Using the LI-6400 / LI-6400XT

Portable Photosynthesis System

Version 6

LI-COR
Biosciences

NOTICE

The information contained in this document is subject to change without notice.

Macintosh® is a registered trademark of Apple, Inc.

New editions of this manual will incorporate all material since the previous editions. Update packages may be used between editions which contain replacement and additional pages to be merged into the manual by the user.

The manual printing date indicates its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change).

U.S. Patent Numbers: 5,332,901 & 5,340,987. Other patents pending in the U.S. and other countries.

LI-6400XT software is subject to the following license agreements:

•binutils, busybox, dbus, gcc, glibc, linux, module-init-tools, pcmciautils, rsync, samba, u-boot, udev: GNU General Public License Version 2
•sysfsutils: GNU Lesser General Public License Version 2.1
•boost: Boost Software License Version 1.0
•howl: Apple Public Source License Version 2.0
•openssh: OpenSSH License
•openssl: OpenSSL License
•xxd: (c) 1990-1998 by Juergen Weigert (jnweiger@informatik.uni-erlangen.de)
•zlib: A Free License
• mDNSResponder: Apache License 2.0

Software license agreements are printed at the end of Book 4.

LI-COR Biosciences, Inc. • 4421 Superior Street • Lincoln, Nebraska 68504
Phone: 402-467-3576 • FAX: 402-467-2819
Toll-free: 1-800-447-3576 (U.S. & Canada)

Contents

Contents

4 Making Measurements

The fundamentals of good measurements

Part II: Useful Details

5 Standard Tools

Trees, Menus, Editors, and File Dialogs

6 New Measurements Reference

Viewing real time data using text and graphics

Contents

Contents

Contents

19 Maintenance & Service

The care and feeding of your new pet

20 Troubleshooting

When things go wrong

Contents

21 Diagnostics and Utilities

Useful programs

Part VI: Programming

22 Programming with LPL

An introduction to the LI-6400's programming language

23 LPL Topics

Programming with LPL

Contents

24 LPL Reference

Keyword Summary

25 AutoProgramming Reference

Making them do what you what them to do

26 Customizing Open

"Cry 'Havoc' - and let slip the dogs of war"

Contents

Part VII: Accessories

27 Leaf Chamber Fluorometer
Using the 6400-40 Leaf Chamber Fluorometer

28 Soil CO2 Flux Chamber
Using the 6400-09 Soil Chamber

A License Agreements
The fine print

Contents

Welcome to the LI-6400XT

A Bit of History

The LI-6400 is LI-COR's third generation gas exchange system. In the beginning was the LI-6000; it used a third-party CO_2 analyzer having the novel (for 1983) features of small size, light weight, and low power. Photosynthesis was computed by measuring the rate of change of CO_2 with time with a leaf enclosed in a relatively large chamber - a closed system. The LI-6000 was limited by the signal noise of the analyzer (1-2 μmol mol^{-1}) and an unfortunate choice of method for computing slopes. These problems were corrected in 1986 in the LI-6200, which sported LI-COR's first CO_2 analyzer and much improved software. It was still a closed system for CO_2, but had the potential for steady-state transpiration measurements. The steadiness of the state, however, depended heavily on a motivated, attentive operator continually adjusting a knob.

While the LI-6200 was fairly well suited for survey measurements, a growing number of customers (and potential customers) were looking to answer deeper questions. The deeper answers could be found in response curves, and that meant using a system that could control the environmental quantities important to photosynthesis: CO_2, light, humidity, and temperature. Several innovators attempted to do various response curves with the LI-6200, with varying degrees of success. Meanwhile, we were having a very hard time in our attempts to transform that instrument into a next-generation system.

By 1990, having abandoned the idea of enhancing the LI-6200, we restarted with the question: "What would the ideal gas exchange system be like?" It would do response curves, of course, so it would need the ability to control chamber conditions. To us, *ideal* response curves meant *labor-free* response curves, so manual controls were out, computer controls were in. Response curves should be seen, not imagined, so we needed built-in graphical capability. The ideal system would also do survey measurements: you should be able to carry it around, clamp onto a leaf using only one hand (our previous systems needed two or three), and be done with the measurement quickly. Ques-

Welcome to the LI-6400XT
A Bit of History

tions of weight, power, and compactness thus emerged. Could the capabilities of a laboratory system (response curves) be squeezed into a field system (survey measurements)?

Challenges loomed, two of them formidable: CO_2 and H_2O control is done best by having gas analyzers right in the leaf chamber, but suitable IRGAs did not exist. Secondly, chamber CO_2 could in principle be controlled by mixing scrubbed air and pure CO_2 (available in very portable 12 gram cylinders), but again, a suitable mixing device did not exist. So, we set about inventing them.

Five years and one name change[1] later, we shipped the first LI-6400 - and many more since then. We've kept working on it, too, adding many innovations and enhancements, including soil respiration and a leaf chamber fluorometer.

In 2002 we changed the digital board (200 MHz processor, Linux operating system, 128 MBytes memory, 64 MByte file system), and brought out version 5 software, opening the door to many exciting new possibilities.

2007 brought the LI-6400XT, which was another digital board change, with version 6 software, a 400 MHz processor, and an expansion slot for flash memory or Ethernet connectivity, making possible remote control of multiple units, and seamless integration of LI-6400XT consoles into local area networks. Data file recomputation got simpler: version 6 added the capability of generating data files as binary Excel files, with the equations built-in.

Old units continue to be upgradable. LI-6400s with the 200 MHz board (i.e. version 5 software) are able to run version 6 with just a software change. If you want the expansion slot capability, then you can turn any LI-6400 into an LI-6400XT with an upgrade kit. Thus, in the nearly 15 year history of the LI-6400, we haven't left any users behind; their investment remains solid.

We thank you for your investment, and trust that this instrument will serve you well. We stand ready with support and help as you put it to work. Welcome to the LI-6400XT.

---

[1] It started out as the LI-6300.

About this Manual

When confronted with a new instruction manual, most people turn to the section named "About this manual" for one simple reason: to find out how much they can skip. Since this manual has 28 chapters, you are probably eager to skip most of them.

You're in luck. You can.

Where to Start?

Select the category that best describes you, and follow the suggested itinerary.

- **Experienced with earlier versions of the LI-6400**
  Read **Version 6.2 Summary** on page xvi.

- **New to the LI-6400; new to gas exchange; methodical and thorough**
  Work through Chapters 1, 2, 3, and 4. You'll understand the LI-6400 and the fundamentals of making measurements fairly well by the end. Then, if you are also doing fluorescence, Chapter 27.

- **New to the LI-6400; new to gas exchange and/or fluorescence; and fairly impatient**
  Chapter 2: Assembling the system.
  Chapter 3: The minimum for learning the software is:
      Tour #1, Stop after Step 5.
      Tour #2, The whole thing.
      Tour #3, Experiments 1, 2, and 5.
      Tour #4, Experiments 1 through 3
  Chapter 4: Do the check lists, and the simple experiments.
  Chapter 27: Fluorescence.
  Having measured some leaves (and satisfied your impatience), you might want to go back and pick up the parts you skipped in Chapters 3 and 4.

- **New to the LI-6400, experienced at gas exchange measurements**
  Skim Chapter 1.
  Chapter 2: Assembling the system.
  Chapter 3: Tours 1 through 5.
  Chapter 4: The check lists are important; pick and choose after that.
  Chapter 27: How to connect and operate the LCF.

Electronic version

This manual is also available as an Adobe® Acrobat® file, on CD and from our web site (www.licor.com). All cross references in the text, table of contents, and index are hyper-linked, allowing one-click access.

Version 6.2 Summary

There are a number of interface changes that came with OPEN 6.2, and the major ones are summarized here. If you are familiar with earlier versions of OPEN, then this summary will get you started very quickly.

Version 6.2 can run on either the third generation 400MHz LI-6400XT, or the second generation 200 MHz board (currently running version 5.x).

Controlling Flow, CO_2, Temperature, Light

The "control panels" in New Measurements mode are a bit different in version 6.2. The differences are summarized below, and fully described in **Interface Fundamentals** on page 7-3.

Old Style:
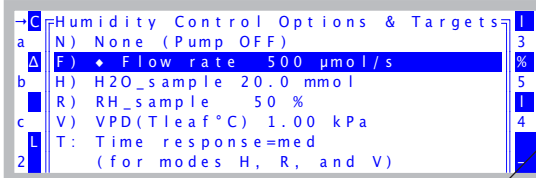
New Style:
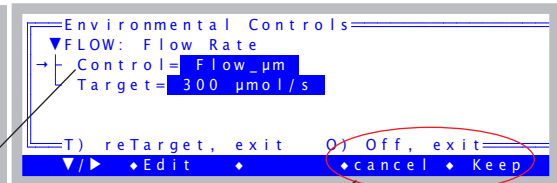
1. It's a modal dialog now.

2. Edit the *Control* node to get all the options.
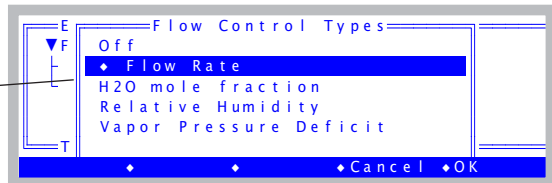
3. The dialog adapts to the selected control type. Here is VPD, for example.
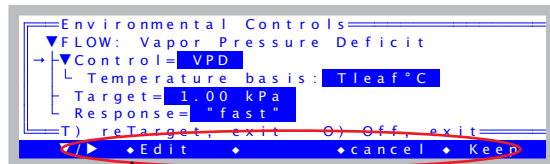
4. Shortcuts: T and O.
O - (for Off): Turns off the control and exits.
T - enter new target value, and exit.

Constant C_i

One of the CO_2 Control options is to maintain a constant C_i.

```
┌─E ┌────────CO2 Control Types────┐
▼C  │  ◆ Off                       │
 L  │  Reference CO2               │
    │  Sample CO2                  │
    │  Control Signal              │
─T  │  Leaf Intercellular CO2      │
    └──────────────────────────────┘
    ◆          ◆          ◆Cancel ◆OK
```
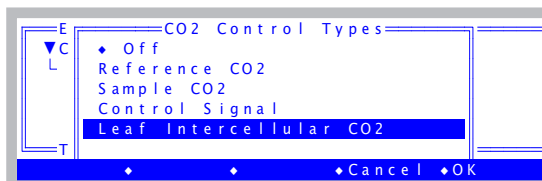
You would use this option to try and maintain C_i while varying some environmental parameter (*other* than CO_2) such as light. See **Intercellular CO2** on page 7-17.

New Log Options

Version 6.2 brings two new Log Options:

Old Options:

```
┌──────────Log Options──────────┐
│ Beep: ON                       │
│ Header: Embedded (normal)      │
│ Remarks: Embedded (normal)     │
│ Stability Details: Not logged  │
│ Statistics: None (normal)      │
│ Excel File: YES                │
│ Echo to COMM: No               │
└────────────────────────────────┘
```

New Options:

```
┌──────────Log Options──────────┐
→ ├ Beep: on                     │
  ├ Header: normal               │
  ├ Remarks: normal              │
  ├ Stability: no                │
 ▼├* Statistics: active (15 s)   │
  ├ * Means->log file: yes       │
  ├ Stats->.STATS file: no       │
  └ period: 15                   │
  ├ Excel file: yes              │
  ├ Control changes: yes         │
  └ Echo to Comm: no             │
└────────────────────────────────┘
```
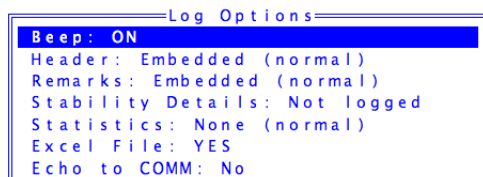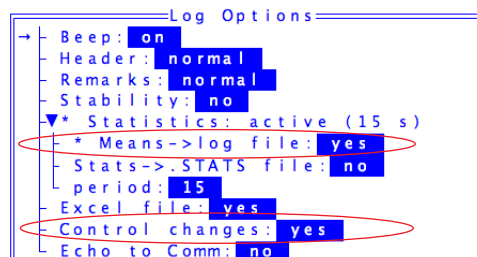
Means➜log file

Normally when you log data, the values recorded are averaged over a fairly short time period, typically 4 seconds or less. When the *Means➜log file* option is enabled, what is logged can be much longer, such as the 15 seconds in the above example. See **...Means➜Log File** on page 9-16.

Control Changes

When the Control Changes option is enabled, anytime you have a log file open, and a control (flow, CO_2, temperature, or light) changes, the change is logged in the log file as a remark. See **Control Changes** on page 9-17.

Welcome to the LI-6400XT

Version 6.2 Summary

AutoPrograms

Version 6.2 brings a number of changes to AutoPrograms, as summarized below. For details, see **AutoPrograms** on page 9-31.

Old Style:

```
Light Curve
Desired lamp settings (µmol/m2/s)
2000 1500 1000 500 200 100 50 20 0
Minimum wait time (secs) 120
Maximum wait time (secs) 200
Match if |ΔCO2| less than (ppm) 20 _

DelLn    ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```

New Style:

```
═══ (2/4) LightCurve2 Setup═══════
     ▶Flr Actions
→▶Summary: 8 SetPts for Qntm
     ▶Stability Definition: 4 items
     ▶Log Opts: beep ctrl

+   ▼/▶   ◆        ◆        ◆cancel ◆ START
```

1. It's a modal dialog now.

2. The input is contained in an editable tree, with nodes you can expand as needed.

3. The programs adapt to whatever light source you are using. in this case, it is an LCF.

4. New and improved matching options (on, off, conditional).

5. Stability and Log Options are part of the input setup.

```
    ═══ (2/21) LightCurve2 Setup═══════
  ▶Flr Actions
→▼Summary: 8 SetPts for Qntm
  ├ Lamp control= PAR
  └▶SetPts: 8 total, 1st= 1800 0 0
    ├▼Stability wait= 180 to 360 s
    │ ├ Minimum (secs)= 180
    │ └ Maximum (secs)= 360
    ├▼Match before log= "If one of..."
    │ ├ ...Elapsed time (min) >    30
    │ ├ ...|CO2 change| (ppm) >    100
    │ ├ ...|ΔCO2| (ppm)<   5
    │ ├ Post-match recovery min (s)= 10
    │ └ Post-match recovery max (s)= 300
    └ Log= Log w/ Fs Fm'
  ▶Stability Definition: 4 items
  ▶Log Opts: beep ctrl
```

6. Read / Write your setup to named files.

```
+   ▼/▶   ◆Edit    ◆        ◆cancel ◆ START
+Open...◆saveAs.◆       ◆        ◆Default
```

'Pick From List' Prompts

Version 6.2 adds a new type of Prompt. This feature lets you pre-enter a list of strings (complicated plot identifiers, etc.), and when you are prompted for one, you can pick it from the list. You can also modify your list on the fly. See **'Pick From List' Prompts** on page 9-25.

New Configuration Builder

The interface for building configurations (Config Menu | New...) has evolved from sequential questions to a dialog.

Old Style:

```
Leaf Temperature:
Measured or Energy Balance ? (M/E)
Enter Leaf type:
Needles or Broadleaves ? (N/B)
```

New Style:

```
═Config for 2x3═══
→└▼LightSource= "6400-02 or -02B LED So
  └ LampSerialNumber= "SI-2262"
─ Bottom= Opaque
─ Material> Broadleaves
└ LeafTemp= Measure

 ▼/►   ◆Edit    ◆        ◆Cancel ◆DoIt
```
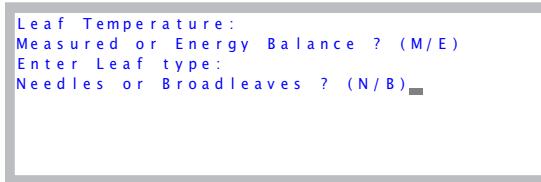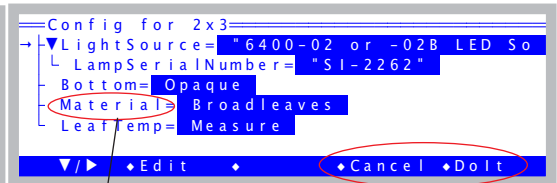
1. It's a modal dialog now.

2. Edit *Material* to generate a mass-based configuration for any chamber. See **The Material= Node** on page 16-51.

```
═Select Material═══
Broadleaves
Needles
Mass-based
```

Improved IRGA Zero

Version 6.2 simplifies the method of zeroing that eliminates the long waits for the sample cell to flush (and dry) out. See **Setting the CO2 and H2O Zero** on page 18-11.

1. With the match valve off, zero the reference (only) analyzers.

```
         Zeroing The IRGA
Mch:off        Fan:Fast          Pump:ON
CO2R_µml   CO2S_µml   H2OR_mml   H2OS_mml
  1.1        2.3        0.127      0.350
Slp(CR)    Slp(CS)    Slp(HR)    Slp(HS)
-4.1E-01   6.1E-01   -2.3E-03    1.1E-02

2 CO2R→0              H2OR→0
```
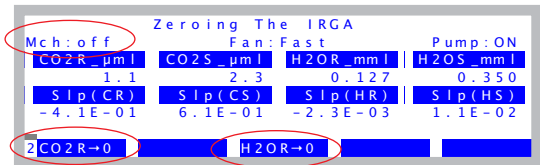
2. With the match valve on, make the sample IRGAs match the reference.

```
         Zeroing The IRGA
Mch:ON         Fan:Fast          Pump:ON
CO2R_µml   CO2S_µml   H2OR_mml   H2OS_mml
  0.5        2.0        0.031      0.290
Slp(CR)    Slp(CS)    Slp(HR)    Slp(HS)
-2.5E-02   7.1E-01    9.3E-02    2.9E-03

2          CO2S→R              H2OS→R
```
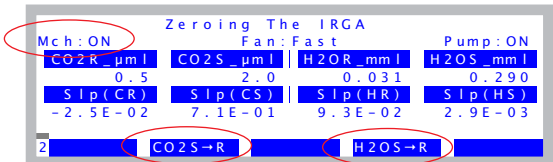
Welcome to the LI-6400XT

Version 6.2 Summary

System Description

1

....

What it is, what it does, and how it does it

1

System Description

This chapter acquaints you with the LI-6400XT's operating principle, major components, and how it computes gas exchange quantities.

An Open System

Measuring Differentials

The LI-6400 is an open system, which means that measurements of photosynthesis and transpiration are based on the differences in CO_2 and H_2O in an air stream that is flowing through the leaf cuvette (Figure 1-1).
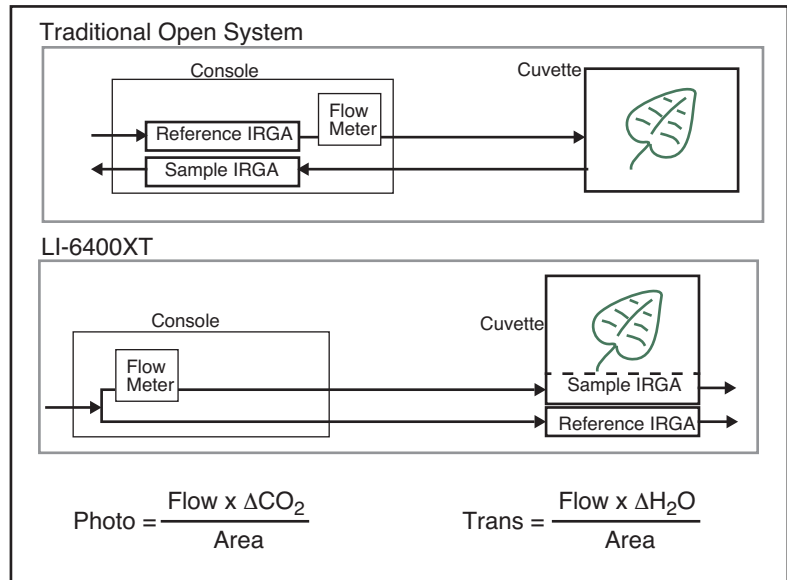
*Figure 1-1. In an open system, photosynthesis and transpiration are computed from the differences in CO_2 and H_2O between in-chamber conditions and pre-chamber conditions. The equations are given in **Equation Summary** on page 1-7.*

The LI-6400 improves upon traditional open systems by having the gas analyzers in the sensor head. This eliminates plumbing-related time delays, and allows tight control for responding to leaf changes. For example, if stomata close, the control system immediately detects the drop in water vapor and can compensate. Similarly, a sudden change in light level will cause an immediate change in photosynthetic rate, which will be detected as a change in the CO_2 concentration. The speed of detection is not a function of the system's flow rate, as in traditional systems, since the sample IRGA is in the cuvette.

There is a second advantage of having the IRGAs in the sensor head. The traditional system has the potential for concentration changes (because of water sorption and CO_2 diffusion) as the air moves from the reference IRGA to the chamber, and again from the chamber back to the sample IRGA. This is not a problem for the LI-6400, because the IRGA measurements are made *after* the air has travelled through the tubing.

The Air Supply

One strength of an open system is that the incoming air stream can be conditioned. That is, its humidity, CO_2 concentration, temperature, etc. can be established by some means prior to entering the system.

Regardless of what is done to the incoming air, however, one thing is crucial:

> *Incoming concentrations must be stable.*

This is especially true for CO_2, where the potential for fluctuations is huge (your breath, for example, is typically 30,000 μmol CO_2 (mol air)$^{-1}$. Fluctuations in incoming concentration will cause concentration differences to be very erratic.

Leaks

"Pressure inside the leaf chamber is slightly above ambient to ensure that leaks are outward, so outside air does not enter the chamber and affect the CO_2 and H_2O concentrations." This argument for the advantage of an open system has been made for a long time, and it's true - as far as it goes. There is another process at work: diffusion. CO_2 will always diffuse from higher concentrations toward lower, even against a pressure gradient, and is only limited by the material through which it is moving. The black neoprene gaskets that we use on the LI-6400 (except for the light source) have the lowest

diffusivity to CO_2 of all the gasket material we have tested, but it's not perfect. Diffusion effects are measurable at low flow rates when there is a large concentration difference between chamber and ambient. See **Diffusion Leaks** on page 4-44 for more details.

The Flow Schematic

The LI-6400 provides mechanisms for modifying the incoming air's CO_2 and H_2O concentrations (Figure 1-2 on page 1-5). There are chemical tubes for scrubbing CO_2 and H_2O, and air can be diverted through these tubes in any proportion desired. CO_2, however, is best controlled by scrubbing *all* of it from the incoming air, and using the 6400-01 CO_2 mixer to inject just enough CO_2 to provide a stable concentration at the desired value. If the 6400-01 is not part of your system, you will need to use a buffer volume. For a more complete discussion of buffer volumes, see **Air Supply Considerations** on page 4-50.

Humidity control in the leaf cuvette is achieved by regulating the flow rate that is going though the cuvette. In units without a CO_2 mixer, the pump speed controls the flow, and in units with a CO_2 mixer, a flow diverter regulates this flow.

Schematic without a 6400-01 CO2 Mixer
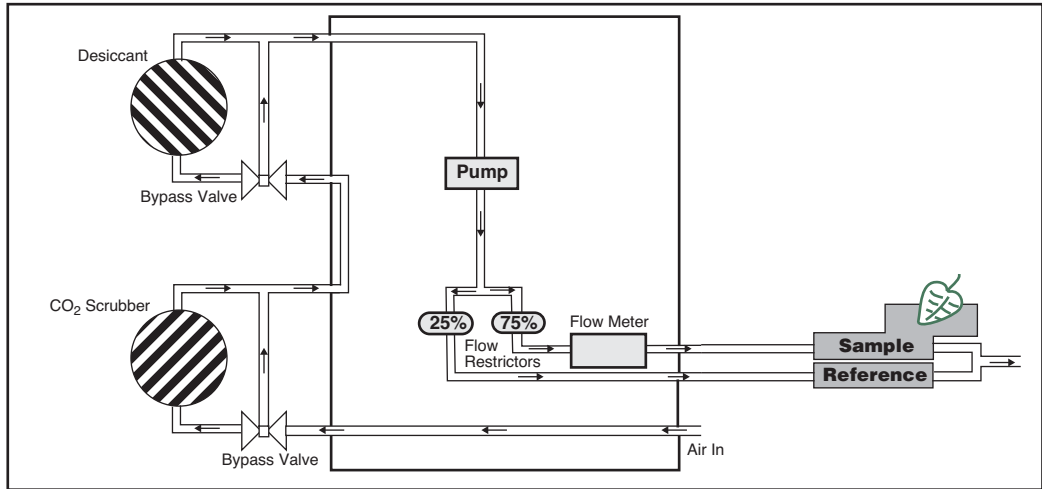
Schematic with a 6400-01 CO2 Mixer
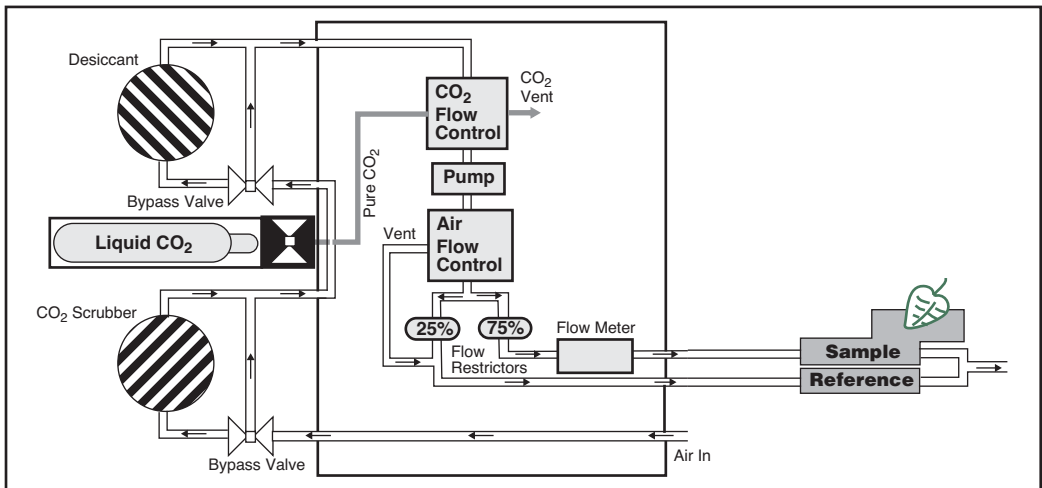
Figure 1-2. LI-6400XT flow schematic, with and without a 6400-01 CO$_2$ mixer.

Matching the IRGAs

The heart of the computation of transpiration and assimilation is the measurement of the concentration differences. Since these differences are measured by two independent infra-red gas analyzers, provisions must be made to check the IRGAs against one another. One way to do this is to compare the IRGAs when the leaf chamber is empty, and adjust one so that they match. Matching in this manner is a problem, however, because you don't want to remove the leaf from the chamber in the middle of an experiment. The LI-6400 provides a mechanism to match the IRGAs without disturbing the leaf: it is called match mode, and it is illustrated in Figure 1-3.
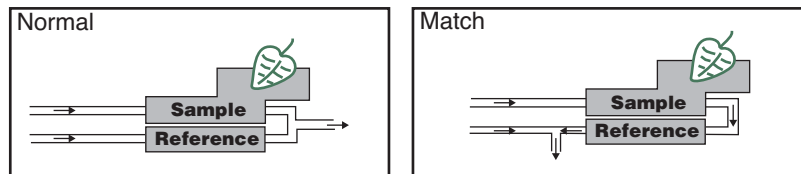
Figure 1-3. In Match Mode, leaf chamber air is measured by both sample and reference IRGAs, allowing the sample IRGA to be adjusted to match the reference IRGA.

Match mode is something that you do at least once at the start of the day, and periodically throughout the day. Matching is important when the ΔCO_2 or ΔH_2O value is small (low rates, small leaf areas). For example, a $1\,\mu$mol mol^{-1} difference between the two CO_2 IRGAs is trivial when the ΔCO_2 is 75 μmol mol^{-1}, but represents a significant error if the ΔCO_2 is only $3\,\mu$mol mol^{-1}.

Equation Summary

If you are not interested in the details of the LI-6400's gas exchange calculations, you can safely skip this section.

The equations for net photosynthesis, transpiration, etc. are essentially those derived by von Caemmerer and Farquhar[1]. Note also that these equations represent the instrument's defaults, and can be modified or replaced as desired by the user (Chapter 15).

Transpiration

The mass balance of water vapor in an open system (Figure 1-4) is given by
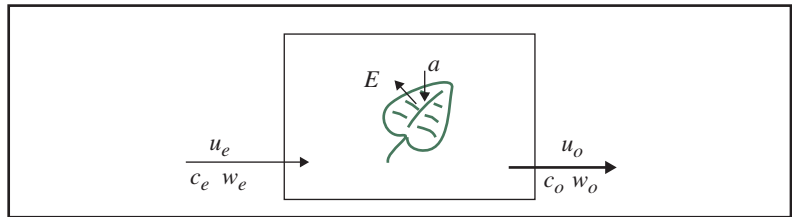
*Figure 1-4. Measuring fluxes in an open system. Transpiration rate (*E*) and photosynthetic rate (*a*) change the water and CO_2 concentrations of air as it passes through the chamber. Transpiration also causes the exit flow* u_o *to be greater than the incoming flow rate (*u_e*).*

$$sE \; = \; u_o w_o - u_e w_e \qquad\qquad (1\text{-}1)$$

where s is leaf area (m^2), E is transpiration rate (mol m^{-2} s^{-1}), u_e and u_o are incoming and outgoing flow rates (mol s^{-1}) from the chamber, and w_e and w_o are incoming and outgoing water mole fractions (mol H_2O mol air^{-1}). Since

$$u_o \; = \; u_e + sE \qquad\qquad (1\text{-}2)$$

we can write

$$sE \; = \; (u_i + sE)w_o - u_e w_e \qquad\qquad (1\text{-}3)$$

---

[1]·S.von Caemmerer and G.D.Farquhar (1981) Some relationships between the biochemistry of photosynthesis and the gas exchange of leaves, *Planta* 153:376-387.

which rearranges to

$$E = \frac{u_e(w_o - w_e)}{s(1 - w_o)} \qquad (1\text{-}4)$$

The relationships between the terms in (1-4) and what the LI-6400 measures are

$$u_e = F/10^6$$
$$w_e = W_r/10^3$$
$$w_o = W_s/10^3 \qquad (1\text{-}5)$$
$$s = S/10^4$$

where F is air flow rate (μmol s^{-1}), W_s and W_r are sample and reference water mole fractions (mmol H_2O (mol air)$^{-1}$), and S is leaf area (cm^2). The equation that the LI-6400 uses for transpiration is thus

$$E = \frac{F(W_s - W_r)}{100S(1000 - W_s)} \qquad (1\text{-}6)$$

Total Conductance to Water Vapor

The total (includes stomatal and boundary layer) conductance of the leaf g_{tw} (mol H_2O m^{-2} s^{-1}) is given by

$$g_{tw} = \frac{E\left(1000 - \dfrac{W_l + W_s}{2}\right)}{W_l - W_s} \qquad (1\text{-}7)$$

where W_l is the molar concentration of water vapor within the leaf (mmol H_2O (mol air)$^{-1}$), which is computed from the leaf temperature T_l (C) and the total atmospheric pressure P (kPa)

$$W_l = \frac{e(T_l)}{P} \times 1000 \qquad (1\text{-}8)$$

The function $e(T)$ is saturation vapor pressure (kPa) at temperature T (C). The formula used by the LI-6400 is (14-24) on page 14-13.

Stomatal Conductance to Water Vapor

The stomatal conductance g_{sw} to water vapor (mol H_2O m^{-2} s^{-1}) is obtained from the total conductance by removing the contribution from the boundary layer.

$$g_{sw} = \frac{1}{\dfrac{1}{g_{tw}} - \dfrac{k_f}{g_{bw}}} \qquad (1\text{-}9)$$

where k_f is a factor based on the estimate K of the fraction of stomatal conductances of one side of the leaf to the other (termed *stomatal ratio* throughout this manual),

$$k_f = \frac{K^2 + 1}{(K + 1)^2} \qquad (1\text{-}10)$$

and g_{bw} is the boundary layer conductance to water vapor (mol H_2O $m^{-2}s^{-1}$) from one side of the leaf. The boundary layer conductance correction thus depends on whether the leaf has stomata on one or both sides of the leaf.

Net Photosynthesis

The mass balance of CO_2 in an open system is given by

$$sa = u_e c_e - u_o c_o \qquad (1\text{-}11)$$

where a is assimilation rate (mol CO_2 m^{-2} s^{-1}), c_e and c_o are entering and outgoing mole fractions (mol CO_2 mol air^{-1}) of carbon dioxide. Using (1-2), we can write

$$sa = u_e c_e - (u_e + sE)c_o \qquad (1\text{-}12)$$

which rearranges to

$$a = \frac{u_e(c_e - c_o)}{s} - Ec_o \qquad (1\text{-}13)$$

To write (1-13) in terms of what the LI-6400 measures, we use (1-5) and

$$c_e = C_r / 10^6$$

$$c_o = C_s / 10^6 \qquad (1\text{-}14)$$

$$a = A / 10^6$$

where C_r and C_s are sample and reference CO_2 concentrations (μmol CO_2 (mol air)$^{-1}$), and A is net assimilation rate of CO_2 by the leaf (μmol CO_2 m^{-2} s^{-1}). Substitution yields

$$A = \frac{F(C_r - C_s)}{100S} - C_s E \qquad (1\text{-}15)$$

Eqn (1-15) was used by the LI-6400 until version 6.0. To get to the new equation, we substitute for E from (1-6) to get

$$A = \frac{F\left(C_r - C_s - C_s\left(\frac{W_s - W_r}{1000 - W_s}\right)\right)}{100S} \qquad (1\text{-}16)$$

which reduces to

$$A = \frac{F\left(C_r - C_s\left(\frac{1000 - W_r}{1000 - W_s}\right)\right)}{100S} \qquad (1\text{-}17)$$

Why does transpiration (or W_r and W_s in (1-17)) appear in the equation for photosynthesis? (Asking this question means you didn't follow the derivation…) The short answer is that it serves as a dilution correction; as the leaf adds water vapor to the chamber, it dilutes all other gasses, including CO_2.

Intercellular CO$_2$

The intercellular CO$_2$ concentration C_i (μmol CO$_2$ mol air^{-1}) is given by

$$C_i = \frac{\left(g_{tc} - \frac{E}{2}\right) C_s - A}{g_{tc} + \frac{E}{2}} \qquad (1\text{-}18)$$

where g$_{tc}$ is the total conductance to CO$_2$, and is given by

$$g_{tc} = \frac{1}{\dfrac{1.6}{g_{sw}} + \dfrac{1.37 k_f}{g_{bw}}} \qquad (1\text{-}19)$$

1.6 is the ratio of the diffusivities of CO$_2$ and water in air, and 1.37 is the same ratio in the boundary layer.

Everything Else

There are many other relationships that the LI-6400 uses (calibration equations for sensors, dew point temperatures, relative humidity, etc.), which are documented in Chapter 14.

Summary of Symbols

a = net assimilation rate, mol CO_2 m^{-2} s^{-1},

A = net assimilation rate, μmol CO_2 m^{-2} s^{-1}

c_e = incoming CO_2 concentration, mol CO_2 mol air^{-1}.

c_o = outgoing CO_2 concentration, mol CO_2 mol air^{-1}.

C_s = mole fraction of CO_2 in the sample IRGA, μmol CO_2 mol^{-1} air

C_r = mole fraction of CO_2 in the reference IRGA, μmol CO_2 mol^{-1} air

C_i = intercellular CO_2 concentration, μmol CO_2 mol air^{-1}

E = transpiration, mol H_2O m^{-2} s^{-1}

F = molar flow rate of air entering the leaf chamber, μmol s^{-1}

g_{bw} = boundary layer conductance to water vapor, mol H_2O m^{-2} s^{-1}

g_{sw} = stomatal conductance to water vapor, mol H_2O m^{-2} s^{-1}

g_{tc} = total conductance to CO_2, mol CO_2 m^{-2} s^{-1}

g_{tw} = total conductance to water vapor, mol H_2O m^{-2} s^{-1}

$k_f = (K^2 + 1)/(K + 1)^2$,

K = stomatal ratio (dimensionless); estimate of the ratio of stomatal conductances of one side of the leaf to the other

s = leaf area, m^2

S = leaf area, cm^2

u_e = incoming flow rate, mol air s^{-1}.

u_o = outgoing flow rate, mol air s^{-1}.

w_e = incoming H_2O mole fraction, mol H_2O mol air^{-1}.

w_o = outgoing H_2O mole fraction, mol H_2O mol air^{-1}.

W_s = sample IRGA mole fraction of water vapor, mmol H_2O mol air^{-1}.

W_r = reference IRGA mole fraction of water vapor, mmol H_2O mol air^{-1}.

W_l = mole fraction of water vapor within the leaf, mmol H_2O mol air^{-1}.

The System Components

If you have just taken delivery of your LI-6400XT check the packing list to verify that you have received everything that you ordered. Or, if you've just inherited an LI-6400 from someone else, check to see that you have everything. Here's a brief description of what should be there:

Figure 1-5. The LI-6400XT Portable Photosynthesis System.

The Standard Parts

Console
The console has an environmentally sealed, 64-key, full ASCII keypad and 8 line × 40 character LCD. On the right side of the console are the sensor head connectors, 6400-03 battery compartments, and RS-232C connector. CO_2 scrubber and desiccant tubes attach to the left side of the console, along with the optional 6400-01 CO_2 source assembly or CO_2 tank connector block. A field stand is normally attached to the underside of the console.

Sensor Head/IRGA
The sensor head/IRGA includes a leaf chamber, spring-loaded latching handle (squeeze and release to open, squeeze and release to close), two Peltier thermoelectric coolers, and the sample and reference gas analyzers. Up to two light measurements are provided for: most leaf chamber tops have a Gallium Arsenide Phosphide (GaAsP) PAR sensor, and a mounting fixture is provided for a 9901-013 external quantum sensor, if desired. (For a discussion of the logic of using two light sensors, see **Why Two Sensors?** on page 8-2.) Leaf temperature is measured with a thermocouple held in the bottom of the 2x3 and 2x6 cm leaf chambers; other chambers use energy balance to compute leaf temperature.

Cable Assembly
The cable assembly has two electrical cables and two air flow hoses, and connects the console to the sensor head / IRGA. These are held together with a flexible, outer wrapping.

Spare Parts Kit
This box contains replacement parts for your LI-6400. As you become familiar with the system you will learn which items to keep close at hand and which items can be safely stored away.

Chemical Tubes
These tubes are used during operation to remove CO_2 and water vapor from the incoming air stream. One tube should contain soda lime, and the other tube should contain Drierite, a desiccant. Each tube has an adjustment valve at the top for partitioning the flow through the chemical contained in the tube.

6400-03 Rechargeable Batteries
The 6400-03 Rechargeable Batteries are shipped tested and fully charged. Because they slowly self-discharge with time, it is a good idea to test your batteries periodically. Leaving them discharged for an extended period can result in damage. (See **6400-03 Batteries** on page 19-8 for instructions con-

cerning testing and charging batteries.) One 6400-03 battery provides approximately 1-2 hours of operational life. Recharge them with the LI-6020 Battery Charger.

LI-6020 Battery Charger

The LI-6020 can charge four 6400-03 Rechargeable Batteries simultaneously. It runs on 92-138/184-276 VAC, 47 to 63 Hz, but a selector switch on the back of the LI-6020 must be set to the appropriate mains voltage. The unit ships with no fuse installed - install the correct fuse based on your mains voltage (0.5A for 115 V, 0.25 A for 230 V).

RS-232C Cable

Part number 9975-016 has a 9 pin to 9 pin cable, and a separate 9 to 25 pin adapter. Figure 11-27 on page 11-25 illustrates how to use it.

6400-25 Compact Flash Memory Card

Can be installed in the expansion slot in the XT console.

6400-26 Ethernet Adapter Card

For use in the expansion slot in the XT console.

Field Stand

When shipped from the factory, the LI-6400's field stand base plate is attached to the bottom of the console. It normally remains there, and you can attach or detach the legs as needed. Store the legs in the narrow front slot of the carrying case.

6400-60 CD

The CD contains a hyper-linked, electronic version (.pdf file) of this manual, plus a number of support programs for tasks such as (re-)installing instrument software, and remotely controlling the LI-6400 from a computer. See **Support Software** on page 11-2. Note: New versions of all of our software are released periodically. You can get the most recent version by contacting LI-COR, or by downloading it from our web site (www.licor.com).

Calibration Sheet

This data sheet lists the calibration information entered into the LI-6400 at the factory. Keep it in a safe place for future reference.

Carrying Case

The hard-shell, foam padded carrying case can hold the console, sensor head, cables, some batteries, legs, and a few other accessories.

The Optional Accessories

There are several optional accessories that you may have ordered with your LI-6400. Any of them can also be purchased later, and (with the exception of the 6400-01 CO_2 Mixer), they do not require factory installation:

6400-01 CO2 Mixer

This consists of three components:

- **A control module that is inside the console**
  This part is factory installed in the console.

- **A cartridge holder and regulator**
  For use with the disposable 12 gram CO_2 cartridges, this part is user-installable between the chemical tubes on the outside of the console (see Figure 2-6 on page 2-8).

- **An adapter block for CO_2 tanks**
  This alternative to the cartridge holder and regulator allows tanks of compressed pure CO_2 to be used instead of the 12 gram cartridges (see Figure 2-8 on page 2-11).

6400-02B LED Light Source

The 6400-02B replaces the top half of the standard leaf chamber and provides light from 0 to over 2000 μmol quanta m^{-2} s^{-1}. The intensity of the light is software adjustable to a resolution of 1 μmol m^{-2} s^{-1}. The 6400-02B replaces the 6400-02, which used only red LEDs. See **Spectral Considerations** on page 8-8 for a comparison of the 6400-02 and 6400-02B.

6400-05 Conifer Chamber

A cylindrical chamber suitable for short-needled shoots. Leaf temperature is obtained by energy balance, for which an external PAR sensor reading is required. (External PAR sensor not included with this option.)

6400-07 Needle Chamber

A 2x6 cm chamber with Propafilm® top and bottom windows. Leaf temperature is not measured, but is computed using an energy balance. A GaAsP light sensor is included.

6400-08 Clear-Bottom Chamber

A 2x3 cm chamber bottom with a Propafilm® window. Leaf temperature is computed using an energy balance. This chamber can be used with any 2x3 cm chamber top (such as the fluorometer adapters) or 6400-02 or -02B LED source.

6400-09 Soil Chamber
For measuring soil CO_2 efflux.

6400-11 Narrow Leaf Chamber
A 2x6 cm chamber with a Propafilm® top window, and an opaque bottom that holds the 6400-04 leaf temperature thermocouple. A GaAsP light sensor is included.

6400-13 Thermocouple Adapter
Allows a type E thermocouple to be connected to the 37 pin connector on the LI-6400 console. This adapter is included with the 6400-09 Soil Chamber.

6400-15 Extended Reach 1 cm Chamber
This is a chamber designed for Arabidopsis and other small leaves. The aperture is 1.0 cm in diameter, and has a Propafilm® top and bottom. No light sensor is included. Leaf temperature comes from an energy balance analysis, for which an external quantum sensor is necessary (but not included).

6400-17 Whole Plant Arabidopsis Chamber
Circular chamber, 7 cm diameter, for whole plant measurements of plants in small pots.

6400-18 RGB Light Source
Designed for the 6400-17 chamber, but can be used on other chambers.

6400-22 Opaque Conifer Chamber
Accommodates short needle shoots. Designed for use with the 6400-18 RGB Light Source. Top shell of chamber is clear, bottom is opaque.

6400-24 Bryophyte Chamber
This is the 7 cm diameter 6400-17 with a solid bottom, making it suitable for measuring unattached clumps of whatever material you'd care to drop into it.

6400-40 Leaf Chamber Fluorometer
Chapter 27 covers installation and operation of this accessory.

6400-70 AC Adapter
This optional accessory fits in the battery compartment, and allows the LI-6400 to be powered from mains power. It can simultaneously (but slowly) recharge one battery.

9901-013 External Quantum Sensor
A LI-COR LI-190SA quantum sensor can be mounted on the sensor head. (The 9901-013 is an LI-190SA with a short cable.)

Display Backlight

If installed, the display backlight is toggled on and off by holding down **shift** + **ctrl** then pressing **home**.

Assembling the LI-6400

2

Putting it all together

2

Assembling the LI-6400

This chapter guides you through the assembly and preparations necessary to operate the LI-6400.

Preparations

This section explains how to prepare the console and sensor head for operation.

The CO_2 Scrub and Desiccant Tubes

The CO_2 scrub and desiccant tubes can remain attached to the console at all times, except when changing chemicals. Figure 2-1 shows the position of these tubes.

Caution: Never unscrew the top cap while a tube is full of chemicals. To change the chemical, grasp the tube barrel (not the top cap) and unscrew the bottom cap. If the top cap is unscrewed with chemicals inside, damage to the air mufflers will occur.

Remove the *bottom* cap of the CO_2 scrub tube, and fill the tube with soda lime (in the spares kit) to within 1 cm of the tube's end. Replace the bottom cap and attach the tube to the console using the lower of the two knurled knobs.

Follow the same procedure with the desiccant tube. Indicating Drierite desiccant is provided in the spares kit.

Keep the threads on the end cap and barrel clean

Do not over-tighten the attachment screws (Figure 2-1). Slightly snug (finger power only - *never* pliers) is sufficient. The O-rings do the sealing.

Complete information on maintenance and service of the chemical tubes is found on page 19-2.
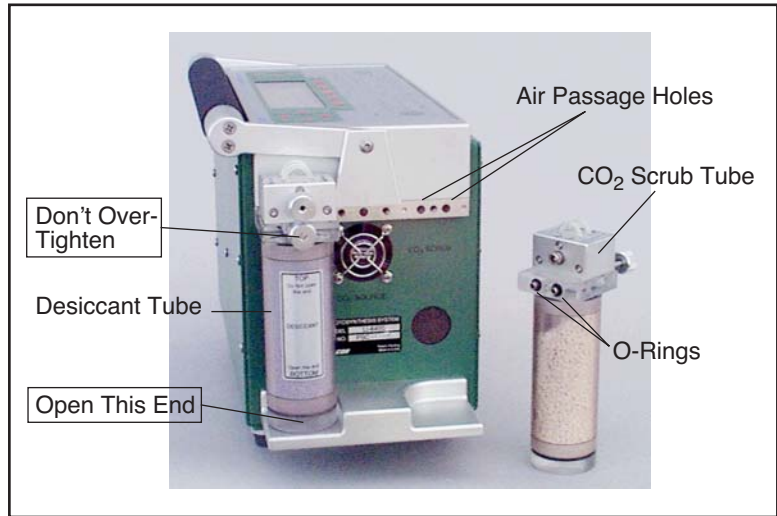
Figure 2-1. Desiccant and CO_2 scrubber tubes.

Cables And Hoses

Air inlet and outlet ports and electrical connectors are located on the right side (end) of the console (Figure 2-2).

Electrical Connectors

Plug the female 25-pin connector into the receptacle labeled **IRGA**, and the male 25-pin connector into the receptacle labeled **CHAMBER**. These connectors are gender specific, and cannot be interchanged. Tighten (slightly) the screws on the connectors, but be careful: these screws can break off if tightened too much. See **Replacing Connector Screws** on page 19-19.

Air Inlet

The port labeled **INLET** is located to the right of the ON/OFF switch. This is the intake through which the pump draws in the air that flows through the system.

If your system does not have a CO_2 injector, attach tubing from a buffer volume to the INLET port. The buffer volume can be as simple as a clean, dry 2-liter plastic soft drink bottle. The larger the volume, the better. For more details, see **Air Supply Considerations** on page 4-50.

Air Outlets

The two sections of Bev-a-line tubing attached to the sensor head must be connected to the console air outlet ports. One of the tubes has a black band near the end of the hose. Attach this hose to the **SAMPLE** port of the console. Attach the other hose to the **REF** port.
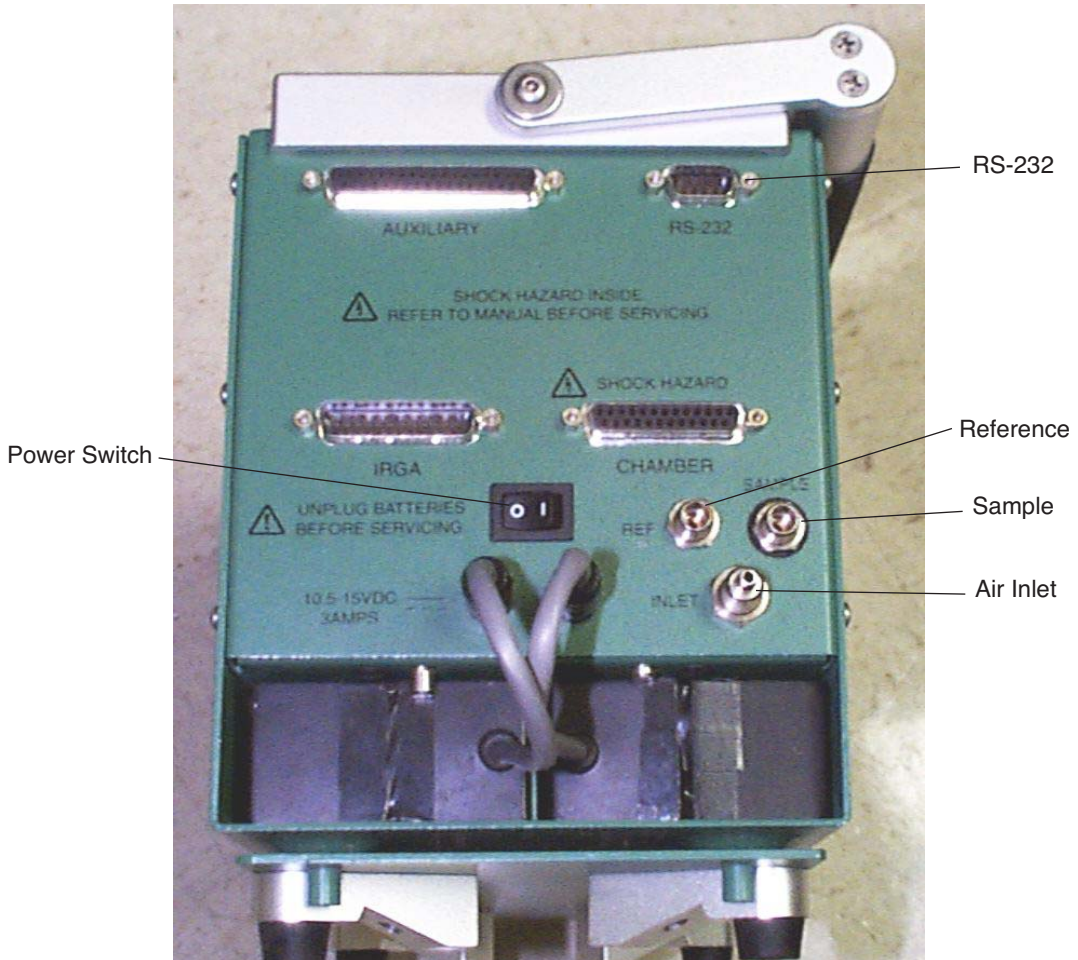
Figure 2-2. Console tubing and cable connections.

Connecting the Chamber / IRGA

The sensor head/IRGA end of the electrical cables attach as shown in Figure 2-3. Be careful not to overtighten the screws on the 26 pin D connector. To plug in the round connector, first line up the red dots, then push the connector all the way in until the red dots meet and there is a click.
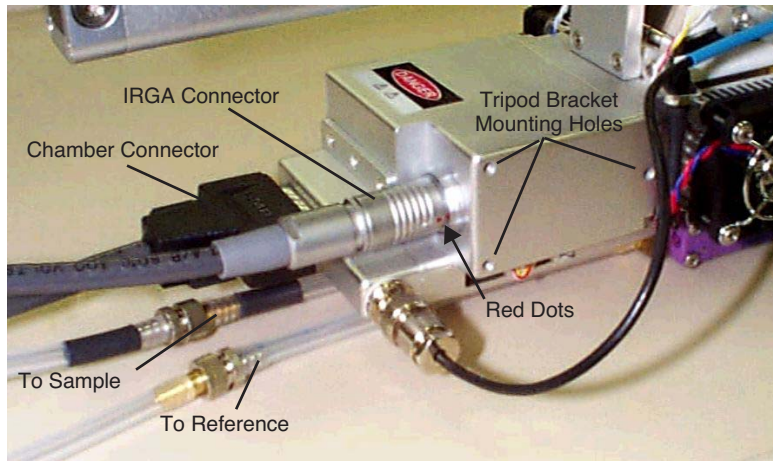
Figure 2-3. Electrical connectors, air hoses, and tripod bracket mounting holes on the sensor head / IRGA.

Interchanging IRGAs? Don't.
If you have more than one LI-6400 at your disposal, can you interchange the IRGA/chambers? The simple answer is (probably) no; they are not designed to do it. Each LI-6400 console is adjusted at the factory for a particular head. If you mismatch them, you may not able to zero the IRGAs.

Using a Tripod

A mounting bracket is included in the spare parts kit for mounting the sensor head on a tripod. A tripod is a virtual requirement when making long-term measurements in the absence of cooperative graduate students.

The three screws included with the mounting bracket are threaded into holes on the right side of the analyzer housing on the sensor head (Figure 2-4). The tripod mounting bracket is threaded for use with standard 3/8-16 and 1/4-20 tripod heads.
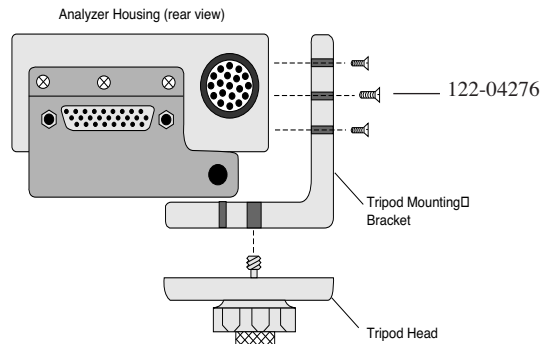
Figure 2-4. Mounting holes for the tripod bracket are found on the right side of the sensor head.

6400-01 CO$_2$ Injector Installation

The optional 6400-01 CO$_2$ Injector consists of a controller that is factory installed within the LI-6400 console, and an external part that attaches between the chemical tubes on the end of the console. This external part can be either

• the 9964-026 Source Assembly
which uses 12 gram CO$_2$ cartridges (described below), or

• the 9964-033 Tank Connector Block
for using a tank of pure CO$_2$ with a regulator, described on page 2-10.

CO$_2$ Cylinder

CO$_2$ Cylinder Cover

O-ring

Piercing Pin

CO$_2$ Cylinder □ Piercing Block

CO$_2$ Source □ Mounting Block

Regulator

> **Warning:** CO$_2$ cylinders contain 12 grams of high pressure liquefied CO$_2$. Follow the handling precautions on the cylinder and cylinder cover carefully.

> **Note:** 12 gram CO$_2$ cartridges last about 8 hours from the time they are pierced, regardless of whether the system is in use or not. However, every once in a while - say, every 100 or 200 cylinders - you might encounter one that provides considerably less, such as only 1 or 2 hours.

Figure 2-5. 9964-026 External CO$_2$ Source Assembly

Using CO$_2$ Cartridges

CO$_2$ source attaches here.

Check for O-rings in these two locations

Cylinder Cover

12 gram cylinder

Oil Filter

Figure 2-6. Location of external CO$_2$ source assembly.

■ **To install the 9964-026 Source Assembly**

1   Attach the assembly block. Is the O-ring present?
Make sure that the O-ring seal on the mounting block is properly seated. Tighten the two knurled knobs on the mounting block to secure the assembly to the console.

2   Unscrew the CO$_2$ cylinder cover.

3   Install a new O-ring in the groove around the piercing block.
Use your finger to press the O-ring into the groove (Figure 2-7). If the O-ring is not in place when the CO$_2$ cartridge is pierced, gas will rapidly vent out a hole on the underside of the mounting block.
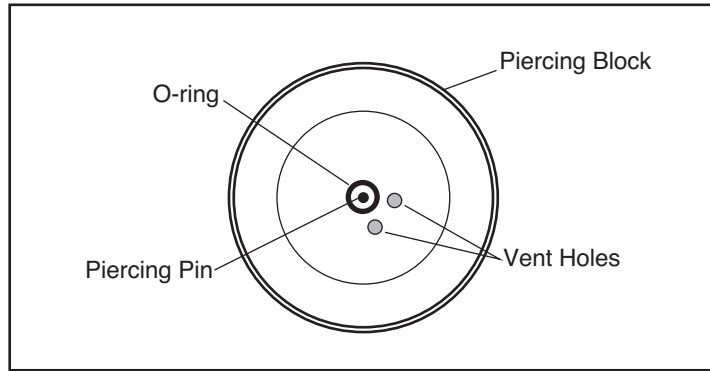
Figure 2-7. Top view of piercing block showing the O-ring location.

Important Note: Although the O-ring may perform properly for several cylinders, *we recommend that it be replaced with each new cylinder*. After being subjected to several high pressure cycles the O-ring weakens and becomes perforated, and easily tears or splits. If the O-ring is slightly torn or perforated, gas slowly leaks through the vent hole shortening the life of the cylinder. If the O-ring is split, gas rapidly vents until the cylinder is empty.

4   **Check the oil filter (every 3 or 4 cylinders)**
It's actually a cigarette filter that is located in the "T" fitting on the rear of the source assembly (Figure 2-6). It's a good idea to unscrew the cap (*before* you install the cylinder!) and look down in at the end of the filter to check for any oil accumulating on the white filter material. If the filter is getting discolored, change the filter. See **Servicing the External CO2 Source Assembly** on page 19-38 for more details. (With LI-COR cylinders, the filter should last for 25 cylinders. We have encountered other cylinders, however, that contain <u>much</u> more oil, notably Copperhead™ and Curtis™ brands, so beware.)

5   **Place a new CO$_2$ cylinder into the cylinder cover**
The cylinder goes in large end first, although one of our engineers accidentally got one to work the other way around.[1]

> *Warning*: Use only the proper size 12 gram cartridges.

---

[1].He's now in management.

6   **Screw the cylinder cover onto the piercing block.**
You may feel some resistance as the piercing pin contacts the cylinder. A short burst of venting CO_2 may occur as the cylinder is pierced; the leak is minimal if you continue to quickly tighten the cylinder cover. Tighten the cover until snug; there's no need to overtighten.

Using other sizes
The mixer cap is designed for a 12g CO_2 cylinder, which is 83 mm in length. You can also use a shorter cylinder, such as the 8g ones readily available in Europe, by inserting a spacer into the cylinder cap. The spacer's length should be such that the cylinder plus the spacer is 83mm (+/- 2 mm).

External CO$_2$ Tanks
The Tank Connector Block replaces the 6400-01 External CO_2 Source Assembly, and is useful in situations where CO_2 tanks or other high volume supplies are available.

The tank connector block is designed for use at pressures between 180 and 220 PSIG (lbs in^{-2} gauge pressure) of CO_2. *Use a regulator, and do not exceed 250 PSIG CO_2, as the pressure relief valve may vent your source*.

The Tank Connector Block uses a 1/8" male NPT to 1/8" tubing fitting. This fitting has a flow restrictor installed (10 cm^3 min^{-1}). *Do not remove this fitting*. A 1/8" to 4mm compression union is also provided for users who may be unable to obtain 1/8" copper tubing. Directions for installing the Tank Connector Block to a CO_2 source using 4mm copper tubing are given in **Installation Using 4mm Copper Tubing** on page 2-11.

■   **To install the 9964-033 Assembly**

1   **Mount the CO$_2$ Tank Connector Block**
Use the two knurled knobs to mount the block between the CO_2 and H_2O scrub tubes. Make sure that the O-ring seal on the back of the block is properly seated.

2   **Insert copper tubing**
Insert the 1/8" copper tubing between the 1/8" connector nut and the ferrule (Figure 2-8).

Important: Note the orientation of the ferrule. One of the tapered ends of the ferrule is longer than the other; the long end must be oriented toward the connector on the mounting block. When the nut is tightened onto the connector,

the ferrule will be permanently crimped to the copper tubing, and you will not be able to remove it.
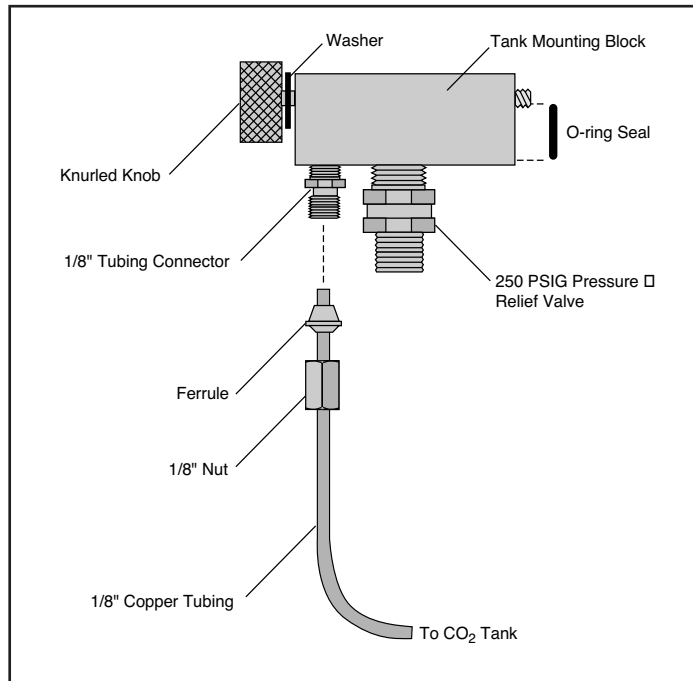
Figure 2-8. Insert tubing through nut and ferrule. Note proper orientation of ferrule.

3   **Tighten the nut until snug, plus 3/4 of a turn.**

4   **Connect your CO$_2$ source.**
The other end of the copper tubing connects to your CO$_2$ source. Adjust your regulator pressure to between 180 and 220 PSIG.

Installation Using 4mm Copper Tubing
If you are unable to obtain 1/8" copper tubing, you can connect the Tank Connector Block to a CO$_2$ source using 4mm tubing and the compression fitting (LI-COR part #300-04439) included with the Tank Connector Block.

1   **Install the Tank Connector Block and copper tubing**
This is described in steps 1-4 above.

2   **Connect the 1/8" and 4mm tubing**
Use the 1/8" to 4mm compression fitting to connect the two pieces of tubing (Figure 2-9). Be sure to orient the ferrules correctly; the narrow tapered end of each ferrule must be oriented *toward* the compression fitting. Tighten the nuts on the compression fitting until snug, plus 1 1/4 turn.

3   **Connect the 4mm tubing to your CO$_2$ source.**
Adjust the regulator's pressure to between 180 and 220 PSIG.
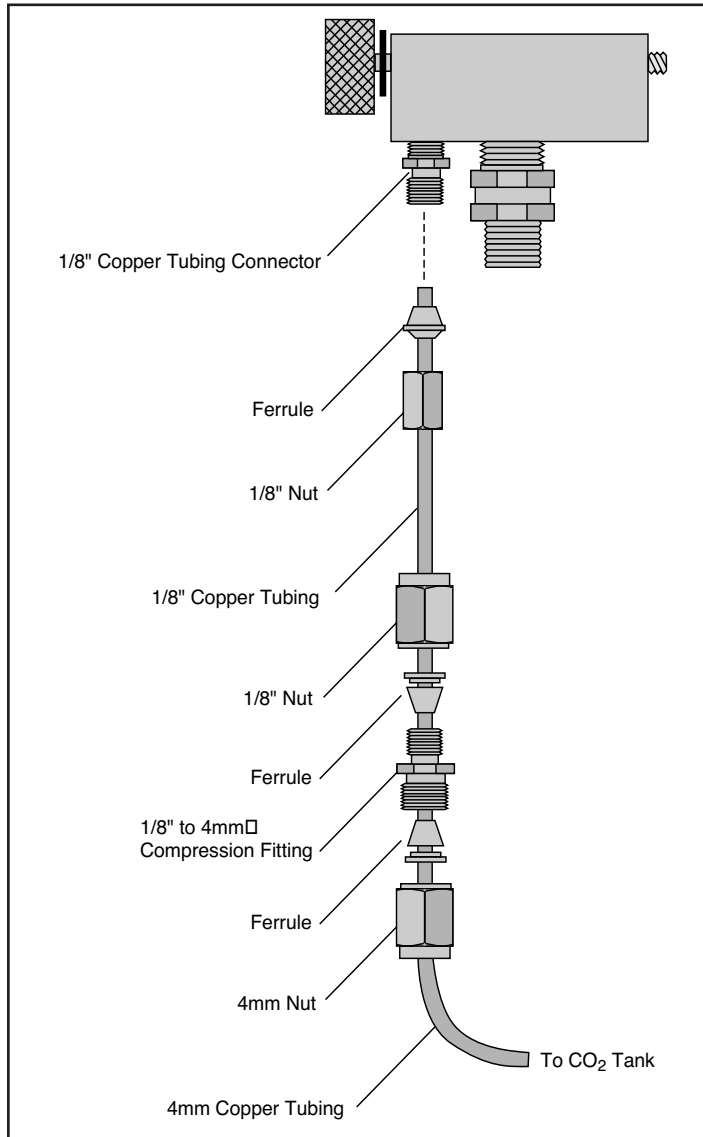
Figure 2-9. Use the compression fitting to connect 1/8" and 4mm tubing.

External Quantum Sensor Installation

The external quantum sensor is held in its mounting bracket with a small set screw (turned with a 0.050" hex key provided in the spares kit). The BNC connector plugs in at the rear of the chamber.
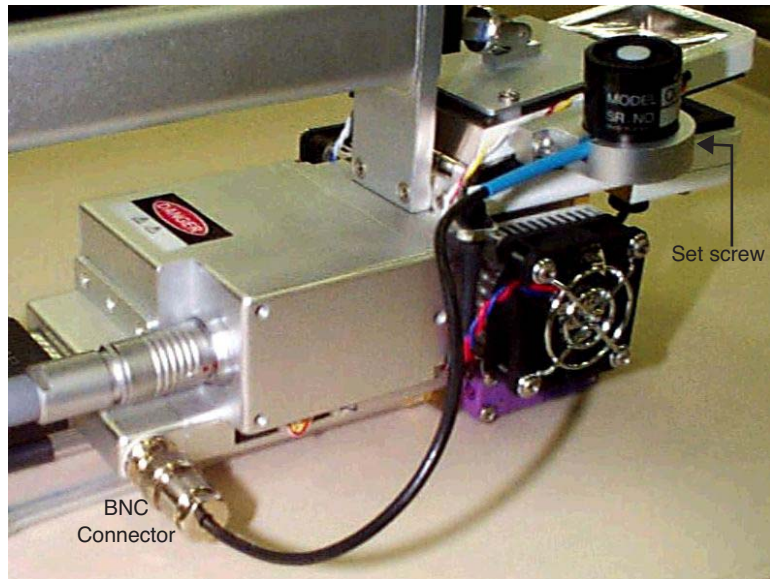
Figure 2-10. The external quantum sensor installed.

If the LI-6400 was shipped from the factory with an external quantum sensor, its calibration factor will have already been entered into the instrument. Otherwise, you will have to do this. See **View / Edit Accessories** on page 18-8.

Attaching the LED Light Source

The optional 6400-02 or -02B LED Light Source is mounted to the sensor head by removing the upper half of the leaf chamber and replacing it with the lamp assembly. Follow these steps to install the lamp:

1   Remove the tripod mounting bracket
This is necessary to access the connector for the in-chamber PAR sensor.

2   Disconnect the light sensor
Pull the connector straight out (<u>don't wiggle side to side</u>) with a pair of long nose pliers (or your fingernails) gripping the connector (Figure 2-11).

Log switch connector

Light sensor connector
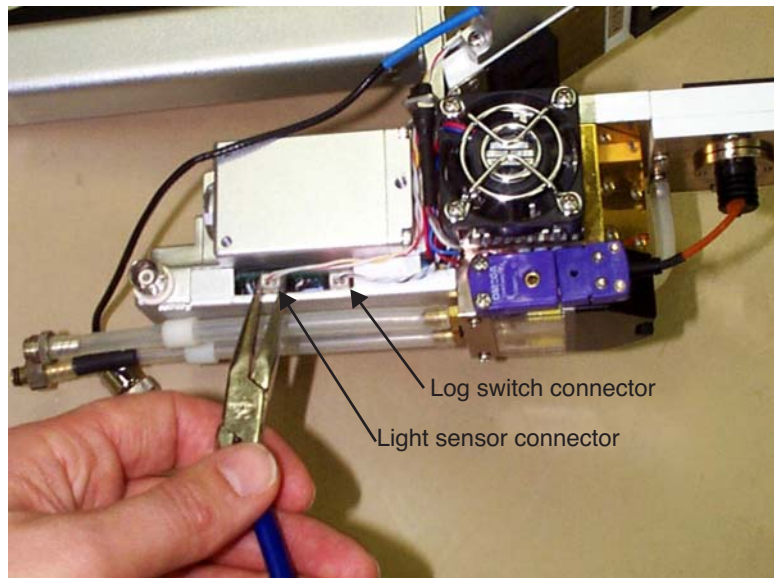
Figure 2-11. Disconnecting the in-chamber PAR sensor connector.

Or, just grasp *both* wires with your fingers and pull straight out. The wires will bring the connector with it.

3   Remove the top leaf chamber
Use the 3/32" hex key provided in the spare parts kit to remove the two long screws that hold the chamber top in place (Figure 2-12).
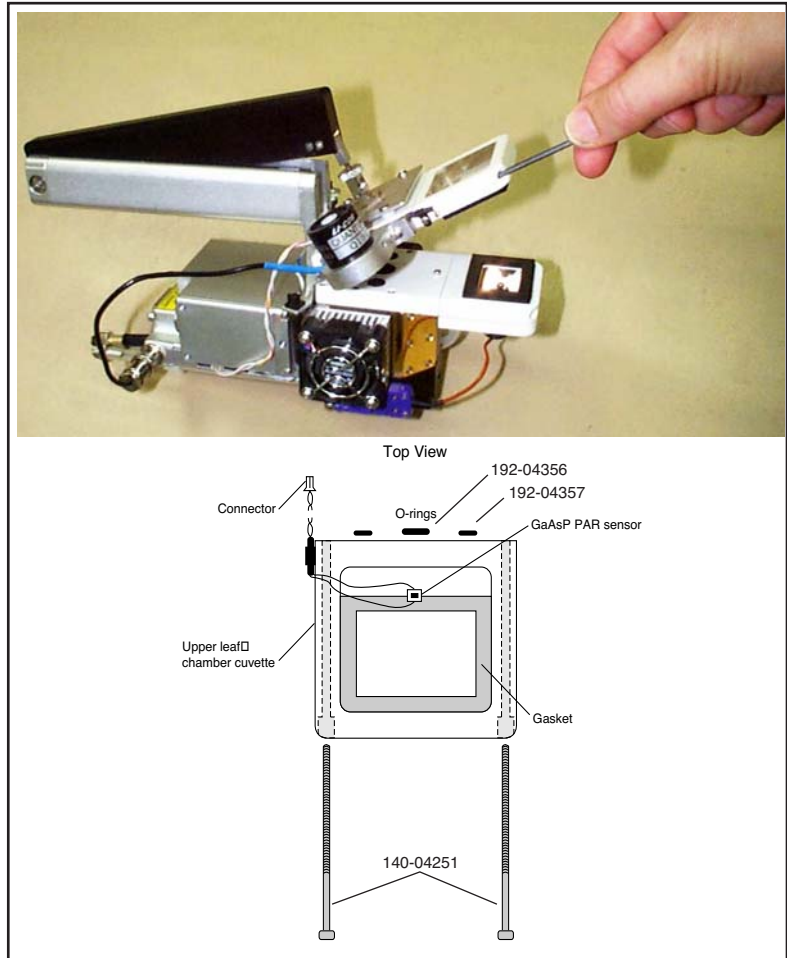
Figure 2-12. The top and bottom chamber halves are held on with two hex head bolts. Use the 3/32 inch hex key to loosen and tighten them.

4   Install the O-rings
Ensure that there are O-rings in the air passage holes.

5   **Install the lamp assembly**

Attach the lamp connector and PAR sensor connector as shown in Figure 2-13. Ensure that the PAR sensor is attached to the connector near the rear of the analyzer housing, *not* to the log switch connector.
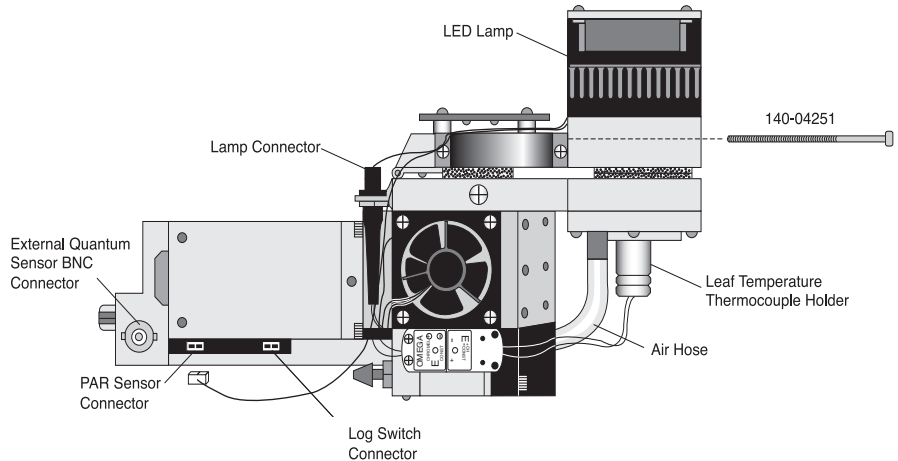
Figure 2-13. Attach the lamp and PAR sensor connectors.

If the LED source was purchased with the LI-6400, its calibration factor will have been installed in the console. Otherwise, you will have to do this. See **Example: 6400-02B LED Source** on page 16-6 for how to do this.

6400-40 Leaf Chamber Fluorometer

Installation and operation are described in Chapter 27.

Powering the LI-6400

LI-6020 Battery Charger
The LI-6400 cannot be operated from mains power alone using this battery charger. (For independent mains power operation, see **6400-70 AC Module** on page 2-20.) It can, however, be operated with the charger *and* a single 6400-03 Rechargeable Battery. To use the LI-6020, plug a fully charged 6400-03 battery into one of the LI-6400 battery jacks, and plug the LI-6020 into the other jack using the 9960-062 cable (in spares kit).

Note: This will not provide indefinite operation. The battery will slowly discharge, and eventually need to be swapped. (Procedure to swap: Disconnect the plug from the *charger*, plug in a fresh battery, then disconnect the old battery and reconnect the plug from the charger.)

6400-03 Batteries
Two **battery jacks** are located beneath the **ON/OFF** switch. Insert two 6400-03 batteries into the battery compartment and connect both batteries. The 6200B batteries used with other LI-COR instruments can also be used, but with less convenience since they will not fit into the battery compartment of the LI-6400.

The 6400-03 batteries have a capacity of approximately 3 Amp-hours each. The LI-6020 Battery Charger produces about 1.5A. The LI-6400, on average, draws 1.5A; therefore, if the LI-6400 is drawing 1.5A, the LI-6020 used with a 6400-03 will power the LI-6400 indefinitely. At maximum draw (with LED light source on, running the coolers, etc.), the LI-6400 will use about 3A. Without the light source, it will draw about 2A. Table 2-2 shows the approximate battery life when the LI-6400 is used with two 6400-03 batteries or with the LI-6020 Battery Charger and one 6400-03.

Table 2-1. Approximate hours of battery life for 6400-03 batteries (at 25 °C ambient).

Power Supply	Power requirement of LI-6400		
	1.5A	2A	3A
Two 6400-03 Batteries	4 Hours	3 Hours	2 Hours
LI-6020 and one 6400-03	Indefinite	6 Hours	3 Hours

- **Two batteries are better than one**
  You will get better battery life when they are used in pairs.

- **At least one battery is required**
  You cannot run the system using only the LI-6020 charger.

- **You are warned when the batteries are low**
  The system will beep regularly when the batteries are low, and there are display indicators as well (see **Low Battery Warning** on page 5-18). One low battery can be removed and replaced with a fresh one while the system continues to run without interruption. Immediately replace the second low battery with a freshly charged battery to ensure maximum operation time.

> **Shooting yourself in the foot…**
> When you are swapping batteries on a running, data-collecting instrument, be careful not to bump the on/off switch with your thumb as you unplug a battery. It's easy to do (I speak from experience), and it quickly brings your measurements to an abrupt halt.

Note that you can also power the LI-6400 using any 12 volt battery with sufficient capacity; a car battery, for example, will run the system for 24 to 48 hours before recharging is necessary.

Other Batteries

You can power the LI-6400 with any 12 V battery (car, marine, etc.) that has at least a 1.5 Amp-hour capacity. To connect an alternative battery to the console, you'll need a 318-02031 connector (the kind that is on the 6400-03 battery). One simple way to get this connector already attached to a cable is to remove it from a dead 6400-03 battery. Alternatively, you can order these connectors by themselves, or else order part number 9960-120, which is this connector attached to 10 feet of cable, or else obtain a 9960-062 (the cable that runs between the console and an LI-6020 charger), and cut it in half to get two short (2.5 ft.) versions of a 9960-120.

The 6400-03 batteries are protected with a 10 Amp automotive fuse (see **Replacing the Battery Fuse** on page 19-9). You might wish to do the same with your alternative battery.

6400-70 AC Module

This optional accessory allows the LI-6400 to be powered from mains power. It consists of a transformer, and a battery-shaped box that can fit in the console (Figure 2-14).

Figure 2-14. The 6400-70 AC Module fits in a battery compartment in the console. If you install a battery in the other compartment, and plug it into the module (as shown here), that battery will trickle charge, but more importantly, provide continued operation for 1-2 hours in the event of a mains power failure.

■ **Notes on using the 6400-70 AC Module**

• **A battery is not required**
The module will power the LI-6400 by itself. However, plugging a battery into the module will provide 1 or 2 hours continued operation in case of mains power failure.

• **Use both plugs - most of the time**
The AC module has two plugs that go into the console's battery connectors. For normal operations, plug both of them in.

• **Hot swapping**
You can switch from AC operation to battery operation by unplugging one of the module's plugs from the console, and replacing it with a fresh battery. Then disconnect the other plug, and replace it with second battery if you wish. Reverse the process to switch from battery to AC operation.

Either way, observe the caution in Figure 2-15.

Safe for extended operations

Battery safely trickle charges, and is available to run the instrument if mains power fails.
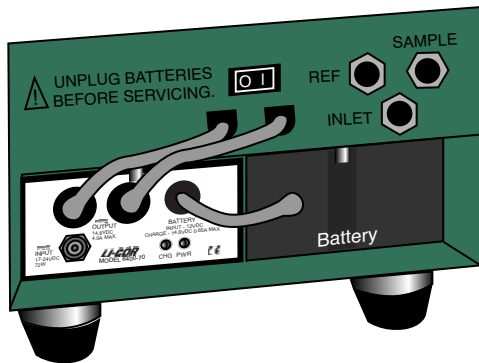
Don't do this for more than a minute

Battery is being charged in an uncontrolled manner. Damage could result to the battery and/or the transformer.
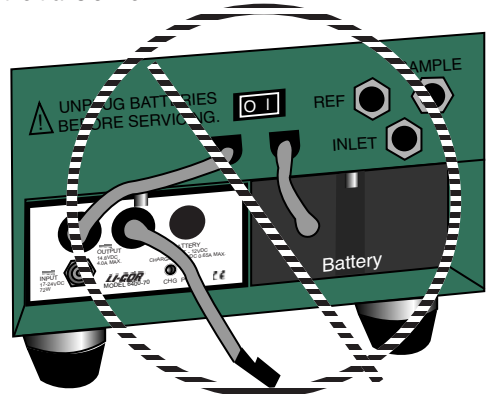
Figure 2-15. Avoid extended (more than a minute) operations with a battery and one of the AC module's plugs connected to the console. In this configuration, you will be providing uncontrolled charging of the battery, which could damage the battery and/or the transformer.

Important Safety Notice

To minimize shock hazard:

The 6400-70 AC Module MUST be connected to an electrical ground through a three-conductor power cable, with the third wire firmly connected to an electrical ground (safety ground) at the power outlet.

Any interruption of the protective (grounding) conductor or disconnection of the protective earth terminal may result in a potential shock hazard at the LI-6400 instrument chassis that could result in personal injury.

ALSO: If you connect an analog output from the LI-6400 to the LI-610 Dew Point Generator, and are powering both units by AC power, a "ground loop" can develop, causing unwanted signal noise that can affect the operation of the LI-610. If you are using the LI-6400 and LI-610 in this manner, we recommend that you isolate the two circuits by operating one or both of the instruments with battery power.

Installing System Software

Note: Installing software is <u>not</u> something you have to do to make the LI-6400 work when you get it from the factory. It comes with software installed and ready to use. We change this software periodically, to fix bugs and add enhancements[2], and to take advantage of these changes, you have to install the new software into the LI-6400. To see what version of software you have installed, select "About this unit" in the Home Menu (see Figure 3-6 on page 3-9). You can see what the latest available system software is (and download it) by checking our web site: www.licor.com.

Hardware Requirements

Version 6.x runs on either the current 400 MHz board (LI-6400XT), or the older 200 MHz board (version 5.x software). If an LI-6400 has version 3.x or 4.x, it has the original digital board (18 MHz). Any LI-6400 can be upgraded to an LI6400XT via part number 6400-926. Any version 5.x LI-6400 can install and run version 6.x software, but will not have any of the flash memory or Ethernet capability.

---

[2]·Or fix enhancements and add bugs…

Guided Tours

3

Learning how to make it work

3

Guided Tours

This chapter teaches you a) how to operate the LI-6400, and b) how the LI-6400 accomplishes its tasks. We do this with a series of guided tours. We recommend that you follow along on your instrument or LI6400Sim simulator. You won't need plant material for these tours - that will come in Chapter 4.

Before You Start

Here are some things you should know about the display and keypad (Figure 3-1 on page 3-3).

Cursor Control Keys

The cursor control keys (↑, ↓, ←, →, **pgup**, **pgdn**, **home**, and **end**) appear on either side of the front panel. The left group does the same thing as the right group, and it doesn't matter which you use. Similarly, there are two **enter** keys and two **labels** keys.

Function Keys

Keys labeled **f1** through **f5** below the display are called *function keys*, and often have labels associated with them on the bottom line(s) of the display. When there are multiple definitions for these keys, the **labels** key can be used to cycle through them (and **shift** + **labels** to go backwards). Sometimes, the labels remain hidden even though function keys are defined and active; pressing **labels** will make the labels temporarily appear.

Display

The display has independent text (8 lines, 40 characters per line) and graphics (64 dots high, 240 dots wide) modes. In this tour we will use both.

You can adjust the contrast by pressing **ctrl** + **shift** + ↑ and **ctrl** + **shift** + ↓. Also, if it is so equipped, you can toggle the display backlight on and off by pressing **ctrl** + **home**. All the hot keys are listed in Table 3-1 on page 3-3.
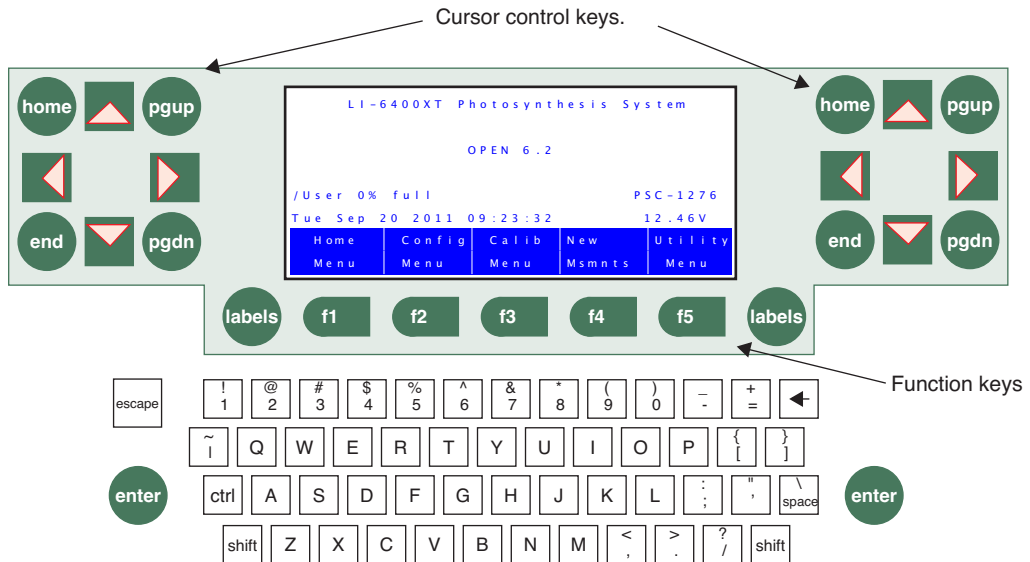
Figure 3-1. The LI-6400 keypad. The cursor control keys, labels, and enter are paired to facilitate access with either hand.

Table 3-1. Hot key combinations - valid anytime.

Press...	To Do...
ctrl shift ↑	Darkens display contrast.
ctrl shift ↓	Lightens display contrast.
ctrl shift home	Toggles display backlight (if installed).
ctrl shift →	Turns graphics mode on. If already on, turns text mode off.
ctrl shift ←	Turns text mode on. If already on, turns graphics mode off.
ctrl escape	Aborts the current application.
ctrl shift escape	Reboots.
ctrl shift end	(XT Only) Unmounts compact flash card (if inserted).

Table 3-2. New Measurements key summary.

Press...	To Do...
[	Enters/Exits Diagnostics Mode
]	Enters/Exits Graphics Mode
a..z	(Text) Selects display line. (Diagnostics) Selects display screen (a - i). (Graphics) Selects graphics screen (a - h).
0..9	(Text and Graphics) Selects function key level.
home **end** **pgup** **pgdn**	(Text) Implements a Display Group key.
ctrl home **ctrl end** **ctrl pgup** **ctrl pgdn**	(Text) Defines a Display Group key.
↑ ↓	(Text) Selects new display line. (Graphics) Selects new display screen.
← →	(Text) Changes to next display on selected line. (Graphics) Changes chart selection.
ctrl z	Toggles warning messages on/off.
ctrl s	(Graphics) Stores current graphics display to /User/Images/RTG_*yyyy-mm-dd_hh-mm-ss*. (Text & Diagnostics) Stores a system snapshot (variables, values, settings, etc.) to a text file names /User/Snapshot_*yyyy-mm-dd_hh-mm-ss*.

Tour #1: OPEN Overview

Running OPEN

OPEN is the name of the program that controls normal operation of the LI-6400XT and the one you'll be running most of the time. This program begins to run automatically after you power on (unless you intervene), as you will see in the following steps.

1   Turn the LI-6400 ON

About 10 seconds will pass while the display shows[1]:

    INITIALIZING.

followed by a series of start up messages. Eventually, you should see:
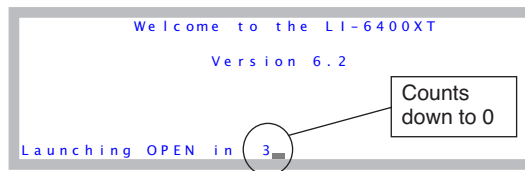
Figure 3-2. OPEN's Autostart countdown is 5 seconds long. Press **enter** *to skip it, or press* **escape** *to prevent OPEN from loading.*

2   Press enter or wait

If you press **escape**, you'll prevent OPEN from loading, and will access the LPL screen (**The LPL Screen** on page 5-19). If you press **enter** (or nearly any other key), OPEN is loaded (Figure 3-3), which takes about 5 seconds.
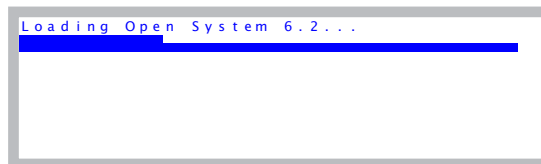
Figure 3-3. OPEN's bar chart is displayed while loading.

[1.]If your LI-6400 doesn't behave as described here, refer to **Power On Problems** on page 20-2.

3   **If asked, select a Configuration**
Once OPEN's bar chart finishes, you might be asked to pick a configuration
file (Figure 3-4). If there is only one configuration file on your instrument
(e.g. a new instrument), then you won't be asked to do this.

```
Dir:  /User/Configs/UserPrefs
══════════Select  a  new  configuration══════
FactoryDefault_6.2.xml
LCF.xml
LED.xml
OpaqueConifer EB.xml
RGB on2x3.xml
+ Dir      ♦          ♦        ♦ CANCEL♦ SELECT
```
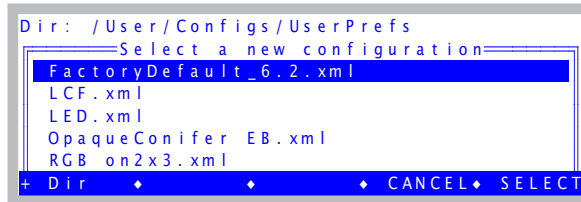
*Figure 3-4. If multiple configuration files exist, you are asked to select one
when OPEN first runs.*

A configuration file contains settings and values used by OPEN. Configura-
tion files are easy to create and modify. There is an introductory tour later in
this chapter on page 3-90.

For now, however, select FactoryDefault_6.2. When it is highlighted, press **enter**.

4   **Power up the IRGAs.**
After a few more seconds, and several more messages, you'll be asked

```
Ci - ok
C
C  Is the Chamber/IRGA connected ?▮
R
C                              ═(Y/N)═
AHs/Cs - ok

Cleanup...
```
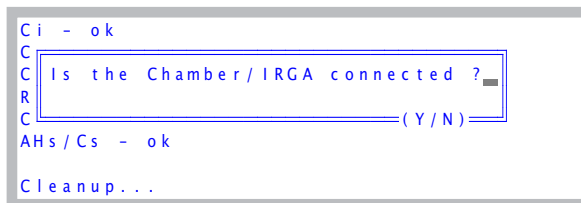
Press **Y** when you have both electrical cables between the console and the sen-
sor head connected.

OPEN's Main Screen
After some more messages, OPEN's main screen appears (Figure 3-5). This
screen represents the home base of operations for OPEN. The function keys
(**f1** through **f5**) have 2-line labels above them on the display, which are used
to access the various menus and routines available in OPEN.

Normally at this point, one would do a series of checks before making mea-
surements, and these are described in **Preparation Check Lists** on page 4-2.
We'll skip these checks for purposes of our tours.

Connecting and Disconnecting the Chamber/IRGA.

Once the OPEN screen appears, you should *not* connect or disconnect the chamber or IRGA cables while OPEN is running without first putting the instrument "to sleep" via the Sleep Mode function in OPEN's utility menu. You run the risk of blowing fuses, and (when the 6400-02 light source is on) the chamber connector can carry dangerous voltage (>100 V).

Powering Off
OPEN's main screen is a safe place to be (or return to) when you are ready to power the instrument off. If you are anywhere else, you may have files open, and could risk losing data if you power off then.
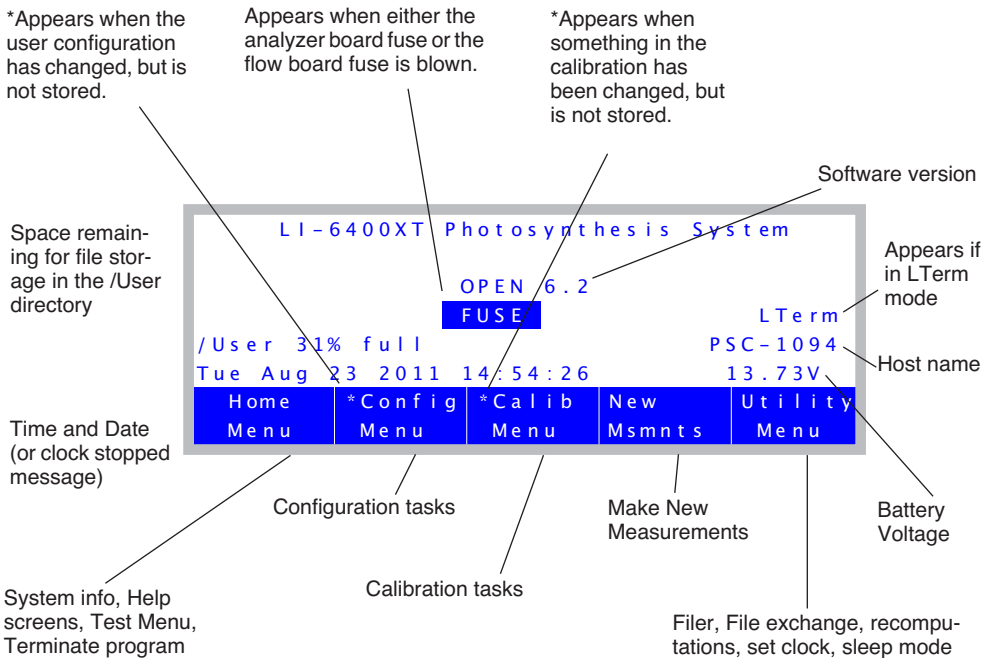
Figure 3-5. OPEN's main screen.

Alert Messages in OPEN's Main Screen

OPEN's Main Screen can show four alert messages. Here's what they mean, and what to do about them.

FUSE

If the instrument has serial number PSC-401 or above, or if it has been up-graded with a new backplane board, the FUSE message will appear if the flow board fuse or the analyzer board fuse is blown. See **Replacing the Fuses** on page 19-11. (To find out if your upgraded instrument is capable of producing this alert message, check for "Fuse Detection" in the "About this Unit" screen, found in the Home Menu, as shown in Figure 3-6 on page 3-9.)

* in Calib Menu label

This message appears if something calibration-related has changed and not been stored. To see what it is, press **f3** (**Calib Menu**), and follow the asterisks. For details, see **Managing Calibration Data** on page 18-2.

* in Config Menu label

The instrument's configuration has been changed, but is not stored. To see what has changed and/or store it, press **f2** (**Config Menu**) and follow the asterisks (Figure 3-101 on page 3-92).

Clock Stopped

This message will appear instead of the time and date if the real time clock is not operating. See **Real Time Clock Problems** on page 20-5.

The Home Menu

The Home Menu is not used very often, but we will point out some things that might bring you here.

Figure 3-6. The Home Menu is a gateway to status information, a suite of diagnostic tests, and the exit.

1   **Access the Home Menu**
From OPEN's Main Screen, press **f1** to access the Home Menu.

2   **Select "About this Unit"**
The "About this Unit" entry will show software versions, date of last service, and other useful information (Figure 3-6).

3   **Press escape**
To return to the Home Menu

4   **Select "Contrast & Backlight..."**
This will open a screen that lets you adjust the contrast (press ↑ or ↓), and turn the display backlight on and off (press **home**).

Note: you do not have to actually be in this screen to make these adjustments. They can be done anytime by pressing **shift**+**ctrl**+↑, **shift**+**ctrl**+↓, and **shift**+**ctrl**+**home**.

5   **A word about exiting OPEN**
The bottom entry in the Home Menu will ask if you wish to terminate OPEN. If you press **Y**, the program will turn off the flow control board and the analyzer board. If you press **N**, the boards are left on. It is not necessary to do either of these when turning off the instrument: you can just press the power switch while in OPEN's main screen.

```
Exit OPEN: Power IRGAs off?

 Y - power off
 N - exit, leave on
<escape> - do not exit

```

6   **Return to OPEN's Main Screen**
Press **escape** to exit the Power off screen, then **escape** again to exit the Home Menu.

The Config Menu

The Config Menu (Figure 3-7 on page 3-11), accessed from OPEN's Main Screen by pressing **f2**, is used fairly often. As the name implies, anytime you need to do something involving the instrument's configuration, you will likely be visiting this menu. This menu has its own tour that starts on page 3-90.

Building a New Configuration on page 16-5.

```
═════Make a NEW Config for...═══════════
      ▸Light Sources
      ▸Clear 2x3 Tops
      ▸Larger Chambers
      ▸Other Chambers
      ▸Special Configurations
  ▾/▸      ◆          ◆        ◆ EXIT  ◆ SELECT
```

Open a Stored Configuration on page 16-9.

```
Dir: /User/Configs/UserPrefs
═══════Select a new configuration═══════
FactoryDefault_6.2.xml
 LCF.xml
 LED.xml
 OpaqueConifer EB.xml
 RGB on2x3.xml
+ Dir     ◆          ◆        ◆ CANCEL◆ SELECT
```

```
═════Config Menu═══════════════════════
   New...
   Open...
   Save as...
   View/edit...

  ▾/▸      ◆          ◆        ◆ EXIT  ◆ SELECT
```

Save the Current Configuration on page 16-10.

```
Dir: /User/Configs/UserPrefs
File:MyConfigExample.xml
Filter: ver 6 config files (*.xml)
═════════════Save this config═══════════
 Default+LED.xml
 Default+RGB.xml
 FactoryDefault_6.1.xml
+ Dir     ◆          ◆ Filter◆ CANCEL◆ SELECT
```

Editing the Current Configuration on page 16-11.

```
═════FactoryDefault_6.2.xml═════════════
→└▾open
   ▸light
   ▸comps
   ▸prompts
   ▸constants
   ▸fctkeys
  ▾/▸      ◆          ◆        ◆        ◆Done
```

Figure 3-7. The Config Menu.

The Calib Menu

The Calib Menu (Figure 3-8) is used when calibrating a CO_2 mixer or light source, and for zeroing and spanning the IRGAs.

```
═══Calib Menu════════════════════════
    ▸View Settings
     Flow meter zero...
    ▸IRGA
    ▸CO2 Mixer
     ParIn zero...

 ▼/▸    ♦        ♦       ♦ EXIT  ♦ SELECT
```
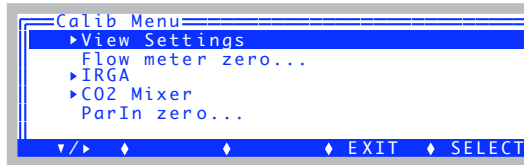
Figure 3-8. The Calib Menu.

When expanded, the Calib Menu looks like this:

▼ **View Settings**
    View Current...
    View History...
    View / Edit Accessory Cals...
  Flow meter zero...
▼ **IRGA**
    IRGA zero...
    IRGA span...
▼ **CO2 Mixer**
    Calibrate...
    Plot...
  ParIn zero..

Chapter 18 discusses this menu, but for now we will just do a simple task: view the current settings.

■   **View Current Settings**

We will view the calibration "tree", which is a structure that contains all the calibration-related values and settings.

1   **Select "View Current..."**

If necessary, expand the View Settings entry by highlighting it and pressing **f1**. Then highlight View Current... and press **f1**. You should see a tree whose root node is li6400.

```
═══Current Settings═══════════════════
→ └▼ li6400
     ├▶factory
     └▶user



 ▼/▶    ♦        ♦       ♦       ♦Done
```

There are two sections to this tree: *factory* and *user*. *Factory* holds calibration coefficients that are determined, as you might expect, at the factory. *User* holds calibration values that are determined when you do calibrations, such as zeros and spans.

▼ **li6400**
    ▼ **factory**   *These settings are determined at the factory.*
        unit= "PSC-1094"
        serviced= "7 Feb 2007"
        fuseaware= 0
        co2mixer= Yes
        ▼ **co2**
            coeffs= 0 0.2053 3.3708e-05 1.1175e-08 -4.2031e-13 2.1027e-17
            dvdt= -3
        ▼ **h2o**
            coeffs= 0 0.00567039 2.28394e-06 5.70608e-11
            dvdt= -2.3
        flow= 0 0.3788
        press= 88.692 0.00552
    ▼ **user**   *These values change when you calibrate.*
        flow_zero= -2.45
        ▼ **irga_zero**
            co2= 78.1 -859.4
               at= 26.352
            h2o1= 17.2 78.1
               at= 27.07
        ▼ **irga_span**
            co2= { 0.994 0.983 }
            h2o= { 1.01 1.001 }
        ▼ **irga_match**
            co2= { 0.26388 -2.49324 }
            h2o= { 0 0.0681999 }
        ▼ **co2_mixer**
            pump_mv= 4600
            ppm= { 2265.32 1068.76 424.10 258.87 117.53 67.910 44.25 42.19}
            mv= { 5000 3000 1500 1000 500 300 200 100 }
        parin_offset= 0.433045
        ▼ **led_cal**
            unit= "SI-1267"
            mv= { 10 25 50 100 1000 2000 3000 4000 5000
            qntm= { 5.04 15.9 38.14 89.58 1053.9 2057.2 2996.5 3888.9 3940.0}
        ▼ **lcf_cal**
            unit= "LCF-0304"/unit
            ▼ **red**
               mv= { 50 100 200 400 800 1500 }
               qntm= { 6.07527 13.6724 33.1146 80.8811 183.756 363.114 }
            ▼ **blue**
               mv= { 100 500 1000 2000 3000 4000 5000 }
               qntm= { 2.85 8.66 14.95 25.24 34.68 42.88 50.3077 }

2   **Explore**
Expand various nodes of the tree to see what's there. For details, see **Managing Calibration Data** on page 18-2.

3   **Return to the Calib Menu**
We're done, so press **escape** or **f5** (**Done**).

4   **Return to OPEN's Main Screen.**
Press **escape** to leave the Calib Menu.

The Utility Menu

The Utility Menu (Figure 3-9) has a collection of useful things that you will occasionally need to do while using OPEN.

```
══════Utility Menu═══════════════════
     ▸Files
     ▸Communications
     ▸Graphics
      Set the Clock...
      Sleep...

   ▾/▸    ♦          ♦         ♦ EXIT  ♦ SELECT
```
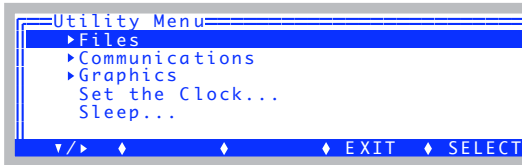
Figure 3-9. The Utility Menu.

▾ **Files**
    Access the Filer..   *Chapter 10*.
    Create a new AutoProgram...   *Making Your Own AutoPrograms* on page 9-47.
    Create a new (empty) file...   *Standard Edit* on page 5-15.
    Recompute a stored data file...   *Discussed in Chapter 13*.
    Reinstall basic config files...   *Discussed on page 16-2*.
    ComputeList List->Module...   *Converting Lists to Modules* on page 15-33.
▾ **Communications**   *Discussed in Chapter 11*.
    Network Status...   *Discussed on page 11-9*.
    Configure the COMM port...   *Discussed on page 11-26*.
    File Exchange Mode...   *Discussed on page 11-28*.
    Connect to li6400.licor.com...   *Connection over the Internet* on page 11-38.
▾ **Graphics**
    Graph a data file...   *Discussed in Chapter 12*.
    View Stored Images...   *Graphics Restore on page 21-15*.
Set the Clock...   *SETCLOCK on page 21-17*.
Sleep...

New Measurements

The final stop on this first tour is New Measurements mode (press **f4**). This is where you make measurements, control chamber conditions, log data, and the like. It has its own tour, so keep reading.

Tour #2: New Measurements Mode Basics

New Measurements mode is entered by pressing **f4** (**New Msmnts**) while in OPEN's main screen. When using the LI-6400, you will likely spend most of your time here, coming out only to do configuration changes, calibrations, lunch, and other ancillary operations.

The New Measurements screen (Figure 3-10) has three rows of variables with highlighted labels above each. A row of function key labels appears at the bottom.
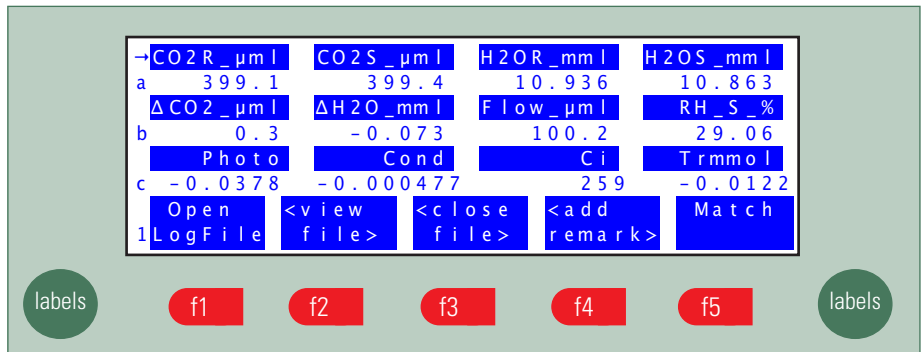
Figure 3-10. New Measurements screen text display.

In New Measurements mode, there are two other display modes for providing information. In addition to this text mode, there is Graphics mode and Diagnostics mode. Figure 3-11 summarizes how to switch between these modes.

.

Figure 3-11. Getting around in New Measurements between the three display modes.] *is the right square bracket key, and* [*is the left square bracket key.*

We'll spend the most time discussing text mode, but before we do, here's a quick glimpse of the other two modes:

1   Switch to Graphics
Press] (the right square bracket key), and you'll see real time graphs. Press] again, and you'll return to the text screen.

2   Switch to Diagnostics
Now press [, and you'll enter the diagnostics mode. Press [again, and you'll return to the text screen.

We'll be visiting graphics and diagnostics again in our tours in this chapter, and will cover more details then. The "reference section" for all three display modes is in Chapter 6.

Function Keys

Text mode has 7 sets of function key definitions (or 10 with the Leaf Chamber Fluorometer). The current function key level number is displayed in the bottom left hand corner of the display. If you press **labels** seven times, you'll see them all. Here is a shortcut: press the number keys (**1** through **7** on the keypad), and you'll jump directly to that function key level.

Open 1 LogFile	<view file>	<close file>	<add remark>	Match

■ **To change function key levels:**

1   Press 1 through 7…
This directly accesses the selected level.

2   …or, press labels or shift labels.
labels takes you ahead a level, and **shift labels** takes you back a level.

The labels are summarized in Figure 3-12. Note that the level 7 key labels are all blank in the default configuration, and are available for your own use (described in **Defining Fct Keys** on page 16-22).

1	Logging control; IRGA matching	
2	Environmental control keys (fan, CO_2, humidity, temp, light)	
3	System and user-defined constants	
4	Real Time Graphics control	
5	AutoProgram control, defining what is logged	
6	Text display control	
7	User definable	

Open 1 LogFile	<view file>	<close file>	<add remark>	Match
LeafFan 2 Fast	*Flow→ 500 µm	Mixer OFF	Temp OFF	Lamp OFF
Area= 3 6	StmRat= 1	Sys&Usr Consts	Prompts* off	Prompt* All
 4	Import GRAPH	View GRAPH	Setup GRAPH	
AUTO 5 PROG		LOG OPTIONS	Define Stablty	Define Log Btn
Display 6 QuikPik	Display List	What's What	Display Editor	Diag Mode
 7	Oxygen= 21.0			

*Figure 3-12. Summary of New Measurement's default function key labels. Keys **f4** and **f5** of level 3 (marked with *) only show these labels when user constants are defined (described in **Defining Prompts** on page 9-21).*

■ **Example: Changing the chamber fan speed**

1 **Access the chamber fan control function key**
Press **2**. The function key labels will change as shown in Figure 3-13.

2 **Access the control display**
Press **f1**. The fan speed control box will pop up on the display.
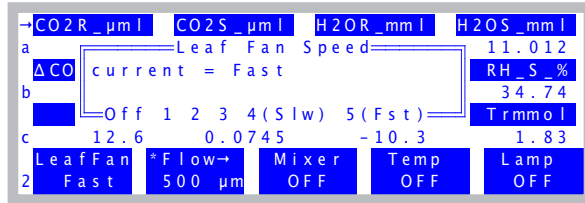
*Figure 3-13. Controlling the fan speed. Press **F**, **S**, or **0** for fast, slow, or off, or **1** thru **5**.*

3 **Turn the fan off**
Press **0** (the letter, or the number **0** - both happen to do the same thing here), and the fan will stop. If you listen, you should hear a drop in the noise level, especially if the chamber is open.

4 **Turn the fan back on (fast)**
Press **f1**, then **f**, and the fan will resume running.This is the speed you should normally use. However, a reduced fan speed is helpful for keeping the surface humidity of the leaf high, which can prevent stomatal closure in stressed plants during measurement.

Text Display

Twelve variables are displayed in Figure 3-10 on page 3-15, but there are many more available. Here's how you display them:

■ **To change a display line:**

1 **Use ↑ or ↓ to select a line**
To the left of each label value line is a letter, and to the left of one label line is an arrow (→). The → indicates the "change line". You can select the change line by pressing the ↑ or ↓ keys.

2 **Press a letter**
Up to 26 display lines can be defined (this corresponds to keys **A** through **Z**), although the default display defines only *a* through *l*. For example, put the → marker on the bottom line, and press **A**. The display will change as shown in Figure 3-14.
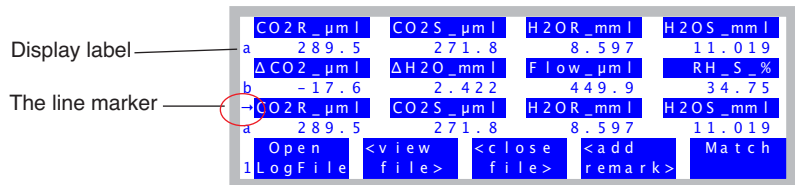
Figure 3-14. To change a display line, position the marker using ↑ or ↓, then press a letter key. In this figure, we've put the marker on the bottom line and pressed **A**.

3 **Alternatively, press ← or →**
Pressing the horizontal arrow keys will scroll the change line through all the possible displays, but it is faster to use the letter shortcuts.

4 **Go exploring!**
Table 3-3 lists the default displays, and their variables. Use the arrow keys and letter keys to view them yourself. Note that these display definitions can be modified for what variables are shown and where (Chapter 6).

Display Groups

You can change all three lines of the text display with a single keystroke. There are 4 keys reserved to do this: **home**, **pgup**, **pgdn**, and **end**.

To define (or redefine) a display group, get the display arranged the way you want it, then hold the **ctrl** key down and press the group key (**home**, **pgup**, **pgdn**, or **end**) of your choice. That group key is now defined. Once a group key is defined, you can change all three display lines to that arrangement sim-

ply by pressing that group key.
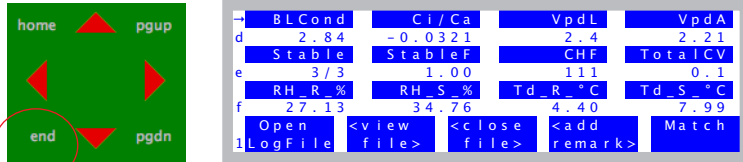
*Figure 3-15. Group keys (**home**, **pgup**, **pgdn**, and **end**) will change all three display lines. The default setting for end, for example, is to bring up display lines d, e, and f.*

■   **Example**
Make the **home** key display lines *a*, *b*, and *c*, and the **pgup** key display lines *g*, *h*, and *k*.

1   **Define home key**
Put lines *a*, *b*, and *c* on the display, and press **ctrl** + **home**.

2   **Define pgup key**
Put lines *g*, *h*, and *k* on the display, and press **ctrl** + **pgup**.

3   **Switch displays**
To view *a*, *b*, and *c*, press **home**. To view *g*, *h*, and *k*, press **pgup**.

You have seen how to define the group keys "on the fly". Actually, the display arrangement (lines *a* through *z*) as well as the group key definitions are all user-definable, either through the level 6 function keys in New Measurements mode (**Display Editor** on page 6-6), or via View/Edit in the Config Menu.

Table 3-3. The variables and how they are grouped for viewing in the default display configuration. See also Table 14-10 on page 14-23.

Group	Label	Description
A	CO2R_μml	Reference cell CO_2 (μmol CO_2 mol^{-1})
	CO2S_μml	Sample cell CO_2 (μmol CO_2 mol^{-1})
	H2OR_mml	Reference cell H_2O (mmol H_2O mol^{-1})
	H2OS_mml	Sample cell H_2O (mmol H_2O mol^{-1})
B	ΔCO2_μml	CO_2 delta (sample - reference) (μmol CO_2 mol^{-1})
	ΔH2O_mml	H_2O delta (sample - reference) (mmol H_2O mol^{-1})
	Flow_μml	Flow rate to the sample cell (μmol s^{-1})
	RH_S_%	Relative humidity in the sample cell (%)
C	Photo	Photosynthetic rate (μmol CO_2 m^{-2} s^{-1})
	Cond	Conductance to H_2O (mol H_2O m^{-2} s^{-1})
	Ci	Intercellular CO_2 concentration (μmol CO_2 mol^{-1})
	Trmmol	Transpiration rate (mmol H_2O m^{-2} s^{-1})
D	BLCond	Total boundary layer conductance for the leaf (includes stomatal ratio) (mol m^{-2} s^{-1})
	Ci/Ca	Intercellular CO_2 / Ambient CO_2
	VpdL	Vapor pressure deficit based on Leaf temp (kPa)
	VpdA	Vapor pressure deficit based on Air temp (kPa)
E	Stable	Stability status: # Stable / # Checked
	StableF	Stability status as a decimal value
	<letters>	Stability flags: 1's and 0's for each variable
	TotalCV	Sum of the CVs of the stability variables
F	RH_R_%	Relative humidity in the reference cell (%)
	RH_S_%	Relative humidity in the sample cell (%)
	Td_R_C	Dew point temp in the reference cell (C)
	Td_S_C	Dew point temp in the sample cell (C)

Table 3-3. (Continued)The variables and how they are grouped for viewing in the default display configuration. See also Table 14-10 on page 14-23.

Group	Label	Description
G	Prss_kPa	Atmospheric pressure (kPa)
	ParIn_μm	In-chamber quantum sensor (μmol m^{-2} s^{-1})
	ParOutμm	External quantum sensor (μmol m^{-2} s^{-1})
H	Tblock°C	Temperature of cooler block (C)
	Tair°C	Temperature in sample cell (C)
	Tleaf°C	Temperature of leaf thermocouple (C)
	CTleaf	Computed leaf temp (C). Same as Tleaf°C unless doing energy balance.
I	HH:MM:SS	Real time clock
	Program	Shows AutoProgram status
	CHPWMF	Status word (summary of line J)
	Battery	Battery voltage (V)
J	CO2	Status of CO_2 IRGAs
	H2O	Status of H_2O IRGAs
	Pump	Status of pump
	Flow	Status of Flow controller
	Mixr	Status of CO_2 mixer
	Fan	Speed of chamber fan
K	Program	Shows AutoProgram status
	ProgPrgs	AutoProgram step counter
	FwMxCrLp	Numerical summary of the four stability flags
	Stable	Stability status
L	CRagc_mv	Reference CO_2 AGC (automatic gain control) signal, in mV
	CSagc_mv	Sample CO_2 AGC signal
	HRagc_mv	Reference H_2O AGC signal
	HSagc_mv	Sample H_2O AGC signal
M	matchCO2	CO2R at last match
	matchH2O	H2OR at last match
	mchElapsed	Time elapsed since last match

Real Time Graphics Displays

New Measurements mode also provides a method to monitor these variables graphically. To access the real time graphics (RTG) displays, press **4**, then **f3 (View Graphs)**. (There is also a short cut: **]** the right square bracket key.) The display should show something like Figure 3-16.
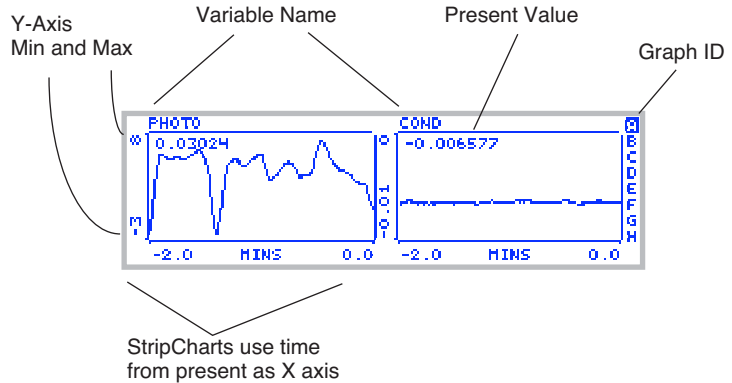
Figure 3-16. The default RTG display A in New Measurements mode shows photosynthesis and conductance.

There are 8 RTG displays available in New Measurements mode, and they are designated by the letters *A* through *H*. Figure 3-16 shows the *A* display. Each display can have 1, 2, or 3 plots. Each plot can be a StripChart - a variable plotted against time with a continuous line, or an XYChart (Figure 3-17) -one variable plotted against another using discrete points, usually representing logged observations.
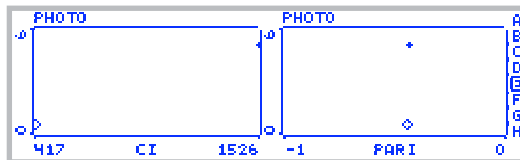
Figure 3-17. An XYChart. Present value is shown by the diamond. Logged data are shown with a +.

Here are two navigational rules:

• **To change RTG Displays**
Press the letters **A** through **H** to jump directly to a particular display. You can also use the arrow keys ↑ or ↓ to step through them sequentially. The indicator at the right side of each graph will show which graph you are viewing.

If no plots are defined for a particular RTG display, it will appear as in Figure 3-18.
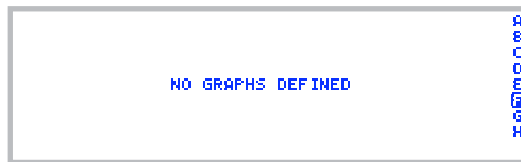
Figure 3-18. An undefined graphics display.

• **To exit RTG Mode**
Press **escape** or **]** to leave graphics mode and go back to the text displays.

Graphics Function Keys

There are three levels of function keys associated with New Measurement's graphics displays. Activate them by pressing **labels**. The plots shrink a bit to make room for the labels.
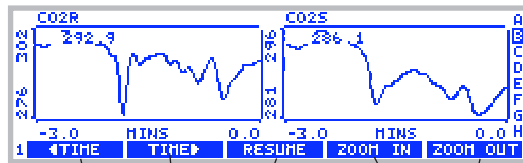
Figure 3-19. Real Time Graphics has three levels of function keys. Use **labels***, or* **1***,* **2***, and* **3** *to change them.*

■ **Graphics Function Key Tour**

Let's try out some of the function keys. Find a graphics display that has strip charts on it, and press **labels** to bring up the labels.

1 **Note that labels are an attribute of each graph**

The function key labels for each graphics display are independent. Change to another display, and the labels go away. That is, turning them on for one graph does not turn them on for any other.

2 **There are three levels of function keys**

You can navigate them just like in text mode: press **labels**, or press **1**, **2**, or **3**. You can also use this shortcut to bring up the labels. When real time graphics labels are not visible, the function keys are not operative.

3 **Turn off the labels**
Press the **HIDEKEYS** key (**f5** level 2), and the labels disappear. Another way to turn off the labels is to press the number **0**.

4 **Mark the graphs**
Press **MARK** (**f1** level 2**)**, and a small arrow (↑) will appear on each StripChart on this display.
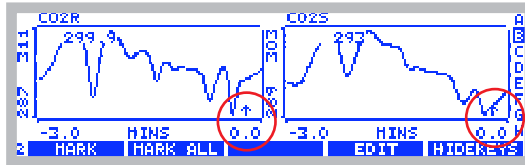
*Figure 3-20. **MARK** puts a small marker on StripCharts. This also happens when data is logged.*

As you might expect, **MARK ALL** marks all of the displays (A - H). Plots are marked with a small '+'. These marks also happen automatically to all graphs when you log data (a topic that lies ahead of us).

5 **Travel back in time**
Our StripCharts in Figure 3-20 are showing 120 seconds of activity, but they remember more than that (10 minutes by default, but it's user definable). Press **1** to access the time control function keys, and press **F1**(◄**TIME**) several times (Figure 3-21). Note that the time axis continues to show 2 minutes worth of data, but at earlier and earlier times. Note too that the plots stop updating while we scroll back, although data continues to be collected and saved. The axis time labels still reflect time since the present.
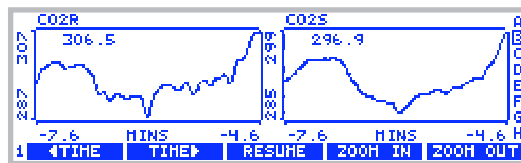
Figure 3-21. Scroll back in time. Here we are looking at data between 4.6 and 7.6 minutes ago.

To resume normal plotting, you can press **f3** (**RESUME**), or scroll back to the present with **f2** (**TIME**►), or simply wait for about 15 seconds, in which case plotting automatically resumes.

6 Zoom in, Zoom out

The remaining two time control keys, **ZOOM IN** and **ZOOM OUT**, change the range of the time axis on StripCharts. Thus, if you wish to see the big picture, start pressing **f5** (**ZOOM OUT**) (Figure 3-22).
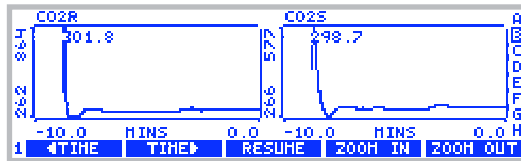
Figure 3-22. The ZOOM OUT key shows more data on the plots. Here we have gone to 10 minutes.

Now, press **ZOOM IN** several times to return to a 2 minute time axis.

7 Selecting a Chart

Normally, the time control keys operate on all of the StripCharts in the display being viewed. (They do nothing to XYCharts). You can, however, select an individual chart by pressing the left or right arrows (← →). An inverse bar appears over the selected plot (Figure 3-23). Now, pressing a time control key will operate on only that plot.

This plot is selected.
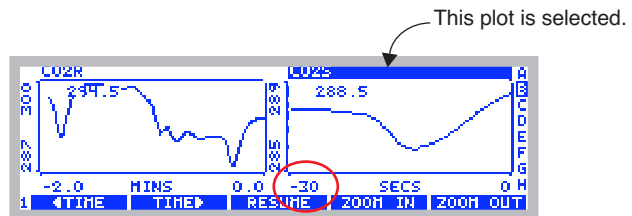
Zoom In only operated on the selected plot.

Figure 3-23. Select plots using ← or →. When no plot is selected, the time control keys work on all plots in the display.

8 Exit graphics mode

There are two ways back to normal text mode: **escape** or **]**. Note that the **]** key brings you into graphics mode, and also takes you out.

Tour #3: Controlling Chamber Conditions

Chamber conditions are controlled from New Measurements mode via the function keys on level 2 (Figure 3-24).

CO_2 control (requires optional 6400-01). Set reference or sample CO_2 target.

Light control (requires optional 6400-02(B) or 6400-18 RGB or 6400-40 LCF. Says "-none-" if not configured for a controllable light source.

2 LeafFan Fast	*Flow→ 500 µm	Mixer OFF	Temp OFF	Lamp OFF

Controls flow through chamber, either as a constant, or variable to hold humidity constant.

Controls coolers on the IRGA. Can target constant block temp, or constant leaf temp.

Figure 3-24. Summary of the environmental control function keys.

The main control areas are chamber wind speed, flow/humidity, CO_2, temperature, and light. The key labels for **f2** through **f5** indicate the current state of the control, along with the target value of active controls. This tour will acquaint you with each control. Details of OPEN's chamber control are given in Chapter 7.

Fixed Flow Operation

Flow and humidity are grouped together into one control. You can specify a fixed flow rate (and let humidity vary) or a fixed humidity (and let flow vary). Confused? Keep reading.

Experiment #1   **Humidity vs. Flow Rate**
This experiment illustrates the relationship between chamber humidity and flow rate.

1   **Simulate a leaf with filter paper**
Use Whatman® #1 filter paper (or even towel paper), folded a couple of times, and moist but not dripping. Clamp your "leaf" into the leaf chamber

(Figure 3-24). Use the adjusting nut to make the chamber seal snugly, but not too tight.

Adjustment knob

Moistened filter paper, folded twice
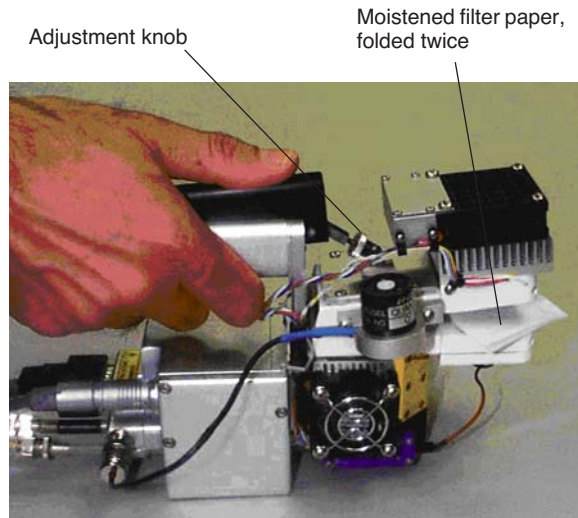
Figure 3-25. Set the adjustment knob so that the chamber gaskets are slightly compressed when closed with no leaf. Then, open the chamber, tighten one more half turn, and close onto the leaf or filter paper.

2   Set soda lime to full bypass, desiccant to full scrub
The soda lime tube should be the one closest to you on the left side of the console. Roll the adjustment knob toward you for bypass. On the desiccant tube, roll the knob away from you for scrub. There's no need for strong fingers here; when the knob gets resistive at the end of its travel, quit. It is far enough.

3   Use a high flow rate: 700 μmol s^{-1}
Press **2** (if necessary, to bring up the level 2 function keys). Then press **f2** and select a flow rate of **700** (Figure 3-26).

1. If not already set for "Flow Rate", move cursor to the Control node, press **Edit**,....

```
┌──Environmental  Controls══════
│▼FLOW:  Flow  Rate
→├─ Control=  Flow_µm
│ └ Target=  500 µmol/s

│══T) reTarget, exit   O) Off, exit══
│ ▼/▶   ◆Edit    ◆    ◆cancel ◆ Keep
```

...and select it from the list.

```
┌──E──┌═════Flow  Control  Types══════
│▼F │ Off
│   │ ◆  Flow  Rate
│   │ H2O  mole  fraction
│   │ Relative  Humidity
│   │ Vapor  Pressure  Deficit
│──T│      ◆       ◆      ◆Cancel ◆OK
```
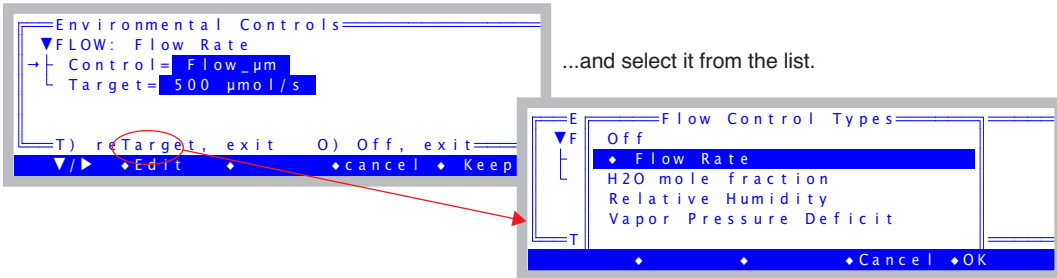
2. Move the cursor to the Target line, and press **Edit**.

```
┌──Environmental  Controls══════
│▼FLOW:  Flow  Rate
│├─ Control=  Flow_µm
→└ Target=  500 µmol/s

│══T) reTarget, exit   O) Off, exit══
│ ▼/▶   ◆Edit    ◆    ◆cancel ◆ Keep
```

3. Enter 700, press **Enter**.

```
┌──Environmental  Controls══════
│ ┌──Target  for  Flow_µm──
│ │700▪
│ └──500µmol/s──────────
│Note: Enter 'c' to use the current
│ value of Flow_µm (450)
│ or '#' for a variable target.
│DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```

4. Now press **Keep**.

```
┌──Environmental  Controls══════
│▼FLOW:  Flow  Rate
│├─ Control=  Flow_µm
→└ Target=  700 µmol/s

│══T) reTarget, exit   O) Off, exit══
│ ▼/▶   ◆Edit    ◆    ◆cancel ◆ Keep
```

Useful short cut: You can eliminate Step 4 by pressing **T** for Step 2.

Figure 3-26. Pressing **f2** *brings up the humidity control panel (part a). Press* **F** *to select fixed flow rate, then enter the target value (part b).*

4   **Note the reference and sample water vapor channels**
If it is not already there, put display line *a* on the top line. The two variables

a | CO2R_µml | CO2S_µml | H2OR_mml | H2OS_mml |
|---|---|---|---|
| 399.7 | 365.9 | 0.143 | 15.13 |

on the right (*H2OR_mml* and *H2OS_mml*) are reference and sample water mole fractions, in mmol mol^{-1}. The reference should be near zero, since we are forcing all the air through the desiccant. The sample in this illustration is

near 15 mmol mol^{-1}; what your value is depends on how wet the "leaf" is, what the temperature is, etc.

5   Note the flow rate and relative humidity
Put display line *b* on the second line. Line *b* contains the flow rate

b	ΔCO2_µml	ΔH2O_mml	Flow_µml	RH_S_%
	−33.8	14.91	700.8	52.01

(*Flow_µml*), which should match (usually within 1 μmol s^{-1}) the target value shown on the flow control function key label (**f2** level 2). Another item on line *b* we'll be watching is relative humidity in the sample cell, in percent (*RH_S_%*).

6   Change the desiccant to full bypass
Observe what happens when you change the desiccant from full scrub to full bypass. The reference water vapor concentration (*H2OR_mml*) will increase to the ambient value, since we are now not drying the incoming airstream at all. The sample water vapor concentration (*H2OS_mml*) will not increase as much, since the paper is not able to evaporate as much water into the more humid air.

7   Change to a low flow rate: 100 *µ*mol s^{-1}
Press **f2** then type **T 100**. (Notice the short cut: when you aren't changing control *modes*, the quickest way to change a target and exit is to type **T**.

8   Note the sample and reference humidities
The reference value has not changed, but the sample value increased. Why? Because the air is going through the chamber at a much slower rate, providing a longer exposure to the evaporating paper. In fact, you may start getting "High Humidity Alert" flashing on the center line of the display. If it does, ignore it. (Warning messages are discussed later, on page 3-35.)

9   Turn the desiccant to full scrub
Observe the reference and sample humidities falling, and note a) the sample value comes down slowly because of the slow flow rate[2], and b) the reference

---

[2.]If you have a CO_2 mixer, the reference value will decrease rapidly, because "excess" flow is routed to the reference line (Figure 1-2 on page 1-5).

goes back to near zero, but c) the sample doesn't return to the value you had in Step 4, because now the flow is slower.

Points to Remember

• At equilibrium, reference humidity is determined solely by the desiccant tube scrub setting; it doesn't depend on flow rate. It can also depend on the moisture content of the soda lime, which we had bypassed for this experiment.

• At equilibrium, sample humidity is determined by
   a) desiccant tube scrub setting,
   b) flow rate,
   c) evaporation from the leaf.

• Lowest humidity: high flow, full scrub. Highest humidity: low flow, full bypass. Between these limits, any chamber humidity can be achieved by various combinations of flow rate and scrub setting. (For a picture of this, see Figure 3-71 on page 3-71.)

One last comment: it can potentially take many minutes (or tens of minutes) to reach equilibrium after a substantial humidity change, since water adsorbs to all surfaces in the instrument. This is more obvious at low flow rates, less so at high flow rates.

Fixed Humidity Operation

One of the powerful features of the LI-6400 is its ability to operate in a fixed humidity mode. It does this by actively regulating the flow rate to maintain a target water mole fraction (or relative humidity, or vapor pressure deficit in the chamber). This mode is useful for maintaining humidity while measuring the leaf's response to something else.

Experiment #2   **Maintaining a constant humidity**
Continuing from Experiment #1 with wet filter paper clamped in the leaf chamber, we will now do some constant humidity operations, starting in fixed flow mode to see what range of humidities are achievable.

1   **Set flow to 400**
Press **2**, **f2**, then **T 400 enter**.

2   **Put desiccant mid-range, soda lime on bypass**
Set the knob midway between scrub and bypass. It's about one complete knob turn from either extreme, but you don't have to look; you can feel the midpoint. This is the region where the knob is the loosest.

Set the soda lime scrub tube knob for full bypass.

3   Switch to humidity control
Press **f2**, switch to mole fraction, and enter the current H2OS_mml value as the target.

1. Edit Control and pick H2O mole fraction.

```
                          ═E══════Flow Control Types══════
                          ▼F   Off
                          └    ◆ Flow Rate
                               H2O mole fraction
                               Relative Humidity
                               Vapor Pressure Deficit
══Environmental Controls══   ─T
▼FLOW: Flow Rate
├ Control= Flow_μm              ◆         ◆       ◆Cancel ◆OK
└ Target= 700 μmol/s
──T) reTarget, exit   O) Off, exit══
 ▼/▶  ◆Edit    ◆      ◆cancel ◆ Keep
```

2. Then press **T** then **c** then **Enter**.

```
                    ══Environmental Controls══
                    ─Target for H2OS_mml─
                    c
                     20.0mmol/mol
              Note: Enter 'c' to use the current
               value of H2OS_mml (18.3)
               or '#' for a variable target.
              DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```

20.0 is the default (or previous) mole fraction target.
18.3 is the actual current value of (in this case) H2OS_mml.

*Figure 3-27. When prompted for a target, the current value of the relevant target variable is shown. The shortcut for using that value is to enter a **c** instead of a number.*

4   Note the function key labels
The **f2** label should reflect the new type of control and the target.

* indicates humidity is not yet on target

```
    LeafFan  *H2OS→   Mixer    Temp    Lamp
  2   Fast  18.3 mm   OFF      OFF     OFF
```

*Figure 3-28. **f2** indicates constant mole fraction control, and the target value.*

5   Observe the flow rate
The flow rate (display line *b*) will vary a bit, finally settling down once the humidity is on target. Eventually, the asterisk on the **f2** label (Figure 3-28) will disappear, indicating that the water mole fraction is on target and stable.

6   **Raise the target value by 2**

Press **f2**, and enter a new target that's 2 mmol mol^{-1} *higher* than the present target (e.g. 20.3 mmol mol^{-1}). The flow rate should fall, and eventually settle on a new value that maintains this higher humidity.

7   **Lower the target by 4**

Now press **f2**, and enter a value that's 2 mmol mol^{-1} *lower* than the original target (e.g. 16.3 mmol mol^{-1}). The flow rate will eventually settle on a higher value that maintains this lower humidity.

8   **Enter a target that's too dry**

Now change the target to a much lower value, like 5 mmol mol^{-1} below the original target value (e.g. 12 mmol mol^{-1}). The flow rate will go as high as it can, but if it's not high enough, eventually the message

>> FLOW: Need ↑SCRUB or wetter target <<

will begin flashing on the center of the display. Try to remedy the situation by increasing the amount of scrubbing by the desiccant tube. If you provide enough scrubbing, eventually the target humidity will be achieved, and the flow rate will be less than the maximum. If the humidity still won't go low enough, the only recourse is to raise the target value. Hence the message.

9   **Enter a target that's too wet**

Press **f2** and enter a wet target, such 5 mmol mol^{-1} above the original value used back in Step 3 (e.g. 24). Soon the message

>> FLOW: Need ↓SCRUB or drier target <<

will be flashing on the center of the display. Notice that the flow rate is about 30 μmol s^{-1} if you have a CO_2 mixer, or near zero if you don't. Change the desiccant to full bypass and wait; the target humidity may or may not be achieved, and in fact you may even get "High Humidity Alert" messages while you are waiting.

10   **Return to the original target**
We end this experiment by returning the target to the original value, and the desiccant to the mid-range setting.

Points to Remember
• Sample humidity is a balance between what is coming from the leaf and the flow from the console: how dry (desiccant scrub setting, and soda lime scrub tube) and how fast (flow rate).

• If you ask for humidities outside what can be achieved, given a desiccant tube scrub setting and leaf transpiration rate, you'll get a warning message.

• These flow warnings can be remedied by adjusting the scrub knob, or changing the target value, or sometimes just by waiting.

About Warning Messages
While in New Measurements mode, there are a few situations that will cause OPEN to alert you to possible problems. Experiment #2 illustrated two. The complete list of possible causes is in **New Measurements Mode Warning Messages** on page 20-5. When such a situation occurs, a warning message is displayed on the middle label line on the display, as illustrated in Figure 3-29.

If you wish to disregard the displayed message and get it out of the way, press **ctrl Z** to turn it off (hold the **ctrl** key down and press **Z**). Note that this is a toggle: if you press **ctrl Z** again, the message will re-appear. Also, note that each time you re-enter New Measurements mode, OPEN resets this flag, so that warnings will again be displayed.

Blinking ———

→CO2R_µml	CO2S_µml	H2OR_mml	H2OS_mml
a 512.6	565.4	5.494	5.332
>> The Pump Is OFF... <<			
b 52.8	-0.161	-0.4	21.46
Photo	Cond	Ci	Trmmol
c 0.0377	5.81E-06	-9.69E+03	0.000116
LeafFan	Flow	Mixer	Temp	Lamp
2 Fast	OFF	OFF	OFF	-none-
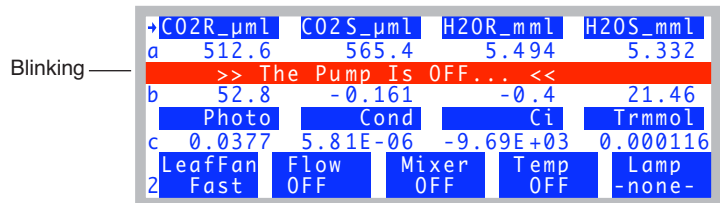
Figure 3-29. New Measurement Mode's warning messages appear on the 3rd line of the display.

Dynamic Response of Humidity Control

We will do one more humidity control experiment, but this time track what is happening using real time graphics. Part A of the experiment defines the graphics display, and part B does the work.

Experiment #3   **Watching the Dynamic Response of Humidity Control**

■   **Part A: Set up an RTG display for RH_S and Flow**
The editor we will be using is fully described in **Real Time Graphics** on page 6-14. But for now, just follow along and you'll get the job done.

1   **Select a Graphics Display**
Press **4** to view the Real Time Graphics control keys, then press **f3** (**View Graph**). Find an unused graphics display (F in our example).

2   **Launch the Plot Editor**
Press **2** to bring up the function key labels, then press **f4** (**EDIT**). The display will look like this:

```
✦CO2═════════════════Graph F═══════
a      Plot #1  < none >
  ΔCO   Plot #2  < none >
b      Plot #3  < none >

c  -0.353   0.000365   1.51E+03      0.0129

       ♦TurnON ♦Import ♦      ♦Done
```
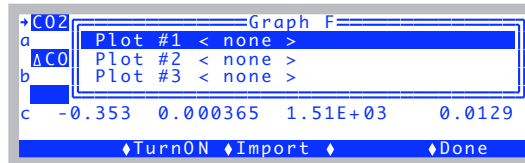
Figure 3-30. Editing Graphics Display F.

3   **Enable Plot #1**
Press **f2** (**TurnON**). When offered a choice between StripChart and XYChart, press **S** (or **f1**) for StripChart.

4   **Define the StripChart**
The Plot Editor for StripCharts will appear (Figure 3-31). The top line, "Y Axis =" shows the variable to be plotted, and presently there is none defined.

```
                 ═══════Graph F════════
  B:  │ Plo║══════Plot #1══════════║
  C:  │ Plo║ Y Axis = RH_S         ║
  D:  │ Plo║  A:Min=(0), Max=(1), Δ=1 ║
  E:  │    ║ X Axis = (Strip Chart Time) ║
  F:  │    ║ Keep 10.0 mins (Show 2.0) ║
  ════════════════════════════════
   Edit    ♦        ♦     ♦Cancel ♦Done
```
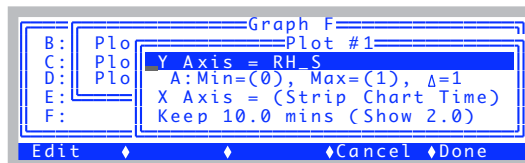
Figure 3-31. StripChart plot editor

5   Select RH_S

With the top line (Y Axis =) line highlighted, press **f1** (**Edit**). A list of variables will appear. Page down (press **pgdn**) a few times until you get to the line that says –15:RH_S and select it by pressing **f5** (**Select**) or **enter** (Figure 3-32).
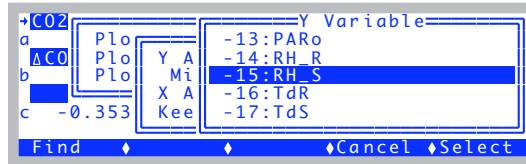
Figure 3-32. Selecting a variable to plot.

Then press **f5** (**Done**) to end the plot definition. The display should appear as in Figure 3-33.
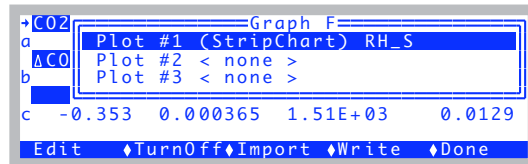
Figure 3-33. Done with first one

6   Add a Flow Plot

Press ↓ to highlight Plot#2, and press **f2** (**TurnOn**). Select StripChart. Then edit the Y Axis entry and pick flow rate, ("–7:Flow"). Press **Done** (**f5**).
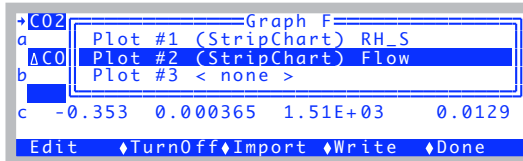
Figure 3-34. Two plots defined.

7   **Exit the Graph F editor**
Press **Done** (**f5**). You will go back to viewing Graph F, and it will start operating (Figure 3-35).
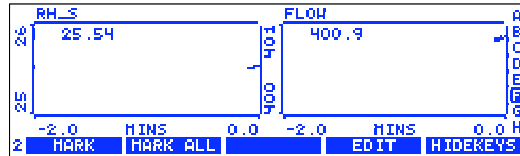
Figure 3-35. The finished product.

8   **Press escape or] to return to text mode**

■   **Part B: Do the Test**
We will now change the input humidity from dry to ambient and back, and watch how the flow rate adjusts.

1   **Fixed Flow at 400, Desiccant mid-range. Pick a target.**
Press **f2** (level 2), set for flow rate, then **t 400 enter**. Note the RH value (*RH_S_%*, line *b*). This value will be the target in the next step.

2   **Constant RH**
Switch to Relative Humidity mode, enter the current *RH_S_%* value as the target (**f2**, pick Relative Humidity, then **t c enter**).

3   **View the Graph**
Press **]**.

4   **Turn the desiccant to full bypass**
Wetter air will enter the chamber, so flow must increase to maintain the target humidity.

5 Wait about 15 or 20 seconds, turn to full scrub
The flow drops to a new value, but humidity stays the same. You should see something like Figure 3-36.
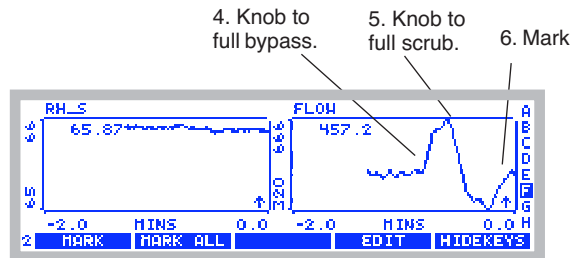
Figure 3-36. Flow increased when desiccant was put on full bypass, then dropped when desiccant was put on full scrub.

6 Mark the Plot
Press **f1** (**MARK**). This will mark on the plots the time we changed response time (next step).

7 Change to medium response.
Press **]** to stop viewing the graph. At level **2**, press **f2,** then change the response to *medium*. Then press **Keep**. Resume watching the graph by **]**.
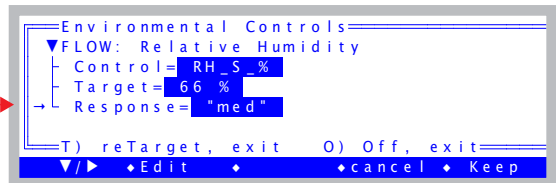
8 Full bypass, Full Scrub, Mark
Repeat the sequence: Full bypass, wait 10 secs. Full scrub, wait 10 secs, then back to middle, wait 10 secs, and mark (Figure 3-37).
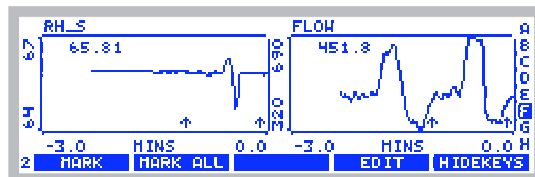
Figure 3-37. Scrub / bypass cycles at fast and medium response. The mark shows when the response was changed from fast to medium.

9   **Once more at slow response.**
Change to slow response (**]**, **f2**, change response to *slow*, **keep**, **]**). After you've scrubbed and bypassed, you should see something like Figure 3-38.
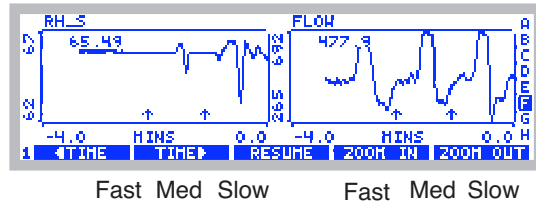
Figure 3-38. Results from the dynamic response test. Note that on fast response, the humidity is the most stable, and the flow rate the most unstable. On slow response, the flow rate is most stable, but the humidity is least stable, reacting slowly to the changing incoming humidity.

We will leave humidity control, but for more detail see **Humidity Control** on page 7-7.

Points to Remember
• There is a control trade-off: stable humidity and unstable flow vs. unstable humidity and stable flow. (For details and suggestions, see the discussion under **Humidity Time Response** on page 7-13.)

• Typically, you will be best served by operating with Response=fast for the "tightest" humidity control. Response=med will have reduced system noise, however.

CO_2 Control - Without a 6400-01 Mixer

In the absence of a 6400-01 CO_2 Mixer, the LI-6400's means of controlling CO_2 is limited to the soda lime adjustment knob. The following experiment illustrates how this works.

Experiment #4 **Adjusting the Soda Lime**
For this experiment, the chamber should be empty and closed.

1 **Fixed flow of 500 μmol s^{-1}**
Press **2**, **f2**, flow rate, **t 500 enter**.

2 **Soda lime full scrub, desiccant full bypass**
This will provide incoming air that is free of CO_2.

3 **Note the reference and sample CO_2**
They should both be near zero, and stable. Photosynthesis (*Photo*, line *c*) should be stable. If *Photo* is not zero, it is because the sample and reference IRGAs are not reading the same thing. Matching (page 4-33) will take care of that, but we will ignore it for now.

4 **Desiccant full scrub.**
Watch *CO2R_μml* or *CO2S_μml*, and note the burst. That's CO_2 that flushed out of the desiccant tube; the desiccant buffers CO_2 chemically as well as volumetrically.

5 **Soda lime full bypass, desiccant full bypass**
Now we will let ambient air into the chamber. Watch how *unstable* photosynthesis becomes.

Unlike Step 3 when we were scrubbing all the CO_2 from the air, the reference and sample CO_2 are now fluctuating as unmodified, ambient air enters the system. If you want to see *real* instability, breathe near the air inlet on the right side of the console, and watch what happens to photosynthesis.

> **Points to Remember**
> • The soda lime scrub setting controls reference CO_2, from ambient down to zero.
>
> • You will need a buffer volume to make stable measurements. See **Air Supply Considerations** on page 4-50
>
> • The desiccant buffers CO_2.

CO_2 Control - With a 6400-01 Mixer

Life is simpler with a CO_2 mixer, as the following will demonstrate.

Experiment #5 **Using the CO_2 Mixer**

Prepare the 6400-01 (see **6400-01 CO2 Injector Installation** on page 2-7), and install a 12 gram CO_2 cartridge.

1 **Set the flow control for a fixed flow at 500 μmol s^{-1}.**
Press **2**, **f2**, flow rate, **t 500 enter**.

2 **Turn on mixer and set a 400 μmol mol^{-1} reference target.**
Press **f3** and make the screen look like this,

```
════Environmental  Controls════
▼CO2:  Reference  CO2
→├   Control=  CO2R_µml
 └   Target=  400  µmol/mol


══T)  reTarget,  exit    O)  Off,  exit══
  ▼/►   ◆Edit     ◆       ◆cancel ◆ Keep
```
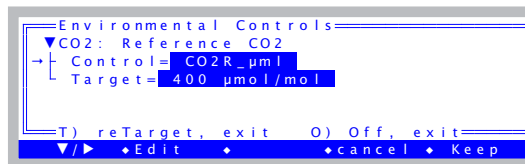
then press **f5** (**Keep**). Set the soda lime to full scrub.

3 **Put status line J on the middle line**
The mixer status (Mixr) can show OK, Low, or High.

```
→  CO2   H2O  Pump  Flow  Mixr   Fan
 j   OK    OK    OK    OK   LOW  Fast
```

If you've just installed the CO_2 cartridge, Mixr will probably show LOW (not enough CO_2 pressure). Eventually (after 2 or 3 minutes), it should show OK, and the *CO2R_µml* value (line *a*) should be close to 400, the target.

4 **Once *CO2R_µml* is stable, change target to 200 μmol mol^{-1}**
Press **f3**, **t**, **200**, **enter**. Notice how much faster *CO2R_µml* drops than rises. It may overshoot or undershoot the target value, but will correct itself eventually. Calibrating the mixer (we skipped it, but it's described on page 18-25) can improve this performance.

5 **Once *CO2R_µml* is stable, change to 20 μmol mol^{-1}**
Press **f3**, **t**, **20**, **enter**. *CO2R_µml* won't make it to 20, but will probably stabilize between 30 and 50 μmol mol^{-1}. Notice the Mixer status (*Mixr* on line *j*) will be High.

6   Change to 2000 μmol mol^{-1}
Press **f3**, **t**, **2000**, **enter**. It will take a few minutes to reach this, and *Mixr* will be Low for much of this time. When *CO2R_μml* finally does reach 2000 μmol mol^{-1}, it should be fairly stable.

7   Change to 400 μmol mol^{-1}
Press **f3**, **t**, **400**, **enter**. *CO2R_μml* should drop to 400 μmol mol^{-1} much faster than it rose to 2000 μmol mol^{-1}.

Points to Remember
- Soda lime must remain on full scrub when using the mixer

- The lowest stable value is typically between 30 and 50 μmol mol^{-1}. See Application Note 7 for a modification to control to lower concentrations.

- Mixer adjusts faster when lowering the concentration, than raising it.

Experiment #6   Dynamic Response of the CO$_2$ Mixer
We will need StripCharts for CO2R and CO2S (where we are headed is Figure 3-39 on page 3-44). They are likely already defined, on Graph B. If not, set them up yourself, following the example on page 3-36. (Hint: *CO2R* and *CO2S* have ID numbers -1 and -2 in the list of variables that appears when you edit the Y axis.)

1   Close the empty chamber
It's another no-leaf experiment.

2   Fixed flow at 500 μmol s^{-1}
Press **2**, **f2**, flow rate, **t**, **500**, **enter**.

3   Set the mixer for a 400 μmol mol^{-1} reference target
Press **f3**, reference, **t**, **400**, **enter**. Watch the graph (**]**).

4   After 1 minute, change reference target to 900 μmol mol^{-1}.
When the *CO2S* curve flattens out, press **]**, **f3**, **t**, **900**, **enter**. Resume watching the graph (**]**).

5   After 1 minute, change the reference target to 100 μmol mol^{-1}.
When the *CO2S* curve flattens out, press **]**, **f3**, **t**, **100**, **enter**, followed by **]** to continue viewing the graph.

---

6   **Redo Steps 4 through 5 with Flow = 100.**
Reduce the flow rate, and repeat the cycle. Note the slower response of the sample cell, and the faster response of the reference cell.

Your results should be similar to Figure 3-39.
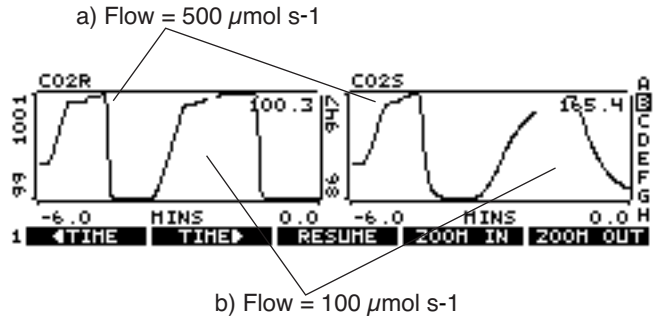
Figure 3-39. Controlling on reference CO_2 concentration, with a closed, empty chamber. Note the slower response of the sample cell (due to its larger volume) is aggravated at lower flow rates. Note also the faster response when lowering CO_2 than when raising it.

Point to Remember
• Controlling on reference concentration is faster than controlling on sample concentration, for 3 reasons:
    1. The larger volume of the sample cell / leaf chamber.
    2. Flow changes when controlling constant humidity.
    3. Possible photosynthetic rate changes.

Temperature Control

Temperature control has two target options: a constant block temperature (the block is the metal block that encompasses the sample and reference cells of the IRGA), or constant leaf temperature. **Temperature Control** on page 7-18 has details, but the next two experiments illustrate the differences.

We will use StripCharts for both of these experiments, monitoring block temperature, air temperature in the sample cell, and leaf temperature. Graph D should already be configured for this, but if not, the ID numbers for our three temperatures are -8, -9, and -10.

Note: you can extend the temperature range of the coolers by using the expanded temperature range control kit 6400-88.

Experiment #7   **Controlling on Block Temperature**

This is the more direct and stable of the temperature control options.

1   **View the temperature line.**

Display line *h* of the default display map has block, air, and leaf temperatures.

2   **Set block target to 3°C cooler than current value**

Press **f4** (level 2)**,** block temp, **t** <*value* - 3> **enter** (where *value* is the block temperature from Step 1).

1. Set for Block
Temperature.

```
═══Environmental  Controls═══
▼TEMP:  Block  Temperature
→├  Control= Tblock
 └  Target= 20.0 C

══T)  reTarget,  exit    O)  Off,  exit══
 ▼/►    ♦Edit    ♦        ♦cancel ♦ Keep
```
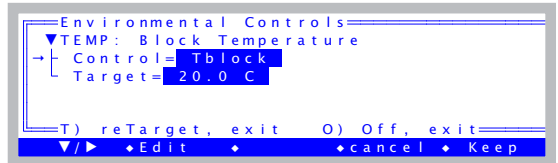
2. Enter the target:
Tblock - 3.

```
═══Environmental  Controls═══
──Target  for  Tblock──
 23
 ──20.0C──
Note:  Enter  'c'  to  use  the  current
value  of  Tblock  (26.0)
or  '#'  for  a  variable  target.
DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```
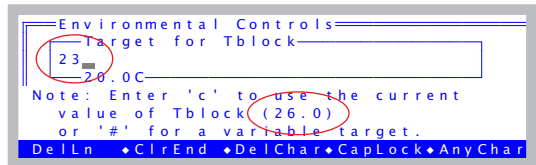
The external fans on either side of the chamber should start to run. Watch the block temperature slowly drop to this new target value.

3   **Note the temperature gradient**

You should see that *Tblock < Tair < Tleaf*.

4   Now make the target 3 °C warmer than ambient
Press **f4**, **t**, <*value* + 3> **enter**. The block temperature will rise a bit faster to this new target. Heating is always more efficient than cooling.

5   Note the temperature gradient
You should see that *Tblock* > *Tair* > *Tleaf*.

6   Return to ambient
Set the block temperature back to the starting point.

The graph for this experiment might look like Figure 3-40.
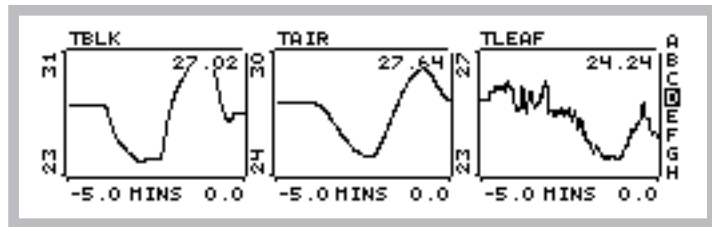
Figure 3-40. Results of Experiment 7.

Points to Remember

• Controlling on block temperature is slow but steady.

• Limit of control is generally within 7 °C of ambient.

• The further *Tblock* is from ambient, the larger the temperature gradient through the leaf chamber and IRGA.

Experiment #8   **Controlling on Leaf Temperature**
For this experiment you will need slightly damp filter (or towel) paper to act like a leaf.

1   **Observe Tleaf**
Tleaf°C is on display line *h*.

2   **Control leaf temperature to 1 °C cooler than its present value**
Press **f4** (level 2), leaf temp, **t**, <*value* - 1> **enter**.

1. Set for Leaf Temperature.

```
════Environmental Controls═══
▼TEMP: Leaf Temperature
├ Control= Tleaf
→└ Target= 20.0 C

═T) reTarget, exit   O) Off, exit═
 ▼/▶   ◆Edit   ◆        ◆cancel ◆ Keep
```

2. Enter the target: Tleaf - 1.

```
════Environmental Controls═══
┌─Target for Tleaf─────
│ 25.5_
└──20.0C───
Note: Enter 'c' to use the current
  value of Tleaf (26.5)
  or '#' for a variable target.
DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```
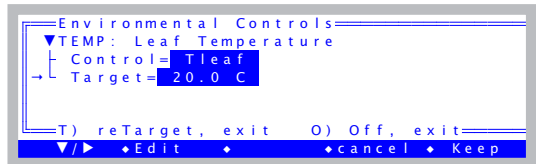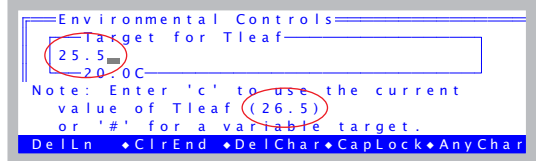
3   **Watch what the block temperature does**
Press **]** to view the strip charts (Figure 3-41a). The block temperature will drop in discrete increments as the control algorithm works to get the leaf temperature down to where it belongs.

4   **Now add 1 degree to the target leaf temperature**
Press **]** to get back to text mode, then **f4**, **t**, <*value* + 1> **enter**. Then **]** to view
the strip charts (Figure 3-41b).
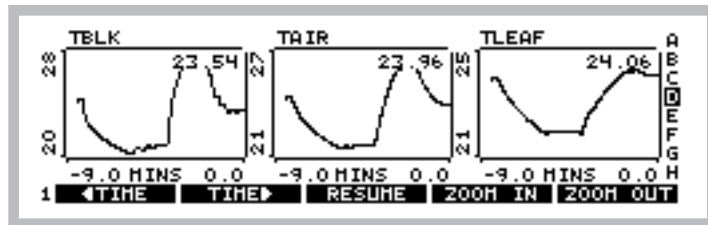
Figure 3-41. Results of Experiment 8.

Points to Remember
• Block temperature will warm or cool as needed in its efforts to control
leaf temperature.

• Leaf temperature control is not as fast or stable as block temperature
control, since the mechanism is to control the temperature of the air that
is blowing by the leaf, whereas the real factors driving leaf temperature
include radiation balance and physiology.

Lamp Control

If the light source is attached (the procedure is described on page 2-15), but OPEN is not presently configured to use it,[3] then do the following:

■ **To configure OPEN for a controllable light source:**

1   Access the system configuration
This is done by `Config Menu|View/edit`. Press **escape** to leave New Measurements mode. In OPEN's main screen, press **f2** (**Config Menu**), then highlight the entry "`View/edit...`", and press **enter**.

```
┌═Config Menu════════════════════┐
│   New...                        │
│   Open...                       │
│   Save as...                    │
█  *View/edit...                  │
│                                 │
│ ▼/▶    ♦        ♦      ♦ EXIT ♦ SELECT│
└─────────────────────────────────┘
```

2   Attach a Light Source
Use whatever you have available: 6400-02B LED, 6400-18 RGB, or 6400-40 LCF.

3   Navigate to the open - light - source node
(Remember **f1** will open closed nodes.)

```
┌═FactoryDefault_6.2.xml══════════┐
└▼open                            
 ┌▼light                          
 │ ┌ source= "Sun+Sky"           
█ │▶par_in= Measured  in  Chamber 
 │ └▶par_out= Measured            
 └▶comps                          
 ▼/▶   ♦Edit    ♦      ♦      ♦Done│
└─────────────────────────────────┘
```
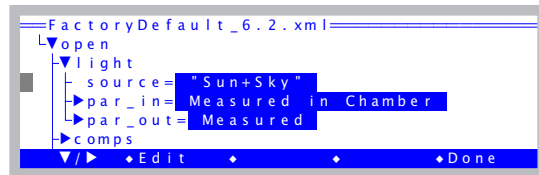
[3.] The control key label, **f5** level 2, will show "-none-" if OPEN is not configured for a light source.

4   Press f2 (Edit) and pick a source
A menu of possible light sources appears. Choose the appropriate 6400-02, 6400-18, or 6400-40 entry, and press **enter**.

```
═Fa┌═════Select Light Source═════════
 └▼*║"6400-40 Fluorometer" 1.0 0
  ├▼║"6400-40 (Light source only│
  │├║"6400-02 LED Source" 1.0 0.│
  └├║"6400-18 RGB Source" 1.0 0.│
   └║"Sun+Sky" 1.0  0.19
  ├▶║"Quartz Halogen" 1.11 0.26
   Print ♦ Find  ♦ ReFind♦ CANCEL♦ SELECT
```
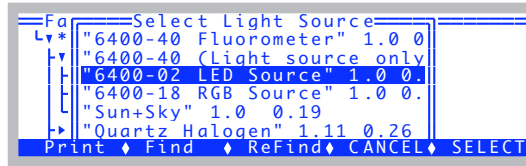
5   Return to New Measurements mode
Press **f5** (**Done**), to return to the Config Menu, then **escape** to return to OPEN's main screen, then **f4** (**New Msmnts**).

The lamp control is the most straight-forward of all the control algorithms. The feedback is immediate, since there is a sensor located in the light source measuring its output, and there is no other mechanism (such as flow rate) that will interfere with the light value[4].

Experiment #9   Simple Lamp Control

1   Close the chamber, no leaf

2   Monitor in-chamber PAR on the display
Press **G**. The in-chamber PAR value is *ParIn_μm* on display line *g*.

3   Set the lamp for PAR, 1000 μmol m^{-2} s^{-1}.
Press **f5** (level 2), PAR, **t**, **1000 enter**.

```
═Environmental Controls═══════
▼Light: PAR
→├ Control= ParIn_µml
 └ Target= 1000 µmol/m2/s


═T) reTarget, exit    O) Off, exit═
 ▼/▶  ♦Edit   ♦      ♦cancel ♦ Keep
```
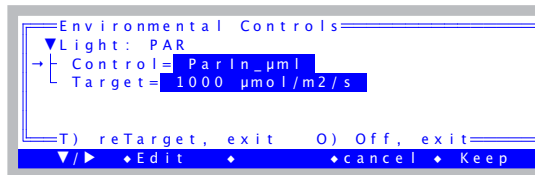
4   Watch ParIn_μml
The *ParIn_μml* value will go to a value fairly close to the target, and then a few seconds later shift to the exact target value.

---

[4]·Leaf reflectance plays a big role, but it is stable while the leaf is in the chamber, so we don't worry about it.

5 Open the chamber
When you open the chamber the *ParIn_µml* value will drop about 10% due to the sudden decrease in reflectance. Note: this is NOT true for the 6400-18 RGB. It has some on-board circuitry that keeps it locked on to the target when you change the amount of light reflecting back into the lamp (**Internal Feedback** on page 8-18).

6 Watch ParIn adjust
After several seconds, the *ParIn_µml* value will increase to the target, as the light source brightens to make up for the decreased reflectance.

7 Turn the lamp off.

Points to Remember
• As with the other controls, the light source is active, adjusting for changing conditions in the leaf chamber or light source itself.

• The light source control uses a "first guess" when given a target. If the guess is not on target, it adjusts itself to the correct value. There is a calibration routine (in the Calib Menu) that can be run that generates data for these first guesses. This is described on page 18-30 (for the light source) or page 27-75 for the LCF.

Control Summary

Even though the four control areas that we have just explored have very differing hardware (a pump and/or proportioning valve for humidity control, LEDs for light control, etc.), they have common interfaces and logic. In fact, this control software offers some powerful features that our tour did not touch upon. Chapter 7 provides a more thorough discussion of these controls, their options and limitations. At some point in your experience with the LI-6400, you should take time to read this material, and acquaint yourself more fully with these tools.

Tour #4: Matching

We now take a look at the single most important issue for making a worthwhile measurement with the LI-6400: it's something we call *matching*.

<u>What is Matching?</u> Photosynthesis is computed from the difference in CO_2 between the reference and sample cells of the IRGA. That difference tells us how much CO_2 the leaf has taken out of the air. Similarly, the difference in H_2O tells us how much water vapor the leaf has contributed, and from that we get transpiration and conductance. If the sample and reference cells are not matched, that is, if they do not report the same concentrations when seeing the same air stream (an empty chamber, for example), then the computations of photosynthesis and transpiration will be wrong.

A simple method of seeing how well the IRGAs are matched is to close the leaf chamber empty (no leaf), and compare sample and reference readings. Unfortunately, it is not always convenient to empty out the chamber - suppose you wish to check the match at each step in a light curve - so the LI-6400 has a method of routing the air that has exited the leaf chamber into the reference cell, allowing the reference and sample cells to see the same air without affecting what is going on in the chamber. We call this *match mode*, and there is an illustration of the match valve and how it changes the flow schematic in the next chapter (Figure 4-2 on page 4-34).

The Match Valve

First, we will get you acquainted with where the match valve is and how it moves.

Experiment #1 **Making the Match Valve Move**
Start from OPEN's main screen. If you are in New Measurements mode, press **escape**.

1 **Find the Diagnostics and Test menu**
Press **f1** (**Home Menu**), navigate to and open the *Diagnostics & Tests* node.

```
┌═Home Menu════════════════┐
│    About this unit...     │
│    Contrast & Backlight...│
│▼Diagnostics & Tests       │
│    Auxiliary DAC Test...   │
│    CO2 Mixer Test...       │
│    DAC Status...           │
│▼/▶  ◆      ◆      ◆ EXIT ◆ SELECT│
└══════════════════════════┘
```
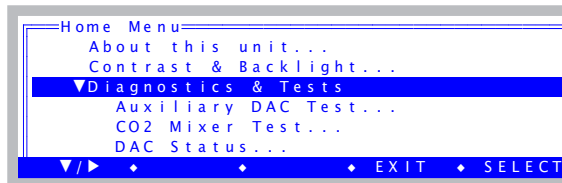
2   **Launch Match Valve Tester**
Scroll down to Match Valve Tester, and press **enter** (or **f5 Select**).

```
═══Home Menu════════════════════
       CO2 Mixer Test...
       DAC Status...
       Digital Status...
       Log Button Tester...
       Match Valve Tester...
       Pressure Sensor...
   ▼/▶     ◆         ◆         ◆  EXIT  ◆  SELECT
```

```
       Match Valve Test Program

   When in match mode, the right end
   of the paddle should be down.

   Match mode is OFF

    TOGGLE                          Quit
```
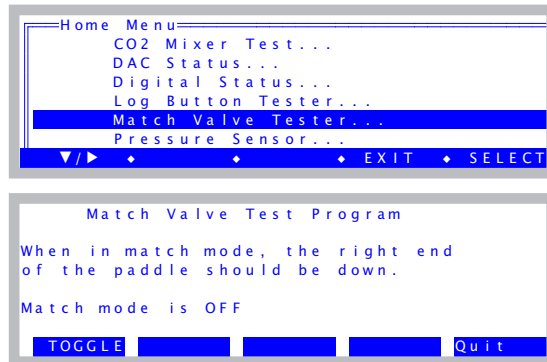
Figure 3-42. The match valve test program.

3   **Press f1 to move the valve**
Turn the sensor head over so you can see the valve. It should change positions
each time you press **f1** (**Toggle**).
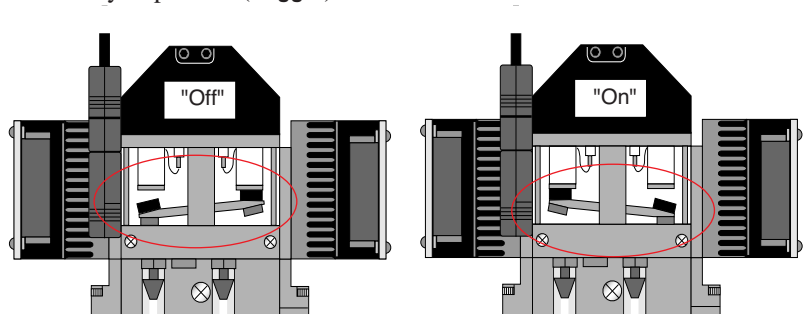
Figure 3-43. The two positions of a properly functioning match valve.

If your match valve is not behaving properly, see **Match Valve Problems** on
page 20-22.

4   **That's all**
Press **escape** to get back to OPEN's main screen. The match valve will return
to the OFF position automatically when you exit the test program.

Match Mode

The match valve is just a valve, whereas match *mode* refers to the software and display screen that do something useful with that valve.

Experiment #2 **Match Mode in New Measurements**

Match mode can be entered manually during New Measurements mode, and allows you to check how well the IRGAs are matched, and to match them if you choose. This occurs without changing the conditions a leaf sees in the chamber.

1 **Start with stable conditions in New Measurements mode**

Set a flow rate of 500 μmol s^{-1}. If you have a functioning CO_2 mixer, set it up for controlling reference CO_2 to 400 μmol mol^{-1}. (If you do not, just set CO_2 scrub to full). Set desiccant midway between scrub and bypass. Close the chamber, and wait for *CO2S* and *CO2R* to stabilize.

2 **Enter match mode**

Press **1 f1** (**Match**). You will see a transition display (Figure 3-44a) for a few seconds, followed by the match mode display (Figure 3-44b).
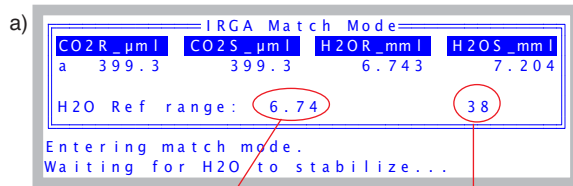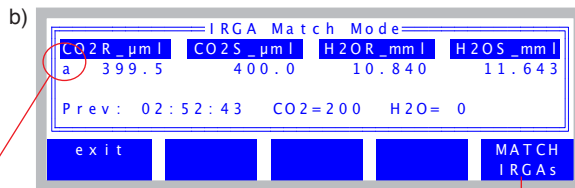
Figure 3-44. Entering match mode. (a) The transition display. (b) The match mode display.

3   **Match the IRGAs**

If there are significant differences between *CO2R* and *CO2S*, or between *H2OR* and *H2OS*, you can remove them by pressing **f5** (**MATCH IRGAs**).
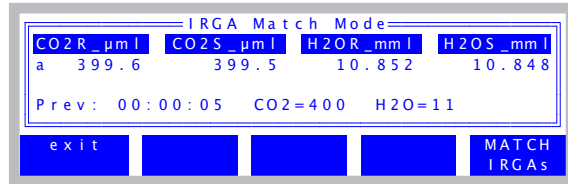
Figure 3-45. Five seconds after pressing **f5** *to match the IRGAs.*

Once you press **MATCH IRGAs**, you are still in match mode, and you can verify that the IRGAs do now more or less agree. If you matched at an inopportune time and the agreement is not so great, you can press **MATCH IRGAs** again, as often as you wish.

What actually happens when you press **MATCH IRGA**s is just mathematical: two correction factors, one for CO_2 and one for H_2O, are computed for the sample cells. So, *CO2S* and *H2OS* will be the values that you see change.

4   **Exit, and compare again.**

Press **f1** (**exit**) to leave match mode, and then compare sample and reference back in New Measurements mode.
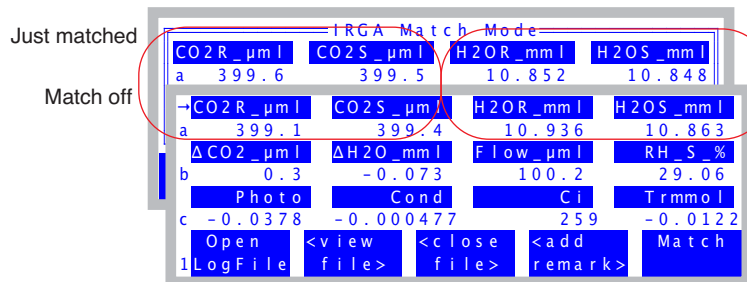
Figure 3-46. Comparison of readings just after matching with what they are back in New Measurements mode.

Notice in Figure 3-46 that the sample concentrations barely changed when match mode was exited, and that's a good thing, since the match valve position should not change sample concentrations. The reference concentrations, however, did change slightly: *CO2R* dropped 0.5 μmol mol^{-1}, and *H2OR* increased by 0.12 mmol mol^{-1}. If there were a leaf in the chamber, the reference

changes would have been much bigger, but the chamber was empty. So why were there any changes in the reference concentrations?

Well, what we have just done is test the chamber for leaks, diffusion, cleanliness, and anything else that might alter the H_2O and CO_2 concentrations of air passing through it. Certainly diffusion is the likely cause of the CO_2 change, since the air surrounding the chamber in this particular case was (I am opening the chamber now to check...) 550 μmol mol^{-1}, so there would be a small influx of CO_2 through the gaskets. (For more on this, see **Diffusion Leaks** on page 4-44). The reason for a water change is less clear: it's not likely a diffusion issue, since the water gradient through the gasket was minimal. Most likely the slight H_2O losses through the chamber were due to ad- or absorption by chamber surfaces - especially any dirt on those surfaces.

Points to Remember

• A measurement is only as good as the match.

• Sample concentrations should not change when entering or leaving match mode. If you match, however, that will change them.

• Comparing reference concentrations in and out of match mode when the chamber is empty is a sensitive test for chamber leaks and cleanliness.

Experiment #3

Matching at different concentrations
If you match at one concentration, do you have to match again if the concentration changes? Let's find out.

1   **Match at a low concentration**

Set the mixer for a 200 μmol mol^{-1} reference concentration, and let it stabilize. Then enter match mode and match the IRGAs. (If you just did the previous experiment, where you matched at 400 μmol mol^{-1}, note the difference, if any, changing from 400 to 200 made.)

2   **Change to a high concentration**

Now set the mixer for 1500 μmol mol^{-1}. When the mixer finally stabilizes, compare *CO2S* and *CO2R*. *CO2S* is likely to be slightly lower than *CO2R* due to diffusion losses out of the chamber gasket (assuming the ambient concentration is lower than 1500 μmol mol^{-1}). Next, we'll remove the diffusion difference by going in to match mode.

3   **Enter match mode**

We will not match, but just want to compare *CO2S* and *CO2R* when they are seeing the same airstream.

How close are they? If the difference is a few tenths of a μmol mol^{-1}, then you can feel proud or lucky, because ideally, that's what it should be. However, if it is more (or a *lot* more) than that, then it is an indication that the IRGAs are not very well zeroed and/or spanned, and you might want to spend some time going over **User Calibration (Zero and Span)** on page 18-10.

4   **Go back to normal**

Exit match mode, and set the mixer for 400 μmol mol^{-1}.

Whether or not your system happens to be well calibrated at the moment, however, does not mean it will always be that way, so changing concentrations is one thing that should always trigger a little voice in your head that says "check the match". And that brings up the question: how do you know what the conditions were the last time you matched?

5   **View display line 'm'**

The default version 6.2 displays have previous match information on display line *m* (Figure 3-47).
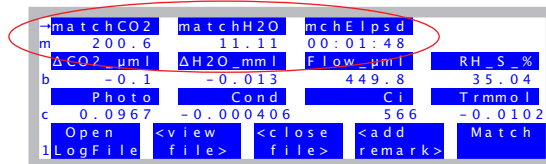
Figure 3-47. Previous match information: what concentrations, and how long ago (hh:mm:ss).

If your display configuration does not have these values on line *m*, then you can add them easily enough: See **Display Editor** on page 6-6; the items you want to add have system IDs -112, -113, and -114.

Experiment #4   **"Matching is so easy. What could *possibly* go wrong..."**

1   **Low and slow**

Set the flow rate to 30 μmol s^{-1}. You'd normally not do this, but we are forcing an issue.

2   Enter match

Press **f5** (level 1). You should get this message (Figure 3-48):

```
═══════════WARNING═══════════
Flow is low (   31 µmol)
Still want to match?_

═══════════════════(Y/N)═══
```
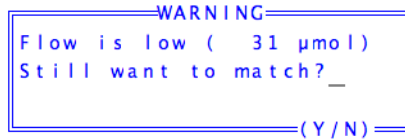
Figure 3-48. Warning displayed if entering match mode with a very low flow rate.

(You can press **n**, then set the flow back to 500 μmol s^{-1}).

Why is flow important? The lower the flow rate to the sample cell, the longer it will take to flush out the reference cell when the match valve turns on. That is why there is a 45 second potential wait during the transition. At the limit, of course, is no flow, and matching cannot work without flow through the sample cell. So we warn about flows that are < 50 μmol s^{-1} (with mixer) or < 100 μmol s^{-1} (no mixer).

3   Make a bad starting point

To illustrate the next point, we need to create a situation[5] with very poorly matched IRGAs, so we will do it with a little behind the scenes magic: 1) Press **escape** to OPEN's main screen. 2) Press **k**, and a little ok: prompt will appear. 3) Type this: **20 &co2_2_offset =** (there is a space after the 20, and one before the =).

```
ok:20 &co2_2_offset =__
 DelLn   ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar
```
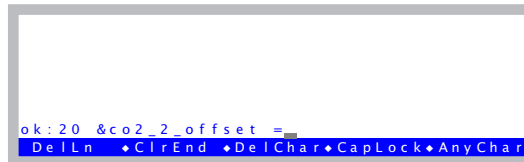
Figure 3-49. Forcing a 20 ppm mismatch in the IRGAs.
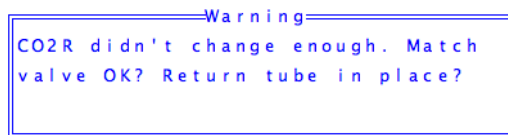
Press **enter**. If nothing happens, that's good. (If there is an error message, type it again, but correctly). Then press **escape** to get back to the main screen, then return to New Measurements. Set the mixer to control on *CO2R* to 400 μmol mol^{-1}. You should now have a very poorly matched IRGA, with *CO2S*

[5.]More typically, novice users create this situation for themselves by matching at the wrong time (chamber open, mixer still adjusting to new target, etc.)

about a 20 μmol mol^{-1} greater than *CO2R*.

4   Fix it by matching
Press **1 f5** to enter match mode. After the transition period, you should see this warning message (Figure 3-50):

```
══════════Warning══════════
CO2R  didn't  change  enough.  Match
valve  OK?  Return  tube  in  place?
```

Figure 3-50. Warning when the expected change in CO2R *doesn't happen when entering match mode.*
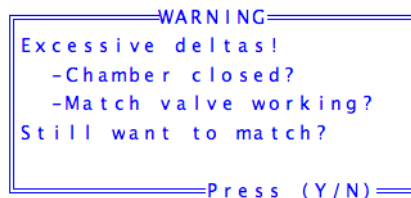
What is this all about? When match mode is entered, the system has some expectations for what should happen: mainly, *CO2R* should move toward *CO2S*. If someone had forgotten to install the return tube that connects the chamber to the match valve, or if the match valve fails to move, that would prevent *CO2R* from changing, and that is why the warning message provides the hints that you see.

However, we got that message with an empty chamber, and poorly matched IRGAs. The lesson is that this message does not necessarily mean a hardware issue. Any time you enter match mode with a difference between *CO2R* and *CO2S* of > 10 μmol mol^{-1}, *and* the *CO2R* value changes less than 1.5 μmol mol^{-1} during the transition, the "CO2 didn't change enough" message will appear.

5   Proceed with the match
Press **enter** to clear the "CO2R didn't change" warning. Then press **f5** to match the IRGAs and fix our mismatch problem...

...and you will get another warning (Figure 3-51):

```
══════════WARNING══════════
Excessive  deltas!
   -Chamber  closed?
   -Match  valve  working?
Still  want  to  match?
══════════Press (Y/N)══════════
```

Figure 3-51. Message to caution against matching at an inappropriate time.

When you attempt to match and the difference in CO_2 is $> 10\ \mu$mol mol^{-1} or the difference in H_2O is > 1 mmol mol^{-1}, then this message appears. This message is designed to prevent matching when you should not, since that large of a difference could be due to the mixer valve not functioning or other problems.

Press **y** to complete the match, and then exit match mode.

6   Change the mixer target
Drop the mixer target from its current 400, down to 200 μmol mol^{-1}.

7   Enter match mode now. (Hurry - Don't wait!)
Once through the transition period, you should get the leak warning (Figure 3-52):
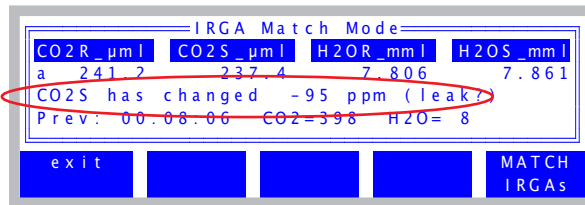
Figure 3-52. The leak warning appears if CO2S, *which should be stable, is not.*

CO2S should remain constant while in match mode, unless there is a leak. Now, this was clearly not caused by a leak, but the system assumes that you are acting rationally, and entering match mode right after a mixer change is not a rational act. You can do it, and it does not hurt anything (as long as you wait for things to stabilize before actually matching), but you will be warned.

This warning can have causes besides the obvious (leaks, opening the chamber, etc.): it could be the leaf being measured. If the light level changes while in match mode, photosynthesis will change, and *CO2S* will change.

The threshold for this message to appear is 3 μmol mol^{-1}. Some of the thresholds and actions involved in match mode are configurable. This is described in **Matching Variations** on page 16-19.

Points to Remember
• Warning messages have multiple potential causes. It's worth figuring out why a warning appeared so you can decide if it can be safely ignored or not.

Tour #5: Logging Data

Recording data in New Measurements mode is discussed in detail in Chapter 9, but here we show the basics. Logging control functions are found on function key levels 1 and 5 (Figure 3-53).

Pick a destination for logged data

Inactive when not logging

Automatic logging
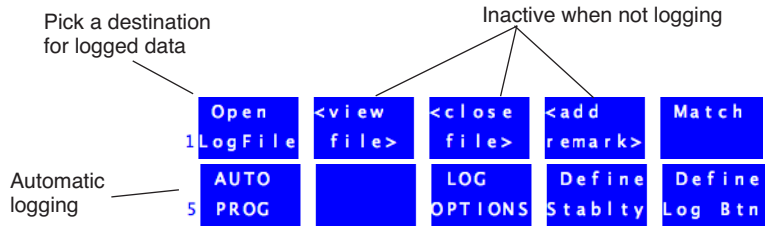
Figure 3-53. Logging begins with choosing a destination for the logged data.

Logging Data Manually

Below is a step-by-step example in which we will open a file, and store some data.

Experiment #1   **Log a data set manually**
Start out in New Measurements mode.

1   **Press 1 (if necessary) to bring up the logging control keys**
Shown in Figure 3-53.

2   **Press f1 (Open LogFile)**
This is how to select where data are to be recorded. Normally this is a file, but the comm port is also a possible destination.

3   **The Standard File Dialog**
You are shown the Standard File Dialog. This screen will appear anytime you need to enter a file name. This dialog is fully described on page 5-9, but we will cover some important features of it here.

Current directory

Files in the current directory
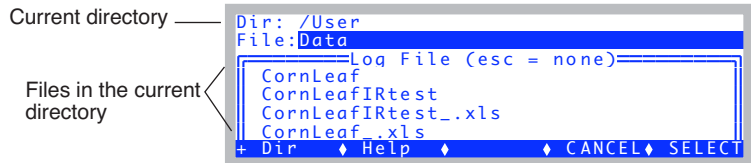
The top line of the dialog shows the directory in the file system where you will be writing your log file. Normally, this should be /User. Below that is a

list of the files in that directory. If you wish to change directories, press **f1** (**Dir**).

4   **Exploring the file list**
You can scroll through the file list, and even sort it, by pressing the up or down arrow key (↑ or ↓). The highlighted bar will jump from the file entry line to the list below it (Figure 3-54), and you can then scroll up and down through the list, using ↑, ↓, **home**, **end**, **pgup**, and **pgdn**.

```
Dir: /User
File:Data
═══════════Log File (esc = none)═══════════
 CornLeaf
 CornLeafIRtest
 CornLeafIRtest_.xls
 CornLeaf_.xls
+ Dir    ♦ Help  ♦          ♦ CANCEL♦ SELECT
```
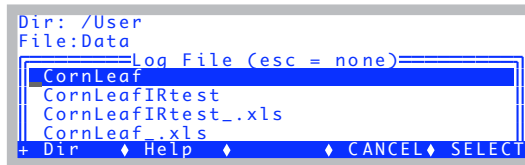
Figure 3-54. Highlighted bar dropped down to the file list.

While you are down in the file list, there are some things that you can do (Figure 3-55). Press **labels**, and you'll see the second level of fct keys:

Press **f1** (or just **M**) to toggle the display through the following:

Name only

```
Dir: /User
File:Data
═══════════Log File (esc = none)═══════════
 CornLeaf
 CornLeafIRtest
 CornLeafIRtest_.xls
 CornLeaf_.xls
+ More  ♦ Sort  ♦          ♦          ♦
```
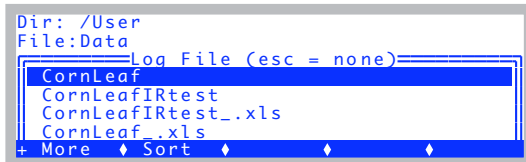
Date and time of last modification, and Name

```
Dir: /User
File:Data
═══════════Log File (esc = none)═══════════
 2008-05-20 09:21:23 CornLeaf
 2008-05-20 13:36:28 CornLeafIRtest
 2008-05-20 13:36:28 CornLeafIRtest_.x
 2008-05-20 09:21:23 CornLeaf_.xls
+ More  ♦ Sort  ♦          ♦          ♦
```

Size and Name

```
Dir: /User
File:Data
═══════════Log File (esc = none)═══════════
   74069 CornLeaf
   67399 CornLeafIRtest
  802580 CornLeafIRtest_.xls
  878162 CornLeaf_.xls
+ More  ♦ Sort  ♦          ♦          ♦
```
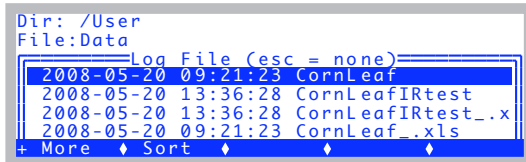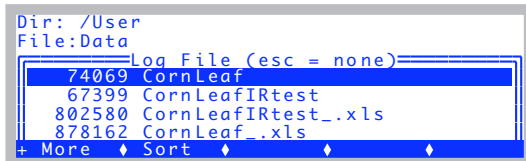
Figure 3-55. Manipulating the file list in the Standard File Dialog.

Another thing you can do is sort the list. Press **f2** (or just **S**). You can sort by Name, Date, or Size, and in Ascending or Descending order (Figure 3-56).

```
Dir: /User
Fi┌──────────────────────────────────┐
  │Sort by: Name, Date, or Size ?▁   │
  │                        (N/D/S)   │
  │CornL┌────────────────────────────┴──────┐
  │CornL│Dir: /User                         │
         │File:                             │
         │┌─────────────────────────────────┐│
         ││Ascending or Descending?▁         ││
    Cor  │                          (A/D)   ││
    Cor  └─────────────────────────────────┘│
    CornLeafIRtest_.xls                      │
    CornLeaf_.xls                            │
```
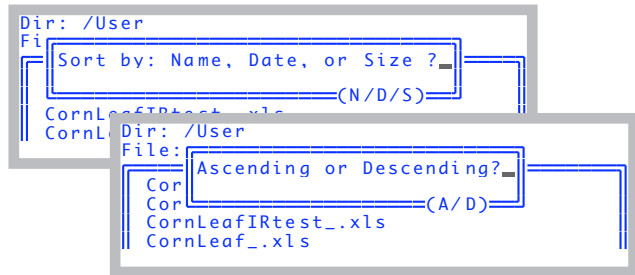
Figure 3-56. Sorting options.

Note that after the sort, the highlighted file remains the highlighted file; you'll have to scroll up or down to see where you are in the list.

Example: To find the most recently created or modified file in the list, press the following key sequence: **S D D home**. That is, Sort on Date in Descending order, then jump to the top of the list (Figure 3-57).

```
Dir: /User
File:Data
════════Log File (esc = none)════════
 2008-09-04 17:32:23 Data
 2008-09-04 17:32:23 Data_.xls
 2008-08-27 17:02:28 Junk
 2008-08-27 17:02:28 Junk_.xls
+ More   ♦ Sort   ♦          ♦          ♦
```
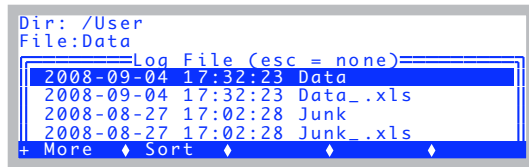
Figure 3-57. File list arranged in descending order by date.

You "exit" the bottom box by pressing **enter** or **escape**: If you wish to pick one of those existing files as the destination, highlight it, and press **enter**. Otherwise, just press **escape** and the file name entry line will remain unchanged (Figure 3-58).
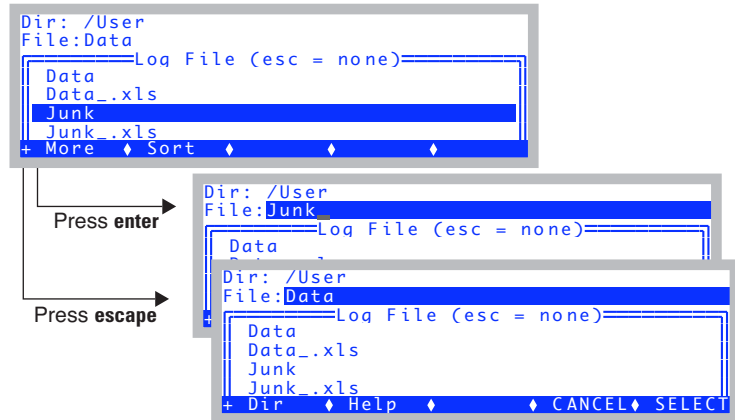
Figure 3-58. "Exiting" the bottom file list box. **Enter** *selects the name,* **escape** *does not.*

Press **labels** to access the editing keys (Figure 3-59), then **f1** (**DelLn**) to clear the default name, then type **Experiment 1**, and press **enter**.
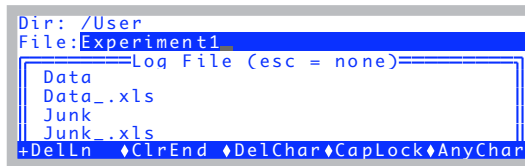
Figure 3-59. The editing fct keys for the Standard File Dialog box.

There are some illegal characters for file names (such as ':' and '/'), but this dialog will not let you type them. Spaces are ok.

If you enter the name of a file that already exists, you will be given the choices of overwriting it, appending to it, or cancelling to enter a different name.

5   **Enter initial remarks.**

You will be shown the remark entering box (Figure 3-60). Type a remark if you wish, and press **enter**.
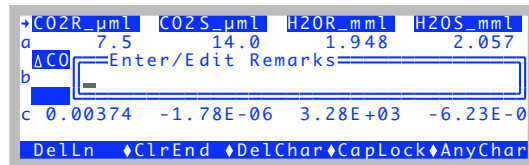
Figure 3-60. The prompt for entering remarks into a log file.

6   **Redo Experiment #1 on page 3-28.**

At each equilibrium value (that is, at the end of steps 4, 6, 8, and 9 of Experiment #1), press **Log** (**f1** level 1) to record data at those moments.

While the log file is open, the level 1 **f1** function key label (Figure 3-61) will show the number of observations logged.
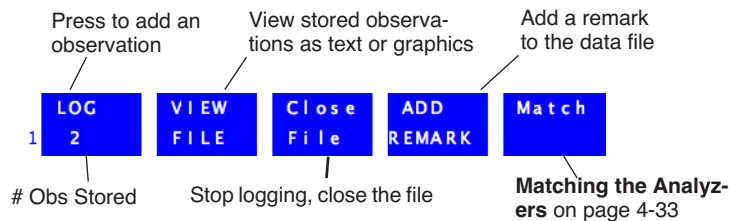
Figure 3-61. Level 1 function key labels when logging to a file.

7   **Graph the data**

Earlier we showed how to view strip charts of data in real time. Now we will show how to graph data you have recorded before closing the data file.

Viewing Stored Data

It is convenient - and sometimes necessary - to examine the specific data that has been logged, to see if the experiment is going as planned, or if an adjustment (such as a career change) is necessary.

When logging is active (to a file), you can view logged data by pressing **View File** (**f2** level 1).

Experiment #2   **View Data Logged Thus Far**

You can view data in your log file right from New Measurements mode, as long as the file is still open.

OPEN's Graphics Packages

OPEN uses two graphics packages: Real Time Graphics is used to plot measurements as they happen. GraphIt, on the other hand, is only used with data that is stored in a file.

---

1   **Access GraphIt (View File)**

Press **1** (if necessary), then **f2**.

LOG	VIEW	Close	ADD	Match
1 4	FILE	File	REMARK	

After a few seconds, you will see something like Figure 3-62.

The file to which you are logging ⎯

```
              GraphIt
Source= Experiment 1
(no graphs defined)



Import              View    Edit
1GrafDef            Data    GrafDef
```

Pick from some pre-determined graph parameters   View the data file text   Define graph parameters
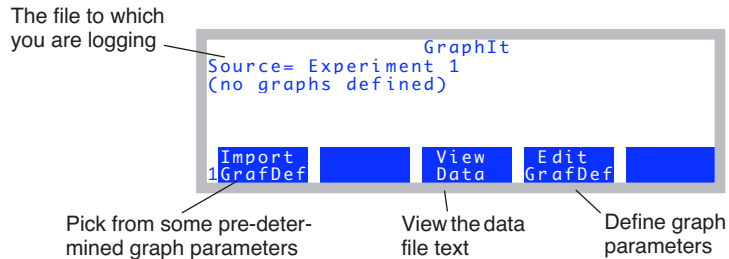
Figure 3-62. The first time GraphIt is entered from New Measurements mode.

This is GraphIt, a useful program that appears in several contexts, and is explained thoroughly in Chapter 12. For now just follow along, and we'll make some graphs.

2 **View the data file as stored**

Press **f3**, then **F** (Figure 3-63).

Appears only if graph
axes are defined

Appears only if curve-
fits are defined

```
View Options:
F - File as stored
H - Header
D - Data set (all vars and obs)
G - Graphed vars & obs only
C - curvefit Coefficients
E - curvefit Errors
```

Figure 3-63. GraphIt's View File screen provides options to view the file or subsets thereof, or to view curve fit coefficients and errors.

After pressing **F**, you should see something like Figure 3-64a. If you press **pgdn**, you will see more of the file (Figure 3-64b).

a)
```
========/User/Experiment 1====== 0=
"OPEN 6.2"
"Wed Sep 14 2011 17:01:39"
<open><version>"6.2"</version>
<open><configfile>"/User/Configs/UserPre
<open><light><source>"Sun+Sky"</source><
<open><comps><file>"/User/Configs/Comps/
<open><prompts><onlog>off</onlog><items>
```

b)
```
========/User/Experiment 1====== 2527=
<li6400><factory><unit>"PSC-1646"</unit>
<li6400><user><flow_zero>5000.0</flow_ze
"17:01:40  "
$STARTOFDATA$
"Obs"■"HHMMSS"■"FTime"■"EBal?"■"Photo"■"
1■"17:01:42"■10.5■0■12.7■0.0748■-10.9■1.
2■"17:01:43"■12.0■0■12.7■0.0748■-10.9■1.
```

*Figure 3-64. Viewing a file "as stored". a) This is the header information at the top of the file. b) After pressing **pgdn**, the rest of the file is shown. An explanation of the top line banner in each figure is given in Figure 5-2 on page 5-3.*

The little blocks (•) are tab characters, which is the default delimiter.

If you could see all of the file at once, it would look like Figure 3-65. If this

seems an inconvenient way to view the data, keep reading (it gets better).

```
"OPEN 6.2"
"Wed Sep 14 2011 17:01:39"
<open><version>"6.2"</version></open>
<open><configfile>"/User/Configs/UserPrefs/FactoryDefault_6.2.xml"</configfile></open>
<open><light><source>"Sun+Sky"</source><par_in>1<sensor>"GA-1646"<cal>0.73<actinity>1.00</actinity><transm>1.00</transm>...
<open><comps><file>"/User/Configs/Comps/StdComps_6.2"<header>""</header><extras></extras></file><energybal>no<f_parln>...
<open><constants><oxygen>21 %</oxygen><bb_vapor>1.5</bb_vapor><bb_oxy>0.9</bb_oxy></constants></open>
<open><stability><items>"Std Stability"<items[1]><id>-2</id><size>15</size><pcv><onoff>0</onoff><value>1</value></pcv>...
<open><log><format>"StdLogFmt_6.0"<items>{ -35 -21 -36 -76 30 23 36 21 25 221 -33 111 -34 11 -9 -10 -8 -1 -2 -4 -5 -14 -15 -7 -12 -13 ...
<open><a2d><avgtime>4.0 secs</avgtime><userchans><ch20>"Off"</ch20><ch21>"Off"</ch21><ch22>"Off"</ch22><ch23>"Off"</ch23>...
<li6400><factory><unit>"PSC-1646"</unit><serviced>"22 Nov 2004"</serviced><fuseaware>0</fuseaware><co2mixer>yes</co2mixer>...
<li6400><user><flow_zero>5000.0</flow_zero><irga_zero><co2>303.9 592.7<at>49.6493</at></co2><h2o>-262.1 -461.5<at>88.216...
"17:01:40 "
$STARTOFDATA$
"Obs"•""HHMMSS•""FTime•""EBal?•""Photo•""Cond•""Ci•""Trmmol•""VpdL•""CTleaf•""Area•""BLC_1•""StmRat•""BLCond•",...
1•"17:01:42•"10.5•0•12.7•0.0748•-10.9•1.83•2.4•26.51•6•1.42•1•2.84•24.56•26.51•26.02•289.37•,...
2•"17:01:43•"12.0•0•12.7•0.0748•-10.9•1.83•2.4•26.52•6•1.42•1•2.84•24.56•26.52•26.02•289.59•,...
3•"17:01:44•"13.0•0•12.8•0.0744•-14•1.83•2.4•26.52•6•1.42•1•2.84•24.56•26.52•26.02•289.54•,...
4•"17:01:46•"14.5•0•12.5•0.0747•-7.88•1.83•2.4•26.52•6•1.42•1•2.84•24.56•26.52•26.02•289.45•,...
5•"17:04:57•"206.0•0•12.7•0.0746•-12.8•1.83•2.4•26.52•6•1.42•1•2.84•24.56•26.52•26.02•289.65•,...
```

Figure 3-65. Experiment 1 as stored. The lines have been shortened to fit the figure (• represents tabs).

3   View the Header
Press **escape** to stop viewing the file "as stored", and then press **H** (for Header). You will see, in convenient tree form, all the configuration and calibration information for the system as it was when you opened the file (Figure 3-66).


```
┌──Header  from  /User/Experiment  1══════╗
║ ►open
║ ►li6400
║
║
║
║ ▼/►    ◆        ◆          ◆  EXIT   ◆  SELECT
```
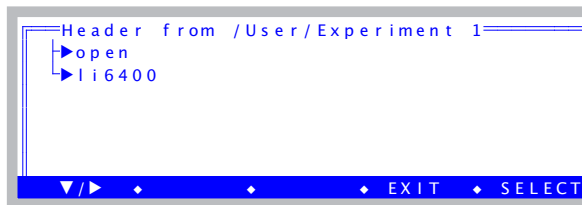
Figure 3-66. Configuration information is contained in the open *node, and calibration information is in the* li6400 *node. Expand and browse as you wish.*

4   **View the data file in columns**

Press **escape** to stop viewing the header tree, then press **D** (for Data Set). You will see a more legible display (Figure 3-67).

At the upper left corner of the data array.

```
┌Obs          HHMMSS        FTime        EBal?┐
│1            17:01:42      10.5         0     │
│2            17:01:43      12.0         0     │
│3            17:01:44      13.0         0     │
│4            17:01:46      14.5         0     │
│5            17:04:57      206.0        0     │
│6            17:04:58      207.0        0     │
     Sort    ♦  Find    ♦  ReFind♦  JumpTo♦  OK
```

After paging to the right (**shift** + →) five times.

```
┌Tleaf        TBlk          CO2R         CO2S ┐
│26.51        26.02         289.37       271.7│
│26.52        26.02         289.59       272.0│
│26.52        26.02         289.54       271.8│
│26.52        26.02         289.45       272.0│
│26.52        26.02         289.65       271.9│
│26.52        26.02         289.64       271.8│
     Sort    ♦  Find    ♦  ReFind♦  JumpTo♦  OK
```

Figure 3-67. Viewing the data set in column format. Only the data is shown. Use **shift** → *and* **shift** ← *to page left and right, and* **ctrl** → *and* **ctrl** ← *to move one column at a time.*

5   **Define a plot**

Press **escape** twice to stop viewing the data in columns and return to Graph-It's main screen. Then press **f4** (**Edit Config**) to define the axes for the plot.

```
Edit which item:
X - X (horizontal) axis
Y - Y (left vertical) axis
Z - Z (right vertical) axis
L - Logic for including observations
C - Curves

Press choice or <esc>
```

Figure 3-68. GraphIt's configuration editor screen. For details, see **Using Edit GrafDef** *on page 12-7.*

6   **Put *Flow* on the X axis, autoscaled**

Press **X**. If you are presented with

```
V – change variable
R – change range
```

then press **V**. (This choice will not appear if the X axis is currently undefined.) You will see a menu of variables, obtained from the label line in the data file. Scroll down (probably 5 **pgdn**'s from the top) until you find Flow. Highlight it and press **enter**.

Now set the range. (If you had to press **V** to change the variable, you will have to press **R** now to set the range.) You will be asked for the minimum and the maximum of the axis (Figure 3-69). For each, just press **enter** to accept * for the value.

```
Enter MIN (* to autoscale)
*
Enter MAX (* to autoscale)
*

 DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

*Figure 3-69. Range information is obtained in two prompts. Enter a
* for either the minimum or the maximum to let that value be adjusted
according to the data being plotted.*

When you are done, if the display does not return to the menu shown in Figure 3-68, you will have to press **escape**.

7   Put *RH_S* on the Y axis
Press **Y** to pick a Y variable. (If you get the V/R choice again as in Step 6, press **V**). Highlight RH_S, and press **enter**.

Now press **escape** to signal no more Y variables (you will be asked for up to 5 variables for this axis, and you only want one).

Now enter * for the Min and again for the Max. (If the minimum and maximum are not automatically asked, press **R** to do this).

Press **escape** until you get back to GraphIt's main screen.

8   Plot the graph
If it did not plot automatically upon leaving the configuration editor, then press **f2** to plot the graph (Figure 3-70).
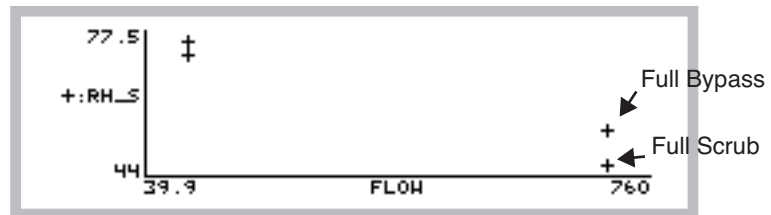
Figure 3-70. Sample humidity as a function of flow rate from Experiment #1.

These data illustrate the concept of an operating envelope for humidity control; if we draw a line around these data, we draw the envelope (Figure 3-71).
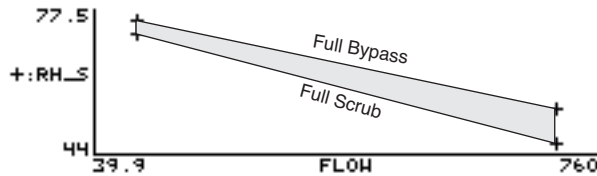
Figure 3-71. The operating envelope from Experiment #1. Inside the shaded area is achievable, outside is not

9 **Save this definition for future use.**
Press **escape** to stop viewing the graph, and return to GraphIt's main screen. Press **f5** (**Config SaveAs**), and name the file "RH & Flow". (Storing this file uses the Standard File Dialog again, just as you saw when you opened a log file.)

10 **Quick! Plot something else!**
From GraphIt's main screen, press **f1** (**Import Config**). You will be presented with a menu of plot definitions, which should include "RH & Flow", or whatever you named it in the last step. Just for fun, let's pick another one. Try selecting "A-Ci Curve" at the top of the list. You will get a very uninspiring plot (Figure 3-72) but it illustrates how to quickly change plot definitions.
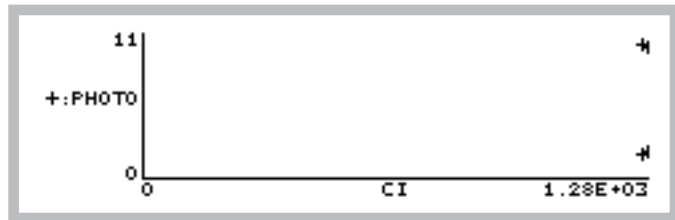
Figure 3-72. A-Ci data from Experiment #1.

11 **Replot RH and Flow…**
Press **escape**, then **f1 (Import Config)**, and select "RH & Flow", to redraw to our first plot.
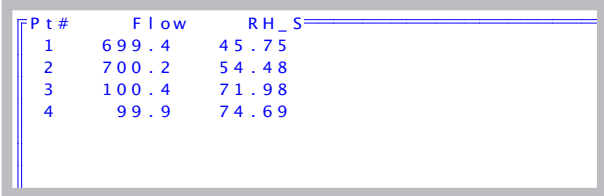
12   **…and view the plotted data**
Instead of pressing **escape** to stop viewing the graph, you can press **V**, and a scrollable menu of the data points that are plotted will appear (Figure 3-73).

```
┌Pt#      Flow      RH_S
│  1     699.4     45.75
│  2     700.2     54.48
│  3     100.4     71.98
│  4      99.9     74.69
```

Figure 3-73. Pressing V after drawing a plot in GraphIt will generate a list of the data points plotted.

Press **escape** to exit from this.

13   **Return to New Measurements mode, and close the file**
Press **escape** until you get back to New Measurements mode, then press **f3** (**Close File**).

That concludes the introduction to GraphIt, the tool for viewing logged data. It is accessible from New Measurements mode for a file that is open for logging, and also from the Utility Menu for previously logged files. GraphIt is fully described in Chapter 12.

Automatically Logging Data

OPEN's mechanism for automatic operation is the AutoProgram. An Auto-Program is a list of instructions that the instrument should do, which typically includes setting controls and logging data. OPEN includes a small collection of AutoPrograms for various common tasks (described in **AutoPrograms** on page 9-31), but you can modify these and add to the collection.

Experiment #3   **Log Data At Regular Intervals**
We will demonstrate AutoPrograms with one that logs data at regular intervals. Start in New Measurements mode.

1   **Open a data file**
Press **Open LogFile** (**f1** level 1) to name and open a data file.

2   Pick the AutoProgram "AutoLog2"
Press **5** then **f1**. A menu of AutoPrograms will appear. Select the one named
AutoLog2, and press **enter** (Figure 3-74).

```
Dir: /User/Configs/AutoProgs
═════════════Select a Program═════════
  A-CiCurve2
  AutoLog2
  CO2Curve_MutipleLight
  FlrLoop2
  LightCurve2
+ Dir      ◆         ◆         ◆ CANCEL◆ SELECT
```

Figure 3-74. Selecting an AutoProgram.

3   Append to current log file?
Whenever you launch an AutoProgram with a data file open, you'll be asked
if you'd like to append to that file (Figure 3-75).

```
→CO2R_µml  CO2S_µml  H2OR_mml  H2OS_mml
a                                      .851
  ┌─────────────────────────────────┐ S_%
  │ Append to the current log file?_ │
b └──────────────────────────────────┘ 6.57
  ═══════════════════════(Y/N)═══════ mmol
c  0.0355  -0.000102  1.02E+03  -0.00169
```
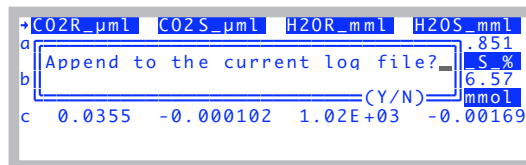
Figure 3-75. Starting an AutoProgram with a data file already open.

4   Configure the program
You will be shown a configuration screen (Figure 3-76). Set it to log every 15
seconds over 3 minutes.

1. Expand the Summary node.

```
  ═══ (1/2) AutoLog2 Setup═══════
→▶Summary: Every 10s for 00:03:00
  ▶Log Opts: beep ctrl



+  ▼/▶   ◆         ◆         ◆cancel ◆ START
```
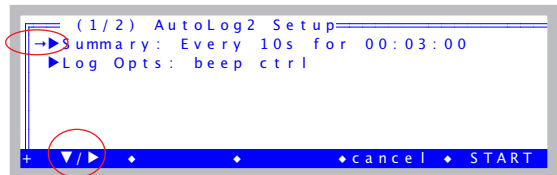
2. Set as shown.

3. Press Start.
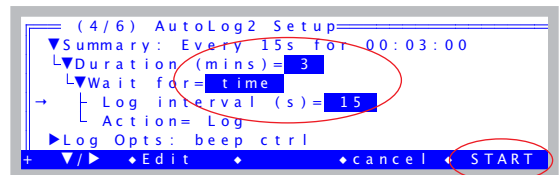
```
  ═══ (4/6) AutoLog2 Setup═══════
  ▼Summary: Every 15s for 00:03:00
   └▼Duration (mins)=  3
    └▼Wait for=  time
→   ├ Log interval (s)=  15
    └ Action= Log
  ▶Log Opts: beep ctrl
+  ▼/▶  ◆Edit   ◆         ◆cancel ◆ START
```

Figure 3-76. Configuring AutoLog2.

5 Watch
Press **1** to view the Log key label. Every 15 seconds, the number of observations will increment by 1.

AutoProgram active

Number of observations

Press **K** to watch the AutoProgram status line (Figure 3-77).

Figure 3-77. The AutoProgram status line includes Program, *the time remaining, and* ProgPrgs, *the current and total program steps.*

Normally, *Program* shows the number of seconds until the next event (typically, that event is data logging) will happen, and *ProgPrgs* ("Program Progress") shows the number of logging events so far, and the total number when done. With *AutoLog2*, however, *Program* shows the time remaining in the whole program.

6 Terminate it
To stop an AutoProgram before it is done, press **escape**, then **A** (Figure 3-78).
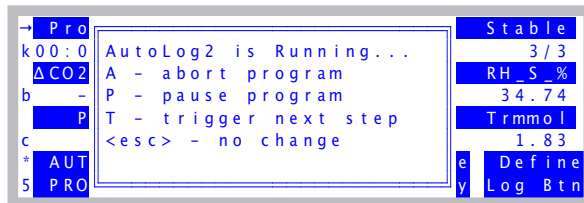
Figure 3-78. The AutoProgram Exit screen. Pressing **A** *will terminate the Auto-Programs,* **P** *will pause,* **T** *will trigger the next step in the AutoProgram, and* **escape** *will let the AutoProgram resume*

Notice the *T* option in Figure 3-78. Normally, this triggers the next step in an AutoProgram. With *AutoLog2*, however, it too will terminate the program. (If you want to log early with *AutoLog2*, just press the **Log** key.)

7   **Close the data file.**

Press **1** then **f3** (**Close File**) to do this. Notice that AutoPrograms don't automatically close data files. You can open a file for logging, then run several AutoPrograms, accumulating data in that file. However, if you launch an AutoProgram without having a log file open, you will be asked to open one. A complete discussion of AutoPrograms is in Chapter 9.

Stability

An important question concerning data logging is when to do it. That is, do I log now, or should I wait longer for things to be more stable? Manual logging usually involves that question being handled at some conscious level by the operator. Running an AutoProgram, however, leaves that decision to the machine.

Are there objective criteria that can be used by both the machine and the operator to decide on stability?

We would not ask that if the answer were not yes, and indeed, OPEN provides a fairly powerful technique for you to define and use stability criteria. **f4** level 5 (**Define Stablty**) lets you select your standards for stability, and we'll go there next.

Experiment #4   **Set Up System Stability Criteria**

Start in New Measurements mode.

1   **Press f4 level 5.**

You will see a screen similar to Figure 3-79, listing the current stability criteria.

```
╔════════════Stability Editor═══════════╗
║ Label: Std Stability                   ║
║ 1: CO2S (-2) 15 Slp<1                  ║
║ 2: H2OS (-5) 15 Slp<1                  ║
║ 3: Flow (-7) 15 Slp<1                  ║
║                                        ║
╟────────────────────────────────────────╢
║ ◆Edit   ◆Add    ◆Remove ◆Cancel ◆OK    ║
╚════════════════════════════════════════╝
```
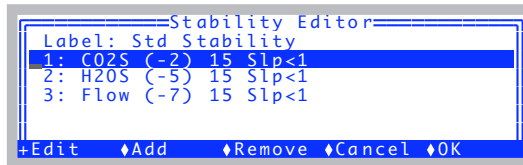
Figure 3-79. Defining stability, accessed by f4 level 5.

Each variable to be considered is listed, along with the time period and relevant thresholds.

2   Clean out the list
If there are items in the list, highlight each and press **Remove** (**f3**) until there are none. This is not normal operating procedure, but it gives us a clean slate to work with.

```
═══════════Stability Editor═══════════
 Label: Std Stability



+Edit    ♦Add      ♦        ♦Cancel ♦OK
```
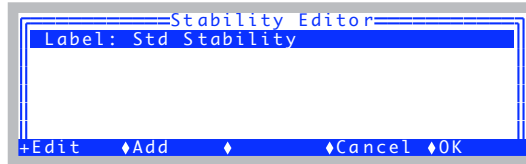
Figure 3-80. All entries removed (except the label).

3   Build a definition
Now, we need a definition of stability. An obvious one would be to look at photosynthesis and conductance. Over the course of a minute, if photosynthesis changes by less than 0.1 μmol m^{-2} s^{-1}, and conductance changes by less than 0.05 mol m^{-2} s^{-1}, then we could consider that stable.

For purposes of an introductory tour, however, let's pick reference cell CO_2, sample cell CO_2, and leaf temperature. (There's no logic to this combination, but it will work fine for our example.) A good rate of change threshold for the CO_2 values might be 1.0 ppm per minute, and 0.1 °C per minute for the leaf temperature.

We have been referencing rates of change as "per minute". Does this mean we have to wait 1 minute after stability is achieved to see if it holds? No. Instead, we will specify some time period, such as 10 or 20 seconds, over which OPEN will keep running statistics. If the rates of change over the last 10 or 20 seconds meet our criteria, then stability is reached.

There is a danger to doing it this way, however. A fluctuating signal could happen to have a rate of change near zero for an instant during a transition from "going up" to "going down", which could cause an AutoProgram to be fooled into logging too soon.

We can prevent this by specifying a second parameter, such as standard deviation or coefficient of variation, in addition to a rate of change (hereafter called slope, or abbreviated Slp). Table 3-4 illustrates our "tour stability cri-

teria", designed to avoid being fooled by a fluctuating signal.

Table 3-4. Example stability criteria

Quantity	Slope	Std Dev
CO2R	< 1.0	< 1.0
CO2S	< 1.0	< 1.0
Tleaf	< 0.1	< 0.1

4   Add CO2R, CO2S, and Tleaf to the list
Press the **Add** key (**f2**). A list of variables will appear (Figure 3-81).
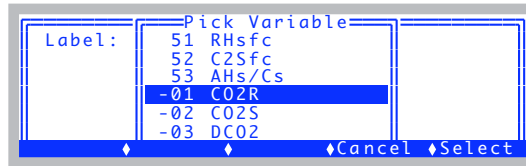
Figure 3-81. Adding a variable.

Select **−1:CO2R**, and press **enter**. Press **Add** (**f2**) again, and this time select **−2:CO2S**. Do it a third time and select **−10:Tleaf** (Figure 3-82).
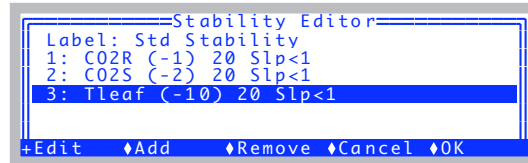
Figure 3-82. After clearing the list, and adding CO2R and CO2S, and Tleaf.

Notice the default setting for each item is to check for slopes, with a threshold of 1.

5   **Change the label**
Set the label to Exp#4.
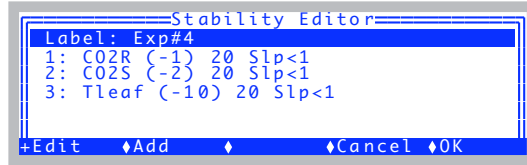
Figure 3-83. Changed the label.

6   **Want to make changes?**
Highlight CO2R, and press **Edit** (**f1**) or **enter** (Figure 3-84). Here we see the options for each item in the stability list: the variable (quantity to be tracked), time period (seconds), coefficient of variation (in percent), rate of change (per minute), and standard deviation.
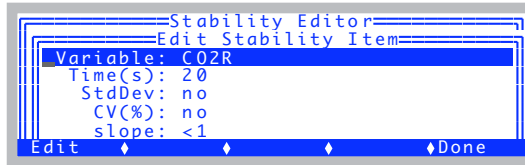
Figure 3-84. Editing an entry in the stability list. Time(s) *is time period for the running computations,* StdDev *is standard deviation,* CV(%) *coefficient of variation, as percent, and* Slp *is slope (rate of change with time).*

This is how you can select the time period, and enable or disable any combination of the three possible statistics for a variable. Press **Done** (**f5**) to exit the Stability Item Editor.

7   **Exit the editor**
Press **OK** (**f5**) to exit the Stability Editor, and return to New Measurements mode.

In the next part of the tour, we will see how to check on the system's stability during operation.

Experiment #5   **Monitor Stability In New Measurements Mode**

1   **Bring up display line 'e'**

Display line *e* contains two system variables that tell you moment by moment if the system has achieved your definition of stability. In the example in Figure 3-85, one of the three is stable.

Figure 3-85. Display line e shows three stability indicators. Stable shows the number of stable variables and the total number checked. StableF is the decimal equivalent of that fraction. The third quantity CCT (in this case) indicates which variables are stable or not.

2   **Close the chamber, Pump On, Soda Lime Full Scrub**

Close the chamber, turn on the pump, put the soda lime on full scrub, and watch display line *e,* as one by one, the variables become stable.

If you are thinking that waiting for "000" to become "111", or "0/3" to become "3/3" is not particularly informative, we agree.

There is a way to see the details of your stability checking:

3 **Diagnostic Mode, Screen A**
Press **[** (left square bracket) to enter Diagnostics Mode. (That's the short cut. The long way is to press **f5** on level 6, labelled **Diag Mode**). You should see something like Figure 3-86. (If not, press **A**, and it will come into view.)

Not Stable

```
(A) Stability    Status=1/3
    1)CO2R*  2)CO2S*  3)Tleaf

Sec  20        20        20
 Mn  -0.682    10.6      26.9
SDv  1.7E-01   3.2E+00   6.4E-03
%CV  2.4E+01   3.0E+01   2.4E-02
Slp -3.3E+00  -6.3E+01  -1.3E-01
```

Figure 3-86. Diagnostic display A monitors the details of the stability computa-tions. If there are more than 4 quantities in the definition, ← and → will scroll the columns, while **home** *and* **end** *jump to the left and right limits.*

This is one of several diagnostic screens. (Quick Lesson: You press letters **a**, **b**, etc. to jump between the screens in diagnostics mode. To leave diagnostics mode and return to the normal New Measurements text display, press **[**.) In diagnostic display A, you can monitor all the details of your stability criteria. When a variable is not stable, it is marked with an asterisk, and the reason (such the rate of change too large) is highlighted.

Thus, there are two ways to check on stability: line *e* for a quick glance, or **[** (then **a**, if necessary) for the details.

For more on Diagnostics Mode, see **Diagnostics** on page 6-24.

AutoPrograms, Stability and Matching
Most of the default AutoPrograms can make use of this stability feature, and it takes the form of a *Stability Definition* node in the setup window. To see an example, try the next experiment.

Experiment #6 **View stability and matching setup in an AutoProgram**

1 **Re-open AutoLog2**
Re-open *AutoLog2*, but do not launch it yet.

2   Change from time to stability

Change the node *Wait for* from *time* to *stability* (Figure 3-87). There will be some extra nodes that appear.

```
══ (1/10) AutoLog2 Setup═══════════
→▼Summary: Every 30-120s for 00:03:00
 └▼Duration (mins)= 3
  └▼Wait for= stability
   ┌▼Stability wait= 30 to 120 s
   ├ Minimum (secs)= 30
   └ Maximum (secs)= 120
   ├ Match before log= "never"
   └ Action= Log
 ▼Stability Definition: 3 items
  └▼items= "Std Stability"
   ├ items[1]= CO2S (-2) 15 Slp<1
   ├ items[2]= H2OS (-5) 15 Slp<1
   └ items[3]= Flow (-7) 15 Slp<1
```
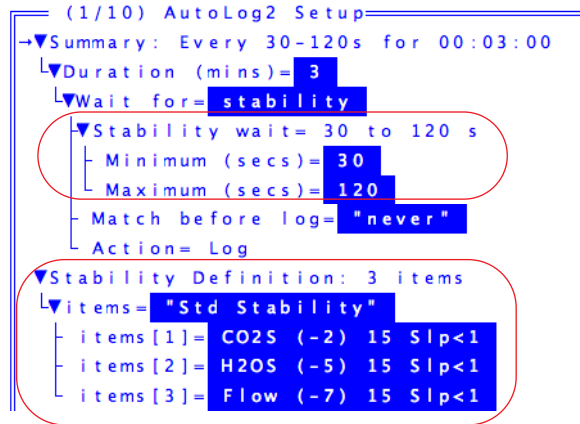
Figure 3-87. When stability is part of an AutoProgram, there will be two parts: The min and max wait times, and the stability definition.

Stability wait contains the *minimum* and *maximum* waiting times for stability to occur. *Stability Definition* contains the stability definition that will be implemented if the AutoProgram is launched. Note that this definition is not necessarily the currently active one - it is independent, and stored with the AutoProgram's other parameters. If you launch the AutoProgram, its stability definition is implemented, and remains in effect once the AutoProgram finishes. It will be the stability definition present the next time that AutoProgram's setup screen appears

AutoPrograms use stability along with a minimum and maximum wait time. When logging (each point on a light curve, for example), the program will wait the minimum time (after setting a new light level), then start checking stability and log when the stability definition is met. If this doesn't happen by the maximum time, it will log anyway, and move on to the next step.

You can change the currently active stability definition anytime, even while an AutoProgram is running.

3   **Navigate up to the *Match before log* node**
This is the node that control matching. Navigate to it and press **f2** (**Edit**). You will see it toggles through three settings (Figure 3-88).
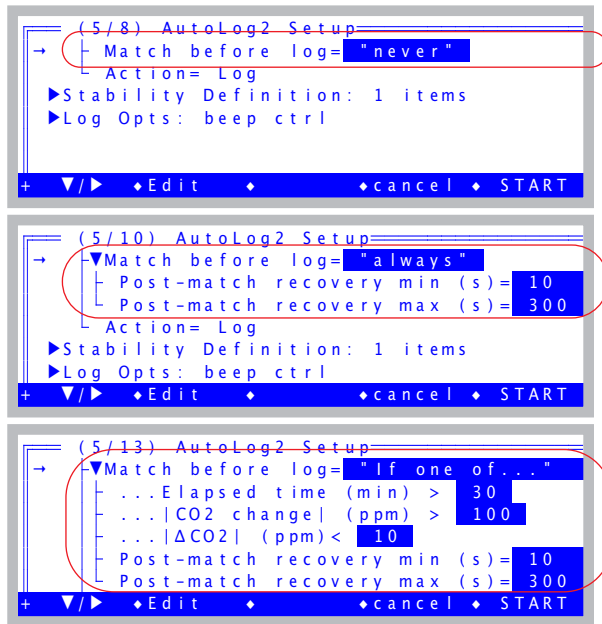
Figure 3-88. Configuring matching for an AutoProgram.

These are your matching options for AutoPrograms that have this node: you can <u>never match</u>, <u>always match</u> before logging, or only <u>match if</u> certain conditions are met. The conditional match occurs if at least one of the conditions is met: a) time since last match, b) change in CO_2 since last match, or c) a small CO_2 differential (the smaller the differential, the more important the match).

When matching does occur, there is a post-match recovery time to allow stability to be re-achieved, and the minimum and maximum wait times for that are editable.

4   **Cancel the AutoProgram**
Press **f4** (**cancel**).

Log Options
The next stop on the data logging tour will look at Log Options.

Experiment #7   **Log Stability Variable Statistics**
Start out in New Measurements mode.

1   **Access Log Options**
If you have a log file open, close it (**f3** level 1). **Log Options** is **f3** level 5. When you press it, you will see something like Figure 3-89.
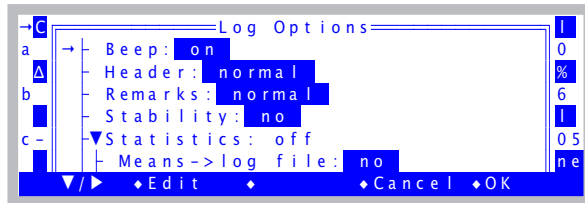
Figure 3-89. Log Options.

The entries in this menu give you several options for how your log file will look, and what is logged. The details are discussed in **Log Options** on page 9-14, but for now, focus on the fourth item, Stability. Setting this to logged (Figure 3-90) by highlighting it, and pressing **Edit** or **enter**, will cause the details (mean, standard deviation, coefficient of variation, and rate of change) of your stability variables to be included in your log file with each observation.
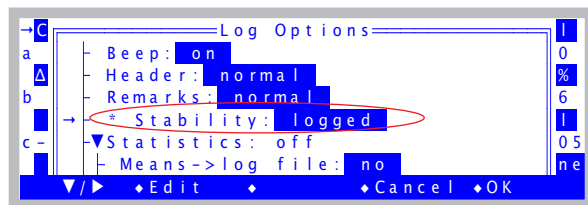
Figure 3-90. Stability details will be logged.

2   **Enable Stability Details**
Highlight Stability, and press **Edit** (**f2**). The line should change to Stability: Logged.

3   **Exit Log Options**
Press **OK** (**f5**) to leave **Log Options**.

4   **Open a log file, and log a couple of dummy observations**
Press **1** then **f1**, and enter a destination file name, remarks, etc. Then press **f1** (**Log**) a couple of times to log some data.

Guided Tours

5 **View the file**

Press **f2** (**View File**), then **f3** (**View Data**), then **D** (view Data set). You should see the first 4 columns of the data file:

```
┌Obs          HHMMSS      FTime      EBal?┐
│1            16:12:44    760.0      0    │
│2            16:12:45    761.0      0    │
│3            16:12:46    762.5      0    │
│                                         │
│                                         │
│   Print  ♦ Find   ♦ ReFind♦ JumpTo♦ OK │
```

Figure 3-91. Viewing the data in columns.

Now scroll to the right by pressing **shift** + →. (You can get to the end of the line quickly by pressing **shift** + **end**). Eventually, you will see something like Figure 3-92.

```
┌Status    ■ N(CO2R)    MN(CO2R)    SD(CO─┐
│110205      29          0.35       2.7E-│
│110205      31          0.347      2.8E-│
│110205      34          0.349      2.7E-│
│                                         │
│                                         │
│   Print  ♦ Find   ♦ ReFind♦ JumpTo♦ OK │
```

Figure 3-92. Scrolled way to the right.

Status is normally the last column. There will be 15 columns appended, containing the stability details. For each of the *CO2R*, *CO2S*, and *Tleaf* stability variables, there will be a N(), MN(), SD(), CV(), and SLP() columns, with the variable name in the parentheses (). N() is the number of samples, MN() is the mean value of those samples, SD() is standard deviation, CV() is coefficient of variation, and SLP() is slope, or rate of change.

6 **Return to New Measurements**

Press **escape** three times to get back to New Measurements, then close the file by pressing **f3** (**Close File**).

If you wish, you can go back and turn off stability logging now, or just leave it. The Log Options always revert back to their default settings at power on.

Experiment #8 **Make an Excel File**

You may have noticed one of the log options is `Excel File:` This option (which by default is `Yes`) creates an Excel version of your data file - with all equations built-in - along with the normal text version. For example, with this option set to Yes, when you open a log file named `Data`, there will also be another file created named `Data.xls`.

1   Access Log Options
If you have a log file open, close it (**f3** level 1). Then access **Log Options** (**f3** level 5) (Figure 3-93).
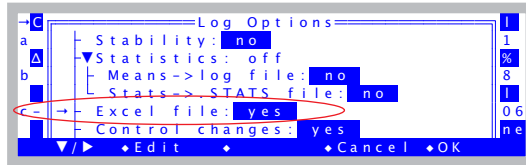
Figure 3-93. The Excel File option is in Log Options.

2   Create an Excel file (if necessary)
If the Excel File option has been enabled, and you have logged some data, you will already have an Excel file to work with. Otherwise, set it to Yes, and go open a log file (**f1** level **1**), log some data, and close the file.

3   Move an .xls file to your computer.
If you have not done this before, there are many options, depending on your computer (Windows, Mac OSX, Linux), and whether you have a LI-6400XT or not (Ethernet or RS-232); this is all spelled out in detail in Chapter 11. For purposes of this step, we will assume you have made either a direct Ethernet connection between the LI-6400XT and your PC, or else both are plugged into the same local sub network.

a) To verify you are connected, and to see your host name and IP address, access Network Status in the Utility Menu (Figure 3-94).
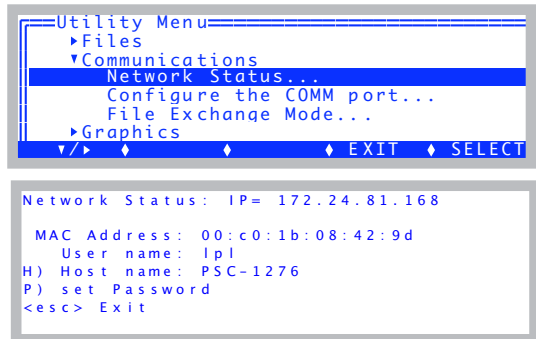
Network Status shows an IP address, if you are connected. The User name is always lpl (lower case LPL), as is the password, unless someone has changed it.

Figure 3-94. Network Status

b) Connect the LI-6400XT file system to your computer using Windows Explorer (Figure 3-95) or Max OS X Finder (Figure 3-96).

For Windows:
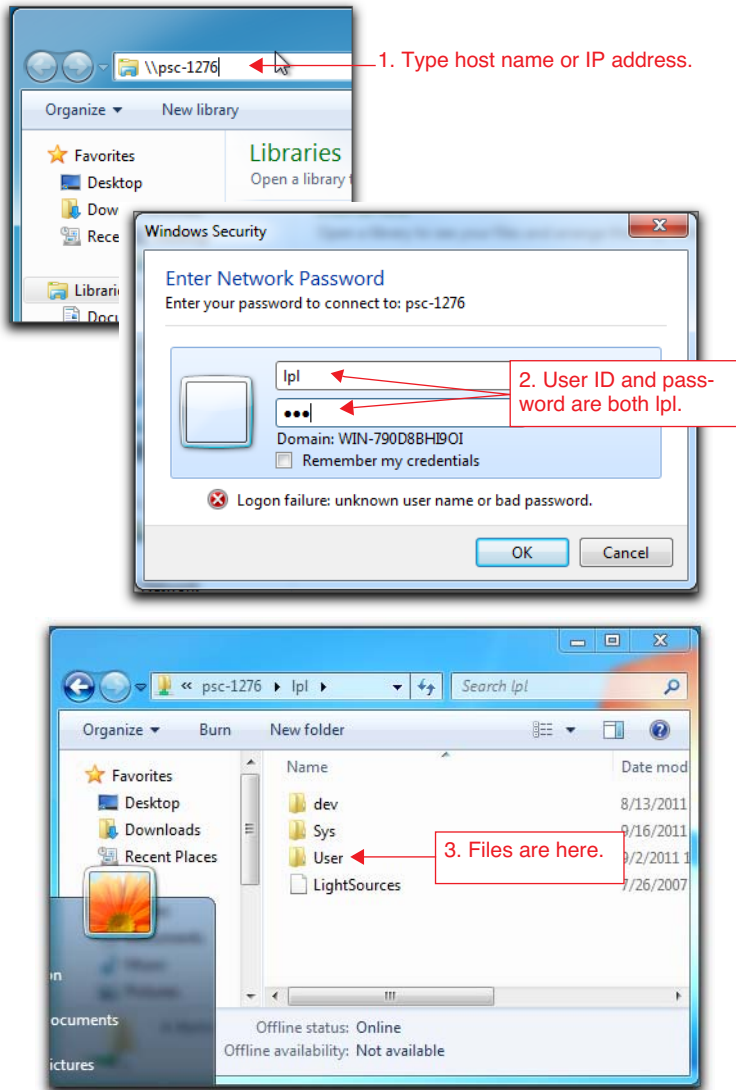
Figure 3-95. Connecting to the LI-6400XT file system using Windows.

For Mac OS X:
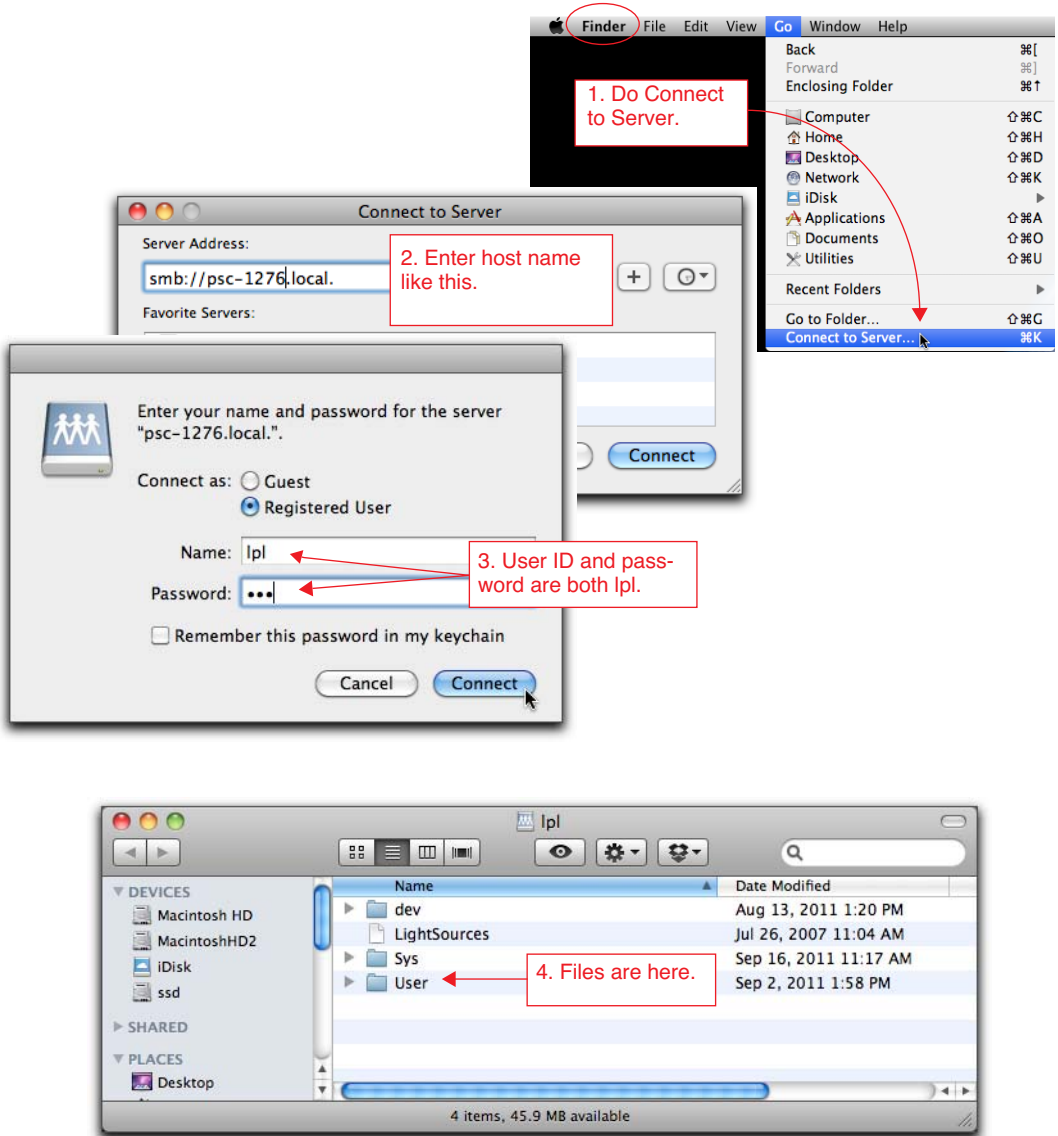
Figure 3-96. Connecting to the LI-6400XT file system using Finder.

c) Navigate to the User directory.

d) Drag the Excel file you like to view to the desired folder or desktop on your computer, and open it with Excel (Figure 3-97). You could simply double click it to open it without moving it to your PC, but note the cautionary box on the next page.

Figure 3-97. Sample .xls data file. Columns marked with "out" (Row 10) are defined by equations.

e) Change leaf area, and see it recompute automatically for that row. Note that some columns are marked "in", and some are marked "out" (row 10). The "outs" have equations associated with them so are computed, and the "ins" are simply inputs. Thus, if you change an input value, such as *Area* (column M), all the out columns in that row that are a function of area (such as *Photo*, column G) will change automatically.

Opening an Excel file that is on the console
This is OK.

This is OK even if the file is still active. However, as new observations are logged, you will not see them in Excel unless you re-open the file each time you want to check on it.

One thing to avoid <u>if it is still an active file</u>: Do NOT write to it. That is, do not overwrite the .xls file on the console. (Save it as an .xlsx, or save it somewhere else, but don't write to the original while it is still being used by the LI-6400, or it may become corrupted.)

Tour #6: Configuration Introduction

OPEN can be configured to do many things, such as controlling a light source, or measuring soil respiration. This tour will acquaint you with some basic concepts about managing configurations in OPEN. If you want more details on configurations, see Chapter 16.

Configuration Basics

We will start with the factory default configuration, make some changes, and see how to track and store them.

Experiment #1   Making Configuration Changes

1   Start with Factory Default

To implement this, go to the Config Menu and select **"Open..."**. If asked to select a configuration (this happens if there is more than one file in the list) (Figure 3-98), select **"FactoryDefault_6.2.xml"**.

```
Dir: /User/Configs/UserPrefs
══════════Select a new configuration══════
 ExtRch_1cm EB.xml
 ExtRch_1cm RGB EB.xml
 FactoryDefault_6.2.xml
 LCF.xml
 LED.xml
+ Dir     ◆       ◆      ◆ CANCEL◆ SELECT
```
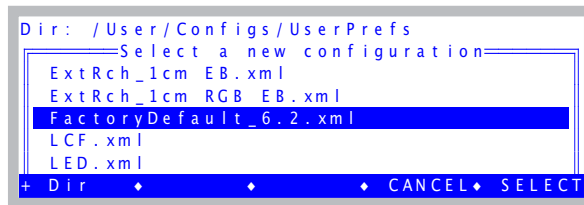
Figure 3-98. Prompting the user for a configuration file.

2   View the Configuration Tree

Select the bottom item, **"View/edit..."**. This will show you all the settings in the current configuration. You will be looking at a tree (Figure 3-99). **f1** will open and close nodes in the tree.

```
════════FactoryDefault_6.2.xml════════
→└▼open
  ├▶light
  ├▶comps
  ├▶prompts
  ├▶constants
  ├▶fctkeys
  ▼/▶   ◆      ◆      ◆      ◆Done
```
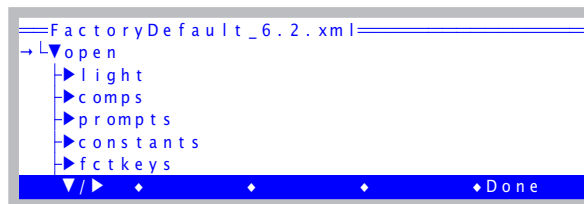
Figure 3-99. The configuration tree view.

Now, let's make some changes to the configuration and see what happens.

3   **Change leaf area**
Press **escape** a couple of times to get to OPEN's main screen, then enter New Measurements mode. Press the **AREA=** key (**f1** level 3), and enter **3** for the leaf area.

4   **Change stomatal ratio**
Press the **STOMRT=** key (**f2** level 3), and enter 0.5 for the new value.

5   **Return to OPEN's Main Screen**
Press **escape** to leave New Measurements mode. There will be a message waiting for you in the form of an asterisk in the Config Menu key label (Figure 3-100).

Figure 3-100. The asterisk in the Config Menu label in-dicates that something has changed.

This asterisk alerts you to the fact that your configuration has been changed and not saved. To save a configuration, simply select Save as... from the Config Menu.

Rather than doing that quite yet, however, continue reading, and we will see something interesting about the Configuration Tree.

Experiment #2   Using the Configuration Tree

1   **Track the changes**

Tracking what changed is a matter of following the trail of asterisks[6]. Start by pressing **f2** to get to the Config menu (Figure 3-101).

1. Press **f2**.
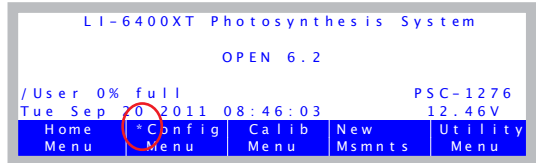
2. Scroll down to here, and press **f1**.

3. Scroll down to here, and press **f1**, to expand the node.
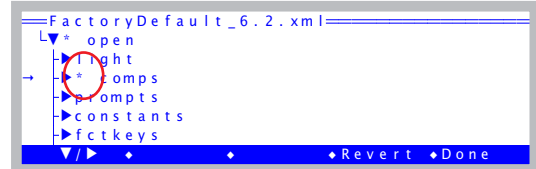
4. Press **f1** again here...
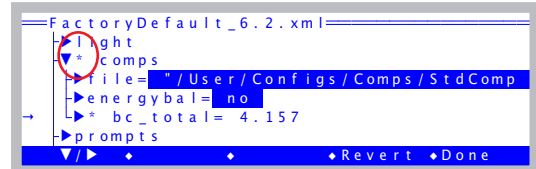
5. ...and again here...

6. ...and finally here.
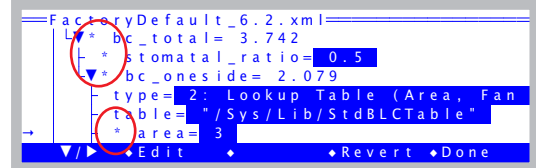
Figure 3-101. Changed nodes are shown with an asterisk.

In the tree view, modified nodes (different from how the config was stored)

[6.]Like following a trail of bread crumbs, but the results are less Grimm.

are shown with an asterisk. Also, any node that contains modified subnodes is shown with an asterisk. Thus, you can track down the specifics that have changed. In this example, the changed area and stomatal ratio nodes are at the end of the trail, as we would have expected.

Let's look closer at the tree view, because the structure here tells us something. The total boundary layer conductance (*bc_total*) is computed from *stomatal_ratio* and a one-sided boundary layer value (*bc_oneside*). The one-sided value, in turn, comes from a lookup table, leaf area, and fan speed (fan speed - *fan* - is just below *area* - you can scroll down and see it).
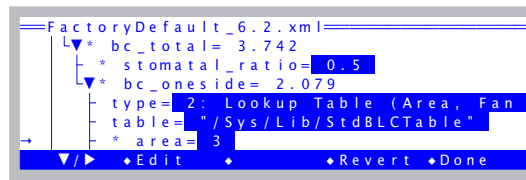
$$g'_{bw} = \frac{g_{bw}(K+1)^2}{K^2+1}$$

g'_{bw} = bc_total
K = stomatal_ratio
g_{bw} = bc_oneside
= Table(s, v_f)
s = area
v_f = fan

Figure 3-102. The node structure often reflects functional relationships - in this case, between the boundary layer conductance variables. See Eqn (1-9) and (1-10) on page 1-9.

2   **Change area again from here**
Put the cursor on the *area* node, and press **f2** (**Edit**). Notice that you can change area (and a lot more) from here - you need not go to New Measurements mode to do so.

Change the area back to 6...
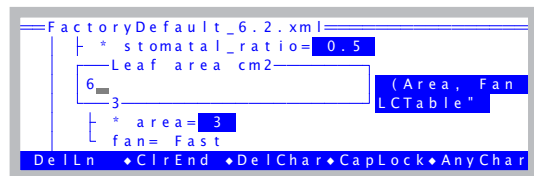
...and notice the asterisk goes away.
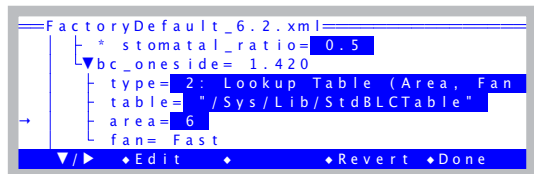
The asterisk on area goes away because area has returned to its original (saved) state. Notice too that the value of *bc_oneside* has changed from 2.079 to 1.420. This changed *bc_total* from 2.742 to 2.556.

3   Experiment

Try out different leaf areas, and fan speeds by changing them from right here in the tree view using the **Edit** key (**f2**), and see the effect on boundary layer. Also, try out the *type* node, which sets how boundary layer is computed.

```
══FactoryDefault_6.2.xml═══════════
        ═══════BLC Styles═══════
        0: Fixed Value
        1: Function of Area                 Fan
        2: Lookup Table (Area, Fan)         "

      └ fan= Fast

            ◆           ◆          ◆Cancel ◆Select
```
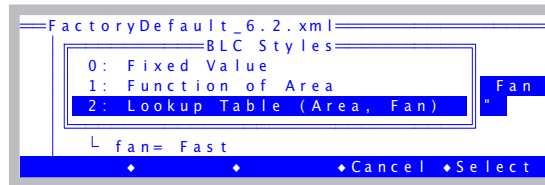
You will notice that the tree changes structure a bit depending on how type is set. The other two (type 0 and 1) are shown below:

In the Fixed Value mode, boundary layer only depends on stomatal ratio and a one-sided value.

```
══FactoryDefault_6.2.xml═══════════
  └▼ * bc_total= 2.556
   ├  * stomatal_ratio= 0.5
  ─▼ * bc_oneside= 1.420
   └  * type= 0: Fixed Value
 ─►prompts
 ─►constants
  ▼/►   ◆Edit      ◆         ◆Revert  ◆Done
```
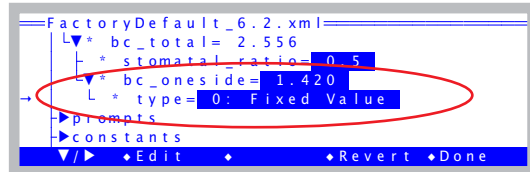
You can specify a slope and offset for the relation between area and one-sided boundary layer, using this mode.

```
══FactoryDefault_6.2.xml═══════════
  └▼ * bc_total= 2.556
   ├  * stomatal_ratio= 0.5
  ─▼ * bc_oneside= 1.420
   ├  * type= 1: Function of Area
   ├  area= 6
   ├  * slope= 0
  ▼/►   ◆Edit      ◆         ◆Revert  ◆Done
```
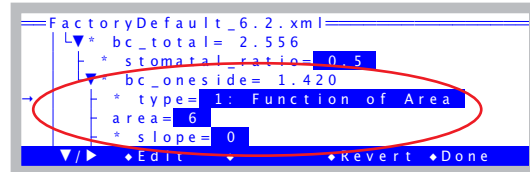
4   Revert Everything

Notice the **f4** key (**Revert**). With the cursor on a *-ed node, press **f4**. and you will be given a choice (Figure 3-103) of revert all nodes back to the original state by pressing **A**, or just the current node by pressing **T**.

The node in question.

```
  ┌►open comps bc_total bc_oneside offset┐
  │T - revert This node
  │A - revert ALL nodes
  │esc - Cancel ▁
  │                              ═══(T/A/esc)═══
  │    └ * offset= 1.42
```
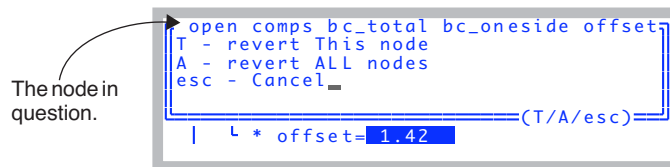
Figure 3-103. Reverting one or more nodes.

Adding Prompts

Suppose you wish to make survey measurements, where each observation you log is on a different leaf, with a potentially different leaf area. Also, you would like to add a column to the data that contains some identifier of that leaf, such as "Plot#". Finally, you wish to be prompted for leaf area and this identifier automatically each time you log a data record.

Experiment #3   **Add Prompts for Plot# and Leaf Area**
We will start in `Config Menu|View/edit`.

1   **Access the Prompts Node**
Navigate to the <open> <prompts> <items> node (Figure 3-104).

```
══FactoryDefault_6.2.xml══════
 └▼open
  ├►light
  ├►comps
  ├▼prompts
  ├  onlog=  off
  └  items=  "Default (none)"
→
 ▼/►   ♦Edit      ♦        ♦        ♦Done
```
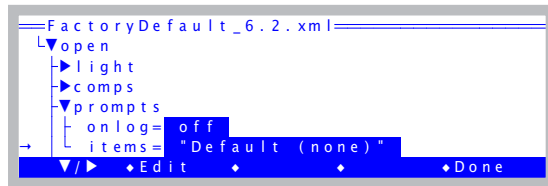
Figure 3-104. The list of prompts in the configuration tree.

2   **Open the Prompts Editor**
With the cursor on the *items* node, press **f2** (**Edit**). You will see a label, and an empty list. Press **f1** (**Edit**), and name the list "My Prompts".

```
═════════════User Prompts═══════════════
Label:Default (none)




+Edit    ♦Insert◻♦       ♦Cancel ♦OK
```

```
═══════════════User Prompts═══════════
 Label:Default (none)
  ══════Prompt list label══════
  My Prompts_


DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```
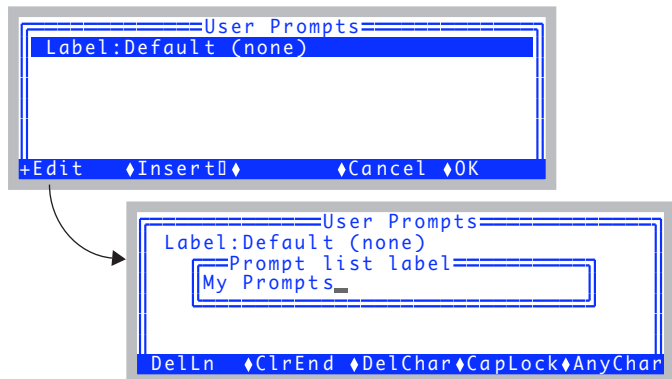
Figure 3-105. Editing the list of prompts.

3   **Add Area to the list**

Press **f2** (**Insert↓**). You will be shown a list of constants. Find -33 Area, and press **f5** (**ok**) (Figure 3-106).
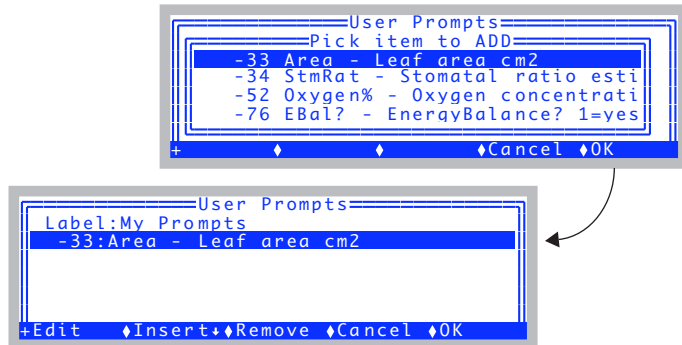
Figure 3-106. Adding area to the prompt list.

4   **Add a user-definable system constant to the list**

There is no system constant named "Plot#", so we will create one, using one of the nine user-definable system constants. Press **f2** (**Insert↓**) and scroll down to item −101 aux1, and press **f5** (**ok**) (Figure 3-107).
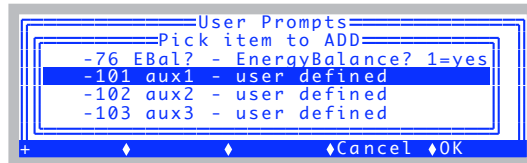
Figure 3-107. Selecting a user-defined system constant.

5   Change the label to "Plot#"

You will be shown a dialog that lets you define some attributes of our system constant (Figure 3-108). Make the label "Plot#".

You can define several characteristics of this item. For now, we will just change the Label.

```
┌─────────────Prompt  Item  Editor──────────┐
│ Label: aux1                                │
│ Description:                               │
│ Type: String     Entry: Type in value      │
│ Screen width 8                             │
│ Initial Value:                             │
│                                            │
│  Edit              Test    Cancel    OK    │
└────────────────────────────────────────────┘
```
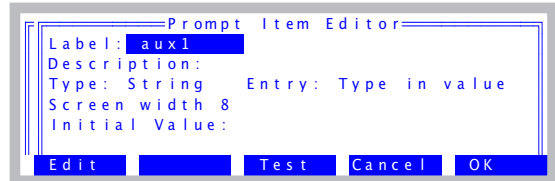
Change the label to Plot# by highlighting the label field, and pressing **f1** (**Edit**).

```
┌─────────────Prompt  Item  Editor──────────┐
│ Label: aux1                                │
│ De┌──Data label──────────────────────┐     │
│ Ty│ Plot#_                            │ue   │
│ Sc└───────────────────────────────────┘     │
│ Initial Value:                             │
│ DelLn    ◆ClrEnd ◆DelChar◆CapLock◆AnyChar  │
└────────────────────────────────────────────┘
```
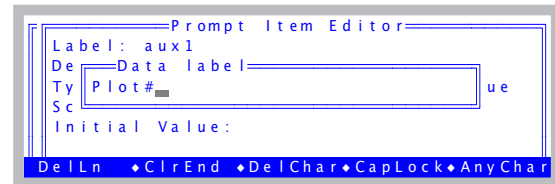
Figure 3-108. Defining the attributes of our system constant #-101.

After you press **f5** (**OK**) to keep your change, the list should look like Figure 3-109.

```
┌═══════════════User  Prompts═══════════════┐
│ Label:My Prompts                           │
│  -33:Area - Leaf area cm2                  │
│ -101:Plot# -                               │
│                                            │
│+Edit    ◆Insert◆◆Remove ◆Cancel ◆OK        │
└────────────────────────────────────────────┘
```
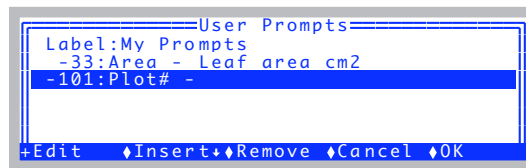
Figure 3-109. Plot# added to the prompt list.

6   Exit the Prompt Editor

Press **f5** (**OK**) to exit the prompt editor. You will be asked a series of questions about any items that you have added to the prompt list (Figure 3-110).

```
┌═══════════════User  Prompts═══════════════┐
│                                            │      Press N
│ Add Area (ID#-33) to Display (Y/N)?         │      Press N
│ Add Plot# (ID#-101) to Display (Y/N)?       │      Press Y
│ Add Plot# (ID#-101) to LogList (Y/N)?       │
│                                            │
└────────────────────────────────────────────┘
```
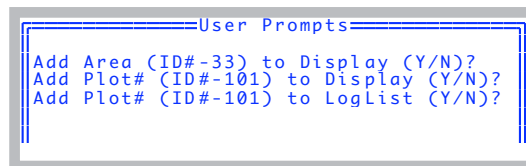
Figure 3-110. Items that were added to the prompt list can also be automatically added to the New Measurements display and Log List, if they aren't already there.

Once you are back in the View/Edit screen, the items node should look like Figure 3-111.
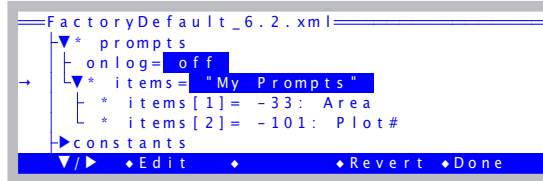
Figure 3-111. Two items in the prmpt list.

7   **Set for Automatically Prompting**
Move the cursor up to the *onlog* node, and press **f2** (**Edit**) to change it from *off* to *on* (Figure 3-112)
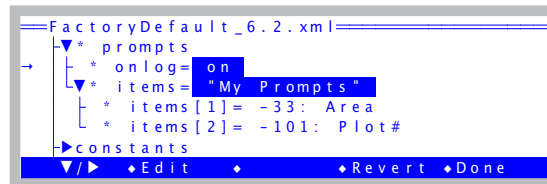
Figure 3-112. The onlog *node controls whether prompts are asked each time* **Log** *is pressed.*

8   **Try it out**
Escape out of the Config Menu, go to New Measurements mode, open a log file, and log a record. You should be prompted for area and plot number (Figure 3-113).
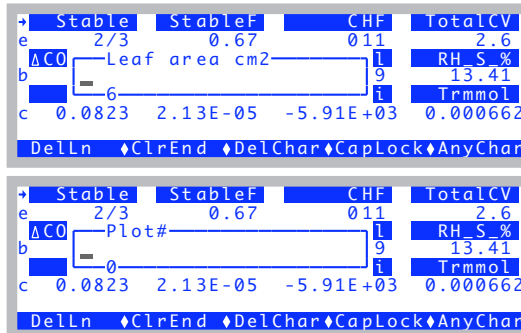
Figure 3-113. Automatically being prompted when **Log** *is pressed.*

9   **Prompt Control in New Measurements**

Level 3 of the function keys have some prompt related items: **f3**, **f4**, and **f5** (Figure 3-114). Try out each one.
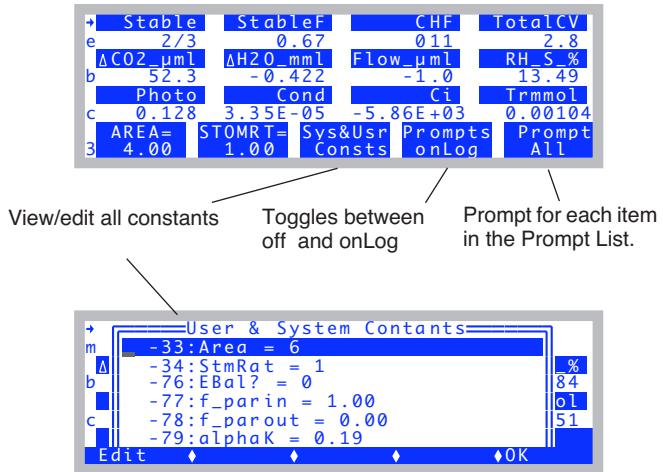
Figure 3-114. Prompt-related function keys on level 3.

10   **The Prompting String**

One last detail: Back in Step 5, we left the description of our Plot# item blank. If you wish to have a more elaborate prompting sequence for a user constant, put it there. For example, if we go back into the configuration menu and change Plot#'s description as shown in Figure 3-115, when it is prompted, that description is used, rather than the label.
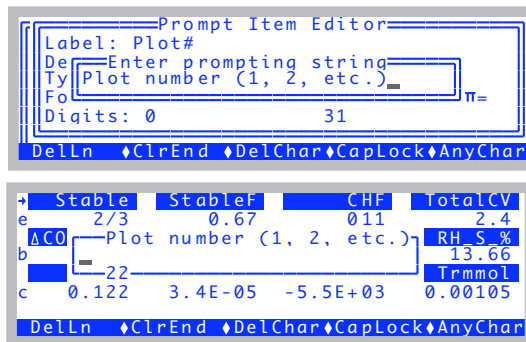
Figure 3-115. If there is a description, it will be used for the prompt.

Adding Computed Items

Suppose you wish to add an item that, unlike some constant like Plot#, needs to be computed. We illustrate that with a step-by-step example of adding a water use efficiency computation.

Experiment #4   **Add Water Use Efficiency**

First, we need a formula for water use efficiency. If photosynthetic rate is *A* (μmol m^{-2} s^{-1}), and transpiration *E* (mol m^{-2} s^{-1}), then water use efficiency *W* (in %) is

$$W = \frac{A \times 10^{-6}}{E} \times 100$$

$$= \frac{A}{10^4 \times E}$$

(3-1)

(The 10^{-6} converts μmol to mol). So, what we need to add is something labelled *WUE* (for water use efficiency) that is computed from the quantities *Photo* and *Trans* that are already being computed. OPEN's user-defined computations, like *Photo* and *Trans*, are defined in a ComputeList file (described in Chapter 15). One way to add *WUE* would be to change this file. OPEN gives us another method of adding user variables without actually touching the ComputeList file. Here's how:

1   Go to the *extras* node in the configuration tree

Go to Config Menu|View/edit. Navigate to the <open> <comps> <file> <extras> node (Figure 3-116).

Scroll down to *comps*

Press **f1** to expand *comps*'
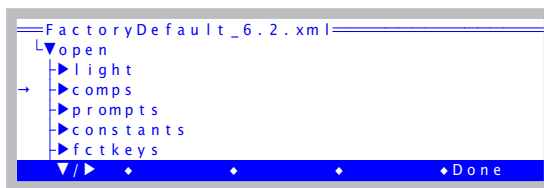
Scroll down to *file*, and expand it. (**f1**)

Scroll to *extras*
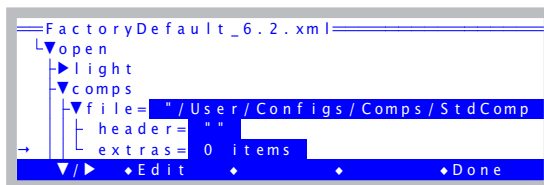
Figure 3-116. Navigating to the extras *node.*

2   **Add a new item.**

With the cursor on the *extras* node, press **f2** (**Edit**), followed by **f2** (**Add**) (Figure 3-117).

This is the list of ex-
tras. It is empty.
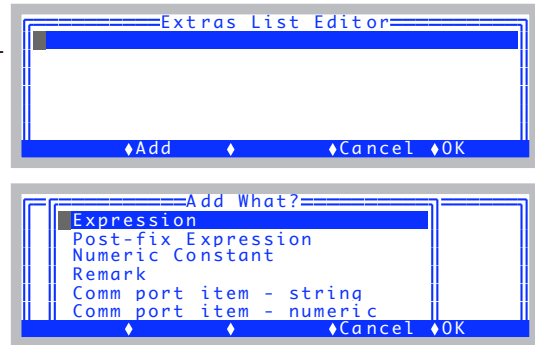
Figure 3-117. Adding an Extra.

What we are adding is an expression, so highlight Expression, and press **f5** (**OK**).

3   **Define the expression**

Enter the label and the description, as shown in Figure 3-118.
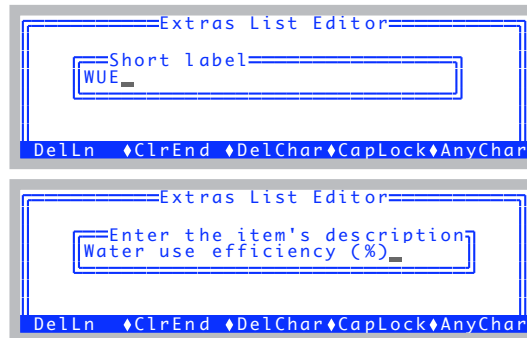
Figure 3-118. The label, and the description.

4   **Pick when it is to be computed**
With expressions, you can decide if they are to be computed **before** or **after** the normal user computations (items in the ComputeList). Since we are using items that are computed in the ComputeList (*Photo* and *Trans)*, we want to compute *WUE* afterwards.

Press **A**

```
==================Extras List Editor=============
===============Computed Item============
Compute this Before or After
normal computations?_

===========(B/A)===========
```
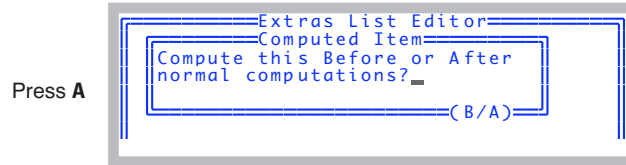
Figure 3-119. Selecting when it is computed.

5   **Enter the equation**
The default formula for a new expression always starts out with our variable equalling 1.0 (Figure 3-120). We need to edit this to implement the formula for *WUE* (Eqn (3-1) on page 3-100). To do this, a dialog known as the The Code Editor is employed, which is described in **Editing Code** on page 15-10.

1. Starting formula. *u1000* is the variable name for *WUE*.

2. Erase the 1.0, then press **ctrl** + **v**.

```
=============Enter the expression========
u1000 = 1.0
```

```
==========Enter the expression==========
u1000 = _

+DelLn   ◆ClrEnd ◆DelChar◆Cancel  ◆OK
```
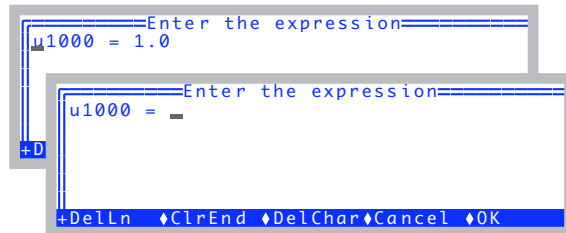
3. Scroll down to Photo, and press **f5** (**ok**).

```
=__ID_ __Label__  ___Variable_____
    23 "Cond" u23
    24 "vp_kPa" u24
    25 "VpdL" u25
    27 "VpdA" u27
    30 "Photo" u30
    35 "CndCO2" u35
       ◆       ◆        ◆Cancel  ◆OK
```
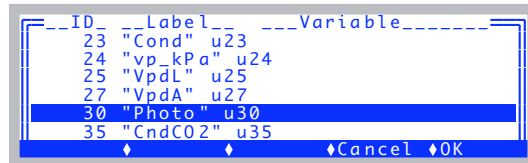
The variable name for Photo (*u30*) is inserted.

```
============Enter the expression========
u1000 = u30_

+DelLn   ◆ClrEnd ◆DelChar◆Cancel  ◆OK
```
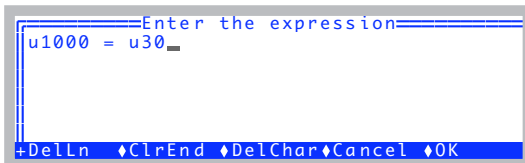
Figure 3-120. Entering the formula for WUE, part 1.

Finish the expression (Figure 3-121), either by using the **ctrl+v** method of in-serting the variable name for transpiration in the right place, or by simply typ-ing **u20**.

```
╔═══════════Enter the expression═══════════╗
║u1000 = u30/u20/10000_                     ║
║                                           ║
║                                           ║
║                                           ║
║                                           ║
║+DelLn   ♦ClrEnd ♦DelChar♦Cancel ♦OK       ║
```
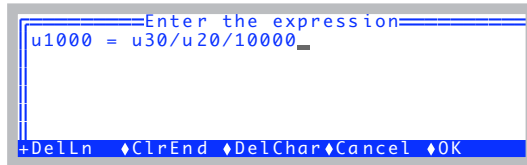
Figure 3-121. Entering the formula for WUE, part 2.

When the equation is correctly entered, press **f5** (**OK**).

6   Final details

You will be presented with a dialog for this extra item (Figure 3-122). Modify any of the fields you choose by highlighting the field, and pressing **f1** (**Edit**).

```
╔═══════════Extras Item Editor═══════════╗
║ID: 1000    Label: WUE                   ║
║Desc: Water use efficiency               ║
║Variable name: u1000                     ║
║Type: Expression                         ║
║Format: Sig. Digits      Digits: 3       ║
║Code: u1000 = u30/u20/10000              ║
║ Edit           Test   Cancel    OK      ║
```
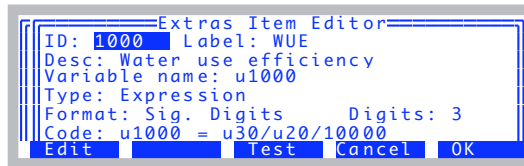
Figure 3-122. The Extras Item Dialog lets you change any attribute.

7   Test it

The **test** key (**f3**) will test the computation, and the display format (Figure 3-123). This will allow you to experiment with various settings of *Format* and *Digits*, for example.

With Format = Sig. Dig-its, *WUE* looks like this.

```
╔═══════════Edit User Item═══════════╗
║      WUE                            ║
║     0.639                           ║
║ok. Example display shown.           ║
║Press any key                        ║
```
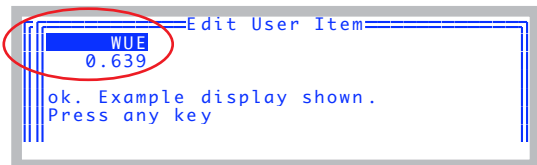
Change the format to Scientific, and press **Test** again.

```
╔═══════════Edit User Item═══════════╗
║      WUE                            ║
║6.394E-01                            ║
║ok. Example display shown.           ║
║Press any key                        ║
```
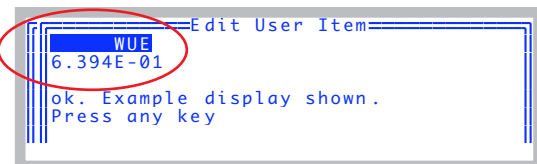
Figure 3-123. Using the Test key.

8   **Return to the List Editor**
Press **f5** (**OK**) to return to the List Editor (Figure 3-124).

```
════════════Extras List Editor═══════════
1000: WUE (Water use efficiency)



Edit    ♦Add     ♦Remove ♦Cancel ♦OK
```
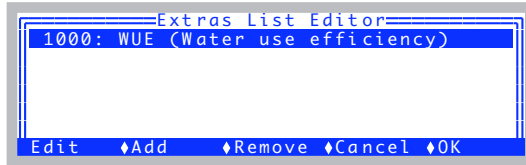
Figure 3-124. Our extra added to the list.

9   **Done Adding Extras**
Press **f5** (**OK**) to exit the List Editor. Because we have added something, we will be asked if we would like to add it to the New Measurements display and the Log List (Figure 3-125). Press **Y** for both.

```
═══════════Extras Item Editor══════════

Add WUE (ID#1000) to Display (Y/N)?
Add WUE (ID#1000) to LogList (Y/N)?█


```

```
Display Additions...
WUE -> line m
Press any key
```

Note this letter

```
═══FactoryDefault_6.2.xml═══════
  ▼* comps
  ▼* file= "/User/Configs/Comps/StdCo
    header= ""
  ▶* extras= 1 items
  ▶energybal= no
  ▶bc_total= 2.840
  ▼/▶  ♦Edit   ♦       ♦Revert ♦Done
```
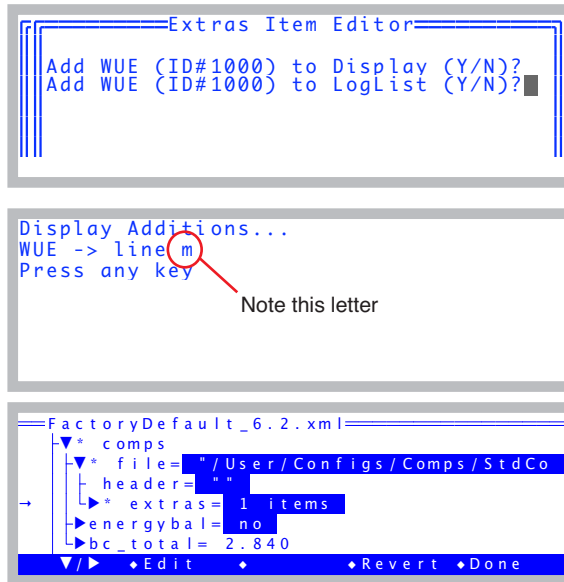
Figure 3-125. Leaving the Extras List Editor. If you have added items, you will be asked about adding them to the Display, LogList, and (if appropriate) PromptLists. You are then shown where (letter) the new display items will be.

10 **Try it out.**

Go to New Measurements mode, and press **m** (or whatever line that was indicated back in Step 9) to see *WUE (*Figure 3-126*)*.
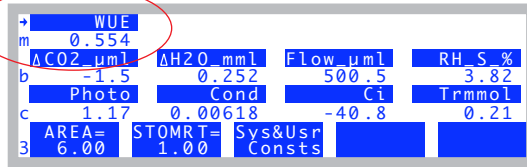
Figure 3-126. WUE in New measurements mode.

Saving Configuration Changes

You can save a configuration anytime by using the `Save as...` entry in the Config Menu.

Experiment #5 **Saving your work**

We will take a slow tour through what is normally a 1 or 2 step process, to point out some things that could be important to understand.

1 **Access the configuration save dialog**

It is `Save as...` in the Config Menu.
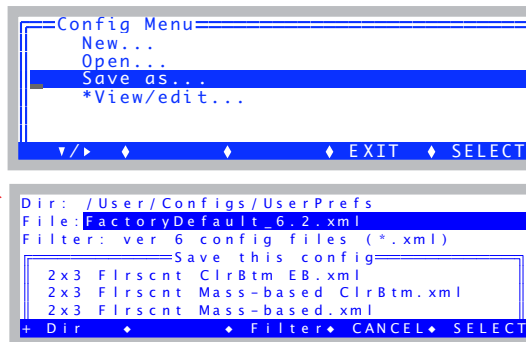
Figure 3-127. Config Menu|Save as.

2 **Note the directory: /User/Configs/UserPrefs**

Configuration files are all stored in the directory /User/Configs/UserPrefs. These are the files that are shown when you are asked to pick a configuration, when you first run OPEN, or when you change to a previously stored configuration.

3   **Note the filter: *.xml**
There are two types of files in /User/Configs/UserPrefs: those that end with
".xml", and those that don't. There is a **Filter** key (**f3**) that will toggle the filter
setting between "*" (show all files) and "*.xml" (show only .xml files). Try
it (Figure 3-128).

Filter = *.xml

```
Dir:  /User/Configs/UserPrefs
File: FactoryDefault_6.2.xml
Filter: ver 6 config files ( (*.xml)
═════════════════Save this config═════════
 2x3 Flrscnt ClrBtm EB.xml
 2x3 Flrscnt Mass-based ClrBtm.xml
 2x3 Flrscnt Mass-based.xml
+ Dir      ◆         ◆ Filter◆ CANCEL◆ SELECT
```

Filter = *

```
Dir:  /User/Configs/UserPrefs
File: FactoryDefault_6.2.xml
Filter: all files (*)
═════════════════Save this config═════════
 2x3 Flrscnt ClrBtm EB.xml
 2x3 Flrscnt Mass-based ClrBtm.xml
 2x3 Flrscnt Mass-based.xml
+ Dir      ◆         ◆ Filter◆ CANCEL◆ SELECT
```

*Figure 3-128. There are two filter settings: show all files, or just
those that end with .xml.*

*.xml files are version 6.1 or greater configuration files. If there are other files
in this directory, they are from earlier versions (like 6.0). They *are* compati-
ble. However, when you save a configuration, it will be a .xml file, regardless
of how the filter happens to be set (i.e., you cannot save a configuration in the
old format).

4   **Try overwriting FactoryDefault_6.2.xml**
Try saving the configuration over FactoryDefault_6.2.xml. It is a protected
file, so you will be prevented from overwriting it (Figure 3-129).

```
Dir:  /User/Configs/UserPrefs
F
F  This file is protected. Please
   modify the name and try again
┌
│
│                           ═════Press Enter═════
│  FactoryDefault_6.2.xml
+ Dir      ◆         ◆ Filter◆ CANCEL◆ SELECT
```
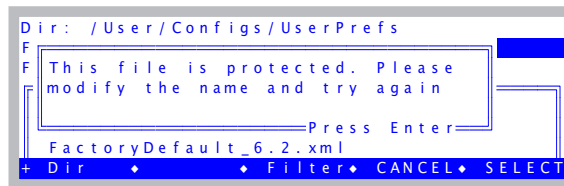
Figure 3-129. Trying to overwrite a write-protected file.

5 **Save it**

Make the name in the *File* box be what you want it to be, and press **enter** (Figure 3-130). It does not matter if you include the .xml or not. The file will get that extension regardless.

```
Dir: /User/Configs/UserPrefs
File:MyConfigExample.xml
Filter: all files (*)
═══════════Save this config═══════
 2x3 Flrscnt ClrBtm EB.xml
 2x3 Flrscnt Mass-based ClrBtm.xml
 2x3 Flrscnt Mass-based.xml
+ Dir     ♦         ♦ Filter♦ CANCEL♦ SELECT
```
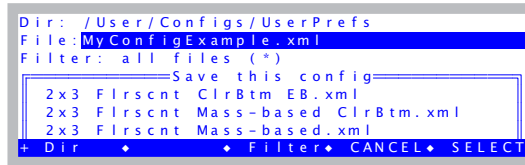
Figure 3-130. Save the configurtion under a new name.

The next time you are prompted to pick a configuration, the file you just created will be the default choice.

```
Dir: /User/Configs/UserPrefs
  ════════Select a new configuration════════
 MyConfigExample.xml
 OpaqueConifer EB.xml
 prompts.xml
 RGB on2x3.xml
 SoilChamber_6.2.xml
+ Dir     ♦         ♦         ♦ CANCEL♦ SELECT
```
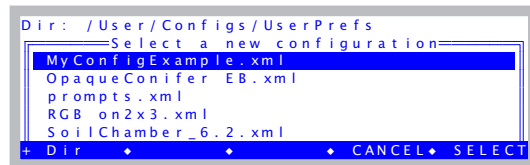
Figure 3-131. The default choice when opening a config-uration is the last one saved or opened.

Congratulations - you made it to the end of the introductory tours. Chapter 4 builds on what you have learned here, and introduces you to proper measurement technique.

Making Measurements

4

The fundamentals of good measurements

4

Making Measurements

The presentation in this chapter presumes that you have assembled the LI-6400, learned how to operate the software - especially the chamber control functions - and are ready to make measurements on plants.

Preparation Check Lists

We present a checklist of things that should be done prior to making measurements. They take about 5 minutes, but if you are careful to do this each session, it can save you a lot of time and frustration later. You may wish to copy and clip this checklist summary:

A. During Warm Up
1) Air Supply: Prepare CO_2 Mixer or Buffer Volume
2) Temperatures: Values OK? T_{leaf} responding?
3) Light Source, Sensors: Responding? Values OK?
4) Pressure Sensor: Value OK? Stable?
5) Leaf Fan: Running?
6) Flow Control: Max flow OK? Chemical tube restrictions?

B. After Warm-up
1) Check the flow zero
2) Adjust latch, close chamber
3) Check CO_2 zero
4) Check H_2O zero
5) Mixer Calibration (optional)
6) Lamp Calibration (optional)
7) Check T_{leaf} zero
8) Set Reference CO_2 and H_2O
9) Test for leaks
10) Match the IRGAs. Valve working?

C. Measuring the First Leaf
1) Set Light
2) Set Flow to 400 μmol s^{-1}
3) Set Reference CO_2
4) Temperature?
5) Clamp onto leaf
6) Set Area and Stomatal Ratio
7) Set constant humidity?

Figure 4-1. The checklists to prepare for making measurements.

During Warm Up

Once OPEN is loaded, and while the gas analyzers are warming up, you should do these steps.

1 **Air Supply - Cartridge or Buffer Volume?**
If you are going to be using the 6400-01 CO_2 mixer, install a cartridge now, so the system can begin pressurizing. Otherwise, prepare a buffer volume (see **Air Supply Considerations** on page 4-50).

2 **Check the Temperatures**
The three measured temperatures (block, air, and leaf) are together in display group *h*. Check to see that they read reasonable values, and are within a few degrees of each other.

Position the thermocouple properly, either just above the gasket (Figure 19-23 on page 19-25) for leaf measurement (normal), or pulled down for air temperature measurement (energy balance).

3 **Check the Light Source and Sensors**
Check to make sure that the instrument is configured for the light source that you are using. See **Specifying the Source and Sensor** on page 8-3.

The light sensors (*ParIn_µm* and *ParOut_µm*) are both in default display group *g*. See that they respond as expected when the light sensors are illuminated and darkened.

If you get negative *ParIn_µm* values, there is probably a mismatch between the real light source, and the one OPEN thinks it has. A trip to <open> <light> <source> in `Config Menu | View/edit` (page 8-4) will fix that.

4 **Check the Pressure Sensor**
The pressure measurement (*Prss_kPa*) is shown in display group *g*. See that it shows reasonable, stable values. (Typical values: 100 kPa near sea level, 97 kPa at 1000 ft., 83 kPa at 5000 ft., etc., but this varies with the weather.)

5 **Check the Leaf Fan**
Turn the leaf fan off and on (**f3** level 3), and listen for sound changes in the sensor head as the fan motor stops and starts. If you do not hear a sound when the fan should be on, it could mean a blown fuse (fan or flow board), a fan jammed with debris, or other problems (see Chapter 20). Leave the fan on when you are done.

6   Is Flow Control OK?

Use the flow control key (**f2** level 2) to fix the flow at 1000 μmol s^{-1}. Watch the *Flow_μms* (display group *b*) to determine the actual maximum flow. The value is typically in the 700's if a CO_2 mixer is installed, or higher if not.

Now test the chemical tubes for flow restrictions by changing each from full bypass to full scrub, and watching the effect on flow rate. Normally, scrubbing will drop the maximum flow by 5 or 10 μmol s^{-1} per tube. Larger drops may indicate that the air mufflers in the chemical tubes are getting clogged, or that a flow diversion tube is pinched shut. See **Pump/Flow Problems** on page 20-13 for more details.

Finally, set the flow to 500 μmol s^{-1}.

After Warm Up

After the IRGAs have been on for about 10 minutes[1], continue with the following steps:

1   Check the Flow zero

In New Measurements mode, monitor *Flow_μms* (display line *b*) and turn the pump off (**2 f2 N**) and the chamber fan off (**3 f3 0** for <u>o</u>ff)[2]. The flow should drop to within 1 or 2 μmol s^{-1} of zero. If it doesn't, re-zero the flow meter (**Zeroing the Flow meter** on page 18-23). Turn the fan back on when done.

2   Adjust the latch, and close the chamber

1) Adjust the latch so that the chamber lips are slightly apart when the chamber is closed. 2) With the chamber closed, close the adjustment knob until it starts to become snug. 3) Open the chamber, and turn the knob one or two more half turns. Now the chamber is adjusted properly for sealing when empty, or with thin leaves. Close the chamber for the next two steps.

3   Check the CO_2 IRGA zero

In New Measurements mode, with the mixer off (**2 f3 N**), and the flow set to 500 μmol s^{-1} (**f2 F 500 enter**), monitor CO_2 reference and sample (display line *a*). Turn the soda lime on full scrub, and the desiccant on full bypass. The

---

[1] Or longer, if the system has just been moved from one temperature to another.
[2] Why turn off the chamber fan? If it's on when the pump is off, it will actually push a bit of air (1 or 2 μmol s^{-1}) back through the flow meter, throwing off your zero reading.

reference should quickly approach zero, while the sample will approach zero a bit more slowly. If they are within 5 μmol mol^{-1} of zero, it will be adequate.

4   Check the H2O IRGA zero
Turn the desiccant to full scrub, and watch sample and reference H_2O. The reference will again approach zero faster than the sample does. It will zero more slowly than the CO_2 IRGA did, however, because of water sorption. Rather than wait the 10 or 20 minutes to get a really good zero, use your judgement. If after a minute or so, the reference is down to 0.2 or 0.3 mmol mol^{-1} and falling slowly, that's good enough. The sample will be higher than that. Clearly, if it's negative and falling after only 1 minute, it will be going too low, and re-zeroing may be in order.

If the CO_2 or H_2O IRGAs need zeroing, refer to **Setting the CO2 and H2O Zero** on page 18-11. The important thing is that the reference IRGAs are reasonably well zeroed (let's say ±5 μmol mol^{-1} CO_2, ±0.5 mmol mol^{-1} H_2O). The first time you match (Step 10, coming up), the sample IRGAs will be taken care of, as they are adjusted to match the reference IRGAs.

Important Note about CO$_2$ and H$_2$O Zeros:
If your chemicals are not fresh, then you will do more harm than good by setting the zeros with them.

The IRGA zeros are quite stable, especially in the absence of big temperature changes. Therefore, the exercise of checking zeros each day is really a diagnostic. If the indicated concentration doesn't change when it should (that is, if it doesn't drop when you start scrubbing), then something is wrong, and it's good to find that out early.

---

5   Mixer Calibration
If you are using the 6400-01 CO_2 Mixer, run the routine found in `Calib Menu | CO2 Mixer | Calibrate`, described in **6400-01 CO2 Mixer** on page 18-25. The chamber can be open for this. Make sure that the soda lime is on full scrub.

6   Lamp Calibration
If you are using the 6400-02 or -02B LED Source, or the 6400-40 LCF, run its calibration (`Calib Menu|LED Source|Calibrate` described on page 18-30 , or `Calib Menu | LCF Source | Calibrate described` on page 27-75). You will do the best calibration by having the chamber closed onto a

representative (with respect to its reflectance) leaf. This isn't critical however, but the chamber should at least be closed.

7   Check T$_{leaf}$ zero
Unplug the leaf temperature thermocouple connector (it's purple colored), and compare the leaf and block temperatures. If they differ by more than 0.1°, then adjust the leaf temperature zero (see **Zeroing the Leaf Temperature Thermocouple** on page 18-24).

Finally, reconnect the thermocouple, open the chamber, and verify that "Tleaf_°C" responds when the thermocouple is warmed by touching it.

8   Set Desired Reference Values for CO$_2$ and H$_2$O
If you are using the CO$_2$ mixer, set it to control on reference concentration with a target of 400 μmol mol^{-1}. Make sure that the soda lime is on full scrub. If you are not using the CO$_2$ mixer, monitor the reference CO$_2$ concentration. Is *CO2R_uml* sufficiently stable? (Over a 30 s period, it should change less than 2 μmol mol^{-1}.) If not, use a larger buffer volume.

For H$_2$O, set the desiccant at mid-range (between scrub and bypass) for now.

9   Leaks?
Set the flow rate to 200 μmol s^{-1}. With the chamber closed and empty, exhale around the chamber gaskets, and look for any fluctuations in the sample cell CO$_2$ concentration (*CO2S_µml*, display group *a*). If there are no leaks, the *CO2S_µml* value should not increase by more than 1 μmol mol^{-1}.

10   Match the IRGAs
Matching the IRGAs is easily accomplished whether the chamber is empty or not, but it's a good policy to do this once right before starting a measurement. Refer to **Matching the Analyzers** on page 4-33 for how to do this.

Verify that the match valve is in fact working. Figure 4-2 on page 4-34 shows what to look for.

You are now ready to clamp onto a leaf and begin measurements.

Clamping Onto the First Leaf
Once you've got the system behaving well with no leaf in the chamber, you are ready to start. The basic procedure is quite simple: set the conditions in the chamber, insert the leaf, adjust the conditions if necessary, then wait for stability.

1   Light
If using the LED source, set the light to the desired value (ambient is a good value to start with - it won't be an abrupt change for the leaf). If you aren't using the LED source, then orient the chamber so that no shading of the leaf by the chamber walls will occur once you've installed the leaf.

2   Flow

Set the control for Fixed flow, about 400 μmol s^{-1}, and desiccant mid-range between scrub and bypass. We'll come back to this in Step 9.

3   CO$_2$
If you are using the CO_2 mixer, set it to control reference CO_2 with a target slightly above ambient (say, 400 μmol mol^{-1}). If you aren't using the CO_2 mixer, but using a buffer volume instead, set the soda lime scrub knob to give you the concentration you want. Usually, that means full bypass.

4   Temperature
(Optional) If you are going to be in direct sun, you will probably want to use the coolers to control the temperature. Check the temperatures to see their present values, then set the control accordingly.

5   Insert leaf
Check the latch adjustment for a good seal. Snug is fine; be careful it's not too tight, however. If you aren't using the LED source, be careful with the chamber's orientation; avoid shading part of the leaf with the walls of the chamber.

6   Set Stability (optional)
Use **Define_Stablty** (f4 level 5) to setup the stability criteria you wish to use (**Defining Stability** on page 6-29).

7   Log Options, Log Button (optional)
This is a good time to set your log options (**Log Options**, **f3** level 5) and your log button behavior (**f5** level 5), if you wish. See **User Definable Log Button** on page 9-6 and **Log Options** on page 9-14.

8   Set Area and Stomatal Ratio
In New Measurements mode, press **3**, and set the leaf area and stomatal ratio for this leaf. Leaf area is simply the area exposed inside the chamber. If you are using a 2x3 chamber and filling it, the area is 6 cm^2. Stomatal ratio is an estimate of the ratio of stomata on one side of the leaf to the other. Use 1 for equal stomatal density on top and bottom; 0 for stomata on only one side. If you aren't sure, use 0.5. It doesn't matter if you use the ratio of top to bottom, or bottom to top. Thus, 0.5 is the same as 2; 0.333 is the same as 3, etc.

9 Revisit the flow control
Decide how you will operate: control flow to maintain constant humidity, or use a constant flow. (If you aren't sure what to do here, you probably skipped over **Tour #3: Controlling Chamber Conditions** on page 3-28. There may still be hope for you, however; work through **Humidity Control Experiments** on page 4-9.)

From this point on, what you do is going to depend on your experiment, or what it is you wish to accomplish. For example, you might wish to measure a response curve (light, for example, is discussed on page 4-24), or make survey measurements (page 4-21) by going from leaf to leaf and only taking a minute or so for each measurement.

If you are new to gas exchange measurements on plants, continue on with the next section (**Some Simple Experiments**). It will take you through some principles that should help you make valid measurements.

Some Simple Experiments

If you have not had much experience making gas exchange measurements, you may wish to work through some of the experiments in this section. To do them, first establish a leaf in the chamber:

■ **Do this first**

1 Select a plant and leaf to measure
The preferred plant material is an adequately watered plant that is growing in full or partial sun. By contrast, measurements will be more difficult if done on a dry, neglected house plant that has only seen dim (for the plant) fluorescent lights its whole life.

2 Do Steps 1 through 6 in the previous section
Set the controls (light, flow, CO_2, temperature), area, and stomatal ratio.

3 Observe the CO_2 concentrations
Note *CO2S_μml*. Is it below *CO2R_μml*? If so, that's good, because it means there is more photosynthesis than respiration. (Net photosynthetic rate will be on display line *c*, under *Photo*.) *CO2S_μml* should stabilize (within 0.2 or 0.3 μmol mol^{-1}) after 30 seconds or so of clamping onto the leaf. If it's not stable, check the stability of *CO2R_μml*. Perhaps the mixer hasn't stabilized yet, or you need a buffer volume. Consult **Unstable Photosynthetic Rates** on page 20-9 if you need help fixing this instability. If *CO2S_μml* is not below *CO2R_μml*, then perhaps you need to match (page 4-33).

4   Observe the humidity values

The *RH_S_%* value on display line *b* is the relative humidity in the sample IRGA. It's calculated from the water IRGA signal, which is *H2OS_mml*. Other things being equal, you want to make gas exchange measurements in as high a humidity as possible, without letting the flow rate (which is determining that humidity) go too low. 200 or 300 μmol s^{-1} is a reasonably low flow range; but you can drop to 100 if you need to. (Leaks are a bigger problem at low flow rates - see **Leaks** on page 4-44.)

You are now ready to do some or all of the following elementary experiments.

Humidity Control Experiments

Step 9 on page 4-8 (**Revisit the flow control**) asks you to decide how you wish to operate while making measurements: either a fixed flow rate (with a potentially variable humidity), or constant humidity (with a potentially variable flow rate). The following experiment will acquaint you with the capabilities and trade-offs involved.

Experiment #1   Finding the Humidity Limits

If you are using the CO_2 mixer, set it to control the reference concentration at a value slightly above ambient, such as 400 μmol mol^{-1} if you're outdoors.

1   Operate in fixed flow mode

With the desiccant midway between scrub and bypass, operate in a fixed flow mode at 400 μmol s^{-1}.

2   Match the IRGAs

When *CO2S_μml* and *H2OS_mml* become stable, match the IRGAs.

3   Note the conditions

After matching, note the values that pertain to photosynthesis (*CO2R_μml*, *CO2S_μml*, *ΔCO2*, and *Photo*) and the values that pertain to conductance (*H2OR_mml*, *H2OS_mml*, *RH_S_%*, and *Cond*).

4   Find the upper humidity limit

Set the desiccant on full bypass, and the flow rate to 100 μmol s^{-1}. Wait about one minute, then observe the water numbers. The value of *H2OS_mml* will be about as high as you'll be able to achieve with this leaf at this stomatal conductance.

Question #1: How can the *RH_S_%* value (as opposed to *H2OS_mml*) be further raised and lowered? (Answer on page 4-56.)

Note that we just dropped the flow rate F by a factor of 4 for this step. Since $A = (\Delta CO2)/F$ and $E = (\Delta H2O)/F$ (the complete equations for photosynthesis A and transpiration E are in Chapter 1) it might lead you to believe that $\Delta CO2$ and $\Delta H2O$ should each have also increased by a factor of 4.

Question #2: Did you observe a 4-fold increase in $\Delta CO2$? How about $\Delta H2O$? If you didn't, why not? (Answer on page 4-56.)

5 Find the lower humidity limit

Now set the desiccant on full scrub, and increase the flow to 800 μmol s^{-1} (it probably won't achieve that value). Give it a minute or so to stabilize, and observe the new set of values. This *H2OS_mml* value represents your lower humidity limit for this leaf.

How is stomatal conductance (*Cond*) behaving: steady, dropping, or increasing?

Question #3: We just lowered the humidity in the chamber. If there are water sorption effects on the chamber walls, will they make stomatal conductance too high or too low? (Answer on page 4-56.)

Question #4: How might you differentiate real stomatal changes from water sorption effects? (Answer on page 4-56.)

6 Return to starting Conditions

Return the flow to 400 μmol s^{-1}, and the desiccant to mid-range between scrub and bypass.

Points to Remember

• Changing the flow rate affects both CO_2 and H_2O concentration in the chamber.

• Chamber humidity control is via flow rate. The highest humidity is achieved by low flow and bypassing the desiccant. The lowest humidity is achieved by high flow and full scrub.

Experiment #2   **Maintaining a Constant Humidity**
Often it is desired to make measurements or to conduct an experiment at a consistent chamber humidity. This experiment lets you see the automatic humidity control in action.

1   **Pick a humidity target**
Start out in fixed flow mode at 400 μmol s^{-1}, with the desiccant adjustment knob midway between scrub and bypass. When *H2OS_mml* is stable, change to constant humidity control (use the H option: constant H_2O mole fraction), and target that current value of *H2OS_mml*. Your flow rate should settle into the 300 or 400 μmol s^{-1} range. (If flow jumps to either extreme, and messages appear about targets being too dry or too wet, make sure you selected the H option and entered a reasonable value, in mmol mol^{-1}.)

Note the *CO2S_μml* value.

2   **Dry the incoming air**
Once the chamber humidity is on target, and the flow rate is stable, turn the desiccant knob to full scrub. Observe what happens to reference humidity, flow rate, and sample humidity (*H2OR_mml*, *Flow_μml*, and *H2OS_mml*). Reference humidity should go to zero, flow will decrease, and H2OS_mml will remain unchanged. Also, keep an eye on how quickly or slowly *CO2S_mml* gets to its new value.

Question #5: What does it mean if the reference humidity (*H2OR_mml*) doesn't get within 0.5 mmol mol^{-1} of zero? (Answer on page 4-56.)

Question #6: Will the sample cell CO_2 (*CO2S_μml*) increase or decrease during this step? Why? (Answer on page 4-56.)

3   **Moisten the incoming air**
Now turn the desiccant knob to full bypass, and watch the flow increase, reference humidity increase, and the sample cell water mole fraction remain unchanged. The sample cell CO_2 will go the *other* direction from how it changed in the previous step (not to give the answer to Question #6 away…).

4   **Change to constant RH**
Return the desiccant knob to midway, and after the flow rate stabilizes, note the value of *RH_S_%*. Now change to constant RH control, targeting that value of *RH_S_%*.

5   Turn on the coolers
Turn on the temperature controllers (**f4** level 2), by setting the block temperature for a target about 5°C below its present value (display line *h*). But, before you do this, try another question:

Question #7: What will cooling the chamber do to flow rate (owing to the constant RH control mode) and why? (Answer on page 4-56.)

6   Watch RH and flow
As the air temperature drops in the chamber, watch the flow compensate for the changing RH. Notice that the control is not quite as tight as when we were controlling on a constant mole fraction; RH will drift off target a little bit as the temperature changes. (Why? Read about RH control on page 7-11.)

7   Change to constant VPD
Note the current value of *VpdA* (display line *d*), the vapor pressure deficit based on air temperature. Then change to controlling to a constant VPD, based on *Tair*, and target that value. (See the VPD discussion on page 7-11.)

8   Change the temperature control to ambient + 5°C
Before you do, test your understanding with this question:

Question #8: How will flow rate respond as the temperature increases, since we're holding a constant vapor pressure deficit? (Answer on page 4-56.)

Now try it, and see who is correct. Wait 2 or 3 minutes for things to stabilize.

9   Watch it warm
You might notice that warming is more efficient than cooling. You may also notice that the VPD gets away from the target a little bit as the temperature changes.

When the block temperature achieves its target, see that the VPD settles back to its target. Then set the target temperature to ambient, and bring the chamber back to normal.

You can turn the temperature control off if you like.

Points to Remember

• Constant humidity mode will compensate for changes in incoming air stream, or leaf transpiration changes.

• Controlling to a constant mole fraction is the tighter control, while controlling to a constant RH or VPD can have small lags in the face of rapidly changing temperature (see the discussion about the various humidity control options on page 7-11 for more details).

Controlling CO_2

The next two experiments are best done with a 6400-01 CO_2 mixer. If you don't have one installed, you can still do approximate CO_2 control between ambient and zero by adjusting the soda lime tube flow adjust knob.

Experiment #3 **CO_2 and Humidity Control Interactions**

Start with the conditions described by **Do this first** on page 4-8. Make sure the desiccant knob is mid-range.

1 **Set flow control for constant water mole fraction**

Target the current value of *H2OS_mml* (Step 1 on page 4-11).

2 **Change to constant sample cell CO_2**

If you have a CO_2 mixer, switch over to controlling a constant sample cell concentration. Target the present value of *CO2S_µml*. Wait for *CO2S_uml* to stabilize.

3 **Turn the desiccant knob to full scrub**

Watch *CO2R_µml* and *CO2S_µml*. The latter is supposed to be held constant. *CO2S_µml* will drift well off target, then come back to where it was as *CO2R_µml* adjusts.

Question #9: Will *CO2R_µml* increase or decrease? (Answer on page 4-57.)

Note: Keep in mind that this test of abruptly changing the incoming humidity while trying to control both chamber humidity and CO_2 is an artificial worst case. Typically, the flow control system balances stomatal changes, which happen less rapidly, so the sample cell CO_2 control option isn't faced with large swings in flow rate. The next two steps will illustrate a more typical sequence of events.

4   Return the desiccant knob to mid-range
Watch the sequence reverse itself. Also note the order in which things happen: Once *Flow_µml* stabilizes, then *CO2S_µml* can stabilize.

5   Shade the leaf
If you are using a light source, cut the light value in half. If you are not, just shade the leaf with your hand. Before you do, however, here's another learning opportunity:

Question #10:   What do you expect to happen to photosynthesis (*Photo*), stomatal conductance (*Cond*), and intercellular CO_2 (*Ci*) when you cut the light in half? How do you expect the control systems to compensate: specifically, how will flow rate change, and how will reference CO_2 change? (Answer on page 4-57.)

6   Watch the response
Photosynthesis will immediately start to drop, and (if you wait 10 or 15 minutes), conductance will eventually decrease as well. Some species can react faster than this, however.

7   Restore the light
Return the leaf to its original light value and watch the control system respond to the changes.

Points to Remember
• Constant humidity control interacts with sample cell CO_2 control. Abrupt (artificial) changes can be problematic, but when tracking leaf changes, the control system should be able to handle it. Be patient.

Experiment #4 **A Manual CO$_2$ Response Curve**

CO$_2$ response curves are described in detail later (page 4-29), including how to generate them automatically. This experiment will give you a step-by-step guide to manually generating one. If you don't have a CO$_2$ mixer, don't despair[3], you can still do the experiment.

As always, we start with the conditions that **Do this first** on page 4-8 describes.

1 **Set the controls**
Flow: Constant mole fraction, target the current value (Step 1 on page 4-11).
CO$_2$: If you have a CO$_2$ mixer, set it to control reference CO$_2$ to a bit above ambient, such as 400 μmol mol^{-1}. If you don't, set the soda lime knob on full bypass.
Temperature Control: Constant leaf temperature, and target the current value.
Light: Use 1000 μmol m^{-2} s^{-1}. (If you don't have a light source, do the experiments in a growth chamber, or outdoors. But beware: this experiment is meaningless without steady light).

2 **Open Log File**
Name it "Sample CO2 curve", or whatever you like. (**f1** level 1)

3 **Wait for stability, and log the first point.**
When the CO$_2$ and humidity controls are stable and on target, log the starting datum (**f1** level 1)

4 **Next CO$_2$ value**

If you are using the CO$_2$ mixer, lower the reference target by 100 μmol mol^{-1}. If you aren't, turn the soda lime knob a bit toward scrub, so that the reference CO$_2$ drops to more or less what you want. Use these targets for reference concentration: 400, 300, 200, 100, and 30 (C$_3$) or 0 (C$_4$). The last point is designed to be below the compensation point.

Question #11: Notice when you change the CO$_2$ concentration (whether you did it with the mixer, or the soda lime tube) the indicated photosynthetic rate (*Photo*) becomes quite erratic. Why? (Answer on page 4-57.)

Question #12: If you are controlling CO$_2$ by varying the soda lime scrub knob, under what circumstances might you expect changes in this knob set-

---

[3]·Don't despair. Just buy one. It's worth it.

ting to affect the flow rate through the chamber? (Hint: it's a humidity control question. Answer on page 4-57.)

5   Wait for stability, then match and log
Wait for a minute or so for the photosynthetic rate to stabilize, match the IRGAs, then log another record (**f1** level 1).

6   Repeat until done
Repeat Steps 4 and 5 until you are done. Try to get about 4 or 5 points between your starting value and ending points. Go down to 30 μmol mol^{-1} or so for C_3 plants, and 0 for C_4's. (Hint: If you are using the CO_2 mixer, you get 0 μmol mol^{-1} by turning the mixer off.)

At any point along the way, you can view a graph of your logged data by doing Step 8.

7   Finish the curve back at the starting point
Repeat the starting point. See how long it takes for the photosynthetic rate to return to normal. (Hint: don't spend too long at lowest CO_2 value.)

If you have a CO_2 controller, do some points above ambient, such as 600, 800, and 1000 μmol mol^{-1}. (If global climate change is keeping you funded, go right on up to 2000.)

8   View the Graph
You can view your curve with GraphIt (press **View File** (**f2** level 1 in New Measurements mode). If the axes are not defined for an A-Ci, press **QuikPik Config** (**f1**) and select "A_Ci Curve". Press **REPLOT GRAPH** (**f2**) and draw it. Hint: If your low CO_2 point had negative photosynthesis (respiration), you may want to change the default A-Ci plot to automatically scale the axis minimum for photosynthesis. Otherwise, it won't show that point.

9 **Analyze the data**

Use GraphIt to generate plots to answer these questions: What's the CO_2 compensation point? Did humidity stay constant over the experiment? How much did the stomata change over the measurement?

> **Points to Remember**
> • Changes in CO_2 target value are accompanied by a brief disruption in the system's stability.
>
> • Plot of logged data can be examined during a measurement with GraphIt.

Light Experiment

Photosynthesis is first and foremost driven by light, so a natural experiment is to measure this relationship.

Experiment #5   **Sun and Shade Dynamics**

For this experiment, select a fully sunlit leaf. An LED source is not required for this experiment; we'll be changing back and forth between sunlit and shaded conditions, so you can simply use your hand to block the sun (or other light source) from the leaf when you need low light. It's low tech, but effective.

We start once again with the conditions of **Do this first** on page 4-8.

1 **Set the controls**

Flow: Constant mole fraction, target the current value. (Step 1 on page 4-11).
CO_2: If you have a CO_2 mixer, set it to control reference CO_2 to a little above ambient, such as 400 μmol mol^{-1}. If you don't, set the soda lime knob on full bypass.
Temperature Control: Constant leaf temperature, targeting the current value.
Light: If you have an LED source, set it to match full sun, or whatever the ambient light on the leaf is.

2 **Clamp onto the leaf**

3   **Use Real Time Graphics**

Set up strip charts for viewing photosynthesis, conductance, and C_i. The default configuration has the first two already defined, so you can add Ci to that screen, or put it on another one.

4   **Simulate brief shade (fast cloud)**

Activate the strip charts. When there are reasonably flat lines displayed (indicating stability), try decreasing the light by 80% (from 1500 μmol m^{-2} s^{-1}) down to 300, for example) for 20 or 30 seconds, then returning it to its original state. (If you aren't using the light source, do this by shading the leaf with your hand. If you are using the light source, do this by **escape** (to stop viewing the graph), then **2 f5** _<low value>_ **enter**, wait 15 seconds, then **f5** _<high value>_ **enter**, then view the graph again by **4 f3**.)

Question #13: How would you expect _Photo_, _Cond_, and _Ci_ to react to this brief drop in light? (Answer on page 4-57.)

View the strip chart to see what really happened.

5   **Simulate longer shade (slow cloud)**

Now try decreasing the light by 80% for 2 minutes, then returning it to its starting value. Was this long enough to get the stomata to start to respond? (If you are patient, you might find out how long it takes for the stomata to _stop_ responding when the light drops. That is, how long before they stabilize in the new conditions. It might be 10 to 15 minutes, or longer.)

Question #14: Why does stomatal conductance decrease when light is reduced, and what determines the degree of stomatal closure? (Answer on page 4-57.)

6   **Change to sample cell CO$_2$ control**

Change from controlling on reference CO$_2$, to controlling on the sample cell CO$_2$. Target the current value of _CO2S_μml_. Repeat Steps 4 and 5. How does the photosynthesis response differ from the first time you tried it?

Question #15: Suppose you want to do some sun / shade dynamics measurements, and you a) want the sample cell CO$_2$ concentration to be as consistent as possible, and b) don't want the slower time response of the sample cell CO$_2$ control algorithm to interfere with your measurements. How could you do it? (Answer on page 4-57.)

7   Change to a shade adapted leaf

Change to a leaf that has been in the shade for some time. If you are using an LED source, don't forget to adjust the light to a low value, before putting the leaf in the chamber. Change to CO_2 control back to a constant reference concentration.

8   Provide a brief sunfleck

Give the leaf full sun for 30 or 40 seconds, and observe the response of *Photo*, *Cond*, and *Ci*.

9   Provide a long sunfleck

Now give it full sun, and see how long (if ever) it takes for *Photo* and *Cond* to reach the values that you found for the sunlit leaf.

Points to Remember

• Light changes produce immediate photosynthetic rate changes. These changes can be compensated by controlling sample cell CO_2, but some adjustment time is necessary, typically 1 minute or less.

• Light changes will cause stomatal changes, but only after many minutes. These changes are continuously compensated when using constant humidity control.

Equilibrium is reached faster by decreasing light on a sun-adapted leaf, than by increasing light on a shade-adapted leaf.

Experiment #6   Sun And Shade Leaf Survey

This experiment uses the LI-6400 in a survey mode in which a succession of leaves is measured, and each measurement lasts a minute or less.

Should you use the LED light source for this experiment? If you have this choice, here are some things to consider. This experiment will measure sun and shade leaves that are adapted to their radiative environment. If you don't use the light source, you won't be affecting that environment very much when you clamp onto the leaf with the clear chamber top. If you are using the light source, you'll have to be sure and set the light to match this ambient value before clamping onto each leaf. If you have an external quantum sensor and a light source, you can use the Tracking mode in the New Measurements mode light control screen (**f5** level 2), and have the source track ambient (if it is reasonably stable, of course) as measured by the quantum sensor.

Prepare the system: Use fixed flow at about 400 μmol s^{-1}, and control reference CO_2 to 400 μmol mol^{-1}. Match the IRGAs after stability is reached.

1 Open a log file
If you want, you can record your work. Open a log file, and name it "Survey Experiment", or whatever you'd like.

2 Measure 5 sunlit leaves
Clamp onto another leaf, and wait for the photosynthesis and conductance values to stabilize. One minute is usually sufficient. Then press **LOG** (**f1** level 1), or else press the button on the chamber handle for about 1 second. After logging, move on to the next leaf.

If you aren't using a light source, be careful about shading these sunlit leaves. If a leaf is tipped away from the sun prior to measurement, the chamber walls will cast a shadow on the leaf when you place it in the chamber. Changing the orientation to avoid this shading will cause other problems since you've suddenly increased the light. For this experiment, it's best to choose sunlit leaves that are directly facing the sun.

If your leaves aren't filling the chamber aperture, be sure that the entered value of leaf area (**f1** level 3) matches the actual one for each leaf.

3 Measure 5 shaded leaves
Now measure 5 leaves that have been well shaded for some time. If you are using a light source, remember to lower its value to match the typical shade leaf's environment.

4 Plot the results
Enter GraphIt (**f2** level 1) to view your data file so far. Try plotting it using the "Light Curve" configuration. When you are done, exit GraphIt and close the log file.

Points to Remember
• Measurements can be made fairly quickly provided the chamber conditions are not too different from ambient.

• Don't let the chamber walls cast shadows on the leaf.

Where to Go From Here

This section has introduced you to survey. light response, and CO_2 response measurements that you can do with the LI-6400. The next sections describe these measurements in more detail, providing physiological and operational considerations to help guide you as you determine measurement protocols for your experiments. The remaining sections in this chapter describe some operational hints and considerations with which you should be familiar.

Making Survey Measurements

The goal of survey measurements is usually to characterize a community, which means measuring a lot of leaves in a short period of time. This means spending a minimum amount of time on any one leaf in order to maximize the sample size.

Operational Considerations

If the ultimate goal is to be able to say something about a community, or at least about a number of plants, it follows then that each leaf needs to be measured in similar conditions. The conditions in the chamber should be as close as possible to what the leaf was experiencing prior to the measurement. This provides a time savings as well; you will only be waiting for the leaf chamber to equilibrate (flush out), rather than waiting for the leaf to equilibrate.

Light

Light is the most important variable, so be careful how it changes before and during the measurement. Avoid shading the leaf as much as possible as you move it into the chamber. During the measurement, keep the chamber orientation constant. Be cognizant of the recent light history of the leaf. If you are measuring sunlit leaves, don't select one that happens to be in a small sunfleck, or one that just became sunlit when you moved some stems out of the way. When you put a leaf into a clear-top chamber, the light incident on the leaf will be reduced by about 10%. Photosynthesis may respond fairly quickly to that reduction, and should equilibrate in a few seconds. Stomatal responses take longer, but a 10% light reduction will usually not cause a measurable change in conductance.

Avoid large changes of light. A common error is to reorient the chamber during a measurement. Whether you do it inadvertently (busy watching the display) or intentionally (avoiding shade), it's bad.

For outdoor survey measurements, clear days are a blessing, but partly cloudy days are a curse. With only short periods of uninterrupted sun, the leaves will

be in perpetual disequilibrium. Snapshots of photosynthesis and conductance taken against that sort of backdrop will be nearly impossible to interpret, and therefore meaningless. Use of a light source will guard against the odd cloud shadow interrupting a measurement on a nearly clear day. With more abundant clouds, the most a light source can offer is the chance to let each leaf equilibrate for 10 or 15 minutes in constant light, and that makes for very slow survey work.

CO_2

Since photosynthesis is a function of CO_2, it is important to have the chamber CO_2 concentrations as consistent as possible.

If you do not have a CO_2 mixer, you'll have to carry along a buffer volume to dampen the potentially huge fluctuations in CO_2 that will occur should you choose to breathe while you work. Buffer volumes are discussed in **Air Supply Considerations** on page 4-50. If you use a long tube[4] and a pole to draw the "clean" air from well above your head, you will still need a buffer volume, albeit a smaller one might suffice. Whatever you use, experiment until you have fairly steady reference CO_2 concentrations.

Life is much easier with a CO_2 mixer. You only need to decide whether to control reference or sample CO_2. If measurement speed is important, then by all means control on reference. If you want near-ambient values in the sample cell, set the reference for the right amount above ambient. Try a leaf or two until you get it right. On the other hand, if you can afford 2 or 3 minutes per measurement and want consistent sample CO_2 concentrations, try the S option (**f3** level 2).

Flow / Humidity

Use a fixed flow rate, medium or high, with little or no desiccant scrubbing. Here's the rationale: Fixed flow rate minimizes the time for system equilibration, once a leaf is installed. Minimal scrubbing along with a high flow rate means the chamber humidity will be reasonably close to ambient.

There are interactions with CO_2 control. If you are using the CO_2 mixer, the soda lime must be on full scrub, and that usually means the incoming humidity will be below ambient, even with the desiccant on full bypass. You can moisten the soda lime (see **Humidifying Incoming Air** on page 4-52) and/or reduce the flow rate a bit to offset this.

---

[4]Make sure its inside diameter is *larger* than 1/8 inch, to avoid pressure drops and reduced pump performance.

Temperature

There are two schools of thought about temperature control and survey measurements: one is that you shouldn't use the coolers so that your battery life is maximized. The other says you should use the coolers to maintain ambient temperature, so that the chamber doesn't get hot from being in the sun. You decide.

Matching the IRGAs

Match once on the first leaf (or a "trial leaf"), and perhaps every 30 minutes or so after that, especially if the temperature is changing.

When to Log?

Stability considerations are important here, because you want to take data quickly, but not before it's ready. You could monitor the stability indicators (**Stability Indicators** on page 4-41). You may want to shorten their time period to about 10 seconds.

Logging Considerations

You will need to decide some other logging issues, besides when to do it:

- **Leaf Area?**
  Is it changing from leaf to leaf? How and when will it be measured? Do you wish to be prompted for leaf area as you log data?

- **Extra Data?**
  Are there extra data you wish recorded, such as numbers or remarks the operator is to enter, to help identify the data later?

- **How Many Log Files? Log Options?**
  Are all the measurements destined for one file, or should there be several? If several, what's the rationale for the grouping? Does it matter in which order the measurements are done?

- **Use Stability Checking?**
  Do you want to guess at when to log, or use some objective criterion?

- **Log Button Behavior**
  Are you going to use it? Do you want it to generate prompts?

The simplest approach is log all the data into one file. If for some reason you desire multiple files, then make your measurements so that File1 is finished before File2 is started. (Appending data to an existing file adds a new header as well, so it's not a very efficient use of disk space.)

Judicious use of prompts and remarks (see **Prompts and Remarks** on page 9-20) can make the single file approach very workable, since you can go back later with your spreadsheet program and extract or sort the data records using these. Also, if leaf area and/or stomatal ratio is changing from leaf to leaf, they can be automatically prompted as you log each observation.

Light Response Curves

Starting from total darkness, in which there can be no photosynthesis, the first few photons to be absorbed by the leaf will be used with greatest efficiency. As light increases, the efficiency drops, and eventually subsequent increases in light yield little or no increase in photosynthesis. Thus, a light response curve can provide measures of dark respiration rate, the light compensation point (absorbed quantum flux for which photosynthesis and respiration are balanced), the quantum efficiency (initial slope), and the maximum photosynthetic rate. Shade adapted species tend to have lower dark respiration rates, lower compensation points, and lower maximum photosynthetic rates than sun adapted leaves. Quantum efficiency tends to be conservative, however.

Light Curve Strategies
Depending upon what you are trying to measure, there are a couple of approaches to light curves.

Rapid
Since the photosynthetic apparatus responds almost immediately to light, especially drops in light, the quickest method is to start with a leaf equilibrated to high light, and decrease the light, spending perhaps 1 or 2 minutes at each light value, and dropping in steps of 200 μmol mol^{-1} or less. When you do this, you'll find that the stomata have not had time to adjust, and tend to be more open at the low light values than they normally would. This manifests itself as a steadily rising C_i throughout the measurement. There's nothing wrong with this, but be careful how you use the conductance measurements from a rapid light curve, because they are not equilibrated values.

Slow
Another approach is to do a slow curve, giving the stomata time to equilibrate at each light level. Going slowly, you can work from dark to light, or light to dark. (If you are using a red only light source, however, beware; the stomatal behavior will be artificial. Our comparisons of the red+blue LED source and sunlight show them to have the same influence for opening stomata, however.) If you wait 15 or 20 minutes at each light level, you will find that C_i will

be fairly constant throughout the measurement, indicating that the stomata are fully adjusted. In fact, you could use C_i as an indicator of when to log the next record at all but the darkest light levels.

Survey

A third approach is to generate a light curve using multiple leaves that are equilibrated at a range of light values. Experiment #6 on page 4-19 uses this approach. This has the advantage of being fairly quick, yet has equilibrated values. The potential for difficulty comes from using multiple leaves, thus bringing age differences and other factors into the response curve. The survey approach is better suited for some species than others. In deciduous trees, for example, leaf age is not particularly related to position in the canopy. With this approach you can achieve a range of light levels by selecting leaves that are tilted with respect to the sun, and in varying degrees of shade. The orientation of the sunlit leaves is a problem, however, unless you are using a light source when you clamp onto them. With a clear chamber top, leaves that are tilted with respect to the sun will be shaded by the chamber wall, and this is to be avoided at all costs. If, however, you use a light source, you can set the appropriate value first, or have it automatically track the ambient light as measured by an external PAR sensor.

Sunfleck / Shade Method

The fourth approach offered here is to separate each new light level with the starting light value, with time to equilibrate. That is, use a sequence such as: 1800, 1000, 1800, 500, 1800, 300, 1800 μmol m^{-2} s^{-1}. (The starting point needn't be high; you could work the other direction with shade leaves.) Data collected in this manner might be most appropriate for addressing questions of light dynamics in canopies.

Operational Considerations

Once you decide on the strategy you wish to take, you then need to decide on how the chamber controls should be set, and on how data is to be collected.

Light

The best light source for light response curves is the red+blue 6400-02B or 6400-40 LCF. The red only 6400-02 source has the potential problem of allowing excessive (that is, more than normal) stomatal closure as light decreases, or delaying stomatal opening as light increases.

Without the LED light source, a light curve cannot be automated, but is still possible. Neutral density filters, for example, can provide means to reduce sunlight or other sources by known amounts. The survey technique discussed above could be done without a light source.

CO$_2$

It is important to maintain the chamber CO$_2$ concentrations as constant as possible while measuring a light response curve. Otherwise, the effects of CO$_2$ on photosynthesis will be confounded with the effects of light. If you have a CO$_2$ mixer, this is simple to do: set it to control on sample CO$_2$ concentration.

Temperature

Ideally, the response curve should be measured at a constant leaf temperature.

Humidity Control

Operate the flow control for constant water mole fraction. If you go from light to dark, expect conductances and transpiration rates to fall, so leave room for the flow to fall as well (or rise, if you are going from dark to light).

Matching

Since the concentrations in the IRGAs aren't going to be changing much during a light curve, there's no real reason to match after every measurement. Match once before starting. If you are doing a slow curve, however, matching won't hurt anything, since you'll have time to burn.

With OPEN version 3.2 and above, you are asked for a matching threshold (the absolute value of the ΔCO$_2$ value). Thus, you don't have to decide whether to match before each observation or not; it will match on the ones with ΔCO$_2$ smaller than your threshold, and skip the rest.

AutoPrograms

There are at least two possibilities here: "LightCurve" (described on page 9-43) and "TimedLamp" (described on page 9-45). "LightCurve" lets you specify the sequence of light values you want. A minimum and maximum wait time is specified. (Logging can't occur before the minimum time expires; after that, a record is logged when stability is achieved. "TimedLamp" also lets you specify a sequence of light values, but at each one, you specify a) how long to maintain that light level, and b) how often to log data within that period. This program is good for recording events throughout the experiment, letting you record how the leaf responded with time, as well as with light.

Rapid Light Curve, Step-By-Step

Here's how to make an automatic light response curve. It uses "LightCurve", and does a rapid response curve.

1 Prepare the chamber

Light: Typically 1500 μmol s^{-1} for C$_3$ plants, or 2000 for C$_4$ plants.

CO$_2$: Constant reference CO$_2$, about 400 μmol mol^{-1}, or your choice. (This is temporary - we'll switch to constant sample in a few minutes.)

Flow: Constant flow, 500 μmol s^{-1}.

2 Clamp onto the leaf

3 Set the temperature

Set the temperature control for constant leaf temperature.

4 Set the chamber humidity

After the chamber has been clamped onto the leaf for a few minutes, note the *H2OS_mml* value, then change the flow control to constant mole fraction control, and target that value.

5 Set the chamber CO$_2$

Control constant sample CO$_2$, targeting the desired value.

6 Set log options, and open a log file

Make sure you've got the computations, prompts, log list, etc. that you need.

7 Area and Stomatal Ratio

Are they correct?

8 Match the IRGAs

Be sure *CO2S_μml* is stable before you do this.

9 Launch the "LightCurve" Autoprogram

Press 5 then **f1**. Pick "Light Curve" from the list.

When asked "`Append to current data file?`" Press **Y**

When asked "`Desired lamp settings (`μ`mol/m2/s)`", edit the list as needed, and press **enter**.

When asked "`Minimum wait time`", enter the desired value. 120 seconds is usually adequate. This is the time after each light level change that the system will wait before checking stability to see if it can log.

When asked "`Maximum wait time`", enter the desired time, in seconds. After the minimum time, it will check stability up to this time to see if it can log. Enter 200. That gives it 80 seconds after the initial 120 second delay for photosynthesis to stabilize.

When asked "`Match if |ΔCO2| less than ppm`", enter **15**.

When asked "`Stability Definition OK (Y/N)`", press **Y** to keep it, or **N** to change it.

10   Trigger the first point
If the first point is the current value, there's not much point in waiting. Press **escape**, then **T** to log it, and start the next one.

11   Watch the curve develop
Press **4** then **f3**, and watch the curve develop.

12   When it's done...
Once the curve is done, you may want to set the light high again by hand, to let the leaf recover. Or just take the leaf out of the chamber if you are done with it.

13   After the fact analysis
Before you close the data file, you may want to access GraphIt (press **View File** (**f2** level 1) in New Measurements mode). If the axes are not defined for a light curve, press QuikPik Config (**f1**) and select "Light Curve". Press REPLOT GRAPH (**f2**) and draw it.

Answer these questions by plotting the appropriate data. Did sample cell CO_2 stay constant? Did the sample cell humidity stay constant? How did stomatal conductance behave as a function of light? What does a graph of photosynthesis vs. conductance look like?

14   Exit GraphIt, and close the file
Press **escape** until you get back to New Measurements mode, then press **CLOSE_FILE** (**f3** level 1) to close the file.

CO$_2$ Response Curves

Why Measure CO$_2$ Response?

An A-C$_i$ curve (assimilation rate plotted against intercellular CO$_2$ concentration) can provide a number of insights into the biochemistry of a leaf or plant:[5]

- **CO$_2$ compensation point**
  The value of C$_i$ where photosynthesis and respiration are in balance.

- **Carboxylation efficiency**
  The initial slope provides an *in vivo* measure of the activity of Rubisco in the leaf. This is sometimes called the mesophyll conductance.

- **Stomatal limitations**
  Stomatal limitation of photosynthesis can be separated from mesophyll limitations.

- **Carboxylation limitations**
  Within the mesophyll, carboxylation limitations can be separated from electron transport limitations.

Operational Considerations

Some things to consider when doing a CO$_2$ response curve.

Light

Even the 6400-02 LED (red only) source will work just fine for CO$_2$ response curves, since the goal is to maintain constant light during the measurement. Stomatal behavior, which the blue light controls, is not as important for this measurement, provided the stomata stay reasonably open. Differential closing ("patchiness") can be a problem, however.

CO$_2$

Speed is important, not precise, predetermined, in-chamber values. Therefore, use the mixer in the constant reference mode. If you wish to eliminate the time that the system takes to lock in on a particular reference value, you

---

[5.]See, for example, G.D. Farquhar and T.D. Sharkey (1982) Stomatal conductance and photosynthesis. *Annual Review of Plant Physiology* 33,317-45. Also G.D.Farquhar, S. von Caemmerer, J.A.Berry (1980) A biochemical model of photosynthetic (CO$_2$) assimilation in leaves of C$_3$ species. *Planta* 149,78-90.

could also run the mixer in the constant control signal mode (option C). If you do, you'll be entering the target values for the AutoProgram in mV instead of μmol mol^{-1}.

In what order should the curve be measured? There are a couple of constraints to consider. One is that high CO_2 concentrations may induce some stomatal closure, so if you are including high CO_2, they should be done last. The other constraint is that if too much time is spent near the CO_2 compensation point, enzyme deactivation may occur. A suggested measurement scheme is to start at ambient, go down to the compensation point, return to ambient, then increase to the upper limit.

Temperature
The response curve should be measured under constant temperature conditions. Operate the coolers at a constant leaf temperature.

Humidity Control
Operate the flow control for constant water mole fraction. Expect higher conductances and transpiration rates at the low CO_2 values, so choose a mole fraction target that gives a flow rate that has room to increase (e.g. 500 or 600 μmol s^{-1}).

Matching
Since the concentrations of CO_2 are covering a large range, match before each reading.

With OPEN version 3.2 and above, you are asked for a matching threshold (the absolute value of the ΔCO_2 value). Thus, you don't have to decide whether to match before each observation or not; it will match on the ones with ΔCO_2 smaller than your threshold, and skip the rest.

Diffusion
This can be a problem for A-Ci curves, since there can be a large concentration gradient between the chamber and ambient. See **Diffusion Leaks** on page 4-44.

Step-By-Step
Here's how to make an automatic CO_2 response curve. It uses the AutoProgram "A-CiCurve", described on page 9-39.

1   Set the chamber conditions

Light: Set the desired value. If not using the LED source, note that constant light is critical for this measurement. It should typically be saturating light (usually > 1500 μmol m^{-2} s^{-1} for C3 plants).

Flow: Fixed at 300 μmol s^{-1}. You want 50% RH or higher in the chamber.

CO$_2$: Constant reference CO$_2$, at about 40 or 50 μmol mol^{-1} above ambient.

2   Clamp onto the leaf

3   Set the humidity control

Note the value of *H2OS_mml*. Then change the flow control to constant mole fraction, and target this value. The flow should be 300 μmol s^{-1} or so. We'll need room for it to increase, because the conductance will likely increase during the measurement as the chamber CO$_2$ decreases.

4   Set the temperature

Set the temperature control for constant leaf temperature.

5   Open a log file

Make sure you have the computations, prompts, log list, log options, etc. that you need.

6   Area and Stomatal Ratio

Are they correct?

7   Real time graphics

Set up a screen for plotting A-Ci. (*PHOTO* on the Y axis, *Ci* on the X). If there already is one, clear its data.

8   Match the IRGAs

Be sure *CO2S_µml* is stable before you do this.

9   Launch the "A-CiCurve" Autoprogram

Press **5** then **f1**. Pick "A-CiCurve" from the list.

When asked "`Append to current data file?`" Press **Y**

When asked "`Enter the desired values?`", edit the entries until they are what you want. For example, use 400 300 200 100 50 400 400 600 800. (If it's a C$_4$ plant, use 0 instead of 50). Notice there are two 400's in a row after the low value. That is not an error, but a trick to give the leaf some recovery time after the low CO2 measurement. Later on, we can discard the first of those readings, if it doesn't fit.

When asked "`Minimum wait time`", enter the desired value. 60 seconds is usually adequate. This is the time after each CO_2 level change that the system will wait before checking stability to see if it can log.

When asked "`Maximum wait time`", enter the desired time, such as 120. This is the longest any one point will take.

When asked "`Match if |ΔCO2| < ppm`", enter **15**.

When asked "`Stability Definition OK? (Y/N)`", respond as needed.

The experiment will then start automatically.

10   **Watch the curve develop**
Press **4** then **f3**, and watch the curve develop.

11   **When it's done...**
Once the curve is done, you may want to set the CO_2 back to the starting value, to let the leaf recover. Or just take the leaf out of the chamber if you are done with it.

12   **After the fact analysis**
Before you close the data file, you may want to access GraphIt (press **View File** (**f2** level 1) in New Measurements mode). If the axes are not defined for an A-Ci curve, press QuikPik Config (**f1**) and select "A-Ci Curve".

Answer these questions by plotting the appropriate data. Did the sample cell humidity stay constant? Did leaf temperature stay constant?

13   **Exit GraphIt, and close the file**
Press **escape** until you get back to New Measurements mode, then press **CLOSE_FILE** (**f3** level 1) to close the file.

Matching the Analyzers

The purpose of matching is to remove offsets between the sample and reference analyzers caused by small variations in temperature, flow rate, calibration, drift with time, etc.

Matching the analyzers improves the accuracy of your measurements, especially when working with low photosynthesis rates. Recall from Equation (1-15) on page 1-10 that photosynthesis is proportional to the measured CO_2 differential:

$$A \propto C_r - C_s \qquad\qquad (4\text{-}1)$$

If $C_r = 360 \, \mu\text{mol mol}^{-1}$ and $C_s = 330 \, \mu\text{mol mol}^{-1}$, and there is a $1 \, \mu\text{mol mol}^{-1}$ offset between the IRGAs, then the photosynthetic rate A is in error by 1/30 or 3.3%. If, however, the differential is small (for example $C_r = 360$ and $C_s = 355$), then the error in photosynthetic rate due to a $1 \, \mu\text{mol mol}^{-1}$ offset is 1/5 or 20%. Clearly, the smaller the differentials, the more important matching becomes.

The first step to matching is making the IRGAs see the same air. This is a mechanical exercise accomplished by a small valve on the bottom of the chamber/IRGA (Figure 4-2). Outgoing chamber air is sent to the reference cell, and the air that normally goes to the reference cell is diverted. The second step is to make the IRGAs read the same, and this is a mathematical operation. The equations for sample H_2O and CO_2 (page 14-6 and page 14-8) contain adjustment terms (W_{ms} and C_{ms}), and it is these that are changed when matching. Thus, the sample, not the reference, CO_2 and H_2O values are adjusted.

NOTE: There are alternative methods of matching. See **Matching Variations** on page 16-19.
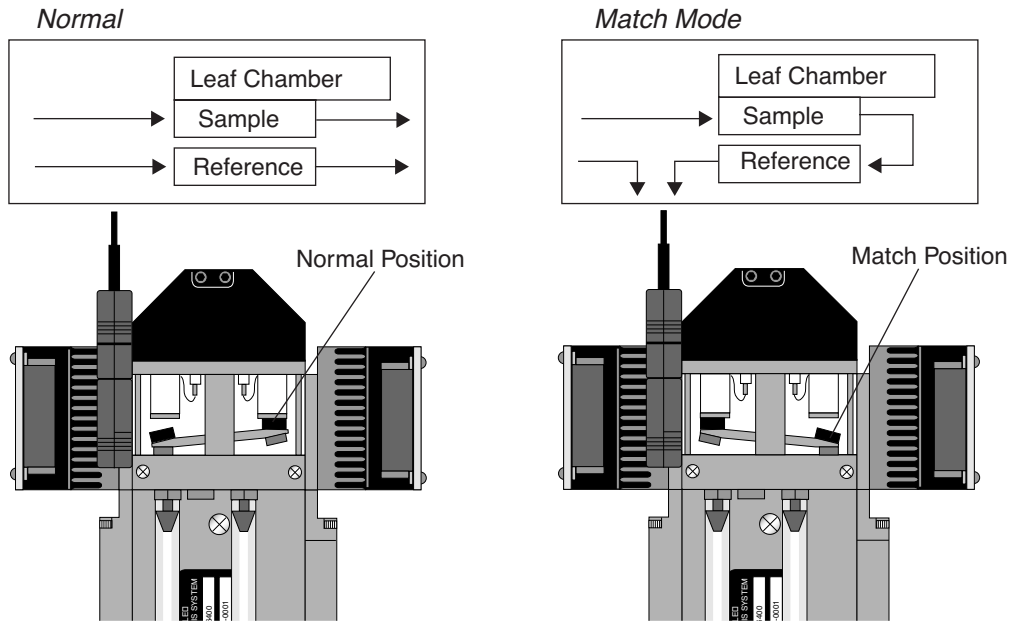
Figure 4-2. The match valve puts exhaust air from the sample cell into the reference cell, allowing both cells to be matched without altering conditions in the leaf chamber.

How to Match

Match mode is entered by pressing **MATCH** (**f5** level 1 in New Measurements mode).

> Only match the IRGAs if the sample cell concentrations (*CO2S_μml* and *H2OS_mml*) are stable.

The match valve will toggle (as shown in Figure 4-2), and a countdown is displayed (Figure 4-3). The entry countdown covers the time period in which the reference cell is being flushed with air from the sample cell. The length of this delay period is based on the stability of the reference H_2O IRGA. After a change in incoming air, water will take longer to come to a new equilibrium than CO_2, because of sorption effects. During this period, the display will

show the time remaining, and the range in the reference H_2O readings over the last 4 seconds. The delay ends when a) 45 seconds elapses, or b) the H_2O range falls to < 0.1 mmol mol^{-1}.
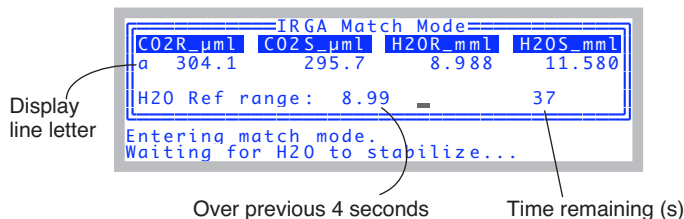
Figure 4-3. When match mode is entered, a countdown is displayed. The delay gives the reference cell time to flush out.

If match mode is entered with a flow rate less than the minimum recommended flow (50 mol s^{-1} with a CO_2 mixer, 100 μmol s^{-1} without), then a warning will appear (Figure 4-4) giving you a chance to cancel match mode. If you choose to enter, the countdown will likely last the full 45 seconds.
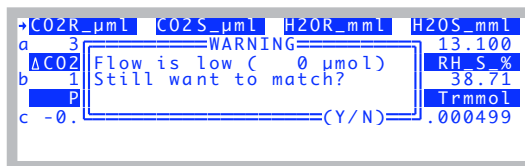
Figure 4-4. The low flow warning.

Note that the entry and exit delays can be cut short by pressing **escape** during the countdown.

Changing Displays

To change displays in match mode, simply press the letter of the desired display. For example, to see the status display line j, press **j** (Figure 4-5).

```
════════IRGA Match Mode════════
 CO2  H2O Pump Flow Mixr  Fan
j OK   OK   OK   OK  off Fast

Prev: 00:08:43  CO2=316  H2O= 9

   exit                      MATCH
                             IRGAs
```
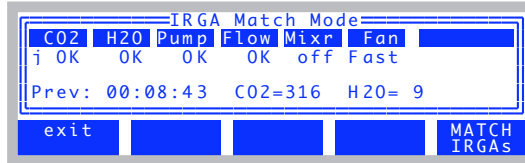
Figure 4-5. Press the display line letter to change the display.

The default display line for match mode is set in the <open> <matching> <disp> node of the system configuration. See Chapter 16.

What Happens in Match Mode

Once in match mode (Figure 4-6), your choices are provided by the function keys: **f5** (**MATCH IRGAs**) matches the IRGAs (computes a new C_{ms} and W_{ms}), or **f1** (**exit**) quits. The display indicates the values of sample CO_2 and H_2O when last matched, and the elapsed time since the last match. Pressing **f5** (**MATCH IRGAs**) will cause C_{ms} and W_{ms} to be adjusted so that the sample and reference values become the same. You can do this as often as you like while in match mode, or not at all. Pressing **f1** (**exit**) will cause the match valve to toggle back to normal position, and the exit countdown will commence.

```
════════IRGA Match Mode════════
CO2R_µml  CO2S_µml  H2OR_mml  H2OS_mml
a  312.1     308.5     9.346     9.226

Prev: 00:00:21  CO2=316  H2O= 9

   exit                      MATCH
                             IRGAs
```
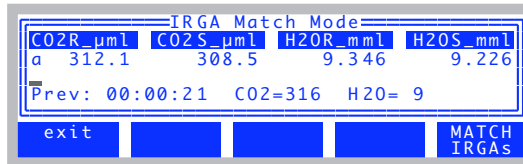
Figure 4-6. The display while in Match Mode. The elapsed time is the time since the previous match, and the previous values are the concentrations at which the last match occurred.

Messages in Match Mode

Match mode has some messages for alerting you to possible problems:

"CO2R Didn't Change"

After the H_2O reference reading has stabilized, if the message
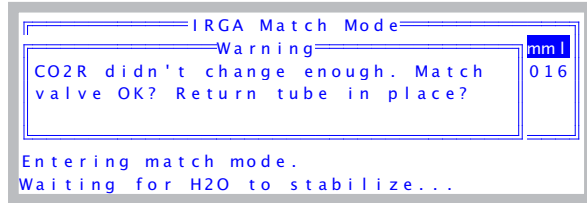
```
┌──────────IRGA Match Mode──────────┐
│┌─────────Warning──────────┐    mm │
││ CO2R  didn't  change  enough.  Match │  016
││ valve OK? Return tube in place? │
│└──────────────────────────┘
│
│ Entering  match  mode.
│ Waiting  for  H2O  to  stabilize...
└───────────────────────────────────┘
```

appears, it is because the CO_2 reference reading changed less than 1.5 μmol mol^{-1} after the match valve closed, and the expected change was much larger than that. Reasons for this would be a match valve that is sticking, or the air flow tube connecting the chamber to the match valve not being in place, or some other flow related problem.

"CO2S Has Changed"

The sample cell CO_2 concentration at the start of match mode is retained, and periodically compared to subsequent values as a stability check. Whenever the difference exceeds 3.0 μmol mol^{-1}, a warning appears (Figure 4-7).

```
┌──────────IRGA Match Mode──────────┐
│ CO2R_µml  CO2S_µml  H2OR_mml  H2OS_mml │
│ a  316.1    308.5     9.430     9.233 │
│ CO2S has changed   -3 ppm (leak?) │
│ Prev: 00:01:33  CO2=316  H2O= 9   │
│                                   │
│  exit                      MATCH  │
│                            IRGAs  │
└───────────────────────────────────┘
```
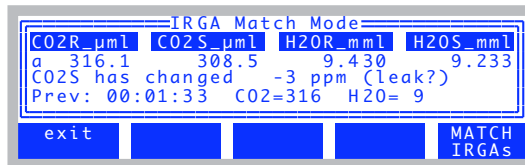
Figure 4-7. The leak message appears if the CO2S_µml value is more than 3 µmol mol^{-1} from the value it had when match mode was entered.

Since the sample CO_2 concentration should not be affected by match mode, drift in this value indicates a problem. The cause for such drift is either unstable incoming CO_2, a leak, or a sudden change in photosynthetic rate. While in match mode, if the sample CO_2 is stable but the reference varies, then there is a leak in the chamber exhaust tube, or else a problem with the match valve itself. See page 20-22 for troubleshooting help.

The best way to prevent this message is to not enter match mode when the sample cell concentrations are unstable.

"Excessive Deltas"
If **MATCH** is pressed, and the difference between sample and reference IR-GAs exceeds 10 μmol mol^{-1} for CO_2 or 1 mmol mol^{-1} for H_2O, a warning will be displayed:

```
             ═════════WARNING═════════
CO2R │ Excessive deltas!           │ 2OS_mml
 a  3 │   –Chamber closed?          │  11.012
CO2S │   –Match valve working?      │ ?)
Prev │ Still want to match?         │

 exi          ═════Press (Y/N)═════      MATCH
                                          IRGAs
```
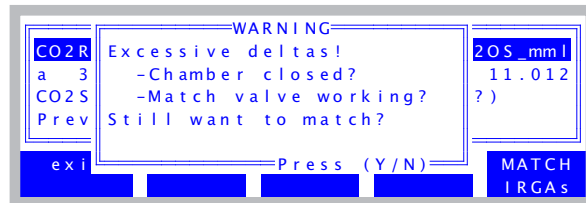
Note that the limits of 10 μmol mol^{-1} and 1 mmol mol^{-1} are user adjustable. See **Match Settings** on page 16-21.

As the message indicates, a bad leak in the chamber or a stuck match valve can cause the reference - sample differences to be this large. Badly zeroed and/or spanned IRGAs can cause this to happen as well.

See page 20-22 for troubleshooting help.

When To Match

- **When you start**
  Remember to match before getting very far with the first leaf of the day.

- **When rates are low**
  With low photosynthesis or transpiration rates, sample - reference differences will be small, and any offsets will be important.

- **After large concentration changes**
  If the IRGAs are well zeroed and spanned, matching once at any concentration will suffice for all other concentrations. If the IRGAs are not be so well zeroed and spanned, matching will be a function of concentration.

- **After large flow rate changes**
  This counteracts any potential mixing-flow rate interactions in the chamber.

- **Periodically**

  How often? That depends most on temperature changes, so it's hard to pre-scribe a definite time. Start with 30 minutes or so, and adjust as needed. Use of the coolers to stabilize temperatures will help to minimize zero drift and thus reduce the need for matching.

Viewing Previous Match Information Anytime

In New Measurements mode, you can find out how long it has been since the last match, or what conditions were for the last match from display line 'm' (Figure 4-8). *matchCO2* is the reference CO_2 reading at the previous match, *matchH2O* is the reference H_2O reading at the previous match, and *mchElpsd* is the elapsed time (HH:MM:SS) since the previous match.
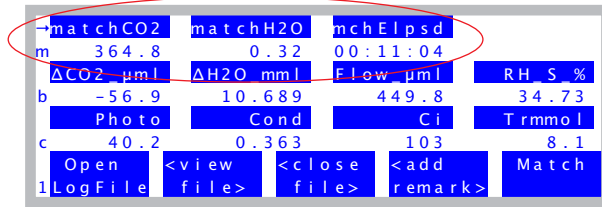
Figure 4-8. Display line m (on standard displays starting with version 6.2) show pervious match information.

If you haven't matched yet, these values will be
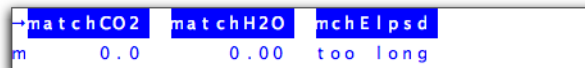
These three variables are system variables with IDs of -113, -114, and -115 (See **List of Open 6.2 System Variables** on page 14-23), so you can add them to any display you wish, if they aren't there already, and /or include them in your data files.

Logging Match Adjustments

The adjustment factors that are computed when a match occurs can be stored each time you log data (Figure 4-9).

"Flow"	"PARi"	"PARo"	"Press"	"CsMch"	"HsMch"	"Status"
172.8	1026	1311	98.07	−2.2	0.04	111115
215.7	1070	1354	98.07	−1.2	0.03	111115
233.3	1062	1337	98.07	−0.8	0.02	111115
199.4	1164	1450	98.07	−0.6	0.01	111115
200.2	1160	1440	98.07	−0.5	−0.01	111115
199.7	1113	1374	98.06	−0.6	0.00	111105

Figure 4-9. The default log file format includes the adjustment factors that are set when the IRGAs are matched. This provides a record of what happened.

Match Mode and AutoPrograms

When match mode is entered automatically as part of an AutoProgram, there will be no messages sent to the display, since it is assumed no one is there to respond to them. Instead, OPEN will make some choices, and tell you what it did with messages (and time stamps) sent to the log file (or .REM file if you are storing remarks separately).

Low Flow

If flow control is fixed, then matching is skipped, with the logged message

    "14:20:33 Didn't match! Flow too low"

If flow control is not fixed, but set for maintaining constant humidity (or mole fraction, or vpd, etc.), and if the current flow is too low, the flow will be temporarily increased to 500 μmol s^{-1} for the duration of the match. A message to that effect will be logged with a time stamp:

    "14:20:33 Low Flow. Increased for matching"

CO2S Has Changed

If the sample CO_2 value never stabilizes sufficiently after 1 minute to do a match (see **"CO2S Has Changed"** on page 4-37), this message will be logged:

    "14:20:33 Didn't match! Unstable CO2S"

The 1 minute time limit is user adjustable. See **MaxAutoTime** on page 16-21.

CO2R Didn't Change

If the CO_2 reference value doesn't change enough when match mode is entered (see **"CO2R Didn't Change"** on page 4-37), this message will be logged:

```
"14:20:33 Didn't match! Valve stuck or flow
problem."
```

Excessive Deltas

If the required change is excessive (**"Excessive Deltas"** on page 4-38), this message is logged, but the match still occurs.

```
"14:20:33 Warning! Large Deltas in match: ΔCO2=n,
ΔH2O=n"
```

Stability Considerations

In New Measurements mode, the LI-6400 measures and computes continually, regardless of the state of equilibrium of the leaf in the chamber. It can also log data with the same disregard for stability. The question is, how do you know when the system is stable enough to record meaningful data? Also, when you log an observation, how can you tell, when you look at the data later, how stable the reading was?

Stability Indicators

Version 5 introduces a powerful technique for determining system stability (**Stability Indicators** on page 6-29). It allows you to set up criteria based on measured and computed variables, to determine when the system is stable. For each variable chosen, stability can be based on statistics (any combination of standard deviation, rate of change, and coefficient of variation) over a time period of your choosing. Table 4-1 indicates a typical definition.

Table 4-1. Typical stability definition

Variable	Time (s)	Standard Deviation	Coefficient of Variation (%)	Rate of Change (per minute)
PHOTO	20	< 0.5		< 0.1
COND	20	< 0.1		< 0.05

The shaded entries are "don't care". That is, they don't enter into the decision. Therefore, with the definition given in Table 4-1, if one or more of the non-shaded conditions is not met, the system would be considered unstable.

What variables should be checked?

There are two approaches to determining system stability: Check all the inputs, or check the results. By inputs, we mean the list of variables that go into calculating photosynthesis and conductance: *CO2R*, *CO2S*, *H2OR*, *H2OS*, *Flow*, *Tleaf*. Add to that list *ParIn*, since light is driving photosynthesis. If all of those were stable, then certainly *PHOTO* and *COND* will be stable.

The merit of checking just the results, *PHOTO* and *COND*, is that it is simple. On the other hand, keeping statistics on all of the inputs is good for diagnostic purposes. For example, if *PHOTO* is not stable, why is it that way? Is the flow rate varying (automatic humidity control)? Is *CO2R* unstable? If *CO2R* is stable, but *CO2S* is not, it could be resulting from flow fluctuations, or from *ParIn* fluctuations. And so on.

What statistics should be checked?

At first glance, time rate of change seems sufficient. If that's all you check, you could run the risk of calling a signal stable that was noisy or in transition, and just happened to have a rate of change (slope of a straight line fitted to values of the signal plotted against time), near zero at one instant in time. If you were only considering one variable in the stability definition, then just looking at only the rate of change could be risky. If you have several variables included, then the chances of getting fooled go down.

Logging Stability Indicators

The Log Options part of the configuration includes something that provides information on system stability at the time of logging (Figure 4-10).
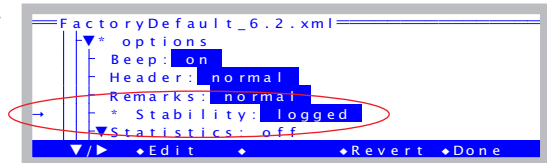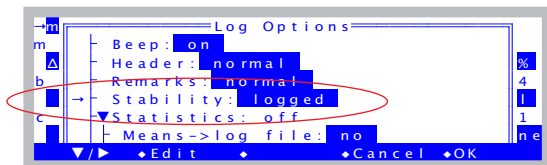
Figure 4-10. Logging stability details can be accessed in two ways.

The *stability* entry can be "no" or "logged". When it is set to "logged", there will be 5 extra columns in your data file for each variable that you have in your stability criteria (Figure 4-11).

N(CO2S)	MN(CO2S)	SD(CO2S)	CV(CO2S)	SLP(CO2S)	N(H2OS)	MN(H2OS)
in	in	in	in	in	in	in
26	307.583459	0.06531	0.02123326	0.00600133	26	11.0142
30	307.571838	0.08927235	0.02902488	-0.006642	30	11.0132

N - sample count
MN - mean
SD - standard deviation
CV - coefficient of variation
SLP - rate of change (per minute)

Figure 4-11. Extra columns are added to your data file when the Log Option stability *is enabled. Circled above is an example for when* CO2S *is one of the stability variables.*

For more on this and other Log Options, see **Log Options** on page 9-14.

Real Time Graphics

A useful visual indicator of stability can be had with judicious use of New Measurement's strip chart mode (**Real Time Graphics** on page 6-14). You can plot photosynthesis and conductance as a function of time, and watch for any trends over the past minute or two, or more.

If you are using a buffer volume, it is useful to keep an ongoing plot of reference CO_2, so that if photosynthesis doesn't seem to be stabilizing, you can quickly tell if the problem is physiology (reference is stable) or mechanical (reference is unstable).

Average Time

You have at your disposal a configuration parameter that influences stability. It is the <open> <a2d> <avgtime> node in the configuration tree (**Averaging Time** on page 16-29). By default, it is 4 seconds, but you can raise or lower that number if you wish.

If you require the quietest possible signal, and can afford longer equilibrium times, then you may want to raise this number to 10 or 15 seconds. If, however, you are trying to measure transients, or the system dynamics need to be optimized, then set this to 0.5. (Any value at or below 0.5 will give you 0.5

second averaging, which is the frequency of new measurements.) The price will be slightly greater IRGA noise than you'd get at 4 seconds.

Leaks

In photosynthesis systems, there are two types of leaks: bulk flow and diffusion. Bulk flow leaks occur when there is a hole (apart from the system inlet and outlet) that allows air to move into or out of the system. Diffusion occurs when a particular gas, such as CO_2, moves through the walls of the system in response to a concentration gradient.

Bulk Flow Leaks

The pressure inside the leaf chamber is slightly positive so that bulk flow leaks tend not to be a problem. At low flow rates, however, this positive pressure is more than offset in certain parts of the chamber by the chamber circulation fan. This can have dramatic consequences: if the center O-ring on a leaf chamber is missing, for example, ambient air can be sucked into the chamber.

- **Check the O-rings**
  They have a way of escaping when you are changing chambers, so make sure they are all in place.

- **Check the gasket material**
  If the white gasket on the light source is flattened, change it. The black gasket material (neoprene) can recover if left uncompressed overnight. When the chamber is not being used, you can preserve your gaskets by adjusting the thumb nut so that the gaskets are not compressed when the chamber is latched.

- **Check the seal around leaves and stems**
  When the gaskets are compressed around thicker leaves and stems, small gaps will be created. At high flow rates, this may not be a problem, but at lower rates, be sure to seal them with putty or gum.

If bulk flow leaks are there, they will be a bigger problem at low flow rates, than at high flow rates.

Diffusion Leaks

CO_2 moves from high concentrations to low concentrations. It does this not only through air, but also through solids, including many plastics and synthetic rubbers. CO_2 can pass through Bev-a-line tubing (polyethylene, lined with ethylene vinyl acetate), through gasket material (ethylene and neoprene), and

through O-rings (butyl rubber). It goes through just about everything that's not glass or metal. There are some thin-film materials that have very low permeabilities to CO_2, however, such as Teflon, Saran (polyvinylidene chloride), Mylar, and Propafilm[®] (polypropylene coated with Saran).

Diffusion of CO_2 into or out of the LI-6400 leaf chamber is proportional to the difference between the inside and outside CO_2 concentrations. It is useful to think of this diffusion leak as a flux of CO_2. When the bulk flow rate through the chamber is high, this diffusion flux will affect the chamber concentration very little. When the bulk flow is low, that same diffusion flux will have a much larger effect.

The effects of diffusion in a gas exchange system are proportional to the ratio of surface area (tubing, gaskets) to leaf area (i.e. large leaf area is good, small leaf area is bad). Thus, while you might be able to ignore diffusion problems with a 2x3 or 2x6 chamber, you absolutely cannot ignore them when using the 6400-15 Extended Reach 1cm Chamber.

A diffusion model is presented in Figure 4-12, and predicts a linear relation when normalized leak is plotted against 1/flow rate. (The diffusion leak is $(c_o - c_i)$; the normalizer is the gradient $(c_a - c_o)$.)

$$\rho v \frac{\partial c_o}{\partial t} = uc_i - uc_o + kc_a - kc_o$$

$$0 = u(c_i - c_o) + k(c_a - c_o)$$

$$u(c_o - c_i) = k(c_a - c_o)$$

$$\boxed{\frac{c_o - c_i}{c_a - c_o} = \frac{k}{u}}$$
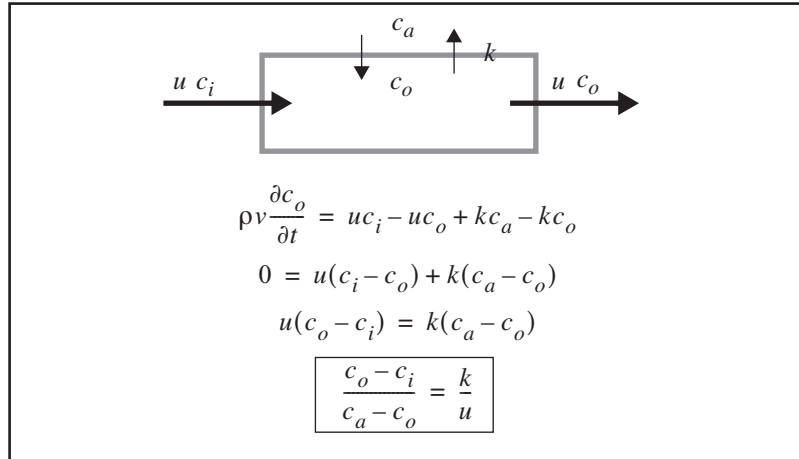
Figure 4-12. A model for diffusion in an open photosynthesis system, based on conservation of mass. The walls of the system have CO_2 leak rate k (mol s^{-1}). u is bulk flow rate (mol s^{-1}), and c_i, c_o, and c_a are incoming, outgoing, and ambient CO_2 concentrations (mol mol^{-1}). In steady state, the change of CO_2 with time in the system is 0.

To verify this model, we performed a simple experiment[6]. We used the CO_2 mixer to generate a variety of reference cell CO_2 concentrations, and recorded the difference between the sample and reference cell concentrations at various flow rates. (No leaf in the chamber.) We took the following steps to ensure the ambient CO_2 concentration was as stable as possible: 1) data collection was done with an AutoProgram, so no one had to be standing there breathing; 2) the instrument was set up in a vacant, well mixed greenhouse; 3) an external fan continually ventilated the chamber; 4) the ambient CO_2 concentration was monitored with a second gas analyzer. Sample cell CO_2 concentrations were measured by the reference IRGA using the match valve. This eliminated any potential errors due to IRGA drift, since the same IRGA

---

[6].Simple in hindsight - it took a couple of weeks to get it right.

(reference) was used for both sample and reference measurements. The results are plotted in Figure 4-13.
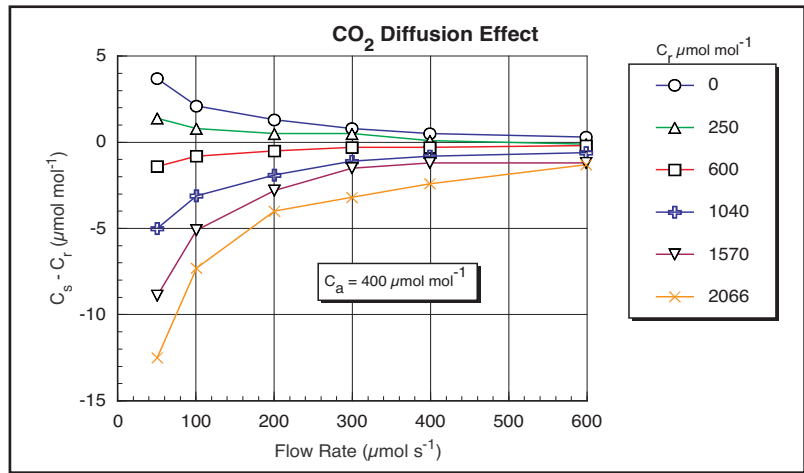
Figure 4-13. CO_2 diffusion into a closed, empty LI-6400 chamber with black neoprene gaskets as a function of flow rate, for various reference concentrations C_r.
C_a (ambient CO_2) was 400 $\mu mol\ mol^{-1}$. Diffusion effects are highest at low flow rates and large gradients. The lines in the figure connect data of common C_r.

The diffusion effect $(C_s - C_r)$ plotted in Figure 4-13 is equivalent to the $(c_o - c_i)$ term in the model in Figure 4-12. Normalizing by the gradient $(C_a - C_s)$ should make this data fall on one curve if the model is correct, and it does to a reasonable degree (Figure 4-14). Curve fitting yields a diffusion coefficient *k* of 0.46. (The easiest curve fit is to plot (1/*Flow*) on the X axis, rather than *Flow*; *k* is then the slope of the line.) Note that the outliers tend to be the data collected with the smallest gradient, so uncertainties in what the true gradient was are largest.
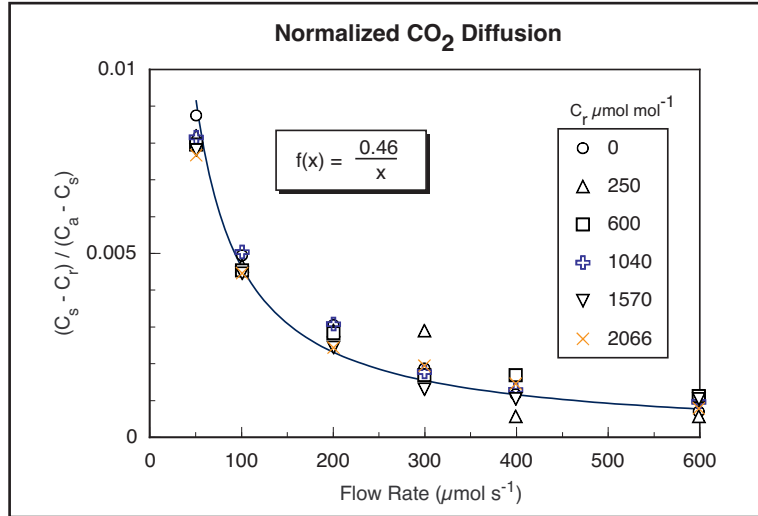
Figure 4-14. Data from Figure 4-13 normalized by the CO_2 gradient, $(C_a - C_s)$.

In order to correct photosynthesis calculations for the effects of CO_2 diffusion, we re-visit the derivation of the photosynthesis equation. The mass balance equation ((1-11) on page 1-9) becomes

$$sa = uc_i - uc_o + k(c_a - c_o) \qquad (4\text{-}2)$$

The $k(c_a - c_o)$ term accounts for diffusion. The final equation ((1-15) on page 1-10) becomes

$$A = \frac{F(C_r - C_s)}{100S} - C_s E + \frac{k}{100S}(C_a - C_s) \qquad (4\text{-}3)$$

Note that there are now two "correction terms": one for transpiration, and one for diffusion. The diffusion correction term is insignificant for measurements with near-ambient CO_2 concentrations in the chamber (Table 4-2). Near the CO_2 compensation point it's a different matter; diffusion becomes signifi-

cant, and failure to account for it will lead to a large relative overestimation of assimilation rate.

Table 4-2. Typical values of the three terms of the net photosynthesis equation under two sets of conditions

Term	Equation	Near Ambient	Near CO_2 Compensation
CO_2 Uptake	$\dfrac{F(C_r - C_s)}{100S}$	$\dfrac{300(380 - 340)}{100 \times 6} = 20$	$\dfrac{200(50 - 48)}{100 \times 6} = 0.67$
Transpiration Correction	$C_s E$	$340 \times 0.005 = 1.7$	$48 \times 0.007 = 0.33$
Diffusion Correction	$\dfrac{k(C_a - C_s)}{100S}$	$\dfrac{0.4(370 - 340)}{100 \times 6} = 0.02$	$\dfrac{0.4(370 - 48)}{100 \times 6} = 0.21$
Net Photosynthesis		$20 - 1.7 + 0.02 = 18.32$	$0.67 - 0.33 + 0.21 = 0.55$

A protocol for measuring at concentrations well away from ambient is:

- **Minimize the gradient**
  Keep high CO_2 (breath) away from the chamber. Keeping the chamber well ventilated will help do this. If possible, collect data with AutoPrograms, so an operator doesn't need to be nearby.

- **Use the diffusion corrected formula for photosynthesis**
  Implement the correction in the Compute List.

Recent Developments

Flexas et al[7]. show evidence that the bulk of diffusion leaks occur at the interface of the two sealing gaskets, and not through the gaskets themselves. They suggest a method of quantifying leak rates by clamping onto leaves rendered photosynthetically inert.

---

[7].J. Flexas, A. Dıaz-Espejo, J. A. Berry, J. Cifre, J. Galmes, R. Kaldenhoff, H. Medrano and M. Ribas-Carbo. 2007. Analysis of leakage in IRGA's leaf chambers of open gas exchange systems: quantification and its effects in photosynthesis parameterization. *Journal of Experimental Botany*, Vol. 58, No. 6, pp. 1533–1543.

Operational Hints

Air Supply Considerations

An open system, such as the LI-6400, is only as good as the incoming air stream is stable, especially with respect to CO_2 concentration. When the incoming air is fluctuating in CO_2 concentration, there will be phase differences as those fluctuations pass through the reference IRGA and the sample IRGA, resulting in fluctuations in the CO_2 differential - even with no leaf in the chamber.

There are essentially three options for making the incoming air stable:

1 **Use the 6400-01 CO_2 Mixer**
All the incoming air is scrubbed by the soda lime column, and the mixer adds whatever CO_2 is necessary to hold your requested concentration.

If you are using the mixer, you'll need to connect your CO_2 source (a CO_2 cartridge or tank - see **6400-01 CO2 Injector Installation** on page 2-7) about 5 or 10 minutes before you expect to use the system, and let it pressurize the internal workings of the mixer.

2 **Use a buffer volume**
When air is moved through a large, mixed volume, fluctuations in incoming CO_2 are greatly dampened, and can be stable enough to use for gas exchange purposes (Figure 4-15).
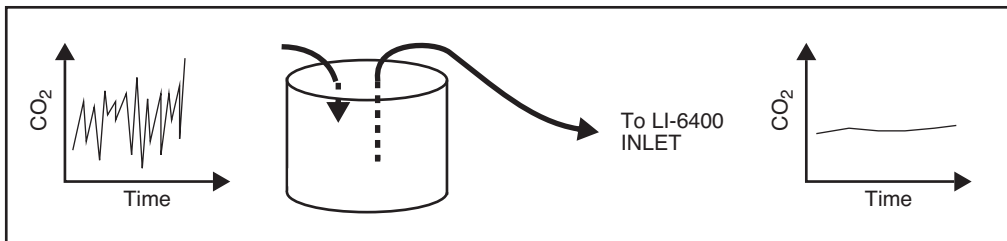
Figure 4-15. A buffer volume will dampen fluctuations in concentration.

Acceptable volumes depend on the magnitude of the fluctuations that need to be dampened, but several liters is a good starting volume. A plastic five gallon enclosed bucket is a good buffer volume, or - if nothing else - use the LI-6400 carrying case as a buffer volume.

3   Use tank air, or some other source

An advantage of an open system is that you can condition the incoming air stream, using CO_2 tanks, humidifiers, oxygen generators, etc., prior to introducing that air to the leaf chamber.

You should plumb the system so that the LI-6400's pump is used, even if the air supply has its own flow control (e.g. compressed gas). Figure 4-16 illustrates how to do this using a T arrangement.
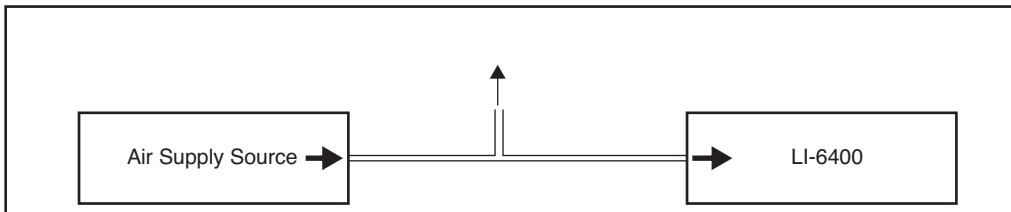
Figure 4-16. When supplying air to the LI-6400 from a supply with its own flow control, make sure the supplied air flow exceeds that required by the LI-6400, and that there is excess flow coming out the T. Make sure the open leg of the T is long enough to prevent diffusion from contaminating the air to the LI-6400.

What's the Light Source?

Make sure the <open> <light> <source> configuration node matches as closely as possible the actual light source you are using (**Specifying the Light Source** on page 8-4).

There are a couple of reasons for doing this. If you are using the 6400-02 or -02B Light Source, the function key for controlling this optional device depends on this configuration item. But even if you aren't, the calibration of the in-chamber light sensor is adjusted to account for the light source, to minimize spectral errors in its calibration. See **Light Sources and Sensors** on page 8-1 for more details.

Also, don't mix the 6400-02 (or -02B) and the LCF. That is, don't configure the software for one, and use the other. The *parIn* readings won't work.

Dealing With Low Rates

Measuring very low rates of photosynthesis or transpiration becomes problematic in an open system. Eventually, the CO_2 or H_2O differential becomes so small that it is in the noise level of the analyzers. Some things to try:

- **Use as much leaf area as possible**
  The more leaf area you can get in the chamber, the larger the differential that you can measure.

- **Use as low a flow rate as possible**
  Consider about 100 μmol s^{-1} an effective lower limit. If you have a CO_2 mixer, then 50 is the low value. Leaks may be a problem, however. See **Bulk Flow Leaks** on page 4-44.

- **Match the IRGAs**
  As the differentials become small, the effect of any offset error is magnified.

Using Closed Mode
We have experimented with using a closed system technique in the LI-6400 to handle low rates. This is accomplished by turning off the pump for 10 or 15 seconds, and measuring the rate of change of CO_2 with time, and H_2O with time. The LI-6400 can be programmed to accommodate closed measurements interspersed at will among normal, open measurements. However, our tests indicate that the repeatability and accuracy of the closed technique is no better (and sometimes worse) than using the open method with a low (100 μmol s^{-1}) flow rate. Contact LI-COR for further information.

Humidifying Incoming Air
The LI-6400's humidity control balances the leaf's transpiration with drier incoming air to maintain a desired humidity (see **Humidity Control** on page 7-7). How dry the incoming air is depends on the manual adjustment knob of the desiccant tube. The limitations of this approach become apparent when measuring a small leaf area and/or low transpiration rate when it is desired to have high humidity in the chamber. Another source of water besides the leaf is needed.

One solution to this problem if you are using the 6400-01 CO_2 Mixer is to add a small amount of water (10 ml) to the soda lime tube (Figure 4-17). After about an hour of subsequent use, the water output becomes quite stable, and remains so for many hours thereafter.

Note
The soda lime used for the experiment in Figure 4-17 was a dry variety, which is no longer available. Most soda lime available now (for example, LI-COR part number 9964-090) is quite moist, so this procedure may not be necessary.

Caution

When adding water to soda lime, do it slowly, letting the chemical absorb the liquid. Then, hold the tube horizontally and shake it, to distribute the moistened (and clumpy) pellets. <u>Avoid adding too much water</u>; if liquid gets out of the tube during operation, downstream metal parts could become oxidized.

Another solution is to use a brand of soda lime that has a higher water output. In situations where you need fairly dry input air to the chamber, however, this material should be avoided, as it can saturate the desiccant tube in an hour or two, making frequent changes necessary.

Adding water to soda lime has an added benefit of helping to prolong or rejuvenate the CO_2 scrubbing capacity of the soda lime. When it is used in dry environments (such as in a closed loop with a desiccant - not a normal configuration for the LI-6400), the scrubbing capacity of soda lime can be greatly diminished.

In dry environments when the 6400-01 CO_2 Mixer is not in use (and little or no air is being routed through the soda lime tube), you can humidify the incoming air stream either by adding a tube with moistened filter paper or sponge to the system air inlet, or by replacing the desiccant in the desiccant tube with this humidifying material. The former method adds a tube to the system, but maintains a wide humidity control, while the latter method adds no hardware but sacrifices the ability to dry incoming air.
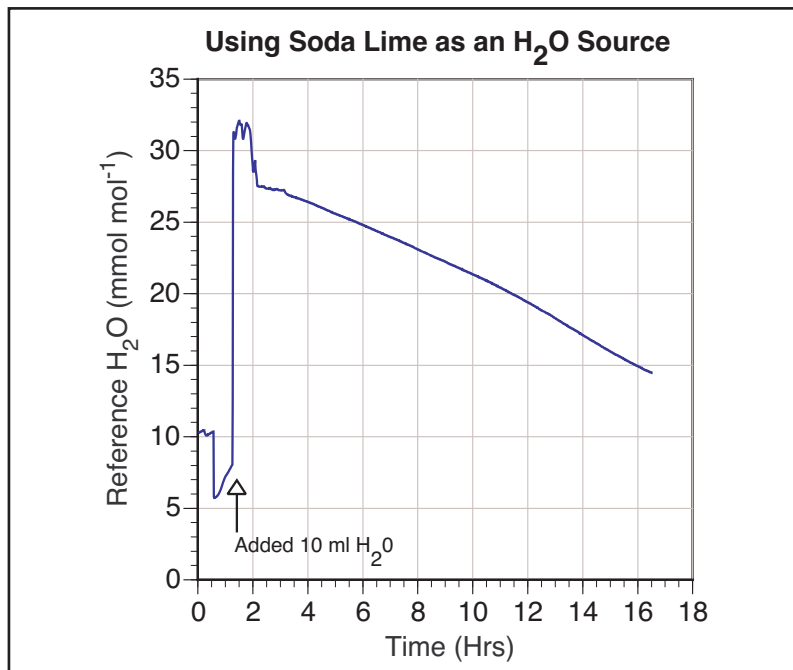
Figure 4-17. Reference H_2O concentration plotted against time. At hour 0, the desiccant and soda lime tubes were fully bypassed, and ambient humidity was measured. After 30 minutes, the soda lime was turned full on, and the water concentration dropped, then began to climb back toward the ambient value. At 1.4 hrs, the soda lime tube was removed, 10 ml. of water was added to the chemical, and the tube replaced. After about an hour, the water concentration output became very stable, dropping very slowly over the next 10 hours.

Controlling Low Flow Rates

When the 6400-01 CO_2 Mixer is installed, flow control is done with a flow diverter (Figure 4-18). The excess flow from this device is dumped into the reference leg, providing faster response times in the reference IRGA.
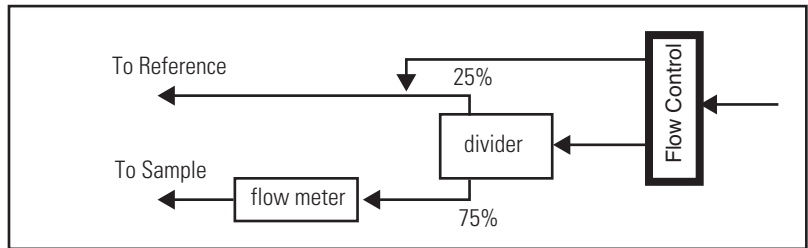
Figure 4-18. Flow schematic for units with a 6400-01 CO_2 mixer. Flow control is done with a diverter valve, and the excess flow is dumped into the reference flow.

(The complete schematic is in Figure 20-19 on page 20-46). However, when the flow control unit is routing <u>most</u> of the flow to the reference cell, there comes a point when some of that flow comes *back* through the flow divider and in fact goes to the sample cell. Typically with the 6400-01 CO_2 Mixer, the lowest attainable flow is about 20 or 30 μmol s^{-1}.

If low flows are needed for some special experiment, this behavior can be prevented quite simply by replacing the "Y" connector with a straight union (Figure 4-19), and letting the flow from the flow controller vent to the atmosphere. This will allow precise flow control down to 0, but at the expense of horrendously slow response times in the reference IRGA at these low rates.
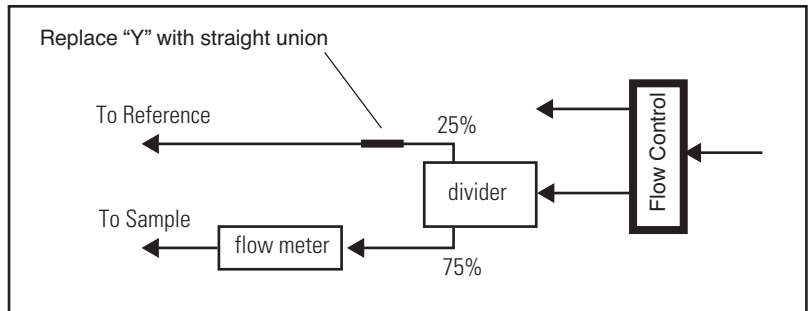
Figure 4-19. To achieve flow control down to zero, replace the T downstream of the diverter with a straight union.

Answers to Questions

Answer 1: Relative humidity is also a function of temperature, so using the coolers to bring down the chamber temperature will increase the *RH_S_%* value (or warming the chamber will decrease it) even though *H2OS_mml* remains unchanged. Conversely, raising the chamber temperature will lower the *RH_S_%*.

Answer 2: Most likely, neither ΔCO_2 nor ΔH_2O changed by a factor of 4. The ΔCO_2 value won't because we are holding the reference CO_2 constant, so decreasing the flow lowers the ambient CO_2 for the leaf, and the photosynthetic rate drops (unless you are on the flat part of a CO_2 response curve, such as C_4 plants at high CO_2). If, however, we had been maintaining a constant sample cell CO_2, we would have seen a 4-fold increase in ΔCO_2, at least until some stomatal changes occurred. The ΔH_2O value, on the other hand, can't increase by 4 because the transpiration rate has to fall as the humidity increases in the chamber.

Answer 3: If the chamber walls were equilibrated at the higher humidity, a drop in humidity will cause water to come off the walls and be added to the air stream. This would make the ΔH_2O too big, and the conductance high.

Answer 4: Both leaf conductance and water sorption affect the water vapor in the chamber, but they have different time scales. Water sorption effects will be most pronounced in the first minute or two after a change in chamber humidity. Stomata usually take many minutes to change. Therefore, apparent conductance changes in the first minute after a large humidity change are mostly due to water sorption; after that, it's probably stomatal change.

Answer 5: It either means the desiccant isn't very good, or the IRGA is not properly zeroed.

Answer 6: Decrease. The incoming air is drier, so the flow rate drops (we're maintaining a constant chamber humidity), so the leaf has more time to remove CO_2 from the air as it passes through the chamber.

Answer 7: Cooling will cause relative humidity to increase, so flow rate will have to increase to compensate.

Answer 8: Warming the air will not directly change the vapor pressure of the air, but it has a profound effect on the *saturation* vapor pressure of the air. Therefore, warming will increase the vapor pressure deficit and lower the rel-

ative humidity. The flow rate will have to *drop* for the system to maintain the vapor pressure deficit and relative humidity. But there's more: there will be increased transpiration (absent stomatal closure) into this drier air, which will dampen the need for decreased flow.

Answer 9: *CO2R_µml* will increase. Here's the sequence: desiccant knob to full scrub, incoming air dries, the humidity controller drops the flow rate, the decreased flow rate causes *CO2S_µml* to drop (if the leaf is photosynthesizing), so the CO_2 controller has to increase *CO2R_µml* to bring *CO2S_µml* back up.

Answer 10: When the light is reduced by half, photosynthesis will drop immediately, causing *CO2S_µml* to increase, so *CO2R_µml* will decrease to keep it on target. Stomatal conductance will initially remain the same, so C_i will increase, but then decrease as the stomata eventually start to close. When this happens, the flow rate will decrease, since we are doing constant water mole fraction control. It can take 10 or more minutes for all this to happen, however.

Answer 11: The sample cell and reference cell have different volumes, and different flow rates through those volumes. Thus, any change in incoming concentration will wash through the two cells at different rates, creating oscillations in the differential. The wildly fluctuating photosynthetic rate isn't real - it's just reflecting this phase difference. After a minute or two, it should stabilize, however.

Answer 12: The soda lime will release water vapor and change the humidity of the air stream. If the desiccant is largely bypassed, then these changes get through to the sample cell, and the humidity control system will respond.

Answer 13: During this brief shade event, photosynthesis will drop, and conductance will not change, so C_i will increase. Once the light returns to normal, the reverse will happen.

Answer 14: The leaf is not consuming CO_2 as fast in reduced light, so the stomata do not have to be so open to take CO_2 in. Water can thus be conserved. How much will stomata close? One notion is that plants tend to operate at constant C_i. This would mean that the stomata would close until the intercellular CO_2 concentration gets back down to where it "belongs".

Answer 15: Operate in fixed flow mode with as high a flow rate as you can, and set the mixer to control on reference CO_2. High flow rates will do three

things for you, two of them good. High flow rate will 1) minimize the difference in the sample cell CO_2 concentration as the photosynthetic rate changes; 2) minimize the time necessary to flush out the leaf chamber, which gives you the best dynamic response; 3) make the humidity in the chamber low. This last feature can be overcome by moistening the incoming air stream. See **Humidifying Incoming Air** on page 4-52, for example.

Standard Tools

5

Trees, Menus, Editors, and File Dialogs

5

Standard Tools

As you use the LI-6400, you will encounter a few interface tools again and again. This chapter fully describes these tools, along with some other ones that you may not see so often.

Tree Views

Tree Views (Figure 5-1) are used extensively for menus and configuration related interfaces.

Closed node

Open node

Use **f1** to open or close a node.
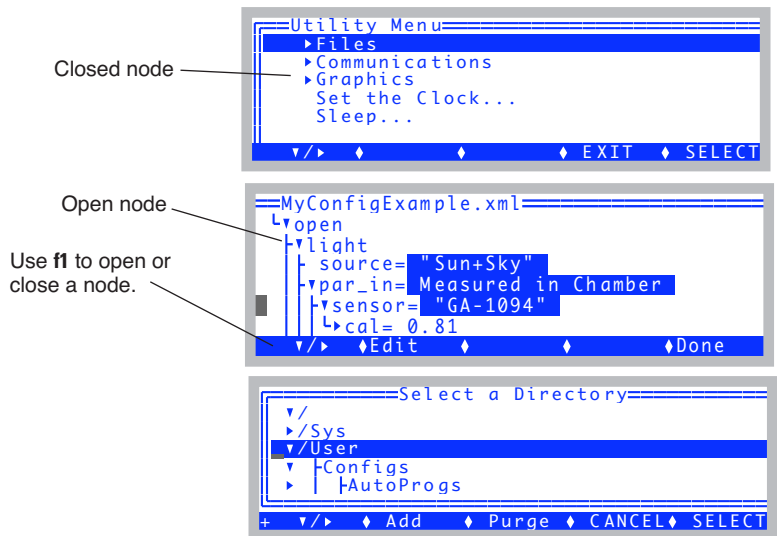
Figure 5-1. Three examples of tree views: The utility Menu, the configuration tree, and the directory tree in the Filer.

The style varies a bit: menus don't show lines on the left; the config tree doesn't use a highlighted bar to mark the cursor, etc. But they all mark nodes with a solid triangle on the left; an open node points down, a closed node points to the right.

Standard Menu

Menus have two basic uses (Figure 5-2):

1 Picking one of a number of possibilities

OPEN's Home Menu is as example. You are presented with a list of options, and you scroll the highlighted bar up or down to select the desired one.

2 Viewing uneditable text

This doesn't look like a menu, because there is no highlighted bar that runs across the display that follows the cursor. Viewing files from the Filer (see **The Filer** on page 10-4) is an example of this use of Standard Menu.

Whichever the use, the function key behavior and cursor key definitions are the same.
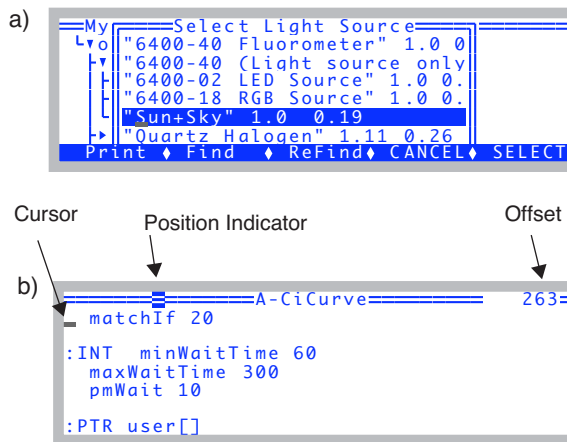
Figure 5-2. Two examples of Standard Menu in action. a) a list of selections with a highlighted bar that the user can move up or down with the arrow keys. b) viewing uneditable text. In this example, there is a banner that includes the file name. To the right is the cursor location, expressed as bytes into the file. The inverse video block moves across the top to reflect the relative location (left = start, right = end) in the file of the cursor.

Function Keys

Standard Menu provides five function key definitions. Whether or not the function key labels remain on-screen depends on how big the menu's window is: if it covers the bottom row of the display, then the function key labels are
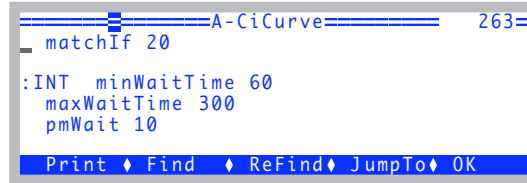
hidden, but can be made to appear by pressing **labels**. A subsequent keystroke (such as an arrow key, or a function key) will make the labels disappear again.

Whether or not the labels are displayed, the function keys themselves are *always* active in Standard Menu.

Figure 5-3 illustrates the function keys, and Table 5-1 describes their meaning.

```
═══════════════A-CiCurve══════════    263═
  matchIf 20

:INT   minWaitTime 60
  maxWaitTime 300
  pmWait 10
    Print ♦ Find   ♦ ReFind♦ JumpTo♦ OK
```

Figure 5-3. Pressing **labels** *causes the function key labels to appear until the next keystroke.*

Table 5-1. Standard Menu function key definitions.

Label	Action
Print	Outputs the contents of the menu to the RS-232 port.
Find	Prompts the user for a string, then searches the menu for the next occurrence of that string.
Refind	Searches for the next occurrence of the search string.
JumpTo[a]	Jump to some byte offset (specified by you) in the file.
CANCEL	Exit the menu without making a selection. (Same as pressing **escape**.)
SELECT	Select the currently highlighted item. (Same as pressing **enter**.)

a. When present, replaces the CANCEL key.

Cursor Keys

The cursor key definitions are given in Table 5-2.

Table 5-2. Standard Menu cursor key definitions

Key	Action
↑ ↓	Moves cursor up or down one line.
← →	Moves cursor back or ahead one character. Moving back will "line-wrap", but not moving ahead.
home	Jump to upper left (first byte of the data).
end	Jump to first character of the last line of the data.
shift home	Jump to start of current line
shift end	Jump to end of current line
pgup	Page up, if text rows outnumber window height.
pgdn	Page down, if text rows outnumber window height.
shift ← →	Scroll display left or right (*regardless of line length of data*, so it's possible to scroll the window blank).
ctrl ← →	Jump to start of previous or next word. This is useful for moving left and right by columns (if data is space delimited)

Standard Line Editor

The Standard Line Editor is used for entering a single line of data, such as a remark in a data file or a numeric value.
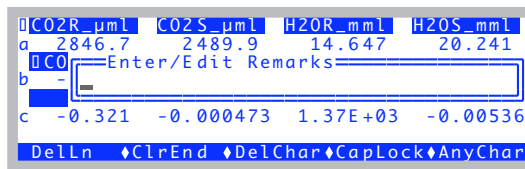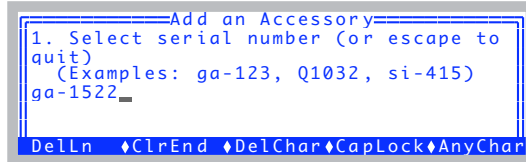
Figure 5-4. The prompt for a remark for logged data is an example of the Standard Line Editor

The cursor may or may not be in a window with a border containing prompts or default values (Figure 5-5).
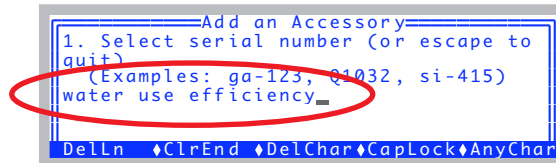
Figure 5-5. The succession of questions asked when adding an accessory use the Standard Line Editor, but without visible windows.

The Recall Buffer

Pressing ↑ while in the line editor will bring back previous entries made anywhere else in the line editor. You can scroll backwards or forwards through the list by pressing ↑ and ↓.

Figure 5-6. The recall buffer will contain entries made anytime the Std Line Editor was used. The entries won't always make sense in any particular context, of course. The same can be said of politicians.

Function Keys

Unless the window covers the bottom line of the display, function key labels will be visible. If the window does obstruct the bottom display line (renaming a file in the Filer will bring about this situation), then the **labels** key can be used to display the function key labels, which disappear again on the next keystroke.

Whether or not the labels are displayed, the function keys themselves (Table 5-1) are *always* active in Standard Line Editor.

Table 5-3. Standard Line Editor function key definitions.

Label	Action
DelLn	Clears the line.
ClrEnd	Clears from the cursor to the end of the line.
DelChar	Deletes the character at the cursor location, but does *not* move the cursor.
Caplock	Toggles caps lock on and off. This applies only to letter keys, not number keys.
AnyChar	Access the AnyChar routine, for generating any key code or character.

Cursor Keys

The cursor key definitions are given in Table 5-2.

Table 5-4. Standard Line Edit cursor key definitions

Key	Action
← →	Moves cursor back or ahead one character.
home	Jump to start of the line.
end	Jump to the end of the line.
shift ← →	Scroll display left or right (*regardless of line length of data*, so it's possible to scroll the window blank).
ctrl ← →	Jump to start of previous or next word. This is useful for moving left and right by columns (if data is space delimited)
↑ ↓	Steps back/ahead through the recall buffer.

AnyChar Routine

When **AnyChar** is pressed, nothing seems to happen. That's because the program is waiting for your next keystroke to decide what to do. You may choose one of the following:

• **Type the 3 digit decimal code for the character desired.**
  For example, 065 will result in A.

- **Press enter to access the character menu**
  This menu (Figure 5-7) allows you to view all the characters in the character set, and select the one you want (Figure 5-7). The character codes are viewable in decimal or hex. Pressing **D** and **X** toggles between decimal and hex modes.

- **Press a non-ascii key (with shift and/or ctrl, if desired)**
  This will generate an escape sequence that specifies the key stroke.

The cursor is a highlighted box that you can move with the arrow keys

The decimal code of the current character. To view hex, press **x**. (**x** and **d** toggle between decimal and hex modes).
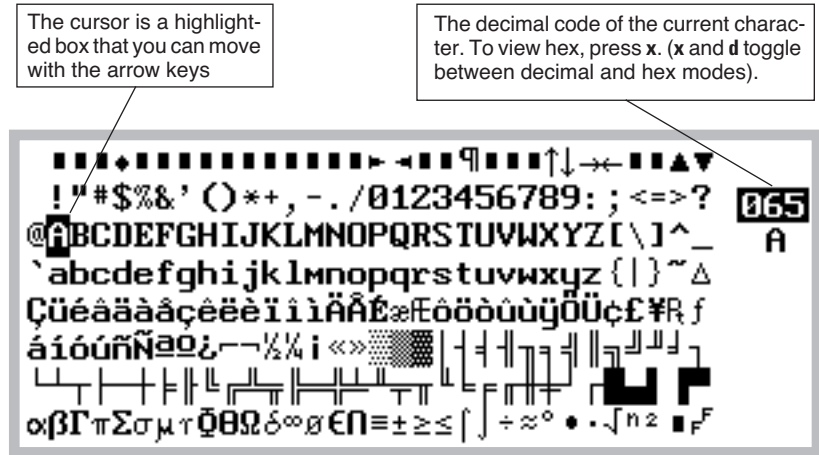
Figure 5-7. The AnyChar routine lets you pick the character you wish to type from a map.

When do you need to use AnyChar?
One example is entering units or a label in which you want the symbol μ (as in μmol, for instance). You can generate a μ by pressing **AnyChar**, followed by **2 3 0**. Or press **AnyChar**, then **enter**, then pick μ from the menu.

Standard File Dialog

In most cases, when you are asked to select a file or enter a file name, the Standard File Dialog is displayed. In this dialog you can select any directory in the file system, and view existing files in any directory. Figure 5-8 shows the dialog boxes for a) selecting existing files (usually for reading), and for b) selecting new or existing files (usually for writing).

Current directory   The prompt

a) For reading

```
Dir: /User/Configs/UserPrefs
===========Select a new configuration====
  DefaultFluorometer_6.0.xml
  FactoryDefault_6.0
  FactoryDefault_6.0.xml
  IRTTest.xml
  junk.xml
+ Dir     ♦        ♦        ♦ CANCEL♦ SELECT
+ More   ♦ Sort   ♦        ♦        ♦
```

File List   2nd level

b) For writing

Entry line, default name

```
Dir: /User/Configs/AutoProgs
File:Data_
=========Log File (esc = none)=======
  A-CiCurve
  AutoLog
  Data
  Data_.xls
+ Dir    ♦ Help  ♦        ♦ CANCEL♦ SELECT
+DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

File List   2nd level

Figure 5-8. The Standard File Dialog for selecting a file name. a) for existing files. b) for new or existing files.

For Reading

When selecting a file for reading (Figure 5-8a), the dialog is essentially a menu. There are two levels of function key definitions (Figure 5-9), and **labels** is used to toggle between the two. Note that this dialog is simply a vari-

ation of the Standard Menu; an extra layer of function key definitions has been added.
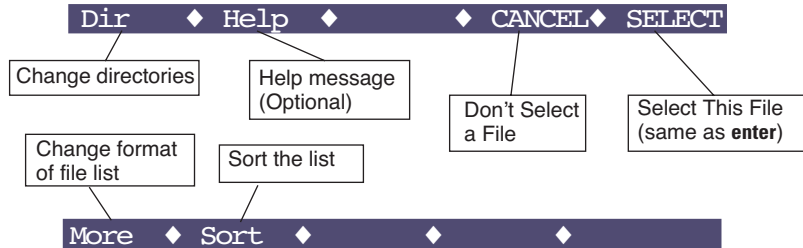
Figure 5-9. Function keys for Standard File Dialogs for existing files.

Manipulating the File List

The second level function keys contain two keys for manipulating the file list. **More** (**f1**, or just press **M**) will toggle what is displayed in the list besides file name: namely modification date and time, and file size (Figure 5-10).
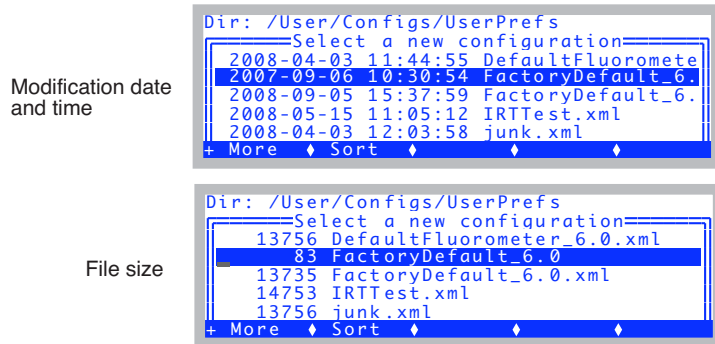
Figure 5-10. f1 (More) toggles alternate file information.

Sort (**f2** or just press **S**) lets you sort the list by Name, Date, or Size, and in Ascending or Descending order.

```
Dir: /User/Configs/UserPrefs
  Sort by: Name, Date, or Size ?_ l
  ================(N/D/S)================
     Dir: /User/Configs/UserPrefs
                                    n
     1 Ascending or Descending?_ .xml
     1 ================(A/D)================
  14753 IRTTest.xml
  13756 junk.xml
```
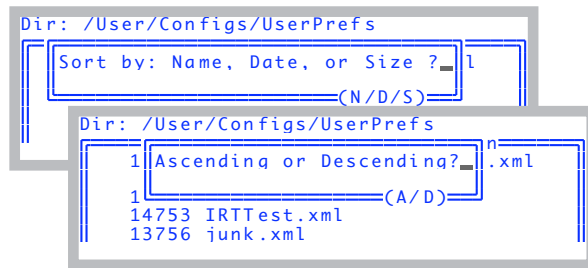
Figure 5-11. The two sorting prompts.

Thus, for example, if you wish to see the newest files in the list, press **S D D**, then **home**, to jump to the top of the list. Note, to actually see the dates, you may have to press **M** once or twice to bring up that display format.

For Writing

When selecting a file for writing, you have the opportunity to type the name of the file. Two levels of function key definitions are available, with the second level being that of the Standard Line Editor (which is used here).

```
Dir      ◆ Help  ◆        ◆ CANCEL◆ SELECT
```

| Change directories | Help message (Optional) | Don't Select a File | Select This File (same as **enter**) |

| Standard Line Editor keys |

```
DelLn   ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```
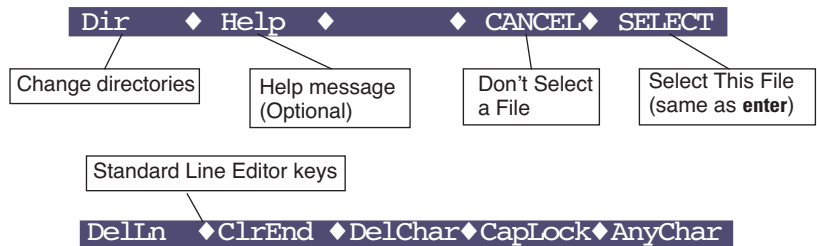
Figure 5-12. Function keys for Standard File Dialogs for new or existing files.

Note that the file list is scrollable with the ↑ ↓ **pgup pgdn**; Any of those keys will cause the focus to jump from the entry line down to the file list (Figure 5-13.

Press ↑ or ↓ or
pgdn or **pgup**...

```
Dir: /User/Configs/AutoProgs
File:Data_
===========Log File (esc = none)=========
  A-CiCurve
  AutoLog
  Data
  Data_.xls
+ Dir    ♦ Help   ♦        ♦ CANCEL♦ SELECT
```

... and focus will
shift down to the
file list.

```
Dir: /User/Configs/AutoProgs
File:Data
===========Log File (esc = none)=========
  A-CiCurve
  AutoLog
  Data
  Data_.xls
+ Dir    ♦ Help   ♦        ♦ CANCEL♦ SELECT
```

2nd level changes ———
```
+ More   ♦ Sort   ♦        ♦        ♦
```
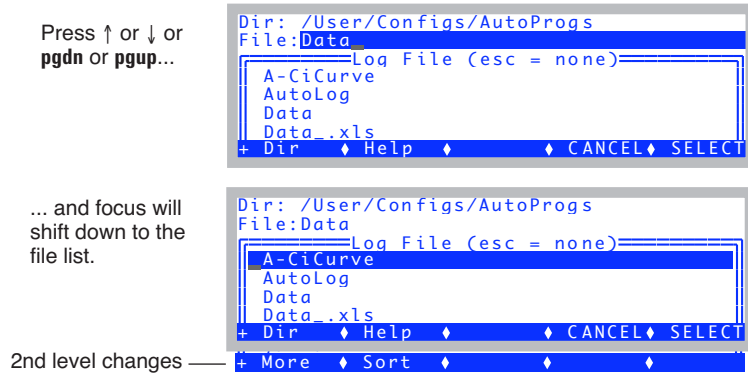
Figure 5-13. To shift focus from the file name entry line down to the file list, press one of the four keys indicated. To get back, use **escape** *or* **enter**.

While the focus is in the file list, you can scroll (↑ ↓ **pgup pgdn home end**), and also manipulate the file list (**Manipulating the File List** on page 5-10). If you wish to copy the name of an existing file up to the entry line, press **enter**. Otherwise, to change the focus back to the entry line, press **escape**.

What if I enter an existing file name?

If you enter a name that already exists, you will be notified when you press **enter** or **Select**.

If appending is allowed

```
File 'Data' exists
Overwrite, Append, or Cancel?

                      (O/A/C)
```

If appending is not allowed

```
File 'Data' exists
Replace it?

                      (Y/N)
```

Figure 5-14. When an existing file is specified (that is different from the default file name), you will be notified with one of these prompts. The message depends on whether appending is permitted.

Pressing **C** (cancel) or **N** (no) or **escape** will return you to the Standard File Dialog so you can modify the file name.

Changing Directories

The items in the file list for either type of Standard File Dialog are the files in the current directory. To change directories and get a new list of files, press **Dir** (**f1**). This will bring up a Standard Menu of all the directories presently in the file system, as illustrated in Figure 5-15.

Expand or
collapse
directories

Make new
directory

Remove a
directory

```
D╔══════════Select a Directory═══════╗
F║ ▼/                                 ║
 ║ ▶/Sys                             ║
 ║▼/User                             ║
 ║ ▼ ┌Configs                        ║
N║ ▶ │ ┌AutoProgs                    ║
 ║                                    ║
+║ ▼/▶   ♦ Add   ♦ Purge ♦ CANCEL♦ SELECT║
 ╚════════════════════════════════════╝
+ Find  ♦fnd Nxt♦ Currnt♦ View   ♦ Rename
```

Find a directory

Jump to
current
directory

Tree view
or full name
view

Rename a
directory

```
D╔══════════Select a Directory═══════╗
F║ ▼/                                 ║
 ║ ▶/Sys                             ║
 ║▼/User                             ║
 ║ ▼/User/Configs                    ║
 ║ ▶/User/Configs/AutoProgs          ║
 ╚════════════════════════════════════╝
+ Find  ♦fnd Nxt♦ Currnt♦ View   ♦ Rename
```
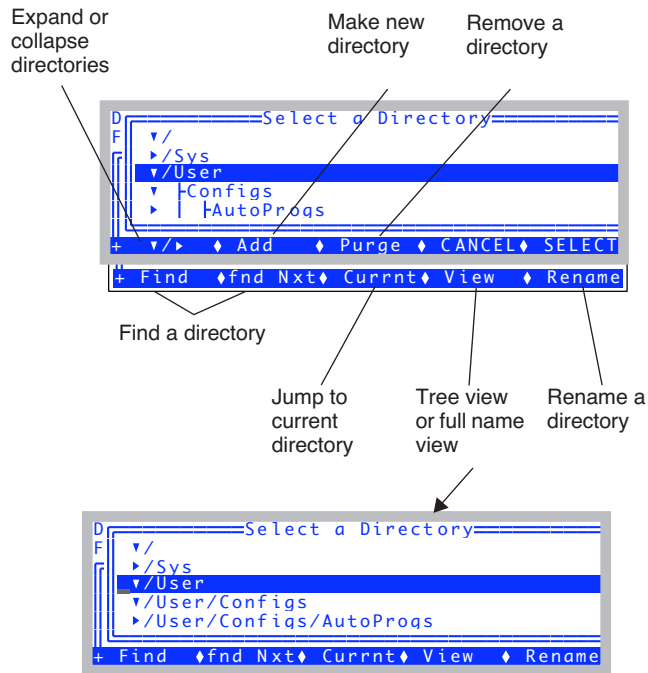
Figure 5-15. Selecting a directory in a Standard File Dialog is done from a Standard Menu accessed by pressing the **Dir** *function key.*

As an example, when you open a data file in New Measurements mode, you will see the Standard File Dialog in Figure 5-8b. Notice that the current directory is listed (/User), followed by the default file name (Data), which can be edited if desired. A list of the files in the /User directory is also shown. To change directories, press **Dir** (**f1**), and then use ↑ or ↓ keys to highlight a different directory. Press **Enter** or **Select** (**f5**) to select the directory; a new list of files in that directory is displayed. (The file system is described in Chapter 10.)

Standard Edit

The Standard Line Editor, described above, is the tool for editing a single line. This is actually a special case of a more general tool, Standard Edit, which is a multiple line text editor. One typically uses Standard Edit for editing files, such as customizing an AutoProgram (if you're clever), or massaging a data file (if you're too clever). For whatever purpose, Standard Edit can be accessed in several ways:

- **From the LPL Copyright Screen**
  As described in **Editing a File** on page 5-20.

- **From the Filer**
  As described in **Viewing and Editing a File's Contents** on page 10-13.

- **From OPEN's Utility Menu**
  Select "Create a new (empty) file..." entry in the Utility Menu of OPEN. You are given an empty buffer to type into, and can name it and store it on exit from the editor.

When the editor runs, the name of the file is displayed on the top line of the display (Figure 5-16).
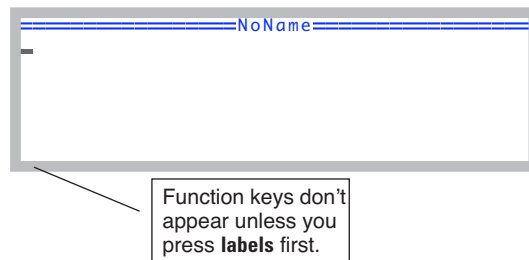
Figure 5-16. Standard Edit shows the file name on the top line (NoName for new files).

Function Key Definitions

In Standard Edit, there are four levels of function keys defined, even though no function key labels normally appear on the display. Key labels only appear after the **labels** key is pressed, and disappear after any other key is pressed. The reason for this behavior is to maximize the view area of the edit window, and not lose a line to key labels.

```
DelLn   ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
BlkSTRT◆BlkEND  ◆BlkCPY ◆BlkMVE ◆BlkDEL
BlkSTRT◆BlkEND  ◆BlkPRNT◆BlkWRIT◆BlkREAD
Print  ◆ JumpTo◆        ◆ Find  ◆ReFind
```

Figure 5-17. The four levels of function keys defined in Standard Edit.

Table 5-5. Function key definitions for Standard Edit.

Label	Action
BlkSTRT	Mark the current cursor location as the start of the selected text block.
BlkEND	Mark the current cursor location as the end of the selected text block.
BlkCPY	Copy the selected text block to the current cursor location, where it is inserted.
BlkMVE	Move the selected text block to the current cursor location, where it is inserted. It is removed from the original location.
BlkDEL	Delete the selected text block.
BlkPRNT	Copy the selected text block to the RS-232 port.
BlkWRT	Store the selected text block as a file, whose name you are asked to enter using Standard File Dialog.
BlkREAD	Insert a file at the current cursor location. You are asked to name the file using Standard File Dialog.
PRINT	Print the current contents of the buffer being edited to the comm port.
JumpTo	Jumps to an offset (in bytes) that you specify.
Find	Search for a target string, which you enter. The search starts at the current cursor location, and goes to the end of the file.
ReFind	Find the next location of the current target.

Cursor Control Keys

Any lines that are longer than the display width are not wrapped, but only the visible portion of the line is shown. The window adjusts as necessary to keep the cursor location visible. The cursor control keys defined in the system editor are the same as for Standard Menu (Table 5-2 on page 5-5).

Tabs

When editing files that have originated in other text editors (as is the case for most of the LPL program files), you may observe lines that begin with small squares rather than spaces, such as this

        ■■■This line starts with 3 tabs

The little squares are tab characters, and Standard Menu and Standard Edit treat tabs like spaces, so ignore them. (If you feel the need to generate a tab character in Standard Edit, press **ctrl i**).

The Exit Menu

When you are done editing, press **escape**, to access the system editor's **Exit Menu**, whose options are listed in Table 5-6.

```
══════════════Exit Menu═══════════════    0═
<esc> - Resume editing
S - Store as new file
Q - Quit editor
```
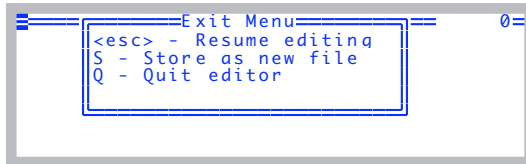
Figure 5-18. Standard Edit's exit menu.

Table 5-6. Standard Edit's exit menu options

Press	Action
escape	Resume editing.
U	(If the file exists and is not write protected) Overwrite the current file (named in the prompt) with the contents of the edit buffer.
S	Store the edit buffer as a new file. Select the name using the Standard File Dialog Box.

Table 5-6. Standard Edit's exit menu options

Press	Action
Q	Quit the editor. If you have made changes, but have not done Update or Store, you will be asked if you wish to abandon the changes.
X	(If the contents appears to be an LPL program) Execute the contents of the edit buffer as an LPL program.

Low Battery Warning

When the LI-6400 battery voltage drops below 11V, a low battery warning appears on the top line of the display, regardless of what application (if any) is running. The message appears every 2 seconds until the voltage rises above 11V, at which time a "Battery OK" message replaces the warning. If the voltage drops below 10.5V, the warning changes to a 60 second countdown, after which the unit is powered OFF. If the batteries are not replaced, or you do not turn off some of the power draining devices (i.e., LED light source, thermoelectric coolers, etc.), the LI-6400 will turn itself OFF when the count reaches zero.

Performing without a net

If you ever need to extend operation a few more minutes, and are facing an imminent shutdown due to a low battery, here is a way to buy some time. From OPEN's main screen, press **K**, and type the following at the ok prompt:

```
0 lowwarn
```

and press **enter**. Then press **escape** to get back to the main screen. This command disables the software low battery warning behavior. The instrument will now operate to about 7 or 8 volts, at which point it will die with no warning. Three notes of caution: 1) There is no guarantee of the time you have gained. It depends on how much current you are drawing from the batteries, and their state. 2) As voltage drops toward the death point, you start running a risk of measurement errors, due to reference voltages shifting. 3) Get those batteries on a charger as soon as you can. Do not let them sit around after pulling them down that far.

To re-enable low battery warnings, enter

```
1 lowwarn
```

at the ok prompt. Also, cycling the power re-enables this warning.

The LPL Screen

The LPL screen (Figure 5-19) represents the user interface to the LPL operating system.

You normally do not see this screen, because the Autostart feature (page 5-23) automatically launches OPEN.
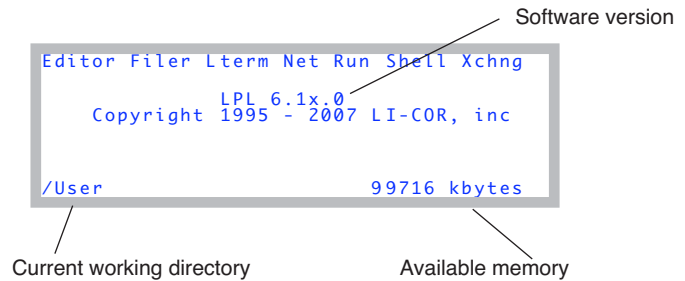
Figure 5-19. The LPL Copyright screen.

Table 5-7 summarizes the LPL Screen's options.

Table 5-7. Tools available in the LPL Copyright screen.

Press...	To Do...
E	Edit an existing file, or to create a new (empty) file. See below.
F	Access the Filer, the file system utility program whose description begins on page 10-4
L	Enable/disable LTerm
N	Runs the Network Status program
R	Run an LPL program.
S	Launch the shell program
X	Enter file exchange mode (for use with the LI6400FileEx, LI6400Sim, or FX/Mac applications) running on an external computer.

Editing a File

You are prompted to enter the name of the file to edit. If you wish to start with a new (empty) file, enter nothing for the file name (press **f1** to clear the entry line if necessary); when you have finished typing in the file's contents, you get to name it on exiting from the editor. If you aren't sure of the name of the file to be edited, you can use the wild card character *. For example, if you enter

```
/User/*
```

you will be shown a list of all the files in the /user directory, as well as any subdirectories, and you can pick a file, or navigate backwards or forwards through the directory tree.

Running a File

LPL program files can be run from the LPL Copyright screen by pressing **R**. The prompt for the file name is similar to that of the Editor, described above, so use of a wildcard will allow file selection from a menu of matches.

LTerm

This toggles LTerm mode, which is used for remote terminal control via RS-232 (as opposed to Ethernet, which doesn't require any special mode). For details, see **Using LI6400XTerm** on page 11-30.

Network Status

Shows the status of the network connection, if any. Details are in **Verify the configuration** on page 11-8.

```
Network Status: IP= 172.24.81.8

 MAC Address: 00:c0:1b:07:f7:b2
    User name: lpl
H) Host name: PSC-1094
P) set Password
<esc> Exit_
```

The Shell Program

Pressing **S** launches the file "/Sys/Lib/StdShell"[1]. When run, this program will appear as in Figure 5-20.
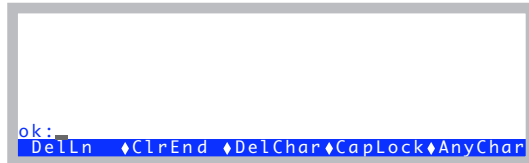
Figure 5-20. The program StdShell prompts for LPL commands or statements, and executes them.

The rules are pretty simple: your one line entry must be a valid LPL function definition, followed by **enter**. Entering a blank line will bring back the previous entry so you can edit it and try again. Pressing **escape** quits the program.

A Calculator
You can use StdShell as a calculator, by typing in the expression to be evaluated, then pressing enter. Use postfix notation, or else begin with a $. For example

```
ok:5.5 2 ^ 3.14159 * print
```

will print 95.0331 on the display, which is the area of a circle of radius 5.5. The same expression, using in-fix notation, could be entered as

```
ok:$ print(3.14159*5.5^2)
```

[1.]Pressing **K** from OPEN's main screen will also launch StdShell.

A Stop Watch

Figure 5-21 illustrates a method to use StdShell to make a stop watch.

Step 1: Type this in.

```
ok:getms getkey drop getms swap - print
 DelLn  ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar
```

Step 2: Press **enter** to start.

Step 3: Press **enter** to stop. The elapsed time (mS) is printed.

```
3055
ok:_
 DelLn  ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar
```

Step 4: Press **enter** to recall the line. Go to Step 2.

```
 3055
ok:getms getkey drop getms swap - print
 DelLn  ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar
```

Figure 5-21. A stop watch example using StdShell.

Power ON Hooks

Between the time the Boot program launches the LPL application, and the time the LPL Copyright screen appears, two things can happen: file exchange mode might be entered, and after that the Autostart files might be run automatically.

File Exchange Mode

Prior to the LPL Copyright screen appearing, if the LI-6400 detects a particular sort of incoming data on the Comm Port, file exchange mode will be entered automatically. The particular sort of incoming data that will do this is the sort that comes from a computer that is running FX and trying to establish communications with the instrument. Specifically, a certain character is looked for: a hex 04.

This is the mechanism that allows you to connect to the computer, run FX, and then turn on the LI-6400 to establish communications.

The Autostart Folder

After the file exchange test, LPL executes in alphabetical order the contents of the folder named /Sys/Autostart. This is the mechanism by which Open is launched when the instrument is powered on.

CheckContrast

Yes, the name starts with a space. This puts it at the top of the alphabetical list, so it executes before any of the others. What it does is set the display contrast to a reasonable value if it is full on or off.

BackLightON

Turns on the display backlight, if present.

InitComm

Sets the comm port baud rate.

Welcome

Does the 5 second countdown, then launches Open.

New Measurements Reference

6

Viewing real time data using text and graphics

New Measurements Reference

New Measurements has three display modes: Text, Graphics, and Diagnostics (Figure 6-1). This chapter discusses all three, plus New Measurement's stability tracking feature.
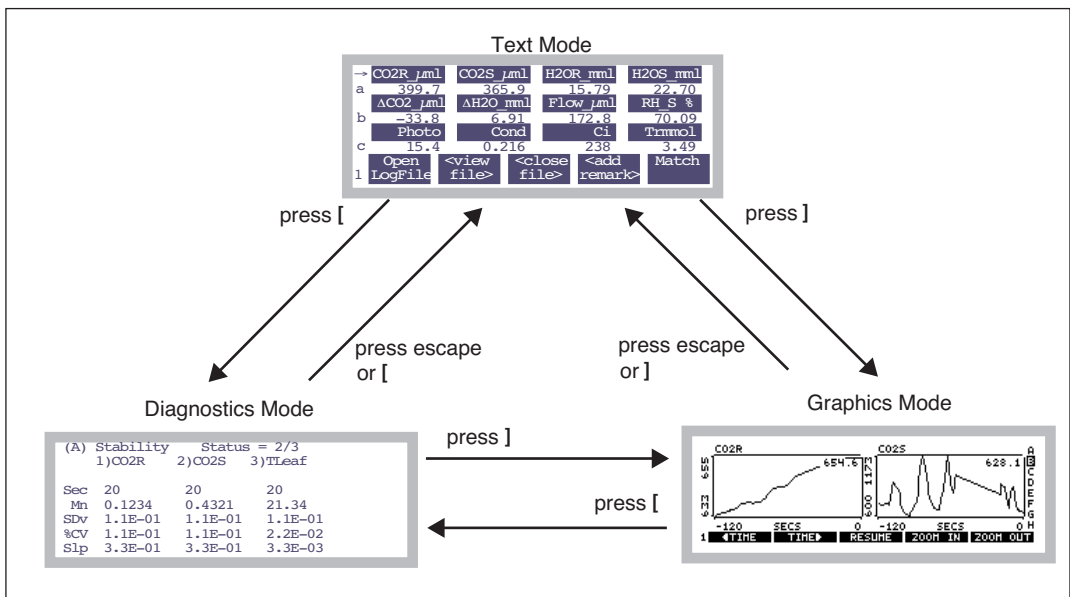
Figure 6-1. Getting around in New Measurements between the three display modes.

Real Time Text

There are 7 (or 10, with the LCF) levels of function keys available in the Real Time Text mode, and up to 12 variables can be viewed at once on the display (Figure 6-2).

Standard Function Key Summary

Table 6-1 lists the standard function keys available in New Measurements mode.

Table 6-1. Summary of function keys, New Measurements mode

Label	Description
1 [Open LogFile] [<view file>] [<close file>] [<add remark>] [Match]	
Open LogFile	Opens a log file, or (if open) adds an observation. See **Open Log-File** on page 9-4.
View File	If a log file is open, allows you to view and graph that data. See Chapter 12.
Close File	Close a log file.
Add Remark	Add a remark to the log file. See **Logging Remarks** on page 9-5.
Match	Enters Match Mode. See **Matching the Analyzers** on page 4-33.
2 [LeafFan Fast] [Flow= 500μms] [Mixer OFF] [Temp OFF] [Lamp= OFF]	
LeafFan	Chamber fan control.
Flow=	Sets flow control mode. See **Humidity Control** on page 7-7.
Mixer=	Sets CO2 mixer control. See **CO2 Control** on page 7-14.
Temp=	Sets chamber cooler control. See **Temperature Control** on page 7-18.
Lamp=	Sets lamp control. See **Light Control** on page 7-20.
3 [AREA= 6.00] [STOMRT= 1.00] [Sys&Usr Consts] [Prompts off] [Prompt All]	
AREA=	Sets leaf area.
STOMRT=	Sets stomatal ratio.

Table 6-1. Summary of function keys, New Measurements mode (Continued)

Label	Description
Sys&Usr Conts	View/edit all relevant system and user constants (even if not in the defined as prompts.
Prompts	Sets prompts behavior (if defined). See **Prompts and Remarks** on page 9-20.
Prompt All	Prompt for all defined prompts. See **Prompts and Remarks** on page 9-20.
4 `GRAPH QuikPik` `View Graph` `GRAPH Setup`	
GRAPH QuikPik	Setup a group of real time graphics screens. See **Real Time Graphics** on page 6-14.
View Graph	View graphics screens. Same as pressing **]**.
GRAPH Setup	Setup one or more graphics screens. See **Real Time Graphics** on page 6-14.
5 `AUTO PROG` `Log Options` `Define Stablty` `Define Log Btn`	
AUTO PROG	Launch an AutoProgram. See **AutoPrograms** on page 9-31.
Log Options	Log Options are discussed in **Log Options** on page 9-14.
Define Stablty	Stability is discussed in **Stability Indicators** on page 6-29.
Define Log Btn	Define the log button behavior. See **User Definable Log Button** on page 9-6.
6 `Display QuikPik` `Display List` `What's What` `Display Editor` `Diag Mode`	
Display QuikPik	See **Display QuikPik** on page 6-5.
Display List	See **Display List** on page 6-6.
What's What	See **What's What** on page 6-6.

Table 6-1. Summary of function keys, New Measurements mode (Continued)

Label	Description
Display Editor	See **Display Editor** on page 6-6.
Diag Mode	Enters Diagnostics Mode, same as pressing **[**.

Text Data Control

The contents of the three display lines can be changed by using ↑ ↓ to position the change line marker, and pressing the desired group code letter. Alternatively, pressing ← or → will cycle through the available choices.
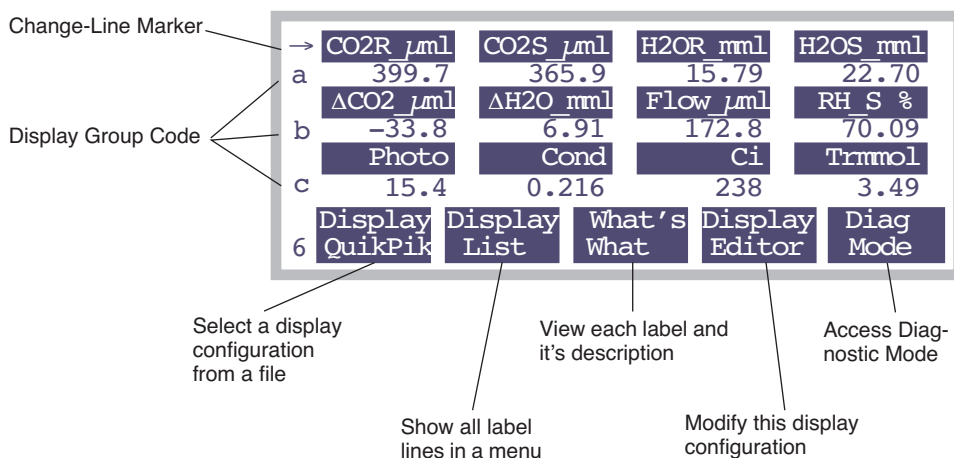
Figure 6-2. New Measurement's text display, showing the Display Function keys (level 6)

New Measurements function key level 6 contains the text display function keys (Figure 6-2). They are:

Diag Mode

One method of accessing Diagnostics Mode, described on page 6-24. The other method is to press **[**.

Display QuikPik

Allows rapid selection of another display format by selecting it via Standard File Dialog (page 5-9) from those that have been stored in the file system in

the directory /User/Configs/Displays. These files are generated by the Display Editor, explained below.

Display List

Brings up a menu showing the list of currently defined label lines.

```
a:CO2R_μml CO2S_μml H2OR_mml H2OS_mml
b:ΔCO2_μml ΔH2O_mml Flow_μml    RH_S_%
c:   Photo     Cond        Ci   Trmmol
d:  BLCond    Ci/Ca       VpdL     VpdA
e:  Stable   StableF       CHF  TotalCV
f:  RH_R_%    RH_S_%   Td_R_°C  Td_S_°C
g:Prss_kPa ParIn_μm ParOutμm
h:Tblock°C    Tair°C   Tleaf°C   CTleaf
```

Figure 6-3. The current list of display labels is shown by the Display List *function,* **f2***, level 6*

This is an instance of Standard Menu (page 5-3), so the cursor control keys and function keys are active.

What's What

Uses Standard Menu to show the list of displayed variables and their descriptions.

```
════════════Displayed Items════════════
CO2R_μml - Ref. CO2 μmol/mol
CO2S_μml - Chmbr. CO2 μmol/mol
H2OR_mml - Ref. H2O mmol/mol
H2OS_mml - Chmbr. H2O mmol/mol
ΔCO2_μml - Ch - Ref CO2 μmol/mol
ΔH2O_mml - Ch - Ref H2O mmol/mol
  Print ♦ Find  ♦ ReFind♦ CANCEL♦ SELECT
```

Figure 6-4. Descriptions of displayed items are shown by the What's What *function,* **f3***, level 6*

This list is generated from the display groups, and is arranged in order of occurrence for groups A to Z, and left to right within a group.

Display Editor

The Display Editor (Figure 6-5) allows you to modify the current display configuration, store it as a file, and/or retrieve a previously stored configuration. It functions as a dialog, so nothing that you do affects the current display configuration if you quit by pressing **cancel**. **OK** implements your changes.

Note: The Display Editor can also be accessed from Config Menu | View/edit, and editing the node <open> <display> <text>,

or <open> <display> <text> <lines>.

Modify current line

Undo any changes you may have made

Clear all items from the line

Copy one or more lines to another line

See **Storing and Retrieving Displays** on page 6-10.
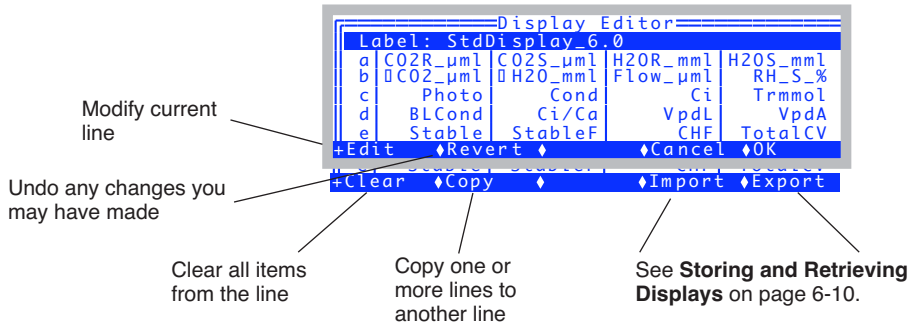
Figure 6-5. The Display Editor allows you to modify the current display list, store it, or retrieve a new one.

The entire list consists of a label, lines a - z, then home, end, pgup, and pgdn. The bottom 4 items are the display groups (Figure 6-6). For example, pressing **home** (in New Measurements mode), sets up displays a, b, and c.

Figure 6-6. The bottom of the list in Display Editor.

The cursor line is highlighted, and is moved up and down by using ↑, ↓, **home**, **end**, **pgdn**, and **pgup**. In addition, you can jump to any defined line simply by pressing the corresponding letter **A** through **Z**.

To edit something, highlight it and press **enter** or **Edit** (**f1** level 1). There are three types of lines in the display definition list (Figure 6-7): You can edit the label line (the label is just for your reference), you can edit any of the lines 'a' through 'z', or you can edit any of the four group lines (pgup, pgdn, home, and end).

Editing the label line

Editing lines a - z (see Figure 6-8)

Editing home, end, pgup, or pgdn
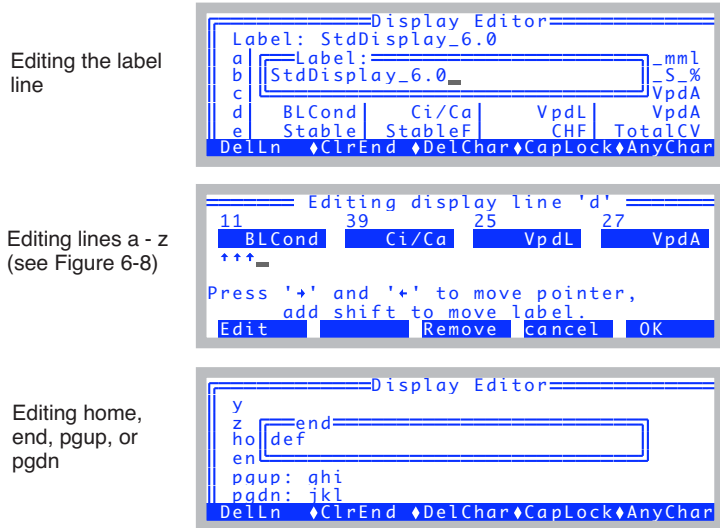
Figure 6-7. Editing the three types of lines in the Display Editor.

The editor for display lines 'a' - 'z' lets you replace individual variables (Figure 6-8), remove them, or shuffle them around (Figure 6-9),

Example: To replace the second item (Ci/Ca), press → to move the pointer, then press **f1** (**Edit**)...
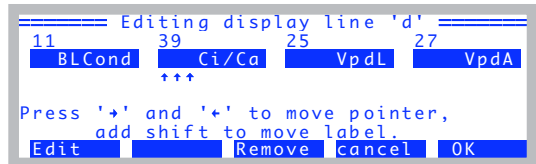
...then pick the new variable from the list.
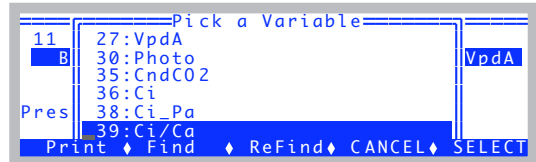
Figure 6-8. Display Line Editor: Replacing a variable.

Incidentally, you can access this line editor from `Config Menu | View/edit`, by editing any node below <open> <display> <text> <lines>.

To move a variable, use **shift** + ← or →. Here, we swap Ci/Ca and Vpdl.
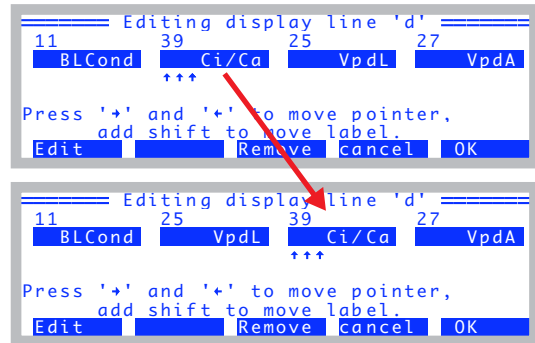
Figure 6-9. Rearranging variables in the Display Line Editor.

Display Groups

You can change all three lines of the text display with a single keystroke. There are 4 keys reserved to do this: **home**, **pgup**, **pgdn**, and **end**. You can set this up in the Display Editor, as we've just shown (Figure 6-7), or "on the fly", in New Measurements mode while viewing live data.

To define (or redefine) a display group "on the fly", get the display arranged the way you want it, then hold the **ctrl** key down and press the group key (**home**, **pgup**, **pgdn**, or **end**) of your choice. That group key is now defined. Once a group key is defined, you can change all three display lines to that arrangement simply by pressing the group key.
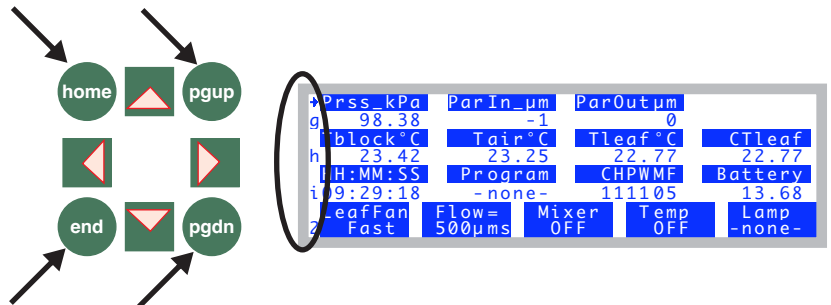
*Figure 6-10. Define a display group by holding the **ctrl** key down and pressing any of the 4 indicated group keys: **home**, **pgup**, **pgdn**, and **end** (on either side of the display). To change all three lines to a previously defined group, simply press that group key.*

Storing and Retrieving Displays

Display information is part of OPEN's configuration tree, and as such, is stored whenever you do `Save as..` in the Config Menu. For most users, that's all you need to know, so you can quit right there.

The Display Editor also lets you store and load display information separately, as a hold-over from earlier versions of OPEN. Separate display information used to be stored in the folder /User/Configs/Displays, and in fact, still can be.

Export

To save display information separately, press **Export** (**f5** level 2) in the Display Editor, to bring up dialog for picking the destination. (Figure 6-11).

```
Dir: /User/Configs/Displays
File:StdDisplay_6.0
=======Save This Display Config=======
 Closed Mode
 Diagnostic
 FlrDisplay_6.0
 mytest
+ Dir     ♦          ♦    ♦ CANCEL♦ SELECT
```
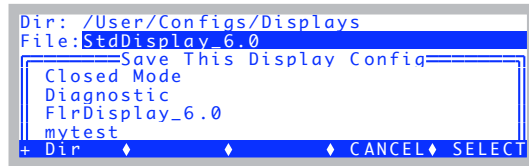
Figure 6-11. Saving a display configuration separate by itself.

The default file name will be whatever is in the Label line.

Import

You can import display information from two sources: 1) Previously exported files in /User/Configs/Displays, or 2) Any configuration in /User/Configs/User-Prefs. Thus, when you press **Import** (**f4** level 2) in the Display Editor, you are shown a tree-like list of possible sources (Figure 6-12).

The two sources

```
┌══Pick one of these══════════════════╗
║  ▸Files in /User/Configs/Displays    ║
║  ▸Files in /User/Configs/UserPrefs   ║
║                                      ║
║                                      ║
║                                      ║
║ ▾/▸    ♦          ♦       ♦ EXIT  ♦ SELECT ║
```
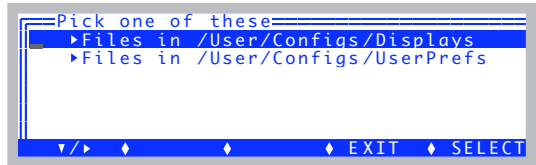
A list of the files in /User/Configs/Displays

```
┌══Pick one of these══════════════════╗
║  ▾Files in /User/Configs/Displays    ║
║     Closed Mode                      ║
║     Diagnostic                       ║
║     FlrDisplay_6.0                    ║
║     StdDisplay_6.0                    ║
║     StdSoilDisplay_6.0                ║
║ ▾/▸    ♦          ♦       ♦ EXIT  ♦ SELECT ║
```
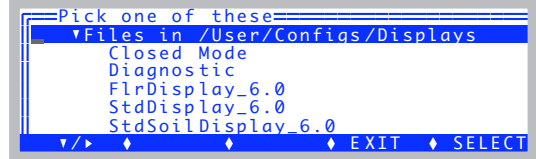
Each file in /User/Configs/UserPrefs that has display information in it is shown here. To see the label, expand the file name node.

```
┌══Pick one of these══════════════════╗
║  ▸Files in /User/Configs/Displays    ║
║  ▾Files in /User/Configs/UserPrefs   ║
║    ▾2x3 LED.xml                       ║
║      Label="My Improved Display"      ║
║    ▸DefaultFluorometer_6.0.xml        ║
║    ▸FactoryDefault_6.0.xml            ║
║ ▾/▸    ♦          ♦       ♦ EXIT  ♦ SELECT ║
```
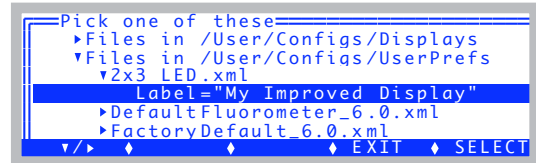
Figure 6-12. The Display Editor can read from either the old style display format files, stored in /User/Configs/Displays, or it can extract the display nodes from the newer .xml config files stored in /User/Configs/UserPrefs.

Exported Display Files

Display files that are stored in the /User/Configs/Displays directory have used the same format that dates back to the Clinton administration, which is illustrated in Figure 6-13. The format used as part of system configurations is shown in Figure 6-14 on page 6-13.

Marker label, number of display lines defined

ID values for line A.

ID values for line B.

ID values for line K

Display group definition (**Display Groups** on page 6-9).

home key
end key
pgup key
pgdn key

```
DISPLAYMAP= 11
-1 -2 -4 -5
-3 -6 -7 -15
30 23 36 21
25 41 42
-71 -72 -73
-14 -15 -16 -17
-11 -12 -13 -32
-8 -9 -10
-21 -22 -23 -24
-31
-22 -56 -57 -52

DISPLAYGROUP=
1 2 3
5 4 6
9 7 3
11 10 2
```
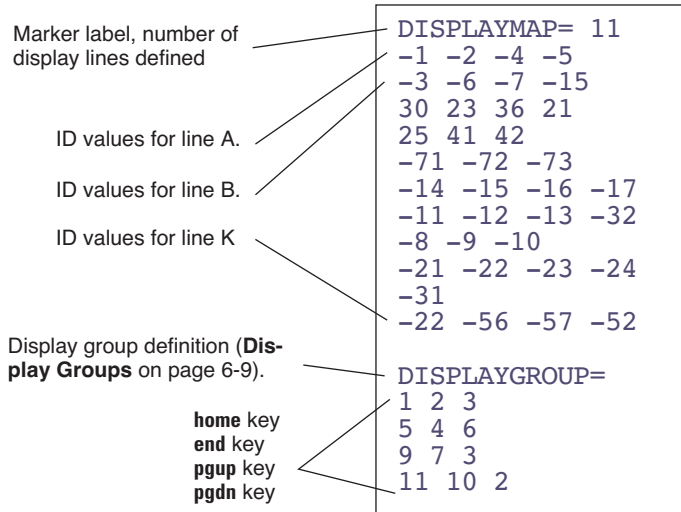
Figure 6-13. Display format files have a marker (DISPLAYMAP=), followed by the number of display lines to be defined. Following this are the ID values of each item. System variables have ID values <0, while user items have positive ID values.

```
<open>
    <display>
        <text>"Std Display"
            <lines>
                <a>{ -1  -2  -4  -5  }</a>
                <b>{ -3  -6  -7  -15  }</b>
                <c>{ 30  23  36  21  }</c>
                <d>{ 11  39  25  27  }</d>
                <e>{ -71  -72  -73  -74  }</e>
                <f>{ -14  -15  -16  -17  }</f>
                <g>{ -11  -12  -13  }</g>
                <h>{ -8  -9  -10  221  }</h>
                <i>{ -21  -22  -23  -24  }</i>
                <j>{ -31  }</j>
                <k>{ -22  -56  -57  -71  }</k>
                <l>{ -48  -49  -50  -51  }</l>
                <m>{ }</m>
                <n>{ }</n>
                <o>{ }</o>
                <p>{ }</p>
                <q>{ }</q>
                <r>{ }</r>
                <s>{ }</s>
                <t>{ }</t>
                <u>{ }</u>
                <v>{ }</v>
                <w>{ }</w>
                <x>{ }</x>
                <y>{ }</y>
                <z>{ }</z>
            </lines>
            <groups>
                <home>"abc"</home>
                <end>"def"</end>
                <pgup>"ghi"</pgup>
                <pgdn>"jkl"</pgdn>
            </groups>
        </text>
    </display>
</open>
```

Figure 6-14. Format used for New Measurments text mode display, when it is part of the overall system configuration file.

Real Time Graphics

Real Time Graphics (RTG) provides a graphical method of monitoring recent activity in New Measurements mode (Figure 6-20). Up to 8 graphics screens can be defined, each holding up to three plots. Plots can be made to scroll horizontally and vertically to keep the curve on-scale.

RTG control is provided with a combination of function keys. In Text mode, level 4, there are 3 function keys for configuring graphs (Figure 6-15).

4   `Import GRAPH`   `View GRAPH`   `Setup GRAPH`

Define all 8 graphics displays by reading their configurations from a file, stored in "/User/Configs/RTG_Defs"

View graphs (same a pressing **]**).

Define individual graphs. See **Defining Graphs** on page 6-14
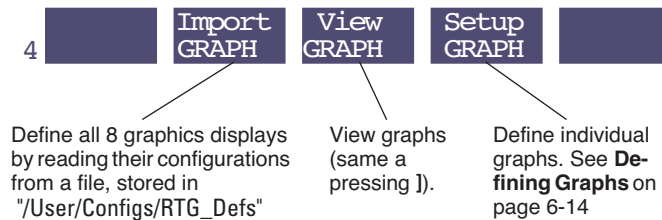
Figure 6-15. The graphics control function keys.

Defining Graphs

Setup GRAPH (**f4** level 4) brings up a list of the 8 RTG screens (Figure 6-16), providing an overview of how they are configured. Note: You can also access this same editor from `Config Menu | View/edit`, by editing the <open> <display> <graphs> node.

From this editor, you can read or save an individual RTG configuration (**Import1** and **Export1**), or read or save them all as a group (**Import**, **Export**). (**Import** does the same thing as **Import GRAPH**, except it doesn't put you in graphics view mode.) Individual definitions (**Import1** and **Export1**) are stored in /User/Configs/RTG_Defs/Graphs, while the group definitions (**Import**, **Export**, and **Import GRAPH**) use either /User/Configs/RTG_Defs (old format), or can extract graphics configurations from the .xml configuration files stored in /User/Configs/UserPrefs.

File formats are described in **RTG Configuration Files** on page 6-20.

See Figure 6-17

Figure 6-23 on page 6-21

```
┌────Real Time Graphics Screens────┐
│  Label: Std Gas Exchange         │
│  A: PHOTO, COND                  │
│  B: CO2R, CO2S                   │
│  C: H2OR, H2OS, Flow             │
│  D: Tblk, Tair, Tleaf            │
├──────────────────────────────────┤
│+Edit    ♦Import1♦Export1♦    ♦Done│
└──────────────────────────────────┘
 +        ♦Import ♦Export ♦     ♦Done
```

Figure 6-22 on page 6-21
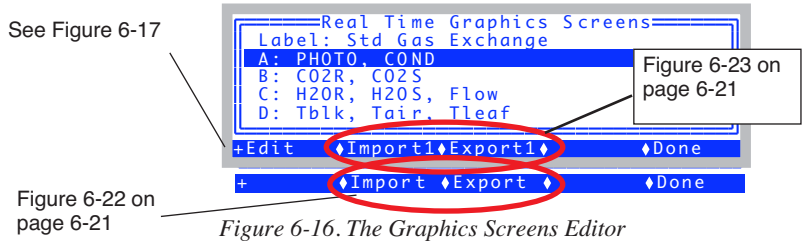
Figure 6-16. The Graphics Screens Editor

To edit a graph, highlight the desired entry and press **Edit** to bring up the Graph Editing Dialog (Figure 6-17).

If a plot is defined

```
┌──────────────Graph A─────────────┐
│ La│ Plot #1 (StripChart) Photo  │
│ A:│ Plot #2 (StripChart) Cond   │
│ B:│ Plot #3 < none >            │
│ C:│                             │
│ D: Tblk, Tair, Tleaf            │
├──────────────────────────────────┤
│ Edit   ♦TurnOff♦Import1♦eXport1♦Done│
└──────────────────────────────────┘
```

If a plot is not defined

```
┌──────────────Graph A─────────────┐
│ La│ Plot #1 (StripChart) Photo  │
│ A:│ Plot #2 (StripChart) Cond   │
│ B:│ Plot #3 < none >            │
│ C:│                             │
│ D: Tblk, Tair, Tleaf            │
├──────────────────────────────────┤
│        ♦TurnON ♦Import1♦     ♦Done│
└──────────────────────────────────┘
```
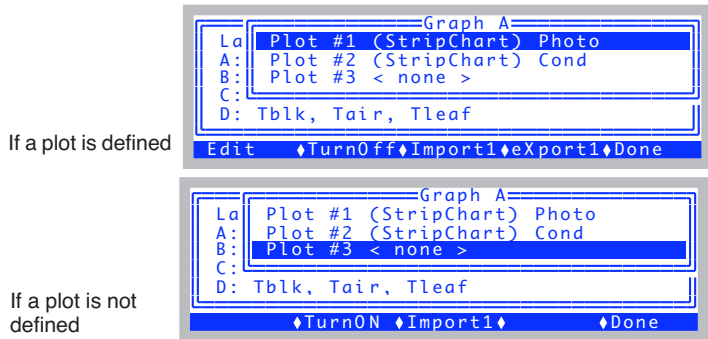
Figure 6-17. Editing a graph.

Up to three plots on a screen can be defined. **Edit** lets you adjust a particular definition (Figure 6-18). **Import1** and **Export1** let you save or retrieve an individual screen definition. **TurnOn** lets you define a plot's type: Strip Chart or XY Chart. To change from one type to another, you must first **TurnOff** the plot, then **TurnOn**.

A plot definition consists of what variable you want on each axis, and how you want it scaled. For strip charts, you have no control over the X axis or its scaling (scaling can be done dynamically from the graphics level 1 function keys, Figure 6-20 on page 6-18), but you can specify how much data is to be retained. The default is 10 minutes.

Editing a Strip Chart
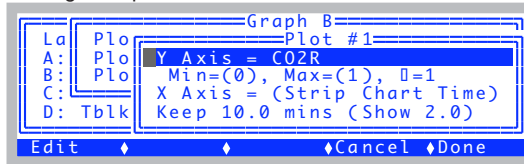
Editing an XY Chart
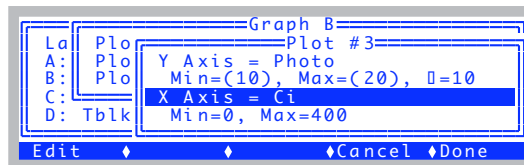
Figure 6-18. Editing a plot definition

There are five scaling options available (Figure 6-19):
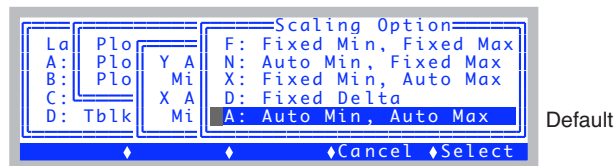
Figure 6-19. Scaling options

- **F: Fixed Min, Fixed Max**
  You specify the minimum and maximum, and they don't change. If data goes off scale, it is not shown. When selected, this option is displayed as

  ```
  Min= 0, Max= 10
  ```

- **N: Auto Min, Fixed Max**
  The maximum value of the axis is fixed, but the minimum varies with the data so that it stays low enough to encompass the displayed data set. You specify the increment by which the minimum can change. When selected, this option is displayed as

  ```
  Min=(0), Max= 10, ∆= 1
  ```

- **X: Fixed Min, Auto Max**
  The minimum value is fixed, but the maximum varies to contain the displayed data set. You specify the increment by which the maximum can change. When selected, this option is displayed as

  ```
  Min=0, Max= (10), Δ= 1
  ```

- **D: Fixed Delta**
  The maximum - minimum difference stays fixed at a value you specify. When selected, this option is displayed as

  ```
  Fixed Delta= 10
  ```

- **A: Auto Min, Auto Max**
  This is the default setting. You specify the increment by which you'd like the maximum and minimum values to change. The axis is then adjusted automatically to contain the displayed data set. When selected, this option is displayed as

  ```
  Min=(0), Max=(10), Δ= 1
  ```

RTG Dynamic Control

Dynamic control of RTG displays is available via three levels of function keys (Figure 6-20).
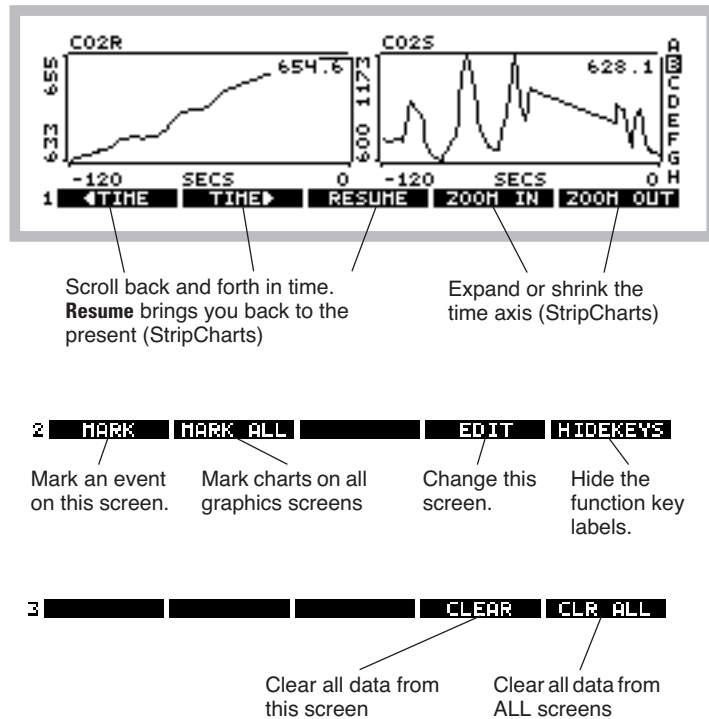
Figure 6-20. The Real Time Graphics function keys

The functions keys are not normally visible. To make them appear (and activate them), press **labels**, or else press **1**, **2**, or **3**. To make them disappear (and become deactivated), press **HIDEKEYS** or **0**.

Changing RTG Screens

The indicator (A-H on the right hand side) shows which screen is presently being viewed. Use ↑ and ↓ or the letters **a** through **h** to change screens.

Changing Time Scales (Strip Charts)

The level 1 function keys control the time axis on strip charts. **◄TIME** scrolls back in time, **TIME►** brings you forward in time, and **RESUME** snaps you back to the present. While scrolled back in time, the real time updates for that plot will stop (although the data is still coming in and retained). After 15 seconds of inactivity, the chart will resume automatically. To expand or contract the time axis, use **ZOOM IN** and **ZOOM OUT**.

Selecting

Press ← or → to select a plot (Figure 6-21). Pressing that key again will select the next plot, and so on, until none is selected.
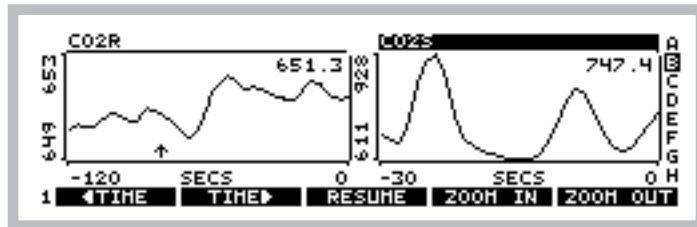
Figure 6-21. A selected plot has its title highlighted.

When a plot is selected, the time scale control keys operate only on that plot. When no plot is selected, they operate on all plots on that screen.

Marking

Strip charts are marked with a small vertical arrow (↑) that stays on the time axis at the time of the marking. XY Charts are marked with a +. Marking happens automatically to all plots on all RTG screens when logging occurs. Charts can also be marked manually without logging by **MARK** or **MARK ALL** on level 2.

MARK marks all plots on that screen, and **MARK ALL** marks all plots on all screens.

Clearing

Plots can have their data cleared by pressing **CLEAR** or **CLR ALL** on level 3.

CLEAR clears all plots on that screen, and **CLR ALL** clears all plots on all screens.
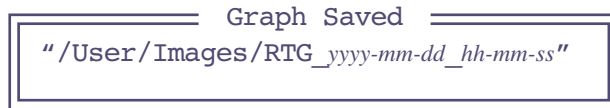
Editing

A graphics screen can be edited directly from graphics mode by pressing **EDIT** on level 2. This will access the editor shown in Figure 6-17 on page 6-15.

RTG Image Files

Anytime you are viewing an RTG screen, you can capture the image that you see by pressing **ctrl** + **s**. This action will save the image in /User/Images in a file named RTG_*yyyy-mm-dd_hh-mm-ss*, where *yyyy-mm-dd_hh-mm-ss* is the date and time of the instant you pressed **ctrl** + **s**. A message will briefly be displayed, which you can clear away early by pressing **enter**.

```
================= Graph Saved =================
 "/User/Images/RTG_yyyy-mm-dd_hh-mm-ss"
```

To view a stored image later, use the program named View Stored Graphics Images in OPEN's Utility Menu.

These files are binary files, so if you transfer them to another computer via RS-232, you should treat them as binary files, not text (**Text vs. Binary Files** on page 11-29).

RTG Configuration Files

Inidividual graphics screen definitions are stored in /User/Configs/RTG_Defs/Graphs, while groups of 8 are stored in /User/Configs/RTG_Defs. An individual screen definition might look like (Figure 6-24 on page 6-22). The group definitions in /User/Configs/RTG_Defs are collections of eight <GRAPH>...</GRAPH> sets.

Anytime you read a group configuration, such as **Graph Import** (**f4** level 4 in New Measurements mode) or **Import** (**f2** level 2 in the Graphics Screen Edi-

tor), you will be shown the two possible sources:

The two sources for graphics configurations...

```
┌─Pick one of these═══════════════════════┐
│ ▶Files in /User/Configs/RTG_Defs        │
│  ▶Files in /User/Configs/UserPrefs       │
│                                          │
│                                          │
│  ▼/▶   ◆        ◆       ◆ EXIT  ◆ SELECT │
```
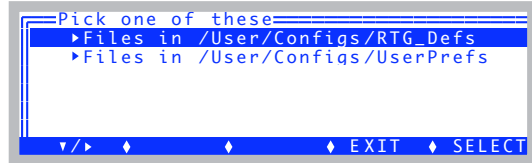
...old format files in /User/Configs/RTG_Defs, or...

```
┌─Pick one of these═══════════════════════┐
│ ▼Files in /User/Configs/RTG_Defs        │
│     Closed Mode                          │
│     Std Fluorescence                     │
│     Std Gas Exchange                     │
│     StdSoilEffluxRTG_6.0                 │
│  ▶Files in /User/Configs/UserPrefs       │
│  ▼/▶   ◆        ◆       ◆ EXIT  ◆ SELECT │
```
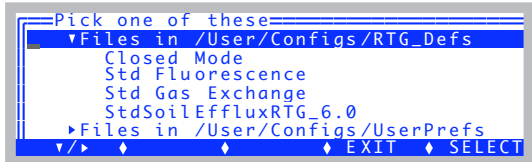
...you can extract graphics information from .xml files in the user preferences directory.

```
┌─Pick one of these═══════════════════════┐
│  ▶Files in /User/Configs/RTG_Defs        │
│ ▼Files in /User/Configs/UserPrefs        │
│    ▼2x3 LED.xml                          │
│      Label="Really exotic graphs"        │
│    ▶DefaultFluorometer_6.0.xml           │
│    ▶FactoryDefault_6.0.xml               │
│  ▼/▶   ◆        ◆       ◆ EXIT  ◆ SELECT │
```
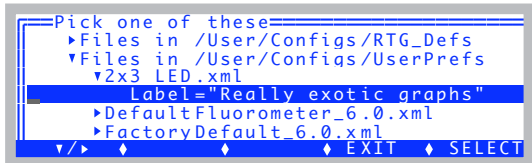
Figure 6-22. Eight-screen graphics definitions are read from one of two places.

When you read an individual screen definition (**Import1** - **f2** in the Graphics Screen Editor, or **Import** - **f3** in the individual screen editor), you are reading a file from the directory /User/Configs/RTG_Defs/Graphs.

```
┌──────────────────────────────────────────┐
│ Dir: /User/Configs/RTG_Defs/Graphs        │
│ ┌════════Read Graph Definition═══════════┐│
│ │ CO2R, CO2S                             ││
│ │ H2OR, H2OS, Flow                       ││
│ │ PHOTO, COND                            ││
│ │ PHOTO-Ci, PHOTO-PARi                   ││
│ │ TBlk, Tair, Tleaf                      ││
│ │+ Dir    ◆        ◆      ◆ CANCEL◆ SELECT│
```
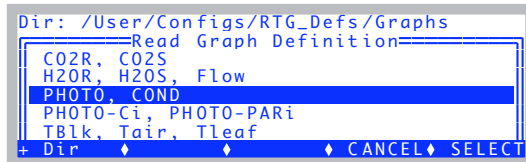
Figure 6-23. Inidivual graphics screen definitions can be stored and read from /User/Configs/RTG_Defs/Graphs.

Graphics configurations for one or more displays that is stored independently in /User/Configs/RTG_Defs/Graphs use a format shown in Figure 6-24. When the information is stored as part of the overall system configuration, the format is a bit different (Figure 6-25).

```
<GRAPH>
    <TITLE>"CO2R, A-Ci"</TITLE>
    <PLOT1>
        <ACTIVE>TRUE</ACTIVE>              ──────  Strip chart
        <STRIP>TRUE</STRIP>            ◄────
        <SHOWTIME>120</SHOWTIME>              ──────  Retain 600 secs of
        <SAVETIME>600</SAVETIME>          ◄────       data
        <YAXIS>
            <VAR>-1</VAR>            ◄─────────  Plot CO2R
            <MIN>0</MIN>
            <MAX>1</MAX>
            <INC>1</INC>
            <SCALE>4</SCALE>
        </YAXIS>
        <XAXIS>
            <VAR>0</VAR>
            <MIN>0</MIN>
            <MAX>1</MAX>
            <INC>1</INC>
            <SCALE>4</SCALE>
        </XAXIS>
    </PLOT1>
    <PLOT2>
        <ACTIVE>TRUE</ACTIVE>
        <STRIP>FALSE</STRIP>        ◄─────────  XY Chart
        <SHOWTIME>120</SHOWTIME>
        <SAVETIME>600</SAVETIME>              ──────  Y axis is PHOTO
        <YAXIS>
            <VAR>30</VAR>            ◄─────────
            <MIN>0</MIN>                       ──────  X axis is Ci
            <MAX>1</MAX>
            <INC>1</INC>
            <SCALE>4</SCALE>
        </YAXIS>
        <XAXIS>
            <VAR>36</VAR>        ◄─────────────
            <MIN>0</MIN>
            <MAX>1</MAX>
            <INC>1</INC>
            <SCALE>4</SCALE>
        </XAXIS>
    </PLOT2>
    <PLOT3>
        <ACTIVE>FALSE</ACTIVE>
    </PLOT3>
</GRAPH>
```

Figure 6-24. Format for storing an RTG screen definition.

```
<open>
     <display>
          <graphs>"Really exotic graphs"
               <a>"PHOTO, COND"
                    <plot1>
                         <active>1 </active>
                         <strip>1 </strip>
                         <showtime>120 </showtime>
                         <savetime>600 </savetime>
                         <yaxis>
                              <var>30 </var>
                              <min>0 </min>
                              <max>1 </max>
                              <inc>1 </inc>
                              <scale>4 </scale>
                         </yaxis>
                         <xaxis>
                              <var>0 </var>
                              <min>0 </min>
                              <max>1 </max>
                              <inc>1 </inc>
                              <scale>4 </scale>
                         </xaxis>
                    </plot1>
                    <plot2>
                         :
                    </plot2>
                    <plot3>
                         :
                    </plot3>
               </a>
               <b>
                    :
               </b>
               :
               <h>
                    :
               </h>
          </graphs>
     </display>
</open>
```

Figure 6-25. Format for graphics display information when it is part of the overall system configuration file.

Diagnostics

The third mode for viewing real time data in New Measurements mode is Diagnostics Mode. Diagnostics mode is entered by pressing **[** from text or graphics mode, or by pressing **Diag Mode** (**f5** level 6 from text mode).

There are 9 screens, labelled A through I.

A: Stability

This display (Figure 6-26) is probably the most useful during normal operations, as it shows the current statistics on the list of variables that you have in your stability list (described in **Stability Indicators** on page 4-41).

```
(A) Stability    Status=2/3
    |1)CO2S* |2)H2OS  |3)Flow

Sec| 15       | 15       | 15
 Mn| 695      | 14.7     | 0.389
SDv| 1.7E+01  | 4.6E-02  | 3.3E-02
%CV| 2.5E+00  | 3.1E-01  | 8.6E+00
Slp|-2.3E+02  |-6.2E-01  |-3.6E-01
```

*Figure 6-26. Diagnostic display A monitors the details of the stability computations. If there are more than 4 quantities in the definition, ← and → will scroll the columns, while **home** and **end** jump to the left and right limits.*

Values that exceed your stability criteria are shown in inverse. If a statistic is not being checked, it is still displayed, but will never be highlighted.

B: Flow Control Status

The flow control status shows the current state of the flow (and humidity) control hardware (Figure 6-27). There's not much here of interest unless you are troubleshooting.

```
(B) Flow Control Status

Pump mv: std=4500, mxr=4600, now=4492
 SetPt = 1320   Null Balance = 0   Rng=2
  Flow = 0mV (0 µmol/s)
DAC offsets: H = 0, F = 0, Lim= 50
Status: LOW (2)
```

Figure 6-27. The flow control status display.

Units with CO_2 mixers have two set points for the pump speed: one with the

mixer (mxr=) and one when it's off (std=). Now= shows the current pump D/A output setting. SetPt= is the flow or humidity control set point. For fixed flow, this is the flow meter's target, and the actual value is on the next line (Flow=). When doing humidity control, SetPt= is the sample cell water IRGA target, and it is shown (H2OS=) instead of Flow=. Sometimes there is an offset in what a D/A is told to do, and what it really does. The DAC offsets line shows the apparent offsets for this unit, one for flow and one for humidity. Lim= is the maximum offset value (mV) that the software is willing to compensate for.

C: CO2 Mixer Control Status

The CO_2 Mixer Control Status display (Figure 6-28) shows if the mixer board is powered, and if the solenoid including the mixer in the flow circuit is activated.

```
(C) CO2 Mixer Control Status

 Power=1 Solenoid = 0
 Setpt = 0 mV
  CO2R = 637.0 ppm
Status: off (0)
```

Figure 6-28. CO_2 Mixer Control Status screen. Power= should always be 1 (if a mixer is installed), but Solenoid= will be 1 only when actually doing CO_2 control. SetPt= is the mixer's target, and CO2R= is the result.

D: Temperature Control Status

The temperature control status screen is shown in Figure 6-29.

Controlling block temperature

```
(D) Temperature Control Status

Cooler=1 Target = 20.00 SetPt = 19.92 C
Tblk = 19.95
On Target
```

Controlling leaf temperature

```
(D) Temperature Control Status

Cooler=1 Target = 24.00 SetPt = 23.42 C
Tleaf = 23.95, d/dt = -0.04 (stable=1)
On Target
```

Figure 6-29. Temperature Control Status screen

E: Lamp Control Status

The Lamp Control status screen is shown in Figure 6-30.

With no light source attached

```
(E) ParIn Status

ParIn = 0 mV (-9 µmol/m2/s)
   Cal = 0.81   Offset = 10 mV
```

With 6400-02(B) LED Source

```
(E) Lamp Control Status

Power = 1
SetPt = 939 mV
ParIn = -1270 mV (999 µmol/m2/s)
   Cal = -0.78   Offset = 10 mV
```

With 6400-18 RGB Source

```
(E) RGB: Connect=1  On=1  FB=1
Status: 15.41V  FAN
 µmol:    433     433     433    1300
% Max:   28.6    46.4    45.8    48.6
ParIn: 1299 umol (1363 mV)
   Cal = 0.95 (Trans=1.00 Offset=0 mV)
Stable
```

With 6400-40 Leaf Chamber Fluorometer

```
(E) LCF Actinic Status

Power  RedSetmV  BlueSetmV  FarSetmV
   1       444        742      -117
BlueCal=-1.85  RedCal=-2.63  Cal=-2.52
ParIn = 538 µmol/m2/s (-212.9 mV)
%Blue = 10.2   Offset = 0.4 mV
Target=500 (Red:450 + Blue:50)
```

*Figure 6-30. Lamp Control Status Screen. The Cal= value is the calibration constant being used for the light source, and of Offset value is the zero offset adjustment, set by the routine in the Config Menu (see **Zeroing the ParIn Signal** on page 18-29). In the case of the LCF, separate red and blue calibration factors exist, so the final value (Cal=) is a weighted average that depends on the fraction of blue light. For the 6400-18 RGB, it is a weighted average of the red, blue, and green fractions.*

F: IRGA Diagnostic Screen

The IRGA Diagnostic Display (Figure 6-31) shows raw millivolt signals, computed concentrations, and the temperatures and pressure that also go into the calculations. (Figure 6-31).

```
(F) IRGA Diagnostic Display
____ _____mV _____ppx _____AGC _ok
CO2R     24.1     5.97    732.1    1
CO2S   1829.4   622.88   2312.9    1
H2OR   1914.5    22.996 -1012.8    1
H2OS   1738.1    19.759   141.6    1
Tirga=26.49 Tcham=24.27 Tblk=24.43
Pressure = 97.30 kPa (1559.1 mV)
```

Figure 6-31. IRGA Diagnostic Screen. ppx means ppm or ppt - that is, $\mu mol\ mol^{-1}$ for CO_2, or $mmol\ mol^{-1}$ for H_2O

This is a valuable screen for troubleshooting. See page 20-14.

G: System Diagnostic Screen

The System Diagnostic Screen is shown in Figure 6-32.

```
(G) System Diagnostic Display

Thr Sep 11 2008 10:15:41
LPL Version: 6.1x.0
OPEN Version: 6.1x2
Available Memory = 99663872 bytes
Stack Items = 4
```

Figure 6-32. System Diagnostic Display shows version numbers and available memory.

H: User and Sys Variables

The User and Sys Variable screen shows all variables in a list you can scroll through. The list is updated twice per second.

```
(H) User and Sys Variables
10 fda= 0.00123
11 BLCond= 2.84
20 Trans= 2.6E-06
21 Trmmol= 0.0026
22 CndTotal= 0.000227
23 Cond= 0.000227
24 vp_kPa= 1.7
```

I: Network Status

The network status display is shown in Figure 6-33.

```
(I) Network Status: IP= 172.24.81.8

   User name: lpl
   Host name: PSC-1094
    #Clients: 1
```

Figure 6-33. The Network Status diagnostic display. #Clients is the number of remote terminal clients at the moment (Remote Control on page 11-29

Stability Indicators

OPEN allows you to set up stability criteria based on measured and computed variables, to determine when the system is stable. For each variable chosen, stability can be based on statistics (any combination of standard deviation, rate of change, and coefficient of variation) over a time period of your choosing. (For computational details, see **Stability Variables** on page 14-13.)

Defining Stability

The Stability Dialog (Figure 6-34) is accessed by pressing **Define Stablty** (**f4** level 5) in New Measurements mode. It can also be accessed directly from some AutoPrograms during set up by responding **N** to the prompt

```
Current Stability Definition
uses n variables. OK? (Y/N)
```

The Stability Dialog lists the variables currently in the stability list. All user variables, and all floating point system variables (e.g. no strings) are available for inclusion in the stability list. For each variable in the list, the mean, standard deviation, coefficient of variation, and rate of change with time is continuously tracked while in New Measurements mode.

The Stability Dialog also lets you specify what combination of statistics should be considered in order to determine system stability. The current criteria may be viewed in the Stability Dialog's main screen. If no stability statistics are being checked, the label no is shown.

Add or remove variables

```
═══════════Stability Editor═══════════
Label: Std Stability
█1: CO2S (-2) 15 Slp<1
 2: H2OS (-5) 15 Slp<1
 3: Flow (-7) 15 Slp<1

+Edit    ♦Add    ♦Remove ♦Cancel ♦OK
+         ♦      ♦       ♦Import ♦Export
```

Editing the highlighted entry produces this dialog

Stability definitions can be stored and recalled separately from the overall system configuration. See **Saving Stability Definitions** on page 6-32.

```
═════════Stability Editor═════════
══════Edit Stability Item══════
█Variable: CO2S
 Time(s): 15
 StdDev: no
 CV(%): no
 slope: <1
Edit  ♦      ♦      ♦      ♦Done
```

Edit Variable:The variable is selected from a list of user and system variables.

```
═════════Pick Variable═════════
        █-02 CO2S
Variab  -03 DCO2
Time(   -04 H2OR
StdD    -05 H2OS
CV(     -06 DH2O
slo     -07 Flow
        ♦      ♦      ♦Cancel ♦Select
```

Edit Time(s): The time period for the running statistics

```
═════════Stability Editor═════════
══════Edit Stability Item══════
V ┌─Running stats over __ secs┐
  │─15─────────────────────────U
   CV(%): no
   slope: <1
DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Each of the statistical items (std dev, cv, and rate of change) has a 2 part edit: The first ("Check Y/N") determines if that statistic is to be examined for determining stability. If you specify Yes, then you are asked for the threshold value.

```
═════════Stability Editor═════════
                         tem═══
Variab┌Check Std Dev ?┐
Time( │               │
StdD  └═══════(Y/N)═══┘
CV(%): no
```

```
═════════Stability Editor═════════
══════Edit Stability Item══════
V ┌─Std dev < what value─┐
  █─1─────────────────────U
   CV(%): no
   slope: <1
DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Figure 6-34. The Stability Dialog.

Viewing Stability Status

There are three summary variables you can monitor or log that provide system stability information, and they are listed in Table 6-2.

Table 6-2. The stability indicators.

Variable	ID	Example	Description
Stable	-71	" 4/5 "	#stable / total # as a string
StableF	-72	0.80	#stable / total # as a value
\<Initials>	-73	01111	Who's stable and who isn't. The label consists of the first letter of each variable checked. 0 means unstable, and 1 means stable.
TotalCV	-74	0.543	Sum of the CVs, in %.

If a variable is in the list, but has no stability thresholds (that is, has `<don't care>` for all the statistical entries), it will always be considered stable.

The real time values of all of these statistics are available for viewing in Diagnostics Screen A (Figure 6-35).
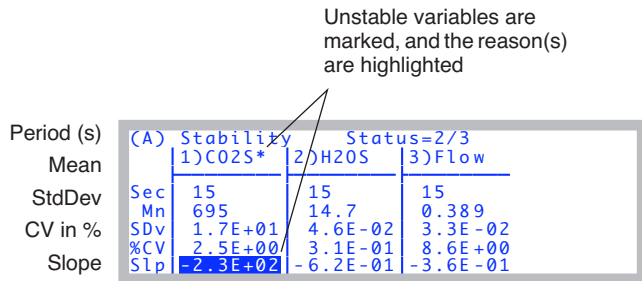
Figure 6-35. The stability diagnostic display.

The display will update each second with the latest values. If more than 4 variables are in the stability definition, the list can be scrolled left and right by pressing → or ←.

Saving Stability Definitions

Stability definitions can be automatically stored as part of the entire system configuration. Thus, the "`Open...`" and "`Save as...`" entries in the Config Menu take care of this.

Stability definitions can also be saved or retrieved separately by using the Stability Editor's **Import** and **Export** keys (Figure 6-34 on page 6-30). Separate stability definitions (e.g. ones created with versions of software before 6.1) normally live in the /User/Configs/StableDefs directory.

Reading a Stability Definition

When you press **Import** (**f4** level 2) in the Stability Editor, you get to select a stability definition from one that was exported, or one that is contained within the system configurations.

The two sources of stability definitions:

```
═══Pick one of these══════════
  ▸Files in /User/Configs/StableDefs/
  ▸Files in /User/Configs/UserPrefs

  ▾/▸    ♦       ♦      ♦ EXIT  ♦ SELECT
```
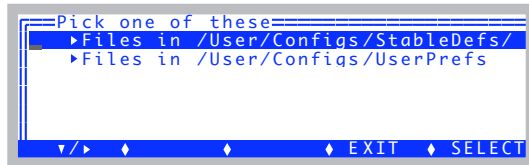
...old format defs in "/User/Configs/StableDefs".

```
═══Pick one of these══════════
  ▾Files in /User/Configs/StableDefs/
      Flr Stability
      Std Stability
      duh...
  ▸Files in /User/Configs/UserPrefs
  ▾/▸    ♦       ♦      ♦ EXIT  ♦ SELECT
```
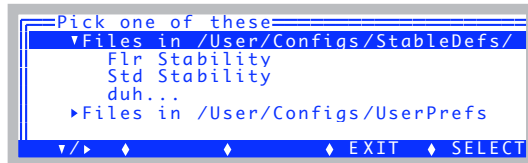
...stability defs found in system configurations (.xml files in "/User/Configs/UserPrefs").

```
═══Pick one of these══════════
  ▸Files in /User/Configs/StableDefs/
  ▾Files in /User/Configs/UserPrefs
    ▾2x3 LED.xml
        Label="My world-class stability
    ▸DefaultFluorometer_6.0.xml
    ▸FactoryDefault_6.0.xml
  ▾/▸    ♦       ♦      ♦ EXIT  ♦ SELECT
```
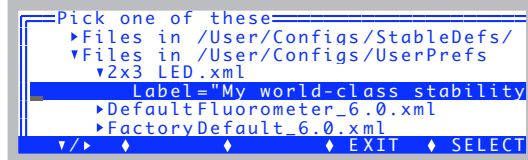
Figure 6-36. You can import a stability definition from one of two places.

Exporting a definition

If you press **Export** (**f5** level 2) in the Stability Editor, you can export the current stability to a file in /User/Configs/StableDefs.

```
Dir: /User/Configs/StableDefs
File:My Stab Def_
═════Save stability definition═════
│ duh...
│ Flr Stability
│ Std Stability
│
+ Dir      ♦         ♦        ♦ CANCEL♦ SELECT
```
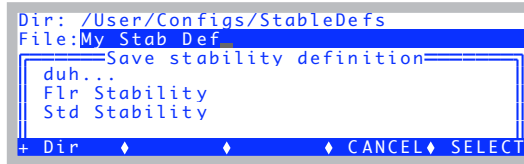
Figure 6-37. Exporting a stability definition.

The format of stability definitions changed with version 6.1. The old format is shown in Figure 6-38, and the new format is shown in Figure 6-39.

	`<STAT TRACKER>`
	`    <TITLE>"MyDef"</TITLE>`
	`    <ITEM>`
30 = Photo	`        <ID>30</ID>`
20 seconds	`        <SIZE>20</SIZE>`
Don't check %CV	`        <PCV_CHECK>FALSE</PCV_CHECK>`
	`        <PCV_VALUE>1</PCV_VALUE>`
Do check slope	`        <SLP_CHECK>TRUE</SLP_CHECK>`
	`        <SLP_VALUE>0.5</SLP_VALUE>`
Don't check std dev	`        <SDV_CHECK>FALSE</SDV_CHECK>`
	`        <SDV_VALUE>1</SDV_VALUE>`
	`    </ITEM>`
	`    <ITEM>`
23 = Cond	`        <ID>23</ID>`
20 seconds	`        <SIZE>20</SIZE>`
Don't check %CV	`        <PCV_CHECK>FALSE</PCV_CHECK>`
	`        <PCV_VALUE>1</PCV_VALUE>`
Do check slope	`        <SLP_CHECK>TRUE</SLP_CHECK>`
	`        <SLP_VALUE>0.01</SLP_VALUE>`
Don't check std dev	`        <SDV_CHECK>FALSE</SDV_CHECK>`
	`        <SDV_VALUE>1</SDV_VALUE>`
	`    </ITEM>`
	`</STAT TRACKER>`

Figure 6-38. Listing of a typical stability definition file (Old Format).

```
<open>
     <stability>
          <items>"MyDef"
               <items[1]>
                    <id>30 </id>
                    <size>20 </size>
                    <pcv>
                         <onoff>0 </onoff>
                         <value>1 </value>
                    </pcv>
                    <std>
                         <onoff>0 </onoff>
                         <value>1 </value>
                    </std>
                    <slp>
                         <onoff>1 </onoff>
                         <value>0.5 </value>
                    </slp>
               </items[1]>
               <items[2]>
                    <id>23 </id>
                    <size>20 </size>
                    <pcv>
                         <onoff>0 </onoff>
                         <value>1 </value>
                    </pcv>
                    <std>
                         <onoff>0 </onoff>
                         <value>1 </value>
                    </std>
                    <slp>
                         <onoff>1 </onoff>
                         <value>0.001 </value>
                    </slp>
               </items[2]>
          </items>
     </stability>
</open>
```

30 = Photo
20 seconds
Don't check %CV

Don't check std dev

Do check slope

23 = Cond
20 seconds
Don't check %CV

Don't check std dev

Do check slope

Figure 6-39. The version 6 format for stability definitions.

Keyboard Summary

New Measurement - specific control shortcuts are shown in Table 6-3.

Table 6-3. New Measurements key summary.

Press...	To Do...
[	Enter/Exit Diagnostics Mode
]	Enter/Exit Graphics Mode
a..z	(Text) Selects display line. (Diagnostics) Selects display screen (a - i). (Graphics) Selects graphics screen (a - h)
0..9	(Text and Graphics) Selects function key level.
home **end** **pgup** **pgdn**	(Text) Implement a Display Group key.
ctrl home **ctrl end** **ctrl pgup** **ctrl pgdn**	(Text) Define a Display Group key.
↑ ↓	(Text) Select new display line (Graphics) Select new display screen
← →	(Text) Change to next display on selected line. (Graphics) Change chart selection
ctrl z	Toggle warning messages on/off.
ctrl s	(Graphics) Store current graphics display to /User/Images/RTG_*yyyy-mm-dd_hh-mm-ss*. (Text & Diagnostics) Store a system snapshot (variables, values, settings, etc.) to a text file names /User/Snapshot_*yyyy-mm-dd_hh-mm-ss*.

.

Environential Control

How OPEN controls chamber conditions

7
Environmental Control

One of the things that makes the LI-6400 a powerful system is its ability to control conditions in the leaf chamber. A clear understanding of how this happens is vital to proper use of the system. This chapter discusses the mechanisms and options for controlling chamber conditions, and goes along with **Tour #3: Controlling Chamber Conditions** on page 3-28, which provides some step by step experiments designed to give you a feel for using these controls.

OPEN's Control Manager

In OPEN's New Measurements mode, the environmental control function keys (Figure 7-1) allow control of fan speed, humidity (using flow rate), CO_2 (with optional 6400-01 CO_2 Mixer), temperature, and light (with optional 6400-02B LED Source, or 6400-18 RGB, or 6400-40 LCF).

Fan	Flow/Humidity	CO_2	Temperature	Light

| LeafFan | Flow→ | *CO2R→ | Temp | Lamp |
| 2 Fast | 500 µm | 400µml | OFF | –none– |
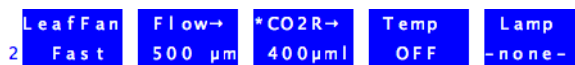
Figure 7-1. New Measurement's level 2 function keys contain the environmental controls.

Four of the five control areas represented by the key labels above share a common control manager. The control manager is simply the software that defines the user interface and how and when the user's selections are implemented.

Interface Fundamentals

Each control key label reflects the type of control, the current target, and the stability status. Pressing one of the four function keys brings up a configuration dialog for that control. The flow/humidity key (**f2**), for example, is shown in Figure 7-2.

```
┌─────Environmental Controls═══════════════┐
│ ▼FLOW:  Flow Rate                         │
│→├ Control= Flow_µm                        │
│ └ Target= 500 µmol/s                      │
│                                           │
│                                           │
│══T) reTarget, exit    O) Off, exit════════│
│+   ▼/▶  ◆Edit    ◆      ◆cancel ◆ Keep    │
└───────────────────────────────────────────┘

          ┌══════Flow Control Types══════┐
          │ Off                           │
          │ ◆ Flow Rate                   │
          │ H2O mole fraction             │
          │ Relative Humidity             │
          │ Vapor Pressure Deficit        │
          └───────────────────────────────┘

┌═════Environmental Controls════════════════┐
│ ▼FLOW:  Vapor Pressure Deficit            │
│→├▼Control= VPD                            │
│  └ Temperature basis: Tleaf°C             │
│ ├ Target= 1.00 kPa                        │
│ └ Response= "fast"                        │
│══T) reTarget, exit    O) Off, exit════════│
│+   ▼/▶  ◆Edit    ◆      ◆cancel ◆ Keep    │
└───────────────────────────────────────────┘
```

Figure 7-2. A typical control screen. There are usually two nodes: Control (type) and Target. Some control types may have other nodes, which represent options specific for that control type, as illustrated above by the additional nodes when Control is VPD.

Normal procedure is to edit the appropriate nodes in the tree, and when it is to your liking, press **Keep** (**f5**) to implement the new control settings, or **cancel** (**f4**) to exit and leave the control unchanged.

Shortcuts

There are two shortcuts; **T** and **0**. Pressing **T** will immediately edit the target value (Figure 7-3) and, unless you press **escape** to cancel, will then exit and implement the new settings. Pressing **0** (as in Off) does the equivalent of editing the Control= node, selecting the Off option, and then pressing **Keep**.

Editing the Target

When you edit a Target= node, the screen appears as in Figure 7-3.
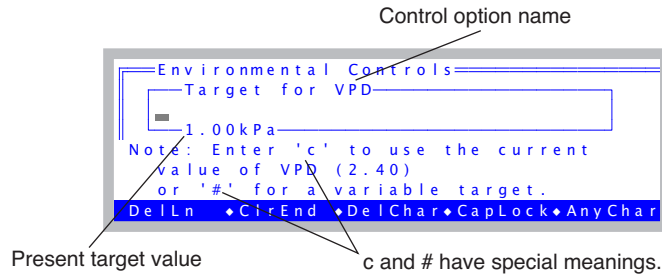
Figure 7-3. The prompt for a target value.

A blank entry (just press **enter**) will retain the current value.

If you enter the letter **c**, the target will be set to the current value of the appropriate variable.

If you wish to enter a variable target, use the character **#**, either by itself, or with an ID value (e.g. **#20**). See **Variable Targets** on page 7-5.

Behind The Scenes

Once the user has specified a control and target, the control manager sets the appropriate hardware. The system then begins a regime of periodically checking the stability status and fine tuning the control as needed.

Each control option has three software components: a fast component that executes every 3 seconds, a medium component that executes every 10 seconds, and a slow component that executes every 30 seconds. These components are called interrupt service routines, or ISRs. A particular control may not actually use all three ISRs; this depends upon how much work is being done by the hardware, and on the response time of the quantity being controlled. ISRs execute at regular intervals while the control manager is active.

For example, the LED light source responds nearly instantaneously to control changes, so fine tuning adjustments to keep it at a target value can be done with the 3 second ISR, while the 10 and 30 second ISRs do nothing. Controlling leaf temperature, however, is a much slower process, so the 10 and 30 second ISRs are needed.

Active vs. Inactive Control Manager

The control manager is *inactive* when it is not doing the fine tuning necessary for target tracking. The consequences of this depend on the control. Some controls are fully implemented in hardware, leaving nothing important for any of the three ISRs to do[1]. Usually, however, the ISRs *are* important, and so it is important to know when the control manager is active, and when it is not.

When is the Control Manager Active?

The control manager is active in New Measurements mode (which includes AutoProgram activity), *except* during IRGA matching.

The practical consequences of an inactive control manager are minor, at least over short periods of time. Some controls are implemented in hardware (such as block temperature control and null balance water vapor control), so even without the control manager, they will still actively track. The others will simply have stable targets that won't be adjusted to account for changing temperature. Specifics can be found in the ISR descriptions for each control described below.

Variable Targets

Any control mode whose target entry box includes a message about a pound sign (#) after the default label (e.g. Figure 7-3 on page 7-4) can have a variable target instead of a fixed value target. For example, you may wish to maintain a relative chamber humidity that is the same as the humidity measured with an external humidity sensor. Rather than entering a number for the target, enter a pound sign

    #

and press **enter**. You will be shown a list of system and user defined variables, from which you can pick the quantity that is to be tracked.

Alternatively, if you know the ID number for the system variable or user variable, you can enter it directly and bypass the menu selection.

    #–13

[1.]A trivial example is turning off the light source; once it's off, there's no further adjustment necessary. A not-so-trivial example is fixed flow mode, which maintains itself independent of software because there is hardware to do the job.

(Variable number -13 is the external quantum sensor, for example.) When a variable target is specified, the target is updated every 30 seconds, but only while the control manager is active.

■ **Example: Tracking External Humidity**

Suppose we want to make chamber humidity track a value[2] provided by user channel #1001 named *xtrnRH*. Once this is all configured (Chapter 15), we can press **2 f2** (i.e. **f2** in level 2) to select the flow/humidity control, set Control to Relative Humidity, edit the target and enter **#1001 enter** (Figure 7-4). Or, you could just enter **#** and pick *xtrnRH* from the menu.

Enter # followed by the variable ID.

Or, just enter #....

```
══Environmental  Controls══
 ┌─Target  for  RH_S_%──────
 │ #1001▮
 └──50%▮
Note: Enter  'c'  to  use  the  current
   value  of  RH_S_%  (35)
   or  '#'  for  a  variable  target.
DelLn   ♦ClrEnd   ♦DelChar♦CapLock♦AnyChar
```
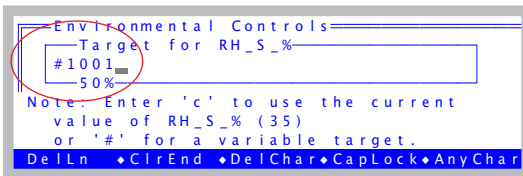
```
══Environmental  Controls══
 ┌─Target  for  RH_S_%──────
 │ #▮
 └──35%▮
Note: Enter  'c'  to  use  the  current
   value  of  RH_S_%  (35)
   or  '#'  for  a  variable  target.
DelLn   ♦ClrEnd   ♦DelChar♦CapLock♦AnyChar
```
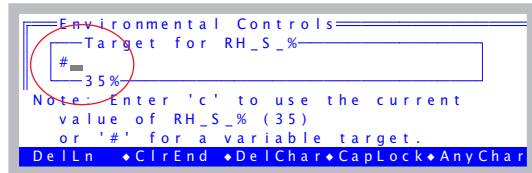
...and you'll be prompted for the ID by picking it from the list of all sys and user variables.

```
E═══════Pick  Target  Variable═══
 ┌──223:h2o_i
 │  224:h20diff
 │  225:CTair
Not│226:SVTair
v  │1001:xtrnRH
o  │2213:Tair_K
Sort   ♦  Find   ♦  ReFind♦  CANCEL♦  SELECT
```

```
══Environmental  Controls══
▼FLOW:  Relative  Humidity
 ┌ Control= RH_S_%
▮┤ Target= (xtrnRH)  35  %
 └ Response= "fast"
══T)  reTarget,  exit   O)  Off,  exit══
+   ▼/▶  ♦Edit   ♦   ♦cancel  ♦  Keep
```
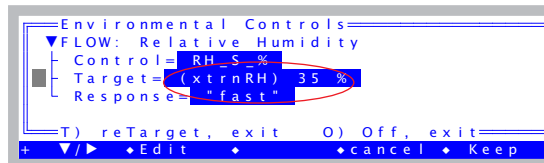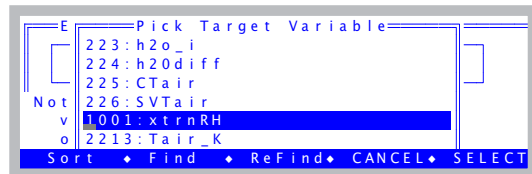
The Target will show the variable, and its current value.

```
══Environmental  Controls══
▼FLOW:  Relative  Humidity
 ┌ Control= RH_S_%
▮┤ Targe = (???)  0  %
 └ Response= "fast"
══T)  reTarget,  exit   O)  Off,  exit══
+   ▼/▶  ♦Edit   ♦   ♦cancel  ♦  Keep
```
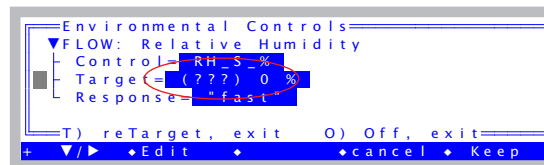
If you pick a non-existent variable, it looks like this.

Figure 7-4. Instead of entering a fixed target value, you can specify a variable, system or user. The target value comes from this variable, and is updated every 30 seconds.

─────────────────────────────

[2.]That value should be ambient plus something to account for the chamber boundary layer conductance.

Then, the flow/humidity control function key will show the humidity target as the latest value of the variable we specified.

Latest value of *xtrnRH*

Figure 7-5. When a control has a variable target, the value is updated every 30 seconds. This value is shown on the control key label.

When a control has a variable target, the 30 second ISR for that control is skipped; instead, the control manager retargets the control to the current value of the specified variable. The end result is that every 30 seconds *while the control manager is active*, the target for the control will change.

Variable targets are a powerful tool, but do have a possible limitation when combined with AutoPrograms. See **AutoPrograms and the Control Manager** on page 25-30.

Humidity Control

Humidity control in the LI-6400 is done by a combination of two mechanisms:

1   **Incoming humidity**
The mechanism for controlling incoming humidity is manual: the amount of incoming air that is routed through the desiccant is set by the valve on the top of the desiccant tube, allowing incoming humidities to range from ambient to dry. (You could also moisten the air - see **Humidifying Incoming Air** on page 4-52).

2   **The flow rate of air through the chamber**
The flow of air through the chamber is controlled by the software, either by controlling pump speed (CO_2 mixer not installed) or diverting excess flow (CO_2 mixer installed).

The Operational Envelope
The manual bypass control defines the operating envelope within which the automatic flow controls can operate. By forcing more of the flow through the desiccant, the operating window (upper and lower limits of humidity) decreases; when less air is forced through the desiccant, the window increases (Figure 7-6).

Figure 7-6. The operating envelope is determined by the transpiration from the leaf. To maintain any given humidity, a combination of flow rate and desiccant knob settings might achieve it. Typically, the knob remains fixed, and the flow is adjusted automatically.

Control Options

When **f2** level 2 is pressed in New Measurements mode, the following control options are presented:

```
                                        ▼FLOW: Flow Rate
                                      → ├ Control= Flow_µm
                                        └ Target= 500 µmol/s

                                        ▼FLOW: H2O mole fraction
                                      → ├ Control= H2OS_mml
    ═══Flow Control Types═══             ├ Target= 20.0 mmol/mol
    Off                                  └ Response= "fast"
    ◆ Flow Rate
    H2O mole fraction                   ▼FLOW: Relative Humidity
    Relative Humidity                 → ├ Control= RH_S_%
    Vapor Pressure Deficit              ├ Target= 50 %
                                        └ Response= "fast"

                                        ▼FLOW: Vapor Pressure Deficit
                                      → ├▼Control= VPD
                                         └ Temperature basis: Tleaf°C
                                        ├ Target= 1.00 kPa
                                        └ Response= "fast"
```
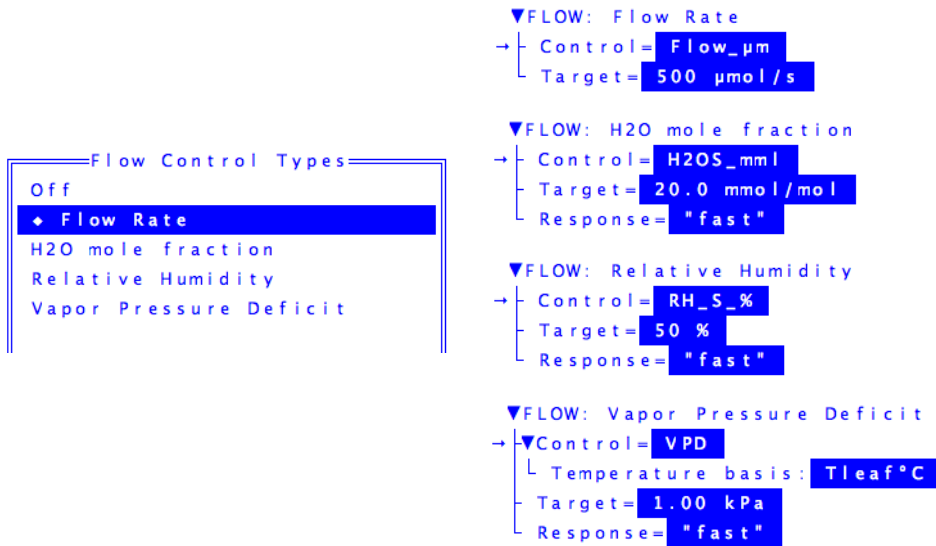
Figure 7-7. The humidity control screen presents one fixed flow option and several constant humidity options.

Table 7-1. Flow/Humidity control menu options

Option	Description
Off	Turns off the pump. Not normally used, except for diagnostics, or to save the battery or motor when not taking measurements. Or, to provide some peace and quiet.
Flow rate	Maintains a fixed flow rate through the chamber. This is a good default option.
H_2O mole fraction	Maintains a constant water mole fraction in the sample cell. This is useful during response curves for light or CO_2.
Relative humidity	Maintains a constant relative humidity in the sample cell.
Vapor pressure deficit	Maintains a constant vapor pressure deficit in the sample cell. This VPD can be based on a leaf or air temperature (you choose the variable).

For best results, start off using the Flow rate option, at a mid-range flow rate ($400 \ \mu\text{mol s}^{-1}$). Then, adjust the desiccant knob as necessary to give the desired humidity in the chamber. You may have to modify the flow rate to achieve a particular humidity. For example, if you can not get the humidity high enough at $400 \ \mu\text{mol s}^{-1}$ even with the desiccant on full bypass, then slow the flow down.

High flow rates are good from a system response point of view, but reduce the CO_2 and H_2O differentials on which the calculations are based. Low flow rates will increase these differentials, making their measurement less prone to error, but at the cost of increasing the system response time. Try to keep the flow rates above $100 \ \mu\text{mol s}^{-1}$ (without a CO_2 mixer installed) or $50 \ \mu\text{mol s}^{-1}$ (with a mixer installed).

Constant Humidity Operation

The mole fraction, relative humidity and VPD options *actively regulate* the flow rate to maintain a constant humidity (either water vapor mole fraction, or relative humidity, or vapor pressure deficit) in the sample cell / leaf chamber.

> **Note**: For constant humidity options to work well, the leaf *must* be supplying a reasonable source of humidity. Thus, small leaf areas, low transpiration rates, or an empty leaf chamber, can make these options problematic.

The leaf chamber water vapor concentration that can be maintained is dependent upon a number of factors, including the transpiration rate of the leaf, the vapor pressure of the incoming air stream, and the volume of air that is being diverted through the desiccant tube.

■ **For best results with constant humidity options:**

1 Start out with the constant flow rate option
Manually find the flow rate / desiccant tube setting that provides the desired humidity or vapor pressure deficit.

2 Switch over to the humidity option
Your target value will be close to what is actually being achieved, so the control system will not have to do very much to get it there and hold it there.

H₂O Mole Fraction

Constant mole fraction is a fairly tight control circuit. The work is done by hardware circuitry, so no software intervention is needed. The software does figure out what signal (mV) the sample cell water IRGA will be putting out when the mole fraction is at the target value; this is communicated to the control circuit via an analog output signal. Since this target signal is also a function of pressure, temperature, and how the IRGA is zeroed and matched, there needs to be periodic updates of this target signal, even when the target mole fraction does not change.

Relative Humidity

The constant RH control is a simple extension of the H₂O mole fraction mode. The actual control is the same, it is just that more frequent updates are needed, since temperature is now in the mix. If temperature is changing rapidly, the temperature updates may not be frequent enough, and the actual RH may drift off target a little bit (usually not more than a couple of percent).

Vapor Pressure Deficit

This is the difference in vapor pressure between location X and what is actually in the air. Where is *X*? Well, you have at least two options: the chamber air, or the substomatal cavity of the leaf. In the case of the former,

$$VPD \ = \ e(T_{ca}) - e_s \tag{7-1}$$

and in the case of the latter,

$$VPD \ = \ e(T_l) - e_s \tag{7-2}$$

where $e(T)$ is the saturation vapor pressure function (Equation (14-24) on page 14-13), T_{ca} is computed chamber air temperature, ID# 225 (defined in the StdComps_6.2 ComputeList file)[3], T_l is leaf temperature, and e_s is vapor pressure in the sample cell (Equation (14-21) on page 14-12). The constant VPD option is the loosest humidity control, especially for VPD with respect to leaf temperature. The reason is that the only thing the humidity control has to work with is the flow rate, while there are a number of other things directly affecting the air or leaf temperature, such as incident radiation, transpiration rate, etc.

When you select the *Vapor Pressure Deficit* option, there will appear a subn-

_____

[3]·If you are doing energy balance, it is measured by the leaf temp thermocouple. Otherwise, it is estimated as the mean between sample IRGA cell and leaf temp.

ode named *Temperature basis* (Figure 7-8). This determines the temperature used in the calculation of VPD. value, followed by a temperature.

Currently selected temperature

▼FLOW: Vapor Pressure Deficit
→ ▼Control= VPD
     └ Temperature basis: Tleaf°C
     ├ Target= 1.00 kPa
     └ Response= "fast"
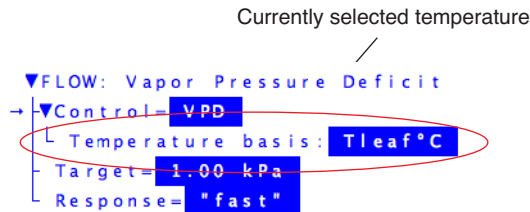
Figure 7-8. The VPD control needs a temperature basis.

When you edit this node, you are shown a menu of all user defined variables, and two system variables: *Tleaf°C* and *Tair°C*. (Figure 7-9).

Choose a Temperature
▼F 2213: Tair_K
─▼ 2214: Twall_K
  └ 2216: R(W/m2)
  ├ 2218: Tl-Ta
      -9: Tair°C
─T -10: Tleaf°C
Sort ♦ Find ♦ ReFind♦ CANCEL♦ SELECT
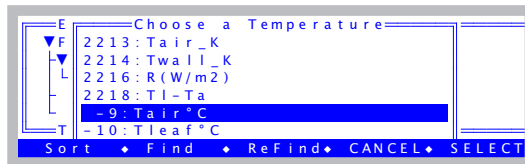
Figure 7-9. When asked to select a temperature for the VPD calculation, be sure what you select is a <u>temperature</u>. Don't select a VPD! The system does its own VPD computation based on the value of the (temperature) variable you select.

The reason the menu in Figure 7-9 includes all user variables is to allow you to select a temperature that you may be measuring or computing besides the two system temperatures. Normally through, you would select either *Tleaf°C* or *Tair°C*. Don't make the mistake of selecting VpdL or VpdA. <u>Remember, you are not selecting a VPD value, you are selecting a *temperature* for the system to use for its own VPD calculation.</u>

Humidity Time Response

When one of the constant humidity options is active, the setup dialog will include a Response node.

```
═══════Environmental  Controls═══════
▼FLOW: Vapor Pressure Deficit
┌▼Control= VPD
│  └ Temperature basis: Tair°C
├ Target= 1.00 kPa
├ Response= "fast"
└─T) reTarget, exit    O) Off, exit═══
▼/▶   ◆Edit   ◆        ◆cancel ◆ Keep
```
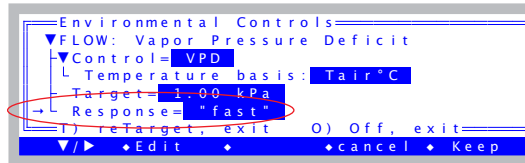
Figure 7-10. The response time can be fast, medium, *or* slow. Fast *is the de-*

When edited, the time response will cycle between *fast*, *medium*, and *slow*. This determines the rapidity of response of the control circuit (see Figure 3-38 on page 3-40). The *fast* setting will give you tight humidity control, but at the expense of "jumpy" flow readings. This is best for survey measurements when you are doing humidity control, and want to lock in rapidly on the target humidity as you go from leaf to leaf. The *medium* setting will drop the variability of the flow in half and still do a reasonable job of maintaining humidity on target. This is best for response curves and AutoPrograms, when you would like stable flow readings and stable humidities, and are not expecting abrupt changes in incoming humidity or transpiration rate. The *slow* setting is not particularly useful, unless you want the smallest possible flow variations and are not expecting any appreciable excursions in transpiration rates or incoming humidities.

CO$_2$ Control

If you do not have the 6400-01 CO$_2$ Mixer, then your CO$_2$ control options are limited to the knob on the soda lime tube, which will allow you to manually regulate CO$_2$ between ambient and zero for the incoming air stream.

If you are using the 6400-01, then you should leave the soda lime knob on full scrub.

When **f3** (level 2) is pressed in New Measurements mode, the following control options are presented:
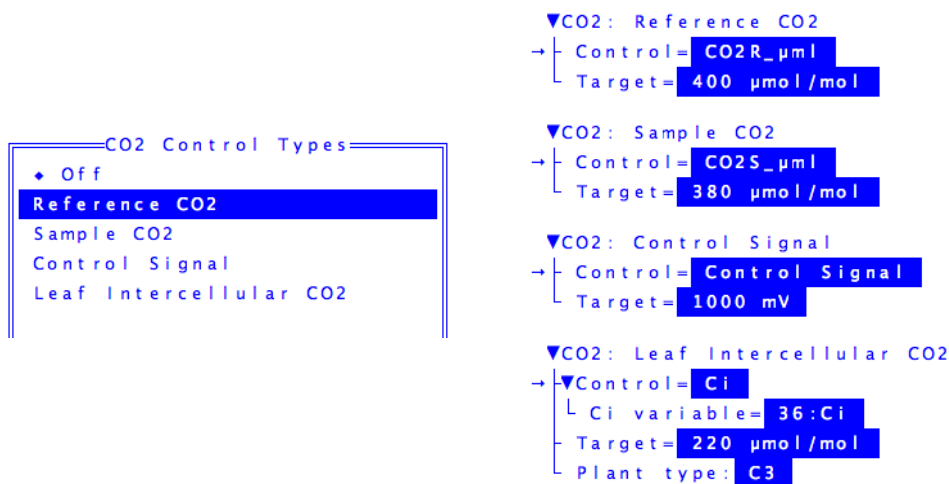
Figure 7-11. The CO$_2$ control screen options.

Table 7-2. CO$_2$ control options

Option	Description
Off	Turns off the mixer. (It's actually not turned off, but its output is switched out of the air stream and has no effect on the system's CO$_2$ concentration.)
Reference CO$_2$	Maintains a constant CO$_2$ concentration in the reference cell. That is, the incoming chamber CO$_2$ concentration is held constant at a target value. This is a good default option.

Table 7-2. (Continued)CO$_2$ control options

Option	Description
Sample CO$_2$	Maintains a constant CO$_2$ concentration in the sample cell / chamber.
Control Signal	The mixer control signal is set and held at a target value. This is mostly a diagnostic tool.
Intercellular CO$_2$	Tries to maintain constant intercellular CO$_2$

Reference CO$_2$

Usually the preferred option is constant reference, or incoming CO$_2$ concentration. When you enter a target concentration, the injector will adjust itself to bring the reference cell concentration to that value and hold it. Depending on how big the adjustment is between the present value and the one you want, it may take a few seconds or a few minutes to get there (going up always takes longer than coming down). Once the concentration is achieved, there is very little that will cause it to change,[4] so maintaining it is usually not a problem. If it does drift more than 1.5 ppm, the software will attempt to bring it back on target.

Sample CO$_2$

This option has the advantage of maintaining what the leaf actually "sees" as a constant, but it is not as tight a control loop. This is because there are things that affect this concentration that the control loop *cannot* control, like photosynthetic rate of the leaf, and flow rate through the chamber (the latter being fully in the domain of the humidity controller). All the CO$_2$ controller can do is regulate CO$_2$ concentration coming into the chamber, once those other things are stable.

■ **For best results controlling Sample CO$_2$:**

1 **Start out with the constant Reference option**

Specify a target 20 or 30 μmol mol^{-1} *above* what you want in the chamber (to allow for photosynthesis), and wait for the system to stabilize.

[4.]Temperature and pressure changes, bad soda lime, or the soda lime tube not on full scrub.

2   Switch to the S option

Once things are stable, the controller can quickly lock in and hold a constant sample concentration. Subsequent targets can then be specified for the sample concentration, but if it really loses control, you can always drop back to the Reference option and bring things back under control.

Unlike the constant humidity control options, which depend on the leaf providing lots of water, the Sample option doesn't depend on *anything* from the leaf, other than perhaps reasonably stable CO_2 assimilation. In fact, it works well with no leaf at all (as long as the chamber is closed, of course[5]).

Interaction with Humidity Control

There is an interaction between the constant sample CO_2 option, and the constant humidity control options. If the humidity controller is changing the flow rate to try to achieve some target humidity, the CO_2 concentration in the leaf chamber is going to be responding as well. Generally this causes the CO_2 controller (when it's doing the Sample option) to wait until things stabilize, before attempting adjustments.

The result is that the Sample CO_2 option will work with constant humidity control, but expect longer system equilibration times.

Control Signal

The Control Signal option has two purposes: as a diagnostic, and as an option that will provide the fastest equilibration time after a change in CO_2 target. This option simply sets the controller to a target value (voltage, not CO_2 concentration), and makes no further adjustment. Thus, when you specify a target, you may not know exactly what the final concentration will be. You will be assured the controller will be making no changes, so any change or fluctuation you see in reference CO_2 concentration will be coming from something else (leaks, bad soda lime, total flow rate changes, long term drift of the controller, etc.).

---

[5].Opening the chamber might seem to be a way to really mess up the constant sample CO_2 option, but usually the subsequent sample cell CO_2 fluctuations are sufficient to keep the controller from even *trying* to control it.

Intercellular CO$_2$

The Intercellular CO$_2$ option is a bit of a different animal than the other options. Its purpose is to try to maintain a constant C$_i$ in the face of *other* changes going on, such as during a light response curve. You should *not* try to generate a CO$_2$ response curve with this control. Here's why: Leaf intercellular CO$_2$ is approximately (see Eqn (1-18) on page 1-11)

$$C_i \approx C_s - \frac{A}{g} \qquad\qquad (7\text{-}3)$$

where C_s is the sample cell CO$_2$ concentration, A is photosynthetic rate, and g is conductance. Of the three terms on the right side of the equation, only C_s comes close to being directly controlled by the CO$_2$ mixer, and even that is not direct, since it depends on flow rate and photosynthetic rate A. What really drives C_i is the ratio of A to g, and the CO$_2$ mixer has no immediate affect on those. There are indirect effects, of course, and they can work against you. For example: suppose we start with C_s = 380, A = 20, and g = 0.1. That makes C_i = 380 - 200 = 180. Now suppose you want to make C_i go to 300. Your first guess might be to make C_s 500, since A/g is 200. So you do that. However, because C_s has gone up, A also goes up, and the stomates close a bit. Now we have A = 25, g = 0.07, and A/g is now 357, and that makes C_i only 143, and somehow we have gone the wrong direction.

So, use the Intercellular CO$_2$ option to maintain C_i while changing light levels or other environmental variables; do not use it to do CO$_2$ responses.

CO$_2$ Mixer Calibration

There is a relation between the CO$_2$ mixer's control signal, and the resulting CO$_2$ concentration measured in the reference cell. In fact you can see a plot of the relation that your instrument is currently using by selecting "Plot..." under "CO2 Mixer" in the Calib Menu. A typical plot is shown in Figure 18-20 on page 18-28.

The CO$_2$ control software uses this calibration to come up with a first guess when you have specified some target CO$_2$ concentration. If you find (when operating in constant reference concentration mode) that the first guesses do not seem very good, you can generate a new set of calibration points for it to use, described in **6400-01 CO2 Mixer** on page 18-25.

Temperature Control

Temperature control is achieved using dual Peltier devices on the sides of the IRGA / sensor head. These devices heat or cool the air that is circulated through the leaf chamber.

When **f4** (level 2) is pressed in New Measurements mode, the following control options are presented:
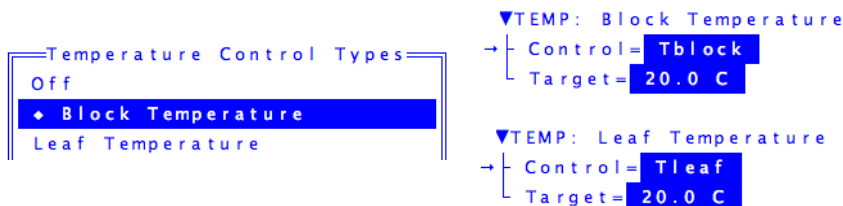
*Figure 7-12. The temperature control screen options.*

Table 7-3. Temperature control options

Option	Description
Off	Turns off the chamber coolers.
Block Temperature	Maintains a constant block temperature. Good default option.
Leaf Temperature	Maintains a constant leaf temperature.

Constant Block Temperature

Block temperature is measured in the wall of the IRGA, and is primarily used for the feedback control on the coolers. This control loop is fairly straightforward, and should be able to hold temperatures at targets that are within 7 degrees of ambient (larger if warming, since warming is more efficient than cooling).

Constant Leaf Temperature

The constant leaf temperature option is not a tight control loop, for two reasons: 1) the control of leaf temperature is indirect, via air temperature, and 2) there are factors beyond the reach of the controller that affect leaf temperature, including leaf transpiration rate and incident radiation.

For these reasons, probably the best use of the leaf temperature control option is to maintain leaf temperatures at or near ambient levels in the face of chang-

ing light levels or transpiration rates.

Note: The feedback signal for this option comes from the leaf temperature thermocouple, regardless of whether that signal is actually being used for leaf temperature or not[6].

Condensation

It is possible (but certainly not advisable) to specify a target temperature that will cause the coolers to bring the IRGA below the dewpoint temperature.

While in New Measurements mode, OPEN keeps track of the humidity in the IRGA and leaf chamber, and if it gets above 95%, will display a blinking warning

```
>> High Humidity Alert <<
```

For a complete discussion of this message, see **"High Humidity Alert"** on page 20-6.

---

[6]·For example, using the leaf temperature thermocouple to measure air temperature and calculate leaf temperature from an energy balance.

Light Control

This control is available only when the system is configured for an LED source. See **Specifying the Light Source** on page 8-4.

When **f5** (level 2) is pressed in New Measurements mode, the control options depend on the light source.

6400-02B Options

The options are shown in Figure 7-13.
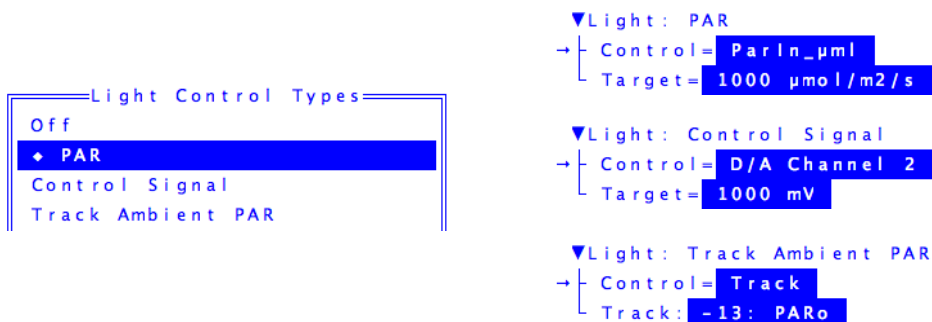
Figure 7-13. The lamp control screen options when using the 6400-02/02B LED Source.

PAR

This is the usual option for controlling the light source. Values between 0 and 2000 μmol m^{-2} s^{-1} can be specified, and are maintained by the control system. This is a fairly tight control loop, and within a few seconds of specifying a target, stability should be achieved. If something happens to change the light level (opening the chamber, for example), the control system will adjust the lamp to bring the light back to the target.

When you specify a target value, the software makes a first guess of the required control signal it will take to achieve that light level in the chamber. After a few seconds, that guess is adjusted based on what the light sensor is actually reading. If you notice that the first guess is not very close, you can generate a new and improved relationship by selecting the light source calibration routine from the Calib Menu.

Control Signal
The constant control signal option is a diagnostic tool, in which the control signal to the lamp is set directly (in mV, not μmol m^{-2} s^{-1}). The range is 0 to 5000 mV.

Track Ambient PAR
The tracking option works just like the Quantum Flux option, except the target value potentially changes every 3 seconds, and the target value comes from the external quantum sensor.

6400-18 Options
The options are shown in Figure 7-14.
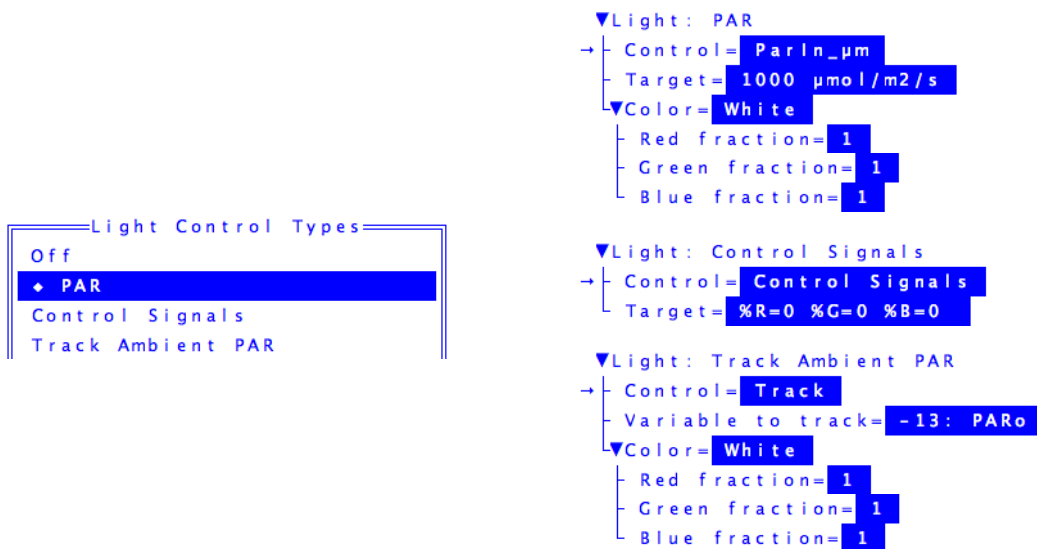
Figure 7-14. *The lamp control screen options for the 6400-18 RGB Light Source.*

PAR
This is the usual option for controlling the light source. Values between 0 and 2000 μmol m^{-2} s^{-1} can be specified, and are maintained by the control system. This is a fairly tight control loop, and within a few seconds of specifying a target, stability should be achieved. If something happens to change the light level (opening the chamber, for example), there is a feedback circuit that immediately compensates.

When you specify a target value, the software makes a first guess of the required control signal it will take to achieve that light level in the chamber. After a few seconds, that guess is adjusted based on what the light sensor is actually reading. If you notice that the first guess is not very close, you can generate a new and improved relationship by selecting the light source calibration routine from the Calib Menu.

Track Ambient PAR
The tracking option works just like the Quantum Flux option, except the target value potentially changes every 3 seconds, and the target value comes from the external quantum sensor.

Color

Color is an option for both PAR and Track Ambient PAR modes. You can specify color either by picking it from a menu, or specifying the mix of red, green, and blue fractions (Figure 7-15).
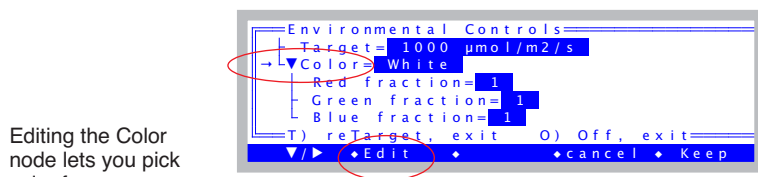
Editing the Color node lets you pick color from a menu.

You can add a color to the list by pressing **Add**...

...or just **select** one that's already there.

You can also set the color by directly setting the Red, Green, and Blue proportions.
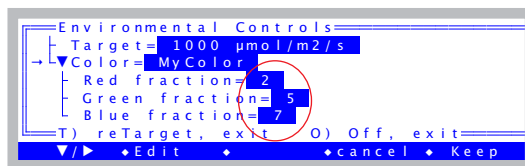
Figure 7-15. Picking a color, and/or adding to the color list.

The list of colors lives in the file /User/Configs/RGBColors. The state of that file after doing Figure 7-15 (adding a color) is shown in Figure 7-16. (If that file is missing, it will be generated from a master copy, which is /Sys/Lib/RGBColors.

```
"White" 1 1 1
"Red" 1 0 0
"Yellow" 1 1 0
"Green" 0 1 0
"Cyan" 0 1 1
"Blue" 0 0 1
"Magenta" 1 0 1
"RedBlue Source" 0.94 0 0.06
"MyColor" 2 5 7
```

Figure 7-16. Example contents of the file /User/Configs/RGBColors.

Each line in the file consists of a quoted string, followed by three values: red, green, and blue. When used, the three values are normalized to sum to 1. White, for example, could be 1 1 1 or 10 10 10 or .33 .33 .33.

The "RedBlue Source" simulates the spectral output of the 6400-02B, which has about 6% blue.

Control Signals
The constant control signal option is a diagnostic tool, in which the control signal to the lamp is set directly in percent of maximum output for each of the three colors, red, green, and blue. The range for each is 0 to 100. When in this mode, the feedback circuit is not enabled, so changes in light due to varying reflection back into the source are not compensated.

6400-40 Options

The options are shown in Figure 7-17.

```
                                         ▼Light: PAR
                                       → ┌ Control= ParIn_µml
                                         ├ Target= 1000 µmol/m2/s
                                         ├ Blue portion= percentage
                                         └ Blue %= 10


        ╔═══Light Control Types═══╗
        ║ Off                      ║      ▼Light: Control Signals
        ║ ♦ PAR                    ║    → ┌ Control= Control Signals
        ║ Control Signals          ║      └ Target= R=1000 B=1000 mV
        ║ Track Ambient PAR        ║
        ╚══════════════════════════╝
                                         ▼Light: Track Ambient PAR
                                       → ┌ Control= Track
                                         ├ Variable to track= -13: PARo
                                         ├ Blue portion= percentage
                                         └ Blue %= 10
```
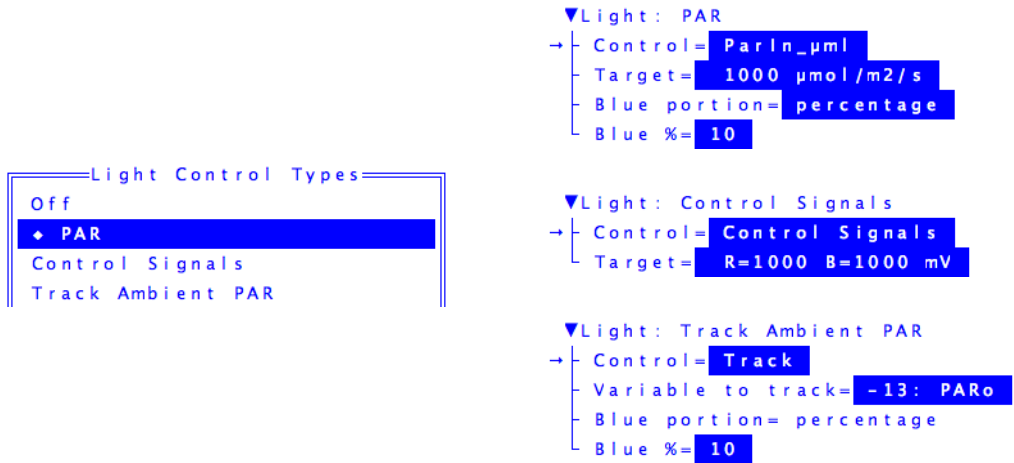
Figure 7-17. The lamp control screen options for the 6400-18 LCF.

This is all discussed in **The LCF as a Light Source** on page 27-18.

Light Sources and Sensors

8

The most important parameter is the hardest to measure

8

Light Sources and Sensors

Radiation is the most important environmental factor in photosynthesis, and the most difficult to measure. If you could measure the irradiance on a leaf with an absolute accuracy of 5%, you'd be doing very well. In reality, the measurement error is more like 10%[1]. But 5% or 10% errors would be intolerable for temperature, CO_2 concentration, or humidity, since we can measure those parameters to better than 1%. This chapter explains how the LI-6400 measures photosynthetically active radiation.

Why Two Sensors?

The optional External Quantum Sensor (part # 9901-013) measures photosynthetic photon flux density (PPFD) over the 400nm to 700nm waveband. It gives accurate results with most light sources over a wide range of incident angles, and is mounted in a position that minimizes errors due to shading. It's a good quantum sensor, but there's a problem: it's in the wrong place.

The external quantum sensor does not have the same field of view as the leaf element in the chamber, nor is it subject to similar shading conditions, angular responses, or attenuation by the chamber window. In an extreme case, when the quantum sensor is shaded and the leaf isn't, the quantum sensor could be measuring a factor of 10 lower than the actual irradiance on the leaf. In more typical conditions, there will easily be differences of 10%.

To address this problem, the standard light sensor in the LI-6400 is an unfiltered gallium arsenide phosphide (GaAsP) device that is small enough to be placed in the chamber very near the leaf plane. A calibration coefficient weighted for a "sun+sky" spectrum is provided in the LI-6400 Configuration list. The spectral properties and calibration of the GaAsP sensor are discussed below, but in essence, we are getting a second measurement that's very nearly in the right location, but with a less-than-ideal sensor.

There is a third type of light sensor that is used in the LI-6400. It is a silicon

---

[1.]See "Radiation Measurement" for a discussion of all the sources of error. This article is found in the LI-COR brochure, "Radiation Measurement Instruments", publication number LM1-11/94.

diode that monitors and controls the optional 6400-02, -02B, -18, and -40 Light Sources. The silicon diode light sensor has a wide response that covers the red and blue LED emission range. It is also an unfiltered sensor, but calibration is simplified because it only views radiation from the LED light source. A calibration coefficient is provided with each 6400-02 or -02B LED Light Source.

Specifying the Source and Sensor

One of the configuration parameters of OPEN involves specifying the light source (and indirectly, the sensor). There are reasons this information needs to be known:

- **Light Source Control**
  Is a light source installed? The answer to this question determines whether or not the lamp control key in New Measurements mode is active, and/or whether the Leaf Chamber Fluorometer keys are enabled. It also tells the software what type of in-chamber light sensor is connected.

- **Calibration Issues**
  Whatever the type of light sensor being used, it has a calibration factor which converts raw mV to μmol m^{-2} s^{-1} photon flux. This conversion factor depends on the spectral characteristics of the incident radiation. The software can adjust this factor depending on what the light source is.

- **Energy Balance Issues**
  When leaf temperature is not directly measured, it is computed using a leaf energy balance (described in Chapter 17). One of the inputs of this computation is the absorbed radiant energy by the leaf, and this again depends on the spectral characteristics of the incident radiation.

Calibration Equations
The equations that OPEN uses for computing the readings of its light sensors are (14-17) through (14-19), starting on page 14-10.

Specifying the Light Source

This is set by the configuration node <open> <light> <source> in `Config Menu|View/edit`.
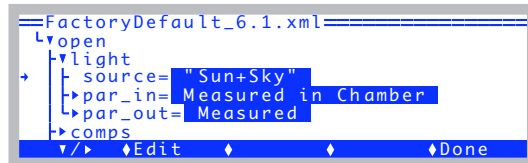
Figure 8-1. Viewing the configuration tree, in View/Edit in the Config Menu.

When you edit the *source* node, the Light Source Selection Menu (Figure 8-2) will appear.
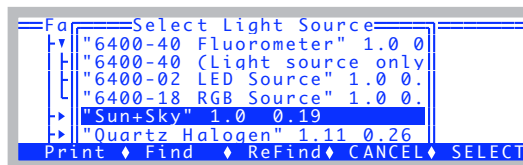

```
"6400-40 Fluorometer" 1.0 0.16
"6400-40 (Light source only) 1.0 0.16
"6400-18 RGB Source" 1.0 0.16
"6400-02 Red Blue Source" 1.0 0.16
"Sun+Sky" 1.0 0.19
"Quartz Halogen" 1.11 0.26
"Fluorescent" 0.88 0.18
"Metal Halide" 0.87 0.17
```

Adjustment factor f_a (Eqn (14-18) on page 14-11).

Conversion factor αk for energy balance Equation(17-9) on page 17-3.

Figure 8-2. The menu for picking a light source.

Each item in the list consists of a string plus two values. The string becomes the value for <open> <light> <source>. The first value is the actinic adjustment factor (f_a in Eqn (14-18) on page 14-11). It becomes the value for <open> <light> <par_in> <sensor> <cal> <actinity>. The second value is αk, used to convert incident μmol m^{-2} s^{-1} to absorbed W m^{-2} in the energy balance equation (17-9) on page 17-3. It becomes the value for <open> <comps> <energybal> <alphaK>.

The lower part of the light source list ("Sun+Sky" and below) comes from the file /User/Configs/LightSources. If that file is not there, it is created from /Sys/Lib/LightSources. If you wish to add light sources, do so by editing /User/Configs/LightSources. For an example, see **Generating a Calibration Correction** on page 18-38.

Specifying the Light Sensors and Methodology

<open> <light> <par_in> determines how *ParIn_μm* (system ID #-12) is to be determined (Figure 8-3). It can be skipped, directly measured, or inferred from the parOut measurement.

```
════╤═════════Select In-Chamber PAR option═════════
    │  None
    │  Measured in Chamber
    │  ParOut * transm
    │
    │
           ♦              ♦          ♦Cancel  ♦Select
```
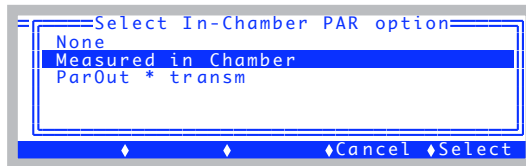
Figure 8-3. ParIn can be directly measured, or inferred from the parOut measurement. The config tree's nodes will adjust accordingly.

The value selected will shape the configuration tree appearance (Figure 8-4).
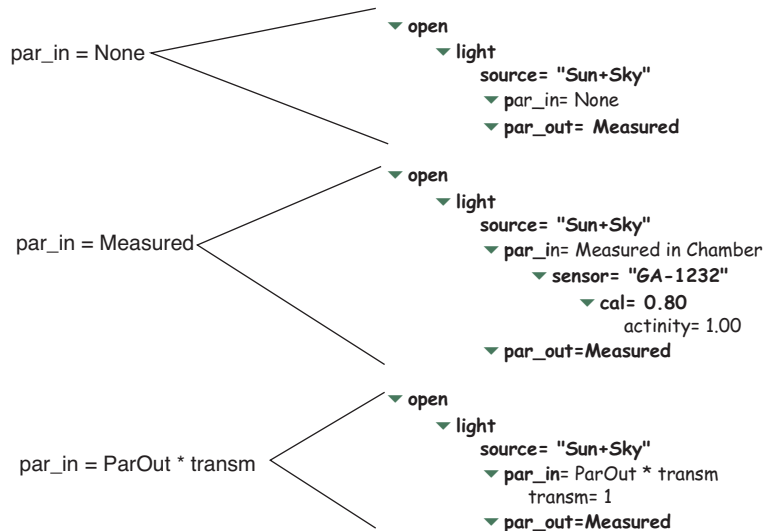
Figure 8-4. Effect on the configuration tree of the three settings of par_in.

If PAR is being measured, the *sensor* node will determine which sensor is being used in. These are also selectable from a list, if more than one of the relevant types are in your accessories list ("`View/Edit Accessory List`" in the Calib Menu).

For example, suppose the accessories list looked as shown in Figure 8-5. There are two chamber tops (GA-1097 and GB-223) listed, and two Red Blue LED light sources. Editing the <open> <light> <par_in> <sensor> node will let you choose between the appropriate sensors for each light source.

Config Tree (Config Menu)

```
▼ open
    ▼ light
        source="Sun+Sky"
        ▼ par_in= Measured in Chamber
            ▼ sensor=GA-1097
                ▼ cal =0.81
                    actinity= 1
```

The choice is from the <tops> node when source is "sun+sky".

Accessories List (Calib Menu)

```
▼ li6400
    ▼ accessories
        ▼ tops
            ▶ GA-1097= 0.81
            ▶ GB-223 =0.78
        ▼ led
            ▶ SI-1267= -0.73
            ▶ SI-1389= -0.71
        ▼ quantum
            ▶ Q30292= 7.32
```

The choice is from the <led> node when source is "6400-02B".

```
▼ open
    ▼ light
        source="6400-02 LED Source"
        ▼ par_in=Measured in Chamber
            ▼ sensor=SI-1267
                ▶ cal =-0.73
                    transm= 1
        ▼ par_out= Measured
            ▼ sensor= "Q30292"
                cal= 7.32
```

Figure 8-5. Interaction between the accessories list in the Calib Menu, and the light source selector in the Config Menu: Editing par_in *lets you pick from the appropriate list.*

External Quantum Sensor

Similarly, the external quantum sensor is handled by the <open> <light> <par_out> node, and its subnodes. Editing this node toggles between None and Measured. If it is measured, the <sensor> subnode can be edited to select the specific LI-190 sensor being used (Figure 8-6).
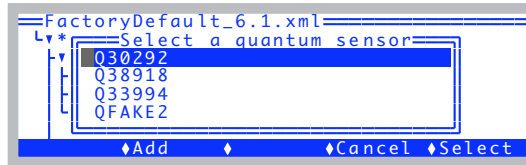
Figure 8-6. Selecting an LI-190 sensor.

6400-02 and -02B Light Sources

Installation

The 6400-02B light source comes with a calibration sheet that contains the correct multiplier for that sensor. This must be added to the accessories list (Figure 8-7).
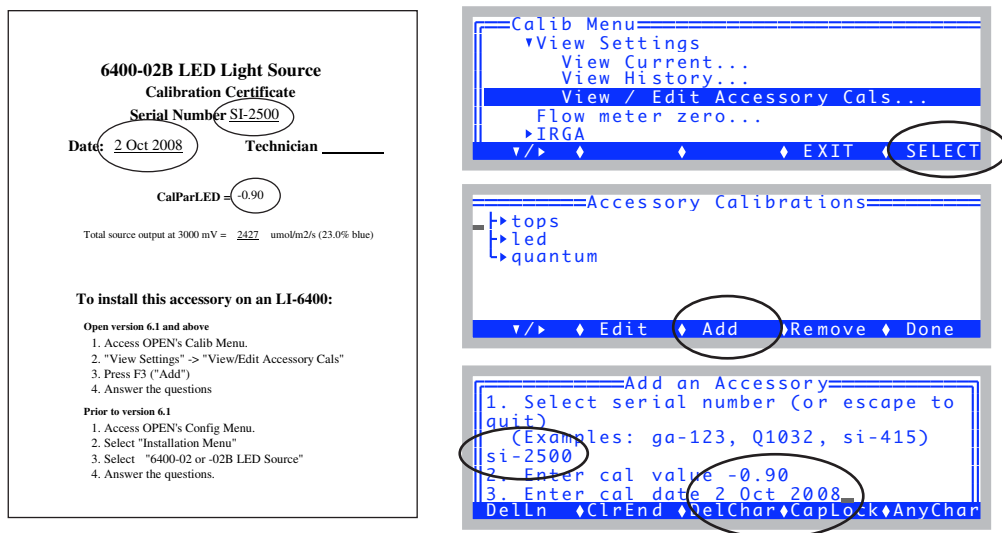
Figure 8-7. Adding an LED source to the accessories list.

Once it is installed, you enable its operation from the calibration node <open> <light> <source>, as described earlier (**Specifying the Light Source** on page 8-4.

Spectral Considerations

The 6400-02 spectral output has one peak centered at about 670 nm, while the 6400-02B has a secondary peak centered at about 465 nm (Figure 8-8). While the red only LED source provides a very suitable light source for photosynthetic studies (Tennessen, et al, 1994[2]), the addition of the blue LEDs in the

---

[2.]Tennessen D.J., D.L. Singsaas, T.D. Sharkey, 1994. Light-emitting diodes as a light source for photosynthesis research. *Photosynthesis Research* **39**: 85-92.

6400-02 enlarges the scope of suitable applications to include stomatal kinetics.

In-chamber light is measured with an unfiltered silicon photodiode that is part of the LED source. See **Light Source Calibration** on page 18-31.
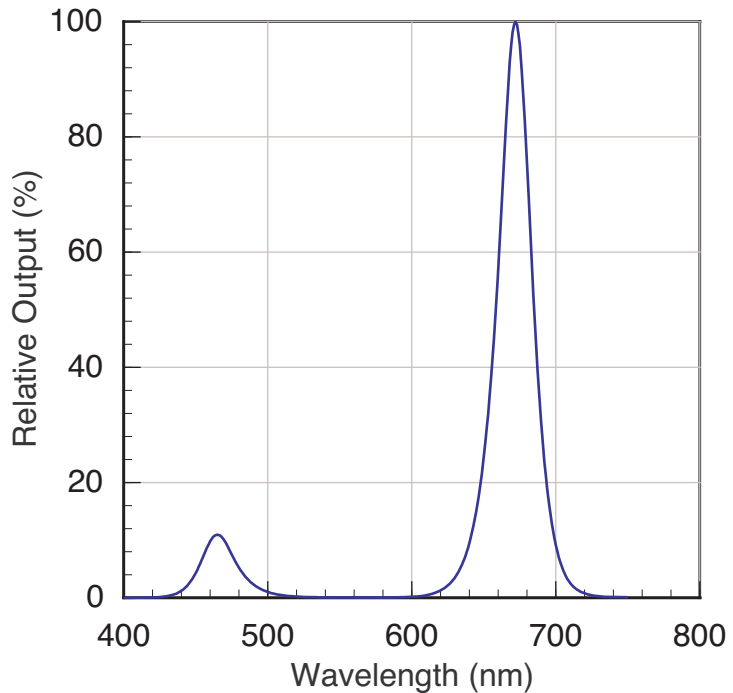
Figure 8-8. Typical output of a 6400-02B LED source at 25°C. (A 6400-02 source has the same red peak, but no blue peak.)

Temperature Effects

The LED source's efficiency and spectral characteristics depend a bit on temperature. At 50C, the efficiency will be about 75% of what it is at 0C. This means an increase in the power required for a given output (and a reduced maximum output) at higher temperatures. The spectral shift is strongest for the red LEDs. Typically, the peak wavelength is shifted towards longer wavelengths by 7 or 8 nm when comparing 0C to 50C performance. The blue peak shifts up by only 2 nm over those temperatures.

Aging

Like all light sources, the LED source is subject to aging. In a 2 year test with a 6400-02 running <u>continuously</u>, its output dropped by 30%: (10% the first 6 months, 10% the second 6 months, and 10% over the last 12 months).

If you find that your light source cannot achieve high enough light levels, and you suspect it's simply due to the age of the source, then there is a possible remedy. See **Source Isn't Bright Enough** on page 20-33.

6400-18 RGB Light Source

The 6400-18 RGB Light Source contains three colors of LED: red (635 nm), blue (460 nm), and green (522 nm) that can be controlled independently. Normally, it is used to generate "white" light, which we define to be "equal quantum output" of the three colors (Figure 8-9).
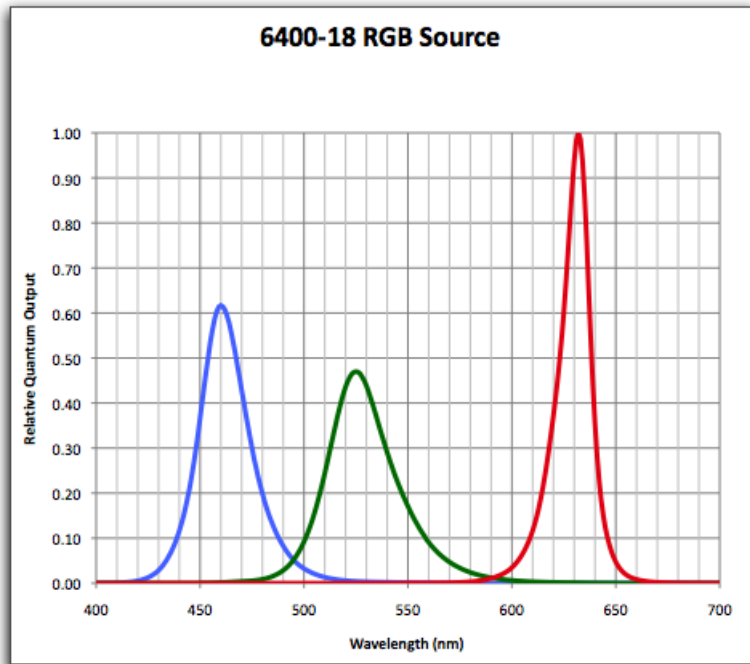
Figure 8-9. Relative quantum output of the 6400-18 RGB Light Source for white light. For white (or quantum white), the areas under the three curves are balanced.

Installation

Hardware

To install the 6400-18 RGB Light Source, connect the communications cable between it and the 37 pin port on the console, and the power cable to the supply (Figure 8-10).
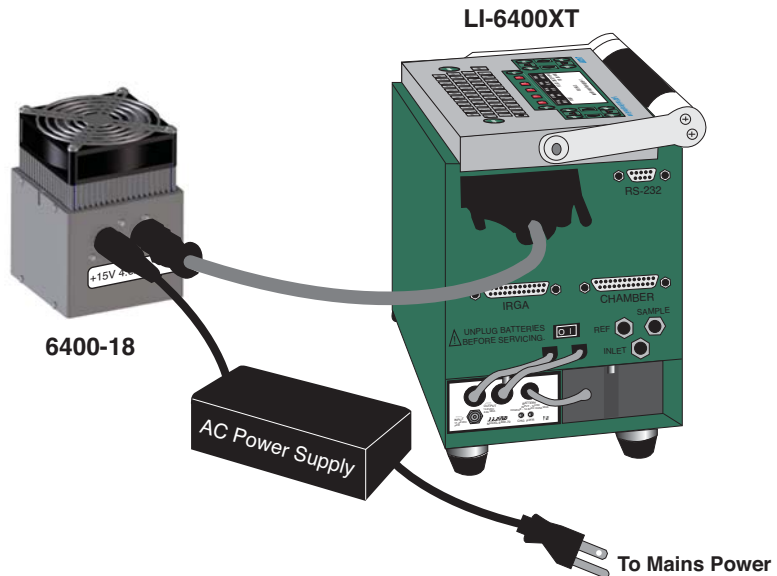
Figure 8-10. Cable connections for the 6400-RGB Light Source.

There may be a slight flicker or flash when the power is first applied. Also, the fan may or may not run briefly. This is all normal behavior.

Software

To operate the 6400-18 RGB Light Source, OPEN (version 6.1 or above is required) must be configured to have this device as its light source. You can build a configuration as shown below, or simply enable the console for RGB source operations (page 8-14).

■ **To build a new configuration for a specific chamber**

If you have a specific chamber you wish to use this light source for, follow these steps to create and store a configuration.

1   Go to the Config Menu
From OPEN's main screen, press **f2**.

2   Select New...
Highlight the *New*... entry, and press **enter**.

```
╔══Config Menu═════════════════════╗
║    New...                         ║
║    Open...                        ║
║    Save as...                     ║
║    View/edit...                   ║
║                                   ║
║  ▼/▶   ♦         ♦      ♦ EXIT  ♦ SELECT
╚═══════════════════════════════════╝
```

3   Select the RGB Source
Highlight the *6400-18 RGB Source*... entry, and press **enter**.

```
╔══Make a NEW Config for...════════╗
║   ▼Light Sources                  ║
║      6400-02B LED Source...       ║
║      6400-18 RGB Source...        ║
║      6400-40 Fluorometer...       ║
║   ▶Clear 2x3 Tops                 ║
║   ▶Larger Chambers                ║
║  ▼/▶   ♦         ♦      ♦ EXIT  ♦ SELECT
╚═══════════════════════════════════╝
```
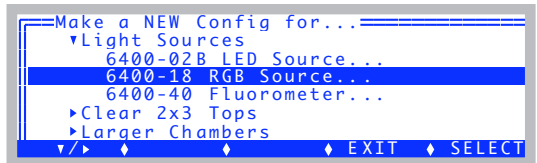
4   Select the desired chamber
Pick the chamber you wish to use, and press **enter**. Depending on the chamber, there may be some additional prompts you'll have to answer.

```
╔════Select an underlying chamber══╗
║ a) Any 2x3 Chamber                ║
║ b) 6400-07/-11 2x6 Chamber        ║
║ c) 6400-05 Conifer Chamber        ║
║ d) 6400-15 ExtRch1cm Chamber      ║
║ e) 6400-17 Whole Plant            ║
║ f) 6400-22 Opaque Conifer Chamber ║
╚═══════════════════════════════════╝
```
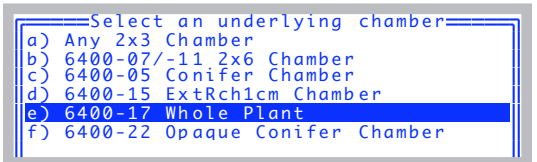
5   Pick the N option
Press **N**.

```
Config Modifications Ready. Pick one:

N) implement them as a New config:
   named 'WholePlantChamber  EB RGB'
C) apply them to the Current config
V) View the modifications

<escape> - discard and quit
```
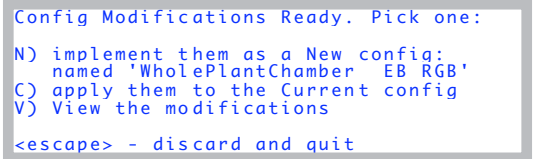
6   Wait
The configuration will then be implemented. When it comes time to communicate with the RGB Source, you'll see a display as shown to the right.

```
Ci - ok
Ci_P╔════════RGB Comm═══════════╗
Ci/C║30 31 30 39 00 00 20 46 00 00║
RHsf║ 31 03 00  _                 ║
C2Sf╚═══════════════════════════╝
AHs/

Linking /Sys/Lib/RGBTools...
```
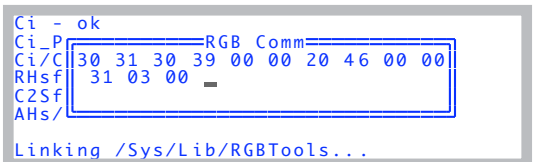
If there is a problem with the communications between the console and the

RGB Source, you'll see the display in Figure 8-12 on page 8-15.

7   Store the Configuration
At this point your con-
figuration has been im-
plemented, and you are
prompted to store it.
Modify the name if you
wish, and press **enter**. (If
you do not wish to store
it at this time, press **escape**, instead).

```
Dir: /User/Configs/UserPrefs
File:WholePlantChamber  EB RGB_
Filter: ver 6 config files (*.xml)
════════════Save this config════════════
 Default+LED.xml
 Default+RGB.xml
 FactoryDefault_6.1.xml
+ Dir     ♦        ♦ Filter♦ CANCEL♦ SELECT
```

Press **escape** a couple of times to return to OPEN's main screen, and you are
ready to use your new configuration.

■ **To enable RGB operations <u>without</u> building a configuration**
You can simply enable RGB operations from the configuration tree.

1   Access the system configuration
Go to `Config Menu|View/edit` and press **enter**.

```
═Config Menu════════════════════
    New...
    Open...
    Save as...
    View/edit...

    ▼/▶    ♦        ♦        ♦ EXIT   ♦ SELECT
```

2   Navigate to <open> <light> <source>
Navigate down to the
source node, and press
f2.

```
═FactoryDefault_6.1.xml═══════════
 └▶open
  ├▼light
▶ │ ┌ source= "Sun+Sky"
  │ ├▶par_in= Measured in Chamber
  │ └▶par_out= Measured
  ├▶comps
  ▼/▶  ♦Edit    ♦        ♦        ♦Done
```

3   Select RGB Source
Highlight the *6400-18
RGB Source* entry, and
press **enter**.

```
═Fa╔════Select Light Source════╗══
 └o║"6400-40 Fluorometer" 1.0 0║
  ├▼║"6400-40 (Light source only║
  │ ║"6400-02 LED Source" 1.0 0.║
  │ ║"6400-18 RGB Source" 1.0 0.║
  │ ║"Sun+Sky" 1.0  0.19        ║
  ├▶║"Quartz Halogen" 1.11 0.26 ║
  Print ♦ Find  ♦ ReFind♦ CANCEL♦ SELECT
```

4 Wait for communications

Synchronization with the source will occur.

```
═FactoryDefault_6.1.xml═══════════
 └▼o═════════════RGB Comm══════════
  ┌▼ 30 31 30 39 00 00 20 46 00
  ├
  ├
  ├
  ├▸comps
Linking /Sys/Lib/RGBTools...   ♦ SELECT
```

5 Done

You are done. **Escape** back to the main screen.

```
═FactoryDefault_6.1.xml═══════════
 └▼* open
  ├▼* light
→ │ ├ * source= "6400-18 RGB Source"
  │ ├▸* par_in= Measured in Chamber
  │ └▸par_out= Measured
  ├▸* comps
  ▼/▸  ♦Edit   ♦   ♦Undo   ♦Done
```

Communications

When an RGB Source configuration is implemented, the console queries the RGB Source for some information (Figure 8-11). If there is a problem, you'll see the message shown in Figure 8-12, instead.

```
Ci - ok
Ci_P┌═══════════RGB Comm═══════════
Ci/C│30 31 30 39 00 00 20 46 00 00
RHsf│ 31 03 00 ▃
C2Sf└═════════════════════════════
AHs/

Linking /Sys/Lib/RGBTools...
```

Figure 8-11. The normal synchronization between console and source takes a few seconds, during which time this display is shown.

```
Ci - ok
Ci_P┌═══════════════════════════════
Ci/C│RGB Lamp not detected.
RHsf│Try to communicate anyway?▃
C2Sf│
AHs/└══════════════════(Y/N)═══

Linking /Sys/Lib/RGBTools...
```

Figure 8-12. The communications problem display. Make sure both cables are connected, and the power supply is plugged in.

The console can detect if the 37-pin connector is plugged in and powered. If it doesn't think it is, you get Figure 8-12. As soon as the situation is remedied (e.g. power plugged in), communications will begin automatically.

Operation

When configured for the RGB Source, the lamp control screen in New Measurements mode (**f5** level 2) will appear as in Figure 8-13.
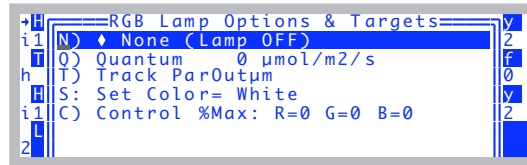
Figure 8-13. The RGB Lamp Control Screen.

The Q, T, and S Options

The *Q* option lets you specify a target value for the source to maintain, and the *T* option will make the target value follow the external quantum sensor. The color of the light for both the *Q* and *T* options is given by the *S* option.

■ **Example: Set 2000 *µ*mol m^{-2} s^{-1} white light.**

Set the controls as shown in Figure 8-14.

Press **Q** in the control panel. (The color is already white).
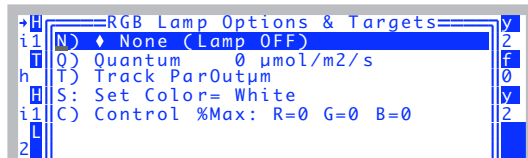
Type in the target value.
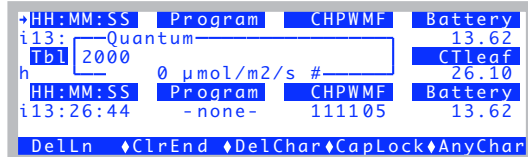
In New Measurements mode, the ParIn_*µ*m value should go to the target.
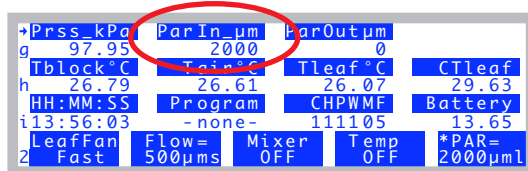
Figure 8-14. Set for 2000 µmol m^{-2} s^{-1} white light.

After a target is specified, you can watch the *ParIn_µm* value to track the

lamp's progress. If the current system averaging time[3] is 4 seconds (the default), then it will take 4 seconds for the displayed value to get to the first guess, which is usually within a percent of the target. After another few seconds, the value should lock on to the target very closely.

■ **Change to 2000 μmol m^{-2} s^{-1} red light.**
Set the controls as shown in Figure 8-14.

Press **S** to change the color.

```
→P┌════RGB Lamp Options & Targets═══
g │N) None (Lamp OFF)               │
T │Q) ♦ Quantum 2000 µmol/m2/s      │f
h │T) Track ParOutµm                │1
H │S: Set Color= White              │y
i1│C) Control %Max: R=0 G=0 B=0     │9
L │                                 │
2 │                                 │ml
```

Pick Red.

```
→P┌══RG┌══════Pick Color══════┐ets══
g │N) Non│ White               │   │
T │Q) ♦ Q│ Red                 │   │f
h │T) Tra│ Yellow              │   │3
H │S: Set│ Green               │   │y
i1│C) Con│ Cyan              0 │   │5
L │      │ Blue                │   │
 Add      ♦         ♦    ♦Cancel ♦Select
```

Press **Q** to pick a target.

```
→P┌════RGB Lamp Options & Targets═══
g │N) None (Lamp OFF)               │
T │Q) ♦ Quantum 2000 µmol/m2/s      │f
h │T) Track ParOutµm                │2
H │S: Set Color= Red                │y
i1│C) Control %Max: R=0 G=0 B=0     │4
L │                                 │
2 │                                 │ml
```

Type in the target value.

```
→Prss_kPa  ParIn_µm  ParOutµm
g      ┌───Quantum───────────┐   CTleaf
 Tbl│ 1000                   │    27.99
h   └──── 2000 µmol/m2/s #───┘
 HH:MM:SS   Program    CHPWMF   Battery
i13:49:50   -none-     111105    13.64

 DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Figure 8-15. Change to 1000 μmol m^{-2} s^{-1} red light.

The C option

The C option lets you directly specify how much each color should be on. The values are in terms of percent of maximum power. This option is mostly for diagnostic purposes.

_____

[3.]In the configuration tree at <open> <a2d> <avgtime>.

■ **Use 10% Red, 20% Green, and 30%Blue**

Access the control screen, press **C**, and type in 10 20 30 (Figure 8-16).
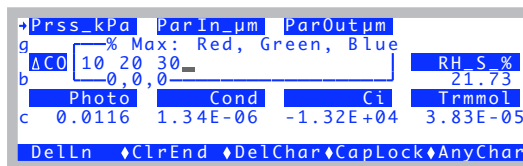
Figure 8-16. The control option.

This will result in a fairly bright, bluish white light. The quantum flux will depend on the particular unit, but should be 800 or 1000 μmol m^{-2} s^{-1}.

Internal Feedback

When operating in the Q (or T) modes, there is internal feedback circuitry used to maintain a constant light output. This is not there with the C option. For example, with the light configured as in Figure 8-16, and the lamp aimed at a dark surface, the ParIn_μm value shows about 1000 (Figure 8-17). If it is looking closely at a reflective surface (white paper, for example). the output is more like 1500.
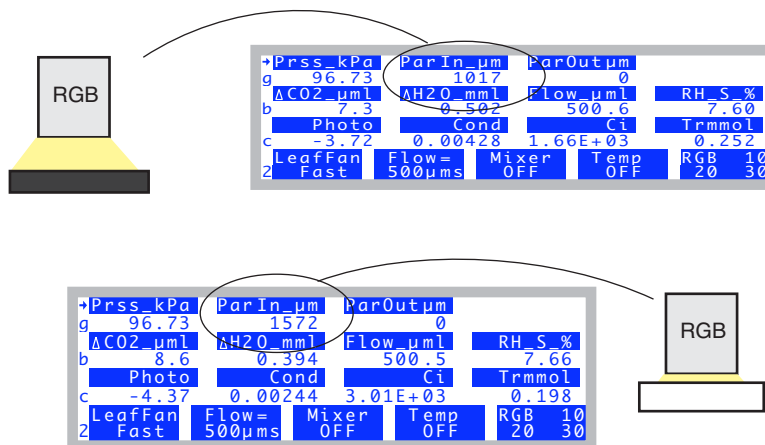
Figure 8-17. Without feedback control (mode C), there is a big effect on the lamp output of what the lamp is looking at.

On the other hand, if configured the Q option and the same experiment is done, the *ParIn_*μm value remains unchanged. This is because the feedback

circuit drops the lamp output as the white paper closes in on the lamp, thereby dropping the output to maintain a constant signal on the detector.

RGB Lamp Diagnostic Screen

When configured for the RGB Source, the Lamp Diagnostic screen (E) looks like Figure 8-18. (Read about the Diagnostics screens in New Measurements mode in **Diagnostics** on page 6-24).
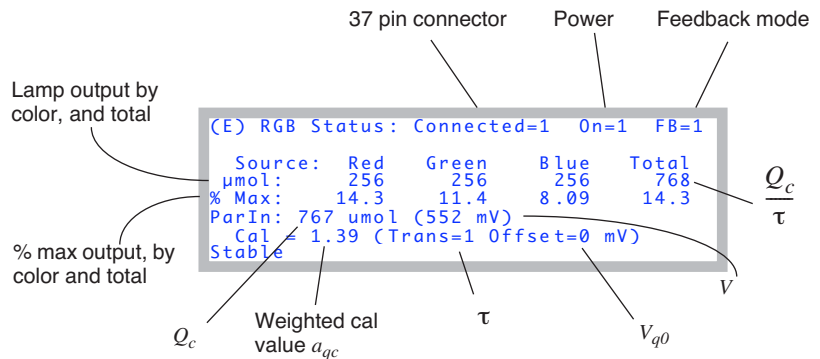
Figure 8-18. The RGB Diagnostics display.

Connected is 1 if the lamp is powered and the 37-pin connector is detected. Otherwise, it is 0. *On* is 1 if the lamp is supposed to be on (as set from the control panel), otherwise 0. *FB* indicates if the feedback circuit (**Internal Feedback** on page 8-18) is enabled. It will be for Q and T modes, but not C mode.

The *μmol* line shows the outputs of red, green, and blue (estimated), and the total (measured), all in μmol m^{-2} s^{-1}. These values refer to the quantum flux leaving the clear plastic face plate of the light source. The next row, *% Max*, represents the percent of full power that each color is operating at.

The *ParIn* line shows a value in μmol m^{-2} s^{-1}, and this is the *ParIn_μm* value normally shown in the display on line *G*. It is the value incident on a leaf in the chamber. The value comes from the remaining four values on the display: The *ParIn_μm* value (Q_c) comes from a signal (V), an offset (V_{qo}), and a multiplier (a_{qc}) (Eqn (14-17), pg 14-10, but reproduced here).

$$Q_c = a_{qc}(V - V_{q0}) \tag{8-1}$$

When configured for the RGB source, the signal V is not the normal in-chamber quantum sensor signal V_{qc}, but rather a spare channel (20) on the 37-pin connector. V_{qo} is still the software offset parameter (**Zeroing the ParIn Signal** on page 18-29). The multiplier a_{q0}, for the RGB Source, is determined by a weighted average of the red, green, and blue factory calibration values (a_r, a_g, and a_b) of the lamp, and a transmittance factor τ to account for any losses between the lamp window and the leaf.

$$a_{qc} = (f_r a_r + f_g a_g + f_b a_b)\tau \tag{8-2}$$

where f_r, f_g, and f_b are the weighting factors for the three colors, and

$$f_r + f_g + f_b \equiv 1 \tag{8-3}$$

The values of f_r, f_g, and f_b define the color.

The Transm Factor

The *transm* factor (τ in Eqn (8-2)) allows you to operate the lamp with reference to what the leaf sees for radiation, instead of what the lamp is putting out. The key is to pick the right value of τ. The value can be viewed and edited in the configuration tree (Figure 8-19), where it lives in the node <open> <light> <source> <par_in> <sensor> <cal> <transm>.
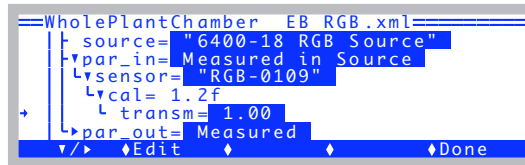
Figure 8-19. The configuration tree with an RGB Source.

Also visible is the current value of the weighted multiplier a_{qc} (the <cal> node).

Table 8-1 lists some measured values of τ for the two chambers designed for

use with the 6400-18.

Table 8-1. Suggested *transm* values for two chambers.

Chamber	Location	transm
6400-17 WPA	Just under propafilm	0.80
	Midway between top, bottom	0.75
	Chamber floor	0.70
6400-22 Opaque Conifer	Middle	0.80

For non-standard chambers, such as setting the RGB Source above a standard 2x3 chamber, here are some guidelines:

$$\tau \cong t \times h(d) \qquad\qquad (8\text{-}4)$$

where *t* is the transmittance of the chamber top (e.g. the propafilm), and $h(d)$ is a reduction factor based on the distance between the bottom of the RGB window, and the leaf surface (Figure 8-20).
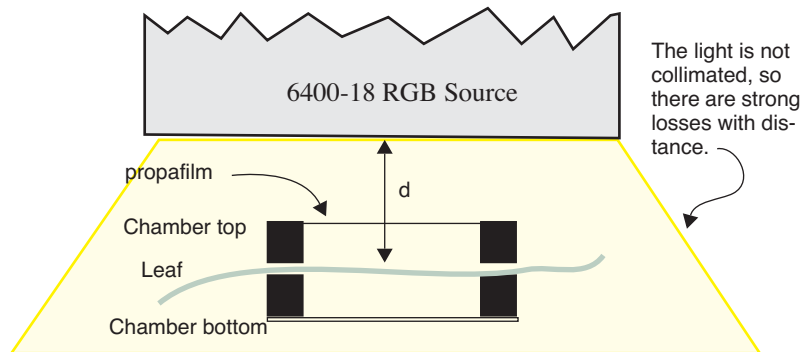
Figure 8-20. Schematic for a non-standard installation. The leaf is distance d *from the bottom of the light source, and the chamber top has transmittance* t.

We have measured the loss function h(d), in the absence of any walls or reflectors between the RGB Source and the chamber top, to be

$$h(d) \ = \ \frac{1}{1 + 0.022d} \qquad\qquad (8\text{-}5)$$

for *d* in mm. Thus, for example, if the RGB source were 30 mm above the leaf, and the chamber top were propafilm ($t = 0.95$), then the transm factor τ would be

$$
\begin{aligned}
\tau &= 0.95 \times h(30) \\
&= 0.95 \times 0.6 \\
&= 0.57
\end{aligned}
$$

$(8\text{-}6)$

This calculation can be done automatically for you. When configured for the RGB Source, editing the <transm> node gives you this option (Figure 8-21).

1. Edit the *transm* node.

2. Press **C**.

3. Answer the prompts, press **Y**

4. Done

Figure 8-21. Computing transm for non-standard installations. Not for use with the 6400-17 WPA or 6400-22 Opaque Conifer chambers.

User Calibration

The user calibration of the RGB Source is discussed in **6400-18 RGB Light Source** on page 18-34.

6400-40 Leaf Chamber Fluorometer

The 6400-40 Leaf Chamber Fluorometer has independently controlled red and blue LEDs for providing actinic light to drive photosynthesis. The spectral output of these LEDs is shown in Figure 8-22;
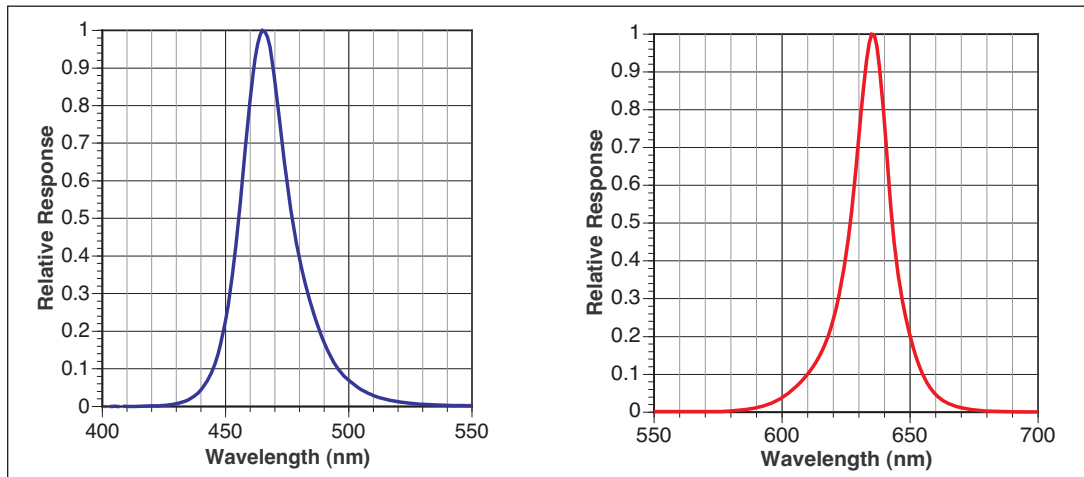
Figure 8-22. Relative spectral outputs of the red and blue LEDs used in the 6400-40 Leaf Chamber Fluorometer.

the blue LEDs are essentially the same as the 6400-02B, while the red LEDs are centered at a shorter wavelength, typically 635 nm, rather than the 670 for the -02B. The aging and temperature effects discussion above for the -02B also applies to the LCF.

Since the LCF allows the red and blue LEDs to be independently controlled, a complication arises when converting the in-chamber light sensor's signal into μmol m^{-2} s^{-1}: One needs to know a calibration factor for the red LEDs, and also one for the blue LEDs. This is provided as part of the factory calibration. But one also needs to know how to weight these two values for a particular situation. The method used by OPEN is based on knowing the fraction of blue radiation at any point in time, and this is determined from the DAC (digital to analog converter) settings that drive the two sets of LEDs. The relation between the DAC setting for a set of LEDs (red or blue) and the actual quantum output comes from the LED calibration curves (generated by the user performing the calibration menu item described in **Calibrate...** on page 27-75). Thus, the accuracy of the in-chamber light sensor depends on the user calibration, when using the 6400-40 LCF with both red and blue LEDs on.

Gallium Arsenide Phosphide (GaAsP) Sensor

The spectral response of a GaAsP sensor is shown in Figure 8-23 on page 8-24, along with the spectral response of an ideal quantum sensor for comparison. The GaAsP sensor spectral response is similar to the ideal quantum sensor, but it begins to drop dramatically at 650nm, the slope from blue to red is generally steeper than ideal, and it's nonlinear. While this is certainly not an ideal quantum response, corrections for spectra of sources commonly encountered in photosynthesis work can be made, and are usually less than ± 15%.

Mixed light sources will cause complications, but if one light source predominates, the appropriate value for that source can be used, or the GaAsP sensor can be calibrated to the specific lighting conditions using a LI-COR quantum sensor (**Generating a Calibration Correction** on page 18-38).
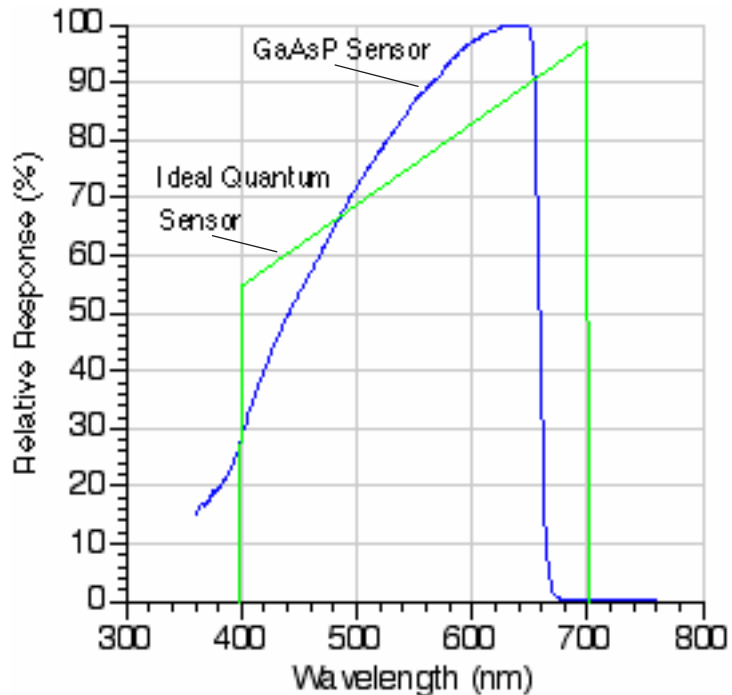
Figure 8-23. Spectral response of GaAsP sensor and ideal quantum response.

Temperature

Our tests have indicated that the GaAsp sensor has a temperature coefficient of about +0.2% C^{-1}.

View Angle

The GaAsP sensor, especially as mounted in the LI-6400 leaf chamber, is subject to serious errors when measuring irradiance at non-normal incidence angles. The primary reason for this is reflections from the chamber walls. For example, at angles when the sun reflects off the wall nearest the sensor, the sensor's readings can be boosted by 25%. For this reason, you should not trust this sensor very much at non-normal incidence angles.

A stronger reason to be careful of non-normal incidence radiation is the potential for the chamber walls to shade the leaf. Uniform lighting is critical to good gas exchange measurements. If sunlit and shaded leaf areas are measured together, it becomes very difficult to interpret the results.

Data Logging

9

What you want, where you want, when you want

9

Data Logging

The LI-6400 provides a great deal of flexibility in choosing what is logged, and how it is logged. This chapter explains what goes on, and how you can modify logging to suit your purposes.

For an introduction to this topic, see **Tour #5: Logging Data** on page 3-61.

Basic Concepts

The LI-6400, in its default configuration, is an open system. As such, it shows a continuous stream of measurements and calculations. Over the course of several minutes, the LI-6400 will make thousands of measurements. The question is: which of these will you want to retain for future use?

One application might only need two sets of data: one when conditions were stable at light level X, and another 5 minutes later at light level Y. Another application, however, might be focused on the dynamics of the change from X to Y, and require data as frequently as possible over that 5 minutes.

Where does it go?
When the LI-6400 records data, it generally does so in its file system, which is described in Chapter 10. Data can also be sent out the RS-232 and/or Ethernet ports in real time. Files are named, allowing you to go back and retrieve your prized data that you stored under some meaningful name like "junk", or "test 1", or the ever-popular "A-Ci curve right before lunch".

What gets stored?
Data files contain "snapshots". That is, whenever you think it appropriate, you can store in your file a set of data for that moment in time. A file can contain any number of such observations. The observations can be recorded manually at your whim; and/or they can be recorded automatically at regular intervals or based on stability criteria.

What does it look like?

A sample file with logged data is shown in Figure 9-1.

Header

```
"OPEN 6.1y"
"Thr Oct  9 2008 13:04:01"
<open><version>"6.1y"</version></open>
<open><light><source>"Sun+Sky"</source><par_in>1 <sensor>"GA-1094"<cal>0.81 <ac
<open><comps><file>"/User/Configs/Comps/MyComputeList"<header></header><extra
<open><prompts><onlog>off</onlog><items>"Default (none)"</items></prompts></ope
<open><stability><items>"Std Stability"<items[1]><id>-2 </id><size>15 </size><pcv><op
<open><log><format>"StdLogFmt_6.0"<items>{ -35 -21 -36 -76 30 23 36 21 25 221
<li6400><factory><unit>"PSC-1094"</unit><serviced>"7 Feb 2007"</serviced><fuseawa
<li6400><user><flow_zero>-195.3</flow_zero><irga_zero><co2>273.4 -468.8<at>26.35
"13:04:06 test"
$STARTOFDATA$
"Obs""HHMMSS""FTime""EBal?""Photo""Cond""Ci""Trmmol""VpdL""CTleaf""Area""BLC_1
1•"13:04:08"•9.5•0•0.03•-6.4E-05•620•-0.00122•1.83•22.01•6•1.42•1•2.84•21
2•"13:04:12"•13.5•0•0.0394•-0.000103•481•-0.00197•1.84•22.00•6•1.42•1•2.8
3•"13:04:15"•17.0•0•0.0428•-0.00011•484•-0.00212•1.85•22.00•6•1.42•1•2.84
```

Observations

Figure 9-1. Sample data file. Each • represents a tab character.

At the start of the file is the header. The header consists of a series of lines containing selected parts of the current system configuration. The data consists of labels identifying each column and rows of observations.

Data files can have Excel counterparts. When this option (described on page 9-17) is enabled, logging data results in two files. A text file (as shown in Figure 9-1) *and* an Excel file (.xls), shown in Figure 9-2. The Excel file has minimal header information, but will always contain all of the information (including the equations) necessary to recompute, should you wish to change an input value such as leaf area.

Figure 9-2. Sample Excel file. Columns with an "out" label (e.g. Row 9) have equations built-in for them. The "in" columns are inputs.

Getting Started

Logging is done in New Measurements mode, and is initiated with the function keys in level 1.

	Open	<view	<close	<add	Match
1	LogFile	file>	file>	remark>	

Open LogFile

Open LogFile (**f1** level 1) will use the Standard File Dialog (page 5-9) to allow you to specify the destination file name (Figure 9-3).

```
Dir: /User
File:Data_
========Log File (esc = none)========
 .DS_Store
 CornLeaf
 CornLeafIRtest
 CornLeafIRtest_.xls
+ Dir    ♦ Help  ♦         ♦ CANCEL♦ SELECT
```
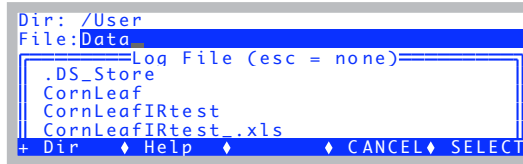
Figure 9-3. Specifying a destination name.

If you do not want to log to a file, press **escape** to get the alternate destination prompt (Figure 9-4), which allows you to log to the Comm port instead of to a file.

```
Log to Comm port ?

              (Y/N)
```

*Figure 9-4. The alternate destination prompt. Press **Y** for Comm port, or* **escape** *to cancel logging.*

If you wish to log to a file <u>and</u> to the Comm port simultaneously, use the log options feature described on page 9-18.

Logging Remarks

Once the destination is established, you'll be prompted for a remark (Figure 9-5). This initial remark is recorded on the line just prior to the label line. See Figure 9-1 on page 9-3.
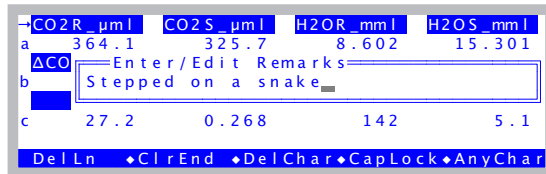
Figure 9-5. The prompt for entering remarks into a log file. The details of this type of dialog box are described in **Standard Line Editor** *on page 5-5.*

Remarks can also be entered at any time while the file is open by pressing **Add_Remark** (**F4** level 1), and will take the form of a quoted string on its own line in the file. The start of a remarks line contains the time (HH:MM:SS) the remark was entered (Figure 9-6).


```
6·1393.97·−2.97·0.219·27.1·3.85·1.76·3.5·13…
"11:58:30 Stepped on a snake"
7·1585.53·21.4·0.207·759·3.64·1.76·3.5·1·3·…
```
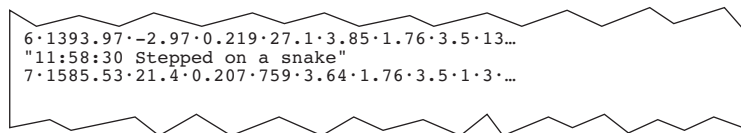
Figure 9-6. Remarks are logged as quoted strings, including the time stamp.

Alternatively, user-entered remarks or constants can take the form of additional data columns, rather than occupying an entire record. See **Prompts and Remarks** on page 9-20.

Logging Observations

Once the destination and initial remarks are established, the **f1** level 1 function key label will show the number of observations logged (Figure 9-7).
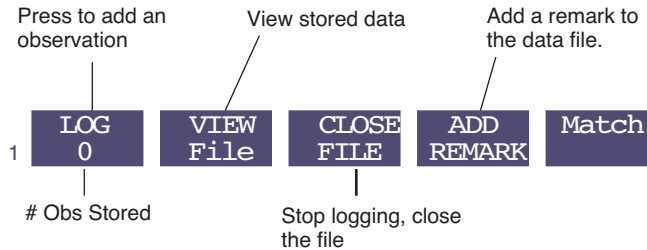
Figure 9-7. Level 1 function key labels when logging is active.

Observations can be added to the logged data by pressing **f1** (**LOG**) or the log button on the left hand side of the sensor head handle. To log with the button, press and hold it for about 1 second, or until you hear the beep (if it is enabled) indicating an observation has been logged. The log button normally does not function unless logging is active.

The log button can be disabled by unplugging it (Figure 2-13 on page 2-17), or redefining its action (described next).

User Definable Log Button

What the log button does when pressed (while in New Measurements mode) is user-definable (Figure 9-9).
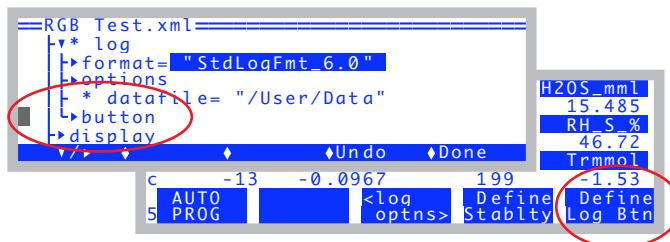
Figure 9-8. Define the log button from the Config Tree, or else from New Measurements mode
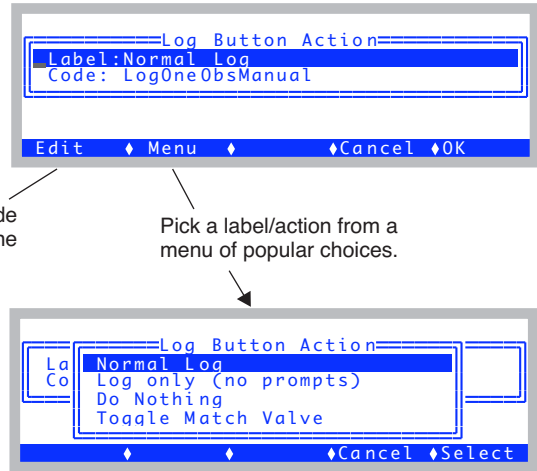
Figure 9-9. The Log Button definition dialog.

Items in the list of possibilities shown by the **Menu** key (**f2**) come from /Sys/Open/StdLogButtonMethods, and - if configured for the 6400-40 LCF - /Sys/Open/FlrLogButtonMethods, and also from /User/Configs/UserLogButton-Methods, if it exists. Thus, if you wish to add some favorites of your own, put them in the latter file. The format is illustrated in the listings shown in Figure 9-10 and Figure 9-11, and is quite simple: two quoted strings on each line. The first string is the label, and the second is the code to be executed.

The difference between "Normal Log" and "Log only" is that user prompts (described in **Prompts and Remarks** on page 9-20) are skipped in the latter.

```
"Normal Log" "LogOneObsManual"
"Log only (no prompts)" "LogOneObsNoComp"
"Do Nothing" "Noop"
"Toggle Match Valve" "IsIrgaMatch IF IrgaMatchOff ELSE IrgaMatchOn THEN"
```

Figure 9-10. Contents of /Sys/Open/StdLogButtonMethods.

```
"MeasureToggle_OnOff" "IsFMEasOn IF FMEAS_OFF ELSE FMeas_on THEN"
"FarRedToggle_OnOff" "IsFarRedOn IF FarRed_Off ELSE FarRed_On THEN"
"ActinicToggle_OnOff" "IsActinicOn IF Actinic_Off ELSE Actinic_on THEN"
"Do Fm" "KDoFm"
"Do FoFm" "KDoFoFm"
"Do Fm'" "KDoFmp"
"Do FsFm'" "KDoFsFmp"
"Do Fo'" "KDoFop"
"Do FsFm'Fo'" "KDoFsFmpFop
"Do Fo + Log" "SetFo LogOneObsManual"
"Flash (No assign)" "DoFlash"
"Dark (No assign)" "DoDark"
```

Figure 9-11. Contents of "/Sys/Open/FlrLogButtonMethods".

The fluorescence options are described in Table 27-14 on page 27-86, but essentially correspond to many of the LCF control function keys. If prompts are set to "On Log" (see Figure 9-34 on page 9-24), then the fluorescence commands that include logging ("Do..") will also trigger prompts, when executed via the log button.

Determining What is Logged

The values logged and their arrangement are determined from the \<open\> \<log\> \<format\> and \<open\> \<log\> \<options\> nodes in the Configuration editor (Figure 9-12).
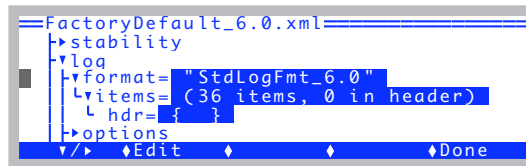
Figure 9-12. The nodes responsible for logging format, accessed by View/Edit *in the* Config Menu.

Log Format

Editing the <log> <format> node brings up the dialog that shows a label and the list of items to be logged in the file, in the order in which they are logged (Figure 9-13). The Label: line is not logged - it just provides a reference name, for your information.

'r' items are required for recomputing.

'H' items are constants which will be put in the header, instead of columns.
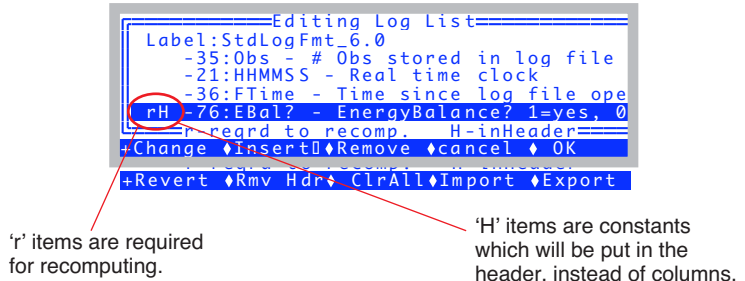
Figure 9-13. The Log Format Editor.

f1 (**Change**) and **f2** (**Insert↓**) allow you to replace or add a variable in the list by picking from the list of all currently defined user and system variables. Items already in your log list are marked with an asterisk (Figure 9-14).
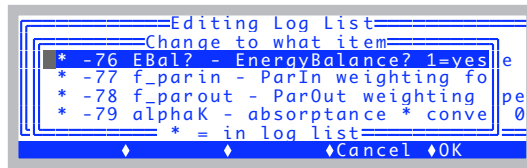
Figure 9-14. Picking from the list of user and system variables.

f3 (**Remove**) will remove an item (except the label).

f1 level 2 (**Revert**) will undo all changes. (Same as **Cancel**, then re-entering the dialog, but with much less fuss).

f2 level 2 (**Add Hdr** / **Rmv Hdr**) is present only when the cursor is on a constant value (as opposed to a computed value), such as Area, or EBal?, and serves to toggle the designation of that item as a header or not. See **Header Constants** on page 9-12.

You can rearrange items in the list using **shift** + ↑ and **shift** + ↓ (Figure 9-15).
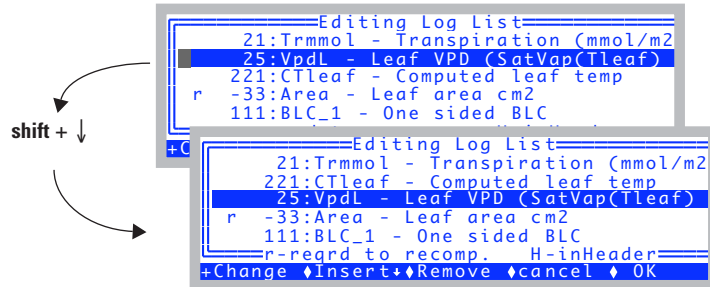
Figure 9-15. Rearranging the order. Items will appear in the text log file in the order they are listed here.

The **Import** and **Export** keys allow you to save or recall just the log format, separately from the complete system configuration. NOTE: Normally, you never need to mess with this, because the log format information *is* saved with the complete system configuration.

Exported log formats are stored in /User/Configs/LogFormats. When you press **Import**, you are prompted to pick one of those files, or you can extract a log format from a complete system configuration, which live in /User/Configs/UserPrefs.

Pick an exported log list file...

... or extract log list information from a system configuration file.
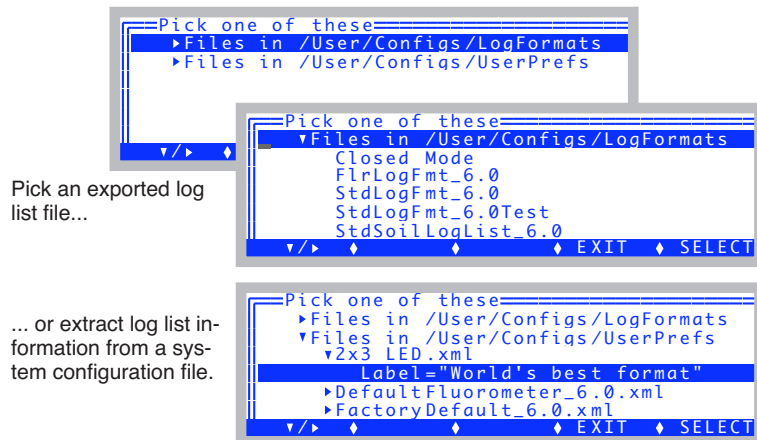
Figure 9-16. Importing a log format. It can come from an exported list, or be extracted from another system configuration file.

File Formats

Log Lists, as stored in the /User/Configs/LogFormats directory use the old format (Figure 9-17). The log information in the new system configuration .xml files, have a different format (Figure 9-18).

```
LogFormat=
-35 -36
30 23 36 21 25
-33 -34 -32 -9 -10 -8 -1 -2 -4 -5 -14 -15 -7 -12 -13 -11 -65 -66 -23
```

Figure 9-17. Old format for log list files. The numbers following the marker indicate which items (system or user variables) are to be logged. These numbers can be on any number of lines; it does not matter how they are distributed.

```
<open>
  <log>
    <format>"StdLogFmt_6.0"
      <items>{ -35  -21  -36  -76  30  23  36  21  25  221  -33…}
        <hdr>{  }</hdr>
      </items>
    </format>
    <options>
      <beep>on</beep>
      <hdr>normal</hdr>
      <rem>normal</rem>
      <stab>no</stab>
      <stats>no
        <period>15 </period>
      </stats>
      <excel>yes</excel>
      <comm>no</comm>
    </options>
    <datafile>"/User/Data"</datafile>
    <button>
      <action>"Normal Log"
        <code>" LogOneObsManual "</code>
      </action>
    </button>
  </log>
</open>
```

Figure 9-18. Logging information is stored in system configuration .xml files.

Header Constants

Normally, all variables (such as "PHOTO") and constants (such as "AREA") in LI-6400 data files are in columns. It is possible, however, to move quantities that you don't expect to change over the course of logging your data out of their column and into the file header. For example, if we were to declare "EBal?" and "AREA" as header variables, the data file would look like Figure 9-19.

```
"OPEN 6.1"
"Thr Oct  9 2008 14:07:44"
<open><version>"6.1y"</version></open>
<open><light><source>"Sun+Sky"</source><par_in>1 <sensor>"GA-10!
<open><comps><file>"/User/Configs/Comps/StdComps_6.1"<header>""
<open><prompts><onlog>off</onlog><items>"Default (none)"</items:
<open><stability><items>"Std Stability"<items[1]><id>-2 </id><s
<open><log><format>"StdLogFmt_6.0"<items>{ -35  -21  -36  -76
<li6400><factory><unit>"PSC-1094"</unit><serviced>"7 Feb 2007"<
<li6400><user><flow_zero>-195.3</flow_zero><irga_zero><co2>273.
"Const= · "-76 · "EBal? · "0
"Const= · "-33 · "Area · "6
"14:07:49 test"
$STARTOFDATA$
"Obs · ""HHMMSS · ""FTime · ""Photo · ""Cond · ""Ci · ""Trmmol ·
1 · "14:07:51 · "20.5 · -10.4 · -0.00583 · -2.31E+03 · -0.126 · 2.
2 · "14:07:53 · "22.5 · -8.79 · -0.00595 · -1.83E+03 · -0.126 ·
3 · "14:07:56 · "25.5 · -8.42 · -0.00615 · -1.66E+03 · -0.13 · 2
4 · "14:07:58 · "27.5 · -8.39 · -0.00631 · -1.59E+03 · -0.134 ·
"Const= · "-33 · "Area · "3
5 · "14:08:06 · "35.0 · -20 · -0.0128 · -1.95E+03 · -0.274 · 2.
6 · "14:08:08 · "37.0 · -21.4 · -0.0121 · -2.27E+03 · -0.259 ·
```

Figure 9-19. Text file format with "EBal?" and "Area" declared header constants. As such, they appear in the header, and anytime while logging that they change.

The values are no longer in the data columns, but instead, appear in "Const=" lines in the header. If a declared header constant value changes while the log file is open (e.g. you press the **Area** function key, and enter a new value), another "Const=" row will be inserted in the data file to note the change.

Header constants are handled in a similar manner in Excel files.

Figure 9-20. Excel file format with "EBal?" and "Area" declared header constants. If you change the Area value in cell B9, observations 1, 2, and 3 will update (rows 13-16). If you change the value in Cell A19, observations 5, 6, and 7 will update (rows 21-23).

Log Options

Log Options can be set either from the Config Menu (View/Edit), or from New Measurements mode. The **Log Options** key (**f3** level 5) brings up the list of options (Figure 9-21). Note: some options will not be editable if a log file is open.
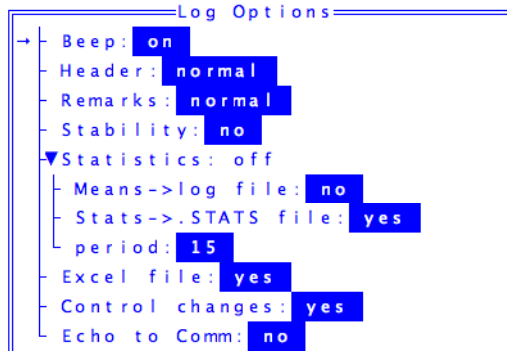
Figure 9-21. The Log Options editor, and configuration tree nodes.

Use the **Edit** (**f2**) key to change any of the settings. Exit with either **OK** (**f5**) to keep the changes, or **Cancel** (**f4**) to ignore the changes.

Beep
The choice is "on" or "off". Controls whether the console beeps when an observation is logged.

Header
The choice is "normal" or "separate". If you select "separate", then your data file will not have any of the normal header information above the line labels. This will instead be put into a separate file (Figure 9-22).

"MyData"

```
$STARTOFDATA$
"Obs","Time","Photo","Cond","Ci","Trmmol","VpdL","Area",…,"Status"
1,78.8,10.2,41.9,1.27E+03,19.8,0.709,6,  111105
2,138.8,10.1,11.6,1.28E+03,12.7,0.529,6,  111105
3,206.3,1.85,1.44,1.27E+03,2.92,0.289,6,  111105
4,261.0,2.09,2.49,1.28E+03,4.61,0.328,6,  111105
```

"MyData.HDR"

```
""OPEN 6.2"
"Thr Sep  1 2011 14:07:44"
<open><version>"6.2"</version></open>
<open><light><source>"Sun+Sky"</source><par_in>1 <sensor>…
<open><comps><file>"/User/Configs/Comps/StdComps_6.1"<head…
<open><stability><items>"Std Stability"<items[1]><id>-2…
<open><log><format>"StdLogFmt_6.0"<items>{ -35  -21  -36  …
<li6400><factory><unit>"PSC-1094"</unit><serviced>"7 Feb …
<li6400><user><flow_zero>-195.3</flow_zero><irga_zero><co2…
```

Figure 9-22. A data file, and its separate .HDR file.

Remarks
The choice is "normal" or "separate". If you select "separate", all remarks will go into a .REM file. This can really clean up a fluorescence file, which generates remarks from flash and dark pulse events.

Stability
The choice is "no" or "logged". If you choose "logged", your data file will contain extra columns of data. The mean, standard deviation, CV, and rate of change (slope) of each variable in the stability list will be appended to each line. The rate of change is on a per minute basis (Figure 9-23).

```
Regular data    Photo stats                                      Cond Stats


"Obs",..."Status",N(Photo),MN(Photo),SD(Photo),CV(Photo),SLP(Photo),N(Cond),...



              # Samples, Mean, Std Dev, CV (%), Slope
```
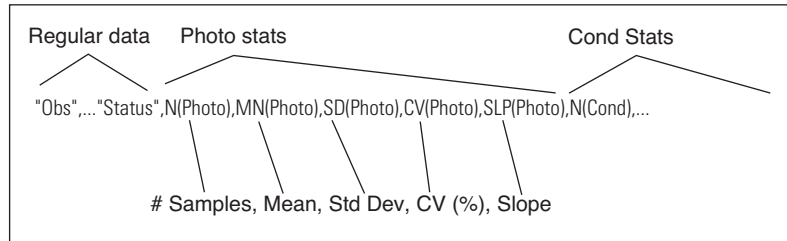
Figure 9-23. Stability details (in this case for PHOTO and COND) are appended to the normal data record. There are five extra columns for each variable in the stability list.

Statistics

Normally, logged values of measurements and computations are an average of the previous few seconds ('few' depends on how <open> <a2d> <avg-time> is set). If either of the statistics options (below) is on, it will cause statistics (mean and standard deviation) to be kept on the appropriate floating point variables[1] that are being logged. The time period for those statistics is determined by the Period entry. 15 or 20 seconds is generally appropriate, but it can be any length that makes sense for the experiment.

...Means→Log File

This can be "off" or "on". When on, this option causes the following two additional steps to be taken when a data record is logged, before anything is actually written to the file:

1. The latest values of variables are replaced by their mean values from the running statistics.

2. User computations (Photo, Cond, etc.) are performed again.

Thus, the logged data record will have inputs (e.g. CO2S) that are 15 second (or whatever) averages, and computed quantities (e.g. Photo) that are computed from those long averages.

Note: This feature is ignored for closed mode operation, which includes configurations for the Soil Chamber or the Custom Closed Chamber configuration.

---

[1] Excluded: sys and user constants, and variables relating to fluorescence events, matching, time of day, etc.

...Stats→.stats File

The choice is "yes" or "no". When set to "yes", the complete set of statistics is logged in a .STATS file that mirrors your log file. For example,

The data file: "SampleData"

```
                ⋮
"Obs","HHMMSS","FTime","Photo","Cond","Ci","Trmmol","VpdL","Area","StmRat","BLCond","Tair",...
1,"12:39:09",3.2,1.74E-07,3.19E-09,-39.8,1.84E-08,0.505,6,1,2.84,0.00,0.01,0.00,47.33,47.28,...
2,"12:39:13",8.2,2.34E-05,-1.66E-07,271,-9.59E-07,0.505,6,1,2.84,0.01,0.01,0.01,47.45,47.30,...
3,"12:39:18",14.2,-1.47E-05,-3.01E-08,-733,-1.73E-07,0.505,6,1,2.84,0.01,0.01,0.01,47.23,47.30,...
```

The stats file: "SampleData.STATS"

```
"Thr Jun 20 2002 12:39:05"
Period= 15 secs
"Obs","HHMMSS","FTime","MN(Photo)","SD(Photo)","MN(Cond)","SD(Cond)","MN(Ci)","SD(Ci)",...
1,"12:39:09",3.2,-3.471e-06,4.57e-05,-4.978e-08,2.407e-07,-186.8,381.8,-2.871e-07,1.387e-06,...
2,"12:39:13",8.2,8.319e-06,3.591e-05,-8.377e-08,2.229e-07,-51.49,1003,-4.829e-07,1.284e-06,...
3,"12:39:18",14.2,5.004e-06,3.601e-05,-6.486e-08,1.849e-07,-499.2,2526,-3.739e-07,1.065e-06,...
```

Figure 9-24. Example of a .STATS file. Each relevant floating point variable logged in the data file has a corresponding mean (MN) and standard deviation (SD) logged in the stats file.

...Period

This entry appears only if Stats→.stats File or Means→Log File is set to "yes". It determines the time period (secs) of the running stats to be computed.

Excel File

The choice is "yes" or "no", and it determines if an Excel file (.xls) is built along with the normal text file. If you aren't using Excel for post processing, turn this feature off, and you'll save a lot of file space, since .xls files are typically 10 times larger than the text file.

Control Changes

The choice is "yes" or "no". If "yes", anytime you make a control set point change (flow/humidity control, CO_2 control, temperature control, light control), a remark will be added to your log file (if one is open, of course) indicating the new setting (Figure 9-25).

```
     :
"12:48:41 test"
$STARTOFDATA$
"Obs""HHMMSS""FTime""EBal?""Photo""Cond""Ci""Trmmol"...
1 "12:49:09 "41.0 0 6.37 0.0441 -7.16 0.914 2.4
"12:49:25 CO2 Mixer: CO2R -> 300 uml"
2 "12:50:09 "101.5 0 10.7 0.0441 -191 0.914 2.4
     :
```

Figure 9-25. Logging control changes, which appear as time-stamped remarks in the log file.

Echo to COMM

The choice is "yes" or "no". When "yes", each time an observation is logged, it is also written to the RS-232 port. If you are using LI6400XTerm (RS-232 or Ethernet), you can capture this stream of data by using the Comm window.
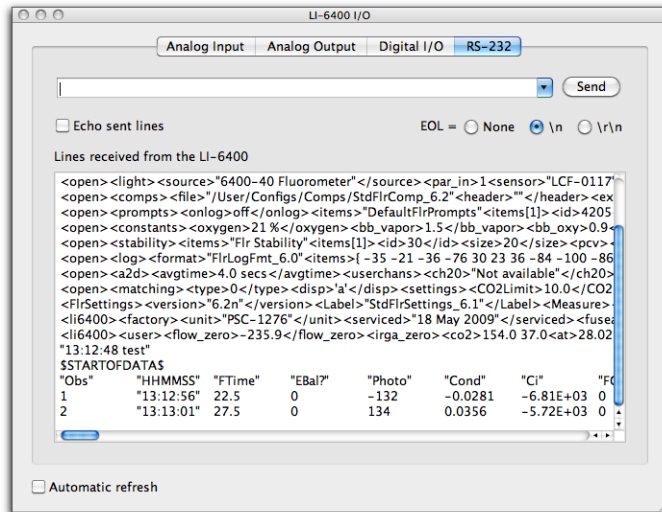
Figure 9-26. Using the Comm Monitor window of LI6400XTerm to monitor data with OPEN's "Echo to Comm" log option enabled.

Logging Times and Dates

The default log list includes an observation time in HH:MM:SS format, and also the number of seconds that have elapsed since the file was opened (*FTime*). Table 9-1 lists all the system variables that contain time and date information. To include any of these in your log file, add them to your log list (**Log Format** on page 9-9).

Table 9-1. System variables involving time and date.

ID	Label	Description	Example
-21	HH:MM:SS	Clock time string, 24 hour.	"12:02:54"
-36	FTime	Number of seconds since the file was opened.	1234.5
-64	DecHour	Decimal hour of observation.	15.0237
-69	DOY	Day of the year (1…366)	125
-70	YYYYMMDD	Year, month, and day	20070811

Plotting HH:MM:SS (-21)

If you are plotting data on the LI-6400 using GraphIt (Chapter 12), then you can use HH:MM:SS as the time variable, because GraphIt will convert the string to decimal hours automatically when plotting. For example, "10:40:17" would be treated as 10.67139. If you need to plot the data using your spreadsheet, you will need to consider whether or not it will handle HH:MM:SS in a similar manner.

Prompts and Remarks

What Are My Options?

If you wish to add extra information to your log file, and want it to come from typing, as opposed to being measured, here are your options:

- **Use the Remarks Feature**

    Whenever you want to record something, press **f4** level 1 (**Add Remark**), and a line will be inserted into the file with your comments, along with a time stamp. The advantage of this is that there is nothing to set up - this is available anytime.

    ```
    "Obs"•"HHMMSS"•"FTime"•"Photo"•"Cond"•"Ci"•"Trmmol"•"VpdL"•"Area"•"StmRat"•...
    1•"12:39:09"•3.2•1.74E-07•3.19E-09•-39.8•1.84E-08•0.505•6•1•2.84•0.00•0.01•0.00•...
    2•"12:39:13"•8.2•2.34E-05•-1.66E-07•271•-9.59E-07•0.505•6•1•2.84•0.01•0.01•0.01•...
    "12:48:16 something's out there..."
    3•"12:49:58"•113.2•-1.47E-05•-3.01E-08•-733•-1.73E-07•0.505•6•1•2.84•0.01•0.01•0.01...
    ```

    Figure 9-27. A remark line in a data file.

- **Use the Prompts Feature**

    The Prompts feature is a list of items you'd like to be prompted for. The list can include system constants like leaf area, as well as constants you define, like Plot#, etc. Unlike the standard remarks, these values will be columns in the data set, unless you declare them header constants - see **Header Constants** on page 9-12.

    ```
    "Obs"•"HHMMSS"•"Plot#"•"LeafCode"•"Trtmnt"•"FTime"•"EBal?"•"Photo•…
    1•"10:21:37"•5512•"AC-12-FF"•"994426G-01"•67.0•0•-0.189•-0.000171•…
    2•"10:21:40"•5512•"AC-12-FF"•"994426G-01"•70.0•0•-0.193•-0.000176•…
    3•"10:22:13"•5520•"AC-13-FF"•"994426G-02"•103.5•0•-0.278•-0.000248•…
    4•"10:22:15"•5520•"AC-13-FF"•"994426G-02"•105.0•0•-0.294•-0.000263•…
    ```

    Figure 9-28. Three user defined constants in the data columns. "Plot#" is numeric, formatted as integer values. "LeafCode" and "Trtmnt" are handled as strings.

    Constants can come from two sources: 1) use one of the nine system constants reserved for your use (described next), or 2) create a user-defined object (described in Chapter 15) that is a constant.

Defining Prompts

Prompts are defined in the <open> <prompts> node of the system configuration.

Edit the <items> node, or any of its subnodes, to access the Prompt Editor

2nd level->
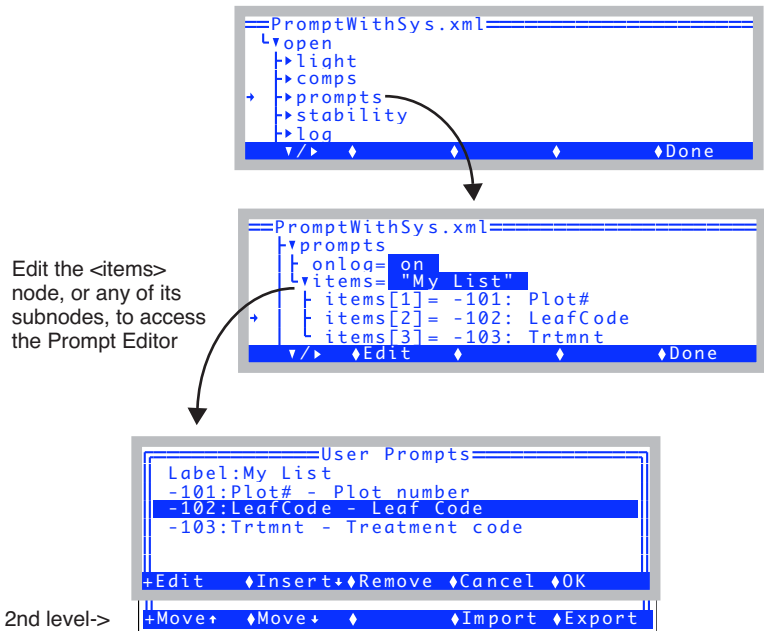
Figure 9-29. The Prompt Editor is accessed from the system configuration tree.

The prompt configuration shown in Figure 9-29 will prompt for "Plot#" (an integer), "LeafCode" (a string), and "Area" (a floating point value) whenever **f5** level 3 is pressed in New Measurements mode. This sequence can also be made to happen whenever the Log key is pressed by setting the <open> <prompts> <onlog> node to ON.

Edit (**f1**) allows you to edit the current item, or label. **Insert↓** allows you to add to the list. Either way, you pick the new item from a list of available constants (Figure 9-30 on page 9-22).

Remove (**f3**) removes an item from the list (except for the label).

Move↑ (**f1** level 2) and **Move↓** (**f2** level 2) can be used to shift the order of the items in the list, except for the label. Note that **shift+↑** and **shift+↓** will also shift items.

The **Import** and **Export** keys allow you to import and export prompt configurations separate from the overall system configuration. See **Prompt List Files** on page 9-29.

Adding/Editing Constants

Adding or editing a Prompt item brings up the dialog shown in Figure 9-30.
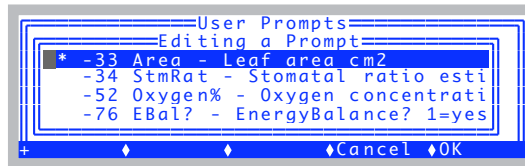
Figure 9-30. Pick the item to be added from the list of system and user defined constants.

The list of items from which you pick is arranged in descending order of ID#. User items (if any) are first, followed by system items (Figure 9-32).
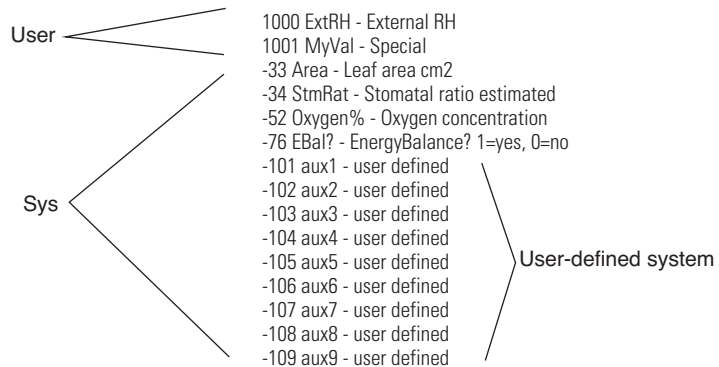
Figure 9-31. Typical list of constants. User items have IDs > 0, while system items have IDs of 0 or less.

User-defined System Constants

If the selected ID is -101 to -109, (a system constant reserved for users), then you get to define several of its attributes (Figure 9-32).
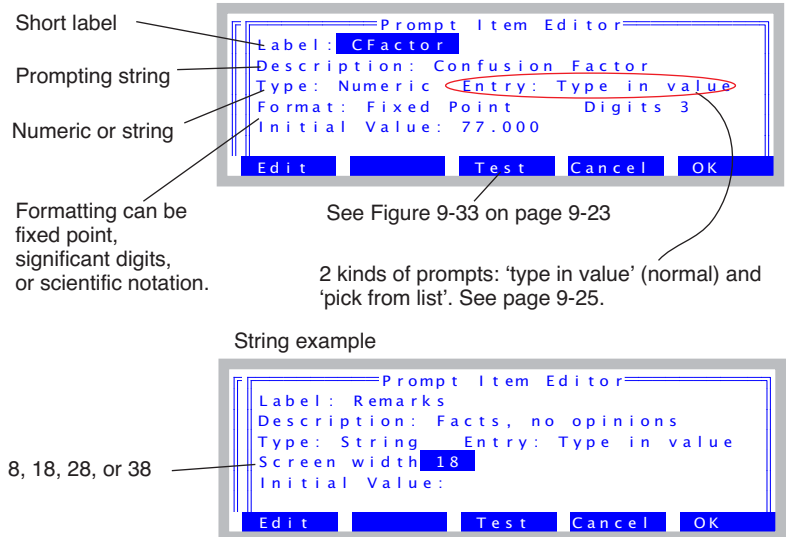
Figure 9-32. Defining a

The **Test** button (**f3**) lets you "try out" your definition (Figure 9-33). You are prompted for the value, then shown what it will look like in New Measurements mode, if you add it to the display.
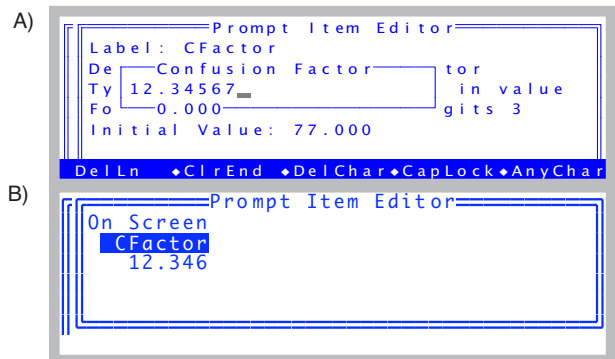
Figure 9-33. The Test button sequence: A) The test prompt B) The sample display.

Prompts in New Measurements Mode

Function keys in level 3 include the prompt controls. **PromptAll** (**f5**) will prompt you with the items in your prompt list. **Prompts on/off** (**f4**) toggles whether or not you are taken through the prompt list when you manually log an observation. The **Sys&User Consts** key (**f3**) allows you to view and edit all currently relevant constants, whether they are in the list of things being prompted for or not. There are also 7 keys you can make prompt for any constant.

Pick and choose what you edit from a list

Prompt for everything in sequence

Any of these 7 keys can be assigned to a prompt for particular constant.
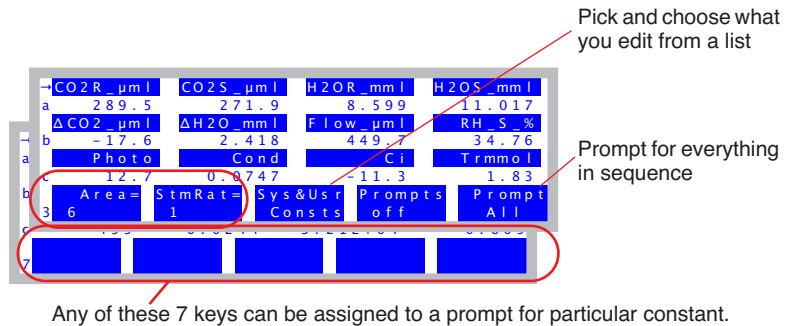
Figure 9-34. Level 3 and 7 function keys are prompt-related.

Interaction With Real Time Graphics

When you press the log button with "Prompt on Log" and RTG is active and displaying graphs, you are taken out of graphics mode for the prompts, then put back in.

Timing relative to logging

When you manually log, here is the sequence of events:

1. Latest measurements are captured.
2. If "Prompt on Log" is enabled, prompts are presented, user responses recorded.
3. Computations performed.
4. Record added to log file.
5. Beep (if enabled).

Thus, if you are prompting for values that are used in computations (e.g. leaf area), the computations done for the logged data will use the latest values that you have entered.

Assigning a Fct Key to a Prompt Item

To assign a fct key to prompt for a constant, go to the Config Menu, select View / Edit..., and edit the *fctkeys* node (Figure 9-35).

Pick the fct key to define, press Edit.

Each key has a mode. Select 'Prompt for a constant'.

Then set the Target to the constant to be prompted.
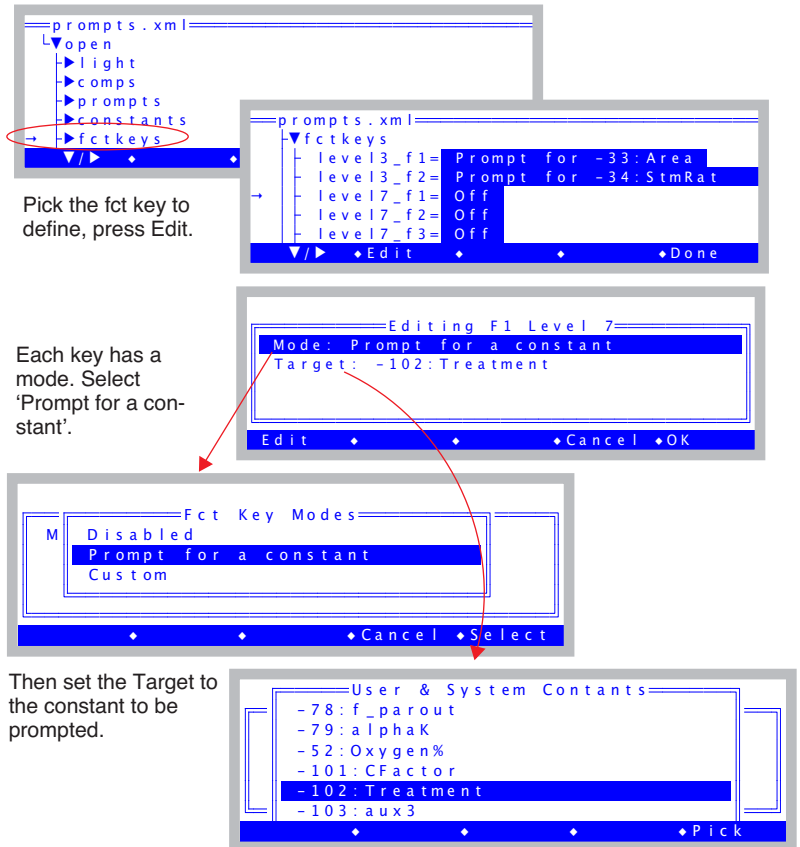
Figure 9-35. Defining a fct key to prompt for a constant.

'Pick From List' Prompts

When editing prompt items that have IDs from -101 to -109, you have a choice of making them 'type in value' or 'pick from list'. When prompted for the latter type, the user will be shown a list of potential replies, and she simply picks from the list. For example, the list could be a bunch of fairly complicated number / letter combinations defining treatment types.

Figure 9-36 shows how to set this up.

1. Set *Entry:* to *Pick From List*.

```
                ┌──Prompt  Item  Editor──┐
┌─│─────────────────────────────────────────────┐
│ │ Label:  Treatment                          │
│ │ Description:  Treatment  Code              │
│ │ Type:  String      Entry:  Pick  From  List│
│ │ Screen  width  18                          │
│ │ Initial  Value:                            │
│ │ PickFile:                                  │
│ │ Edit              Test    Cancel    OK     │
└─────────────────────────────────────────────┘
```
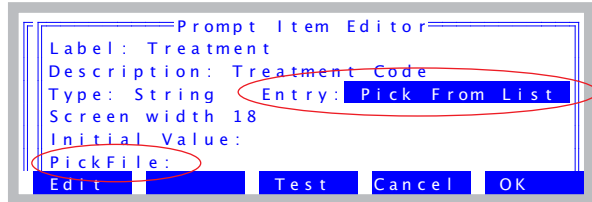
2. Highlight *Pick-File*, and press Edit.

...you can make a new file, or pick an existing one. If you press N for new file...

```
                ┌──Prompt  Item  Editor──┐
┌─│─────────────────────────────────────────────┐
│ │ ┌──Picking  entries  from  a  file──┐       │
│ │ │ E  –  select  an  Existing  file  │ List │
│ │ │ N  –  make  a  new  file          │       │
│ │ │                    ( E / N )       │       │
│ │ └────────────────────────────────────┘       │
│ │ PickFile:                                   │
└─────────────────────────────────────────────┘
```
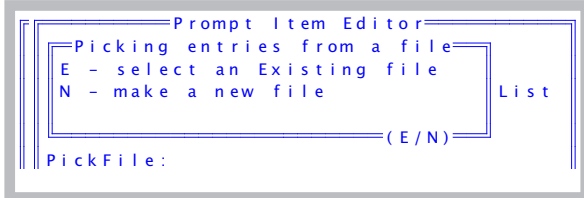
...you can type in the list of possible responses. Press OK and you can name it and save it.

```
                ┌──Type  one  entry  per  line──┐
┌─────────────────────────────────────────────┐
│ AT–189376–001                               │
│ AT–163809–070                               │
│ AT–435985–001                               │
│ AT–446410–071                               │
│ XC–011–011–02                               │
│                                             │
│ +DelLn    ◆ClrEnd  ◆DelChar◆Cancel  ◆OK    │
└─────────────────────────────────────────────┘
```
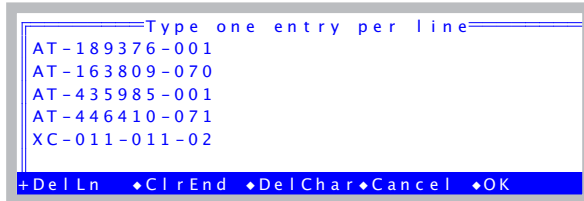
3. Done with the setup. Press OK.

```
                ┌──Prompt  Item  Editor──┐
┌─│─────────────────────────────────────────────┐
│ │ Label:  Treatment                          │
│ │ Description:  Treatment  Code              │
│ │ Type:  String      Entry:  Pick  From  List│
│ │ Screen  width  18                          │
│ │ Initial  Value:                            │
│ │ PickFile:  ..Configs/myTreatmentCodes      │
│ │ Edit              Test    Cancel    OK     │
└─────────────────────────────────────────────┘
```
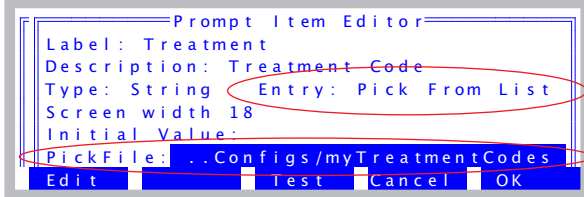
In New Measurements mode (or when you press Test), this prompt will look like this:

```
→CO2 ┌──Select  value  for  Treatment──┐ _mml
a   2 │ AT–189376–001                  │ .013
ΔCO  │ AT–163809–070                  │ _S_%
b  –1 │ AT–435985–001                  │ 4.74
     │ AT–446410–071                  │ mmol
c    │ XC–011–011–02                  │ .611
  A  └─'E'–sel+edit  '^E'–edit  all──┘ rompt
  Sort    ◆  Find   ◆  ReFind◆ CANCEL◆ SELECT
```
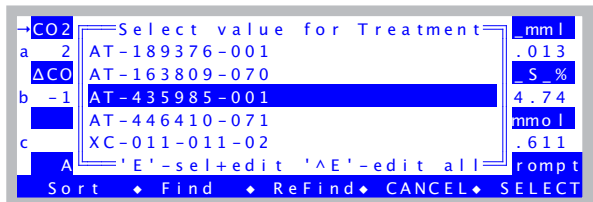
Figure 9-36. Setting up a prompt to be 'Pick From List' and using it.

Notice the bottom label in the prompting window when a 'pick from list' prompt is asked for (bottom of Figure 9-36):

```
⌐ 'E'-sel+edit '^E'-edit all⌐
```

This is a reminder of how you can handle exceptions when you are actually using the pick list. In addition to simply picking an entry, you could select one and press **E**, or just press **ctrl+E**.
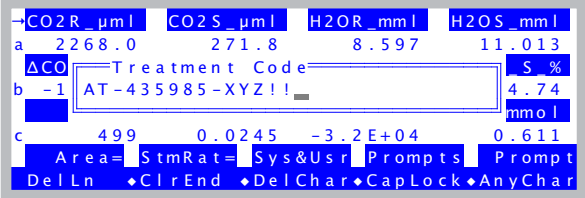
E: Temporary change

Figure 9-37 illustrates how to use the **E** option to select an item and modify it temporarily. This option does *not* change the stored file of responses: the first time you pick a different entry, the edited one will disappear from the list.

Press **E**...

```
→CO2═══Select value for Treatment═══_mml
a   2 AT-189376-001                 .013
ΔCO AT-163809-070                   S %
b  -1 AT-435985-001                 4.74
      AT-446410-071                 mmol
c     XC-011-011-02                 .611
  A═══'E'-sel+edit  '^E'-edit all═══rompt
  Sort   ◆ Find  ◆ ReFind◆ CANCEL◆ SELECT
```

...to edit and use the selection

```
→CO2R_μml   CO2S_μml   H2OR_mml  H2OS_mml
a  2268.0     271.8     8.597    11.013
ΔCO═══Treatment Code═══         S %
b  -1 AT-435985-XYZ!!_           4.74
                                 mmol
c      499     0.0245   -3.2E+04     0.611
   Area= StmRat= Sys&Usr Prompts   Prompt
  DelLn   ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```

The *next* time you are prompted, the edited item is appended to the list

```
→CO2═══Select value for Treatment═══_mml
a   2 AT-163809-070                 .018
ΔCO AT-435985-001                   S %
b  -1 AT-446410-071                 4.77
      XC-011-011-02                 mmol
c     AT-435985-XYZ!!               0.611
  A═══'E'-sel+edit  '^E'-edit all═══rompt
  Sort   ◆ Find  ◆ ReFind◆ CANCEL◆ SELECT
```

Figure 9-37. Temporarily modifying an entry in the list.

Ctrl+E: Permanent Change

If you highlight an entry and press **ctrl+E**, you get to edit the *entire* list, and then pick your item. This option *does* change the stored list of responses.

System Variables for Prompts

Table 9-2 lists the system variables that are available for use with prompts. These are the items that will be in the list in Figure 9-31 on page 9-22.

Table 9-2. System Variables that can be used as prompts.

ID	Label	Description	LPL Variable Name[a]
-33	AREA	Leaf area (cm^2)	*area_cm2*
-34	STMRAT	Stomatal Ratio	*stom_rat*
-52	Oxygen%	Percent oxygen	*oxyPct*
-55	BLC1_mol[b]	One sided BL cond.	*condBL_one*
-76	EBal?	Energy balance flag	*doEB*
-86	Fo	Minimal fluorescence[c]	*flr_fo*
-88	Fm	Maximal fluorescence	*flr_fm*
-101	Aux1[d]		
-102	Aux2		
-103	Aux3		
-104	Aux4		*auxn.floatVal*
-105	Aux5	Can be string or numeric.	or
-106	Aux6		*auxn.stringVal*,
-107	Aux7		where *n* = 1,9
-108	Aux8		
-109	Aux9		

a.Necessary if you wish to use the value in a computation, such as in a Compute List, where you need to refer to the item by its name, not its label.
b.Available only when configured for a fixed value boundary layer conductance.
c.Available when configured for fluorescence.
d.Items -101 through -109 have user defined labels. What's shown here are default labels.

If you need more than 9 extra constants, you can define an unlimited number of new ones. See Chapter 15.

Prompt List Files

Prompt List definitions are part of the overall system configuration, and as such are stored along with everything else when you do Save As... from the Config menu.

The Prompt Editor, however, does allow you to read prompt lists from the file system, from one of two sources: Either the directory /User/Configs/Prompts, in which previously exported definitions can be found, or they can be extracted from system configuration files stored in /User/Config/UserPrefs.The **Import** key (**f4** level 2) brings up a dialog that lets you pick (Figure 9-38).
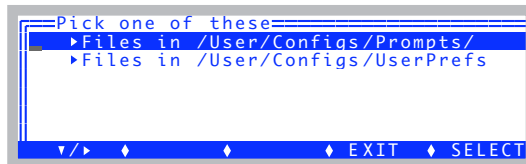
Can import a pre-
viously exported
file stored here...
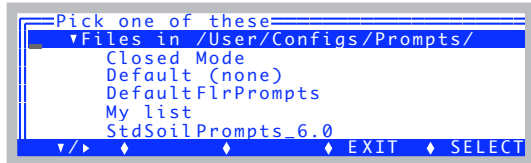
...or extract a
prompt definition
from a system
configuration,
stored here.
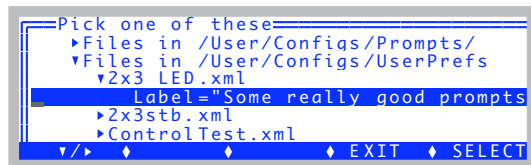
Figure 9-38. Importing a prompt list.

File Format

The format of Prompt definitions as part of the system configuration is shown in Figure 9-39.

```
<open>
<prompts>
<onlog>off</onlog>
<items>"MyList"
<items[1]>
 <id>-33 </id>
 <label>"Area"</label>
 <desc>"Leaf area cm2"</desc>
 <type>0 </type>
 <strscrlen>0 </strscrlen>
 <fmt1>0 </fmt1>
 <digs>0 </digs>
 <pick>0</pick>
 <pickFile>""</pickFile>
</items[1]>
<items[2]>
 <id>-101 </id>
 <label>"Plot#"</label>
 <desc>"Plot number"</desc>
 <type>0 </type>
 <strscrlen>0 </strscrlen>
 <fmt1>1 </fmt1>
 <digs>0 </digs>
  <pick>0</pick>
 <pickFile>""</pickFile>
</items[2]>
<items[3]>
 <id>-102 </id>
 <label>"LeafCode"</label>
 <desc>"Leaf code"</desc>
 <type>1 </type>
 <strscrlen>2 </strscrlen>
 <fmt1>1 </fmt1>
 <digs>0 </digs>
 <pick>1</pick>
 <pickFile>"/User/Configs/LeafCodes"</pickFile>
 </items[3]>
 </items>
 </prompts>
</open>
```

Figure 9-39. In OPEN 6.1 and above, prompt definitions are stored in an XML format.

AutoPrograms

One mechanism by which the LI-6400 can operate automatically is the Auto-Program.

What are AutoPrograms?

AutoPrograms are small LPL application programs designed to run on top[2] of OPEN. While there is practically no limit to the scope of what an AutoProgram can be made to do, typically these programs log data in some sort of automatic fashion, while perhaps maintaining control over one or more conditions in the leaf chamber.

A number of AutoPrograms (Table 9-3) are installed with OPEN, and are described below. You can modify these, or write your own. There is a utility (**Making Your Own AutoPrograms** on page 9-47) for generating AutoPrograms, and Chapter 25 describes many useful commands available for Auto-Programs.

Table 9-3. Standard AutoPrograms

Name	Description
A-CiCurve2	Generates a CO_2 response curve.
AutoLog2	Logs data at regular intervals.
LightCurve2	Generates a light response curve.
TimedLamp2	User selects LED source values, logging frequencies, and time intervals. Useful for sunfleck simulations.
CO2Curve_MultipleLight	Does CO_2 response curves at several light values.
LightCurve_MultipleCO2	Does light curves at several CO_2 values.

---

[2] That is, they cannot be launched unless OPEN is running.

Launching AutoPrograms

All AutoPrograms are launched in the same way.

1   **Press AutoProg**

AutoPrograms are launched by pressing **AutoProg** (**f1** level 5 in New Measurements mode of OPEN).

2   **Select destination**

If logging is not active, you'll be asked to open a log file, just like when you press **Open_LogFile** (**f1** level 1).

If logging is active, you'll be asked

```
Append to the current log file? (Y/N)
```

Press **Y** if you wish to add observations to the current log destination, or **N** if you wish to close it and open another. If you press **N**, you'll be asked to pick a log destination.

3   **Select the AutoProgram**

The user is then prompted to pick an AutoProgram. This list of programs shown (uses Standard Menu) consists of all files in the directory /User/Configs/AutoProgs.

4   **Answer the questions**

Most AutoPrograms will give you a chance to set parameters before they run. Prior to version 6.2, this was done by sequential prompts. Starting with 6.2, the standard AutoPrograms changed to an editable tree interface, allowing you to scroll and view everything, and set what you want in any order you choose. The interfaces for the standard AutoPrograms are described below in **AutoProgram Details** on page 9-34.

While an AutoProgram is Active

Once an AutoProgram is active, the display and function keys will appear much like they do in New Measurements mode. There are some subtle differences, however.

- **The AutoProgram Asterisk**
  While an AutoProgram is active, an asterisk is on the display (Figure 9-40).

Figure 9-40. An asterisk appears to the left of the function keys while an Auto-Program is active.

- **You can't leave New Measurements mode**
  If you attempt to exit New Measurements mode, you will be presented with the AutoProgram Exit Screen (Figure 9-42 on page 9-34).

- **You can monitor an AutoProgram's progress**
  There are two fields in the standard display map that are useful for monitoring an AutoProgram. Both are available on the *k* display line (Figure 9-41).

If the program is paused, the Program value will alternate between "PAUSED" and the time until the next step.

Figure 9-41. "Program" indicates the time remaining to the next step, while "ProgPrgs" indicates the program is presently working on the 8th of 18 steps.

- **You can still log and control manually**
  While an AutoProgram is running, even one that is operating a control (example: LightCurve controls the LED source), you can still log data by pressing **Log** (**F1** level 1), and change any control setting (**F1** through **F5** level 2), enter match mode (**F5** level 1), etc.

Controlling an AutoProgram

Once it is launched, there's not much one can do to an AutoProgram's course of action, other than pause/resume, terminate it early, or trigger the next step

before it would otherwise occur. To do one of these, press **escape** to access the AutoProgram Exit screen (Figure 9-42).

A running program

```
█CO2R_                         H2OS_mml
a      1  AUTOLOG is Running...   13.114
█CO2_     A - abort program      RH_S_%
b     72  P - pause program       41.52
     Ph   T - trigger next step  Trmmol
c     -   <esc> - no change_       -7.83
* AUTO                          e Define
5 PROG                          y Log Btn
```

A paused program

```
█CO2                           2OS_mml
a         AUTOLOG is PAUSED...   13.198
█CO       A - abort program     RH_S_%
b         R - resume program     41.78
          T - trigger (and resume)  Trmmol
c         <esc> - no change_      -8.56
* AU                            Define
5 PR                            Log Btn
```
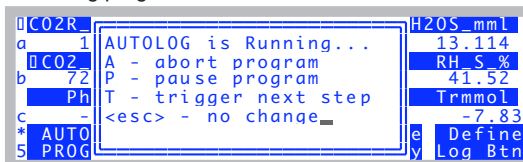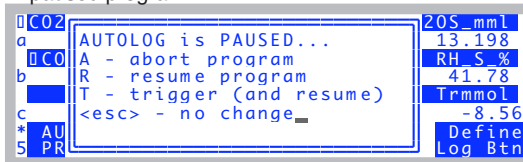
*Figure 9-42. The AutoProgram Exit screen. Pressing **A** will terminate the AutoProgram, **P** will pause (or **R** resume) , **T** will trigger the next step in the AutoProgram, and **escape** makes no change in the AutoProgram state.*

AutoProgram Details

Old and New

Version 6.2 includes a number of changes to the standard AutoPrograms described below. These programs (e.g. A-CiCurve2, LightCurve2, etc.) use a setup editor based on a tree structure, while their predecessors (e.g. A-CiCurve, LightCurve) simply prompted for information in an irreversible sequence.

The new AutoPrograms also adapt to the light source being used, and in particular, if you are configured for fluorometry, will add some options specific for fluorometry. Thus, several fluorometer-specific AutoPrograms have gone away in 6.2, since that job is now covered by the new standard AutoPrograms.

Version 6.2 will still support old AutoPrograms. In fact, if you upgraded an earlier instrument to version 6.2, the entire suite of AutoPrograms that you used to have is still there, but reside in a different directory (/User/Configs/AutoProgs_old). If you wish to run one of them, just navigate to that directory when being prompted to pick an AutoProgram to run, and select it.

Common Setup Features

The setup screen for each of the standard AutoPrograms consists of a tree with expandable, editable nodes. For example, the one for A-CiCurve2 is shown in Figure 9-43.

Present only when configured for fluorometry.

Settings specific to this program.

Common to all programs.

Save all settings, and start the program.

```
═ (1/4) A-CiCurve2 Setup═
→▶Flr Actions= DarkAdapt +
 ▶Summary: 8 (400→600) of Sample CO2
 ▶Stability definition: = 4 items
 ▶Log Opts: beep ctrl

 +   ▼/▶   ◆          ◆          ◆cancel  ◆ START
+Open...◆saveAs.◆          ◆          ◆Default
```

Read/Write program settings to/from a named file.

Reset all program settings to factory defaults.

Figure 9-43. The setup screen for the AutoProgram A-CiCurve2.

Flr Actions:

This node is there only if you are configured for an 6400-40 LCF Fluorometer. The node handles things like dark adaptation.

```
→▼Flr Actions= DarkAdapt +
 ┌▼Dark adapt before starting= yes
 │ ┤ Dark time (min) before FoFm= 30
 │ ┤ Light time (min) after FoFm= 30
 └▼Measure dark photo at Fo= no
    └ Dark photo rate= -1
 ┤ Save each flash= no
 ┤ Save each dark pulse= no
 └ Flr Recording= "Off always"
```

If you are doing a dark adaptation, the LCF actinic and measuring beam will be shut off the prescribed time, followed by 1) (optional) match and set dark respiration rate to the current value of photosynthesis, 2) measuring beam on, 3) DoFoFm, 4) actinic on, and 5) (optional) wait for the prescribed light adaptation time.

Stability Definition:

The stability node contains a stability definition that will be implemented when (if) you start the AutoProgram. Note that what is shown is not necessarily the current stability definition, but if you do launch the AutoProgram, it *will* become the definition, and remain so once the AutoProgram ends. See **Defining Stability** on page 6-29.

```
→▼Stability definition: = 3 items
 └▼items=  "Std Stability"
   ├ items[1]=  CO2S  (-2)  15  Slp<1
   ├ items[2]=  H2OS  (-5)  15  Slp<1
   └ items[3]=  Flow  (-7)  15  Slp<1
```

Log Opts:

The log options node is a subset of the complete list of log options (**Log Options** on page 9-14). Only the ones that are changeable once a log file is open are included here.

```
→▼Log Opts: beep mean ctrl
 ├ Beep:  on
 ├▼Statistics: active (15 s)
 │ ├ Means->log file:  yes
 │ ├ Stats->.STATS file:  no
 │ └ period:  15
 ├ Control changes:  yes
 └ Echo to Comm:  no
```

Stability Wait=

Most AutoPrograms will contain a "Stability wait" option in the Summary node.

```
→  ├▼Stability wait= 60 to 300 s
   ├ Minimum (secs)=  60
   └ Maximum (secs)=  300
```

Minimum is the time between whatever change just happened (new CO_2 mixer target, for example), and when the program will *begin* to check stability to see if an observation can be logged. No logging will occur during this period. When the AutoProgram is running and in this minimum wait period, the *Pro-*

gram variable on display line 'k' will show the wait time with semi-colons instead of colons:

Following the minimum and but before the <u>Maximum</u> is when logging could occur, if stability is reached. Otherwise, logging will occur at the end of the maximum time. During stability checking period, the *Program* variable will show asterisks instead of colons:

Match Before Log=

Most AutoPrograms will contain a "match before log" option in the Summary node. The match node can be in one of three states: "never", "always", or "if one of...".

The "if one of..." option will match if at least one of three conditions is true. 1) The time since the previous match exceeds some threshold time, 2) The reference CO_2 concentration has changed (up or down) since the previous match more than the specified threshold value, or 3) the absolute difference between CO2R and CO2S is less than the specified threshold.

If matching does occur, the will be a post-match recovery time for which you specify the minimum and maximum. After the minimum time, the system starts checking for stability, and stops waiting once it is achieved (or the maximum time expires).

Saving AutoProgram Settings

AutoPrograms with the new (version 6.2) interface provide a mechanism not only for remembering what the settings were the last time the program was run, but also for storing and retrieving settings from named files, and resetting to factory defaults.

```
══  (1/4) A-CiCurve2 Setup════
→▶Flr Actions= DarkAdapt +
  ▶Summary: 8 (400→600) of Sample CO2
  ▶Stability definition: = 4 items
  ▶Log Opts: beep ctrl


+  ▼/▶  ◆          ◆          ◆cancel ◆ START
+Open...◆saveAs.◆        ◆          ◆Default
```
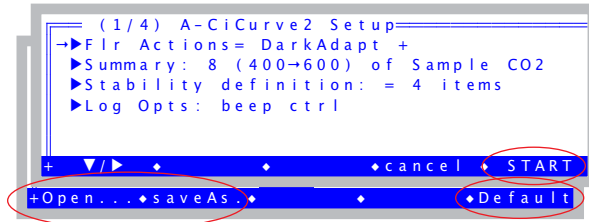
START - launches the program, but also automatically saves the parameters to the storage directory in a file named "previous".

Default - This key will be present if there is a file named ".default" in the storage directory. Pressing the key will load the settings from this file (these are factory defaults).

saveAs.. - Save the current parameter settings to the storage directory. You can name the file.

Open.. - Load parameters from a file in the storage directory. You can pick "previous", or ".default", or any file you have previously named and stored.

"A-CiCurve2"

This program controls the 6400-01 CO_2 mixer, and logs data when stability is reached. The Summary node of the setup dialog is shown in Figure 9-44.

```
══ (1/4) A-CiCurve2 Setup══════
→▶Flr Actions= DarkAdapt +
  ▶Summary: 8 (400→600) of Sample CO2
  ▶Stability definition: = 4 items
  ▶Log Opts: beep ctrl


→▼Summary: 8 (400→600) of Reference CO2
  ├ CO2 control= Reference CO2
  └▼CO2 Setpoints= { 400 300 200 100 50
  ├▼Stability wait= 60 to 120 s
  │ ├ Minimum (secs)= 60
  │ └ Maximum (secs)= 120
  ├▼Match before log= "If one of..."
  │ ├ ...Elapsed time (min) > 30
  │ ├ ...|CO2 change| (ppm) > 100
  │ ├ ...|ΔCO2| (ppm)< 10
  │ ├ Post-match recovery min (s)= 10
  │ └ Post-match recovery max (s)= 300
  └ Log= Log
```

page 9-35

page 9-36

page 9-36

page 9-36

page 9-37

Figure 9-44. The setup dialog for the A-CiCurve2 program.

CO2 Control=

This specifies the type of CO_2 mixer control to be used.

```
═════════CO2 Control═════════
 Reference CO2
 ◆ Sample CO2
 Control Signal
 Leaf Intercellular CO2
```

Current mode.
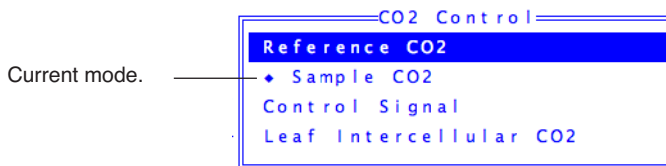
For a CO_2 response curve, the best choice is probably "Reference CO2", since it will be quite fast to get the mixer stabilized at each new value. If you use "Sample CO2", there will be considerably more delay while the mixer adjusts to each new photosynthetic rate. Since you want a range of CO2 concen-

trations, and don't really care that they be exact values, this extra time is not worth it. Just use "Reference CO2".

If you want to go the fastest, and eliminate any "hunting" time for the mixer to lock in on CO_2 reference values, you can use "Control Signal" mode. When you do this, an extra node will appear giving you the choice of specifying the control set points in mV or in ppm.

```
▼Summary: 8 (400→600) of Control Signa
→ ▼CO2 control= ◆ Control Signal
    └ Setpoint units= ppm
  └▼CO2 Setpoints= { 400 300 200 100 50
  ▼Stability wait= 60 to 120 s
```
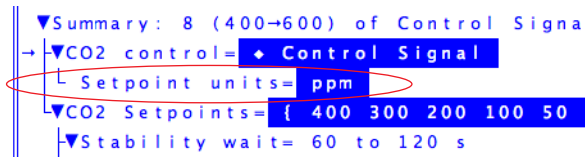
If you specify ppm, each target will be converted to mV before actually setting the mixer. This conversion uses the mixer calibration curve, so is only as good as your most recent mixer calibration.

The worst possible choice for doing a CO_2 response curve is "Leaf Intercellular CO2", and you'd never use it for a typical A-Ci curve.

CO2=
Enter the CO_2 values you wish to achieve. Type the values, separated by spaces, such as

```
┌═CO2 Setpoints═
│ 400 300 200 100 50 400 400 600 _
```

When editing this node, the edit box will show an arrow on the left if there are more values scrolled off to the left.

```
┌═CO2 Setpoints════════════════════n
←│ 400 600 _
```

There will also be an arrow on the right end if there are values scrolled off that way, as well. **home** and **end** can jump to the beginning and ending of the line.

"AutoLog2"

AutoLog2 is designed to log instantaneous data at regular intervals.

```
═════ (1/4) AutoLog2 Setup═══════════════
→▶Flr Actions
 ▶Summary: Every 30-120s for 00:10:00
 ▶Stability definition: = 4 items
 ▶Log Opts: beep ctrl
```

page 9-35 ⟶
page 9-36 * ⟶
page 9-36 ⟶
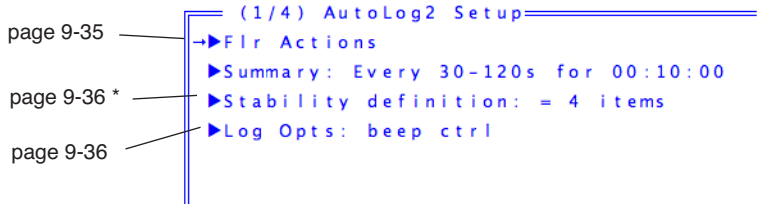
Figure 9-45. The setup dialog for the AutoLog2 program. *The Stability definition node only appears when "Wait for= stability"

The Summary node has two looks, depending on if you are waiting for time or for stability.

```
→▼Summary: Every 120s for 00:10:00
  └▼Duration (mins)= 10
   └▼Wait for= time
    ├ Log interval (s)= 120
    └ Action= Log
```

```
→▼Summary: Every 30-120s for 00:10:00
  └▼Duration (mins)= 10
   └▼Wait for= stability
    ▶Stability wait= 30 to 120 s
    ├ Match before log= "never"
    └ Action= Log
```

page 9-36
page 9-37
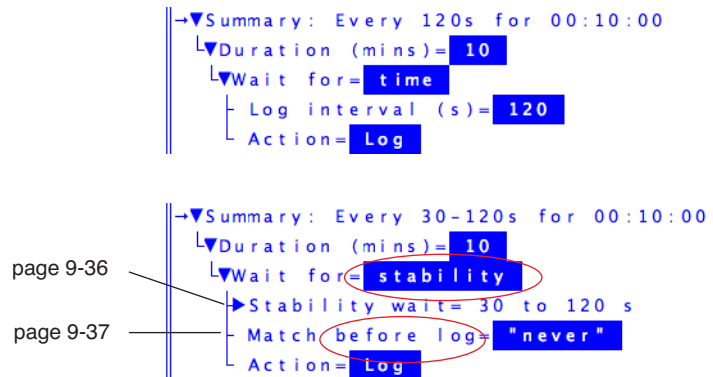
Duration (mins)=
Duration (in minutes) specifies how long AutoLog2 is to run.

Wait for=
This determines the time between logging events. You can wait for time (in seconds), which is controlled by the sub-node "Log interval (s)=". Or you can elect to do stability checking, in which case other nodes will appear that let you specify stability, and a matching option.

"CO2Curve_MultiLight"

This program has two control loops, one inside of another. The outer loop changes light, and the inner one changes CO_2 concentration.
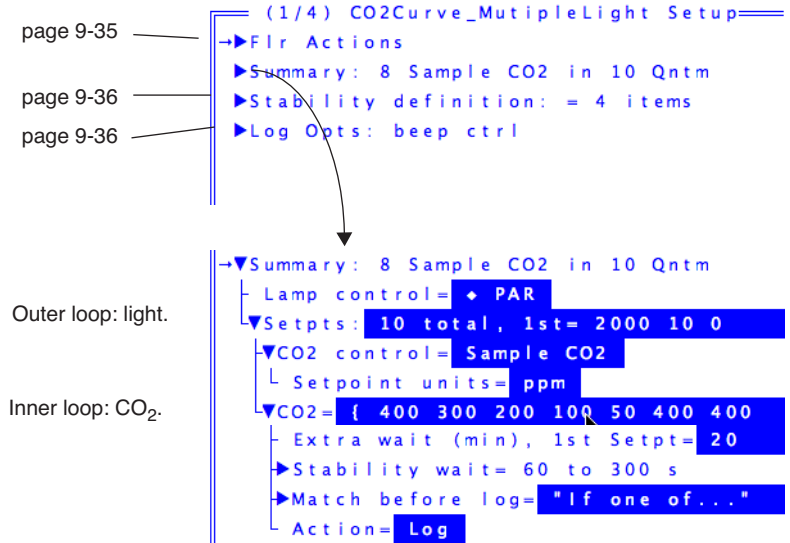
Figure 9-46. The setup screen for the CO2Curve_MultiLight program.

Lamp Control=

See **Lamp Control=** on page 9-43.

Setpts:

See **Setpts:** on page 9-44.

Extra wait (min), 1st Setpt=

This is the time (minutes) that the program will wait after setting each light level. Note that this wait comes right *after* setting the CO_2 to the first value in the repeating list, so that the extra equilibration time will have the plant at both the new light level, and the first CO_2 level.

CO2 Control=

See **CO2 Control=** on page 9-39.

CO2=

See **CO2=** on page 9-40.

"LightCurve2"

Controls a light source, and logs data when stability is reached.

page 9-35

page 9-36

page 9-36

```
       ═══ (1/4) LightCurve2 Setup═══════
 →►Flr Actions
  ►Summary: 5 SetPts for Qntm
  ►Stability definition: = 3 items
  ►Log Opts: beep ctrl




 →▼Summary: 10 SetPts for Qntm
  ├ Lamp control= PAR
  └▼SetPts: 10 total, 1st= 2000
    ►Stability wait= 120 to 300 s
    ►Match before log= "If one of..."
  ├ Log= Log
```
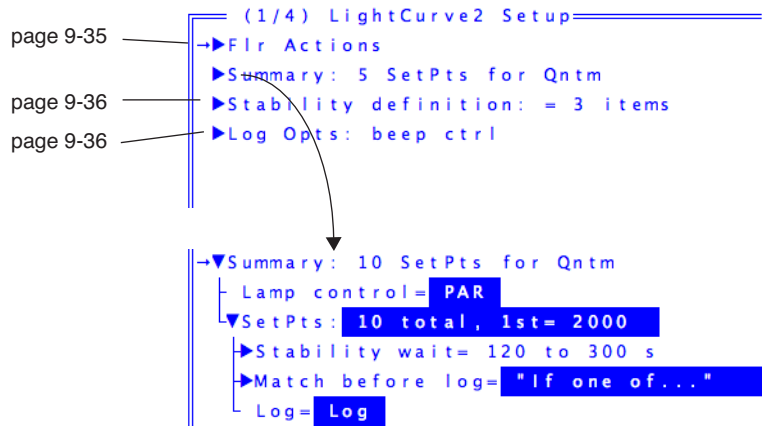
Figure 9-47. Setup screen for the program LightCurve2.

Lamp Control=

This specifies how the light source is to be controlled.


```
       ═══Light Control═══════
 ▌PAR
  Control Signal
```

Normally one would want to use PAR, but Control Signal is in there for completeness sake.

Setpts:

When you edit the Setpoints, you get a dialog box appropriate for the light source being used and the mode you are in (Figure 9-48).

Lamp control = PAR   Lamp control = Control signal
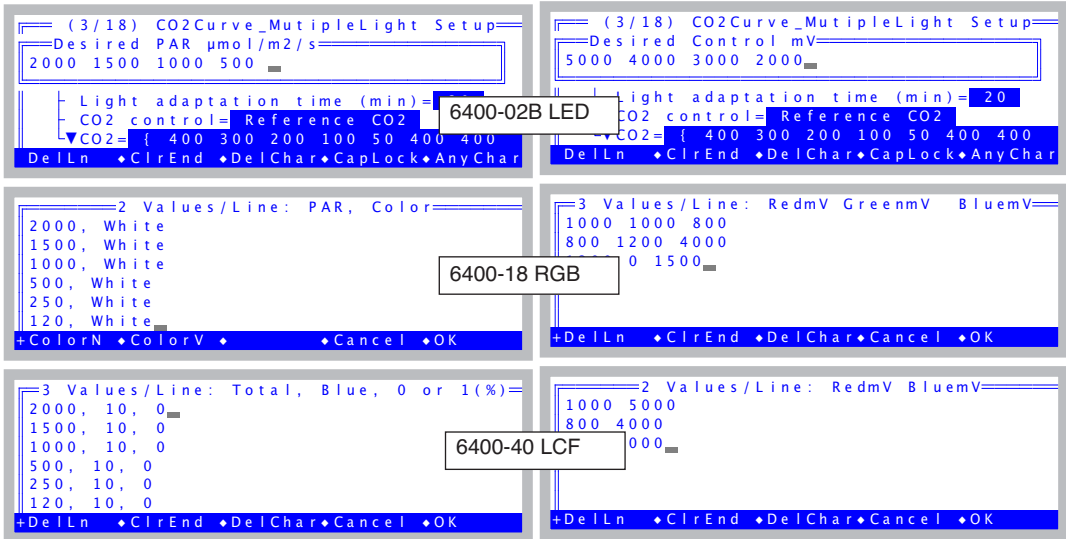
Figure 9-48. Prompting for PAR setpoints depends on what light source you're using.

"LightCurve_MultipleCO2"

This program does light curves at one or more CO_2 settings (Figure 9-49).

page 9-35

page 9-36

page 9-36

Outer loop: CO_2.

Inner loop: light.

```
═══ (1/4) LightCurve_MultipleCO2 Setup═══
→▶Flr Actions
 ▶Summary: 7 Qntm in 4 Sample CO2
 ▶Stability definition: = 4 items
 ▶Log Opts: beep ctrl
```

```
→▼Summary: 7 Qntm in 4 Sample CO2
 ├ CO2 control= Sample CO2
 ▼CO2= { 385 280 550 700    }
 ├ Lamp control= ◆ PAR
 ▼Setpts: 7 total, 1st= 2000 10 1
 ├ Extra wait (min), 1st Setpt= 20
```
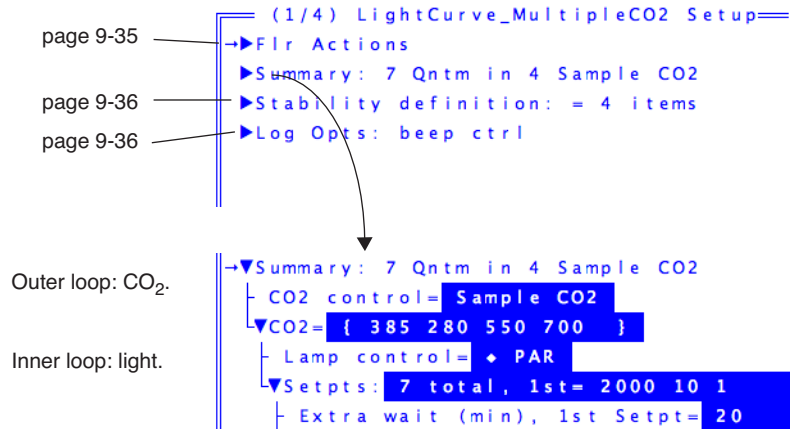
Figure 9-49. Setup screen for the program LightCurve_MultipleCO2.

CO2 Control=
See **CO2 Control=** on page 9-39.

CO2=
See **CO2=** on page 9-40.

Lamp Control=
See **Lamp Control=** on page 9-43.

Setpts:
See **Setpts:** on page 9-44.

Extra wait (min), 1st Setpt=
The extra time (in minutes) that is to be waited for the first light level (right after a change in CO_2).

"TimedLamp2"

TimedLamp allows you to program timed changes in a light source (Figure 9-50). For example, you may want to set the light level to 200 μmol m^{-2} s^{-1} for several minutes, then jump to 2000 μmol m^{-2} s^{-1} for 20 seconds, then drop back to the original value for several more minutes.

You can also specify the logging frequency for each lamp level. You may, for example, want to record data less frequently during an initial stability period, and very frequently after a light change.
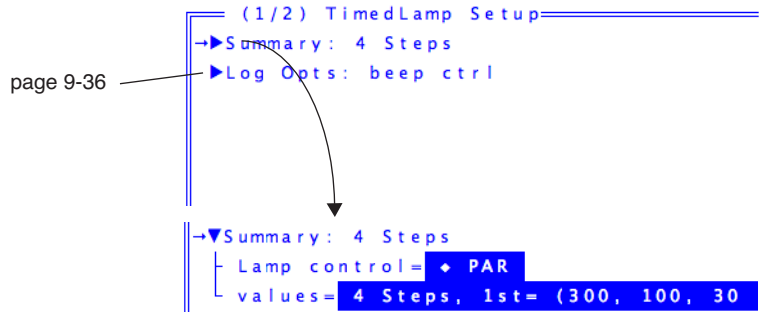
Figure 9-50. Setup screen for the program TimedLamp2.

Lamp control=
The lamp control is a choice between PAR and Control Signal. Usually, you would want PAR unless you want the sharpest possible step change in light with no potential for subsequent adjustments to get an exact value.

values=
The editor for the values entry is looking for three values per setpoint: time duration, intensity, and a log interval (Figure 9-51).

Figure 9-51. Each step in TimedLamp has three values: duration, intensity, and log interval.

The intensity value has to be interpreted based on the light source being used, and the lamp control method (Table 9-4).

Table 9-4. How to interpret Intensity when entering programmed setpoints for TimedLamp2.

Light Source	PAR	Control Signal
6400-02B LED	Total μmol m^{-2} s^{-1}.	mV
6400-18 RGB	Total μmol m^{-2} s^{-1}. Color remains unchanged.	% of total red. (Green and Blue % of total remain unchanged.)
6400-40 LCF	Total μmol m^{-2} s^{-1}. Blue settings remain unchanged.	red mV. (Blue mV remains unchanged)

Making Your Own AutoPrograms

If none of the standard AutoPrograms does what you need to do, you can make your own. There is a tool that makes this task very easy: `"Create a new AutoProgram"`, found in OPEN's Utility Menu. This program lets you build an AutoProgram by picking items from a menu, and requires no programming whatsoever. You always have the option of editing an AutoProgram after it's built. AutoPrograms are stored in /User/Config/AutoProgs.

Note: AutoProgram Builder is a "legacy" utility, and AutoPrograms generated with it will use the old sequential prompt method of entering inputs, not the editable tree method introduced in version 6.2.

What the AutoProgram Builder Does

The AutoProgram builder constructs an AutoProgram from your responses. You pick, in order, the events that are to occur when the AutoProgram runs. Some events are simple, while other events require extra information from you. For example, suppose you build an AutoProgram that does only two things: 1) wait a fixed amount of time, and 2) log data. When you pick the "wait a fixed amount of time" option from the menu, you will be asked a) whether you want to specify time in minutes or seconds, b) what the prompt should be that is used to prompt the user to enter this time, and c) what the default value of the wait time will be.

Thus, building an AutoProgram with the AutoProgram Builder consists of picking events in order, and answering the questions (if any) associated with each event. When you are done, the AutoProgram builder will create the file

and store it for you. To run the new AutoProgram, select it just as you would any other AutoProgram, by pressing **f1** (level 5) in New Measurements mode.

Plan Ahead!
Have a clear idea of what you want the AutoProgram to do before using the Builder. If you make a mistake, such as leaving out a step, you will either have to start over, or else edit the AutoProgram file once the Builder is done creating it.

To illustrate the use of the AutoProgram Builder, we present two examples.

Example: Humidity Response Curve
This AutoProgram will expose the leaf to a range of humidities, and allow time to equilibrate before logging. Since flow rate is the system's basic humidity control, we will simply control flow rate and let the resulting humidity be whatever it will be. This guarantees that the AutoProgram will generate the widest range of humidity, regardless of what ambient conditions are, or what the leaf is doing. Thus the program would have the following structure:

```
User enters flow rates
User enters wait time
BEGIN LOOP over flow rates
  Set flow rate
  Wait some time
  Log
END LOOP
```

Before using the AutoProgram builder, it is important to have this sort of outline in front of you.

■   **To Build This AutoProgram**

1   Launch the AutoProgram Builder
Select the entry entitled "`Create a New AutoProgram`" in the Utility Menu. Press **enter** when prompted with "`Press <enter> to start`".

```
=================Main Event #1=================
A) End Program
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
G) Set a control...
```
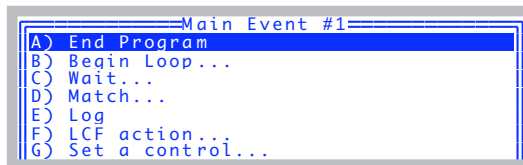
Since the first thing on our outline involves prompting the user for information, you might think the first thing we want to do would involve choice G

"Prompt for user constants", but that's not how the AutoProgram Builder works. When we get done with entering program steps, we'll get to choose how the user inputs wait times and flow rates and any other constant that pertains to this program (Step 11 on page 9-52). So for now, let's just ignore the user entry steps, and get straight to the loop.

2   Select "`Begin Loop`" **from the menu**
Highlight "`B) Begin Loop`" and press **enter**, or else simply press **B**.

```
════════════════Main Event #1════════════
A) End Program
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
G) Set a control...
```
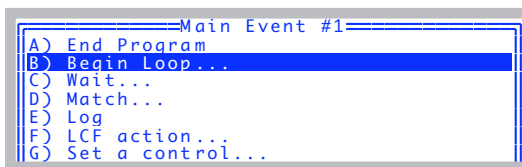
3   Make it a Control Loop
Press **1** to select a control loop.

```
┌═══════════════Loop Options═══════════┐
│Loop N times, where                   │
│1) N = # of settings for a control    │
│2) N = user entered                   │
│                                      │
└══════════════════════════(1/2)═══════┘
F) LCF action...
G) Set a control...
```
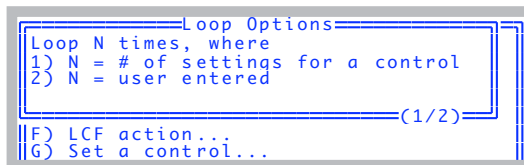
4   Specify the type of Control Loop: Fixed Flow Rate
When the control loop options are shown, select "`A) Flow/Humidity`" and press **enter**, or simply press **A**.

```
═══════════════Main Event #1═══════════
A) ┌═══Loop sets what control?═══┐
B) │A) Flow/Humidity             │
C) │B) CO2                       │
D) │C) Temp                      │
E) │D) LED Source                │
F) │E) LCF                       │
G) │F) Fan                       │
```
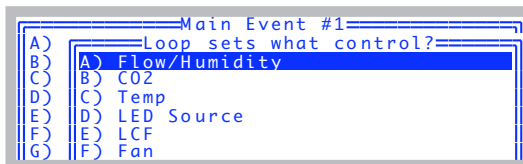
Follow that with what type of flow control: Flow (press **A**) again.

```
═══════════════Main Event #1═══════════
A) ┌═══Pick Flow/Humidity Option═══┐
B) │A) Flow                        │
C) │B) H2OS_mml                    │
D) │C) RH_S_%                      │
E) │D) VPD                         │
F) │                               │
G) │                               │
```
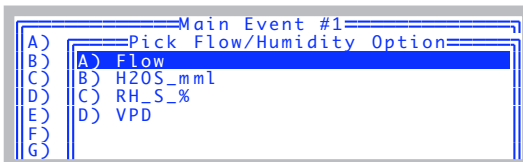
5   **Edit the prompt string for the flow control loop**
You will be given the chance to edit the prompting string for the flow rates.

```
═══════════════Main Event #1═══════════════
A) End Program
B) ┌══════Enter prompt string═══════════════
C) │Flow rates (µmol/s):_
D) └═════(DelLn to never prompt)═══════════
E) Log
F) LCF action...
DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Just press **enter** if the prompt is OK. (You would press **DelLn** (**f1**) to clear this
or any prompt if you want a fixed value that is never prompted.)

6   **Edit the default values for the flow control loop**
You will be given the chance to set the default values for the flow rates.

```
═══════════════Main Event #1═══════════════
A) End Program
B) ┌══════Enter default value═══════════════
C) │300 500 700_
D) └
E) Log
F) LCF action...
DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Adjust them as you wish, and press **enter** when done.

7   **Specify the first action within the flow control loop: Waiting**
The first action is the wait, so highlight "`C) Wait: Fixed Time`" and press
enter, or press **C**.

```
═══════════════Main Event #1═══════════════
════════Loop (Flow) Event #1═══════════
A) End Loop
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
```

You will be asked to specify what sort of wait you want. Press **A** to select the
fixed time wait.

```
═══════════════Main Event #1═══════════════
══════════Pick Wait Option═════════
A)│A) Wait: Fixed Time
B)│B) Wait: Min,Max,Stability
C)│C) Wait, with Logging at intervals
D)│
E)│
F)│
```

You will be asked how you wish to specify this wait time. Press **M** to select minutes.

```
╔══════════════Main Event #1══════════════╗
║ ┌──────────────────────────────────┐  1  ║
║ │A) En│Enter time in               │     ║
║ │B) Be│Minutes or Seconds?_        │     ║
║ │C) Wa└──────────────────(M/S)═════╝      ║
║ │D) Ma─────────────────────────────┘      ║
║ │E) Log                                   ║
║ │F) LCF action...                         ║
╚═════════════════════════════════════════╝
```
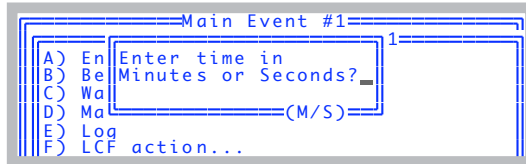
You'll be asked to edit the time delay prompt. Notice that the default units reflect your earlier choice of minutes or seconds. Press **enter**.

```
╔═══════════════Main Event #1══════════════╗
║ ┌─────────Loop (Flow) Event #1═════════┐ ║
║ │A)┌───Enter prompt string────────────┐│ ║
║ │B)│Wait time (minutes):_             ││ ║
║ │C)└────(DelLn to never prompt)───────┘│ ║
║ │D) Match...                           │ ║
║ │E) Log                                │ ║
║ DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar║
╚══════════════════════════════════════════╝
```
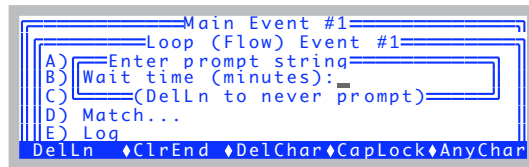
You'll be asked to specify the default wait time value. Press **enter** when done.

```
╔═══════════════Main Event #1══════════════╗
║ ┌─────────Loop (Flow) Event #1═════════┐ ║
║ │A)┌───Enter default value─┐           │ ║
║ │B)│1_                      │           │ ║
║ │C)└───────────────────────┘           │ ║
║ │D) Match...                           │ ║
║ │E) Log                                │ ║
║ DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar║
╚══════════════════════════════════════════╝
```
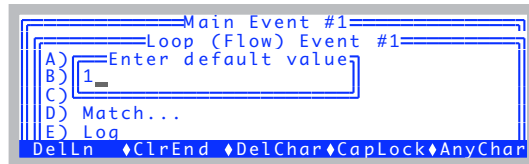
8 **Specify the second action in the flow control loop: Logging.**
The second action is logging, so highlight "E) Log" and press **enter**, or simply press **E**.

```
╔═══════════════Main Event #1══════════════╗
║ ┌─────────Loop (Flow) Event #2═════════┐ ║
║ │A) End Loop                           │ ║
║ │B) Begin Loop...                      │ ║
║ │C) Wait...                            │ ║
║ │D) Match...                           │ ║
║ │E) Log                                │ ║
║ │F) LCF action...                      │ ║
╚══════════════════════════════════════════╝
```
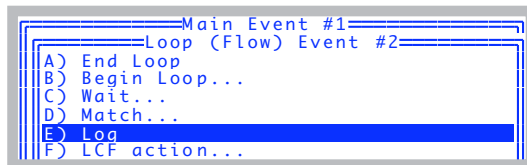
9   **End the control loop**

Highlight "`A) End Loop`" and press **enter**, or simply press **A**.

```
========================Main Event #1=========================
========================Loop (Flow) Event #3==================
A) End Loop
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
```

10   **End the program**

We are now out of the flow control loop, and since there is nothing more for the program to do, we'll quit by highlighting "`A) End Program`" and pressing **enter**, or else pressing **A**.

```
========================Main Event #2=========================
A) End Program
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
G) Set a control...
```

11   **Specify your prompting preference**

There are three options: 0)You can have the program never prompt you, but always use the values you have specified as the defaults. This is nice for rapidly starting an AutoProgram. 1) You can have it prompt you, always using the default values. 2) You can be prompted each time, with the values entered the last time used as the default. Press **0**, **1**, or **2**.

```
Enter prompting preference:
  0) Never prompt
  1) Prompt with fixed defaults
  2) Prompt with last time defaults

Enter (0/1/2)_
```

12   **Store the AutoProgram**

You are given the opportunity to store the AutoProgram.

```
Enter prompting preference:
  0) N=========================================
  1) P| Store this AutoProgram_         |
  2) P|                          |ults
    |====================(Y/N)====
Enter (0/1/2)
```

If you press **Y**, the Standard File Dialog is accessed for storing the file. The default name of the file will be the sequence of selections you made, but you can change it to something meaningful. In this case, the default name will be

BaaCaEAA

and the default directory will be /User/Configs/AutoPrograms. Renaming it to Humidity Curve would be a good choice. The program is then exited. If you wish to view or edit the AutoProgram that you have created, you can do so by accessing the Filer, selecting the file, and pressing **E**.

If you press **N**, to not store the file, you will be given a chance to edit the AutoProgram you have created.

```
Enter prompting preference:
  0╔═══════════════════════════╗
  1║ View/Edit this AutoProgram?_║
  2╚════════════════════(Y/N)══╝ts
Enter (0/1/2)
```

When you exit the editor (Standard Editor), you can save the file.

The AutoProgram Builder Reference

The major options of the AutoProgram Builder are shown in Figure 9-52.

Action List

```
A) End Program (or End Loop)
B) Begin Loop...
C) Wait...
D) Match...
E) Log
F) LCF action...
G) Set a control...
H) Prompt for user constants
I) Add a remarks record
```

F) LCF Action List

```
A) Measure On
B) Measure Off
C) Far Red On
D) Far Red Off
E) Actinic On
F) Actinic Off
G) Flr Recording On
H) Flr Recording Off
I) Do Fo
J) Do Fm
K) Do FoFm
L) Do Fo'
M) Set Fs
N) Do FsFm'
O) Do FsFm'Fo'
P) Set Zero
Q) Flash (No assign)
R) Dark (No assign)
```

D) Match Options List

```
A) Match always
B) Match if |ΔCO2| < x
C) Match if |ΔH2O| < y
D) Match if |ΔCO2|<x OR |ΔH2O|<y
```

C) Wait Options List

```
A) Wait: Fixed Time
B) Wait: Min,Max,Stability
C) Wait, with Logging at intervals
```

B) Loop

```
Loop N times, where
1) N = # of settings for a control
2) N = user entered
```

Controls: See Figure 9-54 on page 9-58

Figure 9-52. AutoProgram Builder's major options. The Type 1 Loop uses multiple control values, while the "Set a control…" event uses one control value. The Type 2 Loop sets no control values - it is a fixed count loop.

A) End Program

This item is labelled "`A) End Loop`" when selecting items within a control loop. When selected from within a control loop, it ends the loop. When selected outside of a control loop, it marks the end of the AutoProgram.

B) Begin Loop…

There are two types of loops: Type 1 sets a control, and the number of loops is determined by the number of control settings. The possible controls are shown in "Control List" in Figure 9-54 on page 9-58, and they essentially correspond to the controls found in the level 2 function keys of New Measurements mode. (Note: if you specify a Type 1 loop, but then **escape** out of selecting the control to be set, the loop becomes Type 2.) A Type 2 loop does not set any control automatically, but simply loops a user-specified number of times.

Once the prompting sequence and default values are established, you select items for "inside" the control loop.

The loop sets fixed flow control.

This loop is the 3rd event in the Auto-Program

We're picking the 1st item in the flow control loop. (This will be the first thing to happen *after* the flow rate is set to the next value.)

```
═════════════ Main Event #3 ═════════════
══════ LOOP (Flow: Flow #1) Event #1 ══════
A) End Loop
B) Control Loop...
C) Wait...
D) Match...
E) Log
F) Set a control...
```
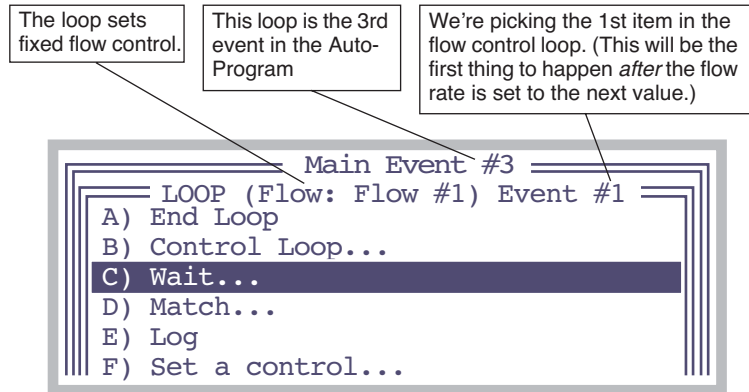
Figure 9-53. Picking items within a loop. A Type 1 loop sets the control to a new value automatically; you have to select everything else that you wish to have happen after that, however.

Note that you can have nested loops, since "`B) Begin Loop…`" is one of the in-loop possibilities. To terminate a loop, select "`A) End Loop`".

C) Wait…

There are three wait options:

```
A) Wait: Fixed Time
B) Wait: Min,Max,Stability
C) Wait, with Logging at intervals
```

During these waits, the instrument will appear to be in New Measurements mode, allowing you to monitor channels, view real time graphics, etc.

A) Wait: Fixed Time: The A option will generate a prompt for the wait time (it can be in minutes or seconds), and a prompt for the default value to be used.

B) Wait: Min, Max, Stability: The B option generates a number of prompts: The min and max wait times can be specified in minutes or seconds. There is a prompt for each, and a default value for each.

C) Wait, with Logging at intervals: The C option is similar to the fixed time wait, except that logging will occur automatically at regular intervals during this wait. The total wait duration is always specified in minutes, and the logging interval is specified in seconds. Prompts and default values for both are established.

The C option represents a short-cut to doing a Type 2 loop that contains a fixed time wait and a log. If you wish to do other events, such as matching before each log, then you should specify a Type 2 loop, rather than using this wait option.

D) Match…
There are some options for matching:

```
A) Match always
B) Match if |ΔCO2| < x
C) Match if |ΔH2O| < y
D) Match if |ΔCO2|<x OR |ΔH2O|<y
```

Matching can happen always or conditionally, based on the ΔCO_2 and/or the ΔH_2O value. Options B, C, and D will cause prompts and default values to be established.

E) Log
This selection will cause data to be logged. It requires no extra information.

F) Set a control…
This selection is for setting a control once, as opposed to looping over several values for the control. You select the control from the list shown in Figure 9-54 on page 9-58. For example, if you were building a light curve Au-

toProgram, and wanted to set the CO_2 controller for controlling on sample concentration, and the temperature controller for controlling on leaf temperature prior to looping over light values, you would use the "Set a control..." option for the CO_2 and again for temperature, but then use the "Control Loop" option for setting the light values.

G) Prompt for user constants
This will do the same thing as pressing **F5** level 3 (**Prompt All**) in New Measurements mode: any user constants that have been defined will be prompted for. (Note: This suspends operation until the user responds.)

H) Add a remarks record
This option does the same thing as pressing **F4** level 1 (**Log Remark**) in New Measurements mode: it adds a remark record to the data file. (Note: This suspends operation until the user responds.)

```
A) CO2R_µml
B) CO2S_µml
C) Control Voltage
```

```
A) Flow
B) H2OS_mml
C) RH_S_%
D) VPD
```

Control List

```
A) Flow/Humidity
B) CO2
C) Temp
D) LED Source
E) LCF
F) Fan
G) RGB
```

```
A) Tblock
B) Tleaf
```

```
A) Quantum Flux
B) Control Voltage
```

```
A) Fan Speed
```

```
*A) Quantum Flux
*B) Set Color
*C) Control %Max (3 color)
```

```
*A) Quantum flux, %Blue
*B) Quantum flux, fixed Blue
*C) Control Red_mV, Blue_mV
D) Meas Intensity
E) Meas Modulation
F) Meas Filtering
G) Meas Gain
H) Flash Type
I) Flash Intensity
J) Flash Duration
J) Phase1 Duration
K) Phase2 Duration
L) Phase3 Duration
N) Flash Modulation
O) Flash Filtering
P) Dark Duration
Q) Far Red Intensity
R) Dark Pre-Time
S) Dark Post-Time
T) Dark Modulation
U) Dark Filtering
```

* Items with an asterisk are multi-valued. When you are prompted for them, the interface used is a multi-line editor. As an example, see **Prompting for Color** on page 9-59.

Figure 9-54. AutoProgram Builder's control options

Prompting for Color

Obviously, you'll have to be configured for the 6400-18 RGB Source. When you run the program, after the preliminaries to get the log file squared away, the first thing you'll be prompted for is the color list (Figure 9-55).
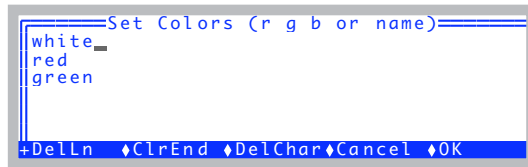
Figure 9-55. Prompting for colors

You can enter names or numbers for colors. An allowable name is anything that appears in your current color list[3]. Instead of a name, you can enter three values, for the fraction of red, green, and blue that you want. Below are some possibilities:

```
white
red
yellow
green
cyan
blue
magenta
1 1 1
.33 .33 .33
10 11 18
```

If you enter three values, leave one or more spaces between them. The order is red - green - blue, so "10 11 18" would be 10 parts red, 11 parts green, and 18 parts blue. If all three values are the same, that specifies white. "1 0 0" is red, "0 1 0" is green, and "0 0 1" is blue. Thus, you could set your list to be something like Figure 9-56, and press **f5**.
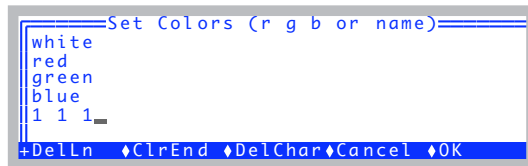
Figure 9-56. white, red, green, blue, then white again.

[3.]The file **/User/Configs/RGBColors**. See **Color** on page 7-23.

10

The LPL File System

Managing your data storage space

10

The LPL File System

This chapter describes the LI-6400's file system, and a very useful tool for its management: the Filer.

Files and Directories

The LI-6400's file system contains the programming that makes the instrument work, as well as the data files that result from that work. To efficiently manage all of this information, the storage space is partitioned into groupings of directories and files.

What Are Files?

Files on the LI-6400 are just like files on your computer: each is a collection of data that has a name and a time stamp (when it was last modified), among other attributes. You can copy, delete, view, edit, and otherwise manage these files just as you would on a computer. Some of the files on the LI-6400 are programs to control the LI-6400, while other files are your own creation, containing data you've collected.

Naming Convention

File names on the LI-6400 can be as long as you'd care to make them, and can consist of any combination of numbers, letters, spaces, and punctuation, except the following six characters:

```
/ \ * ? ; :
```

Upper and lower case *does* matter for letters, so the following could be distinct files:

```
MyData
mydata
MYDATA
```

Dots carry no significance in the LI-6400's file system, so names like

```
WheatData.plot2.aci.joe
```

are acceptable.

What Are Directories?

Directories provide a mechanism for grouping files logically. Directories contain the header information (name, date, etc.) for all of the files and sub-directories contained therein. Directories have the same naming convention as files.

To identify a particular file in the LI-6400's file system, use its name and the names of all the parent directories that contain it. For example,

> /User/Configs/AutoProgs/AutoLog

specifies a file named AutoLog, that is located in a directory named AutoProgs, that is in turn found in a directory named Configs, that is in the directory named User. Note that a slash (/) is used to separate directory and file names in the specifier. A slash in either direction will work, so the following are equivalent:

> /Sys/Open/Open
> \Sys\Open\Open

/Sys, /User, and /dev

There are three permanent directories: /Sys, /User, and /dev. These directories cannot be renamed or deleted.

/Sys contains all of the programming required by the LI-6400. You are not prevented from modifying any of these files or from storing files of your own in /Sys, but this is not recommended. Also, when you update software, everything in /Sys will be lost (and presumably replaced by something better).

/dev contains unit-specific calibration information. Files include: /dev/.factory - factory calibration coefficients, /dev/.user - user calibration coefficients (zero and span, for example), /dev/.lcd - the last display setting for the display contrast, /dev/.vcal - reference voltage values for your particular instrument.

/User contains data files and configurations that are generated and maintained by the user. As the name implies, it is your space to use, with one slight exception: the directory /User/Configs contains a number of files and subdirectories that are expected to be there.

Partitions

The file system has two partitions (Figure 10-1). Partition 1 holds the /Sys directory, and any directories or files that you might put in the root. Partition 2 holds /User and /dev.
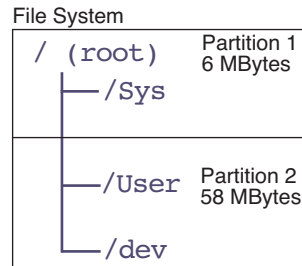
File System

Figure 10-1. The file system has two partitions.

When software is updated, all of the first partition is wiped out, but the second is left intact. For this reason, all user files should be kept in the /User - /dev partition (that is, be contained in the /User directory).

The Filer

The Filer is a general purpose tool for managing files and directories on the LI-6400 file system. Some of the functions that you can perform with the Filer include copying, deleting, renaming, and viewing files.

■ **Accessing the Filer**
There are a couple of ways to get to the Filer:

• **From OPEN**
From the OPEN main screen, press the Utility Menu function key (**f5**). Select "Access the FILER", which is in the "Files" node, and press **enter**.

• **From Power ON**
Power ON and press **escape**, to get to the LPL copyright screen. Then press **F**.

The main Filer screen shows directory statistics, a list of files, and function key labels (Figure 10-2). To exit the Filer, press **escape** while viewing this screen.

Current Directory

Use the cursor control keys to move the highlight bar.

```
/User             A-Ci bad
Filter=           A-Ci Iris leaf
Total: 34         A-Ci Test 4
Bytes: 71962      Data
Tagged: 0         Data (Jason)
Bytes: 0          MixerT...CurveExperimnt#1
                                    ← →
 1 Dir    ◆ fiLter◆ Sort   ◆ Tag    ◆ purGe
```

The file list

Which files to display. "*" or "" for all.

Press labels for more keys, or use the short cut keys.

3 dots (...) mean a long file name has been shortened to fit.

Press ← or → to view file time stamp and size.

```
 2 Copy   ◆ Move   ◆dUp    ◆ Rename◆ Print

 3 View   ◆ Bview  ◆ Edit   ◆ grapH ◆Exec

 4 space  ◆ trAsh  ◆        ◆ eJect ◆scaN

 5 Work ◆          ◆        ◆ Find  ◆
```

Figure 10-2. The Filer's main screen.

The Function Keys

To navigate through the Filer's function keys, press **labels** until the desired key appears, or press the number 1 through 5 to jump to that level. To execute a particular task, press the corresponding function key, or else type the letter associated with that key. This shortcut key is the capitalized letter[1] in the function key's label. Table 10-1 summarizes the function keys.

---

[1] Capitalizing a character in the label to indicate a key short cut is not usually true for other screens in Open, but it is in the Filer.

Table 10-1. The Filer's main function key labels and shortcuts.

Label	Short Cut	Description
Dir	**D**	Select a directory. See **Directory Dialog** on page 10-8.
fiLter	**L**	Change the filter for displayed files. The default filter is nothing, which selects all files. See **Viewing a Subset of Files** on page 10-11.
Sort	**S**	Sort the displayed list of files. See **Sorting the File List** on page 10-12.
Tag	**T**	Tag one or more files in the list. See **Tagging Groups of Files** on page 10-15.
purGe	**G**	Purge (move to the trash) all tagged files in the file list. See **Removing Files** on page 10-15.
Copy	**C**	Copy tagged files to another directory. See **Copying and Moving Files** on page 10-16.
Move	**M**	Same as Copy, except the original is purged.
dUp	**U**	Duplicate all tagged files. The new files will have "copy" appended to their name. See **Duplicating Files** on page 10-16.
Rename	**R**	Rename the highlighted file. See **Renaming Files** on page 10-17.
Print	**P**	Print all tagged files. See **Printing Files** on page 10-17.
View	**V**	View the highlighted file as a text document. See **Viewing and Editing a File's Contents** on page 10-13.
Bview	**B**	View any (text or binary) highlighted file.
Edit	**E**	Edit the highlighted file.
grapH	**H**	Run GraphIt (Chapter 12) for the highlighted file.
eXec	**X**	Run the highlighted file (if it's an LPL program). See **Executing Programs** on page 10-18.
space	**space**	Checks space on both disk partitions. See **Space Available** on page 10-19.
trAsh	**A**	Empty trash. **Emptying The Trash** on page 10-19.

Table 10-1. (Continued)The Filer's main function key labels and shortcuts.

Label	Short Cut	Description
eJect	**J**	Unmount the flash memory card, allowing it to be safely ejected.
scaN	**N**	Re-build the list of directories, and the list of files.
Find	**F**	Find matching file names in the entire file system. **Viewing a Subset of Files** on page 10-11.
Work	**W**	Set the current directory to be the default once the Filer is exited.

Filer's Directory Operations

Directory Dialog

The Directory Dialog (Figure 10-3), accessed by pressing **D** (or the **Dir** function key), shows all directories on all disks in a menu. To select a particular one, highlight it, and press enter.

```
══════════════Select a Directory══════════════
▼/User
▼  ┌Configs
▶  │ ┌AutoProgs
▼  │ ┌Comps
▼  │ ┌CustomClosed
+  ▼/▶   ♦ Add    ♦ Purge ♦ CANCEL♦ SELECT
+ Find   ♦fnd Nxt♦ Currnt♦ View  ♦ Rename
```

Open / Close directories. Directories that do not contain other directories are always shown as open (▼).

View (or **V**) alternates between the full path names (above) and a tree view (left).

```
══════════════Select a Directory══════════════
▼/User
▼/User/Configs
▶/User/Configs/AutoProgs
▼/User/Configs/Comps
▼/User/Configs/CustomClosed
+ Find   ♦fnd Nxt♦ Currnt♦ View  ♦ Rename
```

*Figure 10-3. The Directory Dialog is accessed by pressing **D** (or the **Dir** function key) from the main menu. The function keys are described in Table 10-2.*

In addition to allowing you to navigate the file system, the Directory Dialog allows some manipulation and housekeeping, such as renaming, creating, and removing directories. Table 10-2 summarizes the two levels of function keys.

Table 10-2. The Directory Dialog function keys.

Label	Short Cut	Description
▶/▼	(none)	Collapse / Expand a directory
Add	**A**	Add a new directory

Table 10-2. The Directory Dialog function keys.

Label	Short Cut	Description
Purge	P	Purge the highlighted directory. If the directory is not empty, you will be notified and asked if you really want to purge it.
CANCEL	escape	Exit the directory selection screen, and don't change directories.
SELECT	enter	Change to the highlighted directory, and exit the change screen.
Find	F	Enter a target string, and jump to the next directory that has that string within its name.
fnd Nxt	N	Find the next occurrence of the target string.
Currnt	C	Highlight the current working directory.
View	V	Toggle between tree view and full name view.
Rename	R	Rename the highlighted directory.

Creating

Use the **Add** function key in the Directory Dialog (or press **A**) to create a new directory. For example, if you wish to make a directory /User/MyData, highlight /User in the Directory Dialog, and press **A**. You will be prompted

```
====== Make New Directory ======
/User/_
```

Type in the rest of the name, and press **enter**.

Removing

Use the **Purge** function key (or press **P**) in "Directory Select". A directory that is not empty (has files or other directories) can be purged, but you will be warned first, and will have a chance to cancel the operation.

NOTE: You cannot remove /Sys, /User or /dev, nor should you wish to.

Files purged using the Filer go into a trash directory. One of the ways of emptying the trash is to select the _Trash directory in the Directory Dialog, and

press **Purge**. For more trash talk, see **Emptying The Trash** on page 10-19.

> **Warning**: Purging directories really does remove them, and any files they contain. They aren't moved to the trash.

Filer's File Operations

Viewing File Date and Size

The main Filer screen displays small left and right arrows (Figure 10-2 on page 10-5), which is a subtle clue that additional file information can be viewed by pressing the ← or → arrow keys. Pressing ← or → toggles the file list to show the time or date of last modification, or the file size, in bytes.

Figure 10-4. The ← and → keys will cycle through display modes.

Viewing a Subset of Files

Within a Directory

The file list can include all files in a directory, or a subset of them. This is controlled by the Filter setting (Figure 10-5)
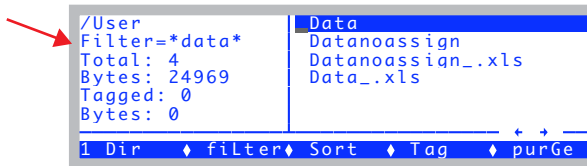
Figure 10-5. The Filter determines which files are shown. A blank, or *, will show all files. The characters * or ? are wild cards.

To show a particular set of files, specify a mask using the **fiLter** key (or just press **L**). You are prompted to enter the new Filter string. Examples are given in Table 10-3.

Table 10-3. Examples of Filters, and the files they would include and exclude.

Filter	Includes Files	Excludes Files
*	(all)	(none)
123	123.dat Data123456 123	12.3
Data.?	data.1 data.x	data data.123
*.???	junk.dat this is long.123	junk.12 junk.1234

Note that upper or lower case in the filter doesn't matter.

In the Entire File System

Press **F** (the shortcut for the **Find** function key). You will be prompted for a filter (Table 10-3). The entire file system is then searched for matches, and the results displayed in a list, grouped under their respective directories. If the filter is an empty string (or a *), you will get every file.
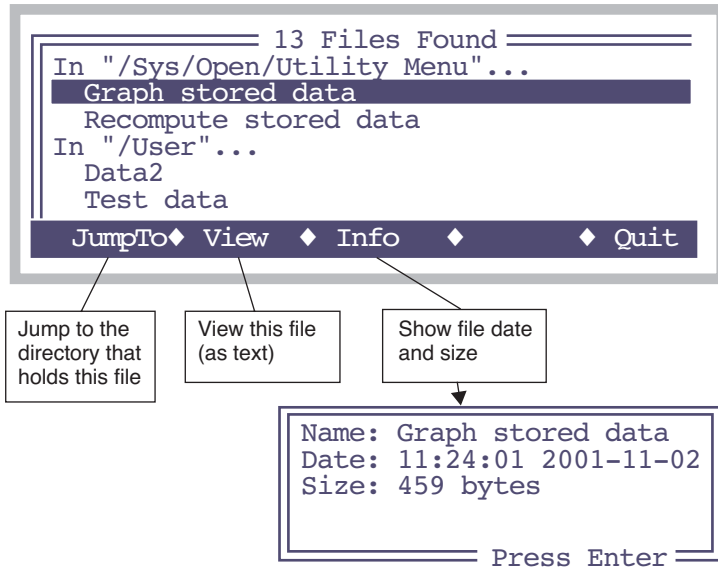
Figure 10-6. Results of Find. The target in this example was "*data*", so any file containing the word 'data' is listed.

Sorting the File List

Press the **Sort** key (or just **S**) to arrange the files alphabetically, or by size, or by date. The key labels will change to those shown in Figure 10-7. Following your selection, you are asked Ascending or Descending. The file list is then sorted accordingly.
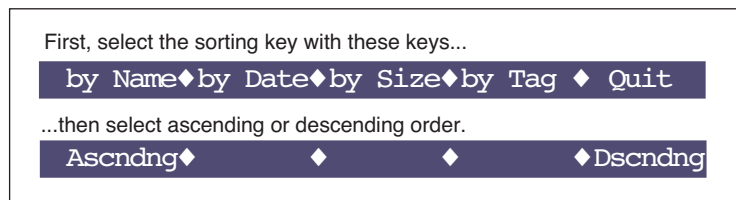
*Figure 10-7. Function keys for sorting. You can also press **N**, **D**, **S**, **T**, or **Q** for the sort options, and **A** or **D** for the direction choice.*

Viewing and Editing a File's Contents

Text Files

To view the contents of a file, highlight it and press **View** (or **V**). Refer to **Standard Menu** on page 5-3 for details. To edit a file, press **Edit** (or **E**) instead. Refer to **Standard Edit** on page 5-15 for details on using this editor. Whether viewing or editing, the display will look the same (Figure 10-8) until you press the labels key; different functions keys are defined for viewing and editing.
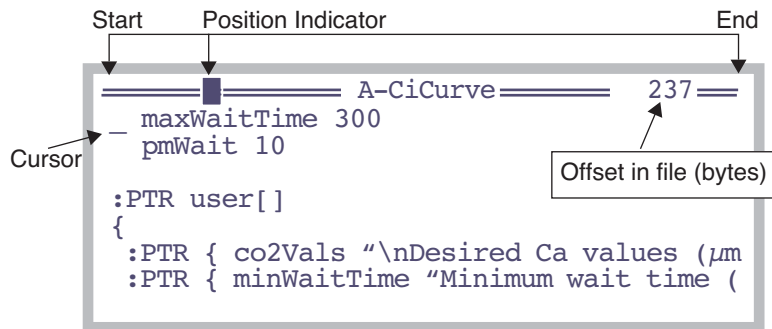
Figure 10-8. Viewing or editing a file looks the same. Viewing will not allow you to make changes, however.

You can also do graphical operations on a text file by pressing **H** (the **grapH** function key). This is described in **GraphIt** on page 12-1.

Binary Files

Any file can be viewed with the binary file viewer, accessed by pressing **B** (or the **Bview** function key).

```
FILE: A-CiCurve 1377
    0:2F 2A 0A 09 41 2D 43 69   /*..A-Ci
    8:20 43 75 72 76 65 0A 09    Curve..
   16:72 65 76 20 32 20 33 2F   rev 2 3/
   24:31 36 2F 39 35 0A 09 0A   16/95...
   32:09 39 36 30 38 32 32 20   .960822
   40:2D 20 54 65 63 68 20 4E   - Tech N
Press Any Key
```

Byte offsets into file

The 43rd byte of the file is a hex 65, or Ascii 'e'

Figure 10-9. Binary file view. Use **home**, **end**, **pgup**, **pgdn**, ↑ *and* ↓ *to move through the file. To jump to any arbitrary byte in the file, press* **J** *and enter the offset value.*

Tagging One File

Many of the Filer commands require that files or directories be selected ("tagged") before the command can be executed. These operations include purging, copying and printing files.

Press **enter** to tag the highlighted file (or the long way: press **Tag**, followed by **tag One**). A tag symbol appears next to the file name (Figure 5-3), and the highlight moves to the next file. When a file is tagged, press the **enter** key again to remove the tag. Additional function keys associated with the Tag function (below) are accessed by pressing **Tag**.

Tagged files are marked.

Number of tagged files, and size in bytes

```
/User            │  Datanoassign_.xls
Filter=          │  Data_.xls
Total: 71        │ ▸EX1
Bytes: 6900140   │ ▸EX1_.xls
Tagged: 2        │  EX2
Bytes: 16414     │  EX2_.xls
                 ─────────────── □ ▯ ──
1 Dir   ♦ filter♦ Sort  ♦ Tag   ♦ purGe
```
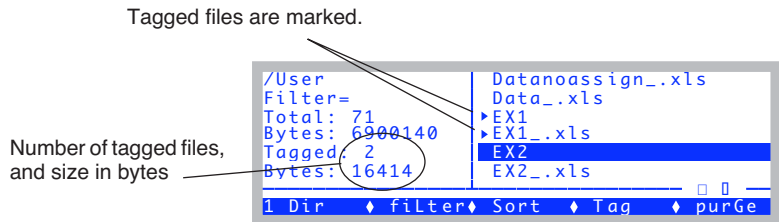
Figure 10-10. The Filer's main screen, showing two tagged files.

Tagging Groups of Files

To tag a group of files, press **T** (or the **Tag** function key), and use the tag function keys (Table 10-4).

Table 10-4. Tag function keys

Label	Short Cut	Description
Tag all	**T**	Tag all files in the current directory list.
Clr all	**C**	Remove all tags from files in the current directory list.
Retag	**R**	After you have performed certain functions (e.g., copying files), the tag marker will change to a - (hyphen) to indicate that the command was executed on the tagged files. Press **Retag** to tag these files again for further operations.
Invert	**I**	Tag all files that are currently *not* tagged, and remove the tag from those files that *are* tagged.
tag One	**O**	Tag the currently highlighted file. (A faster way to tag one file is to press **enter** in Filer's main screen.)

Removing Files

To remove files, you must first tag them (above), and then press the **purGe** key (or **G**). Press **Y** (Yes) at the prompt to delete the file(s). Purged files are placed in the trash directory (_Trash) of the ultimate parent directory. For example, if you purge the file /User/Joe/BadData, it will be moved to /User/_Trash. The trash must be emptied to actually remove old files and free up disk space. See **Emptying The Trash** on page 10-19.

Copying and Moving Files

The only difference between moving and copying is that the original file is purged after moving.

■ **To move or copy files from one location to another:**

1   **Tag the file(s) to be copied**
Tag by pressing **enter** or **space**, or by the Tag function keys.

2   **Press M for Move, or C for Copy**
Or use the function keys.

3   **Select the destination**
A menu of destinations is presented. Highlight the desired destination directory, and press **enter**. Or press **escape**, if you've changed your mind.

4   **Select the overwrite option**
Your choices are shown in Figure 10-11. Press **Y** (Yes) to overwrite existing files with the same name, **N** (No) to prevent files from being overwritten, or **A** (Ask) to bring up another prompt (**Y**/**N**) each time such a file is found.

Or Move

```
Copy to /User/Ralphy

Overwrite duplicate files ? (Y/N/A)
  (Yes, No, Ask when it happens)


   Yes      ♦          ♦ Ask     ♦          ♦ No
```
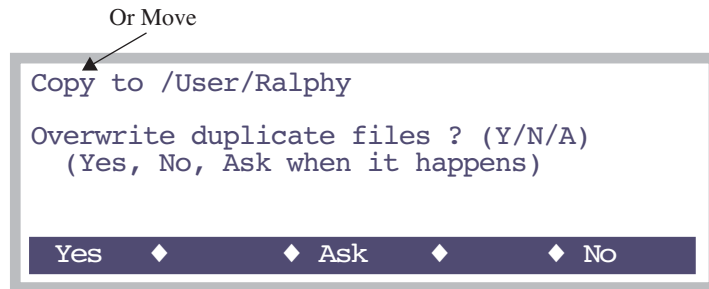
Figure 10-11. When moving or copying, you are given options on what to do if there is a duplicate file in the destination directory.

There is a utility program (/Sys/Utilities/Nested Directory Copy) you can use to copy all files and directories that are contained within a particular directory. See **Nested Directory Copy** on page 21-16.

Duplicating Files

The purpose of duplicating files is to allow copying to the same directory. All tagged files generate a duplicate, with "copy" appended to their name. To duplicate tagged file(s), press **U** or **dUplct**.

Renaming Files

■ **To rename a file:**

1 **Highlight the file to be renamed**

2 **Press Rename (or R)**
You are prompted for a new name (Figure 10-12).

3 **Press escape or enter.**
enter renames the file, **escape** aborts.

```
┌──────────────────────────────────────────────┐
│  /User              │  A-Ci bad              │
│  ┌─ New Name ────────────────────────────┐   │
│  │ MixerTestResults_                      │   │
│  └────────────────────────────────────────┘   │
│  Tagged: 0          │  Data (Jason)          │
│  Bytes: 0           │  MixerTestResults      │
│                                  ← → ─        │
│   DelLn  ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar   │
└──────────────────────────────────────────────┘
```
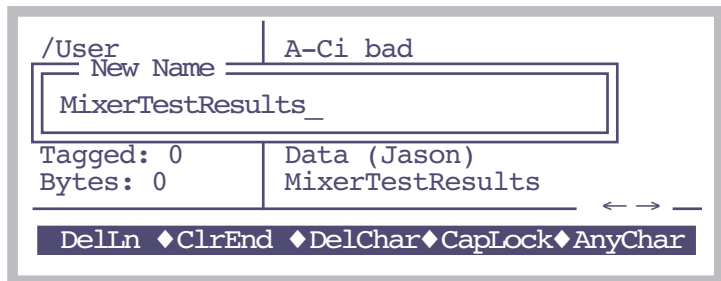
Figure 10-12. Renaming the highlighted file. Standard Line Edit is used (page 5-5).

Printing Files

Downloading files is discussed in Chapter 11. One "last-resort" method of downloading them is to tag them in the Filer, and press **P** (or **Print**). The print program that then runs will prompt you for the following information (Figure 10-13):

```
Print headers? (Y/N)N
Append FormFeeds? (Y/N)N
Conversions  N) None
  S) to Space delimited
  C) to Comma delimited
  T) to Tab delimited
```
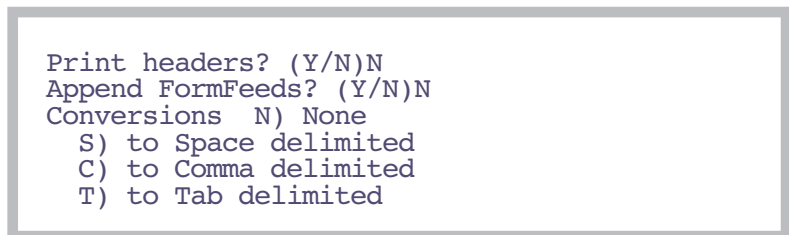
Figure 10-13. The print program, for sending selected files to the Comm Port.

- **Print headers? (Y/N)**

  If you press **Y**, a banner will be output before each file. The banner will include the file name, modification date, and current date.

- **Append FormFeeds? (Y/N)**

  If you press **Y**, each file will be followed by a form feed (decimal 12), which will cause most printers to eject the page. This allows each file to begin on a new page, if you are printing multiple files to a printer.

- **Select Conversion**

  You will be prompted for a conversion type. Data files are stored as tab delimited. You can convert this when you print, if you like: **N** leaves the file alone, **S** puts the data into columns, **C** uses comma delimiters, and **T** uses tab delimiters.

  The file(s) are then sent to the Comm Port.

Executing Programs

LPL programs can be run from the Filer by highlighting the file containing the program and selecting **eXec** (or **X)**. After the program is executed, you will return to the Filer. As an example, you can set the communications parameters from the Filer by executing the /Sys/Utility/Setcomm program file.

Utility programs in the directory /Sys/Utilities are described in **/Sys/Utility Programs** on page 21-11.

Housekeeping
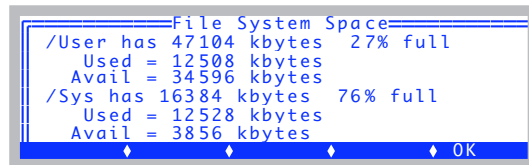
Space Available

The space available on both partitions (as well as on a compact flash card if it is installed - LI-6400XT only), is displayed by pressing **space** (or the function key labelled **space**).

■ **Find the space available on the file system:**

1 **In the Filer, press the space function key (or press space).**
The space on both partitions will be displayed. (Figure 10-14). Press any key to return to the Filer.

```
═════════════File System Space═════════════
/User has 47104 kbytes  27% full
     Used = 12508 kbytes
    Avail = 34596 kbytes
/Sys has 16384 kbytes  76% full
     Used = 12528 kbytes
    Avail = 3856 kbytes
    ♦         ♦         ♦         ♦ OK
```

Figure 10-14. The available space display. /Sys and /User represent two disk partitions.

Emptying The Trash

It is necessary to empty the trash directory to reclaim space occupied by purged files. Each directory in the root will potentially have its own trash directory, named _Trash. If no files have been purged from that directory or any of its children, there will be no _Trash.

■ **There are two ways to empty trash:**

• **From the Directory Dialog**
Select the trash directory you wish to empty (typically, it would be /User/_Trash), and press **P** (or the **Purge** function key).

• **While viewing a file list**
Press **A** (or the **trAsh** function key) to empty the trash associated with the file list you are viewing.

Flash Memory

The expansion slot in the LI-6400XT will accept Compact Flash cards.

Figure 10-15. Inserting a Compact Flash card into the expansion slot.

1   **Insert a Compact Flash card**
A few seconds after one is inserted into the slot, you should hear three quick beeps, then see the following message.

```
Compact Flash Device ready at

    /Flash

                       Press any key
```

2   **Use it**

At this point, the flash device will show up in the file system under the directory /Flash. If you go to the Filer, and press **D** to view the directories, you will see the Flash directory listed (Figure 10-16).

```
═══════════ Select a Directory ═══════════
▼/
►/dev
► Flash
►/Sys
►/User

 +   ►/▼  ◆  Add    ◆  Purge ◆  CANCEL◆ SELECT
```
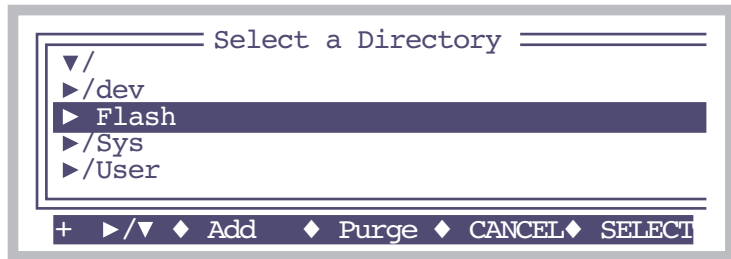
Figure 10-16. When mounted, the compact flash will appear as a directory named "Flash". You can add files and/or subdirectories to it as you wish.

While it is mounted, you can treat /Flash just like the /User disk: You can open and write files to it, make directories, copy files to and from it, etc.

3   **Before removing, do a software unmount.**

Before physically removing the compact flash card from the slot, you should always do a software unmount first. There are a number of ways to do this:

(Anytime) Hold the **ctrl** and **shift** keys down, and press the **end** key. This is one of the system hot keys (Table 3-1 on page 3-3 lists them all).

(In Filer) Press **J** (shortcut for **f4** level **4**).

```
4 space  ◆  trAsh  ◆        ◆  eJect ◆scaN
```

(In Filer). In the directory list, highlight /Flash and press **Purge** (**f3**) (Figure 10-16).

(LPL Command). From LPL's command line, type CFUNMOUNT (case doesn't matter), and press **enter**.

```
ok:cfunmount
```

Any of these methods will unmount the /Flash disk, so that it can be safely (for

your data) removed. There will be an accompanying message:

```
Compact flash device

    /Flash

can now be removed.
                    Press any key
```

If there is a problem with unmounting the disk, such as if there is still an open file, you will get a message about that instead.

To physically remove the flash card, press the eject button (Figure 10-15 on page 10-20).

Connecting to a Computer

11

....

Retrieving your data, and remote control

11

Connecting to a Computer

Eventually you will want to move files from the LI-6400 to your computer or load files back onto the LI-6400. Also, there may be times when you need to control the LI-6400 from your computer, such as when making measurements in a small growth cabinet, or doing a classroom demonstration with the LI-6400 display projected onto a large screen.

There are several options for all of these tasks.

Support Software

Download and Install

A collection of LI-6400 Apps for OS X, Windows, and Linux is available on the CD that ships with the instrument, or from the LI-COR website.

Windows

The installer is an executable that will install three apps, and put their icons on your desktop.

Mac OS X

The installer is a .dmg. Once mounted, simply drag the LI-6400Apps to the Applications folder shortcut. Once you have done that, you can eject the disk, and use the Finder to navigate to /Applications/LI-6400Apps, and double click the app you wish to run.

Linux

The source code for the apps is in a .tar.gz file, which contains a .spec file for installing and building.

LI6400XTerm

This program allows remote control of any LI-6400 through RS-232, Ethernet, or over the Internet via the li6400.licor.com server. You can also use LI6400XTerm for file exchange between the instrument and your computer. See **Using LI6400XTerm** on page 11-30.
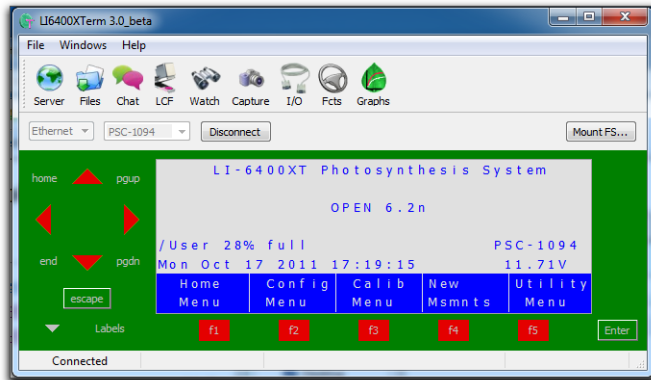
Figure 11-1. The Windows version of LI6400XTerm.

LI6400Group

This program allows control of multiple LI-6400s. See **Using LI6400Group** on page 11-34.
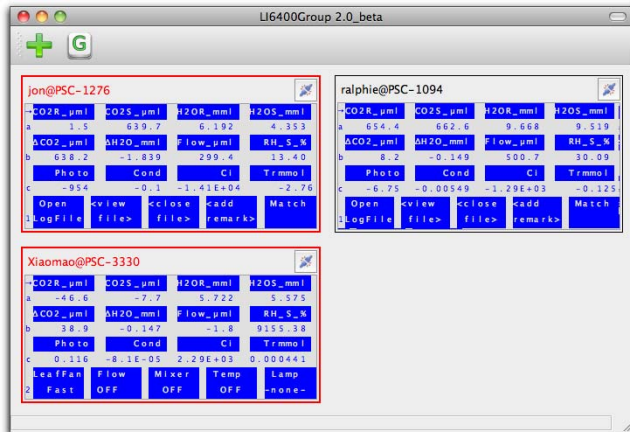
Figure 11-2. The Mac version of LI6400Group.

LI6400Sim

This program simulates an LI-6400, and is useful for seeing how things work (testing AutoPrograms, for example) when there is no instrument at hand to use (Figure 11-3). The simulator supports previous versions of OPEN as well. In the preferences dialog, you can select versions all the way back to 3.4.3.
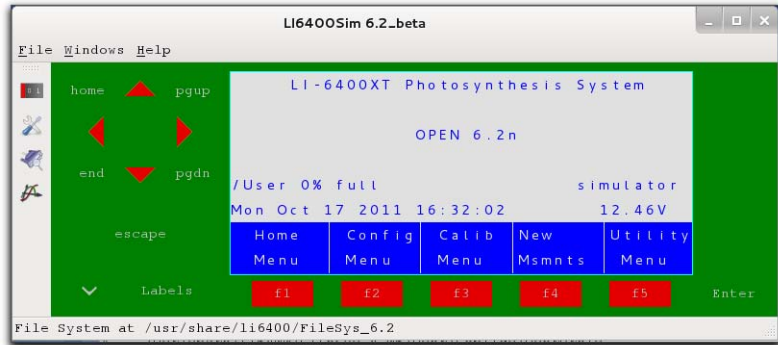
Figure 11-3. The LI6400Sim program, Linux version.

LI6400Term for iOS

There is an LI6400 terminal program available from Apple's App Store (Figure 11-4 on page 11-5), and another for iPad (Figure 11-5 on page 11-6) These apps allow control of an LI-6400XT via WiFi (the LI-6400XT must be connected to a wireless router or a LAN with wireless capability), or any LI-6400 via the li6400.licor.com server.
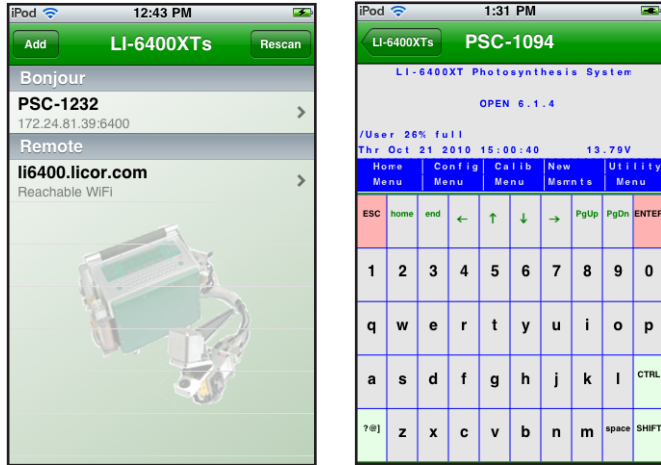
Figure 11-4. The LI6400Term app for iPod and iPhone.
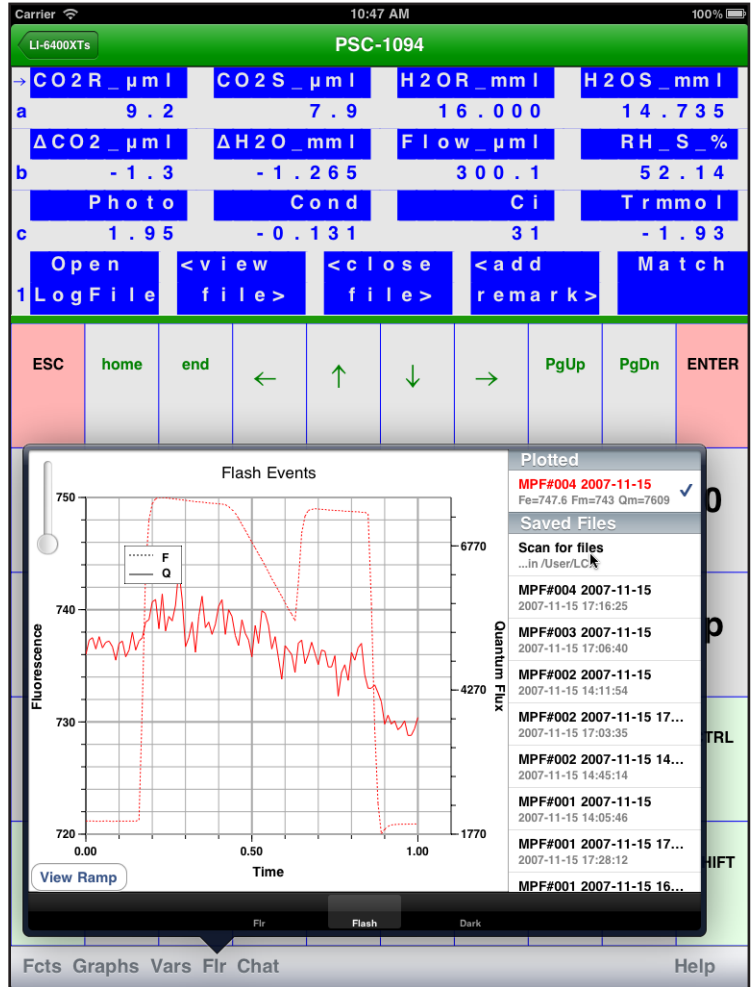
Figure 11-5. The LI6400Term iPad app.

Connecting with Ethernet

The LI-6400XT has an expansion slot that accepts an Ethernet adapter (6400-26)[1], included in the spares kit. Configuration is automatic: plug in the card and cable, plug the other end of that cable into your computer or switch, and in a few seconds the LI-6400 will be configured and ready to go.
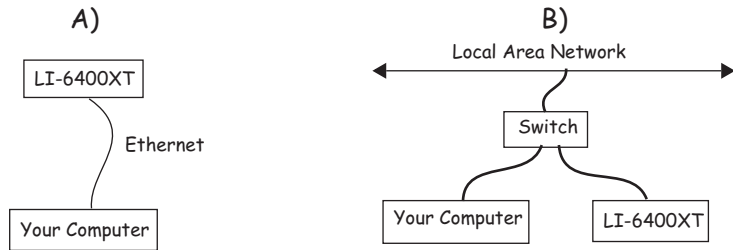
Figure 11-6. You can connect the LI-6400XT directly to your computer, or directly to the LAN to which your computer is connected. If you need more ports, use a multi-port Ethernet Switch.

Note: If the direct connection (A) doesn't seem to work, try plugging both computer and LI-6400XT into an Ethernet Switch, as in (B).

The LI-6400XT supports Secure Shell (SSH), SSH File Transfer Protocol (SFTP), Samba (file sharing on Windows, Mac OX, Linux, Unix), and Zero Configuration Networking.

The following sections provide step-by-step instructions to guide you through the various operations available via Ethernet connection to your LI-6400XT.

Note: For all Ethernet interactions, the LI-6400XT does not have to be in any special mode. It can be doing anything - New Measurements mode, LPL screen, etc.. The only requirement is that the console is powered on, with the network card and cable plugged in.

[1.] Caution: other vendors' cards may or may not work in this slot. Also, the 6400-26 has an installation CD. You can ignore it: it does NOT need to be used when using the card in the LI-6400XT.

Installing Ethernet

1   Plug in the Ethernet adapter card
It goes in label side up. A few seconds after you plug it in, you should hear three quick beeps.

Eject button

Figure 11-7. Ethernet adapter card inserted in the LI-6400XT expansion slot.

2   Verify the configuration
You can verify the configuration on the display of the LI-6400 in the Network Status program, described on page 11-9.

Operational hint:
If network operations become unresponsive, try ejecting and re-connecting the Ethernet adapter card.

The Network Status Program

The Network Status program is accessible from either the LPL screen, or OPEN's Utility Menu.

From OPEN's Utility Menu
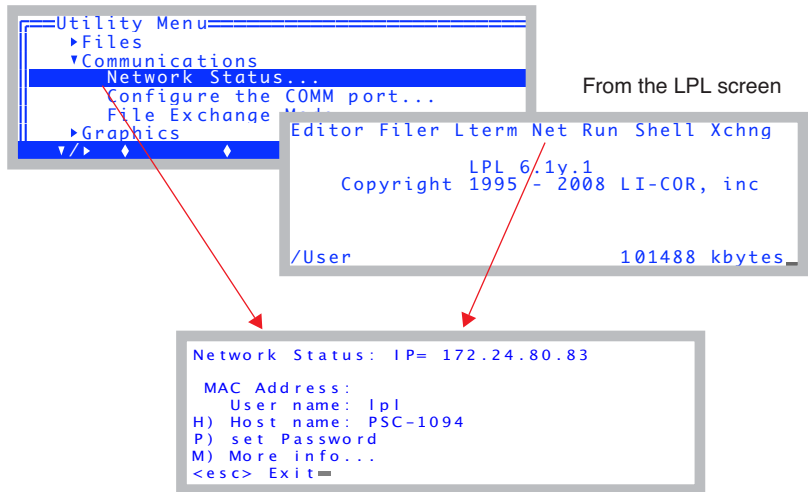
Figure 11-8. The Network Status display shows the LI-6400's connection status and host name.

If you are successfully connected, you'll see the IP address at the top of the display. Alternative messages include:

"Not possible" - this LI-6400 is not an XT and has no compact flash expansion slot.[2]

"Card not inserted" - the Ethernet card is not inserted in the compact flash slot.

"Cable not connected" - an Ethernet cable is not connected.

"Acquiring..." - The LI-6400 is still configuring itself to the network.

_____

[2]Upgrades are available, of course. Just have your people contact my people...

Normally, during the few seconds after connecting to a network, the message line will change several times between an IP address, the "Cable not connected" message, and the "Acquiring..." message, as the self-configuration takes place.

The <u>Host name</u> is the name the LI-6400XT will have on your LAN. By default, this will be the instrument's serial number, but you may change it to any valid name (start with a letter, avoid spaces, case insensitive). Examples: qwerty, my6400, psc-1234, BoSox07.

The <u>User name</u> is always lpl (lower case LPL). The default <u>password</u> is also lpl. The User name and password will be needed to access the LI-6400XT for file sharing from a remote computer.

The <u>MAC Address</u> will be the answer to a question your IT people might ask if they need to make special allowances for you and your LI-6400XT. Otherwise, ignore it.

The "More info..." option (press **M**) will show a screen similar to Figure 11-9.

```
══════More  Network  Information══════
Services  active:  6400  6409
Clients  active:    <none>
To  attach  this  XT's  file  system
to  your  PC,  you  can  use  the  host  name:
Windows→  \\PSC-1094
      Mac→  smb://PSC-1094.local.
   Linux→  nautilus  smb://lpl:lpl@PSC-10
or  you  can  use  the  IP  address:
Windows→  \\172.24.80.83
      Mac→  smb://172.24.80.83
   Linux→  nautilus  smb://172.24.80.83
 Sort    ♦  Find   ♦  ReFind♦ JumpTo♦ OK
```

Figure 11-9. Services should be 6400 and 6409. Clients shows the number of remote connections using 6400 or 6409. That is, connections using LI6400XTerm, LI6400Group, etc. The remaining information pertains to mounting the LI-6400XT's file system on your PC. This is discussed below in this chapter.

Making the LI-6400XT Wireless
While the LI-6400XT has no inherent wireless capability, you can add it via the Ethernet card, by connecting to a wireless router.

(Windows) File Transfer with Explorer

1 Open an Explorer Window
Click on, for example, "My Computer".

Enter the host name or
address. e.g. either
\\PSC-1232
or
\\172.24.81.39 ...

... or navigate by
starting at "Network".
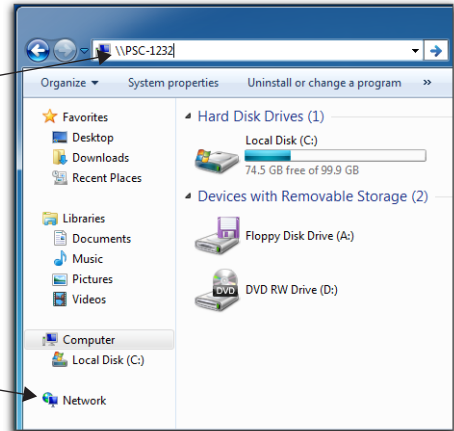
Figure 11-10. Get to the LI-6400 either by entering the host name, or by navigating.

You can either navigate to the LI-6400 by clicking on icons, or get there in
one step by entering the host name. If you navigate there, you may have to go
through (depending on what version of Windows you have, etc.) "My Net-

work Places" and "Entire Network". If you see a work group named Licor, look in that. Eventually, you should find your XT in a list:
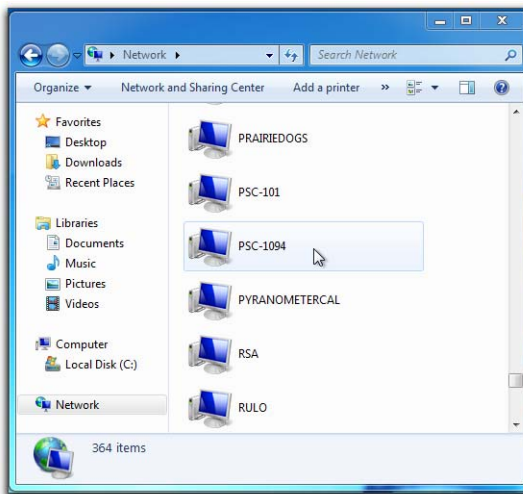
Figure 11-11. Navigating to an LI-6400XT with Windows Explorer

2 Authenticate
When you open the server for the first time (since login), you will have to authenticate the connection by entering the user name (lpl) and password (lpl again, unless you have changed it):
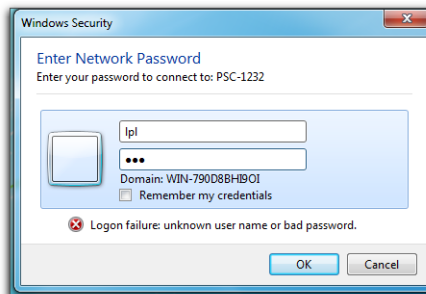
Figure 11-12. Use lpl for the user name and the password.

3 Navigate to the User directory
When Windows opens the LI-6400XT server, you will see an "lpl" folder. Double click it.

This will bring you to the file system you are familiar with on the LI-6400. Your data and configuration files are in the User folder, and you can drag and drop from there to your desktop or other Explorer windows.
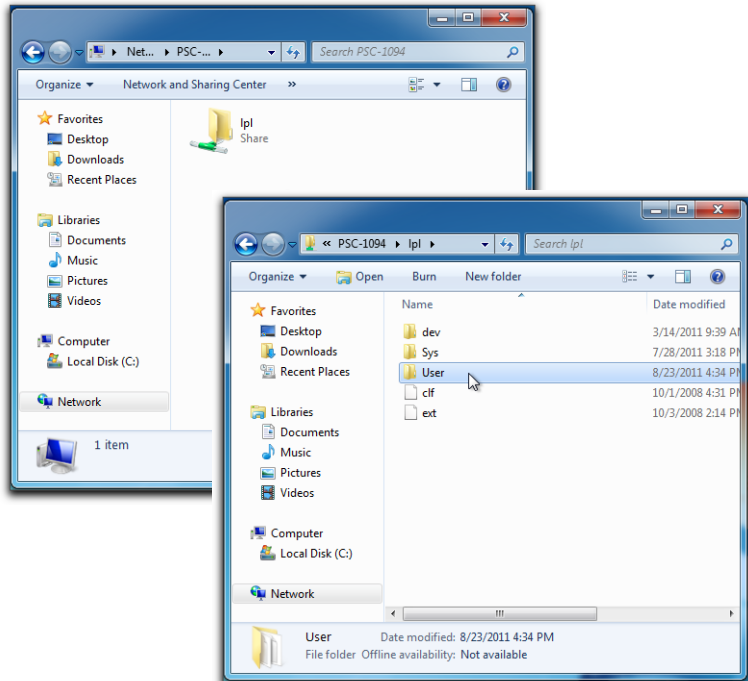
Figure 11-13. Your data files will be in the User folder. Sys contains programs, and dev contains instrument specific calibration files.

Opening an Explorer Window from LI6400XTerm
Another way to open an Explorer window is to start with LI6400XTerm, and click the Open File System button. (You need not be communicating with any instrument to do this.)
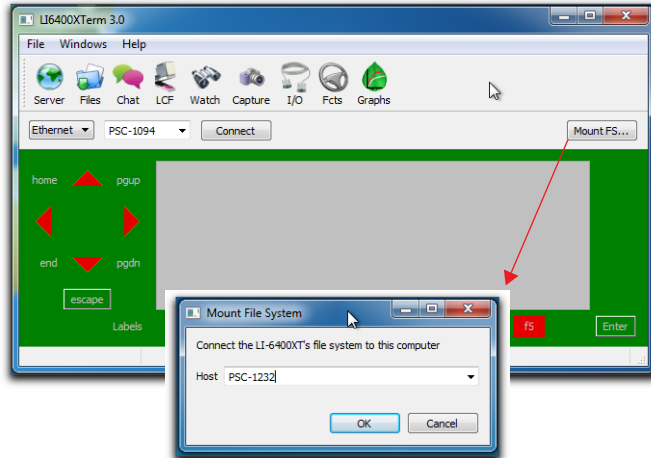
Figure 11-14. Using LI6400XTerm to mount an XT file system.

When you select an instrument from the drop down list, and click OK, you will then either be show the authentication window (Figure 11-12 on page 11-12), or the Explorer window for lpl (Figure 11-13 on page 11-13).

(Windows) File Transfer with WinSCP
WinSCP is a freely downloadable (winscp.net) Windows application that supports SFTP. It is a reliable alternative if you are having trouble making Windows recognize the LI-6400XT. For example: when we plug an LI-6400XT into the LAN here at LI-COR, it can sometimes take several minutes before it becomes "visible" in the Microsoft Windows Network. WinSCP and LI6400XTerm, on the other hand, will find it immediately.

1   **Run WinSCP, and log in**
Once installed and launched (after a restart), WinSCP presents its connection dialog (Figure 11-15). Enter the Host name, User name, and password, and then press Login. The host name and password should match your instrument (Figure 11-8 on page 11-9).
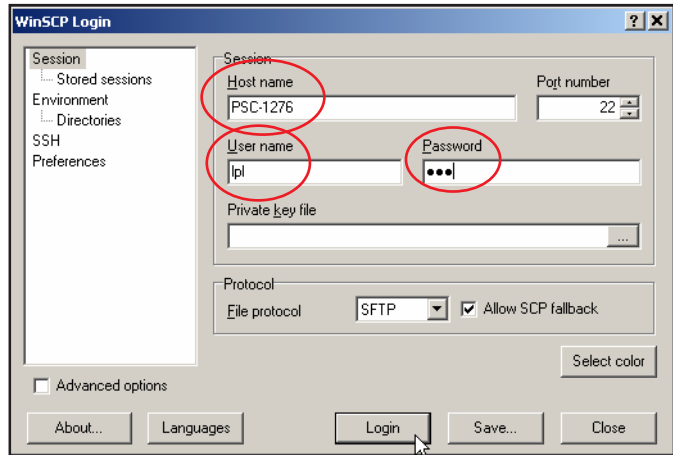
Figure 11-15. Connecting via WinSCP.

2   Drag and Drop as needed

WinSCP will open a window to the LI-6400XT's file system, from which you can drag and drop files just like with Explorer (Figure 11-16).

Notice that dev, Sys, and User aren't at the root, like they seem to be when you are using the Filer on the LI-6400XT. WinSCP is showing you the true file structure: these folders actually reside in /home/lpl/, and if you navigate back toward the true root, you'll find some of the files you'd expect to find on a Linux machine, which is in fact what the LI-6400 is.
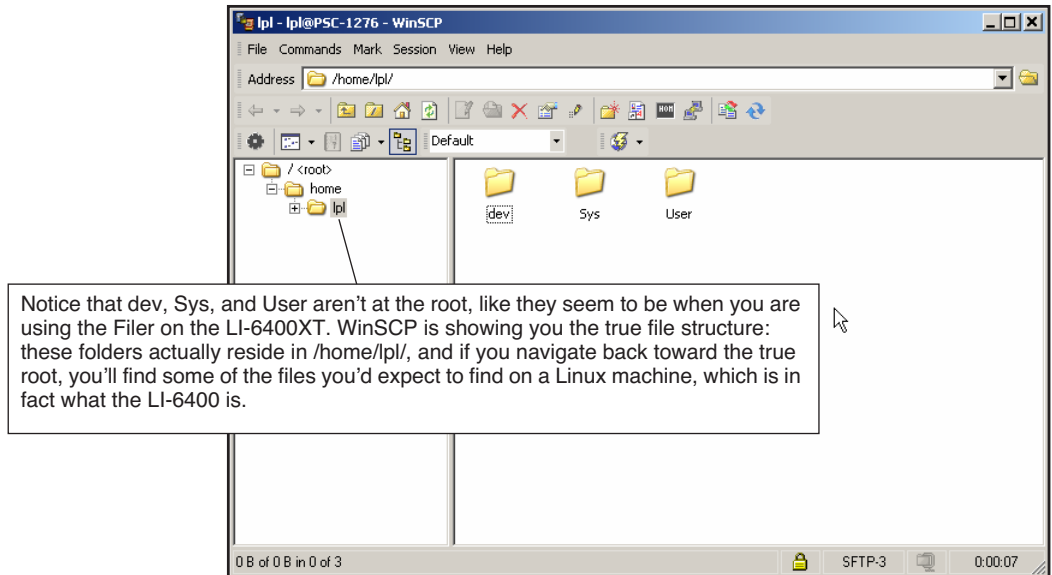
Figure 11-16. Using WinSCP.

(Mac OS X) File Transfer with Finder

From a Mac with OS X, you can connect to the LI-6400XT with Finder, either by directly entering the host name, or navigating to it:

1   Enter the host name or IP address

Open the "Connect to Server..." dialog, under Go in Finder's Menu Bar. Precede the host name with `smb://` and put `.local.` behind it (Figure 11-17).
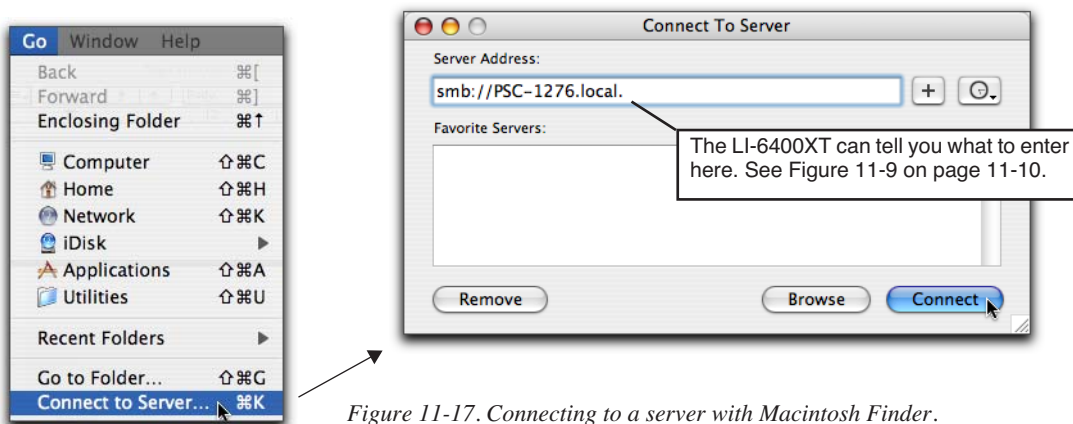
Figure 11-17. Connecting to a server with Macintosh Finder.

If you enter the IP address, put "`smb://`" in front of it (e.g. `smb://172.24.80.33`).

2   Verify

You will be asked to select a share. Your only choice is lpl, so just click OK (Figure 11-18).
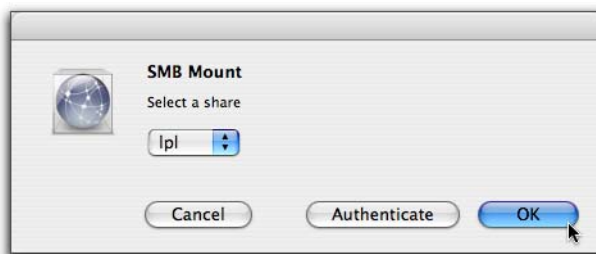
Figure 11-18. The share name is lpl.

3   Authenticate
The workgroup should default to LICOR. Enter the user name (always lpl) and password (also lpl, unless you have changed it), and click OK.
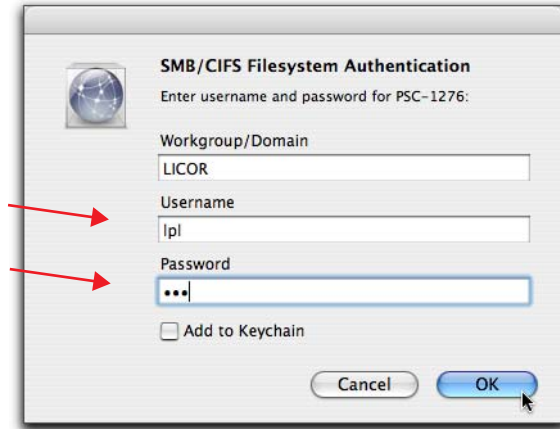
Figure 11-19. Enter the username and password.

4   Open the LPL server
Finder will open a window for lpl (the LI-6400XT's file system), and add it to the Shared items on the left side of the Finder window.
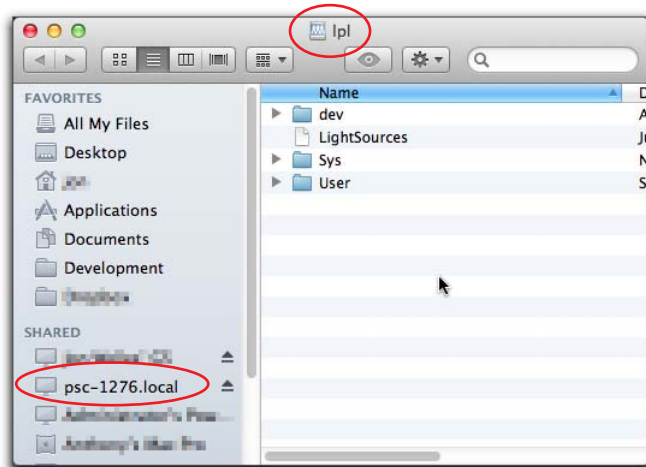
Figure 11-20. The file system of an LI-6400XT shown in Finder.

You can now drag and drop files or folders, just as if the LI-6400XT were a mass storage device connected to your computer.

When you are done, you can disconnect if you wish by selecting Eject from Finder's File Menu, or clicking the eject icon to the right of its Shared list entry.
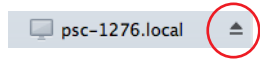
Opening a Finder Window with LI6400XTerm
You can also open a Finder window for the XT using LI6400XTerm, by clicking on the Mount FS button.
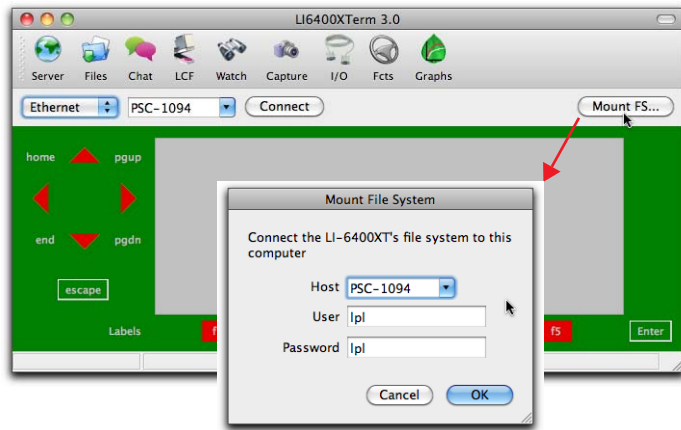
Figure 11-21. Opening a Finder window with LI6400XTerm.

Pick the instrument, click OK, and a Finder window will open in a few seconds.

Finder mounts the LI-6400XT in the Mac file system at /Volumes/lpl.

(Mac OS X) File Transfer with Fetch

Fetch is a nearly-free, downloadable (fetchsoftworks.com) file transfer application for Mac OS that supports SSH and SFTP, but with a graphical interface.

1 **Run the application and log in.**

The New Connection dialog lets you specify host name, user name, and password (Figure 11-22).
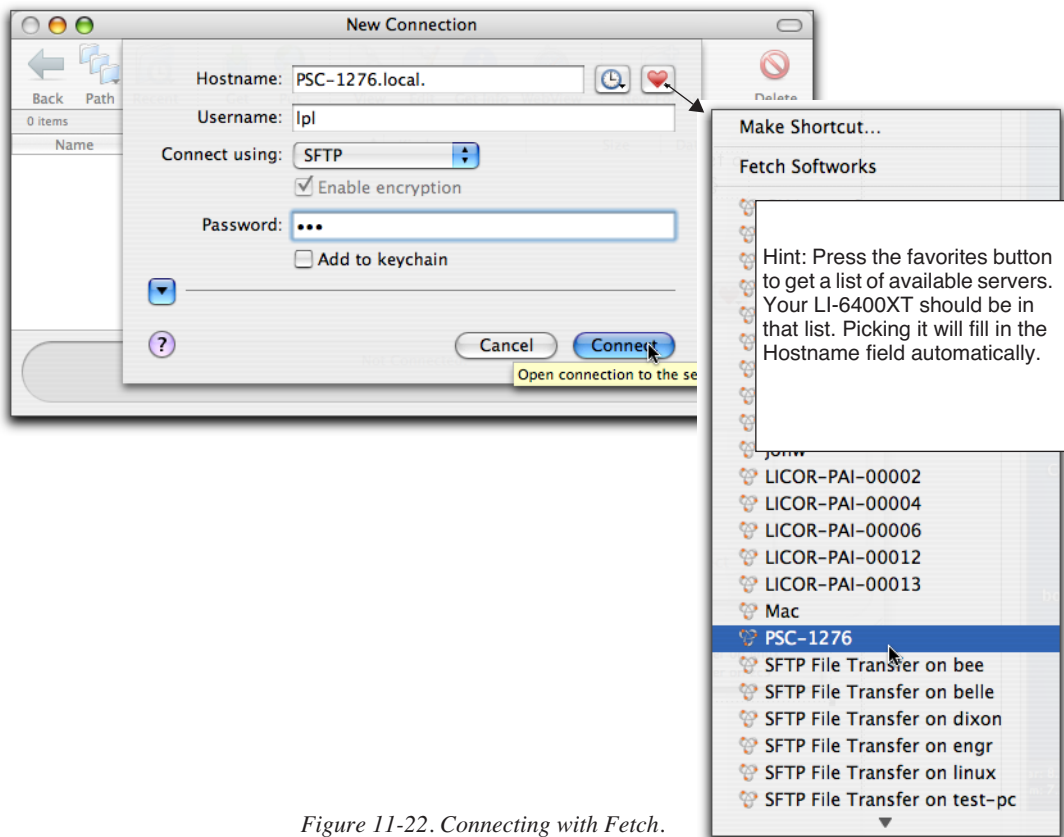
Figure 11-22. *Connecting with Fetch.*

2   Explore, Drag, Drop, etc.

Once you are logged in, the LI-6400XT's file system will appear in the Fetch window (Figure 11-23). You can then drag, drop, and otherwise manipulate files.
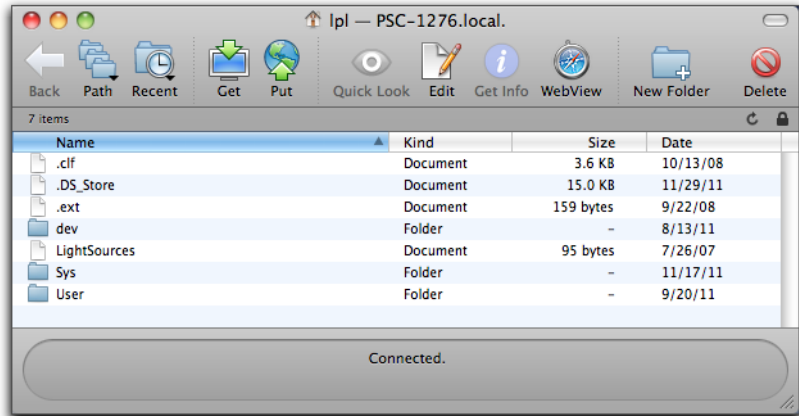
Figure 11-23. Fetch's view of the LI-6400XT file system.

(Linux) Connecting with Nautilus

The examples below are from Fedora 15.

Method 1: "Connect to Server"

Use the "Connect to Server" entry in the Places drop down.
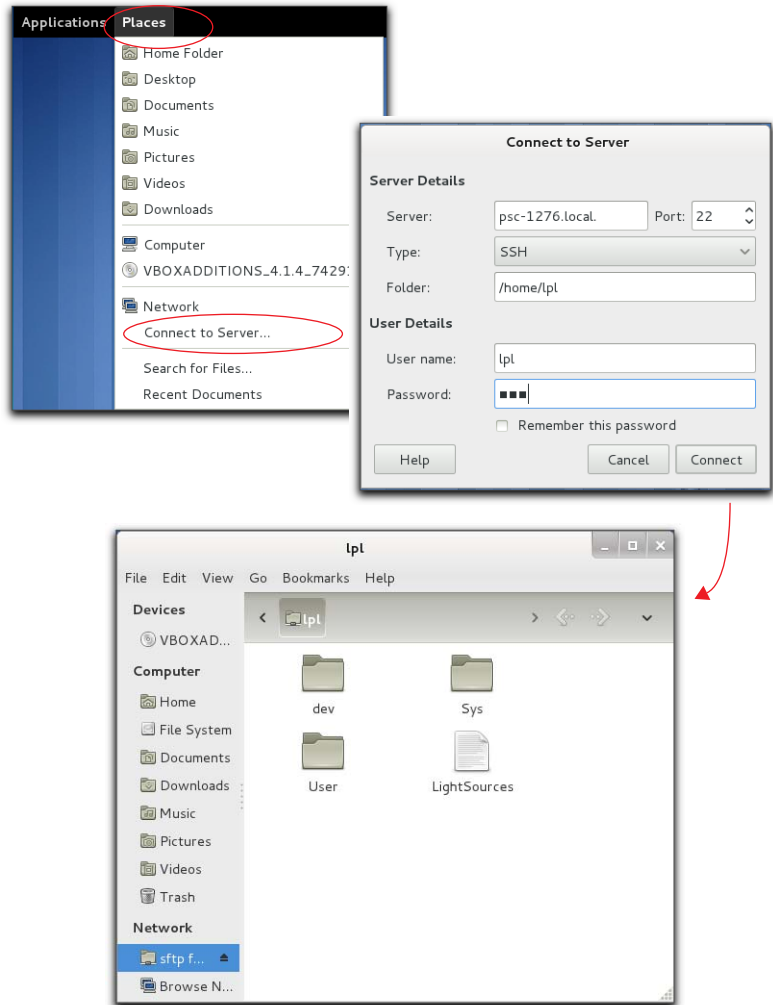
Figure 11-24. Connecting to the LI-6400XT with Connect To Server.

Method 2: Browse Network

From a Nautilus window, select Browse Network in the side bar, find your LI-6400XT, and double click it (Figure 11-25).
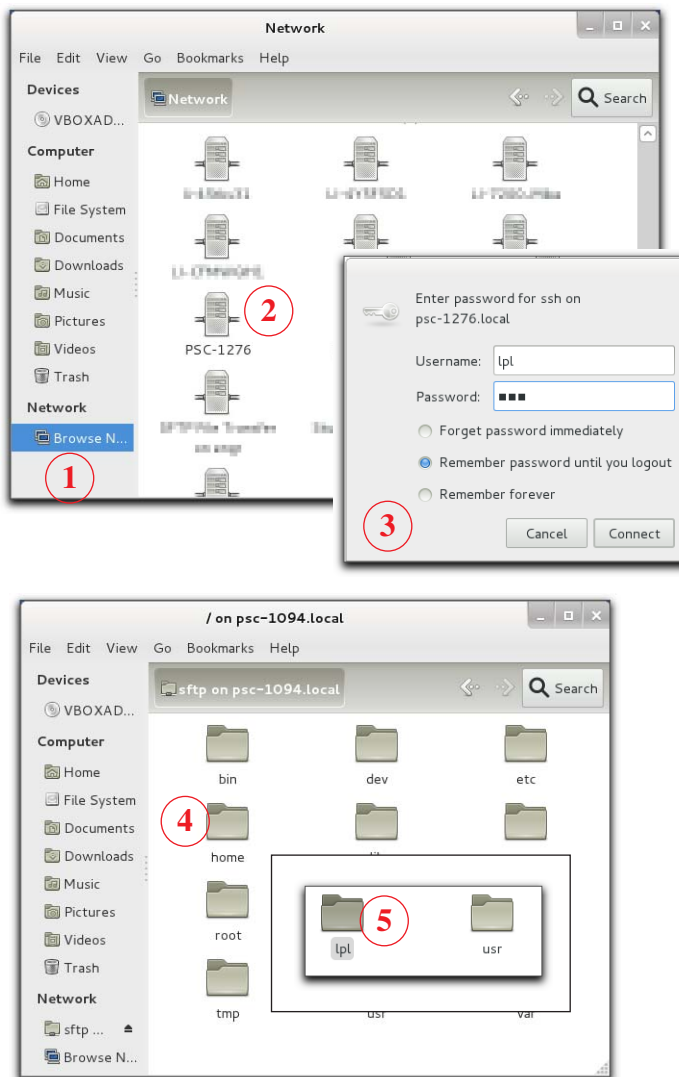
Figure 11-25. Using Browse Network to get to an LI-6400XT file system.

Method 3: Launch Nautilus from a Command Line

You can start from the command line in Terminal, and launch a file browser aimed at your LI-6400XT (Figure 11-26). The example uses psc-1276, so substitute the host name of your own LI-6400 (see Figure 11-8 on page 11-9).

(1) `$ nautilus smb://lpl:lpl@psc-1276.local.`
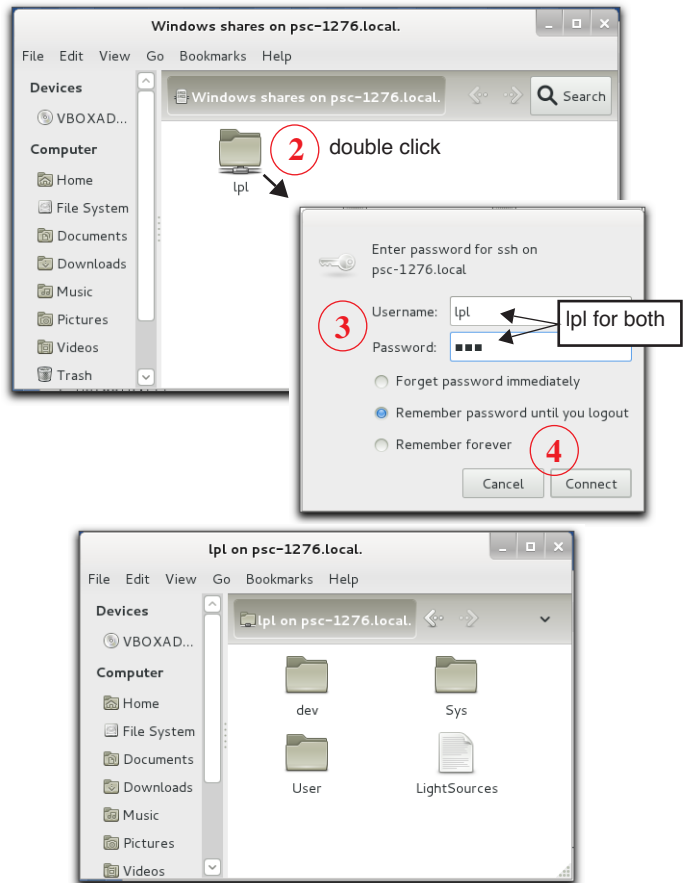
Figure 11-26. Opening a GUI interface to the LI-6400XT file system from Linux.

(Mac OS X, Linux) Command Line File Transfer

If you don't mind working with a command line, you can do all manner of evil with the LI-6400XT, using rcp, rsynch, and the like.

You need to know that the real path to the file system that you see on the LI-6400XT is /home/lpl/. Thus, the data file /User/data is really /home/lpl/User/data. The user name is lpl, and the password is lpl.

Here are two simple examples:

Copy a data file named 'mydata' from the /User directory on the LI-6400XT to the destination directory /Users/jon/test on your computer:

```
rcp lpl@psc-1094.local:/home/lpl/User/mydata /Users/jon/test
```

To copy the *entire* contents of the /User directory into the existing directory /Users/jon/Test, preserving file time stamps and permissions, do this:

```
rcp -rp lpl@psc-1094.local.:/home/lpl/User/* /Users/jon/Test
```

NOTE: If you have mounted the LI-6400XT's file system to your computer, then you can treat its files as "local", and you needn't worry about host names and passwords. For example, with OS X, you can mount the file system with Finder (**(Mac OS X) File Transfer with Finder** on page 11-16), then all the LI-6400XT files will be in the location */Volumes/lpl*. So, for example, from Terminal, you can get a list of the files in the LI-6400XT's /User directory by typing this:

```
ls /Volumes/lpl/User
```

or you can copy the data file 'mydata' from the /User directory on the LI-6400XT to the existing destination directory /Users/jon/test on your computer:

```
cp /Volumes/lpl/User/mydata /Users/jon/test
```

Connecting with RS-232

The LI-6400 is shipped with a communications cable (part number 9975-016) that has a 9-pin female D connector on each end. Included in that package is a 9- to 25-pin adapter (part number 392-5688).

The LI-6400 is configured with a male 9-pin AT connector on the console. Plug either end of the 9975-016 cable into the console, and the other end into the serial port on your computer. If your PC has a 25-pin male RS-232 connector, use the 9- to 25-pin adapter (Figure 11-27). Better still, get a new computer.

If your computer has USB ports instead of a serial port, then you must use a USB/Serial adapter. USB/Serial adapters are available from LI-COR (part number 6400-27) as well as other sources (e.g. www.keyspan.com).
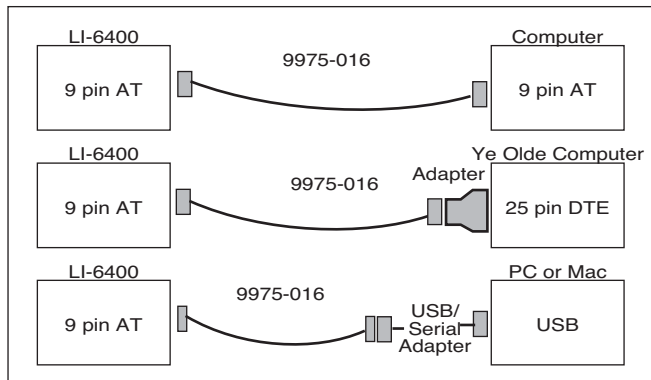
Figure 11-27. Typical cable connections using the 9975-016 cable, with 9- to25-pin adapter.

A schematic of the 9975-016 cable is shown in Figure 24-4 on page 24-30.

Configuring the Comm Port

OPEN's Utility Menu has two entries useful for RS-232 interfacing: "`Con-figure the COMM port`" and "`File Exchange Mode`".


```
╔═Utility Menu══════════════════╗
║ ►Files                        ║
║ ▼Communications               ║
║   Network Status...           ║
║   Configure the COMM port...  ║
║   File Exchange Mode...        ║
║   Connect to li6400.licor.com...║
║ ▼/►    ◆        ◆        ◆ EXIT  ◆ SELECT║
```

See **Transferring Files with RS-232** on page 11-28.

Using LI6400XTerm on page 11-30.

See **Echo to COMM** on page 9-18.

```
        Comm Port Configuration

LTerm:  off
Echo logged data to Comm:    No
Incoming Comm:  Ignore

Config: 9600 8 1 N
   Edit
```

Options are:
1) Ignore,
2) Execute as LPL (see **Control via LPL Commands** on page 11-44),
3) Parse for UCOMM(). See **UCOMM()** on page 15-27,
4) Prompts and Remarks. See **Data for Prompts and Remarks** on page 11-56.

Configure the baud rate, bits per second, etc. You can ignore this when using LI6400XTerm or LI6400FileEx, since it is taken care of for you.

Figure 11-28. Most RS-232 tasks begin in the Utility Menu.

Editing the configuration string will bring up the following prompt (Figure 11-29):

```
              Comm  Port  Configuration
       Baud  Data  Stop  Parity
L  9600  8  1  N
E
Incoming  Comm:  Ignore

Config:  9600  8  1  N
 DelLn    ◆ClrEnd   ◆DelChar◆CapLock◆AnyChar
```
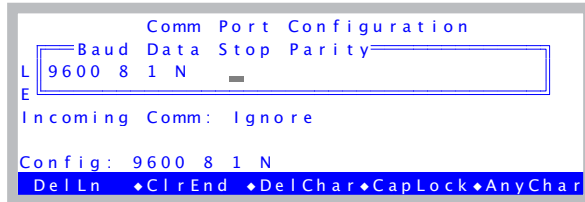
Figure 11-29. The RS-232 configuration prompt.

Table 11-1 shows the acceptable values of these parameters, and Table 11-2 illustrates some sample configurations.

Table 11-1. Communication configuration parameters.

Parameter	Acceptable Values
Baud Rate	300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
Data Bits	7 or 8
Stop Bits	1 or 2
Parity	Odd (O), Even (E), or None (N)
Handshaking	X (for XON/XOFF only), H (Hardware only), or HX (XON/XOFF and Hardware handshaking)

Table 11-2. Sample configurations

Entry	Meaning
19200 8 1 N	19200 baud, 8 data bits, 1 stop bit, no parity
115200 8 1 N H	115299 baud, 8 data bits, 1 stop bit, no parity, hardware handshaking
9600 7 1 E HX	9600 baud, 7 data bits, 1 stop bit, even parity, hardware and software handshaking

Transferring Files with RS-232

There are three options: LI6400XTerm, LI6400FileEx, or a generic data capture program.

Option 1: Use LI6400XTerm

The remote control program provides a method for transferring files. See **Using LI6400XTerm** on page 11-30.

Option 2: Use LI6400FileEx

LI6400FileEx is a (Windows only) program that interacts with the LI-6400 while it is in File Exchange Mode.

There are two ways to get the LI-6400 into File Exchange mode (Figure 11-30).
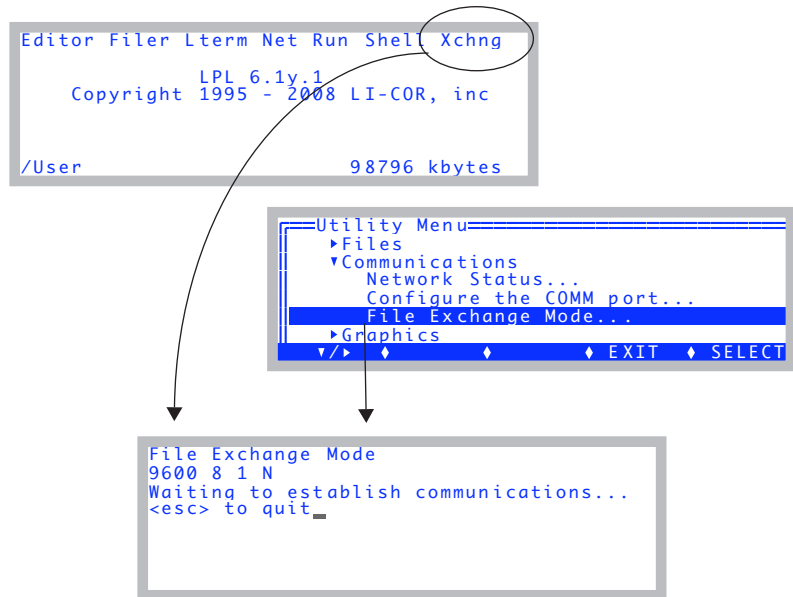
Figure 11-30. The two methods to enter File Exchange mode for RS-232 file transfer.

Note: When LTerm (**Using LI6400XTerm** on page 11-30) is active, File Exchange Mode is unnecessary, and is disabled.

Option 3: Use a Generic Data Capture Program

Generic data capture is the last resort for file transfer via RS-232, but we mention it anyway for completeness sake. Any generic serial communications program, such as HyperTerminal for Windows, or Z Term for Macintosh, can be used to capture LI-6400 *text* files. This method has three basic steps:

1   Set the communications parameters

Described below.

2   Open a destination file

Once you have configured the LI-6400's communications parameters to match those used by your data capture program, you will need to open a file on the computer to store the data, then send the LI-6400 file(s).

3   Output the Files

To send one or more files, use the Filer. First tag the desired files (page 10-14), then press **P** to print them (page 10-17).

Text vs. Binary Files

Most files on the LI-6400 are text, but not all. The exceptions are Excel files (.xls), and files generated by capturing graphics displays. When moving files back and forth via Ethernet, you don't have to worry about whether they are binary or text. When moving files using RS-232 and the FileExchange program, or some generic data capture software, this is a concern. The FileExchange program gives you an option for specifying text or binary, or letting the software figure it out. Generic data capture programs may assume it's text. The FileExchange program will modify end of line character(s) of text files based on the source and destination. It will not try to do that for binary files.

Remote Control

There are several methods to remotely control an LI-6400:

- **Terminal Software (complete control)**

  The computer acts like a terminal to the LI-6400, providing all of the normal front-panel capability locally or even over the internet. The connection can be via Ethernet or RS-232. The instrument can continue to be used from the front panel, as well as from the computer. This is described in **Using LI6400XTerm** below, and **Using LI6400Group** on page 11-34.

- **Send LPL Commands (very limited control)**

  The LI-6400 can be made to compile and execute incoming LPL commands via RS-232 while OPEN is running. This provides a method for an external

device to have as much or as little control as is desired. For example, a computer controlled imaging system could provide the LI-6400 with leaf areas, and trigger logging events. This method is described in **Control via LPL Commands** on page 11-44.

Using LI6400XTerm

The program LI6400XTerm allows you to connect to any LI-6400XT on your local area network (Ethernet) to passively monitor and/or actively control. Further, a single LI-6400XT can support any number of such connections simultaneously. The LI-6400XT does not need to be in any special mode of operation - you can connect or disconnect at will.

The program can also connect with RS-232, and this allows it to connect to any LI-6400 that has version 5.3 software or above. The RS-232 connection requires that the LI-6400 have LTerm active (press **L** from OPEN's main screen).

1   **Launch the program**
(Windows) Double click the shortcut icon on the desktop, or find the program under Start menu | All Programs | LI-6400_Apps.

(Mac OS X) The program will be in the /Applications/LI-6400Apps/ folder.

(Linux) The program is in the Applications | Education menu.

2   Connect to an LI-6400
Pick the host name, and click the Connect button.
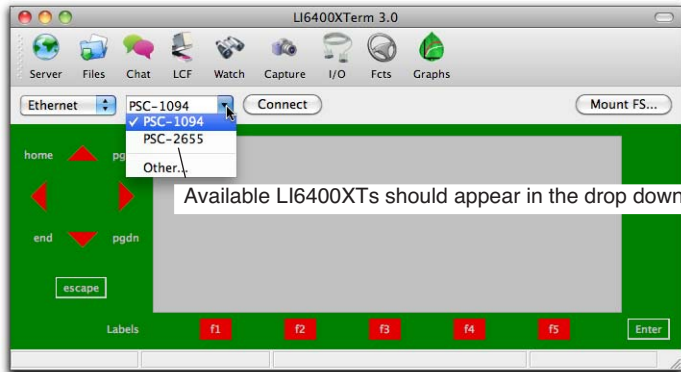
or, select "Other..." and
enter the IP address
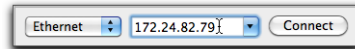
If connecting via RS-232, the drop down will show the port selection...

... on Mac OS X
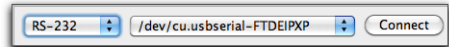
... on Windows
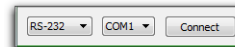
... on Linux
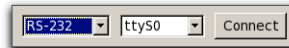
Figure 11-31. Connecting with LI-6400XTerm.

3   The display is duplicated on your PC
Once connected, the display will show exactly what is on the instrument's display. The buttons around the simulated display will behave just like on the instrument (Figure 11-32).
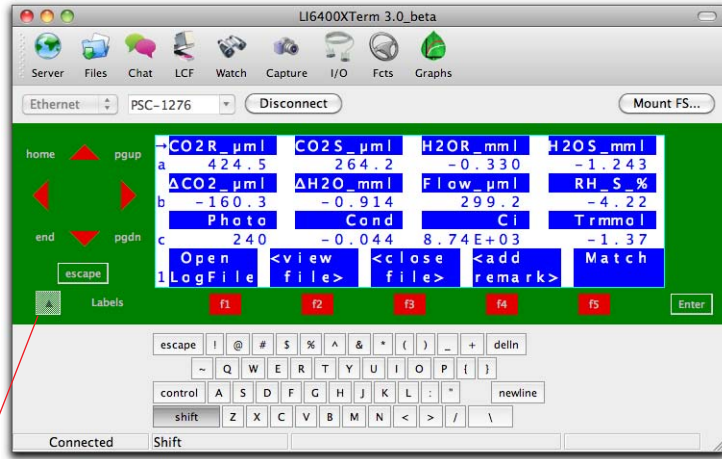
Hide / Show keyboard

Figure 11-32. The LI6400XTerm main screen.

4   Function Keys

LI6400XTerm can display all available function keys as active buttons.
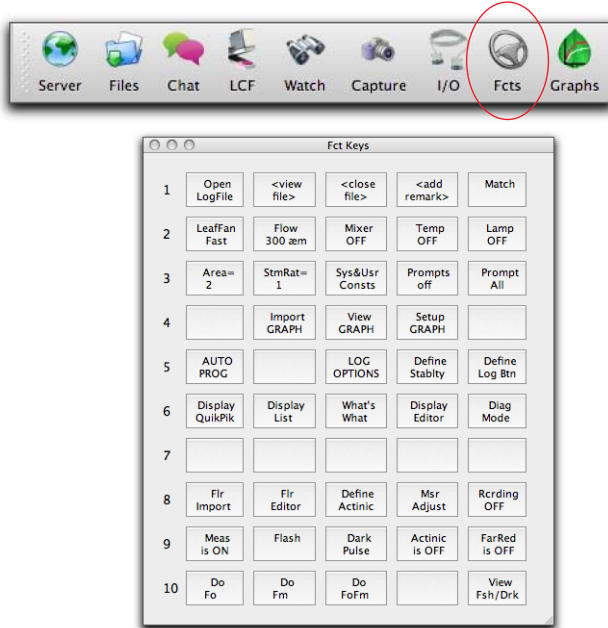
Figure 11-33. The Fct Keys window will display the current state of the function keys as active buttons. Thus, in New Measurements mode when configured for a fluorometer, there are 10 levels.

5 **File Transfers**
Use drag and drop to move files between the computer and the LI-6400 (Figure 11-34).
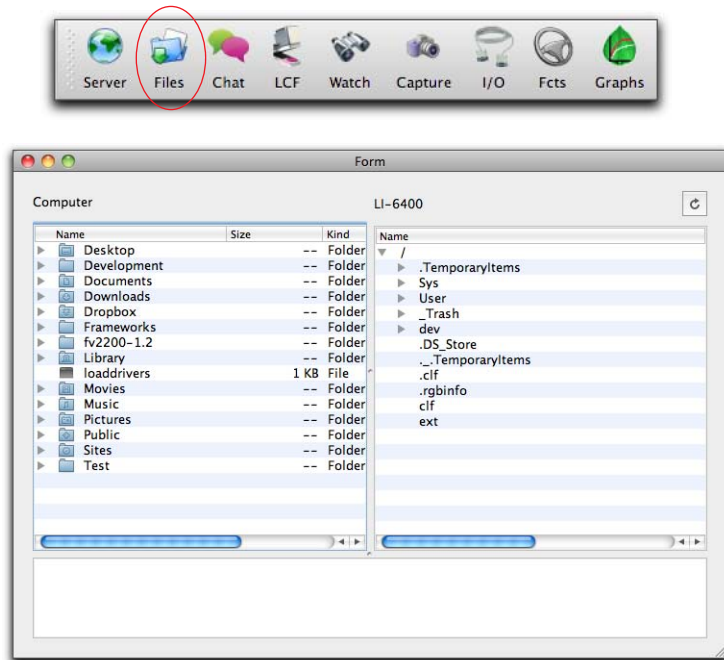
Figure 11-34. The file exchange window in LI6400XTerm.

Using LI6400Group

LI6400Group is similar to LI6400XTerm, except it allows you to monitor/control multiple LI-6400s. A user guide is available under Help in the menu bar.

1 **Launch the program**
(Windows) Double click the shortcut icon on the desktop, or find the program under Start menu | All Programs | LI-6400_Apps.

(Mac OS X) The program will be in the /Applications/LI-6400Apps/ folder.

(Linux) The program is in the Applications | Education menu

2   Add a connection
Click the add connections button to add as many LI-6400 connections (Ethernet or RS-232) as you wish.

Click the Add tool button to add a connection.
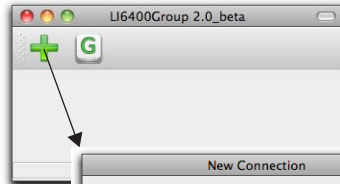
Select the LI-6400 to add.
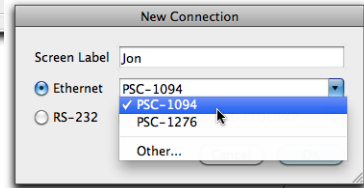
By default, you are "listening" only. Click the header bar to open/close keyboard communication with the LI-6400.
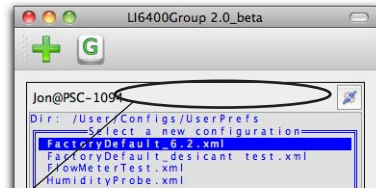
You can use your keyboard and/or Group's on-screen keyboard.
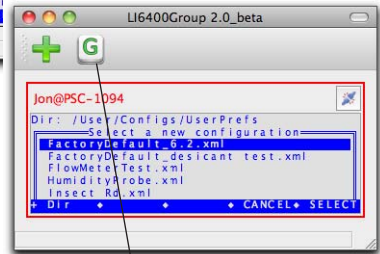
Figure 11-35. Adding a connection to the LI6400Group program.

At this point, it is just like the LI6400XTerm program. If you type with the main window active, those keystrokes will be sent to the instrument if the control connection is enabled (red box around display). The difference is you can add more connections.

3   **Add another connection**

Click the add button and add another connection. The window will resize to accommodate it. (In the preferences dialog, you can select the number of columns to allow.)
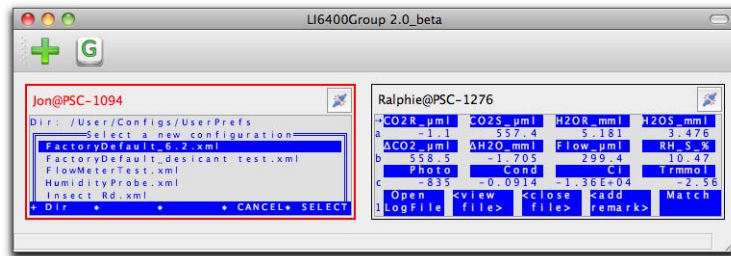
Figure 11-36. LI6400Group allows multiple instruments to be monitored and/or controlled.

4   **Simultaneous control**

Click on a display header bar to enable/disable it accepting key strokes (when enabled, the entire display will be bordered in red).
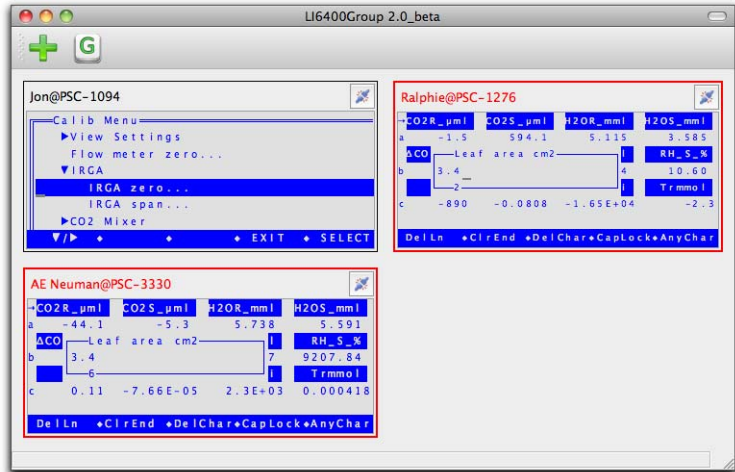
Figure 11-37. Setting leaf area on two units simultaneously.

5   **Disconnecting**

Click a display's disconnect button to stop communications. You can click again to re-establish (with the same or another instrument). An inactive display can be removed by clicking the remove button.
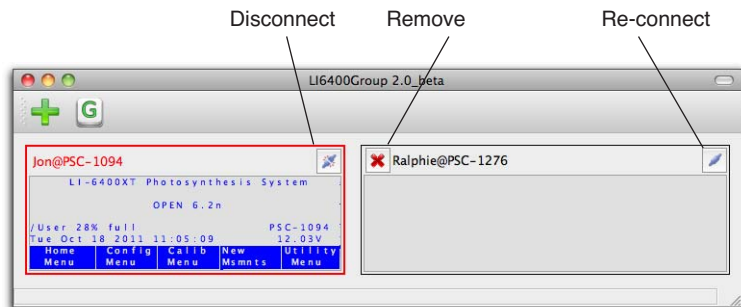
Figure 11-38. Cleanup operations.

Connection over the Internet

There is a method for connecting to and controlling an LI-6400XT via the internet, using the server li6400.licor.com. Usually this is something LI-COR uses for remote diagnostics, but you can use it as well without any interference from us. Using our server, you can remotely access and control any (cooperating) LI-6400XT anywhere in the world. You can do it using LI6400XTerm (on Windows, Linux, or Mac), or an iOS device (iPhone, iPod Touch, iPad). You can also do this with an older (non-XT) LI-6400, but that requires a computer running LI6400XTerm to be attached via RS-232 to the LI-6400, since it cannot access the server on its own.

What li6400.licor.com does

Our server looks for connection requests. Each request has two identifiers bundled into it: a remote id, and a local id. If the request looks legitimate, the server will accept the connection, and then look through its list of pending connections to find one with a corresponding pair of ids. For example, if I request a connection with local id = abc, and remote id = 123, the server will try and connect me to any waiting connection that has local id = 123, and remote id = abc. If there are no matching connections, it will keep me around for a while[3] to see if one comes along.
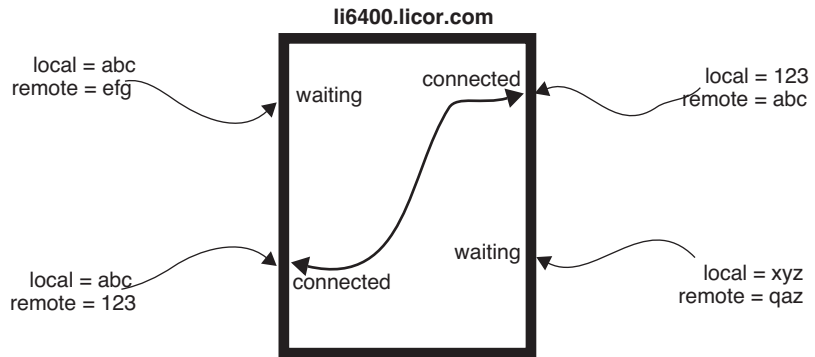
Figure 11-39. Illustration of remote and client id use by li6400.licor.com

■ **Connecting an LI-6400XT to li6400.licor.com.**
Connect the console to an Ethernet cable with Internet access. (You can check Network Status in the Utility Menu to make sure the LI-6400XT sees the local

---

[3.]The client software is written to keep reconnecting when connections time out.

network at least). Then select "Connect to li6400.licor.com" in the Utility Menu.

```
┌─Utility Menu══════════════════════════════┐
│   ▼Communications                          │
│        Network  Status...                  │
│        Configure  the  COMM  port...       │
│        File  Exchange  Mode...             │
│        Connect  to  li6400.licor.com...    │
│   ▶Graphics                                │
│   ▼/▶     ◆          ◆          ◆ EXIT  ◆ SELECT│
└────────────────────────────────────────────┘
```

```
┌──────────────────────────────────────────┐
│ Connect  to  LI6400.licor.com            │
│                                           │
│ L)ocal  id:  psc1094                     │
│ R)emote  id:  rmt                        │
│ C)onnect                                  │
│                                           │
│ Choose  L/R/C/T:_                        │
└──────────────────────────────────────────┘
```
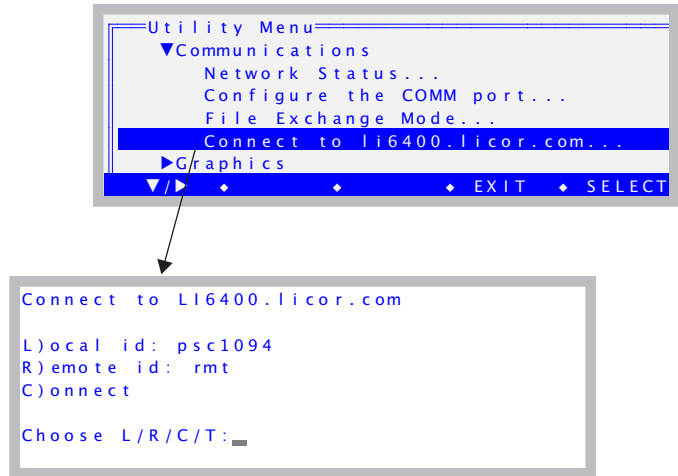
Figure 11-40. The dialog for connecting to li6400.licor.com.

The connection will require a local and a remote id. The default local id is based on the host name (with "illegal" characters removed), but you can make either id anything you want by pressing **L** or **R**. Just use letters and numbers only.

You can test to see if the LI-6400XT can get to the server by pressing **T**. The resulting screen (Figure 11-41) will show failure or success.

```
┌──────────────────────────────────────────┐
│ Querying  server...                       │
│ Failed  to  open  server.  Press  Any  Key_│
│        ┌──────────────────────────────────┐
│        │ connections:                      │
│        │ packets:  26171680  91775        │
│        │ uptime:  2117:28:33              │
│        │                                   │
│        │                                   │
│        │ Press  Any  Key_                 │
└────────┴──────────────────────────────────┘
```
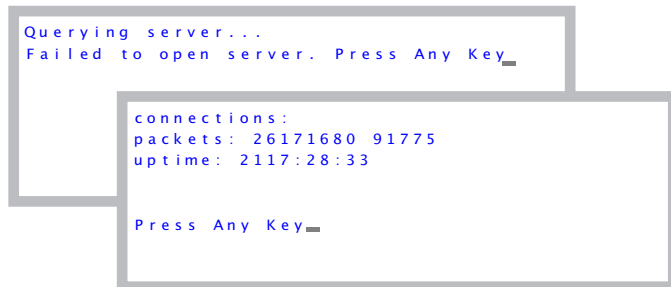
Figure 11-41. The failure and success screens when testing the connection to li6400.licor.com.

When you are ready to connect, press **C**.

When a successful connection is made to the server, you will see

```
Connected to LI6400.licor.com

Local id: psc1094
Remote id: rmt
Status: Waiting

Press D) to disconnect, or
any other key to exit
```

Figure 11-42. Connected to the server, but waiting for the other party.

"Waiting" means the instrument is connected to li6400.licor.com, and is wait-
ing for a remote connection from someone else. At this point you can press
any key to exit the screen, and use the LI-6400XT normally. Whenever you
return to this screen (or even if you stay there), it will show its current status,
which will be one of three conditions: disconnected (Figure 11-40), waiting
(Figure 11-42), or talking (Figure 11-43).

```
Connected to LI6400.licor.com

Local id: psc1094
Remote id: rmt
Status: Talking

Press D) to disconnect, or
any other key to exit
```

Figure 11-43. Completed connection to a controlling device via the server.

■ **Connecting LI6400XTerm (Ver 3) to li6400.licor.com.**
To connect a computer running LI6400XTerm to LI-COR's server, click on the Server button (or select Windows | Server from the menu bar). Then enter the IDs into the connection dialog (Figure 11-44).
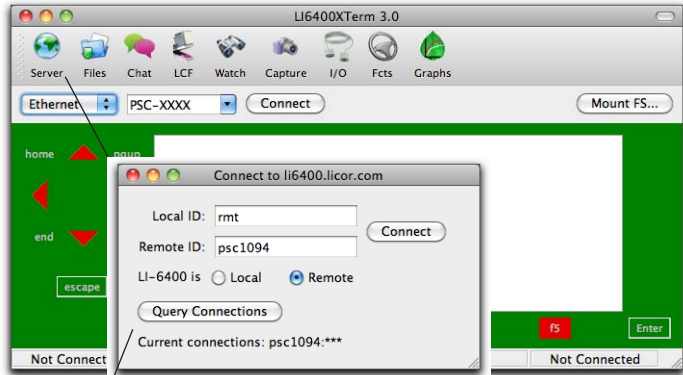
Use the Query Connections button to see
what Remote IDs the server is waiting for.

Figure 11-44. The Connection dialog in LI6400XTerm.

When you click the Connect button, you should see the remote LI-6400's display appear on the screen.
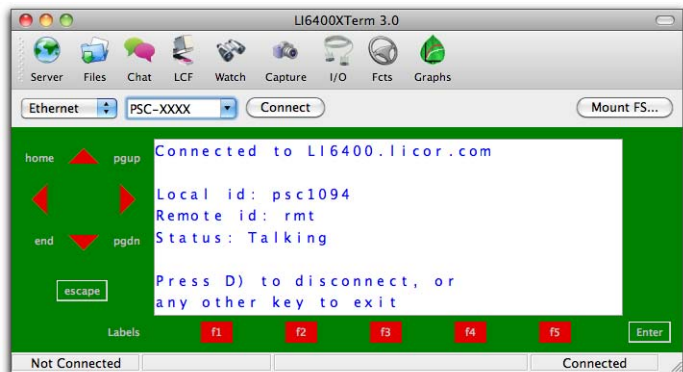
Figure 11-45. LI6400XTerm connected via the LI-COR server.

You can then operate the instrument remotely. To disconnect, click on the Server button, and then click Disconnect in the Connections box.

■ **Connecting an iOS device to li6400.licor.com**
Launch the LI6400Term app, and select li6400.licor.com from the menu
(Figure 11-46).
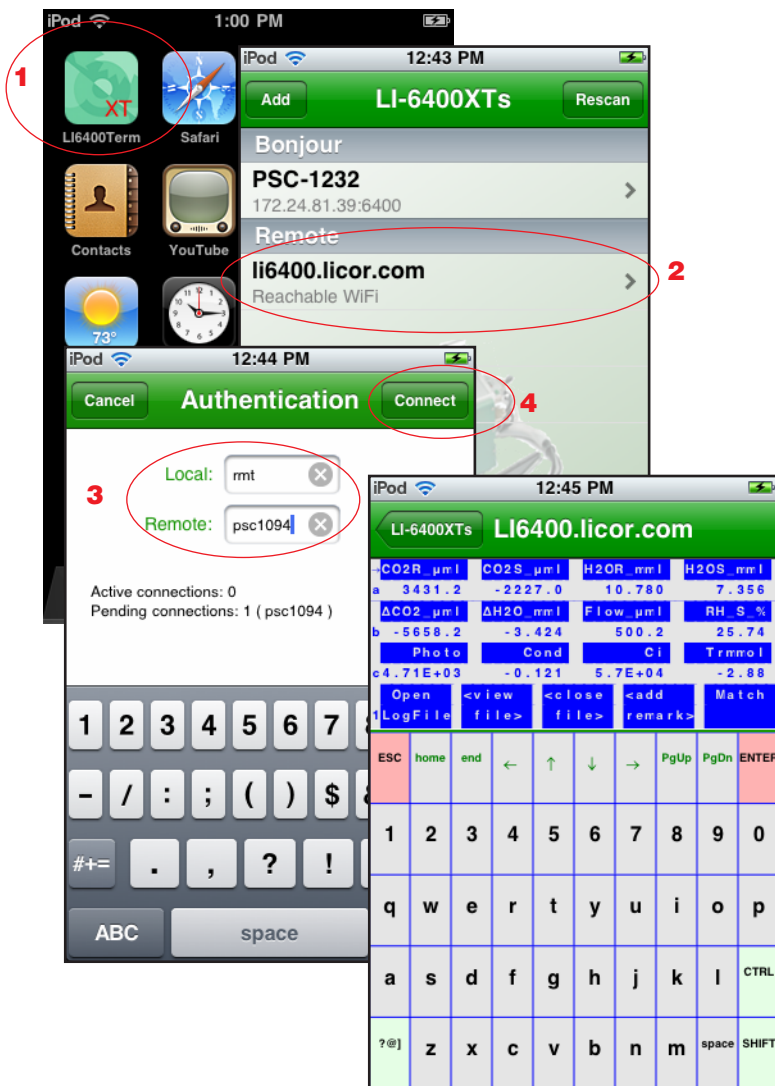
Figure 11-46. Connecting to li6400.licor.com with an iOS device.

■ **Connecting a non-Ethernet LI-6400 (ver 5 or above) to li6400.licor.com**
This will require a computer running LI6400XTerm (Ver 3) next to the
LI-6400.

1 **Connect the computer and LI-6400 with an RS-232 cable.**
See Figure 11-27 on page 11-25.

2 **Configure LI6400XTerm**
Configure the program for using RS-232 and the proper port (Figure 11-47).
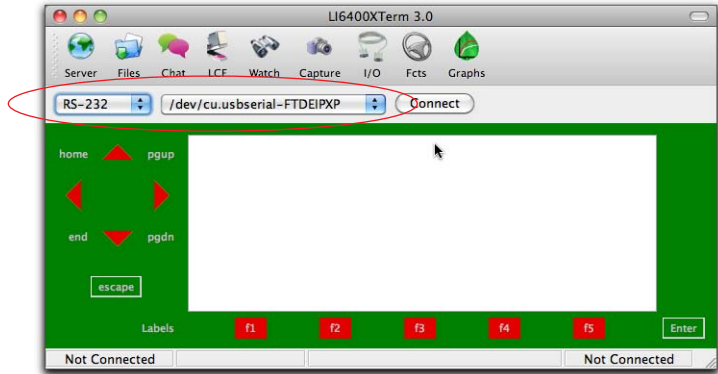
Figure 11-47. LI6400XTerm configured for RD-232 communications.

3 **LTerm on the LI-6400**
Put the LI-6400 into LTerm mode by pressing **L** from Open's main screen.
The LTerm label should be visible on the display
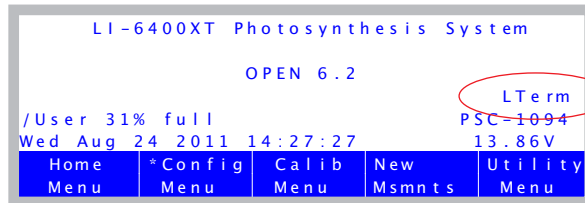
*Figure 11-48. LTerm is toggled by pressing **L**. When active, LTerm allows remote monitoring and control via the RS-232 port.*

4 **Start Talking**
Press the connect button in LI6400XTerm's window, and you should be connected.
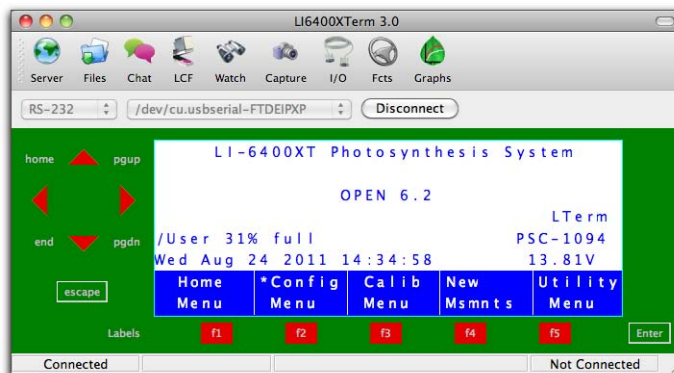
5 **Connect the Computer to li6400.licor.com**
Click on the Server icon (or select Server from the Windows menu bar entry), and fill in the IDs. Make sure the Local radio button is selected.
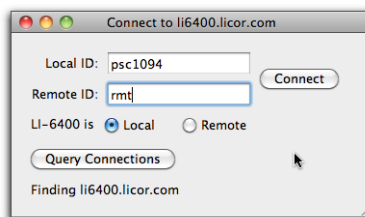
Press Connect, and your LI-6400 (through your computer) will be available to the outside world via li6400.licor.com server.

Control via LPL Commands

Suppose you have a data logger, or a data logging computer, that you want to coordinate somehow with the LI-6400. For example, you may want to get an occasional value of CO_2 or photosynthesis, or have the computer tell the instrument to start or stop an AutoProgram.

One of the fields in the Comm Port configuration lets you enable a mode whereby you can send and receive data while the LI-6400 is otherwise going about its business (Figure 11-49).

From the Utility Menu...

```
╔═Utility Menu═══════════════════════╗
║    ▸Files                          ║
║    ▾Communications                 ║
║       Network Status...            ║
║       Configure the COMM port...   ║
║       File Excha                   ║
║    ▸Graphics       Comm Port Configuration    ║
║ ▾/▸    ♦      ♦                                ║
        ║  LTerm: off                              ║
        ║  Echo logged data to Comm:  No           ║
        ║  Incoming Comm: Execute as LPL           ║
        ║                                          ║
        ║  Config: 9600 8 1 N                      ║
        ║   Edit                            OK     ║
        ╚══════════════════════════════════════════╝
```
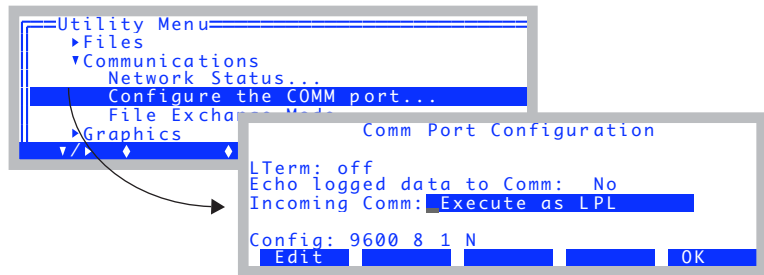
...or from Config Menu -> View Config...

```
╔═Test2.xml══════════════════════════╗
║  ┌▸a2d                             ║
║  ┌▾* comm                          ║
║  ├ config= 9600 8 1 N              ║
║  ├ lterm= Don't care (it's off)    ║
║  └ * incoming= Execute as LPL      ║
║  ┌▸matching                        ║
║ ▾/▸    ♦Edit    ♦      ♦Undo   ♦Done ║
╚════════════════════════════════════╝
```
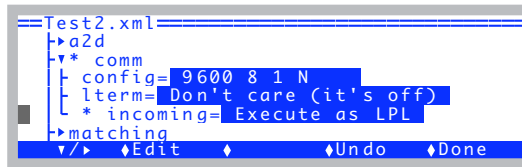
Figure 11-49. Configuring for handling incoming LPL commands.

Note: Changes to "Incoming Comm" mode do not take effect until you return to OPEN's Main Screen.

The incoming data should consist of LPL commands. LPL is the language that all of the OPEN programming is written in (refer to Part VI, **Programming**), so this option provides great flexibility. While "Execute as LPL" is enabled, every time the LI-6400 receives a newline character (decimal 10), it attempts to compile and execute the data that has been received since the last newline. Thus, if you sent the following line

```
"Hello from the LI-6400\n" comm print
```

followed by a newline, the string

```
Hello from the LI-6400
```

would be sent out the LI-6400's Comm port, followed by a newline character. Notice, we embedded a newline character into the output string by slash-n (\n).

Getting The Values of Variables

Many of the variables of interest in OPEN have ID numbers associated with them. (See Chapters 14 and 15 for a discussion of system and user defined variables.) A very useful way of getting any of these variables sent out the Comm port in a nicely formatted fashion is to use the function *idout*, which expects a destination and an ID number on the stack. For example,

```
30 comm idout
```

will get the following in return:

```
Photo= 12.34
```

The function *idout* prints the log format label for that value, followed by a '=', followed by the present value of the variable, formatted just as for a log file, followed by a newline character. You can get multiple values by putting an array of id numbers on the stack:

```
:INT { 30 -1 -2 -4 -5} comm idout
```

will result in

```
Photo= 12.34
CO2R= 378.1
CO2S= 372.3
H2OR= 12.34
H2OS= 20.45
```

If you know the variable name of something (see Table 14-10 on page 14-23 lists system variable names and ID numbers, but this applies to any public variable in OPEN), you can create an output line formatted however you choose. For example, the variable name that holds leaf area is *area_cm2*. Thus, if you want the value to be digits with no other label, you can send

```
area_cm2 "%1.5f\n" comm print
```

to receive

```
1.23456
```

The variable name for user defined items is usually *unnn*, where *nnn* is the ID number. Thus, photosynthesis is *u30*, and transpiration is *u20*.

Setting the Values of Variables

You have to be a little careful here, since many of the user defined variables are recomputed once or twice a second, so setting a value won't accomplish very much. For example, you could tell the LI-6400 to make the value of photosynthesis 20.00, but it would only stay that way a fraction of a second until it was computed again. User defined constants and remarks are a different matter, since they are strictly determined by outside influences - i.e., a user typing on the keyboard. Control set points are also a different matter, since many of them have functions for getting and setting, as we'll see in **Control Set Point Variables** on page 11-48.

In general, variables are set in LPL by the following statement:

```
<value> &<variable name> =
```

The first item pushed onto the stack is the desired value. Next comes the address of the variable to be set, then finally an '=' sign. For example

```
2.34 &area_cm2 =
```

will set the area to 2.34. The & before *area_cm2* is important. That tells the compiler to put the address of that value on the stack, rather than the value of the variable. (Note: setting area in this way will *not* by itself change the **f1** level 3 function key label, but it *does* change the area.)

There is a handy function for converting variable ID numbers to addresses: *FmtGetVarAddr*. Thus, you can set any system or user variable using only the ID number. For example, leaf area (area_cm2) is -33, so

```
2.34 -33 FmtGetVarAddr =
```

would accomplish the same thing as

```
2.34 &area_cm2 =
```

and

```
2.34 30 FmtGetVarAddr =
```

is the same as

```
2.34 &u30 =
```

which (pointlessly) sets the photosynthetic rate.

Control Set Point Variables

Information about LED Source control, flow control, temperature control, and humidity control can be obtained and changed remotely. See **LED Source Control** on page 25-22 and subsequent sections for a number of useful functions, designed for use with AutoPrograms, but serve nicely for remote control as well. For example, you can use *LampGetTarget* in the sequence

```
LampGetTarget "Type=%d, Val=%f\n" comm PRINT
```

to learn

```
Type=2, Val=2000
```

which means the lamp is in constant PAR control (Type=2) with a set point (Val=) of 2000 μmol m^{-2} s^{-1}.

To set a control and setpoint for the lamp, you can use *LampSetTarget* or *LampSetNewTarget*. The former just changes the target, and the latter changes both the control type and the target. For example,

```
1500 3 LampsetNewTarget
```

changes an LED light source to constant control signal (3) mode, with a target of 1500 mV. If the device is an LCF, it sets it to fixed blue mode, with a target (red + blue) of 1500. This function does not output anything, but if you want some sort of verification sent out the comm port, you could add it yourself:

```
1500 3 LampsetNewTarget "Set Lamp!\n" comm PRINT
```

or

```
1500 3 LampsetNewTarget LampGetTarget "Type=%d,
Val=%f\n" comm PRINT
```

The command lines can be as long as you like, as long as no newline characters are present until the end.

LCF Control

Refer to **Programming Commands** on page 27-86, which describes commands such as

```
DoFsFmp
```

which will trigger the measurement of F_s and F'_m.

Variables and Commands for Logging

Any of the LP_ commands (Chapter 25), such as *LPLog*, can be used with the "Execute as LPL" option. Thus, if a log file is open, an external device can force a data record to be logged by sending

```
LPLog
```

If you want to send a string to be added to a log file as a remark, do this:

```
"The sky is falling!" LogTSRemark
```

Testing Commands

It is wise to test the commands you will be using, before you commit them to code in your data logger or computer program. A convenient way to do this testing is with LI6400XTerm or LI6400Sim (below) or Telnet (next section).

- **Use LI6400XTerm and an LI-6400**

    Establish the connection (**Using LI6400XTerm** on page 11-30). Then, enable the incoming LPL option (Figure 11-49 on page 11-45) and return to OPEN's main screen. Finally, open the Comm Monitor window. Enter commands in the upper window, see results in the lower (Figure 11-50).
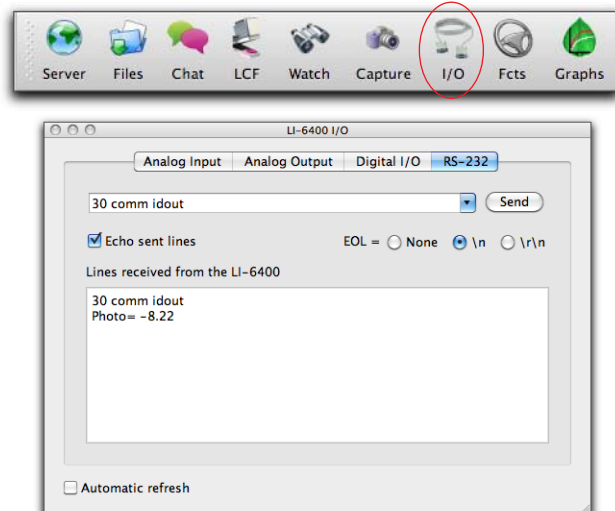
Figure 11-50. Using the Comm Monitor window of LI6400Term to test LPL commands.

- **Use LI6400Sim by itself**

   Using the Windows simulator, enable the incoming LPL option (shown in Figure 11-49 on page 11-45) and return to the OPEN's main screen. Then, open the Control window, then click on the Comm tab. Type in the upper window, see results in the lower (Figure 11-51).
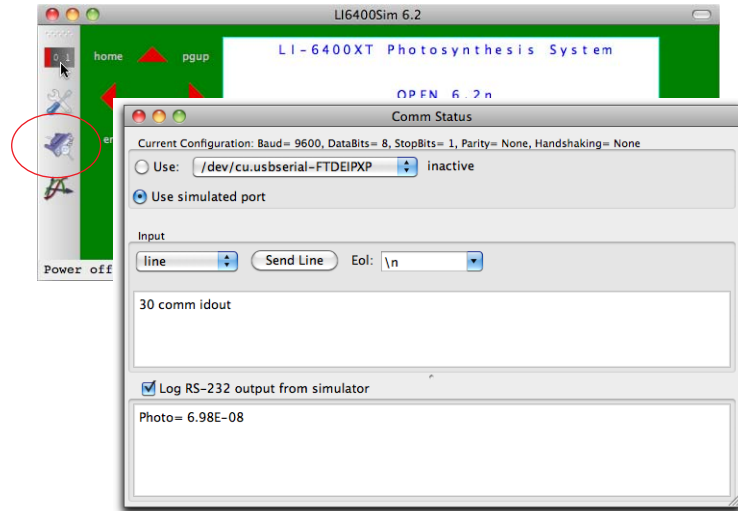
Figure 11-51. The Comm window of LI6400Sim.

- **Use Telnet**

   With your PC and the LI-6400XT on the same LAN, determine the IP address of the LI-6400XT, and launch Telnet from a command line prompt on your PC. Be sure to specify port 6409.

   ```
   $ telnet 172.24.81.168 6409
   ```

   A successful connection results in a message like

   ```
   Trying 172.24.81.168...
   Connected to 172.24.81.168.
   Escape character is '^]'.
   ```

   To send a command, simply type it on the PC, followed by return. In this mode (Telnet using port 6409), RS-232 output form the LI-6400XT is sent to your PC, and anything you type on the PC is sent to the XT.

   To exit Telnet, type **ctrl +]**, then **q**.
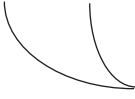
Getting Data From Serial Devices

The LI-6400 can be configured to collect input from an RS-232 device (such as GPS, bar code reader, etc.). There are two basic use cases: getting data for real time measurements, and getting data once in a while when answering a prompt, or adding a remark.

Data for Real Time Measurements

Suppose you have some sort of instrument that outputs data on RS-232 that you'd like to feed into the LI-6400. Suppose that each record that it output had a time stamp, a temperature, and a relative humidity, so a short piece of the data stream might look like what is shown in Figure 11-52.

```
      Item #1    Item #2   Item #3

      10:15:42  24.23    45.66
      10:15:43  24.24    45.65
      10:15:44  24.22    45.65


                          Delimiter should be space(s), tab,
                          or comma.
```

Figure 11-52. A sampling of data records from a fictitious instrument. We are interested in capturing the 2nd and 3rd items.

For each item you'd like to pick off from this data stream, you need to define a variable. We'll add two: one for the temperature we'll call *xTair*, and one for humidity we'll call *xRH*. Once that's done, we need to configure the Comm port for this.

■   **Adding xTair and xRH**

1   **Access the "extras" node in the configuration tree**
In the configuration tree (`Config Menu | View/edit`), find the <open>
<comps> <file> <extras> node, and edit it (Figure 11-53).
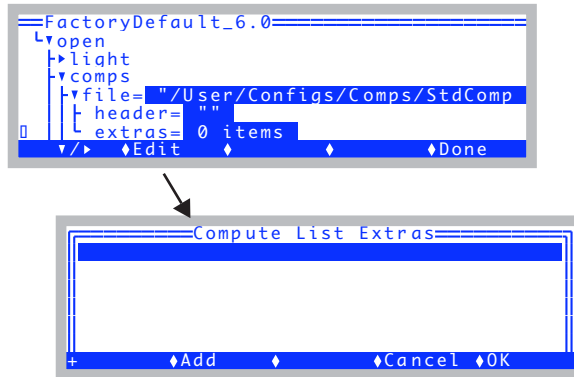
Figure 11-53. Accessing the Extras list.

2   **Add a numeric comm port item.**
Press **f2** (**Add**), and you will be prompted to identify the type of item you wish
to add. Pick Comm port item - numeric (Figure 11-54).
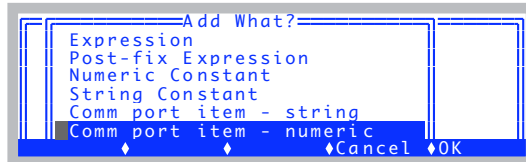
*Figure 11-54. There are two comm port items. We want
the numeric one.*

3   **Fill in the details**
You will be prompted for a label, a description, and what position in the incoming RS-232 record to look for the value (Figure 11-55).

You can adjust anything you like here. Note we made the format a fixed point, with 2 digits to the right of the decimal.

Figure 11-55. Adding an extra item.

4   **Finish it**
Press **f5** (**OK**).

5   **Now add an RH variable**
Repeat the process to add a second item. Call it xRH. It will be the 3rd item in the incoming data stream.

Figure 11-56. The setup for the RH variable.

After you OK it, your extras list should look like (Figure 11-57).

```
════════════Extras List Editor════════════
1000: xTair (external temp)
 1001: xRH (external RH)



Edit    ♦Add      ♦Remove  ♦Cancel  ♦OK
```
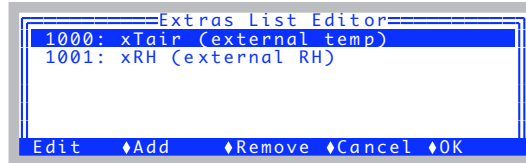
Figure 11-57. Both items successfully added.

6   **OK the List**
Press **f5** (**OK**) to keep the list. You will be asked if you want to add these items to the New Measurements display, and to the log list. You do, so press **Y** in response to each question (Figure 11-58).

Adding the new items to the display and log lists.

```
════════════Extras List Editor════════════
Add xTair (ID#1000) to Display (Y/N)?Y

Add xTair (ID#1000) to LogList (Y/N)?Y

Add xRH (ID#1001) to Display (Y/N)?Y
Add xRH (ID#1001) to LogList (Y/N)?█
```
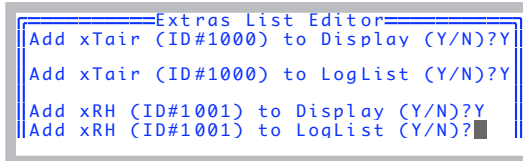
If you add things to the display, you are told where they can be found.

```
Display Additions...
xTair -> line m
xRH -> line m
Press any key
```

Figure 11-58. When you exit the Extras List Editor, you are given a chance to add new additions to the New Measurements display, and to the Log List.

■   **Configuring the Comm Port**
To make this work, we need to configure the comm port, in two regards: 1) the baud rate, data bits, etc., and 2) Tell OPEN to parse incoming data. This latter item will happen automatically, since you have a user item that requires it.

1 **Access the Comm Config node**

In the configuration tree, find the <open> <comm> node (Figure 11-59).


```
═══FactoryDefault_6.0════════════════════
 ┣▸a2d
 ┣▾* comm
 ┃ ┣ config= 9600 8 1 N
 ┃ ┣ lterm= Don't care (it's off)
 ┃ ┗ * incoming= Parse for UCOMM()
 ┣▸matching
   ▾/▸   ♦Edit   ♦         ♦Undo    ♦Done
```

```
═╒═════Select Incoming COMM option═════
  │ Ignore
  │ Execute as LPL
  │ Parse for UCOMM()
  │ Prompts and Remarks
  │
       ♦        ♦          ♦Cancel ♦Select
```
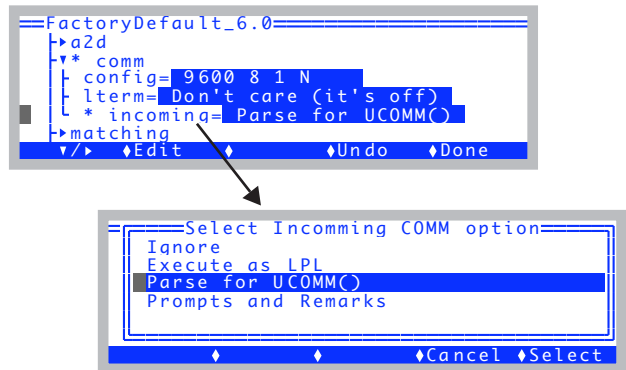
Figure 11-59. Configuring the comm port. Make sure the baud rate, etc. is what you need, and also that the <incoming> node is set for parsing.

Some Rules to Remember

Incoming data records are parsed according to the following rules:

1 **Records end with a newline character**

This character has a decimal value of 10.

2 **Delimiters are space, tab, and comma**

This record:

```
123,45.67     89
```

would be considered to have three items available; Item #1 is "123", and item #2 is "45.67", and item #3 is 89 (note there are multiple spaces between items 2 and 3).

3 **Double quotes override delimiter characters**

The record

```
10.56 "123,45.67",The End
```

would be considered to have four items: Item#1 is "10.56", item #2 is "123,45.67", item #3 is "The", and item #4 is "End". If an item is quoted, the quotes are stripped from the returned string.

4   String and Numeric
Lines are parsed as strings. If you are specifically looking for a numeric item, the string piece that is parsed has one more step to convert it to a floating point value. If the string is not a valid number, the value will be 0.

Data for Prompts and Remarks

Suppose you have a bar code reader, and want to be able to scan a label associated with each measurement site right before the measurement. The destination for this information might be the remarks field in a file, or a constant that you are prompted for prior to each measurement.

All you have to do to make this happen is configure the comm port to match your (RS-232) bar code reader, and set the incoming comm node in the configuration to "Prompts and Remarks".

Method 1: The Config Menu. (Select "View/Edit Config...")

```
══FactoryDefault_6.0═══════════════════
 ├►a2d
 ├▼* comm
 │ ├ config= 9600 8 1 N
 │ ├ lterm= Don't care (it's off)
 │ └ * incoming= Prompts and Remarks
 ├►matching
 ▼/►    ♦Edit    ♦        ♦Undo    ♦Done
```
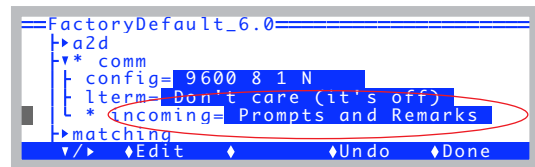
Method 2: The Utility Menu

```
══Utility Menu═══════════════════════
  ►Files
  ▼Communications
    Network Status...
    Configure the COMM port...
    File Exchange Mode
```

```
          Comm Port Configuration

LTerm: off
Echo logged data to Comm:  No
Incoming Comm: Prompts and Remarks

Config: 9600 8 1 N
 Edit                            OK
```
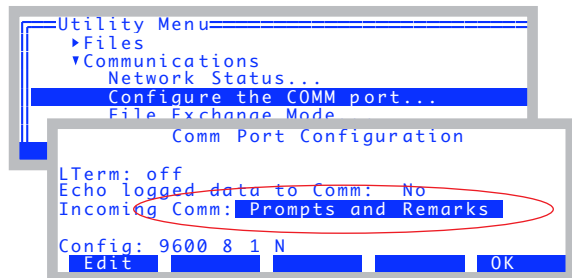
Figure 11-60. Two places you can configure for prompts and remarks: either the configuration tree, or else in the Comm Port Configuration screen from the Utility Menu.

GraphIt

12

A tool for viewing and graphing data files

12

GraphIt

GraphIt is a utility included with the LI-6400 programming that allows you to view data in labeled columns, plot selected variables, and do polynomial curve fitting.

An introductory tour of GraphIt can be found in **Viewing Stored Data** on page 3-65.

Accessing GraphIt

There are several ways to access GraphIt:

- **From the Filer**
  Highlight the file to be viewed, and press **H**.

- **From OPEN's New Measurements mode**
  GraphIt can be used on the file being logged by pressing **ViewFile** (**f2** level 1) while logging to memory or a file.

- **From OPEN's Utility Menu**
  The menu selection "Graph a data file…" will run the GraphIt Utility program described in the next paragraph.

- **Run the program GraphIt Utility**
  It's stored in the /Sys/Utility directory. The program uses the Standard File Dialog to prompt you to select a file, then runs GraphIt on that file.

- **From the data recompute program**
  The "Recompute a stored data file…" program in OPEN's Utility Menu uses allows you to view source and destination files using GraphIt.

When GraphIt Starts

Before GraphIt's main screen (Figure 12-1 on page 12-3) appears, the program examines the data file to find a label line (the line that is used to provide names for the data columns). Acceptable data file formats are discussed on page 12-4. If there's a problem with your data file format, you'll be asked to pick the label line (Figure 12-9 on page 12-11). In the absence of a label line, arbitrary names will be given to data columns.

GraphIt's View Control:

```
View Options:
F - File as stored
H - Header
D - Data set (all vars and obs)
G - Graphed vars & obs only
C - curvefit Coefficients
E - curvefit Errors
```

GraphIt's Editor:

```
Edit which item:
X - X (horizontal axis)
Y - Y (left vertical) axis
Z - Z (right vertical) axis
L - Logic for including observations
C - Curves
```

Target File Name

Current Plot Definition (if any)

observations. (Appears only after data has been read and graphed)

```
                    GraphIt
Source= Experiment 1
X axis: 0,*,"Ci";
Y axis: 0,*,"Photo";

                            (7 Obs)
  Import   REPLOT   View    Edit     Export
1 GrafDef  GRAPH    Data    GrafDef  GrafDef
  Pick     REPLOT            Store    Load
2 DataSet  GRAPH             Image    Image
```

Change to a previously defined and stored plot definition

Draw axes, points, and curves using current plot definition (appears only if there is a plot defined)

Store the current plot definition, for later retrieval with QuickPik. (Appears only if there is a plot defined)

Select Data Set:

```
Data Set Options:
P - Pick Label, and 1st, last Data lines
S - Subsample data (use every __ line)
```

Graphics Image Control: storing and retrieving pictures

Figure 12-1. GraphIt's main screen shows the name of the data file being used, and the current plot definition (if any). There are 2 levels of function keys defined, and one can toggle between them by pressing **labels** *or* **1** *or* **2**. *Descriptions of GraphIt's functionality begin with* **Defining Plots** *on page 12-5.*

Data File Format

GraphIt expects its target data file to have a particular structure. This structure is not hard to come by - OPEN's default data structure fits this scheme, as will nearly any spreadsheet-ready ASCII (text) data file.

- **Rows = Observations**
  Lines in the file need to consist of observations of variables, and one row (line) is one observation. A single observation of all the variables should be on one line.

- **Columns = Variables**
  Within a line, the values for each variable can be separated by spaces, or tabs, or any non-numeric character, such as comma, semicolon, etc.

- **A label line**
  There needs to be a line in the file that lists names of the data columns. If each name is quoted, they need not be delimited, but they can be with spaces and/or commas. If each name is not quoted, then they need a space or comma separating them, and (obviously) no spaces within a label.

- **Data lines need sufficient data**
  Any line that doesn't parse into enough values will be ignored when scanning for data. Quoted items are considered 1 value.

Figure 12-2 illustrates some acceptable data file formats.

OK
```
"Header info"
"This is skipped"
"The label line is next"
"Obs", "Photo", "PAR", "Tleaf", "Tair"
1,14.2,1001,23.4,22.3
2,12.2,1.1E3,23.5,22.4
3,13.8,2E3,22.4,22.7
"This is a remark"
4,18.3,1098,23.7,22.9
5,11.1,1008,22.4,22.6
6,17.7,1023,23.3,22.2
```

OK
```
Photo PAR TleafTair
14.2  1001 23.4 22.3
12.2 1031 23.5 22.4
13.8 1065 22.4 22.7
18.3 1098 23.7 22.9
11.1 1008 22.4 22.6
17.7 1023 23.3 22.2
```

Problem
```
Photo: 14.2 12.2 13.8 18.3 11.1 17.7
PAR: 1001 1031 1065 1098 1008 1023
Tleaf: 23.4 23.5 22.4 23.7 22.4 23.3
Tair: 22.3 22.4 22.7 22.9 22.6 22.2
```

Figure 12-2. Illustration of the data file format expected by GraphIt.

Finding the Label Line

When GraphIt first runs, it looks for the label line. It does this by looking for an identifier string

    $STARTOFDATA$

and expects the column labels to be the next line. If this target is not found, you will be asked to identify the column label line using Standard Menu (Figure 12-9 on page 12-11).

Once the label line is identified, GraphIt reads the labels; they are used to identify data columns.

Finding the Data Lines

The data set is assumed to extend from the first data line following the label line, to the first blank line or the end of the file. If this is not the case, you can specifically identify the label line, first data line, and last data line (described in **Selecting Observations** on page 12-10).

Once the label line is identified and read, GraphIt's main screen appears (Figure 12-1).

Defining Plots

A plot definition consists at a minimum of:

* **An X axis definition**
  Minimum and maximum scaling values, and variable name.

* **A Y or Z axis definition**
  Minimum and maximum scaling values, and from one to five variable names (Figure 12-3).
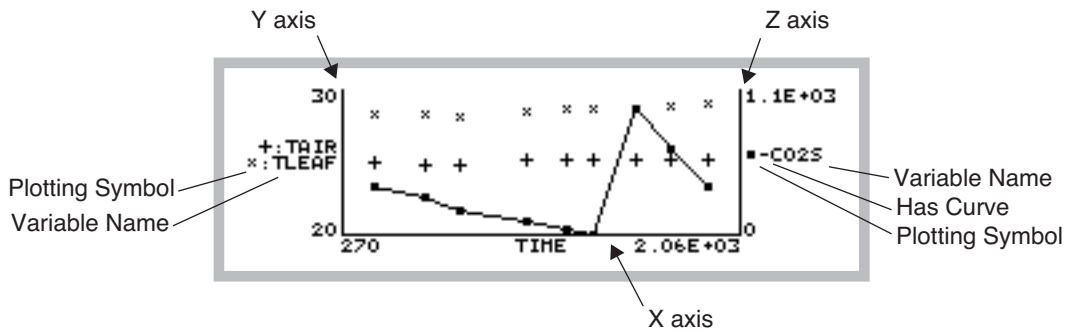
Figure 12-3. A typical GraphIt plot. One or two vertical axes can be used, and up to five variables can be plotted on each axis. Curve fitting options consist of dot-to-dot (as shown), or polynomials.

There are two methods of defining a plot: the quick method is to select a previously defined and stored definition from a menu (press **Import GrafDef**). The general method is to use the plot configuration editor (press **Edit GrafDef**).

Using Import GrafDef

This is quite simple, as the following example illustrates.

■ **To configure for an A-Ci curve using Import GrafDef:**

1 **Press Import GrafDef**
The directory /User/Configs/PlotDefs is accessed using the Standard File Dialog, and you can select from the definitions stored there.

2 **Select "A-Ci Curve" from the menu**
Figure 12-4 illustrates the resulting plot definition and plot (provided, of course, that the data were actually an A-Ci measurement!).

Plot Definition

X axis: 0,*,"Ci";
Y axis: 0,*,"Photo";

Y Axis max

Variable

Plot symbol

Y Axis min

26.4

+:PHOTO

0

0   CI   832

X Axis min   X variable   X axis max
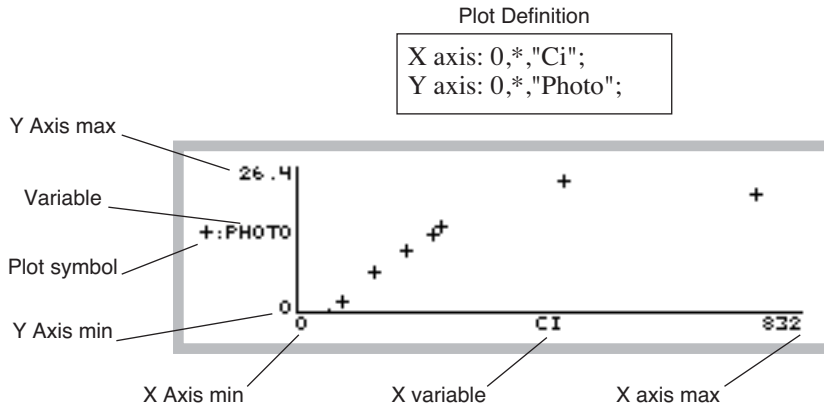
Figure 12-4. A plot using an A-Ci plot definition. A * for either min or max scaling will result in that min or max value being computed based on the data.

Using Edit GrafDef

To define a graph, or to modify the current configuration, press **Edit GrafDef**. The edit menu will appear:

This option only appears if there is currently a valid plot definition

```
Edit which item:
X - X (horizontal) axis
Y - Y (left vertical) axis
Z - Z (right vertical) axis
L - Logic for including observations
C - Curves

Press choice or <esc>
```

Figure 12-5. GraphIt editor's main window. Note: you can by-pass this window by typing **X***,* **Y***,* **Z***,* **L***, or* **C** *directly from GraphIt's main screen.*

Editing Axis Definitions

Pressing **X**, **Y**, or **Z** will allow you to edit the definition of that axis. The display will show:

Min   Max   Label

```
X axis: 0,*,"Ci":        ┌─────────────────────┐
V – change variable      │ Axis being edited, and
R – change range         │ the current definition
                         └─────────────────────┘
Select option (or <esc>) ┌──────────┐
                         │ Options  │
                         └──────────┘
```
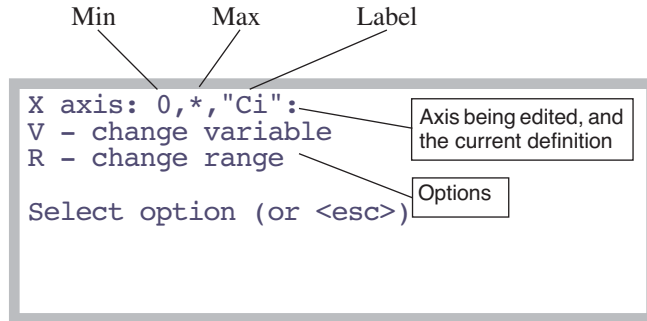
Figure 12-6. Editing an axis has two parts that can be done independently: selecting the variable(s), and selecting the scaling.

Press **V** to change the variable(s) to be plotted on that axis, and **R** to change the min and/or max scaling values. An asterisk (*) for either the min or max values means that the value will be determined based on the data set (autoscaling).

- **V: Change Variable**
Selecting variables is done via a Standard Menu (Figure 12-7 on page 12-9) using the list of variables read from the label line. Only one variable is needed for the X axis, but up to five can be drawn on either the Y or Z axes (you'll want all the variables for an axis to be of the same magnitude - temperatures, for example - because they are all plotted with a common scaling).

> **Curve Setup Short-Cut**
> A shortcut is provided for setting up any Y or Z variables for drawing curves instead of just plotting points: if you want a curve for a variable, instead of selecting it by pressing **enter**, press a number (**0** through **5**). **0** will cause the points to be connected by straight line segments, and **1** through **5** specifies a polynomial of that power.

Selecting the X variable

Pick 1 variable.

═══════X Axis═══════
"Obs" 1
"HHMMSS" 2
"FTime" 3
"Photo" 4
"Cond" 5
"Ci" 6
"Trmmol" 7

> The X variable is selected from a menu of all the data columns in the data file.

> The variable count for that axis. Up to five can be selected.

> The Y or Z variable(s) are selected from the same menu. A shortcut is provided for drawing curves: press a number rather than enter to make a selection. Also, up to five entries can be selected. Press escape after you have selected all the ones you want.

Selecting Y or Z variables

Pick up to 5 vars:
Press ENTER for each
(or '0' - dot2dot
or 1 to 5 for poly fit)
Press <esc> to stop.

┌Y Axis plot #1┐
"Obs" 1
"HHMMSS" 2
"FTime" 3
"Photo" 4
"Cond" 5
"Ci" 6
"Trmmol" 7

Figure 12-7. Selecting the variable(s) for an axis is done via Standard Menu. Y and Z axes allow up to five variables to be included, and provide a short-cut for doing curves. When you've selected all the variables you want, press **escape***.*

- • **R: Change Range**

  Pressing **R** (Figure 12-6 on page 12-8) will allow the max and min to be specified for the axis being edited. If available, the range of the data will be displayed for your information.

```
Data range is 27.1 to 769
Enter MIN (* to autoscale)
0
Enter MAX
*
```
DelLn ◆ClrEnd ◆DelChar◆CapLock◆AnyChar

Figure 12-8. An axis's min and max are prompted for in sequence, with the default value or response displayed.

If you enter * (or any non-numeric entry, for that matter) for either the max or min value, that parameter will be computed for you, based on the variables defined for that axis.

> **Note:** The axes max and min aren't the actual data values max and min. The axes are scaled a bit larger to encompass the plotting symbols.

Storing Plot Definitions

The **Export GrafDef** function key (**f5** level 1) allows you to store the current plot definition for later recall using the **Import GrafDef** function key (**f1** level 1). Definitions should be stored in the directory /User/Configs/PlotDefs. GraphIt prompts you for the directory and file name using the Standard File Dialog.

Selecting Observations

GraphIt provides three tools for selecting data subsets for plotting or viewing. In order of priority of implementation, they are:

1   First and Last Data Lines

Data is found by searching for non-blank lines whose first character is not a quote ("), starting with the First Data Line. The search continues until a blank line is found, or until the Last Data Line has been searched.

2   Use Every n[th] Line

Between the First and Last Data Lines, every n^{th} line is converted to numbers, provided it does not start with a quote ("). The default value of n is 1, but the user can set this. This is useful for subsampling large data files.

3   Inclusion Based On Data Values

Once a line is converted to numbers, any variable(s) or combination thereof can be tested to see if that observation should be included.

Picking the First and Last Observation

The starting and ending data lines can be manually selected. Press "**Pick DataSet**" (**f1** level 2), then **P**. You will be prompted to select three lines in succession using Standard Menu: the label line, the first data line, and the ending data line (Figure 12-9).

```
═ Select the LABEL LINE ═══════
"LightSource=","Sun+Sky",1,0.19
"LogFormat=","/User/Configs/LogFormats/
"1: grass"
"Time","Photo","Cond","Ci","Trmmol","A
20.2,15.4,0.216,238,3.49,1.62,3.5,1,3.49
104.2,11.3,0.202,180,3.33,1.65,3.5,1,3.4
  Print ◆ Find ◆ Refind◆ CANCEL◆ SELECT
```
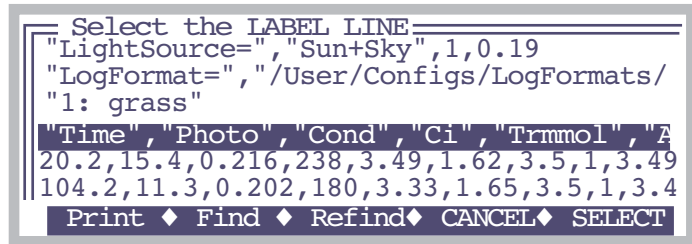
Figure 12-9. Prompting for the Label Line. Put the highlighted bar on the label line, and press **enter**. *When selecting the data lines, the prompts will be "Select the FIRST DATA LINE" and "Select the LAST DATA LINE"*

Using Every nth Observation

If your data represents a long time series with many observations, it will be advantageous to subsample the data when plotting it. For example, instead of plotting each observation, you may wish to plot every 10th observation.

To define a subsample interval, press "**Pick Data Set**" (**f1** level 2), then **S**. You will be prompted to enter a value *n*, where *n*=1 means use every line, *n*=2 means use every other line, etc.

Logic Based Inclusion

Sometimes you may wish to include or exclude observations based on the values of the data for that observation. For example, suppose you have a file of survey data for sunlit and shaded leaves, and you wish to plot photosynthesis vs. conductance for sunlit leaves only. You could do this with GraphIt's logic option by specifying only observations with PAR greater than some threshold value, such as $1000 \ \mu$mol m^{-2} s^{-1}.

Inclusion logic is defined by pressing "**Edit GrafDef**" (**f4** level 1) then **L** from GraphIt's main screen (short cut: just press **L** from the main screen). The logic screen (Figure 12-10) appears, and you can type in or edit the logical statement to be used.

The keyboard entry
line is inverse video

```
Logic for selecting observations.
Examples:
"Obs" > 10
("Obs" > 10) AND ("Photo" < 5.2)

_

+ DelLn  ◆ClrEnd  ◆DelChar◆CapLock◆AnyChar
```
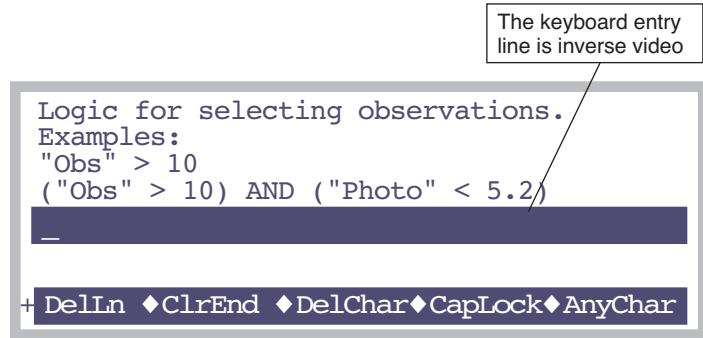
Figure 12-10. Entering selection logic. Put data set variable names in quotes.

For example

```
"Obs" > 10
```

will include only those lines for which the Obs value is greater than 10. More complicated statements are possible with appropriate use of parentheses:

```
("Obs" > 10) AND ("Photo" < 5.2)
```

includes all lines of data for which Obs is greater than 10, and Photo is less than 5.2. Note that variable names are always quoted and must match a variable (column label) in the data set.

You can also use the actual variable names. System variables are listed on page 14-23. For user defined items, you can use *unnn*, where *nnn* is the ID number. Thus, photosynthesis is *u30*.

```
("Obs" > 10) AND (u30 < 5.2)
```

Table 1 lists some useful symbols for use in logic statements.

Table 9-1. LPL keywords for logical expressions.

LPL	Action
+ - * /	Add, subtract, multiply, divide
< >	Less than, greater than

Table 9-1. LPL keywords for logical expressions.

LPL	Action
<= >=	Less than or equal, greater than or equal
<> ==	Not equal, is equal
AND OR	Logical and, logical or

For this logical expression, spaces don't matter. For example

```
("CO2S" / "Ci") > 2
```

is the same as

```
("CO2S"/"Ci")>2
```

Also note that the variables (if any) that appear in the logical statement don't have to be the ones being plotted. The above example includes C_a/C_i ratios of greater than 2, but we might be plotting something else entirely. It also illustrates that the logic can include quantities that aren't already computed in the data set, since the computation can be done in the logical statement itself.

The logical expression must evaluate to a number. If that number is non-zero, it is assumed true, and that observation is included. A null logical expression (in the screen shown in Figure 12-10 on page 12-12, press **DelLn** then **enter**) means that all observations are included. A logical expression that would accomplish the same thing would be simply

```
1
```

while one that would exclude all observations (not very useful) would be

```
0
```

Dealing With Strings

When GraphIt scans lines in the file for data, any line that does not have sufficient data is skipped. Note that each quoted string constitutes one data "value"[1]. Also, "sufficient data" depends on which variables you are plotting. If you are plotting the 1st, 3rd, and 4th variables in the list, then four values are

---

[1] The value is 0 if there are no numbers in it. If the string is quoted time, such as "14:22:45", the value will be decimal hours (14.3792)

sufficient data. Therefore, be aware that if your data set includes a labels line, then it will likely be considered as data.

Curve Fitting

GraphIt's curve fitting capability is fairly limited: you have a choice of straight line segments between data points (dot-to-dot), or polynomial fits of power 1 to 5.

Configuring for Curves

There are two ways to implement curves:

- **When first selecting variables**
  This is explained in **Curve Setup Short-Cut** on page 12-8.

- **For an existing configuration**
  From GraphIt's main screen, press **Edit GrafDef**, then **C** (short cut: just press **C**). If only one Y or Z variable is defined, the curve is simply selected from a menu (Figure 12-11).


```
Y axis, 1st variable      The variable        Current curve choice

Y1 "Photo" none                    ═══Curves═══
                                   -1:none
                                    0:Dot-to-Dot
                                    1:StaightLine
                                    2:2ndOrderPoly
                                    3:3rdOrderPoly
                                    4:4thOrderPoly
                                    5:5thOrderPoly
```
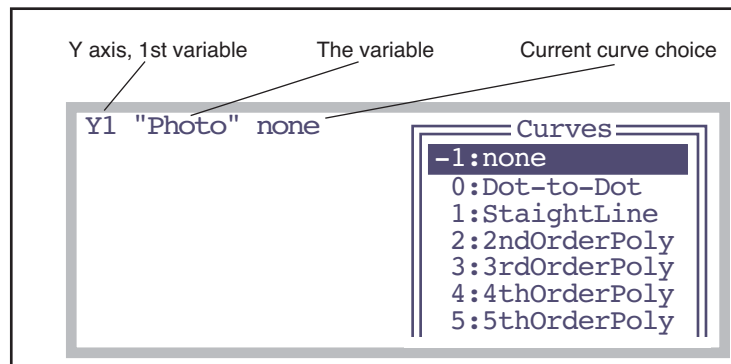
Figure 12-11. Selecting a curve option when only one Y or Z variable.

If multiple Y and/or Z variables are defined, then it is a two part process (Figure 12-12).

a) Select the variable

```
╔════════ Curve Status ═══════╗
║ Y1 "Photo" none             ║
║ Z1 "Tair" none              ║
║ Z2 "Tleaf" none             ║
║                             ║
║                             ║
╠═════════════════════════════╣
║ Print ♦ Find  ♦ ReFind♦ Cancel ♦ Select ║
╚═════════════════════════════╝
```

b) Select the curve

```
╔════ Curve Status ═══╗╔═══ Curves ═══╗
║ Y1 "Photo" none     ║║ -1:none      ║
║ Z1 "Tair" none      ║║  0:Dot-to-Dot║
║ Z2 "Tleaf" none     ║║  1:StaightLine║
║                     ║║  2:2ndOrderPoly║
║                     ║║  3:3rdOrderPoly║
║                     ║║  4:4thOrderPoly║
╠═════════════════════╩═══════════════╣
║ Print ♦ Find  ♦ ReFind♦ Cancel ♦ Select ║
╚═════════════════════════════════════╝
```
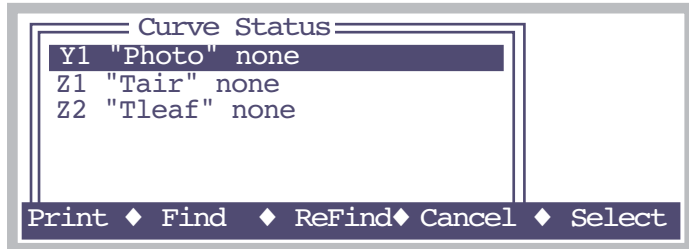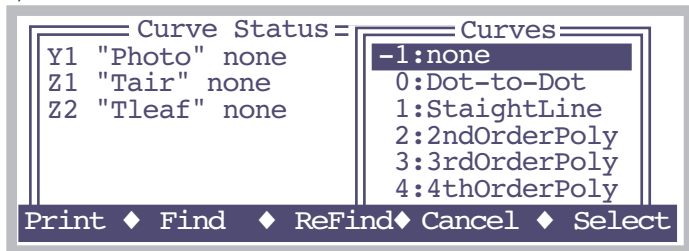
Figure 12-12. Selecting curve options for multiple variables is a two step process: a) pick the variable, and b) pick the curve.

Viewing Curve Fit Results

If one or more polynomials have been selected, the resulting coefficients can be seen by pressing **View Data** and selecting either option **C** (curve fit coefficients) or **E** (curve fit errors).

■ **Example: Find The Initial Slope Of An A-C$_i$ Curve**

1 **Define the plot**
Use **Import GrafDef** or **Edit GrafDef** to set up a plot of photosynthetic rate as a function of intercellular CO_2.

2 **Select a straight line curve**
Press **Edit GrafDef**, then **C**, and select a straight line as the curve type.

3 **Ignore the higher Ci values**
While in the editor, press **L** to select observation logic. Use an expression like

```
"Ci" < 150
```

to include the observations on the low end of the curve.

4   Plot and curve fit
Press **REPLOT GRAPH**. Press **escape** after viewing the plot.

5   View the slope
Press **View Data**, then **C**. The display will look similar to Figure 12-13.

Offset   Slope

```
══════════ CurveFit Information ══════════
Photo vs Ci coeffs:
-5.22275 0.0939082
Y Intercept = -5.22275
X Intercept = 55.6155
RMS= 0.7112
```
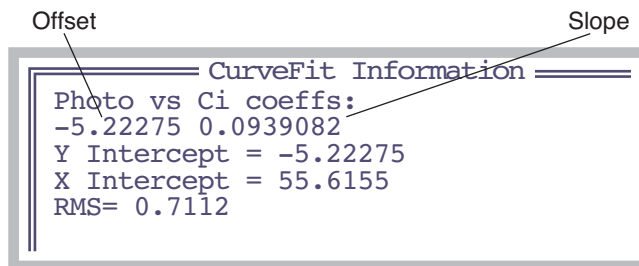
Figure 12-13. Polynomial coefficients are presented in ascending order of power. X intercept(s) are computed for 1st and 2nd order polynomials.

6   Store the info?
Press escape to stop viewing CurveFit Information. You will then be given an opportunity to store that data to a file.

```
Store curvefit info to disk? (Y/N)
```

If you press **Y**, a Standard File Dialog will be used to let you specify a destination file. If you pick an existing file, you can choose to append the curve fit information to its end.

Finding Initial Slopes
Finding the initial slope of a curve is important for light response curves and A-Ci curves. This section leads you through a step-by-step example of how to find the initial slope of either curve.

1   Use GraphIt
This is the graphing routine accessed by pressing **View File** (**f2** level 1) in New Measurements mode, or (for a file that is closed) by highlighting a file in the Filer, and pressing **H**.

2   Define the curve
Plot "Photo" vs "parIn_μm" for a light curve, or "Photo" vs "Ci" for an A-
Ci curve. Note the value of "Photo" where the curve becomes non-linear:
we'll be entering that value in the next step.

3   Limit the Data Range
In order to include only the data from the linear part of the curve, select **Edit**
GrafDef (**f4**) and **L** for logic. Type

        "Photo" < x

and press **enter**, where x is the value you determined in the last step.
Figure 12-14 shows how this looks on the display.


```
Logic for selecting observations.
Examples:
"Obs: > 10
("Obs" > 10) AND ("Photo" < 5.2)
"Photo" < 8_
```

Your entry →
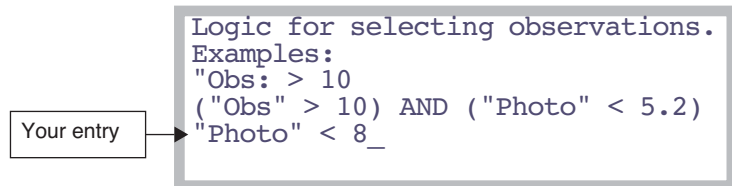
*Figure 12-14. A Ci Curve initial slope logic. Enter the maximum photosynthesis
rate on the linear part of the curve.*

4   Fit a Curve
To fit a curve to this reduced data set, press **Edit GrafDef** and select **C** for
curves. Choose "1: Straight line" from the list, and press **enter**. When you
exit the editor, the fitted line will be drawn.

5   View the slope value
Press **View Data** (**f4**) then **C** for select Curvefit Coefficients. You can save
them to a file by pressing **Y** when prompted. One suggestion is to append them
to the end of your data file.

Viewing Data

The View Data function key brings up a list of the options (Figure 12-15).

```
View Options:
F - File as stored
H - Header
D - Data set (all vars and obs)
G - Graphed vars & obs only
C - curvefit Coefficients
E - curvefit Errors
```

Figure 12-15. The screen displayed after pressing **View Data***.*

F - File as stored

This option lets you see the data file just as it exists on the disk (Figure 12-16).

```
=============/User/Data=============
"OPEN 6.1z"
"Fri Oct 10 2008 14:09:21"
<open><version>"6.1z"</version></open>
<open><light><source>"Sun+Sky"</source><
<open><comps><file>"/User/Configs/Comps/
<open><prompts><onlog>off</onlog><items>
<open><stability><items>"Std Stability"<
```

Figure 12-16. The F option. Viewing a file.

H - Header

This option lets you view the XML header information in a more convenient manner (Figure 12-17).

```
==Header from /User/Data===========
┌►open
└►li6400



  ▼/►    ♦         ♦         ♦ EXIT  ♦ SELECT
```

Figure 12-17. Viewing header information. The two main nodes are configuration and calibration information.

D - Data set (all vars and obs)

This option views the currently selected data set in column format (Figure 12-18). The observations reflect the first and last, as well as any regular skipping that you may have defined (as described in **Using Every nth**

Observation on page 12-11). It does NOT reflect logic statements (**Logic Based Inclusion** on page 12-11).

Cursor

Position Marker (left = start, right = end)

Fixed Header ⟶

TBlk	CO2R	CO2S	H2OR
23.17	399.7	365.9	15.79
22.97	299.4	279.6	17.15
23.02	199.7	187.9	17.37
23.43	100.7	96.2	15.42
23.35	49.5	49.9	15.52
23.34	−0.3	4.9	15.60

Scrollable text

Print ◆ Find ◆ ReFind◆ JumpTo◆ OK

Figure 12-18. Viewing with the **D** *option: all variables and observations of the data set. Use shift→ and shift←to scroll left and right four columns at a time. An inverse box in the fixed header indicates relative position in the data of the cursor.*

G - Graphed vars and obs only
This options shows the data plotted in the last plot. You can also see this after viewing a plot by pressing **V** instead of **escape**.

C - Curvefit Coefficients
This option shows the coefficients and RMS values for each defined curve. Figure 12-13 on page 12-16 is an example. Press **escape** when done viewing, and you are given the option of storing the information to a file:

```
Store curve fit info to disk? (Y/N)
```

E - Curvefit Errors

This option shows the coefficients, RMS values, and the data points and errors for each defined curve.

```
Photo vs Ci coeffs:
-5.22275 0.0939082

     Ci    Photo y-f(x)
======== ======== ========
     180     11.3  -0.3807
     127      7.2   0.4964
    76.8     2.16   0.1706
    51.4    -0.39 0.005872
    27.1    -2.97  -0.2922
RMS = 0.7112
```

Figure 12-19. The **E** *view option for the Initial Slope example (page 12-15).*

As with the **C** option, you are given the option of storing the information to disk when you exit from viewing it.

Measuring Graphs

GraphIt provides methods of determining points and intervals on a plot. After a plot has been drawn and you are looking at it, instead of pressing **escape** to continue on, you can bring up one of the following graphics cursors.

The XY Cursor (+)

Pressing **+** will bring up the XY cursor, a cross-hair cursor that you can move around the display by pressing the arrow keys (add **shift** for bigger steps) and see the coordinates of any location. **Escape** quits.
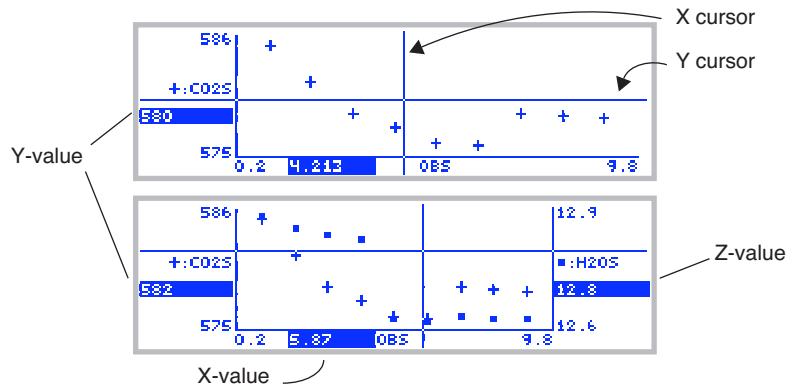
Figure 12-20. The XY cursor.

The YY Cursor (=)

Pressing **=** brings up two Y cursors (Figure 12-21), which are used to measure the interval between vertical axis locations. There are two horizontal cursors to control here, but only one is active at a time. The active cursor is marked by little boxes at its ends, and is moved by pressing ↑ or ↓ (add **shift** for bigger steps). To swap the active cursor, press → or ←. Press **escape** to exit.
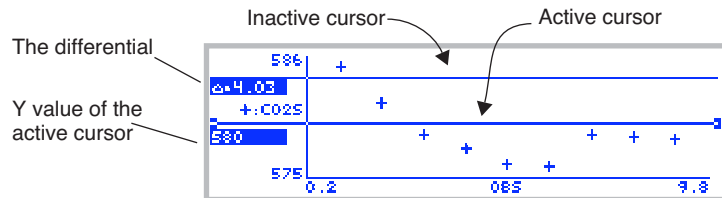
Figure 12-21. The YY cursor.

XX Cursor (‖)

Pressing **|** brings up two X cursors (Figure 12-22), which are used to measure the interval between horizontal axis locations. Just as with the YY cursor, the active cursor is marked by little boxes at its ends, and is moved by pressing → or ← (add **shift** for bigger steps). To swap the active cursor, press ↑ or ↓. Press **escape** to exit this mode.
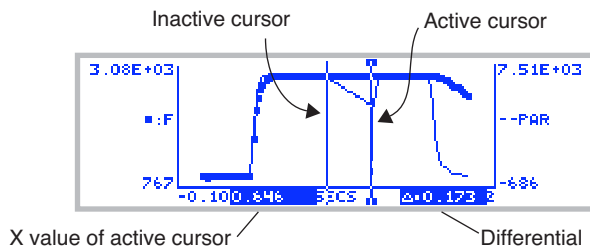
Figure 12-22. The XXcursor, showing a 170 ms phase 2 of a multiphased fluorescence event (Chapter 27).

Accuracy and Resolution

The graphics cursors' resolution is the pixel size (240 wide by 64 high). The coordinates it computes are based on the current scaling and the pixel location of the cursor. Thus, for example, if you plotted observation number against observation number, you know the points are even integers (1,1), (2,2), etc., but the graphics cursors may indicate something slightly different, such as (4, 4.019) (Figure 12-23).
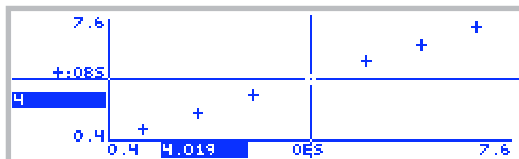
Figure 12-23. The graphics cursor accuracy is limited by the pixel resolution.

PlotDef File Format

Figure 12-24 illustrates the format of a plot definition file. These files are expected in, but not confined to, /User/Configs/PlotDefs
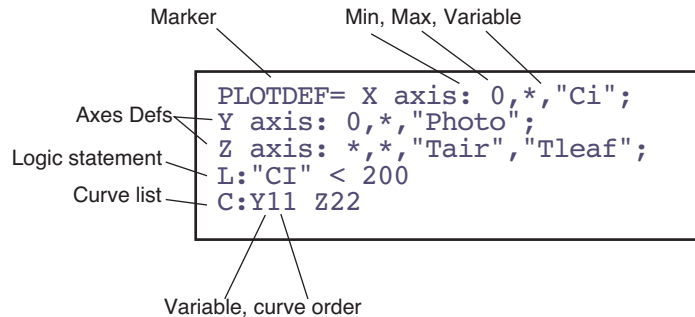
Figure 12-24. An example plot definition. All entries are optional, except the Marker (plotdef=).

Storing and Retrieving Graphics Images

Graphics images can be stored and retrieved by using the function keys **Store Image** (stores an image to the file system) and **Load Image** (reads an image from the file system). The resulting files are binary, and are not legible as text. If you transfer them to a computer using RS-232 (Chapter 11), be sure to treat them as binary files (**Text vs. Binary Files** on page 11-29).

Store Image lets you pick the destination file name and directory. There is also a short-cut method to storing graphics images, that automatically generates the destination file name. Press **ctrl + s** when viewing a GraphIt graph. A message will be displayed briefly indicating the name of the file, which is automatically generated:

```
======= Graph Saved =======
"/User/Images/QP_yyyy-mm-dd_hh-mm-ss"
```

The graphics image will be stored in a file named /User/Images/QP_*yyyy-mm-dd_hh-mm-ss*, where *yyyy-mm-dd* is the year, month, day, and *hh-mm-ss* is the hour, minute, second when you pressed **ctrl + s**.

If you wish to use these binary files on another computer (that is how the plot images were created for this manual, for example), then you will have to write a program to handle the data, and will need to know the "encoding" scheme (Figure 12-25).
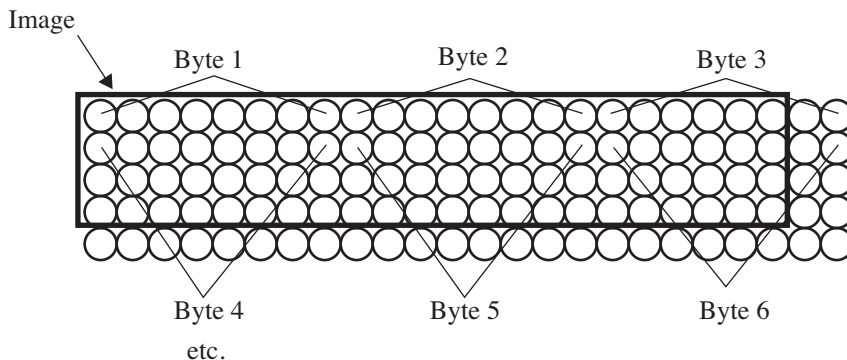
Figure 12-25. If the image border is not divisible by 8, then there will be unused portions of some of the bytes. Note that the image length and width must be preserved, or else the image will be shuffled.

Recomputing Data Files

13

How to recompute data files

13 Recomputing Data Files

Version 6 with its Excel file generation option, *almost* renders this chapter obsolete. But not quite. If you had turned off the Excel feature (whether accidentally or because of a predisposition to avoid all things Microsoft[1]) when generating a data file and you now need to recompute the data file, you can do so with the technique described in this chapter.

A Step-By-Step Example

Suppose you have a file named /User/MyData that has an incorrect leaf area (you used 6.0, and it should have been 4.4) and stomatal ratio (you used 0 and it should have been 0.5). This example shows how to make a new file named /User/MyData (RCMP) that incorporates those changes.

1   **Run the Recompute Program**
You'll find the data file (text version) recomputation program under `Utility Menu|Files` (Figure 13-1).

```
╔══Utility Menu═══════════════════════
║  ▼Files
║      Access the Filer...
║      Create a new AutoProgram...
║      Create a new (empty) file...
║      Recompute a stored data file...
║      Reinstall basic config files...
║  ▼/▶   ♦        ♦        ♦ EXIT  ♦ SELECT
```
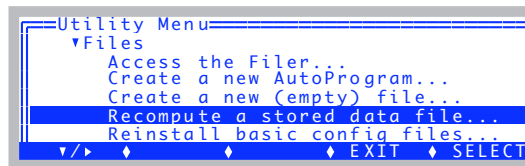
Figure 13-1. OPEN's recomputation program is found in its Utility Menu

---

[1.]A predisposition that I have great affection for.

2   **Specify the source file**

Highlight "SourceFile:", and press **f1** (**edit**) (Figure 13-2).
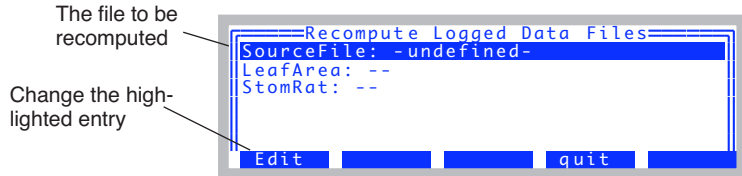
Figure 13-2. The recompute program's main screen. Use ↑ and ↓ to move the high-light bar up and down, and press edit to change the highlighted entry.

The Standard File Dialog will appear. Highlight the file "MyData", and press **enter** (Figure 13-3).
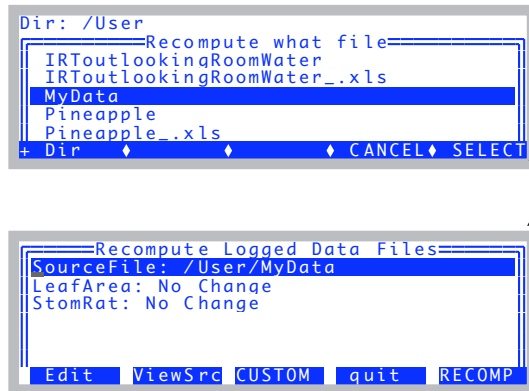
Figure 13-3. The Standard File Dialog is used to select the source file.

The **ViewSrc** key (**f2**) views the source file using GraphIt (Chapter 12). The **CUSTOM** key (**f3**) provides some flexibility in what gets recomputed, and is explained in **Customizing Recompute** on page 13-6.

Recomputing Data Files
A Step-By-Step Example

3 **Specify a change in leaf area**
Highlight the "`LeafArea:`" line, and press **Edit** (Figure 13-4).
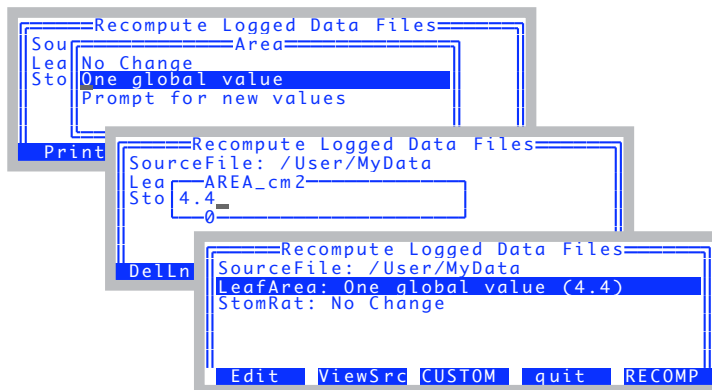
Figure 13-4. To change leaf area, change the `LeafArea:` *entry from* `No Change` *to one of the other choices.*

4 **Specify the new stomatal ratio**
Highlight `StomRat:` and press **Edit**. Select "`One global value`", and enter 0.5. The display should now look like Figure 13-5.

Figure 13-5. Ready to recompute. We'll be setting a new leaf area, and a new stomatal ratio. The recomputed data will be written to an as yet un-named file.

5   Recompute the file

Press **f5** (**RECOMP**). You will be asked to name the destination file (Figure 13-6). The default name will be the original name with (RCMP) appended to it.

```
Dir: /User
File:MyData (RCMP)_
==========Write to what file=========
 .DS_Store
 ._MyData
 CornLeaf
 CornLeafIRtest
+ Dir    ♦         ♦         ♦ CANCEL♦ SELECT
```
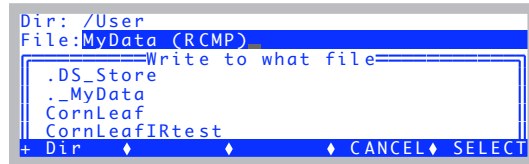
Figure 13-6. You are prompted for the destination file when you start the recomputation, if the Destination: entry is set to "Write to a file".

As the file recomputes, the display will show a dot for every observation processed:

    Recomputing.....

If an 'x' is printed instead of a dot, it means that a line in the data file has been skipped over. See **Skipping Observations** on page 13-8.

6   View the results

After the file is generated, GraphIt (Chapter 12) is called to allow you to view the new file either graphically or textually.

```
                GraphIt
Source= /User/MyData (RCMP)
(no graphs defined)



  Import           View    Edit
1 GrafDef          Data    GrafDef
```
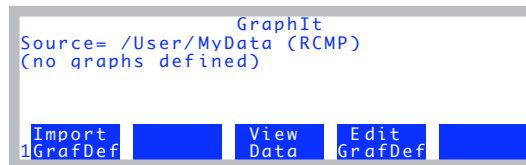
Figure 13-7. GraphIt run for the destination file.

The Details

Source File Considerations

The source data file contains a header[2] that includes all the relevant configuration information (Figure 13-8). For best results, the source file should have one header, and one set of data below it. If it has multiple headers and data sets, subsequent headers will be ignored, and if subsequent data sets differ in format, the results may be invalid.

Destination File Considerations

Destination File Header

The recompute program puts its own header information at the start of the recomputed data file, as well as a copy of the original header (Figure 13-8). If you recompute multiple times, you'll get multiple headers.

```
"OpenRecomp 6.1"
"/User/MyData"                               New Header
"Sat Nov  8 2008 13:37:04"
"*************"
"OPEN 6.1"
"Fri Nov  7 2008 12:58:14"
<open><version>"6.1ac"</version></open>
<open><configfile>"/User/Configs/UserPrefs/FactoryDefault_6.1.xml…
<open><light><source>"Sun+Sky"</source><par_in>1 <sensor>"GA-1094…
<open><comps><file>"/User/Configs/Comps/StdComps_6.1"<header>""…
<open><prompts><onlog>off</onlog><items>"Default (none)"</items>…
<open><stability><items>"Std Stability"<items[1]><id>-2 </id><size>…
<open><log><format>"StdLogFmt_6.0"<items>{ -35  -21  -36  -76   30 …
<open><a2d><avgtime>4.0 secs</avgtime><userchans><ch20>"Off"</ch20>…
<open><matching><type>0 </type><disp>'a'</disp><settings><CO2Limit0…
<li6400><factory><unit>"PSC-1094"</unit><serviced>"7 Feb 2007"</ser…
"12:58:16 "
$STARTOFDATA$
"Obs" "HHMMSS" "FTime" "EBal?" "Photo" "Cond" "Ci" "Trmmol" "VpdL" …
1 "12:58:19 "9.0 0 32 0.313 138 3.54 1.15 20.01 4.4 1.77 0.5 3.19  …
2 "12:58:27 "17.0 0 32.3 0.313 136 3.54 1.15 20.00 4.4 1.77 0.5 3.1…
3 "12:58:34 "25.0 0 32.1 0.312 137 3.54 1.15 20.01 4.4 1.77 0.5 3.1…
4 "12:58:41 "32.0 0 32.1 0.311 136 3.53 1.15 20.01 4.4 1.77 0.5 3.1…
```

Figure 13-8. An example recomputed data file.

Customizing Recompute

The function key **CUSTOM** (**f3**) allows some control over what happens to each variable. Pressing this key will generate a list, such as the one shown in

---

[2.]This header information may also be in a separate .HDR file. If it is, the program will find and use it (if that file still exists).

Figure 13-9. The list is of the variables that will be written to the recomputed file. In general, system variables are left unchanged, and user defined variables will be recomputed, but you can override individual cases, if you choose.

Obs number and time are fixed. You can't change them.

```
════════════════Action List════════════
-35 Obs    :No Change
-21 HHMMSS :Doesn't change
-36 FTime  :Doesn't change
-76 EBal?  :No Change
 30 Photo  :Always recomputed
 23 Cond   :Always recomputed
  Edit                              done
```

User defined variables are usually 'Always Recomputed'.

Area and StmRat can be set here or from the main screen.

```
════════════════Action List════════════
 25 VpdL   :Always recomputed
 221 CTleaf :Always recomputed
-33 Area   :One global value (4.4)
 111 BLC_1  :Always recomputed
-34 StmRat :One global value (0.5)
 11 BLCond :Always recomputed
  Edit                              done
```

System variables, such as H2OR, can also be set to 'One global value' or 'Prompt for new values'.

Some system values that derive from others are always recomputed.

```
════════════════Action List════════════
-1 CO2R    :No Change
-2 CO2S    :No Change
-4 H2OR    :No Change
-5 H2OS    :No Change
-14 RH_R   :Always recomputed
-15 RH_S   :Always recomputed
  Edit                              done
```

Figure 13-9. The CUSTOM function key allows you to edit the entire list of variables, selecting what happens to each (with some limitations, of course).

Hints

Skipping Observations

While recomputing, every line from the data file that is read and interpreted as being a valid data line produces a '.' on the display. Otherwise, an 'x' is printed. The basic reason for skipping a record is that there are fewer than expected items in the line, based on the label line for the data file. Some reasons for this are

- **Remark line**
  Remarks in the file consist of one item, a quoted string.

- **Recomputing a file with multiple files**
  This is discussed in detail below.

- **A corrupted label line**
  If you are having trouble getting recompute to see any observations (that is, all the lines produce 'x' instead of '.'), examine the label line to see if all of the variable names are there. Also, look for any double commas.

  'x' lines are written as-is to the destination file.

Multiple (Appended) Files

If the source data file is actually a collection of multiple data sets (created by specifying an existing file when opening the log destination, and choosing the "Append" option when notified that a file exists already), then there are some constraints you should be aware of:

1 **The recompute program reads until the end of the source file is reached.**
   Blank lines, remarks, etc. will not stop it.

2 **Only the first data header is considered**
   If you changed log formats, or compute lists, etc., they will not be taken into consideration. If this is the case, you should split up the source file into multiple files, before recomputing.

3 **Non-data lines:**
   All lines that do not have enough data to meet the expected number of variables (string or numeric) are transferred as is to the destination. The program will print out a "." for each good data line, and a "x" for any other line. Thus, if you see

   `.....x...xxxxxxxx.....`

It probably means it found a remark after 5 observations, and then after another three observations, it found another data set appended on, with 4 observations. Or, someone logged nine remarks one after the other.

OPEN's System Variables

14

Quantities provided by OPEN

14

OPEN's System Variables

OPEN defines, computes, and maintains a number of quantities that are of interest to you, such as CO_2 concentrations, temperatures, time of day, etc. These are referred to as *System Variables*, and you can view them, log them, plot them, and wonder about them. Should you find yourself doing the latter, then this is the chapter for you.

There is another group of variables that *you* can define, compute, and maintain, and should you wonder about those, Chapter 15 (**Defining User Variables**) may satisfy your curiosity.

Background Information

Properties of Variables

Every system variable (and every user variable, as well) has the following properties:

1   **A unique ID number**
    This is used as a reference in format files (display, log, and strip chart, for example). System variables have negative ID values, user defined variables have ID values greater than or equal to 0.

2   **Labels for Display and for Logging**
    Display labels are what you see in New Measurements mode. Log labels are how the variable is labelled in your output file. Sample cell CO_2 concentration, for example, has a display label of *CO2S_µml* and a log label of *CO2S*. Most system variables use the same label for each, however.

3   **Formats for display and logging**
    Formats dictate how much space to take up when displaying the variable's value, how many significant digits to show, right or left justified, etc. Typically, display formats take up 8 spaces, while logging formats are more compressed.

4   A short description.
This is what you see in the list generated by the **What's What** function key in New Measurements mode.

5   A variable name
The variable's name is how you refer to it in a program. For example, if you are writing an AutoProgram (Chapter 25) or defining a user variable (Chapter 15) and need to refer to the sample cell CO_2 concentration, the variable name is *co2_2_um*.

Table 14-10 on page 14-23 lists the System Variables defined by OPEN, including their labels, variable names, descriptions, and references to where they are defined or discussed.

When Are They Computed?

In New Measurements mode, system and user variables are computed as follows:

1   Measured Quantities
The instrument's A/D converter has a set of new readings available every 0.5 seconds. These readings are the raw signals from each sensor. Each time these new A/D readings are available, OPEN computes the sensor's readings in meaningful units, and some ancillary values, such as relative humidity, which are based on multiple sensors. Thus, system variables that are associated directly or indirectly with sensors are computed and available with updated values every 0.5 seconds. These variables are documented under the headings **Measured Variables** and **Computed Variables**, below. It should be noted that these values that are available every 0.5 seconds are themselves an average of multiple readings, depending on the channel (Table 14-1), averaged for whatever the system averaging time is set to (contained in the configuration node <user> <a2d> <avgtime>).

Table 14-1. The three sampling rates used in the LI-6400

Resolution	Samples/sec	Variable	Description
Low	20	*lowResSPS*	Most channels
High	200	*highResSPS*	IRGA channels
Fluor.	1000	*flrSPS*	Flr signal from 6400-40

2   User variables (photosynthesis rate, conductance, etc.)
Each time a new set of measured quantities is available, the user variables are computed. These typically include photosynthesis, conductance, etc.

3   **Status variables**

There's a group of system variables that convey some system status information. These are also updated every 0.5 seconds, and are described in **Status Variables** on page 14-15.

Measured Variables

These system variables are measured signals or computed quantities associated with the various analog sensors of the LI-6400. This section discusses analog measurements and computations, presenting equations for each sensor used.

Pressure

Atmospheric pressure P (kPa) is measured with a transducer located in the console of the LI-6400. The signal (mV) is V_p.

ID: -11[1]

$$P = a_{p0} + a_{p1}V_p \qquad (14\text{-}1)$$

where a_{p0} and a_{p1} are the 1st and 2nd calibration coefficients respectively specified by a node in the factory calibration tree.

```
<li6400> <factory> <press> { a_p0  a_p1 }
```

You can view the values in the Calib Menu (Figure 14-1).
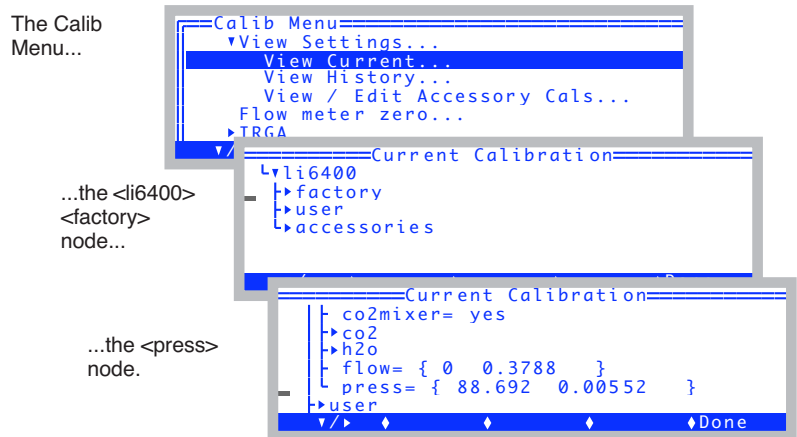
*Figure 14-1. Viewing the <li6400> <factory> <press> node. Use the **f1** key to expand nodes (turn ▶ into ▼).*

[1.] These ID numbers refer to the system ID number. See Table 14-10 on page 14-23.

Air Temperature

The air temperature within the IRGA sample cell T_a (C) is computed from the signal V_a (mV) of a linearized thermistor located just beneath the circulation fan in the sample IRGA (shown in Figure 19-33 on page 19-37). The equation is

ID: -9

$$T_a = \frac{V_a}{100} \qquad (14\text{-}2)$$

Block Temperature

The temperature T_b (C) of the metal block containing the optical path of the IRGAs is also measured with a linearized thermistor, whose signal (mV) is V_b.

ID: -8

$$T_b = \frac{V_b}{100} \qquad (14\text{-}3)$$

IRGA Temperature

The IRGA's detectors view mostly radiation that has come through the optical path. However, a tiny amount of peripheral radiation can find its way into the detector as well. For this reason, we measure the temperature T_x (C) of this background with a chip thermistor whose signal in mV is V_x, using

ID: -58

$$T_x = \frac{1}{5.3375 \times 10^{-6} V_x + 0.0090167} - 22.696 \qquad (14\text{-}4)$$

We then do a software correction for zero drift that is described below in (14-8) and (14-13).

Water Concentrations

The equations relating reference and sample IRGA signals V_{wr} and V_{ws} (mV) to reference and sample H_2O concentrations W_r and W_s (mmol mol^{-1}) are

ID: -4

$$W_r = \left[f\left((V_{wr} - V_{cr}x_{cwr} + z_w)G_{wr}\frac{101.3}{P} \right) \left[\frac{273 + \frac{T_b + T_a}{2}}{273} \right] + W_{mr} \right] F_{O_2} \quad (14\text{-}5)$$

ID: -5

$$W_s = \left[f\left((V_{ws} - V_{cs}x_{cws} + z_w)G_{ws}\frac{101.3}{P}\right)\left[\frac{273 + T_a}{273}\right] + W_{ms}\right]F_{O_2} \quad \text{(14-6)}$$

where the function $f(x)$ is the factory calibration function.

$$f(x) = a_{w0} + a_{w1}x + a_{w2}x^2 + a_{w3}x^3 \qquad \text{(14-7)}$$

The coefficients $a_{w0}...a_{w3}$ are specified by a node in the factory calibration tree

    `<li6400> <factory> <h2O> <coeffs>` { a_{w0} a_{w1} a_{w2} a_{w3} }

x_{cwr} and x_{cws} are the direct cross sensitivity correction factors for CO_2 on water vapor for the reference and sample cells respectively, and are specified in the factory calibration node

    `<li6400> <factory> <h2o> <xs>` { x_{cwr} x_{cws} }

G_{wr} and G_{ws} are gain factors set in the "IRGA Span" routine of the Calib Menu of OPEN (**Setting the H2O Span** on page 18-20), and visible in the node

    `<li6400> <user> <irga_span> <h2O>` { G_{wr} G_{ws} }

The zero drift correction term z_w is based on the difference between the IRGA's background temperature now (T_x) and the temperature when the IRGA's water channel was last zeroed (T'_{xw}).

$$z_w = (T'_{xw} - T_x)S_w \qquad \text{(14-8)}$$

S_w is the zero shift calibration term for water, found at

    `<li6400> <factory> <h2o> <dvdt>` S_w

Note the different temperatures used in the density corrections; chamber air temperature T_a is appropriate for the sample cell, since that is where the sensor is located. Since the reference cell is bored through the metal block, we use the average of the block temperature T_b and T_a for adjusting reference concentration.

Both W_{ms} and W_{mr} are small adjustment factors used by the "IRGA Zero"

routine to compensate for the resolution of the D/A converters used to zero the IRGAs. W_{ms} is also used for the match correction that is determined every time the IRGAs are matched in New Measurement's Match Mode.

F_{o_2} is a correction for oxygen concentration based on Bunce (2002)[2].

$$F_{o_2} = 0.96 + \frac{0.04}{21}X_o \qquad (14\text{-}9)$$

where X_o is the oxygen concentration in percent. This is a user-entered system constant (ID: -52).

Carbon Dioxide Concentrations

The equations relating reference and sample IRGA signals V_{cr} and V_{cs} (mV) to reference and sample CO_2 concentrations C_r and C_s (μmol mol^{-1}) are

ID: -1
$$C_r = g\left(\frac{(V_{cr} - V_{wr}x_{wcr} + z_c)G_{cr}}{B_r}\frac{101.3}{P}\right)\left[\frac{273 + \frac{T_b + T_a}{2}}{273}\right]B_r + C_{mr} \quad (14\text{-}10)$$

ID: -2
$$C_s = g\left(\frac{(V_{cs} - V_{ws}x_{wcs} + z_c)G_{cs}}{B_s}\frac{101.3}{P}\right)\left[\frac{273 + T_a}{273}\right]B_s + C_{ms} \qquad (14\text{-}11)$$

where the function $g(x)$ is the factory calibration polynomial.

$$g(x) = a_{c0} + a_{c1}x + a_{c2}x^2 + a_{c3}x^3 + a_{c4}x^4 + a_{c5}x^5 \qquad (14\text{-}12)$$

The coefficients $a_{c0}...a_{c5}$ are specified by the configuration command

    <li6400> <factory> <co2> <coeffs> { a_{c0} a_{c1}...a_{c5}}

x_{wcr} and x_{wcs} are the direct cross sensitivity correction factors for H_2O on CO_2 for the reference and sample cells respectively, and are specified in the factory calibration node

---

[2]J.A.Bunce, 2002. Sensitivity of infrared water vapor analyzers to oxygen concentration and errors in stomatal conductance. *Photosynthesis Research* 71:273-276.

```
<li6400> <factory> <co2> <xs> { x_wcr  x_wcs}
```

G_{cr} and G_{cs} are gain factors set in the "IRGA Span" routine set in the Calib Menu of OPEN (**Setting the CO2 Span** on page 18-17), and found at

```
<li6400> <user> <irga_span> <co2> { G_cr  G_cs }
```

The zero drift correction term z_c for CO_2 is analogous to that for water (14-8).

$$z_c = (T_{xc} - T_x)S_c \qquad (14\text{-}13)$$

with S_c found at

```
<li6400> <factory> <co2> <dvdt> S_c
```

The band broadening correction terms B_r and B_s for water vapor and oxygen in the IRGA cell are computed from

$$B_r = 1.0 + 0.0005\,W_r - 0.001\dot{X}_O$$
$$B_s = 1.0 + 0.0005\,W_s - 0.001X_O \qquad (14\text{-}14)$$

This formulation for the water band broadening correction is derived in **IRGA Corrections** on page 14-27.

T_a and T_b are used for the density corrections for the same reasons described above for water, and C_{ms} and C_{mr} are the CO_2 versions of W_{ms} and W_{mr} also described above. X_o is oxygen concentration in percent (ID: -52).

Leaf Temperature

The leaf temperature T_l is measured with a chromel-constantan thermocouple junction. The reference junction is the IRGA block, whose temperature (T_b) is known. The thermocouple's signal is amplified to become V_l. (mV), and is related to leaf temperature T_1 (C) by

ID: -10

$$T_l = T_b + \frac{V_l}{100} \qquad (14\text{-}15)$$

Flow Meter

The flow meter is located in the console, and its signal V_f (mV) relates to flow rate F (μmol s^{-1}) by

ID: -7

$$F = a_f V_f \qquad (14\text{-}16)$$

The calibration factor a_f is found in the calibration node

    `<li6400> <factory> <flow> { 0 `a_f` }`

In-Chamber PAR

The in-chamber PAR sensor is a GaAsP sensor located in the top of the standard chamber. However, when the 6400-02 or -02B Light Source is being used, the sensor is a silicon photodiode located in the light source itself. Either way, the relation between the sensor's signal V_{qc} and the light reading Q_c is

ID: -12

$$Q_c = \begin{cases} a_{qc}(V_{qc} - V_{qo}) & \text{if parInMode = 1} \\ \tau_x Q_x & \text{if parInMode = 2} \end{cases} \qquad (14\text{-}17)$$

If there is no internal sensor, the in-chamber PAR value can be estimated (*parInMode* = 2) from an external sensor measurement Q_x multiplied by a transmittance factor, τ_x . See **Specifying the Light Sensors and Methodology** on page 8-5. The controlling factor is the node

    `<open> <light> <par_in> `*parInMode*

and τ_x is contained in the configuration node

    `<open> <light> <par_in> <sensor> <cal> <transm> `τ_x

If Q_c is measured (*parInMode* = 1), the value of the offset term V_{qo} is contained in the calibration node

    `<li6400> <user> <parin_offset> `V_{qo}

but can be changed by the user by performing the routine **Zeroing the ParIn Signal** on page 18-29. The value of a_{qc} can be found in the configuration node

    `<open> <light> <par_in> <sensor> <cal> `a_{qc}

and is determined by a number of other configuration settings, such as what the light source is (if any), and what sensor is used.

$$a_{qc} = \begin{cases} a_e \tau & \text{if LightSource is 6400 LED} \\ a_g f_a & \text{if LightSource is anything else} \end{cases} \qquad (14\text{-}18)$$

The actinity correction factor f_a for the GaAsP sensor allows the "same" calibration factor a_g to be used for various light sources.

The term τ in Eqn (14-18) is a transmittance factor, normally 1.0. If you have a light source mounted in a non-standard way, say above a chamber not normally designed for that source, then τ can be used to account for transmission losses, so that the *parIn_μm* value will still be relative to the leaf in the chamber. See, for example, **The Transm Factor** on page 8-20. The value τ is contained in the configuration node

    <open> <light> <parin> <sensor> <cal> <transm> τ

Fluorescence note: Normally, the sensor signal V_{qc} comes from analog input channel 15. When the instrument is configured for using the 6400-40 LCF as the light source, V_{qc} comes from channel 23.

External PAR

The optional external quantum sensor is a LI-COR LI-190 Quantum Sensor, whose signal V_{qx} relates to reading Q_x (μmol m^{-2} s^{-1}) by

ID: -13

$$Q_x = \frac{10}{a_{qx}} V_{qx} \qquad (14\text{-}19)$$

where the calibration factor a_{qx} ($\dfrac{\mu A}{1000\ \mu\text{mol m}^{-2}\ \text{s}^{-1}}$) is held in the configuration node

    <open> <light> <par_out> <sensor> <cal> a_{qx}

Computed Variables

Computed variables do not have sensors of their own, but instead derive from measurements of multiple sensors. There is no relative humidity sensor, for example, but we can compute relative humidity from water mole fraction, pressure, and temperature.

Humidity Variables

Vapor Pressure

The water IRGAs measure vapor concentration in mmol mol^{-1}. To convert this to vapor pressure, convert to mol mol^{-1} and multiply by the total pressure P (kPa). Thus, given a reference measurement of water W_r, the reference vapor pressure e_r (kPa) is

ID: -53
$$e_r = \frac{W_r P}{1000}$$
(14-20)

while in the sample cell, the vapor pressure e_s (kPa) is

ID: -54
$$e_s = \frac{W_s P}{1000}$$
(14-21)

Relative Humidity

Relative humidity is the ratio of vapor pressure to saturation vapor pressure. The reference and sample relative humidity h_r and h_s (in percent) are given by

ID: -14
$$h_r = \frac{e_r}{e(T_a)} 100$$
(14-22)

ID: -15
$$h_s = \frac{e_s}{e(T_a)} 100$$
(14-23)

Saturation Vapor Pressure

The saturation vapor pressure function $e()$ used by OPEN is from Buck (1981)[3]:

[3.]Buck, A.L. (1981) New equations for computing vapor pressure and enhancement factor. J. Appl. Meteor. 20:1527-1532.

$$e(T) = 0.61365e^{\frac{17.502T}{240.97 + T}}$$

<div align="right">(14-24)</div>

where the argument T is in degrees C. The LPL implementation of function $e()$ is named *SatVap*. However, for Excel compatibility, Eqn (14-24) appears explicitly in ComputeLists (see Figure 15-18 on page 15-20).

Dewpoint Temperature

Dewpoint temperature, the temperature at which condensation will occur, in the reference and sample cells T_{dr} and T_{ds} is computed from the corresponding vapor pressures using a dew point function $d()$:

ID: -16
$$T_{dr} = d(e_r)$$
<div align="right">(14-25)</div>

ID: -17
$$T_{ds} = d(e_s)$$
<div align="right">(14-26)</div>

The dew point function $d()$ is Equation (14-24) solved for temperature T, along with a check to keep bad things from happening should vapor pressure be less than or equal to 0.

$$d(e) = \begin{cases} -99.9 & \text{if } e < 0.01 \\ \dfrac{240.97z}{17.502 - z} & \text{if } e > 0.01 \end{cases}$$

<div align="right">(14-27)</div>

where the argument e is vapor pressure in kPa, and $z = \ln\left(\dfrac{e}{0.61365}\right)$. The LPL

implementation of function $d()$ is named *DewPoint*.

Stability Variables

Any number of quantities can be included in the system stability test (**Stability Indicators** on page 4-41). For each, a mean, standard deviation, coefficient of variation (in percent), and a rate of change (per minute) is computed. In addition, the coefficient of variation and slope are always computed for the fluorescence signal (when configured for fluorescence mode), and are stored in variables *FlrCV_%* (ID:-97), *dF/dt* (ID:-98), and *Fmean* (ID:-115).

For n observations of quantity x, the formulae for mean, standard deviation, and coefficient of variation are given below:

Mean	$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$	*(14-28)*
Standard Deviation	$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$	*(14-29)*
Coeff Variation (%)	$$c = 100(s/\bar{x})$$	*(14-30)*

Four system variables provide summary information on the stability list variables. They are *Stable* (ID:-71), *StableF* (ID:-72), and a third (ID:-73) whose label consists of the first letter of each variable in the stability list. The final one is *TotalCV* (ID:-74) which is the sum of the CVs of the stability variables. *Stable* is a string that shows the number of stable variables, and the total number in the list, such as "2/5". *StableF* is this same information as a decimal (e.g. 0.40). The third one (ID: -73) is a string of 1's and 0's that indicate stability of each variable (1=stable, 0=not).

Time and Logging Variables

ID: 0

"Time"
The number of seconds since the instrument was powered on. Note: this value will start to lose adequate resolution (0.5 seconds) after about 92 days. This means you need to power off at least every three months or so, or else the real time graphics and fluorescence events (flash, dark pulse) will begin to lose horizontal resolution.

ID: -35

"Obs"
The number of observations that have been stored *since the log file was last opened*. Note that this will not be the number of observations actually stored in the file if an existing file is opened for appending.

ID: -36

"FTime"
The number of seconds (floating point) since a log destination has been opened. If no log destination is active, this number is meaningless.

ID: -64

"DecHour"
The time of day in decimal hours. For example, 02:15 pm would be represented by 14.2500.

ID: -21   **"HH:MM:SS"**
Time of day is shown as an 8 character string containing HH:MM:SS on a 24 hour basis.

ID: -69   **"DOY"**
The day of the year. January 1st is 1, and December 31 is 365 (366 for leap years). The value is an integer.

ID: -70   **"YYYYMMDD"**
The year, month, and day represented as an 8 digit integer.

Status Variables

Several status indicators are available as system variables:

ID: -31   **"CO2 H2O Pump Flow Mixr Fan"**
This is a string designed to show the status of six hardware components, and is available on level J of the standard display map. It is simply a composite of the status variables described below.

ID: -25, -26   **"CO2" and "H2O"**
There are 4 possible values of these string variables:

Table 14-2. CO_2 and H_2O IRGA status.

| | | What is Sensed | | | |
| | | CO$_2$ IRGA | | H$_2$O IRGA | |
Value	**Meaning**	**0x0202[a]**	**0x0203**	**0x0200**	**0x0201**
OK	IRGAs are OK.	1	1	1	1
errR	Reference IRGA error.	0	1	0	1
errS	Sample IRGA error.	1	0	1	0
err	Error both IRGAs.	0	0	0	0

a. Digital inputs. 0x0201 is port 2, pin1, etc.

The error condition is triggered by too much light blockage in the cell, or by the IRGAs not being connected. See **"IRGAs Not Ready"** on page 20-6.

ID: -27 **"PUMP"**

The pump status. An error condition indicates a blocked inlet, causing the pump to draw too much current. Without the 6400-01 CO_2 Mixer, this error condition is not sensed.

Table 14-3. Pump Status states.

Value	Meaning	What is Sensed		
		Without Mixer	With Mixer	
		0x0300[a]	0x0300	0x0100[b]
Off	Pump switched off.	0	1	ignored
OK	Pump OK.	1	1	1
err[c]	Pump drawing too much current.	not applicable	1	0

a.Pump control output.
b.Digital input port 1 pin 0.
c.Possible only when the 6400-01 CO_2 Mixer is installed.

ID: -29 **"MIXR"**

The 6400-01 CO_2 Mixer can indicate 4 possible values:

Table 14-4. Mixer Status.

Value	Meaning	What is Sensed		
		0x0303[a]	0x0101[b]	0x0102
Off	Mixer off.	0	ignored	ignored
OK	Mixer operating in balance.	1	1	1
Low	Mixer under-pressured.	1	1	0
High	Mixer overpressured.	1	0	ignored

a.CO_2 solenoid control output, port 3 pin 3.
b.Digital inputs port 1, pins 1 and 2.

The "Low" value indicates an under-pressurization, common right after installing a new CO_2 cartridge or when changing from low to high CO_2 concentrations. A spent cartridge will always cause this. The "High" value will briefly appear when lowering the concentration, or when asking for too low a value (see the related troubleshooting discussion starting on page 20-27).

ID: -28 **"FLOW"**

The flow control hardware can indicate four values:

Table 14-5. Flow status.

Value	Meaning	What is Sensed		
		0x0300[a]	0x0103[b]	0x0104
OK	Flow control in balance.	1	1	1
Low	Can't reduce flow low enough.	1	0	1
High	Can't raise flow high enough.	1	ignored	0

a.Pump control output, port 3 pin 0.
b.Digital input port 1, pins 3 and 4.

A "High" condition occurs when asking for unattainably high flow rates. The obvious way to make this happen in fixed flow mode (**F**) is to enter an unreachable target value, such as 2000 μmol s^{-1}. The more subtle way to make this happen is in the constant humidity control mode when the target is too dry, and/or the incoming air is too moist.

A "Low" condition occurs when asking for unattainably low flow rates on units equipped with the 6400-01 CO_2 Mixer. See also **"Flow is Too Low"** on page 20-7.

ID: -30 **"FAN"**

The sample cell and leaf chamber mixing fan can have three values:

Table 14-6. Fan status.

Value	Meaning	What is Sensed
Off	Fan switched off.	Voltage on DAC channel 7, the fan control.
Low	Fan at the Low voltage.	
High	Fan at the High voltage.	

The fan is controlled by setting a DAC (channel 7) between 0 (off) and 5 (high) Volts. The setting for "Low" is defined by the configuration command FanSlow=, and defaults to 4 volts.

ID: -23

"CHPWMF"

This is a six digit numerical composite of the status flags.

Table 14-7. Six digit status summary.

Code Letter	C		H		P		W		M		F	
Item	CO_2 IRGAs		H_2O IRGAs		Pump		Flow Control		CO2 Mixer		Chamber Fan[a]	
Possible Values	1	OK	1	OK	0	Off	0	Off	0	Off	0	Off
	2	errR	2	errR	1	OK	1	OK	1	OK	4	Slow
	3	errS	3	errS	2	err	2	Low	2	Low	5	Fast
	4	err	4	err			3	High	3	High		
LPL Function	StatusCO2		StatusH2O		StatusPump[b] IsPumpOn		StatusFlow		StatusInj		chFanState[c]	

a. The value for F is the number of volts that the fan control DAC is set to. By default, 5 is fast and 4 is slow, but this can be changed via the configuration command FanSlow=.

b. If 6400-01 Mixer is installed. Otherwise, use IsPumpOn.

c. An INT, not a FCT.

To get just one component of this number (such as the water IRGA status) in a Compute List or AutoProgram, use the LPL Function names given in the table. Figure 14-2 illustrates a piece of LPL code that does one thing if the IRGAs are OK, and another if they are not.

```
StatusCO2 StatusH2O OR IF
  /* Not ok */
...

ELSE
  /* both ok */
...

THEN
```

Figure 14-2. Using status information from an LPL program.

ID: -22   **"Program"**

When an AutoProgram is active, this string (default display level *k*) indicates the time remaining until the next step during the commands LPMeasure and LPMeasureTilStable (see **Useful AutoProgram Commands** on page 25-15).

Table 14-8. Uses of the Program variable.

When	Program	Meaning
In LPMeasure	HH:MM:SS	Time to next step.
In LPMeasureTilStable	HH;MM;SS	Time until minWaitTime expires.
	HH*MM*SS	Stability checking. Time until max-WaitTime expires.
Otherwise	- None -	No AutoProgram running.

ID: -56   **"ProgPrgs"**

Autoprogram progress (shown on default display level *k*). This 8 character string usually indicates the number of steps done in an AutoProgram, and the total number, such as "8 / 16". The AutoProgram command LPSetProgress controls this string (**Useful AutoProgram Commands** on page 25-15).

ID: -57   **"FwMxCrLp"**

This four digit (it's actually an 8 character string) value provides a control manager status indicator (Figure 14-3). A control is marked with a 1 if it is on target and stable (or turned off). (The control manager is discussed in Chapter 7.)
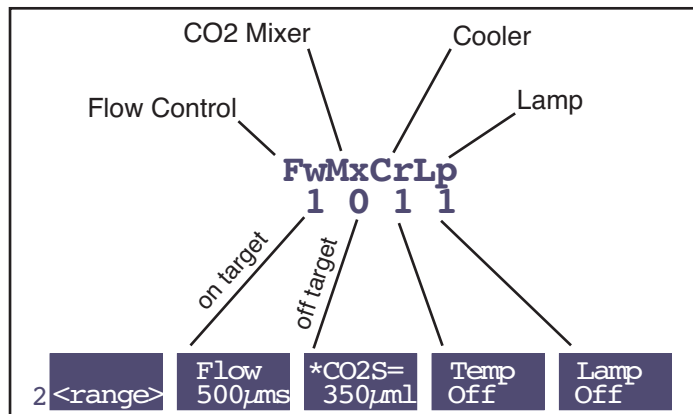
Figure 14-3. The control manager status variable indicates if a control is on target and stable.

A '0' in the FwMxCrLp display corresponds to an asterisk on the related control manager function key.

Boundary Layer Variables

The one-sided boundary layer conductance to water vapor is computed in the standard compute list (see Chapter 15) from the formula

ID:111
$$g_{bw} = sg_1 + g_0 \qquad\qquad (14\text{-}31)$$

where s is leaf area, and g_1 (ID:-18) and g_0 (ID:-19) are the slope and offset of boundary layer as a function of leaf area. Effective boundary layer conductance to water vapor g'_{bw} is then computed from stomatal ratio (K) from

ID: 11
$$g'_{bw} = \frac{g_{bw}(K+1)^2}{K^2 + 1} \qquad\qquad (14\text{-}32)$$

Where g_1 and g_0 come from depend on the configuration flag g_{type}, which lives in the configuration node

    `<open> <comps> <bc_total> <bd_oneside> <type>` g_{type}

where g_{type} can be one of three values (Table 14-9).

Table 14-9. The three boundary layer modes.

g_{type}	Meaning
0	g_0 user entered, $g_1 = 0$
1	g_1, g_0 user entered
2	g_1, g_0 from Table

Type 0: bc_oneside is directly editable.

```
==FactoryDefault_6.0=============
| ├ stomatal_ratio= 1
| └▼* bc_oneside= 1.420
|   └ * type= 0: Fixed Value
├▶prompts
├▶stability
├▶* log
▼/▶  ♦Edit   ♦      ♦Undo   ♦Done
```

Type 1: specify a slope and offset.

```
==FactoryDefault_6.0=============
| ├ stomatal_ratio= 1
| └▼* bc_oneside= 1.420
|   ├ * type= 1: Function of Area
|   ├ area= 6
|   ├ * slope= 0
|   └ * offset= 1.42
▼/▶  ♦Edit   ♦      ♦Undo   ♦Done
```

Type 2: slope and offset come from a lookup table.

```
==FactoryDefault_6.0=============
| ├ stomatal_ratio= 1
| └▼bc_oneside= 1.420
|   ├ type= 2: Lookup Table (Area, Fan
|   ├ table= "/Sys/Lib/StdBLCTable"
|   ├ area= 6
|   └ fan= Fast
▼/▶  ♦Edit   ♦      ♦Undo   ♦Done
```

Figure 14-4. The configuration tree changes to reflect the boundary layer mode.

Boundary Layer Tables
Figure 14-5 illustrates the format of a boundary layer lookup table.

Figure 14-5. Listing of the boundary layer lookup table "/Sys/Lib/StdBLCTable". The data follows the BLCTABLE= string, and two values are expected in that line: the minimum leaf area, and the maximum leaf area. Following this comes 6 lines, one for each fan speed voltage (0 to 5). On each line is a pair of values: the one-sided boundary layer conductance for the low area, and the boundary layer conductance for the high area. The values have units of mol m^{-2} s^{-1}.

The values in the table are from measurements using saturated filter paper. Since direct leaf temperature measurements are problematic in these conditions (wet paper temperature much cooler than air temperature), the leaf temperature thermocouple is used to measure air temperature, and leaf temperature is computed after the fact using an interactive energy balance solution. The program used is **ENERGYBAL** on page 21-13.

There are four boundary layer tables in version 6.2, located in the /Sys/Lib directory:

BLCTable_2x6	*For the 2x6 chambers*
BLCTable_arabidposis	*6400-16 Extended Reach Chamber*
BLCTable_LCF	*For the 6400-40 LCF*
StdBLCTable	*For the 2x3 chambers*

List of Open 6.2 System Variables

Table 14-10 lists the system variables provided by OPEN. Any of these can be displayed, logged, plotted, etc., as well as used in computations and Auto-Programs.

Table 14-10. System Variables - Labels and Variable Names.

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
0	Time	Seconds since power on	*a2dTime*	page 14-14
-1	CO2R_μml CO2R	Reference CO_2 μmol mol^{-1}	*co2_1_um*	C_r Eqn (14-10), pg 14-8
-2	CO2S_μml CO2S	Sample CO_2 μmol mol^{-1}	*co2_2_um*	C_s Eqn (14-11), pg 14-8
-3	ΔCO2_μml DCO2	ΔCO_2 μmol mol^{-1}	*co2_diff_um*	$\Delta C = C_s - C_r$
-4	H2OR_mml H2OR	Reference H2O mmol mol^{-1}	*h2o_1_mm*	W_r Eqn (14-5), pg 14-6
-5	H2OS_mml H2OS	Sample H2O mmol mol^{-1}	*h2o_2_mm*	W_s Eqn (14-6), pg 14-7
-6	ΔH2O_mml DH2O	ΔH_2O mmol mol^{-1}	*h2o_diff_mm*	$\Delta W = W_s - W_r$
-7	Flow_μml Flow	Flow Rate μmol s^{-1}	*flow_um*	F Eqn (14-16), pg 14-10
-8	Tblock°C Tblk	IRGA Block Temp °C	*tblk_c*	T_b Eqn (14-3), pg 14-6
-9	Tair°C Tair	Chamber Air Temp °C	*tcham_c*	T_a Eqn (14-2), pg 14-6
-10	Tleaf°C Tleaf	Leaf Temp °C, measured with the thermocouple	*tleaf_c*	T_l Eqn (14-15), pg 14-9
-11	Prss_kPa Press	Atm Press kPa	*press_kpa*	P Eqn (14-1), pg 14-5
-12	ParIn_μm PARi	In-chamber PAR μmol m^{-2} s^{-1}	*parIn_um*	Q_c Eqn (14-17), pg 14-10
-13	ParOutμm PARo	External PAR μmol m^{-2} s^{-1}	*parOut_um*	Q_x Eqn (14-19), pg 14-11
-14	RH_R_% RH_R	Reference RH%	*rhIn*	h_r Eqn (14-22), pg 14-12
-15	RH_S_% RH_S	Sample RH%	*rhOut*	h_s Eqn (14-23), pg 14-12

Table 14-10. (Continued) System Variables - Labels and Variable Names.

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-16	Td_R_°C TdR	Reference Dew Point °C	*tdIn*	T_{dr} Eqn (14-25), pg 14-13
-17	Td_S_°C TdS	Sample Dew Point °C	*tdOut*	T_{ds} Eqn (14-26), pg 14-13
-18	BLCslope	slope as function of area	*blcSlope*	g_1 Eqn (14-31), pg 14-20 <open> <comps> <bc_total> <bc_oneside> <slope>
-19	BLCoffset	offset as function of area	*blcOffset*	g_0 Eqn (14-31), pg 14-20 <open> <comps> <bc_total> <bc_oneside> <offset>
-21	HH:MM:SS HHMMSS	Real time clock	*clocktime*	page 14-15
-22	Program	Auto Program time status	*lpTimeStat*	page 14-19
-23	CHPWMF Status	Numerical status code	*statusWord*	page 14-18
-24	Battery	Battery voltage	*battery_v*	Reported in Volts
-25	CO2	CO2 IRGA status	*stat_co2*	page 14-15
-26	H2O	H2O IRGA status	*stat_h2o*	page 14-15
-27	PUMP	Pump status	*stat_pump*	page 14-16
-28	FLOW	Flow Control status	*stat_flow*	page 14-17
-29	MIXR	CO2 Mixer status	*stat_inj*	page 14-16
-30	FAN	Chamber fan status	*stat_fan*	page 14-17
-31	CO2...FAN	The status line	*statLineVar*	page 14-15
-32	BLC_mol BLCond	(Not used by version 6)	*condBL_mol*	
-33	AREA_cm2 Area	In-chamber leaf area cm^{-2}	*area_cm2*	<open> <comps> <bc_total> <bc_oneside> <area>
-34	STMRATIO StmRat	Stomatal ratio estimate	*stom_rat*	<open> <comps> <bc_total> <stomatal_ratio>
-35	Obs	# Obs stored in log file	*obsInPad*	page 14-14
-36	FTime	Time since logging started (s)	*obsTime*	page 14-14
-37	CO2R_mv	Ref CO_2 IRGA mV	*co2_1_mv*	V_{cr} Eqn (14-10), pg 14-8
-38	CO2S_mv	Sample CO_2 IRGA mV	*co2_2_mv*	V_{cs} Eqn (14-11), pg 14-8
-39	H2OR_mv	Ref H_2O IRGA mV	*h2o_1_mv*	V_{wr} Eqn (14-5), pg 14-6
-40	H2OS_mv	Sample H_2O IRGA mV	*h2o_2_mv*	V_{ws} Eqn (14-6), pg 14-7

Table 14-10. (Continued) System Variables - Labels and Variable Names.

ID	Display Label / Log Label[a]	Description	Variable Name	Definition or Reference
-41	Tblk_mv	IRGA Block temp mV	*tchamblk_mv*	V_b Eqn (14-3), pg 14-6
-42	Tair_mv	Sample air temp mV	*tcham_mv*	V_a Eqn (14-2), pg 14-6
-43	Tleaf_mv	Leaf temp mV	*tleaf_mv*	V_l Eqn (14-15), pg 14-9
-44	flow_mv	Flow meter mV	*flow_mv*	V_f Eqn (14-16), pg 14-10
-45	press_mv	Pressure mV	*pressure_mv*	V_p Eqn (14-1), pg 14-5
-46	parIn_mv	In-chamber PAR mV	*parIn_mv*	V_{qc} Eqn (14-17), pg 14-10
-47	parOutmV	External PAR mV	*parOut_mv*	V_{qx} Eqn (14-19), pg 14-11
-48	CRagc_mv	Ref CO_2 IRGA AGC mV	*agc_c1_mv*	
-49	CSagc_mv	Chamber CO_2 IRGA AGC mV	*agc_c2_mv*	**AGC voltages** on page 20-15
-50	HRagc_mv	Ref H_2O IRGA AGC mV	*agc_h1_mv*	
-51	HSagc_mv	Chamber H_2O IRGA AGC mV	*agc_h2_mv*	
-52	Oxygen%	Oxygen concentration (%)	*oxyPct*	x_o Eqn (14-9), pg 14-8 and Eqn (14-14), pg 14-9
-53	vapR_kPa	Ref vapor press kPa	*eAir_1_kPa*	e_r Eqn (14-20), pg 14-12
-54	vapS_kPa	Sample vapor press kPa	*eAir_2_kPa*	e_s Eqn (14-21), pg 14-12
-55	BLC1_mol	Not used by version 6	*condBL_one*	
-56	ProgPrgs	AutoProgram Progress	*lpProgress*	page 14-19
-57	FwMxCrLp	Control Panel Stability: Flow Mixer Cooler Lamp	*cpStable*	page 14-19
-58	Tirga°C / Tirga	IRGA Temp C	*tIrga_c*	T_x Eqn (14-4), pg 14-6
-59	Tirga_mv	IRGA Temp mV	*tIrga_mv*	V_x Eqn (14-4), pg 14-6
-60	uc_20_mV	User Channel 20 mV	*chan20_mv*	
-61	uc_21_mV	User Channel 21 mV	*chan21_mv*	See **Analog Input Channels** on page 16-29
-62	uc_22_mV	User Channel 22 mV	*chan22_mv*	
-63	uc_23_mV	User Channel 23 mV	*chan23_mv*	
-64	DecHour	Decimal time of day	*decHour*	page 14-14
-65	CsMch	Sample CO_2 offset μmol mol^{-1}	*co2_2_offset*	C_{ms} Eqn (14-11), pg 14-8
-66	HsMch	Sample H_2O offset mmol mol^{-1}	*h2o_2_offset*	W_{ms} Eqn (14-6), pg 14-7
-67	CrMch	Ref CO_2 offset μmol mol^{-1}	*co2_1_offset*	C_{mr} Eqn (14-10), pg 14-8

Table 14-10. (Continued) System Variables - Labels and Variable Names.

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-68	HrMch	Ref H_2O offset mmol mol^{-1}	*h2o_1_offset*	W_{mr} Eqn (14-5), pg 14-6
-69	DOY	Day of the year (0…366)	*clockDOY*	page 14-15
-70	YYYYMMDD	Date code (integer)	*clockDate*	page 14-15
-71	Stable	Stable / Total	*StableStat*	**Stability Indicators** on page 4-41
-72	StableF	Stable / Total as fraction	*fractStable*	
-73	<letters>	Status as string. (e.g. "1101")	*stableFlags*	
-74	TotalCV	Sum of stab. variables' CV	*totalCV*	
-76	EBal?	Do energy balance?	*doEB*	**Using Energy Balance in OPEN** on page 17-6
-77	f_parin	Fraction of *ParIn_μm* to use for EB	*f_parin*	
-78	f_parout	Fraction of *ParOut_μm* to use for EB	*f_parout*	
-79	alphaK	Used in the conversion of μmol mol^{-1} to W m^{-2}	*alphaK*	
-80	F	Fluorescence signal (zero subtracted)	*flr_f*	**Display Summary** on page 27-28
-81	%Blue	Blue fraction	*bluePct*	
-82	FlrMax	Max F during last flash	*flrMax*	
-83	FPeak_μm FPeak	Max PAR during last flash	*flashMax*	
-84	FCnt	Flash count	*flashCount*	
-85	Fzero	Fluorescence zero value	*flrZero*	
-86	Fo	Minimal F, dark adapted	*flr_o*	
-87	Fo'	Minimal F, light adapted	*flr_op*	
-88	Fm	Maximum F, dark adapted	*flr_m*	
-89	Fm'	Maximum F, light adapted	*flr_mp*	
-90	Fs	Steady state fluorescence	*flr_s*	**Display Summary** on page 27-28
-91	FlrEvent	Last action	*flrStat*	
-92	M:Int...Gn	Measuring beam configuration	*msrStat*	
-93	F:Dur...Hz	Flash settings	*flashStat*	
-94	D:Dur...Hz	Dark pulse settings	*darkStat*	
-95	FlrMin	Lowest F during last dark pulse	*flrMin*	
-96	ParIn@Fs	PAR when F_s last set	*parIn_fs*	

Table 14-10. (Continued) System Variables - Labels and Variable Names.

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-97	FlrCV_%	CV (in percent) of F	*flrRms*	page 14-13
-98	dF/dt	Rate of change of F (per minute)	*flrSlope*	
-99	FMaxStd	Simple F_{max} during MPF	*FlrMaxStd*	
-100	Dcnt	Dark pulse count	*darkCount*	
-101	Aux1[b]			
-102	Aux2			
-103	Aux3			
-104	Aux4		*auxN.floatVa*l or *auxN.stringVal* where $N = 1,9$	**System Variables for Prompts** on page 9-28
-105	Aux5	Can be string or numeric.		
-106	Aux6			
-107	Aux7			
-108	Aux8			
-109	Aux9			
-111	chan15_mV	Analog input channel 15.	*chan15_mV*	Used for *parIn_mV* except when using LCF or RGB
-112	matchCO2	CO2R at previous match	*matchPrevCO2*	**Viewing Previous Match Information Anytime** on page 4-39
-113	matchH2O	H2OR at previous match	*matchPrevH2O*	
-114	mchElpsd	Time since last match (string)	*matchElapsed*[c]	
-115	Fmean	Flr mean value	*flrAvg*	page 14-13

a. Same as Display Label, if none shown

b. Default. -101 thu -109 have user defined labels.

c. The variable *matchPrevTime* is the time (seconds since power on) of the previous match. To get elapsed time in seconds, subtract *matchPrevTime* from the current time, *a2dTime*.

IRGA Corrections

Temperature

Temperature has two effects: one effect is via density, and the other is due to electronic and/or optical drift.

Density

The number of absorbing molecules in the path changes with temperature because the path is open and not a sealed container. Thus, the mole fraction doesn't change, even though the raw output of the analyzer does. This is corrected for by adjusting for the concentration predicted by the IRGA for the absolute temperature. (A more rigorous formulation is derived below in Eqn (14-33), pg 14-29 to Eqn (14-39).) The challenge is to get the correct temperature. Experience has shown that for the sample cell, the chamber air temperature works the best. This is not too surprising, since sensor for this is right under the mixing fan in the sample cell itself. Thus, T_a appears in the density correction term in the sample cell concentration equations: Eqn (14-6), pg 14-7 and Eqn (14-11), pg 14-8.

For the reference cell, it turns out that an average of the block temperature and the chamber air temperature works the best, so that's what is used in Eqn (14-5), pg 14-6 and Eqn (14-10), pg 14-8.

Zero Drift

The IRGA detectors are temperature controlled for stability, but this does not entirely eliminate drift with temperature. This drift can be isolated from the effects of density changes by seeing what the effect of temperature is when measuring dry, CO_2-free air. This in fact is done as part of the factory calibration, and the results are used in a software correction for zero drift.

The correction scheme is this: we know the response from the factory calibration (i.e. mv/°C for H_2O and also one for CO_2). This is stored in the instrument, and applied to measurements based on how much the temperature has changed from the last time the zero was set (either by the user or at the factory). Thus, one of the hidden side effects of zeroing the instrument is updated that "temperature when last zeroed" term. See Eqn (14-8), pg 14-7 and Eqn (14-13), pg 14-9. The temperature that is used for all of this comes from a sensor within the IRGA's electronics box. This value is also available to view: it is system variable -58, *Tirga*.

Pressure

Pressure, like temperature, has an effect on the raw output of the IRGA, but it is not quite the same as that of temperature. Rather than correcting the concentration for pressure like we do density, instead we correct the raw signal for pressure before we compute concentration. There is good theoretical reasoning for this (as well as experimental verification), and it is shown below in Eqn (14-33), pg 14-29 to Eqn (14-39).

Dilution Effects of H_2O

This is a correction we *don't* do, at least when computing CO_2 concentration in the LI-6400. The dilution effect is simply this: as you add molecules of a gas (water vapor, for example) to a mixture, the fraction of that mixture that is made up of something else (mole fraction of CO_2, for instance) has to decrease, since the total number of molecules in the mixture has increased.

Now for an airsteam flowing though a chamber containing a transpiring leaf (or in a chamber sitting on moist soil), there very definitely is dilution. However, we ignore that effect when computing CO_2 concentration, but account for it when computing photosynthetic rate (or soil CO2 efflux). Thus, the LI-6400 IRGA is always indicating the actual CO_2 concentration, not what the CO_2 concentration would be if there were no water vapor in it.

Band Broadening of O_2 and H_2O

Absorption in the infrared involves vibrational and rotational energy transitions. The 4.26 μm CO_2 absorption band is due to infrared energy absorption by a particular bond stretching mode that is coupled to rotational energy transitions that produce a large number of individual absorption lines. Individual absorption line widths are sensitive to intermolecular collisions and become broader with increasing pressure. Therefore, total absorption across a band per mole of absorber increases with pressure.

Full description of an absorption band is complex, but approximate expressions can be used over limited ranges of absorber mole fraction, pressure and pathlength. It can be shown that the "non-overlapping line approximation" applies at ambient pressure and CO_2 mole fraction over the short pathlengths found in LI-COR infrared gas analyzers (Wolfe and Zissis, 1978). This leads to a "scaling law" that allows absorption measured under one set of conditions to be scaled to other conditions (Jamieson, et al., 1963),

$$\frac{A}{P} = g(u/P) \qquad\qquad (14\text{-}33)$$

where A is total band absorption, P is total pressure (kPa), u is absorber amount (mol m^{-2}) $= \rho L$; ρ is molar density (mol m^{-3}), and L is pathlength (m); g is a general unspecified function.

From the ideal gas law, the absorber mole density ρ can be expressed as

$$\rho = \frac{p}{RT}$$
$$= \frac{XP}{RT} \tag{14-34}$$

where p is absorber partial pressure and X is absorber mole fraction (mol absorber / mol air). Therefore,

$$\frac{u}{P} = \frac{XL}{RT} \tag{14-35}$$

Substituting (14-35) into (14-33) and incorporating the constants L and R into a new function h gives,

$$\frac{A}{P} = h(\frac{X}{T}) \tag{14-36}$$

In principle, (14-36) can be solved for mole fraction, giving

$$X = h^{-1}(\frac{A}{P})T \tag{14-37}$$

Since LI-COR gas analyzers produce an output voltage that is proportional to absorptance,

$$V = KA \tag{14-38}$$

substituting (14-38) into (14-37) yields

$$C = f(V\frac{P_o}{P})\frac{T}{T_o} \tag{14-39}$$

where C is the CO_2 mole fraction in μmol mol^{-1}, and the constant K is included in the calibration function f; P_o = 101.3 kPa, T_o =273 K. Equation (14-39) is the fundamental LI-COR gas analyzer calibration function, and $f(x)$ takes the form of a polynomial. Note the difference in how temperature and pressure affect the calibration curve for a LI-COR gas analyzer: temperature serves as a concentration scaling factor, while pressure scales the raw voltage, much like a gain adjustment.

All gases are not equally effective in causing pressure broadening of absorption lines. The equivalent pressure P_e is defined as

$$P_e = P_{N_2} + \sum \alpha_i P_i + {}^b CO_2 \qquad (14\text{-}40)$$

where P_{N_2} is the partial pressure of nitrogen, and P_i gives the partial pressures of other diluent non-absorbing gases. The partial pressure of each non-absorbing gas is multiplied by a weighting factor α_i called the foreign gas broadening coefficient. The coefficients α_i reflect the ability of each diluent gas to cause pressure broadening relative to α_{N_2}; b is the self-broadening coefficient for the absorbing gas, and it gives the relative effect of its own partial pressure on absorption (Burch et al., 1962).

Broadening coefficients for the effect of various gases on CO_2 absorption at 4.26 μm are given in Table 14-11. For example, the equivalent pressure of a binary mixture that is 80% (v/v) nitrogen and 20% oxygen at a total pressure of 100 kPa is $P_e = 80\,(1) + 20\,(.81) = 96.2$ kPa.

Table 14-11. Typical foreign gas and self-broadening coefficients for the CO_2 4.26 μm absorption band (Burch et. al. 1962).

Gas	Broadening Co-efficient	Comment
N_2	1.00	Foreign Gasses
O_2	0.81	
H_2	1.17	
A	0.78	
He	0.59	
H_2O	?	
CO_2	1.30	Self Broadening

Equivalent pressure can be written in terms of mole fractions and total pressure. For air with dry gas mole fractions x_i, and water vapor mole fraction w,

$$P_e = P[1 + (\alpha_w - 1)w + \sum (\alpha_i - 1)x_i] \qquad (14\text{-}41)$$

x_{CO_2} is typically small (3.5×10^{-4}) for gas exchange measurement condi-

tions and is neglected. If we consider oxygen, water vapor, and nitrogen, then (14-41) becomes

$$P_e = P[1 + (\alpha_w - 1)w + (\alpha_O - 1)x_O]$$
$$= P\,c(w, x_O)$$

(14-42)

A calibration equation similar to (14-38) that includes the pressure broadening effects of variable water vapor can be obtained by substituting P_e for P in (14-33) and (14-34), and carrying through the subsequent steps to give

$$C = c(w, x_O)F\left[\frac{VP_0}{Pc(w, x_O)}\right]\frac{T}{T_O}$$

(14-43)

Finding appropriate values for α_w and α_O for use in (14-43) forms the basis of the pressure broadening correction for water vapor in LI-COR gas analyzers. We use empirically determined values of 1.5 and 0.9 respectively, and are contained in the configuration nodes

<open> <constants> <bb_vapor> α_w

<open> <constants> <bb_oxy> α_O

Thus, Equation (14-14) comes from

$$c(w, x_O) = 1 + (\alpha_w - 1)w + (\alpha_O - 1)x_O$$
$$= 1 + 0.5w - 0.1x_O$$
$$= 1 + 0.0005W - 0.001X_O$$

(14-44)

where W has units of mmol mol^{-1}, w is in mol mol^{-1}, X_O is in %, and x_O is in mol mol^{-1}.

Direct Cross Sensitivity

Direct cross sensitivity is what you have left after you have accounted for all the other possible effects of varying mole fractions in a mixture (i.e. dilution and band broadening). In a non-dispersive IR device like the LI-6400 analyzer, the potential for cross sensitivity arises from at least two potential sources: optics and electronics. On the optical side, filters might "see" a little beyond

where they are supposed to. In the electronics, there are several opportunities in the signal processing and measurement to have cross talk. Obviously, we design to minimize these effects, and in the LI-6400, they are quite small and can usually be ignored. But not always.

So, in May of 2010, we began measuring direct cross sensitivity for each unit during factory calibrations, and those parameters are part of the calibration. We have also included some on-board software (starting in version 6.1.4) to assist users in determining these coefficients for themselves, if they have need to do so. See **XSensitivity** on page 21-19.

Literature Cited

Burch, D. E., E. B. Singleton and D. Williams. 1962. Absorption line broadening in the infrared. *Applied Optics* 1: 359 - 363.

Jamieson, J. A., R. H. McFee, G. N. Plass, R. H. Grube and R. G. Richards. 1963. Infrared Physics and Engineering. McGraw-Hill, New York, 673 pp.

Wolfe, W. L. and G. J. Zissis. 1978. The Infrared Handbook. The Infrared Information and Analysis Center, Environmental Research Institute of Michigan.

Defining User Variables

15

....

Adding Your Own Equations

15 Defining User Variables

OPEN provides some methods of extending the quantities computed beyond the set of system variables described in the previous chapter. The user can define variables and constants that can, like the system variables, be viewed, logged and plotted.

User defined variables are controlled by the Configuration tree node <open> <comps> <file> and its subnodes, <header> and <extras>.

The ComputeList file, page 15-14

Allows customization. See **Hooking the Extras Module** on page 15-14

Additional items defined by the user
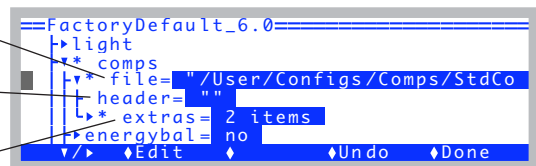
Figure 15-1. The <open> <comps> <file> node.

The bulk of the work is done by the ComputeList file (<file> node). There are come some ways of making modifications without actually having to touch the ComputeList: <extras> allows you to define more user items outside of the ComputeList, and <header> allows you to enter text to be compiled and processed independent of the ComputeList.

We'll start with the <extras> node, which is likely the place you'll want to start if you wish to add some variables.

Extras

Examples of Extras in operation can be found in **Add Water Use Efficiency** on page 3-100, in which a water use efficiency variable is added, and **Data for Real Time Measurements** on page 11-51, which shows how to incorporate variables measured with an external device and transmitted to the LI-6400 via RS-232.

Types of Extras

There are three basic types of Extras in OPEN 6.2:

Expressions (Normal and Post-fix)

This type of Extra item is computed from some combination of system and/or user variables, using an equation that you specify. Expressions can be written using in-fix notation (i.e. normal, algebraic), or in post-fix notation. The code for expressions (i.e, the equation) gets executed about two times per second while measurements are being made. Note: the code usually is an equation defining the variable, but it does not have to be confined to that. You could, for example, also drive a DAC (digital to analog output channel) based on photosynthetic rate, or output something to RS-232, etc.

Constants (Numeric and String)

A constant is something that the system is not going to try to update, like it does expressions. Rather, a constant Extra starts with some initial value that you specify, and there it stays until you change it. Constant Extras are prime candidates for inclusion in your list of prompts.

There is an alternative way to make constants, using one or more of the nine system variables reserved for that purpose (**Defining Prompts** on page 9-21). Extras provides a way to add an unlimited number of constants, should you be so inclined.

Comm port items (Numeric and String)

Comm port Extra items have values that are extracted from incoming RS-232 data. This data could be coming from a GPS device, or temperature/humidity instrument, or a gas analyzer. See **Data for Real Time Measurements** on page 11-51 for an example.

Adding an Extra

To add an Extra, go to the <open> <comps> <file> <extras> node, and edit it (Figure 15-2). You'll first encounter the Extras List Editor, which shows all

The Extras List Editor (empty).

Pick the type

Answer some questions

The Extras Item Editor.

Back at the List Editor, with one Extra now in the list.
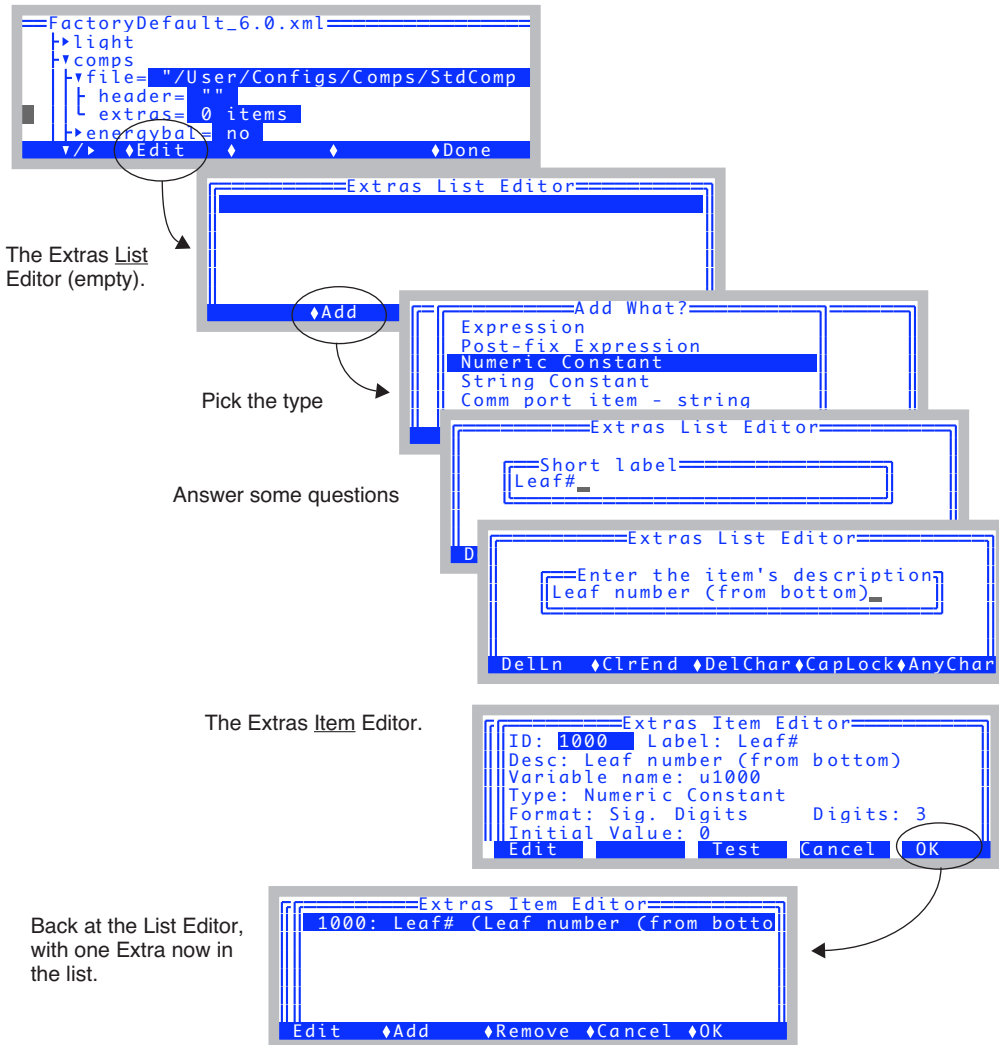
Figure 15-2. Adding an Extra.

of the Extras presently defined, if any. To add an item, press **Add** (**f2**). You'll be asked a few preliminary questions, and then taken to the Extras Item Editor (Figure 15-2 bottom). The preliminary questions just get you started - you can change anything from the Extras Item Editor, even the type.

The Extras Item Editor

The Extras Item Editor (Figure 15-3) is a dialog that lets you specify the attributes of an Extra item. Use the ↑ and ↓ arrows to move the highlighted field, and press **Edit** (**f1**) to change any attribute. It is a dialog, so **Cancel** (**f4**) undoes any changes you may have made, and **OK** (**f5**) keeps them.

The fields, or attributes, shown in the Extras Item Editor are only those relevant for the type (Figure 15-3). There are 12 possible attributes, and they are explained next.
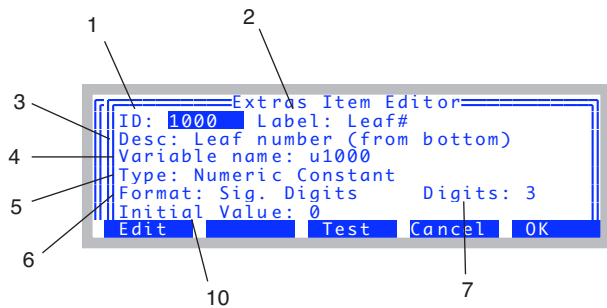
Figure 15-3. The attributes of a numeric constant.

Item Attributes

1   ID

This should be an integer > 0 that is not already used by another user variable (either in the current ComputeList, or already in use by another Extra item). System variables, by contrast, have ID values <= 0.

2   Label

A short string used for the label when and if you display this quantity in New Measurements mode, and for the label if you include the item in the log file.

3   Desc

The description is for your reference, and is only used for the "**What's What**" display in New Measurements mode (**f3** level 6), and (if it is a constant) for the prompting string. If you leave the description blank, you will be prompted with the Label instead.

4   Variable name

The variable name will default to *unnn*, where *nnn* is the ID number. You can edit it and make it whatever you like, but the safest thing is to leave it as the default. If you do edit it, and pick a name that is illegal or already in use, the program will modify it to make it legal. Caution: even with a legal name for the variable, there is a chance that it can cause problems later when you link some other module (e.g. change light source, run a utility program, etc.) that is using that name. This is because the Extra variable names are declared as PUB (public), and are accessible to all modules. This allows you to access them from an AutoProgram, for example. Therefore, leave the default variable name alone, and you'll be safe.

5   Type

When selecting type, pick from the list of possibilities (Figure 15-4).

```
╔════════════Select Type════════════╗
║ Expression                         ║
║ Post-fix Expression                ║
║ Numeric Constant                   ║
║ String Constant                    ║
║ Comm port item - string          3 ║
║ Comm port item - numeric           ║
║    ♦         ♦         ♦Cancel ♦OK  ║
╚════════════════════════════════════╝
```
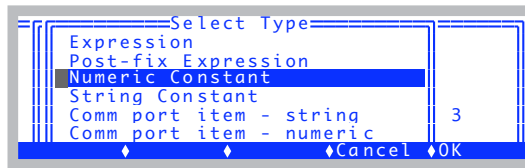
Figure 15-4. There are six types of Extras.

6   Format

This is used for the numeric types (expression, constants, or comm items), and can be one of Fixed Point, Significant Digits, or Scientific. Table 15-1 gives some examples.

7   Digits

Meaning depends on Format. Digits can be 0 to 9. See Table 15-1.

Table 15-1. The value of π shown by combinations of Format and Digits.

Digits:	Format:		
	Fixed Point	**Significant Digits**	**Scientific**
0	3	3	3E+00
1	3.1	3	3.1E+00
2	3.14	3.1	3.14E+00
:	:	:	:
9	3.141592654	3.14159265	3.141592654E+00

8   Code

(Expressions only) The code part of the expression. More explanation can be found in **Editing Code** on page 15-10.
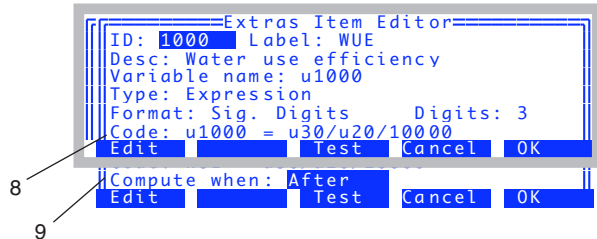
Figure 15-5. The attributes of an expression.

9   Compute when

(Expressions only). This specifies when the computations are to be performed in relation to the ComputeList items. The choice is Before or After. Computations are triggered when the analog-to-digital converter has new readings available, which is twice per second. The order of computations is shown below.

1. System variables
2. Extra Expressions (Before)
3. ComputeList
4. Extra Expressions (After)

10   Initial Value

(Numeric or string constants only. Figure 15-3). The initial value of the constant.

11   Screen width

(String items only. Figure 15-7). String items can be any length, but if one is displayed in New Measurements mode, the displayed length will be limited to 8, 18, 28, or 38 characters, which corresponds to 1/4, 1/2, and 3/4 of a full row on the display (Figure 15-6).

A String constant with
Screen width = 18.

Figure 15-6. An example of Screen width.

12   **Item # in comm string:**
(Comm items only). Comm items come from parsing incoming RS-232 lines.
Tabs, spaces, and commas are the delimiter characters (unless embedded in a
quoted field). This field determines which parsed item is to become the vari-
able, starting with 1. If you want the whole line with no parsing, specify 0.
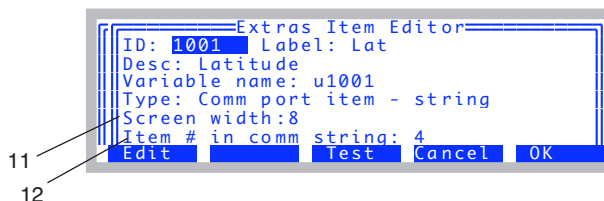
Figure 15-7. The seven attributes of a string comm port item.

For example, if the incoming data record looks like this:

        10:14:22 24.43 55.12

Item #1 as a string is "10:14:22", but as a numeric value is 10. Item #3 is
55.12. Item #0 as a string is "10:14:22 24.43 55.12".

The Test Key
The **Test** key (**f3**) lets you try out your variable (Figure 15-8). In the case of a
constant, you are prompted for it, then shown what it will look like in New
Measurements mode, according your current definition. If it is an expression,
the code is compiled and executed (and any errors reported), and then how it
will look in the New Measurements display is shown.

A) Testing a Numeric Constant

B) Testing a String Constant

C) Testing an Expression

Figure 15-8. Testing A) a numeric constant, using the specified format, digits, description, and initial value; B) a string constant, using screen width of 18; C) an expression, with a typing error so it won't compile.

Editing Code

When the code attribute is edited, the Code Editor dialog (Figure 15-9) lets you type in what you need, and also provides a variable name lookup feature.


```
=======Extras Item Editor========
ID: 1000   Label: WUE
Desc: Water use efficiency
Variable name: u1000
Type: Expression
Format: Sig. Digits     Digits: 3
Code: u1000 = u30/u20/10000
 Edit          Test  Cancel   OK
```

The Code Editor

```
========Code for u1000=========
u1000 = u30/u20/10000_




+DelLn   ◆ClrEnd ◆DelChar◆Cancel ◆OK
```

4 fct key levels

```
+VarName◆Find    ◆Re-find◆AnyChar◆OK
+BlkStar◆BlkEND ◆BlkCpy ◆BlkMve ◆BlkDel
+BlkRead◆BlkWrit◆        ◆       ◆OK
```
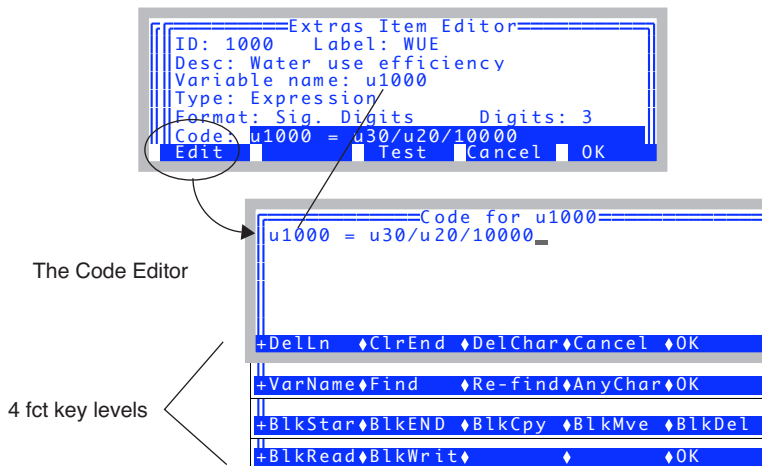
Figure 15-9. The Code Editor. Typically, you set the value of your variable here.

Normally, this code makes an assignment of whatever you are calculating to the variable you are calling this Extra item. Note that case doesn't matter in LPL (the language being used here). In the example above, the variable name is *Wue*, and the code refers to it as *wue*.

Variable names

In the water use efficiency example shown in Figure 15-9, the *u30* and *u20* are variable names, referring to two other user variables - photosynthetic rate and transpiration - defined in the ComputeList. The Code Editor provides a method for looking up variable names (Figure 15-10).
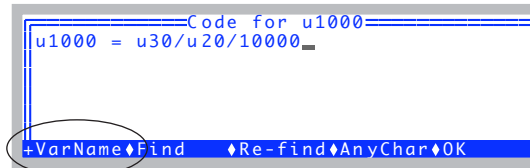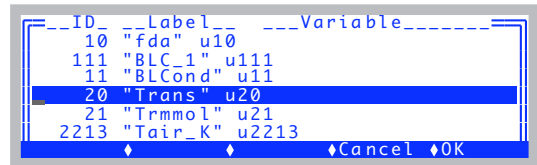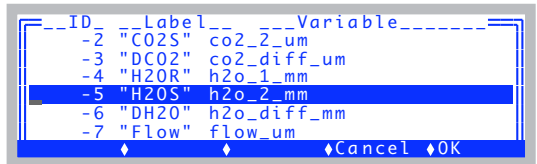
Figure 15-10. Use **VarName** *to insert the variable name for any system or currently defined user variable.*

Code Editor Short Cuts

The Code Editor has some short-cut keys (Table 15-2).

Table 15-2. Short-cut keys in the Code Editor.

Short-cut	Same As	Description
ctrl + y	**f1 (DelLn)**	Delete current line
ctrl + e	**f2 (ClrEnd)**	Clear from cursor to end of the line
ctrl + c	**f3 (DelChar)**	Delete next character
ctrl + v	**f6 (VarName)**	Lookup variable name and insert it
ctrl + f	**f7 (Find)**	Enter target, jump to next occurrence
ctrl + g	**f8 (Re-Find)**	Jump to next occurrence
ctrl + a	**f9 (AnyChar)**	Described in **AnyChar Routine** on page 5-7
ctrl + →		Shift right to start of next word
ctrl + ←		Shift left to start of previous word

Table 15-2. Short-cut keys in the Code Editor.

Short-cut	Same As	Description
shift + →		Page right
shift + ←		Page left
shift + home		Jump to start of current line
shift + end		Jump to end of current line

Some Simple Rules (in-fix notation)

The basic mathematical operators are + - * /. Raising to a power is done by a ^, and you can use parentheses to group operations. So, a sample assignment might be

```
xyz=(a+b)^2
```

Spaces don't matter, so the following is equivalent

```
xyz = (a + b)^2
```

Some Simple Rules (Post-fix notation)

Post-fix is stack oriented, and one doesn't need parentheses. Spaces DO matter: they have to be there. The post-fix equivalent of xyz = (a+b)^2 is

```
a b + 2 ^ &xyz =
```

Notice the assignment is done by putting the address of *xyz* on the stack (done by the & operator at the start of the name), then using the = operator.

The Extras List Editor

The Extras List Editor, accessed by editing the <open> <comps> <file> <extras> node (Figure 15-11) shows the list of currently defined Extras, and lets you edit, add, or remove any item. It is a dialog, so any changes can always be undone by pressing the **Cancel** key (**f4**).

```
╔══════════Extras List Editor══════════╗
║ 1000: WUE (Water use efficiency (%)) ║
║                                      ║
║                                      ║
║                                      ║
║                                      ║
║ Edit    ♦Add    ♦Remove ♦Cancel ♦OK  ║
╚══════════════════════════════════════╝
```
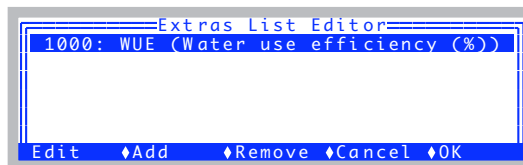
Figure 15-11. The Extras List Editor, with one item in the list.

Including Extras in a Log File, Prompts, and Displays

When you add items to the Extras List, and press **OK** (**f5**), you are prompted about whether you'd like the items added to the New Measurements display, LogList, and (in the case of constants) PromptList (Figure 15-12).

You are asked a se-
ries of Y/N questions
for each item you've
added.

```
╔══════════════Extras Item Editor═══════════════╗
║Add Plot# (ID#1000) to Display (Y/N)?           ║
║Y                                               ║
║Add Plot# (ID#1000) to LogList (Y/N)?           ║
║Y                                               ║
║Add Plot# (ID#1000) to PromptList (Y/           ║
║N)?                                             ║
╚════════════════════════════════════════════════╝
```

If you added some-
thing to the display,
you are told where it
was added.

```
Display Additions...
Plot# -> line m
Press any key
```

Figure 15-12. When you leave the Extras List Editor with OK after adding items, you are prompted about including the new items in the New measurements display, the lists of things to be logged and prompted.

One final note: from the configuration tree, you can get directly to the Extras Item Editor, if you wish (Figure 15-13).

```
═FactoryDefault_6.0═══════════════════════════
└▾* comps
 └▾* file=  "/User/Configs/Comps/StdCo
  ├ header=  ""
  └▾* extras=  1 items
    └ * extras[1]=  1000: WUE (Water us
 ├▾energybal=  no
 ▾/▸   ♦Edit   ♦        ♦Undo   ♦Done
```

Edit an <extras>
subnode

Edit the <extras> node

```
══════════Extras Item Editor═════════
║ID: 1000   Label: WUE              ║
║Desc: Water use efficiency (%)     ║
║Variable name: WUE                 ║
║Type: Expression                   ║
║Format: Sig. Digits    Digits: 3   ║
║Code: WUE = u30/u20/10000          ║
║ Edit        Test  Cancel   OK     ║
```

```
══════════Extras List Editor═════════
║ 1000: WUE (Water use efficiency (%))║
║                                     ║
║                                     ║
║ Edit   ♦Add    ♦Remove ♦Cancel ♦OK  ║
```

Add or Edit

Figure 15-13. Extras Editor Navigation. You can get to the Item Editor directly, or go through the List Editor.

Hooking the Extras Module

Extras are implemented by created a module written in LPL that can be linked to OPEN. This Extras Module is very much analogous to the ComputeList Module that will be discussed in the ComputeList section below. This module has a "hook", however that you can use to implement extended behavior, such as defining New Measurements function keys, that may have nothing to do with Extras. That hook is the text located in <open> <comps> <file> <header> is included in the Extras Module

The content of the <header> node is simply put at the beginning of the Extras module. Thus, it should be valid LPL, and can define whatever structures or objects you like.

The ComputeList File

A ComputeList is a file that contains a list of the variables and constants that you wish to define. It can take one of two forms:

- **List of User Variables ("List")**
  A text file that contains a list of user variables, showing their ID value, Label, Description, and Code. A piece of the default ComputeList (*StdComps_6.2*) is shown in Figure 15-15 on page 15-15. The whole thing is shown on page 15-37.

- **LPL Module ("Module")**
  A text file that contains LPL code that is able to be compiled and linked to the OPEN program. The Module form of `StdComps_6.2` is shown on page 15-39.

When the ComputeList is a List, a Module is built automatically when it is implemented. If the ComputeList is already a Module (as it is for the soil respiration configuration- see page 15-39), then that step is skipped.

Why two formats? The List format is simpler to deal with, and you don't need to know very much about LPL. It limits your flexibility, however. The Module provides great flexibility, but dealing directly with that does mean you have to know some things about LPL.

It is possible to start with a List and let the system build a Module, then modify the Module to suit your purposes. There is a utility program for turning a List into a Module (`"ComputeList List->Module..."`, under `"Files"` in the Utility Menu).

Whatever the format, ComputeList files are specified in the configuration tree

in the <open> <comps> <file> node (Figure 15-14).
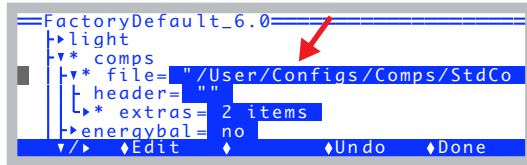
Figure 15-14. The ComputeList file declaration.

Format for Lists

A ComputeList file in List format has some simple rules. We illustrate by showing the first few variables of the default ComputeList, a file named "/User/Configs/Comps/StdComps_6.2". The complete listing is on page 15-37.


```
/* boundary layer

*/
##10 "fda" "flow / area with units conversion"
" (flow_um * 1E-6) / (area_cm2 * 1E-4) "

##111 "BLC_1" "One sided BLC"
" area_cm2 * blcSlope + blcOffset "

##11 "BLCond" "Effective BLC"
" #111 * (stom_rat + 1) * (stom_rat + 1) / (stom_rat * stom_rat + 1)"
```
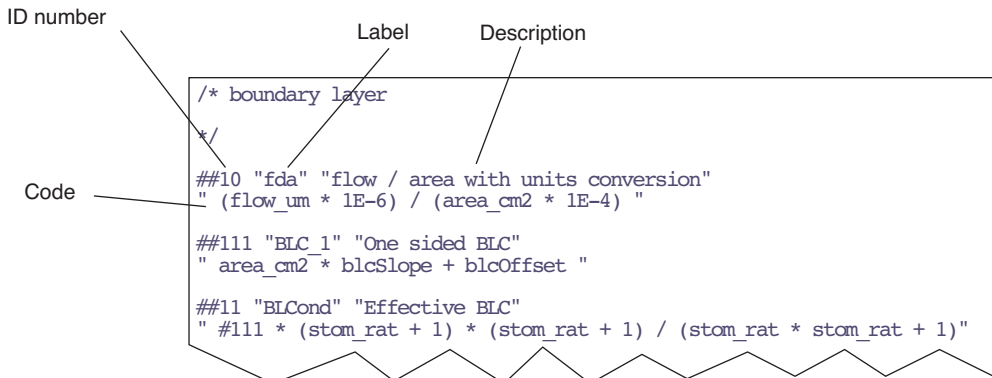
Figure 15-15. The beginning of StdComps_6.2.

In general, the format for each item consists of ## followed by a number, followed by three strings. There is an optional format code following the number.

```
##<ID number>[Fmt Code] <label> <description> <equation>
```

The ID number

The ## marks the start of a definition of a user variable, and is followed by a positive integer that serves as the ID number of that variable. This corresponds to the ID number in Extras (**ID** on page 15-5).

Format Code (optional)

Immediately following the ID number is the optional format code, which is a letter (F or G or S) and a digit (0 through 9). The default format for displaying and logging user computations is to use 3 significant digits (G3), resulting in values such as

```
1.23
0.0123
-1.23E+7
```

You can specify other formats by appending a format specifier to the ID string. The F code specifies a fixed format, and the number indicates the number of places to show to the right of the decimal. The G code specifies significant digits, as given by the number. The G format will use exponential notation if necessary (Table 15-3). S is for string items. See UREM and UCOMM on page 15-27.

Table 15-3. π, 1000π, and $\pi/1000$ expressed using the various format codes

Code	π	1000π	$\pi/1000$	Code	π	1000π	$\pi/1000$
F0	3.	3141.	0.	G1	3	3E3	3E-3
F1	3.1	3141.6	0.0	G2	3.1	3.1E3	3.1E-3
F2	3.14	3141.59	0.00	G3[a]	3.14	3.14E3	3.14E-3
F3	3.141	3141.592	0.003	G4	3.141	3.141E3	3.141E-3
F4	3.1416	3141.5927	0.0031	G5	3.1416	3.1416E3	3.1416E-3
F5	3.14159	3141.59265	0.00314	G6	3.14159	3.14159E3	3.14159E-3

a. Default.

Some examples of explicitly specifying a format code in *StdComps_6.2* can be seen in Figure 15-17 on page 15-19.

The Label String

The first of the three quoted strings following the ID/Format sequence is the label, which should be 8 characters or less. The variable's label is used to identify the variable in text, graphics, and file output. This corresponds to the label field in an Extra (**Label** on page 15-5).

The Description String

The second quoted string is a short description. This corresponds to the Desc: field in an Extra (**Desc** on page 15-5).

The Equation

The third string is the actual mathematical definition of the variable. This definition is given as an algebraic expression that can contain mathematical operators such as +, -, *, /, parentheses, system variables (referred to by name - see Table 14-10 on page 14-23). You can also refer to system and other user variables by *#n*, where *n* is the ID number. So, for example, *area_cm2* could also be written *#-33*, since that is the ID for area.

Note: This is similar to but not exactly like the code part of an Extra Expression (**Code** on page 15-7). One difference is that you cannot use the *#n* reference method in Extras Code strings. The actual variable names of items defined in the ComputeList is *un*, where *n* is the ID number. Thus, the variable name for *fda* above (item #10), would be *u10*. The relation between variable names and ID values doesn't necessarily hold for Extra expressions, since you explicitly define the variable name.

A bigger difference between code in a List file and code in an Extra expression is that the Extra code must explicitly assign the expression to the variable. In List equations, that assignment is implicit. Taking #10 again for an example, the code is

```
" (flow_um * 1E-6) / (area_cm2 * 1E-4) "
```

but what is actually implemented when the Module is created is

```
" $ u10 = (flow_um * 1E-6) / (area_cm2 * 1E-4) "
```

LPL functions are normally written in post-fix, but the $ toggles between post-fix and in-fix. Thus, the " $ u10 = " tells the compiler to switch to in-fix, and then assign what is left on the stack to variable *u10*.

There is a directive for this style ComputeList that you can include if for some reason you don't want this implicit assignment happening. It is NOASSIGN, discussed on page 15-25.

The transpiration section of StdComps_6.2 is shown in Figure 15-16.

```
/* transpiration          Note reference to other user variables by #
*/

##20 "Trans" "Transpiration (mol/m2/s)"
"(h2o_2_mm – h2o_1_mm) / (1000.0 – h2o_2_mm) * #10"

##21 "Trmmol" "Transpiration (mmol/m2/s)"
" #20 * 1E3"
```

Eqn 1-6 on page 1-8

Figure 15-16. Transpiration in StdComps_6.2.

The energy balance section of StdComps_6.2 is shown in Figure 15-17.

Note F1 display format
(e.g. 305.1)

Eqn 17-9 on page 17-3

```
/* energy balance deltaT
*/

##2213F1 "Tair_K" "air temp in K"
"tleaf_c + 273.15"

##2214F1 "Twall_K" "Twall temp K"
" tCham_c + 273.15"

##2216 "R(W/m2)" "incoming radiation"
" (parIn_um * f_parIn + parOut_um * f_parOut) * alphaK "

##2218 "Tl-Ta" "energy balance delta t"
"  (#2216 + 1.0773E-7 * ((#2214 ^ 4) - (#2213 ^ 4)) - #20 *
44100.0)/(#111 * 51.4 + 4.3092E-7 * (#2213 ^ 3)) "

/* leaf temp
*/
##221F2 "CTleaf" "Computed leaf temp"
" Tleaf_c + #2218 * doEB"
```

Figure 15-17. Energy balance implementation in StdComps_6.2.

The leaf conductance section of StdComps_6.2 is shown in Figure 15-18. Notice the definition of *CTair* (#225).

Eqn 1-8 on page 1-8

Eqn 1-7 on page 1-8

Eqn 1-9 on page 1-9

```
/* leaf conductance
*/
##222 "SVTleaf" "SatVap(Tleaf)"
" ( 0.61365 * EXP(17.502 * #221 / (240.97 + #221))) "

##223 "h2o_i" "intercellular h2o"
" #222 * 1000 / press_kPa "

##224 "h20diff" "diff"
" #223 – h2o_2_mm"

##225 "CTair" "Computed chamber air temp"
" $ doEB IF Tleaf_c ELSE Tcham_c Tleaf_c + 2 /  THEN "

##226 "SVTair" "SatVap(Tair)"
" ( 0.61365 * EXP(17.502 * #225 / (240.97 + #225))) "

##22 "CndTotal" "Total conductance"
" $ #224 0 <> IF 1000 #223 h2o_2_mm + 2 / – #224 / #20 * ELSE 0 THEN "

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" $ #22 0 <> IF 1.0 1.0 #22 / 1.0 #11 / – / ELSE 0 THEN "

##24 "vp_kPa" "vapor pressure chamber air"
" h2O_2_mm * press_kPa / 1000 "

##25 "VpdL" "Leaf VPD (SatVap(Tleaf) – eair)"
" #222 – #24"

##27 "VpdA" "Air VPD (SatVap(tair) – eair)"
" #226 – #24"
```

Figure 15-18. Leaf conductance section of StdComps_6.2.

CTair, the air temperature in the leaf chamber, is computed from either the leaf temperature channel (if we are doing energy balance, so the thermocouple is not touching a leaf), or else it is estimated to be the average of the chamber air temp thermistor (back in the IRGA) and the measured leaf temperature.

$$T_{ca} = \begin{cases} T_l & \text{if energy balance = yes} \\ \dfrac{T_l + T_c}{2} & \text{if energy balance = no} \end{cases} \qquad (15\text{-}1)$$

To implement this in the ComputeList, we drop into post-fix notation (with a $), and use an IF...ELSE...THEN formulation.

```
$ doEB IF Tleaf_c ELSE Tcham_c Tleaf_c + 2 /  THEN
```

The assignment is still implicit. This could have been done with in-fix notation by writing it this way:

```
NOASSIGN
IF(doEB)
   u225 = Tleaf_c
ELSE
   u225 = (Tcham_c + Tleaf_c)/2
THEN
```

There is an important reason why this was NOT done, and that is the Excel files option. See **Excel File Considerations** on page 15-34.

The photosynthesis section is shown in Figure 15-19.

Eqn 1-17 on page 1-10

Eqn 1-19 on page 1-11

Eqn 1-18 on page 1-11

```
/* photosynthesis
*/
##30 "Photo" "Photosynthesis (Êmol/m2/s)"
" (co2_1_um – co2_2_um * (1000 – h2o_1_mm) / (1000 – h2o_2_mm)) *
#10 "

##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / #11)"

##36 "Ci" "Intercellular CO2 (Êmol/mol)"
" ((#35 – #20/2) * co2_2_um – #30) / (#35 + #20/2)"

##38 "Ci_Pa" "Intercellular CO2 (Pa)"
" #36 * press_kPa * 1E–3"

##39 "Ci/Ca" "Intercellular CO2 / Ambient CO2"
" #36 / co2_2_um "
```

Figure 15-19. The photosynthesis section of StdComps_6.2.

The Ball-Berry section is shown in Figure 15-20.

```
/* ball berry
*/
##51 "RHsfc"  "Surface Humidity (%)"
" (1.0 - (#20 * press_kpa)/#222/#23) * 100"

##52 "C2sfc"  "Surface CO2 (Êmol/mol)"
" co2_2_um - #30 / (#11 / 1.35)"

##53 "AHs/Cs" "Ball-Berry parameter "
" #30 * #51 /100.0 / #52 "
```

Figure 15-20. The Ball-Berry section of StdComps_6.2.

List and Extras Code Comparison

It is instructive to compare how to implement a variable in a List and as an Extra. As an example, consider surface humidity (#51 in Figure 15-20), and how it could be done as an Extra (Figure 15-21).

Figure 15-21. Surface humidity as an Extra. Both compared in New Measurements mode.

Combining Multiple List Files

The ## marker can also be used in a List style ComputeList to include multiple files. The format is

```
##"file name"
```

For example, the standard ComputeList file for the Leaf Chamber Fluorometer (/User/Configs/Comps/StdFlrComp_6.2) looks like this:

```
##"/User/Configs/Comps/StdComps 6.2"
##"/User/Configs/Comps/FlrOnly"
```

It joins two List files that contain ## definitions. The first file is the default ComputeList, and the second contains the ## definitions for fluorometry.

The included file will be inserted wherever the *##filename* occurs in the file. If must occur where a ## command is valid, however. You can't, for example, include a file as part of the equation in a ## definition.

Modifying ComputeList Files

Edit the configuration node <open> <comps> <file> (Figure 15-22).

```
══FactoryDefault_6.0.xml═══════════════════════
  ┣►li┌═ComputeList Options═┐
 ┣▼*  │ E - Edit this file  │
 ┃▼*┤ │ P - Pick a new file │mps/StdCo
 ┗,   │                     │
  ┣►  └═══════(P/E)═════════┘
  ┣►energybal= no
 ▼/►  ♦Edit    ♦         ♦Undo   ♦Done
```

If the file is protected, you'll get this message before editing.

P

E

```
Dir: /User/Configs/Comps
   ═══Select a ComputeList file═══
  MyComputeList
 ▌StdComps_6.0
  StdFlrComp_6.0
  StdSoilEqns_6.0.LPL
+ Dir    ♦        ♦       ♦ CANCEL♦ SELECT
```

This one is already
in Module form

The Code Editor, described in
Editing Code on page 15-10

```
Note: The current ComputeList file

/User/Configs/Comps/StdComps_6.0

is write protected. If you wish to make
changes, you'll have to save it under
a different name.
```

```
══════════StdComps_6.0═══════════
║/* boundary layer
║*/
║##10 "fda" "flow / area with units comv
" (flow_um * 1E-6) / (area_cm2 * 1E-4)
║
║##111 "BLC_1" "One sided BLC"
+DelLn   ♦ClrEnd ♦DelChar♦Cancel ♦OK
```

```
Your Options:

E - Keep editing
S - Save as different file
Q - Abandon changes
```

If you have made and saved changes,
you have the option of switching to that
new file, or staying with the old one.

```
Dir: /User/Configs/Comps
File:MyComputeList
   ═══════Store ComputeList File═══════
 ║ .DS_Store
 ║ Closed Mode.LPL
 ║ FlrOnly
 ║ InsectRD
+ Dir    ♦        ♦        ♦ CANCEL♦ SELECT
```

```
Make file

/User/Configs/Comps/MyComputeList

the new ComputeList? (Y/N)
```

*Figure 15-22. Changing a ComputeList, by picking
a new one, or editing the current one.*

When you edit List or Module ComputeList files, there are some special reserved words that you can use:

NOASSIGN

(List format only). When OPEN converts a List to a Module, each user variable that you define causes a variable named *un* to be created (where *n* is the ID number of your variable. Thus, *u20* is the variable name of #20). The code that is generated will have a line for each variable assigned to its equation string. Thus, if the equation string for ##20 was `"10 * #15"`, the line for *u20* in the Module would appear as in Figure 15-23.

```
                          u20 = 10 * u15


              System adds this…   …to your equation
```
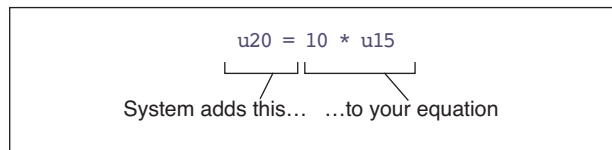
Figure 15-23. When building the code for each user variable, the system makes the equation by inserting the variable name and an = sign before your equation.

Note that "#15" becomes "u15". You can write it either way.

If you don't want this automatic assignment taking place, it can be suppressed with the word NOASSIGN. Suppose, for example, we wanted a variable that indicated 1 for photosynthesis, or 0 for respiration (photosynthesis < 0). You could define it this way:

```
##102 "Flag" "1 if photo, 0 if resp"
" NOASSIGN
 if (#30 > 0)
u102 = 1
ELSE
u102 = 0
THEN "
```

The word NOASSIGN in the first line of the equation string will prevent the program from putting a "U102 =" in front of our equation, so we'll have to do it ourselves, and we did. Alternatively, we could write the expression in postfix as

```
##102 "Flag" "1 if photo, 0 if resp"
" NOASSIGN
 $ #30 0 > if 1 else 0 then &u102 =
```

Another example would be making a variable that never gets assigned. Instead, we use the equation to do something totally unrelated, such as write a message to the RS-232 port.

```
##999 "dummy" "nothing"
```

```
"NOASSIGN
 PRINT( tair_c, "Tair = %6.2f\n", comm) "
```

This will cause `"Tair = 23.34"` (or whatever the air temperature really is), followed by a line feed to be written to the comm port every time user variables are computed.

NOASSIGN has a second effect: it prevents the equation from being "test compiled" when a List is made into a Module. During test compilation, each variable is listed on the display followed by an "ok" or an error. This usually flies by pretty fast, unless there is an error message. If you make a syntax error in an equation string with a NOASSIGN, you won't know it until the LPL module that is being generated fails to link.

ASSIGN()
(Deprecated in Version 6.) For List and Module Formats. The Assign() function assigns (i.e. does the = operation) during New Measurements mode, but not during recomputations done using the Utility Menu recompute function.

UCON()
You can create a numeric constant in a List file by using UCON(). This is equivalent to the numeric constant in Extras. A numerical user constant can be defined by using the function

```
UCON(0)
```

in the equation definition string. For example, Figure 15-24 defines two constants, named "Plot#" and "Rate".

```
##7001F0 "Plot#" "Plot ID Number" "UCON(0)"

##7002 "Rate" "Fert rate (lbs/acre)" "UCON(50)"

##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"
```

Figure 15-24. Two user defined constants with ID's 7001 and 7002. Since Plot# is to be an integer value, we specify a format of F0 so it will be displayed that way.

The argument of the UCON function is the initial value of the user constant.

In version 6.2, you can do UCON(0,1) or UCON(0,2). 1 will automatically include the constant in your Prompts List, and 2 will exclude it.

UREM()

You can create a string constant in a List with UREM(). (This is equivalent
to the string constant type in Extras). The string length is not restricted, but if
you want to specify a display screen width (8, 18, 28, or 38), do it by putting
an S code after the ID number.

```
##415S18 "MyLabel" "whatever"
" UREM() "
```

UCOMM()

The macro function for List formats

```
UCOMM(n)
```

will retrieve the nth item from the latest incoming line of RS-232 data the in-
strument has received while in New Measurements mode. This is equivalent
to an Extra comm item. UCOMM(1) retrieves the first item, according to
parsing rules explained below, UCOMM(2) gets the second item, and so on.
To get the whole line with no parsing, use UCOMM(0).

Suppose we have a hypothetical GPS device that outputs a stream of data reg-
ularly, and wish to use the 2nd (longitude), 3rd (latitude) and 5th (status)
items in each record as variables in the LI-6400. Further, suppose a data
record looks like this:

```
0,-90.46573,40.37865,11991,0010010,8765.332
```

We need to add three variables. We'll want latitude and longitude to be dis-
played with enough resolution, so we'll use a format specifier of "F5" for
both (fixed point, 5 decimal places). Let's say our status variable has mean-
ingful 0's at the start or end, or might contain letters instead of numbers.
We'll want to treat it as a string, instead of a floating point value. This is done
by using the "S" format specifier, as shown in Figure 15-25.

```
##876F5 "Long." "Longitude" "UCOMM(2)"

##875F5 "Lat." "Latitude" "UCOMM(3)"

##874S8 "XStat" "Status of the device"
"UCOMM(5)"
```

*Figure 15-25. Three user variables are added to hold three fields from incoming
RS-232 records. Long and Lat are floating point variables, and XStat will be a
string, display length of 8.*

CommGet()

CommGet() is the corresponding Module function for the List macro UCOMM(). The code section of the Module that corresponds to Figure 15-25 is shown in Figure 15-26.

```
CommGet(&u876, 2)
CommGet(&u875, 3)
CommGet(&874, 5)
```

Figure 15-26. The Module version of UCOMM.

Comments

If you wish to put comments in the ComputeList file, put them *between* variable definitions. This works because once an equation definition string is read, the program searches ahead looking for the next ## to start the next definition. This means, of course, that you should *not* put a ## within your comment.

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

This is a comment,
and can go on for as
long as you'd care to type...

##20 "Trans" "Transpiration (mol/m2/s)"
" #10*(h2o_2_mm - h2o_1_mm)/(1000.0 - h2o_2_mm)"
```

Figure 15-27. If you need comments, put them between ## definitions.

In a Module, comments must be between /* and */.

Compilation Errors

If you make a mistake defining a variable, you may get a message to that effect when it is compiled. While a List is being compiled, each variable's label is printed to the display, followed by an "ok" if it compiles correctly[1]. If there is a problem, however, you will get a message describing the problem. That variable will then not be available for subsequent operations. EXAMPLE: If the definition for Trmmol (#20) is changed to

```
##21 "Trmmol" "Transpiration (mmol/m2/s)"
```

---

[1]·This doesn't happen when using :include (**Open's Hooks** on page 26-14) or NOASSIGN.

```
" #20 * xyz"
```

The symbol *xyz* will not be found, so when *Trmmol* is compiled, an error will be reported (Figure 15-28).

```
Trans – ok
Trmmol – ERROR: Unidentified 'xyz'
Object: FCT recompile
Module: "/sys/open/clfconvert"
Press Any Key
```

Figure 15-28. Example error message, when the unknown variable xyz *is put into the defining equation for* Trmmol.

User variables that fail to compile will always have a value of 999 when you view them later. Thus, if you do the above example, then go to New Measurements mode, the Trmmol value will be 999.

If we modify the above example like this

```
##21 "Trmmol" "Transpiration (mmol/m2/s)"
" NOASSIGN
 u20 = #20 * xyz"
```

to prevent the test compile, the problem with xyz being undefined will cause the entire Module to fail to link. The message will look like Figure 15-29. No user defined quantities will be available. Either edit the ComputeList and fix the problem, or select another ComputeList.

```
══════════════ Errors ══════════════
ERROR: Undefined 'xyz'
Object: FCT ComputeUserValues
Module: "/User/Configs/Comps/MyComputeLi

   View    ◆        ◆Debug   ◆        ◆quit
```

Figure 15-29. When a module generated from a ComputeList fails to link to open, a message such as this is shown. No user defined variables will be available.

Format for Modules

Figure 15-30 shows a condensed view of the Module form of StdComps_6.2.

```
/*
  File = /User/Configs/Comps/StdComps_6.2
*/
:STATIC 1
:CHAR s8[] "%8s"
:STATIC 0

:DOUBLE u10 0.000000
u111 0.000000
u11 0.000000
u20 0.000000
u21 0.000000
  :
u36 0.000000
u38 0.000000
u39 0.000000
u51 0.000000
u52 0.000000
u53 0.000000
:PTR userList[]
{
    :PTR { 10 "fda" s8  u10 g83 "flow / area with units conversion" 0 g13  }
    :PTR { 111 "BLC_1" s8  u111 g83 "One sided BLC" 0 g13  }
    :PTR { 11 "BLCond" s8  u11 g83 "Effective BLC" 0 g13  }
    :PTR { 20 "Trans" s8  u20 g83 "Transpiration (mol/m2/s)" 0 g13  }
    :PTR { 21 "Trmmol" s8  u21 g83 "Transpiration (mmol/m2/s)" 0 g13  }
  :
    :PTR { 36 "Ci" s8  u36 g83 "Intercellular CO2 (Êmol/mol)" 0 g13  }
    :PTR { 38 "Ci_Pa" s8  u38 g83 "Intercellular CO2 (Pa)" 0 g13  }
    :PTR { 39 "Ci/Ca" s8  u39 g83 "Intercellular CO2 / Ambient CO2" 0 g13  }
    :PTR { 51 "RHsfc" s8  u51 g83 "Surface Humidity (%)" 0 g13  }
    :PTR { 52 "C2Sfc" s8  u52 g83 "Surface CO2 (Êmol/mol)" 0 g13  }
    :PTR { 53 "AHs/Cs" s8  u53 g83 "Ball-Berry parameter " 0 g13  }
}
:FCT ComputeUserValues
{ $
 u10 =  (flow_um * 1E-6) / (area_cm2 * 1E-4)
 u111 =  area_cm2 * blcSlope + blcOffset
 u11 =  u111 * (stom_rat + 1) * (stom_rat + 1) / (stom_rat * stom_rat + 1)
 u20 = (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm) * u10
 u21 =  u20 * 1E3
 :
 u36 =  ((u35 - u20/2) * co2_2_um - u30) / (u35 + u20/2)
 u38 =  u36 * press_kPa * 1E-3
 u39 =  u36 / co2_2_um
 u51 =  (1.0 - (u20 * press_kpa)/u222/u23) * 100
 u52 =  co2_2_um - u30 / (u11 / 1.35)
 u53 =  u30 * u51 /100.0 / u52
}
```

Declaration Section

List of user variables, and attributes

Computation Section

Figure 15-30. An abbreviated view of the Module version of StdComps_6.2.

The Module consists of a declaration section, in which all user variables are defined. There is a computational section, which is the function (named *ComputeUserValues*) that is called each time user values are to be computed. The middle section, the declaration of a pointer array named *userList*, requires a bit of explanation.

The *userList* Pointer Array

The pointer array named *userList* contains one object for each user variable or constant that is defined. Each item in the pointer array is itself a pointer that must contain 8 items (Figure 15-31), and optionally a 9th item (Table 15-4).
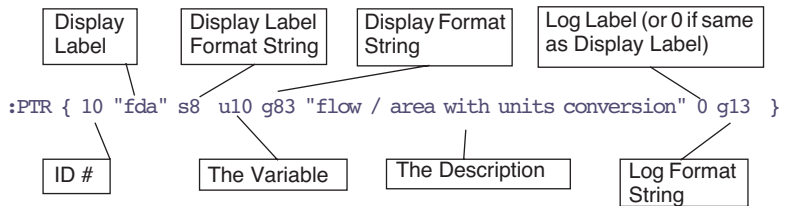
Figure 15-31. The elements of each :PTR array that make up userList[ ].

Table 15-4. Descriptions Of The Elements In Each *userList[ ]* Entry

#	Name	Description
1	ID#	Any integer 1... 2^{31}
2	Display Label	This should be no more characters than you wish the display length to be (typically 8).
3	Display Label Format String	For New Measurements mode. Typically something like "%8s". This is the mechanism by which a short label is made to take up the requisite number of spaces.
4	The Variable	The variable name. Must be defined somewhere, as a :FLOAT, or :INT, etc.

Table 15-4. (Continued) Descriptions Of The Elements In Each *userList[ ]* Entry

#	Name	Description
5	Display Format String	The format string that determines how the value of the variable is displayed in New Measurements mode. This is typically a string like "%8.1f" or "%8.3G", etc. The variable g83 is defined in OPEN.DF2 and is the string "%8.3G".
6	Description	Any descriptive (or vague and confusing, if you prefer) string.
7	Log Label	The label that will be printed in the log file, if this variable is logged. If you wish to use the display label for this, just enter a 0 instead of a string.
8	Log Format String	The format you wish to use for the variable if it is logged. Note that while you may display a number with something like "%8.1f", you may not want to consume 8 spaces all the time when you log, so you would use the corresponding "%1.1f".
9	(Optional)	Can be nothing, or 1, or 2: 1. A user defined constant, that <u>should be</u> included automatically in the active Prompt List. 2. A user defined constant that <u>should not be</u> automatically included in the active Prompt List.

Some format strings are defined in OPEN.DF2, since they are used in the system variable definitions (system variables are defined there with a :PTR array similar to *userList*). The actual C-style format string used in the Module for display and logging is shown in Table 15-5 below.

Table 15-5. Relation Between List Codes and Module format strings[a]

List File Display Code	Display Format		Logging Format	
	Variable Name	String	Variable Name	String
F0	f80	"%8.0f"	f10	"%1.0f"
F5	f85	"%8.5f"	f15	"%1.5f"
G1	g81	"%8.1f"	g11	"%1.1G"
G6	g86	"%8.6f"	g16	"%1.6G"

a. Not all are shown, but enough to give you the idea.

Converting Lists to Modules

There is a utility program under Files in the Utility Menu that will convert a ComputeList in List format into one in Module format (Figure 15-32).

```
┌─Utility Menu═══════════════════════════
│       Access the Filer...
│       Create a new AutoProgram...
│       Create a new (empty) file...
│       Recompute a stored data file...
│       Reinstall basic config files...
│       ComputeList List->Module...
│ ▼/▶    ♦         ♦         ♦ EXIT  ♦ SELECT
```

1. Pick a List type file to convert.

```
Dir: /User/Configs/Comps
┌═══════════Select ComputeList═══════════
│ jonComps_6.0
│ MyComputeList
│ StdComps_6.0
│ StdFlrComp_6.0
│ StdSoilEqns_6.0.LPL
│+ Dir    ♦         ♦         ♦ CANCEL♦ SELECT
```

2. Press Y if you want to view the Module.

```
Ci - ok
Ci_Pa - ok
Ci/Ca - ok
RHsfc - ok
C2Sfc - ok
AHs/Cs - ok

View/Edit result? (Y/N)_
```

3. escape to quit viewing.

```
/*
  Converted from

  /User/Configs/Comps/MyComputeList

  Thr Oct  2 2008 16:29:52
*/
```

4. Press Y if you want to save the Module.

```
Store result? (Y/N)_
```

5. If saving, enter a name. The LPL suffix is the convention, but it is not necessary.

```
Dir: /User/Configs/Comps
File:MyComputeList.LPL
┌═══════════Store ComputeList═══════════
│ .DS_Store
│ Closed Mode.LPL
│ FlrOnly
│ InsectRD
│+ Dir    ♦         ♦         ♦ CANCEL♦ SELECT
```

Figure 15-32. Utility program for converting ComputeList List files into a linkable LPL Module.

Another method to get a Module is from the file /User/Configs/.clf. Whenever OPEN implements a ComputeList that is in List format, it writes the Module version to /User/Configs/.clf. If you wanted to capture that to use as a starting point for a customized Module, just move it to /User/Configs/Comps and re-name it.

Excel File Considerations

The introduction of the Excel file generation option in version 6 brought with it some constraints on what could be done in ComputeLists. The binary Excel files generated meet Excel version 2 standards, which is the earliest standard that is documented in OpenOffice.org documentation. It has minimal features, but is compatible with all subsequent versions of Excel.

Limited function calls

A look at StdComps_6.2 shows that saturation vapor pressure is computed twice - once for leaf temperature (#222) and once for air temperature (#226). There is a saturation vapor pressure function defined in OPEN, but it can't be used here, because it will not translate into Excel version 2. Thus, the equation is explicit in both locations, and no function calls are made. Otherwise, #222

```
##222 "SVTleaf" "SatVap(Tleaf)"
" ( 0.61365 * EXP(17.502 * #221 / (240.97 + #221))) "
```

could have been written as

```
##222 "SVTleaf" "SatVap(Tleaf)"
" SatVap(#221) "
```

What function calls can be made? Only those in Table 15-6.

Table 15-6. LPL functions supported in Excel version 2.

Function	Description
ABS	absolute value
ATAN	Four quadrant tangent
CHS	change sign
COS	cosine
EXP	*e* raised to a power
LGT	log to base 10
LOG	natural logarithm

Table 15-6. LPL functions supported in Excel version 2.

Function	Description
MOD	Modulus
SIN	sine
SQRT	square root
TAN	tangent

No multiple assignments

This is a constraint of the LPL engine that converts a function into Excel equations. When you assign a value to a variable, it has to be in one and only one place. If you have code that looks like this

```
IF (a != 0)
  u21 = x*y/a
ELSE
  u21 = 0
THEN
```

it will work just fine as far as LPL and OPEN are concerned, but it will not translate into the Excel file correctly, because the assignment (u21 =) is happening in two places. If you need this structure, do it in post-fix notation:

```
$ a 0 != IF x y * a / ELSE 0 THEN &u21 =
```

since now there is only one assignment for u21.

No local variables

LPL functions have local variables. However, don't use them if you want things to work correctly in Excel.

◼ **Before and After Example**

Figure 15-33 shows a function before version 6 that would have problems making a correct Excel data file, and the Excel-friendly replacement.

Version 5.3

```
:FCT ComputeUserValues
{
  0 :FLOAT density  ─────┐
  0 :FLOAT va       ─────┴──→ Local variables
  $                                              ──→ Function call
  rhsc = 100.0 * eAir_2_kPa / SatVap(Tleaf_c)
  tSoil = chan21_mv / -10.0
  soilChamSysVol = soilChamBaseVol - soilArea * soilInsDepth
  va = soilChamSysVol / soilArea

  IF (opMode == 3)
    density = press_kPa * 1.2028 / (tleaf_c + 273)
    dcdt = density * (co2Slope + co2Mean * h2oSlope / (1000.0 - h2oMean))
  THEN
  IF (opMode == 4)
    dcdt = dcdtSlope * co2Mean + dcdtOffset  ────── Multiple assignments of dcdt
  THEN
  soilEfflux = dcdt * va
}
```

Version 6.0

```
:FCT ComputeUserValues
{
  $1
  tsch_C = tLeaf_c
  satVapTsch = 0.61365 * EXP(17.502 * tsch_C / (240.97 + tsch_C))

  rhsc = 100.0 * eAir_2_kPa / satVapTsch
  tSoil = chan21_mv / -10.0
  soilChamSysVol = soilChamBaseVol - soilArea * soilInsDepth

  $0 /* post fix to make the dcdt formula excel-friendly  (the if statement) */
  opMode 3 == IF
    press_kPa 1.2028 * tsch_C 273 + / h2oSlope 1000.0 h2oMean - / co2Mean * co2Slope + *
  ELSE
    opMode 4 == IF
      dcdtSlope co2Mean * dcdtOffset +
    ELSE
      0                                    ────── Just one assignment of dcdt
    THEN
  THEN
  &dcdt = ◄──────
  $1
  soilEfflux = dcdt * soilChamSysVol / soilArea
}
```

Figure 15-33. The ComputeUserValues function from the soil respiration ComputeList Module, before (version 5.3) and after it was made Excel-friendly.

The Default ComputeList

The default ComputeList, /User/Configs/Comps/StdComps_6.2, is shown below in two forms:

Listing of StdComps_6.2, List Form

```
/* boundary layer

*/

##10 "fda" "flow / area with units conversion"

" (flow_um * 1E-6) / (area_cm2 * 1E-4 ) "

##111 "BLC_1" "One sided BLC"

" area_cm2 * blcSlope + blcOffset "

##11 "BLCond" "Effective BLC"

" #111 * (stom_rat + 1) * (stom_rat + 1) / (stom_rat * stom_rat + 1)"

/* transpiration

*/

##20 "Trans" "Transpiration (mol/m2/s)"

"(h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm) * #10"

##21 "Trmmol" "Transpiration (mmol/m2/s)"

" #20 * 1E3"

/* energy balance deltaT

*/

##2213F1 "Tair_K" "air temp in K"

"tLeaf_c + 273.15"

##2214F1 "Twall_K" "Twall temp K"

" tCham_c + 273.15"

##2216 "R(W/m2)" "incoming radiation"

" (parIn_um * f_parIn + parOut_um * f_parOut) * alphaK "

##2218 "Tl-Ta" "energy balance delta t"

" (#2216 + 1.0773E-7 * ((#2214 ^ 4) - (#2213 ^ 4)) - #20 * 44100.0)/(#111
* 51.4 + 4.3092E-7 * (#2213 ^ 3)) "

/* leaf temp

*/
```

```
##221F2 "CTleaf" "Computed leaf temp"
" Tleaf_c + #2218 * doEB"
/* leaf conductance */
##222 "SVTleaf" "SatVap(Tleaf)"
" ( 0.61365 * EXP(17.502 * #221 / (240.97 + #221))) "
##223 "h2o_i" "intercellular h2o"
" #222 * 1000 / press_kPa "
##224 "h20diff" "diff"
" #223 – h2o_2_mm"
##225 "CTair" "Computed chamber air temp"
" $ doEB IF Tleaf_c ELSE Tcham_c Tleaf_c + 2 /  THEN "
##226 "SVTair" "SatVap(Tair)"
" ( 0.61365 * EXP(17.502 * #225 / (240.97 + #225))) "
##22 "CndTotal" "Total conductance"
" $ #224 0 <> IF 1000 #223 h2o_2_mm + 2 / – #224 / #20 * ELSE 0 THEN "
##23 "Cond" "Stomatal cond. (mol/m2/s)"
" $ #22 0 <> IF 1.0 1.0 #22 / 1.0 #11 / – / ELSE 0 THEN "
##24 "vp_kPa" "vapor pressure chamber air"
" h2O_2_mm * press_kPa / 1000 "
##25 "VpdL" "Leaf VPD (SatVap(Tleaf) - eair)"
" #222 – #24"
##27 "VpdA" "Air VPD (SatVap(tair) - eair)"
" #226 – #24"
/* photosynthesis */
##30 "Photo" "Photosynthesis (\xe6mol/m2/s)"
" (co2_1_um – co2_2_um * (1000 – h2o_1_mm) / (1000 – h2o_2_mm)) * #10 "
##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / #11)"
##36 "Ci" "Intercellular CO2 (\xe6mol/mol)"
" ((#35 – #20/2) * co2_2_um – #30) / (#35 + #20/2)"
```

```
##38 "Ci_Pa" "Intercellular CO2 (Pa)"

" #36 * press_kPa * 1E-3"

##39 "Ci/Ca" "Intercellular CO2 / Ambient CO2"

" #36 / co2_2_um "

/* ball berry */

##51 "RHsfc"   "Surface Humidity (%)"

" (1.0 - (#20 * press_kpa)/#222/#23) * 100"

##52 "C2sfc"   "Surface CO2 (\xe6mol/mol)"

" co2_2_um - #30 / (#11 / 1.35)"

##53 "AHs/Cs" "Ball-Berry parameter "

" #30 * #51 /100.0 / #52 "
```

Listing of StdComps_6.2, Module Form

```
/*
  File = /User/Configs/Comps/StdComps_6.2
*/
:STATIC 1
:CHAR s8[] "%8s"
:STATIC 0

:CHAR

:DOUBLE
PUB u10 0.000000
PUB u111 0.000000
PUB u11 0.000000
PUB u20 0.000000
PUB u21 0.000000
PUB u2213 0.000000
PUB u2214 0.000000
PUB u2216 0.000000
PUB u2218 0.000000
PUB u221 0.000000
PUB u222 0.000000
PUB u223 0.000000
PUB u224 0.000000
PUB u225 0.000000
PUB u226 0.000000
PUB u22 0.000000
PUB u23 0.000000
PUB u24 0.000000
PUB u25 0.000000
PUB u27 0.000000
PUB u30 0.000000
PUB u35 0.000000
PUB u36 0.000000
PUB u38 0.000000
```

```
PUB u39 0.000000
PUB u51 0.000000
PUB u52 0.000000
PUB u53 0.000000

:PTR userList[]
{
   :PTR { 10 "fda"  s8  u10 g83 "flow / area with units conversion" 0 g13  }
   :PTR { 111 "BLC_1" s8  u111 g83 "One sided BLC" 0 g13  }
   :PTR { 11 "BLCond" s8  u11 g83 "Effective BLC" 0 g13  }
   :PTR { 20 "Trans" s8  u20 g83 "Transpiration (mol/m2/s)" 0 g13  }
   :PTR { 21 "Trmmol" s8  u21 g83 "Transpiration (mmol/m2/s)" 0 g13  }
   :PTR { 2213 "Tair_K" s8  u2213 f81 "air temp in K" 0 f11  }
   :PTR { 2214 "Twall_K" s8  u2214 f81 "Twall temp K" 0 f11  }
   :PTR { 2216 "R(W/m2)" s8  u2216 g83 "incoming radiation" 0 g13  }
   :PTR { 2218 "Tl-Ta" s8  u2218 g83 "energy balance delta t" 0 g13  }
   :PTR { 221 "CTleaf" s8  u221 f82 "Computed leaf temp" 0 f12  }
   :PTR { 222 "SVTleaf" s8  u222 g83 "SatVap(Tleaf)" 0 g13  }
   :PTR { 223 "h2o_i" s8  u223 g83 "intercellular h2o" 0 g13  }
   :PTR { 224 "h20diff" s8  u224 g83 "diff" 0 g13  }
   :PTR { 225 "CTair" s8  u225 g83 "Computed chamber air temp" 0 g13  }
   :PTR { 226 "SVTair" s8  u226 g83 "SatVap(Tair)" 0 g13  }
   :PTR { 22 "CndTotal" s8  u22 g83 "Total conductance" 0 g13  }
   :PTR { 23 "Cond" s8  u23 g83 "Stomatal cond. (mol/m2/s)" 0 g13  }
   :PTR { 24 "vp_kPa" s8  u24 g83 "vapor pressure chamber air" 0 g13  }
   :PTR { 25 "VpdL" s8  u25 g83 "Leaf VPD (SatVap(Tleaf) - eair)" 0 g13  }
   :PTR { 27 "VpdA" s8  u27 g83 "Air VPD (SatVap(tair) - eair)" 0 g13  }
   :PTR { 30 "Photo" s8  u30 g83 "Photosynthesis (\xe6mol/m2/s)" 0 g13  }
   :PTR { 35 "CndCO2" s8  u35 g83 "Total Conductance to CO2" 0 g13  }
   :PTR { 36 "Ci" s8  u36 g83 "Intercellular CO2 (\xe6mol/mol)" 0 g13  }
   :PTR { 38 "Ci_Pa" s8  u38 g83 "Intercellular CO2 (Pa)" 0 g13  }
   :PTR { 39 "Ci/Ca" s8  u39 g83 "Intercellular CO2 / Ambient CO2" 0 g13  }
   :PTR { 51 "RHsfc" s8  u51 g83 "Surface Humidity (%)" 0 g13  }
   :PTR { 52 "C2sfc" s8  u52 g83 "Surface CO2 (\xe6mol/mol)" 0 g13  }
   :PTR { 53 "AHs/Cs" s8  u53 g83 "Ball-Berry parameter " 0 g13  }
}


:FCT ComputeUserValues
{ $
 u10 =  (flow_um * 1E-6) / (area_cm2 * 1E-4 )
 u111 =  area_cm2 * blcSlope + blcOffset
 u11 =  u111 * (stom_rat + 1) * (stom_rat + 1) / (stom_rat * stom_rat + 1)
 u20 = (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm) * u10
 u21 =  u20 * 1E3
 u2213 = tLeaf_c + 273.15
 u2214 =  tCham_c + 273.15
 u2216 =  (parIn_um * f_parIn + parOut_um * f_parOut) * alphaK
 u2218 =   (u2216 + 1.0773E-7 * ((u2214 ^ 4) - (u2213 ^ 4)) - u20 *
44100.0)/(u111 * 51.4 + 4.3092E-7 * (u2213 ^ 3))
 u221 =  Tleaf_c + u2218 * doEB
 u222 =  ( 0.61365 * EXP(17.502 * u221 / (240.97 + u221)))
 u223 =  u222 * 1000 / press_kPa
 u224 =  u223 - h2o_2_mm
   $ doEB IF Tleaf_c ELSE Tcham_c Tleaf_c + 2 /  THEN  &u225 = $
 u226 =  ( 0.61365 * EXP(17.502 * u225 / (240.97 + u225)))
   $ u224 0 <> IF 1000 u223 h2o_2_mm + 2 / - u224 / u20 * ELSE 0 THEN  &u22 = $
   $ u22 0 <> IF 1.0 1.0 u22 / 1.0 u111 / - / ELSE 0 THEN  &u23 = $
 u24 =  h2O_2_mm * press_kPa / 1000
 u25 =  u222 - u24
```

```
u27 =  u226 - u24
u30 = (co2_1_um - co2_2_um * (1000 - h2o_1_mm) / (1000 - h2o_2_mm)) * u10
u35 =  1.0 / (1.6 / u23 + 1.37 / u11)
u36 =  ((u35 - u20/2) * co2_2_um - u30) / (u35 + u20/2)
u38 =  u36 * press_kPa * 1E-3
u39 =  u36 / co2_2_um
u51 =  (1.0 - (u20 * press_kpa)/u222/u23) * 100
u52 =  co2_2_um - u30 / (u11 / 1.35)
u53 =  u30 * u51 /100.0 / u52
}
```

16

Configuration Topics

There must be some way to do this...

16

Configuration Topics

The LI-6400 can do a lot of tasks, from fluorescence to soil respiration, and it has much flexibility on how it does them. Obviously, configuration is an important issue.

Managing Configurations

Configuration files live in the /User/Configs directory (folder), and a number of subdirectories within it.

Sharing Between Instruments

If you wished to provide identical configurations to multiple LI-6400s, you could simply configure one unit the way you want it, then copy the /User/Configs directory to the other LI-6400(s) (provided they all have version 6.1 or above software). So, configurations can be shared. *Calibration* files, on the other hand, are instrument-specific, and reside in the /dev directory. The /dev directory should *never* be moved between instruments.

Regenerating Default Configuration Files

When OPEN runs, it checks to be sure there is in fact a /User/Configs directory. If it is missing, it will automatically create one, and populate it with all the default folders and files it needs.

There is also a program in OPEN's Utility menu that lets you pick and choose the types of files to regenerate (Figure 16-1 on page 16-3). Thus, for example, if you wanted to make sure the default AutoPrograms were in place, you could select only AutoPrograms to reinstall. Table 16-1 on page 16-3 shows what files are installed by this program.
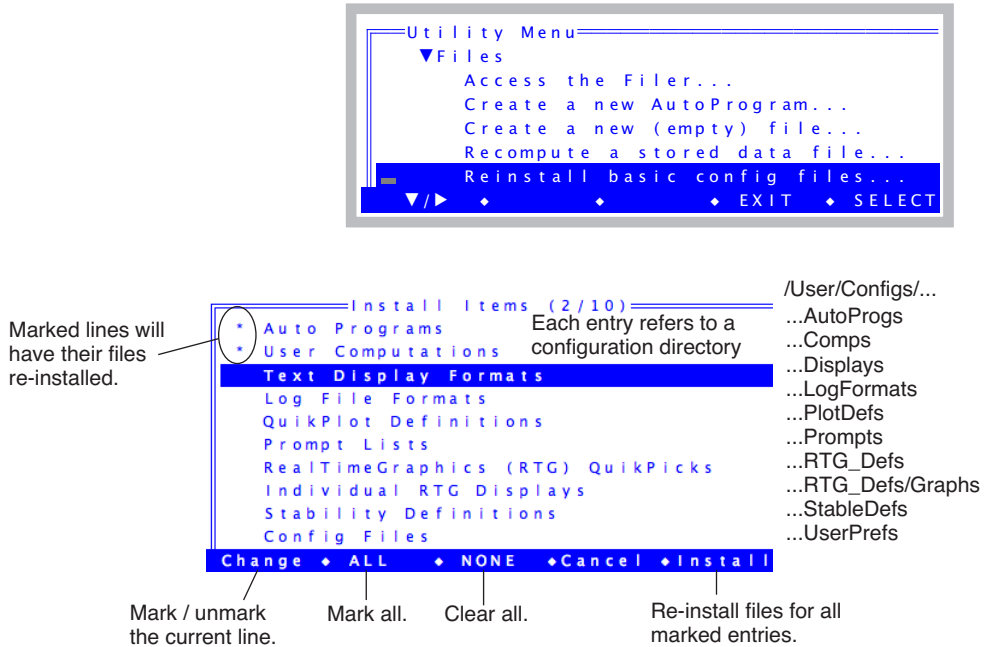
Figure 16-1. The Utility menu entry "Reinstall basic contig files..." allows selected (or all) default files to be re-installed. If /User/Configs does not exist, it is created.

Table 16-1. The default configuration files.

Directory in /User/Configs/	Files
AutoProgs	AutoLog2, LightCurve2, A-CiCurve2, TimedLamp2, StandardSystemTest, CO2Curve_MutipleLight, LightCurve_MultipleCO2
Comps	StdComps_6.2, PhotoTrans_Area, PhotoTrans_Mass, T_RH_PhotoTrans_Area, T_RH_PhotoTrans_Mass
Displays	StdDisplay_6.2, Diagnostic
LogFormats	StdLogFmt_6.0
PlotDefs	Photo, Cond vs Obs, Light Curve, A Ci Curve
Prompts	Default (none)

Table 16-1. (Continued) The default configuration files.

Directory in /User/Configs/	Files
RTG_Defs	Std Gas Exchange
RTG_Defs/Graphs	PHOTO-Ci, PHOTO-PARi, Tblk, Tair, Tleaf, H2OR, H2OS, Flow, CO2R, CO2S, PHOTO, COND
StableDefs	Std Stability
UserPrefs	FactoryDefault_6.2.xml

The Config Menu

The Config Menu (**f2** from OPEN's Main Screen) allows you to create, store, read, and modify the instrument's configuration (Figure 16-2).

Build a new configuration by answering questions. See page 16-5.

Read an existing configuration. See page 16-9.

```
 Config Menu
   New...
   Open...
   Save as...
   View/edit...

  ▼/▶   ♦        ♦       ♦ EXIT  ♦ SELECT
```

Save the current configuration. See page 16-10

View/edit the current configuration. See page 16-11.
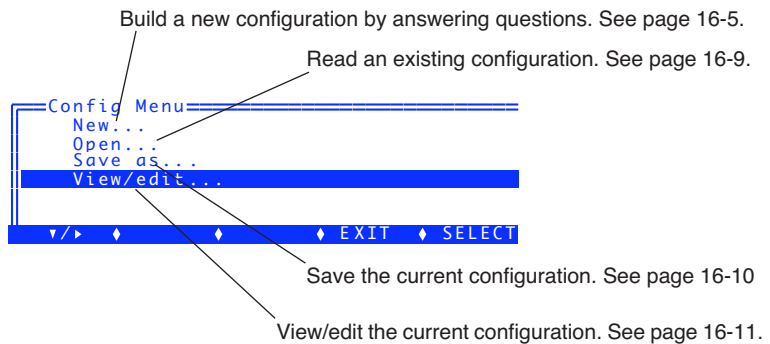
Figure 16-2. The Config Menu

Building a New Configuration

The simplest way to get a good first guess on a configuration for a particular scenario is to use Config Menu|New... (Figure 16-3).
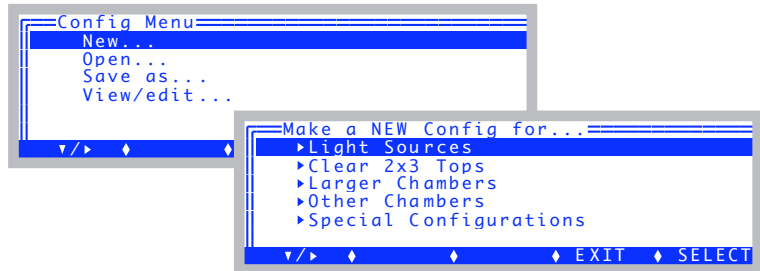
Figure 16-3. The New *entry of the Config Menu.*

Most of these entries (Figure 16-4) will run utility programs that ask you some questions and give you a configuration that should be close to what you are trying to do.

▼ **Light Sources**
    6400-02B LED Source   *See example on page 16-6.*
    6400-18 RGB Source   *See example on page 8-12.*
    6400-40 Fluorometer   *See example on page 27-14.*
▼ **Clear 2x3 Tops**
    Std 2x3 Chamber Top
    6400-06 PAM Interface
    6400-08 Clear Bottom
    6400-10 MiniPAM Interface
    6400-14 OptiSciences Interface
▼ **Larger Chambers**
    6400-07 or -11 2x6 Chamber
    6400-05 Conifer Chamber
    6400-17 Whole Plant Arabidopsis   *See page 16-52.*
    6400-22 Opaque Conifer Chamber *See page 16-54.*
    6400-24 Bryophyte Chamber   *See page 16-54.*
▼ **Other Chambers**
    6400-09 Soil Chamber   *See example on page 28-20.*
    6400-15 Extended Reach Chamber
▼ **Special Configurations**
    Insect Respiration   *See page 16-54.*
    6400-13 TC Adapter   *See example on page 16-37.*
    Custom Chamber - Closed   **See** *Page 16-73.*
    LI-610 Control via Fct Key   *See Figure 16-28 on page 16-26*

Figure 16-4. The New *node map.*

The first step is to pick where to begin, and in some cases, you have multiple options. For example, if you want the RGB light source on top of a 2x3 cham-

ber with a clear top (never mind that it's not designed to do that - it works if you mount it somehow), you could start with the "`6400-18 RGB`", or you could start with the "`Std 2x3 Chamber Top`".

■ **Example: 6400-02B LED Source**

This is for the 6400-02 (red LEDs only) and the 6400-02B (red and blue LEDs).
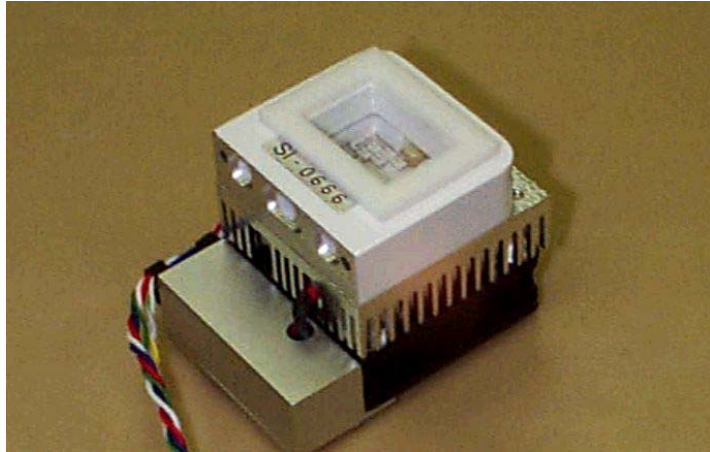
Figure 16-5. The 6400-02B sources have serial numbers starting with SI-.

1 **Pick the source to be used**

The list of LED sources in your accessories list[1] is shown (Figure 16-6). If the one you are going to use is not shown, press **f2** (**Add**) and go to Step 2. Otherwise, skip to Step 3.
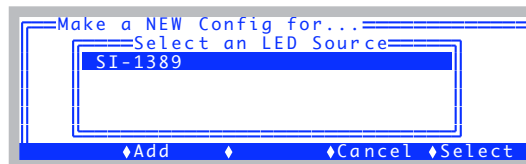
Figure 16-6. Selecting the specific LED light source.

---

[1]See **View / Edit Accessories** on page 18-8.

2 **Adding the source to the list (if necessary)**

To add a source, you'll need the serial number, calibration value, and calibration date (the latter isn't necessary - it's just for your information), as illustrated in Figure 16-7.

6400-02B LED Light Source
Calibration Certificate
Serial Number SI-2209

Date: 31 July 2007   Technician _____

CalParLED= -0.76

```
═════════════Add an Accessory═══════════
1. Select serial number (or escape to
quit)
  (Examples: ga-123, Q1032, si-415)
SI-2209_
```
`DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar`

```
═════════════Add an Accessory═══════════
1. Select serial number (or escape to
quit)
  (Examples: ga-123, Q1032, si-415)
SI-2209
2. Enter cal value -0.76_
```
`DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar`

```
═════════════Add an Accessory═══════════
1. Select serial number (or escape to
quit)
  (Examples: ga-123, Q1032, si-415)
SI-2209
2. Enter cal value -0.76
3. Enter cal date 31 July 2007
```
`DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar`

The new entry will now be in the list.

```
═══Make a NEW Config for...═══════════
   ════Select an LED Source════
   SI-1389
   SI-2209
```
`        ◆Add    ◆        ◆Cancel ◆Select`

Figure 16-7. Entering a new accessory.

3   Configure the Settings

There are several options with 2x3 chambers, and you get to set them now (Figure 16-8).

```
═══Config for 2x3═══
→ ─▼LightSource= "6400-02 or -02B LED So
    └ LampSerialNumber= "SI-1234"
  ─ Bottom= Opaque
  ─ Material= Broadleaves
  └ LeafTemp= Measure

   ▼/▶   ◆Edit    ◆        ◆Cancel ◆DoIt
```
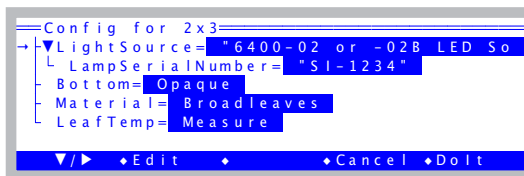
Figure 16-8. The 2x3 chamber configuration dialog.

LightSource=

At this point, you can change your mind about the light source, and select anything else you might want. The subnode under LightSource will be asking for a relevant serial number, if any. If the light source implies a clear topped chamber (e.g. Sun+Sky), then you'll be asked for the serial number of that top.

Bottom=

The bottom is a choice between Opaque and Clear.

Material=

Set as needed. See **The Material= Node** on page 16-51.

LeafTemp=

Measured or energy balance. For more details on energy balance, see the discussion on page 17-6.

4   Final Step

Once you press **DoIt** (**f5**), you are presented with some options (Figure 16-9):

```
Config Modifications Ready. Pick one:

N) implement as a New config named:
   '2x3 LED'
C) apply to the Current config
V) View the modifications
E) return and Edit your settings
<escape> – discard and quit
```
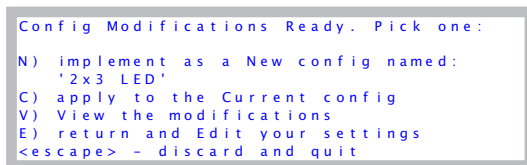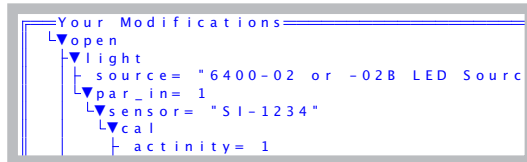
Figure 16-9. This Final Step screen is typical for many of the configurations options.

N - Exit, and make a new configuration file (you pick the name) stored in /User/Configs/UserPrefs. It will be in the list of files presented when you start

OPEN, or when you select `Config Menu|Open...`.

C - Exit, and apply these changes to the current configuration. What exactly would be changed? You can see this by first selecting the **V** option. (This does not save anything to disk. You can do that later, if you choose, with `Config Menu|Saveas`.)

V - View the parts of the configuration tree that you have affected by this configuration. When done viewing, you are returned to the display shown in

```
╔══Your Modifications════════════════
║ └▼open
║ ├▼light
║ │ ├ source= "6400-02 or -02B LED Sourc
║ │ └▼par_in= 1
║ │   └▼sensor= "SI-1234"
║ │     └▼cal
║ │       ├ actinity= 1
```

Figure 16-9, where you can then choose an option, or exit.

E - return to the dialog and edit your settings.

escape - Exit, and implement nothing.

Open a Stored Configuration

`Config Menu|Open...` allows you to implement any previously stored configuration.

```
╔═Config Menu══════════════════════
║   New...
║   Open...
║   Save as...
║   View/edi┌──────────────────────────────────
║           │Dir: /User/Configs/UserPrefs
║           ╞════Select a new configuration═══════
║           │Closed Mode
║ ▼/▶   ◆   │Default+LED.xml
            │Default+RGB.xml
            │FactoryDefault_6.1.xml
            │LCF.xml
            │+ Dir    ◆        ◆    ◆ CANCEL◆ SELECT
```
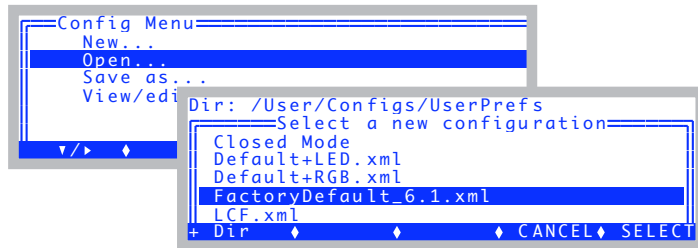
The Standard File Dialog for reading files (page 5-9) is used, which lets you pick a file from a list of those stored in the directory /User/Configs/UserPrefs.

This dialog also will appear when OPEN runs at power on if there is more than one file in /User/Configs/UserPrefs.

Version 6.1 and later configuration files end in .xml; earlier version configuration files (which may be there if you upgraded to a new version) do not.

Save the Current Configuration

`Config Menu|Save as...` will save the current configuration to a file (Figure 16-10).
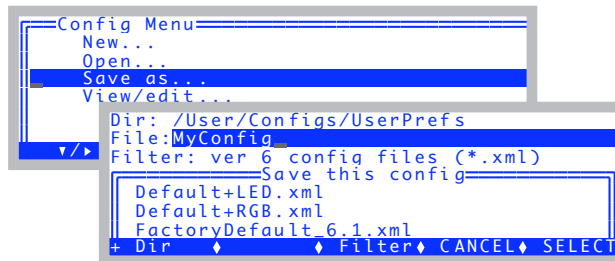
Figure 16-10. Config Menu | Save as.

The routine uses the Standard File Dialog for writing files (page 5-11). The default destination directory is /User/Configs/UserPrefs. File names will automatically have .xml appended to them.

Write Protected Files

If you try overwriting FactoryDefault_62.xml (or any other write protected file), you will get a warning (Figure 16-11). Just modify the name and save it again.

Figure 16-11. The write protected file warning message.

Editing the Current Configuration

`Config Menu|View/edit` allows the current configuration to be viewed and edited.
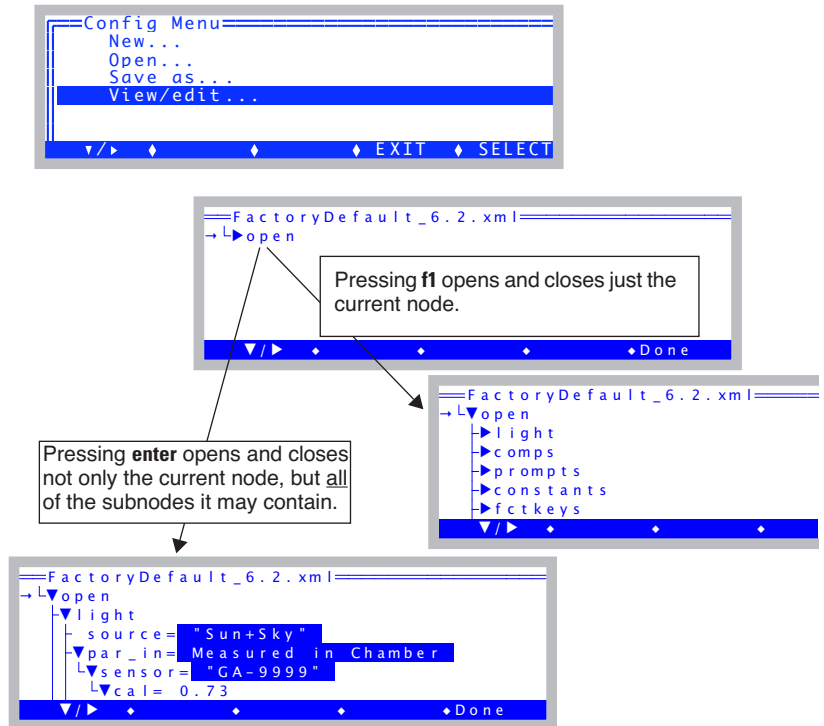
Figure 16-12. View/edit shows the entire <open> configuration node. Nodes can be opened and closed with **f1** *and (unless the node is editable) with* **enter***.*

The configuration is shown in a tree view with nodes that can be expanded or collapsed. A blinking box cursor and a small left arrow indicate the current location in the tree. If the current node is editable, the **f2** label will show **Edit** (Figure 16-14).

■ **Example: Change the Light Source**
As an example of editing a configuration, let's start with the default configuration, and change the light source from "`Sun+Sky`" to something else.

1   **Expand the <light> node**
     Scroll down to <light>, and press **f1** (Figure 16-13).
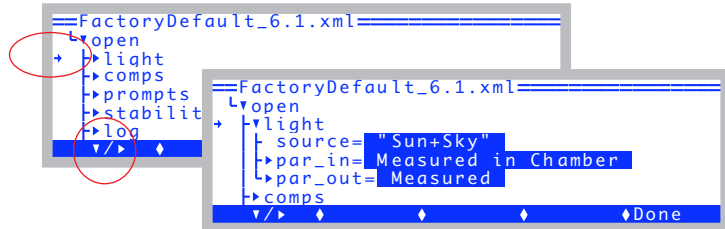
Figure 16-13. Using **f1** *to expand a node.*

2   **Edit the <source> node**
     Scroll down to the <source> node. Press **f2**, which will be labelled **Edit**
     (Figure 16-14).
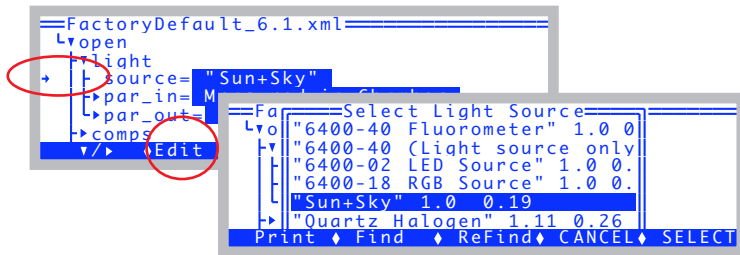
Figure 16-14. Using **f2** *to edit a node.*

3   **Pick a new source, such as Quartz Halogen**
     Scroll down to **"Quartz Halogen"**, and press **enter** or **f5** (Figure 16-15).
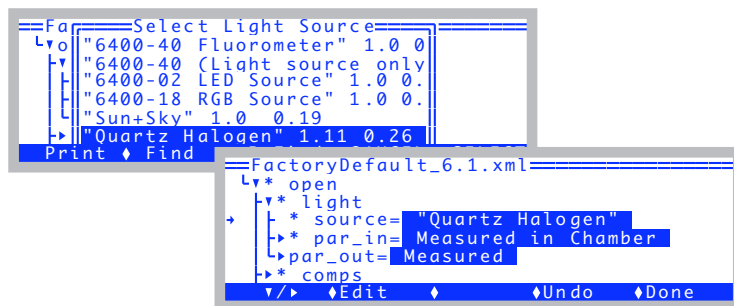
Figure 16-15. Changing the light source.

Notice that many nodes now are preceded by an asterisk (*). This means that the node or one of its subnodes has changed.

4   View all the places a change occurred
If you scroll around and open the * nodes, you will find that changing the light source from **"Sun+Sky"** to **"Quartz halogen"** also resulted in changes to two other nodes (Figure 16-16).
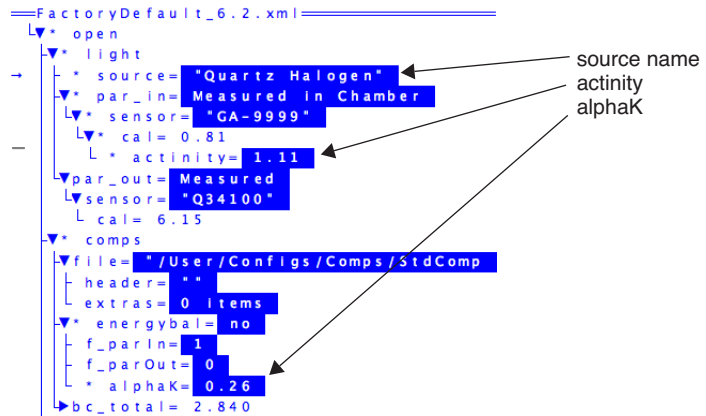
Figure 16-16. Changing the light source changes some system constants.

In addition the trail of asterisks leads all the way out to the main screen (Figure 16-17).
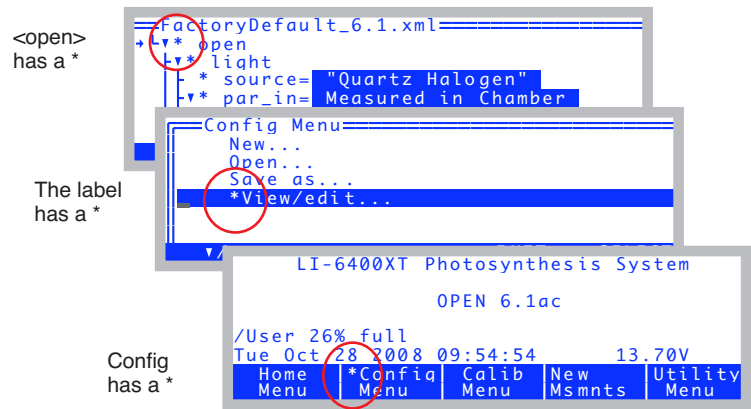
Figure 16-17. Signs that the configuration has changed.

■ **Reverting a Change**

The Configuration Editor has a revert feature, that lets you change back one or all changed nodes.

1 **Locate the node to be reverted**

Put the cursor on a node with an asterisk. **f4** will be labelled **Revert**. Pressing **Revert** will provide two choices: `T – undo This change`, and `A – undo ALL changes`. The **T** choice will undo just the node (and all subnodes, if any) that the cursor is currently on, while **A** will revert all nodes, which essentially sets the configuration back to its previously stored state.
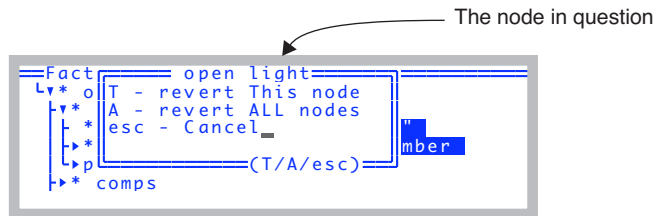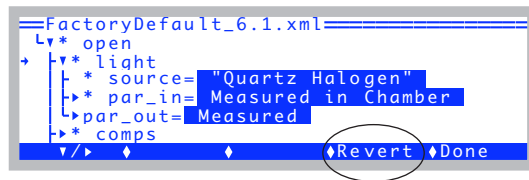
Figure 16-18. Reverting nodes.

An alternative to Revert ALL is to simply use `Config Menu|Open` and reread the file.

The Configuration Tree

Node Reference

The nodes of the configuration tree are shown below, along with references to more information about them.

▼ **open**
   ▼ **light**
      source= Sun+Sky   *Specifying the Light Source* on page 8-4.
      ▼ **par_in**=Measured in Chamber   *Figure 8-3 on page 8-5.*
         ▼ **sensor**= "GA-1232"   *Figure 8-5 on page 8-6.*
            ▼ **cal**= 0.8 a_{qc} *in Equation (14-17) on page 14-10.*
               actinity =1.00 f_a *in Equation (14-18) on page 14-11.*
                 transm= 1 τ *in Equation (14-18) on page 14-11.*
           transm= 0.95 τ_x *in Equation (14-18) on page 14-11.*
      ▼ **par_out**= Measured
         ▼ **sensor**= "Q31295"   *Figure 8-6 on page 8-7.*
            cal= 6.7 a_{qx} *in Equation (14-19) on page 14-11.*
   ▼ **comps**
      ▼ **file**= "/User/Configs/Comps/StdComps_6.2"   *See page 15-14.*
         header= ""   *Hooking the Extras Module* on page 15-14.
         ▼ **extras**= 1 items   *Extras* on page 15-2.
            extras[1]=1000: WUE (Water use eff.)
      ▼ **energybal**= no   *Using Energy Balance in OPEN* on page 17-6.
         f_parIn= 1 f_{in} *in Equation (17-11) on page 17-8.*
         f_parOut= 0 f_{out} *in Equation (17-11) on page 17-8.*
         alphaK= 0.19   *Table 17-1 on page 17-4.*
      ▼ **bc_total**= 0.200   *Boundary Layer Variables* on page 14-20.
         stomatal_ratio= 1 K *in Equation (14-32) on page 14-20.*
         ▼ **bc_oneside**= 0.100 g_{bw} *in Equation (14-31) on page 14-20.*
            type=2: Lookup Table (Area, Fan)
            table= "/Sys/Lib/StdBLCTable"
            area= 6   ID:-33.
            slope= 0 g_1 *in Equation (14-31) on page 14-20.*
            offset= 0.1 g_0 *in Equation (14-31) on page 14-20.*
            fan= off
   ▼ **prompts**   *Defining Prompts* on page 9-21.
      onlog= off   *Figure 3-112 on page 3-98.*
      ▼ **items**= "My List"
         items[1]=-33: Area
         items[2]=-52: Oxygen%
         items[3]=-101: Plot#
   ▼ **fctkeys**   *Defining Fct Keys* on page 16-22
      level3_f1
      level3_f2
      level7_f1
      level7_f2
      level7_f3
      level7_f4
      level7_f5

▼ **constants**
    oxygen= 21 % X_o *in Equation (14-44) on page 14-32.*
    bb_vapor= 1.5 α_w *in Equation (14-44) on page 14-32.*
    bb_oxy= 0.9 α_o *in Equation (14-44) on page 14-32.*

▼ **stability**
    ▼ **items**= "Std Stability"   *Defining Stability* on page 6-29.
        items[1]= CO2S (-2) 15 Slp<1
        items[2]= H2OS (-5) 15 Slp<1
        items[3]= Flow (-7) 15 Slp<1

▼ **log**   *Determining What is Logged* on page 9-8.
    ▼ **format**= StdLogFmt_6.0
        ▼ **items**= (37 items, 0 in header)
            hdr=( )   *Header Constants* on page 9-12.
    ▼ **option**s   *Log Options* on page 9-14.
        beep= normal   *Page 9-14.*
        hdr= normal   *Page 9-14.*
        rem= normal   *Page 9-15.*
        stab= no   *Page 9-15.*
        ▼ **stats**= no   *Page 9-15.*
            period= 15
        excel= yes   *Page 9-17.*
        comm= normal   *Page 9-18.*
    datafile= "Data"
    ▼ **button**   *User Definable Log Button* on page 9-6.
        ▼ **action**= "Normal Log"
            code= " LogOneObsManual "

▼ **display**
    ▼ **text**= "StdDisplay_6.2"   *Display Editor* on page 6-6.
        ▼ **lines**
            a= CO2R_ ml CO2S_ ml H2OR_mml H2OS_mml
            ...
            z= ( )
        ▼ **groups**   *Display Groups* on page 6-9.
            home= abc
            end= def
            pgup= ghi
            pgdn= jkl
    ▼ **graphs**= "Std Gas Exchange   *Defining Graphs* on page 6-14.
        a= PHOTO, COND
        ...
        h=

▼ **controls**
    ▼ **defaults**   *Specifying Default Control Settings* on page 16-27.
        fan= 5
        ▼ **flow**   *Page 16-27.*
            type= 2
            target= "500"
        ▼ **co2**   *Page 16-28.*
            type= 1
            target= "400"
        ▼ **temp**   *Page 16-28.*
            type= 1
            target= "25"

▼ **light**   *Page 16-28.*
    type= 1
    target= ""
▼ **a2d**   *Configuring Spare Channels* on page 16-29.
  ▼ **avgtime**= 4.0 secs</avgtime>   *Averaging Time* on page 16-29.
  ▼ **userchans**   *See page 16-31.*
    ch20= "ON, gnd=3, res=0"
    ch21= "Off"
    ch22= "Off"
    ch23= "Off"
▼ **comm**   *Configuring the Comm Port* on page 11-26.
  config= 9600 8 1 N
  lterm=-1
  incoming= ignore   *Data for Real Time Measurements* on page 11-51.
▼ **matching**   *Matching Variations* on page 16-19.
  type= normal
  disp= 'a'
  ▼ **settings**   *Match Settings* on page 16-21.
    CO2Limit= 10.0
    H2OLimit= 1.0
    MaxAutoTime= 60
▼ **hooks**   *Hooking Events* on page 16-37
  ▼ **items**= 1 items
    items[1]= set the DAC (On NM Enter)

Configuration File Format

When stored in a file (Figure 16-19), the configuration looks similar to the tree, except the node names are surrounded with angle brackets (<>), creating what is called a tag. Every tag has a corresponding closing tag of the same name with a prefixed '/'. (e.g. <light> ... </light>).

```
<open>
     <version>"6.2"</version>
     <configfile>"/User/Configs/UserPrefs/FactoryDefault_6.2.xml"</configfile>
     <light>
          <source>"Sun+Sky"</source>
          <par_in>1
               <sensor>"GA-1094"
                    <cal>0.81
                         <actinity>1.00</actinity>
                    </cal>
               </sensor>
               <transm>1 </transm>
          </par_in>
          <par_out>1
               <sensor>"Q30292"
                    <cal>7.32 </cal>
               </sensor>
          </par_out>
     </light>
     <comps>
          <file>"/User/Configs/Comps/StdComps_6.2"
               <header>""</header>
               <extras></extras>
          </file>
          <energybal>no
               <f_parIn>1 </f_parIn>
               <f_parOut>0 </f_parOut>
               <alphaK>0.19</alphaK>
          </energybal>
          <bc_total>0.200
               <stomatal_ratio>1</stomatal_ratio>
               <bc_oneside>0.100
                    <type>2 </type>
                    <table>"/Sys/Lib/StdBLCTable"</table>
                    <area>6</area>
                    <slope>-0 </slope>
                    <offset>0.1 </offset>
                    <fan>0 </fan>
               </bc_oneside>
          </bc_total>
     </comps>
     <prompts>
          <onlog>off</onlog>
          <items>"Default (none)"</items>
...
```

Figure 16-19. LI-6400 configuration files are stored in an XML-style format

Matching Variations

The <open> <matching> part of the configuration tree provides some degrees of freedom regarding matching.

Matching Methods

The <open> <matching> <type> node determines how matching is done. There are three possibilities (Figure 16-20): 1) use the match valve (see **Matching the Analyzers** on page 4-33); 2) turn off the mixing fan; 3) use an empty chamber.
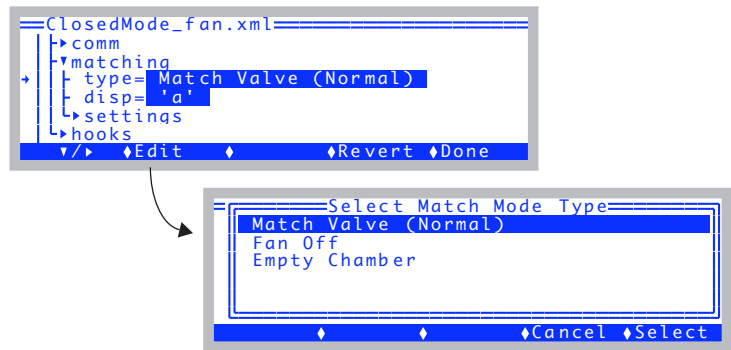
Figure 16-20. Specifying the match method.

Fan-based Matching.

Suppose you wish to make soil CO_2 flux measurements in flow-through mode (i.e. flux is proportional to the sample - reference CO_2 difference). Since the chamber sits on a porous surface (turf or soil), it may be difficult to collect outflow to send back to the match valve. Matching will be a problem.

In such a case, you can match by shutting the fan off[2] thereby isolating the sample cell of the IRGA from the rest of the chamber. The sample IRGA will see the same incoming air as the reference IRGA, assuming the normal flow is into the IRGA, then on into the chamber.

You get this by setting <open> <matching> <type> to "Fan off".

---

[2]·Also, to zero and span, the fan will need to be shut off, but the zeroing and spanning programs provide a method for doing that already.

In normal matching, the reference side changes because it sees air from the chamber. When doing fan-based matching, the sample CO_2 and H_2O change when match mode is entered (and the fan automatically shuts off). Mathematically, matching remains the same with the adjustment being made on the sample side.

The match mode display for a fan-based match is shown in Figure 16-21.
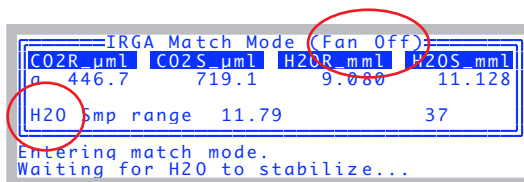
Figure 16-21. Entering match mode when matching via the fan instead of the valve.

Empty Chamber Matching

Another way to match is to simply remove all plant material from the chamber and close it. While this is conceptually simple, since the flow configuration for the match is identical to normal operations, it carries the disadvantage of not being able to be done automatically, or without disturbing an on-going measurement. Also, leaks into the chamber may be different with and without a leaf in place (see **Diffusion Leaks** on page 4-44).

The match mode display for empty chamber matching is shown in Figure 16-22.
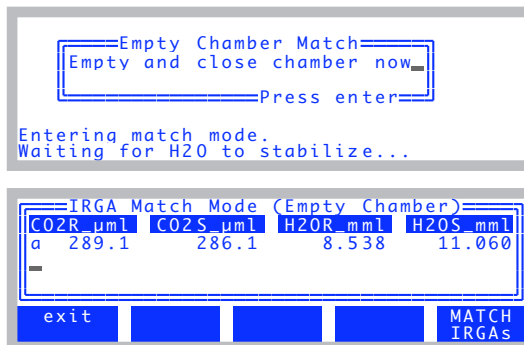
Figure 16-22. The match mode display for empty chamber matching.

Other considerations

The <open> <matching> <type> configuration only affects New Measurement's match mode. Thus, for example, the match valve exerciser diagnostic program (**Match Valve Tester** on page 21-8) will always toggle the match valve, regardless of the <open> <matching> <type> setting.

The match method is controlled by a single variable named *matchType*. When *matchType* = 0, normal matching is in effect; *matchType* = 1 is fan-based matching, and *matchType* = 2 is empty chamber matching. Thus, you can also set match type programmatically, such as in an AutoProgram.

Match Settings

The <open> <matching> <settings> node contains three more settings that affect match mode (Figure 16-23).

```
══FactoryDefault_6.1.xml═══════════
  ┌ disp= 'a'
  └▾settings
→   ├ CO2Limit= 10.0
    ├ H2OLimit= 1.0
    └ MaxAutoTime= 60
  └▸hooks
  ▾/▸   ♦Edit    ♦       ♦Revert ♦Done
```
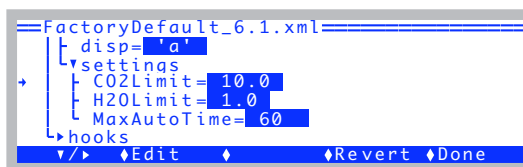
Figure 16-23. The <open> <matching> <settings> nodes.

CO2Limit

This value (μmol mol^{-1} CO_2) determines how big a CO_2 difference will be allowed before warning that it is too big (if manually matching) or skipping entirely (if matching during an AutoProgram). This variable that holds this value is *matchCO2Limit*.

H2OLimit

This value (mmol mol^{-1} H_2O) determines how big a H_2O difference will be allowed before warning that it is too big (if manually matching) or skipping entirely (if matching during an AutoProgram). The variable that holds this value is *matchH2OLimit*.

MaxAutoTime

The maximum time (secs) that match mode will wait for stability during an AutoProgram. The variable that holds this value is *matchAutoTimeLimit*.

Defining Fct Keys

OPEN 6.2 provides a simple method for defining seven of the fct keys in New Measurements mode. It is done by editing the nodes contained in <open> <fctkeys>.
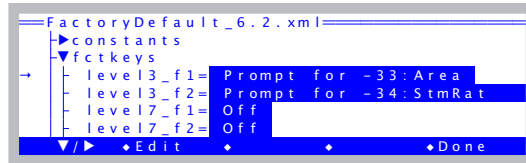
The definition of each of the 7 keys available for this method includes a Mode, which can be

Disabled

Prompt for a Constant

Custom

To change the mode, simply highlight it and press Edit.
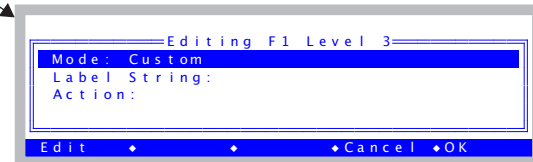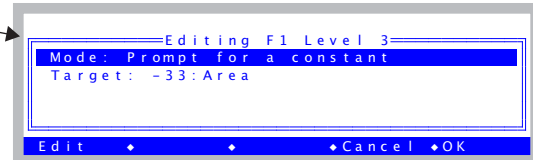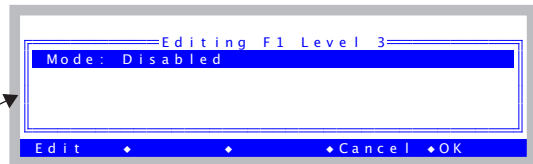
Figure 16-24. The <open> <fctkeys> node defines 2 keys on level 3, and all 5 keys on level 7 in New Measurements mode.

Prompting for Constants

The "prompt for a constant" mode is quite simple: the only extra piece of information needed is the system ID for that constant, which is shown on the editor line *Target* . Editing the *Target* line brings up a list of sys and user constants to pick from.

The method for labelling a fct key defined in this manner is to put the variable's log label on the top line, and the current value on the 2nd line. If either

is too long to fit, it is shortened (Figure 16-25).

The definition...
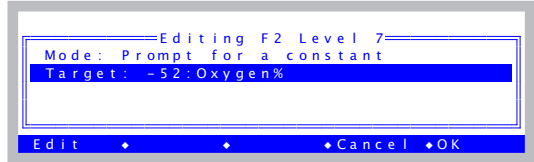
...how it looks...
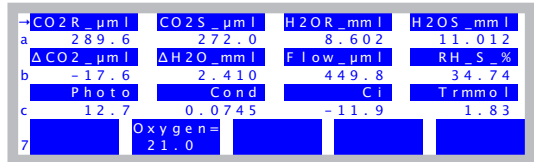
...and how it acts.
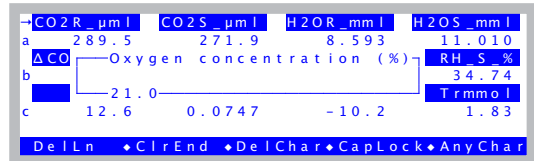
Figure 16-25. Defining f2 level 7 to prompt for Oxygen.

Custom Action

Fct keys defined with Mode=Custom need two extra pieces of information: what to use for a label, and what to execute when pressed.
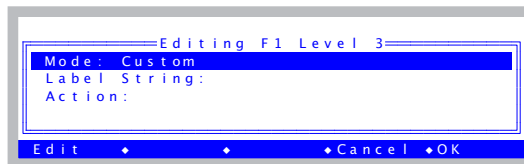
The label can be either a static string, or else a function to be called to make a string. The latter is needed if you wish to have the label contain a current value, for example. The action is a function that is called when the key is pressed.

Example #1

Suppose we want to make a key that simply sends a message out the RS-232 port when the fct key is pressed. Figure 16-26 shows how to set that up.

1. Set Mode to Custom.

```
═══════════Editing F2 Level 7═══════
Mode:  Custom
Label String:
Action:

Edit      ◆         ◆        ◆Cancel ◆OK
```
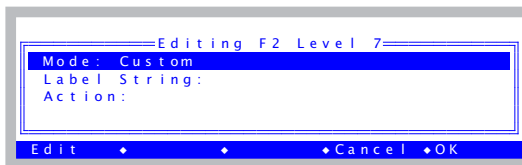
2. Highlight Label, press **Edit**.

3. Press **S**.

```
Label via String or Function ?

                              (S/F)
```
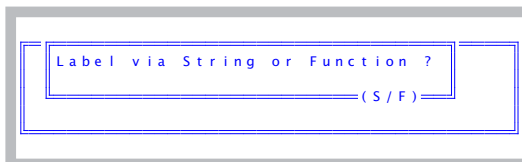
4. Enter the label, press **Enter**.

```
═══════════Editing F2 Level 7═══════
Mo ┌───Key Label (14 Chars max)═══┐
La │ Send   Now▁                  │
Ac

DelLn    ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```
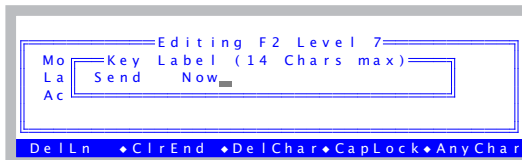
5. Highlight Action, press **Edit**.

6. Enter the code to be executed. Then press **OK**.

```
════════════════Action Fct════════
"Now\n" comm print▁

+DelLn    ◆ClrEnd ◆DelChar◆Cancel ◆OK
```

7. Done. Press **OK**.

```
═══════════Editing F2 Level 7═══════
Mode:  Custom
Label String:  Send   Now
Action: "Now" comm print

Edit      ◆         ◆        ◆Cancel ◆OK
```
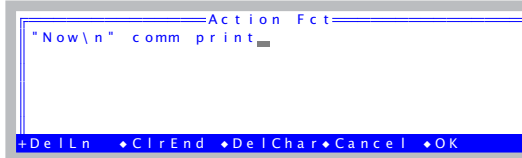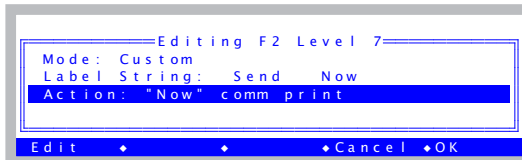
→CO2R_µml	CO2S_µml	H2OR_mml	H2OS_mml
a 289.6	271.9	8.601	11.008
ΔCO2_µml	ΔH2O_mml	Flow_µml	RH_S_%
b -17.7	2.406	449.7	34.73
Photo	Cond	Ci	Trmmol
c 12.8	0.0743	-14.6	1.82
7	Send Now		

In New Measurements mode, our key will look like this.

Figure 16-26. Defining f2 level 7 to send a message out the Comm port when pressed. The funny spacing in Step 4 is to make it look good on the label.

Example #2

A more useful example would be to define a key to set a new Dew Point temperature on an LI-610 Dew Point Generator. Also, we'd like the key label to reflect the current dew point setting. We'll use D/A channel 3 for this. Figure 16-27 shows how this configuration looks when done. Mode should be *Custom*, and when you edit the Label, select **F** for function.
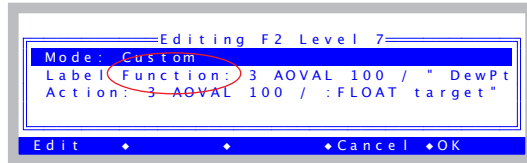
Figure 16-27. Configured for driving an LI-610.

The *Label* function should be

```
3 AOVAL 100 / " DewPt= %1.2f" to_string
```

"3 AOVAL" gets the current value of D/A channel 3. "100 /" divides it by 100 (and thus converts mV to °C). The rest of the line puts it into a string formatted to show temperature with 2 digits to the right of the decimal.

The *Action* should be:

```
3 AOVAL 100 / :FLOAT target
"610 Target Temp (C)" "%g" &target AskNewFloat IF
    target 100 * 3 AOSET
THEN
```

The first line creates a temporary variable named *target*, that contains the current temperature, based on the current D/A setting of channel 3. The next line prompts for a new value, and if one is entered, the third line sets D/A channel 3 to the new value (temperature * 100, to convert to mV).

Note: If you actually want to do this example, don't type this in. Instead, go to Config Menu, select New..., and then select the installer for the LI-610 example (Figure 16-28).

Pick the key (f1 to
f5, level 7)

Pick the DAC channel

```
═══Make a NEW Config for...═══
  ▶Other Chambers
  ▼Special Configurations
     Insect Respiration...
     6400-13 TC Adapter...
     Custom Chamber - Closed...
     LI-610 Control via Fct Key...
  ▼/▶    ◆        ◆        ◆ EXIT  ◆ SELECT
```

```
This program adds a Fct Key for
driving an LI-610 DewPoint Generator
to the current configuration.
═══════Configuration Details═══════
Use FctKey 1 (level 7)
Use DAC channel 3 (pin 12)

Edit    ◆        ◆        ◆Cancel ◆OK
```

Figure 16-28. The LI-610 configuration builder for version 6.2. You can select the key and the DAC channel. The implementation is done with the <fctkeys> node.

Example #3

The third example reimplements the area key, but uses the custom mode instead of the 'prompt for a constant' mode. We do this because in addition to prompting for area, we want to report the result in a message sent out the RS-232 port.

```
═══════Editing F1 Level 3═══════
Mode: Custom
Label Function: -33 KeyLabThisID
Action: -33 AskThisID-33 FmtGetVarAd


Edit    ◆        ◆        ◆Cancel ◆OK
```
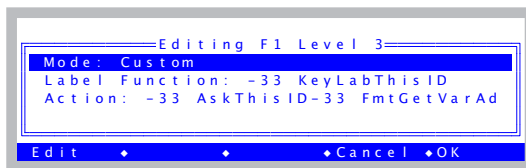
Figure 16-29. Redoing the area key.

The Label function (not string) passes the ID for area (-33) to a convenient function *KeyLabThisID* that does the labelling work.

```
-33 KeyLabThisID
```

The Action should be

```
-33 AskThisID
-33 FmtGetVarAddr VAL "Area is now %1.1f\n" comm PRINT
```

The first line handles the prompting, again with a convenient function *Ask-ThisID*. The second lines sends a formatted message with the new area value. We could have replaced *-33 FmtGetVarAddr VAL* with the actual variable name (*area_cm2*), but the method used works for any ID, and you don't need to know variable names to use it.

Specifying Default Control Settings

The first time you enter New measurements mode after implementing a configuration, the controls (fan, pump, CO_2 control, temperature, and light) are set according to the nodes contained in <open> <controls> <defaults>.

```
══FactoryDefault_6.1.xml═══════════
  ┌▼controls
  ↳ ▾defaults
→  ├ fan= 5
  ├▸flow= Flow rate 500
  ├▸co2= None (Mixer Off)
  ↳▸temp= None (Cooler OFF)
  ▼/▶    ◆        ◆      ◆Revert ◆Done
```
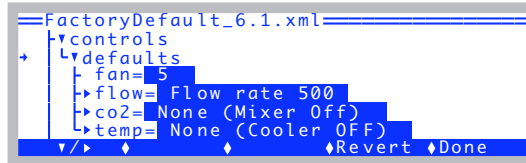
Figure 16-30. Default control settings.

Fan

The <open> <controls> <defaults> <fan> node determines the default fan speed (Figure 16-31). Set a value of 0 (off) through 5 (fast).

```
══FactoryDefault_6.1.xml═══════════
  ┌▼controls
  ↳ ┌──Enter speed for Default (0..5)┐
         █
  └──5─────────────────────────┘
  ├▸co2= None (Mixer Off)
  ↳▸temp= None (Cooler OFF)
  DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```
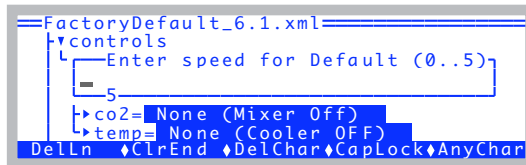
Figure 16-31. Setting the default fan speed.

Flow / Humidity

The <open> <controls> <defaults> <flow> node determines the default humidity control (Figure 16-32).

```
══Environmental Controls═══════
▼FLOW: Flow Rate
→ ├ Control= Flow_µm
  └ Target= 500 µmol/s

  └ T) reTarget, exit   O) Off, exit═══
  ▼/▶  ◆Edit   ◆       ◆cancel ◆ Keep
```
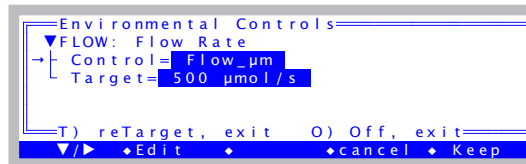
Figure 16-32. Setting the default flow / humidity controls.

CO$_2$

The <open> <controls> <default> <co2> node determines the default CO$_2$ mixer settings (Figure 16-33).
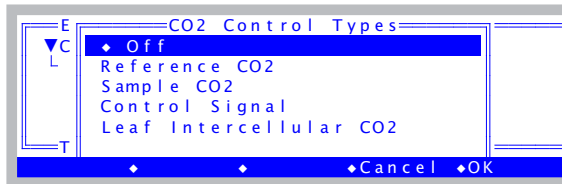
Figure 16-33. Setting the default mixer settings.

Temperature

The <open> <controls> <default> <temp> node determines the default temperature controller settings.
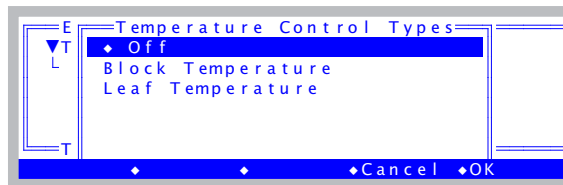
Figure 16-34. Setting the default temperature settings.

Light

The <open> <controls> <default> <light> node is visible only if a controllable light source is specified in the <open> <light> <source> node. If one is, the default settings can be set by editing the <default> <light> node Figure 16-35.
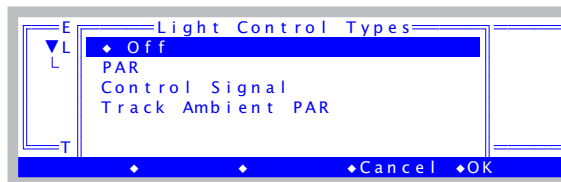
Figure 16-35. Default light settings. The dialog will be the one associated with whatever the light source is specified to be.

Configuring Spare Channels

Analog Input Channels

The LI-6400 has 24 analog inputs, four of which (Table 16-2) are available to the user as spare channels. The <open> <a2d> node (a2d is short for analog-to-digital) lets you control the user input channels, as well as specifying the averaging time (Figure 16-36) for all channels.

Averaging Time

In general, longer averaging times result in lower noise. The default averaging time is 4 seconds, and is set in the node <open> <a2d> <avgtime> (Figure 16-36).
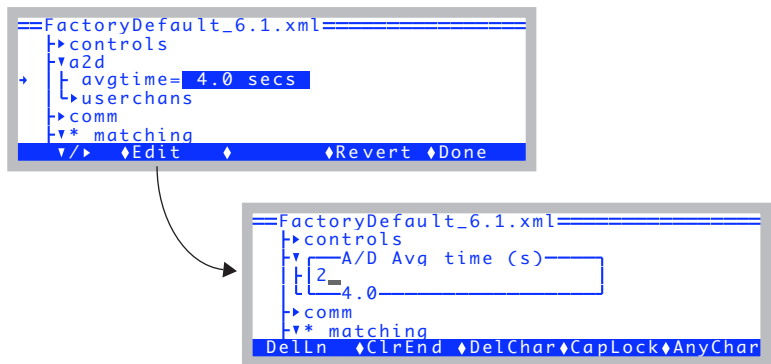
Figure 16-36. The <open> <a2d> <avgtime> node controls the averaging time used for all analog input channels (except the fluorometer signal).

Note: If you wish to do average times longer than 5 or 6 seconds, do not do it via the *avgtime* setting. Instead, use the statistics LogOption, described on page 9-16.

Wiring User Analog Input Channels

Measurements are referenced to a ground signal that is software selectable (Table 16-3). All of the LI-6400's analog inputs are over-voltage protected to ±35V, with or without power.

To wire an analog input, select a signal pin (Table 16-2) and a ground pin (Table 16-3).

The range on each of these channels is +/- 5V. The resolution depends on the sampling rate (see Figure 16-37): At 25 Hz, it is 0.27 mV, and at 100 Hz, it is 0.06 mV.

Table 16-2. User defined analog input channels.

LPL ID#	Pin # (37 Pin Connector)
20	36
21	19
22	18
23	37

Table 16-3. Ground channels for analog input signals

LPL ID#	Pin # (37 Pin Connector)	Also used for
3	31	Chamber signals
4	14	Pressure sensor
5	32	
6	15	
7	17	

Configuring User Analog Input Channels

OPEN takes care of the details of programming for the spare analog input channels. All you have to do is define the channel (Figure 16-37) so that it reflects the signal and ground channels being used by the sensor.
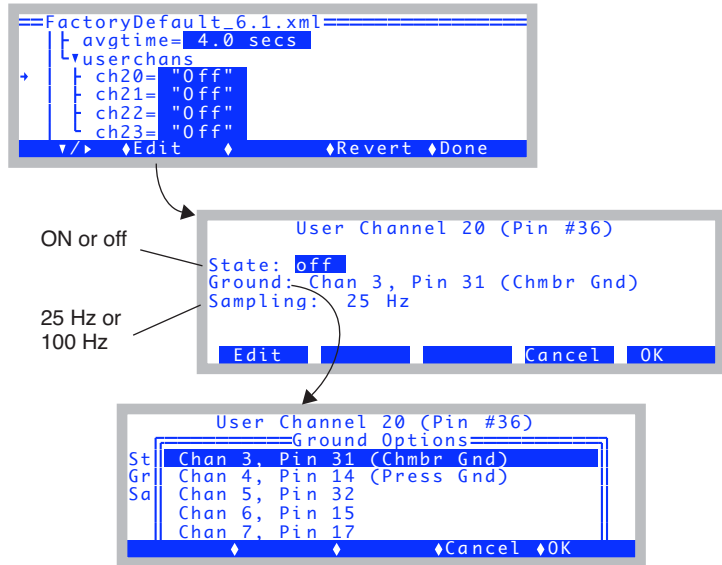
Figure 16-37. Configuring a spare analog input channel.

You are offered two <u>sampling rates</u>: most of the LI-6400's sensors are sampled at 25 Hz. The IRGA-related channels are sampled at 100 Hz. Thus, if you specify a 4-second averaging time, channels that are sampled at 25 Hz will have 25 x 4 = 100 samples taken and averaged each time a new measurement is ready (every 0.5 seconds). If you wanted each new reading to be made up of an entirely different set of raw readings, specify an average time of 0.5 seconds, or less. Then, each reading on a 25 Hz channel will be the average of 12 readings.

For a general discussion of analog input programming in LPL (i.e. details you just don't want to know), see **Analog Measurements** on page 23-71.

Analog Output Channels

Six digital to analog output channels are available for use (Table 16-4).

Table 16-4. Spare D/A channels.

LPL ID#	Pin # (37 Pin Connector)	Range (V)	Resolution (mV)	Current (mA)
3	12	-5 to +5	2.44	±5
8	10	0 to 5	19.5	+5 / -2
9	27	0 to 5	19.5	+5 / -2
10	9	0 to 5	19.5	+5 / -2
17	29	-5 to +5	39	+5 / -2
18	11	-5 to +5	39	+5 / -2
19	30	-5 to +5	39	+5 / -2

Connections

Use pin 13 as the ground for these outputs.

Software

An analog output is controlled with the LPL command AOSET. This and related commands are discussed in **Analog Output Control (D/A)** on page 23-78. Thus, to set channel 10 to 4V you would do the following:

```
4000 10 AOSET
```

Where and when do you do this? See **Hooking Events** on page 16-37 for several possibilities. For example, to drive D/A channel 10 with the computed value of Photosynthesis, put this piece of code into the <open> <hook> list using the On tic option:

```
u30 100 * 10 AOSET
```

If you just want to set a D/A channel manually from the keyboard, press **K** in OPEN's main screen to run the LPL Shell program (described on page 5-21). At the ok: prompt, type the command and press **enter**. For example,

```
ok:2450 3 AOSET
```

will set channel 3 to 2450 mV.

Digital Output

Several digital channels are available for use (Table 16-5):

Table 16-5. Spare digital channels.

LPL ID#	Pin # (37 Pin Connector)	LPL ID#	Pin # (37 Pin Connector)
0x0400	23	0x0404	25
0x0401	5	0x0405	7
0x0402	24	0x0406	26
0x0403	6	0x0407	8

Connections

Use pin 20 for the digital ground. These open-drain digital outputs require the use of an external pull-up resistor. When the digital line is set high, the voltage will go to 0. When it is set low, it will go to whatever voltage you have on the other side of the resistor (Figure 16-38.) Note: they are low power, and cannot directly drive most solenoids.
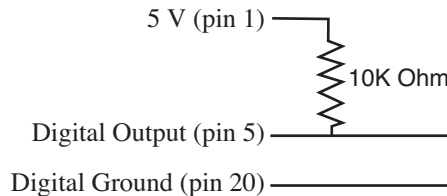
Figure 16-38. To connect a digital output, use a 10K resistor between the output pin and the 5V supply, and measure the voltage between the output pin and digital ground. When the output is set (high), you'll measure 0 volts between pins 5 and 20, and when not set (low), you'll measure 5 volts.

Software

Use the DIOSET command. (This and related commands are discussed in **Digital I/O** on page 23-80.) For example,

```
1 0x0402 DIOSET
0 0x0401 DIOSET
```

will set 0x0402 high (0V), and 0x0401 low (5V, if using pin 1 for voltage supply).

See **Hooking Events** on page 16-37 for possible ways to implement commands.

Digital Input

Two digital input channels are available, besides the log switch. (Table 16-5):

Table 16-6. Spare digital channels.

LPL ID#	Pin # (37 Pin Connector)
0x0106	4
0x0107	22

These inputs can be used to detect switch closures or to count pulses (provided they are slow - less than about 10Hz).

Connections

Use pin 20 for the digital ground. Digital input signals should be kept between 0 and +5 volts. Figure 16-39 illustrates a method of connecting a switch.
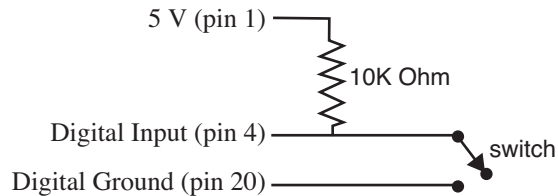
Figure 16-39. To connect a digital input, use a 10K resistor between the input pin and the 5V supply. When the switch is open, the digital input will be high, and when closed, the input will be low.

Software

For detecting the position of the switch, use DIOGET.

```
0x0106 DIOGET
```

It will return 1 if the input is high, and 0 if the input is low. An input channel can be used to count pulses (<10 Hz), and this is discussed in **Digital I/O** on page 23-80.

See **Hooking Events** on page 16-37 for possible ways to implement commands.

Pulse Counting

The LI-6400 has one pulse counting channel, that can count pulses up to about 4 KHz.

Connections

Use pin 20 for the digital ground, and pin 3 for the pulsed signal. A pulse is counted each time the signal drops from >5V to 0. Figure 16-40 illustrates a method of connecting a switch.

Pulse counting (pin 3) ———————— pulse generator

Digital Ground (pin 20) ————————
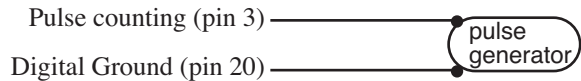
Figure 16-40. To connect the pulse counting, use pins 3 and 20. Pulses should be between 0 and (5 < x < 8) Volts.

Software

Accessed via the HSCOUNTS LPL keyword.

Summary of the Console's 37-Pin Connector

The LI-6400 console's 37-pin connector assignments are listed in Table 16-7. The shaded entries (pins 10, 21, and 28) are outputs that are wired in parallel with some component in the LI-6400, and would not normally be controlled by a user application.

Table 16-7. Summary of the console's 37-pin connector.

Pin #		Name	Comments
1	F,T[a]	+ 5V	
2	F	Battery	
3		Pulse Counting Input	**Pulse Counting** on page 16-35
4	R	Digital Input 0x0106	**Digital Input** on page 16-34
5	F, R	Digital Output 0x0401	
6	F, R	Digital Output 0x0403	**Digital Output** on page 16-33
7	F, R	Digital Output 0x0405	
8	F	Digital Output 0x0407	
9	F	Analog Output 10 (0 to 5V, 8 bit)	
10		Analog Output 8 (0 to 5V, 8 bit)	**Analog Output Channels** on page 16-32
11	F,T	Analog Output 18 (-5 to +5V, 8 bit)	
12		Analog Output 3 (-5 to +5V, 12 bit)	
13	T, R	Analog Ground	Use with analog outputs
14		Analog Input Ground 4	Used by pressure sensor
15	T	Analog Input Ground 6	**Analog Input Channels** on page 16-29
16		Sample H2O raw analog output	
17		Analog Input Ground 7	
18		Analog Input 22	**Analog Input Channels** on page 16-29
19	T	Analog Input 21	
20	F, R	Digital Ground	Use with digital I/O
21		Digital Output 0x0007	Match mode
22	F, R	Digital Input 0x0107	**Digital Input** on page 16-34
23	F, R	Digital Output 0x0400	
24	F	Digital Output 0x0402	See **Digital Output** on page 16-33
25	F	Digital Output 0x0404	
26	F	Digital Output 0x0406	
27		Analog Output 9 (0 to 5V, 8 bit)	**Analog Output Channels** on page 16-32
28		Analog Output 16	Flow meter zero
29	F	Analog Output 17 (-5 to +5V, 8 bit)	**Analog Output Channels** on page 16-32
30	F	Analog Output 19 (-5 to +5V, 8 bit)	
31		Analog Input Ground 3	Used by leaf chamber
32	F, R	Analog Input Ground 5	**Analog Input Channels** on page 16-29
33		Sample CO2 raw analog output	
34		Reference H2O analog output	
35		Reference CO2 raw analog output	
36	F, R	Analog Input 20, ±5V	**Analog Input Channels** on page 16-29
37	F	Analog Input 23, ±5V	

a.F = Used by 6400-40 LCF. T = 6400-13 T/C Adapter, R = 6400-18 RGB Source

Hooking Events

The <open> <hooks> node provides a method for you to add customized behavior at key points in the operation of OPEN. There are four places that OPEN provides hooks for <open> <hooks>:

- **On Config**
  When a configuration is selected at power on or from `Config Menu|Open`, immediately after implementing the selected configuration, the *On Config* hooks are executed.

- **On NM Enter**
  Each time New Measurements mode is entered, the *On NM Enter* hooks are executed.

- **ON NM Exit**
  Each time New Measurements mode is exited, the *On NM Exit* hooks are executed.

- **On Tic**
  Each second while in New Measurements mode, the *On Tic* hooks are executed.

In the default configuration, all of these hooks are empty.

An Example

Start with the default configuration (FactoryDefault_6.2.xml), and use "Config Menu | New" to add a 6400-13 Thermocouple to your configuration (Figure 16-51).

```
┌─══Make a NEW Config for...══════════
│  ▸Clear 2x3 Tops
│  ▸Larger Chambers
│  ▸Other Chambers
│  ▾Special Configurations
│      6400-13 TC Adapter...
│      Custom Chamber - Closed...
│  ▾/▸    ♦         ♦         ♦ EXIT  ♦ SELECT
```

```
┌───────────────────────────────────────
│  This program will add an
│
│  ** 6400-13 External Thermocouple **
│
│  to the current configuration.
│  OK to continue? (Y/N)_
└───────────────────────────────────────
```
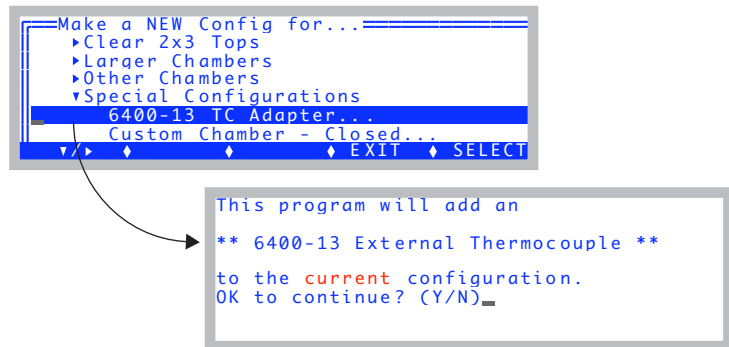
Figure 16-41. Adding a 6400-13 Thermocouple adapter.

When you get done, examine what changed in the configuration tree (Figure 16-42).

```
╔═══Config Menu═══════════════════╗
║    New...                       ║
║    Open...                      ║
║    Save as...                   ║
║    *View/edit...                ║
║                                 ║
║                                 ║
║  ▼/▶   ♦        ♦     ♦ EXIT  ♦ SELECT ║
╚═════════════════════════════════╝
```

```
╔═══FactoryDefault_6.1.xml════════╗
║  ┌─▶comm                        ║
║  ├─▶matching                    ║
║  └─▼* hooks                     ║
║    └─▼* items=  1 items         ║
║ ▶    └ * items[1]=  Set DAC (On NM Enter) ║
║  ▼/▶   ♦Edit    ♦       ♦Revert ♦Done ║
╚═════════════════════════════════╝
```
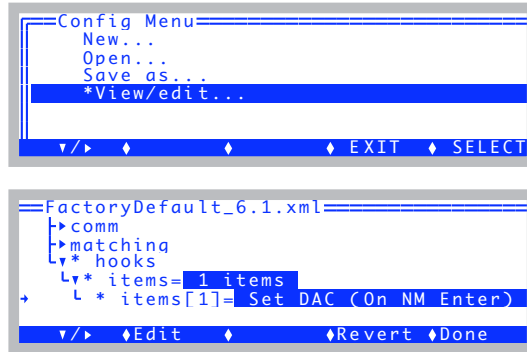
Figure 16-42. Adding the thermocouple adapter adds a hook to the configuration.

If you edit that hook (make sure the cursor is on *items[1]=*), you will see that it consists of three parts: a Label, Code, and a When (Figure 16-43).

```
╔═══════════Hook Item Editor══════╗
║ Label: Set DAC                  ║
║ Code: -3600 18 AOSET            ║
║ When: On NM Enter               ║
║                                 ║
║                                 ║
║  Edit          Test   Cancel  OK ║
╚═════════════════════════════════╝
```
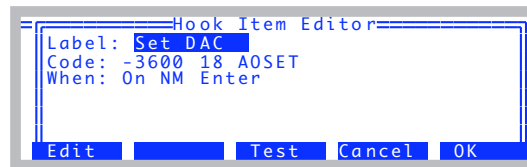
Figure 16-43. The hook for the thermocouple adapter.

The *Label* is just for readability, to remind you what the hook is for. The *Code* does the work. In this case, it sets digital to analog channel number 18 to a value of -3600 mV. The *When* specifies that this is to happen each time New Measurements mode is entered.

If you'd like a *Why*, here it is: The circuitry inside the connector housing for the thermocouple adapter that plugs into the 37-pin connector on the console needs a -3.6V supply at very low current. We are supplying that with a spare D/A channel. It only needs to be done once, so that we could have used a different hook, like On Config, but setting it each time New Measurements mode is entered is a bit safer, in case something strange happens, like someone experimenting with typing AOSET commands at the ok: prompt, for example.

■ **An Experiment**

Let's add a hook that does something we can see, and add it to the list, just to prove that hooks really do work. We'll make a message appear on the screen each time New Measurements is entered, and another each time it is exited.

Start in the Configuration tree, with the cursor on the <open> <hooks> <items> node (Figure 16-44).We'll add our new hooks to the one already there for the 6400-13 Adapter (assuming you followed the example on page page 16-37).

```
==FactoryDefault_6.1.xml=============
├─▶comm
├─▶matching
└─▼* hooks
→  └─▼* items= 1 items
     └ * items[1]= Set DAC (On NM Enter)
   ▼/▶   ◆Edit    ◆        ◆Revert ◆Done
```
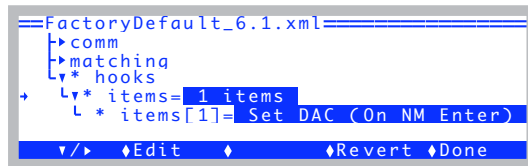
Figure 16-44. Start here, with the cursor on items.

1 **Edit the Hook List**

Make sure the cursor is on *items=*, and press **Edit** (**f2**) to access the list of Hook Items (Figure 16-45).

```
===============Hook Items=============
 1:Set DAC (On NM Enter) -3600 18 AOSE



+Edit    ◆Add    ◆Remove ◆Cancel ◆OK
```
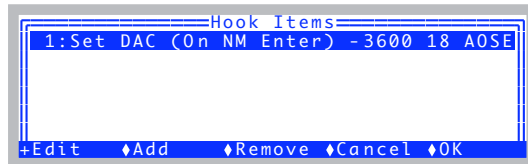
Figure 16-45. The Hook List.

2   Add a hook
Press **Add** (**f2**), and you'll be prompted for the three parts of the hook (Figure 16-46).

1. The label

```
════════════Hook Items════════════
1:Set DAC (On NM Enter) -3600 18 AOSE
╔═════Short label═════╗
║Hello message▄       ║

DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

2. Pick this

```
═══════Execute When?═══════
  On Config              0 18 AOSE
  On NM Enter
  On NM Exit
  On Tic

        ♦        ♦       ♦Cancel ♦OK
```

3. Type this

```
═══════Code for Hello message═══════
"Hello" print getkey▄




+DelLn  ♦ClrEnd ♦DelChar♦Cancel ♦OK
```

Finished

```
════════════Hook Items════════════
1:Set DAC (On NM Enter) -3600 18 AOSE
2:Hello message (On NM Enter) "Hello"



+Edit   ♦Add    ♦Remove ♦Cancel ♦OK
```

Figure 16-46. Add a message for entering New Measurements mode.

What we are doing in the code is putting a string ("Hello") on the stack, then printing it (*print*), then waiting for the user to press any key (*getkey*).

3   Add another hook

We'll add another hook much like our first, only it will say "Good-bye", and will happen when we leave New Measurements mode.

1. The label

```
════════════════Hook Items════════════════
1:Set DAC (On NM Enter) -3600 18 AOSE
2:┌───Short label═══════════════════llo"┐
  │goodbye message▁                      │
  │                                      │
  └──────────────────────────────────────┘

DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

2. Pick this

```
════════════Execute When?════════════
┌──────────────────────────────┐
│On Config               0 18 AOSE
│On NM Enter             ) "Hello"
│On NM Exit
│On Tic
└──────────────────────────────┘

       ♦         ♦       ♦Cancel ♦OK
```

3. Type this

```
═════════Code for goodbye message═════════
"Good-bye" print getkey▁




+DelLn   ♦ClrEnd ♦DelChar♦Cancel ♦OK
```

Finished

```
════════════════Hook Items════════════════
1:Set DAC (On NM Enter) -3600 18 AOSE
2:goodbye message (On NM Exit) "Good-
3:Hello message (On NM Enter) "Hello"



+Edit   ♦Add     ♦Remove ♦Cancel ♦OK
```

Figure 16-47. Add a message for leaving New Measurements mode.

4   **Try it out**
Press **OK** (**f5**) to exit the Hook Items Dialog, and escape out to OPEN's main screen. Then enter and exit New Measurements mode, and you should see your messages (Figure 16-48). You'll have to press a key after each to continue on.

Entering New
Measurements
mode

```
Hello
```

Leaving New
Measurements
mode

```
→CO2R_µml   CO2S_µml   H2OR_mml   H2OS_mml
a    575.0      456.7      9.138     11.209
 ΔCO2_µml   ΔH2O_mml   Flow_µml     RH_S_%
b   -118.2      2.071      499.9      31.13
    Photo       Cond         Ci     Trmmol
c     97.7     0.0767   -1.61E+03       1.75
Good-bye
```

Figure 16-48. The hooks in action.

At this point, you'll either want to go back and remove these items from the list, or change one of the messages to something really sinister[3] that has a 1% chance of appearing, if you'd like to mess with someone's mind.

        rnd .01 < if "good-bye" print then

_____

[3.]Perhaps something like "Your data files have disappeared, along with all hope of graduation".

Adding an External Temp, RH Probe

As an example of using the configuration tools, we add a temperature and humidity probe. The probe in question is a Vaisala Humitter 50Y* (Vaisala, Woburn, MA), although others could be substituted. The important considerations are the range and sensitivity required.

Hardware

Figure 16-49 illustrates how the humidity probe can be connected to the 37-pin input connector on the LI-6400 console (for a summary of all the pins on that connector, see Table 16-7 on page 16-36).
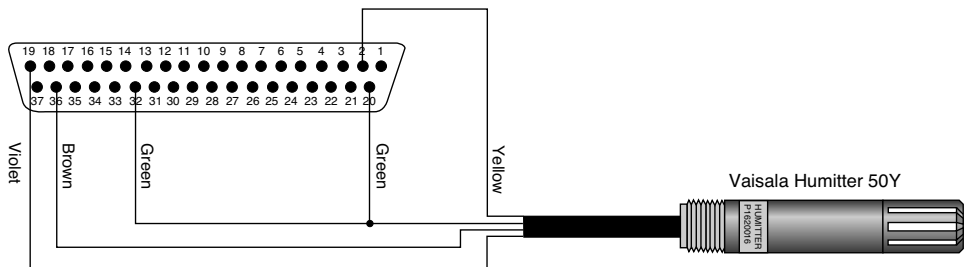
Figure 16-49. A wiring diagram for the humidity probe.

Table 16-8. Wiring the Vaisala Humitter to the 37-pin connector.

Pin	Description
2	Power
19	Channel 21 signal input (Temperature)
20	Power Ground
32	Signal input ground
36	Channel 20 signal input (RH)

Software

In software, we will enable the two physical channels that are measuring temperature and humidity, and also create two user-defined entities so we can compute the values, display them log them, etc. There are a couple of ways to do this (Chapter 15), but we'll use the "extras" method.

1   Edit the <extras> node

Go to `Config Menu|View/edit` and navigate to the <open> <comps> <file> <extras> node.
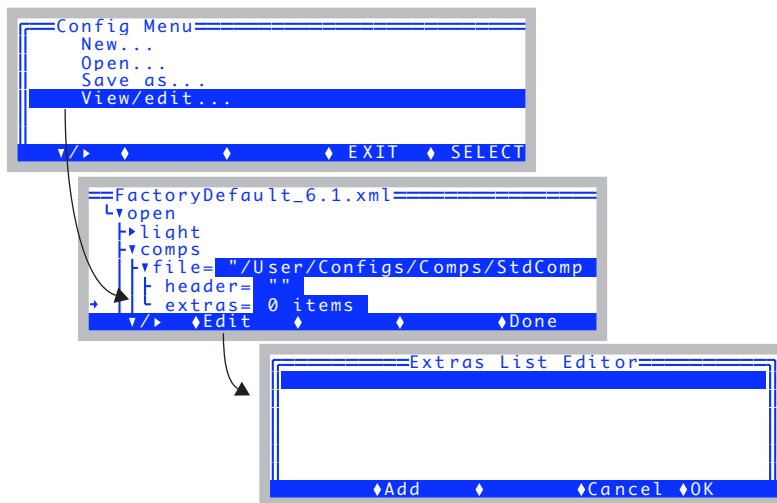
Figure 16-50. The temperature and humidity variables will be created as "extras".

2   Add External Temp

Press **Add** (**f2**), select `Expression`, name it xTair (Figure 16-51).
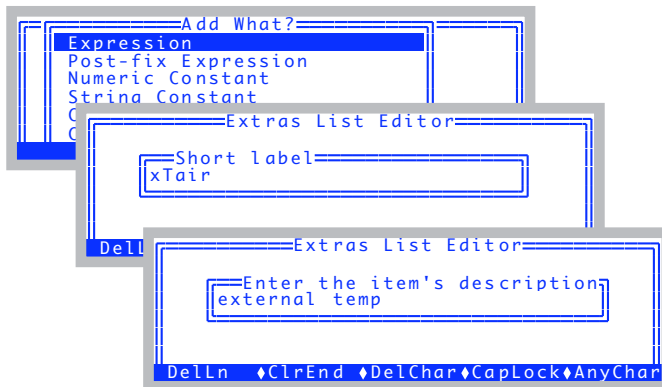
Figure 16-51. Adding xTair as an expression.

3   Add the equation

You will be asked when to do the computation, and what the formula is (Figure 16-52). For this probe, the relation between temperature and voltage is $T = 100V - 40$, but since the signal reported by the LI-6400 is in mV, the multiplier becomes 0.1.

Press **B**

```
════════════Extras List Editor═════════
  ════════════Computed Item══════════
  Compute this Before or After
  normal computations?
  ════════════════════════════(B/A)════
```

This is the equation.
Press **f5** when done.

```
═══════════════Code for u1000═══════════
u1000 = chan21_mv * 0.1 - 40_



◆DelLn  ◆ClrEnd ◆DelChar◆Cancel ◆OK
```
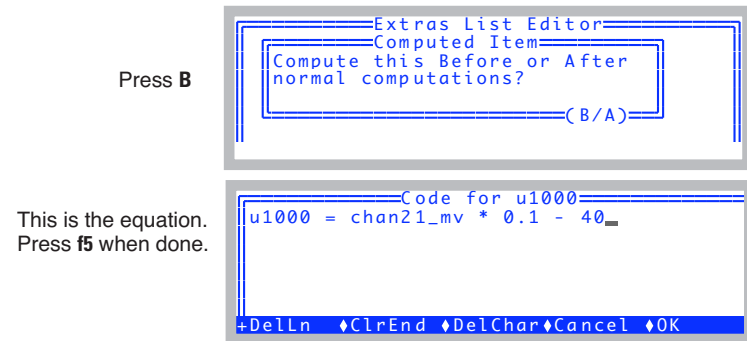
Figure 16-52. Adding the equation.

Before or *After* specifies when in relation to the ComputeList the quantity should be computed. We choose *before*, since the only variable we need to compute *xTair* is *chan21_mV*, which is measured and ready before extras or the ComputeList is computed.

4   Finalize the definition

You then see the Extra Item Editor, from where you can make final adjustments. We'll use it to change the format to *Fixed*, with 2 significant digits (so it will appear as 12.34, for example).

```
═══════════════Extras Item Editor═══════
ID: 1000   Label: xTair
Desc: external temp
Variable name: u1000
Type: Expression
Format: Fixed Point    Digits: 2
Code: u1000 = chan21_mv * 0.1 - 40
  Edit          Test   Cancel   OK
```
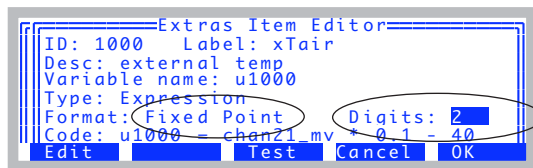
Figure 16-53. The final state.

When done, press **f5** (**OK**), and the Extras List Editor will look like Figure 16-54.
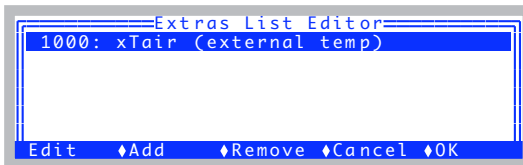
Figure 16-54. Added xTair to the Extras List Editor.

5   **Add xRH**
Follow the same procedure to add xRH. When you get to the Extras Item Editor, it should look like Figure 16-55.
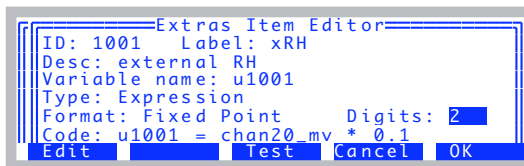
Figure 16-55. The external RH definition.

When it does, press **OK** (**f5**), and the Extras List Editor will show both additions (Figure 16-56).
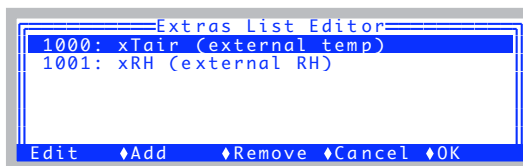
Figure 16-56. The List Editor with both variables.

6   **Add to Display and Log List**
When you press **OK** (**f5**) to exit the list editor, you'll be given the opportunity to add these quantities to the new measurements display, and to what is logged (Figure 16-57).

Press **Y** to each of these.

```
══════════Extras List Editor══════════
Add xTair (ID#1000) to Display (Y/N)?Y

Add xTair (ID#1000) to LogList (Y/N)?Y

Add xRH (ID#1001) to Display (Y/N)?Y
Add xRH (ID#1001) to LogList (Y/N)?
```

We'll find them on display line *m*

```
Display Additions...
xTair -> line m
xRH -> line m
Press any key
```

Figure 16-57. Adding the new variables to the display and LogList.

7   **Enable the channels 20 and 21.**
Navigate to the <open> <a2d> node (Figure 16-58).

```
══FactoryDefault_6.1.xml══════════════
└▼a2d
  ┌ avgtime= 4.0 secs
  └▼userchans
◆   ┌ ch20= "Off"
    ├ ch21= "Off"
    ├ ch22= "Off"
 ▼/▶  ◆Edit  ◆       ◆Undo   ◆Done
```

Figure 16-58. The <open> <a2d> node.

8   **Enable channel 20, with ground 5.**

Edit the <chan20> node.

Turn it on.

```
          User Channel 20 (Pin #36)

State:  ON
Ground: Chan 3, Pin 31 (Chmbr Gnd)
Sampling:  25 Hz


   Edit                        Cancel   OK
```

Set ground to 5.

```
          User Channel 20 (Pin #36)
=============Ground Options============
St  Chan 3, Pin 31 (Chmbr Gnd)
Gr  Chan 4, Pin 14 (Press Gnd)
Sa  Chan 5, Pin 32
    Chan 6, Pin 15
    Chan 7, Pin 17
          ◆         ◆       ◆Cancel  ◆OK
```

Press **OK**.

```
          User Channel 20 (Pin #36)

State:  ON
Ground: Chan 5, Pin 32
Sampling:  25 Hz


   Edit                        Cancel   OK
```
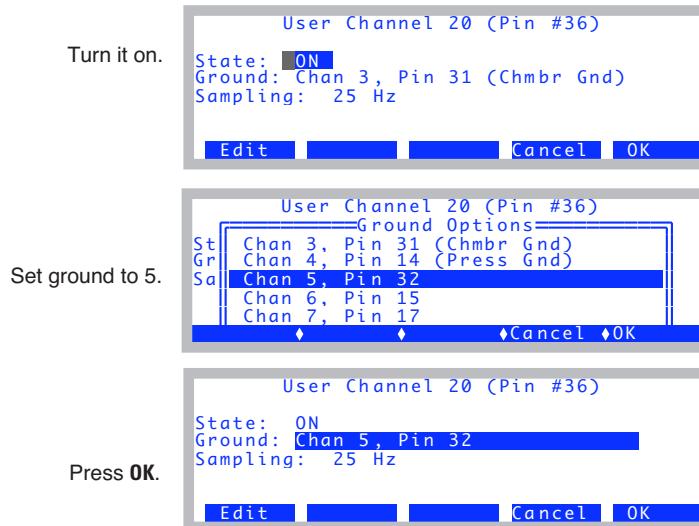
Figure 16-59. Defining channel 20.

9   **Edit the channel 21 the same way.**

Edit the <chan21> node to look like Figure 16-60.

```
          User Channel 21 (Pin #19)

State:  ON
Ground: Chan 5, Pin 32
Sampling:  25 Hz


   Edit                        Cancel   OK
```
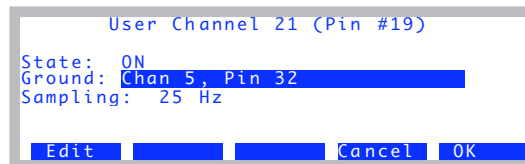
Figure 16-60. Defining channel 21.

The <a2d> nodes should now look like Figure 16-61.
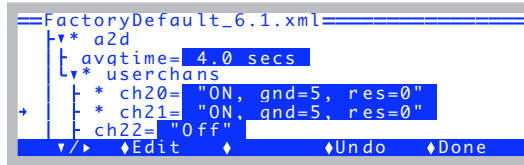
Figure 16-61. Channels 20 and 21 ready.

10 **Save your work**

Save this configuration using `Config Menu|Save as`.
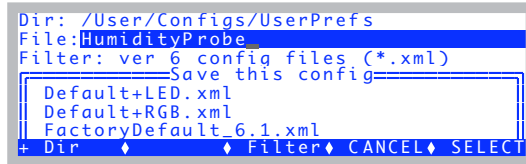
Figure 16-62. Save the config.

The changes you just made can be summarized in the <open> <comps>
<file> <extras> and <open> <a2d> nodes, as shown below.

```
<open>
    <comps>
        <file>
            <extras>
                <extras[1]>
                    <id>1000 </id>
                    <label>"xTair"</label>
                    <desc>"external temp"</desc>
                    <varname>"u1000"</varname>
                    <type>1 </type>
                    <code>"u1000 = chan21_mv * 0.1 - 40"</code>
                    <when>0 </when>
                    <initsval>""</initsval>
                    <initdval>0 </initdval>
                    <strscrlen>1 </strscrlen>
                    <fmt1>1 </fmt1>
                    <digs>2 </digs>
                    <commitem>1 </commitem>
                </extras[1]>
                <extras[2]>
                    <id>1001 </id>
                    <label>"xRH"</label>
                    <desc>"external RH"</desc>
                    <varname>"u1001"</varname>
```

```
                              <type>1 </type>
                              <code>"u1001 = chan20_mv * 0.1"</code>
                              <when>0 </when>
                              <initsval>""</initsval>
                              <initdval>0 </initdval>
                              <strscrlen>1 </strscrlen>
                              <fmt1>1 </fmt1>
                              <digs>2 </digs>
                              <commitem>1 </commitem>
                      </extras[2]>
                  </extras>
          </file>
      </comps>
      <a2d>
          <avgtime>4.0 secs</avgtime>
          <userchans>
              <ch20>"ON, gnd=5, res=0"</ch20>
              <ch21>"ON, gnd=5, res=0"</ch21>
              <ch22>"Off"</ch22>
              <ch23>"Off"</ch23>
          </userchans>
      </a2d>
</open>
```

Configuring for Alternative Materials

While the LI-6400 is normally used on individual broadleaves, it can also be used to measure needles, whole plants, bryophytes, fruit, insects, and just about anything else that you might be able to fit into a chamber.

The Material= Node

For many of the configurations that you can generate with the configuration builder, there is a node (Figure 16-63) for selecting the material being measured. We reduce it to three choices: broadleaves, needles, or mass-based.
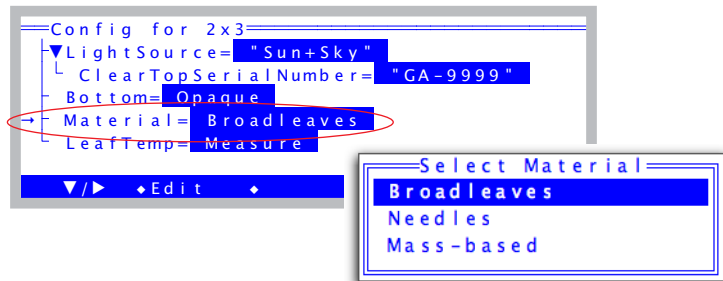
Figure 16-63. Many configurations have a Material= *node, allowing you to tailor the configuration to the material being measured.*

If you pick Broadleaves, the boundary layer conductance will be determined based on lookup table based on leaf area and fan speed, or fixed value, depending on the chamber involved. If you pick needles, boundary layer will be a fixed value and not use a lookup table. Both choices result in photosynthesis and transpiration being area-based, and provide for a user-entered leaf area and stomatal ratio.

If you pick mass-based, there are a number of implications: Only photosynthesis and transpiration are computed (no stomatal conductance, C_i, or other leaf-based quantity.). The units of photosynthesis are μmol kg^{-1} s^{-1}, and the units of transpiration are mmol kg^{-1} s^{-1}. Leaf area, boundary layer conductance, and stomatal ratio are no longer needed; instead, what used to be the Area key (**f1** level 3) will prompt for mass in grams.(Figure 16-64).

mmol kg^{-1} s^{-1}.

μmol kg^{-1} s^{-1}.

grams

Figure 16-64. Mass-based configuration in New Measurements. Only photosynthesis and transpiration are computed, and they have different units; mass rather than area is entered via **f1** *level* **3**.

6400-17 Whole Plant Chamber

When measuring a complete canopy, you have to be concerned about potential sources and sinks of CO_2 and H_2O besides the plant - namely, the container and soil holding the plant, if they are included in the measurement volume. LI-6400 Application Note 4 (available on the 6400 CD) describes ways to minimize these effects with the 6400-17 WPA Chamber.

Stomatal conductance is often ignored in whole plant measurements. There are several reasons for this, including: 1) The value is non-linearly related to leaf temperature, and in a whole plant there will likely be a distribution of temperatures. 2) There are potential sources of transpiration/evaporation through non-stomatal sources (stems, petioles, soil, etc.). 3) Given the variety of leaf sizes and shapes, there will be a distribution of boundary layer conductances in play, making it very dicey to obtain a stomatal conductance from a total conductance.

Once you've eliminated stomatal conductance from the list of things you are interested in calculating, it takes some other things with it: all C_i-related values, leaf (or plant) temperature, and any energy balance options that may go into leaf temperature.

To make a configuration for the Whole Plant Chamber, use Config Menu|New, and pick 6400-17 WPA from the menu.

```
┌─Make a NEW Config for...══════════════┐
│  ▼Larger Chambers                     │
│      6400-07 or -11 2x6 Chamber...    │
│      6400-05 Conifer Chamber...       │
│      6400-17 Whole Plant Arabidopsis..│
│      6400-22 Opaque Conifer Chamber...│
│      6400-24 Bryophyte Chamber...     │
│  ▼/▶   ◆        ◆        ◆ EXIT ◆ SELECT│
└───────────────────────────────────────┘
```

```
┌─Make a NEW Config for...══════════════┐
│      ══Select a quantum sensor══      │
│      │Q34100                      │  .│
│      │                            │   │
│      │                            │sis..│
│      │                            │er...│
│      └────────────────────────────┘   │
│      ◆Add       ◆       ◆Cancel ◆Select│
└───────────────────────────────────────┘
```

```
┌══Config for WholePlantChamber══════════┐
│→├ LightSource= "Sun+Sky"              │
│ ├ ExternalQuantum= "Q34100"           │
│ ├ Material= Broadleaves               │
│ ├ Computations= Photo, Trans, Cond, Ci│
│ └ Matching= Normal                    │
│                                       │
│  ▼/▶   ◆Edit   ◆       ◆Cancel ◆DoIt  │
└───────────────────────────────────────┘
```
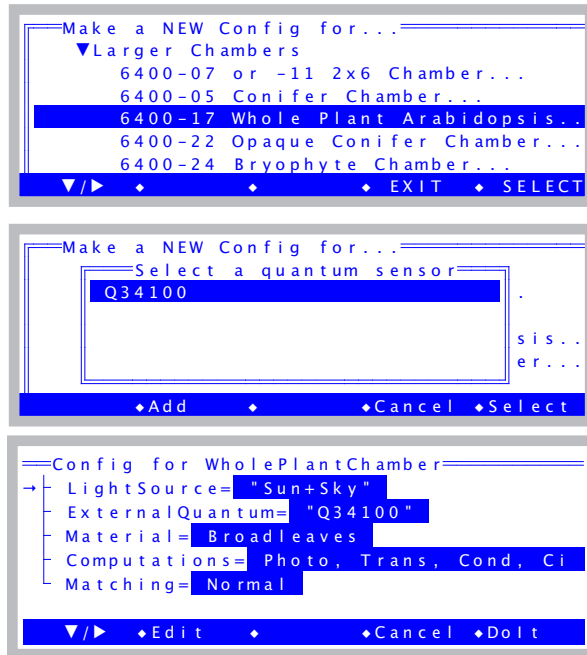
Figure 16-65. Making a configuration for the 6400-15, -17, -22, and -24 chambers all follow this pattern.

The "Computations" node lets you select between the normal gas exchange quantities (photosynthesis, transpiration, stomatal conductance, C_i, etc.) and a restricted set of just photosynthesis and transpiration. The latter is useful for whole plant measurements or when making non area-based measurements. If you set the Material to mass-based, this node will automatically be set to the reduced set.

The "Matching" node lets you enable fan-based matching, as opposed to normal matching which uses the match valve. See **Matching Methods** on page 16-19.

Once the configuration is to your liking, press **f5** (**DoIt**) and **N** to implement and name the configuration.

6400-24 Bryophyte Chamber

The Bryophyte chamber configuration builder is like what is shown in Figure 16-65.

If you are measuring very moist samples (e.g. mosses) in this chamber (or any chamber), there are some operational considerations to keep in mind. If the chamber temperature gets above ambient temperature (not hard to do if you are illuminating it with a light source), then with moist samples, you run a very real risk of condensation forming in the exhaust tube that takes chamber air to the match valve. If that happens, matching will introduce serious errors, since the air entering the reference IRGA will be wetter than the air leaving the sample cell. To prevent this problem, you can do a couple of things: 1) Run the coolers to keep the chamber temperature below ambient. 2) Consider changing the match method to fan based. (See **Matching Methods** on page 16-19). Fan-based matching will take exhaust tube issues out of the picture.

6400-22 Opaque Conifer Chamber

The Opaque Conifer Chamber chamber configuration builder follows what is shown in Figure 16-65.

Insects

The configuration builder has an insect option (Figure 16-66).

Config Menu, then New...

```
═Make a NEW Config for...═
 ►Clear 2x3 Tops
 ►Larger Chambers
 ►Other Chambers
 ▼Special Configurations
   Insect Respiration...
   6400-13 TC Adapter...
 ▼/►    ◆         ◆        ◆ EXIT   ◆ SELECT
```

No options. Just press N.

```
Config Modifications Ready. Pick one:

N) implement as a New config named:
   'Insect Rd.xml'
C) apply to the Current config
V) View the modifications

<escape> - discard and quit
```
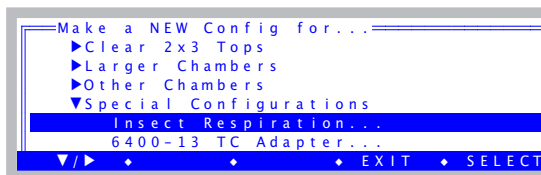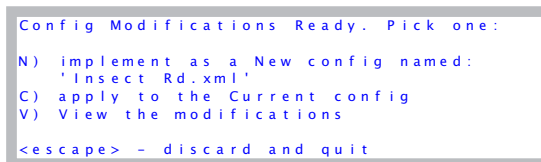
Figure 16-66. Making a configuration for insect respiration.

In New Measurements mode, it looks like Figure 16-67.

*Figure 16-67. The insect respiration configuration in New Measurements mode. Respiration has units of $(\mu g\ CO_2)\ (g\ insect)^{-1}\ min^{-1}$, and **f1** level 3 prompts for insect mass in grams.*

The compute list is shown in Figure 16-68.

```
/* Insect Respiration
*/
##141 "Mass" "Insect mass (g)"
"ucon(1)"

##142 "fdg" "flow / Insect Mass"
" (flow_um * 1E-6) / #141 "


/* humidity
*/
##20 "Trans" "Transpiration (mol/m2/s)"
"(h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm) * #142"

##21 "Trmmol" "Transpiration (mmol/m2/s)"
" #20 * 1E3"

/* respiration */
##145 "Resp" "Respiration (ugCO2/gInsect/min)"
" (co2_1_um - co2_2_um * (1000 - h2o_1_mm) / (1000 - h2o_2_mm)) * -2640.588 * #142 "
```

Figure 16-68. The compute list for the insect respiration configuration. This will be found in "/User/Configs/Insect Rd" once you build the configuration.

Interfacing Custom Chambers

This section was originally published as an Application Note entitled "Interfacing Custom Chambers to the LI-6400 Sensor Head", document number PPS231. It has been updated here for subsequent software changes.

Introduction

The LI-6400 can use custom-made chambers for a variety of applications. Adapting the LI-6400 for whole plant canopy gas exchange measurements is discussed in LI-6400 Application Note 2. Another approach using a smaller chamber is described here, along with some general considerations. We complete the example by measuring photosynthesis of Prostrate Spurge (a weed with tiny leaves growing close to the ground) and the respiration rate of detached fruit. This application illustrates how other shapes and sizes of chambers can also be attached to the sample IRGA cover plate of the LI-6400 sensor head and demonstrates the adaptability of the LI-6400 for a variety of gas exchange configurations.

As a precautionary note, departing from a standard 2x3 cm chamber with light source involves losing control of illumination and also compromising some humidity and temperature control. Sometimes modifying the measurement protocol can avoid the need for a special chamber altogether. This note is for cases where a custom chamber is clearly necessary.

Considerations for Building Your Own Chamber

Two design characteristics of sample chambers are shape and volume. These characteristics not only affect the size and shape of the subject material that can be measured, but also have a bearing on air flow rates and consequently the magnitude of the reference and sample chamber concentration differences. In addition, as chamber volume increases, lack of control of chamber temperature and humidity may become an issue. It may also become necessary to install a fan inside the chamber to achieve adequate mixing.

Chamber Volume, Time Constant and Mixing

For chamber volumes smaller than about two liters, the mixing fan within the sample chamber IRGA provides adequate mixing of the chamber volume. For larger chambers, an additional fan(s) should be installed within the chamber. For a thoroughly mixed, open, flow-through system, the chamber concentration $C(t)$ at time t, is given by

$$C(t) = C_e - (C_e - C_o)e^{-ft/V} \qquad (16\text{-}1)$$

Where C_o is the initial chamber concentration, C_e is the entering concentration, f is the airflow rate and V is the chamber volume. By definition, the time constant τ of the chamber is the time it takes for the chamber concentration to get to within $1/e$ or about 63% of the final concentration. Equation (16-1) shows that τ is given by:

$$\tau = \frac{V}{f}$$ (16-2)

For example, for a well-mixed chamber of volume 2000 cm^3 and an airflow rate of 700 μmol s^{-1} (15.7 cm^3 s^{-1}), the time constant is 127 seconds (Figure 16-69). By contrast, the standard 2x3 cm LED light source chamber has a volume of about 80 cm^3 and at a flow of 700 μmol s^{-1}, its time constant is about 5 seconds.
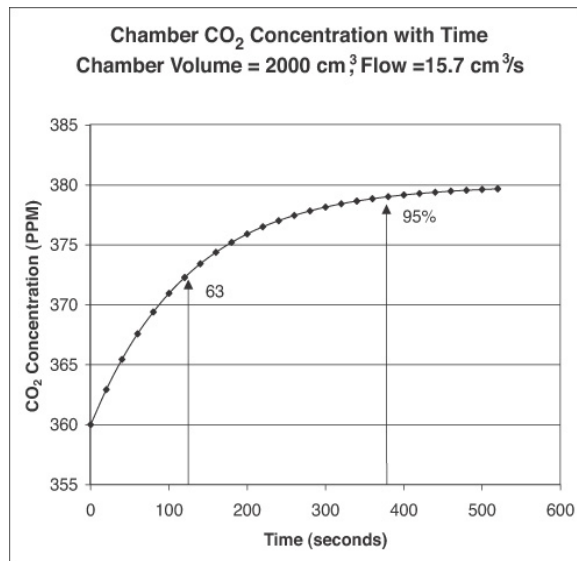
Figure 16-69. The theoretical change in chamber CO$_2$ concentration with time for a 2 liter chamber.

Starting with an initial chamber concentration of 360 ppm, and an incoming flow of 700 μmol s^{-1} (15.7 cm^3 s^{-1}) and concentration C_e of 380 ppm, the chamber concentration changes 95% of the ultimate change after about 380 seconds.

Leaf Area, Flow Rate and CO_2 Differentials

One of the main reasons for custom designing a chamber is to increase the amount of plant material that can be enclosed within the chamber, in order to increase the precision of measuring low rates of exchange (e.g. when measuring plant respiration or low rates of photosynthesis). To find the minimum sample area required to obtain a given measurement precision, do the following analysis:

Assume:
1) The CO_2 noise is ± 0.2 ppm, peak-to-peak.
2) We desire a 2% measurement precision.
3) Minimum flow is 200 μmol s^{-1}.

To get a 2% measurement precision of the CO_2 concentration differential (difference between reference and sample chamber concentration), the differential should be at least 10 ppm (this is obtained by dividing analyzer noise by the required precision, 0.2/0.02, or 10 ppm).

Next we consider how to generate a 10 ppm CO_2 differential. This will obviously depend on the amount of plant material enclosed within the measurement chamber, its CO_2 exchange rate, and the air flow rate through the system.

Ignoring the dilution effects of water vapor (see Von Cammaerer and Farqhuar, 1981), the gas exchange rate in an open flow through system is given by

$$A(\mu mol/m^2/s) = \frac{f(\mu mol/s) \times \Delta CO_2(\mu mol/mol)}{S(cm^2) \times 100} \qquad (16\text{-}3)$$

Assuming a fairly low yet practical airflow rate of 200 μmol s^{-1}, the minimum sample area required to generate a 10 ppm CO_2 differential is given by:

$$S(cm^2) = \frac{f(\mu mol/s) \times \Delta CO_2(\mu mol/mol)}{A(\mu mol/m^2/s) \times 100} \qquad (16\text{-}4)$$

If we wanted to measure a CO_2 exchange rate of 1.0 μmol m^{-2} s^{-1} with a precision of 2%, equation (16-4) predicts the required chamber should enclose a minimum leaf area of about 20 cm^2.

Other ways in which we could improve the measurement precision of low activity material would be to increase the signal averaging time. For example, increasing the signal averaging time from 4 seconds to 16 seconds would cut the noise in half.

Figure 16-70 shows the relationship between flow (μmol s^{-1}) and CO_2 differentials for various multiples of CO_2 assimilation rates (A, in μmol m^{-2} s^{-1}) and leaf area (S, in cm^2). To determine the expected CO_2 differential, multiply the assimilation rate expected by the leaf area to be enclosed within the chamber, and follow the appropriate graph line to the air flow rate desired. For example, assume a 5 cm^2 leaf area with a net assimilation rate of 5 μmol m^{-2} s^{-1}. Use the 25 line (5x5=25) to predict a CO_2 draw-down of 10 ppm when the air flow rate is 250 μmol s^{-1}.
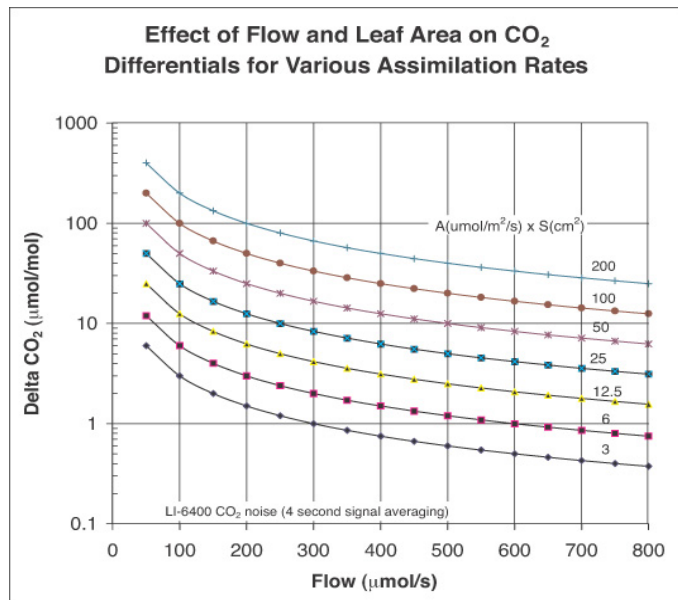
Figure 16-70. Relationship between flow and CO_2 differentials for various combinations of CO_2 assimilation rates and leaf area.

Leaf Area, Flow Rate and Humidity Control
Enclosing actively photosynthesizing and transpiring plant material within the measuring chamber will not only lower the chamber CO_2 concentration,

but will also raise humidity levels. The primary limitation here is the supply of dry air. If transpiration exceeds the flow rate of dry air, then chamber humidity will rise and condensation will occur. Lower flow will raise humidity, while a lower transpiration rate requires a lower flow rate of dry air.

Combining equations (1-4) and (1-7) on page 1-8, stomatal conductance, g (mol m^{-2} s^{-1}), can be expressed as:

$$g = \left[\frac{f(W_s - W_r)}{100S(1000 - W_s)} \right] \left[\frac{1000 - \dfrac{W_L - W_s}{2}}{W_L - W_s} \right] \qquad (16\text{-}5)$$

Where f is the airflow rate (μmol s^{-1}), S is the leaf area (cm^2), W_s is the sample chamber water vapor mole fraction (mmol mol^{-1}), W_r is the incoming or reference chamber water vapor mole fraction (mmol mol^{-1}), and W_L is the water vapor mole fraction (mmol mol^{-1}) within the leaf air spaces (this value is calculated using the leaf temperature measurement).

$$g \approx \left[\frac{f(W_s - W_r)}{100S} \right] \left[\frac{1}{W_L - W_s} \right] \qquad (16\text{-}6)$$

solving for W_s we get:

$$W_s = \frac{kW_L + Wr}{1 + k} \qquad (16\text{-}7)$$

where

$$k = \frac{100gS}{f} \qquad (16\text{-}8)$$

If leaf and air temperatures are the same, then the highest value of W_s possible before condensation would occur would be W_L. The lowest value possible (at very high flow rates) would be W_r. Figure 16-71 shows the relationship between airflow and chamber humidity for various multiples of stomatal conductivity and leaf area. W_s is shown as a fraction of the value between W_r and W_L.

Example: Flow = 700 μmol s^{-1}, dry incoming air (W_r=0), Tleaf (30 °C (W_L = 43 mmol mol^{-1}), leaf area = 50 cm^2, g = 0.15 mol m^{-2} s^{-1}. W_s would be about halfway between W_r and W_L or about 22 mmol mol^{-1}. If W_r were 20, then W_s would be about 32 mmol mol^{-1}.
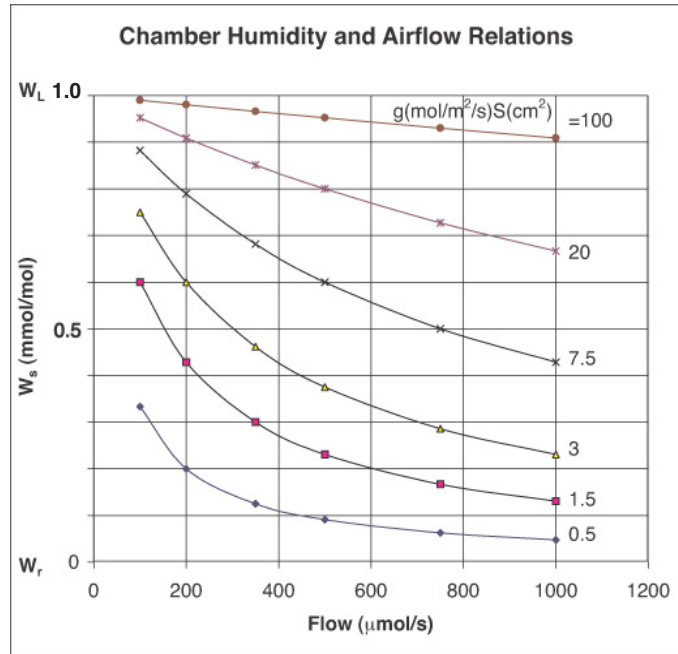
Figure 16-71. Relationship between air flow and chamber humidity for various multiples of stomatal conductivity and leaf area.

Chamber Temperature Control

The standard 2x3 cm LED chamber has a volume of about 80 cm^3. Under normal field conditions, the temperature control range of the LI-6400 is about ±6 °C from the ambient temperature. With a larger chamber, the temperature control range possible with the standard thermoelectric coolers will be reduced.

Interfacing Custom Chambers to the LI-6400 IRGA

Figure 16-72 shows the LI-6400 sensor head from which the latching handle, the leaf chamber and the sample IRGA cell cover plate have been removed. The mixing fan can be seen within the sample IRGA cell.

Figure 16-72. The LI-6400 sensor head, leaf chamber removed.

Attaching a custom chamber involves replacing the sample IRGA cell cover plate with either the mounting plate used for the 6400-09 Soil Chamber (part # 9864-174) or the mounting plate used on the 6400-05 Conifer Chamber (part # 9864-157) (Figure 16-73). The Soil Chamber mounting plate views are on the right hand side. While either mounting plate will provide a suitable base for attaching a custom chamber, the 9864-174 mounting plate provides a larger base area than the 9864-157 plate, and is suitable for larger chambers.

The air inlet manifolds, attached to the reverse side of these plates, can be seen in Figure 16-73. These manifolds are a little different from each other, and from the inlet manifold used with the other standard chambers. The correct air inlet manifold must be used with each type of mounting plate (part # 9864-158 for the Conifer Chamber mounting plate and part # 9864-032 for the Soil Chamber mounting plate). During operation, the mixing fan pushes air out of the two outer holes of the mounting plate. The air returns to the sample IRGA cell via the larger middle hole of the mounting cover plate.
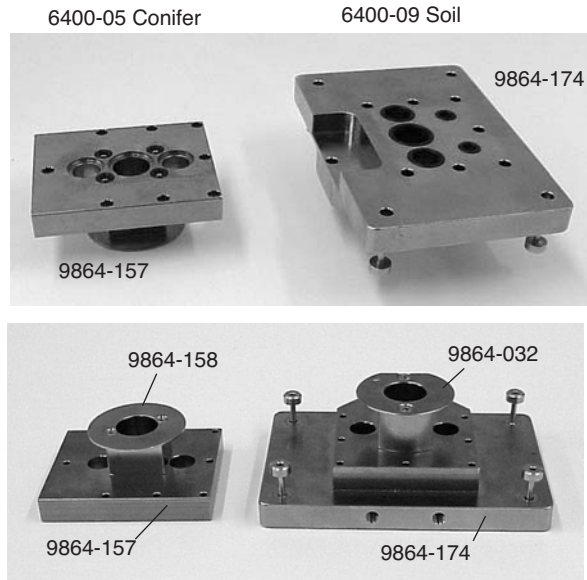
Figure 16-73. Interface plates and fan shrouds for the conifer and soil chambers.

Figure 16-74 shows the 9864-174 Soil Chamber mounting plate and the 9864-157 Conifer Chamber mounting plate attached to the sample IRGA cell. The orientation of the three holes (with respect to the sensor head body) on the conifer chamber mounting plate is at right angle to the orientation of the holes on the soil chamber mounting plate. The difference in orientation does not affect circulation or mixing of the chamber air.

9864-174 (for Soil chamber)

9864-157 (for Conifer chamber)

Figure 16-74. The attachment plates mounted on the sensor head.

Figure 16-75 shows a relatively large (approximately 2 liters in volume) custom chamber made from a piece of Plexiglas tubing. The tube diameter is about 14 cm. The chamber was made by cutting an approximately 13.5 cm length of the tube, and covering one end of the tube with Propafilm (LI-COR part # 250-01885) using double-stick tape. The other end of the tube was left open. Note the three holes in the chamber wall and the machined flat surface for matching up against the three holes of the mounting plate. Also present was another small hole (not visible) to which a chamber exit tube leading to

the Match Valve could be attached (but see Matching and Software Considerations below).

Figure 16-75. A section of Plexiglas tube to be used as a chamber.

Figure 16-76 shows the Plexiglas tube chamber attached to the LI-6400 sensor head using the Soil Chamber and the Conifer Chamber mounting plates. Although in this design a piece of Plexiglas tube was used to make an open-ended cylindrical chamber, the user is free to employ any shape that is appropriate for a particular application.
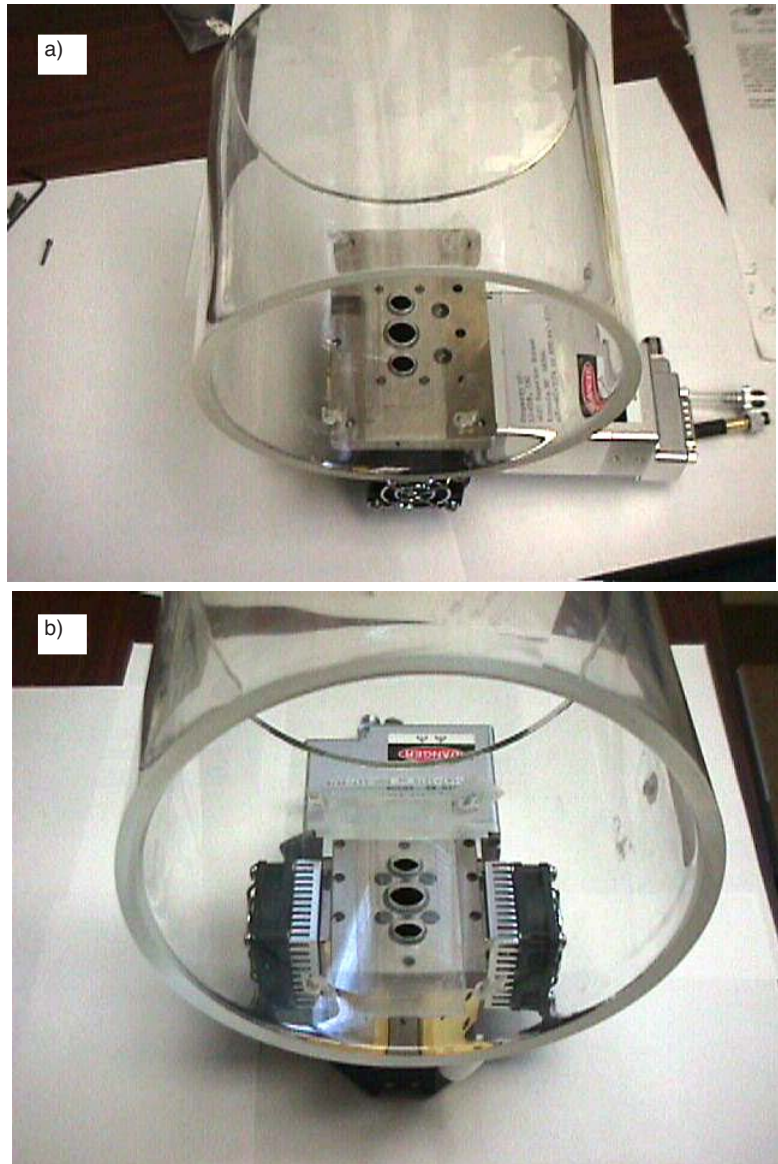
Figure 16-76. The Plexiglas tube chamber mounted using the a) soil chamber plate, and b) the conifer chamber plate.

Chamber Material and Water Vapor

An important consideration in chamber design is the chamber material and its water sorption/desorption properties. Water vapor tends to sorb and desorb slowly on most types of surfaces, and may require long periods of time (approximately one hour was needed for the chamber shown in Figure 16-78 on page 16-69) to establish equilibrium. If adequate time is not allowed for the water vapor readings to stabilize, then transient sorption/desorption effects can cause the apparent leaf transpiration rates to appear smaller or larger than they actually are. Therefore, stomatal conductance readings (or their derivatives such as intercellular CO_2 concentration) will be in error. To minimize these effects the chamber walls can be coated with Teflon tape (LI-COR part # 212-02314). In the above chamber example, stomatal conductance values were not of interest and the chamber walls were not coated.

Matching and Software Considerations

When making open, flow-through measurements, chamber volume is not used in calculations, so the standard configuration in OPEN is fine. The only change needed is the protocol used to match the analyzers. For a large chamber such as the one described above, the chamber time constant even at a high flow rate of 700 μmol s^{-1} is several minutes (Figure 16-69 on page 16-57). This means the sample chamber concentration may take a long time to stabilize, which could cause problems during matching. The simplest way to deal with this is to turn off the mixing fan so that the sample chamber air does not come back into the sample IRGA cell. This is readily accomplished by the configuration node <open> <match> (Figure 16-77).
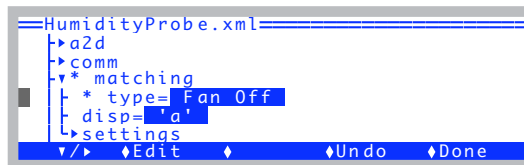
Figure 16-77. Use Config Menu | View/Edit *to set the match type to* Fan Off.

With respect to matching, the Plexiglas chamber described above had a chamber exhaust tube leading to the Match Valve. The main purpose of this exhaust tube was to provide a route for the sample air stream to escape when the chamber was sealed. This occurred during fruit respiration measurements (described below) when the chamber was placed on a flat surface. When using the chamber over soil, the exhaust tube was not necessary, because soil is

porous and the chamber did not seal well against the ground. With a completely sealed chamber design, a chamber exhaust tube should always be installed, even though it may not be used for matching IRGAs.

Measurement Examples

Effect of Temperature on Fruit Respiration

The Plexiglas custom chamber is used for measuring the respiration rate of a banana (Figure 16-78 on page 16-69). Although it is not immediately apparent from the figure, the top end of the Plexiglas tube is covered with Propafilm. The lower end of the Plexiglas tube remained open-ended. Note that the standard leaf thermocouple has been replaced with a longer "E-type" thermocouple wire which has been taped on the fruit for measuring its skin temperature. The banana had a surface area of about 220 cm^2. The CO_2 mixer was used to set the reference CO_2 concentration to 400 μmol mol^{-1}. Temperature was raised using temperature control in New Measurements (**f4** level 2).

With a flow rate of around 700 μmol s^{-1}, the respiring banana generated a delta CO_2 concentration of over 300 μmol mol^{-1}. The measurement was also repeated on a pear with a surface area of about 175 cm^2. Figure 16-79 shows the respiration rate of a banana and pear as affected by the fruit skin temperature.

The advantage of using a large chamber to measure low rates of exchange was evident in that even with respiration rates of less than 2 μmol m^{-2} s^{-1} and flow rates of 700 μmol s^{-1}, the large area of the pear generated ΔCO_2 of over 50 μmol mol^{-1}. This provided stable, high-precision measurements of the respiration rate.
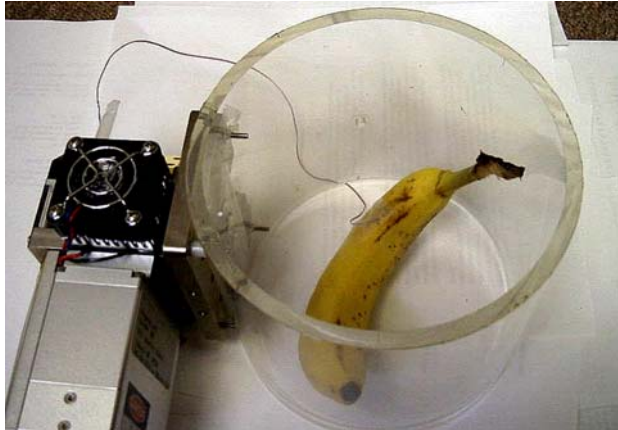
Figure 16-78. Measuring fruit respiration with the custom chamber.
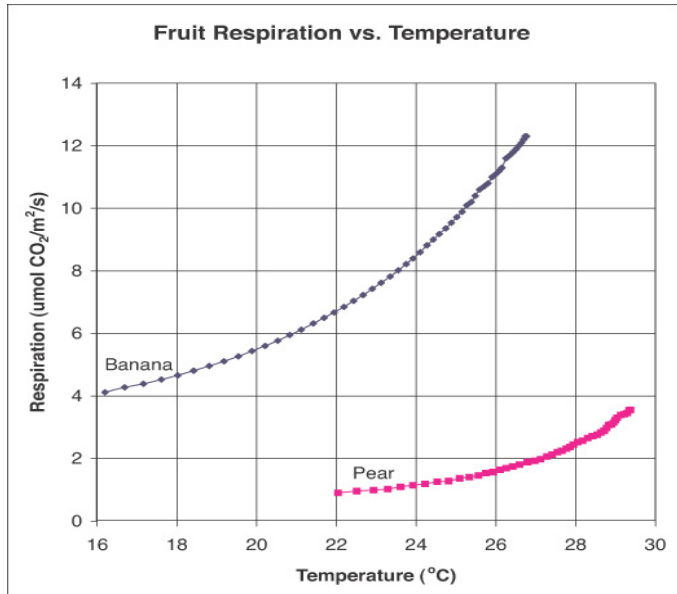
Figure 16-79. Respiration rate of a banana and pear as affected by the fruit skin temperature.

Measurement of Photosynthetic Rate of Prostrate Spurge (Euphorbia prostrata)

The Plexiglas chamber is shown measuring the photosynthetic rate of Prostrate Spurge, a creeping weed (Figure 16-80). An airflow of 700 μmol s^{-1} and an incoming CO_2 concentration of around 400 μmol mol^{-1} were used. The chamber area was 154 cm^2 and photosynthesis was calculated on a unit ground area basis for this exercise. The measurements were taken on a cloudy day, so that light levels varied as clouds moved across the path of the sunlight.

Figure 16-80. Plexiglas chamber measuring Prostrate Spurge

The photosynthetic rate of the Prostrate Spurge as affected by sunlight (Figure 16-81). Placing a cardboard box over the chamber resulted in zero light level, and provided an estimate of the soil respiration component (about 2 μmol m^{-2} s^{-1}), which was subtracted (Figure 16-81).
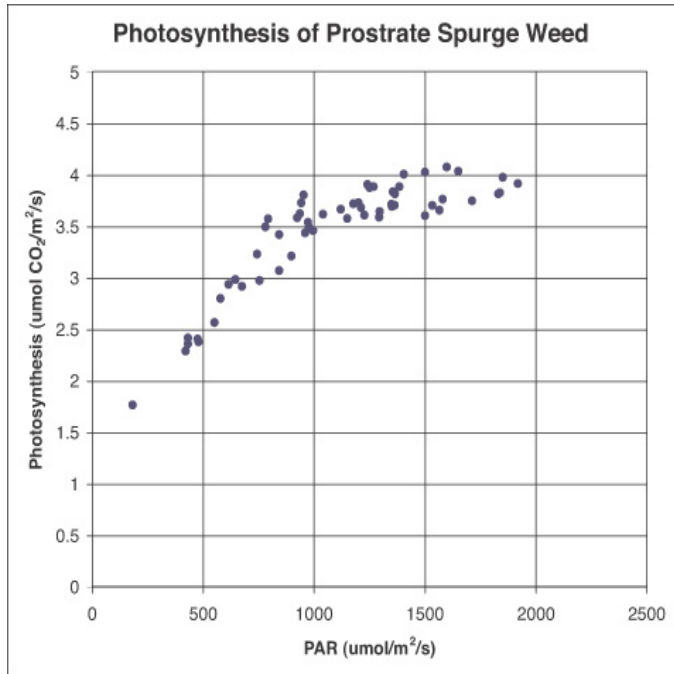
Figure 16-81. Photosynthetic rate of Prostrate Spurge as affected by sunlight.

This chamber was also used to measure the photosynthetic rate on an actively growing grass sward (Figure 16-82). The grass had photosynthetic rates of around 15 μmol m^{-2} s^{-1} (on a unit ground area basis). However, even with an airflow rate of 700 μmol s^{-1} fully diverted through the desiccant, the system was unable to the control chamber humidity, resulting in condensation on the chamber walls during the measurement. A smaller chamber would have been more appropriate for measuring photosynthesis of actively growing grass.

Figure 16-82. Using the custom chamber to measure turf. Note the condensation on the chamber top.

Custom Chamber - Closed

One of the items in `Config Menu|New` is a `Custom Chamber - Closed` configuration builder, for making closed system for flux measurements based on a time rate of change of CO_2. This would allow you, for example, to use a 6400-09 Soil Chamber in a closed mode, bypassing the normal drawdown cycles. This can be advantageous for measuring tiny fluxes. You could also use a normal leaf chamber, and make closed mode measurements on a leaf.

Setup

Do `Config Menu|New` and navigate to the `Custom Chamber - Closed` entry, and press **enter**.
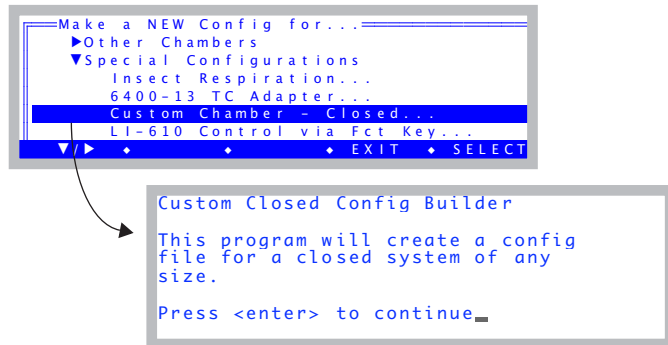
Figure 16-83. Custom Chamber - Closed *is under the* Special Configurations *node.*

1   **"Which chamber will you use?"**
The purpose of this first question is to get the default volume and area. Note that you can change volume and area after you are done - this just gets the initial default values.
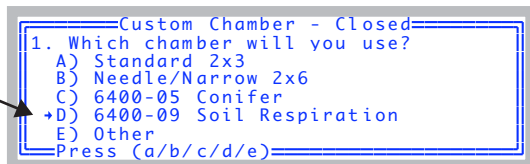
Figure 16-84. The first question gets at volume and area.

2 **"Which IRGA measures the chamber?"**

Closed mode measures gases with just one IRGA. Which should it be? The answer here is likely **S**ample, but if your chamber is somehow connected to the reference side, press **R**.

You can go back to the previous question by pressing the ↑ key

```
========Custom Chamber - Closed=======
2. Which IRGA measures the chamber?
 �→ S) Sample
    R) Reference


 ═══Press (s/r) or ↑ for prev═════════
```
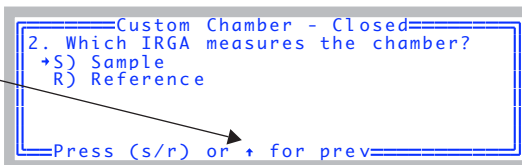
Figure 16-85. Choosing an IRGA

3 **"Chamber air temp measured by...?"**

A closed system requires measuring the air temperature in the chamber. How should it be measured?

```
========Custom Chamber - Closed=======
3. Chamber air temp measured by...
 �→ A) T/C attached to Tleaf connector
    B) T/C attached to aux. connector
    C) Temp/RH probe via aux. connector
    D) Other via aux. connector
 ══Press (a/b/c/d) or ↑ for prev═════
```
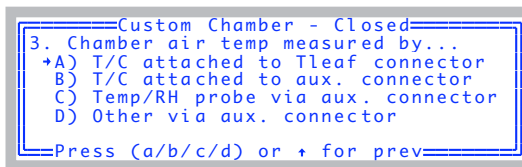
Figure 16-86. Closed mode needs a chamber air temp. Where is it coming from?

Four possibilities are presented here:

A) Use a thermocouple plugged into the sensor head connector that is normally used for the leaf temperature measurement.

B) Use a thermocouple plugged into the 6400-13 external thermocouple adapter, that plugs into the console auxiliary port.

C) Use a temperature/humidity probe (such as the Vaisala Humitter 50Y) connected to the console auxiliary port. If you select this, you will be prompted for the channels and ground being used for Temperature and for RH (Figure 16-87).

```
        ═══════════Temp Channel═══════sed═══
  3. Cham║ 0 - 20 (Pin 36) ║d by...
  →A) T/║→1 - 21 (Pin 19) ║connector
   B) T/║ 2 - 22 (Pin 18) ║onnector
   C) Te║ 3 - 23 (Pin 37) ║ connector
   D) Ot║                 ║or
        ══════════(0/1/2/3)════
  ══Press (a/b/c/d) or ↑ for prev══
```

```
        ════════Temp Ground════
  3.║ 3 (Pin 31 - Chamber Gnd)  ║..
   →║ 4 (Pin 14 - Pressure Gnd) ║ctor
    ║→5 (Pin 32)                ║tor
    ║ 6 (Pin 15)                ║ector
    ║ 7 (Pin 17)
        ══════════(3/4/5/6/7)════
```

```
        ═══════════RH Channel════════sed════
  3. Cham║→0 - 20 (Pin 36) ║d by...
  →A) T/║ 1 - 21 (Pin 19) ║connector
   B) T/║ 2 - 22 (Pin 18) ║onnector
   C) Te║ 3 - 23 (Pin 37)_║ connector
   D) Ot║                 ║or
        ══════════(0/1/2/3)══
  ══Press (a/b/c/d) or ↑ for prev══
```

```
        ════════RH Ground════
  3.║ 3 (Pin 31 - Chamber Gnd)  ║..
   →║ 4 (Pin 14 - Pressure Gnd) ║ctor
    ║→5 (Pin 32)                ║tor
    ║ 6 (Pin 15)                ║ector
    ║ 7 (Pin 17)
        ══════════(3/4/5/6/7)════
```

Figure 16-87. Using a Vaisala Humitter 50Y for chamber temperature. See the wiring diagram in Figure 16-49 on page 16-43.

D) If something else is connected to the console auxiliary port, you will be prompted for the channel and ground for Temperature (Figure 16-87, but for temperature only).

For choices 'C' and 'D', you will also be prompted for the transform for converting Volts to degrees C (and Volts to RH, if you picked 'C'). Enter the right side of the equation in algebraic notation, using the symbol V for measured volts (Figure 16-88).

```
┌═══════Custom Chamber - Closed═══════┐
│═══════Enter the Transform for Temp═══│
│DegC= V * 0.1 - 40_                   │
│└══(Use 'V' for raw Volts)═══════════ │
│  C) Temp/RH probe via aux. connector │
│  D) Other via aux. connector         │
│                                      │
│DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar│
```

```
┌═══════Custom Chamber - Closed═══════┐
│═══════Enter the Transform for RH════│
│RH= V * 0.1_                          │
│└══(Use 'V' for raw Volts)═══════════ │
│  C) Temp/RH probe via aux. connector │
│  D) Other via aux. connector         │
│                                      │
│DelLn  ♦ClrEnd ♦DelChar♦CapLock♦AnyChar│
```

Figure 16-88. Entering the transform. Use upper case V for the raw signal.

4   **Soil Temperature?**
     If wish to measure Tsoil with the 6400-09TC soil temperature probe, pick
     Yes.

```
┌═══════Custom Chamber - Closed═══════┐
│4. Do you want Tsoil measured         │
│with the the 6400-09 Soil Temp probe  │
│  Y) Yes                              │
│ →N) No                               │
│_                                     │
│└══Press (y/n) or ↑ for prev═════════ │
```

Figure 16-89. Using the soil temperature probe.

5   **External fans?**
     The next question (Figure 16-90) comes about because it is possible to use the
     LED light source connector on the sensor head to supply power to a small
     fan(s) in the chamber (provided it uses < 400 mA).

```
┌═══════Custom Chamber - Closed═══════┐
│5. Do you want an ON/OFF FCT key      │
│control for external fan(s) via the   │
│LED light source connector?           │
│  Y) Yes                              │
│ →N) No_                              │
│└══Press (y/n) or ↑ for prev═════════ │
```

Figure 16-90. Powering external fans.

You would wire the mating connector (part number 310-04300) to use the 2

pins (Figure 16-91) that normally supply power to the LED light source fan. (*Do not use pins 3 and 4 that supply 100V power to the LEDs*).
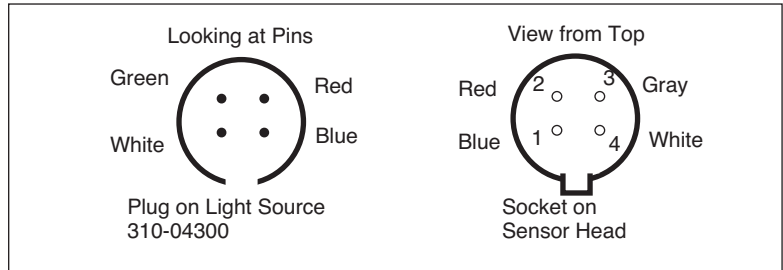
Figure 16-91. The pins that go into holes 1 and 2 are the ones to use for wiring a fan. Blue (1) is ground, Red (2) is power.

If you wish to power fans in this manner, respond **Y** to this prompt, and a function key control will be added that controls this fan power <u>without</u> putting any voltage on the other two pins. (This is much safer than using the normal light source control to simply run the fans.)

6   Using the Pump?
If you need to have the console pump operating to measure CO_2 and H_2O with the IRGA (sample or reference - you picked it back in Step 2), you will want to press **Y** to this question (Figure 16-92).
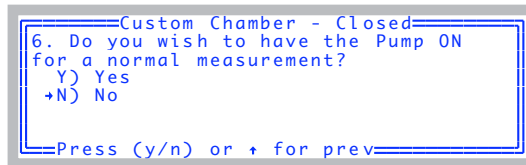
Figure 16-92. Default state of the pump.

If you respond No to this question, your configuration will have the following features:

a) Default flow control state will be Pump OFF.

b) "Pump OFF" and "Low Flow" warnings will be suppressed.

No matter how you respond to this prompt, you will have complete manual flow control (as normal) via **f2** level 2 in New Measurements.

7   **"Done. Ready to name and store?"**
At this point (Figure 16-93) you can quit, or go back to previous questions to check your answers. If you are ready to continue, press **Y**.

```
========Custom Chamber - Closed========
7. We're done. Are you ready to
name and store your configuration?
 →Y) Yes
  N) No

 ═══Press (y/n) or ↑ for prev═══════════
```
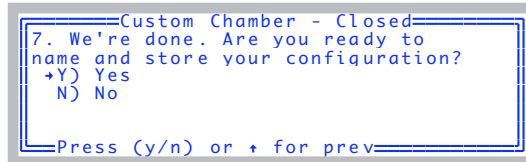
Figure 16-93. Read to finish? If you press **N***, you go back to question 1.*

8   **Configuration Name**
The last step is to name the configuration. This is the name you will see when you are asked to choose a configuration at power up or reset.

```
Dir: /User/Configs/UserPrefs
File:ClosedMode_6.1
Filter: ver 6 config files (*.xml)
══════════Save this config══════════
 Default+LED.xml
 Default+RGB.xml
 FactoryDefault_6.1.xml
+ Dir    ♦         ♦ Filter♦ CANCEL♦ SELECT
```
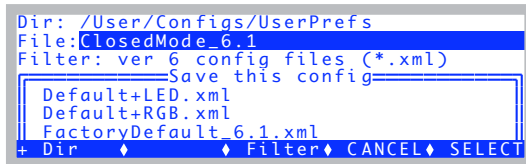
Figure 16-94. Selecting the configuration name.

If you select a name, you will then be asked if you'd like to implement the configuration now (Figure 16-95).

```
Implement now? (Y/N)
```
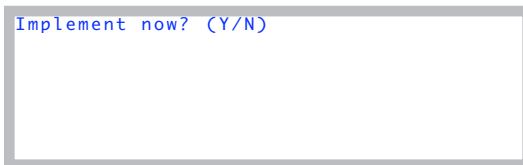
Figure 16-95. If you press **N***, you can implement it later using* Config Menu | Open.

Using The Closed Configuration

When a closed configuration is implemented, the main screen (Figure 16-96) indicates the version of the Closed System it is using, and the version of OPEN that it is running on.

```
      L I - 6 4 0 0 X T   C O 2   F l u x   S y s t e m

           C L O S E D   1 . 5   /   O P E N   6 . 2

/ U s e r   9 9 %   f u l l                    P S C - 1 0 9 4
T u e   A u g   2 3   2 0 1 1   1 5 : 0 3 : 3 9        1 2 . 4 6 V
   Home      Config    Calib     New      Utility
   Menu       Menu     Menu    Msmnts      Menu
```
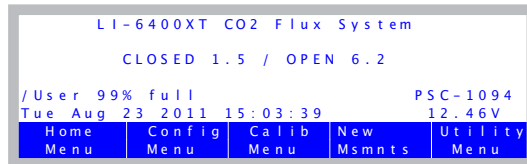
Figure 16-96. A configuration generated by Custom Chamber - Closed.

In New Measurements mode, the function keys on levels 1 through 6 are all the same with normal LI-6400 configurations, with the exception the **Area** and **Stom Ratio** keys (**f1** and **f2** in level 3) are missing. If you requested it, the **Area** key will have been replaced with a chamber fan control key (Figure 16-97).

```
→CO2R_µml  CO2S_µml  H2OR_mml  H2OS_mml
a   475.2     459.3    10.039    12.892
    xdC/dt     xCdry   RHcham   Flow_µml
b  0.0655       465      39.9       0.2
    CFlux       ETime  Program Status
c   0.311        0.0   0:Stopped
  Ch Fan               Sys&Usr  Prompts   Prompt
3   ON                  Consts    off       All
```
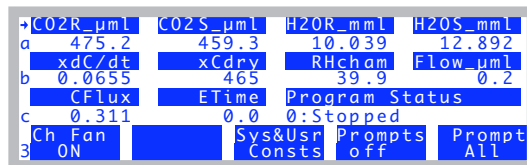
*Figure 16-97. **F1** on level 3 is a fan control. The key labels shows the state of the chamber fan (assumed attached to the light source control connector.)*

The main control for the Flux System is on function key level 7 (Figure 16-98).

```
→CO2R_µml  CO2S_µml  H2OR_mml  H2OS_mml
a   472.1     456.5    10.134    12.915
    xdC/dt     xCdry   RHcham   Flow_µml
b  -0.205       463      39.5       0.3
    CFlux       ETime  Program Status
c  -0.976        0.0   0:Stopped
  Target=   Edit      START
7 400      Params
```
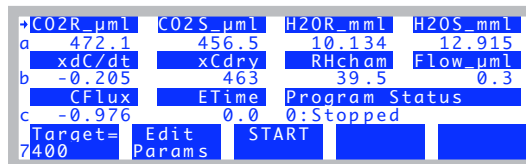
Figure 16-98. Controls for Closed Mode are on function key level 7.

The **Target=** key (**f1**) prompts for the target CO_2 concentration (if AutoTarget = no), and number of repetitions. If AutoTarget= yes, the target value will be in parentheses in the key label.

The **Edit Params** key (**f2**) allows you to view and change any of the closed system parameters (Figure 16-99).

```
════ClosedMode_6.1.xml════
→ ├ AutoTarget= no
  ├ CO2Target= 400
  ├ MeasTime= 20
  ├ Repetitions= 0
  ├ LogInt= 5
  ├ Offset= 0
  ▼/▶    ◆Edit    ◆    ◆    ◆Done
```
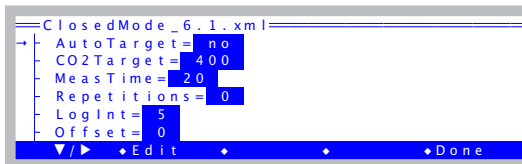
Figure 16-99. Editing the Closed Parameters.

Note that these parameters are also part of the system configuration with Closed System. `Config Menu|View/edit` in fact will look like Figure 16-100.

```
════ClosedMode_6.1.xml════
  ├▶open
→ └▼Closed
    ├ AutoTarget= no
    ├ CO2Target= 400
    ├ MeasTime= 20
    ├ Repetitions= 0
  ▼/▶    ◆    ◆    ◆    ◆Done
```
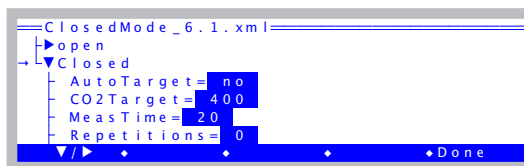
Figure 16-100. Closed parameters are part of the system configuration.

The parameters are listed below:

▼ **Closed**

AutoTarget=no	*When ON, CO2Target is set to CO2 value at start of measurement.*
CO2Target= 400	*Final results extrapolated to what value (ppm).*
MeasTime= 20	*How long the measurement lasts (secs).*
Repetitions= 0	*If > 0, the measurement cycle will be repeated that many times.*
LogInt= 5	*Data logging interval (secs).*
Offset= 0	*h in Eqn (16-9).*
SysVol= 991	*V_{sys} in Eqn (16-9).*
AreaCham= 81	*A_{cham} in Eqn (16-9).*

Offset is collar height on which the chamber sits, and system volume is total volume without collar. The relation between total volume and these three parameters is shown below

$$V_{tot} = V_{sys} + A_{cham}h \qquad (16\text{-}9)$$

where V_{tot} is total volume (cm^3), V_{sys} is system volume (cm^3), A_{cham} is exposed chamber area (cm^2), and h is the height (cm) of the chamber base above the ground (the offset).

The **START** key (**f3**) initiates a measurement. If no log file is open, you are prompted to name one. If there is a file open already, you are asked if you'd like to append the data onto that file. Press **Y** to do that.

During the measurement, the Program Status on display line *c* indicates the time remaining. If you wish to quit early, press **f3** (**STOP**)

Before the
measurement.

```
→CO2R_µml    CO2S_µml    H2OR_mml    H2OS_mml
a   562.0       592.5      10.042      12.934
     xdC/dt      xCdry      RHcham    Flow_µml
b  -0.653          604       35.4          1.0
      CFlux       ETime    Program Status
c   -3.11          0.0    0:Stopped
   Target=    Edit       START
7 600          Params
```

During the
measurement.

```
→CO2R_µml    CO2S_µml    H2OR_mml    H2OS_mml
a   562.3       581.3      10.052      12.868
     xdC/dt      xCdry      RHcham    Flow_µml
b  -0.732          591       35.2          1.0
      CFlux       ETime    Program Status
c   -3.48         13.5    1:Measuring (6)
   Target=    Edit       STOP
7 600          Params
```

Figure 16-101. Program Status and ETime (elapsed time) indicate the state of a measurement.

Once the measurement is done, the display will appear as in Figure 16-102.

After the mea-
surement.

```
→CO2R_µml    CO2S_µml    H2OR_mml    H2OS_mml
a   565.7       576.6      10.101      12.637
     xdC/dt      xCdry      RHcham    Flow_µml
b   0.509          600       34.6          0.9
      CFlux       ETime    Program Status
c    2.42          0.0    0:Stopped
   Target=    Edit       START    Resume
7 600          Params                Updates
```
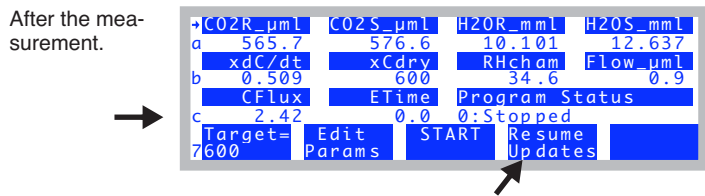
*Figure 16-102. Some variables will be frozen following a measurement. Press **F4** to resume updates for them.*

Some of the displayed variables (Table 16-10 on page 16-83) will be "frozen" following a measurement, and will remain so until you start another measurement, or press **Resume Updates** (**f4**). The rest of the Closed Flux Systems variables are described in Table 16-9 on page 16-82.

The Measurement Description

This closed flux system computes CO_2 flux by measuring the rate of change of dilution-corrected CO_2 with time. Unfortunately, this rate of change continually decreases, as the gradient driving the flux diminishes with time. We attempt to account for this by computing short-term (10 second) averages and rates of change during the measurement, then when the measurement is over, computing a flux appropriate for a user-specified target CO_2 concentration.

Instantaneous Values

Every 0.5 second, new readings are available, and are shown on the display. The average time associated with each reading is user-selectable, and is typically 2 seconds. In addition to the usual LI-6400 measured quantities, Closed System adds the following variables, which can be viewed and logged:

Table 16-9. Real time measurement variables defined by Closed.LPL

Label	Symbol	User ID	Description
Dilution	D	416	$D = 1 - \dfrac{W}{1000}$, where W is the sample cell H_2O
Cdry	C_{dry}	412	$C_{dry} = \dfrac{C}{D}$, where C is the sample cell CO_2
TCham	T_c	411	Chamber temperature. Defaults to the channel normally used for leaf temp (*Tleaf*).
Density	ρ	413	Density of the air: $\rho = \dfrac{1000P}{8.314(T_c + 273.15)}$
RHcham	RH_c	415	Chamber humidity in %

Special Averaging

There is another group of variables that reflect separate statistics kept by the Closed System. They are:

Table 16-10. Special Averaging variables defined by the Closed System.

Label	Symbol	User ID	Description
Mode		430	Useful for interpreting logged data: 1 - During measurement (averaging) 2 - End of measurement (regression) Mode has two other possible values: 0 - Not during measurement, special averaging active -1 - Not during measurement, special averaging not active ("frozen" display)
Smpls		431	Mode 1: Number of samples used in the average. This value will be < 0 while ETime < 10. This serves as a flag when recomputing the data file not to include this record in the vectors used for the final regression. Mode 2: Number of samples used in the regression. This is the number of Mode 1 records that have Smpls > 0.
xDilu	$\bar{D}$	404	Mode 1: Averaged D Mode 2: Regressed to ETime 0 based on first 10 secs of data
xDens	$\bar{\rho}$	403	Mode 1: Averaged ρ Mode 2: Regressed to ETime 0 based on first 10 secs of data
xCdry	$\bar{C}_{dry}$	406	Mode 1: Averaged C_{dry} Mode 2: C_{target}
xdC/dt	$\overline{\dfrac{dC_{dry}}{dt}}$	405	Mode 1: Averaged $\dfrac{dC_{dry}}{dt}$ Mode 2: Regressed to C_{target}
CFlux	F_c	401	CO_2 flux computed from the above variables: $F_c = \dfrac{\bar{\rho}\bar{D}V_{tot}}{100A_{cham}}\overline{\dfrac{dC_{dry}}{dt}}$

During a measurement (*Mode* = 1), the Table 16-10 variables come from 10 second running statistics. Each time a reading is logged (every *LogInt*), the current value of $\bar{D}$, $\bar{C}_{dry}$, $\overline{\dfrac{dC_{dry}}{dt}}$, and $\bar{\rho}$ are added to an internal vector. At

the end of a measurement (*Mode* = 2), the "x_" variables are computed from these vectors by a straight line fit against either elapsed time or *xCdry*, and computing the values at either *time*=0 or *Ctarget*.

For example, Figure 16-103 shows a sample data set using the LI-6400 Soil Chamber. The measurement time was 90 seconds, and data was logged every 10 seconds.

```
"OPEN 6.1ac"
"Fri Oct 31 2008 13:25:18"
<open><version>"6.1ac"</version></open>
<open><configfile>"/User/Configs/UserPrefs/ClosedMode_6.1.xml"</configfile></open>…
<open><comps><file>"/User/Configs/Comps/ClosedMode_6.1.LPL"<header>""</header>…
<open><prompts><onlog>off</onlog><items>"ClosedMode_6.1"<items[1]><id>-104 </id>…
<open><stability><items>"Std Stability"<items[1]><id>-2 </id><size>15
</size><pcv><onoff>0 </onoff><value>1 </value></pcv><std><onoff>0 </onoff><value>…
<open><log><format>"ClosedMode_6.1"<items>{ -35  -21  414  431  430  401  403  404…
<open><matching><type>0 </type><disp>'a'</disp><settings><CO2Limit>10.0</CO2Limit>…
<li6400><factory><unit>"PSC-853"</unit><serviced>"17 Feb 2003"</serviced><fuseawa…
<li6400><user><flow_zero>1058.2</flow_zero><irga_zero><co2>-190.3 -970.7<at>…
"Const= "420 "Vtot "991
"Const= "421 "Vsys "991
"Const= "422 "Offset "0
"Const= "423 "Area "81
"13:25:20 "
$STARTOFDATA$
"Obs" "HHMMSS" "ETime" "Smpls""Mode""CFlux""xDens""xDilu""xdC/dt""xCdry""CTarget"… "
1 "13:25:25 "10.0 -12 1 8.36 39.622 0.9792 1.762 578.48 610 25.66 583.79 39.614…
2 "13:25:35 "20.0 20 1 8.23 39.615 0.9772 1.739 591.23 570 25.64 601.31 39.617 …
3 "13:25:45 "30.0 20 1 8.42 39.612 0.9751 1.782 610.62 570 25.72 619.24 39.607 …
4 "13:25:55 "40.0 20 1 8.13 39.607 0.9739 1.723 628.54 570 25.71 636.72 39.607 …
5 "13:26:05 "50.0 20 1 7.43 39.605 0.9731 1.576 645.15 570 25.72 652.73 39.606 …
6 "13:26:15 "60.0 20 1 7.36 39.603 0.9726 1.561 661.28 570 25.75 668.68 39.601 …
7 "13:26:25 "70.0 20 1 7.41 39.606 0.9722 1.572 676.99 570 25.73 684.48 39.604 …
8 "13:26:35 "80.0 20 1 7.25 39.601 0.9718 1.54 692.56 570 25.76 699.88 39.597 …
9 "13:26:45 "90.0 20 1 7.12 39.600 0.9716 1.512 707.96 570 25.75 715.21 39.601 …
10 "13:26:46 "0.0 8 2 8.64 39.623 0.9799 1.818 570 570 25.76 716.84 39.600 26.66 6…
```

Figure 16-103. Sample data file

Figure 16-104. Excel version of the sample data in Figure 16-103.

Note there are 9 *Mode 1* records, followed by 1 *Mode 2* record. Each *Mode 1* record shows *Smpls* of 20 except the first, which is -14. (This means 14 samples, and the negative sign means the initial 10 second collection period hadn't finished). The *CFlux* column shows the running average values (*Mode 1*), and the final regressed value (*Mode 2*). In all cases, *CFlux* is computed from the four columns to its right (*xDens*, *xDilu*, *xdC/dt*), and *Vtot* and *Area*, further over to the right. Our target was 570 ppm (*Ctarget*), and the final value of *xCdry* reflects this.

xdC/dt vs xC_{dry}
Extrapolated to the target value.
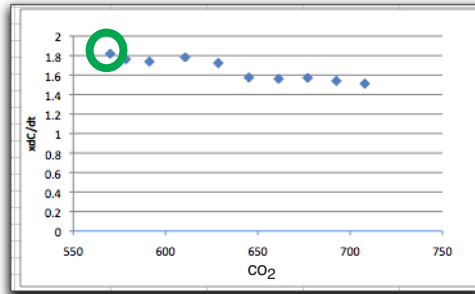
Density vs. Time
Extrapolated to $t = 0$
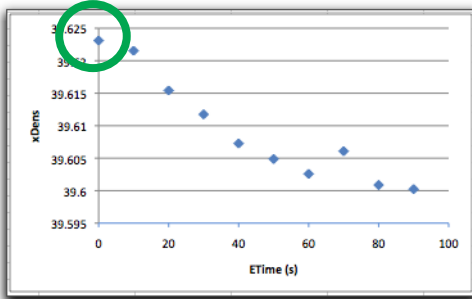
Dilution vs. Time
Extrapolated to $t = 0_t$
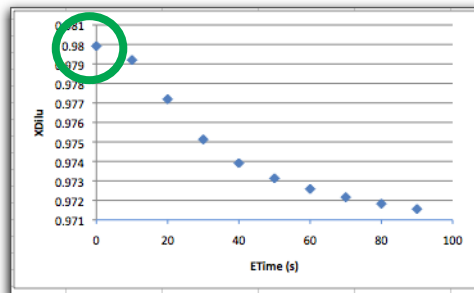
Figure 16-105. Plot of $\overline{\dfrac{dC}{dt}dry}$, $\bar{\rho}$, and $\overline{D}$ from Figure 16-10. The circled points are the Mode 2 values.

The plot of CFlux is shown in Figure 16-106.
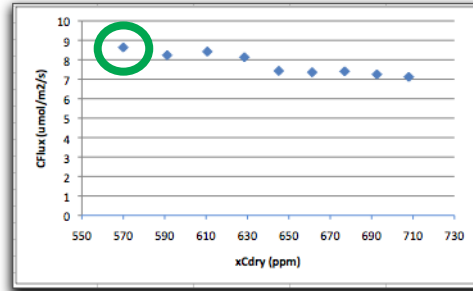
Figure 16-106. Plot of CFlux. The circled point is the Mode 2 value.

Theory

Starting from the LI-6400 soil flux equation (Equation (28-6) on page 28-46)

$$f_c = \frac{\rho v}{s}\left(\frac{dc}{dt} + \frac{c}{1-w}\frac{dw}{dt}\right) \tag{16-10}$$

we see that CO_2 flux (mol m^{-2} s^{-1}) is related to the rate of change of CO_2 ($\frac{dc}{dt}$

in mol CO_2 mol^{-1} s^{-1}) and also to the rate of change of water vapor ($\frac{dw}{dt}$ in

mol H_2O mol^{-1} s^{-1}). The problem with this formulation is w is a much less linear function of time than c. Over short, discrete intervals of 10 seconds (used by the LI-6400 in its soil configuration), this isn't an issue. Over 1 or 2 minutes, however, in a chamber that is not actively controlling the conditions, this can be a problem.

A better approach is to avoid the issue entirely and do the dilution correction on the fly, rather than after the fact. If we let

$$c_{dry} = \frac{c}{1-w} \tag{16-11}$$

and note that

$$\frac{dc}{dt}dry = \frac{1}{1-w}\frac{dc}{dt} + \frac{c}{(1-w)^2}\frac{dw}{dt}$$

$$= \frac{1}{1-w}\left(\frac{dc}{dt} + \frac{c}{(1-w)}\frac{dw}{dt}\right) \qquad (16\text{-}12)$$

$$(1-w)\frac{dc}{dt}dry = \frac{dc}{dt} + \frac{c}{(1-w)}\frac{dw}{dt}$$

We then substitute this result into Equation (16-10) and arrive at

$$f_c = \frac{\rho v(1-w)}{s}\frac{dc}{dt}dry \qquad (16\text{-}13)$$

This is the form of the equation used by the Custom Chamber - Closed configuration. The final equation, with flux F_c in μmol m^{-2} s^{-1}, volume V in cm^3, area S in cm^2, W in mmol H$_2$O mol^{-1}, and C in μmol CO$_2$ mol^{-1} is

$$F_c = \frac{\rho V\left(1 - \frac{W}{1000}\right)}{100S}\frac{dC}{dt}dry \qquad (16\text{-}14)$$

where

$$\rho = \frac{1000P}{R(T_{cham} + 273.15)} \qquad (16\text{-}15)$$

T_{cham} is chamber temperature (C), P is pressure (kPa), R is the universal gas constant (8.314 Nm mol^{-1} K^{-1}), and C_{dry} is

$$C_{dry} = \frac{C}{1 - \frac{W}{1000}} \qquad (16\text{-}16)$$

Computational change in Closed 1.5

The target value has always assumed to be wet (that is, not corrected for dilution). Prior to 1.5, this was the value used with the regression. Starting in

1.5, however, the regression target is dilution corrected. That is, the final flux is computed for $\dfrac{C_{t\arg et}}{1 - \dfrac{W}{1000}}$.

Using an Energy Balance

17

Computing what you can't measure

17 Using an Energy Balance

Sometimes it is not convenient or even possible to measure leaf temperature with a thermocouple. LI-6400 chambers that do not measure leaf temperature directly include the 6400-05 Conifer, the 6400-07 Needle, the 6400-08 Clear Bottomed, and the 6400-17 Whole Plant Chamber. The method of getting leaf temperature in these cases is to use an energy balance. This chapter explains the theory and the practice of using energy balance.

The Theory

The energy balance of a leaf in the LI-6400 cuvette has three major components: net radiation R (W m^{-2}), sensible heat flux Q (W m^{-2}), and latent heat flux L (W m^{-2}).

$$0 = Q + L + R \qquad (17\text{-}1)$$

We consider two components to net radiation, short wave (visible + near IR) and thermal ($R = R_{solar} + R_{thermal}$). The absorbed short wave part we will estimate by multiplying the incoming PAR reading in the chamber R_Q (μmol m^{-2} s^{-1}) by an empirical[1] conversion factor k, and a leaf absorptance α averaged over the spectrum of the light source.

$$R_{sw} = \alpha k R_Q \qquad (17\text{-}2)$$

The net thermal term is the difference in black body radiation balance, determined by the chamber wall temperature T_w and the leaf temperature T_l.

$$R_{thermal} = 2\varepsilon\sigma(T_w + 273)^4 - 2\varepsilon\sigma(T_l + 273)^4 \qquad (17\text{-}3)$$

where ε is the thermal emissivity of the leaf, and σ is the Stefan-Boltzmann constant. The 2's in (17-3) account for both sides of the leaf. Latent heat flux L is the transpiration rate E (mol m^{-2} s^{-1}) converted to W m^{-2}:

---

[1] Measurable by a spectroradiometer.

$$L = 44100E \qquad (17\text{-}4)$$

The sensible heat flux is a function of the leaf - air temperature difference $T_l - T_a$, the specific heat capacity of the air c_P (J mol^{-1} K^{-1}), and the one-sided boundary layer conductance for heat transfer of the leaf g_{bh} (mol m^{-2} s^{-1})

$$Q = 2c_p g_{bh}(T_l - T_a) \qquad (17\text{-}5)$$

g_{bh} is related to the boundary layer conductance for water vapor g_b by

$$g_{bh} = 0.92 g_b \qquad (17\text{-}6)$$

The energy balance equation now becomes

$$\alpha k R_Q + 2\varepsilon\sigma(T_w + 273)^4 - 2\varepsilon\sigma(T_l + 273)^4 = \qquad (17\text{-}7)$$
$$44100E + 2c_p g_{bh}(T_l - T_a)$$

If we let $\Delta T \equiv T_l - T_a$, and note that for small ΔT

$$(T_l + 273)^4 \approx (T_a + 273)^4 + 4(T_a + 273)^3 \Delta T \qquad (17\text{-}8)$$

then we can solve (17-7) for ΔT

$$\Delta T = \frac{\alpha k R_Q + 2\varepsilon\sigma\left[(T_w + 273)^4 - (T_a + 273)^4\right] - 44100E}{2c_p g_{bh} + 8\varepsilon\sigma(T_a + 273)^3} \qquad (17\text{-}9)$$

where $\varepsilon \approx 0.95$ for most leaves, $\sigma = 5.67 \times 10^{-8}$ W m^{-2} K^{-1}, and $c_P = 28$ J mol^{-1} K^{-1}. While α and k can vary widely for different light sources (Figure 17-2 on page 17-5), their product is fairly conservative (Table 17-1).

Using an Energy Balance
The Theory

Table 17-1. Radiation conversion factors (k) and absorption coefficients (α) for various light sources.[a]

Light Source	k[b]	α	$k \times \alpha \equiv B$
Sun + Sky	0.39	0.49	0.19
Mercury	0.33	0.60	0.20
Fluorescent	0.23	0.75	0.18
Tungsten	0.75	0.26	0.20
6400-02 LED (Red only)	0.18	0.84	0.15
6400-02B LED (Red + Blue)	0.19	0.84	0.16
Metal Halide	0.28	0.61	0.17

a.Measured with an LI-1800 spectroradiometer, 400-1100nm. Measurements of α were made on a green ash leaf using an 1800-12 Integrating Sphere.

b.Units of conversion factor k are $\dfrac{W/m^2}{\mu mol/m^2/s} = \dfrac{J}{\mu mol}$

.
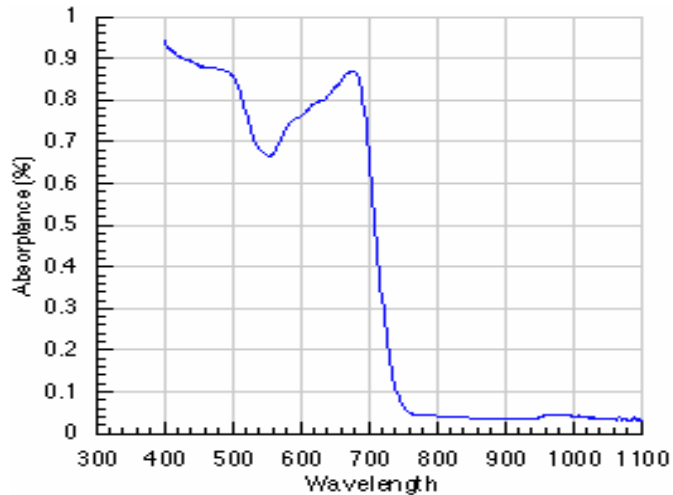
Figure 17-1. Leaf absorptance spectrum used in the computations of Table 17-1.
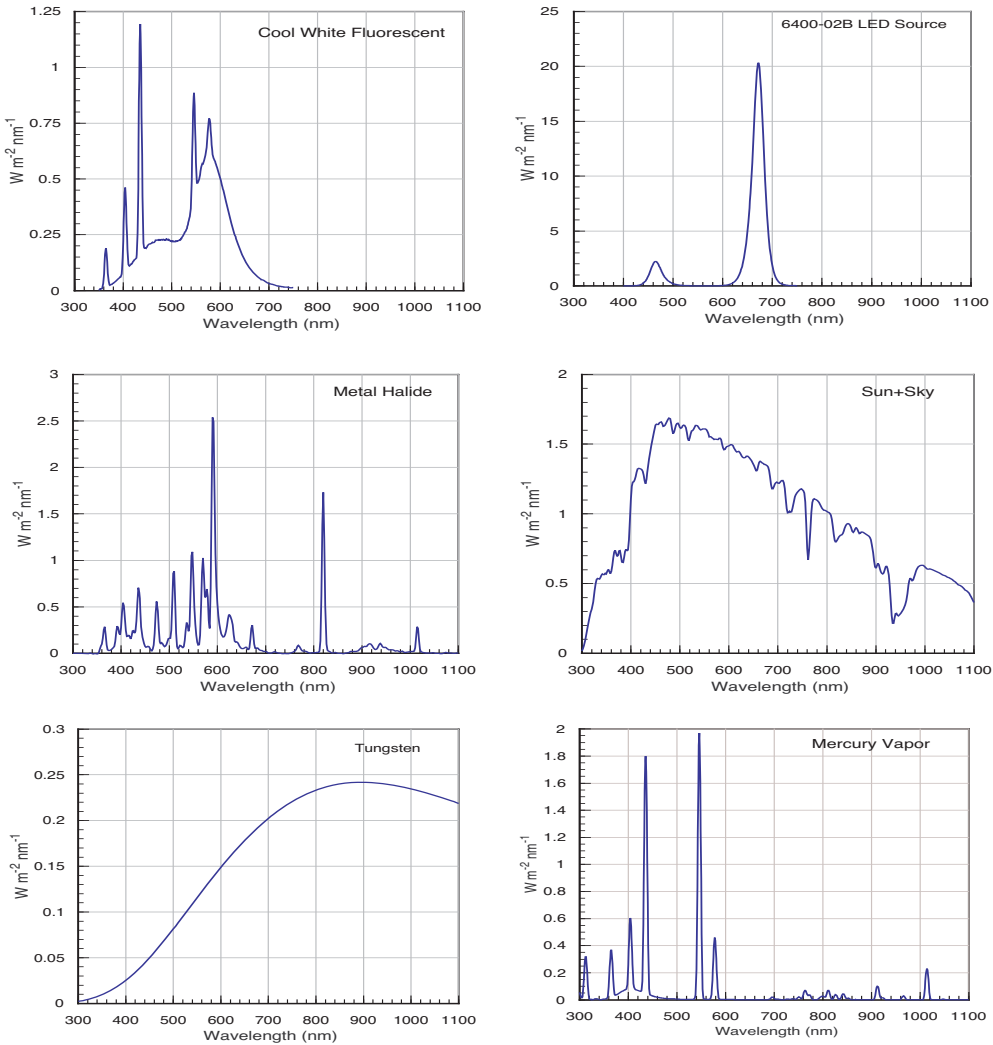
Figure 17-2. Spectra used to compute results in Table 17-1 on page 17-4. Vertical scales have units of
$W\,m^{-2}\,nm^{-1}$ *in all cases.*

Using Energy Balance in OPEN

How To Do It

There are two implementation steps:

1   **Software**

The system constant *Ebal?* (ID: -76) controls whether or not an energy balance computation is done. *Ebal?* lives in the configuration tree at

```
<open> <comps> <energybal> ebal
```

It can also be put in a prompt list, or edited during New Measurements mode by pressing **f3** level 3 (**Sys&Usr Consts**).

```
C ═══════User & System Contants═══════ mL
a     -33:Area = 6                       11
      -34:StmRat = 1                      %
b     -55:BLC1_mol = 1.42                81
      -76:EBal? = 0                      ol
c     -77:f_parin = 1.00                094
      -78:f_parout = 0.00
 Edit      ♦         ♦         ♦    ♦OK
```
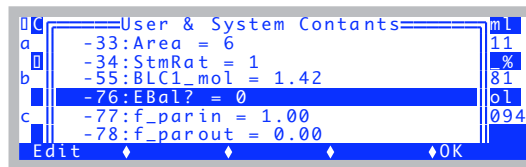
Figure 17-3. Changing EBal? on the fly in New Measurements mode.

2   **Hardware**

Make sure the leaf temperature thermocouple is measuring air temperature. For the chambers designed to be doing energy balance anyway (clear bottomed, conifer, etc.), that's the norm. For the 2x3 and 2x6 chamber bottoms that have a leaf temperature thermocouple in them designed to be touching a leaf, you'll want to pull it down a bit so it is measuring air, not plant material.

The Details

Energy balance computations are implemented via the ComputeList (Chapter 15). A part of the standard compute list is shown in Figure 17-4.

```
/* energy balance deltaT
*/

##2213F1 "Tair_K" "air temp in K"
"tLeaf_c + 273.15"

##2214F1 "Twall_K" "Twall temp K"
" tCham_c + 273.15"

##2216 "R(W/m2)" "incoming radiation"
" (parIn_um * f_parIn + parOut_um * f_parOut) * alphaK "

##2218 "Tl-Ta" "energy balance delta t"
" (#2216 + 1.0773E-7 * ((#2214 ^ 4) - (#2213 ^ 4)) - #20 * 44100.0)/(#111
* 51.4 + 4.3092E-7 * (#2213 ^ 3)) "

/* leaf temp
*/

##221F2 "CTleaf" "Computed leaf temp"
" Tleaf_c + #2218 * doEB"
```

_Figure 17-4. Partial listing of "StdComps_6.2", showing the energy balance section._

ID:2213, ID:2214: Absolute Air and Wall Temperature

We need to have a wall temperature and a chamber air temperature in degrees K. Variables 2213 and 2214 take care of that.

$$T_a\{K\} = T_l + 273.15$$
$$T_w\{K\} = T_c + 273.15$$

$(17\text{-}10)$

```
Twall_K = tLeaf_c + 273.15
Tair_K = tCham_c + 273.15
```

Note that chamber air temperature T_a comes from the leaf temperature thermocouple measurement T_l _(Tleaf_C, ID: -10)_, and chamber wall temperature T_w comes from the chamber (IRGA) temperature T_c _(Tcham_C_, ID:-9).

The LI-6400 doesn't have a wall temperature sensor per se, but the thermistor that measures T_c is located in the IRGA beneath the air circulation fan. Between this location and the leaf chamber, the air comes in contact with a lot of metal wall material, so this sensor's signal is not a bad first guess for wall temperature. However, the metal and Propafilm pieces of the leaf chamber itself will be modified by external air, but we have no way of measuring that with the default instrument. Thus, we use *tCham_C* for wall temperature. *If the coolers are on, then the farther the set point is from ambient temperature, the worse this assumption becomes*. Fortunately, however, wall temperature is not that critical in the analysis.

As for air temperature next to the leaf, some LI-6400 chambers have this (e.g. 6400-05, -07, and -08), and the rest don't. Either way, the variable *tLeaf_C* is the thing to use. If this is normally supposed to measure leaf temperature, such as in the standard 2x3 chamber, you'll have to make it into an air sensor by pulling it down a bit so that the junction is not in contact with any leaf material. Also, try to keep the junction shaded if possible to minimize radiation errors.

ID:2216: Absorbed Energy

Incoming radiation is computed in variable ID:2216 using the in-chamber light sensor (R_{Qin}) and the external LI-190 quantum sensor (R_{Qout}), each multiplied by an appropriate weighting factor.

$$
\begin{aligned}
R_{SW} &= \alpha k R_Q \\
&= \alpha k (f_{in} R_{Qin} + f_{out} R_{Qout})
\end{aligned}
\tag{17-11}
$$

```
R(W/m2) = (parIn_um * f_parIn + parOut_um * f_parOut) * alphaK
```

Absorption factor α and conversion factor k are combined into the system parameter *alphaK* (ID:-79). Typical values are given in Table 17-1 on page 17-4. *alpahK* is held in the configuration node

```
<open> <comps> <energybal> <alphaK> αk
```

The PAR readings R_{Qin} and R_{Qout} are system variables *parIn_um* (ID:-12) and *parOut_um* (ID:-13). The weighting factors f_{in} and f_{out} are system variables *f_parin* (ID:-77) and *f_parout* (ID:-78). These are found in the configuration nodes

```
<open> <comps> <energybal> <f_parin> fin
<open> <comps> <energybal> <f_parout> fout
```

Examples: If the standard 2x3 cm chamber with an opaque bottom and clear top is used,

```
f_parin = 1
f_parout = 0
```

If the chamber has a clear bottom, then the incident PAR would be larger than measured by the uplooking chamber light sensor. You could handle this situation by

```
f_parin = 1
f_parout = 0.1
```

which would add 10% of the external LI-190 quantum sensor's reading to account for this. You are assuming that the ground reflectance and lower chamber window transmittance combine to be 0.1.

2218: Delta-T

The actual calculation is performed in #2218, which computes the leaf-air temperature difference, using Equation (17-9) on page 17-3.

```
Tl–Ta = (#2216 + 1.0773E–7 * ((#2214 ^ 4) – (#2213 ^ 4)) –
         #20 * 44100.0)/(#111 * 56.0 + 4.3092E–7 * (#2213 ^ 3))
```

221: Leaf Temperature

The leaf temperature is computed in variable #221. If energy balance is being used, the computed leaf-air temperature difference is added to the air temperature (as measured by the leaf temperature thermocouple). Otherwise, the leaf temperature is simply the leaf temperature thermocouple reading.

```
CTleaf = Tleaf_c + #2218 * doEB
```

Measuring Boundary Layer

Measuring the boundary layer conductance for broadleaves in LI-6400 chambers can be done using saturated filter paper to simulate a leaf. The problem with this experiment, however, is that the leaf can be 10C cooler than the air, and the leaf temperature thermocouple will always underestimate that difference, because of conduction errors along the wires.

We can bring an energy balance analysis to bear on this problem, with the objective of finding an equation that will compute both leaf temperature, and boundary layer conductance. This requires the following simplifications:

- **There is no significant short wave energy incident on the filter paper**
  $R_Q = 0$ in (17-7). This requires doing the experiment indoors in room light.

- **The only resistance to water leaving the paper is boundary layer**
  This is accomplished by making sure the filter paper is very wet, and allows us to write

$$E = 2g_b\left(\frac{e(T_l) - e_s}{P}\right) \tag{17-12}$$

where the function $e(T)$ computes saturation vapor pressure (Equation (14-24) on page 14-13), T_l is the filter paper temperature, and e_s is the vapor pressure surrounding the paper (Equation (14-21) on page 14-12). Solving for one-sided boundary layer conductance g_b,

$$g_b = \frac{EP}{2(e(T_l) - e_s)} \tag{17-13}$$

and substituting that expression, along with $R_Q = 0$ into (17-7) leaves us with

$$0 + 2\varepsilon\sigma(T_w + 273)^4 - 2\varepsilon\sigma(T_l + 273)^4 = \tag{17-14}$$

$$44100E + 2c_p\left(\frac{EP}{2(e(T_l) - e_s)}\right)(T_l - T_a)$$

Solving (17-13) for E yields

$$E = \frac{2\varepsilon\sigma((T_w + 273)^4 - (T_l + 273)^4)}{44100 + \dfrac{c_p P(T_l - T_a)}{e(T_l) - e_s}} \tag{17-15}$$

Everything in Equation (17-15) is known except leaf (paper) temperature T_l. If T_l can be found by iteration, then boundary layer conductance can be determined from (17-13). An LPL program is provided with OPEN that implements this solution. It is /Sys/Utility/ENERGYBAL, and is described on page 21-13.

Further Reading

Nobel, P. 1990, *Physiochemical and Environmental Plant Physiology*, Academic Press, San Diego, London.

Ehleringer, J.R. 1989. Temperature and energy budgets. in *Plant Physiological Ecology: Field Methods and Instrumentation*. R.W. Pearcy, J. Ehleringer, H.A. Mooney, P.W. Rundel, eds. Chapman and Hall, London, New York.

Norman, J.M. and G.S. Campbell. 1998. *An introduction to environmental biophysics*, Springer, New York.

Calibration Issues

18

The difference between data and good data.

18

Calibration Issues

This section describes the calibration issues for the various sensors in the LI-6400. While some sensors require no such attention, others require periodic user or factory service (Table 18-1).

Table 18-1. Sensors and accessories, and their calibration requirements

Quantity	Sensor	Factory Calibration	User Calibration
H2OR, H2OS	H2O IRGA	2 Yrs	Check zero daily Check span monthly
CO2R, CO2S	CO2 IRGA		
Flow	Flow meter		Check zero daily
PARi (LED)	Si photodiode		Generate control voltage vs. output response as needed (see **Light Source Calibration** on page 18-31).
PARi (no LED)	GaAsP photodiode		Measure adjustment factor for light sources not already in the list. (See **Generating a Calibration Correction** on page 18-38).
PARo	Quantum sensor		None
Press	Pressure sensor		
Tleaf	Thermocouple	None	Periodic zero check
Tair	Linearized Thermistor		None
Tblock			
Tirga	Chip Thermistor		
6400-01 CO_2 Injector	None	None	Generate control voltage vs. output response as needed (see **6400-01 CO2 Mixer** on page 18-25).

Managing Calibration Data

Before discussing details about the various user-calibrations that need to be performed, it is important to cover some basic concepts of managing calibrations on the LI-6400.

The Calib Menu has three entries for managing calibration data

(Figure 18-1):`"View Current..."`, `"View History..."`, and `"View/Edit Accessory Cals.."`

```
╔═══Calib Menu══════════════════════
║ ▼View Settings...
║    View Current...
║    View History...
║    View / Edit Accessory Cals...
║  Flow meter zero...
║ ▶IRGA
║ ▼/▶    ♦        ♦       ♦ EXIT  ♦ SELECT
```

Figure 18-1. The View Settings node.

View Current

View Current shows the current state of the calibration tree (Figure 18-2).

```
╔═══Current Calibration════════════════
 └▼li6400
  ├▶factory
  └▶user



  ▼/▶    ♦        ♦       ♦ EXIT  ♦ SELECT
```

Figure 18-2. The Current Calibration screen shows the state of the factory and user settings.

The <factory> node contains factory settings, and the <user> node shows values that are set by the user in response to performing the various calibration routines described in this chapter. The whole tree is shown below.

▼ **li6400**		
▼ **factory**	*These settings are determined at the factory.*	
unit= "PSC-1094"		
serviced= "7 Feb 2007"		
fuseaware= 0		
co2mixer= Yes		
▼ **co2**	*Eqn (14-12) on page 14-8.*	
coeffs= 0 ...	*Six values, first one 0.*	
dvdt= -3	*S_c in Eqn (14-13) on page 14-9.*	
▼ **h2o**	*Eqn (14-7) on page 14-7.*	
coeffs= 0 ...	*Four values, first one 0.*	
dvdt= -2.3	*Sw in Eqn (14-8) on page 14-7.*	
flow= 0 0.3788	*0, a_f in Eqn (14-6) on page 14-7.*	
press= 88.692 0.00552	*a_{pi} in Eqn (14-1) on page 14-5.*	
▼ **user**	*These values change when you calibrate.*	
flow_zero= -2.45	*D/A value, channel 16.*	
▼ **irga_zero**		
co2= 78.1 -859.4	*D/A values, channels 12, 13.*	
at= 26.352	*T_{xc} in (14-13) on page 14-9.*	
h2o1= 17.2 78.1	*D/A values, channels 14, 15.*	

```
           at= 27.07                    T_xw in (14-8) on page 14-7.
 ▼ irga_span
     co2= { 0.994  0.983  }            G_cr and G_cs in (14-10), (14-11).
     h2o= { 1.01  1.001  }             G_wr and G_ws in (14-5), (14-6).
 ▼ irga_match
     co2= { 0.26388  -2.49324  }       C_ms and C_mr in (14-10), (14-11).
     h2o= { 0  0.0681999  }            W_ms and W_mr in (14-5), (14-6).
 ▼ co2_mixer
     pump_mv= 4600                     V_q0 in (14-17) on page 14-10.
     ppm= { 2265.32    42.19}
     mv= { 5000    100  }
parin_offset= 0.433045
 ▼ led_cal              Only visible when configured for 6400-02 Source.
     unit= "SI-1267"
     mv= { 10   5000
     qntm= { 5.04  3940.0}
 ▼ lcf_cal              Only visible when configured for LCF.
     unit= "LCF-0304"/unit
     ▼ red
         mv= { 50   1500  }
         qntm= { 6.07527   363.114  }
     ▼ blue
         mv= { 100   5000  }
         qntm= { 2.85  50.3077 }
```

For Power Users

The View Current screen has some hidden capability that comes in handy if you know what you are doing. To edit any node, press **ctrl** + **e**. To reset the flow meter and IRGA zeros to zero, and the IRGA spans to 1.0, press **ctrl** + **x**. To show the normally hidden <accessories> node (and allow direct edits), press **ctrl** + **a**. If there are changed nodes, **f3** (**Save**) and **f4** (**Revert**) will be active.

The <factory> values are stored in the file /dev/.factory, the <user> values in /dev/.user, and the <accessory> values in /dev/.accessories. Also, the "at factory" line of the calibration history (described below) is in the file /dev/.fuser. These files are normally set at the factory. If a unit was upgraded to version 6 in the field, these files are generated automatically the first time OPEN runs. There is a routine named *LoadCalXML* that runs when OPEN starts. If .factory, .user, or *.accessories* is missing from /*dev* (which will be the case for a field upgrade), it will generate them (and .fuser) from the old version calibration files, /dev/parm0 and /dev/parm1.

View History

Each time you perform a user calibration (zeroing the flow meter, zeroing the IRGAs, etc.) an entry will be added to the calibration history file (/dev/.history). You can view this file using the View History program (Figure 18-3).
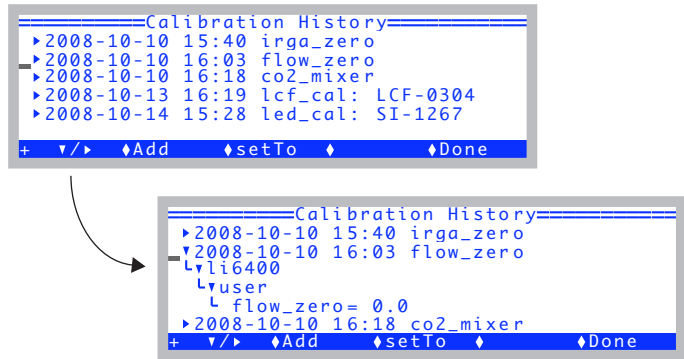
Figure 18-3. The calibration history viewer.

Each entry in the file is a node that expands to show what changed as a result of that calibration.

The top entry in the history file will be labelled ****At Factory****, and it will reflect the state of the user calibration when the unit was last at the factory. The history file does not grow forever - when it gets above a certain size[1], it will drop some of the earliest entries to make room. Since the ****At Factory**** entry is stored separately (/dev/.fuser), it will never be dropped, and will always appear first in the list, regardless of its date.

The function keys available in View History are shown in Table 18-2, and described in more detail following the table.

Table 18-2. Function keys in View History.

Key	Short-cut	Description
▼/▶	**enter**	Expand or collapse a node.
Add	**A**	Add the current \<user\> node as an entry. See below.

---

[1] 50,000 bytes

Table 18-2. (Continued) Function keys in View History.

Key	Short-cut	Description
setTo	T	Revert back to a selected calibration setting. See below.
Edit	E	Change the label of the cursor nodes main entry.
Remove	R	Remove the cursor nodes main entry.
Save	S	(Visible only if modified) Save the history file.

Add

The **Add** key (**f2** or just **A**) will add the entire <user> node to the history file in one "event". This is useful for capturing the current state of your machine (the entire <user> node) into a single history file record. You are prompted for a label.
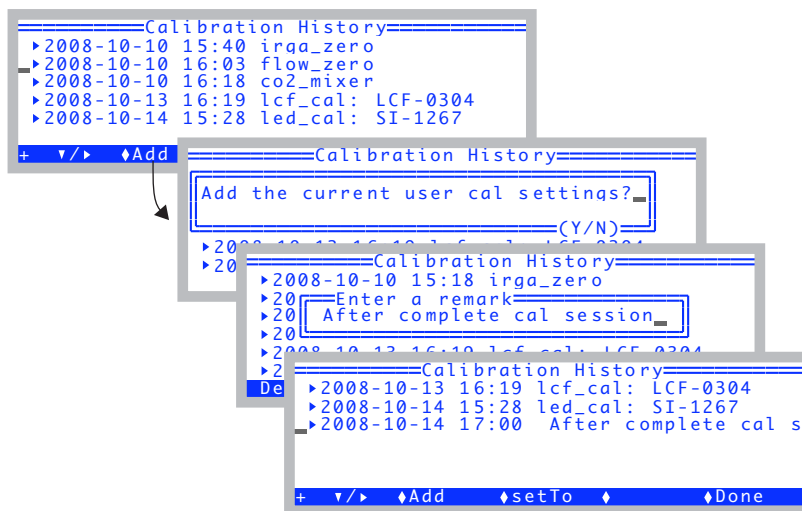
Figure 18-4. Manually adding to the history file. The node added will contain the entire user calibration node.

setTo

The **setTo** key (**f3** or just **T**) will let you revert back to whatever calibration settings the cursor is currently sitting on (and all subnodes from that). For example, if you want to revert the entire <user> node (and subnodes) back to the `**At Factory**` settings, put the cursor on the `**At Factory**` and press **setTo**. However, if you all you wanted was to revert the H_2O zeros, for exam-

ple, then open the `**At Factory**` node, and put the cursor on <li6400> <user> <irga_zero> <h2o> node, then press **setTo**, as shown in Figure 18-5.

1. Position the cursor on the desired node, and open it.

```
============Calibration History============
 ‣2008-08-26 10:30 ** At Factory **
 ‣2008-01-11 14:36 Menu Save
 ‣2008-04-16 14:34 flow_zero
 ‣2008-04-16 14:44 irga_zero
 ‣2008-04-17 14:29 irga_span
 ‣2008-04-17 16:46 co2_mixer
 +  ▾/▸   ♦Add    ♦setTo  ♦       ♦Done
```

2. Position on the desired sub-node.

```
============Calibration History============
 ├ flow_zero= -234.4
 ▾irga_zero
 │▾co2= 273.4 -468.8
 │ ∟ at= 26.352
 ∟▾h2o= 117.2 78.1
   ∟ at= 27.07
 +  ▾/▸   ♦Add    ♦setTo  ♦       ♦Done
```

3. Confirm this is what you wish to revert to.

```
======Change current cal to this?======
 ∟▾li6400
  ∟▾user
   ∟▾irga_zero
    ∟▾h2o= 117.2 78.1
      ∟ at= 27.07
    ▾/▸   ♦       ♦       ♦Cancel ♦OK
```

Figure 18-5. Reverting to a subset of the **At Factory** node.

Edit

The **Edit** key (**f1** level 2, or just **E**) will let you edit the label for the main table entry node that the cursor is currently sitting on (or in, if it is expanded). You cannot edit the label for the `**At Factory**` node at the top of the list, however.

Save

The **Save** key (**f3** level 2, or just **S**) is visible if you have made changes to the history file contents. The file is not actually updated until you save it, however.

Calibration Issues
Managing Calibration Data

View / Edit Accessories

Accessories are considered to be anything that may or may not be attached to the LI-6400 that has calibration information that OPEN needs to know about. The screen is shown in Figure 18-6.

```
============Accessory Calibrations============
┌─►tops
├─►led
└─►quantum


     ▼/►   ♦ Edit   ♦ Add   ♦Remove ♦ Done
```
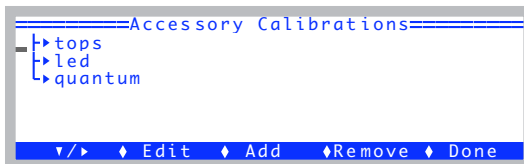
Figure 18-6. The View/Edit Accessories screen.

A typical accessories node fully expanded is shown below:

▼ li6400
    ▼ accessories
        ▼ **tops**
            ▼ GA-1094= 0.81
                date= 9 Jan 2001/date=
            ▼ GA-1052= 0.82
                date= 08/29/01/date=
        ▼ **led**
            ▼ SI-1267= -0.78
                date= 2005?/date=
        ▼ **quantum**=
            ▼ Q30292= 7.32
                date= 28 Nov 2001/date=

The three kinds of sub-node under accessories are shown in Table 18-3.

Table 18-3. Accessory nodes.

Node	Serial number	Description
tops	GA- GB-	Chamber tops with GaAsP light sensors.
led	SI-	6400-02, -02B Sources
quantum	Q	External quantum sensors

Adding an Accessory

To add an accessory, it doesn't matter where the cursor is, just press **Add** (or **f3**, or just **A**). You will be prompted for a serial number (Figure 18-7), the calibration value, and the date.

The three prompts, in sequence.

```
═══════════Add an Accessory═══════════
1. Select serial number (or escape to
quit)
  (Examples: ga-123, Q1032, si-415)
gb-899
2. Enter cal value 0.76
3. Enter cal date 22 Aug 2005
DelLn  ◆ClrEnd ◆DelChar◆CapLock◆AnyChar
```

The result.

```
═══════════Accessory Calibrations═══════════
┌▼* tops
│ ▶GA-1094= 0.81
│ ▶GA-1052= 0.82
│ ▶GA-222= 0.55
└▼* GB-899= 0.76
  └ * date= 22 Aug 2005
  ▼/▶  ◆ Edit  ◆ Add   ◆Remove ◆ Done
```

Figure 18-7. Adding an accessory.

The <u>serial number</u> should start out with GA-, GB-, SI-, or Q. Upper or lower case does not matter. The program will use the serial number prefix to put the accessory into the right node[2]. The <u>cal value</u> comes from the accessory's calibration sheet. If you cannot find the calibration sheet, contact LI-COR. The <u>cal date</u> also comes from the cal sheet, but this is not important. It is only there for your reference; whatever string you enter, if any, will be dutifully put in the date node.

Edit

Use the **Edit** key (**f2**, or just **E**) to change any value or date in the list. You cannot edit a serial number with **Edit**. If you need to do that, use **Add** and **Remove** as necessary.

Remove

The **Remove** key (**f4**, or just **R**) will remove any single accessory.

Saving

When you press **Done** (or **escape**, for that matter), the accessories list will be saved to the file /dev/.accessories.

---

[2]Any prefix besides GA-, GB-, SI-, or Q will be considered a chamber "top".

CO_2 and H_2O Analyzers

Factory Calibration

The factory calibration of the infrared gas analyzers (IRGAs) consists of determining the coefficients of the polynomial *f(x)* for water (Equations (14-5) and (14-6) on page 14-6), and *g(x)* for CO_2 (Equations (14-10) and (14-11) on page 14-8). The source for CO_2 concentrations is a series (usually 13) of standard tanks, ranging in concentration from 0 to about 3000 μmol mol^{-1}. Water concentrations are generated using a LI-COR LI-610 Dew Point Generator. IRGA calibration data is collected at a series of temperatures (typically 15, 30, and 45 C), with the entire instrument in a temperature controlled chamber. The data are then standardized (concentrations are scaled by temperature, voltages are scaled by pressure), and the calibration curves generated: a 5th order polynomial for CO_2, and a 3rd order polynomial for H_2O. The coefficients for these curves are provided on the calibration sheet, and are retained in file system (/dev/.factory).

User Calibration (Zero and Span)

User calibration actions consist of checking and/or adjusting the zero and span. A zero is done with dry, CO_2-free air in the IRGA cell, and the span is done with a known non-zero concentration in the cell.

How Often?

When OPEN first runs, it reads the most recent zero and span information from a file (/dev/.user), and applies it. Thus, user calibrations should carry over from one day to the next, once the instrument is warmed up. As for how often you should re-check them, your own experience should be your guide, but our experience is:

For zeroing, If conditions (temperature, mostly) haven't changed a great deal since the last time you zeroed the IRGAs, it won't need adjusting. But, it doesn't hurt to check it, as long as you know your chemicals are good, or have a tank that you know is CO_2-free. You will do more harm than good, however, if you dutifully re-zero every day using chemicals, but ignore the condition of those chemicals.

For spanning, if you've got a standard you trust, then you can adjust the IRGA to match that standard. It shouldn't need subsequent adjusting for months and months, however. If you don't have a good standard, then don't bother with the span.

Setting the CO_2 and H_2O Zero

The procedure for checking the IRGAs' zero is part of the daily warm-up tasks, described in **After Warm Up** on page 4-4. If you find that an adjustment is necessary, then do one of the following procedures: use the chemical tubes to obtain dry, CO_2 free air (described below), or use CO_2-free air from a tank (described starting on page 18-14).

Zeroing With Chemicals

While it is probably best to use tank air for zeroing, it can be done with chemicals. Being able to trust the chemicals to thoroughly scrub the air involves a certain amount of "procedures and practice", including knowing when and from where the chemicals were purchased, how and where they were stored, etc. Just because you changed the chemicals in the tubes recently does not necessarily mean that they are good.

1 **Select** `"IRGA Zero"` **in the calibration menu**

You'll be greeted with a couple of prompt screens, followed by the main screen (Figure 18-8):

```
          IRGA  Zeroing

This  routine  requires  the  leaf  chamber
to  be  closed  and  empty,  and  the
use  of  fresh  soda  lime  and  desiccant.

OK  to  Continue?  (Y/N)_
```

```
          Zeroing  The  IRGA

Chamber  must  be  closed.  To  zero...
..CO2  only:  SCRUB  CO2,  BYPASS  desiccant
..H2O  only:  SCRUB  desiccant

Press  Any  Key
```

Match Valve <u>on</u> or <u>off</u>.
Press **M** to toggle

Rates of change
(per minute)

Press **F** to change to fast, slow, and off.

Press **P** to toggle pump <u>on</u> and <u>off</u>. (Turn it off when zeroing from a tank.)

```
          Zeroing  The  IRGA
Mch:off          Fan:Fast          Pump:ON
 CO2R_µml    CO2S_µml    H2OR_mml    H2OS_mml
    3.7          3.8        0.002      -0.003
  Slp(CR)     Slp(CS)     Slp(HR)     Slp(HS)
 -2.7E-01    1.1E-01    4.8E-03    -3.7E-03

1 CO2RS→0   H2ORS→0    All→0     Plot     Quit
```

Zero CO_2 IRGAs only

Zero H_2O IRGAs only

Zero both CO_2 and H_2O IRGAs

Strip Chart Viewing

Press this to leave

See **Individually Zeroing the IRGAs** on page 18-16.

For manual zeroing

1	CO2RS→0	H2ORS→0	All→0	Plot	Quit	
2	CO2R→0		H2OR→0			(Match OFF)
2		CO2S→R		H2OS→R		(Match ON)
3	CO2R↑	CO2R↓		CO2S↑	CO2S↓	
4	H2OR↑	H2OR↓		H2OS↑	H2OS↓	

Figure 18-8. The IRGA zero screen. Other variables can be monitored by pressing ← or →, and zeroing of individual IRGAs is available by pressing **labels**, *or the fct key level number (***1** *through* **4***)*

2   Soda Lime: full scrub   Desiccant: full bypass

Figure 18-8 illustrates the IRGA zero screen. We'll do CO_2 first, because it's quickest. Put the desiccant on full bypass, because (if it's Drierite) it will buffer CO_2, resulting in a longer time to reach zero as it flushes out.

3   Wait for stability

Watch *CO2R_µml* and *CO2S_µml* (reference and sample cell CO_2 concentrations), and also their rates of change (*Slp(CR)* and *Slp(CS)*). and when they get as low as they will go, you are ready. (If you wish to see this graphically, press **Plot** (**f4**). A typical plot is shown in Figure 18-9). To return to the normal text display, press **escape** or **QUIT**). It should take less than 2 minutes to get reasonably stable readings.
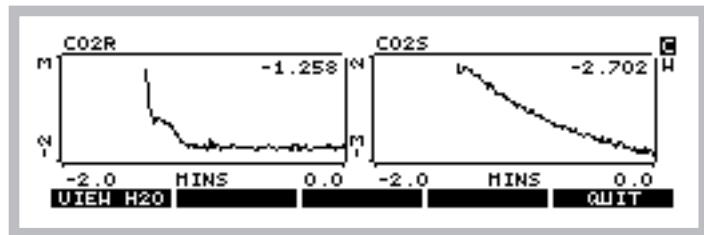
Figure 18-9. Pressing **plot** *(***f4***) brings up strip charts for H_2O and CO_2. Switch between the two by pressing* **C** *and* **H***, or* **f1**.

4   If re-zeroing is necessary, press CO2RS→0 (f1).

If both CO_2 values are within 5 μmol mol^{-1} of 0, you can skip this if you wish[3]. If you do the press **CO2RS→ 0**, it will take about 30 seconds to do its measurements and calculations, and establish new zeros.

When it's done, both the CO2R_μml and CO2S_μml values should be within 1 μmol mol^{-1} of 0.

5   Soda Lime: full scrub   Desiccant: full scrub

Now we do the water zero. Since water sticks to everything, it can take many minutes to come to a reasonably stable "dry" reading, since the chamber and IRGA surfaces are continually giving up water vapor to the suddenly drier air within the system.

Monitor *H2OR_mml* and *H2OS_mml*, and their rates of change *Slp(HR)* and

---

[3]·Note that the zero only affects absolute concentrations; the relative readings (differentials) are taken care of by matching, described on page 4-33.

Slp(HS). (If you want to do this graphically, press **plot** (**f4**) then **H**.)

This process can be speeded a bit by removing the upper half of the chamber from the system. This is done by clamping onto a sheet of something gas impermeable (like Propafilm or Saran) that covers the entire chamber *AND* the three rear holes over the IRGA.

After a couple of minutes, if it's clear that both IRGAs are heading for something close to zero, then call it good (the same rationale as in the footnote to step 4 applies here as well) and jump ahead to step 7. Otherwise, wait it out for 15 minutes[4] or so, and…

6   **Press H2ORS→0 (f2) when it's stable**
It takes about 30 seconds to accomplish the zero, after which you are done.

7   **Press Quit (f5) to exit**

Zeroing With Compressed Gas
Instead of using the chemicals to zero, you can use air from a tank, such as compressed CO_2 free air, or dry nitrogen. Be careful with the latter - these tanks may have 10 or 20 μmol mol^{-1} of CO_2 in them, and would thus require a tube of soda lime in-line to remove it. If you have doubts, use a tube of soda lime to test it. The compressed air should be suitably dry, however.

1   **Use a modest flow rate**
If you have a way to measure flow from your tank, use about 500 ml min^{-1}. If you don't, then simply adjust the flow so that you can just feel it if you hold the tube next to your moistened lower lip. Flow rate is not very important for zeroing, as long as it's adequate to get the cells flushed out in a timely manner. (Safety tip: establish the flow rate before connecting to the sensor head.)

2   **Connect directly to the sensor head**
It is simplest to connect the output of the tank's regulator directly to the sensor head, and bypass the console. There are two ways to do this: either connect to the sample inlet, and use the match valve to get that air into the reference afterwards, or use a "Y" connection and flow air to both sample and reference IRGAs simultaneously. Both methods are illustrated in Figure 18-12 on page 18-19. The preferred method is to use a "Y" connector.

[4.]There's a way to do this faster. See **Individually Zeroing the IRGAs** on page 18-16.

3   **Use the IRGA zero program**

It's the entry `"IRGA Zero"` in the Calib Menu.

4   **Turn off the Pump**

Since you don't need the pump, you can turn it off by pressing **P**.

5   **Match ON (if necessary)**

If you are using a single line connected to the sample inlet, put the match valve in the proper position by pressing **M**, once the IRGA zero program is running. Note that now the reference IRGA will be seeing any leaks that occur in the sample cell, so make sure there are none (chamber closed).

To shorten the dry down time, block the upper half of the chamber, as explained under step 5 on page 18-13.
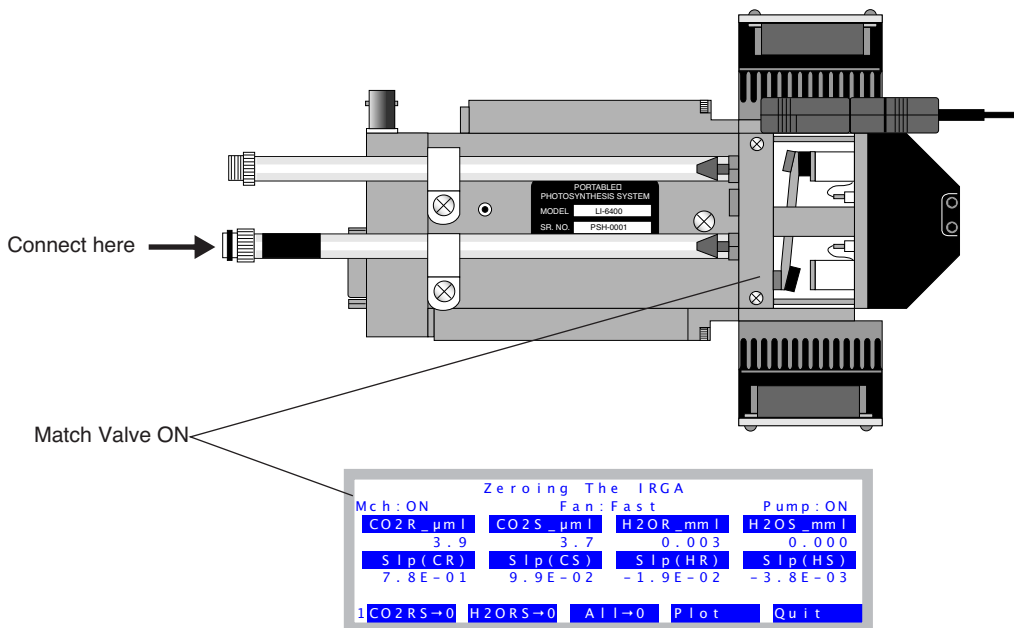
Figure 18-10. When connecting to a tank of CO_2 free air to zero the IRGAs, connect the flow to the sample inlet, and have the match valve turned on. The chamber needs to be closed, as well.

6   **When stable, press All→0**

Since the air source is dry and CO_2 free, you can zero both IRGAs simultaneously.

What Can Go Wrong When Setting The Zero

When automatically zeroing the IRGAs, two things are done for each IRGA: 1) a D/A (digital to analog output) channel is set. The resolution of these channels is fairly coarse (19mV), so to make the zero really zero, 2) an adjustment term is computed. This second step ensures that the IRGAs will always zero - even if you zero the IRGA without having dry or CO_2 free air in them, and are outside the range that the D/A portion can handle. It is therefore a good idea to check the values of these two components to make sure they are both reasonable (**View Current** on page 18-3).

• **Not Zero Air**

The most common problem results from zeroing the IRGAs with air that's not really CO_2 free, or not really dry.

• **Post-Zero Drift**

After zeroing the CO_2 and/or H_2O IRGAs, if you see continued drift before exiting the zeroing routine, it likely means that you zeroed them too soon, before they had equilibrated. This is especially true of water vapor.

• **Didn't zero very well**

CO_2 should be within 1 μmol mol^{-1} of zero after zeroing, and H_2O should be within 0.1 mmol mol^{-1} of zero. If things are off more than that, go check the values (**View Current** on page 18-3). It may be that you had very non-zero air when you zeroed, or else the IRGA has a problem.

Individually Zeroing the IRGAs

The big advantage of zeroing the reference and sample cell individually is that it is much faster. Here's why: first you zero just the reference cell, which always stabilizes much sooner than the sample cell. Then, you turn on the match valve so the reference cell sees the sample cell air, and you adjust the sample cell to match the reference cell reading. Here's the step-by-step:

1 **Connect with a "Y" if you are using tank air.**

See part B of Figure 18-12 on page 18-19.

2 **Match valve OFF.**

Flow the zero air, wait for reference cell stability.

3 **Zero the reference cell**

Press **2** to get the level 2 fct key labels, which should look like this with the match valve off:

Press **f1** to zero CO_2, and/or **f3** to zero H_2O.

4   Match Valve On

Press **m** to turn on the match value, and the fct key label should change to

| 2 | | CO2S→R | | H2OS→R | |

5   Match Sample to Ref

Wait 10 or 15 seconds for the reference cell to get flushed out, and replaced with sample cell air. Then press **f2** (and/or **f4**) to "zero" the sample cell. (It is a zero adjustment, but the target is no longer 0. Instead, the target is whatever the reference cell is reading.)

Manual Adjustment Keys

The manual zero adjustment keys are on levels 3 and 4. You can probably

| 3 | CO2R↑ | CO2R↓ | | CO2S↑ | CO2S↓ |
| 4 | H2OR↑ | H2OR↓ | | H2OS↑ | H2OS↓ |

safely ignore them, since their historical use was automated in the level 2 fct keys in version 6.2. They allow direct control of the 4 D/A channels that do the coarse adjustment of zeroing. When one of the manual zeroing function keys is pressed, the value of the D/A channel that controls that IRGA's zero is displayed in the upper right (Figure 18-11).

Figure 18-11. The 2nd and 3rd level function keys allow individual IRGAs to be adjusted manually.
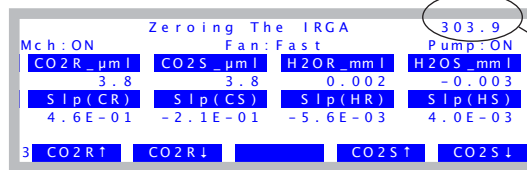
The value of the D/A channel for the IRGA last adjusted manually.

Setting the CO_2 Span

To check the span of the CO_2 analyzer, you'll need a known concentration of CO_2. Generally, this is provided by a tank of compressed gas (CO_2 in *air*, not nitrogen) that has been certified, or (even better) that has been measured using a properly calibrated gas analyzer. The concentration should be at or above where you work most of the time, but anything from 300 to 3000 μmol mol^{-1} would be fine[5].

[5.]At the factory, we use our 1500 μmol mol^{-1} tank for setting spans.

Setting the CO_2 IRGA gain is a process by which the user can manually (using the arrow keys ↑↓) adjust the values of G_{cr} and G_{cs} (these items are defined in Equations (14-10) and (14-11) on page 14-8).

■ **Checking/Setting the CO_2 Span**
Look at Figure 18-12 on page 18-19.

1 Select "IRGA span" **from the Calib Menu**

2 Set the flow from the tank.
See the comments under step 1 on page 18-14.

3 Attach to the IRGA.
You have a choice here, as shown by Figure 18-12: either attach to the sample inlet on the sensor head, and have Match ON,

-or-

Use a Y connector to split the flow and attach to both sample and reference ports simultaneously, and have Match Off.

4 Adjust the span settings as needed
When *CO2R_µml* is highlighted, you are adjusting the CO_2 reference IRGA's span factor. The sample IRGA is *CO2S_µml*.

The span values should be close to 1.0, as described in **What Can Go Wrong Setting the Span** on page 18-22).

```
        I R G A   S p a n   S e t t i n g

T h i s   r o u t i n e   a l l o w s   y o u   t o   a d j u s t   t h e
g a i n   o f   a n y   I R G A .   T h e   c h a m b e r   m u s t   b e
c l o s e d   a n d   e m p t y ,   w i t h   t h e   k n o w n   g a s
f l o w i n g   t o   t h e   C H A M B E R .

O K   t o   C o n t i n u e ?   ( Y / N )
```

```
P l e a s e   1 )   C l o s e   t h e   c h a m b e r   ( n o   l e a f )
            2 )   C o n n e c t   g a s   d i r e c t l y   t o   I R G A

A c t i v e   k e y s   w i l l   b e :
  ↑↓   ( a n d   s h i f t e d )   –   c h a n g e   t h e   s p a n .
  ←→   –   c h a n g e   I R G A
  F , S   –   s e t   f a n   s p e e d ,   M   –   m a t c h   m o d e
O K   t o   b e g i n ?   ( Y / N )
```

Active IRGA. To change which IRGA you are adjusting, use the FCT keys, or else use → and ←.

Span factors for each of the IRGAs. Press ↑ or ↓ to adjust the reading (and this value) up or down for the active IRGA. These values are G_{cr}, G_{cs}, G_{wr}, and G_{ws} used in Equations (14-5) through (14-11) starting on page 14-6.

```
        A d j u s t i n g   C O 2 _ R   ( u s e   ↑↓ )
CO2Rµml   CO2Sµml   TD_R_°C   TD_S_°C
    3.8       3.8     -99.90    -99.90   ◀Mean
5.4E-01 -4.4E-01   3.6E-10   3.6E-10   ◀Slope
  ↑   ↓   ↑   ↓                           Fan:Fst
    0.991     0.992     1.003     0.998   Mch:off
+CO2_R   CO2_S   H2O_R   H2O_S   Done
+Plot
```

Press the function key that corresponds to the IRGA you wish to adjust.
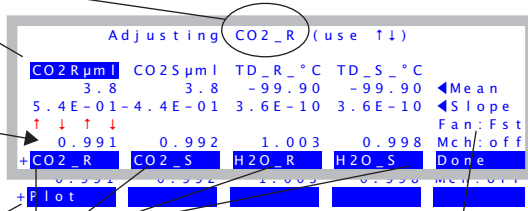
Watch graphs instead of looking at numbers.

Press **F** to adjust the fan speed (fast, slow, off) and **M** to toggle the match valve.

A If you connect only to the sample port, then Match must be ON for the air to get to the reference analyzer as well.
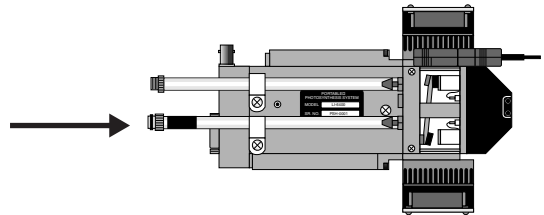
B If you split the flow, Match can be Off. (By turning Match ON and Off while watching the reference concentration, you can see the influence of the sample cell and chamber on the air stream)
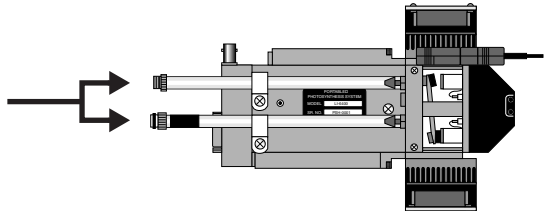
*Figure 18-12. Adjusting the reference CO_2 span. Real-time values for the four IRGA channels are shown (*Mean*), as well as their rates of change (*Slope*). The span factors (which should always be near 1.0) are used as multipliers of the IRGA voltages. Pressing ↑↓ increases or decreases the span factor for the active IRGA by a factor of 0.001. To change the span factor by 0.010, hold **shift** down while you press ↑ or ↓. To change the active IRGA, press the **F1** through **F4**, or else use ← or →.*

Setting the H_2O Span

To check the span of the H_2O analyzer, you'll need a known concentration of H_2O. The best choice for this is the LI-COR Dew Point Generator (LI-610).

> If you do not have an LI-610, or some device of similar accuracy, do *not* adjust the IRGA span values for water.

The H_2O IRGA gain adjustment is a process by which the user can manually (using the arrow keys ↑↓) adjust the values of G_{wr} and G_{ws} (see Equations (14-5) and (14-6) on page 14-6).

■ **To set the H_2O Span**

1 **Setup the LI-610 for an appropriate dew point**
Subtract about 5 °C from room temperature, and use that for the target dew point temperature. Wait until the condensor's temperature (as monitored on the LI-610) reaches this target.

The reason for this 5 °C "buffer" is to avoid condensation in the line between the LI-610's condensor and the IRGA. If condensation happens, you will have large errors.

2 **Set the flow rate.**
Use about 500 ml min^{-1} flow from the LI-610.

3 **Attach to the IRGA**
You have the same choice, as shown in Figure 18-12 on page 18-19. Here, however, we would recommend option B: splitting the flow and connecting both the reference and sample, with Match Off. The reason for this is that you will be able to drastically reduce the equilibrium time, waiting for the sample cell, as you will see.

4 **View water channels**
Press **F3** or **F4** to make the water IRGAs the active ones.

5 **Wait for equilibrium**
Watch the rates of change (slopes). If you are using Option A (connected to sample, Match On), then be prepared to wait about 20 minutes, until the rise in sample and reference concentration is negligible.

If you are plumbed for Option B (sample and reference connected), ignore the

sample, and only wait for the reference to equilibrate; 3 to 5 minutes should be adequate.

Either way, you can watch a graph of the reference and sample concentrations by pressing **Plot** (**f1** level **2**).

6 **Adjust the reference gain as needed**
When $Td\_R\_{}°C$ is highlighted, press ↑ and ↓ to adjusting the H_2O reference IRGA's span factor until $Td\_R\_{}°C$ reads correctly (Figure 18-13).


```
        Adjusting  H2O_R  (use  ↑↓)

CO2Rµml  CO2Sµml  TD_R_°C   TD_S_°C
    3.8      3.8     19.69     19.56  ◀Mean
-2.9E-01-1.1E-02-4.9E-03  3.5E-03  ◀Slope
                  ↑   ↓  ↑   ↓        Fan:Fst
    0.991    0.992   1.003     0.998  Mch:off
+CO2_R    CO2_S    H2O_R     H2O_S    Done
```
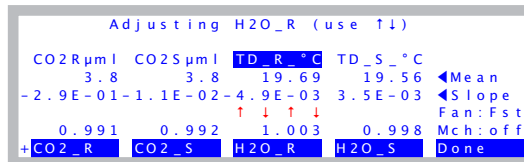
Figure 18-13. Adjusting the H_2O span. Adjust until the displayed dewpoint value matches the output of the LI-610.

7 **Select the sample IRGA**
Press → (or **f4**) to highlight $Td\_S\_{}°C$.

If you are plumbed for Option B, continue with Step 8.

If you're plumbed for Option A, press ↑ and ↓ to adjust $Td\_S\_{}°C$ until it reads correctly. You are done.

8 **Match mode ON**
Press **M** to make the Mch: indicator read ON.

9 **Note the reference dew point value**
Watch the Td_R_C value. It will likely drop a bit, as the still unequilibrated air from the sample cell enters the reference cell. When it stabilizes (30 seconds), set the *sample* IRGA (Td_S_C) to read the Td_R_C value.

Do you see the trick here? We first spanned the reference IRGA so it's reading correctly. The sample IRGA is seeing slightly drier air because water sorption is still going on, and we're losing water to the walls of the sample cell. When we put match mode on, the reference cell changes from seeing air directly from the dew point generator, to air that has been modified by the sample cell. This allows us to measure the sample cell dewpoint. We then use this value as a momentary target for setting the sample IRGA.

What Can Go Wrong Setting the Span

The span factors should be within 0.05 of 1.0 (that is, 0.95 to 1.05)[6]. The far-ther out of this range you go in attempting to make the analyzer read the cor-rect concentration, the more likely it is that something is wrong, such as:

- **Badly zeroed analyzer**
  Make sure you have a good zero before setting the span.

- **Concentration not what you think it is**
  (CO_2) Has the span gas ever been independently checked? Don't believe even the most reputable vendor of calibration gasses; after all, someone could have accidentally put the wrong label on the tank or its paperwork. It happens.

  (H_2O) Is there water in the condensor? Are you asking for a target that is at or above the room temperature? If so, you won't get that dew point, but you will get trouble in the form of condensation somewhere in the line.

- **Leak in the chamber**
  The chamber has to be well sealed for this to work.

- **Match valve set incorrectly**
  If the match valve isn't in the right position, the sample cell will be seeing the tank air, but not the reference cell. Also, is the match valve in fact working? Don't trust the display - look at the underside of the IRGA to see its position.

- **Plumbing mistake**
  (CO_2) Is the tank air really getting to the sensor head?

  (H_2O) Is the LI-610 pump on? Is there flow going to the sensor head?

  (Both) Is the tube connected *directly* to the sensor head. Do NOT connect the tank or LI-610 to the console Input air connector when setting spans. Space does not permit listing all the reasons why that is a bad idea.

- **"IRGAs Not Ready"**
  If this message is flashing on your display, then there are more pressing prob-lems to be addressed (see **"IRGAs Not Ready" Message** on page 20-15), and you certainly shouldn't be setting the span.

- **IRGAs not responding**
  See **IRGA(s) Unresponsive** on page 20-16.

---

[6]·You are not allowed to adjust the span outside of the range 0.9 to 1.1. You can override this with **shift + J**, however.

Flow meter

Flow in the LI-6400 is measured with an electronic mass flow meter in the console.

Factory Calibration

The flow meter is calibrated at three temperatures using a mass flow controller. (The flow controller is, in turn, periodically calibrated with a precision device that operates on a volumetric displacement principle.) The flow data for all temperatures is fit with a straight line, whose slope becomes the a_f term in Equation (14-16) on page 14-10. This value is shown on the LI-6400 calibration sheet, and is viewable. See **View Current** on page 18-3.

Zeroing the Flow meter

The flow meter zero program is run at the user's discretion. The program is fully automatic. When you select "Flow meter zero" in the Calib Menu, the flow and chamber mixing fan will be shut off, and the system will begin a 10 second count down (Figure 18-14). After ten seconds, the flow meter signal should read within 1 mV of zero. Press **OK**.

When entered, the zero routine turns off the pump, waits 10 seconds, and automatically zeros the flow meter.

```
         FLOWMETER ZERO

 7 secs to Zero Flowmeter

Flowmeter signal:   2.9 mV
 NoAuto                          Cancel
```
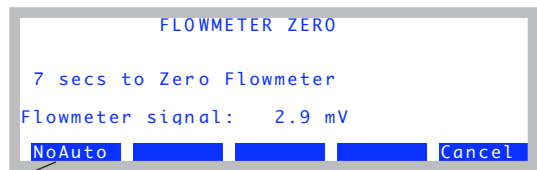
Skip the Automatic Zero

Afterwards, you can monitor the flow meter signal, and manually adjust the D/A channel that is used to zero it. If you do this, the D/A value will be shown.

```
         FLOWMETER ZERO

<-117.2>
Flowmeter signal:   0.8 mV
 Adjst□  Adjst□              OK
```
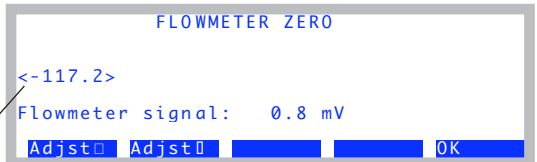
Figure 18-14. Zeroing the flow meter.

The flow meter's zero has a slight temperature sensitivity, but the drift is typically less than 1 μmol s^{-1} per 10 °C. The flow meter zero is controlled by an analog output channel, and should not change much from day to day. Since checking and setting it is so easy, it is worth the effort to make sure.

Zeroing the Leaf Temperature Thermocouple

The only calibration required for the leaf temperature thermocouple is its zero adjustment. It's a good idea to check this at the start of each day; not because it drifts that much, but because it's easy to do, and is part of good pre-operation practice. After all, if you have a broken thermocouple it's better to find out before you make your measurements than after.

1   Unplug the thermocouple connector
Remove the male thermocouple connector by pulling straight out.

2   Monitor TBlock and Tleaf
Configure the New Measurements screen so that you can view both the leaf temperature and block temperature variables (*Tleaf °C* and *Tblock °C*, respectively). In the default display configuration you can view these variables by pressing **H**.

3   Make them read the same
There is a small adjustment screw located on the underside of the sensor head, near the rear of the analyzer housing (Figure 18-15). Once the instrument has been on for about 30 minutes to warm up, use the small flat head screwdriver in the spares kit to turn the adjustment screw until the displayed leaf temperature and block temperature match.
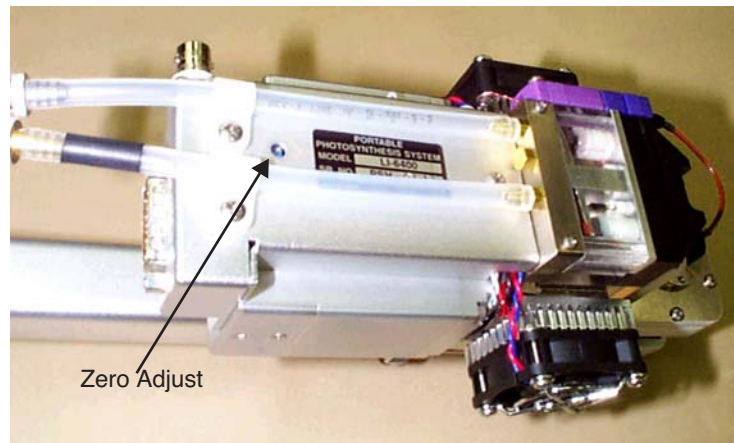
Figure 18-15. Location of the leaf temperature zero adjustment screw.

4   Reconnect the thermocouple connector.

6400-01 CO$_2$ Mixer

The 6400-01 CO$_2$ Mixer regulates CO$_2$ concentration to some specified target, which is communicated to the mixer as a command voltage. The relationship between the command voltage and the resulting CO$_2$ concentration depends on the bulk flow rate, temperature, and condition of the control apparatus. There is no calibration per se of the CO$_2$ mixer, since it depends on the CO$_2$ analyzer to "report back" what concentration is being achieved. However, there is a calibration that relates command voltage to resulting CO$_2$ concentration, and it is described below.

Calibrating the CO$_2$ Mixer

Before you start, make sure the CO$_2$ source (12 gram cartridge or external tank) has been connected for a few minutes. In the case of the cartridge, make sure it's reasonably fresh; they only last 8 hours once pierced, whether used or not.

1   **Run the CO$_2$ Mixer Calibration program**

In OPEN's calibration menu, select `"Calibrate"` under `"CO2 Mixer"` in the calibration menu (Figure 18-16).
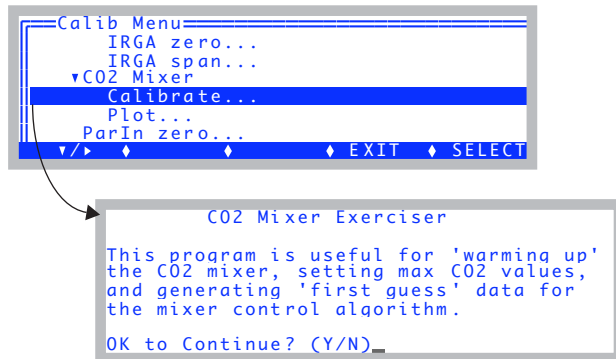
Figure 18-16. Running the mixer calibration program.

When prompted to continue, press **Y**.

2   Wait for the upper limit

If the mixer isn't already on, the system turns it on, and sets its target to the highest possible value (5 volts). The program then waits for stability of the reference CO_2 reading to be achieved (Figure 18-17).

Mixer status is explained in **"MIXR"** on page 14-16.

The range over the past 10 readings (5 seconds)

```
Waiting for mixer to stabilize...

CO2R_µml    Status   Flow_µml   Rangeµml
   1190.3      LOW      189.8      31.21
Pump: 4600 mV       Time:165

earlyOK  Pump+    Pump-              Abort
```

Press this to override the program's wait for stability (Range<1 and Status=OK) and go on to the next step.

You can control the pump speed with these keys while you are waiting, between 1000 and 5000. The slower the pump, the higher the upper limit of CO2 you can achieve.

Quit and exit, restoring everything to its prior state.

Figure 18-17. Waiting for the mixer to achieve its highest possible concentration.

Stability is defined as when the *Rangeµml* value drops below 1.0 while the *Status* indicates "OK". The function key **earlyOK** will override these conditions and force the software to proceed to the next step.

3   OK the upper limit

Once the mixer is stable at its maximum value, this value is presented with an option to change it or continue on (Figure 18-18).

```
The Max CO2 is about  2258 ppm.
Is this OK ?
    Y - Yes, continue
    N - No, adjust pump speed
  <esc> - abort
```

Figure 18-18. The upper limit presented.

(Note: If the max value is less than 2000, then there's a problem. See **Can't Achieve High Values** on page 20-28.)

If this is OK, press **Y** and go to step 4. Otherwise, press **N**, and adjust the pump speed, and press **earlyOK** when you are ready.

> ### Adjusting the Upper Limit
>
> The 6400-01 CO_2 Mixer option is specified as having an upper range of 2000 μmol mol^{-1}, and the lower limit of control is typically 40 or 50 μmol mol^{-1}. This range can be adjusted by varying the pump speed. When the 6400-01 is installed, the pump runs at a constant voltage, and flow control is done by a flow controller downstream of the point of CO_2 injection that diverts air away from the sample path; the excess flow is routed to the reference path (Figure 1-2 on page 1-5). Without the 6400-01, flow rate is controlled by varying the pump speed. For any rate of injection of CO_2, a reduced pump speed results in lower bulk flow and increased CO_2 concentration.

4   Automatic mixer calibration

The program loops through a range of set points, from 5000 mV down to 0 mV, and records the CO_2 concentration for each (Figure 18-19).
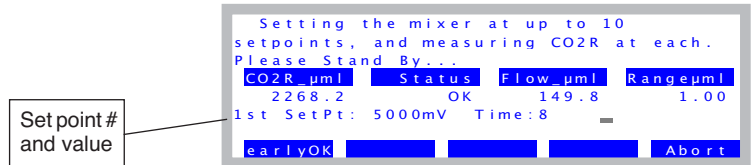
Figure 18-19. While doing the automatic calibration, the display will show CO_2 mixer set point value and resulting CO_2 concentration.

Generally, it should take about 20 to 30 seconds to do each set point. The criteria is that *Rangeµml* must be < 1, and the Status must be OK. Pressing **f1** (**earlyOK**) causes the point to be accepted, and it moves to the next set point.

Normally, this takes 8 setpoints, and the low CO_2 concentration will be about 40 or 50 ppm. If you have modified the LI-6400 console plumbing according to App Note 7 (*Modification of LI-6400 to Control at Low CO_2*, PPS-267), then the program will automatically detect this from the mixer's response, and take two extra steps down to 0 ppm.

5   View the calibration curve

Once the calibration is done, the data is printed and you are given an option to plot it (Figure 18-20).

```
  3000      1068.8        18        0.81
  1500       424.1        18        0.65
  1000       258.9        16        0.95
   500       117.5        17        0.47
   300        67.9        15        0.67
   200        44.3        15        0.40
   100        42.2        15        0.09
Plot this? (Y/N)
```

```
 2.49E+03
          +
   +-CO2
                                              +

   -180
      -390          SETPT          5.49E+03
```
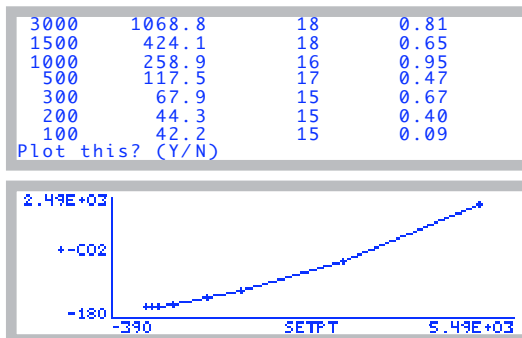
Figure 18-20. The mixer calibration

You can also see this plot at any later time, by selecting "Plot" under "CO2 Mixer" in the Calib Menu.

6   Implement the calibration

You will be asked if you wish to implement this calibration or not (Figure 18-21). Press **Y** if you do.

```
Press Any Key
(V - show values)




Implement this cal? (Y/N)
```

Figure 18-21. Implementing the mixer calibration.

If you implement the calibration, the <user> calibration node is updated, as are the files /dev/.user and /dev/.userhistory.

Internal PAR Sensor (General)

The clear-topped chambers, LED light sources, and the LCF all have a sensor for measuring the PAR in the chamber. Each of these is discussed in subsequent sections, but there is an offset adjustment that is available that is common to all.

Zeroing the ParIn Signal

Some instruments, at various times and for various reasons, might exhibit a slight offset in the ParIn signal. That is, when the sensor is in the dark, the *ParIn_µm* reading will not be exactly zero. If this is a problem, you can work around the situation by applying an offset to the measurement to correct it.

The way to do this is "`ParIn zero...`" in the Calib Menu (Figure 18-22).
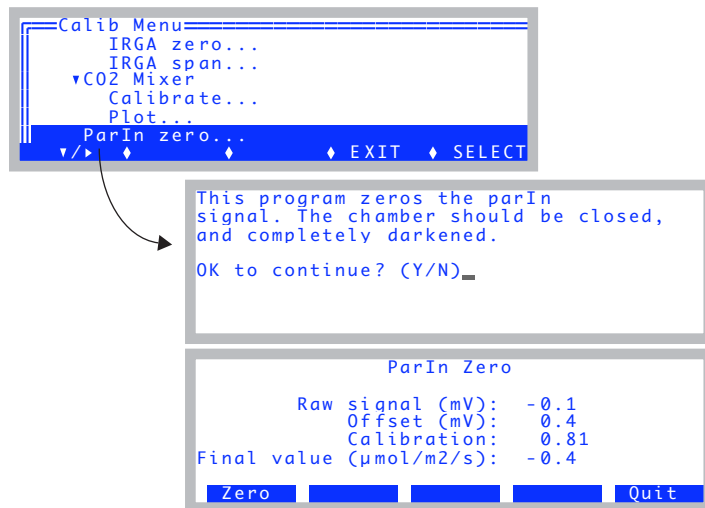
Figure 18-22. The program for zeroing the ParIn signal.

To zero the light sensor, press **Zero** (**f1**) when you are sure the chamber is completely darkened. The offset value will be retained (it will simply be the raw signal at the moment you pressed **f1**), and subsequently applied. (The equation for the final value is (14-17) on page 14-10.) The offset value is retained in the /dev/.user file. The offset value can be also be viewed in New Measurements mode in Diagnostic Screen E (page 6-27).

6400-02(B) LED Source

There are some things to consider in discussing 6400-02 LED Source calibrations:

- **It is a light source**
  The desired brightness of the LEDs is communicated to the source by means of a command voltage. The relation between this signal and light output requires a calibration that is a function of temperature and LED age. This calibration is easily done by the user, and is described below.

- **It contains a light detector**
  The relationship between the internal silicon photodiode's output and the real quantum output of the LEDs is measured at the factory, but will drift as the detector ages. This calibration is done at the factory, and should be checked every two years.

  Aside: We do not recommend using an LI-190 quantum sensor to calibrate an LED light source detectors for two reasons: First, unit-to-unit variations in a quantum sensor's response within the 400-700 nm range (i.e., those "wiggles" in its response function) can potentially cause significant measurement discrepancies when measuring a narrow-peaked source, such as LEDs. Secondly, the closer the LED is to the 700 nm cut-off in the quantum sensor sensitivity, the more potential you have for differences due again to unit-to-unit variations. Also, temperature induced variations in spectral output and sensitivity can contribute to apparent shifts in response when measuring LEDs with a quantum sensor. Factory calibrations of LI-COR LED light sources involve a spectroradiometer, which does not suffer from the aforementioned shortcomings.

- **It ages**
  As the source ages, its maximum output drops. This affects the user calibration, but not the factory calibration. The factory calibration has to do with the how the detector ages, not the LEDs.

Light Source Calibration

Calibration data relating the LED Source's command signal with light output can be generated by running the light source calibration program in the Calib Menu (Figure 18-23).
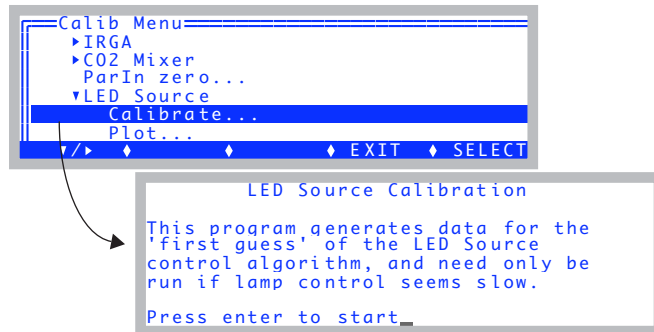
Figure 18-23. The LED Source Calibration's opening screen.

The program goes through a series of command voltages (10, 20, 50, 100, 1000, 2000, 3000, 4000, and 5000 mV), at each waiting for 10 seconds then recording the *QNTM* value (PAR in μmol m^{-2} s^{-1}) (Figure 18-24).
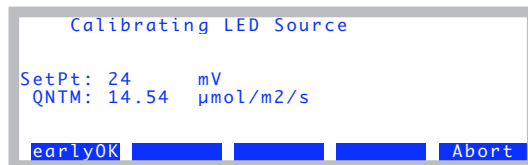
Figure 18-24. The calibration curve is generated automatically. SetPt is the command signal, and QNTM is the resulting PAR (μmol m^{-2} s^{-1}).

When done, the program will show the data and provide you with an opportunity to plot the relation between set point and output (Figure 18-25).

```
    50       39.0
   100       91.1
  1000     1063.5
  2000     2073.6
  3000     3020.2
  4000     3920.0
  5000     3939.9
Plot this? (Y/N)
```
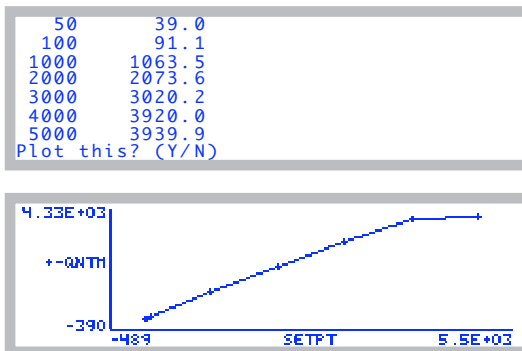

Figure 18-25. The LED source calibration data is displayed, and you are given a chance to plot the graph.

To actually implement this calibration data, you must respond by pressing **Y** when asked (Figure 18-26).


```
Press Any Key
(V - show values)



Implement this cal? (Y/N)_
```

Figure 18-26. To implement the calibration, press Y.

The calibration data is visible in the user calibration tree under the <li6400> <user> <led_cal> node, so is thus visible in the "View Current..." program under "View Settings" in the Calib Menu, and also in the Calibration History screen (Figure 18-27). (Note: the <led_cal> node is only visible in the current calibration while you are configured for the 6400-02B light source.)

Current Calibration Screen

```
┌──=Current Calibration══════════════════
│┌─►co2_mixer
││ ├ parin_offset= 0.433045
││ └▼led_cal
││  ├ unit= "SI-1267"
││  ├ mv= { 10  25  50  100  1000  2000
│    └ qntm= { 4.25933  16.3624  38.5714
   ▼/►    ◆          ◆       ◆ EXIT   ◆ SELECT
```

Calibration History Screen

```
═══════════Calibration History═══════════
▼2008-10-14 15:28 led_cal: SI-1267
 └▼li6400
  └▼user
   └▼led_cal
     ├ mv= { 10  25  50  100  1000  2000
     └ qntm= { 5.04895  15.9909  38.1456
+   ▼/►  ◆Add    ◆setTo  ◆      ◆Done
```

Figure 18-27. Viewing the 6400-02B calibration results after the fact.

6400-18 RGB Light Source

There are two basic calibrations for the 6400-18 RGB Light Source: one is a factory calibration (three values), that relate the internal detector's output to true lamp output for each of the three colors of LED. The second is a user calibration (six values) that relates control signal to light output, again for each of the three colors.

RGB Calibration Editor

These values can be viewed and edited from the Calib Menu (Figure 18-28).
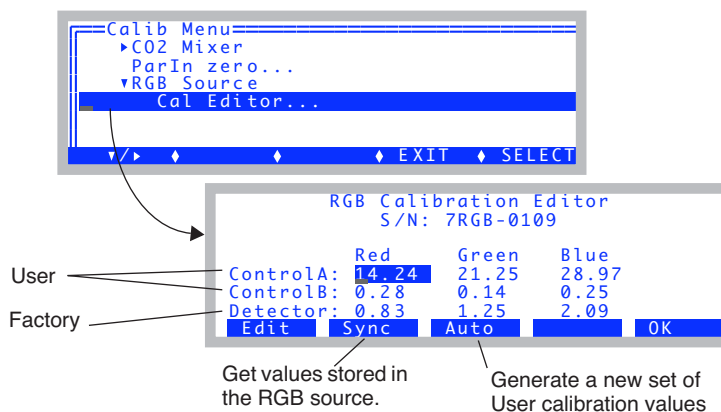
*Figure 18-28. The RGB Calibration Editor. The highlighted box can be moved by ↑ → ↓ and ←. **Edit** to change any value.*

Any of the nine values can be changed by highlighting the value and pressing **f1** (**Edit**). When a new value is entered, it is transmitted to the RGB Light Source and saved in its memory.

Reading back from the RGB source is normally not necessary, but if you want to, use **f2** (**Synch**). The Synch routine *is* necessary if you are switching between multiple units, since that's how to get the calibration coefficients that are stored in the currently attached unit downloaded to OPEN. Synch happens automatically when you first select a configuration with an RGB light source (or change the light source node in the configuration to RGB).

To generate a user calibration, use the **Auto** key (**f3**).

```
          RGB Lamp Calibration

This program performs independent
calibrations of the red, blue, and
green LEDs of the 6400-18 RGB Source.

Press enter to start, or <esc> to quit_
```

Does Red, Green, then Blue ⟶

```
         Calibrating RED LEDs

SetPt: 5       %
 QNTM: 96.67   μmol/m2/s   _

earlyOK                         Abort
```

Does several set points for each color.
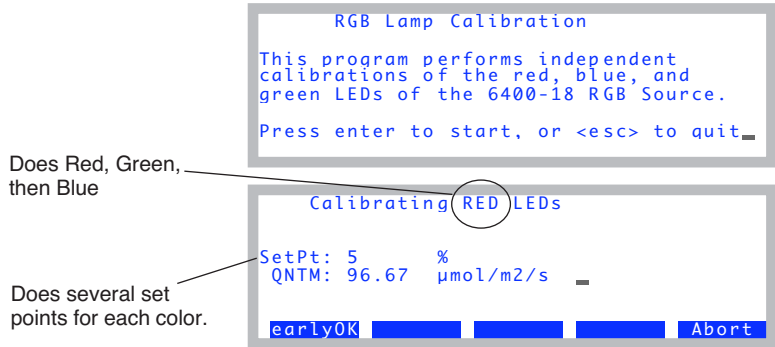
Figure 18-29. The RGB automatic calibration.

Data is collected for each of the three colors relating control signal to light output. The relation between quantum output Q_i of color i and control signal V_i is not quite linear, so we use

$$Q_i = \frac{a_i V_i + b_i V_i^2}{1 + 0.02 V_i} \qquad (18\text{-}1)$$

and determine the coefficients a_i and b_i. When the measurements are over, you can see a plot of the results, both the points and the fitted curve (Figure 18-30). If you implement the calibration, the coefficients will be sent to the RGB Light Source for storage, and the Calibration Editor will display the new values. The three values in the *ControlA* line are a_i, and the three in the *ControlB* line are b_i.

Calibrating the RGB Source leaves no trace in the current calibration or the history file. Rather, all the values are retained in the memory of the light source itself.

```
     1       12.6        33.8        52.1
    10       92.7       280.4       406.1
    20      206.1       518.2       742.2
    40      324.3       921.0      1337.6
    60      444.5      1258.9      1870.0
    80      565.7      1545.7      2346.3
   100     2050.2      1788.1      2766.3
Plot this? (Y/N)
```

Press **Y**

```
2.25E+03

  +:RED

 -191
      -8.9           SETPT        110
```

Press **escape**

```
1.96E+03

  +:GREEN

 -142
      -8.9           SETPT        110
```

Press **escape**

```
3.04E+03

  +:BLUE

 -219
      -8.9           SETPT        110
```

Press **escape**

```
Press Any Key
(V - show values)




Implement this cal? (Y/N)_
```

Press **Y**

```
         RGB Calibration Editor
            S/N: 7RGB-0109

            Red      Green    Blue
ControlA:  23.35    33.72    45.79
ControlB:  0.38     0.20     0.37
Detector:  0.83     1.25     2.09
   Edit     Sync     Auto            OK
```
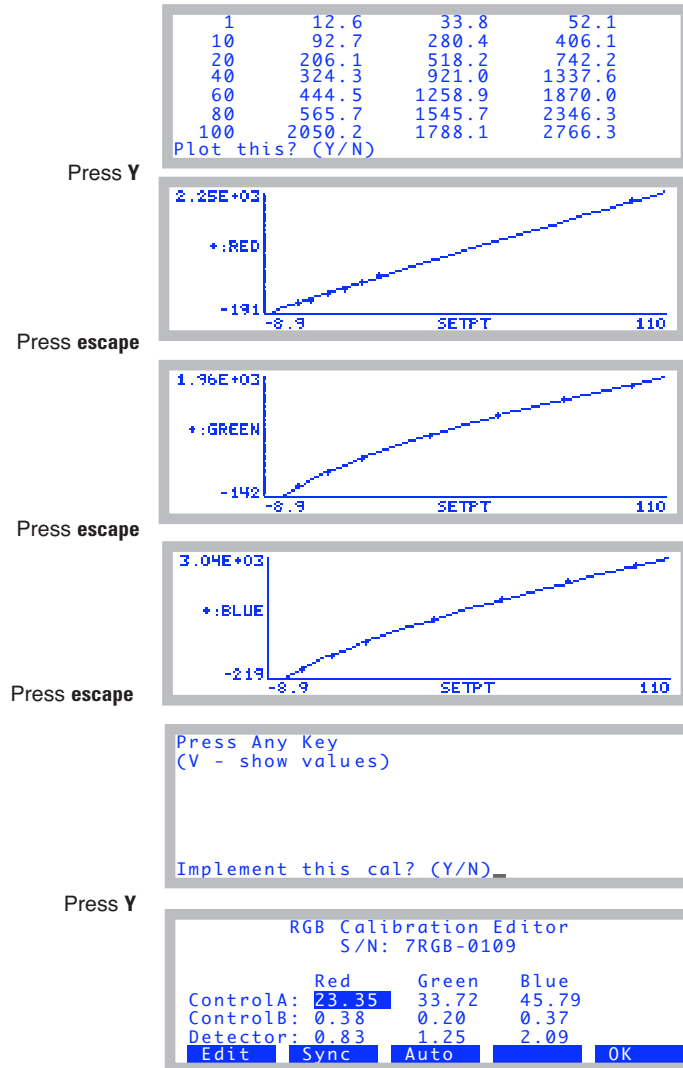
Figure 18-30. After the automatic calibration.

6400-40 Leaf Chamber Fluorometer

There are two separate items that have calibration requirements in this accessory: the light source, and the fluorometer.

Light Source Calibration

The light source consists of red and blue LEDs, and its calibration requirements are similar to the 6400-02. One difference is that the red and blue LEDs are independently controlled, and thus each has its own calibration. See **Calibrate...** on page 27-75 for details.

Fluorescence Calibration

The fluorometer part of the LCF only requires zeroing. See **Zero Fluorescence Signal** on page 27-81.

GaAsP Light Sensors

Gallium Arsenide Phosphide light sensors are used in the standard 2x3 LI-6400 chamber top, as well as many of the optional chamber tops. Standard chambers have serial numbers GA-*nnn*, while accessory chambers with GaAsP sensors have GB-*nnn* serial numbers.

Factory Calibration

The factory calibration of a GaAsP sensor is done by placing the sensor a known distance from a standard lamp, and measuring the sensor's output. The calibration is dependent upon the spectral properties of the source (see Figure 8-23 on page 8-24). The calibration value that we provide is adjusted for a typical sun+sky spectrum, even though it is generated with a tungsten lamp.

The calibration is reported on the calibration sheet, and stored in the instrument, as the object of the CalParGaAs= configuration command. This value is used for the term a_g in Equation (14-18) on page 14-11.

Generating a Calibration Correction

A calibration correction factor (f_a in Equation (14-18) on page 14-11) of the GaAsP sensor in the leaf chamber can be performed for light sources not included in the Light Source menu in the Configuration list. Be warned that the results are extremely sensitive to view factors and incident radiation geometry. For best results, do this procedure with incident radiation that is as perpendicular to the leaf plane as possible, and keep the radiation geometry fixed. This method does not work well with strictly diffuse light sources because of view factor differences between the center of the leaf plane, leaf edges, and the GaAsP sensor.

With due care, proceed as follows:

1 **Install the quantum sensor mounting bracket**
Replace the lower portion of the leaf chamber with the 9864-111 Quantum Sensor Chamber Mount (in the spares kit) using the chamber mounting screws to attach the quantum sensor mount.

2 **Install the external quantum sensor**
With the quantum sensor mount in place and the chamber closed, insert a quantum sensor into the mount until it contacts the leaf chamber gaskets, and secure it with the set screw.

3   Orient the leaf chamber

Orient it so that the incoming radiation is perpendicular to the leaf plane. The distance from the leaf chamber to the light source should not be changed between light sensor calibration and photosynthesis measurements. This will minimize errors due to radiation geometry and view factors.

4   Set the Light Source to `Sun + Sky`

This will apply no correction factor for your calibration readings.

5   Record the readings

Note the PAR values Q_c (ParIn_μm) and Q_x (ParOutμm)

6   Compute the correction factor

The correction factor f_a is

$$f_a = \frac{Q_x}{Q_c} \qquad\qquad\qquad (18\text{-}2)$$

7   Enter the new value

Go to the Filer, pick the /User/Configs directory, highlight the file `"Light Source Control"` (Figure 18-31 top). Add your new correction factor to the list, as shown in the bottom part of that figure. You'll need a source name in quotes, followed by the f_a value, followed by a guess at the absorbed energy conversion factor (αk in Eqn (17-3) on page 17-2). Use 0.2 unless you have measured it. Press **escape** then **U** then **Q** when you are done.
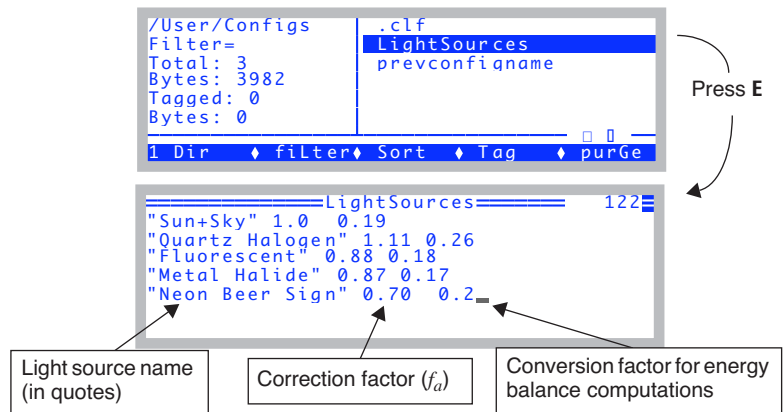
Figure 18-31. Each entry in the list source list needs a quoted name, an actinity correction factor, and a conversion factor (used for energy balance computations).

Maintenance & Service

19

The care and feeding of your new pet

19

Maintenance & Service

This chapter describes the maintenance and service tasks that may be required in the normal course of operating the LI-6400.

Chemical Tubes

The chemical tubes can be left in place on the console until it is time to replace the chemicals, or to service a tube's flow control assembly.

Flow Control Knob

Attachment Screw

TOP☐
Do Not open☐
this end

DESICCANT

Open this end☐
BOTTOM

Bottom Cap

Figure 19-1. A chemical tube. The only difference between the soda lime tube and the desiccant tube is the label.
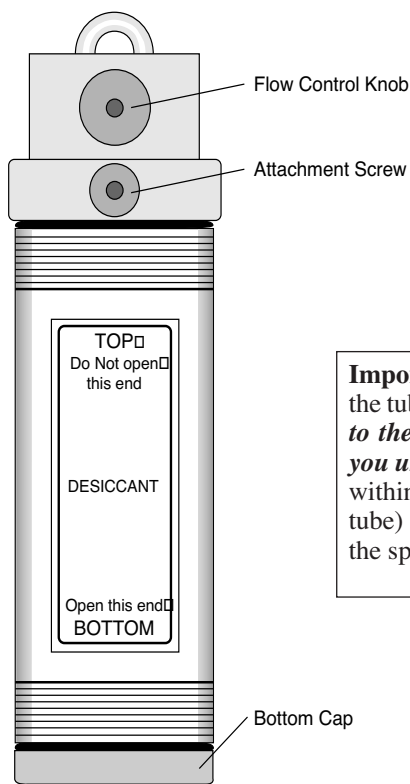
Removing

The tubes are removed from the console by loosening the attachment screw (Figure 19-1). If the screw is too tight to turn (and you should be turning it counterclockwise, as viewed in Figure 19-1), then before you resort to pliers, try wiggling the bottom of the tube left and right (orientation as shown in Figure 19-1) as you attempt to loosen the screw.

Important: To open a tube to replace its chemicals, turn the tube upside down and unscrew the *bottom* cap. ***Damage to the mufflers (Figure 19-2 on page 19-3) will result if you unscrew the top cap with chemicals in the tube***. Fill to within 1 cm of the end of the tube with Drierite (desiccant tube) or soda lime (soda lime tube). Both chemicals are in the spares kit.

Cleaning The End Cap Threads

It is important that the threads be kept very clean. If dust or other debris accumulates on them, you will have a difficult time putting the end cap on far enough to compress the O-ring, and the tube will leak. This is most often a problem with the desiccant tube, and can show up as instability in the CO_2 concentration in both reference and sample cells.

After you dump out the old chemicals from a tube, use a stiff brush to clean the threads of both the end cap and the tube barrel. For especially dirty threads, soak them in water briefly, then wipe them clean and dry.

Do not lubricate the threads. Lubricant will accumulate dust, and aggravate the problem. When you reassemble the tube, make sure the threads are clean and dry. No grease, please.
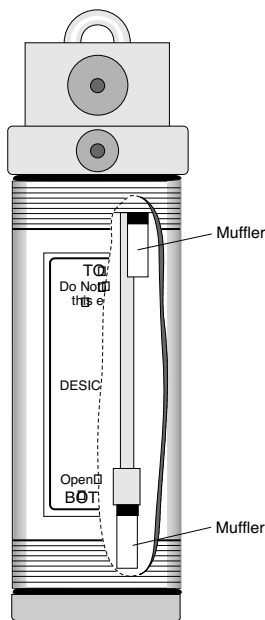
Replacing the Air Mufflers

There are two air mufflers attached to the air hoses inside the chemical tubes (Figure 19-2). These mufflers may become clogged, restricting air flow through the tubes and thus reducing the maximum flow rates. (This is one of several things that can reduce flow rates. See **Can't Achieve High Flow Rates** on page 20-13.)

To replace the mufflers, remove the bottom cap, empty the chemicals and then remove the top cap. The old mufflers can then be unscrewed and replaced with new ones. *The white part is glued to the black part and will break loose, if you try to screw or pull it out while holding the white part. Use a 1/4" wrench and touch only the black threaded end.* Do not overtighten the mufflers, as they can break at the threads very easily. There are air mufflers in the spare parts kit.

Figure 19-2. Air mufflers in soda lime and desiccant tubes.

The Desiccant Tube: Use Drierite

■ **To regenerate Drierite**
LI-COR recommends indicating Drierite (W.A. Hammond Drierite Company, P.O. Box 460, Xenia, OH 45385) for use with the LI-6400. Drierite is anhydrous 97% calcium sulfate ($CaSO_4$) and 3% cobalt chloride. Calcium sulfate is safe, chemically inert except toward water, economical, and can be regenerated. Drierite absorbs approximately 6.6% its weight in water. Indicating Drierite is blue when dry, and changes to pink upon absorption of water to signal when it should be replaced.

1 **90 minutes at 230 °C or 450 °F**
Preheat the oven and a shallow pan. Spread the granules in a single layer one granule deep and heat for the above time period. (Note: less heat for a longer time will not work.)

2 **Seal in a glass container while still hot**
After 90 minutes, place the hot, regenerated material back in its glass container and close the lid.

Note that the color of the indicating Drierite may become less distinct after successive regenerations. If it turns black, you've overheated it.

The CO_2 Scrub Tube: Use Soda Lime

Soda lime (calcium oxide and sodium hydroxide) removes CO_2 from the air stream, and adds a bit of water. Some brands add more water than others. When soda lime becomes very dry, it loses efficacy. This is not typically a problem for the LI-6400, since incoming air, which generally has some moisture in it, goes through the soda lime first.

Wet soda lime is available in the spares kit. The part number is 9964-090.

How often the soda lime needs to be replaced depends upon how much CO_2 it has been forced to remove. Loss of capacity to scrub CO_2 can be recognized by an inability to reduce the CO_2 mole fraction to zero and hold it there.

A Quick Soda Lime Test
Turn the soda lime tube on full scrub, and wait for the reference CO_2 to get as low as it will go. Then blow a puff of air at the inlet tube on the right side of the console, and watch the reference CO_2 reading. A positive spike of more than 2 ppm would indicate that the soda lime should be changed.

Reattaching

Here's a quick checklist for reattaching the tubes:

1   **Chemicals to 1 cm of the top.**
Leave a little room for them to move when shaken. Then you can easily break up channelling, which is the tendency for air to find a dominant passage through the chemicals, thereby saturating the chemicals unevenly in the tube.

2   **Threads clean and dry**
Discussed in **Cleaning The End Cap Threads** on page 19-3.

3   **Are the end cap O-rings slightly compressed?**
Not too tight, but no gaps allowed.

4   **Make sure the air passage tubes have O-rings**
See Figure 19-3. Also, make sure that the mounting surface on the console is clean.

5   **Not too tight**
The attachment knob should be <u>just slightly snug</u>, to squeeze the air passage O-rings a bit. Finger tight is fine, and *never use pliers*. Don't be concerned if the tube is not rigidly held against the console. A bit of wobble is normal.

Figure 19-3. Each chemical tube has two air passage holes protected by O-rings.

Chemical Tube Flow Control

The usual reason to take apart the flow control assemblies on the soda lime or desiccant tube is that one of the small flow tubes becomes pinched, or some debris clogs up the tubing, or gets in the small air passage ways.

■ **To disassemble and service a flow adjustment assembly**

1 **Set the flow adjust knob**
Put it midway between SCRUB and BYPASS.

2 **Remove the knob**
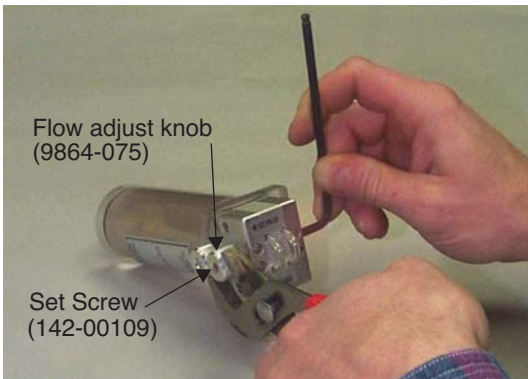
Use a 5/32 hex key to hold the flow adjust bolt, and loosen the flow adjust knob with channel lock pliers. (Turn the nut toward SCRUB to loosen.) Remove the nut and white washer that is beneath it.

The flow adjust knob has to go back on oriented the same way, so you should mark which face is out when it is attached. (The knob has a set screw (142-00109) inserted from the outside that sets the proper depth of the flow adjust bolt. Don't adjust the set screw.)

Flow adjust knob (9864-075)

Set Screw (142-00109)

Figure 19-4. Removing the flow adjust knob.

3 **Remove eight screws**
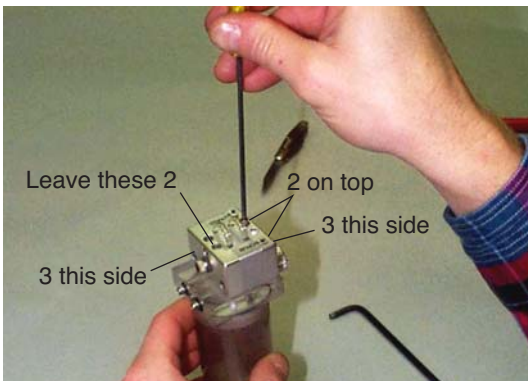
Using a 3/32 hex key, remove the three screws on each side of the assembly, and two of the four screws from the top. Leave the two top screws that are closest to the flow adjust bolt side (opposite the side where the nut was) of the assembly.

Leave these 2

2 on top

3 this side

3 this side

Figure 19-5. Removing eight screws from the flow adjust assembly.

4   Remove the outer and center pieces
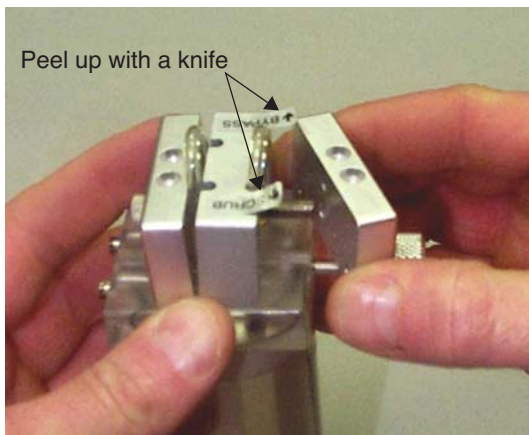
Peel up with a knife

You'll have to peel the ends of the SCRUB and BY-PASS stickers up with a knife, then the two loose pieces of the assembly can be removed.

Figure 19-6. Removing two of the three blocks from the flow adjust assembly.

5   Service as needed
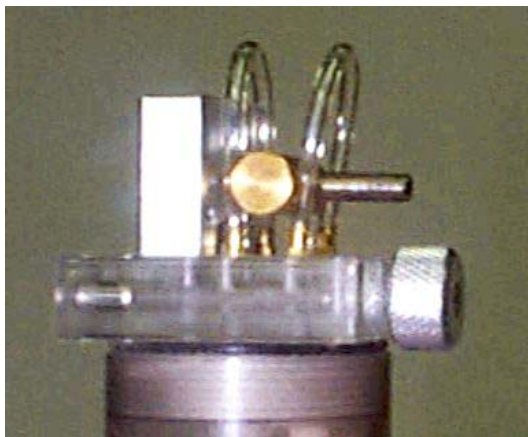
If the small tubing remains compressed long enough (especially in hot weather), they can seal closed. If that is the case, replace them. Also, check carefully in the hose barbs and air passages in the clear plastic base of the assembly for any debris that might be lodged there, blocking the flow.

The small tubing is polyurethane, 1/16 inch inside diameter, and 1/8 inch outside diameter.

Figure 19-7. The flow adjust assembly works by pinching one or the other of the two small air tubes with a rod that moves when the flow adjust knob is turned.

6   Reassemble

Reverse the procedure. When it is time to reattach the flow adjust knob, remember to put the white washer on first, and be sure the knob is oriented correctly. How do you tell? There is a set screw threaded into the knob; the back of the set screw accepts a 3/32 hex key (don't turn it!), and this is the side that should face away from the flow control assembly.

6400-03 Batteries

Charging the 6400-03 Batteries

Batteries are normally charged with the LI-6020 battery changer.

1 Select the proper voltage
Make sure that the voltage selector slide switch on the back of the LI-6020 battery charger is set to the appropriate line voltage (115 or 230 VAC).

2 Plug the charger into mains power
The AC indicator light will illuminate. If the charge indicator lights up instead, you've got the wrong voltage selected.

3 Connect batteries
The CHARGE indicator illuminates if any of the batteries connected to the charger are being charged. One method for testing a battery's charge is to connect it to the charger when no other batteries are attached. If it is charged, the CHARGE light comes on for only a few seconds if at all. If the CHARGE light does not come on, either the battery is fully charged, or else the battery's fuse has blown. (To test, plug the battery by itself into the LI-6400 console, and power it on. If nothing happens, then the problem is the fuse, or the battery is dead.)

A fully discharged 6400-03 battery requires about 3 hours to recharge. Four discharged batteries connected simultaneously would require approximately 10 to 12 hours to recharge. We recommend you not leave a battery on the charger for more than 24 hours after the charge light has gone out.

Charging with the 6400-70 AC Module

One battery at a time can be charged on the 6400-70 AC Module. This will be a trickle charge, so it takes about 6 hours to recharge a battery. (The real purpose of this feature is to keep a battery charged to run the system in case of mains power failure.)

Never recharge a battery by plugging it into the console at the same time that one leg of the 6400-70 AC Module is plugged into the other battery connector on that console (Figure 19-8).

Safe for extended operations

Battery safely trickle charges, and is available to run the instrument if mains power fails.
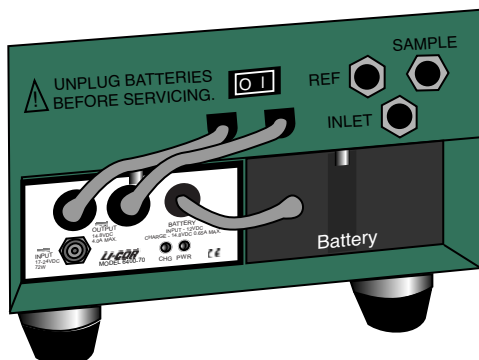
Don't do this for more than a minute

Battery is being charged in an uncontrolled manner. Damage could result to the battery and/or the transformer.
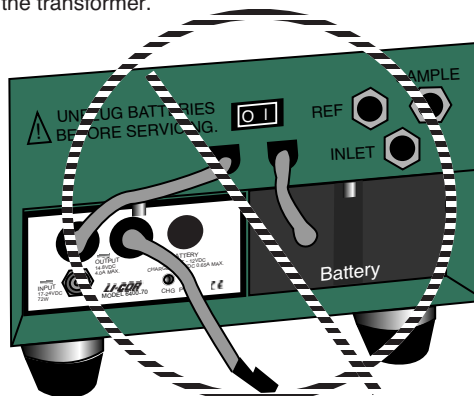
Figure 19-8. Avoid extended (more than a minute) operations with a battery and one of the AC module's plugs connected to the console. In this configuration, you will be providing uncontrolled charging of the battery, which could damage the battery and/or the transformer.

Storing the 6400-03 Batteries

Store batteries fully charged, and in a cool place, if possible. For long term storage, place the batteries on the charger overnight every three months.

Replacing the Battery Fuse

There is a 10A automotive type fuse located inside the metal cover of the 6400-03 battery. If the battery fails to power the LI-6400, and will not light the charge indicator on the battery charger, check to see if the fuse has blown.

To replace the fuse, cut the black tape on the battery pack along the long axis, between the two halves of the battery. Carefully remove the top half of the battery (the half with the electrical cables), and lay it to the side with the wires still attached. Check to see if the fuse has blown. Replacement fuses (part #438-03142, in the spares kit) plug into the spade connectors; no soldering is required. After replacing the fuse, tape the battery covers together.

System Console

Cleaning
Wipe with a soft cloth. Be careful not to scratch the display window.

Opening the Console
You will need to remove the cover to replace any of six different fuses, or to replace the internal air filter (Balston).

■ **To remove the console cover:**

1 **Disconnect everything**
Disconnect all cables and hoses from the console and remove the batteries.

2 **Remove the screws**
Remove the eight screws (122-00007) (nine screws on early units) on each side of the LI-6400 console case with a Phillips head screwdriver.

3 **Drop the cover**
Grip the carrying handle and lift the card cage out of the lower shell.

Internal Air Filter Replacement
The air filter should be replaced annually; more frequently in dirty environments.

The filter is located inside the LI-6400 (Figure 19-9 on page 19-12). Disassemble the console as described above

> **NOTE**: Console serial numbers PSC101-160 that do not have the 6400-01 CO_2 injector option will have two Balston air filters.

Before installing, blow through a new filter in the direction of the white flow arrow to remove any fibers or other debris that may be loose inside.

The filter is attached to the air line with a quick connector. The old filter can be removed by compressing the red ring into the quick connector body and pulling the connector off of the filter. Leave the connector attached to the hose, as repeated removal from the hoses may result in a leak. The filter may

also be removed by inserting a pair of long-nose pliers between the coupling and the filter; gently pry the two apart, using the filter body as a fulcrum.

Install the filter with the white directional arrow aimed in the direction of the air flow; air flows to the filter *from* the desiccant tube. (See Figure 20-19 on page 20-46 or Figure 20-23 on page 20-51 for flow schematics.)

Spare filters can be ordered from LI-COR under part number 300-01961 (one each).

Replacing the Fuses

Remove the card cage as described above. There are six fuses located on two different circuit boards; there are three fuses on both the backplane board and the flow board (Figure 19-9). Table 19-1 lists the replacement fuses that should be used. Extra fuses can be found in the spare parts kit.

Table 19-1. Replacement fuse sizes for the flow and backplane boards.

Flow Board			
Fuse	**Size**	**Protects...**	**Part Number**
F1 (Circ. fan)	3A Fast blow, 250V	Circulating fan	439-04215
F2 (TEC-)	5A Fast blow, 125V	Thermoelectric coolers	439-04214
F3 (TEC+)			
Backplane Board			
Fuse	**Size**	**Protects...**	**Part Number**
F1 (Analyzer)	3A Fast blow, 250V	CO_2/H_2O analyzers	439-04215
F2 (Flow)		Flow controller board	
F3 (Dig. Bd.)	1A Fast blow, 250V	Digital board	439-04216

When installing a fuse, be sure to exactly center it in the holder. If you don't, one end of the fuse will hit the retaining part of the clip and spread that clip too wide, eliminating contact once the fuse is fully inserted.

Flow board

Backplane board

F3 (TEC+) 5A
F2 (TEC-) 5A
F1 (Circulation fan) 3A

F1 (Analyzer board) 3A

Display Connector

F3 (Digital board) 1A
F2 (Flow board) 3A

Air Filter
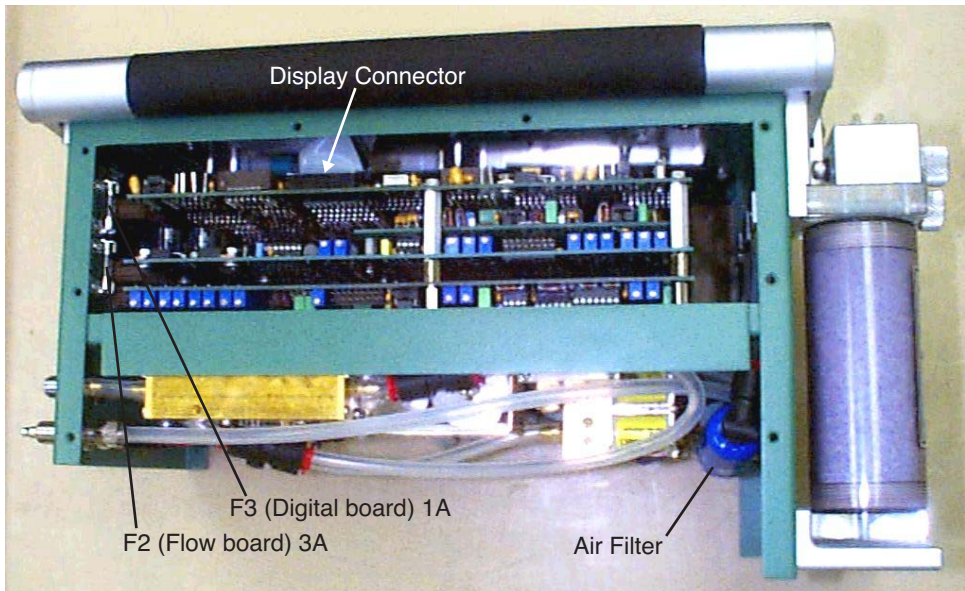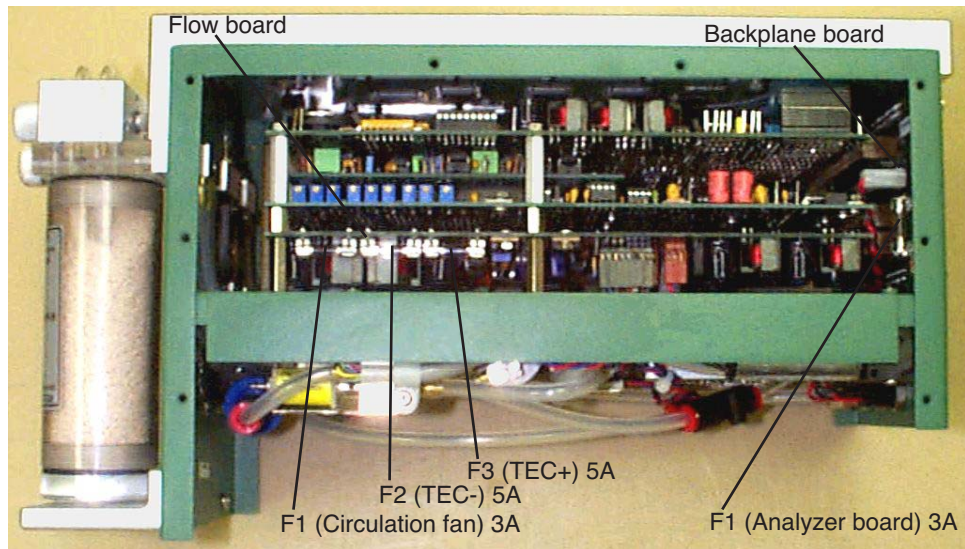
Figure 19-9. Location of console fuses. The backplane board fuses protect the analyzer board, flow board, and digital boards. The flow board fuses protect the thermoelectric coolers and fan.

Removing the PC Boards

This is certainly not part of normal maintenance, but we include these instructions here for reference purposes, should you ever have to perform this operation as directed by some sadistic LI-COR technician.

1   Power the LI-6400 off, and disconnect all cables and batteries

2   Remove the console bottom shell
See **Opening the Console** on page 19-10.

3   Unplug the display and keyboard connectors
Unplug the keyboard connector by carefully lifting it up from the pins on which it sits (A in Figure 19-10). Then, unplug the display connector on the other side of the console (B). Finally, if there is one, unplug the back light connector (C).
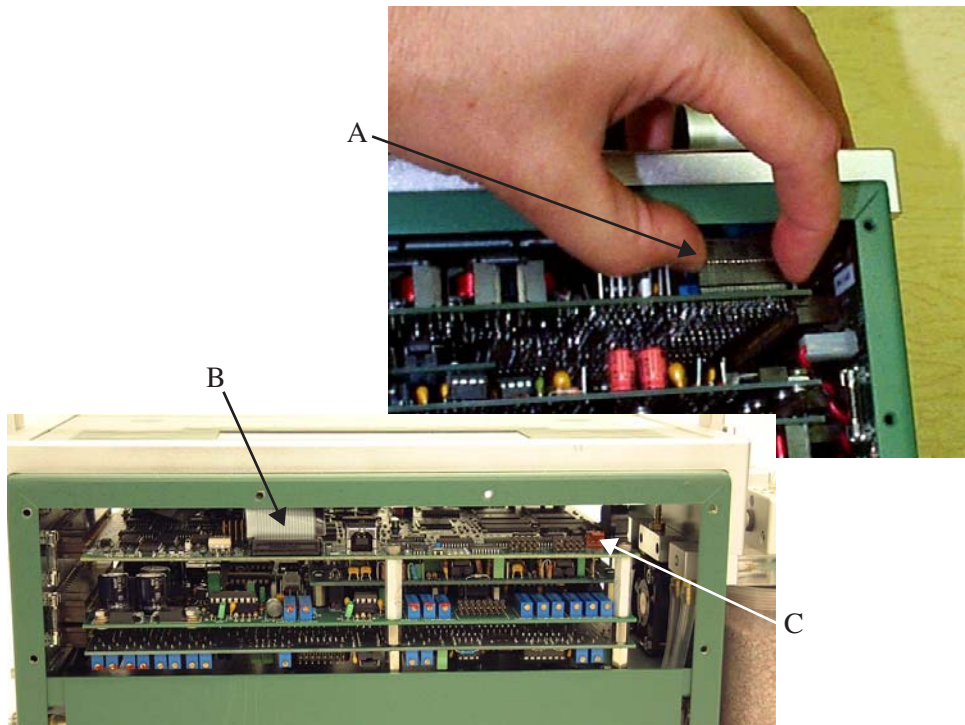
Figure 19-10. The keyboard (A), display (B), and backlight (C) connectors.

4 **Turn the console upside down and disconnect all flow board connectors**
All of the connectors in question go through a square hole to the flow board. (Figure 19-11).

Close-up of the 5 connectors to be unplugged.
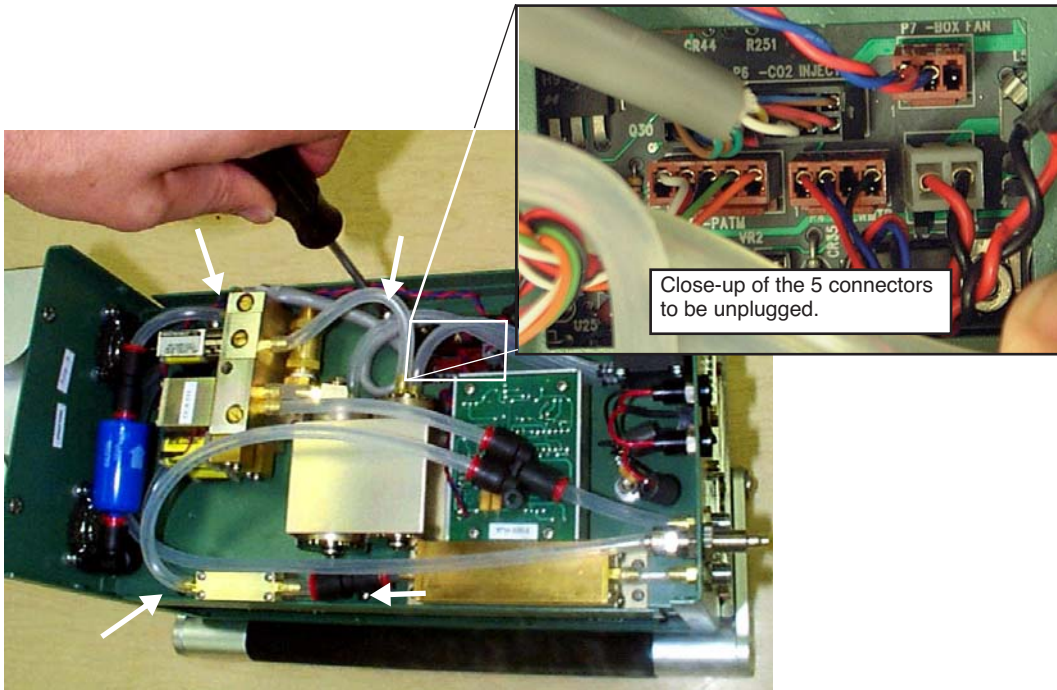
Figure 19-11. The underside of the console, showing the connector well from which all connectors should be unplugged, and the four screws that should be removed that hold the board assembly in place.

5 **Remove the board assembly retaining screws**
There are four shown by the white arrows in Figure 19-11.

6 **Unplug the board assembly**
All of the boards in the assembly connect to the back plane board at the right end of the console. Unplug these boards by hooking your fingers on the stand-offs on the back of the assembly and pulling (and rocking back and forth gently) the assembly out of the connectors (Figure 19-12).

Pull here
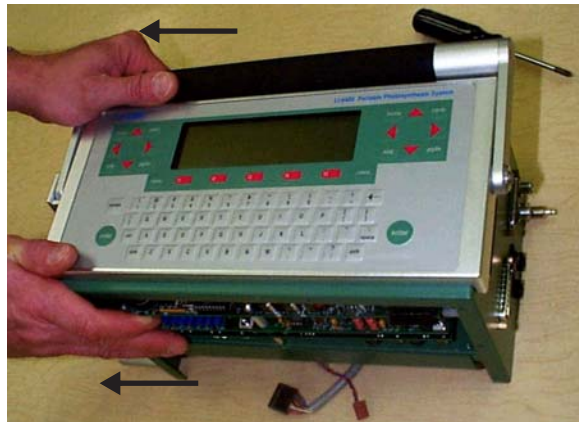
Back plane board

Figure 19-12. Unplug the board assembly from the back plane board by pulling (and rocking side to side gently) on the board standoffs.

7 **Slide the board assembly out of the console**
Be careful that the display and keyboard cable and connectors don't snag on the boards as you slide them out of the console.

Keyboard
Connector

Clock
Battery

Display
Connector

Backlight
Connector
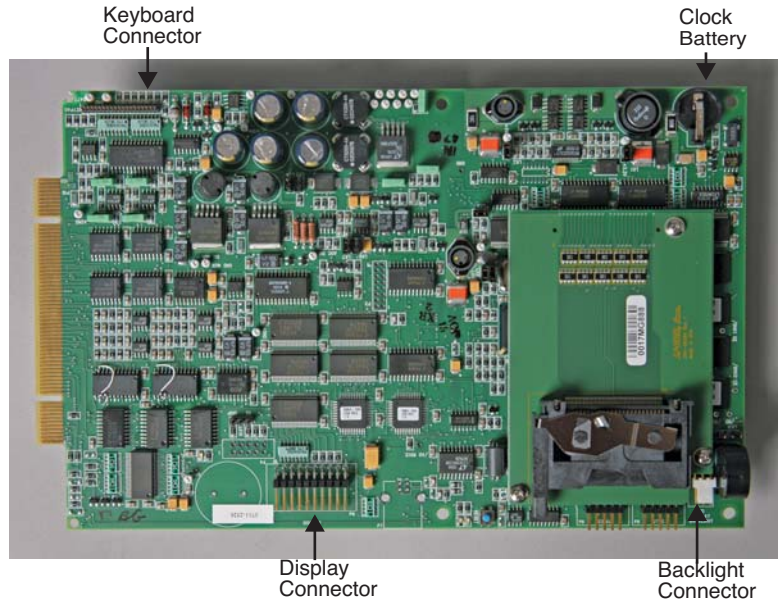
Figure 19-13. The LI-6400XT digital board.

Re-installing the PC Boards

1 **Slide the board assembly back into the console**
Be careful not to snag the keyboard and display connectors and cables.

2 **Re-connect the keyboard, display, and backlight (if present) connectors**
Make sure the connectors are centered, especially the display connector
(Figure 19-14). If you are off a pin to the left or right, you can do damage
when you turn on the power.

Note the alignment lines.
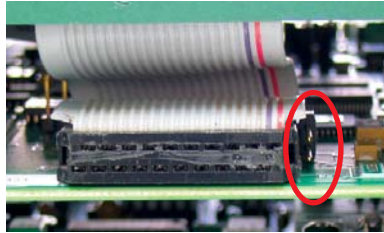
Too Far Left. **Powering on will destroy a part on the board**..
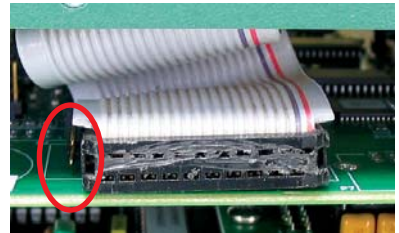
Too Far Right. Doesn't do damage, but it also won't work.
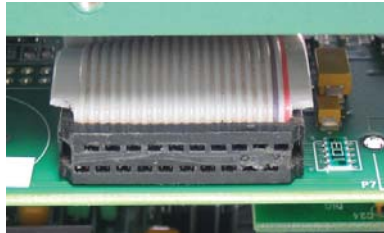
Correctly installed
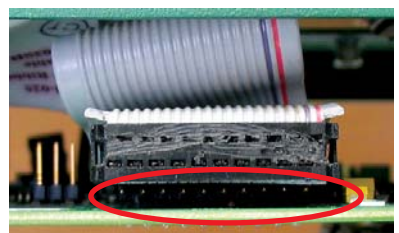
Too high

Figure 19-14. Be especially careful to get the display connector centered. If you don't, powering the system might cause damage.

3   Re-seat the board assembly
Line up the edge connectors of the three boards, and push them into the back plane board by pushing on the standoffs between the boards.

4   Install the retaining screws
Turn the console over, and re-install the four screws. The board assembly has to be fully seated before the screw holes line up.

5   Reconnect the flow board connectors
Refer to Figure 19-15. Note the labels on the flow board by each connector.
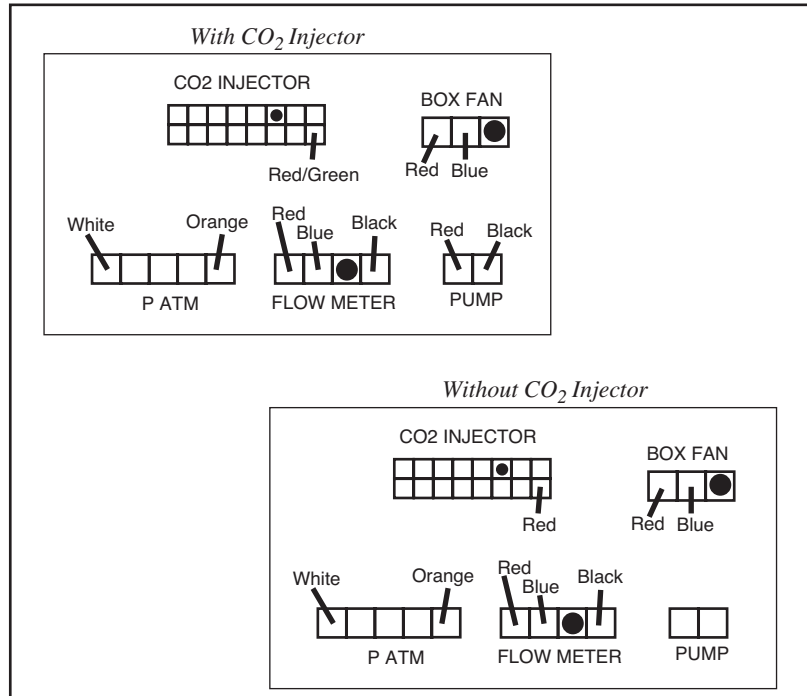
Figure 19-15. Diagram for plugging in connectors to the flow board. On instruments that do not have a CO_2 injector, the pump connector is left empty. The pump plugs in to the CO_2 injector connector, instead.

Real Time Clock

The real time clock is powered from the main battery while the console is turned on. When the console is powered off, the clock is run from a 3V lithium battery. The battery should operate for many years.

When the voltage on the battery drops to 2.7V, the battery (LI-COR part number 442-03791) should be replaced. See **Real Time Clock Problems** on page 20-5 for instructions about measuring and replacing the battery.

Cables

Insulation

We have found that exposure to UV radiation can sometimes cause shrinkage in the cable insulation. This manifests itself in a tendency for a cable to develop coils, and/or to pull away from the back of the connector. If you detect this happening, contact LI-COR.

NOTE: In the Spring of 2000 we changed to a different type of cable that should be immune to this type of problem. The new cables are black, and the older cables are gray.

Replacing Connector Screws

Three of the connectors in the LI-6400's cable assembly have screws to hold them in place. Do not overtighten these screws, or they will break. If they do break, they are easily replaced in the connector shroud (Figure 19-16); the end that breaks off in the mating connector may be difficult to remove.

Part Number
314-04913
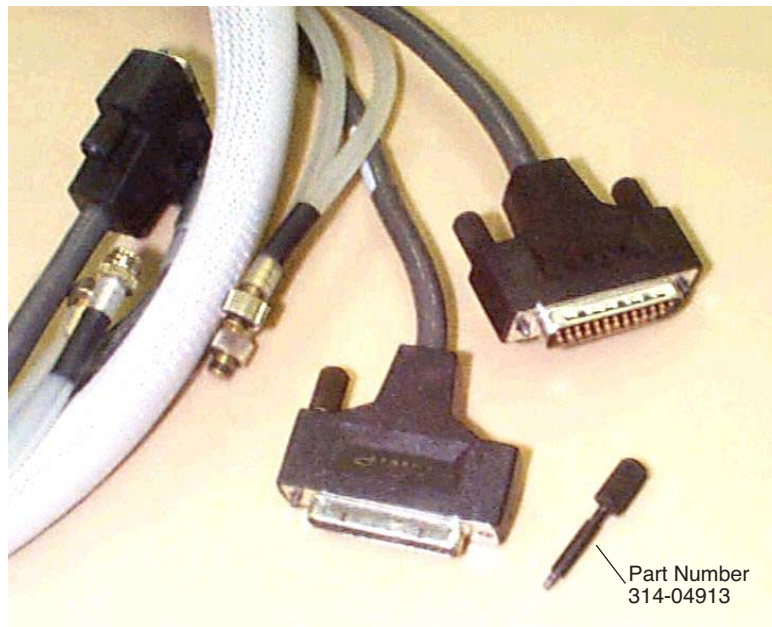
Figure 19-16. A connector screw can be removed by firmly pulling it straight out. It is held in place by friction.

The Chamber Handle

Handle Maintenance

Handle maintenance is simple. The handle is held on with two screws (or three, depending upon the instrument's vintage). If they get loose, tighten them (Figure 19-17).

Handle Screws
122-02284

Tripod Bracket Screws
122-04276
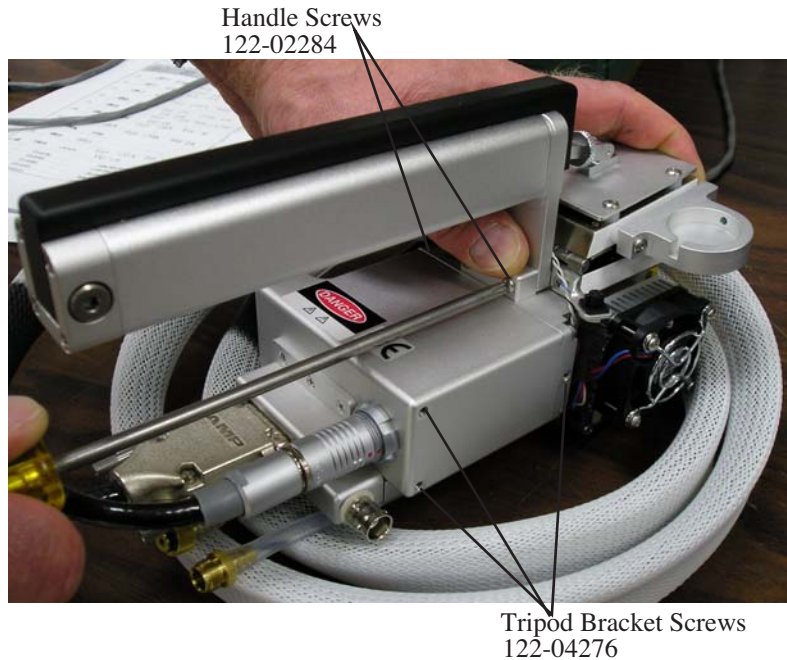
Figure 19-17. Keep the handle screws snug. Older units have three screws.

Latch Maintenance

The key to the latch mechanism is the chamber catch rod, that little wire shown in Figure 19-18. There are a couple of latching problems that this rod can cause:

• **Chamber doesn't latch reliably**
   This rod must have a 90 degree bend at the top, or the chamber will not latch correctly. If it becomes straightened, reach in with a pair of needle nosed pliers and re-bend it.

- **Chamber doesn't unlatch reliably**
  If the rod isn't far enough to the left (Figure 19-18 photo) when the chamber is open, then you might have trouble getting it to unlatch. Hint: If you have trouble unlatching a chamber, just push the black handle piece to the left (looking down on it from the rear) as you squeeze the handle.

Chamber catch rod assembly
Part Number 6400-903

Front view, installed
Side view, exploded

6564-057
(wire only)

90° Bend
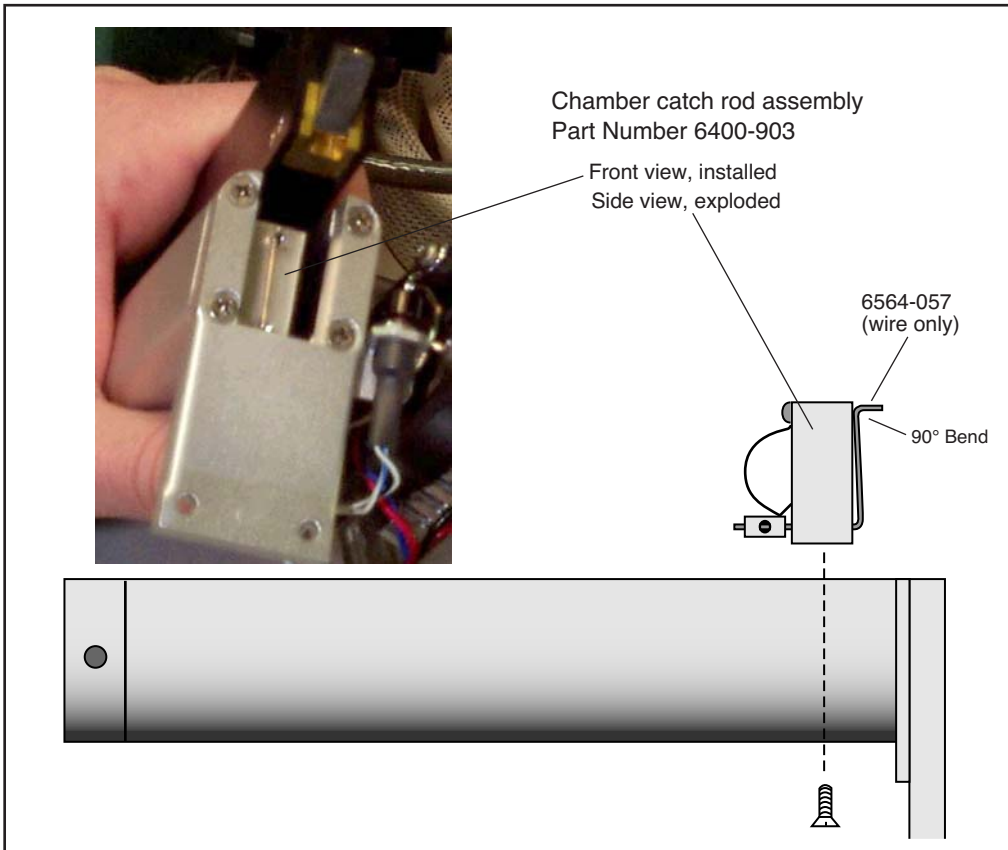
Figure 19-18. The chamber catch rod should have a 90 degree bend at the top. If it doesn't, the chamber will not latch correctly. The catch rod can be re-bent with a pair of needle nosed pliers, if necessary. Should the wire break, you can replace it (6564-057) or replace the entire assembly (6400-903).

Latch Return Spring

The latch return spring is shown in Figure 19-19. It should never need any service, and the only reason you might have to deal with it is if the spring should fall out of place when the handle is disconnected
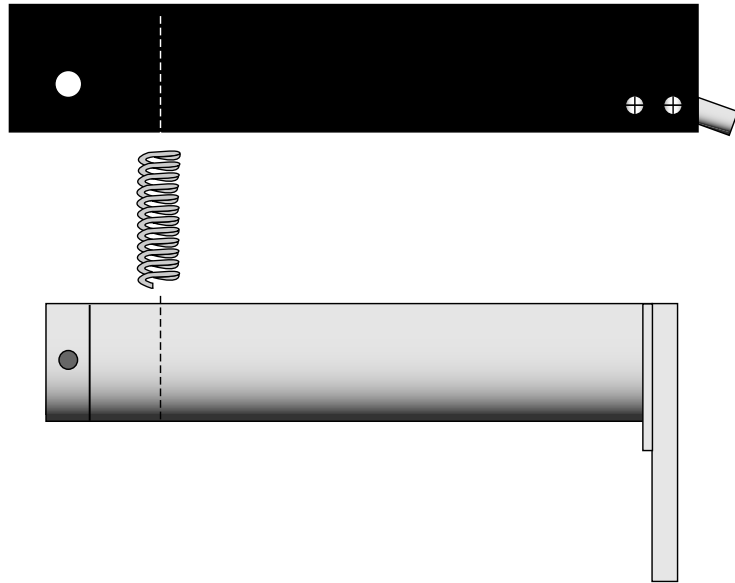
Figure 19-19. The latch return spring is in the rear of the handle. To access, remove the bolt in the rear that serves as a hinge.

Handle Removal

The handle must be removed for certain service operations, and for installing certain chambers (e.g. the 6400-09 Soil Chamber and the 6400-05 Conifer Chamber).

■ **To remove the chamber handle assembly**

1 **Detach the chamber top from the handle**
Open the chamber, and turn the adjustment nut until it is free of the handle
(Figure 19-20).
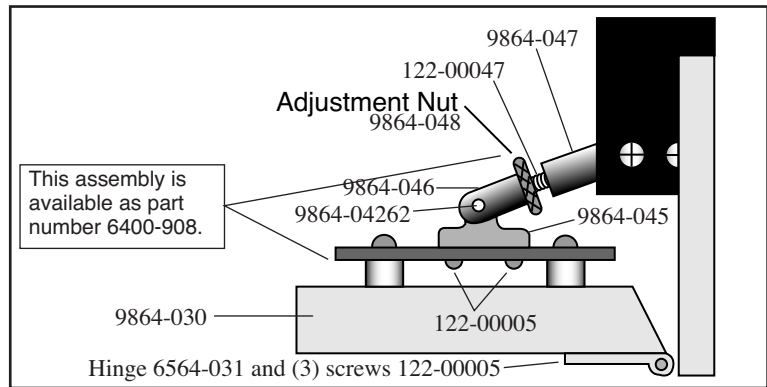
Figure 19-20. Unscrew leaf chamber adjustment nut.

2 **Remove the handle screws**
The screws are located on the back side of the handle, as shown in
Figure 19-21. Use a #1 Phillips head screwdriver to remove them. Note that
the middle screw (if there is one) is shorter than the other two. Allow the han-
dle to rest at the side of the sensor head with the log button attached.
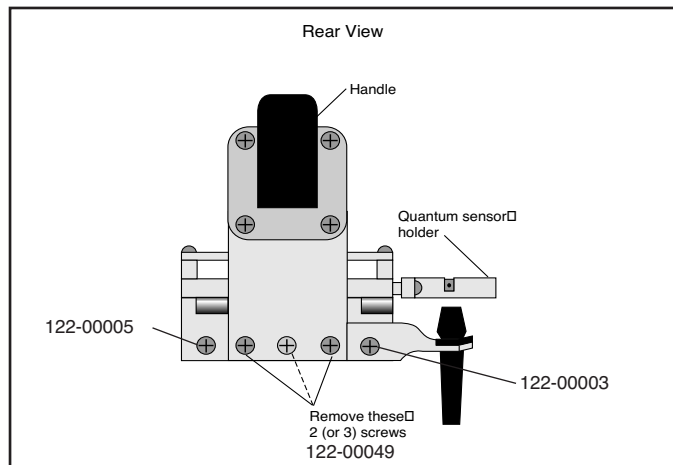
Figure 19-21. Remove the indicated screws.

Leaf Temperature Thermocouple

Thermocouple Maintenance

The thermocouple circuit should be zeroed periodically. The procedure is described in **Zeroing the Leaf Temperature Thermocouple** on page 18-24.

Thermocouple Replacement

The leaf temperature thermocouple is mounted in a plastic holder that is inserted from below the bottom half of the leaf chamber (Figure 19-22). The thermocouple is terminated with a male thermocouple connector. This entire assembly is replaced. If the thermocouple junction is broken, leaf temperature will read the same as block temperature[1].
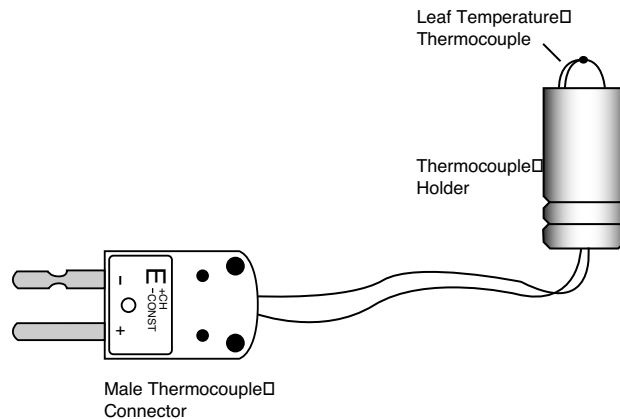
Figure 19-22. Leaf temperature thermocouple and connector.

■ **To replace the 6400-04 leaf temperature thermocouple:**

1 **Unplug the connector**
Remove the male thermocouple connector by pulling straight out.

_____

[1]·Provided the leaf temperature is properly zeroed.

2   **Pull out the holder**

The holder can be removed by pulling straight down and out of the leaf chamber. You may have to twist it a bit, if it is tight. *Do not pull on the thermocouple wires*.

3   **Insert the new holder**

Moisten the thermocouple retainer O-ring slightly, or use a *minuscule* amount of silicon grease. This will make it easier to install the new thermocouple.

4   **Plug in the connector**

Connect the new assembly by re-inserting the male thermocouple connector, and carefully insert the plastic thermocouple holder up through the bottom of the leaf chamber. Do not pinch the thermocouple wires when inserting the holder.

5   **Position the holder**

Insert the holder until the thermocouple bead extends just above the lower foam gasket, when viewed from the side (Figure 19-23). This will ensure that the leaf is in contact with the thermocouple when the chamber is closed.

If you are using an energy balance to compute leaf temperature, then position the thermocouple lower, so that it will not touch any leaf material.
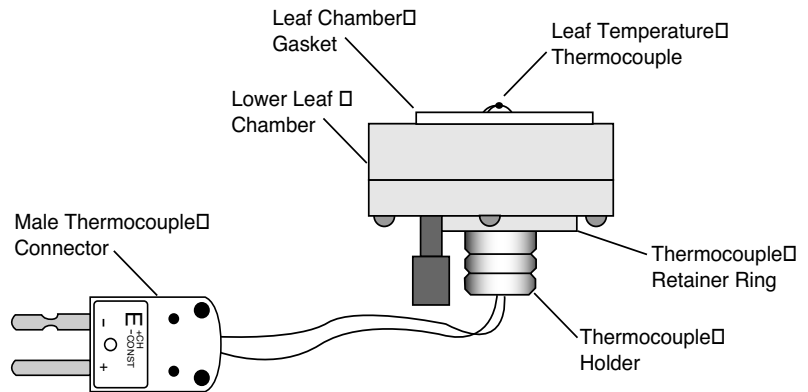
Figure 19-23. Position thermocouple bead just above foam gasket.

Leaf Chambers

Foam Gasket Care

It is important to take care of the foam gaskets on the leaf chambers. *Never latch the chamber closed when it is not in use*, as the gaskets will stay compressed if you leave the chamber closed for several hours. Black neoprene will recover from this condition overnight, but the white gaskets used with the LED light sources will not.

Foam Gasket Replacement

The two gaskets on the leaf chamber and the air seal gasket located behind the chamber gaskets should all be replaced as needed. It is important that you peel off the old gaskets correctly. There is a thin film below the gasket foam called the *carrier* that holds the adhesive. With a fingernail or the flat edge of a knife, try to pry up a corner of the carrier. If you only pry up the foam and adhesive, the carrier will remain on the chamber, and is difficult to scrape off. If you get underneath the carrier, the whole gasket will come off quite readily.

Frequently there is residual adhesive material on the chamber after the gaskets are removed. This can be cleaned up with acetone or other solvents, *if* it is a painted metal chamber part. Never use solvents on the plastic 6400-05 Conifer Chamber. **Note:** A product that we've recently come across that works very well for cleaning up gasket adhesive is Oil-Flo™ , manufactured by Titan Chemical, inc. (1240 Mountain View-Alviso Rd., Sunnyvale, CA 94089, 408-734-2200). This is a water soluble, solvent-based "safety degreaser". It even worked safely on the 6400-05 conifer chamber (acrylic), but marred an old 6200 leaf chamber (polycarbonate), so it would be wise to follow the label's admonition to "test on plastics" before use.

The replacement gaskets are installed by removing the paper backing over the adhesive. Watch for channels where the adhesive sticks to itself when you remove the backing. If you stretch the gasket slightly, the channels disappear. If you apply the gasket to the chamber with channels, you'll have leaks.

For part numbers for replacement gaskets, see the table **Items for Chambers** on page 19-43.

Propafilm® Replacement

Several chambers use Propafilm® (ICI Americas, Inc., Wilmington, DE), such as the standard 2x3 chamber top, the 6400-07 top and bottom, the 6400-08 bottom, the 6400-11 top, and the 6400-17 top. You will need to re-

place the film if it becomes torn or punctured, or excessively dirty. Replacement film (LI-COR part #250-01885) and double-sided tape (part #212-04341) can be found in the spare parts kit.

■ **To replace the Propafilm:**

1 Detach the affected chamber part
Use the 3/32" hex key provided in the spare parts kit to remove the two long screws, or (for the 6400-17) a #2 Phillips screw driver to remove four screws.

2 Remove the old Propafilm and tape
The tape has a fairly strong adhesive; if it does not peel off readily, use a mild solvent (e.g. acetone) to help dissolve the adhesive. Do not use a blade or other sharp instrument to remove the tape, as you could damage the surface of the chamber, thereby making it difficult to achieve a air-tight seal with the new Propafilm.

3 Prepare the new tape
Cut a strip of the double-sided tape that is slightly longer than the leaf chamber. Lay on a flat cutting surface, adhesive side up. The tape may be curled slightly; hold the corners down with cellophane tape, if necessary.

4 Attach the tape and trim
Lower the top surface of the chamber onto the tape and press firmly. Trim the tape around the outer and inner edges of the chamber (Figure 19-24). To get a clean cut, use a fresh blade (an Exacto knife works well) and make sure your first cut as close to the edge as possible.
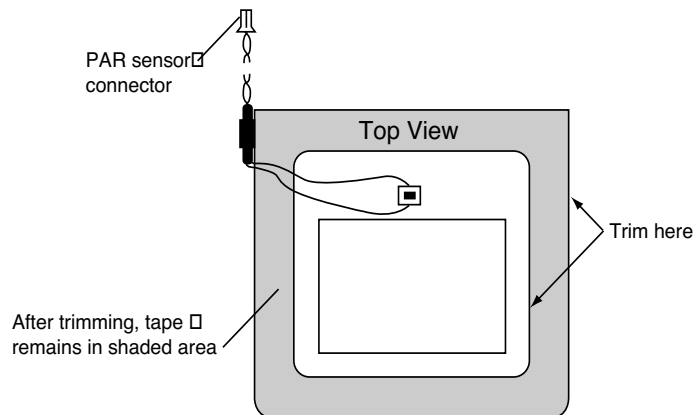
Figure 19-24. Trim tape at leaf chamber edges

5   Prepare a piece of Propfilm
Cut an oversized piece of Propafilm. Stretch the film on a flat, clean surface until taut. You may want to tape the corners with cellophane tape to secure it to the cutting surface.

6   Remove the backing
Peel the backing from the tape and smooth any bubbles that may have formed.

7   Attach the Propafilm
Lay the leaf chamber onto the Propafilm. Turn the chamber over and smooth the film. Bubbles can be lanced and smoothed.

8   Trim to size
Trim the film around the outer edge of the chamber. Make your first cut is as close to the edge as possible.

9   Reassemble the leaf chamber assembly.

Fluorescence Chamber Tops
The 6400-06, -10, and -14 chambers have holders for fluorescence probes built into the chamber tops. The top piece is plastic, lined on the inside with Teflon. If the inner surface is scratched, the Teflon can tear, leaving a shadow-causing blemish.

The chamber tops can be removed by removing the screws that hold it in place. There is a layer of silicone cement underneath the chamber top, but it won't stick to the top because of its Teflon lining.

Self adhesive Teflon is available from LI-COR as part number 212-02314.

Chamber Mixing Fan
See **Noise caused by the Fan Motor** on page 20-40 for a discussion of how to determine if the fan motor is about to fail.

LED Source Maintenance
The most important maintenance item for the LED source is the foam gaskets, concerning which there is one thing to remember:

> Use white polyethylene gaskets on the LED source.

(You can still use black neoprene gaskets on the bottom chamber.) The reason is that the white gaskets helps unify the light distribution across the surface of the leaf. Black gaskets significantly reduce the quantum flux hitting the leaf near the edges.

The white polyethylene gaskets do not recover nearly as well as black neo-prene after being compressed, and they are much more of a pain to remove. Therefore, to avoid changing them anymore often than necessary:

> Don't leave the chamber latched closed (with the gaskets compressed) any more than is absolutely necessary.
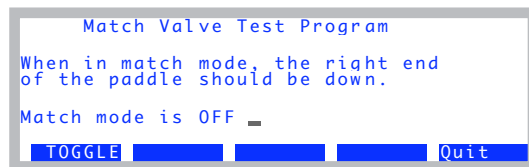
Match Valve Maintenance

Unsticking the Match Valve

The match valve pads are coated with molybdenum disulfide, a grey powdery dry lubricant, used to prevent the pads from sticking. These pads might have a tendency to stick down (usually after a period of storage) and prevent the valve from moving. There are a few courses of action to take, should this happen to you.

Exercise

Home Menu | Tests & Diagnostics | Match Valve Tester is a program that allows you to "manually" (via the function keys) control the match valve.

```
          Match Valve Test Program

When in match mode, the right end
of the paddle should be down.

Match mode is OFF _

    TOGGLE                              Quit
```

Figure 19-25. The Match Valve manual control program. The status line indicates where the valve should be, not necessarily where it actually is.

Often a stuck valve can be freed by cycling the valve several times. Note that when one exits this program, the match valve is returned to the position it was in when the program was launched.

Orthoscopic Surgery

If the "Match Valve Tester" solution fails, there is a direct approach, requiring a thin, stiff piece of wire. Disconnect the IRGA tubing, and insert the wire up the appropriate tube (usually the reference tube) and through the hose barb to pop the match valve free. Subsequent exercise using the "Match Valve Tester" program is recommended.

Pad stuck down

The right end of the paddle should be down when Match is ON.

Insert wire here
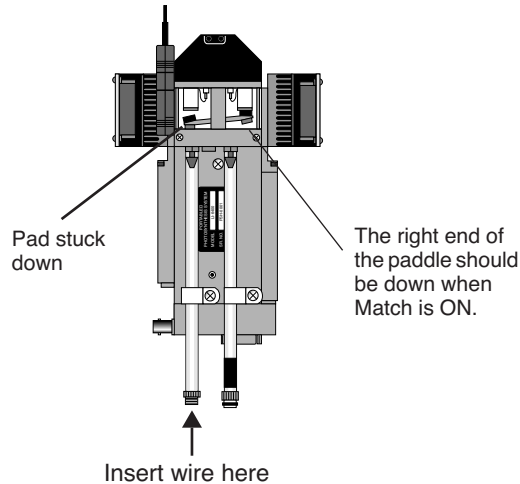
Figure 19-26. If a pad sticks preventing the match valve from operating, insert a thin wire as shown and pop the pad free.

Open Heart Surgery

If all else fails, or if the problem persists, then perhaps a cleaning and lubrication is warranted. To do this, lay the sensor head on its back so it is level.

1 **Disassemble**

Remove the four screws that hold the black cover in place, and lift off the cover. Lift the match valve housing up from the paddle end, exposing the pads. *Do not remove the plastic cover.*
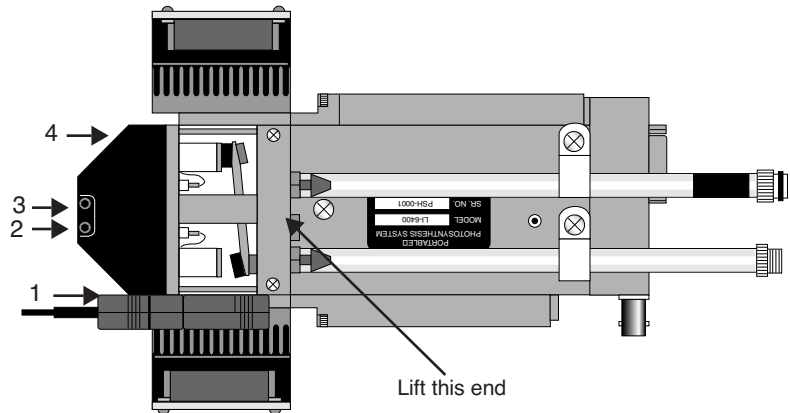
Figure 19-27. Remove the match valve cover by removing four screws, and lifting the cover from the indicated end.

2   Clean the pads

Clean the surface of each pad with alcohol, then apply the molybdenum lubricant[2]. Rub the powder into the surface of the pad with the side of a toothpick. Tap off the excess powder so it will not enter the system plumbing.

3   Clean the holes

Clean the edges of the holes where the pads make contact.

4   Reassemble

Replace the match valve, and the black cover. *Make sure that the two wires under the black cover are not pinched*. Do this by sliding the cover back and forth a bit to see if you can feel the wires between the cover and the block.

[2.]Available from Dow Corning under the trade name MolyKote Z, part number 88050-21, or (on request) from LI-COR.

IRGA Maintenance

Chemical Bottles

There are two small plastic bottles in the analyzer housing designed to keep the detectors free of CO_2 and water vapor. They contain a mixture of anhydrous magnesium perchlorate and Ascarite II that should last for three or more years.

Note: Instruments built or serviced at LI-COR prior to about January 2003 will have soda lime instead of Ascarite for the CO_2 scrubber. That combination needs to be changed annually, since the magnesium perchlorate desiccates the soda lime, rendering it ineffective, ultimately resulting in calibration drift and instability.

■ **To change the sensor head soda lime/desiccant bottles:**

1 **Open the covers, remove bottles**

The plastic bottles are located in the analyzer housing in the sensor head. They are accessed by removing the cover plate on the left side of the analyzer housing (Figure 19-28). There are two bottle covers, each with an O-ring seal, beneath the housing cover (Figure 19-30). Pull up on the bottle covers to expose the bottles; the O-rings can expand to form a very tight seal, so you may have to pull hard.
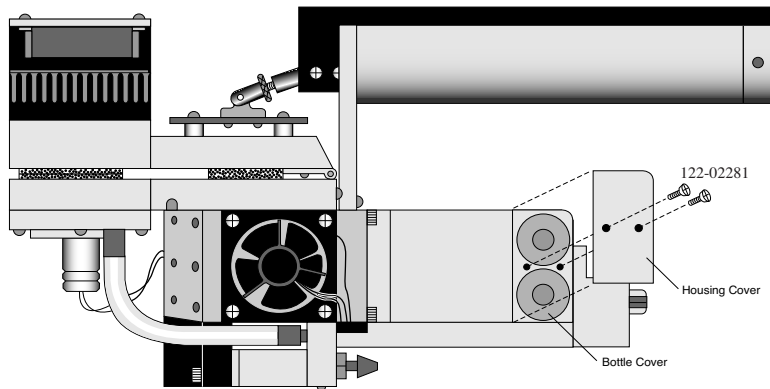
Figure 19-28. Remove the two screws that secure the housing cover.

2   Prepare new bottles
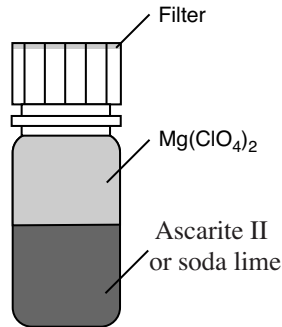
Figure 19-29. Fill the bottles with the CO$_2$ scrubber, followed by magnesium perchlorate.

Part number 6400-950 consists of two internal scrub bottles already filled with magnesium perchlorate and Ascarite II.

If you do not have a 6400-950, you will need to prepare the chemicals yourself. Fill the two bottles with equal parts Ascarite II (or <u>dry</u> soda lime - *do not use the 9964-090 wet soda lime)* and magnesium perchlorate. Fill the bottles half full with Ascarite, followed by the magnesium perchlorate. Place a filter paper disk in the lid to keep the chemicals from spilling into the detector housing.

3   Insert the new bottles
Insert them *lid first* into the analyzer housing. Seat the bottle cover with the attached O-ring, and secure the housing cover with the two screws.

4   Wait before use
When the bottles are changed, allow one day for the detector to equilibrate again.

Magnesium perchlorate is the recommended desiccant. **Do not use any other desiccant**. Several grades of magnesium perchlorate are available from com-
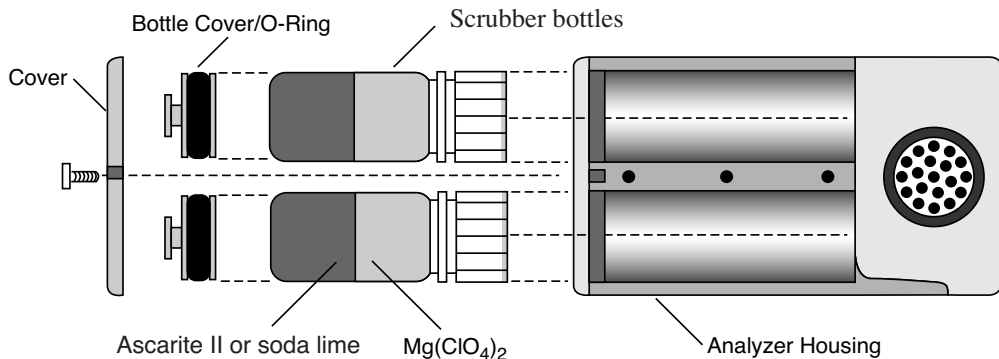
Figure 19-30. Insert the new bottles lid first, followed by the bottle cover/O-ring and housing cover.

mercial suppliers. One type that works well is marketed under the name *De-hydrite*, and is available (catalog number C260-M61) from Arthur Thomas Company, Vine St. & 3rd, Philadelphia, PA 19105. (215)574-4500.

> **CAUTION!** Magnesium perchlorate is a strong oxidizing agent. Contact with skin or mucus membranes may cause irritation. Avoid bringing it into contact with acids and organic substances such as cotton, rubber, grain dust, etc. Consult the container label.

Cleaning the Optical Bench

Because the LI-6400 contains open path analyzers, it is possible for air-borne debris to enter the system and contaminate the sample optical path.

When the analyzers become too dirty, the "IRGA(s) not ready" message will be displayed in New Measurements mode. There are other causes of this message, however (see page 20-15).

Cleaning the Mirrors

The easiest way to open up the optical path for partial cleaning is to remove the mirrors.

■ **To remove and clean the mirrors**

1 Remove the bottom chamber
Remove the chamber bottom by disconnecting the thermocouple, exhaust tubing, and removing the two screws that hold the chamber in place.

2 Remove the six screws from each gold mirror
Use a 5/64 hex key. Be careful. They are small screws.

3 Clean the mirrors
Wash with ethanol or water, and wipe dry. Use a cotton cloth (e.g. an old Tee shirt) on the mirror surface. Do <u>not</u> use a cotton swab, as they tend to scratch.

Hint: A mirror that seems clean when you are looking straight at it may have residue that appears when viewed at an oblique angle. Look at the mirror from several angles before you deem it "clean".

4 Look inside
Check for debris that may be inside the cell. You could blow out the cell with clean, pressurized air. If you are tempted to reach in with a Q-tip or cotton

swab to clean the windows at the back of the optical bench, be careful not to snag the cotton on the sharp corners near the front. If you do get fuzz snagged there, it will blow in the wind when the chamber fan is running, and you'll have increased signal noise from the IRGAs.

5   Reassemble
Be careful when re-installing the mirrors. The screws are small, and excess force will break them, creating a major problem.

Opening the Optical Bench
If access via the mirrors is inadequate, then open the whole thing up:

■   **To disassemble the sensor head and clean the optical path:**

1   Remove the handle from the sensor head.
Described in **Handle Removal** on page 19-22.

2   Remove the upper half of the leaf chamber.
Remove the two screws from the hinge on the rear of the upper half of the leaf chamber (Figure 19-31). The upper portion of the leaf chamber can now be moved aside; unhook the connector from the PAR sensor or LED Light Source, if necessary.
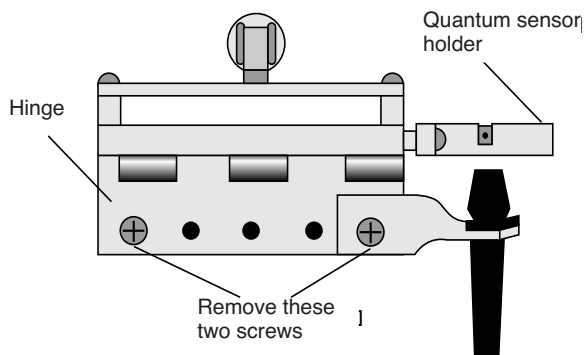
Figure 19-31. Remove the screws shown.

3   Remove the optical bench cover.
Pull off the air hose from the underside of the leaf chamber.

Remove the male end of the thermocouple connector by pulling straight out.

There are eight hex head cap screws on the optical bench cover, as shown in Figure 19-32. Remove the cap screws with a 5/64" hex key (in the spares kit). The cover with attached lower half of the leaf chamber can now be removed.

Top View

Gasket

Lower leaf chamber cuvette

Sensor head manifold 9864-033

Hex head screws (8) 140-04234
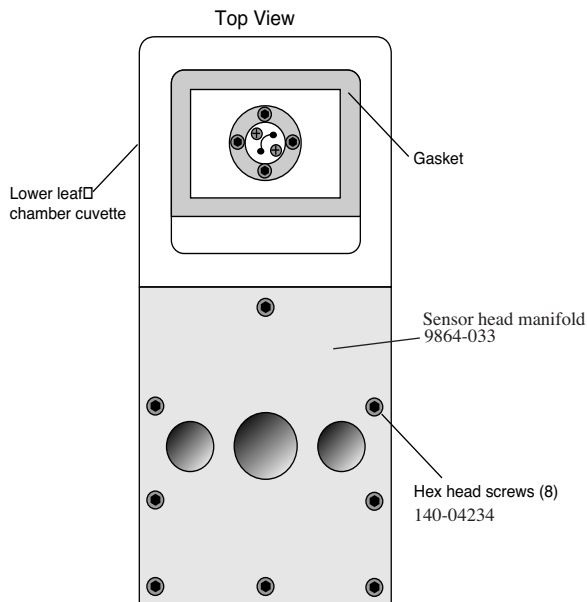
Figure 19-32. Remove the optical bench cover (leave the chamber bottom attached) by removing the 8 screws in the manifold.

4   Clean the windows

Moisten a cotton swab and swab the two optical windows for the sample cell (Figure 19-33).
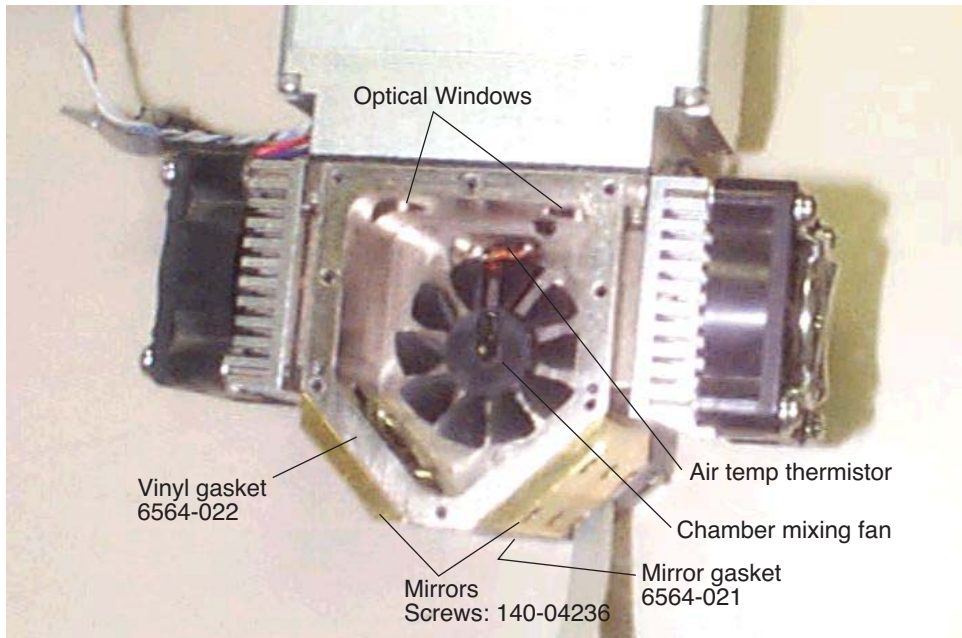
Figure 19-33. Clean the two optical windows.

If you haven't done so already, you can also remove and clean the two gold plated optical mirrors by unscrewing the six cap screws on each mirror. See comments on Step 3 on page 19-34.

Allow the mirrors to air dry before reassembling the sensor head.

5   Reassemble the sensor head.
Note that there is a thin vinyl gasket on the top surface of the optical bench (Figure 19-33). This gasket is reusable; it should adhere to the optical bench. If it becomes detached, be sure to reposition it before reassembly. Tighten (but not overly so - they're small and can break) the eight screws evenly.

Note that one of those screws (nearest the label "Mirrors") in Figure 19-33 goes through an air channel in the optical bench cover. If this screw is not snug, there will be an air leak.

Servicing the External CO$_2$ Source Assembly

Oil Filter Replacement

Inside of every CO$_2$ cartridge is a residue of oil. When the CO$_2$ cylinder is pierced, some of this oil is released along with the CO$_2$. There is a filter attached to the regulator to prevent oil from clogging the flow restrictor. After using 25 CO$_2$ cylinders, the oil filter should be replaced. Instructions are given below. Note: Some brands of cylinders (notably CopperHead™ and Curtis™) contain considerable oil, requiring a filter change every cartridge. It is therefore good policy to check the filter (remove cap and look for discoloration on the filter end) every time you replace a cylinder when using non-LI-COR cylinders.

To install the filter:

1 **Remove the CO$_2$ cylinder cover.**

Warning: Before replacing the filter, the CO$_2$ cylinder cover must be removed in order to depressurize the 6400-01 CO$_2$ Injector. If you attempt to remove the filter cap before the cylinder is exhausted, high pressure CO$_2$ will blow the filter out of its holder.

CO$_2$ Cylinder☐
Cover

192-01919

Flow
Restrictor

Filter Cap

Figure 19-34. Location of oil filter cap.

2 **Remove the filter cap**
After depressurizing the CO$_2$ cylinder, remove the filter cap to reveal the filter (Figure 19-34).

3 **Remove the old filter**
Use the filter hook included to remove the old filter, being careful not to scratch the O-ring seat. (Hint: pointed tweezers work better.)

4 **Prepare the new filter**
Remove the paper from around the new filter. Roll the filter between your thumb and index finger to smooth and compress it to a diameter that just fits into the body of the "T" fitting on the regulator.

5 **Install the new filter**
Insert the filter and push it into the body. Do not use the filter cap to push the filter into the body because some of the filter fibers can become tangled in the O-ring seal and cause leaks.

6 **Reconnect the cap**

> **Rolling Your Own…**
> The oil filters are simply cigarette filters that can be cut from any unused filtered cigarette. When cutting the filter, use a razor blade to cut it 2 cm (0.75 inch) long. Then slit and remove the paper, and insert the filter as described above.

Getting To Know Your Mixer

It is a good idea to keep a record of the dynamic response of your mixer. Specifically, you should monitor how long it takes to go from 100 to 2000 μmol mol^{-1}. With use, oil may get past the filter and begin to clog the flow restrictor in the source. If this happens, the time it takes to make this 100 to 2000 μmol mol^{-1} transition will increase (Eventually, you won't even be able to get to 2000 μmol mol^{-1}). By measuring this transit time periodically, you can get an indication of coming problems before they can actually occur, and can avoid them entirely by servicing the source as described in the next section. Example: a new source (clean restrictor) might make this 100 to 2000 transition in 2 minutes. If after a while this time degrades to 3 minutes, then replacing the flow restrictor would be in order. This is covered in the next section.

If the Flow Restrictor Becomes Clogged

If the oil filters are not replaced on a regular basis, oil from the CO_2 cartridges can enter the copper supply tube and clog the flow restrictor. If you are unable to attain desired CO_2 concentrations while operating the CO_2 injector, and you are using a fresh CO_2 cartridge and oil filter, you may have a clogged flow restrictor. If oil makes it through the flow restrictor and enters the console, it will require factory service. Therefore, it will behoove you to keep close watch on your filters, to prevent any of this from happening.

The flow restrictor must be replaced if it becomes clogged. The restrictor is pressed into the fitting connected to the top of the copper supply tube (Figure 19-35), and cannot be removed; you must replace the entire fitting. A spare fitting with the restrictor in place (part number 9964-042) can be found in the 6400-01 spare parts kit.

■ **To replace the flow restrictor and clean the copper supply tube:**

1 **Loosen two nuts, remove supply tube**
One is at the base of the "T" fitting (Figure 19-35), and the other is at the top of the copper supply tube.
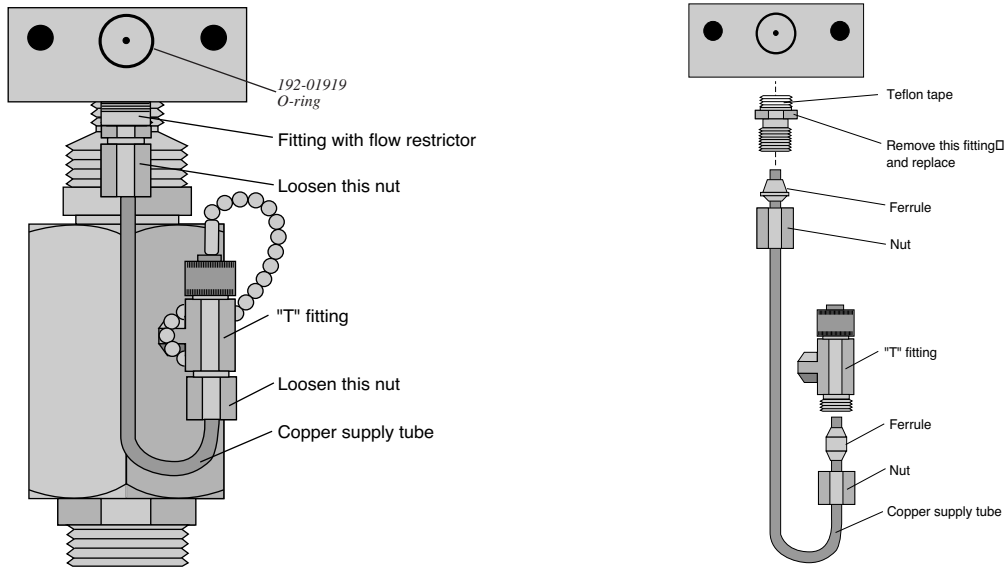
Figure 19-35. Remove the two nuts at each end of the copper supply tube.

2 **Clean the supply tube**
To remove any oil that may be present in the copper supply tube, flush thoroughly with an organic solvent (e.g. acetone) that leaves no residue. If that's not possible, use hot, soapy water (dishwashing soap works well), and rinse thoroughly with clean water.

Blow out the tube to ensure no droplets are left inside. If any droplets get up into the flow restrictor, it will become clogged.

3 **Replace the flow restrictor**
Remove and discard the fitting containing the flow restrictor. Install the new fitting/restrictor (LI-COR part #9964-042). Note that the fitting is wrapped at one end with Teflon tape (Figure 19-35). Insert this end into the mounting block and tighten securely.

4   **Install the copper supply tube**
Tighten the two nuts.

Shipping The LI-6400

If you need to transport the LI-6400 in its shipping case, we recommend that you observe the following tips:

- **Tie the chamber handle closed.**
  Close the leaf chamber and use a piece of tape or a cable tie to prevent the spring-loaded handle from opening. Failure to do so can result in bending or breaking the chamber catch rod (Figure 19-18 on page 19-21) inside the handle. This is especially true when an LED light source is attached to the chamber.

  Be sure to set the latch adjustment knob so the foam gaskets are not compressed.

- **Console in the case**
  The console goes into the case so that a) the display is up facing the handle side of the case, and b) the console handle is on the fat side of the case. This helps in closing the lid, and puts the console in the most protected orientation.

  If shipping to LI-COR for repair or recalibration

- **Call for an RMA (Returned Material Authorization).**

- **Be sure to include…**
  …the console, with chemical tubes attached, the CO_2 source assembly for the mixer (if you have one), the chamber/IRGA, and the cable assembly. Also include all items that will need a light calibration (LED light source, chamber top(s), and external quantum sensor).

- **Don't bother sending…**
  …anything else (batteries, communication cables, charger, or extra chambers), unless you specifically want us to examine them.

- **Back up your data**
  We do <u>not</u> routinely save a copy of the /user disk before we start working on the instrument, nor do we delete files from a customer's /user disk. But if you send us a console with priceless data stored on it and nowhere else, then you may be in for some abnormally bad luck. So please, send us only files you can afford to lose.

Useful Part Numbers

Quantity is "1 each" unless otherwise stated.

Entire System	Part Number
Standard spare parts kit	9964-015

Items for Chemical Tubes	Part Number
Drierite desiccant (1 lb bottle)[a]	622-04299
Soda lime - wet (450g bottle)	9964-090
Ascarite II (500g bottle)	9970-022
Air flow muffler	300-03707
Small O-ring (seals air passage to console)	192-02597
Large O-ring for end caps	192-04291
Chemical tube assembly (entire)	9964-021
Threaded tube only	6564-076
Bottom end cap	6564-071
Labels for desiccant and soda lime tubes	250-04296
Tygon tubing (material for small bypass tubes)	222-00302
Flow adjust knob	9864-075
Set screw for flow adjust knob	142-00109

a.Also available from W.A.Hammond Drierite Company, Box 460, Xenia, OH 45385, part number 24001

Battery Items	Part Number
Battery	6400-03
10 Amp fuse for 6400-03 battery	438-03142
Male connector (as on the battery)	318-02031
Female connector (as on the console)	318-02030
Cable (5 ft.) and connectors for charger - console	9960-062
Cable (10ft.) and connector for external battery	9960-120

Items for Air Flow	Part Number
Balston air filter	300-01961
1/8" ID quick coupler, female	300-04269
1/8" ID quick coupler, male	300-04270
Replacement air pump	286-04198
Air pump repair kit	6400-907
Bev-a-line tubing (per foot) (1 box = 50ft)	222-01824
Lower chamber exhaust tube	6564-154
1/4" OD Quick Connect straight union	300-03123
1/4" OD Quick Connect "Y"	300-03367

Console	Part Number
Clock battery (keeps time while power off)	442-03791
3A, 250V, fast blow (5 x 20 mm)	439-04215
5A, 125V, fast blow (5 x 20 mm)	439-04214
1A, 250V, fast blow (5 x 20 mm)	439-04216
Screws (16 required) for holding bottom shell	122-00007

Items for Chambers	Part Number
4-40, 1.75" hex screw for attaching chamber	140-04251
Leaf Temperature Thermocouple	6400-04
Propafilm (10" wide, order by length)	250-01885
Double sided tape (per foot)	212-04341
Gaskets: 2x3 chamber and 3-hole (10 sets)	6400-30
Gaskets: 2x6 chamber and 3-hole (10 sets)	6400-32
Gaskets: 6400-05 Conifer chamber (5 sets)	6400-31
Gaskets: 6400-02B LED source. Includes 20 white gaskets, 5 black neoprene gaskets for the lower chamber, and 5 3-hole gaskets.	6400-33
Gaskets: 6400-40 LCF	6400-41
Needle holding gaskets (5 sets)	6400-34
Teflon tape (3.5" wide, order by length)	212-02314
Upper half of std. 2x3 chamber	9964-028

Items for Chambers(Continued)	Part Number
Lower half of std 2x3 chamber	9964-029
Upper half of 2x6 chambers	9964-052
Lower half of clear bottom, 2x6 needle chamber	9964-051
Lower half of 2x6 narrow chamber	9964-050
O-ring (smaller) for outer air passage	192-04357
O-ring (larger) for inner air passage	192-04356
6400-17: seal for 2.5 inch pots	6564-279
6400--17: seal for Cone-tainers	192-09261
6400-17: O-ring for chamber top	192-09262

Items for the CO_2 Mixer	Part Number
Entire source assembly for 12 gram cartridges	9964-026
Connector block for CO_2 tanks	9964-033
12 gram CO_2 cartridges (25) (includes 9964-041)	9964-037
Spare parts kit for CO_2 mixer	9964-039
25 little O-rings, and a filter (Spares kit)	9964-041
Flow restrictor assembly	9964-042
Large O-ring between source and console	192-01919

Items for Sensor Head / IRGA	Part Number
Latch repair kit (has 6564-057 installed)	6400-903
Chamber catch rod (wire only)	6564-057
Chamber tension adjustment assembly	6400-908
Log switch replacement kit	6400-904
Fan motor (field installation version)	6400-902
Sensor head manifold	9864-033
Vinyl manifold gasket (top of sample cell)	6564-022
Screws for sensor head manifold	140-04234
Vinyl gasket (mirrors)	6564-021
Screws for chamber mirrors	140-04236

Items for Sensor Head / IRGA	Part Number
Connector screws for chamber/IRGA cables	314-04913
Chamber handle screws	122-00049
Tripod bracket screws	122-04276
Internal scrubber bottles (2), with chemicals	6400-950

Troubleshooting

When things go wrong

20 *Troubleshooting*

The LI-6400 is sufficiently complicated that sooner or later, you will encounter some inexplicable and troubling behavior, whose cause can range from something you've done wrong or misunderstood, to a component failure that requires factory attention. This chapter is designed to help you sort out causes and cures when things go awry.

Power On Problems

Usually, when you turn on the LI-6400, there is about 5 or 6 second wait while the displays shows

    INITIALIZING.

Followed by sequence of messages until

```
            STARTING...
         STARTING NETWORKING
              ######
```

Figure 20-1. The version 6 startup screen. If it is not an XT, the middle line will be blank.

This is followed by the Welcome screen with a 5 second countdown (Figure 20-2).

```
         Welcome to the LI-6400
               Version 6


Launching Open in 5
```

Figure 20-2. The version 5 welcome screen.

This section describes some of the *unusual* things that might happen during the power up period.

Console Doesn't Power On

If nothing happens when you power on:

- **Does the display flicker, or show any lines momentarily?**
  If no: It could be a totally dead battery, or blown battery fuse. Check the digital board fuse (Figure 19-9 on page 19-12). Make sure the fuse holder is making contact with both ends of the fuse. Sometimes, one end gets spread apart when a fuse is installed.

  If yes: It could be a very discharged battery. Try another one.

  If it's not a battery or fuse, then it's either a problem with the display, or a problem with the digital board. If you've had the display unplugged recently, make sure the display connector (Figure 19-9 on page 19-12) is plugged in correctly. If it is wrong, you may have already done some damage (see Figure 19-14 on page 19-17). Contact LI-COR for further assistance.

Screen Becomes Black

If the stored contrast file (/dev/.lcd) is missing, the display contrast may go full dark sometime during the INITIALIZING. sequence. After waiting about 10 seconds, press **ctrl** + **shift** + ↓ several times to adjust the contrast (this automatically regenerates the contrast file). It should work better on the next power on.

Powers off

If the instrument powers off right after you respond to

        Is the Chamber/IRGA connected ?

- **Wrong power configuration?**
  You might be operating in following configuration: The LI-6020 charger connected to the console, and no battery (or a discharged one, or one with a blown fuse) connected to the console. This doesn't work. See **Powering the LI-6400** on page 2-18.

  If you've got what appears to be a dead battery, see **Replacing the Battery Fuse** on page 19-9.

- **Cable or sensor head problem?**
  Try disconnecting all the cables from the console. There may be a problem in the IRGA or chamber that is pulling the voltage low enough to turn the console off. See if you can isolate it.

Welcome screen doesn't appear

The /Sys/Autostart folder or its contents may have been tampered with (see **The Autostart Folder** on page 5-23). Try re-installing system software (**Installing System Software** on page 2-22).

Continually Blows a Fuse

The important questions here are 1) What fuse is blowing? 2) When does it happen? 3) Does it happen with the chamber/IRGA detached (but the cable attached)? 4) Does it happen with no cable attached?

Here are some hints:

- **Flow board fuse**
  Items in this circuit are the pump, chamber mixing fan (see **The Mixing Fan** on page 20-39), and 6400-02 or -02B LED light source (or LCF light source) fan, and the match valve. Try to isolate the problem. One of the steps might be to leave the chamber cable plugged into the console, but not into the sensor head, to determine if it is the cable itself that has a problem.

  Note also that when you power up normally, the match valve first tries to operate just before OPEN's main screen appears. The pump and chamber fan stay off until the first time you enter New Measurements mode.

- **IRGA board fuse**
  Only the gas analyzers are on this circuit. If this fan is blown, you may not even know it, unless you notice the IRGAs have no dynamic response.

- **TEC fuses**
  Protects the circuit driving the thermoelectric coolers on the chamber.

- **Fan fuse**
  Protects against problems with the chamber mixing fan. If the fan is the culprit, however, it may be the flow board fuse that goes first.

- **Digital board fuse**
  If this is blown, you'll have no display, and no keyboard control.

Real Time Clock Problems

Doesn't Keep Time When Power Off

The first thing to check is the clock's battery (Figure 20-3). Measure its voltage relative to ground, as shown; it should be 2.7V or greater.
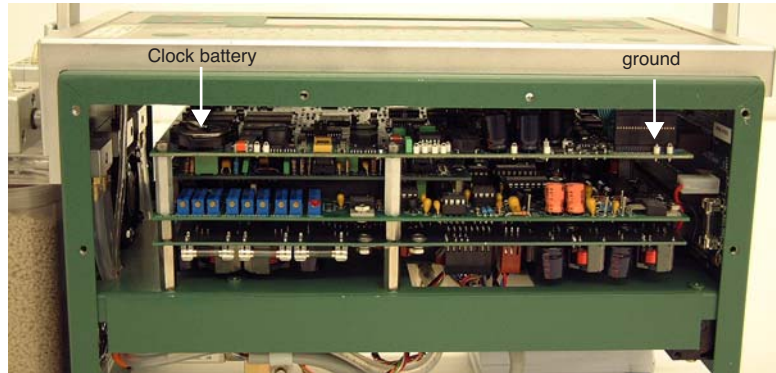
Figure 20-3. Location of the clock battery.

The battery (LI-COR part number 442-03791) is a "coin" type, and is held in place in a holder on the corner of the board. It is easily removed, but be sure the instrument is powered off.

"Clock Stopped" Message

Open's main screen displays the time and date, which should update each second. If the real time clock is not running, the message "Clock Stopped" is displayed instead. If this happens, contact LI-COR.

New Measurements Mode Warning Messages

In New Measurements mode, a variety of messages can appear in the second line of the display. This section lists, in decreasing order of severity, the messages and their meanings.

These warning messages are generated by a function that runs every 10 seconds while in New Measurements mode, so they appear or disappear at 10 second intervals. (A message may linger for up to 10 seconds after the offending condition has been remedied.)

These messages can be suppressed (and re-enabled) by pressing **ctrl** + **z** (in New Measurements mode only). Also note that every time New Measurements mode is entered, the messages are re-enabled automatically.

"BLOWN FUSE (Analyzer or Flow)"

Instruments having serial number 401 and above are equipped with a back plane board that is modified to allow detection of a blown fuse. Two fuses can be detected: the analyzer board fuse, and the flow board fuse. It cannot tell which one is blown, just that one or both are blown. See **Replacing the Fuses** on page 19-11.

"IRGAs Not Ready"

Refer to the discussion starting on page 20-15.

"High Humidity Alert"

The lowest value T of the three temperatures measured in or near the chamber (analyzer block, chamber air, and leaf) is used to compute a relative humidity RH_{alert}:

$$RH_{alert} = \frac{W_s \frac{P}{1000}}{e_s(T)} \times 100 \qquad (20\text{-}1)$$

where W_s is sample water concentration (mmol mol^{-1}), P is ambient pressure (kPa), and the function $e_s()$ computes saturation vapor pressure (kPa) as a function of temperature (C) (Equation (14-24) on page 14-13). If RH_{alert} exceeds 95%, the "High Humidity Alert" message appears. The usual remedy is some combination of the following:

- **Give the coolers a warmer target.**
  Perhaps you're asking for a block or leaf temperature that is too cold.

- **Increase the flow rate.**
  This will reduce the chamber humidity.

- **Dry the incoming air further**
  Increase the desiccant scrubbing.

- **Water IRGA OK?**
  The problem could be with the water IRGA. Do its readings, *H2OR* and *H2OS*, make any sense? Are they responsive? (If the readings don't seem to change, then the analyzer board fuse may be blown (page 19-11); this can happen without causing the IRGAs Not Ready message.)

"Chamber Fan is Off"

This message refers to the internal fan in the sample cell, which is controlled via function key **f1** (level 2), labelled **FAN**. Note that the instrument cannot sense if the fan is actually moving or not; the message is what is *should* be doing. The only way to tell if the fan is functioning is to turn it on and off and listen for sound changes.

"Pump is Off"

If the pump is off (accomplished by selecting "None" in the Flow Control Options (**2 F2 N**)), this message will appear. The software senses the pump's status by checking the digital output that controls it. If the pump is in fact not operating due to other reasons (not being plugged in, blown fuse, etc.), it will *not* cause this message to appear.

"Flow is Too Low"

The minimum recommended flow is 50 μmol s^{-1} with a 6400-01 Mixer, or 100 without. In fixed flow mode, if the flow drops below this value, the "Flow is Too Low" message will appear. Note that if OPEN is in any other flow control mode, flow rates less than this are tolerated (since the system is controlling flow rates, and not the user), and this message will not appear.

If "Flow is Too Low" appears even though the *requested* flow rate is higher than 50 μmol s^{-1}, check for a blown flow board fuse; the *real* flow may be 0 because the pump is not running. If the flow board fuse has failed, a number of other things will not be functioning as well, such as the console cooling fan, light source, light source fan, cooling fans, chamber fan, and CO_2 injector.

"FLOW: Need ↑SCRUB or wetter target"

The "↑SCRUB" refers to the desiccant tube adjustment label, and the direction you should to turn the knob. Specifically, this message means that the target humidity is low, and that the flow rate can't be increased any more in an effort to drive the humidity down. The remedies include:

- **Turn the desiccant tube knob toward SCRUB**
  If possible, drier incoming air is needed.

- **Pick a higher target humidity, (smaller VPD)**

- **Reduce the leaf area**

- **If the humidity is in fact dropping, wait**

A variation of this message is

"FLOW: Need ↑SCRUB - H2OR > Target"

This means the reference water IRGA (and hence the air incoming to the chamber) is already higher than what you've asked for the target to be. Unless you can convince the leaf to remove water vapor from the air, your only recourse is to dry the air a bit, or raise the target humidity (which means *reduce* the VPD target, if that's what you are targeting).

"FLOW: Need ↓SCRUB or drier target"

The water target is high enough to cause the flow to drop to its minimum, perhaps temporarily, while the system waits for the leaf to humidify the chamber. The remedies:

- **Turn the desiccant tube knob toward BYPASS,**
  More moist incoming air is needed. (See **Humidifying Incoming Air** on page 4-52).

- **Pick a lower target humidity, or larger VPD**

- **Increase the leaf area.**

- **Wait for the humidity to climb.**

"Negative PAR! LightSource? Cal?"

The ParIn_μm reading is less than -10 μmol m^{-2} s^{-1}. You likely have the wrong light source specified, since Red/Blue LED sources generate negative signals, while the Red only, and the standard chamber top light sensors generate positive signals.

"No Light Signal - Check Source"

This message will appear when the following conditions are all true: 1) Configured for LED or LCF light source, 2) D/A driving the LEDs is > 1000 mV, 3) measured internal PAR sensor signal is < 20 mV.

There are a variety of ways to get this to happen. Some represent hardware failures, and some human failures.

- **Light Source not plugged in**
  This can be either the lamp itself or the detector connector.

- **LED Source Mix-up**
  You can <u>not</u> treat the 6400-02 LED Source like a 6400-40 LCF, or vice versa. They get their light sensor signals from different places.

- **Lamp Failure**
  See **Source Won't Turn On** on page 20-30.

"IRGAs Warming Up"

If the IRGAs are indicating an unready state within the first 2 or 3 minutes of power on, the displayed message with be this warm-up message. After that time, the dreaded "IRGAs Not Ready", discussed above, will appear.

"RGB Source: WARM / HOT / 5V / LowBatt / NoBatt"

This is a 6400-18 RGB Light Source warning message. See **Warning Message** on page 20-35.

Unreasonable Results

The first indicator of trouble to novices at gas exchange will typically be those values the user is paying attention to: photosynthesis, conductance, C_i, etc. Sometimes, the tendency is to blame the computer ("This thing is computing crazy numbers for photosynthesis!"). However, the program is doing exactly what it has been told to do (that being the abominable nature of computers); crazy numbers come from crazy inputs. You will need to look behind the computations and determine which input is crazy and why.

Photosynthetic Failures

Photosynthetic rate (Equation (1-15) on page 1-10) is primarily based upon a) the difference between sample and reference CO_2 readings and b) flow rate, so look at those three variables (*CO2R_µml*, *CO2S_µml*, and *Flow_µml*) to determine which, if any, is causing problems. There is also a dilution correction, so wild values for *H2OR_mml* and/or *H2OS_mml* can also have an effect.

■ **Unstable Photosynthetic Rates**

If photosynthesis seems to be jumping around, try these suggestions:

1 **Are you just impatient?**

Bear in mind that right after a change in input conditions (such as a substantial change in the CO_2 mixer setting), there will be a short period (up to 1 or 2 minutes) where the photosynthetic rate may be nonsense, since both IRGAs are coming to new equilibrium values.

2 **What's the magnitude of the variation?**

There will always be some variation in the displayed value of any measured or computed quantity. Is the variation you see excessive? Is it due to the normal noise in the analyzers? Remember that at low rates, the noise in the CO_2 differential (typically 0.4 ppm) will become more and more significant. (So: is the variation in $\Delta CO2$ greater than 0.4 ppm?).

3   Watch those flow rates

For purposes of troubleshooting, operate in fixed flow mode, and set the flow to about $500\ \mu\mathrm{mol\ s}^{-1}$. If you are operating in constant VPD mode, or constant RH, you could be having problems by asking for a humidity value that cannot be achieved given the flow limitations and transpiration rate of the leaf. For example, if you have asked for 80% RH, and have a stressed leaf with nearly closed stomata, and are using very dry input air, your flow rate will go to zero (or about $30\ \mu\mathrm{mol\ s}^{-1}$ with a CO_2 mixer installed) while the system waits in vain for the leaf to raise the humidity to what you asked for. Meanwhile, the computed photosynthetic rate will be quite unstable, being the product of a growing CO_2 differential and a near-zero flow rate.

If the photosynthetic rate is low, try operating at a low fixed flow rate (such as $100\ \mu\mathrm{mol\ s}^{-1}$); this will a) keep the flow rate stable, and b) make the CO_2 differential as large as possible. See **Dealing With Low Rates** on page 4-51 for other suggestions.

4   Is the input stable?

Watch reference CO_2 (*CO2R_μml*) for 15 seconds. How much does it vary?

Expect variations near $0.2\ \mu\mathrm{mol\ mol}^{-1}$ at ambient concentrations. If it is much more than that, you may have problems. An open system such as the LI-6400 depends upon stable inputs. Because air flows through the sample and reference cells at different flow rates, and because there are differing volumes involved, any fluctuation in the input will show up in the sample and reference at different times, causing the differential value to oscillate.

If using the CO_2 Mixer: Set it to control reference concentration (R), as this will ensure that the changes aren't coming from the system's attempts to maintain a particular concentration in the leaf chamber, and make sure the soda lime adjustment knob is on full scrub. (To test the mixer stability, see page 20-25).

Not using the CO_2 Mixer: A larger buffer volume is probably called for. See **Air Supply Considerations** on page 4-50. Or, there may be moving debris in the sample cell. If so, it will affect both *CO2S_μml* and *H2OS_mml*. See **Unstable CO2 and/or H2O** on page 20-17.

5   Is the sample cell stable?

If the reference values are stable, but the sample values aren't, try testing for a leak in the chamber. See **Sensor Head Leaks** on page 20-44.

■ **Photosynthesis is stable, but "can't be right"**
An example of this problem would be negative, low, or ridiculously high photosynthetic rates on fully illuminated leaf of a well-watered, healthy plant.

1   Check the chamber conditions
Is CO_2 where you'd like it to be, or is it near zero? (A common mistake: not using the mixer, forgetting to change the scrub tube setting after testing the IRGAs' zero, and not noticing the absence of CO_2 in the reference air.)

Is the light what you think it is, or did you forget to turn on the LED source?

2   Check other inputs
Are you using the correct leaf area? You want the one-sided leaf area that is enclosed within the chamber. Is the flow rate (display line *b*) OK? It is typically between 200 and 700 μmol s^{-1}. Is the pressure sensor OK (display line *g*)? Typical values: 100 kPa near sea level, 97 kPa at 1000 ft., 83 kPa at 5000 ft., etc.

Questionable Conductances

Since conductances are usually between 0 and 1 mol m^{-2} s^{-1}, you might not notice problems with this value unless it makes intercellular CO_2 be negative (next section), or conductance itself is negative.

1   Leaf Area
If the value of leaf area that is being used is much too low, the leaf conductance will exceed the boundary layer conductance, and the stomatal conductance will become very large, eventually becoming negative (Equation (1-9) on page 1-9).

2   Match problem
Compare the sample and reference water IRGAs. Are they well matched? If sample is lower than reference (meaning transpiration is negative), that is a clear sign they aren't well matched.

3   Leaf temperature
Transpiration does not depend on the leaf temperature measurement, but conductance does. If the transpiration number appears OK, but the conductance doesn't, leaf temperature might be the reason. Is the sensor broken? Is it making good contact with the leaf? Is it well zeroed?

Negative Boundary Layer Conductance

Boundary layer conductance is normally computed from a lookup table based on leaf area and fan speed (see **Boundary Layer Variables** on page 14-20). If you change to a chamber that allows large leaf areas, but use the standard 2x3 chamber lookup table, you can get negative boundary layer conductances as an artifact of extrapolating that table's data.

The remedy is to use the appropriate lookup table, or use a constant value. If you've already logged the data, you can recompute (Chapter 13) using an appropriate boundary layer value.

Impossible C_i's

The intercellular CO_2 value is essentially the ratio of photosynthetic rate to conductance (Equation (1-18) on page 1-11). The typical problem is that C_i is too low or negative. Conductance is generally the culprit, but here are three things to check:

1   Transient condition?

Very low C_i's can be real, especially for short time periods. Example: take a low light leaf into bright light. The plant's photosynthetic biochemistry responds much faster than do the stomata, so until the stomata can open wider, CO_2 is consumed within the leaf, and C_i will be low. (Negative C_i's, of course, cannot be real).

2   Photosynthesis too high?

If the value of photosynthesis is too high, C_i will be too low. The primary culprit: mismatched IRGAs.

3   Conductance too low?

If something is making the value of conductance too low, that will drive C_i down low or negative. Possible reasons:

- **Poorly matched IRGAs**
  Is the match valve working ok?

- **A terrible water calibration**
  If you zeroed with wet air (bad desiccant?), all subsequent water readings will be too low, making conductance too low.

- **Bad leaf temperature measurement or computation**
  If leaf temperature is too high, conductance will be too low. If you are measuring leaf temperature, is the thermocouple working? Is there good contact? Is it well zeroed (page 18-24)? If you are computing leaf temperature (energy

balance), do the values seem reasonable? Do you have the right light source specified?

Pump/Flow Problems

A useful flow diagram for troubleshooting is Figure 20-23 on page 20-51.

Can't Achieve High Flow Rates

Maximum flow rates should exceed 700 μmol s^{-1}. If you can't achieve that (use fixed flow control, so you can directly ask for the flow rate you want), then try these steps.

1   **Turn off the mixer.**
If you are using the 6400-01 CO_2 mixer, shut it off, and try a high flow rate again. If this fixes the problem, then the flow was low because the mixer calibration had specified a lower maximum flow in order to raise the upper limit of CO_2 concentration. Refer to **Calibrating the CO2 Mixer** on page 18-25.

2   **Remove the soda lime tube**
If the flow does not increase substantially, go to Step 3.

If the flow did increase substantially, the problem is either with the soda lime tube, or else in the line between the air intake and the soda lime tube. To test the soda lime tube itself, remove the desiccant tube (that will make the flow be highest) and put the soda lime tube in the desiccant tube location. If the flow doesn't change much between having nothing in the desiccant tube location, and having the soda lime in the desiccant location, that means the soda lime tube is ok, and the problem is in the internal tube connecting the air inlet and the soda lime location. Open the console, and inspect that line; it may have debris in it.

If the problem is the soda lime tube, see **Chemical Tubes** on page 19-2.

3   **Remove the desiccant tube**
If this improves the flow, the problem is in the desiccant tube. To find a blockage within the desiccant tube, see **Chemical Tubes** on page 19-2.

If this doesn't fix the problem, open the console. (See Figure 20-20 on page 20-48.) There might be some sort of blockage between the air filter (which is next after the desiccant tube) and the pump. Pull the air intake line from the pump, and see if that increases the flow rate. If it does, you've bracketed the problem.

The problem could also be a bad pump or diaphragm, but this has been quite rare. See **Pump** on page 20-48.

Pump Status: ERR

This can only happen with the 6400-01 installed, and then only when something is restricting the pump. Check for a blockage in the flow path on the vacuum side of the pump, as described above.

IRGA Problems

The IRGA Diagnostic Screen

A very useful display when troubleshooting the gas analyzers is the IRGA Diagnostic Screen (Figure 20-4). In New Measurements mode, press **[**, followed by **F** to see it.

```
               Raw Signal (mV)              AGC Signals (mV)

                      µmol mol-1    mmol mol-1

                                                    Digital Status

    (F) IRGA Diagnostic Display
                mV          ppx          AGC   ok
    CO2R     1663.8      643.31       -47.2   1
    CO2S     1660.2      630.02       173.4   1
    H2OR     1355.4      14.627     -1559.5   1
    H2OS     1368.2      14.819     -1358.9   1
    Tirga=25.95 Tcham= 21.26 Tblk= 19.99
    Pressure = 97.47 kPa (1730.1 mV)
```
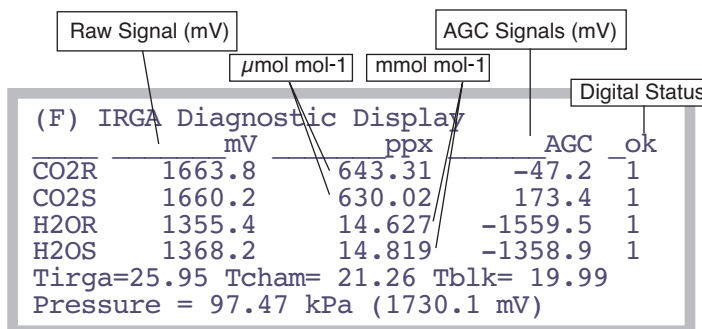
Figure 20-4. IRGA Diagnostic Screen.

Raw Signals

This values represent the IRGA's raw output. They are useful in deducing of a problem is hardware or software related. The final concentrations, which you normally see while operating, are computed from the raw numbers, plus a lot of other things: calibration coefficients, zero and span corrections, temperatures, pressure, etc. If the final values don't make sense, but the raw num-

bers do, then the problem is likely software (that is, bad calibration, etc.). A typical relationship between raw and final values is shown in Table 20-1.

Table 20-1. Approximate (very) relationship between raw signal and concentration.

raw signal mV	CO2 μmol m^{-2} s^{-1}	H2O mmol m^{-2} s^{-1}
0	0	0
500	130	4
1000	300	9
2000	750	24
4000	2300	70

AGC voltages
The AGC signals are shown the IRGA Diagnostic Screen (Figure 20-4 on page 20-14). These signals indicate how much radiation is attenuated in the reference wave band (nonabsorbing for both CO_2 and H_2O). With a good IR source and clean optics, these values are typically 0 or less. As the optics become dirty, these values will increase. Eventually (near 5000mV), the message "IRGAs Not Ready" will appear (and the CO_2 and/or H_2O status indicators (level c in the diagnostics display, j in the standard display) will stop showing OK).

Digital Status
1 is OK, 0 is not ok. If any of these are not OK, then this is the condition for the "IRGAS Not Ready" message, discussed next.

"IRGAs Not Ready" Message
The "IRGAs Not Ready" message will show up in New Measurements mode, and also during the IRGA zeroing and span setting routines, whenever one or more of the gas analyzers is indicating a problem (see discussion under **"CO2" and "H2O"** on page 14-15). What triggers this message is "too much" light attenuation in the non-absorbing reference wave bands for CO_2 and H_2O. Causes:

- **Power**
  IRGA connector not seated, bad cable, blown analyzer board fuse.

- **IRGA not functioning**
  If power is getting to the IRGA, but it is still not functioning, the chopper motor may be stalled.

- **Low Light**
  Too much debris in the sample cell, dirty optics, or a failing source or detector.

■ **Solving "IRGAs Not Ready"**
Here is a logical step-by-step to resolve the problem.

1 **Are the IRGAs warmed up?**
The message should go away within 2 or 3 minutes of when the IRGAs are powered on.

2 **Round connector fully seated?**
One cause of "IRGAs Not Ready" is the IRGA connector not being fully plugged in. The connector in question is the round one that attaches to the sensor head. There are two red dots on the mating halves of the connector, and they will be nearly touching when the connector is fully seated. When making the connection, push the connector in until you hear a snap (which we trust wasn't a weak bone in your wrist giving out).

3 **Is the chopper motor running?**
Determine if the chopper motor is stalled (see **Stalled Chopper Motor** on page 20-20). If the chopper motor is running, continue with Step 4. Otherwise, we are down to three possibilities: 1) a blown analyzer board fuse, 2) a bad cable (between the console and the sensor head), 3) bad chopper motor. If it is one of the latter two things, then contact LI-COR.

4 **Check the AGC voltages**
These values are described on page 20-15. If the reference cell AGC values are well below 5000, but the sample ones are >5000, then clean out the sample cell (page 19-34).

5 **Contact LI-COR**
If you've gotten to this point (the chopper motor is running, and the chamber and optics are clean, but "IRGAs Not Ready" persists), then contact LI-COR.

IRGA(s) Unresponsive

Unresponsiveness is when CO_2 and/or H_2O doesn't seem to respond to what should be large changes, such as blowing into the chamber, or changing the chemical tubes from full scrub to full bypass.

- **Power? (Fuses, Cable, Connector)?**
  Lack of power to the IRGA usually would trigger the "IRGAs Not Ready" message, but not always. The unpowered, digital status lines from the IRGA

could just happen to be in their "ready" states, so perhaps the IRGA is not powered. Check the connector and the fuse.

• **Bad calibration?**
Check the zero and span values in the <user> node in "`View Current...`" under "`View Settings`" in the Calib Menu. The span values should be close to 1.0, and the zeros will be between +/- 5000, but the closer they are to either limit, the more suspect they are. Typically, they are numbers in the hundreds. Try resetting the zero and span values to factory settings to see if that makes a difference.

Unstable CO_2 and/or H_2O

The noise of the LI-6400 analyzers is typically 0.2 μmol mol^{-1} for CO_2, and 0.04 mmol mol^{-1} for H_2O[1]. Leaks, diffusion, inadequate chemicals, and fluctuating inputs are some of the things that can increase apparent signal noise.

■ **Tracking down a noisy signal**
This discussion assumes CO_2 is the noisy quantity, but similar logic applies to H_2O (except for Step 2).

1 Is it really noisy?
Remember that 0.2 μmol mol^{-1} is typical noise for CO_2, and 0.04 mmol mol^{-1} is typical for H_2O. These values will be larger at higher concentrations.
A good method for quantifying these values is to add *CO2R*, *CO2S*, *H2OR*, and *H2OS* to the stability list (see **Stability Indicators** on page 4-41). Then you can monitor the running standard deviations of those signals.

2 Are you using the CO_2 Mixer?
Shut it off, and continue with these steps. If you don't locate the problem, try **6400-01 CO2 Mixer Problems** on page 20-24.

3 Is the instability flow dependent?
Close the chamber, and stop the flow (**2 F2 N**). If the instability seems to go away, then a leak or fluctuating input is suggested. See **Finding Leaks** on page 20-44. If the test was inconclusive, keep reading.

4 Is the instability fan dependent?
If the instability is only in the sample cell, and if it goes away or is much reduced when the chamber fan is turned off, then the problem is likely some

---

[1] 4 second averaging, at 350 μmol mol^{-1} CO_2, and 20 mmol mol^{-1} for H_2O.

fuzz or hair or small particles that are moving around in the sample cell while the fan is on. Clean the cell (page 19-34).

Another possibility is that the fan itself is causing the instability. See **Noise caused by the Fan Motor** on page 20-40.

5   Feed stable air directly to the sensor head
A more definitive test is to feed a constant concentration (e.g. air from a tank) directly to the sensor head, much like the configuration when setting the span. In fact, you can use the span setting routine in the Config Menu to gain control over the match valve (but don't actually change the span). With a flow of stable air going through both IRGAs, is the signal now much more stable? If yes, consult **Finding Leaks** on page 20-44.

6   View the AGC voltages
If the IRGA sample signal is noisy, but an unstable input or leaks don't seem to be the problem, the next thing to check is for foreign objects in the sample IRGA. There is a way to check for this without disassembling the IRGA, however; look at the AGC voltages (discussed in **AGC voltages** on page 20-15).

Observe the sample cell AGC signals. They should not be varying by more than a mV or so. If they are fluctuating more than that, and if the reference ones are stable, that could indicate that there is debris being blown about in the sample cell. If the reference values are jumpy, it could be electronics or a chopper motor problem. Instability in just one of the four channels could indicate a demodulator board problem: go to Step 9.

7   Check the raw voltages
If the problem isn't debris in the sample cell, and isn't leaks or fluctuating input concentrations, then it suggests an IRGA hardware problem, or some other variable (temperature or pressure) is fluctuating, causing the instability. One way to eliminate the latter possibility is to monitor raw IRGA signals. Use the Diagnostics display (described on page 20-50), and view display line *b*. If the raw IRGA signals are not stable (fluctuations > 2 mV), go to Step 9.

8   Check pressure and temperatures
If the IRGA mV values are stable, but the molar concentrations aren't, then look for instability in pressure, IRGA temperature, chamber temperature, and block temperature. (All of these are on the IRGA diagnostic display, page 20-14.) Pressure is used for both sample and reference concentration computations. Block temperature is used for reference concentrations, while the average of block and air temperatures are used for the sample cell. (The equations are on pages 14-6 and 14-8.)

If the culprit is pressure or temperature fluctuations due to sensor or circuitry problems, there's not much to do except the final step in this sequence, which is to contact LI-COR. There is a stop-gap work-around, however, that could keep you going, and that's to use a constant value instead of a measured one for the offending sensor. You could, for example, change the pressure sensor's calibration coefficients to 98 and 0 (offset and slope), which would cause pressure to remain fixed at 98 kPa.

9   Contact LI-COR
If you have gotten to this point by carefully following the logic, you have determined that the instability is not due to fluctuations of concentration in the incoming air, leaks, debris in the cell, or unstable temperature or pressure signals. An IRGA hardware problem is suggested, so contact LI-COR.

Readings Obviously Wrong

If you just don't believe the IRGA readings, then try these steps:

1   Is it responsive?
Watch reference readings, and go from full bypass on both soda lime and desiccant, to full scrub. If the IRGAs don't respond, turn to **IRGA(s) Unresponsive** on page 20-16.

2   Zero and Span
Check the present values found in "View,Store Zeros & Spans" in the Calib Menu. (For a discussion of these numbers, see **Managing Calibration Data** on page 18-2.) Try resetting them to the factory defaults, or follow the zero and span setting procedures starting on page 18-11. A common cause of problems here is zeroing the IRGAs without having a truly CO_2-free or H_2O-free air stream.

3   Verify that the chamber fan is operating
Without mixing the sample cell, a leaf will have little effect on the sample IRGA readings, which can make for strange behavior. The verify the fan's operation, use your ears: turn the fan off (**3 F3 0** (that's a letter, not a number)) and on again (**F3 F**), and listen for the sound changes, if any. (No sound change - no fan - no good.)

Occasional Instability

This problem is characterized by an occasional jump in the IRGA reading, for no apparent reason. Before deciding there is an electronic problem, eliminate a couple of other possibilities:

- **Insects?**
  Flying insects can fairly easily get into the sample cell when the chamber is open, or even into the match valve. Those that find the sample cell are destined to eventually encounter the mixing fan and become debris, but prior to that, you will be seeing the effects of insect respiration on your measurements. So, if you see periodic spikes in the sample CO_2 (such as 5 or 10 μmol mol^{-1} every minute or so), you may have acquired a guest.

- **Leaks?**
  Check for leaks (**Finding Leaks** on page 20-44).

- **Connections?**
  See if there is any relationship between the jumps and movement of the cable. There could be a faulty connection at work. Monitor the CO_2 sample and reference concentrations with a strip chart. That makes it easier to detect a jump.

Stalled Chopper Motor

The chopper motor is the motor that spins the filter wheel in the IRGA/sensor head. This motor should begin to run shortly after the IRGAs are powered on. If the chopper motor does not run, the "IRGAs Not Ready" message will be displayed in New Measurements mode. The typical chopper motor failure is due to the bearings. So, prior to its demise, there may be increased audible noise coming from the motor, and even subsequent electronic noise in the IRGA signals.

■ **Determining if the Chopper Motor is Running**

1 **Turn off the pump and chamber fan**
In New Measurements mode, turn off the pump (**2 F2 N**) and chamber fan (**3 F3 0** [the letter, not the number]), so you can hear the chopper motor.

2 **Put the LI-6400 to sleep**
Go to the Utility Menu, and select "Sleep Mode". Listen for a motor in the sensor head (NOT the console) to wind down once you press **Y** indicating it is OK to sleep.

3 **Wake the LI-6400 up**
Exit the Utility Menu. The fan in the console will begin running immediately, but listen for the chopper motor to begin running after that (press your ear to the IRGA); it should start anywhere from 10 seconds to 1 minute later.

If you don't hear the chopper motor, the problem could be a blown analyzer board fuse, a not-fully-seated IRGA connector (see page 20-16), a bad cable, or a stalled chopper motor.
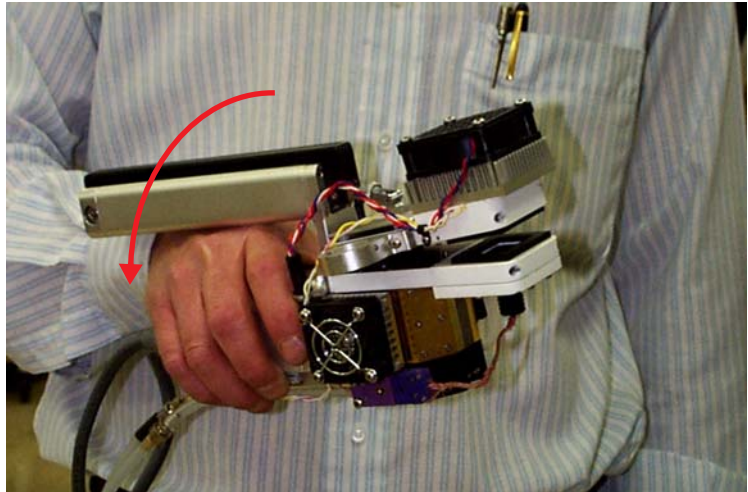
Figure 20-5. Grasp the IRGA as shown, then flick (abruptly turn) your wrist to try to start a stalled chopper motor.

■ **Restarting a Stalled Chopper Motor**
If you didn't hear any evidence of a running chopper motor in the above sequence, then you can try starting it.

1 Make sure the IRGA is powered
Being in OPEN's Main Screen, or in New Measurements mode, is sufficient for this.

2 Have the pump and fans off so you can hear
Same as Step 1 under **Determining if the Chopper Motor is Running** above.

3 Move the filter wheel by inertia
Grasp the IRGA, and give it a quick roll (left or right) (Figure 20-5). You may have to be aggressive; if the bearings are getting bad, it might not start easily.

4 Temperature Control Problem?
Another possibility is that the chopper has stopped due to lack of temperature control. If the IRGA has gotten above the specified operating temperature (50C), or if something has failed in the temperature control circuit, it can

cause the chopper to quit. If the IRGA is hot, cool it down to about 30C and see if it starts working.

Match Valve Problems

"CO$_2$ has Changed" Message

This message appears in match mode when the sample CO$_2$ concentration changes by more than 3 μmol^{-1} since entry. This can be caused by two things:

- **Match mode entered at the wrong time**
  If the light just changed, or you just closed the chamber, or the CO$_2$ mixer target just changed, or you just made an adjustment on a chemical tube flow knob, then the change in CO$_2$ is not indicative of a problem with the system, just with your timing. Wait for *CO2S_µml* to stabilize, then match.

- **Chamber leak**
  If *CO2S_µml* never does stabilize, then you very likely have a chamber leak. See **Sensor Head Leaks** on page 20-44.

"Excessive Deltas" Message

This message appears when you try to match, but the differences between sample and reference are too large. This could be due to very poorly set IRGA span or zero, but may also indicate a problem with the match valve or its plumbing. The default limits are 10 μmol mol^{-1} for CO$_2$, and 1 mmol mol^{-1} for H$_2$O.

- **Is the Match Valve functioning?**
  Figure 4-2 on page 4-34 shows how the match valve should be positioned in and out of match mode.

- **Is the return flow tubing in place?**
  Check to be sure there is a piece of tubing connecting the chamber bottom with the match valve (Figure 20-6).

"CO2R Didn't Change" Message

After the initial delay when entering match mode, during which the H_2O reference reading is supposed to stabilize, if the message

```
=================== Warning ===================
  CO2R didn't change enough. Match
  valve OK? Return tube in place?
```

appears, it is because the CO_2 reference reading changed less than $1.5\ \mu$mol mol^{-1} after the match valve closed, and the expected change (the pre-match difference between sample and reference CO_2) was larger than 10 μmol mol^{-1}. Reasons for this would be a match valve that is sticking, or the air flow tube connecting the chamber to the match valve not being in place, or some other flow related problem. Note: This message could be spurious the first time you match. Suppose the IRGAs badly need matching ($\Delta CO_2 > 10$), and the chamber is empty. When you enter match mode, the reference concentration won't change (because the chamber is empty), and the system gets fooled into displaying this warning.

Match Valve Doesn't Move

Stuck match valve? See **Match Valve Maintenance** on page 19-29.

Chamber exhaust tube
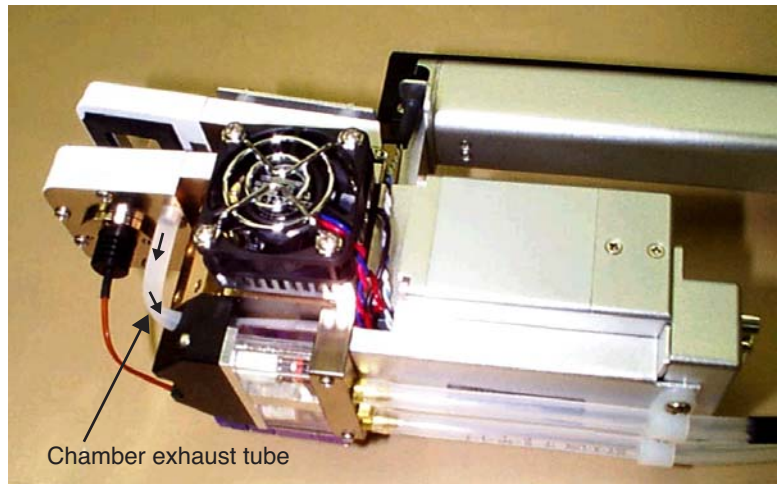
Figure 20-6. The exhaust from the leaf chamber is sent to the match valve.

6400-01 CO$_2$ Mixer Problems

Stays at Zero

If the CO$_2$ seems to stay at zero even though you're asking for something else, then check these possibilities:

1   Cartridge installed recently?
Once pierced, the CO$_2$ cartridge lasts for about 8 hours, whether you are using it or not.

2   Is the mixer actually turned on?
From New Measurements mode, press **2** then **f3** to access the mixer control panel, then press **C** for constant control signal, then enter 2000. If this makes the CO$_2$ values climb from 0, whereas the R or S options don't, then it would indicate a very bad mixer calibration. Go to the Calib Menu, and redo it.

3   Check the regulator's O-ring
Detach the cartridge holder/regulator from the console, and check to be sure the large O-ring that seals the hole connecting the console and the regulator is in place (Figure 20-7). If it is not there, the leak will prevent the mixer from operating correctly.

4   Check the regulator for flow
With a pressurized cartridge installed, there should be a very slight flow coming from the hole in the regulator (within the aforementioned large O-ring). One way to check this is to cover the hole with your finger for a 10 or 15 seconds, then release it suddenly; you should hear a little "ppffft" as the built up pressure is released. Alternatively, put a drop of liquid (soapy water or saliva) on the hole, and look for bubbles.

Lack of flow from the regulator can be due to a clogged filter or flow restrictor. See **Servicing the External CO2 Source Assembly** on page 19-38.

small O-ring   large O-ring
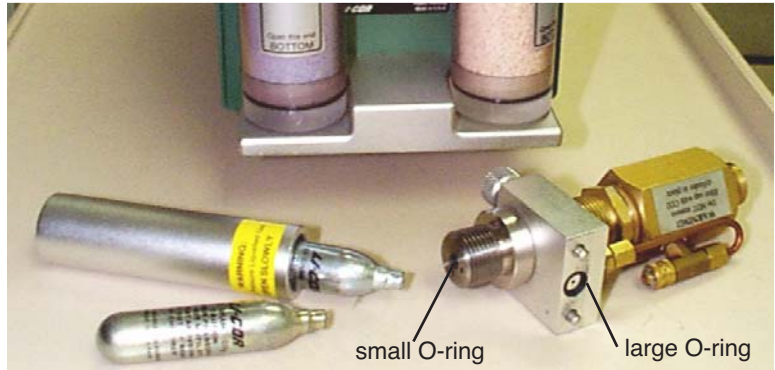
Figure 20-7. The CO_2 regulator's O-rings.

Instability

If the CO_2 concentration is not stable:

1   Is the soda lime OK?
The soda lime must be on full scrub. Shake the tube a little bit to break up any "channelling" that might be occurring. Also, try this test: exhale onto the air inlet connector of the console. If you see any response in *CO2R_µml* to your breath, then your soda lime needs to be replaced (If you see a response in those around you to your breath, then your mouthwash should be replaced as well.)

2   Use C mode
Put the mixer into constant control signal mode (in New Measurements, press **2** then **f3**, then **C**, then enter 2000). If the reference cell is much more stable in this mode, then the problem is due to the mixer hunting for the proper setting for the concentration you asked for. If you were using S mode (constant sample cell concentration), the instability could have been aggravated by flow changes (especially if you were also in constant humidity mode, in which the flow can fluctuate) or leaf photosynthetic changes. Try running in R mode instead.

If C mode doesn't help, then continue on:

3   Is it the IRGA?

Is the IRGA unstable, or is it the mixer? Go through **Unstable CO2 and/or H2O** on page 20-17 to make sure that it is really the mixer that is the cause of the instability.

4   Is the instability worse at high CO_2?

If the instability is not present or small at low concentrations, and noticeably worse at high concentrations, the problem may be due to inadequate flow from the external source assembly. See Step 4, **Check the regulator for flow** on page 20-24.

Cartridge Only Lasts a Few Hours

There is probably a leak in the CO_2 source. To do a leak test, remove the source from the side of the console and install a fresh cartridge. Dip the source regulator assembly into a clear beaker of water, and look for bubbles. *Dip it only as far as the bottom of the aluminum block that the mounting screws pass through*. (Do not dip it any deeper because water should be kept away from the hole where CO_2 exits the source. There is a flow restrictor behind this hole, and if it gets wet, it won't work again.)

The most probable source of a leak is one of the two compression fittings on the copper tubing that goes from the regulator body to the mounting block (shown in Figure 19-35 on page 19-40). If that is the case, tighten the nut on the copper tubing 1/8 turn at a time until the leak stops.

The leak could also be at the cap on the "T" fitting which contains the filter (same Figure). If this is the case, and tightening the cap doesn't help, then check the O-ring in the cap. It may be torn.

When you remove the source from its bath, *do not turn it upside down while it is wet*. This will prevent water from running into the regulator through the large hole on its bottom. Dry the bottom of the source and the inside of that hole with an absorbent towel before turning it over.

Slow Achieving a New Value

Typically, it takes 2 or 3 minutes[2] to go from 100 to 2000 μmol mol^{-1}, and less than 1 minute to come back down. If it takes a long time to go up, then see Step 4, **Check the regulator for flow** on page 20-24. If the problem is erratic closure on the target value (overshooting, undershooting, etc.), then try

_____

[2].See the important hint in **Getting To Know Your Mixer** on page 19-39.

doing the mixer calibration (page 18-25), to improve the mixer's "first guess-es".

Can't Achieve Low Values

Typically, the minimum value achieved by the mixer is 30 to 50^3 μmol mol^{-1}. If your minimum value is considerably above that, then here are some things to check:

1 **What's your maximum flow rate?**
Check the pump speed setting for mixer operations (Figure 20-8).

New Measurements Status display 'B'

```
(B) Flow Control Status

Pump mv: std=4500 mxr=4600  now=4609
 SetPt = 1318  Null Balance = 0  Rng=2
  Flow = 1319mV (499 µmol/s)
DAC offsets: H = 0, F = -2, Lim= 50
Status: OK (1)
```

<li6400> <user> <co2_mixer> <pump_mv>

```
═Current Calibration═══════════════
  ▶irga_match
 ▼co2_mixer
  ├ pump_mv= 4600
  ├ ppm= { 2265.32  1068.76  424.103
  └ mv= { 5000  3000  1500  1000  500
  └ parin_offset= 0.433045
 ▼/▶      ♦        ♦       ♦ EXIT  ♦ SELECT
```

Figure 20-8. Viewing the pump speed setting for mixer operations can be seen in either the New measurements status display 'B', or in the View Current... entry in the Calib Menu.

A "normal" value is 4500 or so. If it is much lower, redo the mixer calibration (**Calibrating the CO2 Mixer** on page 18-25).

2 **Is the soda lime on full scrub and is it good?**
Turn off the mixer, and verify that the reference CO_2 goes to zero. If it doesn't, there is a problem with the soda lime, or else the IRGA needs to be zeroed.

[3.]If you have modified the LI-6400 console plumbing according to App Note 7 (*Modification of LI-6400 to Control at Low CO$_2$*, PPS-267), then it should go all the way down to 0 ppm.

3   Backwards valve
(If someone has been working on the mixer...) If the valve labelled "Bottom valve" in Figure 20-9 on page 20-29 is installed *backwards*, then only very high CO_2 concentrations will be achieved.

Can't Achieve High Values

The CO_2 mixer should be able to achieve an upper limit in excess of 2000 μmol mol^{-1}. 2200 is a typical value. If you can't come close to that, or if it takes a long time to get there, check for the following:

1   Flow from the source
See Step 4, **Check the regulator for flow** on page 20-24.

2   Calibration problem?
Change to C mode (constant control signal), and set a 5000 mV target. If that fixes the problem, then re-do the mixer calibration.

3   Bad or leaking valve
(If someone has been working on the mixer...) If the valve labelled "Bottom valve" in Figure 20-9 on page 20-29 is installed loosely, or without the two O-rings, or has a wire pinched beneath it preventing a tight seal, then it might work OK at lower CO_2 values, but not at higher.

4   Plugged flow restrictor
Oil from the CO2 cartridges may have gotten to the flow restrictor on the source assembly. See **If the Flow Restrictor Becomes Clogged** on page 19-39.

5   Contact LI-COR

If nothing in the previous steps has fixed the problem, then oil may have gotten into the console, and fouled the internal part of the CO_2 mixer. Contact LI-COR.
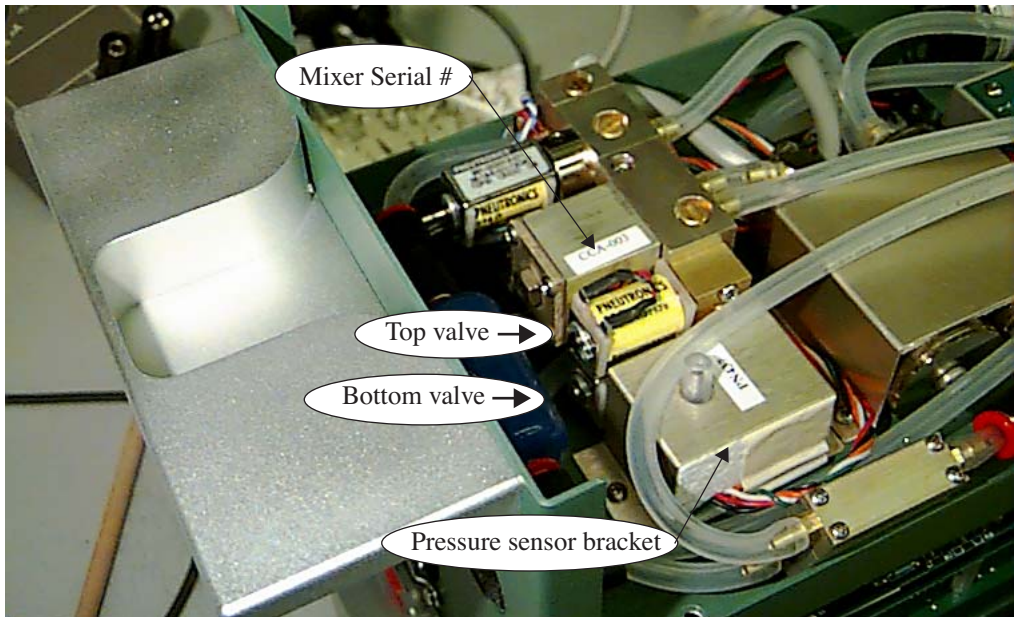
Figure 20-9. The in-console part of the CO_2 mixer.

Calibration Program Gives Erratic Results

If the CO_2 mixer calibration program gives erratic results - for example a curve that looks like steps instead of being smooth - check for leaks inside the console in the path that goes to the reference IRGA. In particular, make sure all the quick connectors have their tubing pushed into them as far as they should (see **Quick Connectors** on page 20-47). If there is a leak in the line going to the reference IRGA, CO_2 will change too slowly after a change in the mixer set point. OPEN 3.3 is less susceptible to this, since the minimum delay time was increased from 6 to 15 seconds with that software release.

Can't Seal the 12 gram Cylinder

If you have difficulty sealing the CO_2 cartridge (that is, if all the CO_2 escapes before you can get the cap screwed down), then

- **Is the small O-ring in place?**
  It is shown in Figure 20-7 on page 20-25.

- **Examine the pierced hole in the top of the cartridge**
  If it is not round, but oblong, then a bent piercing pin on the source may be the culprit.

Light Source / Sensor Problems

No Lamp Control Key
The function key for controlling the lamp is **f5** level 2 in New Measurements mode. If this key is labelled "-none-", then your configuration doesn't specify the LED source as the light source. Go to the Config Menu, select "View/Edit", and set the light source appropriately.

Source Won't Turn On
If the LEDs don't illuminate when they are supposed to, check the following:

1 **Is the lamp fan running?**
   If the lamp fan is running, but the LED's are not illuminated, go to Step 3.

   If the fan is not running, try blowing on it to make it move. If that starts it and brings the light on, the problem is that the lamp fan motor might have a "dead spot", and the lamp will not illuminate if the fan motor is drawing too much current. Or, if the fan is jammed with debris and can not turn, this will prevent the lamp from illuminating. On 6400-02B light sources, there is a thermistor

that is placed in the heat sink (Figure 20-10). Make sure it is not pushed too far in, as it can get into the fan and prevent the blades from turning.
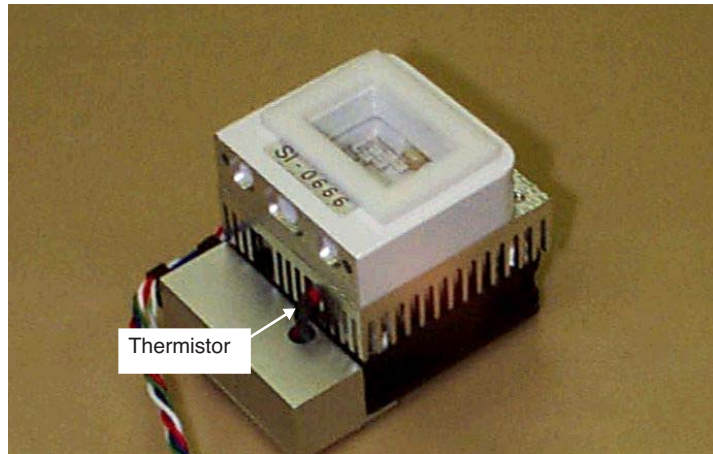
Figure 20-10. The temperature sensor in 6400-02B's.

2   Blown fuse?
The flow board fuse protects the lamp. If it is blown, a number of things will not work, including the lamp, lamp fan, pump and cooling fans.

3   Check the lamp voltage
Unscrew the lamp connector (Figure 20-11) to gain access to the wires in the connector. Measure the voltage (with the lamp turned on high) between the

orange and white wires. *Be careful, there should be voltage in excess of 100 Volts*.
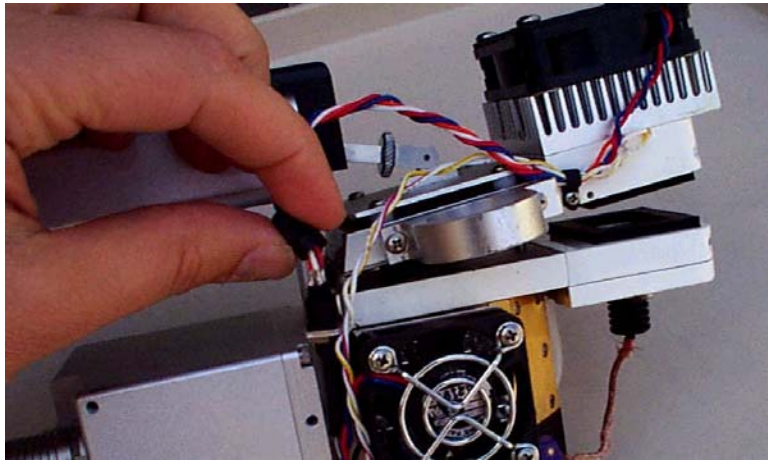
Figure 20-11. Unscrew the lamp connector to access the four wires. Caution - High Voltage.

If the voltage is near 0: A cable or connector problem is indicated. Continue with Step 4.

If the voltage is about 12V: The switching power supply (it's in the console) for the lamp has failed. Contact LI-COR.

If the voltage is over 100V: (Normal)The problem may be a broken connection within the light source itself. Contact LI-COR.

4   **Check the 26 pin D connector**
     Make sure none of the pins have been pushed in or broken (Table 20-2 on page 20-52).

5   **Check the cable**
     Try a different cable, if one is available to you.

Source Blinks On and Off

If the source blinks on and off with about a 3 second period, the problem is one of the following:

- **Light source detector connected?**
  If it isn't the source will blink or go to full intensity.

- **Right calibration?**
  The 6400-02's have positive calibration constants, and the red plus blue 6400-02Bs have negative. If you have the wrong sign on the calibration constant, the source will blink or go to full intensity. Go to the Light Source Control, and select the proper LED light source. If it's not in the list, add it via the Installation Menu.

- **Right Hardware?**
  You can't tell the software you're using a 6400-02 light source, and connect a 6400-40 LCF, for example.

PAR Sensor reads negative

Chances are you are not using a light source, but are configured for a red plus blue one. This will leave you with a negative calibration constant for the in-chamber light sensor. Go to the Light Source Control, and select the proper light source, such as Sun+Sky.

Source Isn't Bright Enough

The 6400-02 and -02B sources will decline in maximum output as they age (see **Aging** on page 8-10). There is a possible remedy if you suspect this could be the problem, and if you have serial number PSC-388 or below. These units operate the light source with a lower power limit than do later units, and this can be changed by a simple factory modification. Contact LI-COR for details.

6400-18 RGB Source

Communications Problem

When it needs to synchronize with the 6400-RGB Source, the console checks to see if the connector is attached. If it is not, you will see the message shown in Figure 20-12.

```
          RGB Calibration Editor
     ┌───────────────────────────────┐
     │RGB Lamp not detected.         │
     │Try to communicate anyway?_    │
 Con │                               │
 Con └──────────────────(Y/N)════════┘
 Detector: 0.83    1.25    2.09
   Edit    Sync    Auto           OK
```
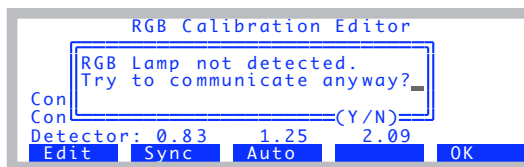
Figure 20-12. Communications problem with the RGB Light Source.

Once the connector is plugged in and the RGB Source powered, the problem (and the message) should go away.

Fan Issues

The RGB source fan operates when it needs to. Here is its logic:

Fan on: IF ($T > 52$ AND $V > 14.0$) OR (ON AND ($T > 30$)).

Fan off: IF ($V < 14.0$) OR OFF OR ($T < 26$)

where ON and OFF refer to the LEDs, T is the heat sink temperature (C), and V is the voltage supply (V). In other words, the fan will run if the heat sink temperature exceeds 52C and the supply voltage is > 14V, or if the LEDs are on and the heat sink temperature gets above 30C. The fan will turn off if the battery is <14V, or if the LEDs are switched off, or if the heat sink temperature drops below 26C.

Temperature T and voltage V can be viewed using the **Query** function key (**f2** level 3) of Home Menu | RGB Control Panel (see **RGB Control Panel** on page 21-9). V can also be viewed in the Light Status window of New Measurements mode (press **[** then **e**).

Heat sink temp, T

```
     Version: 0012
  H.S. Temp: 34.4 C
 Diagnostic: 0030 ( FAN LowBatt)
     Supply: 15.10 V
   Red life: 000002.02 (hrs at 100%)
 Green life: 000003.15
  Blue life: 000003.15
 <escape> exits, any other key rescans_
```

Source voltage, V

```
 (E) RGB: Connect=1   On=1   FB=1
 Status: 15.07V   FAN LowBatt
  umol:     433       433       433      1300
 % Max:    28.7      46.5      45.8      48.7
 ParIn: 1299 umol (1364 mV)
   Cal = 0.95 (Trans=1.00 Offset=0 mV)
 Stable
```

Figure 20-13. Two ways to see RGB Source status information.

Warning Message

If the 6400-18 Light Source is being used, any combination of warning or error conditions can generate a New Measurements mode warning message:

>> RGB Source: FAN / WARM / HOT / 5V / LowBatt / DeadBatt <<

The following words mean the following things:

FAN - The fan is on. (This by itself won't trigger a warning message, but will appear in the list of status items if it is on).

WARM - This is a warning that it is too warm. Specifically the heat sink is warmer than about 70C.

HOT - The heat sink is warmer than 80C, so the LEDs will not be allowed to turn on.

5V - The 5V supply is not in the range of 4.5 to 5.5 V.

LowBatt - The voltage supply is < 15.7 V. This message will only appear if the RGB Source senses that it is connected to a battery. This determination is made when power is first applied. If the detected voltage is 17.25V, it is assumed to be a battery.

DeadBatt - Dead battery indicator. V<14.0V.

Note: The RGB items listed in the New Measurements warning message are also viewable in the two RGB Status screens (Figure 20-13 on page 20-35).

LEDs Switched Off

Check a status display (Figure 20-13 on page 20-35) for the voltage level, or for the `DeadBatt` label. When the supply voltage drops below 14V, the LEDs will switch off. To get them to turn on again, you must first replace the discharged battery, then turn on the light source using the normal method in New Measurements mode (**f5** level 2).

If you aren't using a battery, then verify that all the necessary cables are plugged in.

Doesn't Reach Target

Out of reach?

When operating in Q or T modes (Quantum or Tracking), the light source is color-constrained. That is, it will give highest priority to keeping the color constant, even if it means it can't reach your target. Typically, the maximum red only output is 1300 μmol m^{-2} s^{-1} or more, green is 900, and blue is 1200. So if you are trying to get 2500 μmol m^{-2} s^{-1} with cyan (green + blue, no read), you may not get there.

Transm OK?

There is a multiplier that stands between what the lamp is doing and what the *ParIn_µm* value says. See **The Transm Factor** on page 8-20. Verify that is what you expect it to be for the geometry you are using. A quick way to check it is to look at the Lamp Diagnostics screen in New Measurements mode (press **[** then **E**).

Light at leaf surface

Light output by RGB source

```
(E) RGB Status: Connected=1  On=1  FB=1

  Source:  Red    Green   Blue   Total
  µmol:    1010   1010    1010   3030
% Max:     54.4   51.3    32.1   56.4
ParIn: 1999 µmol (2177 mV)
  Cal = 0.92 (Trans=0.66 Offset=0 mV)
Stable
```
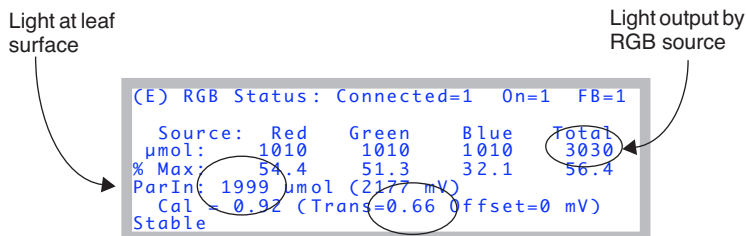
Figure 20-14. The lamp diagnostics screen for the RGB Source.

In Figure 20-14, we see the ParIn value is 2000 μmol m^{-2} s^{-1}, which is our target. But notice that *transm* is 0.66, so the lamp is having to put out 3030 μmol m^{-2} s^{-1} to meet that. (That's a lot, and you may not always get that, especially on a hot day). *Transm* lives in the configuration tree at <open> <light> <source> <par_in> <sensor> <cal> <transm>, and you can adjust it there (Figure 20-15).
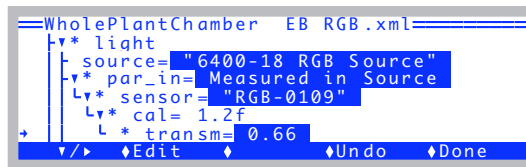
Figure 20-15. The transm *value.*

Gets there slowly?

Start with the lamp off. Then specify a Q target. The parIn_μm value should get to within 1% in the averaging time of the analog measurements (default is 4 seconds). A few seconds after that, it should lock in exactly. If the first guess poor, try re-doing the user calibration (**RGB Calibration Editor** on page 18-34).

Something is Goofy

This is obviously a catch-all, but a good place to spot the root of a myriad of possible problems is in the Calibration Editor (Figure 20-16).

In the Calib Menu, find the Cal Editor...
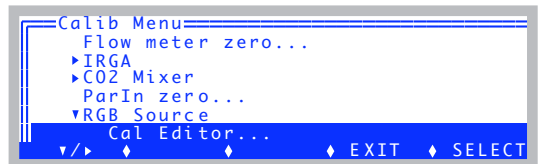
... and make sure the coefficients make sense.
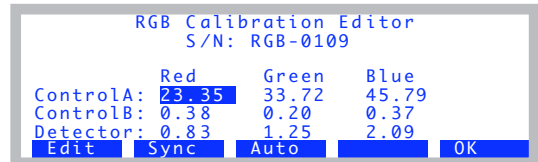
Figure 20-16. The RGB Calibration Editor.

Compare these to the ones on your cal sheet. The top six will change when you run a calibration (**f3**, **Auto**). The bottom three are the factory calibration

values, and they should be as printed. The **Sync** button (**f2**) forces a re-read from the RGB Source.

If you need to re-enter some factory values, position the highlighted box with the arrow keys, and press Edit (f1). If something bad has happened and you need to re-enter everything, including the serial number, here's a hint (and a reward for reading the manual): press **ctrl** + **s** to edit the serial number.

6400-40 Leaf Chamber Fluorometer

See **LCF Troubleshooting** on page 27-82.

Chamber Problems

The Mixing Fan

The chamber mixing fan is shown in Figure 19-33 on page 19-37. It is protected by a fuse (Figure 19-9 on page 19-12), but problems with the fan motor can also cause the flow board fuse to blow.

■ **To verify normal operation**
Use your ears. Turn the fan off and on in New Measurements mode, and you should hear the sound change. (The fan control is via **f1** level 2.)

If the fan isn't behaving normally, then here are some things to check:

1 **Fan motor burned out?**
This typically causes a flow board or fan fuse to blow, but not until the first time New Measurements mode is entered, because that is when the system turns on the fan for the first time.

To verify the motor is burned out, you can measure the impedance across pins 25 and 26 of the 26 pin D connector (Table 20-2 on page 20-52) on the back of the chamber/IRGA. Pins 25 and 26 are on the bottom row, right hand end. It should be about 63 Ohms. If it is 0 Ohms, you need a new motor.

An alternate way to check this is to use the program Control Panel (see **Control Panel** on page 21-12). Power the LI-6400 on, press **escape** to prevent OPEN from loading, access the Filer, select the /Sys/Utility directory, and run Control Panel. Turn on the flow board (item 2), then turn on the fan (item 7 = 5000, then item 6). If that doesn't blow a fuse, try turning on the pump as well (item 9 = 4500, then item 8).

2 **Connector or cable problem?**
Check the 26 pin D connector on the back of the chamber/IRGA to make sure that no pins have been pushed in, or been broken, etc. If possible, try a different cable.

3 **Fan blades jammed?**
(This generally <u>won't</u> cause the fuse to blow.) To check the fan, open the IRGA sample cell (**Cleaning the Optical Bench** on page 19-34) and see if the fan spins freely. If it doesn't, there may be debris wedged under the fan blades, or else the fan is not properly positioned.

A replacement motor and fan is available as part number 6400-902.

■ **Noise caused by the Fan Motor**
The chamber mixing fan inside the sample chamber can cause some unexpected problems in the system when it begins to wear out. The motor for the fan is a dc brush motor, and when the brushes wear down they gradually begin to run roughly on the surface of the commutator. When this happens the brushes begin to make and break contact with the commutator at a very rapid pace. This creates spikes of electrical noise as the motor current is interrupted. These noise spikes emanate from the motor and are coupled into the temperature circuits contained on the chamber head board and the leads of the block, air and leaf sensors.

We have found that this is not a problem until a short time before the brushes are completely worn out, and the motor about to die.

It is easy to test whether the fan motor is causing trouble and to determine the magnitude of the trouble.

The test involves graphing the Tblk and Tair signals on the display and observing the effects of turning the fan on and off. Follow this procedure to do the test:

1   **Turn the chamber mixing fan off.**

2   **Setup graphs**
In New Measurements mode, view the graphs by pressing the right bracket ( ]) on the keyboard. Now press the letter D to show temperature graphs. Tleaf is also included, but we are primarily interested in the effects on Tair and Tblk.

3   **Watch the graphs to get a baseline.**
If the LI-6400 has been turned on for 30 minutes or so, you can expect the line to travel across the screen showing little or no change in temperature. If the instrument has just been turned on you will see the lines creeping upwards as the head reaches temperature equilibrium.

With the fan off, there should not be noise on these signals. If you see it vary up and down by one pixel (Y axis = 1 degree C) it means that the A-D converter is toggling between one value and the next. This is ok.

4   **Turn the fan on.**
Leave the graphing mode and by pressing **escape**. Turn the chamber mixing fan back on and then immediately return to viewing the graph to see the effects.

When the fan is turned back on, if there are no problems, you should not see any fast change in the line on the graph. Now, it *is* normal to see the temperatures (especially Tair) begin a slow trend upwards or downward. This is a real physical temperature change being measured.

If there is a problem caused by the fan, you will see varying degrees of noise on the signal begin to appear. It can range from 2 pixels on the screen to one or two degrees of temperature. An offset may also be observed. This shows up as a very rapid change to another level and is always accompanied by noise showing up as a jumpy line on the graph. The offset is not always present.

The Y axis will automatically adjust to contain the data, regardless of the magnitude of change. You will see the low and high readings to the side of the graph box on the screen change. So be aware that if it changes significantly, you will be seeing a bigger range on the Y axis, and it may not look as noisy as it really is. Pressing the number 3 will reveal a function key labeled **Clear**. This wipes out all data and it will begin to show the graph using the smallest possible range for the Y axis.

If you see the offset or the noise appear, the fan needs to be replaced. The part number for the field-installable mixing fan kit is 6400-902.

The effects of this noise show up in all calculations that use the block or air temperatures. Also the leaf temperature, as it is referenced to the block temperature. We don't recommend basing your judgment of the state of the fan's brushes on the Leaf temperature, because unless the thermocouple is in good contact with a surface of sufficient mass, the readings tend to be a bit noisy anyway. If the thermocouple bead is sitting in the air, you will observe small fluctuations, especially if wind is passing across it.

The effects of fan motor noise can vary from none to so bad that the CO_2 injector system cannot control a setpoint because the calculated CO_2 concentrations are jumping too much.

This chamber mixing fan problem can also show itself in the inability to zero the leaf thermocouple. This is due to the offset caused by the fan noise being larger than the range of voltage change that the leaf thermocouple adjustment potentiometer on the bottom of the sensor head can produce. To test if the fan is the cause of inability to zero the leaf thermocouple, try the zero procedure with the fan turned off. If it will zero with the fan off, but not with the fan on, the fan is bad and needs to be replaced.

It is good practice to do the graphing test described above periodically to that you will know when the fan is nearing the end of its useful life. The noise generally begins very small and increases in severity as the brush-commutator surfaces erode. It is good to catch it early so you can order the field installable fan kit, or send it to LI-COR for repair. It can save you taking noisy data and having to re-do your work.

Bad Temperature Readings

■ **Erroneous Block Temperature**
If the block temperature reads erroneously, such as -50C, there are a couple of things to check:

1 **Check the cable and connectors**
Is the chamber cable plugged in? Check the 26 pin D connector on the back of the chamber/IRGA, in particular pins 22 and 10 (seeTable 20-2 on page 20-52); is the pin pushed in? Check the cable, and for a possible break between pin 22 (on the IRGA end) and pin 1 (on the console end), and between 10 and 3.

2 **Check for a pinched wire**
Check the metal cover (with the serial number label on it) beneath the IRGA (Figure 20-17) and the wires that come out from beneath it. Something may be pinched.
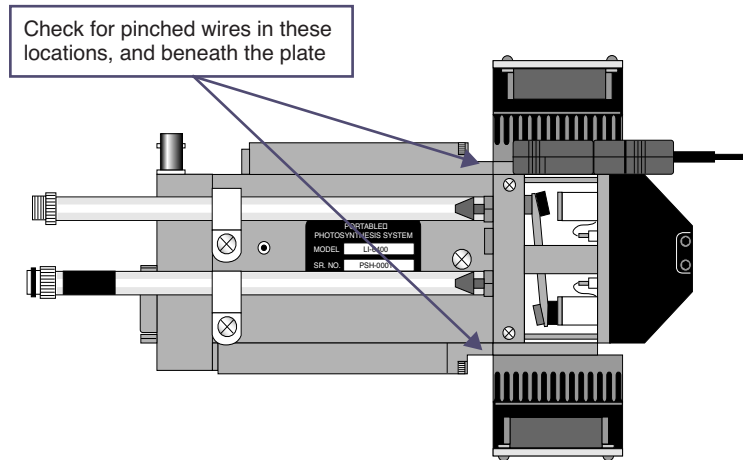
Figure 20-17. Check this metal cover to be sure that it is not pinching any of the wires that come out from beneath it, especially near the match valve.

■ **Erroneous Leaf Temperature**
Leaf temperature is referenced to block temperature. Therefore, if leaf temperature is reading a strange number, the first thing to check is the block temperature - maybe it is the problem. Otherwise...

1 **Is the thermocouple broken?**
If leaf temperature doesn't respond to touch, and always reads the same as block temperature, that is a pretty good indication that the leaf temperature thermocouple is broken.

2 **Check the cable and connectors.**
Leaf temperature signal is carried between pins 14 (chamber end) and 4 (console end).

3 **Is the chamber fan working?**
Sometimes your first clue that the chamber fan is not operating is a leaf temperature sensor that is a few degrees off from where you think it should be.

■ **Noisy (Erratic) Temperatures**
See **Noise caused by the Fan Motor** on page 20-40.

■ **Can't Zero Leaf Temperature**
See **Noise caused by the Fan Motor** on page 20-40.

Finding Leaks

Leaks will get your attention by causing unstable CO_2 readings that you might first suspect are IRGA problems.

■ **Sensor head or console?**
If the reference cell concentration appears to be stable, but the sample cell concentration is unstable, it suggests a leak somewhere in the sensor head. If both are unstable, it suggests a leak in the console.

Sensor Head Leaks

If the reference is stable, try a leak test on the tightly closed chamber: Exhale on and around the leaf chamber, and see if the sample CO_2 responds. You should not see more than a 1 or 2 ppm rise in CO_2 in a properly sealed chamber. There will always be some effect of breathing near the gaskets due to CO_2 diffusion through the gasket material, but it is delayed and fairly small.

If you see rises in excess of 5 or 10 μmol mol^{-1}, you have a leak that should be fixed. To isolate the leak, you might try blowing through a small straw at selected places. Hints:

• **O-rings**
Make sure that all 6 O-rings are in place (between the chamber halves and the IRGA manifold), clean, and *lightly* lubricated.

• **Cap screws**
Another possible leak source is the screw indicated in Figure 20-18.

• **Gaskets**
Are they centered? Is there a tiny wrinkle beneath that acts like a channel? Don't forget the 3-hole rear gasket.

Top View

Gasket

Lower leaf☐
chamber cuvette

This screw must be
snug, or it will leak.
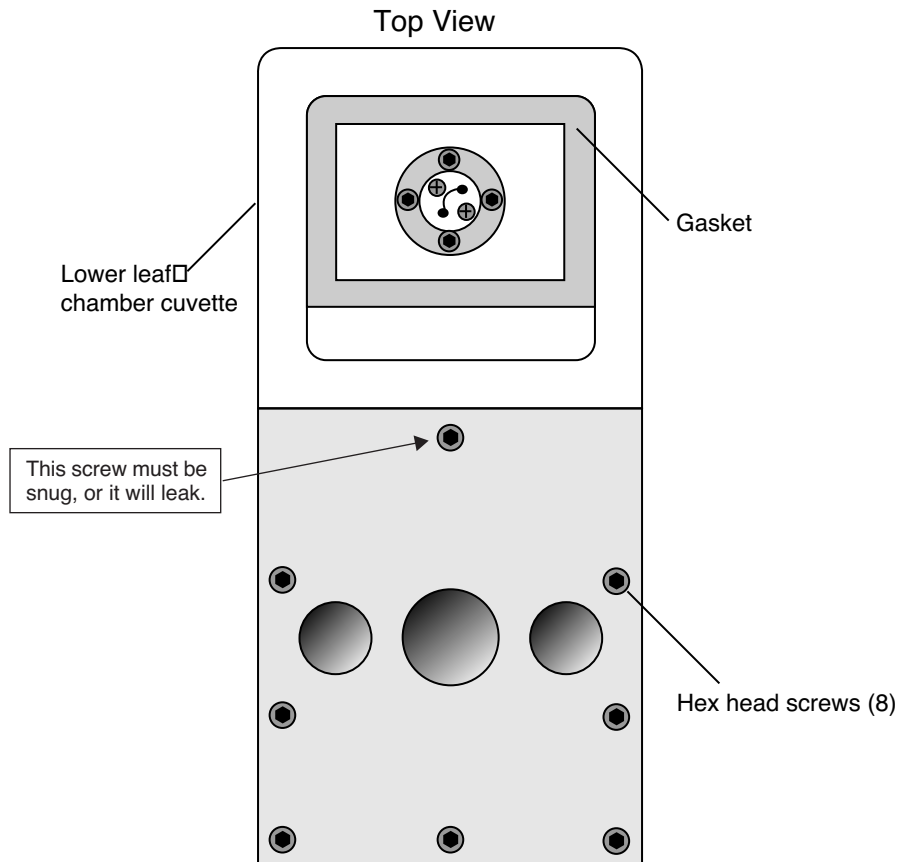
Hex head screws (8)

Figure 20-18. The eight hex head screws must be snug to prevent leaks. The top screw is extra important, however, because it passes through one of the air passages in the manifold, and there is just a metal-to-metal seal beneath the screw head.

Hose Leaks

It's not common, but once in a while we find leaks in a hose connector, due to the O-ring on the connector being dried and worn or cracked. Lubricate with silicon grease.

Console Leaks

It is helpful to have a simple flow schematic (Figure 20-19) in mind when tracking down a leak in the console. Any leak on the vacuum side of the pump will cause ambient air to be sucked into the system, so breathing on the chemical tubes, or into the case, will cause rapid rises in CO_2 a few seconds later in both IRGAs. (This test is best done with the CO_2 scrub ON, so that any of your breath that makes it into the *normal* air inlet will be scrubbed)

Without CO_2 Mixer
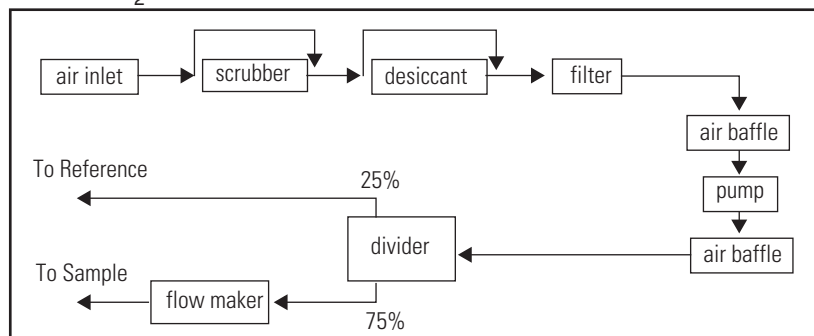
With CO_2 Mixer
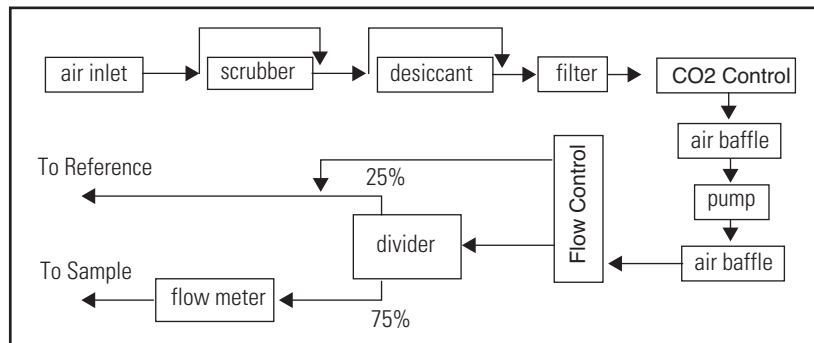
Figure 20-19. Flow schematic with and without the 6400-01 CO_2 Mixer. A more complete schematic is found in Figure 20-23 on page 20-51.

Chemical Tubes

The chemical tubes are common sources of leaks, and leak-like behavior.

- **Bad Soda Lime**

  As soda lime wears out, it begins letting some ambient CO_2 into the system.

- **Orientation**

  If the console sitting on its side, rather than sitting upright, a gap can occur along the length of the chemical tube at the top, creating a channel with reduced CO_2 removal.

- **The Air Passage O-rings**

  There is a small O-ring around each air passage tube to the console (Figure 2-1 on page 2-3). Are they there? Are they clean and lightly lubricated? Are the chemical tubes fully seated against the console? Don't overtighten the attachment screw, or you'll never get it back off without tools. Lightly snug is sufficient.

- **The End Cap O-rings**

  There is a large O-ring on each end cap that should be compressed slightly when the end cap is tightened on. If there is a gap, it's probably because the threads are dirty. Clean the threads with water if necessary, and keep them clean each time you change chemicals. See **Cleaning The End Cap Threads** on page 19-3.

Hose Barbs

Internally, there are a number of single flute hose barbs; visually inspect all hose barbs and check for tightness. Although the tubing might rotate freely[4] on the hose barb flute, the hose barb itself must be threaded tightly into its threaded mounting hole. If you can't turn a hose barb with your fingers, then it is *probably* tight enough to not leak. Hose barbs can be tightened with a 1/4 inch end wrench or (in a pinch) needle nose pliers. They have rubber gaskets, so if they physically touch the surface they screw into, that might be tight enough. *Be careful not to overtighten; hose barbs can break*. Note that there are also four hose barbs on the block to which the chemical tubes are attached.

Quick Connectors

Inside the quick connector is a rubber seal, similar to an O-ring. When a hose is inserted into a quick connector, it pushes in rather easily until encountering the internal clamping device; it must then be pushed an <u>additional</u> 0.25 inch (0.5 cm) to make contact with the rubber seal. Variations in the outside diameter of the Bev-a-line tubing can make the insertion of the tubing into the quick connectors difficult; in such cases, improperly inserted tubing may work loose and cause a leak. Hint: Wet the end of the tubing before insertion. The red collet on the quick connector is for hose removal: press the collet in towards the center of the connector before pulling on the hose to remove it. If you can easily pull a hose out of a quick connector without depressing the red collet, it means that the hose was not inserted far enough. The clamp ring

---

[4.]Actually, if it does rotate freely, it could be leaking.

leaves a mark on the end of the hose; if this mark is 1/4" from the end, the hose was inserted correctly.
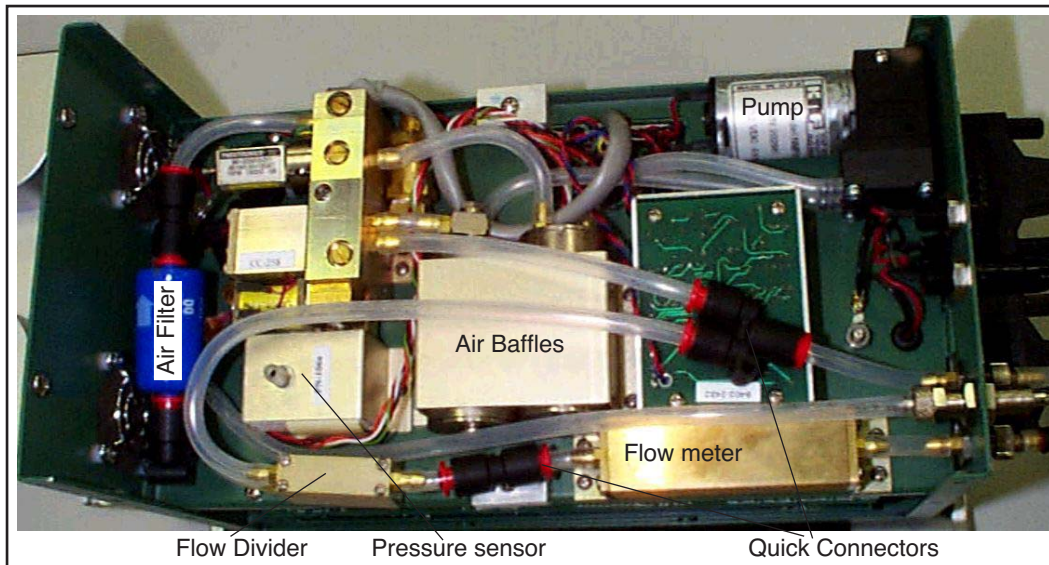
Figure 20-20. Bottom view of the console with the cover removed.

Air Baffles

One of the connections to one of the baffles has a 90° silver elbow fitting and a hose barb. There is an internal O-ring in the silver elbow between the body of the elbow and the air baffle, and another on the other side of the body of the elbow, under the part of the elbow that looks like a nut.

Pump

Make sure there is a good seal on the inlet hose barb of the pump. Inside the pump is a diaphragm and two flapper valves, each sealed with an O-ring. The replacement/repair kit for the pump is part number, 6400-907. If you disassemble the pump, scratch a line down one side so that it can be reassembled in the correct orientation.

Flow Divider

Inside the body of the divider (Figure 20-21) are four flow restrictors through which all the flow passes in parallel. Check the divider assembly screws for tightness. Be careful, as they are small (2-56 thread) and can break. If you disassemble the divider to check the 8 internal O-rings (2 for each flow restric-

tor), put a pencil mark on the outside of the two halves so that it can be reassembled correctly.
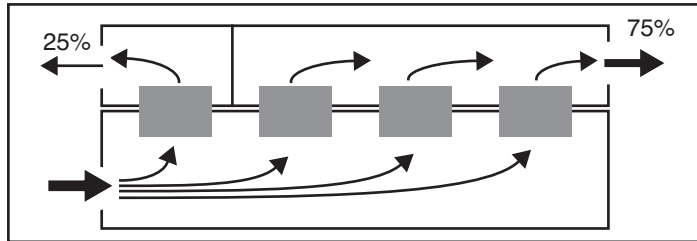
Figure 20-21. Schematic representation of the LI-6400's flow divider, that partitions flow to the sample and reference cells of the IRGA.

Soil Chamber Problems

See **Troubleshooting the Soil Chamber** on page 28-39.

Useful Information

The Diagnostic Text Display

(This display configuration is still around for historical purposes. Don't confuse this display map with the Diagnostic Screens A through G.)

One of OPEN's default displays is named Diagnostic (Figure 20-22). This display can be useful when trouble shooting, as it contains all of the raw input signals, in addition to the standard computed variables.

To enable this display, **Quik Pik** (**f1** level 6) in New Measurements mode.

a	CO2R_µml	CO2S_µml	H2OR_mml	H2OS_mml
b	CO2R_mv	CO2S_mv	H2OR_mv	H2OS_mv
c	CO2 H2O Pump Flow Mixr Fan			
d	Tblock°C	Tair°C	Tleaf°C	Tirga°C
e	Tblk_mv	Tair_mv	Tleaf_mv	Tirga_mv
f	Flow_µml	Prss_kPa	ParIn_µm	ParOutµm
g	flow_mv	press_mv	ParIn_mv	ParOutmv
h	CRagc_mv	CSagc_mv	HRagc_mv	HSagc_mv
i	uc_20_mV	uc_21_mv	uc_22_mV	uc_23_mV
j	HH:MM:SS	Battery		

Figure 20-22. The diagnostic display map for New Measurements mode.

The quantities in levels *a*, *c*, *d*, *f*, and *j* are common system variables, also found in the standard display list. Levels *b*, *e*, and *g* contain the raw signals (mV) for the sensors.

User Channel Voltages

The spare analog input channels are shown in level *i*. These will show 0 mV if they have not been enabled via the 'UserChan=' configuration command.

System Flow Schematic

Figure 20-23 provides a schematic of the flow components of the LI-6400.
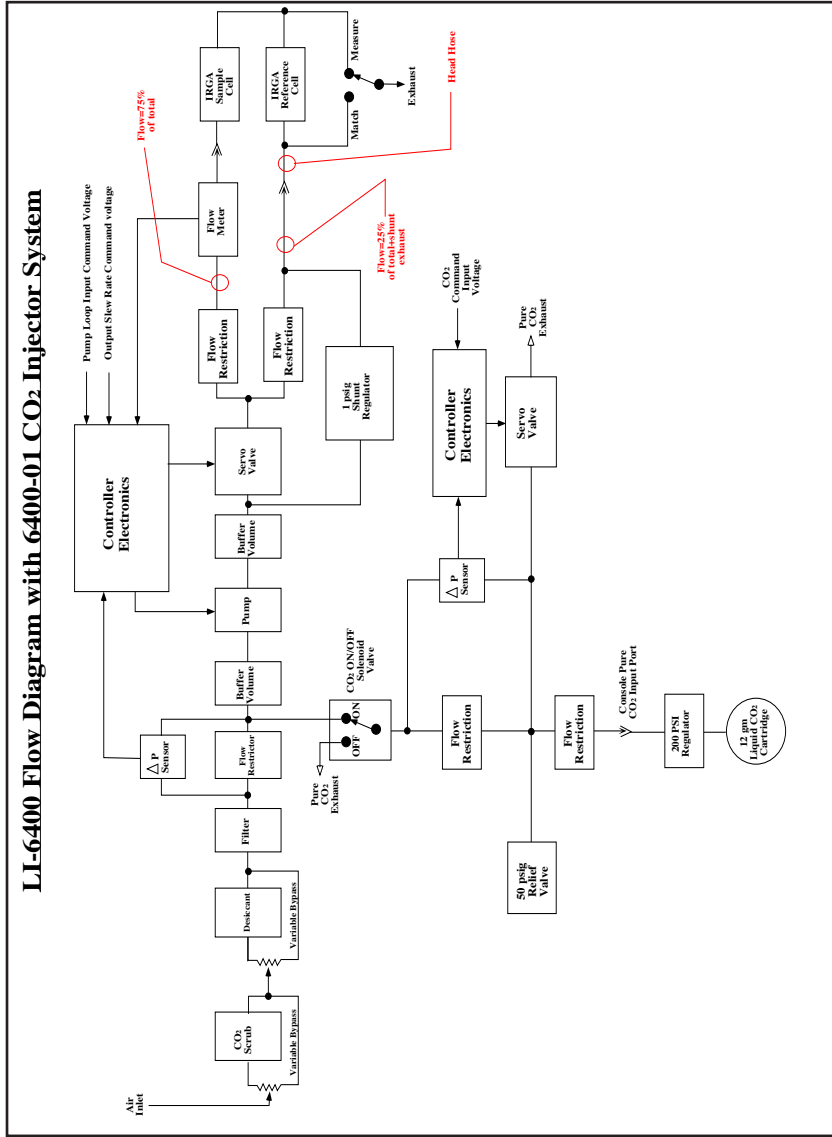
Figure 20-23. LI-6400 system flow schematic, with CO₂ mixer option.

Chamber Connectors

The chamber cable has a 25 pin D connector on the console end, and a 26 pin D connector on the chamber end (Table 20-2).

Table 20-2. Chamber connector cable pin descriptions

Description	Connector		Comments
	26 Pin D Chamber	**25 Pin D Console**	
BlockTemp1	22	1	
BlockTemp2	10	3	
AirTemp1	21	14	
AirTemp2	11	16	
Match+	23	2	
Match-	24	15	
Circ fan +	26	19	A normal fan motor has an impedance of 63 Ohms.
Circ fan -	25	7	
Tleaf	14	4	
Log	1	25	
+10V	7	11	
-9V	6	24	
Signal Gnd	8	12	
Power Gnd	9	13	
PAROut	18	5	
PARIn	15	17	
TEC[a]+	4, 5	10, 23	
TEC-	2, 3	9, 22	
+12V	17	8	
TEC Fan	13	21	
Lamp+	19	18	Warning: voltages can exceed 100V
Lamp-	20	6	
Lamp Fan	16	20	

a.TEC stands for thermoelectric cooler.

Diagnostics and Utilities

21

Useful programs

21 *Diagnostics and Utilities*

This chapter documents the diagnostics and utility programs that are available on the LI-6400.

Diagnostics & Tests Menu

These programs are found in the Diagnostics and Tests Menu, accessed via OPEN's Home Menu.

Auxiliary DAC Test

This program tests four auxiliary DAC outputs (channels 10, 17, 18, and 19) and four user input channels (channels 20, 21, 22, and 23). It requires the proper pins to be connected on the 37-pin connector on the console.

DAC Outputs   Analog Inputs

```
        Aux channel and dac test

Pin  9 1000mV -> Pin 36   995    ok
Pin 29 2000mV -> Pin 19  2040    ok
Pin 11 3000mV -> Pin 18     0    ERROR
Pin 30 4000mV -> Pin 37     0    ERROR
Pin 13 -> Pin 32 (Ground)
Any key changes mV, or <esc> quits
```
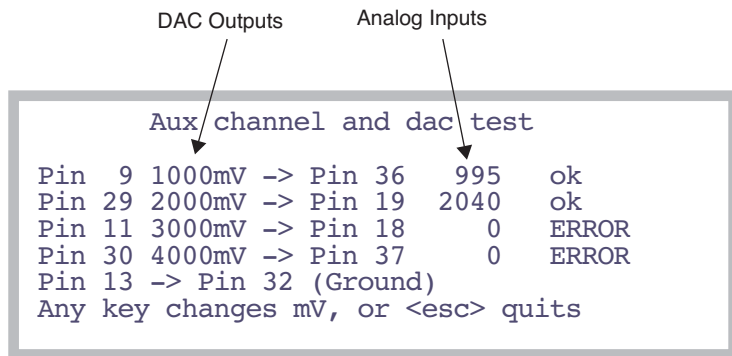
Figure 21-1. The Auxiliary DAC Test display. Press any key to change the mV settings on the DACs. The change cycles between 1-2-3-4 V, 4-3-2-1 V, and 0-0-0-0 V. If the DAC output and the analog input are within 50mV, then the status will show "ok". Otherwise, it will show "ERROR".

CO2 Mixer Test

This program is a performance tester of the 6400-01 CO_2 Mixer. It takes the mixer through a series of test points, and records the time it took the mixer to achieve stability at each point.

The test will use pre-defined points, or ones of your choosing which can be read from a file or entered from the keyboard (Figure 21-2).

```
This program runs a diagnostic
test on the CO2 Mixer.

OK to Continue? (Y/N)_
```

```
This pr
test on  Select Test
           S - Std test
OK to C   K - from Keyboard
          F - from file_
```

Figure 21-2. Opening prompts of the CO_2 Mixer Test program.

The data to be specified are pairs of set points and wait times. The set points are between 0 and 5000 mV, and the wait times (seconds) are the maximum time the program will wait for stability. Stability has two conditions:

1) Mixer status is OK, and

2) Range of values over the past 5 seconds is less than 1 μmol mol^{-1}.

Standard Test

The standard test specifies the following combination of set points and maximum wait times:

Table 21-1. Standard test sequence for the mixer.

Set Point (mV)	Wait Time (s)	Set Point (mV)	Wait Time (s)
0	60	4000	90
500	90	4900	90
1000	90	1500	60
2000	90	200	20

Table 21-1. Standard test sequence for the mixer.

Set Point (mV)	Wait Time (s)	Set Point (mV)	Wait Time (s)
3000	90		

From Keyboard
Enter a series of set points and cut off times in the window provided.

```
══ Each line: SetPt(mV) cutoffTime(s) ══
0 60
1000 120
5000 120
1000 40_
```

After entering the data, press escape. You will have an opportunity to store your entries in a file for use again, by selecting the "F - from File" option.

From File
Any file that has pairs of numbers that can be interpreted as set point and wait time can be used. The file is selected using the Standard File dialog.

Once the input data is established, you are prompted to prepare the chemical tubes (Figure 21-3).

```
Please set the scrub tubes:

SODA LIME: FULL SCRUB
DESICCANT: FULL BYPASS

Then press enter_
```

Figure 21-3. Just before the test starts.

Once you press **enter**, the test begins, and the display will remind you to leave the chemical tubes as set (Figure 21-4).

The first point in the Standard Test has the mixer turned off.

```
Test in progress. Leave
SODA LIME: FULL SCRUB
DESICCANT: FULL BYPASS
CO2R_μml    Status  Flow_μml  Rangeμml
     0.9      off      189.9     15.46
Setting 0mV        Time:15

  earlyOK                        Abort
```
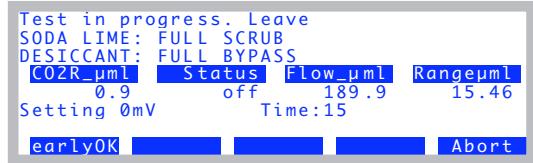
19 seconds in to the second point.

```
Test in progress. Leave
SODA LIME: FULL SCRUB
DESICCANT: FULL BYPASS
CO2R_μml    Status  Flow_μml  Rangeμml
    84.7      OK       189.9     10.91
Setting 500mV      Time:19

  earlyOK                        Abort
```
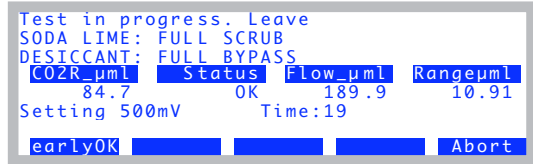
Figure 21-4. During the test.

The **earlyOK** key will quit the current set point, and continue with the next. **Abort** will terminate the program.

When the test is over, the results are shown (Figure 21-5). After viewing the results, press **escape**. You are given the option of storing the results.

```
==========Mixer Test Results==========
 SetPt time  ppm range status
 ===== ====  === ====================Mixer Test Results==========
     0   18  SetPt time              atus
   500   36  ===== ====  Store this  ====
  1000   88      0   18              off
  2000   60    500   36  ====(Y/N)===  OK
   Print ♦ Fi 1000   88  246  0.90     OK
             2000   60  612  0.86     OK
              Print ♦ Find  ♦ ReFind♦ JumpTo♦ OK
```
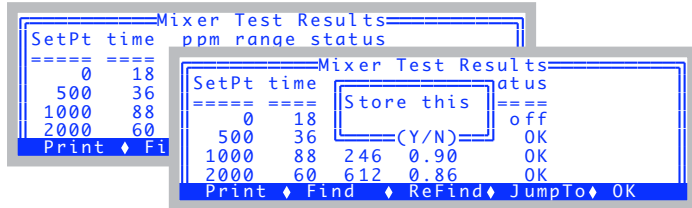
Figure 21-5. Presentation of the results.

If you elect to store the results, the file will look something like this:

```
SetPt time  ppm range status
===== ==== ==== ===== ======
    0   18    0  0.64    off
  500   36   90  0.99     OK
 1000   88  246  0.90     OK
 2000   60  612  0.86     OK
 3000   65 1072  0.98     OK
 4000   91 1629  1.44     OK
 4900   91 2015 14.06    LOW
 1500   20  427  0.89     OK
  200   17   44  0.87     OK
```

DAC Status

The DAC Status program presents the state of the 20 D/A convertors on the LI-6400. Most of the channels are controlled by OPEN, but a 6 are available for user. See **Analog Input Channels** on page 16-29.

```
════════ DAC Status ════════

PORT        mV   Use
════    ════════   ════════════
   0      1321   Flow Set
   1    0.6205   CO2 Set
   2    0.6205   Lamp
   3    0.6205   (pin 12)
   4      1993   Chamber Temp
   5       3.4   unavailable
   6      4491   Pump
   7      4978   Leaf Fan
   8       3.4   (pin 10)
   9       3.4   (pin 27)
  10       3.4   (pin 9)
  11      2150   LCD Contrast
  12     -1093   CO2R_Zero
  13     -2108   CO2S_Zero
  14     -1405   H2OR_Zero
  15     -1834   H2OS_Zero
  16      2652   Flow_zero
  17   -0.9447   (pin 29)
  18   -0.9447   (pin 11)
  19   -0.9447   (pin 18)
```

Figure 21-6. The list produced by the DAC Status program shows each D/A channel and its current value. The channels described by pin n *are available for user use. For range and resolution information on these spare channels, see* **Analog Output Channels** *on page 16-32. (What's the story about port #5? Well, it somehow never made it out to the 37 pin connector, so is unavailable for use. Our mistake.)*

Digital Status

The Digital Status program presents the state of the LI-6400's digital I/O ports.

```
P:01234567 status direction
0:10110011    2      [OUT]
1:01001111    5      [IN ]
2:1111        5      [IN ]
3:01100000    2      [OUT]
4:11000000    2      [OUT]
```

Figure 21-7. Digital Status display.

The display is live, so a change in one of the digital inputs will show up on the screen within a second or two. If the log button is closed, for example, it should make pin 5 port 1 change from 1 to 0.

Press any key (except **shift** or **ctrl**) to terminate the program.

Table 21-2. LI-6400 digital port and pin assignments

Port	Pins 0	1	2	3	4	5	6	7	Status[a]
0	Mixer on/off	RH Control enable	Flow range 1	Flow range 2	TEC on/off	Lamp on/off	Chamber fan on/off	Match valve[b]	Output
1	Pump status	Mixer Hi	Mixer Lo	Flow Hi	Flow Lo	Log Button	37Pin #4	37Pin #22	Input
2	H2O Ref	H2O Smp	CO2 Ref	CO2 Smp					Input
3	Pump	IRGA board	Flow board	CO2 solenoid					Output
4[c]	37Pin #23	37Pin #5	37Pin #24	37Pin #6	37Pin #25	37Pin #7	37Pin #26	37Pin #8	Output

a. In version 5 and 6, no port can be switched from between input and output.

b. This signal is also available on pin 21 of the 37 pin external connector (5V = match off, 0V = match on)

c. All of the pins of port 4 are available for user use. See **Digital Output** on page 16-33.

LCF Control Panel

This program is for the 6400-40 Leaf Chamber Fluorometer, and is described in **LCF Control Panel** on page 27-82. It only appears under Diagnostics and Tests when the LI-6400 is configured for the LCF.

Log Button Tester

The Log Button Tester program indicates if the log button is up or down. Pressing any key (except **shift** or **ctrl**) will terminate the program.
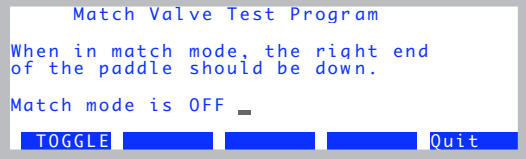
```
Log Button is up  _
```

Figure 21-8. Log Button Tester display.

Match Valve Tester

This program manually moves the match valve. Note that when you leave this program, the match valve returns to the state it was in when the program was entered.

```
      Match Valve Test Program

When in match mode, the right end
of the paddle should be down.

Match mode is OFF _
  TOGGLE                        Quit
```

Figure 21-9. The Match Valve manual control program. The status line indicates where the valve should be, not necessarily where it actually is.

Pressure Sensor

The pressure sensor test program merely runs the program **Geopotential** on page 21-15.

RGB Control Panel

This program appears in the Diagnostics & Tests menu when OPEN is configure for the 6400-18 RGB Light Source. Its main screen is shown in Figure 21-10.

```
        RGB Source Control Panel
RGB: Connected=1  On=1  FB=1
   Source:    Red   Green    Blue    Total
 µmol:        167     167     167      500
% Max:       9.31    7.08    5.07     9.31
ParIn = 500 umol (359 mV)
   Cal = 1.39 (Trans=1.00 Offset=0 mV)
1Config
2 Cont.   Random    Flash
3 Synch    Query            Calib
```

Figure 21-10. The RGB Control Panel program.

Config (**f1**): Shows the configuration in a tree structure, that you can examine and in some cases edit (Figure 21-11).

```
      ══RGB Settings═══════════════
RGB: │  ┌►Control                   otal
  So │  ├►Calibration               500
 µmo │  └►Status= Rescan...         .31
% Ma │                             
ParI │                             
  Ca │                            V)
   ▼/►      ♦      ♦      ♦    ♦Exit
```

Figure 21-11. RGB Settings tree view.

Cont. (**f1** level 2): Displays continuously changing colors: off → white → red → yellow → green → cyan → blue → magenta → red → white etc. until you press any key to stop (Figure 21-12).

```
Continuous. Press any key to stop
RGB: Connected=1  On=1  FB=1
   Source:    Red   Green    Blue    Total
 µmol:        163     171     166      500
% Max:      0.711   0.516    0.37     9.31
ParIn = 500 umol (359 mV)
   Cal = 1.39 (Trans=1.00 Offset=0 mV)
off->White
```
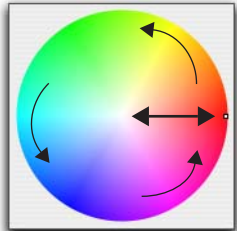
Figure 21-12. Continuous color demonstration.

Random (**f2** level 2): Displays random colors and intensities until you press a key (Figure 21-13).

```
Random Colors. Press any key to stop
RGB: Connected=1  On=1  FB=1
  Source:    Red    Green    Blue    Total
  µmol:       80     256     144      479
% Max:     0.462    1.04   0.426    9.31
ParIn = 479 umol (344 mV)
  Cal = 1.39 (Trans=1.00 Offset=0 mV)
```

Figure 21-13. The Random color program.

Flash (**f3** level 2): Rapidly repeats the sequence of off → add red → add green → add blue → off, and your choice of three intensities (Figure 21-14).

```
Flash. H M L or <esc> quits

HI med low
```

*Figure 21-14. The Flash program. To vary intensity, press **H**, **M**, or **L**.*

Synch (**f1** level 3): **Synch** causes the console to 'synchronize' with the light source (Figure 21-15). It reads the serial number and calibration values from the unit. Each byte received is displayed in the little RGB Comm window that goes up on the display.

```
       RGB Source Control Panel
RGB:┌────────RGB Comm────────┐
  So│30 31 30 39 00 00 20 46 00 │otal
  µmo│                          │500
% Ma│                          │.31
ParI└──────────────────────────┘
  Cal = 1.39 (Trans=1.00 Offset=0 mV)
```

Figure 21-15. Synchronizing with the light source.

Query (**f2** level 3):Query reads some secondary status information from the light source (Figure 21-16).

```
    Version: 0012
 H.S. Temp: 32.6 C
Diagnostic: 0020 ( FAN)
    Supply: 16.08 V
  Red life: 000002.06 (hrs at 100%)
Green life: 000003.22
 Blue life: 000003.22
<escape> exits, any other key rescans_
```

Figure 21-16. The Query display.

<u>Version</u> is the software version of the RGB source's embedded programming. <u>H.S. Temp</u> is temperature of the heat sink. The bits of the <u>diagnostic</u> value each have their own meaning, but it is decoded for you. Thus, bit 5 (0x20) means the fan is running. <u>Supply</u> is the voltage of the source's power input. Below 14.0, the LEDs will shut off. The <u>life times</u> for red, green, and blue are the approximate equivalent hours they have operated at 100% power. 1 second at full power is equivalent to 2 seconds at half power, etc. We do not register aging at levels below 1/8 of full power[1]. The largest value you can achieve for the lifetime of one of the colors is 149130.81, at which time it will roll over and restart at 0 (or bring an end to civilization as we know it).

Calib (**f4** level 3): Runs the calibration routine discussed in **RGB Calibration Editor** on page 18-34.

/Sys/Utility Programs

Programs in the directory "/Sys/Utility" are intended to be general purpose, and do not require OPEN to be running.

BoardsOff

This program will turn off the analyzer board and flow control board. A listing of this program is shown in Figure 22-17 on page 22-27.

```
This program will power OFF the flow
board and the IRGA board.

OK ? (Y/N)
```

[1.]It's like having a car whose odometer doesn't turn if you drive less than 10 mph.

BoardsOn

This program will turn on the analyzer board and flow control board.

```
This sprogram will power ON the flow
board and the IRGA board.

OK ? (Y/N)
```

ConnectToLicorServer

This program is described in **Connection over the Internet** on page 11-38.

Control Panel

This program allows direct access to most of the analog and digital outputs of the LI-6400 (Figure 21-17). The complete list is presented in Table 21-3.
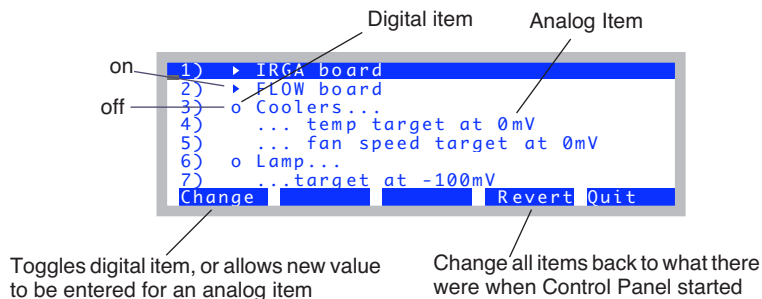
Figure 21-17. The program Control Panel presents a number of analog and digital outputs in a list.

Table 21-3. Items available for viewing/setting in Control Panel.

#	Label	Description and Comments
1	IRGA Board	Turns analyzer board off (o) and on (▲).
2	FLOW board	Must be on (▲) for all other items (3-29) to function correctly.
3	Coolers…	Turns chamber coolers off (o) and on (▲).
4	…temp target	Chamber cooler target value (0-5000 mV). (°C = mV/100, so 2500mV = 25°C)
5	…fan target	Chamber fan speed (0-5000 mV). Normal = 5000.

Table 21-3. (Continued)Items available for viewing/setting in Control Panel.

#	Label	Description and Comments
6	Lamp…	Light source off (o) and on (▲).
7	…target	Light source target (0 - 5000 mV).
8	Pump…	Pump off (o) and on (▲).
9	…target	Pump speed (0 - 5000mV). Typical = 4500.
10	CO2 Mixer Electronics	Mixer electronics off (o) and on (▲). It is normally on all the time.
11	CO2 Mixer Solenoid	Mixer solenoid off (o) and on (▲). (This is what the mixer on/off button in OPEN controls.)
12	…target	Mixer target (0 - 5000 mV).
13	Circulation fan…	Leaf chamber fan off (o) and on (▲).
14	…target	Leaf fan speed (0 - 5000 mV). Fast is usually 5000, slow is usually 4000.
15	Match valve	Match valve on (o) and off (▲). Note reverse logic.
16	User Digital: Pin 23 (0x0400)	
17	User Digital: Pin 5 (0x0401)	
18	User Digital: Pin 24 (0x0402)	
19	User Digital: Pin 6 (0x0403)	Digital outputs available on the 37 pin console connector. LPL address shown in (). See **Digital Output** on page 16-33. Off is (o), on is (▲).
20	User Digital: Pin 25 (0x0404)	
21	User Digital: Pin 7 (0x0405)	
22	User Digital: Pin 26 (0x0406)	
23	User Digital: Pin 8 (0x0407)	
24	Analog Output: Pin 12 (3)	
25	Analog Output: Pin 27 (9)	
26	Analog Output: Pin 9 (10)	Analog outputs available on the 37 pin connector. LPL address shown in (). See **Analog Output Channels** on page 16-32.
27	Analog Output: Pin 29 (17)	
28	Analog Output: Pin 11 (18)	
29	Analog Output: Pin 30 (19)	

ENERGYBAL

This program is useful when measuring chamber boundary layer conductances with wet filter paper. The problem with making these measurement is that the leaf temperature will be wrong. There is such a strong gradient between leaf (wet filter paper) and air, that conduction errors will always cause the measured leaf temperature to be too warm. This program will compute the

leaf temperature based on four inputs: pressure (kPa), vapor pressure (kPa), transpiration rate (mmol mol^{-2} s^{-1}), and air temperature (C). It then prints out a computed boundary layer conductance (mol m^{-2} s^{-1}) and computed leaf temperature (C). For a one-sided value, divide the displayed boundary layer conductance value by 2 (this assumes that the leaf area used in the transpiration rate was a one-sided value).

For more information on how the LI-6400 uses boundary layer conductances, see **Boundary Layer Variables** on page 14-20.

```
Enter: P_kPa vap_kPa Trans_mmol Tair
90.73 1.45 13.7 24.97
BLC = 2.75 Tleaf = 16.6

Enter: P_kPa vap_kPa Trans_mmol Tair
▄

 DelLn   ♦ClrEnd ♦DelChar♦CapLock♦AnyChar
```

Figure 21-18. "ENERGYBAL" *takes four inputs, and computes boundary layer conductance and leaf temperature. It will prompt repeatedly; press* **enter** *with no inputs to terminate it.*

The theoretical basis of this program is described in **Measuring Boundary Layer** on page 17-9.

Geopotential

This program displays real time values from the pressure sensor, and also geopotential height. The latter is the height above sea level in an idealized atmosphere. Real weather will make your indicated height vary from day to day. An interesting feature is that this program can demonstrate the sensitivity of the pressure sensor. Lift the console very slowly, and watch the altitude (in feet) change as the console's height changes. Press **escape** to terminate the program.

```
              Pressure:      97.04 kPa
Geopotential Height:      1205 ft
                           367 m
```

Figure 21-19. The Geopotential program.

Graphics Restore

Note: Graphics Restore is the program that is run when you select "View Stored Graphics Images" from OPEN's Utility Menu.

This program uses the Standard File Dialog (page 5-9) to select files to be loaded back onto the graphics display. These files should be those that were created using **ctrl** + **s**.

QuickPlot (all other graphics)

Real Time Graphics (Strip charts in New Measurements mode)

```
Dir: /User/Images
========Read Graphics from...======
 QP_2008-11-03_16-03-31
 RTG_2007-11-08_17-53-29
 RTG_2008-08-26_13-48-05
 RTG_2008-08-27_09-05-07
 RTG_2008-10-24_12-09-40
+ Dir    ♦          ♦        ♦ CANCEL♦ SELECT
```

```
CO2R                    CO2S
  507.3                   556.7




 -2.0   MINS    0.0  -2.0   MINS   0.0
```
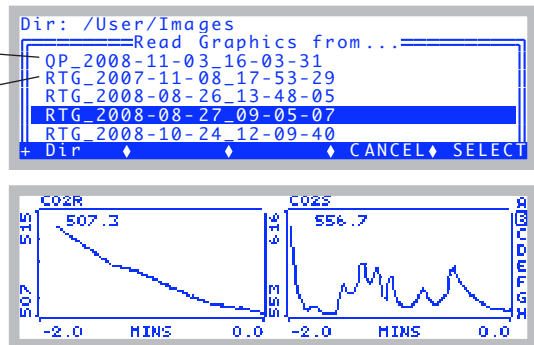
*Figure 21-20. Graphics images stored by pressing **ctrl** + **s** while viewing a graph are stored in /User/Images with a filename that reflects the source (Real Time Graphics or something else) and the date and time.*

Graphit Utility

This program uses the Standard File Dialog (page 5-9) to select a file to be
graphed (Figure 21-21). GraphIt (Chapter 12) is run for the selected file.
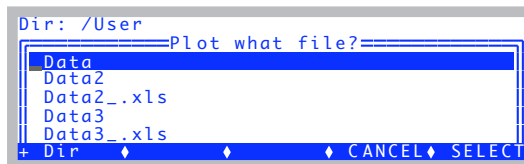
Figure 21-21. Picking a file for Graphit.

Nested Directory Copy

This program prompts you to select a source directory and a destination di-
rectory (Figure 21-22). It then copies all files and subdirectories that are in the
source directory to the destination directory.
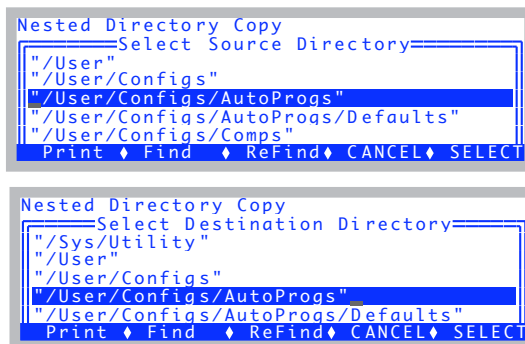
Figure 21-22. Picking source and destination directories.

Pressing **escape** for either prompt will abort the program.

NetworkConfig

This program displays the current network status. It is described in **Connect-
ing with Ethernet** on page 11-7.

SETCLOCK

Sets the system time and date. Set the time zone first (press **SetZone**, **f1**). Then select the field to be changed using ← and →, and change that field's value by pressing ↑ and ↓. Press **Set** (**f5**) to implement your changes, or **Cancel** (**f4**) to abandon them.
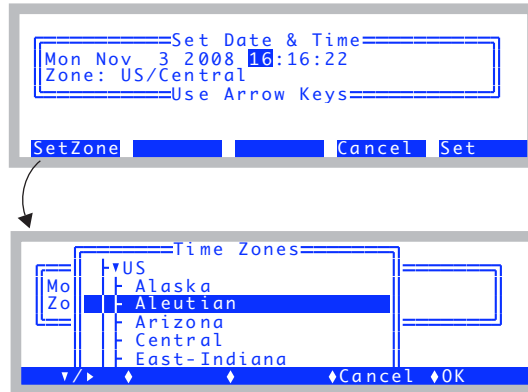
Figure 21-23. Setting the date and time. For best results, pick your time zone first.

SETCOMM

Sets the system's comm port configuration. This program is discussed in **Configuring the Comm Port** on page 11-26.

Simple Terminal

This program sends all keystrokes out of the comm port, and displays all comm port incoming data on the display (Figure 21-24).
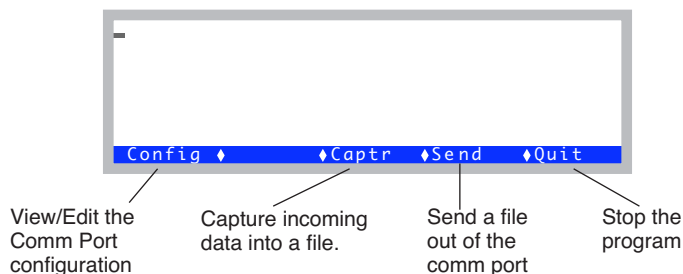
Figure 21-24. The simple terminal program.

In file capture mode (**Captr**), the destination file is selected using the Standard File Dialog. While data is being captured, the display will show only a byte count of captured data, not the actual data (Figure 21-25).
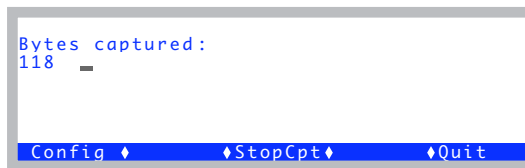
Figure 21-25. The display during data capture to a file.

In the file send mode (**Send**), the display will show the number of bytes transmitted (Figure 21-26).
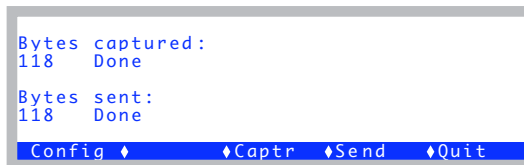
Figure 21-26. The display after sending a file.

XSensitivity

This test measures the apparent cross sensitivity of CO_2 and H_2O in the IR-GAs, and computes correction coefficients that can be used to correct for this effect.

> Note: This test is designed for use on units for which the cross sensitivity was not measured at the factory, which includes any unit last calibrated at LI-COR prior to about May 2010.
>
> If you already have cross sensitivity coefficients (from the factory, or from running this test previously and implementing the results), you will be warned of this fact when you run the test.
>
> You can run the test as a diagnostic and not implement the results, and thereby preserve the factory values.

To determine the cross sensitivity of CO_2 on H_2O, one needs to measure the change in H_2O raw signal caused by a large change in CO_2, all done without changing the true H_2O concentration. Similarly, to determine the cross sensitivity of H_2O on CO_2, one needs to measure the change in CO_2 raw signal caused by a large change in H_2O, all done without changing the true CO_2 concentation.

The best way for the user to perform these measurements is to use the LI-6400's internal CO_2 mixer (if the unit is so equipped), with fresh desiccant and fresh soda lime. The chamber volume should be minimized for this test to minimize water sorption effects. The standard 2x3 cm chamber can be used, but even that can be further minimized by excluding the top half of the chamber with Propafilm or Saran. Better still, completely eliminate chamber lips: seal off the three return holes, and replace the lower lip of the chamber with the single outlet manifold, if you have one.

There are three basic steps to the measurement. 1) Get the IRGAs thoroughly flushed with high CO_2, dry air, and record the raw signals. 2) Drop the CO_2 to 0 without changing the water (turning off the mixer will do that), and record the signals again. 3) Reduce the desiccant scrub to about half way, so that the H_2O will increase, but without affecting the CO_2, and record those readings. Given those three sets of readings, we have what we need to calculate the correction factors.
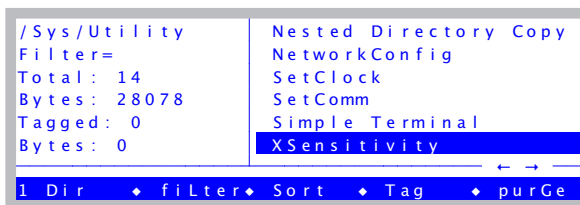
■ **Using the Cross Sensitivity Test Program**

1 **Instrument Preparation**

Make sure you have fresh chemicals, and that the CO_2 mixer is operating properly. Prepare the chamber as described above, and close it. Turn both chemicals to full scrub, set the mixer to a high value, such as 2000 ppm, and let it run for a while. The purpose is to get the chamber well dried down.

2 **Run the Cross Sensitivity Program**

Leave New Measurements mode, go to the Filer, change the directory to /Sys/Utility, highlight XSensitivity, and press **X**.

```
/Sys/Utility        Nested Directory Copy
Filter=             NetworkConfig
Total: 14           SetClock
Bytes: 28078        SetComm
Tagged: 0           Simple Terminal
Bytes: 0            XSensitivity
                                      ←  →
1 Dir    ◆ fiLter◆ Sort   ◆ Tag    ◆ purGe
```

```
      IRGA Cross Sensitivity Test

This program will compute the IRGA
cross sensitivity correction factors.
A CO2 mixer is required, and fresh
soda lime and desiccant.

<enter> starts, or <escape> quits.
```

The next message may or may not appear.

```
This unit currently has correction
factors implemented.

Continue anyway? (Y/N)_
```

Figure 21-27. Launching the cross sensitivity test program.

3   Follow the instructions

You'll a screen full of instructions. Once you're ready, press any key.

```
Do this:
1. Instrument and mixer well warmed up.
2. Desiccant and soda lime FULL SCRUB.
3. Chamber closed.

 Press enter after the instrument
has been running for a while like this
to get the chamber dried down.
```

4   Get first point (automatic)

The mixer will be set to a 4000 mV control setting automatically, and the program will wait for stability to be achieved, or time to run out. The most important thing here is for water to be stable, because we don't want any further dry down between this step and the next.

```
1. High CO2, low H2O



CO2_R_mV CO2_S_mV H2O_R_mV H2O_S_mV
   3346.8    3343.9     -27.5     -29.1
492   dC/dt=0.4 dW/dt=0.5
 earlyOK                          Abort
```
Countdown (s).   Rates of change are in mV per minute.

The program will wait for at least 20s (max 500s) for these stability conditions: $\left| \dfrac{dC}{dt} \right| < 10$, and $\left| \dfrac{dW}{dt} \right| < 1$.

5   Get the second point (automatic)

The mixer will be then be turned off automatically, and the program will wait for stability to be achieved, or time to run out. Now the most important thing is for CO_2 to be stable between this step and the next.

```
1. High CO2, low H2O - ok
2. Low CO2, low H2O


CO2_R_mV CO2_S_mV H2O_R_mV H2O_S_mV
   7.0     630.7     -33.1     -40.4
163   dC/dt=-0.9 dW/dt=-90.3
 earlyOK                          Abort
```

Now the program will wait for at least 20s (max 180s) for these stability conditions: $\left|\dfrac{dC}{dt}\right| < 1$, and $\left|\dfrac{dW}{dt}\right| < 10$.

6   **Get the third point (you have to help)**
You'll be prompted to turn the desiccant knob back to half scrub. DO NOT go any farther toward full bypass than that, as it is important for some flow to continue to go through the desiccant.

```
Now turn the desiccant knob to
HALF scrub.

Then press any key to continue._
```

```
1. High CO2, low H2O - ok
2. Low CO2, low H2O - ok
3. Low CO2, some H2O

CO2_R_mV CO2_S_mV H2O_R_mV H2O_S_mV
     1.7      2.9     717.2     249.1
109  dC/dt=2.6 dW/dt=-1.4
earlyOK                            Abort
```

For this step the program will wait for at least 20s (max 120s) for these stability conditions: $\left|\dfrac{dC}{dt}\right| < 10$, and $\left|\dfrac{dW}{dt}\right| < 30$.

7   View Results

When the third point is recorded, you'll be prompted to put the desiccant back to full scrub, to minimize the time the chamber sees moist air, in case you want to repeat the experiment.

```
Turn the desiccant knob back to
FULL scrub.

Press any key.▄
```

The 1st time through, the Prev values will be 0.

```
CO2 on H2O:      This     Prev      Factory
          Ref:  -1.2E-03  -2.2E-03  +0.00E+00
          Smp:  -1.8E-03  -2.1E-03  +0.00E+00
H2O on CO2:
          Ref:  +7.1E-04  +6.8E-03  +0.00E+00
          Smp:  +9.2E-04  +2.5E-03  +0.00E+00
Repeat ? (Y/N)▄
```

```
CO2 on H2O:      This     Prev      Factory
          Ref:  -1.2E-03  -2.2E-03  +0.00E+00
          Smp:  -1.8E-03  -2.1E-03  +0.00E+00
H2O on CO2:
          Ref:  +7.1E-04  +6.8E-03  +0.00E+00
          Smp:  +9.2E-04  +2.5E-03  +0.00E+00
Repeat ? (Y/N)
Implement ? (Y/N) ▄
```

If not repeating.

The magnitude of these values should be smaller than 10^{-2}. Correction factors $< 10^{-3}$ are for all practical purposes insignificant.

You can repeat the experiment by pressing **Y** to the repeat prompt, and it is usually a good idea to do so at least once. The repeatability of the correction factors will not be very good; you should get the same sign and order of magnitude, but that's about it.

If you implement the results, they will become the "factory" values. However, they are not automatically saved as such. To do that is a separate operation, described below.

Verifying the Correction Factors

The correction factors are applied in computing mole fraction. They are NOT applied to the raw signal readings. That is, the mV signals displayed on the

instrument (e.g. *CO2R_mV*) are always what is actually measured. To see any effect of the correction factors, you must look at mole fraction values (e.g. *CO2R_μm*).

A simple way to verify the effect of the correction factors is to manually re-peat the steps in the test, but do it in New Measurements mode, monitoring mole fractions. Adding a line for mV readings is useful, too.

Step 1. High CO_2, no H_2O.

CO2R_μml	CO2S_μml	H2OR_mml	H2OS_mml
a 1667.7	1670.9	−0.393	−0.320
→ CO2R_mv	CO2S_mv	H2OR_mv	H2OS_mv
n 3304.0	3309.0	−14.2	−9.6
Photo	Cond	Ci	Trmmol
c −2.53	0.0015	4.21E+03	0.0547
Display	Display	What's	Display	Diag
6 QuikPik	List	What	Editor	Mode

Raw signal increased 9mV, but computed concentration only went up .015 ppt.

Step 2. No CO_2, no H_2O.

CO2R_μml	CO2S_μml	H2OR_mml	H2OS_mml
a −0.6	0.4	−0.380	−0.305
CO2R_mv	CO2S_mv	H2OR_mv	H2OS_mv
n −2.3	1.6	−13.7	−0.6
Photo	Cond	Ci	Trmmol
c −0.705	0.00153	716	0.056
Display	Display	What's	Display	Diag
6 QuikPik	List	What	Editor	Mode

Signals dropped slightly, but mole fractions held steady.

Step 3. No CO_2, some H_2O.

CO2R_μml	CO2S_μml	H2OR_mml	H2OS_mml
a −0.6	0.4	6.460	6.049
→ CO2R_mv	CO2S_mv	H2OR_mv	H2OS_mv
n −4.1	−3.9	812.2	779.1
Photo	Cond	Ci	Trmmol
c −0.709	−0.0102	−107	−0.31
Display	Display	What's	Display	Diag
6 QuikPik	List	What	Editor	Mode

Figure 21-28. Verifying cross sensitivity parameters.

Saving the Cross Sensitivity Factors

If the new cross sensitivity factors differ from the currently saved ones, there will be an asterisk on the Calib Menu label in Open's main screen. To save these, go into the Calib Menu, select View Current, and press **f3** (**Save**).
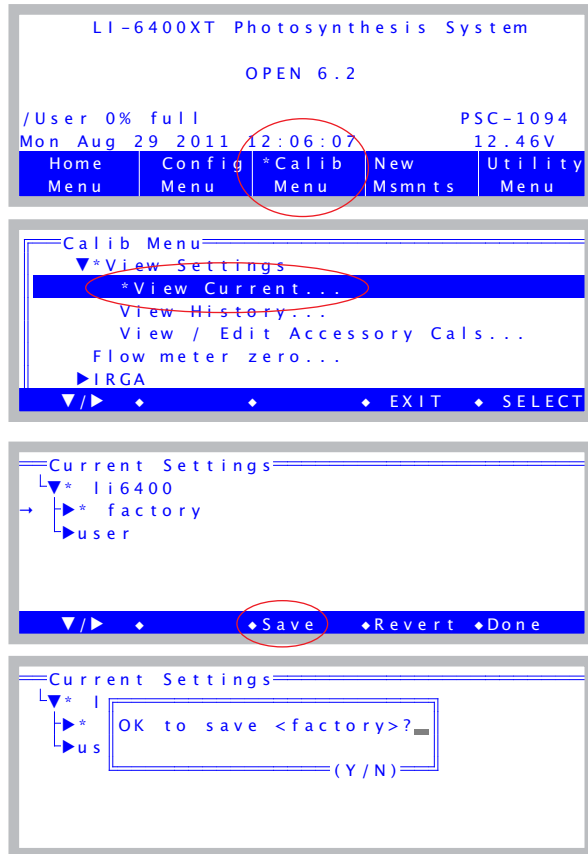
Figure 21-29. Saving the new cross sensitivity factors.

22

Programming with LPL

An introduction to the LI-6400's programming language

22

Programming with LPL

This chapter describes the elements of LPL syntax.

Overview of LPL

LPL is a language designed to program data collection and manipulation applications on the LI-COR LI-6400. LPL programs can also run on other computers to which the LPL Operating system has been ported. LPL itself is platform independent, which means that LPL programs can be developed and tested on a DOS computer, and run on the LI-6400 or a Macintosh without modification.

LPL provides control over a range of hardware, from keyboards and displays, to A/D converters and digital I/O lines. Only the LI-6400 (as opposed to a Macintosh computer, for example) actually has all of this hardware, of course, but the LPL Operating System for each specific platform will provide software simulation of the missing hardware. Thus, for example, it is possible to program and read an A/D converter on a Macintosh just as if it were on the LI-6400; the measured values on the Macintosh, of course, will be entirely mythical[1].

LPL applications take the form of one or more ASCII files. When an application is launched, the file(s) are read and converted into executable form. While it's running, an application can spawn other applications, but the parent application does not continue executing until the child application terminates. Data can be shared between parent and child applications.

---

[1].The Macintosh implementation, however, will allow remote control of an LI-6400, so measured values can be real.

A Simple Example

A simple LPL program is presented in Figure 22-1, which prompts the user for two numbers, computes their average, displays this value, waits for the user to press a key, then quits.

```
/*
  Enter 2 values, compute their mean.
*/

:FLOAT   /* define two floating point variables, and
initialize them to 0 */
    x 0
    y 0

:FCT
Main
{
    "Enter 2 values to average" PRINT

    /* enter two floating point values, store them in X
and Y */
    &y &x "%f %f" ENTER

    x y + 2 /         /* compute the mean */
    y x "\nThe average of %f and %f is %f\n" PRINT

    "\n\nPress Any Key" PRINT
    GETKEY DROP   /* Wait for user */
}
```

22-1. Simple program that prompts for two values, and prints their mean value.

Comments in an LPL program are delimited by **/*** and ***/** and can extend over as many lines as you like. Comments are ignored when the program is actually run; they serve only to clarify things for people editing or reading the program.

The first non-comment we encounter in Listing 22-1 is a **:FLOAT** declaration

```
:FLOAT x 0
       y 0
```

This creates two floating point variables for storing the numbers to be averaged. These variables are given the names x and y, with initial values of zero.

Following that is a function declaration (**:FCT**), used to declare a series of executable steps.

```
:FCT
Main
{
```

The function *Main* does the following: A prompt to enter 2 numbers is printed on the display.

```
"Enter 2 values to average" PRINT
```

The user's response is then parsed into two floating point values, which are stored in variables *x* and *y*.

```
&y &x "%f %f" ENTER
```

x and *y* are then summed, and the sum is divided by 2.

```
x y + 2 /
```

This result is printed, and the program pauses until the user presses a key, at which time the program terminates.

```
y x "\nThe average of %f and %f is %f\n" PRINT
"\n\nPress Any Key" PRINT
GETKEY DROP
```

The View From the Stack

LPL is a *post-fix* language, whose operation centers about an abstraction known as a stack. The stack is a "Last In - First Out" list of objects. The key to understanding an LPL program is to consider each item (keyword, string, number, etc.) in the program, and keep track of what it does to the stack. We'll go through the program in Figure 22-1 again, this time with special emphasis on the stack.

We illustrate stack operations by picturing the stack as a horizontal collection of boxes, with the top of the stack being the right-most box. We start with an empty stack. First, the prompting string's address is pushed onto the stack, and the **PRINT** pops it back off.

Operation	The Stack		
	3	**2**	**1**
1. Stack is empty:			
2. "Enter 2 ..."			Address of "Enter 2..."
3. **PRINT**			

Next we set up for the two values to be entered by the user. **ENTER** captures user keystrokes until the **enter** key is pressed, then partitions what is entered according to what is found on the stack. First, a format string (**"%f %f"**) is popped off. The **"%f %f"** tells the ENTER function to look for two floating point values, and assign them to variables whose addresses are on the stack. The first value is assigned to *x*, and the second to *y*. This pops both addresses from the stack. **ENTER** then quits because the format string tells it to - no more codes - and the number of values assigned is pushed onto the stack.

Operation	The Stack		
	3	**2**	**1**
4. &y			Address of Y
5. &x		Address of Y	Address of X
6. "%f %f"	Address of Y	Address of X	Address of "%f %f"
7. ENTER			2

Now we do some math to find the average value of *x* and *y*. Suppose the user had entered 14 for *x*, and 33.3 for *y*.

Operation	The Stack		
	3	**2**	**1**
8. x		2	14
9. y	2	14	33.3
10. **+**		2	47.3
11. 2	2	47.3	2
12. **/**		2	23.65

Notice how we seem to have an extra 2 being carried along on the stack. This is the 2 that the **ENTER** command left for us, and our example program illustrates slightly sloppy programming; we could have checked this value after the **ENTER** to see if the user in fact entered any values, or enough values, (or even too many values), but we didn't. At the very least we should have disposed of this 2 with a **DROP**, which drops the top entry from the stack.

At any rate, after the **/** is done, we have our average sitting on the stack. Now it's time to display it, along with the values of *x* and *y*. We set this up by first pushing the two values onto the stack, along with a label:

Operation	The Stack				
	5	**4**	**3**	**2**	**1**
13. y			2	23.65	33.3
14. x		2	23.65	33.3	14
15. "The average.."	2	23.65	33.3	14	Address of "The average..."
16. **PRINT**					2

After Step 15, the stack is loaded for **PRINT**. Just like **ENTER**, **PRINT** knows what to do based on the format string it expects to find on the stack. Characters in

```
"The average of %f and %f is %f"
```

are printed as is until the **%f** is encountered, causing it to pop a floating point value from the stack and print it. Similarly the next **%f** causes the 33.3 to be popped and printed, and the final %f pops and prints the 23.65. Thus, **PRINT** cleans the stack except for our left-over 2.

The final thing our program has to do is wait around for the user to press a key to end the program. This is easy.

First, we pop up the standard "Press Any Key" message:

Operation	The Stack		
	3	**2**	**1**
17. "Press Any Key"		2	Address of "Press..."
18. **PRINT**			2

Then we wait until the user presses something (we'll assume the user presses

esc - which generates a keycode of 27):

Operation	The Stack		
	3	**2**	**1**
19. **GETKEY**		2	27
20. **DROP**			2

Making Objects

The objects that can be declared in an LPL program are shown in Table 22-1.

Table 22-1. Declarable Objects

Object	Description
CHAR	Characters are integers that can range in value from -128 to +127, or 0 to 256.
INT	Short integers are two bytes in length, and can range between -32768 to +32767 or 0 to 65536.
LONG	Long integers are four bytes in length, and range between -2147483648 to +2147483647 or 0 to 4294967296.
FLOAT	Floats are 4 bytes in length
DOUBLE	Doubles are 8 bytes in length
PTR	Pointers are simply objects that point at other objects. The information contained in a pointer is the type and address of the object at which it points.
FCT	Functions are collections of commands that perform tasks.

The first five objects (**CHAR** through **DOUBLE**) are for the numbers and strings with which most programs deal. Pointers (**PTR**) are very useful tools for passing information around, and functions (**FCT**) are the collection of commands that do the work of the program.

Syntax

To declare any of the objects listed in Table 22-1, use a colon followed by the object type. For example, the following segment of an LPL program

```
:LONG secs 0
       thisTime 0
```

```
:INT maxCount 100
```

defines *secs* and *thisTime* to be of type **LONG**, and initializes them to 0, while *maxCount* is an **INT** whose value is 100.

Multiple objects can be defined after one declaration, as illustrated by the two floating point variables in Figure 22-1 on page 22-3.

Legal Names
Object names can be any length less than 30 characters. Names may include letters, digits, the underscore character _ , ?, #, %, @, ~, and `. The first character should not be a digit, however, and spaces can not be included. The following are valid object names:

```
x
a12
#lines_@15
```

In LPL, case (upper or lower) does not matter (so **:float** is the same as **:FLOAT** or **:FlOaT**) for any keyword or object name.

Naming Convention
The convention used in this manual is to write LPL keywords in bold upper case, variable names with the first letter lower case, and function names with the first letter upper case (Table 22-2).

Table 22-2. Naming and style convention used in this manual.

Example	Meaning
PRINT	(Bold, UpperCase) LPL Keyword
x, doFlag	(lowercase first letter) User variable name
DoThis	(Uppercase first letter) User function name

When function names are multiple words, the first letter of each is upper case (e.g. *ThisFunction*). Multiple word variables would be similar, except the first letter is lower case (e.g. *finalValue*). This is strictly cosmetic, however, and you can adopt any convention that suits you.

Numerical Objects
Numeric objects (**CHAR**, **INT**, **LONG**, **FLOAT**, and **DOUBLE**) are initialized by following the object's name with the initial value. In the case of inte-

gers (**CHAR**, **INT** or **LONG**), the initial value can be written in decimal, hexidecimal, or as a character (Table 22-3).

Table 22-3. INT and **LONG** initializations

Sequence	Interpretation
abc 15	A decimal integer (15)
def 0xff	Hexidecimal constant (255)
ghi 'A'	Character constant (65)

Floating point objects (**FLOAT** and **DOUBLE**) can be initialized with values written as decimal integers, fixed point values, or exponential notation (Table 22-4).

Table 22-4. FLOAT and **DOUBLE** initializations

Sequence	Interpretation
abc -15.378	Fixed point constant
def 1.234E-5	Exponential notation
ghi 12	Decimal integer

Numerical Arrays

Numerical objects can be declared as arrays. Any array has two extra pieces of information carried around with it, in addition to the actual data: the *size* (the maximum number of objects that the array can hold) and something that's called the *ready* value, which is the actual number of objects that the array is holding.

Arrays are declared by appending square brackets to the object's name. The brackets can be empty, or contain the size of the array. The initial values of the array are given between braces. If the size is not specified in the square brackets (as in def[] above), the size comes from the number of items in the initialization string. The initializing sequence for arrays can have fewer items than the declared size. If there are more initializing items than the declared

size, the final array size matches the initializing sequence count (Table 22-5).

Table 22-5. Declaring arrays.

Declaration	Size	Initial Ready
:FLOAT abc[5] {1.234 1.23E+5 0 4.26 -.001}	5	5
:CHAR def[] {12 20 -15 2 33 6}	6	6
:INT jkl[10] {0xffff 'A' 33}	10	3
:LONG xyz[3] {1 2 3 4 5}	5	5

Char Arrays (Strings)

LPL implements strings as **CHAR** arrays. Characters and character arrays can be initialized exactly like integers, but can also be initialized with string constants. Generally, string constants begin and end with a double quote ("), and everything in between is taken literally, including spaces, newline characters, and comments. Actually, one can use any character to mark the start and end of the string when declaring a **CHAR** array; whatever non-whitespace character is first encountered when the parser is looking for the initializing string is used for the terminating character of that string.

Examples of initializing **CHAR**s and **CHAR** arrays are shown in Figure 22-2:

```
:CHAR
flag   1
decimal   '.'
esc   0x1b
name[80] "SYSTEM"
bad[]   "Illegal Value!"
theBest[] .Penn State.
ok[] {1 2 5 9 20}
```

Figure 22-2. Declaring CHARs and strings.

Character arrays can also be initialized with a combination of characters and numeric values by using escape sequences. For example, suppose you wish to include double quotes within the string. Knowing that the decimal equivalent of the double quote character is 34, one can write

```
:CHAR  abc[]  "Say \34Hello!\34 to her"
```

which would be converted to

```
Say "Hello!" to her
```

Less cryptic methods include:

```
:CHAR abc[] <Say "Hello!" to her>
      def[] "Say \"Hello!\" to her"
```

The Back Slash (\)

The back slash character (\) is a special one within strings. It serves as the trigger for an escape sequence, which simply means that the decimal or hexidecimal value following the \ is to be converted into a character. If you really want a back slash and not an escape sequence, then use two back slashes. Examples are shown in Table 22-6.

Table 22-6. Using \ in string initializations

Initialization String	Interpretation
"sys\\defaults\\config"	sys\defaults\config
"abc\x0d\x0adef"	abc<cr><lf>
"Say \"Good-bye\" to him"	Say "Good-bye" to him
"\65\66\67"	ABC

Following a \ inside of a string constant, LPL expects to find one of the following:

- **up to 3 digits**
  Three digits making a number in the range 0…255 (e.g. \10   \255).

- **A hex value**
  An x or X, followed by up to two hexidecimal digits (0…F) (e.g. \x0d \xff)

- **another back slash **
  (e.g. \\)

- **A short-cut character**
  They are: a, b, f, n, r, t, or "

Table 22-7. Backslash (\) Shortcuts

Sequence	Result
\a	Bell character (decimal 7)
\b	Back space character (decimal 8)
\f	Form feed character (decimal 12)
\n	Newline character (decimal 10)
\r	Carriage return (decimal 13)
\t	Tab character (decimal 9)
\"	Double quote character (decimal 34)

Functions

The general form of a function definition is to follow the function name with braces, between which are the commands that the function is to perform:

```
:FCT reset_all
{
    Reset_menus    Reset_parameters
    Reset_data    Reset_lists
}
```

The Function "Main"

An LPL program can have many functions, and the one named *Main* (if any) is by convention the one that is executed. If there is no function named *Main*, execution begins with the first function that was declared. Note that as far as the operating system is concerned, *Main* (or whatever the name happens to be) will be the ONLY function executed. If there are other functions defined in a program, they are not executed unless they are called from *Main*, directly or indirectly. When *Main* is finally done, the program is done.

The Stack

The stack can be thought of as a list of objects, and as new items are added to the list, they are added to the *top*, pushing all other items down. As items are removed from the list, they are also removed from the top. Many of the built-

in functions of LPL, as well as functions that you may write, expect to find certain types of objects on the stack. These objects are consumed from the stack, and others may be put back onto the stack.

LPL provides a number of commands that directly manipulate the stack, such as **DROP** (discards top stack item), **SWAP** (exchanges top two stack items), and **ROT** (exchanges first and third stack items). Also, stack size (the number of items that can be held at any one time on the stack) can be set. The default size is 10 items.

Post-fix

The default syntax within a function is post-fix. This means that operations are programmed in order of stack operation. To add two values together, we would write

```
17.83  2.5  +
```

If we want to go further, and store the result in a variable named *X*, we would have to put *X*'s address on the stack, then do a store command. In LPL, we would write

```
17.83  2.5  +  &X  =
```

The **&** operator prefixed onto a variable name literally means "put the *address* of this variable onto the stack", rather than the *value* of this variable. Pictorially, the stack operations might look like this:

Operation	The Stack		
	...	2	1
Stack is empty:			
Push 17.83			17.83
Push 2.5		17.83	2.5
Do the **+**			20.33
Push address		20.33	9057:0018
Do the **=**			

In-fix

Post-fix notation is an acquired taste, so for those people who do not like acquiring tastes, and for those occasions when even tasteful people eschew post-fix, there is an alternative: in-fix notation. LPL supports both within function definitions.

Figure 22-3 illustrates the function MAIN from Listing 22-1 re-written using in-fix notation. The magic symbol **$** toggles between in-fix and post-fix within a function definition.

```
Main
{
    $
    PRINT("Enter 2 values to average")
    ENTER(&y, &x, "%f %f")
    PRINT((x+y)/2, y, x, "The avg of %f and %f is %f")
    PRINT("\n\nPress Any Key")
    GETKEY
    DROP
}
```

Figure 22-3. The main function from Figure 22-1 on page 22-3 rewritten using in-fix notation

When writing in-fix, there are a couple of extra rules to keep in mind:

- **Lines are important**
  Use one statement per line, or else use a semicolon (;) as a delimiter. Thus, we could combine the first two statements of Figure 22-3 into

  ```
  PRINT("Enter 2 values to average"); ENTER(&y, &x, "%f %f")
  ```

- **Spaces are not needed**
  When using post-fix notation, each item must be separated by white space (spaces, tabs, or end of line characters). With in-fix, this is not necessary. Parentheses, square brackets, and math operators (+,-,/,*,^) can serve as delimiters between items. Thus, the post-fix sequence

  ```
  1.23 x + &y =
  ```

  can be written with no spaces as

  ```
  y=x*1.23
  ```

Local Objects

Within a function, an object can exist temporarily while that function executes. Such an object is said to be *local*.

```
:FCT
 Main
{
    /* Prints integers from 1 to 10 */

    0 :INT i /* i is local */

    10 NLOOP
       &i 1 + VAL "%d \n" PRINT
    ENDLOOP
}
```

Figure 22-4. Illustration of a local declaration

Local numerical objects are initialized by the value that is currently on the stack. Thus in Figure 22-4, the expression

```
    0 :INT i
```

puts a 0 on the stack, creates a temporary **INT** named *i* and with a value of 0. The 0 is popped from the stack.

Don't confuse normal LPL declarations with local declarations. The **:INT** that appears <u>inside</u> a **:FCT** definition serves a slightly different purpose, and has a different syntax, than an **:INT** that appears <u>outside</u>.

Local Arrays

Local arrays can be declared in a couple of different ways (Figure 22-5).
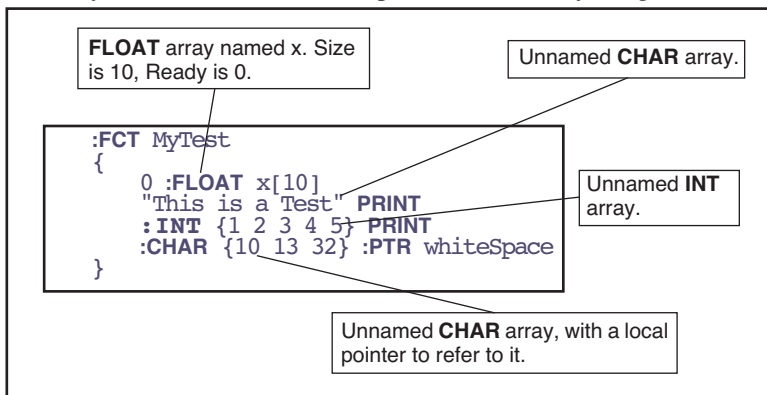
Figure 22-5. Four examples of declaring local arrays.

The sequence

```
0 :FLOAT x[10]
```

creates an empty local **FLOAT** array of size 10. The 0 that's on the stack when this happens serves no purpose other than to be consumed by the declaration. It just has to be there, that's all.

The string

```
"This is a test"
```

declares an unnamed **CHAR** array. When declaring a string within a function, the delimiters must be double quotes.

Unnamed numeric arrays can also be declared, as illustrated by

```
:INT {1 2 3 4 5}
```

This type of declaration creates an array that is full (size = ready = 5).

The sequence

```
:CHAR {10 13 32} :PTR whiteSpace
```

illustrates a method of "naming" an unnamed local array. In this case, a 3 element **CHAR** array is created by the first declaration, and it's address is put

on the stack. Next comes a local pointer declaration leaving us with an object named *whiteSpace* that points at the local array.

NOTE: If you change any values of a local array, or change the ready value, those changes stay in effect *forever* (or at least until the program is re-compiled). For example, consider Figure 22-6. The function *Test* is called two times, and it contains a local CHAR array that gets modified each time *Test* is called.

```
:FCT Main
{Test Test GETKEY}

Test
{
    "ABCD" :PTR x
    x "%s\n" PRINT

Add 1 to each element of the array, and set the size
to 3.
    x 1 +
    x 3 SETREADY
}
```

Figure 22-6. A local array phenomenon illustrated.

Figure 22-6 will produce the following when executed:

```
ABCD
BCD
```

Pointers

Pointers (type **PTR**) are a little more complicated that **INT** or **FLOAT** objects, but provide a lot of power and versatility to the LPL language. The "value" of a pointer is the address and type of the object to which they point. Pointers are initialized by reference to another object. Pointers can remain uninitialized by substituting a 0 for the destination variable name.

We have seen pointers used already in Figure 22-6. But pointers aren't restricted to being local objects. They can be declared out of functions by using the **:PTR** declaration (Figure 22-7):

```
:INT a 5
     b 10

/* MyFct assigned to Test1 */
:PTR myFct Test1

/* myVal assigned to INT a */
    myVal a
:FCT Main
{
    myVal myFct

/* MyFct assigned to Test2 */
    &Test2 &MyFct =
    myVal myFct
}
Test1 { "%d " PRINT }
Test2 { 100 * "%d " PRINT }
```

Figure 22-7. Some PTR declarations

When run, this program will produce

    5 500

You need not worry about the order of appearance when declaring **PTR** objects. Note that in Figure 22-7, for example, we declare *MyFct* as pointing at *Test1*, even though *Test1* has not yet been defined. This is allowed in LPL.

PTR Arrays

Pointer arrays are initialized with groups of object names. Thus, for example, one can create a **PTR** array by

```
:PTR stuff[] {abc MyFct maxLimit
    nameList}
        moreStuff[] {abc stuff}
```

Element of a **PTR** array can be different types of objects. Note that in the above declaration, the 2nd element of *moreStuff* is itself a **PTR** array.

Pointer array declarations can also be nested. For example,

```
:PTR theList[] {
    :PTR { :INT[5] labelOne }
    :PTR { :INT[5] labelTwo }
    }
```

The **PTR** array *theList* has two elements, each one of which is an (unnamed)

pointer array. Each of these unnamed **PTR** arrays has the same size (2 elements) (but they don't have to); the first element is an unnamed **INT** array, and the second is a named object.

PTR arrays can also be used to collect unnamed strings, and even unnamed functions. Figure 22-8 declares a 2 element **PTR** array named *x*. The first element is a string, and the second is a function. The Main function makes the second element operate on the first.

```
:PTR x[] {
    "This is a test"
    :FCT { "***** %s *****" PRINT }
    }
:FCT main
{
    /* Put string on stack. */
    x 1 PICK

    /* Puts fct address on stack, then execute */
    x 2 PICK CALL
}
```

Figure 22-8. PTR arrays can contain unnamed objects.

This program will result in

```
***** This is a test *****
```

being printed to the display.

Passing Parameters

PTRs and **PTR** arrays have many uses in LPL programs. Figure 22-9 illustrates the use of **PTR**s to handle function parameters. In this program, the function named *Go* expects to find three items on the stack waiting for it.

When *Go* executes, three items are popped off the stack and assigned to
PTRs, allowing them to be later referenced.

```
:FCT Main
{
    &Add "Add" "+" Go
    &Sub "Subtract" "-" Go
    &Mult "Multiply" "*" Go
    &Div  "Divide" "/" Go
    GETKEY DROP
}
Go
{
    :PTR symbol
    :PTR label
    :PTR action
    1.23 :FLOAT x
    5.67 :FLOAT y

    label "%s: " PRINT
    x y action :FLOAT result
    result y symbol x "%g %s %g = %g\n" PRINT
}
Add { + }
Sub { - }
Mult { * }
Div { / }
```

Figure 22-9. PTRs and function parameter passing.

When the program in Figure 22-9 is run, it produces the following output:

```
Add: 1.23 + 5.67 = 6.9
Subtract: 1.23 - 5.67 = -4.44
Multiply: 1.23 * 5.67 = 6.9741
Divide: 1.23 / 5.67 = 0.216931
```

Undeclarable Foreign Objects

Another use of **PTR**s is for handling LPL objects that are not declarable. For
example, when a file is opened, a path to the file is created and put on the
stack. Since there is no declarable object in LPL to handle the path, we assign

a **PTR** to it. The example in Figure 22-10 opens a file, writes two lines of text to it, and closes the file.

```
:FCT Main
{
    "w" "test file" OPEN_FILE IF RETURN THEN
    :PTR file

    "This is a new file\n" "%s" file PRINT
    "The end." "%s" file PRINT

    file CLOSE
}
```

Figure 22-10. PTRs and files.

For an example of using **PTR** arrays, we revisit Figure 22-9 on page 22-20, and rewrite the program as follows:

```
:PTR xxx[]
    {
        :PTR { "Add" "+" :FCT { + } }
        :PTR { "Subtract" "-" :FCT { - } }
        :PTR { "Multiply" "*" :FCT { * } }
        :PTR { "Divide" "/" :FCT { / }}
    }
:FCT Main
{
    1 :INT i
    xxx READY NLOOP
        xxx i PICK Go2
        &i 1 + DROP
    ENDLOOP
    GETKEY DROP
}
Go2
{
    :PTR info
    1.23 :FLOAT x
    5.67 :FLOAT y

    info 1 PICK "%s: " PRINT
    x y info 3 PICK CALL :FLOAT result
    result y info 2 PICK x "%g %s %g = %g\n"  PRINT
}
```

Figure 22-11. Using PTR arrays

The **PTR** array *xxx* has 4 elements, each one of which is an unnamed **PTR** array having 3 elements. *Main* goes through the array calling *Go2* with each element found in *xxx*. *Go2* receives 1 parameter, which it assumes is a **PTR** array with 3 elements: 1) the label, 2) the symbol, and 3) the function to execute.

The advantage of Figure 22-11 over Figure 22-9 is that modifications can be made much simpler to the program in Figure 22-11. For example, if we want to add an integer divide test, we can do it all by modifying the PTR array *xxx*: (Figure 22-12).

```
:PTR xxx[]
{
    :PTR { "Add" "+" :FCT { + } }
    :PTR { "Subtract" "-" :FCT { - } }
    :PTR { "Multiply" "*" :FCT { * } }
    :PTR { "Divide" "/" :FCT { / } }
    :PTR { "Integer Divide" "[/]"
        :FCT { :INT b :INT a
            a b / } }
}
```

Figure 22-12. Added integer divide to the options in our program.

Public and Static

Using PUB

When an object is defined in an LPL program, it is by default considered to be a *private* object, because it is only available to the program that defined it. If the program spawns another program, the new program will not know anything about objects belonging to the parent program, unless those objects had been declared as *public*. To declare that an object is public instead of private, precede the object name by the keyword **PUB** (or **pub**).

Figure 22-13 illustrates the use of a public variable.

Program #1

```
:INT
    abc 1
    def 10    /* Private variables */

:FLOAT
    PUB xyz 1.234    /* Public variable */

:FCT
 Main
{
    "UsePubVar" RUN  /* run new program */
    GETKEY DROP
}
```

Program #2 (named "UsePubVar")

```
:FCT
Main
{
    xyz "The value of xyz is %d\n" PRINT
}
```

Figure 22-13. Program #1 calls Program #2, which knows about variable xyz *because it was declared to be public.*

Using STATIC

In large LPL applications, or in library files (files containing code that might be used in any number of applications), it is often convenient to "hide" object names from the rest of the application. Objects whose name is preceded by the label **STATIC** are defined only within the file containing them.

For example, consider a general purpose library file for graphing data. We leave out all the details, and just outline it in terms of a few functions and variables. The interface to this package is the function *GenPlot*, that expects to find the x and y data arrays on the stack, along with the x and y axes labels. All of the functions and variables that *GenPlot* might use are **STATIC**, to prevent name conflicts with any application that might use this file.

File named "/sys/lib/plotter"

```
/* Ploting library */
:FLOAT STATIC xMin 0 STATIC xMax 0
       STATIC yMin 0 STATIC yMax 0
:FCT static FindRange { ... }
STATIC DrawAxes { ... }
STATIC PlotPoints { ... }

PUB GenPlot {
    :PTR xData
    :PTR yData
    :PTR xLabel
    :PTR yLabel

    ...
}
```

Program that uses "/sys/lib/plotter"

```
:INCLUDE "/sys/lib/plotter"
:INT xMin 10  /* Does not conflict! */
:FCT Main
{
    ...
    OpenFile
    ReadData
    "Y" "X" yArr xArr GenPlot
...
}
```

Figure 22-14. Example of using static variables in a library file.

If the above file were named /sys/lib/plotter, we can use it in applications with the **:INCLUDE** directive:

Note that even though our application defines an object named *xMin*, there is no conflict with the one in /sys/lib/plotter, because the latter was defined as static.

There is also a **:STATIC** directive. This saves having to include numerous **STATIC** keywords. We could rewrite the library file described above as shown in Figure 22-15.

```
/* Ploting library */
/******** internal objects *******/
:STATIC 1
:FLOAT  xMin 0  xMax 0
        yMin 0  yMax 0
:FCT  FindRange { ... }
 DrawAxes { ... }
 PlotPoints { ... }

/****** external objects *******/
:STATIC 0
PUB GenPlot {
    :PTR xData
    :PTR yData
    :PTR xLabel
    :PTR yLabel

    ...
}
```

Figure 22-15. The library file of Figure 22-14 rewritten using the **:STATIC** *compiler directive.*

Compiler Directives

We have already seen how to declare objects using the declarative directives **:FCT**, **:PTR**, **:INT**, **:LONG**, **:FLOAT**, **:DOUBLE**, and **:CHAR**. There are other directives as well that control other aspects of building an application. Table 22-8 summarizes some general purpose directives.

Table 22-8. General purpose compiler directives

Directive	Argument	Use
:PRINT	string	Prints a string to the display during compile.
:SEARCH	string	Add a directory to those being searched to satisfy INCLUDE directives.
:INCLUDE	string	Compile an external file as part of this application.
:STATIC	1 or 0	When on, all subsequently defined objects in the current file are STATIC.
:IFNDEF	name	If the name is not defined, ok. Otherwise, skip to the ENDIF
:IFDEF	name	If the name is defined, ok. Otherwise, skip to the ENDIF
:ENDIF	-	Used with :IFNDEF and :IFDEF

As an example, consider the application "/sys/utility/geopotential", which is designed to run either by itself, or on top of the application OPEN on the LI-6400 (see **Geopotential** on page 21-15). This is accomplished by use of the **:INCLUDE** directive, as illustrated in Figure 22-16.

```
/*

    Geopotential Heights

*/

...
:IFNDEF IsFlowBdOn
    :INCLUDE "/Sys/Lib/StdControls"
:ENDIF
:IFNDEF stdPressureCal
    :INCLUDE "/Sys/Lib/StdSensors"
:ENDIF
:IFNDEF Geopotential
    :INCLUDE "/Sys/Lib/UsefulEqns"
:ENDIF
...
```

Figure 22-16. Geopotential Heights program (partial listing)

Near the top of the program are several includes, protected by **:IFNDEF** directives. The names *IsFlowBdOn*, *stdPressureCal*, and *Geopotential* are public objects that are defined in the three library files *StdControls*, *StdSensors*, and *UsefulEqns*, respectively. If the application OPEN is already running, these objects will be loaded and available. Otherwise, they must be included.

As an example of using **:IFDEF**, consider the utility "/sys/utility/boards off" (Figure 22-17). This very short application turns off the IRGA and flow control boards; this can be useful sometimes, but not while some other application that is using these devices is running. To protect against this happening, the program checks to see if the library file StdControls has been linked by a parent application. If it is there, the program warns the user that it can't turn off the boards.

```
/*

    PowerOFF

*/
:IFDEF FlowBdOn
    :FCT main
{
    CLEAR
    "This program should not be run now!

Press Any Key" PRINT GETKEY DROP
}
:ENDIF
:IFNDEF FlowBdOn
    :FCT main
{
    CLEAR
    "This program will power OFF the flow
board and the IRGA board.

OK ? (Y/N)" PRINT GETKEY UPC 'Y' == IF
    0 0x0301 DIOSET
    0 0x0302 DIOSET
THEN
}
:ENDIF
```

Figure 22-17. BoardsOff listing

LPL Topics

23

Programming with LPL

23 *LPL Topics*

Stack Control

LPL provides several stack management tools.

Table 23-1. Stack Control Keyword Summary

Keyword	Description
ADR?	Is the item an address?
DROP	Dispose of the top stack item
DUP	Replicate the top item on the stack
FLUSH	Flush the stack
MATHERR	Enable/disable reporting certain math errors
ROT	Swap the first and third stack items
SHOW	Show what's on the stack
STKCHECK	Enable/disable certain stack error checking
STKREADY	Returns number of items on the stack
STKRESIZE	Change the stack size
STKSHARE	Controls stack sharing between a parent and child applications
STKSIZE	Returns stack size
SWAP	Exchange the top two items on the stack

Basic Stack Manipulation

Basic stack manipulation tools are **FLUSH**, **DUP**, **SWAP**, **DROP**, and **ROT**. Their effect is illustrated below:

Operation	The Stack			
	4	**3**	**2**	**1 (top)**
(Starting Condition)	-	C	B	A
FLUSH	-	-	-	-
DUP	C	B	A	A
SWAP	-	C	A	B
DROP	-	-	C	B
ROT	-	A	B	C

Sometimes it is useful to find out what is on the stack. **STKREADY** returns the number of items on the stack, while **ADR?** indicates if the top item is an address or a numeric value. More specific information can be obtained with **TYPE**, which returns an ID value corresponding to the type of object that is on the top of stack.

Stack Size, Status, Errors

When an application is launched, the default stack size is 10. If a larger stack is needed, the stack can be resized by **STKRESIZE**. The current size of the stack is given by **STKSIZE**.

When one application launches another, the child application can have it's own stack, or share the parent's. Stack sharing is a method of passing data between and parent and child application. **STKSHARE** is the keyword controlling this.

There are two types of errors that can occur during stack operations. Math errors (such as divide by zero), and stack errors (such as pushing too many items onto the stack). The LPL operating system can be made to report either of these types of errors, or ignore them, by use of the keywords **MATHERR** and **STKCHECK**.

Conditionals and Loops

Basic flow control in LPL programs is provided through a loop structure, and a conditional structure.

Table 23-2. Flow Control Keyword Summary

Keyword	Description
IF	Conditional execution. Requires THEN.
ELSE	Optional. Used between IF and THEN.
THEN	Marks the end of a conditional.
LOOP	Marks start of an infinite loop.
NLOOP	Marks start of a finite loop.
ENDLOOP	Marks the end of a loop structure.
BREAKIF	Breaks out of a loop if true.
BREAK	Breaks out of a loop.
RETURN	Exit the current function.

LPL provides two basic flow control tools for program execution within a function: the **IF**…**ELSE**…**THEN** structure and the **LOOP**…**ENDLOOP** structure. Either of these structures can be nested up to 20 deep. Figure 23-1

illustrates both of these structures used together in a little program that prints various messages based on what keys the user presses.

When run, this program will display various messages on line 5 of the display, such as

```
Key #1 is Ascii 'a'
Key #2 is <return>
Key #3 is NonAscii, code = 6301
```

For more information about key codes, refer to **Keyboard Codes** on page 23-29.

```
/* Illustrates IF..THEN and LOOP..ENDLOOP
*/
:FCT Main
{
    CLEAR
    "Press Some Keys (<esc> to quit)" PRINT

    1 :INT counter

    LOOP
        GETKEY :INT k

        /* Quit on escape */
        k 0x1b == BREAKIF

        1 5 POSXY
        counter "Key #%d is " PRINT

        k 255 <= k 0 >= AND IF
/*
This LOOP allows us to use BREAK s, rather than a lot of nested
IF..ELSE..THENs
*/
            1 NLOOP
            k '\r' == IF "<return>" BREAK THEN
            k '\n' == IF "<ctrl+j>" BREAK THEN
            k '\f' == IF "<ctrl+l>" BREAK THEN
            k "Ascii '%c' "
            ENDLOOP

        ELSE
            k "NonAscii, code = %04x"
        THEN

        PRINT CLREOL
/* Increment the counter. */
        &counter 1 + DROP
    ENDLOOP
}
```
Figure 23-1. IF…ELSE…THEN and LOOP…ENDLOOP illustrated.

Array Operations

LPL supports arrays of type CHAR, INT, LONG, FLOAT, DOUBLE, and PTR.

Table 23-3. Array Keyword Summary

Keyword	Description
AMOVE	Shift elements within an array.
APP	Append an item to an array.
COS	Compare two arrays for type and content.
FIND	Find an item in an array.
FREE	Free a dynamically allocated array.
MAKE	Dynamically allocate an array.
MAKE4	Allocate an array for another application.
PDRF	Pointer dereference. (1 step version of PVAL)
PFIND	Pointer version of FIND.
PICK	Select an array element.
PVAL	Pointer version of VAL.
READY	Returns the number of elements in an array.
SETREADY	Sets the array Ready value.
SIZE	Returns the maximum size of an array.
SUBSET	Create an array that is a subset of another.
VAL	Returns the value of an numeric array element.

Also, all of the mathematical transform keywords (listed in **Math Functions** on page 23-18) can also be applied to arrays.

Array SIZE and READY values

All arrays share a common feature - they maintain header information that tells the following:

- **Array maximum size**
  How many elements could fit in the array.

- **Array working size**
  How many elements are actually in the array (the <u>ready</u> value).

Three LPL keywords relate to this header information: **SIZE** returns the size of the array, **READY** returns the ready value, and **SETREADY** sets the ready value. In the declarations below

```
:INT xx[] { 1 2 3 4 5}
:CHAR yy[100] "/user/MyFile"
:FCT test { 0 :LONG temp[200] ... }
```

the size and ready value of *xx* is 5, size of *yy* is 100, but the ready value is 12, and the size of *temp* is 200 while the ready value is 0.

The **SETREADY** keyword allows direct user manipulation of the ready value. If we were to add the line

```
yy 5 SETREADY
```

to the function `test` in the above example, then the value of *yy* would become "`/user`" (1st 5 characters). Alternatively,

```
yy 100 SETREADY
```

would make the array appear full. **SETREADY** never alters actual content, so "artificially" filling an array will leave you with whatever happened to be in there. Figure 23-2 illustrates.

```
:CHAR line[10] "1234567890"
:FCT main
{
    "ABC" line =
    line "'%s'\n" PRINT
    line 10 SETREADY
    line '%s'" PRINT
    GETKEY DROP
}
```

Figure 23-2. SETREADY changes length, not content.

The string *line* is declared with a value of "`1234567890`", but the sequence

```
'"ABC" line =
```

changes the first three characters to `ABC` and sets the ready length to 3. Running the program illustrated in Figure 23-2 will produce the following output:

```
'ABC'
'ABC4567890'
```

Manipulating Arrays

There are many methods of manipulating the content of arrays:

*   **Equating**
    The **=** keyword can be used to set one array equal to another (in ready value and content) or to set the elements [1…Ready] all to a constant value.

*   **Mathematical operators**
    Described in **Math Functions** on page 23-18, these keywords can operate on elements [1…Ready] of arrays.

*   **Appending**
    APP can be used to append data onto an array.

*   **Select 1 element**
    PICK is used to select a specific element in an array, to get or change its value.

*   **Select a subset**
    SUBSET will make an array out of a selected subset of another.

*   **Shift elements**
    AMOVE will shift values within an array.

*   **I/O Tools**
    Some I/O tools (**I/O Programming** on page 23-38) can be used on arrays: **ENTER** will append data onto any array, while arrays of type **CHAR** can be made into Paths, allowing **PRINT** to be used to append data to them.

Figure 23-3 illustrates some basic array manipulation tools.

```
:INT x[10] {1 2 3 4 5 6 7 }
:FCT main
{
    50 x 5 PICK =   /* Set 5th element to 50 */
    ShowX

    8 x APP          /* Append 8 to array */
    ShowX

    1 6 3 x AMOVE   /* Copy pos 6, 7, 8 to pos 1 */
    ShowX

    x 2 7 SUBSET :PTR s
    9 s =         /* Set elements 2..7 to value of 9 */
    ShowX

    GETKEY DROP
}
ShowX
{
    x "%(* )d\n" PRINT
}
```

Figure 23-3. APP, PICK, AMOVE, and SUBSET illustrated.

Running this program will produce the following output:

```
1 2 3 4 50 6 7
1 2 3 4 50 6 7 8
6 7 8 4 50 6 7 8
7 9 9 9 9 9 9 8
```

Address to Value conversion

In Figure 23-3, the line

```
50 x 5 PICK =
```

puts the <u>address</u> of the 5th element of array *x* on the stack, and sets the <u>value</u> of that element to 50. If instead, we simply wanted to know the value of the 5th element, we would write

```
50 x 5 PICK VAL
```

The **PICK** puts the address of an **INT** (the 5th element of *x*) on the stack. The **VAL** pops that address off the stack, looks up the numeric value of that **INT**, and pushes that value onto the stack.

In Figure 23-4, the functions *PrintArray1* and *PrintArray2* are functionally equivalent - they both print out the contents of an integer array whose address

is passed to it. *PrintArray1* does it the hard way, but illustrates the use of **VAL**.

```
:FCT PrintArray1
{
    /* Address of array assigned to 'a'. */
    :PTR a

    1 :INT i
    /* Loop 'READY' times */
    a READY NLOOP
    /* Get the ith value. */
        a i PICK VAL
        "%d " PRINT
    /* Increment the counter */
        &i 1 + DROP
    ENDLOOP

PrintArray2
{
    :PTR a

    /* Much more efficient! */
    a "%(* )d" PRINT
}
```

Figure 23-4. Two methods to print the contents of a numeric array. Function PrintArray1 *illustrates the use of* **VAL***, while* PrintArray2 *is much more efficient.*

There is a variant of **VAL** called **PVAL** that is sometimes used with **PTR** arrays. See **Using PTR arrays** on page 23-13.

Searching and Comparing Arrays

FIND is used to locate the occurrence of an element or group of elements (another array) in an array. **COMPARE** is used to tell if two arrays are identical in type, ready value, and content (elements 1 through Ready only). Another tool - **SEARCH** - is available if the array is a **CHAR** array that has an Path opened to it (see **Paths** on page 23-42).

There is a variant of **FIND** called **PFIND** that can be used with **PTR** arrays, described in **Using PTR arrays** on page 23-13. **FIND** and **PFIND** are illustrated in Figure 23-8 on page 23-15.

Dynamic Allocation

A declared array hangs around for the life of a function (if it's a local array) or the life of the application. Sometimes it is efficient to dynamically allocate an array. That is, create it when you need it, then dispose of it when you're done, and recover that memory. Such arrays are created by the application (**MAKE**), used, then disposed of (**FREE**). Figure 23-5 contains some snip-

pets of LPL code that illustrate differences between declaring arrays and dynamic allocation of arrays.

```
/* scores is a global array */
:INT scores[20] {1 2 3 4 5 6 7 8 9 10}

:FCT Main { ... testFct ...}
TestFct
{
/* text is a local array */
    0 :CHAR text[80]
    "This is a test" text =
    ...
/* Allocate a 1000 element float array.*/
    FLOAT 1000 MAKE IF RETURN THEN
    :PTR values
    ...
/* Dispose of the float array */
    values FREE
}
```

Figure 23-5. Program outline illustrating use of **MAKE** *and* **FREE**.

The **INT** array scores exists for the entire time the program is active. The local array text exists only while the function *TestFct* is active. A 1000 element **FLOAT** array that is created in *TestFct* by the sequence

```
FLOAT 100 MAKE IF RETURN THEN
:PTR values
```

The keyword **MAKE** expects two values on the stack: the size of the array to be created, and a number indicating the type of array to create. We indicate type by the keyword **FLOAT** (note that it's **FLOAT**, not **:FLOAT**!). If **MAKE** succeeds (if there's enough space to make the array), then the address of the new array is pushed onto the stack, followed by a 0. If **MAKE** fails, only a 1 is pushed onto the stack. When **MAKE** succeeds, it is up to you to keep track of the address for future reference, and this is usually done by assigning a **PTR** to it. In Figure 23-5, a local **PTR** is assigned; we can do that because the array is deallocated (using **FREE**) before the function is done. If we wanted to use the new array after the creating function terminates, we would want to be sure to assign a global **PTR** to the array's address. Figure 23-6 illustrates how to do this.

```
:PTR temp 0
:FCT Main
{   MakeSpace NOT IF
        UseSpace KillSpace
    THEN
    GETKEY
}
MakeSpace
{
    CHAR 50 MAKE IF 1 RETURN THEN
/* Don't use a local PTR here! */
    &temp =
/* The 0 signals OK */
    0  }
UseSpace
{
/* Set Ready value to full */
    temp DUP SIZE SETREADY
/* Fill with A's */
    'A' temp =
    temp PRINT
}
KillSpace { temp FREE }
```

*Figure 23-6. Using **MAKE** and **FREE**.*

There is a variation of **MAKE** named **MAKE4** that allows you to specify which application "owns" the allocated array. This allows a child application to create an array for the parent application to use after the child application goes away. If **MAKE** were used for this, instead of **MAKE4**, the array would be removed by the system's garbage collector when the child application terminated.

The Table 23-4 indicates where declared, local, and dynamically allocated arrays are stored.

Table 23-4. Where arrays exist.

Type of Array	Where it lives
Declared global e.g. **:INT** x[5] {1 2 3 }	The application's data segment.
Local e.g. **:FCT** x { "hello" }	The local stack. (It's size is set by the **:LOCAL** directive)
Allocated with **MAKE** or **MAKE4**	Free memory.

Using PTR arrays

Pointer arrays have some subtleties not associated with other arrays. While it is true that each element of a pointer array is a pointer (**PTR**), it is also true that those pointers could be pointing at anything, including other pointers or pointer arrays.

The concept of **VAL** gets ambiguous, then, for a pointer array. Suppose we have the following declarations:

```
:INT x 50
    y 100
:PTR z[2] {x y}
```

Does the sequence "z 1 **PICK VAL**" mean…

1   **…the value of the pointer?**
That is, the type and address of x

 or

2   **…the value of what is being pointed at?**
In this case, 50

It turns out that **VAL** has the 2nd meaning. But how can meaning 1) be achieved, namely getting the address of x on the stack, to change it's value, for example? For that we use **PVAL**. Figure 23-7 illustrates.

```
:INT x 50
     y 100
:PTR z[2] {x y}
:FCT Main
{
/* Print value of x, pointed at by z[1]. */
    z 1 PICK VAL "%d\n" PRINT


/* Change value of x to 75, but get address from z[1].*/
    75 z 1 PICK PVAL =


    x "%d\n" PRINT

/* Change z[1] so it points to y instead of x */
    &y z 1 PICK =


    z 1 PICK VAL "%d" PRINT
    GETKEY DROP
}
```

Figure 23-7. Illustration of **VAL** *and* **PVAL**.

The program in Figure 23-7 will produce the following output

```
50
75
100
```

from which we see that

- **the PICK VAL sequence**
  yields the value (50) of the ultimate object (x);

- **the PICK PVAL sequence**
  yields the address of the final object pointed to by the array element (&x);

- **and PICK by itself**
  yields the address of the pointer array element (z[1]).

Note that intermediate pointers, if any, are hidden. If we change the declarations in Figure 23-7 to be

```
:INT x 50
     y 100
:PTR p x          /* Add a pointer -> x */
     z[2] {p y}   /* 1st element is p */
```

we find the program produces the same result.

Another related tool is **PDRF**, which tracks a pointer just one step. **PVAL** traces links until a non-pointer is found.

A similar situation exists with **FIND** and **PFIND**. **FIND** when used with pointer arrays is value oriented, while **PFIND** is address oriented.

```
:INT a[4] { }
     b[4] { }
     c 5
     d 10

:PTR p1[] {c a}
     p2[] {p1 b}
     p3[] {a p1 b}

:FCT main
{
  p3 p2 FIND "p3 p2 find %d\n" PRINT
  p3 p1 FIND "p3 p1 find %d\n" PRINT
  p3 p2 PFIND "p3 p2 pfind %d\n" PRINT
  p3 p1 PFIND "p3 p1 pfind %d\n" PRINT

  GETKEY DROP
}
```

Figure 23-8. **FIND** *and* **PFIND** *illustrated*

Figure 23-8 will produce the following result:

```
p3 p2 find 2
p3 p1 find 0
p3 p2 pfind 0
p3 p1 pfind 2
```

One way to make sense of these results is to consider what the target object "looks like" when operated on by **FIND** and **PFIND** (**Explanations of the FIND and PFIND in Figure 23-8.**Table 23-5)**.**

Table 23-5. Explanations of the FIND and PFIND in Figure 23-8.

Statement	Meaning
p3 p2 **FIND**	Does {&a &p1 &b} contain {&p1 &b}? Yes, at 2
p3 p1 **FIND**	Does {&a &p1 &b} contain {&c &a}? No

Table 23-5. (Continued)Explanations of the FIND and PFIND in Figure 23-8.

Statement	Meaning
p3 p2 **PFIND**	Does {&a &p1 &b} contain &p2? No
p3 p1 **PFIND**	Does {&a &p1 &b} contain &p1? Yes, at 2

FIND also has some added capability when searching pointer arrays. Normally, the target of **FIND** should be either an array of the same type as the object array (the one being searched), or else the target should be the same type as of one of the elements of the object array. When the object array is a pointer array whose elements are also pointer arrays, **FIND** can be made to search specific parts of those arrays. For example, consider Figure 23-9.

```
:PTR x[]
    { :PTR {"abcdef" "12345"}
    :PTR {"zxywqr" "+-098"}
    :PTR {"jk lmn op q" "1 2 3 4" }
}
:FCT Main
{
    x 1 "wqr" FIND "%d\n" PRINT
    x 2 "2 3" FIND PRINT

    GETKEY DROP
}
```

Figure 23-9. **FIND** *with subscripts*

The pointer array *x* has 3 elements, each one pointing to an array containing two character arrays. The sequence

```
    x 1 "wqr" FIND
```

returns the lowest subscript of array *x* that points to a pointer array whose 1st element contains the string "**wqr**". The result is 2. Similarly, the statement

```
    x 2 "2 3" FIND
```

will produce 3, since this string is contained in the second element of an array pointed at by the 3rd element of *x*.

If this is not complicated enough, further subscripting is supported by **FIND**, allowing statements such as

```
    pArray 1 4 3 target FIND
```

which searches the 3rd element of arrays pointed at by the 4th element of arrays pointed at by the 1st element of the arrays pointed at by the elements contained in *pArray*. Up to 10 (no kidding!) subscripts may thus be specified.

Math Functions

LPL has a wealth of functions that do mathematical operations on numerical objects, including arrays.

Table 23-6. Math Keyword Summary

Keyword	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
<	less than?
<=	less than or equals?
==	are they equal?
>	greater than?
>=	greater than or equals?
^	Raise to a power
ABS	Absolute value
ACOS	Arccosine
AND	are top two items both non-zero
ASIN	Arcsine
ATAN	Arctangent (result -90 to +90)
ATAN2	Arctangent (result -180 to +180)
BINAND	Binary and
BINCMP	Binary complement
BINEOR	Binary exclusive or
BINIOR	Binary inclusive or
BSHIFT	Binary shift left or right
CHS	Change sign

Table 23-6. (Continued)Math Keyword Summary

Keyword	Description
COS	Cosine
DEG	Use degrees for trig functions
EXP	Exponential
HASCOPRO	Is coprocessor installed? (Not applicable for 5.0 and up)
INTERPOL	Interpolate vectors
LGT	Log base 10
LOG	Natural log
LWC	Convert to lower case
MAX	Find maximum value
MIN	Find minimum value
MOD	Modulus
NOT	is (top item) zero?
OR	is one or the other non-zero?
POLY	Polynomial
RAD	Use radians for trig functions
RANDOMIZE	Randomizes the seed for the random generator
REGRESS	Compute coefficients for a polynomial
RND	Random float between 0 and 1
SIN	Sine
SQRT	Square root
SUM	Sum a vector
TAN	Tangent
UPC	Convert to upper case
<>	not equal?

The mathematical functions in LPL can operate not only on values, but also on collections of values. For example, consider the addition operator **+** as used in Figure 23-10 below

```
:INT      iArray[5] {1 2 3 4 5}
:FLOAT    floatVal 1.234
          fArray[5] {10.1 11.1 12.1 13.1 14.1}

:FCT Main
{
/* Add two values 11.734 left on the stack */
  10.5 7 +


/* Add a constant to an array whose address is left on the stack.
 iArray = {6 7 8 9 10} */
  iArray 5 +

/* Add Two Arrays. fArray address left on stack.
fArray = {16.1 18.1 20.1 22.1 24.1} */
  fArray iArray +

/* Add value to variable, by address.
floatVal address left on stack,floatVal = 6.234 */
  &floatVal 5 +

/* Compute the sum of an array.
(6+7+8+9+10) = 30 is left the stack */
  0 iArray +
}
```

*Figure 23-10. Various uses of the **+** operator.*

The **+** operator obviously does more than blindly add whatever is on the stack together. It checks to see what <u>type</u> of objects are sitting on the stack, and operates accordingly. Thus, we can not only add two numbers together, we can also sum an array, add a constant to an array, or add two arrays together element by element, all with the keyword **+**.

In general, LPL's math functions can be classified as <u>Two Object Transforms</u> (e.g. **+**), or <u>Single Object Transforms</u> (e.g. **ABS**, absolute value).

Single Object Transforms

Single Object Transforms can operate on values, addresses of numeric variables, or arrays, so there are a variety of combinations of things that can be on the stack when a Single Object Transform executes. We illustrate the possible before and after stack status by the following shorthand:

```
Initial:    Num a
Final:      DOUBLE or LONG b
```
or

```
Initial:    NAddr c
Final:      NAddr c
```
or

```
Initial:    Array d
Final:      Array d
```

"Num" means DOUBLE or LONG. When a Single Object Transform operates on numeric value a, the result will be either a DOUBLE or LONG pushed back onto the stack, depending on the keyword, and on the type (DOUBLE or LONG) of a. For example, the sine function (**SIN**) will always return a DOUBLE regardless of whether the argument is DOUBLE or LONG, while the sign change function (**CHS**) retains a's type.

"NAddr" means a numeric object's address, such as the address of a CHAR, INT, LONG, DOUBLE, or FLOAT. "Array" means the address of any array. When a Single Object Transform operates on NAddr c or Array d, the address remains on the stack; the <u>values</u> of the variable or array may have changed, however.

When an Array is the object of the transform, each element of the array is transformed. If the array is a PTR array, its elements can potentially point at non-numeric objects, such as function addresses, or other PTR arrays, so what happens then? The rule is this: all pointer elements are tracked down (pointers to pointers to pointers, etc.), and if the final object is numeric, it is transformed. If not, and the transform requires ultimately a numeric value, then an "Illegal Object" fatal error is generated.

Two Object Transforms

Two Object Transforms can operate on any combination of value, address of numeric variable, or array. The <u>target</u> of a Two Object Transform is the 2nd item on the stack, not the top item on the stack. The target can be a "Num", "NAddr", or an "Array", just like with Single Object Transforms. The object

on the top of the stack can also be any of these three items. In our reference section shorthand, we express these possibilities by

Initial: Num *targetNum*, <NObj *c* | Array *d*>
Final: DOUBLE or LONG *resultNum*
or

Initial: NAddr *targetVal*,<NObj *c* | Array *d*>
Final: NAddr *targetVal*
or

Initial: Array *targetArr*, <NObj *c* | Array *d*>
Final: Array *targetArr*

Items in <> brackets, separated by a vertical bar (for example < a | b >) means "one or the other of". Thus, <a | b> means a or b. "NObj" means numerical object: DOUBLE, LONG, or NAddr.

When a Two Object Transform operates on numeric target *targetNum*, the result *resultNum* will be DOUBLE or LONG, depending on the keyword, and depending on type (DOUBLE or LONG) of the original value.

When the target is an address *targetVal* or *targetArray*, the address remains on the stack, while the value(s) of the variable or array are subject to change.

When the top item on the stack is array *d*, but the target is not (*targetVal* or *targetNum*), the transform is done using each element in *d*, with the *targetVal* or *targetNum* holding the cumulative result. For example, you can sum Array *d* by the expression

    0 d +

(Note that the reverse order ("d 0 +") would simply add 0 to each element of *d*, accomplishing nothing). Table 23-7 summarizes the combinations

Table 23-7. Two Object Transform combinations

Target Object	Top-of-Stack Object		
	Num	**NAddr**	**Array**
Num	Resulting value left on stack		Transform performed multiple times, once for each array item. Resulting value left on stack.

Table 23-7. Two Object Transform combinations

Target Object	Top-of-Stack Object		
	Num	**NAddr**	**Array**
NAddr	Target address left on stack.		Transform performed multiple times, once for each array item. Target address left on stack.
Array	Each element of the array is transformed using the top-of-stack value. Target address left on stack		Transform is performed on (and only on) corresponding items between the arrays. Target address left on the stack.

Trig Functions

DEG and **RAD** specify the units of measure (degrees or radians) for trig functions. The default mode when an application runs is radians, unless the application is stack sharing (**STKSHARE**) with it's parent.

Display Control

LPL provides several tools for controlling the display. This section deals only with text; graphical applications are discussed in **Graphics** on page 23-61.

Table 23-8. Keywords for controlling the display.

Keyword	Description
BACKLIGHT	Turns backlight on or off
BLKATTR	Sets attribute in a specified rectangle (window coords)
CONTRAST	Set display contrast (0 to 100)
CLEAR	Clears text and attribute in current window
CLREOL	Clears from cursor location to edge of current window
DISPHEIGHT	Height (chars) of display
DISPWIDTH	Width (chars) of display
DISPUPDATE	Suspend / resume screen updates
GETCONTRAST	Get the current contrast setting (0 to 100)
GETDISP	Gets display text, attribute, and window info

Table 23-8. (Continued)Keywords for controlling the display.

Keyword	Description
GETTEXT	Gets text from a rectangle, (display coords).
GETWINDOW	Get current window info (rectangle, cursor, etc.)
ISBACKLIGHT	Is backlight on or off
MOVETEXT	Move a rectangle to a new location (lcd coords)
POSXY	Positions the cursor (window coords).
PUTDISP	Restores window, text, and attribute
PUTTEXT	Puts text into a rectangle, (display coords)
PUTWINDOW	Set current window info.
SCRLOCK	Locks/unlocks vertical scrolling
SETATTR	Sets attribute for subsequent printing
SETCURSOR	Sets cursor type
WINDOW	Establish a window (display coords)

Display Size

Since LPL is platform independent, there is no fixed display size assumed. Therefore, the keywords **DISPHEIGHT** and **DISPWIDTH** will return the height (in rows) and width (in characters) of the display (Figure 23-11).
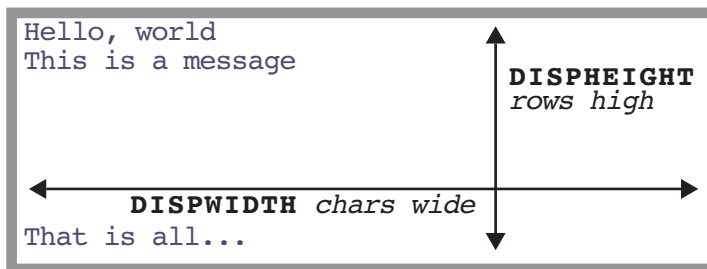
Figure 23-11. Illustration of DISPHEIGHT and DISPWIDTH.

Text Windows

LPL supports windowing the display. A text window is simply a rectangle on the display in which text operations are confined. There are two types of units of measure used: *display* units (absolute) and *window* units (relative) (Figure 23-12).
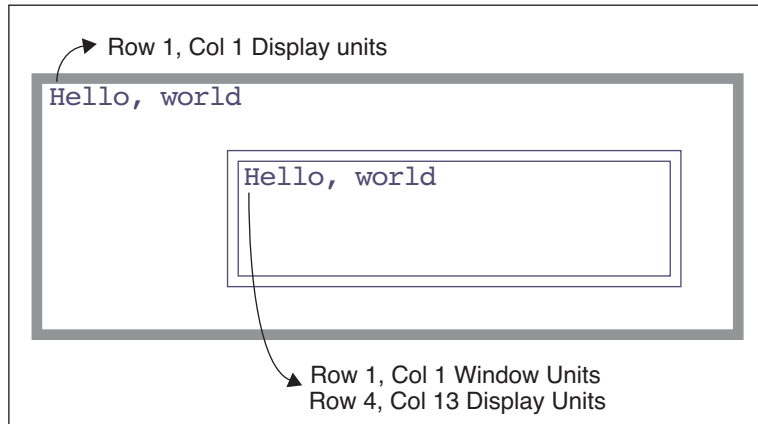
Figure 23-12. Text windows

The **WINDOW** keyword creates a window on the display; the window can be framed by a single or double line border with up to two labels on it, or no border at all. Figure 23-13 illustrates.

```
:FCT Main
{
    CLEAR
    3 2 20 5 2 1 "Win1" 5 "Hello" WINDOW
    "This is in window 1" PRINT

    15 4 35 8 1 3 "Win2" WINDOW
    "This is in window 2" PRINT

    GETKEY
    /* Set window back to full display, borderless */
    1 1 DISPHEIGHT DISPWIDTH 0 WINDOW
}
```

```
  Win1
 This is in windo
 w 1
          Hello      Win2
                    This is in window 2
```
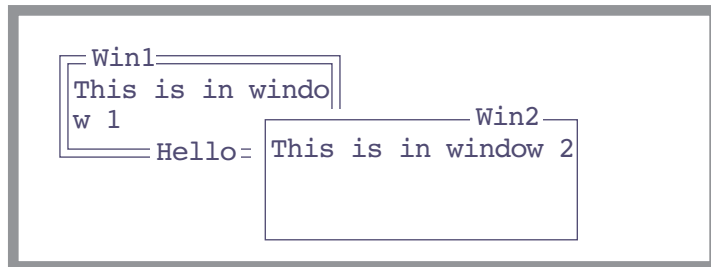
Figure 23-13. Illustration of **WINDOW** *keyword., with resulting display*

By default, all text windows scroll automatically when filled with text. However, they can be made to not scroll, but start overwriting at the upper left corner instead. The keyword **SCRLOCK** controls this feature.

Text and Cursor Attributes

Text can have any combination of 2 attributes besides normal: inverse and blinking. The keyword **SETATTR** sets the attribute for all subsequent printing to the display.

The cursor can be in one of three states: hidden, underline, or full height. The keyword **SETCURSOR** establishes the cursor type. Also, the cursor can be moved to any location within the current window by the **POSXY** keyword.

The keyword **GETWINDOW** loads the current window status, including attribute and cursor information, into an array. It's counterpart, **PUTWINDOW**, will immediately implement a window's scaling, attribute, and cursor location. Figure 23-15 creates two windows with differing attributes, then switches between them as it prints what you type.

```
:INT w1[8] { }  /* Holds attributes for window      /* Inverse. */
1 */                                                    1 SETATTR
         w2[8] { } /* Holds attributes for         /* No cursor.*/
window 2 */                                             0 SETCURSOR
:PTR activeWindow 0                                 /* Save attributes for window 2.*/
:FCT Main                                              w2 GETWINDOW
{
    0 :INT flop                                        LOOP
    CLEAR                                                 GETKEY :INT k
    "Type something (<esc> quits)" PRINT                  k 0x1b == BREAKIF
                                                          &flop NOT VAL IF
/* Make window 1. */                                         w1 &activeWIndow =
    1 3 20 7 2 WINDOW                                     ELSE
/* Normal.*/                                                 w2 &activeWindow =
    0 SETATTR                                             THEN
/* Full height cursor.*/                                  activeWindow PUTWINDOW
    2 SETCURSOR                                           k "%c" PRINT
/* Save attributes for window  1 */                       activeWindow GETWINDOW
    w1 GETWINDOW                                       ENDLOOP
                                                       0 SETATTR
/* Set normal attrib for border. */                    1 SETCURSOR
    0 SETATTR                                           1 1 DISPWIDTH DISPHEIGHT 0 WINDOW
/* Make window 2.*/                                 }
    22 3 40 7 2 WINDOW
```

```
Type something (<esc> quits)


  acegikmo█               bdfhjlnp


```
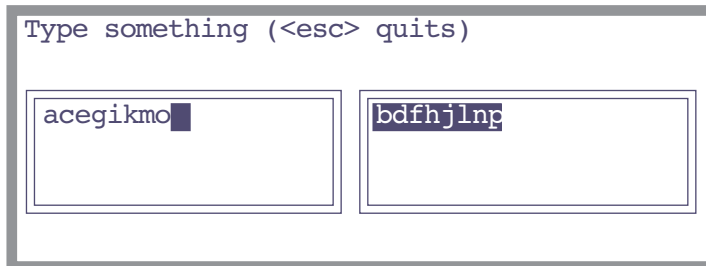
Figure 23-14. Window switching example. Every other character typed goes into the left window, and the other characters go into the right window.

If you run the program illustrated in Figure 23-14 and type "abcdefghijklm-nop", it will appear as shown above, with every character going into alternating windows. (The inverted square in the left hand window is the full cell cursor for that window.)

Manipulating Text

The keyword **CLEAR** removes the text from the current text window, and sets the display attribute for the entire window to the current value. The keyword **CLREOL** clears the text and sets the display attribute for a line of characters extending from the current cursor location to the right edge of the current window.

The attribute for a rectangle within the active window can be changed without changing the text by the keyword **BLKATTR**.

MOVETEXT will copy a rectangle of text within the active window to another location, also within the active window.

The keyword pair **GETTEXT** and **PUTTEXT** are used to copy text between the display (ignoring any windows) and an LPL CHAR array or PATH. Attribute is ignored, although **PUTTEXT** can be made to impose an attribute on the destination rectangle.

Frequently, it is desired to save the contents of the display beneath a window, so that when the window is removed, the display can be returned to the way it was before. **GETDISP** and **PUTDISP** accomplish this easily. It is important to understand the differences between **GETDISP**/**PUTDISP** and **GET-TEXT**/**PUTTEXT**: The _DISP functions deal with text and attribute information, while the _TEXT functions deal just with text. Also, the user takes care of the source or destination space for the _TEXT functions, while **GETDISP** allocates space, and **PUTDISP** disposes of it. Therefore, **GET-DISP** and **PUTDISP** should always occur in pairs, with **GETDISP** first, and you cannot **PUTDISP** twice using the same allocated space.

Buffering Updates

To increase efficiency, and reduce flicker, the user can buffer screen updates. That is, tell the operating system to wait to update the display (text and graphics) until a complete set of changes are made. The keywords **DISPUPDATE** controls this.

Keyboard Control

In LPL, the keyboard can be the source for or destination of generalized data flow, as described in **I/O Programming** on page 23-38. There are several LPL keywords that pertain to the keyboard itself, and these are described in this section.

Table 23-9. Keyboard Control Keyword Summary

Keyword	Description
BEEP	Run the beeper for a period of time
GETKEY	Get the next keystroke
GETKEYDEF	Define behavior during a getkey
KBDCLICK	Enable/Disable keyboard sound for each keystroke
KBDDELAY	Set the delay time for the keyboard (time before repeats)
KBDREPEAT	Set the key repeat frequency for the keyboard
NEXTKEY	What key is available (-1 if none)
UNGETKEY	Put back a keystroke

Keyboard Behavior

When you press a key on the LI-6400, there may or may not be a beep (controlled by **KBDCLICK** - default is no beep). If you hold the key down, there will be a slight delay (controlled by **KBDDELAY** - default 500 ms) followed by regular repetition (time interval controlled by **KBDREPEAT** - default 50 ms).

Keyboard Codes

Keys are identified in LPL by a 2 byte integer value. If the high byte is zero, then the key is an ascii key, and the low byte is the value. Thus, pressing the key **A** will generate a key code of 97 (hex 61), while **shift A** generates 65 (hex 41). If a non-ascii key is pressed, such as a cursor control key, or a function key, then the high byte is non-zero and contains the identifier for that key (Table 23-11 on page 30). The low byte (Table 23-12 on page 23-30) contains modifier key information (the state of the **shift** and **ctrl** keys). The file "/Sys/Lib/StdKeys" provides some LPL names for some non-ascii keys. Note that not all keys supported by LPL are found on any specific platform's keyboard.

Table 23-10. Example Key codes

Key Stroke	Key Code
x	0x0078
ctrl x	0x0018
pgup	0x2100
shift pgup	0x2101

Table 23-11. Non Ascii Key Codes: High Byte

The Key	Variable Name[a]	High Byte	The Key	Variable Name	High Byte
labels	_Labels	0x12	f1	_F1	0x60
home	_Home	0x24	f2	_F2	0x61
end	_End	0x23	f3	_F3	0x62
pgup	_PgUp	0x21	f4	_F4	0x63
pgdn	_PgDn	0x22	f5	_F5	0x64
↑	_UpArrow	0x26	f6		0x65
↓	_DnArrow	0x28	f7		0x66
←	_LeftArrow	0x25	f8		0x67
→	_RightArrow	0x27	f9		0x68
←	_Back	0x08	f10		0x69
escape	_Esc	0x1b	f11		0x6a
enter	_Enter	0x0d	f12		0x6b
ins		0x2d	f13		0x6c
del		0x2e	f14		0x6d
halt[b]		0x1d	f15		0x6e

a. Defined in the file /Sys/Lib/StdKeys
b. The **halt** key can also be generated by **ctrl escape**

Table 23-12. Non Ascii Key Codes: Low Byte

Bits 2 thru 7	Bit 1	Bit 0
unused	Control	Shift

Clock

LPL supports a real time clock.

Table 23-13. Clock Keyword Summary

Keyboard	Description
CTIME	Convert seconds, print time and date in formatted string.
DATE	Convert seconds to year, month, day
DAYOFWK	Convert seconds to day of week
DAYOFYR	Convert seconds to Julian day of year
GETMS	Get ms since power on
GETTDS	Get current date and time in seconds
SECS2TD	Convert seconds to time and date info
SETTDS	Set current date and time using seconds
TD2SECS	Convert time and date info to seconds
TIME	Convert seconds to hour, minute, second

Real Time

Time in LPL is measured from some base time that is platform specific. In the LI-6400, the base time is 1 Jan 1989.

A quick way to find the time base for any particular platform is the LPL sequence

```
0 CTIME
```

This will display the time and date corresponding to 0 time. Since the time is contained in a LONG, each platform has an upper limit of dates it can handle, which can be determined by

```
–1 CTIME
```

The current date and time (in seconds since the base time) is obtained by **GETTDS**, and set by **SETTDS**.

LPL provides several tools for doing time conversions. **DATE** converts sec-

onds since the base time to day, month, and year, while **TIME** extracts the hours, minutes, and seconds. **SECS2TD** converts seconds into date and time, while **TD2SECS** goes the other way. **DAYOFWK** and **DAYOFYR** converts seconds to day of the week and day of the year, while **CTIME** converts seconds to a readable date string.

Time since power on

For timing with a higher resolution than seconds, **GETMS** will return the time in milliseconds since power on or midnight, depending upon the platform.

Event Handling

Event handling is a powerful tool provided by LPL.

Table 23-14. Event Handling Keyword Summary

Keyword	Description
GETKEYDEF	Define behavior during a **GETKEY**.
HALT	Terminates an IDLE
IDLE	Wait for something to happen
OFFA2D	Cancel A/D converter interrupt
OFFCOMM	Cancel comm port interrupt
OFFCYCLE	Cancel regular timer interrupts
OFFKBD	Cancel keyboard interrupt
OFFSOFT	Cancel softkey interrupt
OFFTIC	Cancel 1 second interrupt
OFFTIME	Cancel real time clock one-time interrupt
ONA2D	A/D converter interrupt
ONCOMM	Incoming comm port interrupt (when a specified character arrives)
ONCYCLE	Regular timer interrupts
ONKBD	Keyboard action interrupt
ONSOFT	Softkey interrupt
ONTIC	1 second interrupt
ONTIME	Real time clock one-time interrupt
SLEEP	Main task stops execution for specified time.
TIDLE	Timed IDLE

One can write an LPL program that does some self-contained job, then quits. More often, however, it is necessary for the a program to be aware of the "outside world" - user keystrokes, incoming RS-232 data, ongoing analog measurements, etc. These events are not in general predictable - exactly *when* is

the user going to press a key, for example - so we need a method to "continually test" for the desired event, and if it has occurred, handle it.

Figure 23-15 illustrates (in "pseudo-LPL") how we might structure an LPL program to handle three types of events. Essentially, we sit in a tight loop waiting for something to happen, and when it does, we deal with it. The function `HandleKeys`, for example, would have to get the keystroke, and process it.

```
:FCT EventLoop
{
    LOOP
        KeysReady? IF HandleKeys THEN
        ClockReady? IF HandleClock THEN
        A/DReady? IF HandleA/D THEN
        ...
    ENDLOOP
}
KeysReady? {...}
ClockReady? {...}
A/DReady? {...}

HandleKeys { GETKEY ... }
HandleClock { ... }
HandleA/D { ... }
```

Figure 23-15. Illustration of the hard way to detect events

In order for this outline to work as an LPL program, we'd have to fill in the routines such as *ClockReady?*, to signal us at desired times or time intervals.

Fortunately, LPL hides much of this work from the programmer. It turns out that to actually implement the outline in Figure 23-15, one only needs to a) register the functions to be called, and b) write those functions. Figure 23-16

illustrates how this actually looks in LPL.

```
:FCT Main
{
    &HandleKeys ONKBD
    &HandleClock ONTIC
    &HandleA/D 1 ONA2D

    IDLE

}
HandleKeys { GETKEY ... }
HandleClock {...}
HandleA/D {...}
```

Figure 23-16. Programming for events in LPL.

We register the functions to be called using the **ON...** LPL keywords. **ONKBD**, for example, is for keyboard events. The handling functions are the same as before in Figure 23-15. But notice that the entire event loop is now collapsed into the LPL keyword **IDLE**.

IDLE is just what it sounds like - the LPL program is idle. The operating system, however, is performing the tight loop checking for events. Whenever it finds one for which we had registered a function, it calls that function for us.

LPL can handle a number of events (Table 23-15) in the manner outlined by the listing in Figure 23-16.

Table 23-15. Events Supported by LPL

Event	KeyWord
User presses any key on the keyboard	**ONKBD**
User presses a function key	**ONSOFT**
A certain time and date arrives (alarm clock)	**ONTIME**
1 second intervals	**ONTIC**
User defined intervals	**ONCYCLE**
A/D Readings are ready	**ONA2D**
A certain character's arrival in the Comm port	**ONCOMM**

The Function Keys

The function keys provide a valuable interface tool, in that the options open to a user at any given time can appear on the display as key labels.

Table 23-16. Function Key Keyword Summary

Keyword	Description
MAKESOFT	Make a softkey structure
FREESOFT	Free a softkey structure
HIDESOFT	Hide softkey labels
SHOWSOFT	Show softkey labels
SOFTSETM	Set current menu level
SOFTGETM	Get current menu level
SOFTWIDE	How many softkeys can be shown at once?
GLOBALKEYS	How should softkeys behave across nested IDLEs?

Even though the specific platform on which LPL is installed has a fixed number of physical function keys (the LI-6400 has 5, for example), you can define any number of function keys in LPL; the user can only see them in groups of 5 (or whatever) at a time, but the **labels** key on the keyboard (or the equivalent) will change the displayed group. Groups can also be selected by program.

The function key tools described in this section work during **IDLE** (See "Event Handling" on page 33). The strategy is to define how many function keys you want to "create" (**MAKESOFT**), then what each key's label and function is (**ONSOFT**). During **IDLE**, the user can scroll through the keys (if there are too many to show at one time) using the labels key, and if he presses one, the function happens automatically. **OFFSOFT** and **FREESOFT** disable and dispose of function key definitions. Figure 23-17 illustrates.

```
:FCT Main                                    }
{
    /*  Illustrate use of function keys      /* When FctKey 1 pressed */
    */                                       PrintTime
                                             {
    CLEAR                                        1 1 POSXY
                                                 GETTDS TIME "Time is %2d:%2d:%2d"
    0 ' ' 1 2 0 5 MAKESOFT                   PRINT
    /* 0 - plain text for label              }
delimiter
      ' ' - the delimiter is a space         /* When Fct Key 2 pressed */
       1 - inverse labels                    PrintDate
       2 - labels 2 lines high               {
       0 - don't save background beneath         1 2 POSXY
the labels                                       GETTDS DATE "Year: %d Month:%d
       5 - total of 5 function keys          Day:%d" PRINT
allowed                                      }
    */
                                             /* When Fct Key 3 pressed */
/* Fct Key 1 */                              PrintCTime
    "PRINT  TIME" &PrintTime 1 ONSOFT        {
/* Fct Key 2 */                                  1 3 POSXY
    "PRINT  DATE" &PrintDate 2 ONSOFT            GETTDS CTIME
/* Fct Key 3 */                              }
    "PRINT Ctime" &PrintCTime 3 ONSOFT       /* When Fct Key 5 pressed */
/* Fct Key 5 */                              Quit { HALT }
    "      quit" &Quit 5 ONSOFT
    SHOWSOFT

    IDLE

    FREESOFT
```

Figure 23-17. Defining function keys

There are some other options with function keys. For example, should the labels appear all the time, or just when the user presses the **labels** key? Should the current level of function keys be displayed in the lower left hand corner of the display? Both of these are controlled by parameters in the call to **MAKESOFT**.

Another question is this: what happens to the function key definitions while a function key is being processed (that is, while the function associated with a key is being executed)? Do the key labels stay visible? Should the function key's definition be allowed to change? If another **IDLE** is encountered, do the old key definitions stay active? When the **IDLE** is done, do the old key definitions get restored? All of this is controlled by one flag set with the keyword **GLOBALKEYS**.

- **GLOBALKEYS on:**
  Function key definitions remain active and labels stay visible while function keys are serviced, and while subsequent **IDLE**s are performed. Function keys can be redefined without exiting the **IDLE**.

- **GLOBALKEYS off:**
  While a function key is serviced, the labels disappear, and all function key definitions are "forgotten". When the service is done, the definitions are restored. This means that if you want nested **IDLE**s with new function key definitions, you must do a new **MAKESOFT**. But when the child **IDLE** is done, the old key definitions are automatically restored. When **GLOBALKEYS** is off, you cannot change a function key definition (label or function) without exiting the **IDLE**.

I/O Programming

The ability to move information to and from devices is critical to any computer, and LPL provides an abundance of tools for doing that task.

Table 23-17. I/O Keyword Summary

Keyword	Description
BENTER	Binary enter from a path
BPRINT	Binary print to a path
CLOSE	Close any path
CONVERTIN	Set incoming (writing to) path filter(s)
CONVERTOUT	Set outgoing (reading from) path filter(s)
ENTER	Read formatted or unformatted data from a path
GETCH	Get a byte from a path
GETCONVERTIN	Get current incoming filters
GETCONVERTOUT	Get current outgoing filters
GETDFCIN	Get current default file convertin
GETDFCOUT	Get current default file convertout
IOCLEAR	Clear path error
IOERR	Get latest path error

Table 23-17. I/O Keyword Summary

Keyword	Description
ISEMPTY	Is the path empty
OPEN_BUFF	Open a path to an expandable memory buffer
OPEN_CHARS	Open a path to a character array
OPEN_COMM	Open a path to the comm port
OPEN_FILE	Open a path to a file
OPEN_FILE_ASK	Open a path to a file, with handy user front end.
OPEN_KBD	Open a path to the keyboard.
OPEN_LCD	Open a path to the display
OPEN_QUE	Open a path to a circular queue
PATHSTAT	Get path status
PRINT	Send formatted or unformatted data to a path
PUTCH	Write a byte to a path
RESET	Set fill and empty pointers to 0.
RESETDFC	Reset (to platform default) default file convert in and out
SEARCH	Search a path for the occurrence of something
SETDFCIN	Set default file convertin
SETDFCOUT	Set default file convertout
SETEMPTY	Set empty (reading) pointer for a path
SETFILL	Set fill (writing) pointer for a path
UNGETCH	Character put back function for a path
XFER	Send data from one path to another
PRCOUNT	Parser: how many items from last **PRNEXTLINE**
PRDELIM	Parser: set the delimiter list
PRLINE	Parser: return the last line parsed
PRNEXTLINE	Parser: read next line and parse it

Table 23-17. I/O Keyword Summary

Keyword	Description
ISEMPTY	Is the path empty
OPEN_BUFF	Open a path to an expandable memory buffer
OPEN_CHARS	Open a path to a character array
OPEN_COMM	Open a path to the comm port
OPEN_FILE	Open a path to a file
OPEN_FILE_ASK	Open a path to a file, with handy user front end.
OPEN_KBD	Open a path to the keyboard.
OPEN_LCD	Open a path to the display
OPEN_QUE	Open a path to a circular queue
PATHSTAT	Get path status
PRINT	Send formatted or unformatted data to a path
PUTCH	Write a byte to a path
RESET	Set fill and empty pointers to 0.
RESETDFC	Reset (to platform default) default file convert in and out
SEARCH	Search a path for the occurrence of something
SETDFCIN	Set default file convertin
SETDFCOUT	Set default file convertout
SETEMPTY	Set empty (reading) pointer for a path
SETFILL	Set fill (writing) pointer for a path
UNGETCH	Character put back function for a path
XFER	Send data from one path to another
PRCOUNT	Parser: how many items from last **PRNEXTLINE**
PRDELIM	Parser: set the delimiter list
PRLINE	Parser: return the last line parsed
PRNEXTLINE	Parser: read next line and parse it

Table 23-17. I/O Keyword Summary

Keyword	Description
PRQCOUNT	Parser: How many quoted items in last **PRNEXTLINE**
PRQUOTE	Parser: Set quote characters

Paths

We have seen how **PRINT** can display information on the display, and **EN-TER** can retrieve information from the keyboard. This section will extend your appreciation for **PRINT** and **ENTER**, because we can use these same tools to deal with files, the comm port, buffers in memory, and sometimes even the keyboard.

The unifying concept in all of this is the notion of a Path, which can be thought of as a "data tube". **PRINT** puts data into the tube, **ENTER** takes data out of the tube, and we use **PRINT** and **ENTER** without being overly concerned about where the tube, or Path, *leads*. Thus, **PRINT** and **ENTER** concern themselves with the Path, while the operating system takes care of all the messy details involved with the various devices or memory that are at the *other* end of the Path. LPL supports Paths to a variety of devices (Table 23-18). When an application is launched, four standard paths are provided, and are accessible by the keywords **LCD**, **COMM**, **ARGS**, and **KBD**. Alternate paths to these devices, or paths to other devices are opened by the various **OPEN_** keywords shown in Table 23-18.

Table 23-18. Paths Supported by LPL

Device	Created by	Standard Path Name
Display	**OPEN_LCD**	**LCD**
Keyboard	**OPEN_KBD**	**KBD**
Comm Port	**OPEN_COMM**	**COMM**
Expandable memory buffer	**OPEN_BUFF**	-
Circular memory queue	**OPEN_QUE**	-
CHAR Array	**OPEN_CHARS**	-
File	**OPEN_FILE** **OPEN_FILE_ASK**	-

A simple example of using Paths is shown in Figure 23-18, which writes a line of text to a file. Files are treated more thoroughly in the section **File System** on page 23-47.

```
:FCT main
{

Open file "/user/MyFile" for writing.
    "w" "/User/MyFile" OPEN_FILE IF RETURN THEN
    :PTR file

    "This is a message." file PRINT

Close the path.
    file CLOSE
}
```

Figure 23-18. Accessing a file with a path.

Path Registers

Every path has associated with it four status registers (Table 23-19). The **SETFILL** and **SETEMPTY** keywords allow direct manipulation of the fill and empty registers, whose present value can be determined by **PATHSTAT**. For example, to re-read a file from the beginning, one would set the empty register to 0 (the start of the file). Some operations adjust the registers implicitly; **PUTCH** adds a character to the fill location, and updates the register, while **GETCH** takes a character from the empty location, updating the register.:

Table 23-19. PATH Status Registers

Name	Description
TYPE	Describes what the path is pointing to (Table 24-31 on page 24-73).
SIZE	Number of bytes
EMPTY	Location of next byte of outgoing data
FILL	Location of next incoming byte's destination.

Path Filters

A very powerful feature of paths is the character filtering capability that they have. Filters can be independently enabled for reading data from a path, and for writing data to a path. Table 23-20 indicates the filters available for LPL paths.

Table 23-20. Path Filters

Code	Convert From	Convert To
e	<cr>, <lf>, or <cr><lf>	<lf>
E	<cr>, <lf>, or <cr><lf>	<cr><lf>
n	<cr>, <lf>, or <cr><lf>	<cr>
tn	n spaces (n=0...16)	tab
Tn	tab	n spaces (n=0...16)
s	\xdd or \ddd	ascii character
S	nonprintable characters	\xdd
k	Byte pairs	If 1st byte is null it is stripped; otherwise, \k sequence output.
K	Does reverse of "k" filter: bytes are padded with nulls, except for the \k sequence.	Byte Pairs
Cxy	character x	character y
cxy	1 or more contiguous characters x	character y

Note that filters can be chained together in any sequence. The keywords **CONVERTIN** and **CONVERTOUT** define the filtering (if any) for data inflow and outflow for any path. **GETCONVERTIN** and **GETCONVERTOUT** will return the current filter sequence for any path.

```
:FCT Main
{


Open a path to the display.
    OPEN_LCD IF RETURN THEN
    :PTR disp

Define 8 column tabs.
    "T8" disp CONVERTIN

Send some data with imbedded tabs to 'disp'.
    "123456781234567812345678\nabc\tdef\tghi" disp PRINT


    GETKEY DROP
Close the path 'disp'.
    disp CLOSE
```

Figure 23-19. Simple Filter Example

will produce the lines

```
123456781234567812345678
abc     def     ghi
```

Transferring Data

There are several tools for moving data into or out of paths. **GETCH** and **PUTCH** do a byte at a time. **PRINT** and **ENTER** move formatted ascii data between paths and LPL variables. **BPRINT** and **BENTER** move the values of LPL variables in binary form to and from paths. **XFER** transfers data from one path to another, and (depending on what type of paths are involved) can do it *simultaneously* with LPL program execution. For example, with an overlapped **XFER**, a file can be sent out the comm port at the same time that measurements are being taken. As an example of data transfers, consider Figure 23-20, a simple terminal emulator program.

```
:FCT Main
{

/* Incoming data -> display.*/
    COMM LCD XFER
Typed chars -> comm port.
    KBD COMM XFER
    &Keys ONKBD

    IDLE
}
Keys
{
    GETKEY :INT k
/* Quit if <esc> pressed.*/
    k 0x1b == IF HALT THEN
/* "Echo" the character to the display.*/
    k "%c" PRINT
}
```

Figure 23-20. Simple terminal emulator

Formatted ASCII I/O

Formatted output includes right and left justification, number of significant digits, leading and trailing spaces or zeros, rounding, delimiters between array elements, and other features. These features are controlled by the format string expected by **PRINT**. Formatted input, controlled by the format string expected by **ENTER**, define the rules by which the numbers and strings are extracted from input data.

I/O Errors

The keyword **IOERR** returns the error (if any) that has most recently happened for any path. Possible errors are given in Table 23-21. The keyword **IOCLEAR** resets the error number for a path to 0 (no error).

Table 23-21. I/O Error codes

Error Number	Condition
0	No error
1	Attempting to write to a full path
2	Attempting to read from an empty path
3	Attempting to move a pointer outside of a path's limits

Parsing

LPL 5 introduced a powerful addition to Paths: the Parser. The keyword **PRNEXTLINE** read the next line from the path, and parses it. Other PR keywords retrieve the items parser, set the delimiter characters, etc.

File System

The LPL File System supports a hierarchical directory structure, and provides a number of controls for files, directories, and disks.

Table 23-22. File System Keywords

Keyword	Description
CFSTATE	Is compact flash device present?
CFUNMOUNT	Unmounts compact flash device.
DEVNAME	Returns the hardware location of a disk
DIRALL	Get list of all directories in the file system
DIRCURR	Get current working directory
DIRERASE	Erase a directory
DIRMAKE	Make a directory
DIRSAVE	Make sure file system is up to date
DIRSET	Set current working directory
DSKFORMAT	Format a disk (LPL 4 and below)
DSKISSWAP	Is a disk off-line? (LPL 4 and below)
DSKOFFLINE	Take a disk off-line (LPL 4 and below)
DSKONLINE	Take a disk on-line (LPL 4 and below)
DSKPACK	Defragment a disk (LPL 4 and below)
DSKSPACE	How much space on a disk
FCOPY	Copy a file
FERASE	Erase a file
FGETTDS	Get file time and date

Table 23-22. File System Keywords

Keyword	Description
FGETWP	Get write protect status of a file or directory
FLIST	Get a list of files and/or directories
FMERGE	Merge path and file names into file specifier
FMOVE	Move a file
FPARSE	Parse file specifier into path and file names
FRENAME	Rename a file
FSETTDS	Set file time and date
FSETWP	Write protect a file or directory
FSIZE	Get file size
FTRASH	Returns the trash location
FTYPE	Determine if exists, or file, or directory
FUPDATE	Update changed portion of a file
FWPICK	Pick file from a list of possibilities
OPEN_FILE	Open a a path to a file.
OPEN_FILE_ASK	Open a path to a file, using Standard File Dialog box.

Hierarchical Structure

The LPL file system should be visualized as a tree: Tied to a common root are one or more directories. Within each directory there may be files and/or more directories.

An illustration of a file system containing three disks, named Dir A, Dir B, and Dir C is shown in Figure 23-21.
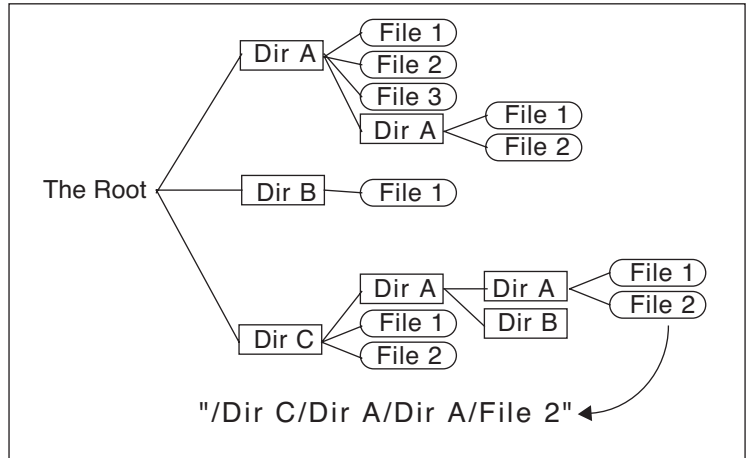
Figure 23-21. File system illustration.

Directory C contains 2 files and 1 directory, which in turn contains two directories, the first of which has two files

Working With Files

Before data can be read from or written to files, they must be opened (**OPEN_FILE** or **OPEN_FILE_ASK**). When a file is opened, a Path is created, and all data transfer operations take place through the path (**I/O Programming** on page 23-38). Files can be opened with various combinations of attributes: write access, read access, and write-append. Am example program is shown in Figure 23-22.

```
:FCT TestReadWrite
{
"w" "/user/testFile" OPEN_FILE NOT IF
    :PTR file

    "This is a test\n" file PRINT
    "This is the end of the test" file PRINT

    file CLOSE
THEN

"r" "/user/testFile" OPEN_FILE NOT IF
    :PTR source

    LCD source XFER

    source CLOSE
THEN
}
```

Figure 23-22. Example of programming file reading and writing. This function writes two lines to a file, then copies the contents of the file to the display.

Other file operations can occur on files and directories without opening them, such as those provided by the Filer (copying, moving, erasing, etc.). LPL provides keywords to do those same operations from a running program (**FCOPY**, **FMOVE**, **FERASE**). File time stamp information can be read or set using **FGETTDS** and **FSETTDS**.

The Trash Directory
When files are deleted via the Filer, they are moved to a Trash directory. A trash directory will exist for each disk that has had at least one file removed. Please note that the Trash directory is a <u>Filer implementation only</u> - removing a file from a running LPL program by using the **FERASE** keyword does <u>not</u> move the file to the Trash. If you wish to use the trash instead of **FERASE**, then use this code sequence:

```
fileName DUP FTRASH FMOVE
```

Networking in LPL

Table 23-23. Networking-related keywords

Keyword	Description
NTSTATE	Get network card state
NTGETHOST	Get host name
NTSETHOST	Set host name
NTGETIP	Get IP address
NTGETMAC	Get Mac address
NTSETPASS	Set password
NTGETUSER	Get user name. (Returns lpl).
NTSETUSER	Set user name. (Not implemented)
NTCLIENTS	Get number of remote clients
NTOPENCLT	Make a client connection
NTCLOSECLT	Close a connection
NTOPENSRV	Start a service
NTCLOSESRV	Stop a service
NTINFO	Get information on clients or services

Many of these LPL commands are used in the program /Sys/Utility/NetworkConfig, which is described in **The Network Status Program** on page 11-9.

Services

The LI-6400XT provides a service on port 6400 with an expected protocol for handling keystrokes, display updates, file transfers, etc. To use port 6400 requires a custom, dedicated client program (we provide LI6400XTerm, LI6400Group, li6400.licor.com server, and the iOS apps) at the other end.

The LI-6400XT also provides a simplified service on port 6409 which can work with a generic program like Telnet. Anything the LI-6400XT sends out the comm port is also sent all port 6409 clients, and anything received from

those clients is treated like incoming RS-232 data. (Note: comm traffic is also handled with clients on port 6400, but it is bundled into packets, and requires the client software to unpackage it. On 6409, no such bundling is done.).

6400 and 6409 services normally start at power up, and remain running while the instrument is powered on. You can manually stop a service (and abruptly terminate any remote clients) with the LPL command NTCLOSESRV. To stop the 6400 service, for example, do

```
6400 NTCLOSESRV
```

and restart it again by

```
6400 NTOPENSRV
```

You can tell what services are running by looking at the More... option in the Network Status program (Figure 11-9 on page 11-10). You can do this programmatically by using the NTINFO command.

```
<path> 3 NTINFO
```

sends a list of active services to <path>, and

```
6409 4 NTINFO
```

which returns a 1 or 0, depending upon if the specified service is active or not

Remote Clients
The information about remote clients that are connected to the LI-6400XT is available from NTINFO and NTCLIENTS.

Local Clients
The LI-6400XT can become a client to an external service via the NTOPEN-CLT command. This is used when connecting through li6400.licor.com. The corresponding command to close the connection is NTCLOSECLT.

Menus and Editors

LPL provides some high level user interface tools for selecting an item from a list (a menu) and for entering numbers or text (an editor). Further, users can customize the behavior of either of these tool types.

Table 23-24. LPL Menu and Editor Keywords

Keyword	Description
MESSBOX	Put up a message in a window
STDLINE	Single line edit
STDEDIT	Multiple line edit
STDMENU	Standard menu function
SETTARGET	Set editor search target
GETTARGET	Return current editor search target
EDOPEN	Set up a custom editor
EDCLOSE	Dispose of a custom editor
EDCTL	Perform an editor function
EDSTAT	Get some editor information
EDWRITE	Custom edit function
EDKEY	Custom edit function
FILER	Access the Filer.
FEDIT	Edit a file (< 64000 bytes)
FVIEW	View a file (any size)

Standard Menus and Editors

Standard Menu
The keyword **STDMENU** displays a list of items in the current text window, and allows the user to make a selection. Options available for **STDMENU** include a menu bar (highlighted band that moves up and down with the cursor), what ascii keys cause exit, what non-ascii keys cause exit, and if/how to adjust the cursor on exit.

Figure 23-23 creates a window, and uses **STDMENU** to display the contents of a CHAR array in that window as a menu with a menu bar. The user can scroll through the contents, then exit using **escape** or **enter**.

```
:CHAR menuItems[] "This is item #1
This is item #2
and item #3
and 4
and 5
and 6\nand 7\nand 8\nand lastly, 9"

:FCT Main
{
/* Make a window */
    5 1 25 8 2 2 "My Menu" WINDOW

/* Use a menu bar, and  exit on esc or enter */
    1 "" "\r" menuItems STDMENU DROP

/* Cleanup the display */
    1 1 DISPWIDTH DISPHEIGHT 0 WINDOW
    CLEAR
}
```

```
┌──────────My Menu──────────┐
│ This is item #1           │
│ This is item #2           │
│ and item #3               │
│ and 4                     │
│ and 5                     │
│ and 6                     │
│ Print ◆ Find  ◆ ReFind◆ CANCEL◆ SELECT │
└───────────────────────────┘
```
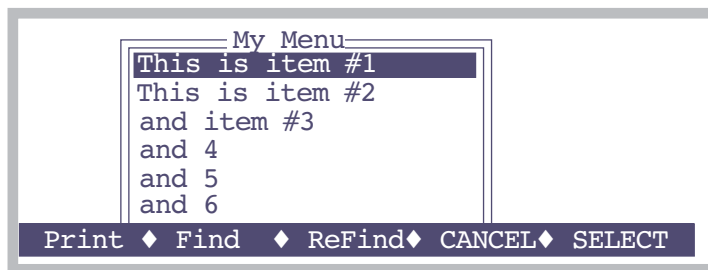
Figure 23-23. Programming with **STDMENU**.

The ↑ and ↓ keys move the highlighted menu bar up and down. In general, the cursor control keys defined for the standard menu function are given in Table 5-2 on page 5-5.

What was selected?
Something very important is missing in Figure 23-23: there is no way to tell what was selected. To do this, we must pass the **STDMENU** function a Path instead of a simple CHAR array, because **STDMENU** uses the Path status registers to tell us where the cursor was when the user exits. We fix this problem in Figure 23-24 by making a Path to the menu items, then after **STDMENU** is finished (if the user pressed **enter**) we read in the selected line.

```
:CHAR menuItems[] "This is item #1
This is item #2
and item #3
and 4
and 5
and 6\nand 7\nand 8\nand lastly, 9"
:FCT Main
{
    5 1 25 8 2 2 "My Menu" WINDOW

/* Open a path to the CHAR array.*/
    menuItems OPEN_CHARS IF RETURN THEN
    :PTR menu

/* Use menu bar, exit on esc or enter, move cursor to left. */
    3 "" "\r" menu STDMENU :INT k

    1 1 DISPWIDTH DISPHEIGHT 0 WINDOW
    CLEAR

/* If user pressed enter...*/
    k '\r' == IF
        0 :CHAR line[80]
/* Read the selected entry.*/
        line "%(.)c" menu ENTER DROP
        line "You chose: '%s'" PRINT
        GETKEY DROP
    THEN
}
```

Figure 23-24. STDMENU example: knowing what was selected.

We can tell if the user pressed **escape** or **enter** by checking the return value of **STDMENU**. We read the selected line using **ENTER**, since **STDMENU** left the empty register of the Path at the last cursor location. What if the user has scrolled to the right, and the cursor was not on the left column? We prevent this from being a problem by setting the line

```
    3 "" "\r" menu STDMENU
```

The 3 indicates a menu bar, and also to snap the cursor to the left column on exit.

Single Line Editor
The keyword **STDLINE** creates a borderless window from the current cursor position to the right edge of the current window, displays in this window the Text object to be edited, and lets the user edit the object. **escape** or **enter** terminate **STDLINE**.

```
:CHAR fileName[80] "myData"
:FCT Main
{
    "Enter a file name:" PRINT
    fileName STDLINE IF
        fileName "\nname is '%s'" PRINT
    ELSE
        "\nName not changed!" PRINT
    THEN
    GETKEY DROP
}
```

Figure 23-25. Using STDLINE.

Multiple Line Editor

The keyword **STDEDIT** invokes the system editor for a Text object. If a file name is specified, the system editor's Exit Menu will be accessible.

FEDIT or **FVIEW** provide a quick way to invoke the system editor on a file.

Customized Editors and Menus

The keywords **EDOPEN**, **EDCTL**, **EDWRITE**, **EDKEY**, and **EDCLOSE** allow you to create menus and editors with customized behavior. Also, **SET-TARGET**, **GETTARGET** allow program control over searching for targets in an editor or menu. The Display Editor in OPEN is an example of a customized menu. This routine shows the makeup of OPEN's display lines, and allows the user to edit them. An abbreviated listing of this routine is shown in Figure 23-26. Note the inclusion of the file "/Sys/Lib/CEDefs", which contains useful tools for these operations.

```
/**********************
                                          DEBuildMenu
    DisplayEditor (partial)               {
                                             :PTR menu
    Called from OPEN                         ...
                                          }
**********************/
:IFNDEF CERefresh                         DEKeys
    :INCLUDE "/sys/lib/CEDefs"            {
:ENDIF                                       GETKEY :INT k

:PTR menu 0 /* our data path */              k _Esc == IF DEQuitTest RETURN THEN
        edPtr 0 /* custom data struct */
                                             k 127 < IF /* is it ascii? */
:FCT Main                                        &k LWC DROP
{                                         /* If 'a' thru 'z', jump to that format
/* Make a path for the menu contents */   line.*/
    1000 OPEN_BUFF IF RETURN THEN               k 'a' >= k 'z' <= AND IF
    &menu =                                         edPtr CEHomeToggle
                                                    k 'a' - 1 + workingFmtList
/* Make a window. */                      READY MIN
    1 1 DISPWIDTH DISPHEIGHT 2 2                     edPtr CEJumpToLine
        "Display Editor" WINDOW                      edPtr CEToggleMenuBar
                                                THEN
/* Fill the content path.*/                     ELSE
    menu DEBuildMenu                      /* If it's non-ASCII, pass it through to
                                          EDKEY */
/* Create a custom menu structure.*/                edPtr CEToggleMenuBar
    menu EDOPEN IF RETURN THEN                       k edPtr EDKEY
    &edPtr =                                         edPtr CEToggleMenuBar
                                                THEN
    &DEKeys ONKBD                             }
    DESoftKeys
/* Turn on the menu bar.*/                    DESoftKeys
    edPtr CEToggleMenuBar                     {
                                                 0 ' ' 1 1 0 10 MAKESOFT
    IDLE
                                          /* We'll just illustrate Fct Key 1 */
    ResetWindow                                  " Edit" &DEEditLine 1 ONSOFT
    edPtr EDCLOSE                                 ...
    menu CLOSE                                    SHOWSOFT
}                                             }
/* Write the initial contents of the menu
to the path passed to this function. */
```

Figure 23-26. Partial listing of OPEN's Display Editor, which uses a custom editor based on EDOPEN.

ListBox

LPL 5 introduced the ListBox, which combines window control, function keys, and editing into a convenient interface tool.

Table 23-25.

Keyword	Description
LBNEW	Create a list box
LBFREE	Destroy a list box
LBSTART	Pick the starting cursor line
LBCOUNT	Returns number of items in the list box.
LBEXECUTE	Let the user interact
LBSEL	Returns the selected line
LBGETSTR	Returns the ith string
LBGETOBJ	Returns the ith object
LBSETSTR	Sets the ith string
LBSETOBJ	Sets the ith object
LBCLEAR	Clears the contents
LBAPPEND	Adds a line
LBDEFKEY	Define ascii or softkeys
LBWINDOW	Defines the window and border
LBBORDER	Defines the border
LBERASE	Hides the softkey labels
LBGETNODE	Returns node map or value for specified line
LBSTARTNODE	Sets pointer line by node map

Real Time Graphics

LPL 5 introduced the Real Time Graphics manager, which provides a convenient method of doing graphics without resorting to the lower level graphics commands describe below in **Graphics** on page 23-61.

Table 23-26. RTG keywords

Keyword	Description
RTGNEW	Create a new RTG manager
RTGMAKE	Redefine an RTG manager
RTGFREE	Destroy an RTG manager
RTGDEF	Define a plot (can have up to three per manager)
RTGCLEAR	Flush data values for a curve
RTGSTART	Draw axes
RTGADD	Add snapshot to strip charts
RTGADD2	Data arrays to be added if the variable is plotted
RTGLOG	Add point to XY curves; mark strip charts
RTGEDIT	User editing
RTGNAME	Name a manager
RTGREAD	Read definition
RTGWRITE	Write definition
RTGVARS	Associate a variable NameList
RTGTIME	Set strip chart time range
RTGSCROLLX	Scroll strip chart time axis
RTGSCROLLY	(does nothing)
RTGSELECT	Select a plot
RTGACTIVATE	Bring it into view

To illustrate RTG managers, we present the relevant functions associated with the graphics during OPEN's IRGA Zeroing routine:

```
/* Creating and destroying
*/
MakeZeroRTGs
{
    OpenVarNameList "/User/Configs/Zero" "CW" 'C' 0 5 RTGNEW
&cRtg =
    OpenVarNameList "/User/Configs/Zero" "CW" 'W' 1 5 RTGNEW
&wRtg =

    "QUIT" &ExitGraph 5 cRTG ONSOFT
    "QUIT" &ExitGraph 5 wRTG ONSOFT
    "VIEW H2O" &ViewH2O 1 cRTG ONSOFT
    "VIEW CO2" &ViewCO2 1 wRTG ONSOFT

    &cRTG zRTGs 1 PICK =
    &wRTG zRTGs 2 PICK =

    -1 0 1 1  4 0 120 600 1 1 cRtg RTGDEF
    -2 0 1 1  4 0 120 600 1 2 cRtg RTGDEF

    -4 0 1 1  4 0 120 600 1 1 wRtg RTGDEF
    -5  0 1 1  4 0 120 600 1 2 wRtg RTGDEF

    cRtg RTGSTART
    wRtg RTGSTART

    1 &zPlotting =
    0 GSHOWPORT
    0 GSETPORT

}

FlushZeroRTGs
{
    1 GSETPORT GINIT GCLEAR
    0 GSETPORT GINIT GCLEAR
    0 GSHOWPORT

    cRTG RTGFREE
    wRTG RTGFREE

}

ZeroRTGAction
{
    zPlotting IF
        cRTG RTGADD
        wRTG RTGADD
    THEN
}
```

Graphics

LPL provides graphics tools for drawing lines, plotting points, etc. on a display device.

Table 23-27. Graphics Keyword Summary

Keyword	Description
ALPHA	Show / hide text display
GBOX	Clear / frame / fill a rectangle (user units)
GCLEAR	Clear graphics window
GDRAW	Graphics pen absolute draw (user units)
GETALPHA	Is alpha visible?
GETGRAPH	Is graph visible?
GGETPORT	Which port is the active one?
GHEIGHT	Height of current graphics window
GICON	Plot a 5x5 image centered at pen location (user units)
GIGET	Copies a graphics image to a buffer (user units)
GINIT	Initializes / Resets graphics
GIPUT	Puts a graphics image on the display (user units)
GISIZE	Gets required size of a graphics image (user units)
GLABEL	Label in graphics mode at pen position
GLSIZE	Determine height and width in pixels of a label
GMDGET	Gets graphics drawing mode
GMODE	Sets graphics drawing mode
GMOVE	Graphics pen absolute move (user units)
GPGET	Get a pixel (user units)
GPLOT	Executes move/draw data stored in a structure.
GPPUT	Put a pixel (user units)
GRAPH	Show / hide graphics display

Table 23-27. Graphics Keyword Summary

Keyword	Description
GRDRAW	Graphics pen relative draw (user units)
GRMOVE	Graphics pen relative move (user units)
GSCALE	Scale graphics area (user units)
GSCGET	Gets current scaling (user units)
GSCROLL	Enable/Disable auto scrolling
GSETPORT	Select a port to be the active one
GSHIFT	Shift a rectangular region (user units)
GSHOWPORT	Make a port the potentially visible one
GSTATUS	Retrieves the number of ports, and the visible one
GTMDGET	Gets graphics-text interaction mode
GTMODE	Sets graphics-text interaction mode
GWHERE	Gets current pen location (user units)
GWIDTH	Width of current graphics window
GWIGET	Get current graphics window (Display Pixels)
GWINDOW	Define graphics window (Display Pixels)

Graphics Hardware

Note that the graphics display and the text display are conceptually different, and may in fact be physically different as well, depending upon the platform. On the LI-6400, the graphics and text displays use the same LCD display, but this device has two separate modes of operation; text and graphics can be shown separately, or together. In the Macintosh implementation of LPL, the text and graphics come out in two separate windows.

The keywords **GRAPH** and **ALPHA** turn on/off the graphics and text displays. **GETALPHA** and **GETGRAPH** return the state of each display: on (visible) or off (not visible). On platforms where text and graphics share common hardware, the "interaction" between text and graphics images is set by **GTMODE** and sensed by **GTMDGET**, and can be any of the following: *or*, *exclusive or*, and *and*.

Note that on the display of the LI-6400, there is "interference" between text highlights (inverse and blinking) and graphics mode: text modes will not be shown when graphics mode is also on.

Graphics Ports
LPL 5 introduced the idea of multiple graphics ports. The active port is the one that all graphics commands affect. This is set by **GSETPORT**, or sensed by **GGETPORT**. This is independent of the port that is potentially visible. That is set by **GSHOWPORT** and sensed by **GSTATUS**, which also indicates the number of graphics ports available.

Windows and Coordinates
In a graphics window, the smallest element is a <u>pixel</u>. The Pixel Hardware Coordinates (<u>PHC</u>) has pixel (0,0) as the lower left pixel in the physical display window. In Pixel Window Coordinates (<u>PWC</u>), pixel (0,0) is the lower left pixel in the display window (a subset of the physical display). Finally, there are User Coordinates (<u>UC</u>), whereby the user can scale the current plotting window to any range of real numbers that is desired.

The keywords **GHEIGHT** and **GWIDTH** return the height and width of the current graphics window. The default graphics window (at power on and after the reset function **GINIT**) is full size: it fills the physical graphics display. The window can be reduced by the keyword **GWINDOW**, which specifies a new graphics window and PWC coordinate system. To find the current graphics window, use **GWIGET**. To scale a graphics window to user coordinates, use **GSCALE**. Its counterpart, **GSCGET** returns the current scaling parameters. Figure 23-27 illustrates the use of some of these graphics key words for scaling.

```
:FLOAT scale[4] {0 1.1 360 -1.1}          GWINDOW

:FCT Main
{                                          /* Re-draw sine curve */
/* Reset the scaling. */                      DrawACurve
    GINIT
/* Turn off text, turn on graphics,        /* Admire the results */
clear the graphics display. */                GETKEY
    1 GRAPH 0 ALPHA                            1 ALPHA 0 GRAPH
    GCLEAR                                  }

                                           DrawACurve
/* Get the dimensions of the graphics      {
display. */                                /* Frame the graphics window */
    GWIDTH :INT pixelsWide                     1 scale GBOX
    GHEIGHT :INT pixelsHigh
                                               0 :INT angle
    DEG                                    /* Move to 0, sin(0) */
/* Scale the plotting area.*/                  0 0 GMOVE
    scale GSCALE                           /* Draw a sine curve */
                                               36 NLOOP
/* Draw a sine curve */                            angle DUP SIN GDRAW
    DrawACurve                                     &angle 10 + DROP
                                               ENDLOOP
/* New window in the lower left 1/3 of     }
the display */
    0 pixelsHigh 3 / pixelsWide 3 / 0
```
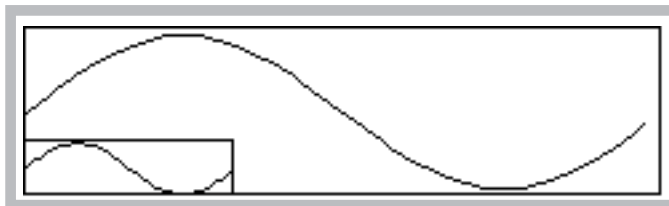

Figure 23-27. Graphics windows illustration, and the program m that generated it.

Drawing and Labelling

Single pixel access is available through **GPGET** and **GPPUT**. More useful, higher level routines are available, however. Figure 23-27 illustrated the fundamentals of drawing in graphics mode when it draws the sine curves. The basis of drawing is a hypothetical graphics pen, whose location is set by **GMOVE** (go to a certain location without drawing) and **GDRAW** (draw from the present location to the given location). There are also a relative move and

draw functions **GRMOVE** and **GRDRAW** that move the pen relative to its current location. The current pen location can be found with **GWHERE**.

GBOX is used to frame, fill, or clear a rectangle. **GPLOT** requires an array of X-Y coordinates, and it can either draw a line connecting them, or plot a character at each coordinate pair.

Labels and symbols can be done with **GLABEL**, which draws a string starting at the pen location, and **GICON**, which plots a 5x5 pixel image at the pen location. **GLORG** specifies where the label is to be relative to the current pen location (Figure 23-28). Thus, if **GLORG** is 8, the label will appear to the left of the pen location, centered vertically. **GLSIZE** returns the height and width of a label in pixels, which can be an aid to positioning it.
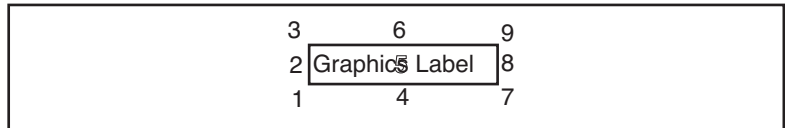

Figure 23-28. **GLORG** *locations for a graphics label relative to pen location*.

```
:FLOAT angle[50] { }
       cosine[50] { }
       fromAngle 0
       toAngle 720
:INT graphBox[4] { 20 60 200 7 }

:FCT Main
{
    GINIT
    1 GRAPH 0 ALPHA
    DEG
    GCLEAR

/* Generate some data */
    FillArrays

/* Draw the axes, and scale */
    DoAxes

/* Draw the curve */
    0 angle cosine GPLOT

/* Plot the points */
    '+' angle cosine GPLOT

    GETKEY 0 GRAPH 1 ALPHA
}

DoAxes
{
/* This labelling is done while we're
still in pixel coordinates */
    20 4 GMOVE
    "0" GLABEL
    200 4 GMOVE
    "720" GLABEL
    100 4 GMOVE
    "DEGREES" GLABEL

    2 7 GMOVE
    "-1" GLABEL
    2 60 GMOVE
    "+1" GLABEL

/* Make a box for plotting, and change
to user coordinates */
    graphBox GWINDOW
    fromAngle 1.1 toAngle -1.1 GSCALE

/* Draw axes lines */
    0 -1 GMOVE
    0 +1 GDRAW
    0 0 GMOVE
    720 0 GDRAW
}

/* This routine fills 'angle' with
angles, and 'cosine' is filled with
corresponding cosines */
FillArrays
{
    angle SIZE :INT n
    toAngle fromAngle - n / :FLOAT delta
    fromAngle :FLOAT x
    1 :INT i
    n NLOOP
/* Store the angle */
        x angle i PICK =
/* Store the cosine */
        x COS cosine i PICK =
        &x delta + DROP
        &i 1 + DROP
    ENDLOOP
/* Make the arrays look full */
    n angle SETREADY
    n cosine SETREADY
}
```

Figure 23-29. Labelling in graphics mode.

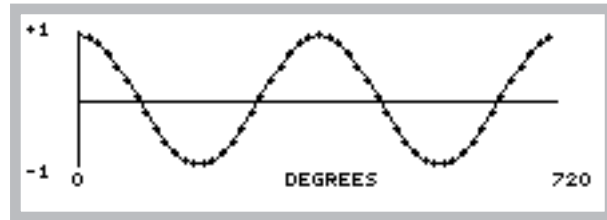The program in Figure 23-29 produces a labelled cosine curve (Figure 23-30).

Figure 23-30. Labelled cosine curve

Drawing Modes

When lines and labels are being drawn in graphics mode, the drawing mode (set by **GMODE**) determines what happens as the new graphics information overwrites old graphics information. For example, where lines intersect, should the pixels be on or off. The possibilities are listed in Table 23-28.

Table 23-28. Graphics Drawing Modes

Mode	Name	Action
0	None	Draw or erase without checking existing graphics state.
1	or	The default after GINIT
2	eor	
3	and	
4	not	Invert, then Mode 0

The effects of the graphics drawing mode is illustrated in Figure 23-31.

```
:INT rect1[] {0 30 40 8}                      :PTR label
    add[] {1 0 1 0}
    rect2[] {10 50 30 10}                      mode1 GMODE
    mode1 0
    mode2 0                                /* Example 1 uses this */
:PTR mode[] {"0 NONE" "1 OR" "2 EOR"          2 rect1 GBOX
             "3 AND" "4 NOT"}            /* Example 2 uses this */
:FCT Main                               /* 1 rect2 GBOX */
{
    1 GRAPH 0 ALPHA                           mode2 GMODE
    GWIDTH 5 / :INT delta
    add delta * DROP                     /* Example 1 uses this */
    GINIT GCLEAR                              1 rect2 GBOX
    1 :INT i                            /* Example 2 uses this */
    5 NLOOP                             /*  2 rect1 GBOX */
        mode i PICK DoBoxes
        &mode2 1 + DROP                       rect2 1 PICK VAL 30 GMOVE
        &i 1 +                                label GLABEL
    ENDLOOP
    0 &mode2 = DROP                           0 GMODE
    GETKEY DROP                               rect2 1 PICK VAL 4 GMOVE
    0 GRAPH 1 ALPHA                           label GLABEL
}                                             rect1 add + DROP
DoBoxes                                       rect2 add + DROP
{                                        }
```


Example 1: The solid rectangles and the lower labels are drawn first, using mode 0. The mode is changed to the indicated values, and the open rectangles and upper labels are drawn

Example 2: The open rectangles and the lower labels are drawn first, using mode 0. The mode is changed to the indicated values, and the solid rectangles and upper labels are drawn.

Figure 23-31. Illustration of drawing modes

Graphics Image Handling

LPL provides some tools for handling graphics images. Images can be copied from the display to a Path with **GIGET**, and restored to the display with **GIPUT**. The utility programs "/Sys/Utility/Graphics Capture" and "/Sys/Utility/Graphics Restore" use these keywords. The programs are discussed on page 21-15.

Graphics images can be moved around on the display by **GSCROLL**. The keyword **GSHIFT** determines the scrolling characteristics. This is the method by which OPEN's New Measurements strip chart mode works.

Serial Communications

LPL provides several tools for dealing with the Comm Port.

Table 23-29. COMM Port Keyword Summary

Keyword	Description
BLKREC	Block receive
BLKSEND	Block send
COMMBREAK	Set Momentary break condition
COMMCONFIG	Configure the comm port: data bits, stop bits, parity, software and hardware handshaking.
COMMLC	Comm line control
COMMLS	Comm line status
COMMPORTCOUNT	Returns number of available comm ports
COMMGETPORT	Return the active comm port
COMMGETTYPE	Return type and capability of the port.
COMMSETPORT	Direct subsequent commands to this port
COMMSETTYPE	Configure a port for USB or Serial
COMMSTATUS	Print the comm configuration to a Path or CHAR array.
COMMUART	Get uart status
FX	Enter file exchange mode
ONCOMM	Incoming comm port interrupt (when a specified character arrives)

RS-232 and USB

LPL 5 introduced new hardware, including a comm port that could be used for either RS-232 or USB, and an additional internal serial port. The active port, and how it is configured, is controlled with the new keywords **COMMGETPORT**, **COMMSETPORT**, **COMMGETTYPE**, **COMMSETTYPE**, and **COMMPORTCOUNT.**

Configuration Example

The utility program "/Sys/Utility/SETCOMM" (described on page 21-17) illustrates the use of **COMMSTATUS** and **COMMCONFIG**. Its listing is shown in Figure 23-32.

```
/*
  Set Comm port params
*/

:INT commRect[] {2 2 38 4}

:FCT
setcomm
{

  0 :CHAR cline[40]

  commRect GETDISP :PTR hold

  commRect 2 1 "Baud Data Stop Parity" WINDOW

  cline COMMSTATUS
  cline STDLINE
  IF
    cline COMMCONFIG

    IF
      1 "Bad Configuration!" MESSBOX
    THEN
  ELSE
    1 "Config unchanged" MESSBOX
  THEN

  hold PUTDISP
}
```

Figure 23-32. Listing of the program SETCOMM.

Analog Measurements

Analog measurements are meaningful only on platforms that have the hardware capability (an A/D convertor) to accomplish this task. However, all platforms support the LPL keywords, and may provide an interface for getting measurements from another device. At any rate, LPL provides a lot of flexibility in its analog measurement capability.

Table 23-30. Analog Measurement Keyword Summary

Keyword	Description
AINUM	Returns the number of analog input channels, ground channels, and maximum number of groups that can be defined.
AIOPEN	Create an internal A/D structure. (obsolete: use **AINEW**)
AICLOSE	Dispose of an A/D structure. (obsolete: use **AIFREE**)
AIGDEF	Define a group.
AICDEF	Define a channel in a group.
AIPREP	Build the action table based on the defined groups and channels.
AISTART	Start or resume the A/D.
AISTOP	Stop the A/D.
AIREADY	Are readings available?
AIFLUSH	Clear waiting readings
AIGET	Get a reading
AITGET	Get a reading with the time stamp
AIFREE	Dispose of the action table.

Groups and Channels

At the heart of LPL's treatment of analog measurements are two concepts: *group* and *channel*.

A <u>channel</u> is a physical measurement channel, and corresponds at some point to where a wire must be connected. Each channel has an address, starting with 0, and the number of channels available is one of the three values obtained via **AINUM**. Thus, if there are 24 channels, they are addressed as 0 thru 23.

A <u>group</u>, on the other hand, has no physical manifestation. A group is simply a collection of channels that are to be measured in the same way. That is, groups define how channels are measured, as all channels in a group are measured the same way, in terms of frequency, number of samples, etc. Groups are identified by number, starting with 0. The number of possible groups (historically 5) is also obtained via **AINUM**.

To illustrate groups and channels, consider the main LI-6400 application, OPEN. The LI-6400 has a four gas analyzer signals (two CO_2 and two H_2O), and about 12 other sensors, such a light sensors, thermistors, etc. We want readings for all of these sensors every, say, 1 second, and we want those readings to be an average over the prior 1 second. But, the gas analyzer signals require higher resolution and move averaging to smooth the noise than the other sensors. We still want 1 second readings, but they should reflect a lot more sub-readings than the other sensors. How do we accomplish this?

We could define two measurement groups: Group 1 could sample at 8 Hz and provides a new reading (the average of those 8 samples) every 1 second. Group 2 could sample at 20 Hz, and provides a new reading every 1 second, but that new reading would be the average over the last 4 seconds, for example, to help smooth the noise. Having defined the groups, we then can identify the channels that are to be included in each group.

Note that in LPL's scheme of things, the same channel can be included in multiple groups. One could, for example, measure a thermocouple in two different ways by including it in two different groups. One group might provide high speed sampling, and the other longer term averaging.

Setting It Up

Analog measurements involve <u>a lot</u> of behind-the-scenes stuff that the operating system takes care of, and you the programmer don't need to worry about (very much). However, there is a right way and a wrong way to get things done, so a few simple rules need to be followed.

The first step in making analog measurements is to tell the operating system to make some work space for itself. This is done with **AINEW**, which returns an address that you must keep track of. The best method is to use a global pointer. It is in this internal structure that the operating system will keep track of our group and channel definitions. Looking ahead, when we are done with a particular set of definitions, we should get rid of them via **AIFREE**.

The next step is to define our groups (with **AIGDEF**) and then channels (with **AICDEF**). Once this is done, we tell the operating system to "compile" our

desires into an "action table", using **AIPREP**.

Now we are ready for measurements. **AISTART** will start our measurements. If for some reason we want to suspend A/D activity, we use **AISTOP**. To resume it again, use **AISTART**.

Easy, right? Well, we've glossed over some important details: Where do the measurements go? How do you get hold of them to display them, or do computations?

Where Do They Go?
When a channel is defined, the following information must be passed to the **AICDEF** function:

• **The address of the internal A/D structure**
This is typically referenced via a pointer from a prior **AINEW**.

• **The signal channel**
Referred to by number (0 to 23).

• **The group to which this channel definition belongs**
Referred to by number (0 to 5). Must have already been defined via **AIGDEF**.

• **The reference (e.g. ground) channel**
Referred to by number (0…7).

• **The range**
This is not currently used, but still must be present. Just say 0.

• **The address of the destination variable**
This must be a FLOAT or a FLOAT array.

Thus, every channel gets linked to a FLOAT variable that you must maintain. (Make sure the destination variable will always exist - using a local variable for this is a bad idea, for example.)

When Do They Get There?
Now that we know where to find our measurements, the question becomes "when do they get there?". You might wonder whether these variables could change suddenly while you're in the middle of using them, which could lead to some surprises. It turns out that the "when" is just as controlled as the "where".

Timing, which is what we're talking about here, is controlled by the groups. When a group is defined (**AIGDEF**), the following information is passed:

- **The address**
  of the internal A/D structure (from a prior **AINEW**).

- **The group being defined**
  Referred to by number (0…4).

- **The time interval.**
  That is, how often (seconds) new readings should be ready for this group.

- **How many sub-samples should be made in this time interval**
  For example, make 10 measurements every 1 second.

- **Averaging information.**
  How many subsamples should be averaged together for the final reading? If you want a 5 second running average and are making 10 readings each second, then use 50 for this.

- **The queue size.**
  How many sets of final readings for this group should the operating system buffer for you. 0 means no buffering.

Let's focus for the moment on the third parameter, the time interval. Each group has some time interval at which new readings will be ready. However, just because readings are available doesn't mean that they automatically go anywhere - they don't. They are buffered until you say you want them, which is done via **AIGET** or **AITGET**.

How do you know readings are ready for you to get them? There are two ways: The function **AIREADY** tells how many readings are available for any group. A more elegant approach, however, rather than sitting around in a tight loop testing **AIREADY**, is to use **ONA2D**. This event command (see **Event Handling** on page 23-33) allows you to specify a function to call whenever readings become available.

An Example

(At last, a complete example!) Figure 23-33 is not particularly useful, beyond illustrating proper technique. The program measures both light sensors on the LI-6400 at 1 second intervals, showing the values on the screen.

```
/*
    Illustrate Analog Measurements
*/

:PTR a2d 0 /* the a/d structrue */


:INT group 0 /* our group # */
    secs 1 /* update time */
    count 10 /* samples / sec */
    avgCnt 10 /* running average */
    qSize 1 /* no buffereing */

/* These channels and grounds are hard-
wired in the LI-6400. */
    qChan 16
    gaspChan 15
    qGnd 3
    gaspGnd 3

/* Our destination variables */
:FLOAT quantum 0
        gasp 0
        dum 0

:FCT main
{
/* Define the A/D stuff */
    Setup IF RETURN THEN

    CLEAR
    "IN-CMBR_mV    OUT_CMBR_mV" PRINT
    1 8 POSXY "Press <esc> to quit" PRINT

    &GetReadings group ONA2D
    &keys ONKBD

    AISTART

    IDLE


    AISTOP
    a2d AIFREE
}
```

```
Setup
{
/* Open a structure.*/
    AINEW IF
        "AIOPEN Failed" PGD 1 RETURN
    THEN
    &a2d =

    qSize avgCnt count secs group a2d
AIGDEF IF
        "AIGDEF Failed" PGD 1 RETURN
    THEN

    1 NLOOP
        &dum 0 0 group 0 a2d AICDEF IF
BREAK THEN
        &dum 0 0 group 1 a2d AICDEF IF
BREAK THEN
        &quantum 0 qGnd group qChan a2d
AICDEF
                IF BREAK THEN
            &gasp 0 gaspGnd group
        gaspChan a2d AICDEF
                    IF BREAK THEN

            a2d AIPREP IF
                "AIPREP Failed" PGD 1
    RETURN
            THEN
            0 RETURN
        ENDLOOP
        "AICDEF Failed" PGD 1 RETURN
    }

GetReadings {
    qSize group AIGET
    UpdateDisplay }

UpdateDisplay {
    1 2 POSXY
    quantum gasp "%7.1f %14.1f" PRINT }

Keys { GETKEY 0x01b == IF HALT THEN }
PGD { PRINT GETKEY DROP }
```

Figure 23-33. Complete A/D example

Zero and Span

Figure 23-33 is concerned with measuring just two channels, but notice that 4 channels are actually defined. In addition to channels 15 and 16, channels 0 and 1 also got defined. What's going on here?

The A/D converter in the LI-6400 needs some reference measurements for accuracy. It must look at a true 0V signal now and again, and also a true 5.0 volt signal. Channel 0 provides the former, and channel 1 provides the latter.

LI-6400 Analog Channels

A standard LI-6400 has 24 analog channels (Table 23-31), and 8 reference channels (Table 23-32). Note that the library file "/Sys/Lib/StdAnalogIn" provides variable names for these channels, which should be used rather than the numbers.

Table 23-31. LI-6400 Analog Input Channels

Channel	Channel Descriptions	Variable Name
0	Zero reference channel	aZeroChan
1	Span reference channel	aSpanChan
2	Battery	aBattChan
3	CO2 Reference	aCO2AChan
4	CO2 Sample	aCO2BChan
5	H2O Reference	aH2OAChan
6	H2O Sample	aH2OBChan
7	IRGA background temp	aTirgaChan
8	CO2 Reference AGC	aAgcCAChan
9	H2O Reference AGC	aAgcHAChan
10	CO2 Sample AGC	aAgcCBChan
11	H2O Sample AGC	aAgcHBChan
12	Cooler block temp	aTblkChan
13	Leaf temp sensor	aTleafChan
14	Flow meter	aFlowChan

Table 23-31. LI-6400 Analog Input Channels

Channel	Channel Descriptions	Variable Name
15	In-chamber PAR sensor.	aParInChan
16	External PAR sensor	aParOutChan
17	Pressure sensor	aPressChan
18	"Lost" spare channel[a]	aLostChan
19	Sample cell air temp	aTairChan
20	Spare channel 1	aUser1Chan
21	Spare channel 2	aUser2Chan
22	Spare channel 3	aUser3Chan
23	Spare channel 4	aUser4Chan

a.This is "lost" because it was not accessible on the 37 pin connector of the first LI-6400s. In LI-6400s having serial numbers 401 and above, it is used for blown fuse detection.

Table 23-32. LI-6400 Reference Channel Descriptions

Channel	Channel Description	Variable Name
0	Reference ground	aRefGnd
1	IRGA ground	aIrgaGnd
2	Flow board ground	aFlowGnd
3	Chamber ground	aChamGnd
4	Pressure ground	aPressGnd
5	Spare ground	aSpareGnd5
6	Spare ground	aSpareGnd6
7	Spare ground	aSpareGnd7

Analog Output Control (D/A)

LPL provides tools for controlling the D/A (digital to analog) converters.

Table 23-33. D/A Control Keywords

Keyword	Description
AONUM	Returns the number of analog output channels
AOMIN	Returns the minimum allowed value of an analog output channel.
AOMAX	Returns the maximum allowed value of an analog output channel.
AORES	Returns the resolution of an analog output channel.
AOSET	Set the value of an analog output channel.
AOVAL	Get the current value of an analog output channel.

Hardware Considerations

The number of digital to analog converters is found from **AONUM**, and is 20 on the LI-6400. They are addressed starting at 0, so number 0 to 19. The range of each, and it's resolution is found using the **AOMIN**, **AOMAX**, and **AORES** functions.

An Example

Figure 23-34 contains a little program that lets you control the light source by pressing the up and down arrow keys.

```
:INT lamp 2 /* d/a for lamp */          {
 delta 200 /* shift amount */            GETKEY :INT k
 uparrow 0x2600                          k escape == IF HALT THEN
 dnarrow 0x2800                          k uparrow == IF
 escape 0x1b                              delta ChangeLamp
                                         THEN
:FCT main                                k dnarrow == IF
{                                         delta CHS ChangeLamp
 1 0x0302 DIOSET /* flow board on */     THEN
 1 0x0005 DIOSET /* lamp on */          }
 &Keys ONKBD
                                        ChangeLamp
 CLEAR "Press up and down arrows\n"     {
PRINT                                     :INT delta
 "(escape to quit)" PRINT
 0 ChangeLamp                            lamp AOVAL delta + lamp AOSET
 IDLE
 OFFKBD                                  1 5 POSXY
}                                        lamp AOVAL "Lamp at %4d mV" PRINT
Keys                                    }
```

Figure 23-34. Program to control the LED source by pressing the up and down arrow keys.

The program turns on the flow control board first, since that must be on to control the lamp. Then it turns on the lamp itself. Then it sets up a keyboard interrupt, so that the function *Keys* is called whenever a key is pressed. *Keys* is concerned with only three keys: **escape**, ↑, and ↓. If ↑or ↓ is pressed, the lamp setting (analog output port number 2) is changed by 200 mV, either up or down. **AOVAL** is used to read the current value, and the change is added to it, and the new value set with **AOSET**.

Digital I/O

Digital I/O is done with the keywords shown in Table 23-34.

Table 23-34. Digital I/O keywords

Keyword	Description
DIONPORTS	Returns the number of DIO ports.
DIONPINS	Returns the number of pins on a particular port.
DIOSTATUS	Get the status and capabilities of a port.
DIODEFPORT	Define a port for input or output.
DIOSETPORT	Set/unset selected pins of a port.
DIOGETPORT	Read selected pins of a port.
DIOSET	Set/unset a pin on a port.
DIOGET	Read a pin on a port.
DIOCOUNTMS	Set sample period for digital counters.
DIODEFCOUNT	Setup a port/pin for use as a counter.
DIOCOUNT	Read and clear a counter.
DIOERR	Returns the number of digital chip "hiccup" recoveries.

Ports and Pins

LPL assumes that digital I/O lines are grouped together into some combination of ports. **DIONPORTS** tells how many ports there are, and **DIONPINS** tells how many pins a particular port has. All pins on a port have the same direction: that is, they are controlled by the LPL program (output), or they are set by some external device (input). Some ports can be programmed to do either input or output, and some are strictly one or the other. The keyword **DIOSTATUS** provides the capabilities and current setting of any port.

A port/pin is designated using a 16 bit integer. The high byte is the port, and the low byte is the pin. Thus, port 2 pin 3 can be most conveniently referenced in hex as 0x0302. The sequence

```
0x0005 1 DIOSET
```

would turn the lamp on, since that is port 0, pin 5 (Table 23-35).

Collections of pins in a port can be dealt with in groups, using **DIOGET-PORT** and **DIOSETPORT**. Thus, to set all pins on port 4 high, you could use the following sequence:

```
0xff 0xff 4 DIOSETPORT
```

The first 0xff specifies the desired pattern of the 8 pins, and the second is a mask pattern.

Table 23-35. LI-6400 digital port and pin assignments

Port	Pins								Status
	7	6	5	4	3	2	1	0	
0	Match valve	Chamber fan on/off	Lamp on/off	TEC on/off	Flow range 2	Flow range 1	RH Control enable	Mixer on/off	Input/<u>Output</u>
1	Pin 22	Pin 4	Log Button	Flow Lo	Flow Hi	Mixer Lo	Mixer Hi	Pump status	<u>Input</u>/Output
2					CO2 Sample	CO2 Ref	H2O Sample	H2O Ref	<u>Input</u>/Output
3					CO2 solenoid	Flow board	IRGA board	Pump	Input/<u>Output</u>
4	Pin 8	Pin 26	Pin 7	Pin 25	Pin 6	Pin 24	Pin 5	Pin 23	Output

Low Speed Counters

Any input pin can be used as a low frequency counter (< 10 Hz). Use **DIODEFCOUNT** to define a port/pin as a counter, and **DIOCOUNT** to sample and clear the counter. The time resolution of these counters is set by **DIOCOUNTMS**. The minimum useful value is 10 (ms); setting it to 0 would effectively stop all counters.

High Speed Counter

For higher speed pulse counting, pin 3 on the 37 pin external connector can be used. It is triggered by the falling edge of a pulse. The keyword **HSCOUNTS** returns the number of pulses detected since the last call.

Digital Errors

The LI-6400 runs a watch dog program that looks for problems in the digital output ports, and restores them to the way they should be if any are found. If this sort of thing happens (usually due to static electricity, for example), a counter is incremented. The number of counts (that is, the number of times this happened) is accessible via the keyword **DIOERR**, which returns the count, and resets the counter.

XML Support

Added in Version 6.0.

Table 23-36. LPL Keywords XML support.

Keyword	Description
XMLOAD	Define an XML structure
XMLOADF	Define an XML structure with input from a file
XMEXEC	Execute a structure
XMEXECF	Execute with source from a file
XMCLEAR	Clear a structure
XMCAPTURE	Set captured values to current values
XMSETCVAL	Directly set captured values
XMCHANGED	Have any nodes changed
XMEDITABLE	Is a node editable?
XMSETVAL	Set a node's value
XMSETATTR	Set the attribute of a node
XMSETSTATE	Set the state of a node(s)
XMVIEW	View a structure
XMINSERT	Insert a node
XMREMOVE	Remove a node(s)
XMREVERT	Set current values from captured values
XMFIND	Find a node

Table 23-36. LPL Keywords XML support.

Keyword	Description
XMGETVAL	Get a nodes value
XMNODES	Count subnodes

Registered Variable Support

.

Table 23-37. Registered variable support.

Keyword	Description
RTMNEW	Create a variable list.
RMTADD	Add a variable to the list
RMTSIGNAL	Signal the remote device
RMTCLEAR	Clear the list
RMTFREE	Dispose of the list

Curve Fitting Support

.

Table 23-38. Curve fitting support.

Keyword	Description
CFNEW	Create a new curve fit structure
CFDEF	Redefine the curve
CFFIT	Fit the curve to data
CFVAL	return f(x)
CFINV	return f^{-1}(y)
CFCOEFFS	Get one or all the coefficients
CFSTATS	Get statistics of the fit
CFVALID	Is the last fit valid?
CFERR	Error from last fit
CFFREE	Dispose of the structure

Application Tools

LPL provides tools for running LPL programs from other LPL programs.

Table 23-39. LPL Keywords for running applications.

Keyword	Description
CALL	Execute a function whose address is on the stack
COMPILE	Compile a function, return compiled code's address
DEBUG	Access the debug menu
FINDADDR	Given an object's name, return it's address
GETMODNUM	Returns an applications module pointer number
ISLINK	Is a module linked?
KBDEXEC	Enters LPL's keyboard command line mode.
LINK	Load and compile source code, add to an application
LPL	Return the version number of the operating system
MEM	Find the amount of memory, and largest contiguous piece.
MEMMAP	Outputs the heap map.
MODSIZE	List compiler size directives for one or more modules
PTRTOLL	Converts an LPL address to address and type
RUN	Launch a program.
SETID	Change the owner of an allocated item
SETMODNUM	Set the value of an application's module pointer
STKSHARE	Control stack sharing
STRIP	Strips token information from an application.
SYSID	Get the ID of the current application
UNLINK	Remove one or more modules from an application
USES	List cross reference information for an object
XREF	Outputs a cross reference for an application.

Running Applications

ID numbers

When an application is launched (with the **RUN** keyword), it is given an ID number. An application can find out it's ID number by the keyword **SYSID**. The first application has an ID of 1. If it launches an application, that one will be 2, etc. The ID number gets used in a number of settings, including

- **Garbage Collection**
Once an application terminates, the LPL garbage collector deallocates any memory that was marked as belonging to that application, that has not already been deallocated.

- **Dynamic Allocation**
MAKE4 allocates arrays for whatever application is specified. This lets a child application create an array for the parent to use, for example. **SETID** allows the ownership of a dynamically allocated structure to be changed.

- **Compiling Functions**
The **COMPILE** keyword lets a child application compile a function that will below to any parent (or itself).

- **Compiling and Linking Structures**
The **LINK** keyword compiles and links any LPL structure (functions, PTR arrays, etc.) to any existing application.

Modules

Applications contain one or mode modules. A module is a source file, and an application collects source files together with the compiler directive **:IN-CLUDE**. As each module is added to the application, it is given a *module number*. One of the attributes of an application is a module counter. This is the number that the next linked module will receive. It starts at 0, and increments by 1. The counter can be examined and modified by **GETMODNUM** and **SETMODNUM**. Why? Keep reading.

An application can be modified after being launched, as well. **UNLINK** can strip one or more modules from an application (based on module number), and **LINK** adds modules. In OPEN 3.0 and above, user defined equations are appended using this method. When changing ComputeLists, the old user defined module(s) are unlinked, and the new ones linked. Prior to OPEN 3.0, it was all done with **COMPILE**.

Debugger

The **DEBUG** keyword brings up LPL's debugger.

The target application name

Shows a list of all active applications. Use this to select the target application for the debugger.

```
========= /sys/lib/stdshell =========
A - Pick App
C - Chars
D - Doubles and floaters
F - Functions
L - Longs and ints
M - Miscellaneous
P - Pointers
U - User private symbols
Y - all public sYmbols
Z - LPL Keywords
```

Show all variables and their values for this type

Show all private and public object names.

Show list of all LPL keywords.

Figure 23-35. DEBUG's main menu.

■ **Debugging tour**

1 Launch the debugger from OPEN's main screen

From OPEN's main screen, one can access the debugger by pressing **K**, then typing **DEBUG** at the ok prompt, and pressing **enter**.

2 Pick an application

At DEBUG's main screen, press **A** to pick an application. A menu will appear (Figure 23-36) of all the applications that are currently active. This represents the "trail" of applications that typically occurs from power on to when open runs. The active one, /sys/lib/stdshell, is what runs when you press K in OPEN's main screen.

```
======= <esc> quits =======
4 - /sys/lib/stdshell
3 - /sys/open/open
2 - /sys/open/start
1 - /sys/autost
```
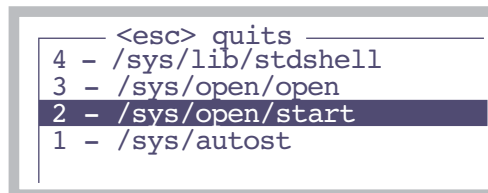
Figure 23-36. Picking the target application.

Select /sys/open/start.

3   **View the functions**
Press **F** to view the function list. Press **escape** when done.

Address   Size   Number of local variables

```
FUNCTIONS
...PRIVATE...
051A1C08 Keys (114 bytes) (1 locals)
051A1BFA main (106 bytes) (0 locals)
051A1C16 ScanDirs (60 bytes) (1 locals)
(None defined) ...PUBLIC...
```

Figure 23-37. The function list of /sys/open/start.

4   **Press M to view the miscellaneous items**
The list of things in this menu is as follows:

Table 23-40. What is displayed in the miscellaneous section of the debugger.

	ITEM	Description	Compiler Directive
	Local Stack		**:LOCAL**
For each module	**Name**	The module name	
	Static Symbols	The symbol table addresses of the root and each block for this module. (Typically, there are no static symbols, because they are stripped after compilation.) The size and the number of used entries in each block is shown.	**:STATCOUNT**
	Private Symbols		**:PRIVCOUNT**
	Public Symbols		**:PUBCOUNT**
	Id Space	Space for names of objects	**:NAMES**
	Data Segment	Data segment information	**:DATA**
	Code Segment	Code segment information	**:CODE**
	Total Wasted.	Wasted space total.	

5   **View the list of LPL keywords**
Press **Z** for the entire list.

Excel Tools

LPL 6 added support for building Excel files.

Table 23-41. Excel file building commands

Keyword	Description
XLOPEN	Open an new .xls file.
XLADDITEMS	Add items to the .xls file
XLADDCOL	Define columns (items in LogList)
XLADDCONTROL	Define header items
XLPREP	Prepare for logging
XLADDOBS	Add an observation
XLCLOSE	Close the .xls file.
XLREOPEN	Reopen the .xls file.

The file "/Sys/Open/Open/log" uses these commands for the Excel log file option in OPEN.

Miscellaneous Tools

NameList

LPL 5 adds a structure that is a list of variables, including names, id numbers, and addresses. It is called a NameList, and several other structures use it, in-

cluding **StatTracking** on page 23-91, and **Real Time Graphics** on page 23-59.

Table 23-42.

Keyword	Description
NLNEW	Create a new namelist
NLFREE	Free a name list
NLADD	Add an item
NLFIND	Find an item
NLGET	Get the ith item id and pointer
NLGETSTR	Get the ith item label
NLCOUNT	How many items?
NLCLEAR	Clear the list

Below is how OPEN builds its Name list of system and user variables. It is created and destroyed elsewhere.

```
PUB BuildVarList
{
    OpenVarNameList NLCLEAR

    /* User */
    userIndexList READY :INT n
    1 :INT i
    n NLOOP
        userIndexList i PICK VAL :INT id

        id FmtGetVarLabel
        id FmtGetLogLab
        id FmtGetVarAddr
        id OpenVarNameList NLADD
        &i 1 + DROP
    ENDLOOP

    /* sys */
    1 &i =
    systemVariables READY NLOOP
        i _FmtIndexToSysRef &id =

        id FmtGetVarLabel
        id FmtGetLogLab
        id FmtGetVarAddr
        id OpenVarNameList NLADD
        &i 1 + DROP
    ENDLOOP
```

```
    UpdateRTGSources
}
```

StatTracking

LPL 5 introduced a useful tool for keeping running statistics and checking for
stability. It is the StatTracker.

Table 23-43.

Keyword	Description
STNEW	Create a new StatTracker.
STADD	Add a variable to the list
STSIZE	How many variables are in the list
STUPDATE	Add a new reading for each variable
STNUMSTABLE	How many variables meet stability criteria
STRESET	Flush buffers, check addresses, set frequency
STNEW	User editing
STFLUSH	Flush the buffers
STREAD	Read a configuration (Deprecated in 6.1)
STWRITE	Write a configuration (Deprecated in 6.1)
STGET	Retrieve a value
STSET	Set a value
STFREE	Destroy a StatTracker

To illustrate its use, we present some functions used for the IRGA zeroing
routine.

```
MakeZeroStatTrackers
{
    OpenVarNameList 0 STNEW &ZTracker =

    tPeriod -1 ZTracker STADD
    tPeriod -2 ZTracker STADD
    tPeriod -4 ZTracker STADD
    tPeriod -5 ZTracker STADD
    hiresupdatetime ztracker STRESET
}
```

```
FlushZeroTracker
{
    ZTracker STFREE
}
Update
{
    A2dReadHiRes
    A2dReadLowRes
    ComputeAllSensors

    a2dTime 60.0 / ZTracker STUPDATE

    ZeroRTGAction

    GETALPHA IF
    ShowDataPtr
        GetStatus

        StatusCO2 1 <> StatusH2O 1 <> OR IF
            10 warningLine POSXY
            "IRGA(s) NOT READY" "\x83%s\x80" PRINT
        ELSE
            1 warningLine POSXY CLREOL
        THEN
    THEN
}
```

Battery and Power

Finally, some miscellaneous tools.

Table 23-44. LPL keywords involving battery and power.

Keyword	Description
LOWWARN	Enable/Disable low battery warning
GETBATT	Returns battery state
POWEROFF	Power off
RESTART	Restarts processor. SImulates power off, then on

LOWWARN enables or disables the behavior described in **Low Battery Warning** on page 5-18: a beep every 2 seconds when battery voltage drops below 11V, and a 60 second countdown to power off when the voltage drops below 10.5V. When the low warning is disabled, the instrument will run until the battery voltage reaches about 7V. **GETBATT** returns the battery state (0, 1, or 2 for ok, low, and critical). **POWEROFF** does what you'd think it does, and **RESTART** is the equivalent of power off then back on.

Reference Voltages

The factory-measured reference voltages are stored in the file "/dev/.vcal". These are accessible to LPL either through this file, or else by two keywords.

Table 23-45.

Keyword	Description
VREFGET	Retrieve the reference voltages
VREFSET	Set the reference voltages

See also Table 24-43 on page 94.

```
<vcal>
    <v5
        ref="5.0016"/>
    <v12bit
        max="4.9967"
        min="-5.0013"/>
    <v8bit1
        max="4.9824"
        min="0.0060"/>
    <v8bit2
        max="4.9636"
        min="-4.9868"/>
    <date>
        2002-03-06 14:47:51
    </date>
</vcal>
```

Figure 23-38. Listing of "/dev/.vcal"

Calling the OS

SYSTEM provides a method of making a direct call to the operating command shell, if there is one. (LPL 5.0 and above).

LPL Reference

24

Keyword Summary

24 *LPL Reference*

This chapter contains a reference for LPL keywords and compiler directories.

Syntax Summaries

LPL Type Declarations

:CHAR

Character declaration
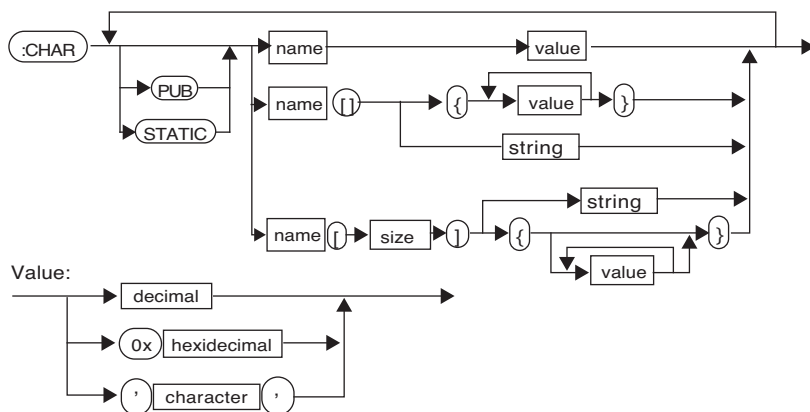
Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
decimal	Any combination of the chars 0…9	-
hexidecimal	Any combination of 0…9 plus A…F	-
character	Any character	-
string	Delimiter char is first encountered	

:DOUBLE   **High precision floating point declaration**
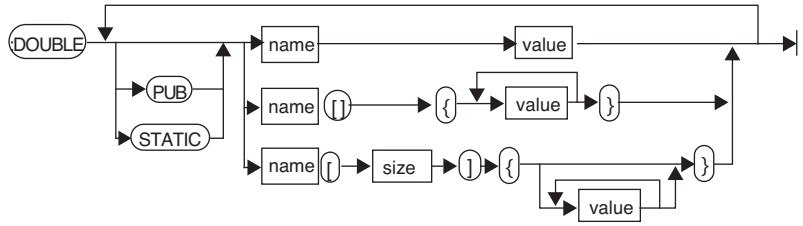
Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
value	integer or floating point	-

:FCT **Function declaration**
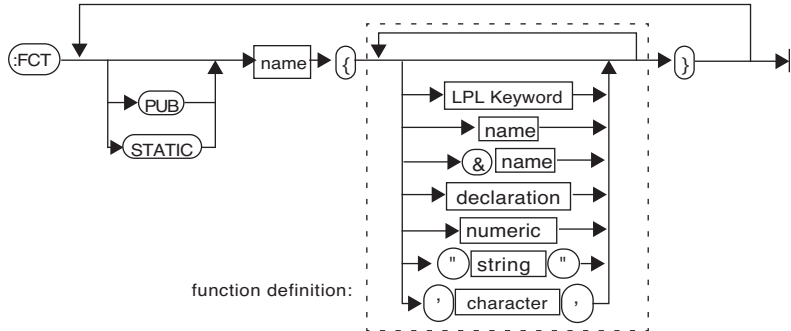
function definition:

declaration:
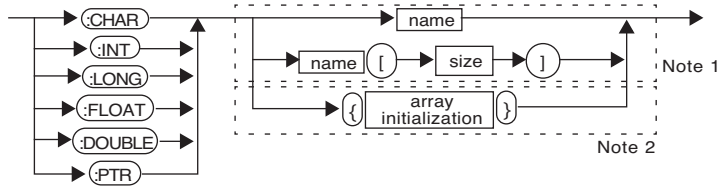
Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
array initialization	specify each array element in the un-named array	.
Note 1	This operation consumes an item from the stack.	
Note 2	This operation consumes nothing from the stack, and leaves the address (of the unnamed array on the stack.	

:FLOAT **Declares a low precision floating point**
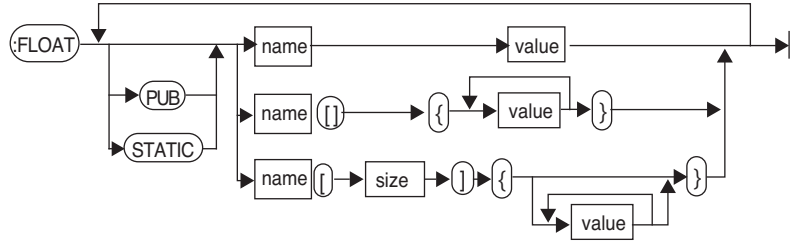
Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
value	integer or floating point	-

:INT **Short integer declaration**

Value:
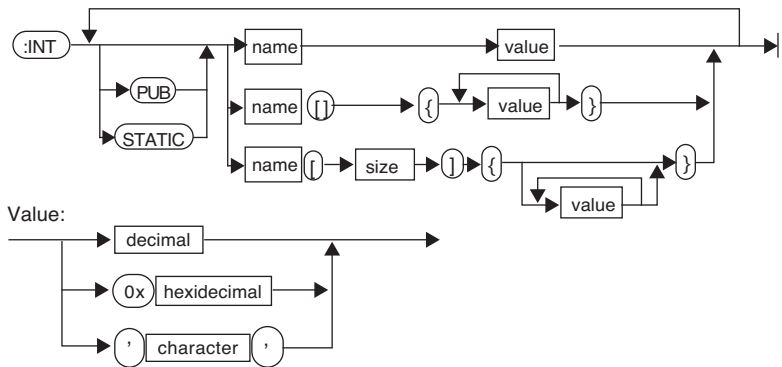
Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
decimal	-	-
hexidecimal	-	-

Item	Description	Range
character	-	-

:LONG **Long integer declaration**
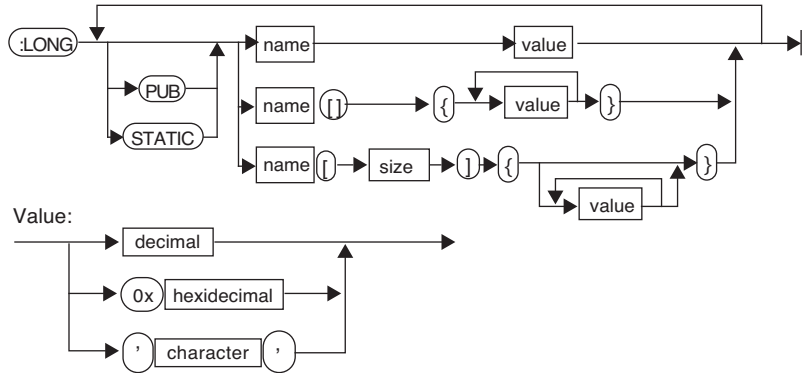
Value:

Item	Description	Range
name	valid LPL name	-
size	integer	1…32767
decimal	-	-
hexidecimal	-	-
character	-	-

:PTR　　　　　　　　　**Pointer declaration**
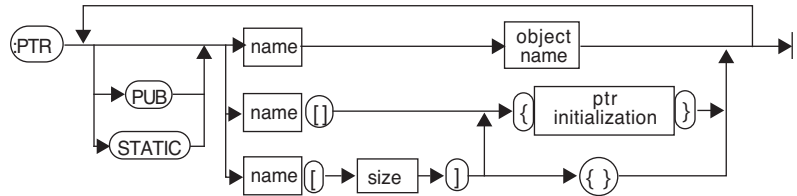
ptr initialization:
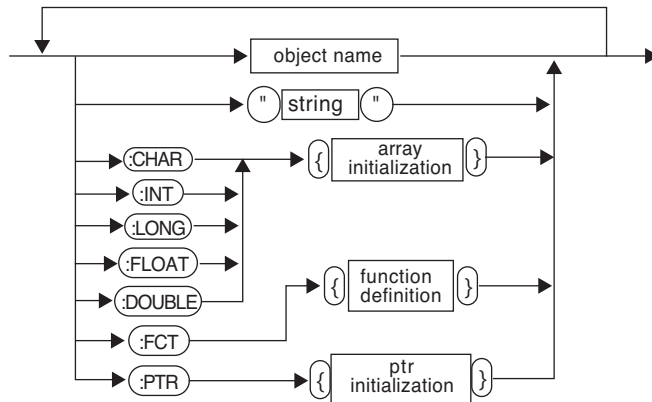
Item	Description	Range
name, object name	valid LPL name	-
size	integer	1…32767
array initialization	specify each array element	
function definition		

Compiler Declarations

:CODE   **Declares the code space for this application**

Item	Description	Range
Bytes	Creates a code segment.	1 to 64000

:DATA   **Declares data space for this application**

Item	Description	Range
Bytes	Creates a data segment.	1 to 64000

:IFDEF
:IFNDEF   **Compiler logic**
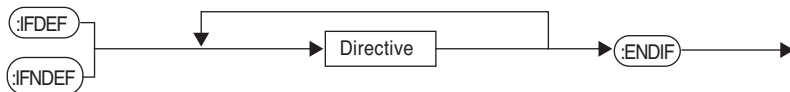
Item	Description	Range
Directive	LPL compiler directive or declaration	-

:INCLUDE   **Include a file**

:INCLUDE ────────► filename ────────►|

Item	Description	Range
filename	The name of the file to be included in this application.	If the name contains one or more spaces, it should be quoted.

:LOCAL   **Create space for local objects**

:LOCAL ────────► Bytes ────────►|

Item	Description	Range
Bytes	The number of bytes to use for local objects. Default = 1000	1 to 64000

:NAMES   **Make space for variable names**

:NAMES ────────► Bytes ────────►|

Item	Description	Range
Bytes	The number of bytes to make a name segment. Multiple segments are allowed.	1 to 64000

:PRINT   **Print a message to the display**

:PRINT ────────► String ────────►|

Item	Description	Range
String	The first non-whitespace character following the :PRINT is used as the string delimiter.	-

:PRIVCOUNT **Declare number of private objects**

Item	Description	Range
count	The number of private objects that this module defines.	1 to 4096

:PUBCOUNT   **Allocate space for N public items**

:PUBCOUNT ——→ count ——→|

Item	Description	Range
count	The number of public objects that this module defines.	1 to 4096

:RTGMGR   **Declare a RealTime Graphics manager**

:RTGMGR ——→ name ——→
{ RTGMGR initialization }
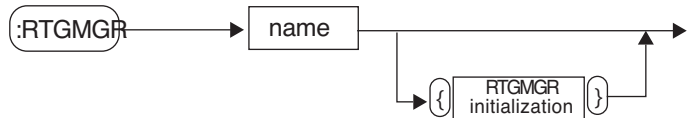
:SEARCH   **Add a directory to the list to be searched**

:SEARCH ——→ directory ——→|

Item	Description	Range
directory	The name of the directory to be added to the list of directories to be searched when finding a file to be included.	If the name contains one or more spaces, it should be quoted.

:STATCOUNT   **Declare number of private objects**

:STATCOUNT ——→ count ——→|

Item	Description	Range
count	The number of static objects that this module defines.	1 to 4096

:STATIC   **Enable / Disable making all subsequent declarations to be static.**

Item	Description	Range
logic	A number. If non-zero, then it's considered true. If zero, then it's considered false.	integer

:XLB   **An XLBuilder object.**

:XML   **An XML Structure**
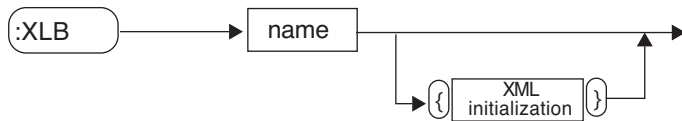
Definitions

It is critical to know what each keyword expects to find (if anything) on the stack, and what information each keyword might leave (if anything) on the stack. Therefore, the following format is used in describing the LPL keywords:
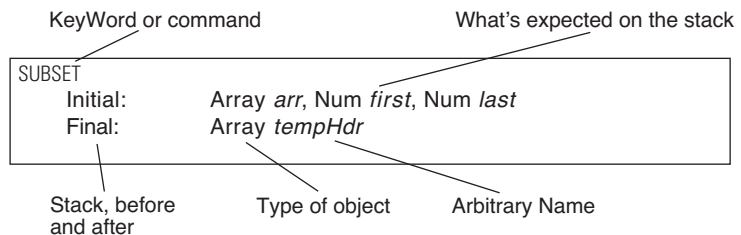
When writing a program with the SUBSET keyword, the post fix order of things matches the order given in the keyword reference. Thus, using the same variable names, we would write

```
arr first last SUBSET    /* Make the subset */
:PTR tempHdr             /* Make local PTR for the result */
```

To do this same function using in-fix notation, the order of the arguments stays the same:

```
...
0 :PTR tempHdr                           /* Create the ptr */
$ tempHdr = SUBSET(array, first, last)    /* Do SUBSET */
...
```

To describe *what* each stack item is, we use the conventions and symbols shown in Table 24-1.

Table 24-1. Symbol Key

Symbol	Meaning	Symbol	Meaning
-	Nothing expected (initial) or nothing returned (final)	Array *name*	NArray, CArray, or PArray
...	Other items	NArray *name*	Address of a numeric array (type NAddr).
[ ]	Optional items are in square brackets	CArray **name**	Address of a Char array
< a | b | c >	Choose <u>one</u> of a, b, or c.	PArray *name*	Address of a Ptr array
Obj *name*[a]	Any object: Addr or Num. (Any stack items is either an address (Addr) or else a numeric value (Long or Double)).	Rect *name*	NArray *window*[b] -or- Num *left*, Num *top*, Num *right*, Num *bottom*
Num *name*	A Long or Double numeric value.	Text *name*	Path or CArray
Addr *name*	Address of any LPL object.	NObj *name*	Long, Double, or NAddr.
Logic *name*	Num that is evaluated to be 1 or 0	Fct *name*	Address of an LPL function
Path *name*	Address of a path	NAddr *name*	Address of numeric object: Char, Int, Long, Double, or Float
STTR *name*	Address of a StatTracker	NameList *name*	Address of NameList
RTG *name*	Address of a RealTimeGraphics	ListBox *name*	Address of a ListBox
XML *name*	An XML structure	XL *name*	Address of an XLBuilder

a. 'name' is arbitrary, and represents how the parameter is used by the keyword.

b.Note that *window* must have a size of at least 4, since elements 1 through 4 are taken to be left, top, right, and bottom respectively.

Single Value Transforms

A number of keywords fall into the class of Single Value Transforms. These have initial and final stack requirements as shown in Figure 24-1.

Initial:	Num *a*
Final:	Double or Long *b*
-or-	
Initial:	NObj *c*
Final:	NObj *c*
-or-	
Initial:	Array *b*
Final:	Array *b*

Figure 24-1. Single value transform stack requirements

Single value transforms can operate on a numeric value, or the address of a value (a variable), or on an array of values. Table 24-2 lists some examples.

Table 24-2. Examples using ABS, a single value transform.

LPL Code	Result
-12 ABS	Puts 12 (LONG) on the stack
48.4 ABS	Puts 48.4 (DOUBLE) on the stack.
arr1 ABS	Sets every element of arr1 to its absolute value.

Two Value Transforms

Some LPL keywords are Two Value Transforms, and their stack require-
ments are shown in Figure 24-2

Initial:	Num *targetNum*, <Num *b* I NObj *c* I Array *d*>
Final:	Double or Long *result*
-or-	
Initial:	NObj *targetVal*,<Num *b* I NObj *c* I Array *d*>
Final:	NObj *targetVal*
-or-	
Initial:	Array *targetArray*, <Num *b* I NObj *c* I Array *d*>
Final:	Array *targetArray*

Figure 24-2. Two value transform stack requirements

Table 24-3 lists some examples using the operator +

Table 24-3. Examples using +, a two value transform.

LPL Code	Result
23 45.6 +	Puts 58.6 (DOUBLE)
100 200 +	Puts 300 (LONG)
&x 1 +	Adds 1 to the value of x, and leaves the address of x on the stack.
1 &x +	If x=3, leaves 4 on the stack
0 xArray +	If xArray=(1 2 3 4), leaves sum (10) on the stack.
xArray yArray +	Adds corresponding elements to xArray. If xArray = (1 2 3 4), and yArray = (10 20 30), then xArray = (11 22 33 4). If yArray = (10 20 30 40 50), then xArray = (11 22 33 44). Address of xArray always left on stack.

Two Value Logical Transforms

These operators are very much like Two Value Transforms, except the resulting values are always 0 or 1.

```
Initial:        Num targetNum, <Num b | NObj c | Array d>
Final:          Long result (0=false, 1=true)
        -or-
Initial:        NObj targetVal,<Num b | NObj c | Array d>
Final:          NObj targetVal (object value will be 1 or 0)
        -or-
Initial:        Array targetArray, <Num b | NObj c | Array d>
Final:          Array targetArray (array elements will be 1 or 0)
```

Figure 24-3. Two value logical operators

Table 24-5 lists some examples.

Table 24-4. Examples using the logical operator ==

LPL Code	Result
12 11 ==	Leaves 0 on the stack
arr1 arr1 ==	Sets every element of arr1 to 1.
arr1 5.5 ==	Sets every element of arr1 to 1 if it was equal to 5.5, otherwise it sets it to 0.
0 arr1 ==	Puts 0 on stack if no element of arr1 is 0, otherwise 1.
1 arr1 ==	Puts 1 on stack if every element of arr1 is 1, otherwise 0.

The Reference

+   **Add values, arrays, or combinations of the two**
Initial / Final: (see **Two Value Transforms** on page 24-15)

-   **Subtract values, arrays, or combinations of the two**
Initial / Final: (see **Two Value Transforms** on page 24-15)

*   **Multiply values, arrays, or combinations of the two**
Initial / Final: (see **Two Value Transforms** on page 24-15)

/   **Divide values, arrays, or combinations**
Initial / Final: (see **Two Value Transforms** on page 24-15)

^   **Raise to a power**
Initial / Final: (see **Two Value Transforms** on page 24-15)

=   **The assignment operator**
Initial: Num *x*, NAddr *target*
Final: -
        -or-
Initial: Addr *x*, Addr *target*
Final: -

The assignment operator is used for numerics, strings, and pointers.

Table 24-5. Examples using the assignment operator =

LPL Code	Description
`5.3 &x =`	Sets variable x to 5.3
`arrX arrY =`	Sets array Y to be equal to array X: The READY value for Y is set to that for X, and elements 1…READY of Y are set to those for X. RESIZED of needed.
`100 arrY =`	Sets elements 1…READY of array Y to 100.
`&x &yPtr =`	(yPtr is a PTR). yPtr now points at object x.

==   **Logical compare**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)
Do not use this operator to compare arrays (use COMPARE for that).

<>   **Logical negative compare**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

>   **Logical greater than**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

>=   **Logical greater than or equals**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

<   **Logical less than**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

<=   **Logical less than or equals**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

ABS   **Absolute value of a value or array**
Initial / Final: (see **Single Value Transforms** on page 24-14)

ACOS   **Inverse cosine**
Initial / Final: (see **Single Value Transforms** on page 24-14)

ADR?   **Is the object on the top of the stack an address?**
Initial: Obj *a*
Final: Obj *a*, LONG *returnVal* (1 = yes, 0 = no)
                            Related Keywords: TYPE

For a discussion of the AI_ keywords, see **Analog Measurements** on page 23-71.

AICDEF   **Define an analog input channel**
Initial: FAddr *dest*, Num *range*, Num *ground*, Num *group*, Num *chan*, A2dPtr *atd*
Final: Long *code* (0=ok, non-zero=failed)
                            See **Analog Measurements** on page 23-71.

Table 24-6. Parameters for AICDEF

dest	The address of the FLOAT variable or array that will hold the new measurements. The actual transfer happens with AIGET or AITGET.
range	The range (unused presently)
ground	The ground channel (0…Ng)
group	The group to which this measurement belongs. This group must have previously been defined with AIGDEF.
chan	The signal channel (0…Ns)
atd	The A/D structure returned by a prior call to AIOPEN.

AICLOSE   **Close an A/D structure (Obsolete. Use** AIFREE**)**
Initial: A2dPtr *atd*
Final: -

See **Analog Measurements** on page 23-71.

AIGDEF   **Define a group for analog measurements**
Initial: Num *qSize*, Num *avgCnt*, Num *count*, Num *secs*, Num *grp*, A2dPtr *atd*
Final: Long *code* (0=ok, non-zero=failed)

See **Analog Measurements** on page 23-71.

Table 24-7. Parameters for AIGDEF

qSize	The number of final readings that the operating system should buffer. This value must be between 1 and N, where (c*4+14)*N < 64000 and c is the number of channels in the group.
avgCnt	The number of raw readings that should be averaged together for each final reading.
count	The number of raw readings that should be taken during the time interval *secs*.
secs	New readings are to be made available at this time interval.
grp	The group number being defined (0…Ng)
atd	The A/D structure returned by a prior call to AIOPEN.

AIGET   **Get a set of A/D readings for a group**
Initial: Num *numWanted*, Num *group*
Final: Long *numRead*

This causes the measured values to be loaded into the destination variables specified in the AICDEF setups. See **Analog Measurements** on page 23-71.

Table 24-8. Parameters for AIGET

numWanted	The number of sets of readings to transfer. This should be less than or equal to the group's qSize, set via AIGDEF. The number of sets that are actually available can be obtained from AIREADY
group	The A/D structure returned by a prior call to AIOPEN.
numRead	The actual number sets of readings read.

AIFLUSH   **Flush all readings for a group**
Initial: Num *group*
Final: -

After a flush, the group will report 0 readings available.

AIFREE   **Free A/D space**
Initial: [A2dPtr a]
Final: -

> For historic compatibility, the A2dPtr is optional. Deallocates internal memory structures used for A/D measurements. See **Analog Measurements** on page 23-71.
> Related Keywords: AINEW

AINEW   **Create a new A/D structure, returns pointer**
Initial: Num sysID
Final: A2dPtr a, Long 0
   -or-
Final: Long 1

> Same as AIOPEN, but returns a pointer. sysID is id of the owning application. Use 0 for current app.
> Related Keywords: AIFREE

AINUM   **Returns status of platforms A/D hardware**
Initial: -
Final: Long *nGroundsAvail*, Long *nChannelsAvail*, Long *nGroupsAvail*

> Provides the number of ground channels, measurement channels, and groups that are available. See **Analog Measurements** on page 23-71.

AIOPEN   **Open an A/D structure (Obsolete. Use** AINEW**)**
Initial: [A2dPtr *model*]
Final: -

> *model*, if specified, should be an existing A/D that will be used as a model for making this new one. See **Analog Measurements** on page 23-71.
> Related Keywords: AICLOSE

AIPREP   **Prepare the A/D for operation**
Initial: A2dPtr *atd*
Final: Long *error* (0=ok, non-zero=failed)

> AIPREP must occur after groups and channels are defined (AIOPEN, AIGDEF, AICDEF), and before the A/D is started (AISTART).

AIREADY   **Group status: readings lost, readings available**
Initial: Num *group*
Final: Long *numLost*, Long *numReady*

> When an A/D is running, this function will return information about readings available, or lost, for a particular group. To capture readings, use AIGET or AITGET.

AISTART   **Start or restart the A/D running.**
Initial: -
Final: Long *error* (0=ok, 1=failed)

> This must be preceded by AIPREP, and all the things that lead up to that.

AISTOP **Pause the A/D operation**
Initial: -
Final: -

> This is a good idea during times that the user will obviously not be interested in getting new numbers, as it frees up the processor for more important things. To restart, use AISTART.

AITGET **Get readings, and the times associated with them**
Initial: FAddr *timeDest*, Num *readings*, Num *group*
Final: Long *actual*

> Get a groups readings along with the time associated with each reading.

Table 24-9. Parameters for AITGET

timeDest	The address of the FLOAT variable or array that will contain the time information. If *readings* is greater than 1, then *timeDest* should be a FLOAT array, of size at least as big as *readings*.
readings	The number of sets of readings to transfer. This should be less than or equal to the group's qSize, set via AIGDEF. The number of sets that are actually available can be obtained from AIREADY
group	The A/D structure returned by a prior call to AIOPEN.
actual	The actual number sets of readings read.

ALPHA **Turns on/off the alpha (text) display.**
Initial: Logic *onOff* (0=off, 1=on)
Final: -

> Related Keywords: GRAPH, GETALPHA, GETGRAPH

AMOVE **Copy elements within an array**
Initial: Num *to*, Num *from*, Num *nElements*, Array *arr*
Final: -

> Copies *nElements*, starting at location *from*, to location *to*, in array *arr*.

AND **Logical AND (true if both non-zero)**
Initial / Final: (see **Two Value Logical Transforms** on page 24-16)

> Related Keywords: NOT, OR

For a discussion of the AO_ keywords, see **Analog Output Control (D/A)** on page 23-78.

AOBATCH   **Set multiple D/As as simultaneously as possible**
Initial: NArray *values*, NArray *addresses*
Final: final
> Does the equivalent of AOSET for each item in the two arrays, getting the address information from the first array, and the value information from the second.
> Related Keywords: AOSET

AOMAX   **Returns max signal that can be put on a D/A channel**
Initial: Num *chanNum* (0..AONUM-1)
Final: Double *mV*
> Related Keywords: AOSET, AORES, AOMIN.

AOMIN   **Returns the min signal that can be put on a D/A channel**
Initial: Num *chanNum* (0..AONUM-1)
Final: Double *mV*
> Related Keywords: AOSET, AORES, AOMAX

AONUM   **Returns number of D/A channels available**
Initial: -
Final: Long *numDacs*

AORES   **Returns resolution of a D/A channel**
Initial: Num *chanNum* (0..AONUM-1)
Final: Double *mV*

AOSET   **Set a D/A channel to a value**
Initial: Num *mV*, Num *chanNum*
Final: -
> Related Keywords: AOVAL

AOVAL   **Returns the current value of a D/A channel**
Initial: Num *chanNum*
Final: Double *mV*
> Related Keywords: AOSET

APP   **Append item or array onto an array**

Initial: Array *toAdd*, Array *dest*

        -or-

Initial: NObj *toAdd*, NArray *dest*

        -or-

Initial: Addr *toAdd*, PArray *dest*

Final: -

                Related Keywords: READY, GETREADY, SIZE. Example: Figure 23-3 on page 23-9

ARGS   **Provides a path to arguments (if any) for that application.**

Initial: -

Final: Addr *path*

                This path is used by the Filer for it's launched applications, such as file copy, file move, etc.

                Related Keywords:LCD, COMM, KBD

ASIN   **Inverse sine**

Initial / Final: (see **Single Value Transforms** on page 24-14)

                Related Keywords: DEG, RAD

ATAN   **Inverse tangent**

Initial / Final: (see **Single Value Transforms** on page 24-14)

                Related Keywords: DEG, RAD

ATAN2   **Inverse tangent given two sides of a right triangle**

Initial / Final: (See **Two Value Transforms** on page 24-15)

                In post-fix, the order is X then Y. Examples: 1.0 1.5 ATAN2 yields 56.3 degrees, whereas -1.0 1.5 ATAN2 yields 123.7, and 1.0 -1.5 ATAN2 yields -56.3.

BACKLIGHT   **Turns on/off the backlight (if installed)**

Initial: Num *n*

Final: -

                $n = 0$: Backlight off, otherwise backlight on.

                Related Keywords:ISBACKLIGHT

BEEP   **Turn on beeper for a specified time**

Initial: Num *milliSecs*

Final: -

                The beep is a background process. It does not halt program execution. If KBDCLICK is on, pressing a key will terminate the beep.

BENTER   **Binary enter**

Initial: Obj *dest*, Path *source*

Final: Long *byteCount* (-1 if error)

                No filtering (**Path Filters** on page 23-43) is done on binary transfers.

                Related Keywords: BPRINT, ENTER

BINAND   **Binary AND of two integers**
Initial / Final: (See **Two Value Logical Transforms** on page 24-16.)

Table 24-10. BINAND Truth Table

	0	**1**
0	0	0
1	0	1

Related Keywords: BINCMP, BINEOR, BINIOR, BSHIFT

BINCMP   **Binary complement**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
Computes the binary complement of a LONG.

Table 24-11. BINCMP Effect

Before	**After**
0	1
1	0

Related Keywords: BINAND, BINEOR, BINIOR, BSHIFT

BINEOR   **Binary exclusive or**
Initial / Final: (See **Two Value Logical Transforms** on page 24-16.)
Computes the exclusive or of two LONGS.

Table 24-12. BINEOR Truth Table

	0	**1**
0	0	1
1	1	0

Related Keywords: BINAND, BINCMP, BINIOR, BSHIFT

BINIOR   **Binary inclusive or**
Computes an inclusive or for two LONGs. If a value is not LONG, an internal con-

version is done first.

Table 24-13. BINIOR Truth Table

	0	**1**
0	0	1
1	1	1

Related Keywords: BINAND, BINCMP, BINEOR, BSHIFT

BLKATTR   **Sets attribute for a rectangle of text on the display.**
Initial: Rect *area*, Num *newAttr*
Final: -
             Related Keywords: PUTTEXT, SETATTR

BLKREC   **Received error-checked data from the Comm port.**
Initial: Path *dest*
Final: Long *errorCode* (0=ok, -1=end of file or time out, -2=user aborted)
             The sending device must be doing the equivalent of LPL's BLKSEND for this to work.

BLKSEND   **Send error-checked data out the Comm port**
Initial: Path *source*
Final: Long *errorCode* (0=ok, -1=end of file or timeout, -2=user aborted)
             The receiving device must be doing the equivalent of BLKREC for this to work.

BPRINT   **Binary print**
Initial: Obj *source*, Path *dest*
Final: LONG *byteCount* (-1 = error)
             If source is a PTR array, then multiple objects can be sent. No filtering is done on binary transfers.
             Related Keywords:BENTER, PRINT

BREAK   **Exit a LOOP…ENDLOOP**
Initial: -
Final: -

BREAKIF   **Conditional exit from a LOOP…ENDLOOP**
Initial: Num *a*
Final: -

             BREAKIF is functionally identical to the sequence IF BREAK THEN

BSHIFT   **Binary shift**
             See **Two Value Logical Transforms** on page 24-16. Shift bits in a LONG left or right.

Related Keywords: BINCMP, BINAND, BINEOR, BINIOR

CALL   **Call the function whose address is on the stack**
  Initial: Fct *code*
  Final: -

> The address can get there via COMPILE, or the & operator, or PICKing from a PTR array.

> For a discussion of the CF_ keywords, see **Curve Fitting Support** on page 23-84.

CFCOEFFS   **Get one or all coefficients**
  Initial: Num *n*, CF *cf*
  Final: Double a_n
      -or-
  Initial: NArray *coeffs*, CF *cf*
  Final: -

> For $n = 0$,...power, returns the nth coefficient.

CFDEF   **(re-)Define a curve**
  Initial: Num *force*, Num *power,* CF *cf*
  Final: -

CFERR   **Last error**
  Initial: Text *dest*, CF *cf*
  Final:

> Writes last error to *dest*.

CFFIT   **Fit a curve**
  Initial: NArray *xData*, NArray *yData*, CF *cf*
      -or-
  Initial: NArray *xData*, NArray *yData*, Num *power*, CF *cf*
      -or-
  Initial: NArray *xData*, NArray *yData*, Num *force*, Num *power*, CF *cf*
  Final: 1 (ok) or 0 (fail)

CFFREE   **Dispose of a CF structure**
  Initial: CF *cf*
  Final:

CFINV   **Invert the function**
    Initial: Num *yVal*, CF *cf*
    Final: Double *xVal*
                     Returns value of *xVal* such that f(*xVal*) = *yVal*.

CFNEW   **Make a new curve fit (CF) structure**
    Initial: Num *sysID*, Num *force*, Num *power*
    Final: CF *cf*
                     Polynomial of degree *power*. if *force* is non-zero, the offset term will be 0.

CFSTATE   **Gets state of compact flash device**
    Initial: -
    Final: Long 1 (there) or <0 (error code)

CFSTATS   **Get statistics**
    Initial: Num *code*, CF *cf*
    Final: Double *val*
                     Related Keywords:Code defined in Table 24-14.

Table 24-14. Options for FILE_OPEN

code	Statistic
1	R^2
2	Mean square error
3	Standard error of the slope
4	Standard error of the intercept
5	Sum of the squares of the residuals

CFUNMOUNT   **Unmounts the compact flash device**
    Initial: -
    Final: Long 0 (ok) or (<0 error code)

CFVAL   **Value of the function**
    Initial: Num *xVal*, CF *cf*
    Final: Double *yVal*
                     Returns value of the function at *xVal*.

CFVALID　　　　　　　　**Is it valid?**
　Initial:　　　　CF *cf*
　Final:　　　　1 (ok) or 0 (not ok)

CHAR　　　　　　　　　**Returns the code value for type CHAR.**
　Initial:　　　　-
　Final:　　　　LONG *typeVal*
　　　　　　　　　　　Related Keywords: INT, LONG, FLOAT, DOUBLE, PTR, MAKE, MAKE4

CHS　　　　　　　　　**Change sign**
　Initial / Final: (See **Single Value Transforms** on page 24-14.)

CLEAR　　　　　　　　**Clear the current text window**
　Initial:　　　　-
　Final:　　　　-

　　　　　　　　　Clears text, and sets the attribute for the current text window to the SETATTR value.
　　　　　　　　　Related Keywords: CLREOL, SETATTR, WINDOW

CLOSE　　　　　　　　**Close a path**
　Initial:　　　　Addr *path*
　Final:　　　　-

　　　　　　　　　Closes any path opened by an OPEN_ keyword (see **Paths** on page 23-42)

CLREOL　　　　　　　**Clear from the cursor location to the right edge of the text window.**
　Initial:　　　　-
　Final:　　　　-

　　　　　　　　　The display attribute for that portion of the line is set to the SETATTR value.
　　　　　　　　　Related Keywords: CLEAR, SETATTR, WINDOW

COMM　　　　　　　　**Provides the standard path to the comm port**
　Initial:　　　　-
　Final:　　　　Addr *path*
　　　　　　　　　This path is common to all applications, and cannot be closed by any of them.
　　　　　　　　　Related Keywords:LCD, ARGS, KBD

COMMBREAK　　　　　**Set a break condition on the Comm port transmit line.**
　Initial:　　　　-
　Final:　　　　-

COMMCONFIG **Configure the Comm port**

Initial: Text *config*

Final: Long *errCode* (0=ok, 1=failed)

The string config should contain

baud dataBits stopBits parity [*handShake*]

Table 24-15. Comm port configuration parameters

baud	Generally 115200, 57600, 38400, 28800, 19200, 9600, 4800, 3600, 2400, 1200, 300, but may be platform specific.
dataBits	7 or 8
stopBits	1 or 2
parity	n (none), e (even), o (odd), 1 (1's) or 0 (0's).
handShake	(Optional) H (hardware handshake), X (xon/xoff), HX or XH (both).

When hardware handshaking, the LPL device will not transmit unless it sees DSR and CTS high. When xon/xoff handshaking, the receipt of an xoff character (0x13) will halt data transmission, and the receipt of an xon (0x11) character will resume it.
Related Keywords: COMMSTATUS

COMMGETPORT **Determine the active comm port**

Initial: -

Final: Long *port*

Related Keywords: COMMSETPORT

COMMGETTYPE **Determines the status and capability of a comm port**

Initial: Num *port*

Final: Long *info*

info bit:

0 (1): Is able to do RS-232

1 (2): Is presently configured for RS-232

2 (4): Is able to do USB

3 (8): Is presently configured for USB

Examples: 3 = RS-232 only

7 = RS-232, but dual capability

13 = USB, but dual capability

Related Keywords: COMMSETTYPE

COMMLC   **Comm port modem line control**
Initial: Num *command* (bit: 1 = RTS; 2 = DTR)
Final: -

COMMLS   **Comm port modem line status**
Initial: -
Final: Long *statusByte* (bit:0:CTS, 1:RTS, 2:DTR, 3:DSR)

```
       LI-6400                            COMPUTER

     1: CD    ◄──────────────        1: CD
     2: RD    ◄──────────►           2: RD
     3: TD    ───────────            3: TD
     4: DTR   ───────────            4: DTR
     5: GND   ───────────            5: GND
     6: DSR   ◄──────────►           6: DSR
     7: RTS   ───────────            7: RTS
     8: CTS   ◄──────────►           8: CTS
     9:                              9:
```
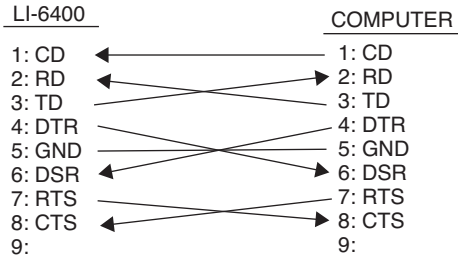
Figure 24-4. Wiring schematic for the 9975-016 cable.

COMMPORTCOUNT   **Returns the number of Comm ports available**
Initial: -
Final: Long *n*

COMMSETPORT   **Activates a Comm port**
Initial: Num *port*
Final: -

                  Once activate, the port receives all subsequent COMM... commands
                  Related Keywords:COMMGETPORT

COMMSETTYPE   **Set the type of a comm port**
Initial: Num *port*, NUM (0=RS-232, 1=USB)
Final: -

                  Related Keywords:COMMGETTYPE

COMMSTATUS   **Returns the Comm status configuration string**
Initial: [Text *dest* (default is LCD)]
Final: -

                  Related Keywords:COMMCONFIG

COMMUART   **Returns the UART status for the Comm port**
Initial: -
Final: Long *statusByte* (bit: 0=break received, 1=framing error, 2=parity error)
                  Example: 6 means framing error and parity error.

COMPARE   **Compare two arrays**

Initial: Array *arr1*, Array *arr2*

Final: LONG *result (1=same, 0=different)*

> Arrays are the same if 1) they are the same type, 2) they have the same ready value, 3) their contents (elements 1…READY) are the same.

COMPILE   **Compile a function**

Initial: Text *source*, [Num *sysID*, Path *errors*, Num 1]

Initial: Text *source*, [Num *sysID*, Path *uses*, Path *errors*, 2]

Final: Fct *code*, Long 0

    -or-

Final: Long 1

> Turn the text of a function definition into a function.

Table 24-16. Compile parameters

errors	If no error path is specified, no error messages are written.
uses	The names of all referenced private and public symbols are recorded in this path.
sysID	If 0, the current application is considered the parent. The parent application determines a) which symbol table is used, and b) who the allocated code space belongs to.
source	The text to be compiled. If a path, must be path to file or buffer.
code	To execute this function, use CALL.

> Related Keywords: KBDEXEC, CALL

CONTRAST   **Set the display contrast**

Initial: Num n

Final: -

> n = 0 to 100.
> Related Keywords: GETCONTRAST

CONVERTIN   **Set the filter for data entering a path**

Initial: Text *filter*, Addr *Path*

Final: -

> String *filter* is made of codes from Table 23-20 on page 23-44.
> Related Keywords: GETCONVERTIN, CONVERTOUT, SETDFCIN, PRINT, ENTER, PUTCH, XFER

CONVERTOUT   **Specify filter(s) for data coming out of a path**

Initial: Text *filter*, Addr ***Path***

Final: -

> String *filter* is made of codes from Table 23-20 on page 23-44.

Related Keywords: GETCONVERTIN, CONVERTIN, SETDFCIN, PRINT, ENTER, PUTCH, XFER

COS   **Cosine**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
Units depend on RAD and DEG.

CTIME   **Converts seconds since base time to a time and date string.**
Initial: [Text *dest* (default is LCD)], Num *tdSecs*
Final: -

The format used is Fri Apr 7 1995 11:22:33. See **Real Time** on page 23-31.
Related Keywords: GETTDS, DATE, TIME

DATE   **Convert seconds to year, month, and day.**
Initial: Num *tdSecs* (secs since base time)
Final: Long *day* (1..31), Long *month* (1..12), Long *year* (e.g.1995)
Related Keywords: TIME, GETTDS, CTIME, SECS2TD, TD2SECS.

DAYOFWK   **Find the day of the week for a certain date**
Initial: Num *day*(1..31), Num *month*(1..12), Num *year* (e.g.1995)
Final: Long *dayNum* (1=Sunday, 7=Saturday)
Related Keywords: DATE, SECS2TD, DAYOFYR

DAYOFYR   **Determines the day of the year**
Initial: Num *day*(1..31), Num *month*(1..12), Num *year* (e.g.1995)
Final: LONG *dayNum* (1=1 Jan, 365 (or 366) = 31 Dec)
Related Keywords: DATE, SECS2TD, DAYOFWK

DEBUG   **Access the debug utility**
Initial: -
Final: -

DEBUGV   **Show the error dialog box**
Initial: Path *errors*
Final: -

errors can contain anything, but normally it would be the message path used for COM-
PILE or LINK.

DEG   **Enable "degrees mode" for trig functions.**
Initial: -
Final: -

The scope of DEG is the application. The default mode is RAD.
Related Keywords: RAD

DEVNAME   **Returns the device name for a disk**
Initial: [Text *dest*], Text *name*
Final: Long *error* (0=ok, non-zero = can't find)

> The device name is the internal file system name, and will be something like /dev/flashdisk1. This name is needed when reading or writing disk images.

For a discussion of the DIO_ keywords, see **Digital I/O** on page 23-80.

DIOCOUNT   **Counts since the last inquiry**
Initial: Num *portPin* (e.g. 0x0301)
Final: Long *counts*

> Use DIODEFCOUNT to define a counter.

DIOCOUNTMS   **Define counting resolution for all counters**
Initial: Num *timeMs*
Final: -

> How often should the operating system check the digital inputs for changes? Minimum useful value is 10.

DIODEFCOUNT   **Define or un-define a portPin as a counter.**
Initial: Num *code* (1=define, 0=undefine), Num *portPin* (e.g. 0x0302)
Final: -

> Use DIOCOUNT to read the counter.

DIODEFPORT   **Set direction of a bidirectional digital port**
Initial: Num *dir* (1=in, 0=out), Num *port* (e.g. 2)
Final: -

> Only works on bidirectional ports. Determine status and ability with DIOSTATUS.

DIOERR   **Number of digital port resets have occurred**
Initial: -
Final: Long *count*

> See **Digital Errors** on page 23-82.

DIOGET   **Returns status of a portPin.**
Initial: Num *portPin* (e.g. 0x0302)
Final: Long *highLow* (1=high or 0=low)

DIOGETPORT   **Get state of multiple pins in a port**
Initial: Num *maskWord*, Num *port*
Final: Long *stateWord*

> Set bits in *mask* determine what pins' status values are returned.
> Related Keywords: DIOSETPORT

DIONPINS   **How many pins in a port**
Initial: Num *port*
Final: Long *pins*

DIONPORTS   **How many ports are available**
Initial: -
Final: Long *numPorts*

DIOSET   **Set a portPin high or low**
Initial: Num *state* (1=high, 0=low), Num *portPin* (e.g. 0x0302)
Final: -

       Related Keywords: DIOGET

DIOSETPORT   **Set multiple pins in a port**
Initial: Num *stateWord*, Num *maskWord*, Num *port*
Final: -

       Related Keywords: DIOGETPORT

DIOSTATUS   **Return directional capability and status of a port**
Initial: Num *port*
Final: Long *status* (bits: 0=can do input, 1=can do output, 2=is set for input).

For a discussion of the DIR_ keywords, see **File System** on page 23-47.

DIRALL   **Get list of all directories in the file system**
Initial: [Path *dest*]
Final: -

       If *dest* is not present, default destination is LCD. Returns the latest list, but will scan if necessary.
       Related Keywords: FLIST, FILER

DIRCURR   **Obtain the current default directory.**
Initial: [Text *dest*]
Final: -

       The default *dest* is the standard path LCD.
       Related Keywords: DIRSET

DIRERASE   **Remove a directory from the file system**
Initial: Text *filename*, [Num *all* (non-zero = remove dir even if not empty)]
Final: Long *error* (0=ok, 1=dir not found, 2=not empty, -1=failed)
       Related Keywords: FERASE

DIRMAKE **Create a new directory**
Initial: Text *newName*, [Num *code*]
Final: Long *error* (0=ok, -1=failed)
If code is present and non-zero, any missing parent directories will also be created.

DIRSAVE **Force an update of the flash file system**
Initial: -
Final: -

This function is deprecated.

DIRSET **Sets the current default directory**
Initial: Text *filename*
Final: Long *error* (0=ok, non-zero=failed)
Related Keywords: DIRCURR

DISPHEIGHT **Returns height of the text display**
Initial: -
Final: LONG *rowsHigh* (e.g. 8)
Related Keywords: DISPWIDTH

DISPWIDTH **Returns width of the text display**
Initial: -
Final: LONG *colsWide* (e.g. 40)
Related Keywords: DISPHEIGHT

DISPUPDATE **Suspends/resumes display (text and graphics) updates**
Initial: NUM *n*
Final: -

n = 0: Suspend updates (nested).
n = 1: Resume updates (nested).
n = 2: Force updates to resume.

DOUBLE **Returns code value for type DOUBLE**
Initial: -
Final: LONG *typeVal*
Related Keywords: INT, LONG, FLOAT, CHAR, PTR, MAKE, MAKE4.

DROP **Drop top item from stack**
Initial: ..., Obj *b*, Obj *a*
Final: ..., Obj *b*
Related Keywords: SWAP, ROT, DUP

For a discussion of the DSK_ keywords, see **File System** on page 23-47.

DSKFORMAT   **Format (completely erase) a disk (LPL 4 and below)**
Initial: Text *fileName*
Final: Long *error* (0=ok, non-zero=failed)
          The disk stated or implied by *fileName* is the one formatted.

DSKISSWAP   **Is there swap space available? (LPL 4 and below)**
Initial: -
Final: Long *code (0=yes, 1=no)*
          Related Keywords: DSKPACK, DSKONLINE, DSKOFFLINE.

DSKOFFLINE   **Take a disk off-line, make available for swap space (LPL 4 and below)**
Initial: Text *filename*
Final: Long *error* (0=ok, 1=failed)
          Note: All files and directories on the disk taken off-line are erased!
          Related Keywords: DSKONLINE, DSKPACK

DSKONLINE   **Bring the off-line disk on-line, and name it (LPL 4 and below)**
Initial: Text *name*
Final: Long *error (*0=ok, 1=failed)
          Related Keywords: DSKOFFLINE, DSKISSWAP.

DSKPACK   **Defragment a disk (LPL 4 and below)**
Initial: [Path *statusDest*], Text *filename*
Final: Long *error* (0=ok, 1=failed)
          Default *statusDest* is LCD. Copied files' names are listed to this path. The disk named or implied in *fileName* is the one defragmented. Swap space must be available.
          Related Keywords: DSKISSWAP, DSKOFFLINE.

DSKSPACE   **Get disk space information**
Initial: Text *fileName*
Final: Long *bytesAvail*, Long *bytesUsed*
          Related Keywords: DSKPACK

DUP   **Duplicate the top item on the stack**
Initial: ..., Obj *a*
Final: ..., Obj *a*, Obj *a*
          Related Keywords: SWAP, DROP, DUP

ECHO   **Enables/Disables LTerm**
Initial: NUM *onoff*
Final: -
          If *onoff* is non-zero, LTerm is enable. If *onoff* is zero, LTerm is disabled.
          Related Keywords:ISECHO

For a discussion of the ED_ keywords, see **Customized Editors and Menus** on page 23-56.

EDCLOSE   **Dispose of an edit structure**

Initial: EditInfoPtr *edInfo*
Final: -

Do not do any further processing once *edInfo* has been disposed.
Related Keywords: EDOPEN.

EDCTL   **Perform an editor function, or series of functions**

Initial: Num *code*, EditInfoPtr *edInfo*
     -or-
Initial: Narray *codes*, EditInfoPtr *edInfo*
Final: -

Perform an editor function or series of functions. *code* values are given in Table 24-17
Related Keywords: EDOPEN.
Related Keywords:

Table 24-17. EDCTL Codes

Hex	Description	Hex	Description
0x01	Refresh the display	0x02	Toggle a menu bar on the cursor line
0x10	Insert a newline at the cursor location	0x13	Delete the character at the cursor
0x11	Split line insert	0x14	Delete the current line
0x12	Delete the char behind the cursor	0x15	Clear to the end of the line
0x20	Insert mode on	0x24	Caps lock off
0x21	Insert mode off	0x25	Caps lock toggle
0x22	Insert mode toggle	0x26	Access the AnyChar routine.
0x23	Caps lock on		
0x30	Move cursor right	0x35	Move to start of previous word
0x31	Move cursor left	0x36	Page left
0x32	Move cursor up	0x37	Page right
0x33	Move cursor down	0x38	Page up
0x34	Move to start of next word	0x39	Page down
0x40	Jump to the beginning	0x45	Jump to bottom of window
0x41	Jump to the start of the last line	0x46	Jump to byte location

Table 24-17. (Continued)EDCTL Codes

Hex	Description	Hex	Description
0x42	Home on current line	0x47	Jump ask
0x43	Jump to end of current line	0x48	Sort ask
0x44	Jump to top of window		
0x50	Set top of block	0x54	Delete block
0x51	Set bottom of block	0x55	Print block to comm port
0x52	Copy block to cursor position	0x56	Store block to file
0x53	Move block to cursor position	0x57	Insert (file) block at cursor position
0x59	Sort block		
0x60	Define/Search for target	0x62	Print all
0x61	Find next target		
0xa0	Node toggle	0xa4	Node toggle 1
0xa1	Node enter	0xa5	Node revert
0xa2	Node exit	0xa6	Node revert all
0xa3	Node edit	0xa7	Node edit or toggle
0xFF00	Cancel and leave	0xFF01	Accept and leave (OK)

EDKEY   **Simulates typing. Handles ascii and cursor control codes.**
Initial: Num *keyCode*, EditInfoPtr *edInfo*
    -or-
Initial: Narray *keyCodes*, EditInfoPtr *edInfo*
    -or-
Initial: Path *text*, EditInfoPtr *edInfo*
Final: -
                  EDKEY interprets nonascii key codes; EDWRITE doesn't.

EDOPEN   **Create a custom edit structure.**
Initial: Text *objectText*
Final: [EditInfo *edInfo*], Long 0
    -or-
Final: Long 1
                  Related Keywords: EDCLOSE

EDSTAT   **Return custom edit status info**

Initial: Num *code*, EditInfoPtr *edInfo*
Final: LONG *result*

          -or-

Initial: Num 4, EditInfoPtr *edInfo*
Final: Path *result*

                    codes and results are given in Table 24-18.

Table 24-18. EDSTAT codes and Results

Code	Result	Code	Result
0	Error code for previous operation	5	The length of the current line of text.
1	Cursor position (offset from start)	6	The offset from the beginning of the cursor.
2	Insert mode status (1 is on, 0 is off)	7	The line number of the cursor line.
3	Character at present cursor location (0 if not on text)	8	Cursor position from left margin.
4	The PATH associated with the edit info structure.	9	The offset to the start of the current line

EDWRITE   **Simulates typing in a custom editor.**

Initial: Num *keyCode*, EditInfoPtr *edInfo*

          -or-

Initial: Narray *keyCodes*, EditInfoPtr *edInfo*

          -or-

Initial: Path *text*, EditInfoPtr *edInfo*
Final: -

                    Related Keywords: EDKEY

ELSE   **Used between IF and THEN**

Initial: -
Final: -

                    See **Conditionals and Loops** on page 23-4.

ENDLOOP   **Terminates LOOP or NLOOP structures**

Initial: -
Final: -

                    See **Conditionals and Loops** on page 23-4.

ENTER   **Read values of variables from a source**

Initial: ..., Addr *var1*, CArray *format*, [PATH *source*]
Final: LONG *numRead*

                    Default *source* is KBD. *format* is explained below. The number of addresses on the
                    stack should correspond to elements in the format string. *numRead* is the count of ob-

jects assigned values. Arrays count as 1 object.

Fill and Empty Registers

ENTER starts reading at the empty register location of *source*, and updates the empty register as characters are consumed. ENTER terminates when the path empties (empty register reaches the fill register value) or when all the directives within the format string are met.

Using Arrays

ENTER *appends* data to arrays. If you wish to overwrite data in an array, use SETFILL prior to ENTER.

The Format String

The format string controls how the source path is read and interpreted. A format string can contain multiple format directives, which take the form

```
% [*] [Reps] [Width] Type
```

Table 24-19. Format string scan format directives

%	Always marks the start of a format directive.
*	Optional. If present, no assignments are made (there doesn't need to be a corresponding address on the stack to accept any values).
Reps	Optional. Used when the target address is an array. If present, it is one of (n), (.), or (*). See Table 24-20
Width	Optional. Maximum number of many characters to consume while satisfying the format directive, not counting leading whitespace.
Type	See Table 24-21

Table 24-20. Repeat Format Codes

Rep Code	Append items until...
(n)	...n are appended
(.)	...until a newline ('\n') character is reached.
(*)	...until the array is full.

Table 24-21. Format Type Codes for ENTER

Type	Description
d or D	Integer

Table 24-21. (Continued)Format Type Codes for ENTER

Type	Description
s	String
c	Character
f or F e or E g or G	Floating point
r<c>	User specified radix. See below.

Skipping and Searching

Any characters that lie outside of a format directive are used as "read through" characters. More sophisticated searching can be done using the %x (or %X) directive. When a %x is encountered in the format string, text is popped from the stack, and the source is searched for a matching string. If one is found, the empty pointer is moved past it. Thus, the LPL code

```
&y "END=" &x "%d%x%f" file ENTER
```

will do the following steps: 1) read an integer, assign it to x. 2) consume characters until the string 'END=' is consumed. 3) read a floating point value, assign it to y.

User Defined Radix

If the source contains numeric values that are not decimal, they can be read and converted using the user defined radix type code 'r'. The character immediately following the 'r' determines the radix, or base, to be used. This character should be the highest value character in the base. (binary = '1', decimal = '9', hex = 'f', base 36 = 'z', etc.) For example, if a file contains the following line

```
ff ef 8e 01101101
```

and we wished to interpret these as 3 hex and 1 binary value, and store them as 4 integer variables (a, b, c, and d), we would write

```
&d &c &b &a "%rf %rf %rf %r1" file ENTER
```

Table 24-22 contains numerous examples.

Table 24-22. Format string examples.

Format String	What is Done
`"%f"`	Read a floating point value.
`"%5c"`	Read the next 5 characters, append to an array.
`"%*5c"`	Read the next 5 characters, make no assignments.
`"%(.)d"`	Read integers to the end of the line, appending to the target array.
`"%(*)f"`	Read as many floating point values as it takes to fill the target array.
`".%d %d"`	Scan the source until a '.' is found, skip over it, read an integer, skip whitespace, read another integer.
`"%f,%f,%f"`	Enter three floating point values that are delimited by commas

EXP **Computes the exponential of e**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
Inverse of LOG.

FCOPY **Copy a file**
Initial: Text *source*, Text *dest*, [Num *overwriteFlag* (0=don't, non-zero OK)]
Final: Long *error* (0=ok, 1=source not found, 2=dest exists or illegal, -1=failed)
Related Keywords: FMOVE

FEDIT **Open a file with the system editor**
Initial: Text *filename*
Final: -
File sizes must be less than 64000 bytes.
Related Keywords: FVIEW

FERASE **Erase a file**
Initial: Text *filename*
Final: Long *error* (0=ok)
Related Keywords: DIRERASE.

FGETTDS **Return the last-modified date for a file or directory**
Initial: Text *filename*
Final: Long *secsSinceBaseTime*
To interpret *secsSinceBaseTime*, see **Real Time** on page 23-31.
Related Keywords: FSETTDS, FTYPE, TIME, DATE, CTIME, SECS2TD

FGETWP   **Get the write protect status of a file or directory**
    Initial: Text *specifier*
    Final: Long *result* (0=ok, 1 = write protected)
                    Related Keywords: FSETWP

FILER   **Access the filer.**
    Initial: [Text *directory* (default is current working directory)]
    Final: -

FIND   **Find an element or array in an array**
    Initial: NArray *arr*, NObj *target*
        -or-
    Initial: PArray *arr*, [Num *s1*, Num *s2*, …,] Addr *target*
    Final: Long *subscript* (or 0 if no match)
                    Related Keywords: PFIND See **Searching and Comparing Arrays** on page 23-10

FINDADDR   **Given a variable name, return it's address**
    Initial: Text *name* [, Num *sysID*]
    Final: Long 1 (not found)
        -or-
    Final: Addr *address*, Long 0
                    The default *sysID* is the current application. FINDTOKEN does the opposite.

FINDTOKEN   **Returns an object's name, given it's address**
    Initial: Text *tokenDest*, Num 1, <Addr *obj* or Long *address*>, Num *sysID*
        -or-
    Initial: Text *ownerDest*, Num 2, <Addr *obj* or Long *address*>, Num *sysID*
        -or-
    Initial: Text *ownerDest*, Text *tokenDest*, Num 3, <Addr *obj* or Long *address*>, Num *sysID*
    Final: -
                    If *sysID* is 0, it specifies the current application. The tokens address can be an LPL
                    pointer, or else a LONG. *ownerDest* will contain the name of the module in which the
                    token is defined.

FLIST **Get a list of files and/or directories**
Initial: Num *code*, Text *pattern*, Text *fields*, [Path *dest* (default = LCD)]
Final: Long *itemsFound*
Use DIRALL to get all directories on the file system.

Table 24-23. Parameters for FLIST

code	Selects files and/or directories: 1 - files only 2 - directories only 3 - files and directories
pattern	Selects the directory to be searched, and the pattern to search for. The characters * and ? are wild cards. For all files in a directory, use "" or "*".
fields	Determines what information to retrieve for each item found: N or n- name S or s- size (bytes) D or d- date last changed T or t- time last changed R or r- version number

FLOAT **Returns code value for type FLOAT**
Initial: -
Final: LONG *typeVal*
Related Keywords: INT, LONG, CHAR, DOUBLE, PTR, MAKE, MAKE4.

FLUSH **Clears the stack**
Initial: …
Final: <empty stack>

FMERGE **Combine path and file name into a file specifier**
Initial: Text *fullSpecifier*, Text *pathName*, Text *fileName*
Final: -
If *pathName* doesn't start with a / or \, the current working directory will be included in *fullSpecifier*. *pathName* can end with a / or \ or neither; it doesn't matter.
Related Keywords:FPARSE

FMOVE **Move a file to another directory.**
Initial: Text *source*, Text *dest*, [Num *overwriteFlag* (0=don't, non-zero=ok)]
Final: Long *error* (0=ok, 1=not found, 2=dest illegal or exists, -1=failed)
The source file is removed.
Related Keywords: FCOPY

FPARSE **Break a file specifier into path and file name**
Initial: Text *pathDest*, Text *nameDest*, Text *fullSpecifier*
Final: -
If *fullSpecifier* doesn't start with a / or \, the current working directory is taken into

account, and will be reflected in the *pathDest* name.
Related Keywords:FMERGE

FREE   **Dispose of a dynamically allocated array or function**
 Initial: Addr *ad*
 Final: -

     Related Keywords: MAKE, MAKE4, COMPILE

FREESOFT   **Discard the current function key structure**
 Initial: [Softkeys *s*]
 Final: -

     If s is present, it frees it. If it is not present, then the currently active softkeys are freed.
     Related Keywords: MAKESOFT, NEWSOFT

FRENAME   **Rename a file or directory**
 Initial: Text *old*, Text *new*
 Final: Long *error* (0=ok, 1=not found, 2=illegal or exists, -1=failed)

FSETTDS   **Sets the last-modified date of a file or directory**
 Initial: Num *secs*, Text *filename*
 Final: Long *error* (0=ok, non-zero=failed)
     To convert *secs* to real time, see **Real Time** on page 23-31.
     Related Keywords: FGETTDS

FSETWP   **Set the write protect status of a file or directory**
 Initial: INT *onOff*, Text *specifier*
 Final: Long *error* (0=ok, non-zero = failed)
     Related Keywords: FGETWP

FSIZE   **Get the size of a file**
 Initial: Text *filename*
 Final: Long *bytes* (-1 if doesn't exist)
     Related Keywords: FTYPE

FTYPE   **Returns the type of a file**
 Initial: Text *filename*
 Final: Long *returnCode* (0=doesn't exists, 1=file, 2=directory)

FTRASH   **Returns the disk's trash directory specifier**
 Initial: [Text *dest*], Text *specifier*
 Final: -
     If *dest* is not specified, LCD is assumed.

FUPDATE   **Update a text file**
Initial: Path *contents*, Text *filename*
Final: Long *error* (1=OK, -1=failed)

LI-6400 Note: If *contents* matches the actual current contents of the file, nothing is done. If the contents don't match, only the changed portion (section of *contents* between first and last difference) is written to disk, thereby potentially saving disk space.

FVIEW   **View a file with the standard menu**
Initial: Text *filename*
Final: -

Any file can be viewed regardless of its size.
Related Keywords: FEDIT

FWPICK   **Pick a file from a menu**
Initial: Text *filename*
Final: Long *code* (0=user aborted, 1=file has no wildcards, or a file was selected)

If *filename* contains a wild card character, a menu pops up of all the files that match the specifier, and the user selects the file he wants. This menu allows navigation to all parent and child directories, but the files shown in each directory are only those that match the pattern. This function is the one that is called when the user enters a wild card character in a filename in the main LPL OS menu (such as when prompted for a file to run or edit). The use of the Standard File Dialog usually renders this function unnecessary.
Related Keywords: OPEN_FILE_ASK

FX   **Enter file exchange mode**
Initial: -
Final: -

GBOX   **Erase, frame, or fill a rectangle**
Initial: Num *code* (0=erase, 1=frame, 2=fill), Rect *box* (user units)
Final: -

Related Keywords: GSCALE, GMODE.

GCLEAR   **Clears the current graphics window**
Initial: -
Final: -

Related Keywords: GWINDOW.

GDRAW   **Draws to a location**
Initial: Point *userUnits*
Final: -

Related Keywords: GSCALE, GMOVE, GRDRAW.

GETALPHA   **Returns state of the alpha display**
Initial: -
Final: Long 1 (visible) or 0 (off)
                 Related Keywords: ALPHA, GRAPH, GETGRAPH.

GETBATT   **Returns battery status**
Initial: -
Final: Long *status* (0=ok, 1=low, 2=critical)
                 Related Keywords: LOWWARN

GETCH   **Get a character from a path**
Initial: Addr *Path*
Final: LONG *result* (or -1 if path empty)
                 Related Keywords: CONVERTOUT, PUTCH

GETCONTRAST   **Returns the display contrast**
Initial: -
Final: Long: 0 to 100
                 Related Keywords: CONTRAST

GETCONVERTIN   **Get the current filter sequence for incoming data**
Initial: [Text *dest* (default = LCD)], Addr *path*
Final: -
                 See **Path Filters** on page 23-43.
                 Related Keywords: CONVERTIN

GETCONVERTOUT   **Get the current filter sequence for outgoing data**
Initial: [Text *dest* (default = LCD)], Addr *path*
Final: -
                 Related Keywords: CONVERTOUT

GETDFCIN   **Get the default convertin filter for files**
Initial: [Text *dest* (default = LCD)]
Final: -
                 Related Keywords: GETDFCOUT, SETDFCOUT, SETDFCIN, RESETDFC.

GETDFCOUT   **Get the default convertout filter for files**
Initial: [Text *dest*]
Final: -
                 Related Keywords: GETDFCIN, SETDFCOUT, SETDFCIN, RESETDFC.

GETDISP   **Store text and attribute information for a rectangle**
Initial: Rect *area*
Final: DispInfo *dest*
                 Related Keywords: PUTDISP. See **Manipulating Text** on page 23-28.

GETGRAPH   **Returns the state of the graphics display**
Initial: -
Final: Long 1 (visible) or 0 (off)
            Related Keywords: ALPHA, GRAPH, GETALPHA.

GETKEY   **Returns the next key code from the keyboard queue**
Initial: -
Final: LONG *keyCode*
            See **Keyboard Codes** on page 23-29 for interpreting *keyCode*. Key codes can be
            placed in the queue by any of the following means: by either the user typing on the
            keyboard, or by sending characters to a keyboard path (e.g. KBD) using PUTCH, PRINT, or
            XFER. A background transfer to the keyboard will provide characters if none are avail-
            able via typing.
            Related Keywords: KBDRDY, ONKBD, IDLE

GETKEYDEF   **Determines system behavior while waiting processing a GETKEY.**
Initial: Logic x
Final: -

            0 is normal behavior: everything stops until a key is pressed. When non-zero, the sys-
            tem processes all interrupts (except keyboard) just like during an IDLE.

GETMS   **Return time (milliseconds) since power on**
Initial: -
Final: Long *ms*

GETMODNUM   **Returns an application's module number**
Initial: Num sysID
Final: Long modNum
            Returns the application's module counter value. This is the value of the last linked
            module, or else the value last set by SETMODNUM if no modules have been linked since
            then. If *sysID* is <= 0, then the current application is assumed.

GETTARGET   **Retrieves present search target for the system editor**
Initial: [Text *destination* (default = LCD)]
Final: -
            Related Keywords: SETTARGET.

GETTDS   **Get seconds since base time**
Initial: -
Final: LONG *tdSecs*
            Related Keywords: SECS2TD, SETTDS. For timing with resolution higher than 1 second,
            use GETMS.

GETTEXT   **Get the text from a window on the display**
Initial: Rect *area*, Text *dest*, [Num *attr* (ignored)]
Final: -
            area is in absolute display coordinates. *dest* should be large enough to allow for 1 byte

per area element. If not, excess characters are ignored.
Related Keywords: PUTTEXT. To save text+attribute, use GETDISP.

GETWINDOW   **Get the status of the current text window**
Initial: NArray *dest*
Final: -

dest must have at least 8 elements (Table 24-24).

Table 24-24. GETWINDOW / PUTWINDOW Array Information

Element	Item	Element	Item
1	Left column	**5**	Cursor Col
2	Top row	**6**	Cursor Row
3	Right column	**7**	Text Attribute
4	Bottom row	**8**	Cursor Type

Related Keywords: PUTWINDOW.

GGETPORT   **Determine the active port**
Initial: -
Final: Long n

Related Keywords:GSETPORT, GSHIFT, GETTARGET

GHEIGHT   **Returns height (pixels) of current graphics window**
Initial: -
Final: Long *pixelsHigh*
Related Keywords: GWIDTH, GWINDOW.

GICON   **Plot a 5x5 pixel symbol centered on the pen location**
Initial: Num *bytes*
Final: -

bytes is taken as a long, from which the pattern is taken (Figure 24-5).

Byte 1 Bit 7	Byte 1 Bit 6	Byte 1 Bit 5	Byte 1 Bit 4	Byte 1 Bit 3
Byte 1 Bit 2	Byte 1 Bit 1	Byte 1 Bit 0	Byte 2 Bit 7	Byte 2 Bit 6
Byte 2 Bit 5	Byte 2 Bit 4	Byte 2 Bit 3	Byte 2 Bit 2	Byte 2 Bit 1
Byte 2 Bit 0	Byte 3 Bit 7	Byte 3 Bit 6	Byte 3 Bit 5	Byte 3 Bit 4
Byte 3 Bit 3	Byte 3 Bit 2	Byte 3 Bit 1	Byte 3 Bit 0	Byte 4 Bit 7
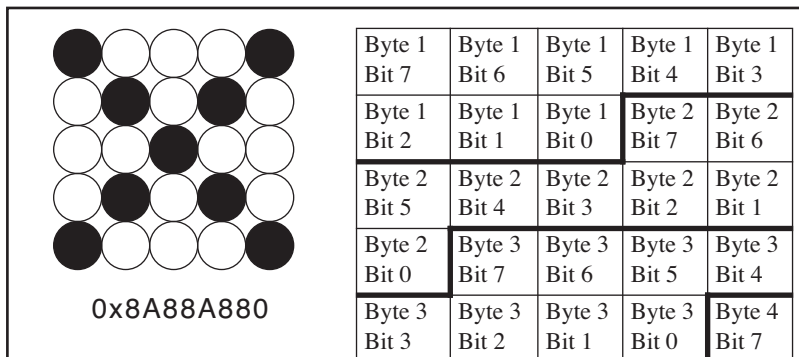
0x8A88A880

Figure 24-5. Relation between location in the 5x5 image and bits in the source Long. In the example on the left, the value 0x8A88A880 results in a 5x5 X image.

Related Keywords: GSCALE, GMODE, GPLOT, GLABEL.

GIGET   **Capture a graphics image**
Initial: Text *dest*, Rect *rect* (user units)
Final: -

The required size of the destination should be obtained with GISIZE, unless the destination is automatically expandable.
Related Keywords: GISIZE, GIPUT.

GINIT   **Initialize the graphics display**
Initial: -
Final: -

The window is set to full display, the scaling is put in pixel units, the scrolling parameters are zeroed, the pen is set to (0,0), the graphics-text mode and graphics-graphics modes are set to *or*.
Related Keywords: GWINDOW, GSCALE, GMODE, GTMODE, GMOVE.

GIPUT   **Put a graphics image back on the display**
Initial: Text *source*, Rect *rect* (user units)
Final: -

Related Keywords: GIGET.

GISIZE   **Returns the array size necessary for a graphics image**
Initial: Rect *rect* (user units)
Final: Long *bytes*

Related Keywords: GIGET, GIPUT.

GLABEL   **Print a label on the graphics screen**
Initial: Text *message*
Final: -

Related Keywords: GMOVE, GLORG, GLSIZE

GLOBALKEYS   **Determines behavior on ONKBD and ONSOFT**
Initial: Logic *onOff* (1=key defs carried over, 0=key defs don't carry)
Final: -

Determines if ONKBD and ONSOFT stay defined if IDLE or TIDLE is encountered during an ONKBD or ONSOFT function call.

GLORG   **Specify the origin position of a graphics label**
Initial: Num *value* (1 through 9)
Final: -

The label's location will be relative to the current pen location, as shown by

```
3          6          9
2 |Graphics Label| 8
          5
1          4          7
```
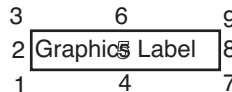
Related Keywords: GLABEL, GLSIZE

GLSIZE   **Get the size (in pixels) of a graphics label**
Initial: Text *theLabel*
Final: LONG *height*, LONG *width*
Related Keywords: GLABEL, GLORG

GMODE   **Set the graphics drawing mode**
Initial: Num *newMode* (0=none, 1=or, 2=eor, 3=and, 4=not)
Final: -
Related Keywords: GMDGET. See **Drawing Modes** on page 23-67.

GMOVE   **Absolute move. Pen moved without drawing.**
Initial: Point *p*
Final: -
Related Keywords: GRMOVE, GWHERE.

GMDGET   **Returns current graphics drawing mode**
Initial: -
Final: Long *oldMode* (0=none, 1=or, 2=eor, 3=and, 4=not)
Related Keywords: GMODE. See **Drawing Modes** on page 23-67.

GPGET   **Is pixel at this location on or off**
Initial: Point *point* (user units)
Final: Long 1 (on) or 0 (off).
Related Keywords: GSCALE, GPPUT, GIGET.

GPLOT **Plot characters or draw a line**
Initial: Num _code_ (0=line, non-zero=plot), NArray _horizUserUnits_, NArray _vertUserUnits_
Final: -

> If _code_ is not zero, it is interpreted as the character to be plotted. Otherwise, a line is drawn connecting the coordinates.
> Related Keywords: GSCALE, GMODE.

GPPUT **Write a pixel at the specified location**
Initial: Point _userUnits_
Final: -

> Related Keywords: GSCALE, GPGET, GICON.

GRAPH **Turns on/off the graphics display.**
Initial: Logic _onOff_ (1=on, 0=off)
Final: -

> Related Keywords: ALPHA, GETALPHA, GETGRAPH.

GRDRAW **Draws relative to the current pen location**
Initial: Point _userUnits_
Final: -

> Related Keywords: GSCALE, GRMOVE, GDRAW.

GRMOVE **Relative move**
Initial: Point _userUnits_
Final: -

> Related Keywords: GMOVE, GSCALE, GWHERE.

GSCALE **Scale the current graphics window**
Initial: Rect _userUnits_
Final:

> All subsequent pen movements are according to this scaling. **Windows and Coordinates** on page 23-63

GSCGET **Returns the current graphics scaling factors**
Initial: -
Final: Double _xMin_, Double _yMax_, Double _xMax_, Double _yMin_
_-or-_
Initial: Rect _scaling_
Final: Rect _scaling_

> Related Keywords: GSCALE.

GSCROLL **Enables/disables automatic graphics window scrolling**
Initial: Num _horizPixels_, Num _vertPixels_
Final: -

> Enables/disables automatic scrolling in the horizontal and vertical. Automatic scrolling will occur whenever the pen is moved or drawn out of the current graphics window. If scrolling is enabled, the graphics window scaling is automatically adjusted to

keep the new pen location visible. Note that when scaling is adjusted, both max and min values are adjusted by the same amount.

GSETPORT   **Set the active port**
Initial: Num i
Final: -

  The active port will receive all subsequent graphics commands.
  Related Keywords: GSHIFT, GETTARGET

GSHIFT   **Shift part of the graphics display**
Initial: Num *dx*, Num *dy*, Rect *userUnits*
Final: -

  Related Keywords: GSCROLL.

GSHOWPORT   **Make a graphics port the (potentially) visible one**
Initial: Num i
Final: -

  The potentially visible port is the one that will be shown when the graphics plane is visible.
  Related Keywords: GRAPH, GETTARGET, GSETPORT, GSHIFT

GSTATUS   **Returns number of graphics ports, and which is potentially visible**
Initial: -
Final: Long visible, Long count

  Related Keywords: GSHIFT, GSETPORT, GSHIFT

GTMDGET   **Returns current graphics-text drawing mode**
Initial: -
Final: Long *currentMode* (1=or, 2=eor, 3=and)

  Related Keywords: GTMODE.

GTMODE   **Set the graphics-text drawing mode**
Initial: Num *newMode* (1=or, 2=eor, 3=and)
Final: -

  This affects what is seen when both graphics and text are visible on the same display device.
  Related Keywords: ALPHA, GRAPH, GTMDGET.

GWHERE   **Returns the pen location**
Initial: -
Final: Double *userHoriz*, Double *userVert*

  Related Keywords: GSCALE.

GWIDTH   **Returns the width of the graphics window in pixels**
Initial: -
Final: Long *pixelsWide*
              Related Keywords: GHEIGHT, GWINDOW.

GWIGET   **Gets the current graphics window**
Initial: -
Final: Long *left*, Long *top*, Long *right*, Long *bottom*
   -or-
Initial: Rect *theWindow*
Final: Rect *theWindow*
              Units are Pixel Hardware Coordinates.
              Related Keywords: GWINDOW

GWINDOW   **Define a graphics window**
Initial: Rect *newWindow*
Final: -
              Units are pixel hardware coordinates. User scaling is <u>not</u> affected by this command -
              the user scaling will be in effect in the new window.
              Related Keywords: GWIGET.

HALT   **Terminate an IDLE or TIDLE**
Initial: -
Final: -
              Related Keywords: IDLE, TIDLE

HASCOPRO   **Is there a coprocessor installed?**
Initial: -
Final: Long *Logic* (1=yes, 0=no)

HIDESOFT   **Hide the function key labels, if any**
Initial: [Softkeys *s*]
Final: -
              Blanks the specified softkeys, or the currently active ones.
              Related Keywords:SHOWSOFT

HSCOUNTS   **High speed counter**
Initial: -
Final: Long *counts*
              Returns the number of counts from the high speed counter since the previous call to
              HSCOUNTS. See **Pulse Counting** on page 16-35.

IDLE   **Wait for interrupt events**
Initial: -
Final: -
              Related Keywords:TIDLE. See **Event Handling** on page 23-33.

IF   **Program flow control**
    Initial: Num *Logic*
    Final: -
                    Related Keywords: ELSE, THEN.

INT   **Returns the code value for type INT**
    Initial: -
    Final: LONG *typeVal*
                    Related Keywords: CHAR, LONG, FLOAT, DOUBLE, PTR, MAKE, MAKE4.

INTERPOL   **Interpolate using two arrays**
    Initial: NArray *yArr*, NArray *xArr*, Num *x*
    Final: Double *y*
                    The *x* value is interpolated using *xArr*, and the corresponding interpolated *yArr* value
                    is returned. The *xArr* array must be sorted in ascending or descending order.

IOCLEAR   **Clears the error code for a path**
    Initial: Addr *path*
    Final: -
                    Related Keywords: IOERR

IOERR   **Returns the latest error (if any) for a path operation**
    Initial: Addr *path*
    Final: LONG *errorNumber* (0=none, 1= read EOF, 2 = write EOF, 3 = seek EOF)
                    See **I/O Errors** on page 23-46.
                    Related Keywords: IOCLEAR

ISBACKLIGHT   **Returns state of the backlight**
    Initial: -
    Final: Long: 1 or 0
                    Related Keywords:BACKLIGHT

ISECHO   **Reports status of LTerm**
    Initial: -
    Final: LONG 1 or 0

                    Related Keywords:ECHO

ISEMPTY   **Is a path empty?**
    Initial: -
    Final: LONG *logic* (1 = empty, 0 = not empty)
                    Related Keywords: PATHSTAT

ISLINK   **Is a module linked**

Initial: Text *name*, Num *sysID*

   -or-

Initial: Num *number*, Num *sysID*

Final: Long logic (1 = yes, 0 = no)

Modules can be referenced by their file specifier (name), or else by their number. If *sysID* is 0, the current application is used.

Related Keywords: LINK, UNLINK, SYSID, SETMODNUM, GETMODNUM

For a discussion of the KBD_ keywords, see **Keyboard Control** on page 23-29.

KBD   **Provides the standard path to the keyboard**

Initial: -

Final: Addr *path*

This path is common to all applications, and cannot be closed by any of them.

Related Keywords:LCD, ARGS, COMM

KBDCLICK   **Enable or disable a beep on each keystroke**

Initial: Num *onOff* (0=off, 1=on)

Final: -

Related Keywords: BEEP

KBDDELAY   **Set the delay time for keyboard repeats**

Initial: Num *milliSecs*

Final: -

KBDEXEC   **Enter LPL's keyboard execution mode**

Initial: -

Final: -

The user is continually prompted ("ok>") to enter function code. Each time the user presses **enter**, the entry is compiled and executed. **esc** exits this mode.

Related Keywords: COMPILE.

KBDREPEAT   **Set the repeat interval when a key is held down**

Initial: Num *milliSecs*

Final: -

LBAPPEND   **Append a line of text to the contents of the list box**

Initial: Text *string*, ListBox *b*

Final: -

Related Keywords:LBSETSTR, LBGETSTR

LPAPPOBJ   **Append a line of text and an associated address to the list box**
Initial: Text *string*, Addr a, ListBox *b*
Final: -

> Related Keywords:LBAPPEND, LBGETOBJ, LBSETOBJ

LBBORDER   **Define the border for the list box**
Initial: Num *border*, [Num *pos*, Text *lab1*, [Num *pos*, Text *lab2*]], ListBox b
Final: -

> *border* is 0 (none), 1 (single line), or 2 (double line). Label positions are shown under WINDOW.
> Related Keywords:LBWINDOW

LBCLEAR   **Clear the contents of the list box**
Initial: ListBox *b*
Final: -

LBCOUNT   **Returns the number of items in the list box**
Initial: ListBox *b*
Final: Long *n*

LBDEFKEY   **Define a function key in the list box**
Initial: Num *action*, Num *keycode*, ListBox *b*
     -or-
Initial: Num *action*, Num *fkey*, Num *keycode*, Text *label*, ListBox *b*
Initial: -

> The first option for defining non-function keys (or unlabeled fct keys). The second is for function keys. The *shortcut* is the ascii key equivalent (C for cut, or whatever). *keycode* is the keycode (**Keyboard Codes** on page 23-29) for the desired key. action is from the list of edit control actions (**EDCTL Codes** on page 24-37). Two common ones are cancel (0xff00) and accept (0xff01).

LBERASE   **Blank the function key labels of a list box**
Initial: ListBox *b*
Final: -

LBEXECUTE   **Begin user interaction with a list box**
Initial: Num code, ListBox *b*
Final: Long *exitKey*

> *exitKey* is the keycode for the key that triggered leaving the list box.
> Code: described below:
> bit 0: 0 = exit on exit keys only, 1 = 1 pass only. exit on every key
> bit 1: 0 - menu bar off, 1 - menu bar on
> bit 2: 0 - print highlight chars, 1 - interpret highlight chars
> bit 3: 0 - no special cursor char, 1 - use special character at cursor location
> bit 4: 0 - normal, 1 - exit on line change

LBFREE   **Free a list box**
Initial: ListBox b
Final: -

Related Keywords:LBNEW

LBGETNODE   **Return a node**
Initial: Text *dest*, Num *code*, Num *index*, ListBox *b*
Final:

If the specified node (0-based *index*) is an XML node, then write to *dest* according to *code*.
code = 0: The node map (list of names, e.g. "li6400 user flow_zero"
code = 1: The node's current value, e.g. "-78.5"
code = 2: The full node value, e.g.
"<li6400><user><flow_zero>-78.5</flow_zero></user></open>".

LBGETOBJ   **Returns the object associated with a line in the list box**
Initial: Num *line*, ListBox *b*
Final: Addr a, Long 0
    -or-
Final: 1

line = 0 is the top line, and it is sent to *dest*.

LBGETSTR   **Returns the string associated with a line in the list box**
Initial: Text *dest*, Num *line*, ListBox *b*
Final: -

line = 0 is the top line, and it is sent to *dest*.

LBNEW   **Create a new List Box**
Initial: Num *sysid*, Num *nkeys*, Num *nlines*
Final: ListBox *b*

nkeys is the number of softkeys the list box will have available (5, 10, etc.), and *nlines* is the number of lines they occupy (typically 1 or 2). *sysid* identifies the owner program. Use 0, unless you want to reassign it.

LBSEL   **Get the selected line number**
Initial: ListBox *b*
Final: Long *index*

index = 0 is the first (top) line.

LBSETOBJ   **Associate an object with a line in the list box**
Initial: Addr *obj*, Num *line*, ListBox *b*
Final: -

line = 0 is the top.

LBSETSTR   **Set a line in the list box**
Initial: Text *string*, Num *line*, ListBox *b*
Final: -

             line = 0 is the top.
             Related Keywords:LBAPPEND

LBSTART   **Pick the cursor line in the list box**
Initial: Num *line*, ListBox *b*
Final: -

             line = 0 is the top line.

LBSTARTNODE   **Pick the cursor line in the list box**
Initial: Text *nodeMap*, Num *startIndex*, ListBox *b*
Final: -

             Starts at line (0-based) *startindex*, and looks for the first line with the matching *nodeMap*.

LBWINDOW   **Define the window for the list box**
Initial: Rect *area*, Num *border*, [Num *pos*, Text *lab1*, [Num *pos*, Text *lab2*]], ListBox b
Final: -

             border is 0 (none), 1 (single line), or 2 (double line). Label positions are shown under WINDOW.
             Related Keywords:LBBORDER

LCD   **Provides the standard path to the display**
Initial: -
Final: Addr *path*

             This path is common to all applications, and cannot be closed by any of them.
             Related Keywords:COMM, ARGS, KBD

LGT   **Log base 10**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
             Related Keywords:LOG

LINK   **Load and compile source code, and link it to an existing application.**
Initial: [Num *sysID*, Addr *messPath*, Num *code*], Text *modName*, Addr *sourcePath*
       -or-
Initial: [Num *sysID*, Addr *messPath*, Num *code*], Text *fileName*, Num *anything*
Final: Long *error* (0=ok, non-zero = failed)

             To link a file, have any numeric value on the top of the stack. Otherwise, have a path containing the source code itself, followed by the name of the module. In the case of

files, the file name is used for the module name.

Table 20-21. The optional code parameter of LINK.

Code	Action
Not there	SysID defaults to the current application
1	Normal compile.
2	Collect cross reference information, retain internally.
3	Collect cross reference information, write it to *messPath*

Related Keywords: UNLINK, ISLINK

LOCK **Lock or unlock an array. A locked array can't be resized implicitly.**
Initial: Num code, Array array
Final: -

If code is 0, unlock. Otherwise, lock the array.

LOG **Natural log**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
Related Keywords: EXP, LGT

LONG **Returns the code value for type LONG**
Initial: -
Final: LONG *typeVal*
Related Keywords: INT, CHAR, FLOAT, DOUBLE, PTR, MAKE, MAKE4.

LOOP **Program looping**
Initial: -
Final: -

Related Keywords: NLOOP, ENDLOOP, BREAK, BREAKIF

LOWWARN **Enable / disable low battery warning**
Initial: Num *onOff* (0=off, 1=on)
Final: -

Related Keywords: GETBATT

LPL   **Return the LPL version number**
  Initial: [Text *dest* (default = LCD)]
  Final: -

LWC   **Lower case**
  Initial / Final: (See **Single Value Transforms** on page 24-14.)
                              Related Keywords: UPC

MAKE   **Dynamic array allocation**
  Initial: Num *typeCode*, Num *numElements*
  Final: Array *newArr*, Long 0
        -or-
  Final: Long 1

                              IMPORTANT: When an application goes out of scope (ends), any arrays that have
                              been allocated but not FREEd are disposed of automatically.
                              Related Keywords: FREE, MEM, MAKE4.

MAKE4   **Dynamic array allocation**
  Initial: Num *typeCode*, Num *numElements*, NUM *sysID*
  Final: Array *newArr*, Long 0
        -or-
  Final: Long 1

                              Allocates an array for a particular application. IMPORTANT: When an application
                              goes out of scope (ends), any arrays that have been allocated but not FREEd are disposed
                              of automatically.
                              Related Keywords: FREE, MEM, MAKE

MAKESOFT   **Create function key structure and activate it**
  Initial: Num *de*, Num *dc*, Num *le*, Num *nLines*, Num *mode*, Num *nKeys*
  Final: -

                              See **The Function Keys** on page 23-36.

Table 24-25. MAKESOFT parameters

de	Delimiter enhancement Table 24-39 on page 24-85
dc	Delimiter character.
le	Label enhancement. Table 24-39 on page 24-85
mode	bit 0 = 0: Key labels always visible. bit 0 = 1: Key labels are visible only after Labels key pressed, and disappear after a non-softkey is pressed. bit 1 = 0: Use '+' for multi-level indicator. bit 1 = 1: Use 1+level for level indicator.
nKeys	number of keys

Related Keywords: FREESOFT, ONSOFT, NEWSOFT

MATHERR   **Enable / disable math error corrections**
Initial: Num *onOff* (0 = report errors, 1=overlook errors)
Final: -

The following math errors normally generate fatal errors, but they can be overlooked with MATHERR:

Table 24-26. Math Errors Fixed by MATHERR

Error	How Fixed
Divide by 0	Result is 0
Argument out of range	Result is 0
Negative number raised to a non-integer power	Result is 0

MAX   **Return maximum value**
Initial / Final: (See **Two Value Transforms** on page 24-15.)

MEM   **Returns memory status**
Initial: -
Final: Long *largestBytes*, Long *totalBytes*

MEMMAP   **Prints allocation information about the heap**
Initial: [Path *dest* (default = COMM)]
Final: -

MESSBOX   **Pop up a boxed message**
Initial: [*aux2*,] [*aux1*,] Num *code*, [Num *pos1*, Text *lab1*, [Num *pos2*, Text *lab2*]] Text *messge*
Final: Long *key*

Displays the text *message* in a framed window, with optional labels. The message can time out, or wait for any or one of an allowed set of keystrokes. This is determined by the *code* parameter.

code values:
bit 0: if set, any keystroke will clear the message. If there are no labels specified, then "Press Any Key" will be put on the lower frame.
bit 1: if set, a particular keystroke will clear. An Narray *aux1* will be popped that is assumed to contain the list of allowed keys (**escape** is always assumed to be an allowed key).
bit 2: if set, a timeout time (secs) will be popped from the stack, either *aux2* (if bit 1 set) or *aux1* (if bit 1 not set).

If code is 0, then the message is displayed, and execution continues with no waiting. Return code *key* will be 0.

MIN   **Returns min value**
Initial / Final: (See **Two Value Transforms** on page 24-15.)
                    Related Keywords:MAX

MOD   **Returns the remainder of A/B**
Initial / Final: (See **Two Value Logical Transforms** on page 24-16.)

MODSIZE   **List compiler size directive settings for all modules in an application**
Initial: Num *low*, Num *High*, Addr *path*, [Num *SysID*]
Final: -

                    Starts at module number *low*, and proceeds through module number *high*. For each
                    module, the following information is written to *path*. This function is useful for "tun-
                    ing" module's compiler directives for efficient use of memory. A program to do this
                    is "/Sys/Lib/UpdateSizes".

```
:CODE nn
:DATA nn
:NAMES nn
:PRIVCOUNT nn
:PUBCOUNT nn
:STATCOUNT nn
```

MOVETEXT   **Copy a block of text and attribute within a text window**
Initial: Num *toLeft*, Num *toTop*, Rect *area*
Final: -

                    toLeft, *toTop*, and *area* are in relative window coordinates.

NEWSOFT   **Create function key structure**
Initial: Num *de*, Num *dc*, Num *le*, Num *nLines*, Num *mode*, Num *nKeys*
Final: SoftKeys

                    Same as MAKESOFT, except it returns SoftKeys pointer instead of implementing it.
                    Related Keywords:MAKESOFT, FREESOFT

NEXTKEY   **See what is next in the keyboard queue, without removing it**
Initial: Num -
Final: NUM *result* (*keyCode* or -1 if queue is empty)
                    To decode *keyCode*, see **Keyboard Codes** on page 23-29.
                    Related Keywords: GETKEY, UNGETKEY

NLADD   **Add an entry to the NameList**
Initial: Text *shortlab*, Addr *ptr*, Num *id*, NameList *list*
Final: -

                    id - the identifier number for the object
                    ptr - the address of the object
                    shortlab - the short label name of the object

NLCLEAR   **Clears all entries in a NameList**
Initial: NameList *list*
Final: -

NLCOUNT   **Returns the number of items in the NameList**
Initial: NameList *list*
Final: Long *size*

NLFIND   **Find an item in the NameList based on it's short name**
Initial: Text *name*, NameList *list*
Final: Long *index*
                  index is 0 based (0 is first item), or -1 if object not found.

NLFREE   **Free a NameList**
Initial: NameList *list*
Final: -

NLGET   **Get the ith item from the list**
Initial: Num *index*, NameList *list*
Final: Ptr ptr, Long *id*, 1
     -or-
Final: 0 (*index* out of range)
                  If index (0 based) is in range, returns the item's *id* and it's pointer.

NLGETSTR   **Get the ith item label from the list**
Initial: Text *dest*, *index*, NameList *list*
Final: -
                  If index (0 based) is in range, writes the item's *label* to *dest*.

NLNEW   **Create a new NameList**
Initial: Num *sysId*
Final: NameList *n*
                  sysId is the id of the owning application. 0 = use current app.

NLOOP   **Loop a finite number of times**
Initial: Num *n*
Final: -
                  Related Keywords: LOOP, ENDLOOP, BREAK, BREAKIF

NOT   **Logical NOT**
Initial / Final: (See **Single Value Transforms** on page 24-14.)

For a discussion of the NT_ keywords, see **Networking in LPL** on page 23-51.

NTCLIENTS   **Returns the number of clients**
Initial: Num service
Final: Long *count*

   -or-
Initial: NArray *dest,* Num service
Final: Long *count*

Returns the number of remote clients for *service*. If *service* = 0, all active services are considered. If *dest* is specified, fills it with the list of client reference numbers. These are numbers that can be used for NTCLOSECLT, for example.

NTCLOSECLT   **Close a client connection**
Initial: Num *clientID*
Final: Long 0 (ok) or <0 (error)

Closes the client connection.
Related Keywords: NTOPENCLT

NTCLOSESRV   **Close a service**
Initial: Num *service*
Final: Long 0 (ok) or <0 (error)

Shuts down the specified service.
Related Keywords: NTOPENSRV

NTGETHOST   **Gets the host name**
Initial: Text *destination*
Final: Long 0 (ok) or <0 (error)

Writes current host name to the *destination* (if it is a path), or sets *destination* to it if it is a string.
Related Keywords: NTSETHOST

NTGETIP   **Returns the IP address**
Initial: Text *destination*
Final: Long 0 (ok) or <0 (error)

Writes current IP address to the *destination* (if it is a path), or sets *destination* to it if it is a string.

NTGETMAC   **Returns the MAC address**
Initial: Text *destination*
Final: Long 0 (ok) or <0 (error)

Writes MAC address to the *destination* (if it is a path), or sets *destination* to it if it is a string.

NTGETUSER   **Returns the user name**
Initial: Text *destination*
Final: Long 0 (ok) or <0 (error)

Writes the user name (lpl) to the *destination* (if it is a path), or sets *destination* to it if it is a string.
Related Keywords:NTSETUSER

NTINFO   **Get information on services or clients**

Initial:	Text *destination,* Num 0
Final:	Long *count*
	-or-
Initial:	Text *destination,* Num *typemask,* Num 1
Final:	Long *count*
	-or-
Initial:	Text *destination,* Num *pid,* Num 2
Final:	Long *count*
	-or-
Initial:	Text *destination,* Num 3
Final:	Long *count*
	-or-
Initial:	Num *service,* Num 4
Final:	Long *0 or 1*

0 - returns a status list of all clients.
1 - returns a status list for all clients whose type matches the mask.
2 - returns a status line for the specified client.
3 - returns a list of all active services
4 - indicates whether the specified service is active (returns 1) or not (returns 0).

Version 0 through 2 writes one or more lines to the destination describing clients. Each line has the following tab-delimited format:

pid   type   address   service [*state   localID   remoteID*]

where *pid* is the internal reference number for the client, *type* is shown below, *address* is the address of the client (if known), and *service* is what service the client is using. The three final fields are only present for type 4 clients.

The *type* entry is one of the following:
1 - RS-232 client
2 - Ethernet client
4 - li6400.licor.com
8 - Comm only client (6409)
16 - Other

The extra fields for type 4 clients are the local and remote ID strings, and an indicator of the state of the connection:

state = 0 - not connected, 1 - waiting to authenticate, 2 - connected, 3 - reconnecting (authentication timed out).

Versions 3 and 4 pertain to services. The string returned or written by version 3 is simply a list of services. e.g., 6400 6409.

NTOPENCLT   **Open a connection**
Initial: [Path *message*],Text *remoteID*, Text *localID*,] Num *port*, Text *address*
Final: Long pid (ok) or -1 (failed)
              Open a connection. If connecting to li6400.licor.com, then include the *remoteID* and
              the *localID* and an optional message path
              Related Keywords:NTCLOSECLT

NTOPENSRV   **Start a service**
Initial: Num *port*
Final: Long 0 (ok), <0 (failed)
              Start a service.
              Related Keywords:NTCLOSESRV

NTSETHOST   **Sets the host name**
Initial: Text *name*
Final: Long 0 (ok) or <0 (error)
              Sets the host name.
              Related Keywords: NTGETHOST

NTSETPASS   **Sets the password**
Initial: Text *password*
Final: Long 0 (ok) or <0 (error)
              Set the password.

NTSETUSER   **Sets the user name**
Initial: Text *name*
Final: Long 0 (ok) or <0 (error)
              Sets the user name. Note: This will always return -1, since we don't let the user name
              change from lpl.

NTSTATE   **Returns the state of the ethernet connection**
Initial: -
Final: Long *state*
              State is described in Table 24-27.

Table 24-27. Return value of NTSTATE.

State	Meaning
-1	Console is not an LI-6400XT.
0	ok
1r	Network card not plugged in.
2	Card in, but cable not connected.

OFFA2D   **Disable an ONA2D condition**
Initial: Num *group*
Final: -

        Related Keywords:ONA2D

OFFCOMM   **Disable ONCOMM condition**
Initial: -
Final: -

        Related Keywords:ONCOMM

OFFCYCLE   **Disable an ONCYCLE timer**
Initial: Num *timerNum* (1, 2, or 3)
Final: -

        Related Keywords:ONCYCLE

OFFKBD   **Disable ONKBD condition**
Initial: -
Final: -

        Related Keywords:ONKBD

OFFSOFT   **Disable ONSOFT for a key**
Initial: Num *keyNum*
Final: -

        Related Keywords:ONSOFT

OFFTIC   **Disable ONTIC timer**
Initial: -
Final: -

        Related Keywords:ONTIC

OFFTIME   **Disable ONTIME alarm**
Initial: -
Final: -

        Related Keywords:ONTIME

ONA2D   **Enable interrupt when an A/D group is ready**
Initial: Fct *funtion*, Num *groupNum*
Final: -

        Related Keywords: IDLE, TIDLE, OFFA2D

ONCOMM   **Enable interrupt when a target char is received on the Comm port**
Initial: Num *character*, Fct *function*
Final: -

        Execute *function* every time *character* is received during the ongoing or subsequent
        (T)IDLE. An overlapped transfer (XFER) from the comm port must be ongoing.
        Related Keywords: IDLE, TIDLE, OFFCOMM

ONCYCLE   **Enable interrupt of a timer**

Initial: Num *seconds*, Num *timerNum* (1, 2, or 3), Fct *function*

Final: -

Define a function to be periodically called during the ongoing or subsequent (T)IDLE. This is the same idea as ONTIC, except the period is user defined, and up to three different periods can be used during one (T)IDLE. Each application has 3 cycle timers that are independent from any other application. The timers are global to any (T)IDLE within an application, however. Thus, any function called from within a (T)IDLE can modify any timer's parameters (period and function address), and the changes remain in effect until another ONCYCLE or OFFCYCLE is encountered.

Related Keywords: IDLE, TIDLE, OFFCYCLE

ONKBD   **Enable interrupt when a keystroke is available**

Initial: Fct *function*

Final: -

Define a function to be called every time a keystroke is available during the *next* (T)IDLE. See also GLOBALKEYS. To access the keystroke that is available, use GETKEY.

Related Keywords: IDLE, TIDLE, OFFKBD

ONSOFT   **Enable interrupt when a function key is pressed**

Initial: Text *label*, Fct *function*, Num *keyNum*

Final: -

Define a label and action for a function key to be performed at the next (T)IDLE (or the ongoing (T)IDLE, depending on GLOBALKEYS). MAKESOFT must first have been performed to build space for the function key information.

ONKBD and ONSOFT interaction: If no function keys are defined (MAKESOFT not in effect), then any function key stroke is passed to the ONKBD function. If function keys are defined (MAKESOFT in effect), then function key strokes do not pass through to the ONKBD function. If function keys are defined, but a function key is pressed for which no ONSOFT function was defined, then that keystroke is ignored.

Related Keywords: IDLE, TIDLE, OFFSOFT

ONTIC   **Enable a 1 second interrupt**

Initial: Fct *functon*

Final: -

Related Keywords: IDLE, TIDLE, OFFTIC

ONTIME   **Enable an alarm clock interrupt**

Initial: Num *timeDateSecs*, Fct *function*

Final: -

Define a function to be executed once when a certain time is reached. If this time occurs when there is no (T)IDLE in effect, then the function will not be executed until the next (T)IDLE.

Related Keywords: IDLE, TIDLE, OFFTIME

For a discussion of the OPEN_ keywords, see **I/O Programming** on page 23-38.

OPEN_BUFF   **Open a path to an expandable memory buffer**
Initial: Num *initialSize*
Final: Addr *path*, Long 0
    -or-
Final: Long 1

        Related Keywords: CLOSE

OPEN_CHARS   **Open a path to an char array**
Initial: CArray *string*
Final: Addr *path*, Long 0
    -or-
Final: Long 1

        The initial fill value is set to the ready value of the array (see **Array SIZE and READY values** on page 23-6) After the path is closed, the ready value is set from the fill value of the path. While the path is open, bear in mind that you have two independent methods of manipulating data in the character array: the path keywords (PRINT, ENTER, etc.) and the normal array keywords (APP, ENTER, etc.), and also that there is no immediate interaction between the SETREADY keyword and the SETFILL or SETEMPTY keywords.
        Related Keywords: CLOSE

OPEN_COMM   **Open a path to the Comm port**
Initial: -
Final: Addr *path*, Long 0
    -or-
Final: Long 1

        Every application has available to it the standard path COMM. One reason to open another path to the comm port would be to implement different filtering without changing the standard filters associated with COMM.
        Related Keywords: CLOSE

OPEN_FILE   **Open a path to a file**
Initial: Text *opts*, Text *name*
Final: Addr *path*, Long 0
    -or-
Final: Long 1

        The opts string is made up of one or more of the items in Table 24-28. For example, "wa" is write-append.

Related Keywords: CLOSE

Table 24-28. Options for FILE_OPEN

opts	**File exists**	**File doesn't exist**
r	ok	Error
w	Opened and truncated	File created
a	Opened for appending	File created

OPEN_FILE_ASK   **Open a file using the Standard File Dialog**

Initial: Num *opts*, Text *name*, Text *prompt*, [Text *helpInfo*, Num *code*]
Final: Addr *path*, Long 0
    -or-
Final: Long 1

Open a file using the operating system's Standard File Dialog box.

Table 24-29. OPEN_FILE_ASK parameters

code	Optional. Value doesn't matter. The presence of a numeric value here indicates that the next item on the stack is help text.
helpInfo	If *code* is present. The text to be displayed in to dialog box if the user presses the **HELP** key.
prompt	The prompt to be used in the dialog box.
name	The starting name of the file to be opened. On return, holds the selected name.
opts	0 = read only, 1 = write only, 2 = read and write.

If *opts* is 1 or 2, and the file exists, the user is asked if the file is to be overwritten or appended.
Related Keywords:CLOSE

OPEN_KBD   **Open a path to the keyboard**

Initial: -
Final: Addr *path*, Long 0
    -or-
Final: Long 1

Every application has available to it the standard path KBD. One reason to open another path to the keyboard would be to implement different filtering without changing the standard filters associated with KBD.
Related Keywords:CLOSE

OPEN_LCD   **Open a path to the display**
Initial: -
Final: Addr *path*, Long 0
  -or-
Final: Long 1

> Every application has available to it the standard path LCD. One reason to open another path to the display would be to implement different filtering without changing the standard filters associated with LCD.
> Related Keywords: CLOSE

OPEN_QUE   **Open a path to a circular queue**
Initial: Num *initialSize*
Final: Addr *path*, Long 0
  -or-
Final: Long 1

> The actual size of the queue may differ from the requested size. Queues are sized 7, 15, 31, 63, 127, 255, 511, 1023, 2047, 4095, 8191, 16383, or 32767 bytes. If a different value is requested, the next largest size is used, but is never larger than 32767.
> Related Keywords: CLOSE

OR   **Logical OR**
Initial / Final: (See **Two Value Logical Transforms** on page 24-16.)
> Related Keywords: AND, NOT

PATHSTAT   **Get status information for a path**
Initial: Num *code*, Addr *Path*
Final: LONG *value*
  -or-
Final: LONG *empty*, LONG *fill*, LONG *size*, LONG *type*
> *code* and *type* are given in Table 24-30, and Table 24-31.

Table 24-30. PATHSTAT Code Values

code	value
1	Path type: See Table 24-31
2	Path size
3	Path fill
4	Path empty

Table 24-31. Path Types

Type	Path to...	Type	Path to...
0	keyboard	**5**	circular queue
1	display	**6**	file (read only)
2	comm port	**7**	file (write only)
3	CHAR array	**8**	file (read + write)
4	memory buffer		

PDRF   **Pointer de-reference**

Initial: Addr *ptr*
Final: Addr *object*

Returns the object directly pointed at, if *ptr* is a pointer. PVAL, on the other hand, tracks a pointer to it's ultimate non-pointer object.
Related Keywords: PVAL

PFIND   **Finds a pointer referenced in a pointer array**

Initial: Parray *arr*, Addr *target*
Final: Long *subscript* (or 0 if not found)

Related Keywords: PVAL, FIND. See **Using PTR arrays** on page 23-13.

PICK   **Return address of an element in an array**

Initial: Array *arr*, Num *subscript*
              -or-
Initial: Num *subscript*, Array *arr*
Final: Addr *address*

If *subscript* is 0 or greater than the array size, an error will result.
Related Keywords: VAL, PVAL, SIZE. See **Array Operations** on page 23-6.

POLY   **Computes a polynomial**

Initial: Double Array *coeffs*, Num *arg*
Final: Double *result*

The power of the polynomial is determined from the number of values in the coefficient array *coeffs*. 2 values = 1st order, 3 values = 2nd order, etc.

POSXY   **Position the cursor within the current text window**

Initial: Num *column*, Num *row*
Final: -

Related Keywords: GETWINDOW.

POWEROFF   **Powers off the instrument**
Initial: -
Final: -

PRCOUNT   **Returns the number of items found in the last line parsed**
Initial: PATH p
Final: Long COUNT
          Related Keywords:PRNEXTLINE, PRQCOUNT

PRDELIM   **Set the delimiters for a path's parsing tool**
Initial: Text *list*, PATH *p*
Final: -

          Default delimiters are space, double quote, and tab. To return to default, do

          " \"\t" path PRDELIM

          Related Keywords:PRNEXTLINE, PRGET

PRGET   **Retrieve a parsed item**
Initial: Text *dest*, Num *index*, PATH *p*
     -or-
Initial: NAddr *dest*, Num *index*, PATH *p*
Final: -
     -or-
Initial: Long *x*, Num *index*, PATH *p*
Final: Long *value*
     -or-
Initial: Double *x*, Num *index*, PATH *p*
Final: Double *value*
          index is 0-based (0 is first item). The item can be retrieved as a string, or converted to
          a numerical value.
          Related Keywords:PRNEXTLINE, PRDELIM,

PRINT   **Print to a path**
Initial: ..., [CArray *fmtString*], [PATH *dest* (default = LCD)]
Final: -

          Print and Pointers
          PRINT starts adding characters to a path at the fill pointer location. As characters are
          added, the fill pointer is updated.

          The Format String
          Characters contained in the format string are output to the destination path, except for
          format specifier fields, which take the form

`% [(`*number* `[`*delim*`]])` `[`*flags*`]` `[`*width*`]` `[.`*precision*`]` *type*

Table 24-32. Format string elements for PRINT

(number [delim])	Optional. Used when outputting an array. Number can be the number of elements to output, or * for all elements. The optional delim character is used, if specified, as the delimiter between elements.
flags	Optional. The flag characters are -, +, and space, as indicated by Table 24-33
width	Optional. If present, specifies the minimum number of characters to print, padding with blanks or zeros.
precision	Optional. If present, specifies the maximum number of characters to print; for integers, minimum number of digits to print.
type	Must be present. Specifies the type of item to be printed. Note that what is popped from the stack is converted to this type. See Table 24-34 on page 24-75

Table 24-33. PRINT Flag Characters

Flag	Meaning
-	Left justifies the result, pads on the right with blanks. If not given, right-justifies the result, pads on the left with zeros or blanks.
+	Numeric values always begin with a + or - sign.
(space)	If the value is non-negative, output begins with a blank instead of a plus; negative values still begin with a minus.

Table 24-34. PRINT Type Codes

Type	Meaning	Type	Meaning
d or i	signed int	f	floating point
o	signed octal int	e	scientific notation with e
u	unsigned decimal int	E	scientific notation with E

Table 24-34. (Continued)PRINT Type Codes

Type	Meaning	Type	Meaning
x or X	unsigned hex int	g	floating point in e or f form, based on precision.
ld or li	signed long	G	floating point in E of f form, based on precision.
lo	signed octal long	c	character
lx or lX	unsigned hex long	s	string
lu	unsigned decimal long	p	pointer address
k	outputs an unsigned int as \kHHLL, where HH is the high byte in hex, and LL is the low byte in hex.		

Table 24-35. PRINT Format Examples

Item on Stack	Format	Result
12 (LONG)	%8d	" 12"
"abcdefg"	%5.5s	"abcde"
1.2345	%7.2f	" 1.23"
1.2345	%-7.1f	"1.23 "
"ABC"	%(*,)d	"65,66,67"

PRINT and PTR Arrays
There should be a % specifier in the format string for each non-pointer array item in the pointer array.

Unformatted PRINT
If no format string is found, then one item on the stack is printed using a default format, based on the type of item that is found (Table 24-36).

Table 24-36. PRINT Default Formats

Stack Object	Format Used	Stack Object	Format Used
LONG	%ld	DOUBLE Addr	%lG
DOUBLE	%lG	CArray	%s

Table 24-36. PRINT Default Formats

Stack Object	Format Used	Stack Object	Format Used
CHAR Addr	%c	INT Array	%(*)d
INT Addr	%d	LONG Array	%(*)ld
LONG Addr	%ld	FLOAT Array	%(*)g
FLOAT Addr	%f	DOUBLE Array	%(*)lg

PRLINE   **Returns the last line that was parsed**
Initial: Text *dest*, PATH *p*
Final: -

PRNEXTLINE   **Parse the next line of a path**
Initial: PATH *p*
Final: Long *count*

        Starting at the path's current empty pointer, data is read and parsed until an end of line of end of path is reached. *count* is the number of parsed items found. Access the items by PRGET. This operation moves the empty pointer.

PRQCOUNT   **Returns the number of quoted items in the last line parsed**
Initial: PATH *p*
Final: Long *count*

PRQUOTE   **Defines the quote character(s)**
Initial: Text *quotes*, PATH *p*
Final: -

        The default is a double-quote character ("\"").

PTR   **Returns the code for type PTR**
Initial: -
Final: LONG *typeVal*

        Related Keywords: INT, LONG, FLOAT, DOUBLE, CHAR, MAKE, MAKE4.

PTRTOLL   **Converts an LPL's address into address and type**
Initial: ADDR *Ptr*
Final: Long *address*, LONG *type*

        Related Keywords:USES

PUTCH   **Add a character to a path**
Initial: Num *theChar*, Addr *Path*
Final: LONG *result* (theChar if ok, or -1 on error)

        Related Keywords: GETCH

PUTDISP   **Redisplay text and attribute information**
Initial: DispInfo *old*
Final: -

        This disposes the structure pointed at by *old*, so don't use it again!
        Related Keywords: GETDISP

PUTTEXT
Initial: Rect *area*, Text *source*, [Num *attr*]
Final: -

        area uses absolute display coordinates. *attr*, if present, is imposed on the rectangle. If missing, attributes are unchanged.
        Related Keywords: GETTEXT.

PUTWINDOW   **Implements window, attribute, and cursor information**
Initial: NArray *source*
Final: -

        Related Keywords: GETWINDOW

PVAL   **Returns the address pointed at**
Initial: PAddr *p*
Final: PAddr *final*
        Related Keywords: VAL

RAD   **Enable "radians mode" for trig functions.**
Initial: -
Final: -

        Related Keywords: DEG, and all trig functions

RANDOMIZE   **Randomize the random generator seed**
Initial: -
Final: -

        Related Keywords: RND

READY   **Returns the number of elements in the array**
Initial: Array *arr*
Final: Long *number*
        Related Keywords: SETREADY, SIZE.

REGRESS   **Do a linear regression**
Initial: Num *power*, NArray *xArray*, NArray *yArray,* DArray *coeffs*, Num *forceZero*
Final: Double *determinant*
        Size of *coeffs* should be at least as large as 1 + *power*. If *forceZero* is true (not 0), the first coefficient will always be 0.
        Related Keywords: POLY

RESET   **Sets fill and empty pointers to 0**

Initial: Addr *path*
Final: -

RESETDFC   **Reset the default convertout and convertin filters for files**

Initial: -
Final: -

Platform specific - See Table 24-37. See **Path Filters** on page 23-43.

Table 24-37. Default File Filters

Platform	ConvertOUT	ConvertIN
LI-6400	"e"	"e"
DOS	"e"	"E"
Macintosh	"e"	"n"

Related Keywords: SETDFCIN, SETDFCOUT, GETDFCIN, GETDFCOUT

RESIZE   **Resize an array.**

Initial: Num n, Array array
Final: -

Resizes the specified array.

RESTART   **Restart the processor (simulates power off then on)**

Initial: -
Final: -

This is equivalent to pressing **shift ctrl escape** on the keyboard, and should be done after loading a disk image.
Related Keywords: POWEROFF

RETURN   **Exit from a function.**

Initial: -
Final: -

For a discussion of the RMT_ keywords, see **Registered Variable Support** on page 23-83.

RMTADD   **Add a variable to the list**

Initial: Text *format*, Text *name*, Num *id*, Addr *address*, RMT *rmt*
Final: -

RMTCLEAR   **Clear the list**
Initial: RMT *rmt*
Final: -

RMTFREE   **Dispose of the list**
Initial: RMT *rmt*
Final: -

RTMNEW   **Create a list**
Initial: Num *sysID*, Text *name*
Final: RMT *rmt*

RMTSIGNAL   **Signal the remote terminal about the list**
Initial: Num *code*, RMT *rmt*
Final:

                    Code = 1: The list is defined. *Code* = 2: The values may have changed.

RND   **Generate and random number between 0 and 1.**
Initial: -
Final: Double *value*
                    Related Keywords: RANDOMIZE

ROT   **Exchange the 1st and 3rd items on the stack**
Initial: Obj *a*, Obj *b*, Obj *c*
Final: Obj *c*, Obj *b*, Obj *a*
                    Related Keywords: SWAP, DUP, DROP

RTGACTIVATE   **Make an RTG manager active (potentially visible)**
Initial: RTG *manager*
        -or-
Initial: Num *port*, RTGArray *list*
Final: -

                    Activate the selected manager, or activate the manager corresponding to the specified port.

RTGADD   **Update an RTG (or list of RTGs)**
Initial: RTG *manager*
        -or-
Initial: RTGArray *list*
Final: -

                    A strip chart will add data. An XY chart will move the pointer.

RTGADD2   **Add arrays of certain data to an RTG (or list of RTGs)**

Initial: RTG *manager*, ADDR *x*, ADDR *y*, NArray *xData*, NArray *yData*

       -or-

Initial: RTGArray *list*, ADDR *x*, ADDR *y*, NArray *xData*, NArray *yData*

Final: -

> x and y are the addresses of the x and y quantities being passed in.
> xData and yData are arrays of x and y data.
> Armed with this information, each RTG manager checks its plots to see if it is plotting any of these quantities. If you, its arrays are updated with the data.
>
> This only affect strip charts. XY plots ignore this.

RTGCLEAR   **Clear data from all graphs in an RTG (or list of RTGs)**

Initial: RTG *manager*

       -or-

Initial: RTGArray *list*

Final: -

RTGDEF   **Define a curve for an RTG**

Initial: 0, Num *plotNum*, RTG *manager*

       -or-

Initial: Num *id*, Num *ymin*, Num ymax, Num *yinc*, Num *yscale*, Num *xid*, Num *xr1*, Num *xr2*, 1, Num *plotNum*, RTG *manager*

       -or-

Initial: Num *id*, Num *ymin*, Num *ymax*, Num *yinc*, Num *yscale*, Num *xid*, Num *xmin*, Num *xmax*, Num *xinc*, Num *xscale*, 2, Num *plotNum*, RTG *manager*

> The third parameter defines the plot: 0 = none, 1 = strip chart, 2 = xy.
>
> For strip charts,
> *xr2* - Number of seconds worth of data to buffer
> *xr1* - Starting number of seconds of data to display
> *xid* - ID of the x (time) variable.
>
> The remaining parameters,
> *xscale* and *yscale* - Scaling option for the x or y axis (see below).
> *xmin* and *ymin* - Min axis value
> *xmax* and *ymax* - Max axis value
> *yid* - ID of the y variable
>
> The scaling option is one of the following:
>
> 0 - Do not adjust axis scaling.
> 1 - Adjust the minimum axis only, based on plotted data.
> 2 - Adjust the maximum axis only, based on plotted data.
> 3 - Adjust both min and max, but keep the delta the same.
> 4 - Adjust both min and max independently, based on plotted data.

RTGEDIT **User editing of an RTG**
Initial: RTG *manager*
Final: Long 1 (changed) or 0 (no change)

RTGFREE **Dispose of an RTG Manager**
Initial: RTG *manager*
Final: -

RTGLOG **Send a log event to an RTG manager (or list of managers)**
Initial: RTG *manager*
-or-
Initial: RTGArray *list*
Final: -

This adds a point to an XY plot. Strip charts mark the event with a small arrow on the bottom of the graph.

RTGMAKE **Redefine an RTG manager**
Initial: NameList *list*, Text *dfltdir*, Text *vname*, Num *char*, Num *gport*, Num *nkeys,* RTG *manager*
Final: -

See RTGNEW for meanings of parameters.

RTGNAME **Returns the name and state of an RTG manager**
Initial: Text *dest*, RTG *manager*
Final: -

RTGNEW **Create a new RTG Manager for a graphics plane**
Initial: NameList *list*, Text *dfltdir*, Text *vname*, Num *char*, Num *gport*, Num *nkeys*
Final: RTG *manager*

nkeys - Number of graphics function keys desired (5, 10, etc.)
gport - Graphics port to be used
char - The char associated with this graph. One of *vname*.
vname - Vertical right-hand label, up to 8 characters in length.
dftdir - Directory for reading and storing graph definitions. Typically /User/Configs/RTG_Defs/Graphs.
list -The NameList from which to choose variables to plot. Also determines the owning application.

RTGREAD **Read plot definitions for an RTG (or list)**
Initial: RTG *manager*, <PATH *source* | CharArray *name*]
-or-
Initial: RTGArray *list*, <PATH *source* | CharArray *name*]
Final: Long 1 (read) or 0 (not read)

With a path, plot definitions are read (XML) from the path.

With a Character array, the string is used for prompting (StdFileDialog) for a file to read.

RTGSCROLLX   **Scrolls x axis (Strip charts only)**
Initial: Num x, RTG manager
Final: -

RTGSCROLLY   **Scrolls y axis**
Initial: Num y, RTG manager
Final: -

Does nothing for either type of chart.

RTGSELECT   **Select a chart in a RTG manager**
Initial: Num *code*, RTG *manager*
Final: -

Selected chart is highlighted by a box.

code = 0: selection off
code = +1: select next one to the right
code = -1: select next one to the left

RTGSTART   **Draw all axes and plot all data in an RTG (or list of RTGs)**
Initial: RTG *manager*
      -or-
Initial: RTGArray *list*
Final: -

RTGTIME   **Change the time range (viewing) for viewed strip charts**
Initial: Num *secs*, RTG *manager*
Final: -

Zooms in and out (strip chart only).

RTGVARS   **Update variable addresses**
Initial: RTG *manager*
      -or-
Initial: RTGArray *list*
Final: -

Updates variables addresses.

RTGWRITE   **Write plot definitions disk**
Initial: RTG *manager*, <PATH *dest* | CharArray *name*]
      -or-
Initial: RTGArray *list*, <PATH *dest* | CharArray *name*]
Final: Long 1 (written) or 0 (not written)

With a path, plot definitions are written (XML) to the path.

With a Character array, the string is used for prompting (StdFileDialog) for a file to write to.

RUN   **Launch an LPL application**
   Initial: Path *sourceCode*
     -or-
   Initial: CArray *fileName*
   Final: -

        Related Keywords: STKSHARE, COMPILE, LINK

SCRLOCK   **Enable/disable print scrolling in a text window**
   Initial: Num *onOff* (0=off, 1=on)
   Final: -

        Scrolling makes the window scroll vertically when filled. No scrolling causes the cursor to wrap back to the upper left corner of the window when it fills, and subsequent printing overwrites the contents character by character. If scrolling is off, cursor goes to top of window.
        Related Keywords: MOVETEXT

SEARCH   **Search a path for a target string**
   Initial: Path *text*, Text *target*, [Num *flag*]
   Final: Long *return* (1 if found, 0 if not)
        flag given in Table 24-38. The path pointers are changed only if *return* is not 0.

Table 24-38. SEARCH Options

flag	Result (if target found)
0	Don't move pointers.
1	Move empty pointer to start of match
2	Move empty pointer to after match
3	Move fill pointer to start of match
4	Move fill pointer to after match

SECS2TD   **Convert seconds since base time to time and date**
   Initial: Num *secsSinceBase*
   Final: Num *secs,* Num *min*, Num *hour,* Num *day,* Num *month*, Num *year*
        Convert seconds since base time to the time and date (numerical)
        Related Keywords: TD2SECS, TIME, DATE, CTIME

SETATTR   **Set text attribute for subsequent printing**
   Initial: Num *newValue*
   Final: -
        Attributes (value of *newValue*) are given in Table 24-39.

Related Keywords: GETWINDOW, WINDOW

Table 24-39. Text attributes

Attribute	Description
0	None
1	Inverse video
2	Blinking
3	Inverse video and blinking

SETCURSOR   **Set the cursor type**
Initial: Num *newValue* (0=none, 1=underline, 2=full height)
Final: -

      Related Keywords: GETWINDOW.

SETDFCIN   **Sets default filters (convertin) for files.**
Initial: Text *string*
Final: -

      See **Path Filters** on page 23-43.
      Related Keywords: SETDFCOUT, GETDFCIN, GETDFCOUT, RESETDFC.

SETDFCOUT   **Sets default filters (convertout) for files.**
Initial: Text *string*
Final: -

      See **Path Filters** on page 23-43.
      Related Keywords:SETDFCIN, GETDFCIN, GETDFCOUT, RESETDFC.

SETEMPTY   **Sets the empty pointer for a path**
Initial: Num *offsetBytes*, Num *Ref* (from: 0=start, 1=current, 2=end), Addr *Path*
Final: -

      Related Keywords:PATHSTAT, SETFILL

SETFILL   **Set the fill pointer for a path**
Initial: Num *offsetBytes*, Num *Ref* (from: 0=start, 1=current, 2=end), Addr *Path*
Final: -

      Related Keywords: PATHSTAT, SETEMPTY

SETID   **Change the owner of on allocated item or A/D structure**
Initial: Num newID, Addr object
Final: -

      Related Keywords: MAKE, MAKE4, SYSID

SETMODNUM   **Set the module number for an application**
Initial: Num value, Num sysID
Final: -

> The next module to be linked will have number 1 plus this value. If *sysID* is <=0, the current application is assumed.
> Related Keywords: GETMODNUM

SETREADY   **Sets the ready value of an array**
Initial: Num *newVal*, Array *arr*
          -or-
Initial: Array *arr*, Num *newVal*
Final: -

> Setting the ready value to 0 effectively empties the array. SETREADY never alters the content of an array - it only changes the ready value in the array's header.
> Related Keywords: READY, SIZE.

SETTARGET   **Sets the search target for the system editor**
Initial: Text *target*
Final: -

> Related Keywords: GETTARGET.

SETTDS   **Sets the system clock**
Initial: Num *secsFromBase*
Final: -

> Related Keywords: GETTDS, TD2SECS.

SHOW   **Show n items on the stack**
Initial: Num *numItems*, [Path *dest* (default = LCD)]
Final: -

> SHOW does not affect the stack (unless pushing *numItems* and *dest* onto the stack to do SHOW causes a stack overrun).

SHOWSOFT   **Cause the function key labels (if any) to be displayed**
Initial: [Softkeys *s*]
Final: -

> Displays the specified softkeys, or the currently active ones.
> Related Keywords: HIDESOFT

SIN   **Computes sine**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
> Related Keywords: DEG, RAD.

SIZE   **Returns maximum size of an array**
Initial: Array *arr*
Final: Long *size*
> Related Keywords: READY

SLEEP   **Puts main task to sleep for specified time**
Initial: Num *secs*
Final:

           Puts main task to sleep for *secs* seconds.

SOFTGETM   **Returns the current function key menu level**
Initial: [Softkeys *s*]
Final: LONG *level* (0=1st, -1 = none defined)
           Related Keywords: SHOWSOFT, SOFTSETM

SOFTISACTIVE   **Is a Softkeys currently active**
Initial: [Softkeys s]
Final: LONG 1 or 0
           Returns 1 if *s* is the currently active Softkey pointer.
           Related Keywords:NEWSOFT, USESOFT

SOFTSETM   **Sets the function key menu level**
Initial: LONG *newLevel* (0=1st), [Softkeys *s*]
Final: -
           Does NOT redisplay key labels.
           Related Keywords: SOFTGETM, SHOWSOFT

SOFTWIDE   **Returns the number of function keys that can be displayed**
Initial: -
Final: LONG *nKeys* (5 for the LI-6400)

SQRT   **Square root**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
           Related Keywords: MATHERR

STADD   **Add a variable to the StatTracker list**
Initial: Num *period*, Num *id*, STTR *owner*
     -or-
Initial: Num *period*, Addr *address*, STTR *owner*
Final: Long (index or -1 on failure)
             id is the id of the variable to be added.
             period is the time period (secs) over which statistics are kept for that variable.
             index is the index (0 based) of the variable in the StatTracker list.

STDEDIT   **Invokes the system editor using the current text window**
Initial: [Text *filename*,] Text *text*
Final: -
           If filename is present, pressing escape will bring up the exit menu.
           Related Keywords:STDMENU, STDLINE

STDLINE   **Edits a line of text**

Initial: Text *line*
Final: LONG *result* (1=**enter** pressed, 0=**escape** pressed)

The window is borderless and lies between the current cursor position and the right edge of the active text window. If the text object is too large to fit in the window, the text scrolls horizontally in the window.

NOTE: The text object changes regardless of whether or not the user exits by pressing **escape** or **enter**. If you wish **escape** to leave the object unchanged, then use STDLINE on a copy of the object.
Related Keywords: STDEDIT.

STDMENU   **Invoke the standard menu interface**

Initial: [Text *bannerLabel*], Num *flag*, [Text *exit_nonascii*], Text *exit_ascii*, Text *menuItems*
Final: LONG *exitKey*

Does Standard Menu. If bits 3 or 4 of *flag* (Table 24-40 on page 24-88) are set, then *bannerLabel* is expected on the stack. A label banner will scroll left and right as the text window scrolls left and right. *exit_nonascii* is a list (int array) taken in highbyte - lowbyte pairs of nonascii keycodes that will cause termination. For example, to specify **F1**, **F2**, and **Shift**+**F2**, the list would be :INT {0x6000 0x6100 0x6101 }. *exit_ascii* is a list (char or int array) of ascii keys that will cause termination. Note that **escape** (0x1b) always terminates, whether it is in the list or not. Note: You can combine exit_nonascii and exit_ascii into one list; you don't need to use both.

Table 24-40. StdMenu *Flag*.

Bit	Value	Description
0	0x01	Highlighted menu bar
1	0x02	Put cursor to start of current line at exit
2	0x04	Use currently defined softkey labels
3	0x08	Fixed banner (e.g. a file name)
4	0x10	Label banner (e.g. column labels).
5	0x20	Show position marker on banner
7	0x80	Show byte marker on banner

Related Keywords:STDEDIT, EDOPEN

STFREE   **Free a StatTracker**

Initial: STTR *list*
Final: -

Related Keywords:STNEW

STFLUSH   **Flush the StatTracker**

Initial: NL *nameList*, STTR *list*

          -or-

Initial: Num 0, STTR *list*

Final: -

STGET   **Retrieve a value from a StatTracker**

Initial: Num *code*, Num *index*, STTR *list*

Final: (depends on code)

Values of code shown in Table 24-41.

Table 24-41. Codes for STGET

code	Value	code	Value	Returns
0x00	The ID			Long
0x01	Period (secs)			
0x02	Sample count			
0x03	Max size	0x23	Check Std Dev?	
0x04	Stable? (0 or 1)	0x24	Check %CV	
0x05	address	0x25	Check Slope?	
0x10	Mean			Double
0x11	Minimum			
0x12	Maximum			
0x13	Std dev	0x33	Std Dev Limit	
0x14	Std error	0x34	Std %CV Limit	
0x15	Slope	0x35	Slope Limit	

For a discussion of the STK_ keywords, see **Stack Control** on page 23-2.

STKCHECK   **Enable/disable stack overflow error reporting**

Initial: Num *onOff* (0=off, 1=on)

Final: -

A stack overflow occurs when too many items are pushed onto the stack. If reported, it is a non-fatal error.

Related Keywords: STKSIZE, STKRESIZE

STKREADY   **How many items are on the stack**
Initial: -
Final: LONG *elements*
                    Related Keywords: SHOW, ADR?

STKRESIZE   **Change the stack size**
Initial: Num *newSize* (number of items)
Final: -
                    This also flushes the stack.
                    Related Keywords: STKSIZE

STKSHARE   **Enable/disable stack sharing**
Initial: Num *yesNo* (0=no sharing, non-zero=stack sharing)
Final: -
                    Stack sharing applies to subsequent child applications launched by the current application.
                    Related Keywords: RUN

STKSIZE   **How many items can the stack hold?**
Initial: -
Final: LONG *numElements*
                    Related Keywords: STKREADY, STKRESIZE

STNEW   **Create a StatTracker**
Initial: NameList *variables*, Num *sysId*
Final: STTR *ptr*
                    sysId is the id of the owning application, or 0 for the current one.
                    variables is the NameList of potential variables that might be used.

STNUMSTABLE   **Returns the number of a StatTrackers variables that are stable**
Initial: STTR *list*
Final: Long *count*

STREAD   **Read a StatTracker definition from a file (*Deprecated in 6.1*)**
Initial: CharArray *fileName*, STTR *list*
    -or-
Initial: Path *source*, STTR *list*
Final: Long 1 (ok) or 0 (aborted)
                    In the first variant, the user is prompted to pick a source file, with *fileName* the default name. In the second variant, *source* contains the XML definition, and the StatTracker directly reads it.

STRESET   **Force a StatTracker to recheck addresses, and clear data buffers**
Initial: Num *freq*, STTR *list*
Final: -

> Each variable in the list gets its address updated, in chase there has been a change. *freq* is the expected frequency (1/sec) with which STUPDATE will be called. This information is used, with the period associated with each variable, to make each variable's buffer.

STSET   **Set a value in a StatTracker**
Initial: Num *value*, Num *code*, Num *index*, STTR *list*
Final: -

> Meanings of code are in Table 24-42.

Table 24-42. Codes for STSET

code	Meaning
0x01	Period (secs)
0x23	Check Std Dev? (1 or 0)
0x24	Check Std Error? (1 or 0)
0x25	Check Slope? (1 or 0)
0x33	Std Dev Limit
0x34	Std Error Limit
0x35	Slope Limit

STSIZE   **Returns the number of items in a StatTracker**
Initial: STTR *list*
Final: Long *count*

STUPDATE   **Add data to a StatTracker, and update all statistics**
Initial: Num *time*, STTR *list*
Final: -

> STUPDATE causes the StatTracker to go get the present value of all its variables, and add a data point for each.
> *time* is the relative time value (units of your choice) to be used, for example, secs since power on. This drives the slope units (per minute, per second, etc.).

STRIP   **Strip token names and cross reference information from the current application**
Initial: -
Final: -

> Note that this recovers space, but destines subsequent COMPILE and KBDEXEC commands

to failure if they contain any references to objects in this application.

STWRITE   **Store a StatTracker definition (*Deprecated in 6.1*)**
Initial: CharArray *fileName*, STTR *list*
      -or-
Initial: Path *dest*, STTR *list*
Final: Long 1 (ok) or 0 (aborted)

> In the first variant, the user is prompted to pick a destination file, with *fileName* the default name. In the second variant, *dest* is the path that will receive the XML definition.

SUBSET   **Create an array subset**
Initial: Array *arr*, Num *first*, Num *last*
Final: Array *tempHdr*

> Creates an array header for a subset of an existing array. This is convenient for limiting searches, or confining transformations to a region of an array whose lower bound is not 1. (When the lower bound is 1, you can accomplish the same thing by setting the ready value (SETREADY) temporarily). IMPORTANT: *tempHdr* is created on the local variable stack, so becomes invalid once the current function is exited.

SUM   **Returns the sum of numeric objects in an array**
Initial: Array *a*
Final: LONG or DOUBLE *sum*

> An alternative is

> ```
> 0 array +
> ```

SWAP   **Exchange the top two items on the stack**
Initial: ..., Obj *a*, Obj *b*
Final: ..., Obj *b*, Obj *a*

> Related Keywords: DUP, DROP, ROT

SYSID   **Returns the system ID number for the current application**
Initial: -
Final: LONG *sysID*

> Related Keywords: SETID

SYSTEM   **Call the host operating system**
Initial: [Text *command*]
Final: Long 1 or 0

> Send the string to the shell of the host operating system (standard c function system (const char *)).
> If *command* is not present, returns nonzero if a command processor is present, and zero otherwise. The return value with *command* is implementation dependent. It generally returns zero if the command was successful, and non-zero otherwise.

TAN   **Tangent**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
                        Related Keywords: DEG, RAD.

TD2SECS   **Convert time and date to seconds since the system base time**
Initial: Num *secs*, Num *min*, Num *hour*, Num *day*, Num *month*, Num *year*
Final: LONG *tdSecs*
                        Related Keywords: SECS2TD, GETTDS, TIME, DATE, CTIME

THEN   **Required termination keyword for IF**
Initial: -
Final: -
                        Related Keywords: IF, ELSE

TIDLE   **Wait for interrupts for a certain period of time**
Initial: Num *seconds*
Final: -
                        Related Keywords: IDLE

TIME   **Convert seconds since base time to time of day**
Initial: Num *tdSecs*
Final: LONG *secs* (0..59),   LONG *mins* (0..59),   LONG *hrs* (0..23)
                        Related Keywords: DATE, GETTDS, CTIME, SECS2TD, TD2SECS.

TYPE   **Returns the type of the object on the stack**
Initial: Obj *a*
Final: Obj *a*, LONG *type*

UNGETCH   **Put a character back into a path**
Initial: INT *charCode*, Addr *path*
Final: LONG *result* (*charCode* or -1 if path is full)
                        Related Keywords: GETCH, PUTCH

UNGETKEY   **Put a back into the keyboard queue**
Initial: INT *keyCode*
Final: LONG *result* (*keyCode* or -1 if queue is full)
                        Related Keywords: GETKEY, NEXTKEY

UNLINK   **Remove one or more modules from an application**
Initial: Num *all*, Num *moduleNumber*, Num *sysID*
        -or-
Initial: NUM *all*, Text *moduleName*, Num *sysID*
Final: Long *error* (0=ok, 1=failed)
                        The module to be removed (the target) can be referenced by name or number. If *all* is
                        non-zero, all modules from the target to the end are removed. If *all* is 0, then only the
                        target module is removed.

Related Keywords:LINK, ISLINK

UPC   **Uppercase function**
Initial / Final: (See **Single Value Transforms** on page 24-14.)
                    Related Keywords:LWC

USES   **List the cross reference information for an object**
Initial: Addr *path*, Num 1, Text *name* [, Num *sysID*]
    -or-
Initial: LArray *longArray*, IArray *intArray*, Num 2, Text *name* [, Num *sysID*]
Final: -

                The symbol *name* is looked up in the symbol table, and cross reference information (if
                any) is written to either a path (in which case names are written), or to numeric arrays
                (in which case addresses and types are written). The cross reference information
                comes from LINK, using control code 2 or 3.
                Related Keywords: LINK, XREF, PTRTOLL, FINDTOKEN

USESOFT   **List the cross reference information for an object**
Initial: [Softkeys *s*]
Final: -

                Switch to a new Softkeys, or remove softkeys (if not there).
                Related Keywords:NEWSOFT

VAL   **Get the numeric value of an object**
Initial: NAddr *a*
Final: LONG or DOUBLE *value*
    -or-
Initial: PAddr *p*
Final: Addr *final*
                Related Keywords: PVAL.

VREFGET   **Retrieve reference voltages**
Initial: DoubleArray *vals*
Final:

                vals should have room for at least 7 values. These A t

Table 24-43. Values for VREFGET and VREFSET

Item	Description
1	5 Volt reference value
2	12 bit D/A 5V actual value
3	12 bit D/A -5V actual value
4	8 bit unipolar D/A 5V actual value

Table 24-43. Values for VREFGET and VREFSET

Item	Description
5	8 bit unipolar D/A 0V actual value
6	8 bit bipolar D/A 5V actual value
7	8 bit bipolar D/A -5V actual value

VREFSET **Set reference voltages**
Initial: DoubleArray *vals*
Final: Long 1 (ok) or 0 (problem)

The first 7 values of *val* are used, and correspond to Table 24-43 on page 24-94.

WINDOW **Open a text window on the display**
Initial: Rect *area*, Num *border*, [Num *pos*, Text *lab1*, [Num *pos*, Text *lab2*]]
Final: -

border is 0 (none), 1 (single line), or 2 (double line). Label positions are shown in Figure 24-6.
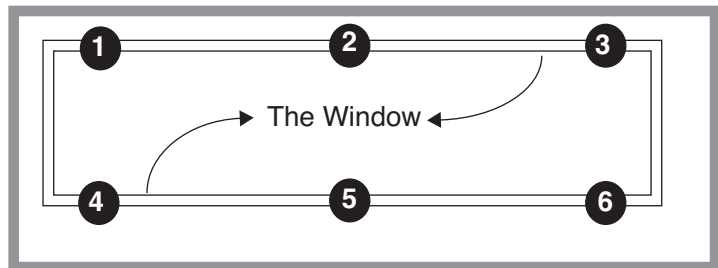
Figure 24-6. Window label position codes

The right border and/or the bottom border can be suppressed by specifying the right (or bottom) boundary to be any value larger than the DISPWIDTH or DISPHEIGHT value. Related Keywords: CLEAR, SCRLOCK, CLREOL

XFER   **Copy from one path to another**
Initial: Addr *fromPath*, Addr *toPath*, [Num *wait* | Fct *onEnd*]
Final: -

Table 24-44. XFER Combinations

		To		
	Keyboard	**Display**	**CommPort**	**File, Buffer, or Queue**
Keyboard	Error	OK	Overlapped	Overlapped
Display	Error	Error	Overlapped	Overlapped
Comm Port	Overlapped	Overlapped	Error	Overlapped
File, Buffer, or Queue	OK	OK	Overlapped	OK

(From)

For a discussion of the XL_ keywords, see **Excel Tools** on page 23-89.

XLADDCOL   **Add a column**
Initial: Text *name*, Addr *source*, XL *XLBuilder*
Final: -

> Add a column for subsequent use with XLADDOBS. *Source* is the address of the LPL variable that will be used for the value (or equation). They correspond to items in OPEN's log list that are not Header items (use XLADDCONTROL for those). *Name* is the label to be used. After adding all the desired columns, do XLPREP, and the columns will be labelled as "in" or "out", depending on whether the column was found to be an input or to be computed.
> Related Keywords:XLADDCONTROL, XLPREP, XLADDOBS.

XLADDCONTROL   **Add a control**
Initial: Num *newLineFlag*, Text *label*, Addr *source*, XL *XLBuilder*
Final: -

> Add an item to the control line. This corresponds to items in OPEN's log list that are declared as "Header" items. If *newLineFlag* is 1, put this item on a new line. If 0, keep on the current line.

XLADDITEMS   **Add items**

Initial: […,] Addr *source*, Num *count*, XL *XLBuilder*

Final: -

> Add one or more items, starting in the first column of the next row. *Count* determines how many items are to be popped from the stack and added. This is for adding header information or comments.

XLADDOBS   **Add an observation**

Initial: XL *XLBuilder*

Final: -

> Adds an observation (row) to the .xls file. The columns for the observation come from those items that had been set with XLADDCOL.
> Related Keywords:XLADDCOL.

XLCLOSE   **Close the file**

Initial: XL *XLBuilder*

Final: -

> Close the active .xls file being operated on by *XLBuilder*.
> Related Keywords:XLOPEN, XLREOPEN.

XLOPEN   **Close the file**

Initial: Num *appendFlag*, Text *name*, XL *XLBuilder*

Final: 0 (ok)

-or-

Final: 1 (fail)

> Create a .xls file named *name*. The .xls will be appended automatically, so *name* should not include it. If *appendFlag* is non-zero, the name will also have a number appended to it. For example, if *name* is /User/Data, and *appendFlag* is not 0, then the file name would be **/User/Data1.xls**, (or */User/Data2.xls*, if the first existed, etc.).
> Related Keywords:XLCLOSE, XLREOPEN.

XLPREP   **Prep the file**

Initial: Fct *faddr*, XL *XLBuilder*

-or--

Initial: 1 or 0

Final: -

> Once the XLADDCOL items are added, call XLPREP to set them up for subsequent XLADDOBS calls. The address of the function passed in (*faddr*) should be the function that computes the outputs in the columns.
> Related Keywords:XLADDOBS, XLADDCOL.

XLREOPEN   **Reopen the file**

Initial: XL *XLBuilder*

Final: 0 (ok) or 1 (fail)

> If the .xls file is already open (XLOPEN), does nothing. If it is closed, will reopen it. The strategy OPEN takes is to keep the .xls log file closed except when actually writing to it, to allow external connections to the console to open the file with Excel during log-

ging. The external Excel connection should not try to save the file, however. Any subsequent writes to the file (by OPEN) will not be seen by Excel without re-reading it.

For a discussion of the XM_ keywords, see **XML Support** on page 23-82.

XMCAPTURE   **Capture a node and all its subnodes**
Initial: Text *nodeMap* XML *xml*
Final: o (ok) or 1 (error)

XMCHANGED   **Has a node or one of its subnodes changed?**
Initial: Text *nodeMap* XML *xml*
Final: 1 (yes) or 0 (no) or -1 (error)

XMCLEAR   **Remove all nodes**
Initial: XML *xml*
Final: -

XMEDITABLE   **Is a node editable?**
Initial: Text *nodeMap* XML *xml*
Final: 1 (yes) or 0 (no) or -1 (error)

XMEXEC   **Execute some text input.**
Initial: [Text *dest*,] Text *source*, XML *xml*
          -or-
Initial: [Text *dest*,] Text *source*, STRUCT *s*
                    Permissible fields between start and end tags in source:
                    • values (more than one for an array)
                    • quoted string for char arrays
                    • ? - returns value - one long string
                    • ?n - bit 0 of n: format the return (tabs and new lines)
                         bit 1 of n - return changed nodes only

                    Examples: LCD "<abc>123</abc>" xm XMEXEC - set the value of <abc>
                    LCD "<abc><def>?</def></abc>" xm XMEXEC - return value of <abc><def>

XMEXECF   **Exec, source from a file**
   Initial: [Text *dest*,] Text *filename*, XML *xml*
        -or-
   Initial: [Text *dest*,] Text *filename*, STRUCT *s*
                        Same as XMEXEC, except the source is a file name.

XMINSERT   **Insert a node into a tree**
   Initial: [Text *dest*,] Text *source*, Num *sysID*, Text *nodeMap,* XML *xm*
   Final: 1 (error) or 0 (ok)

XMFIND   **Is the node present?**
   Initial: Text *nodeMap*, XML *xm*
   Final: 1 (yes) or 0 (no)

XMGETVAL   **Returns the value of a node**
   Initial: Text *dest*, Text *nodeMap*, XML *xm*
   Final: 1 (error) or 0 (ok)

XMLOAD   **Define an XML structure**
   Initial: [Text *dest*,] Text *source*, Num *sysID*, XML *xm*
   Final: 1 (error) or 0 (ok)

XMLOADF   **Define from a file**
   Initial: [Text *dest*,] Text *fileName*, Num *sysID*, XML *xm*
   Final: 1 (error) or 0 (ok)

XMNODES   **Return subnode count**
   Initial: Text *dest*, Text *nodeMap*, XML *xm*
   Final: Long count
        -or-
   Final: -1 (error)

XMREMOVE   **Remove a node**
   Initial: Text *nodeMap*, XML *xm*
   Final: 1 (error) or 0 (ok)

XMREVERT   **Revert node this node and all its subnodes to captured values.**
   Initial: Text *nodeMap*, XML *xm*
   Final: 1 (error) or 0 (ok)

XMSETCVAL   **Set the captured value of a node directly**
Initial: Text *value*, Text *nodeMap*, XML *xm*
Final: 1 (error) or 0 (ok)
                    Value can be some combination of these groups

XMSETVAL   **Set the value of a node**
Initial: Text *value*, Text *nodeMap*, XML *xm*
Final: 1 (error) or 0 (ok)

XMSETATTR   **Set the attribute of a node**
Initial: Text *value*, Text *nodeMap*, XML *xm*
Final: 1 (error) or 0 (ok)
                    Allowable characters in value are shown in **Codes for setting node attributes** on page 24-100.

Table 24-45. Codes for setting node attributes

Node Attribute	Code	Meaning
Read only	r	Run time[a] and Design time[b]
	R	Run time only
	D	Design time only
	d	Never
Hidden	h	Node is Hidden
	i	Node is Visible
	H	Subnodes are hidden
	I	Subnodes are visible (if node is too)
Show as Editable	e	Show as non-editable
	E	Show as Editable

a.Run time is during an XMEXEC, for example.
b.Design time is during an XMLOAD, for example.

XMSETSTATE   **Set the state of a node(s)**
Initial: Num *n2*, Num *n1*, Text *nodeMap*, XML *xm*
Final: 0 (ok) or (-1) error
                    n1 - 1 or 0 for open or closed.
                    n2 - bit 0 - also do descendents
                        bit 1 - also do parents

XMVIEW   **Display a structure**
    Initial: Text *label*, Num *flags*, XML *xm*
    Final: -

The value of *flags* is as follows:

Table 24-46. Codes for setting node attributes

Bit	Meaning
0	0 - No Menu Bar 1 - Menu Bar
1	0: Normal 1: Mark changed nodes with a '*'
2	0: Fct keys off normally 1: Fct keys always visible

XREF   **Generate a cross reference for an application**
    Initial: Text *filename*, Path *dest*

AutoProgramming Reference

Making them do what you what them to do

25

AutoProgramming Reference

There are three levels of dealing with AutoPrograms:

1 **Use the default programs**
OPEN provides some "ready to use".

2 **Use the AutoProgram Builder to make your own**
The AutoProgram Builder was introduced with OPEN 3.2, and is described on page 9-47.

3 **Program your own**
This chapter provides the background necessary to write your own from scratch, or to edit an existing program.

Autoprogram Format

AutoPrograms are small LPL application programs designed to run on top[1] of OPEN. While there is no limit to the scope of what an AutoProgram can be made to do, typically these programs log data in some sort of automatic fashion, while perhaps maintaining control over one or more conditions in the leaf chamber.

Let's look at a simple but fully functional LPL program that can serve as an Autoprogram. This task is very simple: log five observations spaced one minute apart to the log file.

---

[1] That is, they will not run successfully unless OPEN is also running.

```
:FCT main
{
  LPPrep
  5 NLOOP
     60 LPMeasure
     lpAbort BREAKIF
     LPLog
  ENDLOOP
  LPCleanup
}
```

Figure 25-1. A simple AutoProgram.

We start with one of several functions defined by OPEN for use in AutoPrograms:

LPPrep

This is required for any AutoProgram which makes use of the commands *LPMeasure* or *LPMeasureTilStable* is used. (If you don't use the *LPPrep* function prior to *LPMeasure* or *LPMeasureTilStable*, no one will slap your hand, but you might crash the system.)

Repetitive operations can go within LPL's loop structure, the *NLOOP...ENDLOOP* pair of keywords. The value on the stack when the *NLOOP* is encountered specifies the number of times through the loop. Within this structure, the 60 second delay is done by

60 LPMeasure

LPMeasure creates a New Measurements mode-like setting for a fixed amount of time, and when it is over, the global variable *lpAbort* tells us if the user wishes to abort the program. (This happens when the user presses **escape** during *LPMeasure*. This brings up the AutoProgram exit dialog box (Figure 25-2 on page 25-16). If the user presses **A**, *lpAbort* gets set to 1.) To abort (if *lpAbort* is non-zero), we just jump out of the loop.

lpAbort BREAKIF

Data recording is done by

LPLog

which causes a record to be written to the open log file (or RAM file or comm port). When the AutoProgram is done, some internal housekeeping details are

taken care of by

```
LPCleanup
```

which must be used if *LPPrep* was used.

Some AutoProgram Listings

"AutoLog2"

"AutoLog2" is described on
page 9-41.

```
    Auto logging

    112111 – For version 6.2
*/

:INCLUDE "/Sys/Lib/APTools"
:INCLUDE "/Sys/Lib/MatchIF"

:FLOAT logInterval 5
        matchInterval 0
        minWaitTime 0
        maxWaitTime 0
        duration_mins 10
        duration_secs 0

:INT quitAfter 0
    added 0
        method 0
        num_repeats 0

:LONG nextLog_ms 0

:PTR user[]
{
    :PTR { "settings" settingsxml "" "settings" }
    :PTR { "stable" stabilityxml "" "stability" }
}

:FCT main
{
        "Settings loops method match" settingsxml InstallMatchIf

        user APPrompts IF RETURN THEN

        APPrep IF APCleanup RETURN THEN

        duration_mins 60 * &duration_secs =
        &logInterval 0.5 MAX DROP
```

User setup comes from two
xml structures:
"settingsxml" is defined in
this program, "stabilityxml"
is defined by OPEN.

Inserts the MatchIf setup
into settingsxml.
Let user interract with
setup.

If doing wait for stability.

```
            method IF
                duration_secs minWaittime / &num_repeats =
                duration_secs GETTDS + :LONG stop_secs
                num_repeats LPRegLoop NLOOP LPLoopStat
                    GETTDS stop_secs >= BREAKIF
                  minWaittime MaxWaitTime LPMeasureTilStable lpAbort
        BREAKIF

                    LPMatchIf
                    APLogAction
                ENDLOOP LPDeregLoop
            ELSE
```

If doing time-based waiting.

```
                duration_secs logInterval / &num_repeats =
                GETMS logInterval 1000 * + &nextLog_ms =
                &ReadingHook &HookPostComps =
                num_repeats 0 LPSetProgress
                duration_mins 60 * LPMeasure
                &Noop &HookPostComps =
            THEN
            APCleanup
}
```

```
ReadingHook
{
    lpPaused IF RETURN THEN
```

Executes every time new readings are available (timed-based waiting only).

```
    GETMS :LONG now
    now nextLog_ms >= IF

        LogOneObsNoComp

        &added 1 + DROP
        now logInterval 1000 * + &nextLog_ms =
        num_repeats added LPSetProgress
        added num_repeats >= IF HALT THEN
    THEN
}
```

```
GetSummary
{
```

Called from settingsXML.

```
    :CHAR &label[]
    DROP
    0 label SETREADY

    method IF
        duration_mins 60 * TIME maxWaitTime minWaitTime "Every
%d-%ds for %02d:%02d:%02d" label SPRINT
    ELSE
        duration_mins 60 * TIME logInterval "Every %gs for
%02d:%02d:%02d" label SPRINT
    THEN

    updateXML
}
```

Called from settingsXML to dynamically adjust the tree (hide some nodes, expose others, etc.)

```
UpdateXML
{
    method IF
        "h" "settings loops method logInterval" settingsXML
XMSETATTR DROP
        "i" "settings loops method match" settingsXML XMSETATTR
DROP
        "i" "settings loops method stableWait" settingsXML
XMSETATTR DROP
        "i" "stability" stabilityXML XMSETATTR DROP
    ELSE
        "i" "settings loops method logInterval" settingsXML
XMSETATTR DROP
        "h" "settings loops method match" settingsXML XMSETATTR
DROP
        "h" "settings loops method stableWait" settingsXML
XMSETATTR DROP
        "h" "stability" stabilityXML XMSETATTR DROP
    THEN
}

summaryWait
{
    :CHAR &label[]
    DROP
    0 label SETREADY

    maxWaitTime minWaitTime "%d to %d s" label SPRINT
}
```

This structure defines part of the setup configuration tree.

```
:XML
settingsXML
{
    <Settings get=GetSummary disp="Summary" delim=":">
      <loops disp="Duration (mins)" addr=duration_mins edit=-1 >
            <method disp="Wait for" addr=method
toggle="time/stability" oncic=Updatexml >
                <logInterval addr=logInterval disp="Log interval
(s)" edit=-1 />
                <stableWait get=summaryWait disp="Stability wait" >
                    <min disp="Minimum (secs)" addr=minWaitTime
edit=-1 />
                    <max disp="Maximum (secs)" addr=maxWaitTime
edit=-1 />
                </stableWait>
                <match />
            <log disp="Action" addr=apLogType get=APGetLogAction
edit=APEditLogAction attrf=APGetLogAttr />
            </method>
        </loops>
    </Settings>
}
```

The empty match node is a placeholder. The MatchIf info will be inserted here.

"LightCurve2"

"LightCurve2" is described on page 9-43.

```
/*

    Simple Light Curve, with stability checking
        rev 3 3/26/2011

    960822 - Tech Note 14 modification, wait time defaults
    970502 - remembers last time defaults
    980309 - delta-based matching
    020524 - version 5 stability
        110326 - version 6.2
*/

:INCLUDE "/Sys/Lib/APTools"
:INCLUDE "/Sys/Lib/MatchIF"

:INT minWaitTime 120
        maxWaitTime 200


:PTR user[]
{
    :PTR { "settings" settingsxml "" "settings" }
    autoprogStability
}
```

The setup structure. SettingsXML is defined below, and autoprogstability is defined by OPEN.

```
:FCT main
{
    /*Set program name
    */
    "settings light match" settingsxml InstallMatchIf
```

User interaction.

```
    user APPrompts IF RETURN THEN

    APPrep IF APCleanup RETURN THEN

    1 :INT i
```

The Light Loop.

```
    APLampCount LPRegLoop NLOOP LPLoopStat
        /* set the lamp -
        */
        i APSetLamp

        /* wait for stability
        */
        minWaitTime maxWaitTime LPMeasureTilStable
            lpAbort BREAKIF

        /* match and log
        */
        LPMatchIf
        APLogAction

        &i 1 + DROP
```

```
    ENDLOOP LPDeregLoop

    APCleanup
}

/* ---------------------------------------
    cosmetics for the front end
--------------------------------------- */

summaryLight
{
      :CHAR &label[]
      DROP
      0 label SETREADY

    APLampCount "%d SetPts for " label SPRINT
    APLampControlGetShort "%s" label SPRINT
}

summaryWait
{
    :CHAR &label[]
    DROP
    0 label SETREADY

    maxWaitTime minWaitTime "%d to %d s" label SPRINT
}
```

Defines part of the user setup tree.

```
:XML
settingsxml
{
<settings get=summaryLight disp="Summary" delim=":" >
    <lamp addr=apLampControlIndex disp="Lamp control"
get=APLampControlGet edit=APLampControlEdit />
    <light get=APShowLampValues edit=APEditLampValues
disp="SetPts" delim=":">
        <wait get=summaryWait disp="Stability wait" >
            <min disp="Minimum (secs)" addr=minWaitTime edit=-1 />
            <max disp="Maximum (secs)" addr=maxWaitTime edit=-1 />
        </wait>
        <match />
        <Log addr=apLogType get=APGetLogAction
edit=APEditLogAction />
    </light>
</settings>
}
```

"A-CiCurve2"

This program is described on page 9-39.

```
/*
    A-Ci Curve
        rev 3 3/26/2011
```

```
    960822 – Tech Note 14 modification
    970502 – remembers last time defaults
    980309 – delta-based matching
    020524 – version 5 stability
        110326 – version 6.2
*/

:FLOATco2Values[20] {400 300 200 100 50 400 400 600}

:INT minWaitTime 60
      maxWaitTime 300

:INCLUDE "/Sys/Lib/APTools"
:INCLUDE "/Sys/Lib/MatchIF"

:PTR user[]
{
    :PTR { "settings" settingsxml "" "settings" }
    autoprogStability
}

:FCT main
{
    mixerAvail NOT IF
        1 "Sorry.  Need a CO2 Mixer for this." MESSBOX DROP
        RETURN
    THEN

    "settings CO2 match" settingsxml InstallMatchIf

    user APPrompts IF RETURN THEN

    APPrep IF APCleanup RETURN THEN

    1 :INT i
    co2Values READY LPRegLoop NLOOP LPLoopStat
        /* Get current mixer settings
        */
        MixerGetTarget :INT type :FLOAT target

        /* set new settings
        */
        co2Values i PICK VAL MixerAutoProgSet

        /* If mixer was off, wait awhile for it
        */
        type 1 == IF
            mixerStabilizationWait LPAbort BREAKIF
        THEN

        /* wait for stability
        */
        minWaitTime maxWaitTime LPMeasureTilStable
```

The setup tree consits of two parts: settingsxml defined below, and autoprogStability, defined by OPEN.

User intereaction happens here.

The CO2 loop.

```
            lpAbort BREAKIF

        /* match and log
        */
        LPMatchIf
        APLogAction

        &i 1 + DROP
    ENDLOOP LPDeregLoop

    APCleanup
}

mixerStabilizationWait
{
    /* wait from 10 secs to 3 minutes for mixer to stabilize
    */
    ::CHAR old[] ""
    lpProgress old =
    "MixrWait" lpProgress =
    &IsMixerStable 10 180 LPMeasureTilStableFct
    old lpProgress =
}

/* ----------------------------------------
    cosmetics for the front end
   ---------------------------------------- */
summaryCO2
{
    :CHAR &label[]
    DROP
    0 label SETREADY

    co2Values READY :INT n
    n "%d" label SPRINT
    n 3 >= IF
        co2Values n PICK VAL co2Values 1 PICK VAL " (%d\x1a%d) of
" label SPRINT
    THEN
    MixerAPShort label SPRINT
}

summaryWait
{
    :CHAR &label[]
    DROP
    0 label SETREADY

    maxWaitTime minWaitTime "%d to %d s" label SPRINT
}

updateType
{
    mixerAPIndex 4 == IF "I" ELSE "H" THEN "settings mixer"
```

The structure of the setup tree.

```
settingsxml XMSETATTR DROP
}

:XML
settingsxml
{
    <settings get=summaryCO2 disp="Summary" delim=":" >
      <mixer disp="CO2 control" get=mixerAPGet addr=mixerAPIndex
edit=mixerAPEdit oncic=updatetype >
            <units disp="Setpoint units" addr=ppmForControlMode
toggle="mv/ppm" >1</units>
        </mixer>
        <CO2 addr=co2Values edit=-1 >
            <wait get=summaryWait disp="Stability wait" >
              <min disp="Minimum (secs)" addr=minWaitTime edit=-
1 />
                <max disp="Maximum (secs)" addr=maxWaitTime edit=-
1 />
            </wait>
            <match />
            <Log addr=apLogType get=APGetLogAction
edit=APEditLogAction />
        </CO2>
    </settings>
}
```

"TimedLamp"

This program is described on page 9-45.

```
/*

    Timed lamp control

    970502 - Bundled with OPEN 3.0
    980515 - Updated for 3.2
        110425 - Updated for 6.2

*/

:INCLUDE "/Sys/Lib/APTools"

:STRUCT TLSETTINGS
{
    :FLOAT seconds
    :FLOAT intensity
    :FLOAT frequency
    :CHAR  action[]
}

:TLSETTINGS values[] { }

:XML settingsXML
```

```
{
    <settings get=settingsSummary disp="Summary" delim=":" >
        <lamp addr=apLampControlIndex disp="Lamp control"
get=APLampControlGet edit=APLampControlEdit />
        <values addr=values edit=EditValues get=ActionSummary>
        </values>
    </settings>
}

:PTR user[]
{
    :PTR { "settings" settingsXML "" "settings" }
}

:LONG nextLog 0
:INT logInterval 0

:FCT main
{
    "" user LPPrompts3 IF RETURN THEN

    LPPrep

    1 :INT i
    1 :INT firstLamp

    values READY LPRegLoop NLOOP LPLoopStat
        values i PICK :TLSETTINGS x
        x.action READY IF
            x.action openSysID LCD 1 COMPILE NOT IF
                :PTR f
                &f CALL
                &f FREE
            THEN
        ELSE
            firstLamp IF
                x.intensity apLampControlIndex LampSetNewTarget
                0 &firstLamp =
            ELSE
                x.intensity LampSetTargetVal
            THEN
            x.frequency &logInterval =
            x.seconds logInterval - &nextLog =

            &CustomTic x.seconds LPMeasureFct lpAbort BREAKIF
        THEN
        &i 1 + DROP
    ENDLOOP LPDeregLoop
    LPCleanup
}

CustomTic
{
    :LONG remainingSecs
```

User interactions here.

Main loop

```
        logInterval IF
            remainingSecs nextLog <= IF
                LPLog
                remainingSecs logInterval - &nextLog =
            THEN
        THEN

        remainingSecs LPShowSimpleClock
    }

    ActionSummary
    {
        :CHAR &line[]
        DROP

        line 0 SETREADY
        values READY :INT n
        n "%d Steps" line SPRINT
        n 1 > IF
            values 1 PICK :TLSETTINGS x
            ", 1st= " line SPRINT
            x.action READY IF
                x.action "'%s'"
            ELSE
                x.frequency x.intensity x.seconds "(%g, %g, %g)"
            THEN
            line SPRINT
        THEN
    }
    SettingsSummary
    {
        :CHAR &line[]
        DROP

        line 0 SETREADY
        values READY :INT n
        n "%d Steps" line SPRINT
    }

    EditValues
    {
```

The dialog for entering a set of values (frequency, intensity, action).

```
        1000 OPEN_BUFF IF RETURN THEN
        :PTR buff
        1 :INT i
        values READY NLOOP
            values i PICK :TLSETTINGS item
            item.action READY IF
                item.action "\"%s\"\n" buff PRINT
            ELSE
                item.frequency item.intensity item.seconds "%g, %g,
%g\n" buff PRINT
            THEN
```

```
        &i 1 + DROP
    ENDLOOP

    1 STKSHARE
    "3/Line: Time(s), Intensity, logEvery(s)" buff
"/Sys/Lib/PathEditor" RUN :INT k
    0 STKSHARE

    k IF
        '\n' buff PUTCH
        values 0 SETREADY
        ::FLOAT threeFloats[] {0 0 0}
        ::CHAR stringItem[] ""
        ::TLSETTINGS thisLine { 0 0 0 }
        0 :FLOAT floatItem
        LOOP
            buff PRNEXTLINE :INT n
            n NOT BREAKIF
            n 1 == IF
                thisLine.action 0 SETREADY
                thisLine.Action 0 buff PRGET
            ELSE
                1 &i =
                3 NLOOP
                    stringItem 0 SETREADY
                    stringItem i 1 - buff PRGET  /* ith item ->
string */

                        &floatItem "%f" stringItem SENTER IF
                            floatItem threeFloats i PICK  =
                    THEN
                    &i 1 + DROP
                ENDLOOP
                thisLine.action 0 SETREADY
                threeFloats 1 PICK VAL &thisLine.seconds =
                threeFloats 2 PICK VAL &thisLine.intensity =
                threeFloats 3 PICK VAL &thisLine.frequency =
            THEN
            thisLine values APP
        ENDLOOP
    THEN
    buff CLOSE
}
```

Useful AutoProgram Commands

All functions starting with LP... are public LPL functions defined in the file /sys/open/open.lp. The descriptions that follow use the format described in **Definitions** on page 24-12.

Standard Commands

$NOFILE$ **Suppresses prompting for opening a file**

This is not exactly a command, but it this string appears anywhere in the file, such as in a remark at the top, the file prompting will be bypassed. This is useful for AutoPrograms that aren't meant to log in the normal fashion, but instead might do some other sorts of tasks.

LPCleanup   **Required**

Initial: -
Final: -

This should be the last thing before quitting if you did an LPPrep earlier.
Related Keywords: LPPrep

LPDeregLoop   **Deregister a loop**

Initial: -
Final: -

See LPRegLoop.

LPLog   **Logs an observation to the log file.**

Initial: -
Final: -

LPLogComment   **Logs a string to the log file as a comment.**

Initial: Text *remark*
Final: -

This function builds a quoted string made up of the current time (HH:MM:SS), followed by *remark*, and outputs it to the log destination.

LPLoopStat   **Register a loop**

Initial: -
Final: -

See LPRegLoop.

LPMatch   **Matches the IRGAs**

Initial: -
Final: -

This function toggles the match valve, waits for the water reference analyzer to stabilize, then waits up to 1 minute for the CO_2 sample analyzer to stabilize. If it does, the

sample water and CO_2 IRGAs are adjusted to read the reference IRGAs. Otherwise, a warning message is logged.

LPMeasure   **Enter New Measurements mode for fixed time:**
Initial: Num *seconds*
Final: -

Check the numeric variable *lpAbort* after this is through. If it is non-zero, the user triggered the next step.

LPMeasureTilStable   **Enter New Measurements with stability checking.**
Initial: Num *stability*, Num *minTime*, Num *maxTime*
Final: -

stability is the threshold value of total CV%, *minTime* is the minimum time to wait (s), and *maxTime* is the maximum time to wait (s). Check the numeric variable *lpAbort* after this is through. If it is non-zero, the user triggered the next step. These functions "look" just like New Measurements mode, except when you try to exit, or to close a log file, you are shown the AutoProgram exit dialog box (Figure 25-2).

```
<Prog Name> in Progress
A - abort program
T - trigger next step
<esc> - resume program
```
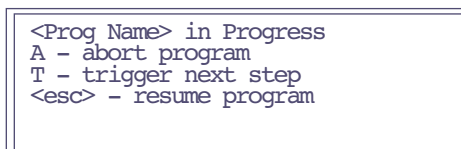
Figure 25-2. The Autoprogram exit dialog.

Pressing **A** will set the global variable *lpAbort* to 1; this should be checked in the AutoProgram immediately after calling LPMeasure or LPMeasureTilStable. Pressing **T** will also terminate LPMeasure or LPMeasureTilStable, but *lpAbort* will be set to 0.

LPPrep   **Required**
Initial: -
Final: -

This is required if LPMeasure or LPMeasureTilStable is going to be used.
Related Keywords:LPCleanup

LPPrompts   **Get user input for a group of items**
Initial: PArray *theList*
Final: Long *code* (1 if user aborted, 0 if ok)

Each item in the pointer array should have the following structure:

NArray *values* CArray *prompt*
Num *value* CArray *prompt*

An example is shown in Figure 25-3.

Related Keywords:LPPrompts2

```
:PTR myList[] {
  :PTR { lampVals "Desired lamp values\n" }
  :PTR { delayTime "Wait time (s)" }
}
```

Figure 25-3. Example pointer array suitable for LPPrompts *or* LPPrompt2.

LPPrompts2   **Get user input, using last time defaults**
    Initial: PArray *theList*, Text *fileName*
    Final: Long *code* (1 if user aborted, 0 if ok)
                    LPPrompts2 is like LPPrompts, with two additions:
                    Default responses are read from file *fileName*, if it exists. User responses are then
                    saved for next time in *fileName*, which is created if it doesn't exist.

                    The structure of theList can be different. In addition to the two elements allowed by LP-
                    Prompts, a third is allowed by LPPrompts2:

                    Addr *stattracker* Text *file* Text *prompt*
                    where *stattracker* is the address of a StatTracker (see STNEW). The standard one in use
                    in New Measurements mode is named StatTrack. The following sample illustrates:

```
/*
    Sample autoprogram: changing stabilities automatically
*/

:CHAR defaultFile[] "XXXDefault4"
StabFile1[80] "/User/Configs/StableDefs/CustomAP1"
StabFile2[80] "/User/Configs/StableDefs/CustomAP2"

:FLOAT
  wait1 20
  wait2 30
  wait3 60
  wait4 180
  values1[50] { 400 300 200 }

:PTR user[]
{
:PTR { wait1 "Min wait time (minutes):" }
:PTR { wait2 "Max wait time (minutes):" }
:PTR { StatTrack StabFile1 "First Stability" }
:PTR { values1 "Ref CO2 values (Êmol/mol):\n" }
:PTR { wait3 "Min wait time (secs):" }
:PTR { wait4 "Max wait time (secs):" }
:PTR { StatTrack StabFile2 "Second Stability" }
}

:FCT main
```

```
{
    CLEAR

    defaultFile user LPPrompts2 IF RETURN THEN

    LPPrep

    /* First stability
    */
    StabFile1 SetStabDef
    wait1 60 * wait2 60 * LPMeasureTilStable

    lpAbort NOT IF
        /* 2nd stability
        */
        StabFile2 SetStabDef
        1 :INT i1
        values1 READY LPRegLoop NLOOP LPLoopStat
            values1 i1 PICK VAL 2 MixerSetNewTarget
            wait3 wait4 LPMeasureTilStable lpAbort BREAKIF
            LPLog
            &i1 1 + DROP
        ENDLOOP LPDeregLoop
    THEN
    LPCleanup
}

SetStabDef
{
    /* in: filename */
    &ActiveSTR =
    StabilityReadFile
}
```

LPRegLoop   **Register a loop**
Initial: Num n
Final: -

(OPEN 3.2)This is an alternative to using LPSetProgress. Register a loop using LPRegLoop prior to starting the loop. Inside the loop, call LPLoopStatus to update the counter, and the display. When the loop is done, call LPDeregLoop to deregister the loop. These are the tools used by the AutoProgram Builder, and were designed to handle nested loops in a reasonable fashion.

LPSetName   **Defines the Autoprogram name**
Initial: Text *name*
Final: -

name will appear in the Autoprogram exit dialog.

LPSetProgress   **Sets the AutoProgram progress indicator:**
 Initial: Num *max*, Num *this*
 Final: -

     max is the number of steps to be done, and *this* is the value of the current step. These two values are built into a string "this/max" that is displayed as the *ProgPrgs* system variable (ID #-56).

UcnAskAll   **Prompt for all user constants**
 Initial:
 Final:

     Same as pressing **User Consts** (**f5** level 3) in New Measurements mode.

LPTicFct   **Specify a function to execute every second during the AutoProgram.**
 Initial: Fct name
 Final: -

     For an example of how this is used, see the TimedLamp AutoProgram.

Fan Control

     These functions affect the chamber mixing fan.

LPSetFanSpeedVal **Set fan speed (volts)**
 Initial: Num *volts* (0 [off] through 5 [full speed])
 Final: -

     Intermediate values (4.38) will work here, as well.

LPSetFanOSF   **Set fan speed (Off Slow Fast)**
 Initial: Num *selection* (0, 1 or 2 for off, slow, fast)
 Final: -

     2 is the highest speed, 1 uses the current definition of "slow" (default is 4 volts).

Flow Control

These functions determine the flow/humidity control.

FlowSetNewTarget   **Set new control mode and target value.**

Initial: Num *value*, Num *typeCode*

Final: -

The units of *value* depend on *typeCode* (Table 25-1).

Table 25-1. Flow control typeCodes.

typeCode	Meaning
1	Pump off
2	Constant flow (μmol s^{-1})
3	Constant sample (mmol mol^{-1})
4	Constant sample RH (%)
5	Constant VPD (kPa)

FlowGetTarget   **Get the current control mode and target value:**

Initial: -

Final: Double *value*, Long *typeCode*

Use this to determine the current set point and control type. *typeCode* values are given in Table 25-1.

FlowSetTargetVal   **Set new target value independent of control mode**

Initial: Num *value*

Final: -

CO_2 Control

These functions require the 6400-01 CO_2 mixer to be installed. They are defined in /sys/open/open.mxr.

MixerSetNewTarget   **Sets the target value and control mode.**
Initial: Num *value*, Num *typeCode*
Final: -

The units of *value* depend on *typeCode* (Table 25-2).

Table 25-2. CO_2 mixer control typeCodes.

typeCode	Meaning
1	Mixer off
2	Constant reference (μmol mol^{-1})
3	Constant sample (μmol mol^{-1})
4	Constant control signal (mV)
5	Constant Ci (ppm)

MixerGetTarget   **Get the current target value and control mode:**
Initial: -
Final: Double *value*, LONG *typeCode*

Use this function to get the current control type and target value. typeCodes are given in Table 25-2.

MixerSetTargetVal   **Sets the target value independent of control mode.**
Initial: Num *value*
Final: -

Sets a new target without changing the control type.

Temperature Control

These functions determine temperature control.

CoolSetNewTarget   **Set new control target and mode.**
Initial: Num *value*, Num *typeCode*
Final: -

The units of *value* depend on *typeCode* (Table 25-3).

Table 25-3. Temperature control typeCodes.

typeCode	Meaning
1	Coolers off
2	Constant block temperature (C)
3	Constant leaf temperature (C)

CoolGetTarget   **Get the current target value and mode:**
Initial: -
Final: Double *value*, Long *typeCode*
Use this to determine the current set point and control type (Table 25-3).

CoolSetTargetVal   **Set the cooling target (independent of control mode)**
Initial: value>
Final: -

LED Source Control

These functions expect the 6400-02 or -02B LED Source, 6400-18 RGB, or 6400-40 LCF to be installed. Commands specific to the LCF and RGB are given in **6400-18 RGB Light Source Control** on page 25-23 and **Leaf Chamber Fluorometer Control** on page 25-24.

LPSetLamp   **Sets the light source.**
Initial: Num *parVal* (in μmol m^{-2} s^{-1})
Final: -

LampSetNewTarget   **Sets the lamp control algorithm, and target value.**
Initial: Num *value*, Num *code*
Final: -

The units of *value* depend on *typeCode* (Table 25-4).

Table 25-4. Lamp control typeCodes

code	6400-02B	6400-18 RGB	6400-40 LCF
1	Lamp off	Lamp off	Lamp off
2	Constant PAR (μmol m^{-2} s^{-1}).	Constant PAR (μmol m^{-2} s^{-1}) using current color.	Constant PAR (μmol m^{-2} s^{-1}) using current blue setting.
3	Constant control signal (mV).	Sets % of max for red.	Sets red control signal (mV).
4	Tracks external quantum.	Tracks external quantum.	Tracks external quantum.

Note: This is a generic lamp control function. For functions that are specific to the 6400-18 and 6400-40 (for example, specify a quantum target AND the color), see the next sections.

LampGetTarget   **Gets the current lamp control id and target value**
Initial: -
Final: DOUBLE *value*, LONG *typeCode*

Us this to determine the current control method and target value. typeCodes are given in Table 25-4.

LampSetTargetVal   **Sets the current target value without changing control type.**
Initial: Num *value*
Final: -

Control type remains the same. Only the target changes.

6400-18 RGB Light Source Control

LPRGBSetPAR   **Sets the control for PAR mode, with the specified color**
Initial: Num *total_um* char[] *color*
Final: -

total_um is total PAR in (μmol m^{-2} s^{-1}), and the color is a string containing the color name, or its RGB proportions (see LPRGBSetColor for more details). Examples:

```
1500 "white" LPRGBSetPAR
800 "10 20 30" LPRGBSetPAR
```

LPRGBSetColor   **Sets the color for PAR or Tracking modes**
Initial: char[] *nameOrSpec*
Final: -

> The string *nameOrSpec* can be either a color name (that is currently defined in the color list (white, red, yellow, green, cyan, blue, magenta are the standards - the user can define more), or the actual red-green-blue proportions. This, the following are valid:
>
> ```
> "green" LPRGBSetColor
> "1 3 4.5" LPRGBSetColor  /* 1 part red, 3 green, 4.5 blue */
> "1 1 1" LPRGBSetColor /* this is white */
> ```
>
> If the RGB Source is currently doing constant quantum flux, or tracking an external sensor, then the color will take effect immediately. If the RGB Source is off, or in control signal mode, the change will not take place until constant quantum flux or external tracking is next begun.

LPRGBSetControl   **Specify the red, green, and blue control settings directly**
Initial: char[] *percentMaxVals*
  -or-
Initial: Num *red%*, Num *green%*, Num *blue%*
Final: -
  -or-

> The string *percentMaxVals* should contain three values separated by whitespace or commas, that specify what each color should be outputting in terms of percent of its maximum possible output. Or, these values can be numerical values on the stack. Examples:
>
> ```
> 10 20 30 LPRGBSetColor /* 10% red, 20% green, 30% blue */
> "10 20 30" LPRGBSetColor /* 10% red, 20% green, 30% blue */
> "100 100 100" LPRGBSetColor  /* everything on full */
> "0 0 0" LPRGBSetColor         /* everthing off */
> "0.001 0.001 0.001" LPRGBSetColor /* everthing real dim */
> ```

Leaf Chamber Fluorometer Control

See also the control functions listed in **Programming Commands** on page 27-86.

LPFlrQPBlue   **Sets total quantum target with the percentage of blue**
Initial: Char[] *twoValues*
  -or-
Initial: Num *total_um*, Num *blue%*
Final: -

> *twoVals* is a string with two floating point values: total target (μmol m^{-2} s^{-1}) and percent blue, separated by a space. The following are equivalent:
>
> ```
> 1500 20 LPFlrQPBlue
> ```

```
"1500 20" LPFlrQPBlue
```

LPFlrQFBlue   **Sets total quantum target and the quantum flux that is to be blue**
Initial: Char[] *twoValues*
    -or-
Initial: Num *blue_um,* Num *total*
Final: -

twoVals is a string with two floating point values: total target and blue, both in
(μmol m^{-2} s^{-1}), separated by a space. The following are equivalent:

```
1500 200 LPFlrQPBlue
"1500 200" LPFlrQPBlue
```

LPFlrPAR   **Sets total quantum target and how blue should be handled**
Initial: Char[] *threeValues*
    -or-
Initial: Num *total_um,* Num *vlue_val* NUM *1 or 0*
Final: -

threeVals is a string with three floating point values: total target (μmol m^{-2} s^{-1}), and
blue value, and a 1 or 0. 1 means the blue value should be interpreted as % of total,
and 0 means the blue value should be interpreted as (μmol m^{-2} s^{-1}). The following are
examples:

```
1500 200 0 LPFlrPAR          1500 μmol m⁻² s⁻¹ total with 200 of it blue.
100 50 1 LPFlrPAR            100 μmol m⁻² s⁻¹ total with 50% of it blue.
```

LPFlrControl   **Sets red and blue control signals**
Initial: Char[] *twoValues*
    -or-
Initial: Num *red_mv* Num *blue_mV*
Final: -

twoVals is a string with two floating point values: red and blue control signals, sepa-
rated by a space. The following are equivalent:

```
1000 5000 LPFlrControl
"1000 5000" LPFlrControl
```

LPFlrSetMeasInt   **Set measuring beam intensity (0-10)**
Initial: Num *value*
Final: -

Value can be any floating point value between 0 and 10.

LPFlrSetMeasMod   **Set measuring beam modulation rate**
   Initial: Int *index*
   Final: -

          Index can be 1..4, as shown in Table 22.

Table 20-22. Indices for setting modulation frequency.

Index	Modulation (kHz)
1	0.5
2	1
3	10
4	20

LPFlrSetMeasFilter   **Set measuring beam bandwidth**
   Initial: Int *index*
   Final: -

          Index can be 1..8, according to Table 23:

Table 20-23. Indices for setting bandwidth

Index	Bandwidth (Hz)	Index	Bandwidth (Hz)
1	0.5	5	20
2	1	6	50
3	5	7	100
4	10	8	200

LPFlrSetMeasGain   **Set gain on measuring beam**
   Initial: Int *index*
   Final: -

          Index can be 1..4:
          1 = 10
          2 = 20
          3 = 50
          4 = 100

LPFlrSetFlashInt   **Set target intensity (0-10) for next flash**
   Initial: Num *target*
   Final: -

          Target should be a floating point value between 0 and 10.

LPFlrSetFlashType **Set the flash type (RF or MPF) for next flash**
Initial: Num *code*
Final: -

        Code should be 0 for a rectangular flash (RF). Any other value will be a multiphase flash flash (MPF).

LPFlrSetFlashDur **Set the duration for the next rectangular flash**
Initial: Num *secs*
Final: -

LPFlrSetPhase1 **Set the phase one timing for the next multiphase flash**
Initial: Num *ms*
Final: -

LPFlrSetPhase2 **Set the phase two timing for the next multiphase flash**
Initial: Num *ms*
Final: -

LPFlrSetPhase3 **Set the phase three timing for the next multiphase flash**
Initial: Num *ms*
Final: -

LPFlrSetFlashMod **Set measuring beam modulation rate for the next flash**
Initial: Int *index*
Final: -

        Index can be 1 to 4, as shown in Table 22 on page 26.

LPFlrSetFlashFilter **Set measuring beam bandwidth for the next flash**
Initial: Int *index*
Final: -

        Index can be 1 to 8, according to Table 23 on page 26.

LPFlrSetDarkDur **Set the duration for the next dark pulse**
Initial: Num *secs*
Final: -

LPFlrSetFarRedInt **Set target intensity (0-10) for next time the far red LED is turned on**
Initial: Num *target*
Final: -

        Target should be a floating point value between 0 and 10.To actually turn the far red

on, do a dark pulse (e.g. DoFop) or directly turn it on with FarRed_on.

LPFlrSetDarkPre **Set the duration for the pre-time of the next dark pulse**
Initial: Num *secs*
Final: -

LPFlrSetDarkPost **Set the duration for the post-time of the next dark pulse**
Initial: Num *secs*
Final: -

LPFlrSetDarkMod **Set measuring beam modulation rate for the next flash**
Initial: Int *index*
Final: -

Index can be 1..4, as shown in Table 22 on page 26.

LPFlrSetDarkFilter **Set measuring beam bandwidth for the next flash**
Initial: Int *index*
Final: -

Index can be 1..8, according to Table 23 on page 26.

LCF Extras
The following functions are used in the factory-suppled autoprograms for the LCF. To use them in your own autoprogram, you'll have to include the following line in your autoprogram file.

```
:INCLUDE "/Sys/Lib/FlrAP"
```

LPFlrSetup **Prompts for standard fluorescence prompts**
Initial: -
Final: Long *Logic* (0=ok, 1=user pressed escape)

See **Flr Actions Node** on page 27-71 for a description of the prompts. After *LPFlrSetup*, one normally uses *LPFlrDarkAdapt*. *LPFlrCleanup* is required at the end. This routine also defines *FlrAction*.

LPFlrDarkAdapt   **Executes the standard fluorescence dark adaption**
    Initial: -
    Final: Long *Logic* (0=ok, 1=user pressed escape)

LPFlrCleanup   **Required after LPFlrSetup**
    Initial: -
    Final: -

PickAct   **Verify the current actinic control type and target(s)**
    Initial: -
    Final: Long *Logic* (0=ok, 1=user pressed escape)

FlrAction   **Performs fluorescence action specified in LPFlrSetup.**
    Initial: -
    Final: Long *Logic* (0=ok, 1=user pressed escape)
                    Does either "FsFm'" or else "FsFm'Fo'".

AutoPrograms and the Control Manager

During an AutoProgram, the control manager will be active while the commands *LPMeasure* and *LPMeasureTilStable* are being executed.

Interaction with Variable Tracking

It is advisable to avoid setting a control option to track a variable (described in **Variable Targets** on page 7-5), and at the same time try to control it with AutoProgram commands. Consider the following example: Suppose we select the lamp control option "Quantum Flux", and make the target #-13 (external quantum sensor). Then, we try to do a light curve. When each of the selected light targets is implemented in the light curve (by, for example, the *LPSetLamp* command) the quantum flux target will indeed be set to that value, but sometime within the following 30 seconds, the control manager will reset the target to the external quantum sensor value.

Thus, the target-setting AutoProgram commands (listed below) will override variable tracking for up to 30 seconds, but after that the variable tracking takes over.

Low Level Control Tools

The control manager is defined as part of OPEN. However, there is a very useful library file that should be included in any LPL program designed to run independently of OPEN that will be doing control related work. The file is /Sys/Lib/StdControls, and it provides several tools, including those documented below.

General Control Functions

AbdOn
   Initial: -
   Final: -

Turns on the flow control board.

This board must be on to control any of the analog or digital inputs or outputs. Uses digital output *dio_flowbd* (0x0302).

AbdOff
   Initial: -
   Final: -

Turns off the flow control board.

Counterpart of AbdOn.

IsAbdOn   **Is flow control board ON?**
   Initial: -
   Final: Long *logic* (1=yes, 0=no)
             Returns the state of digital output *dio_flowbd* (0x0302).

Flow Control Functions

PumpOn   **Turns on the pump, sets speed**
   Initial: -
   Final: -

             The pump speed is set to the current value of *pump_mv*, a variable defined in /sys/lib/stdcontrols. The switch is digital output *dio_pump* (0x0300), and the speed is D/A channel *da_pump* (6).

PumpOff   **Turns off the pump**
   Initial: -
   Final: -

             Unsets *dio_pump*.

IsPumpOn   **Is the pump running?**
   Initial: -
   Final: Long *logic* (1=yes, 0=no)
             Returns the state of the digital output *dio_pump*.

NullOnFlow   **Set flow control for constant flow.**
   Initial: Num *setPoint* (mV)
   Final: -

             Sets flow control circuit to null on flow meter signal, target value (mV) given by *setPoint*. This command unsets digital output *dio_null* (0x0001), and sets D/A channel *da_null_set* (0).

NullOnRH   **Set flow control for constant humidity.**
   Initial: Num *setPoint* (mV)
   Final: -

             Sets flow control circuit to null on water IRGA in the sample cell signal, set point given on the stack. Digital output *dio_null* is set, and D/A channel *da_null_set* is set to the target value.

IsRHNull   **Are we controlling on water (as opposed to flow)?**
   Initial: -
   Final: Long *logic* (1=humidity control, 0=flow control)
             Returns the state of digital output *dio_null*.

GetNullTarget   **Returns the current null set point value (mV)**
Initial: -
Final: Double *setPoint* (mV)
> Returns the value of D/A channel *da_null_set*.

Mixer Control Functions

Co2MixerPowerOn   **Powers on the CO_2 injection circuit**
Initial: -
Final: -

> OPEN does this once when it starts. This is NOT the function to use for turning the mixer on and off. Use Co2MixerOn and Co2MixerOff for that, but leave the power on all the time. Uses digital output *dio_inject* (0x0000).

Co2MixerPowerOff   **Powers off the CO_2 injection circuit**
Initial: -
Final: -

> See comments above.

Co2MixerOn   **Sets solenoid to allow CO_2 injection**
Initial: -
Final: -

> The flow of pure CO_2 is switched into the flow circuit. Uses digital output *dio_co2* (0x0303).

Co2MixerOff   **Sets solenoid to bypass CO_2 injection**
Initial: -
Final: -

> The flow of pure CO_2 is switched out of the flow circuit, so no mixing takes place.

IsCo2MixerOn   **Is CO_2 mixing enabled?**
Initial: -
Final: Long logic (0=mixer off, 1=mixer on)
> Returns the state of the digital output *dio_co2*.

SetCo2MixerMv   **Set the CO_2 mixer control voltage (0 to 5000 mV)**
Initial: Num *setPoint* (mV)
Final: -

> Converting μmol mol^{-1} to mV takes place at a higher level. Sets D/A channel *da_co2_set* (1).

GetCo2MixerMv   **Get the CO$_2$ mixer target voltage (mV)**
  Initial: -
  Final: Double *setPoint* (mV)
                          Returns the current value of D/A channel *da_co2_set*.

Cooler Control Functions

CoolFan_on   **Turns on cooler, and sets cooling fan to full speed.**
  Initial: -
  Final: -

                          Sets digital output dio_cool (0x0004), and D/A channel *da_cool_fan* to 5 V (full speed).

CoolFan_off   **Turns off cooler and cooling fan.**
  Initial: -
  Final: -

                          Unsets digital output dio_cool (0x0004), and D/A channel *da_cool_fan* to 0 V.

Cooling   **Are coolers on?**
  Initial: -
  Final: Long *Logic* (0=off, 1=on)
                          Returns state of digital output *dio_cool*.

Cooling_setpt   **Sets block cooling set point (C)**
  Initial: Num *temp* (C)
  Final: -
                          Sets D/A channel *da_chamber_temp* (4).

Lamp Control Functions

For the 6400-40 LCF, see **Leaf Chamber Fluorometer Control** on page 25-24.

SetLamp_mv   **Sets the lamp control value (0 - 5000 mV)**
  Initial: Num *setPoint* (mV)
  Final: -

                          If mV is 0, lamp is turned off by unsetting digital output *dio_lamp* (0x0005). Otherwise, the digital output is set, and the D/A channel *da_lamp* (2) is set to the target value.

GetLamp_mv   **Returns the current lamp control value (mV).**
  Initial: -
  Final: Double *setPoint* (mV)
                          Returns the state of D/A channel *da_lamp*.

IsLampON **Is the lamp on?**
Initial: -
Final: Long *logic* (0=off, 1=on)
>Returns the state of digital output *dio_lamp*.

LCF Control Functions

These functions are defined in "/Sys/Open/FlrTools", "/Sys/Open/FlrLamp", or "/Sys/Lib/FlrControls". All of these files are linked when the light source is set to the 6400-40 LCF. See also **LED Source Control** on page 25-22. For higher level fluorescence functions, see **Programming Commands** on page 27-86.

Actinic_on **Turns actinic on**
Initial: -
Final: -

Uses the current actinic definition and targets. See **LampSetNewTarget** on page 25-23.
Related Keywords: Actinic_off, IsActinicOn, FarRed_On

Actinic_off **Turns actinic off**
Initial: -
Final: -

Related Keywords: Actinic_on, IsActinicOn

ComputeBluePct **Computes the blue percentage of actinic**
Initial: -
Final: -

The result is in system variable *bluePct*.
Related Keywords:SetRed, SetBlue

FarRed_Off **Turns off the far red LED**
Initial: -
Final: -

Turns the far red LED off by calling *SetNIR_mV* with -100.
Related Keywords:FarRed_On

FarRed_On **Turn on the far red LED**
Initial: -
Final: -

This sets the intensity to the value specified by the FlrEditor and FlrAdjust user interfaces. The value is stored in a variable named *farRedTarget*, and uses a 0 to 10 scale. *FarRed_On* multiples *farRedTarget* by 500, and calls *SetNIR_mV*.
Related Keywords:FarRed_Off

FlrModulationOff   **Turns off modulation of the LCF**
  Initial: -
  Final: -

> This is NOT the method to zero the LCF. Rather, set the modulation intensity to 0.
> Related Keywords:FlrModulationOn

FlrModulationOn   **Turns on modulation of the LCF**
  Initial: -
  Final: -

> Related Keywords:FlrModulationOff

FlrPowerOn   **Power up the LCF**
  Initial: -
  Final: -

> Powers up the LCF and initializes its communications.
> Related Keywords:FlrPowerOff, FlrPowerOnIf, IsFlrPowered

FlrPowerOnIf   **Does the LCF power on sequence only if it is powered off**
  Initial: -
  Final: -

> Related Keywords:FlrPowerOff, FlrPowerOn, IsFlrPowered

FlrPowerOff   **Powers off the LCF**
  Initial: -
  Final: -

> Related Keywords: FlrPowerOn, FlrPowerOnIf, IsFlrPowered

GetBlue_mv   **Get the current blue actinic LED setting**
  Initial: -
  Final: Double *value* (0-5000)
> Related Keywords:SetBlue_mV

GetFlrFilterIndex   **Returns the current LCF filtering index**
  Initial: -
  Final: Long *filterIndex* (1-8)
> The index corresponds to a bandwidth of 0.5, 1, 5, 10, 20, 50, 100, or 200 Hz.
> Related Keywords:SetFlrFilterIndex

GetFlrGainIndex   **Returns current LCF gain setting index (1-4)**
  Initial: -
  Final: Long *gainIndex* (1-4)
> The values of *gainIndex* correspond to gains of 10, 20, 50 and 100.
> Related Keywords:SetFlrGainIndex

GetFlrMeasure_mV   **Get the measuring beam intensity**
     Initial: -
     Final: Double *mV*

                 mV will be -100 when turned off.
                  Related Keywords:SetFlrMeasure_mV

GetFlrRateIndex   **Returns the current modulation rate index**
     Initial: -
     Final: Long *modIndex* (1-4)

                  The values of modIndex correspond to modulation frequencies of 0.25, 1, 10, and 20
                  kHz.
                  Related Keywords:SetFlrRateIndex

GetNIR_mV   **Get far red LED intensity**
     Initial: -
     Final: Double *mV*

                  If the far red LED has been shut off with *FarRed_Off*, the value returned by
                  GetNIR_mV will be -100.
                  Related Keywords:SetNIR_mV

GetPDGainFactor   **Set the LCF photodiode gain**
     Initial: -
     Final: Long *gain* (1 or 5)

                  A gain of 5 is used for normal operation. During a flash, a gain of 1 is used.
                  Related Keywords:SetPDGainFactor

GetRed_mV   **Get current red actinic intensity**
     Initial: -
     Final: Double *mV*

                  Related Keywords:SetRed_mV

IsActinicOn   **Is actinic on?**
     Initial: -
     Final: Logic (1 = on, 0 = off)

                  Related Keywords: Actinic_on, Actinic_off

IsFarRedOn   **Is the far red LED on or not?**
     Initial: -
     Final: Logic (1=on, 0=off)

                  Related Keywords:FarRed_On, FarRed_Off

IsFlrPowered   **Is the LCF powered?**
     Initial: -
     Final: Logic (1 = powered, 0 = off)

                  Related Keywords:FlrPowerOn, FlrPowerOff, FlrPowerOnIf

SetBlue   **Sets blue actinic LED to specified mV level**
  Initial: Num *mV*
  Final: -

                    Calls SetBlue_mV, **then** ComputeBluePct
                    Related Keywords:SetBlue, SetRed

SetBlue_mV   **Set the blue actinic LEDs**
  Initial: Num *value* (0-5000)
  Final: -

                    If *value* > 0, also turns on the blue actinic status LED on the LCF connector.
                    Related Keywords:SetBlue_um, SetBlue, SetRed_mV

SetBlue_um   **Sets blue actinic LEDs to specified μmol m^{-2} s^{-1}.**
  Initial: Num *value*
  Final: -

                    Uses the current blue calibration table to determine the mV value, and calls SetBlue_mV.
                    Related Keywords:SetRed_um, SetBlue_mV

SetFlrFilterIndex   **Set the LCF filtering**
  Initial: Num *filterIndex* (1-8)
  Final: -

                    The index corresponds to a bandwidth of 0.5, 1, 5, 10, 20, 50, 100, or 200 Hz.
                    Related Keywords:GetFlrFilterIndex

SetFlrGainIndex   **Set the LCF gain**
  Initial: Num *gainIndex* (1-4)
  Final: -

                    The values of *gainIndex* correspond to gains of 10, 20, 50 and 100.
                    Related Keywords:GetFlrGainIndex

SetFlrMeasure_mV   **Set the measuring beam intensity**
  Initial: Num *mV* (0-5000)
  Final: -

                    To zero the LCF, turn the measuring intensity off by setting it to 0.
                    Related Keywords: SetFlrMeasure_mV

SetFlrRateIndex   **Set the modulation frequency**
  Initial: Num *modIndex* (1-4)
  Final: -

                    The values of modIndex correspond to modulation frequencies of 0.25, 1, 10, and 20 kHz.
                    Related Keywords:GetFlrRateIndex

SetNIR_mV   **Set far red LED intensity**
Initial: Num *mV* (0-5000)
Final: -

> This is a low level far red LED control. Calling this directly will bypass the normal user interface (FlrAdjust and FlrEditor). A more "neighborly" way to set the far red LED is use *FarRed_On* and *FarRed_Off*. If *mV* > 0, also turns on the far red status LED on the LCF connector.
> Related Keywords:GetNIR_mV

SetPDGainFactor   **Set the LCF photodiode gain**
Initial: Num *gain* (1 or 5)
Final: -

> A gain of 5 is used for normal operation. During a flash, a gain of 1 is used.
> Related Keywords:GetPDGainFactor

SetRed   **Sets red actinic to specified mV level**
Initial: Num *mV*
Final: -

> Calls *SetRed_mV*, then *ComputeBluePct*.
> Related Keywords:SetRed_mV, SetBlue

SetRed_mv   **Set red actinic intensity**
Initial: Num *mV*
Final: -

> If *mV* > 0, also turns on the red actinic status LED on the LCF connector.
> Related Keywords:SetBlue_mV, GetRed_mV

SetRed_um   **Sets red actinic LEDs to specified μmol m^{-2} s^{-1}.**
Initial: Num *value*
Final: -

> Uses current red actinic calibration to determine the mV level, and calls *SetRed_mV*.
> Related Keywords:SetRed_mV, SetBlue_um

IRGA Control Functions

IrgaOn   **Turns on analyzer board.**
Initial: -
Final: -

> This board must be on to operate the IRGAs. This function sets digital output *dio_irgabd* (0x0301).

IrgaOff   **Turns off analyzer board.**
Initial: -
Final: -

> Unsets digital output *dio_irgabd*.

IsIrgaOn   **Is analyzer board ON?**
   Initial: -
   Final: Long *logic* (0=off, 1=on)
                Returns state of digital output *dio_irgabd*.

StatusH2O   **Returns the status of the water IRGAs**
   Initial: -
   Final: Long *status* (1=ok, 2=bad ref, 3=bad sample, or 4=both ref and sample)
                This information comes from 2 digital inputs: *dio_h2o1* and *dio_h2o2* (0x0200 and 0x0201).

StatusCO2   **Returns the status of the CO2 IRGAs**
   Initial: -
   Final: Long *status* (1=ok, 2=bad ref, 3=bad sample, or 4=both ref and sample)
                This information comes from 2 digital inputs: *dio_co21* and *dio_co22* (0x0202 and 0x0203).

IrgaMatchOn   **Turn on the IRGA matching valve.**
   Initial: -
   Final: -
                <u>Unsets</u> digital output *dio_match* (0x0007).

IrgaMatchOff   **Turn off the IRGA matching valve.**
   Initial: -
   Final: -
                <u>Sets</u> digital output *dio_match* (0x0007).

IsIrgaMatch   **Is the match valve in the match position?**
   Initial: -
   Final: Long *location* (0=normal, 1=match)
                Returns the state of *dio_match*, but NOTted.

Chamber Fan Control Functions

ChFanSet   **Sets chamber fan speed**
   Initial: Num *speed* (0, 1, 2, 3, 4, or 5)
   Final: -
                0 is off, 5 is fastest. Value is stored in a public, integer variable *chFanState*. Non zero values (and values <= 5) will cause digital output *dio_fan* (0x0006) to be set, and D/A channel *da_chamber_fan* (7) to be set to 1000 times the speed. Otherwise, *dio_fan* is unset.

Example: Using the Library

As an example of how to use the tools in /Sys/Lib/StdControls, consider the stand-alone program shown in Figure 25-4. This program prompts the user for a command voltage, and then sets the lamp accordingly.

```
:INCLUDE "/Sys/Lib/StdControls"
:FCT main
{
  0 :FLOAT x

  AbdOn /* need flow control board on */
  LOOP
    "\nEnter command signal (0-5000):" PRINT
    &x "%f" KBD ENTER NOT BREAKIF
    &x 0 MAX 5000 MIN DROP /* range check */
    x SetLamp_mv
  ENDLOOP
}
```

Figure 25-4. Listing of a stand alone program to run the lamp.

26

Customizing Open

"Cry 'Havoc' - and let slip the dogs of war"

26

Customizing Open

This chapter explores some of the ways that you can modify OPEN to suit your purposes. The fact that it is one of the last chapters in the manual implies that the things that have come before, especially involving programming in LPL, should not be totally foreign to you.

XML Templates

Configuration

The file /Sys/Open/OpenBaseXML defines the structure and behavior of the configuration tree. A listing follows.

```
<open>
    <version addr="openRevID" attr="rh" />
    <configfile addr="userConfigFile" attr="rh" />
    <light oncic="SyncLightSource" attr="r" >
        <source addr="lightSource" edit="LightSourceMenuPick"
>"Sun+Sky"</source>
        <par_in addr="parInMode" get="getparinmode" edit="editparinmode"
oncic="SyncParInMode" >
            <sensor addr="parInSN" edit="PickParIn" attr="D" >
                        <cal addr="parInCal" fmtout="%1.2f" attr="r"
edit=PickParIn >
                    <actinity addr="actinity" fmtout="%1.2f"
edit="EditAct"  />
                    <transm addr="parInTrans" edit="editparinfactor"
fmtout="%1.2f"  >1</transm>
            </cal>
            </sensor>
            <transm addr="parOutTrans" edit="editparoutfactor"
>1</transm>
        </par_in>
        <par_out addr="parOutMode" get="getparoutmode"
edit="editparoutmode"  >
            <sensor addr="parOutSN" edit="PickParOutxml" attr="D" >
            <cal addr="parOutCal" attr="r" />
            </sensor>
        </par_out>
    </light>
        <comps attr="r" >
        <file addr="computeListFile" edit="EditComputeListXML"
oncic="XMLUpdateComps"  >
            <header addr="extheader" edit="EditExtHeader" />
                <extras subnodearray="extensionList" get="ExtrasLabel"
n_get="ExtensionListFct" edit="EditUserItemsXML"
n_edit="EditUserItemsXML" attr="D" n_attr="H" />
```

```
            </file>
        <energybal addr="doEB" toggle="no/yes" onwrite="CodeEB" attr="D">
            <f_parIn addr="f_parIn"  edit="AskThisID" id=-77 attr="D" />
            <f_parOut addr="f_parOut"  edit="AskThisID" id=-78 attr="D"
/>
            <alphaK addr="alphaK" fmtout="%1.2f" edit="AskThisID" id=-79
attr="D" />
        </energybal>
        <bc_total addr="condBL_mol"  oncic="UpdateBLC" attr="r"
fmtout="%1.3f" >
            <stomatal_ratio addr="stom_rat" fmtout="%1.3g"
edit="AskThisID" id=-34 attr="D" />
            <bc_oneside addr="condBL_one" edit="AskThisID"  id=-55
fmtout="%1.3f" >
                <type addr="compBLC" get="ShowBlcType"
edit="EditBLCStyle" attr="D" />
                <table addr="blcTable" edit="ChangeBLCTable"
onwrite="ReadBLCTable" attr="D" />
                <area addr="area_cm2" fmtout="%1.3g" edit="AskThisID"
id=-33 attr="D" />
                <slope addr="blcSlope" edit="AskThisID" id=-18 />
                <offset addr="blcOffset" edit="AskThisID" id=-19 />
                <fan addr="chFanState" attr="r" edit="changefan"
get="getfan" />
            </bc_oneside>
        </bc_total>
    </comps>
    <prompts attr="r" >
        <onlog addr="ucnAutoPrompt" toggle="off/on" />
            <items addr="PromptList.label" edit="EditUserPrompts"
n_edit="EditUserPrompts" n_get="PromptLabelFct"
subnodearray="PromptList.list" oncic="DecodePromptLabels" n_attr="rH" />
        </prompts>
    <constants>
        <oxygen addr="oxyPCT" id=-52 edit="AskThisID" fmtout="%g
%%">21</oxygen>
        <bb_vapor addr="bbH2O" edit="AskBBH2O" > 1.5 </bb_vapor>
        <bb_oxy addr="bbO2" edit="AskBBO2" > 0.9 </bb_oxy>
    </constants>
    <fctkeys >
        <level3_f1 addr=udk[1] id=1 get=GetUdkLabel edit=EditUdk >{ 1
-33 0 "" "" }</level3_f1>
        <level3_f2 addr=udk[2] id=2 get=GetUdkLabel edit=EditUdk >{ 1
-34 0 "" "" }</level3_f2>
        <level7_f1 addr=udk[3] id=3 get=GetUdkLabel edit=EditUdk >{ 0
0 0 "" "" }</level7_f1>
        <level7_f2 addr=udk[4] id=4 get=GetUdkLabel edit=EditUdk >{ 0
0 0 "" "" }</level7_f2>
        <level7_f3 addr=udk[5] id=5 get=GetUdkLabel edit=EditUdk >{ 0
0 0 "" "" }</level7_f3>
        <level7_f4 addr=udk[6] id=6 get=GetUdkLabel edit=EditUdk >{ 0
0 0 "" "" }</level7_f4>
        <level7_f5 addr=udk[7] id=7 get=GetUdkLabel edit=EditUdk >{ 0
0 0 "" "" }</level7_f5>
    </fctkeys>
    <stability>
            <items addr="StatTrackerState.Label"  oncic="RedefineST"
edit="DefineStability" subnodearray="StatTrackerState.items" n_attr="H"
n_get="StabilityLabelFct" n_edit="DefineStability" />
    </stability>
```

```
<log>
    <format addr="logFormatLabel" edit="edit_log_format">
                <items addr="userLogList" get="logFormatLabelFct"
edit="edit_log_format" >
                            <hdr addr="userHeaderList"
edit=edit_log_format>/>
            </items>
    </format>
            <options>
                <beep disp="Beep" delim=":" addr="LogOpts.Beeper"
toggle="off/on">1</beep>
                    <hdr disp="Header" delim=":" addr="LogOpts.Header"
toggle="normal/separate">0</hdr>
                    <rem disp="Remarks" delim=":" addr="LogOpts.Remarks"
toggle="normal/separate">0</rem>
                    <stab disp="Stability" delim=":"
addr="LogOpts.Stability" toggle="no/logged">0</stab>
                    <stats disp="Statistics" delim=":" get=LogOptsGetFn
oncic=LogOptsUpdate >0
                        <mean delim=":" disp="Means->log file"
addr="LogOpts.StatsLogMean" toggle="no/yes" >0</mean>
                            <file delim=":" disp="Stats->.stats file"
addr="LogOpts.StatsFile" toggle="no/yes" >0</file>
                        <period delim=":" addr="LogOpts.Period"
edit=-1>15</period>
                    </stats>
                <excel disp="Excel file" delim=":" addr="LogOpts.XL"
toggle="no/yes">yes</excel>
                    <controls disp="Control changes" delim=":"
addr=LogOpts.Controls toggle="no/yes">yes</controls>
                    <comm disp="Echo to Comm" delim=":" addr="LogOpts.Comm"
toggle="no/yes">no</comm>
    </options>
    <datafile addr="logFIleName">"/User/Data"</datafile>
    <button>
        <action addr="CurrentLog.Label" edit="EditLogBtn">"Normal
Log"
                <code addr="CurrentLog.code" edit="EditLogBtn">"
LogOneObsManual "</code>
        </action>
    </button>
    <autoprog addr="activeAutoProg" attr="r">
        <active addr="apactive" attr="r" toggle="no/yes" />
    </autoprog>
</log>
<display>
    <text addr="NMDisplay.label" edit="FmtUserEdit">
                <lines edit=FmtUserEdit n_edit="editdispline"
n_get="IDsToNames">
            <a addr="NMDisplay.line[1]" id=1>-1 -2 -4 -5</a>
            <b addr="NMDisplay.line[2]" id=2>-3 -6 -7 -15</b>
            <c addr="NMDisplay.line[3]" id=3>30 23 36 21</c>
            <d addr="NMDisplay.line[4]" id=4>39 25 27</d>
            <e addr="NMDisplay.line[5]" id=5>-71 -72 -73 -74</e>
            <f addr="NMDisplay.line[6]" id=6>-14 -15 -16 -17</f>
            <g addr="NMDisplay.line[7]" id=7>-11 -12 -13 -32</g>
            <h addr="NMDisplay.line[8]" id=8>-8 -9 -10</h>
            <i addr="NMDisplay.line[9]" id=9>-21 -22 -23 -24</i>
            <j addr="NMDisplay.line[10]" id=10>-31</j>
            <k addr="NMDisplay.line[11]" id=11>-22 -56 -57 -71</k>
```
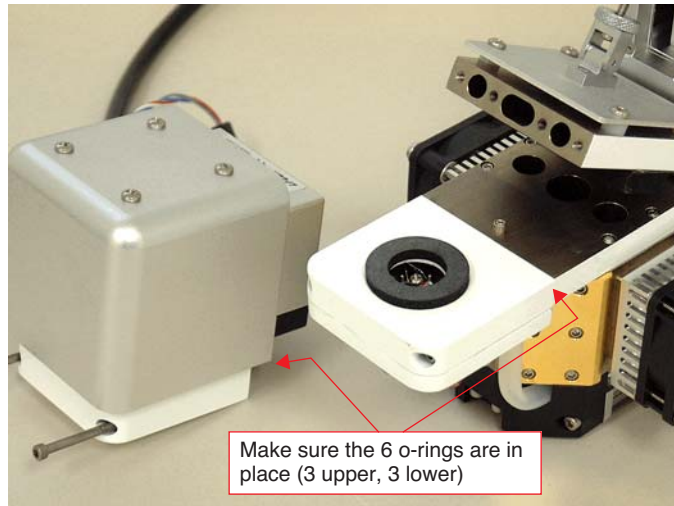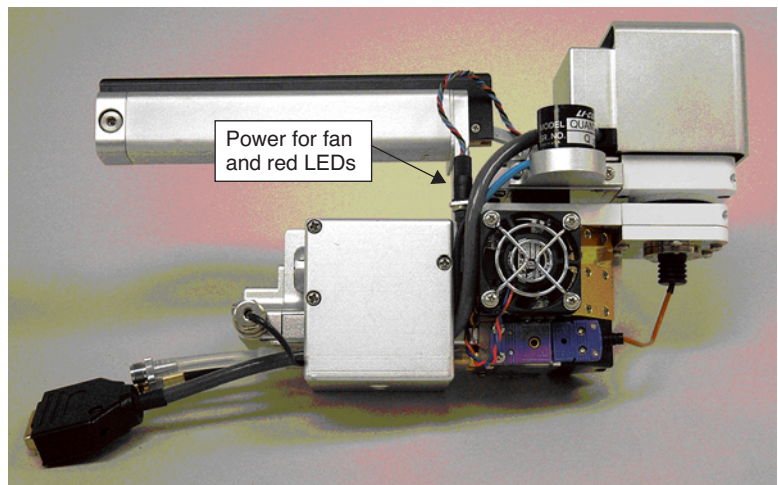
```
                          <l addr="NMDisplay.line[12]" id=12>-48 -49 -50 -51</l>
                          <m addr="NMDisplay.line[13]" id=13></m>
                          <n addr="NMDisplay.line[14]" id=14></n>
                          <o addr="NMDisplay.line[15]" id=15></o>
                          <p addr="NMDisplay.line[16]" id=16></p>
                          <q addr="NMDisplay.line[17]" id=17></q>
                          <r addr="NMDisplay.line[18]" id=18></r>
                          <s addr="NMDisplay.line[19]" id=19></s>
                          <t addr="NMDisplay.line[20]" id=20></t>
                          <u addr="NMDisplay.line[21]" id=21></u>
                          <v addr="NMDisplay.line[22]" id=22></v>
                          <w addr="NMDisplay.line[23]" id=23></w>
                          <x addr="NMDisplay.line[24]" id=24></x>
                          <y addr="NMDisplay.line[25]" id=25></y>
                          <z addr="NMDisplay.line[26]" id=26></z>
                  </lines>
                  <groups>
                          <home addr="NMDisplay.home"
        edit="edithomeset">abc</home>
                          <end addr="NMDisplay.end" edit="editendset">def</end>
                          <pgup addr="NMDisplay.pgup" edit="editupset">ghi</pgup>
                          <pgdn addr="NMDisplay.pgdn" edit="editdnset">jkl</pgdn>
                  </groups>
          </text>
          <graphs addr="rtgLabel" edit="Sc_Edit" >
                  <a addr="_g1.title" edit="g1edit" oncic="g1write" attr="H">
                          <plot1 nodes="_g1.plot1" />
                          <plot2 nodes="_g1.plot2" />
                          <plot3 nodes="_g1.plot3" />
                  </a>
                  <b addr="_g2.title" edit="g2edit" oncic="g2write" attr="H">
                          <plot1 nodes="_g2.plot1" />
                          <plot2 nodes="_g2.plot2" />
                          <plot3 nodes="_g2.plot3" />
                  </b>
                  <c addr="_g3.title" edit="g3edit" oncic="g3write" attr="H">
                          <plot1 nodes="_g3.plot1" />
                          <plot2 nodes="_g3.plot2" />
                          <plot3 nodes="_g3.plot3" />
                  </c>
                  <d addr="_g4.title" edit="g4edit" oncic="g4write" attr="H">
                          <plot1 nodes="_g4.plot1" />
                          <plot2 nodes="_g4.plot2" />
                          <plot3 nodes="_g4.plot3" />
                  </d>
                  <e addr="_g5.title" edit="g5edit" oncic="g5write" attr="H">
                          <plot1 nodes="_g5.plot1" />
                          <plot2 nodes="_g5.plot2" />
                          <plot3 nodes="_g5.plot3" />
                  </e>
                  <f addr="_g6.title" edit="g6edit" oncic="g6write" attr="H">
                          <plot1 nodes="_g6.plot1" />
                          <plot2 nodes="_g6.plot2" />
                          <plot3 nodes="_g6.plot3" />
                  </f>
                  <g addr="_g7.title" edit="g7edit" oncic="g7write" attr="H">
                          <plot1 nodes="_g7.plot1" />
                          <plot2 nodes="_g7.plot2" />
                          <plot3 nodes="_g7.plot3" />
                  </g>
```
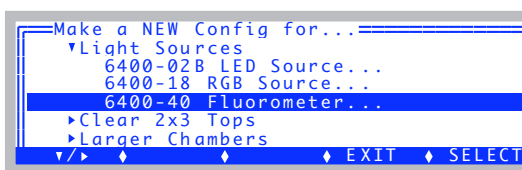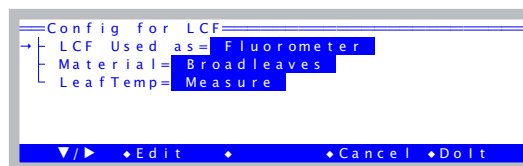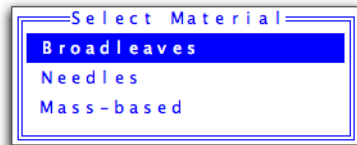
```
            <h addr="_g8.title" edit="g8edit" oncic="g8write" attr="H">
                <plot1 nodes="_g8.plot1" />
                <plot2 nodes="_g8.plot2" />
                <plot3 nodes="_g8.plot3" />
            </h>
        </graphs>
    </display>
    <controls>
        <defaults>
                <fan addr=defaultFanState  edit=editFanDefault >5</fan>
                  <flow edit=EditDefaultFlow get=GetDefaultFlowLabel  >
                        <type addr=defaultFlowTargetIndex
edit=EditDefaultFlow >2</type>
                      <target addr=defaultFlowTarget edit=EditDefaultFlow
>500</target>
                  </flow>
                  <co2 edit=EditDefaultCO2 get=GetDefaultCO2Label   >
                   <type addr=defaultCO2TargetIndex edit=EditDefaultCO2
>1</type>
                        <target addr=defaultCO2Target  edit=EditDefaultCO2
/>
                  </co2>
                <temp edit=EditDefaultCooler get=GetDefaultCoolerLabel
attr="D"  >
                        <type addr="defaultCoolerIndex"
edit=EditDefaultCooler >1</type>
                        <target addr="defaultCoolTarget"
edit=EditDefaultCooler />
                </temp>
                <light edit="editlight" get="lampgetlabelxml" attr="D" >
                        <type addr="defaultLightIndex" edit=editlight
>1</type>
                        <target addr="defaultLightTarget" edit=editlight />
                </light>
        </defaults>
    </controls>
    <a2d>
        <avgtime addr="hiResAvgTime" edit="EditAvgTime" fmtout="%1.1f
secs">4.0</avgtime>
        <userchans>
            <ch20 addr="uC_State[1]" onread="UCR1" onwrite="UCW1"
edit="UCE1"></ch20>
            <ch21 addr="uC_State[2]" onread="UCR2" onwrite="UCW2"
edit="UCE2"></ch21>
            <ch22 addr="uC_State[3]" onread="UCR3" onwrite="UCW3"
edit="UCE3"></ch22>
            <ch23 addr="uC_State[4]" onread="UCR4" onwrite="UCW4"
edit="UCE4"></ch23>
        </userchans>
    </a2d>
    <comm>
        <config addr="nonLTermConfig" onread="GetNLTC" onwrite="SetNLTC"
edit="editNLTC" get="GetFctNLTC">9600 8 1 N</config>
        <lterm addr="ltermonoff" onwrite="SetLTerm" onread="GetLTerm"
get="GetFctLTerm" edit="EditLTerm" >-1</lterm>
        <incoming addr="commInOption" get="CommInGetFct"
edit="CommInEdit" >0</incoming>
    </comm>
    <matching>
        <type addr="matchType" get="MatchGetFct"
```
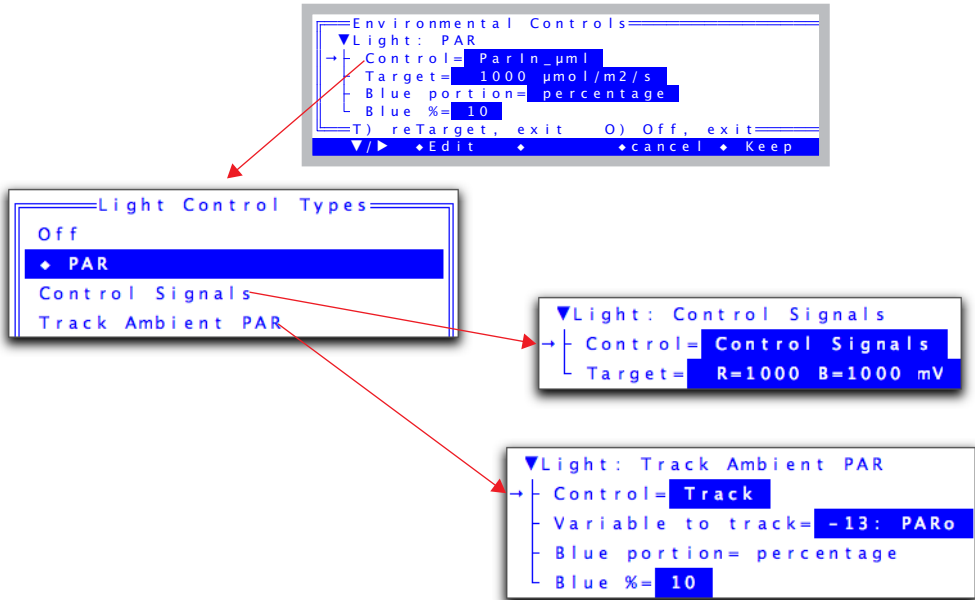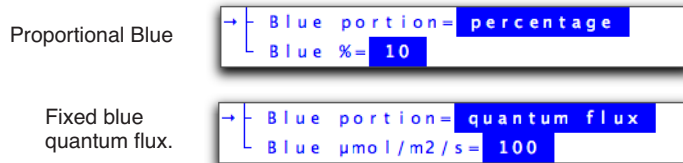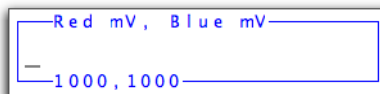
```
edit="MatchTypeToggle">0</type>
        <disp addr="matchDisp" fmtout="'%c'"
edit="matchDispEdit">'a'</disp>
        <settings>
            <CO2Limit addr="MatchCO2Limit" edit="MatchCO2LimitEdit"
fmtout="%1.1f">10</CO2Limit>
            <H2OLimit addr="MatchH2OLimit" edit="MatchH2OLimitEdit"
fmtout="%1.1f">1</H2OLimit>
            <MaxAutoTime addr="matchAutoTimeLimit"
edit="MatchAutoMaxEdit">60</MaxAutoTime>
        </settings>
    </matching>
    <fan attr="X">
        <fast />
        <slow />
    </fan>
    <hooks oncic="CompileXMLHooks" >
            <items subnodearray="hookItems" get="HookLabelxml"
n_get="HookItemGetxml" edit="hookEditxml"  attr="" n_attr="H"
n_edit=hookEditxml >
    </hooks>
</open>
```

Calibration

OPEN's calibration tree structure and behavior is defined by /Sys/Open/CalBaseXML, which is listed here.

```
<li6400>
    <version addr="openRevID" attr="rh" />
    <factory>
        <unit addr="thisUnit">"PSC-????"</unit>
        <serviced addr="lastServiced">"dd mmm yyyy"</serviced>
        <fuseaware addr="parm1FuseAware" />
        <co2mixer addr="mixerAvail" toggle="no/yes" attr="e" />
        <co2>
            <coeffs addr="stdCO2Cal">{2.0E-1 3.7E-5 1.1E-8 -7.7E-13 8.5E-
17}</coeffs>
            <dvdt addr="co2MvPerC">-3</dvdt>
            <xs addr="xsWC">{0 0}</xs>
        </co2>
        <h2o>
            <coeffs addr="stdH2OCal">{5.9E-3 1.8E-6 1.6E-10}</coeffs>
            <dvdt addr="h2oMvPerC">-1</dvdt>
            <xs addr="xsCW">{0 0}</xs>
        </h2o>
        <flow addr="stdFlowCal">{0 0.35}</flow>
        <press addr="stdPressureCal">{88 0.005}</press>
    </factory>
    <user>
        <flow_zero read="XmGetFlowZer" write="XmSetFlowZer" />
        <irga_zero>
            <co2 read="XmGetCO2Zer" write="XmSetCO2Zer">
                <at addr="tLastCO2Zero" />
            </co2>
            <h2o read="XmGetH2OZer" write="XmSetH2OZer">
                <at addr="tLastH2OZero" />
            </h2o>
```

```
        </irga_zero>
        <irga_span>
            <co2 addr="co2Gains">{1 1}</co2>
            <h2o addr="h2oGains">{1 1}</h2o>
        </irga_span>
        <irga_match>
            <co2 addr="co2Match">{0 0}</co2>
            <h2o addr="h2oMatch">{0 0}</h2o>
        </irga_match>
        <co2_mixer>
            <pump_mv addr="mixerPump_mV" attr="C">4500</pump_mv>
            <ppm addr="mixerPPM" attr="">{ 2400 1100  450  270 120  70
50  30 }</ppm>
            <mv addr="mixerSignal" attr="C">{ 5000 3000 1500 1000 500 300
200 100 }</mv>
        </co2_mixer>
        <parin_offset addr="parInOffset">0</parin_offset>
        <led_cal attr="Hh" >
            <unit addr="lastLEDCalSN" >"unknown"</unit>
            <mv addr="lampMv" attr="C">{ 50 100 1000 3000 5000 } </mv>
            <qntm addr="lampQntm" >{ 3 45 555 1500 2500 } </qntm>
        </led_cal>
        <lcf_cal attr="Hh" oncic="FlrFitXML" >
            <unit addr="lastFlrCalSN" >"unknown"</unit>
            <red>
                <mv addr="flrRedmV" attr="C">{ 50 100 500 1000 1500 5000
}</mv>
                <qntm addr="flrRedQntm" >{ 50 100 500 1000 2000 5000 }
</qntm>
            </red>
            <blue>
                <mv addr="flrBlueMv" attr="C">{ 100 200 500 1000 2000
5000 } </mv>
                <qntm addr="flrBlueQntm" >{ 10 20 50 100 200 500 } </qntm>
            </blue>
        </lcf_cal>
    </user>
    <accessories>
        <led>
        </led>
        <tops>
        </tops>
        <quantum>
        </quantum>
    </accessories>
</li6400>
```

System Variable Definitions

In the file /Sys/Open/Open.df2 is a pointer array named *systemVariables* that defines the set of system variables. It is listed below. The meanings of the fields in each record is the same as that described in **The userList Pointer Array** on page 15-31.

```
:CHAR
    gfmt[] "%(* )d\n"
            statLineLab[] "CO2\xde H2O\xdePump\xdeFlow\xdeMixr\xde Fan"
        cpstb[] "Control Panel Stability: Flow Mixer Cooler Lamp"

        s18[] "%18s"
        s28[] "%28s"
        s27[] "%27s"
        s26[] "%26s"
        s38[] "%38s"
        s8[] "%8s"
        s4[] "%4s"
        sq[] "\"%s\""
        ls8[] "\"%-8.8s\""
        ls18[] "\"%-18.18s\""
        ls28[] "\"%-28.28s\""
        ls26[] "\"%-26.26s\""
        ls38[] "\"%-38.38s\""
        ld8[] "%8ld"
        ld1[] "%ld"
        stat6[]"%4s %4s %4s %4s %4s %4s"
        stat6Log[] "\x22%4s %4s %4s %4s %4s %4s\x22"

        e81 "%8.1E"
        e11 "%1.1E"

    markerBase 'a'

PUB systemVariables[]
{
    :PTR {0 "Time" s8 a2dTime f81 "Secs since power on" sad f11 }
      :PTR {-1 "CO2R_\xe6ml" s8 co2_1_um f81 "Ref. CO2 \xe6mol/mol" "CO2R"
f12 }
          :PTR {-2 "CO2S_\xe6ml" s8 co2_2_um f81 "Chmbr. CO2 \xe6mol/mol"
"CO2S" f12 }
            :PTR {-3 "\x7fCO2_\xe6ml" s8 co2_diff_um f81 "Ch - Ref CO2
\xe6mol/mol" "DCO2" f12 }
      :PTR {-4 "H2OR_mml" s8 h2o_1_mm f83 "Ref. H2O mmol/mol" "H2OR" f13 }
      :PTR {-5 "H2OS_mml" s8 h2o_2_mm f83 "Chmbr. H2O mmol/mol" "H2OS" f13 }
      :PTR {-6 "\x7fH2O_mml" s8 h2o_diff_mm f83  "Ch - Ref H2O mmol/mol"
"DH2O" f13}
          :PTR {-7 "Flow_\xe6ml" s8 flow_um f81 "Flow Rate \xe6mol/s" "Flow"
f11}
          :PTR {-8 "Tblock\xf8C" s8 tblk_c f82 "IRGA Block Temp C" "TBlk" f12}
            :PTR {-9 "Tair\xf8C" s8 tcham_c f82 "Chmbr Air Temp C" "Tair" f12 }
             :PTR {-10 "Tleaf\xf8C" s8 tleaf_c f82 "Leaf Temp C" "Tleaf" f12 }
      :PTR {-11 "Prss_kPa" s8 press_kpa g84 "Atm Press kPa" "Press" g14 }
        :PTR {-12 "ParIn_\xe6m" s8 parIn_um f80 "In-chmbr PAR \xe6mol/m2/s"
"PARi" f10}
          :PTR {-13 "ParOut\xe6m" s8 parOut_um f80 "Extrnl PAR \xe6mol/m2/s"
"PARo" f10}
      :PTR {-14 "RH_R_%" s8 rhIn f82 "Ref RH %" "RH_R" f12}
```

```
    :PTR {-15 "RH_S_%" s8 rhOut f82 "Chamber RH %" "RH_S" f12 }
      :PTR {-16 "Td_R_\xf8C" s8 tdIn f82 "Ref DewPoint C" "TdR" f12 }
       :PTR {-17 "Td_S_\xf8C" s8 tdOut f82 "Chmbr DewPoint C" "TdS" f12 }
    :PTR {-18 "BLCslope" s8 blcSlope g83 "Bndry Layer Cond slope (as
f(area))" sad g13 1}
    :PTR {-19 "BLCoffst" s8 blcOffset g83 "Bndry Layer Cond Offset (as
f(area))" sad g13 1}

    :PTR {-21 "HH:MM:SS" s8 clocktime s8 "Real time clock" "HHMMSS" sq }
    :PTR {-22 "Program" s8 lpTimeStat s8 "Auto Program status" sad ls8 }
    :PTR {-23 "CHPWMF" s8 statusWord "  %06lu" "Status:
Co2/H2o/Pump/floW/Mixer/Fan" "Status" "%06lu"}
    :PTR {-24 "Battery" s8 battery_v f82 "Battery voltage" sad f12 }
    :PTR {-25 "CO2" s4 stat_co2 s4 "CO2 IRGA status" sad sq}
    :PTR {-26 "H2O" s4 stat_h2o s4 "H2O IRGA status" sad sq}
    :PTR {-27 "PUMP" s4 stat_pump s4 "Pump status" sad sq}
    :PTR {-28 "FLOW" s4 stat_flow s4 "Flow Control status" sad sq}
    :PTR {-29 "MIXR" s4 stat_inj s4 "CO2 Mixer status" sad sq}
    :PTR {-30 "FAN" s4 stat_fan s4 "Chamber fan status" sad sq}
    :PTR {-31 statLineLab "%29s" statLineVar stat6 "The status line"
statLineLab stat6log } /* needed ? */
    :PTR {-32 "BLC_mol" s8 condBL_mol g83 "Bndry Layer Cond mol/m2/s"
"BLCond" g13 1}
    :PTR {-33 "AREA_cm2" s8 area_cm2 g83 "Leaf area cm2" "Area" g13 1}
    :PTR {-34 "STMRATIO" s8 stom_rat g83 "Stomatal ratio estimate"
"StmRat" g13 1}
    :PTR {-35 "Obs" s8 obsInPad "%8d" "# Obs stored in log file" sad "%d" }
    :PTR {-36 "FTime" s8 obsTime f81 "Time since log file opened" sad f11
}
    :PTR {-37 "CO2R_mv" s8 co2_1_mv f81 "Ref. CO2 IRGA mV" sad f11}
    :PTR {-38 "CO2S_mv" s8 co2_2_mv f81 "Chmbr CO2 IRGA mV" sad f11}
    :PTR {-39 "H2OR_mv" s8 h2o_1_mv f81 "Ref H2O IRGA mV" sad f11}
    :PTR {-40 "H2OS_mv" s8 h2o_2_mv f81 "Chmbr H2O IRGA mV" sad f11}
    :PTR {-41 "Tblk_mv" s8 tchamblk_mv f81 "IRGA Block temp mV" sad f11}
    :PTR {-42 "Tair_mv" s8 tcham_mv f81 "Chmbr air temp mV" sad f11}
    :PTR {-43 "Tleaf_mv" s8 tleaf_mv f81 "Leaf temp mV" sad f11}
    :PTR {-44 "flow_mv" s8 flow_mv f81 "Flow meter mV" sad f11}
    :PTR {-45 "press_mv" s8 pressure_mv f81 "Pressure mV" sad f11}
    :PTR {-46 "parIn_mv" s8 parIn_mv f81 "In-chmbr PAR mV" sad f11}
    :PTR {-47 "parOutmV" s8 parOut_mv f81 "Extrnl PAR mV" sad f11}
    :PTR {-48 "CRagc_mv" s8 agc_c1_mv f81 "Ref CO2 IRGA AGC mV" sad f11}
    :PTR {-49 "CSagc_mv" s8 agc_c2_mv f81 "Chmbr CO2 IRGA AGC mV" sad f11}
    :PTR {-50 "HRagc_mv" s8 agc_h1_mv f81 "Ref H2O IRGA AGC mV" sad f11}
    :PTR {-51 "HSagc_mv" s8 agc_h2_mv f81 "Chmbr H2O IRGA AGC mV" sad f11}
    :PTR {-52 "Oxygen%" s8 oxyPct f81 "Oxygen concentration (%)" sad f11 1}
    :PTR {-53 "vapR_kPa" s8 eAir_1_kPa f82 "Ref vap press kPa" sad f12}
    :PTR {-54 "vapS_kPa" s8 eAir_2_kPa f82 "Chmbr vap press kPa" sad f12}
    :PTR {-55 "BLC1_mol" s8 condBL_one g83 "1-sided Bndry Layer
(mol/m2/s)" sad g13}
    :PTR {-56 "ProgPrgs" s8 lpProgress s8 "AutoProg Progess Indicator" sad
ls8}
    :PTR {-57 "FwMxCrLp" s8 cpStable s8 cpstb sad sq}
      :PTR {-58 "Tirga\xf8C" s8 tIrga_c f82  "IRGA Temp C" "Tirga" f12}
    :PTR {-59 "Tirga_mv" s8 tIrga_mv f81  "IRGA Temp mV" sad f11}
    :PTR {-60 "uc_20_mV" s8 chan20_mv f81 "User Channel 20 mV" sad f11}
    :PTR {-61 "uc_21_mV" s8 chan21_mv f81 "User Channel 21 mV" sad f11}
    :PTR {-62 "uc_22_mV" s8 chan22_mv f81 "User Channel 22 mV" sad f11}
    :PTR {-63 "uc_23_mV" s8 chan23_mv f81 "User Channel 23 mV" sad f11}
    :PTR {-64 "DecHour" s8 decHour f85 "Time of day (decimal)" sad f15 }
    :PTR {-65 "CsMch" s8 co2_2_offset g83 "CO2 Sample Matching Value" sad
```

```
g13 }
    :PTR {-66 "HsMch" s8 h2o_2_offset g83 "H2O Sample Matching Value" sad
g13 }
    :PTR {-67 "CrMch" s8 co2_1_offset g83 "CO2 Ref Matching Value" sad g13
}
    :PTR {-68 "HrMch" s8 h2o_1_offset g83 "H2O Ref Matching Value" sad g13
}
    :PTR {-69 "DOY" s8 clockDOY ld8 "Day Of Year" sad ld1 }
    :PTR {-70 "YYYYMMDD" s8 clockDate ld8 "YearMonthDay" sad ld1 }
    :PTR {-71 "Stable" s8 StableStat s8 "# stable / total" sad sq }
    :PTR {-72 "StableF" s8 fractStable f82 "Fraction stable" sad f11 }
    :PTR {-73 StabVarLabel s8 StableFlags s8 "Stable flags" sad sq }
    :PTR {-74 "TotalCV" s8 totalCV f81 "Total CV" sad f11 }

    :PTR {-76 "EBal?" s8 doEB "%8d" "EnergyBalance? 1=yes, 0=no" sad "%d"
1 }
    :PTR {-77 "f_parin" s8 f_parIn f82 "ParIn weighting for EB" sad f12 1 }
    :PTR {-78 "f_parout" s8 f_parOut f82 "ParOut weighting for EB" sad f12
1 }
    :PTR {-79 "alphaK" s8 alphaK f82 "absorptance * conversion factor" sad
f12 1 }
    :PTR {-80 "F" s8 flr_f f81 "zero corrected flr signal" sad f11 }
    :PTR {-81 "%Blue" s8 bluePct f80 "Blue fraction" sad f10 }
    :PTR {-82 "FlrMax" s8 flrMax f81 "Max flr during flash" sad f11 }
      :PTR {-83 "FPeak_\xe6m" s8 flashMax f80 "Flash peak \xe6mol" "FPeak"
f10 }
    :PTR {-84 "FCnt" s8 flashCount ld8 "Flash count" sad ld1 }
    :PTR {-85 "Fzero" s8 flrZero f81 "Flr zero" sad f11}
    :PTR {-86 "Fo" s8 flr_o f81 "Fo" sad f11 }
    :PTR {-87 "Fo'" s8 flr_op f81 "Fo'" sad f11 }
    :PTR {-88 "Fm" s8 flr_m f81 "Fm" sad f11 }
    :PTR {-89 "Fm'" s8 flr_mp f81 "Fm'" sad f11 }
    :PTR {-90 "Fs" s8 flr_s f81 "Fs" sad f11 }
    :PTR {-91 "FlrEvent" s8 flrStat s8 "Lastest flr event" sad ls8 }
    :PTR {-92 "M:Int  kHz  Hz  Gn" s18 msrStat s18 "Msr Settings" sad ls18 }
    :PTR {-93 flashLabPtr s28 flashStat s28 "Flash Settings" sad s27 }
    :PTR {-94 "D:Dur Far Bfr Aft  kHz  Hz" s26 darkStat s26 "Dark Settings"
sad ls26 }
    :PTR {-95 "FlrMin" s8 flrMin f81 "Min flr during dark" sad f11 }
    :PTR {-96 "ParIn@Fs" s8 parIn_fs f81 "PAR at last Fs" sad f11 }
    :PTR {-97 "FlrCV_%" s8 flrRms f82 "Flr stability CV %" sad f12 }
    :PTR {-98 "dF/dt" s8 flrSlope g82 "Flr slope" sad g12 }
      :PTR {-99 "FMaxStd" s8 flrMaxStd f81 "Simple fmax during MPF" sad
f11 }
    :PTR {-100 "DCnt" s8 darkCount ld8 "Dark pulse count" sad ld1 }

    :PTR {-101 aux1.label aux1.screenLabFmt aux1.floatVal
aux1.screenValFmt aux1.desc sad aux1.logValFmt 1 }
    :PTR {-102 aux2.label aux2.screenLabFmt aux2.floatVal
aux2.screenValFmt aux2.desc sad aux2.logValFmt 1 }
    :PTR {-103 aux3.label aux3.screenLabFmt aux3.floatVal
aux3.screenValFmt aux3.desc sad aux3.logValFmt 1 }
    :PTR {-104 aux4.label aux4.screenLabFmt aux4.floatVal
aux4.screenValFmt aux4.desc sad aux4.logValFmt 1 }
    :PTR {-105 aux5.label aux5.screenLabFmt aux5.floatVal
aux5.screenValFmt aux5.desc sad aux5.logValFmt 1 }
    :PTR {-106 aux6.label aux6.screenLabFmt aux6.floatVal
aux6.screenValFmt aux6.desc sad aux6.logValFmt 1 }
    :PTR {-107 aux7.label aux7.screenLabFmt aux7.stringVal
aux7.screenValFmt aux7.desc sad aux7.logValFmt 1 }
```

```
    :PTR {-108 aux8.label aux8.screenLabFmt aux8.stringVal
aux8.screenValFmt aux8.desc sad aux8.logValFmt 1 }
    :PTR {-109 aux9.label aux9.screenLabFmt aux9.stringVal
aux9.screenValFmt aux9.desc sad aux9.logValFmt 1 }

    :PTR {-111 "chan15_mV" s8 chan15_mv f81 "Analog input channel 15" sad
f11}
        :PTR { -112 "matchCO2" s8 matchPrevCO2 f81 "CO2R at prev match"
sad f11}
        :PTR {-113 "matchH2O"  s8 matchPrevH2O f82 "H2OR at prev match"
sad f12}
        :PTR { -114 "mchElpsd" s8 matchElapsed s8 "Elapsed since prev match
(string)" sad s8}
        :PTR { -115 "Fmean" s8 flrAvg f81 "Flr mean" sad f11}
}
```

Useful Variables

Light Measurements

Table 26-1 lists some important variables used in the computation of light levels, besides those that are explicitly system variables (Table 14-10 on page 14-23.

Table 26-1. Other useful variables for light measurement.

Variable Name	Description
actinity	Actinity correction factor, f_a in Eqn (14-18), pg 14-11. <open> <light> <par_in> <sensor> <cal> <actinity>
parInTrans	τ in Eqn (14-18), pg 14-11. <open> <light> <parin> <sensor> <cal> <transm>
parOutTrans	τ_x in Eqn (14-17), pg 14-10. <open> <light> <par_in> <transm>
parInMode	<open> <light> <par_in>. 0 = None 1 = Measured in chamber 2 = ParOut * parOutTrans
parInCal	<open> <light> <par_in> <sensor> <cal> On configuration, this value comes from one of the nodes in <li6400> <accessories>, or else from the sensor itself, depending on the source. Thereafter, the value may change if the light source can change its color (6400-40, 6400-18).
parOutMode	<open> <light> <par_out> 0 = None 1 = Measured

Table 26-1. Other useful variables for light measurement.

Variable Name	Description
parOutCal	\<open> \<light> \<par_out> \<sensor> \<cal> On configuration, the value comes from one of the nodes in \<li6400> \<accessories> \<quantum>,

A/D Measurements

Table 26-2 lists some variables that pertain to how the LI-6400 makes analog measurements in New Measurements mode.

Table 26-2. Variables defined in /sys/open/open.a2d that are useful for modifying the A/D behavior.

Name	Type	Value[a]	Description
lowResGroup	INT	0	Group number for all non-IRGA channels
hiResGroup	INT	1	Group number for all IRGA channels
groupQueue	INT	1	Buffer size for accumulating readings
hiResUpdateTime	FLOAT	.5	Update time. New readings available every __ seconds. (Low and Hi resolution channels both use this period.)
hiResSPS	INT	200	Samples / second for group 1
lowResSPS	INT	50	Samples / second for group 0
hiResAvgTime	FLOAT	4	Running average periods (secs).
hiResUpdateTime	FLOAT	0.5	How often new readings are available

a. Don't change it if it's darkly shaded!

New Measurements

There are some variables that you can use to define function keys in New Measurements mode, or to change how often user computations are done (Table 26-3).

Table 26-3. Useful New Measurements variables, defined in /sys/open/open.msr

Name	Type	Value[a]	Description
ukey1, ukey2, ukey3, ukey4, ukey5	INT	31…35	Use for defining the five user defined function keys.
maxKeys	INT	35	Determines max # of function keys in New Msmnts mode.

Table 26-3. Useful New Measurements variables, defined in /sys/open/open.msr

Name	Type	Value[a]	Description
matchStableLimit	FLOAT	2	CO_2 range limit (ppm) for determining "stability" when auto-matching (during an AutoProgram).
matchCO2Limit	FLOAT	10	CO_2 adjustment limit for matching during an AutoProgram.
matchH2OLimit	FLOAT	1	H_2O adjustment limit for matching during an AutoProgram.
matchAutoTimeLimit	INT	60	How long (secs) to wait for stability during auto-matching.

a. Don't change it if it's darkly shaded!

Data Logging

Some variables pertaining to logging are presented in Table 26-4.

Table 26-4. Useful New Measurements variables, defined in /sys/open/open.log

Name	Type	Value[a]	Description
logBufferOpen	INT	0 or 1	0=logging inactive 1=logging active. (Do not set this variable!)
logBeepTime	INT	200	Duration of beep when logging, in milliseconds.
obsInPad	INT	0	Number of observations logged since destination opened. (System variable, #-35)
obsOneTime	Long	0	Time and date (seconds) destination was opened.

a. Don't change it if it's shaded!

Open's Hooks

OPEN provides a couple of methods to extend control via hooks. The most user-friendly is discussed in **Hooking Events** on page 16-37.

There is a second method, and that is to build the hook into LPL code, such as a ComputeList in Module format, and that is described here.

There are two approaches to take to make use of a hook.

1   Add a function with a "special" name to a ComputeList.
The safest way to take advantage of OPEN's hooks is put one or more

> **:include** *fileName*

directive(s) into your ComputeList file that will link module(s) containing functions to do the activities that you want. The names of these functions are very important, and Table 26-5 lists the required name along with what the default actions are.

Before a ComputeList is processed, all of these hooks are reset to default values (that's what makes this safe). Once a ComputeList is implemented, OPEN looks for functions in the ComputeList module whose names match the hook names. Each one found is substituted for the default function.

2   Direct pointer manipulation
A second method of using these hooks is to manipulate the hook directly, by reassigning the pointer associated with the hook. The relevant pointer names are also shown in Table 26-5.

Table 26-5. OPEN's Hooks

Pointer Name	When Called	Normally Points to	Looks for Name
HookConfigInit	When a ComputeList changes, just after it is compiled and implemented.	:FCT Nothing { }	*UserConfigInit*
HookMainDisplay	Called in Open's main screen to put the top three lines on the display.	StdMainDisplay	*UserMainDisplay*
HookWatchDog	Called every 10 seconds in New Measurements mode to check for problems (such as IRGA(s) Not Ready). See below.	WatchDog	*UserWatchDog*
HookNewMeasureEnter	Called each time New Measurements mode is entered from OPEN's main screen.	:FCT Nothing { }	*UserNMEnter*
HookNewMeasureStartup	Called each time New Measurements mode is re-started. This happens after HookNewMeasureEnter, and also after AutoPrograms finish.	StdNewMeasureStartup	*UserNMStartup*
HookUserKeys	Called in New Measurements mode after defining all the other function keys.	:FCT Nothing { }	*DefUserKeys*

Table 26-5. OPEN's Hooks

Pointer Name	When Called	Normally Points to	Looks for Name
HookNewReadings	Called in New Measurements mode right after new A/D readings are available and *ComputeSensors* has been called to compute all the measured quantities.	:FCT Nothing { }	*UserNewReadings*
HookPreComps	Called prior to doing the User Computations. (This is also before Extras expressions that are defined as Before).	:FCT CompPrep	*UserPreComps*
HookPostComps	Called just after the User Computations. (And after extra expressions defined as After).	:FCT Nothing { }	*UserPostComps*
HookEverySec	Called every 1 second during New Measurements mode.	:FCT EverySec	*UserEverySec*
HookNewMeasureExit	Called when exiting New Measurement to return to main screen. NOTE: Return value expected: 1 = ok to exit, 0 = do not exit.	:FCT Return1 { 1 }	*UserNMExit*

Some of the items in the WatchDog timer can be disabled by setting the appropriate flag to zero. The appropriate place to do this would be the *UserNMEnter* hook, for example. These flags are automatically reset (to 1) right before *UserNMEnter* is called.

Table 26-6. WatchDog flags (version 5.2 and above)

Flag	Enables
WDC_Fuse	**"BLOWN FUSE (Analyzer or Flow)"** on page 20-6
WDC_IRGA	**"IRGAs Not Ready"** on page 20-6
WDC_Humidity	**"High Humidity Alert"** on page 20-6
WDC_Pump	**"Pump is Off"** on page 20-7
WDC_Fan	**"Chamber Fan is Off"** on page 20-7
WDC_Flow	**"Flow is Too Low"** on page 20-7, and "Flow is low" in Match Mode (page 4-35).
WDC_Light	**"Negative PAR! LightSource? Cal?"** on page 20-8

27

Leaf Chamber Fluorometer

Using the 6400-40 Leaf Chamber Fluorometer

The 6400-40 LCF is an LED-based fluorescence and light source accessory for the LI-6400. This chapter guides you through installing and operating the LCF. The **Operational Summary** on page 27-17 will provide the information necessary to begin operating for those experienced with both fluorescence and the LI-6400. For the less experienced, Chapter 3 is recommended for familiarization with basic LI-6400 operation, followed by **Basic Experiments** on page 27-53.

Getting Started

If you are new to fluorescence, read **Background Information** on page 27-3. To get up and running with the LCF, follow these steps:

1 **Install the Hardware and Software**
If the LCF has not yet been installed on your LI-6400, work through **Installing the LCF** on page 27-11.

2 **Make sure the LCF is working properly**
Work through the **Basic Functionality Test** on page 27-84.

3 **Calibrate the light source**
This is described in **Calibrate...** on page 27-75.

Also note these important sections:

• **Operational Summary on page 27-17**
A summary of the controls, displays, and utilities available when configured for the LCF.

• **Changing Control Parameters on page 27-35**
A discussion of the various LCF settings, what they mean, and what are right for you.

• **Basic Experiments on page 27-53**
A step-by-step guide through some simple experiments, designed to acquaint you with proper operation of the LCF.

Background Information

What is fluorescence?

When a quantum of light is absorbed by a molecule of chlorophyll, the whole energy of the quantum is transferred to the valence electrons of the chlorophyll, raising them to an excited state. The electrons return rapidly to their ground level, releasing the absorbed energy in one of three pathways: 1) fluorescence, 2) heat, or 3) electron transport associated with photosynthetic photochemistry (Figure 27-1).

Figure 27-1. Chlorophyll fluorescence is one of the pathways of energy release following capture of a photon.

This relationship can be expressed as:

$$F + H + P = 1 \tag{27-1}$$

Where fluorescence (F), heat (H), and photochemistry (P) are each given as a fraction of the total absorbed quanta, which is assumed to be 1. P is also known as quantum yield or efficiency. As the light incident on a leaf increases, P decreases while H and F increase. At a saturating light intensity, there will be no further increase in photochemistry with any further increase in light intensity, so P will be zero. When this occurs, F and H will be at their maximal values, F_m and H_m, respectively. If we assume that all of the de-excitation is through heat and fluorescence, then Equation (27-1) becomes:

$$F_m + H_m + 0 = 1 \qquad\qquad (27\text{-}2)$$

so

$$H_m = 1 - F_m \qquad\qquad (27\text{-}3)$$

If we also assume that the ratio of heat to fluorescence de-excitation does not change during a short saturating flash

$$\frac{H}{F} = \frac{H_m}{F_m} \qquad\qquad (27\text{-}4)$$

then it follows that:

$$H = \frac{F(1 - F_m)}{F_m} \qquad\qquad (27\text{-}5)$$

By making two measurements of fluorescence (Figure 27-2), one (F) in non-saturating light conditions and the other (F_m) under saturating light conditions, we can solve for H in Equation (27-5) and P in Equation (27-1).

$$\begin{aligned} P &= 1 - F - H \\ &= 1 - F - \frac{F(1 - F_m)}{F_m} \end{aligned} \qquad\qquad (27\text{-}6)$$

This simplifies to

$$P = \frac{F_m - F}{F_m} \qquad\qquad (27\text{-}7)$$

If the non-saturating light condition is total darkness, and the leaf is completely adapted to that darkness, Equation (27-7) takes the form[1]

ID: 4202[2]

$$P_{dark} = \frac{F_m - F_o}{F_m} = \frac{F_v}{F_m} \qquad\qquad (27\text{-}8)$$

---

[1] We use the nomenclature put forth by VanKooten and Snel (1990). (References are listed in **Chlorophyll Fluorescence References** on page 27-93.)

[2] ID values refer to user defined variable numbers. See **Fluorescence ComputeList** on page 27-87.

where F_o is known as "minimal fluorescence", or the dark-adapted fluorescence value. P_{dark}, the fraction of absorbed photons that are used for photochemistry for a dark adapted leaf, is usually written F_v / F_m. For healthy plants, F_v / F_m is between 0.75 and 0.85. P_{dark} is usually referred to as the maximum or optimal quantum yield.

Figure 27-2. Basic fluorescence measurements. On dark-adapted leaves, we measure minimum F_o *and maximum* F_m *values. On illuminated leaves, we measure the steady state value* F_s, *the maximum value during a saturating flash* F_m', *and the minimum value during a dark pulse* F_o'.

By contrast, if the non-saturating light is non-zero, and the leaf is completely adapted to it (photosynthesis at steady-state), Equation (27-6) takes the form

ID: 4212

$$P_{light} = \frac{F_m' - F_s}{F_m'} = \frac{\Delta F}{F_m'} = \Phi_{PSII} \tag{27-9}$$

where F_s is "steady-state" fluorescence, and F_m' is the maximal fluorescence during a saturating light flash. P_{light}, the fraction of absorbed photons that are used for photochemistry for a light adapted leaf, is usually written Φ_{PSII} or $\Delta F / F_m'$ (Figure 27-2). P_{light} is usually referred to as the effective quantum yield.

A similar relationship that is sometimes used is

ID: 4211

$$\frac{F_v'}{F_m'} = \frac{F_m' - F_o'}{F_m'}$$

(27-10)

which is the efficiency of energy harvesting by oxidized (open) PSII reaction centers in the light. It requires F_o' which is the minimal fluorescence of a light adapted leaf that has momentarily been darkened.

Quantum yield can also be inferred from gas exchange measurements, and is given the symbol Φ_{CO_2}.

ID: 4212

$$\Phi_{CO_2} = \frac{A - A_{dark}}{I\alpha_{leaf}}$$

(27-11)

where A is assimilation rate, A_{dark} is dark assimilation rate (both with units of μmol CO_2 m^{-2} s^{-1}), I is incident photon flux density (μmol m^{-2} s^{-1}), and α_{leaf} is leaf absorptance. A_{dark} is the same magnitude, but opposite sign, of dark respiration rate.

The actual flux of photons (μmol m^{-2} s^{-1}) driving photosystem II (PS II) can be inferred from chlorophyll fluorescence measurements. This is called electron transport rate (*ETR*), and is given by

ID: 4223

$$ETR = \left(\frac{F_m' - F_s}{F_m'}\right) fI\alpha_{leaf}$$

(27-12)

f is the fraction of absorbed quanta that is used by PS II, and is typically assumed to be 0.5 for C3 plants, and 0.4 for some C4 plants like maize (Earl and Tollenaar, 1998).

Photochemical quenching q_P of fluorescence can be computed from

ID: 4216

$$q_P = \frac{F_m' - F_s}{F_m' - F_o'} \qquad (27\text{-}13)$$

Photochemical quenching includes photosynthesis and photorespiration, and tends to be greatest in low light, since that's where leaves use light most efficiently. Non-photochemical quenching q_N of fluorescence includes mechanisms such as heat dissipation, and is computed from

ID: 4220

$$q_N = \frac{F_m - F_m'}{F_m - F_o'} \qquad (27\text{-}14)$$

q_N is highest when light intensities are high, perhaps reflecting a plant protection mechanism to avoid over-energization of the thylakoid membranes. A similar measure of non-photochemical quenching that is also reported is

ID: 4221

$$NPQ = \frac{F_m - F_m'}{F_m'} \qquad (27\text{-}15)$$

Some researchers prefer to use F_o rather than F_o' in the calculation of q_p and q_N.

ID; 4231

$$q_P\big|_{F_o} = \frac{F_m' - F_s}{F_m' - F_o} \qquad (27\text{-}16)$$

ID: 4232

$$q_N\big|_{F_o} = \frac{F_m - F_m'}{F_m - F_o} \qquad (27\text{-}17)$$

The LI-6400 computes these alternative forms as well (starting in OPEN version 5.2), but by default does not display or log them. The user can add or replace them on the display and in the log file as desired. See **Display Editor** on page 6-6, and **Determining What is Logged** on page 9-8.

What is the 6400-40 LCF?

The LCF is an LED-based fluorescence/light source attachment for the LI-6400. It contains a variety of LEDs (3 blue, 1 far red, and the rest red) and two detectors (Figure 27-3). Here is an overview of how it works:

Figure 27-3. View of the LEDs of the LCF, showing the two red modulated measuring beam LEDs (M), the three blue actinic LEDs (B), and the far red LED (F). The remaining LEDs are red, and are used for actinic and saturating flashes.

Figure 27-4. Relative spectral outputs of the three types of LEDs used in the LCF: a) Actinic, b) actinic, measuring, and saturation flashes, c) far red.

Measuring Fluorescence

The LCF uses two red LEDs (center wavelength about 630 nm - see Figure 27-4 on page 27-8) and a detector to measure fluorescence. The LEDs are modulated (turned on and off) very rapidly, at your choice of 0.25, 1, 10 or 20 kHz. This modulated red light is referred to as "the measuring light". The resulting oscillation of incident radiation causes an oscillation in fluorescence which is detected by the LCF.

Why the range of modulation frequencies? There is a trade-off: at the lowest frequency, the photosynthetically active radiation being contributed by the measuring beam is much less than at high frequencies, because the LEDs are being turned on relatively infrequently - only 250 times per second. When measuring minimal fluorescence in a dark adapted leaf (F_o in Equation 27-8 on page 27-4), it is important that the measuring light not induce photosynthesis in the leaf. The problem with 250 Hz, however, is that there are only 250 measurements of fluorescence being made each second, so the averaged value is relatively noisy due to the limited number of points. By contrast, during a saturating flash one doesn't care about contributions of the measuring beam to photosynthesis, so 20KHz modulation can be used. This provides 20,000 samples per second, and the resulting averaged fluorescence signal is much quieter.

The LCF also has software selectable filtering (averaging) of the fluorescence signal. The choices are bandwidths of 0.5, 1, 5, 10, 20, 50, 100, and 200 Hz. Most of the time, 0.5 or 1 Hz can be used, but during saturating flashes when the fluorescence signal is changing fast, 20 or 50 Hz bandwidths are preferred.

Saturating Flashes

Maximal fluorescence, which is measured during a brief period when the photosystem is light saturated, is a key element of fluorometry, and the LCF achieves these light levels ($> 7000 \, \mu$mol m^{-2} s^{-1}) by using 27 red LEDs (center wavelength: 630 nm. See Figure 27-4 on page 27-8). PAR levels during saturating flashes are measured by a calibrated light sensor within the LCF.

Actinic Light

"Actinic" refers to the light provided by the LCF for purposes of driving photosynthesis. The LEDs used for providing actinic radiation are the same ones used for saturating flashes, plus 3 blue LEDs (470 nm). As a light source, the LCF functions much the same as the 6400-02B Red/Blue light source, but with an interesting additional capability: the red and blue are independently controlled. The maximum blue quantum flux achievable is typically about 200 μmol m^{-2} s^{-1}, so the maximum blue fraction can be about 10% of full

Leaf Chamber Fluorometer
Background Information

sun. The LCF's built-in light sensor can be used to control in-chamber PAR levels to specified target values.

Rapid Dark Adaptation
Measuring the minimal fluorescence of a light adapted leaf involves turning off the actinic light briefly while using the far red LED (center wavelength at 740nm - see Figure 27-4c). We refer to this event as a "dark pulse". The far red radiation drives PS I momentarily to help drain PS II of electrons.

Status LEDs
The 37-pin connector for the LCF has 4 status LEDs that light up when the corresponding LEDs in the LCF are illuminated. The exception to this is during a saturating flash, when all four LEDs are illuminated. Also, just before and after a flash, the four status LEDs briefly flicker.

Figure 27-5. The LCF Status LEDs

Installing the LCF

Hardware

The LI-6400 should be off (or in sleep mode) to do this procedure.

1   Install cable in bundle

You have a choice on how to do this. The easiest method is to lash the LCF cable to the LI-6400 cable bundle with the velcro ties found in the LCF spares kit (Figure 27-6a). However, if you are going to be using the LCF frequently, just insert the LCF cable into the bundle (Figure 27-6b). This latter approach takes a few minutes of work, but is more convenient in the long run.

A)

Velcro straps

Either attach the LCF cable to the cable bundle with velcro...

B)

...or insert the LCF cable into the cable bundle sheath. (This operation is rather like a boa constrictor swallowing a large pig: it may seem impossible, but can be done with patience and persistence.)

LCF Cable
15-pin connector

Figure 27-6. The LCF cable can be attached with velcro (A), or inserted into the bundle (B).

2   **Remove the upper and lower chambers from the sensor head**
Use 3/32 hex key provided in the spares kits (LI-6400 and the LCF) to remove the two long screws that hold the chamber top and the chamber bottom in place. Disconnect the chamber top light sensor, and the leaf thermocouple from the lower chamber (Figure 27-7).

Figure 27-7. Remove the chamber top and bottom using a 3/32" hex key

3   **Attach the lower part of the LCF**
Make sure the three o-rings are in place on the LCF bottom chamber, and attach it to the sensor head. Install the leaf thermocouple (Figure 27-8)

4   **Attach upper part**
Make sure the o-rings are in place, and attach the LCF main body.

5   **Connect the cables**
Plug the small 4-pin connector into the sensor head, and connect the 15-pin LCF connector to its cable. The other end of this cable has a 37-pin connector, and that plugs into the LI-6400 console.

.

Make sure the 6 o-rings are in
place (3 upper, 3 lower)

Figure 27-8. The LCF lower chamber attached and the upper chamber ready.

Power for fan
and red LEDs

*Figure 27-9. The LCF attached to the sensor head. The main cable can be routed
behind the quantum sensor, and through the tripod mount (remove the mount to do
this).*

Software

It is necessary to create a fluorometer configuration for subsequent use, and also create an actinic light source calibration file.

■ **To create a Leaf Chamber Fluorescence configuration**

1   Access New... in the Config Menu
Press **f2** from OPEN's main screen to access the Config Menu. Highlight New..., and press **f5** (**select**) or **enter** (Figure 27-10).

Figure 27-10. The starting point for any new configuration.

2   Select 6400-40 LCF under Light Sources
Highlight the Light Sources node and press **f1** to expand it, then scroll down to the 6400-40 entry, and press **f5** (**select**) or **enter** (Figure 27-11).

Figure 27-11. Building an LCF configuration.

3   The LCF Configuration Dialog
The LCF configuration dialog (Figure 27-12) has three items you can edit to suit your requirements.

Figure 27-12. The type of LCF configuration.

4   LCF Used as=
The choice is Fluorometer or LightSource. If you pick the latter, you will only be able to set actinic light; no fluorometry measurements will be done.

5   Material=
The choice is shown to the right. You probably want broadleaves, but the implications of these choices is discussed in **The Material= Node** on page 16-51.

```
════Select Material════
 Broadleaves
 Needles
 Mass-based
```

6   LeafTemp=
(Not an option if Material = Mass-based). Selects how leaf temperature is measured, either with a thermocouple or by energy balance. (If Material = Needles, then this item will automatically be energy balance.)

7   Make a new configuration file
When the configuration dialog is set to your liking, press **f5** (**DoIt**), and   the display will show Figure 27-13. Press **N**.

```
Config Modifications Ready. Pick one:

N) implement as a New config named:
   'LCF'
C) apply to the Current config
V) View the modifications
E) return and Edit your settings
<escape> - discard and quit
```

Figure 27-13. If building a new configuration file, press **N**.

8   Communication with the LCF
At this point, OPEN will implement the configuration, and attempt to communicate with the LCF. If it has a problem doing this, you will see the error message illustrated in Figure 27-14.

```
AHs/Cs - ok

Linking /Sys/Open/FlrTools...00 00 00 00

Erroneous Block 0 ('')
Writing 'ABCD' to Block 0
00 00 00 00 Read '' from Block 0
LCF not responding - Try again? (Y/N)
```

Figure 27-14. Communications error.

If this is happening to you, make sure the cables are connected properly, and press **Y**. If it still doesn't work, press **N** to finish the configuration, and deal

with it later.

9   **Save the configuration file**
You'll be prompted to save the file, and given a suggested default name
(Figure 27-15). Modify it as you wish, and press **enter**. Note that the file name
will always have a .xml appended to it automatically, unless you add it your-
self.

```
Dir: /User/Configs/UserPrefs
File:LCF
Filter: ver 6 config files (*.xml)
══════════════Save this config══════════════
  2x3 Flrscnt ClrBtm EB.xml
  2x3 Flrscnt Mass-based ClrBtm.xml
  2x3 Flrscnt Mass-based.xml
+ Dir    ♦      ♦ Filter♦ CANCEL♦ SELECT
```

Figure 27-15. Saving the configuration.

10   **Return to the Main Screen**
Escape back to OPEN's main screen.

11   **Do an Actinic Calibration**
See **Calibration Issues** on page 27-73.

Operational Summary

Figure 27-16 summarizes the changes to the New Measurements mode displays and function keys associated with the LCF.

Figure 27-16. Summary of the New Measurements displays and function keys added by the default fluorescence configuration. Display explanations are in Table 27-2 on page 27-28, and the new function keys are described in Table 27-3 on page 27-31

The LCF as a Light Source

Note

> The light source control options described here are available whenever the LI-6400 is configured for operating the LCF. That is, when the light source is specified as the 6400-40 LCF. If you specify the 6400-40 as the light source by editing the configuration tree (**Specifying the Light Source** on page 8-4), you are given a choice of using the LCF as a fluorometer, or as a light source only. If you choose the latter, only the light source control options are available in New Measurements mode, and all of the fluorometer controls (function key levels 8, 9, and 0) are hidden.

Caution

> Do not operate the LCF by specifying it in the Light Source Control panel <u>as a 6400-02 Red/Blue Source</u>. This configuration will light the LEDs, but no internal light sensor feedback will be measured, and the LCF could operate, unbeknownst to you, for extended periods at full intensity. This will rapidly reduce its maximum flash potential.

When using the LCF, actinic light is set in New Measurements mode via the lamp control panel, accessed by **f5** level 2. Red and Blue LEDs are independently controlled, allowing operation with a blue value that is either specified in (μmol m^{-2} s^{-1}) or as a proportion of the total. In addition, there is a function key that immediately turns the lamp on and off (**Actinic is ON/OFF**, **f4** level 9), and one that pre-sets the desired actinic target without actually turning it on (**Define Actinic**, **f3** level 8).

The control panel for the light source when configured for the LCF (Figure 27-17) presents the usual options: PAR, Control Signal, and Ambient Tracking.

*Figure 27-17. The lamp control panel (**f5** level 2) for the LCF.*

Because there is adjustable blue in the LCF, the PAR settings have a couple of blue options (Figure 27-18) that determine how much blue there will be.

Figure 27-18. Controlling how much blue light there is.

With tracking ambient PAR, the blue option is automatically set to "percentage".

When setting the target for *Control Signals*, there are two target values. One can enter both of these values, separated by a space or comma, to set both targets. If only one number

```
┌─Red  mV,   Blue  mV─┐
│                     │
│ _                   │
│ └─1000,1000─────────┘
```

is entered, it is taken to be the first target (total, or red). If the entry is a comma followed by one value, that value is used for the second target (blue) (Table 27-1).

Table 27-1. Entering 2-value targets.

Entry	Red Target (mV)	Blue Target (mV)
500, 50	500	50
800 2000	800	2000
1000	1000	unchanged
,5000	unchanged	5000

When configured for *Track Ambient PAR*, the target value comes from an external sensor or user defined channel. Normally, one would choose the external quantum sensor for the target, but if one wanted the light value to be two times larger than ambient, for example, one could set up a user defined variable that computes such a quantity, and use that as the variable to be tracked. *Track Ambient* mode also sets *Blue portion = proportional*, and the user specifies the desired blue percentage.

Measuring Fluorescence

When the fluorescence measuring beam is turned on (**f1** level 9), the raw fluorescence signal can be viewed on display line *n*, under the label *F*.

Real time, raw fluorescence signal. Maximum value is about 8000.

Turns measuring beam ON/OFF.

Figure 27-19. Basic fluorescence monitoring and control

When the chamber is empty, or when the measuring beam is turned off, *F* should be near 0 (say, +/- 5). If it is not, the zeroing routine should be executed (**Zero Fluorescence Signal** on page 27-81). The maximum value that *F*

can be is typically about 8000. (Technical aside: The raw output is generally about -3000mV, and is being measured on a -5V to +5V channel, hence the 8000 maximum. The final *F* value is the raw signal plus an offset.)

Saturating Flashes

The LI-6400 (version 6.0 and above) supports two types of saturating flash: **rectangular** and **multiphase** (Figure 27-20). A rectangular flash provides a saturating pulse of light for a short duration, and the software looks for the maximum value of fluorescence during that pulse. With this method, one assumes that the pulse is strong enough to actually saturate, although this is testable with a simple experiment (see **Optimum Flash Intensity** on page 27-77). The multiphase flash uses varying intensities during the flash, and computes the fluorescence at saturation from the y-intercept of a plot of fluorescence vs. 1/intensity. There is a proper protocol for using the multiphase flash: see **Using the MultiPhase Flash** on page 27-40.

Figure 27-20. Two types of saturating flashes: a) rectangular and b) multiphase with c) regression analysis.

There are several ways to trigger a saturating flash (Figure 27-21). If the leaf is dark adapted and you wish to get a new value for the system variable *Fm* (maximal fluorescence for a dark adapted leaf), press **Do Fm** or **Do FoFm** (**f2** or **f3** level 0). If the leaf is light adapted and you wish to set the value of system

variable *Fm'* (maximal fluorescence of a light adapted leaf), press **Do FsFm'** or **Do FsFm'Fo'** (**f3** or **f4** level 0). To do a flash without assigning any values, press **Flash** (**f2** level 9).

> WARNING: Saturating flashes can be extremely bright, and harmful to the eye if the LEDs are viewed directly.

*Figure 27-21. A saturating pulse (flash) can be triggered from function key level 9 (**f2**) or two keys on level 0 (**f3** and **f2** or **f4** depending on actinic). They only differ in how (or if) the maximal fluorescence value is used.*

Dark Pulses

A "dark pulse" (brief dark period) is designed to measure F_o' on a light adapted leaf. Figure 27-22 illustrates the timing parameters.

Figure 27-22. Illustration of the timing parameters for a dark pulse. The far red LED comes on PreTime *seconds before the actinic light goes off, and remains on for* PostTime *seconds after. The actinic is off for* DarkDuration *seconds.*

Figure 27-23 illustrates how to trigger a dark pulse.

Figure 27-23. A dark pulse is triggered by one of three function keys when actinic is on.

Viewing Flash and Dark Pulse Details

Once a saturating flash or dark pulse has been applied, you may view the flash shape and details via View Fsh/Drk, **f5** level 0 (Figure 27-24).

Figure 27-24. Viewing flash events.

A summary of the flash viewer functions is shown in Figure 27-25 on page 27-25. The flash viewer screen shows the saturating flash file name, which includes the flash number, type (*RF* = rectangular, *MPF* = multiphase), the maximum light intensity (*Qmax*) and maximum fluorescence (*Fmax*) during the flash. If an *MPF* flash is viewed, a *Check* value is also displayed. This value is the difference between the maximum fluorescence reached during the third phase of the *MPF* flash, with what would be predicted using the second phase's regression and the third phase's maximum light. If the flash settings are appropriate, this difference should be within several mV of zero. If a dark pulse is viewed, *Fmin* is the min value of fluorescence.

view Graph: To view the entire flash, press **G** (the shortcut for the **view Graph** key, **f1**). To view the top ten percent of the graph, escape back to the viewer, and press **T** (the shortcut key for the **zoom topTen%** key, **f1** level 2). For an even closer view, press **O** (the letter) from the viewer, the short cut for **zoom topOne%**, **f2**, level2).

view Details: The saturating flash summary details and data can be viewed by pressing **D**, (for **view Details**, **f2**).

view Regrs: The phase 2 regression of the MPF can be viewed by pressing **R**, the short cut for the **view Regrs** key, **f3**.

Save: The flash or dark pulse details and data may be saved (if they haven't been already) by pressing **S** (short cut for **Save**, **f4** level 2). These files are saved by flash number, date, and time stamp in the /User/LCF directory.

Pick event: You may load in previously saved flashes or dark pulses by pressing **P**, the short cut for **Pick event**, **f4**.

See also **LI6400XTerm and the LCF** on page 27-48 for other view options.

Rectangular Flash (RF)

```
=== LCF Flash and Dark Pulse Viewer ===
Latest: F#004 2008-11-06 (Not Saved)
Type: Square Flash
Qmax: 5682
Fmax: 2385
```

view Graph	view Details		Pick event	Quit
zoom topTen%	zoom topOne%		Save	

MultiPhase Flash (MPF)

```
=== LCF Flash and Dark Pulse Viewer ===
File: /User/LCF/FMP#002 2007-10-23
Type: Multiphase Flash
Qmax: 7807
Fmax: 1990 Intercept: 1993.1
Check: 813.6
```

view Graph	view Details	view Regrs	Pick event	Quit
zoom topTen%	zoom topOne%			

view Graph (**f1**) or **G**

view Graph (**f1**) or **G**

zoom topOne% (**f2**) or **O**

zoom topOne% (**f2**) or **O**

view Details (**f2**) or **D**

```
=======Flash Details=======
Flash=  1
Time=   2007-10-23 13:26:13
LCF=    LCF-0481
Duration=      0.8
%Ramp=  0
Fmax=   2865
  Print ♦ Find  ♦ ReFind♦ JumpTo♦ OK
```

view Details (**f2**) or **D**

```
=======Flash Details=======
Flash=  2
Time=   2007-10-23 13:48:40
LCF=    LCF-0481
Duration=      0.7
%Ramp=  40
Fmax=   1990
  Print ♦ Find  ♦ ReFind♦ JumpTo♦ OK
```

view Regrs (**f3**) or **R**   (MPF only)

Figure 27-25. Viewing Rectangular and MultiPhase flashes on the LI-6400.

```
Flash=     1
Time=      2007-10-23 13:26:13
LCF=       LCF-0481
Duration=0.8
%Ramp= 0
Fmax=      2865
Qmax=      7747
RampAnalysis=None
RedTarget=8
Modulation=20
Filter=    50
MeasIntensity=3.0
Time     PAR     Flr
0.000    3       721.9
0.010    3       722.9
0.020    3       723.1
0.030    4       722.5
0.040    3       722.0
0.050    3       723.0
0.060    3       721.9
0.070    2       722.0
0.080    3       722.4
0.090    5       722.6
0.100    2       722.2
0.110    1       722.5
0.120    0       723.3
0.130    3       722.3
0.140    5       722.6
0.150    4       721.0
0.160    3       722.2
0.170    2       722.8
0.180    1031    912.9
0.190    4694    1864.1
0.200    6819    2368.7
0.210    7422    2615.2
0.220    7625    2745.2
0.230    7698    2804.4
0.240    7721    2831.1
0.250    7727    2851.4
0.260    7727    2857.3
0.270    7732    2865.0
...
```

The line shows PAR values.

The dots are fluorescence values.

Figure 27-26. A rectangular saturating flash, graphed and stored.

```
Flash=    8
Time=     2008-09-18 11:06:35
LCF=      LCF-0304
Duration=0.8
%Ramp= 30
Fmax=     854
Qmax=     5988
RampAnalysis=ok
Phase1_ms=300
Phase2_ms=200
Phase3_ms=300
NumPoints=21
Intercept=988.1
SE_Intrcpt=4.0
Slope=    -81.69
SE_Slope=2.10
R2=       0.987641
ramp_dQ/dt=-7839
FC_Q=     5962.2
FC_F=     853.9
FC_Pred= 851.0
FC_DeltaF=2.9
RedTarget=9
Modulation=20
Filter=   50
MeasIntensity=3.0
Time    PAR    Flr    1E4/PAR  Phase
0.000   526    360.8  18.997   0
0.010   526    361.0  18.997   0
0.020   526    360.9  18.997   0
0.030   528    360.1  18.94    0
0.040   526    359.3  18.997   0
0.050   524    359.5  19.084   0
0.060   521    360.1  19.2     0
0.070   520    360.9  19.23    0
0.080   522    361.0  19.142   0
0.090   524    360.1  19.084   0
0.100   525    361.5  19.055   0
0.110   523    361.4  19.113   0
0.120   518    360.5  19.318   0
0.130   524    361.3  19.084   0
0.140   530    360.9  18.855   0
0.150   526    359.7  19.026   0
0.160   759    364.0  13.169   0
0.170   3423   500.8  2.9217   0
0.180   5245   659.5  1.9066   1
0.190   5798   726.3  1.7247   1
0.200   5951   759.7  1.6804   1
...
```


Slope and intercept of regression

Phase of the flash (1, 2, and 3)

Start of Phase 1

Figure 27-27. A MultiPhase flash, graphed and stored.

Display Summary

Table 27-2 summarizes the New Measurements display groups added by the display configuration file "Std Flr Disp".

Also, the default display grouping[3] defined by "Std Flr Disp" is

Key	Displays
home	a, b, c
end	k, e, i
pgup	n, p, r
pgdn	g, h, d

Table 27-2. New Measurement displays added by "Std Flr Disp"

Label	Description	[a]ID #
g	`Prss_kPa` `ParIn_µm` `%Blue` `ParOutµm`	
%Blue	An estimate of the percentage of *ParIn_µm* coming from the blue actinic LEDs.	-81
n	`F` `dF/dt` `FlrEvent`	
F	Fluorescence intensity. Measured continually.	-80
dF/dt	The rate of change (per minute) of the fluorescence signal over the past 10 seconds.	-98
FlrEvent	Indicates the last action performed or variable that was set.	-91
o	`Fo` `Fm` `Fv/Fm`	
Fo	Minimal fluorescence (dark). This value is set by pressing **Do Fo** or **Do FoFm** (**f1** or **f3** level 0).	-86
Fm	Maximal fluorescence (dark). Set by pressing **Do Fm** or **Do FoFm** (**f2** or **f3** level 0).	-88
Fv/Fm	Maximum PSII efficiency. Variable to maximal fluorescence (dark). Equation (27-8) on page 27-4.	4202

---

[3].See **Display Groups** on page 3-19.

Table 27-2. New Measurement displays added by "Std Flr Disp" (Continued)

Label	Description	[a]ID #
p	**Fo'** **Fm'** **Fv'/Fm'** **Fs**	
Fo'	Minimal fluorescence (light). This value is set by pressing **Do Fo'** or **Do FsFm'Fo'** (**f3** or **f4** level 0).	-87
Fm'	Maximal fluorescence (light). This value is set by pressing **Do FsFm'** or **Do FsFm'Fo'** (**f3** or **f4** level 0).	-89
Fv'/Fm'	Variable to maximal fluorescence (light). Equation (27-10) on page 27-6.	4211
Fs	Steady-state fluorescence. Set by pressing **Do FsFm'** or **Do FsFm'Fo'** (**f3** or **f4** level 0).	-90
q	**PhiPS2** **ETR** **qP** **qN**	
PhiPS2	PS II efficiency. Equation (27-9) on page 27-5.	4212
ETR	Electron transport rate. Equation (27-12) on page 27-6.	4223
qP	Photochemical quenching. Equation (27-13) on page 27-7.	4216
qN	Non-photochemical quenching. Equation (27-14) on page 27-7.	4220
r	**Adark** **LeafAbs** **PS2/1** **PhiCO2**	
Adark	Dark photosynthetic rate (μmol CO_2 m^{-2} s^{-1}). (Sign convention: a value < 0 indicates respiration). Set by pressing **Prompt All** (**f5** level 3), or **Flr Editor** (**f2** level 8).	4213
Leaf Abs	Leaf Absorptance. Computed based on the fraction of blue light, and two user-entered (**Prompt All** or **Flr Editor**) absorptances: one for blue, and one for red. Equation (27-18) on page 27-88.	4207
PS2/1	Photosystem distribution factor. Set by pressing **Prompt All** (**f5** level 3). This is f in Equation (27-12) on page 27-6.	4222
PhiCO2	Equation (27-11) on page 27-6.	4215
s	**F M:Int kHz Hz Gn**	
F	Fluorescence intensity. Measured continually.	-80
M: Int … Gn	Measuring beam parameters: intensity, modulation, filter, gain. See Table 27-5 on page 27-38.	-92

Table 27-2. New Measurement displays added by "Std Flr Disp" (Continued)

Label	Description	[a]ID #
t	`FlrMax    RF Dur Int   kHz   Hz`	
FlrMax	Maximal fluorescence obtained during the previous saturating flash.	-82
RF: Dur … Hz MPF% ... Hz	Saturation flash parameters (Table 27-6 on page 27-38): duration (for RF) or percent and phase timing (for MPF), plus intensity, modulation, and filter.	-93
u	`FlrMin   D:Dur Far Bfr Aft   kHz   Hz`	
FlrMin	Minimum fluorescence value during the previous dark pulse.	-95
D: Dur … Hz	Dark pulse parameters (Table 27-7 on page 27-39): duration, pre-time, post-time, modulation, filter.	-94
Other variables available for display		
FPeak_μm	Maximum quantum flux generated by the previous saturating flash, in μmol m^{-2} s^{-1}.	-83
FCnt	Number of saturating flashes performed since power on.	-84
Fzero	The fluorescence value achieved with the measuring intensity set to 0. This is used as an offset to compute F from the actual measured value. To set this, see **Zero Fluorescence Signal** on page 27-81.	-85
ParIn@Fs	The measured value of ParIn_μm when Fs was last set. This is used to keep the electron transport computation from changing because of subsequent changes in ParIn_μm before Fs is measured again.	-96
FlrCV_%	Flr CV (%) over the past 10 seconds.	-97
Fmean	Flr mean over the past 10 seconds.	-115
BlueAbs	Leaf absorptance at 460 nm. User entered constant. See **Leaf Absorptance** on page 27-88.	4205
RedAbs	Leaf absorptance at 640 nm. User entered constant. See **Leaf Absorptance** on page 27-88.	4206
PARAbs	Absorbed PAR. ParIn_μm * LeafAbs.	4208
NPQ	Alternative non-photochemical quenching. Equation (27-15) on page 27-7.	4221

a. ID values < 0 are system variables, and ID values > 0 are user variables.
See **List of Open 6.2 System Variables** on page 14-23, and **Fluorescence ComputeList** on page 27-87.

Function Key Summary

Table 27-3 summarizes the New Measurements function keys enabled when the light source is set to "6400-40 LCF".

Table 27-3. Summary of LCF function keys, New Measurements mode

Label	Description
8 **Flr Import** **Flr Editor** **Define Actinic** **Msr Adjust** **Rcrding OFF**	
Flr Import	Brings up a list of files with saved fluorometer settings. Selecting one of those files implements the settings in that file.
FlrEditor	Allows measurement, flash, and dark pulse settings to be viewed, stored, recalled, etc. See **Flr Editor** on page 27-35.
Define Actinic	Set actinic light, without turning on the LEDs. See **The LCF as a Light Source** on page 27-18.
FlrAdjust	Allows measurement, flash, and dark pulse settings to be changed on the fly, without interrupting measurements. See **Flr Adjust** on page 27-37.
Rcrding ON/OFF	Toggles recording of *F* and time to a data file on/ off. This is independent of normal data logging (f1 level 1). See **Fluorescence Recording** on page 27-33.
9 **Meas is ON** **Flash** **Dark Pulse** **Actinic is OFF** **FarRed is OFF**	
Meas is ON/OFF	Toggles the measuring LEDs on/ off.
Flash	Triggers a saturating flash, but only updates *FlrMax*.
Dark Pulse	Triggers dark pulse routine, but does NOT set *Fo* or *Fo'*.
Actinic	Toggles actinic light on/ off.
Far Red is ON/OFF	Toggles far red light on/ off.
0 **Do Fo'** **Do Fs** **Do FsFm'** **Do Fs Fm'Fo'** **View Fsh/Drk**	(Actinic ON)
Do Fo'	Does dark pulse, and sets *Fo'*, then logs (if data file open).
Do Fs	Sets *Fs* to the current value of *F* and logs (if data file open).

Table 27-3. Summary of LCF function keys, New Measurements mode (Continued)

Label	Description
Do FsFm'	Sets Fs, then does a flash to get Fm' then logs (if data file open).
Do Fs Fm'Fo'	Sets Fs, then does a flash to get Fm', then a dark pulse to get Fo', then logs (if data file open).
View Fsh/Drk	Allows user to view and/ or save last flash, ParIn during flash, or dark pulse. See **Viewing Flash and Dark Pulse Details** on page 27-24.

					(Actinic Off)
0	Do Fo	Do Fm	Do FoFm		View Fsh/Drk

Label	Description
Do Fo	Sets *Fo* to the current value of *F*, then logs (if data file open).
Do Fm	Does flash and sets *Fm*, then logs (if data file open).
Do FoFm	Sets *Fo* to the current value of *F*, does flash, then sets *Fm*, and logs (if data file open).

Logging Considerations

Normal Data Logging

1) Log files are opened by pressing **f1** level 1, or by launching an AutoProgram (**f1** level 5).
2) The user has control over what parameters are stored in the log file (see **Determining What is Logged** on page 9-8.).
3) Logging occurs automatically with an AutoProgram or manually by pressing **Log** (**f1** level 1).

When the LCF is installed, there are some additional capabilities:

1　**There are 4 additional keys that will trigger a Log**

The "Do..." keys on level 0 all trigger logging if a log file is open. For example, **Do FoFm** will log a data record - just like pressing **Log** (**f1** level 1) - but it is done after setting *Fo*, doing a flash, and setting *Fm*.

If you are wondering what happens to the gas exchange values when a saturating flash or dark pulse occurs, and how meaningful values of both can be logged simultaneously, then you'll want to read **Simultaneous Gas Exchange and Fluorescence** on page 27-90.

2   Fluorescence events log remarks

When a log file is open, a remark is logged whenever some fluorescence event occurs, either due to an AutoProgram or manually. These events are summarized in Table 27-4.

Table 27-4. Remarks logged by fluorescence events

Event	Example remark
Fo set	"14:10:32 Fo=225"
Fo' set	"14:10:32 Fo'=815"
Fm' set	"14:10:32 Fm'=1550"
Fm set	"14:10:32 Fm=1760"
Fs set	"10:51:46 Fs=1284 (3 0.25 5 10)"
MPF Flash	"10:52:15 MPF#3 5825 um, Fmax=1360 Int=1404 +/- 6 Slp=-11 +/- 3 Rmp=20 P1=300 P1=300 P3=300 Int=9 Mod=20 Filter=50"
RF Flash	"10:51:46 RF#2 5833 um, Fmax=1510 Int=9 Mod=20 Filter=50"
Dark pulse	"10:52:24 Dark#2 Fmin=464 Dur=6 FarRed=8 Pre=1 Post=4 Mod=0.25 Filter=5"

An example of what the resulting log file looks like can be found in Figure 27-73 on page 27-92.

These remarks can be routed to a separate file. See **Log Options** on page 9-14.

Fluorescence Stability

When doing fluorescence, it is a good idea to include the fluorescence signal (id = -80) in your stability criterion, especially for AutoPrograms. Even if you don't, however, OPEN keeps its own statistics on the fluorescence signal over a 10 second[4] period, including the mean (*Fmean*, id = -115), coefficient of variation as a percentage (*FlrCV_%*, id = -97), and rate of change (*dF/dt*, id = -98). *dF/dt* is on a per minute basis, and is viewed on display line *n*.

Fluorescence Recording

Often it is convenient to record fluorescence as a function of time over the course of an experiment. Fluorescence recording is toggled on and off by **Rcrding On/Off** (**F5** level 8). When fluorescence recording is first turned on, you are prompted for a destination file. While recording remains on, the file records the raw fluorescence signal and times while you are in New Measurements mode. The data is typically spaced every second or two, except during

---

[4.]This is adjustable. See *Period* in Table 27-8 on page 27-39.

saturating flashes and parts of dark events, when the spacing is 0.01 seconds (see next paragraph). There will be gaps in the data during those times in New Measurements mode when you are being prompted for input from the keyboard, such as when launching an AutoProgram, or changing leaf area.

To save the detailed fluorescence data from saturating flashes and/or dark pulses separately (whether fluorescence recording is on or not), use the <Flash> <autosave> and <DarkPulse> <autosave> nodes in the <FlrParams> configuration tree (**Flr Editor**, **f2** level 8). Otherwise, you can save it manually in the Flash / Dark Pulse Viewer, **View Fsh/drk** (**f5** level 0).

A time stamp is written whenever the file is opened or appended, and corresponds to the time of the first observation following it.

The tab-delimited columns are time (s) since power on and Fluorescence.

Flash event (0.01 s data spacing)

```
"Thr Dec 29 2011 10:43:44"
"Time"    "Flr"
467.50    1271.8
468.00    1269.0
468.50    1266.5
469.00    1263.9
469.36    1260.9
469.37    1260.5
469.38    1260.9
469.39    1260.6
469.40    1260.2
469.41    1260.7
469.42    1260.4
469.43    1260.6
469.44    1260.1
469.45    1261.2
469.46    1260.3
469.47    1259.6
469.48    1259.8
469.49    1259.9
469.50    1260.1
469.51    1260.3
469.52    1260.2
469.53    1259.2
469.54    1260.3
469.55    1347.3
469.56    1978.7
469.57    2319.4
469.58    2499.3
  :
```

Figure 27-28. Example of a Fluorescence Recording file.

Changing Control Parameters

The 6400-40 LCF has a number of control parameters, such as measuring beam intensity, flash intensity, and dark pulse duration. They are summarized in Table 27-5 on page 27-38. There are three groups of parameters:

1 **Measuring**

The intensity and modulation frequency of the measuring beam are user controlled. The resulting fluorescence signal coming back from the leaf is averaged with a user-defined bandwidth.

2 **Flash**

Saturating flashes are performed with the red actinic LEDs. The intensity and duration of the flash are user defined, as well as the fluorescence modulation and measuring filter used during the flash.

3 **Dark**

The dark pulse is designed to provide a brief interruption of actinic light to a light equilibrated leaf, in order to measure *Fo'*. The parameters for timing and measuring are user defined, and illustrated in Figure 27-22 on page 27-23.

There are several ways to adjust these control parameters from the keyboard[5]:

Flr Editor

The Flr Editor (Figure 27-29 on page 27-35) is accessed via **f2** level 8. This dialog allows any of the fluorescence configuration parameters to be edited. No changes take effect until the dialog box is exited by pressing **OK**. The entire collection of parameters can be stored as a named file. Also, previously stored files can be read and edited.

Figure 27-29. The Flr Editor, accessed by pressing **f2** *level 8. The editor is a tree view of the Florescence settings.*

[5.] To adjust them from a program, see **Leaf Chamber Fluorometer Control** on page 25-24 and **LCF Control Functions** on page 25-34.

The complete fluorescence configuration tree is shown below.

```
▼ FlrSettings
    Label=  MyLabel
      ▼ Measure                    See Table 27-5 on page 27-38.
          intensity= 3
          rate= 1
          filter= 3
          gain= 1
      ▼ Flash>                     See Table 27-6 on page 27-38.
          type= MultiPhase
          ▼ ramp= 30 %             Visible only for MultiPhase.
              phase1= 300 ms
              phase2= 300 ms
              phase3= 300 ms
          duration= 1 s            Visible only for Rectangular.
          target= 8
          rate= 4
          filter= 6
          autosave= no
      ▼ DarkPulse                  Table 27-7 on page 27-39.
          duration= 6 secs
          ▼ farred= 8
              pre= 1 s
              post= 4 s
          rate= 2
          filter= 1
          autosave= no
      ▼ Calib                      Table 27-8 on page 27-39.
          unit=  LCF-0304
          red= -2.63
          blue= -1.85
          zero= -2928
      ▼ Constants      Table 27-8 on page 27-39.
          red_abs=0.87
          blue_abs= 0.92
          a_dark= -1
          ps_ratio= 0.5
      ▼ Stats
          period                   Table 27-8 on page 27-39.
```

Flr Import

Flr Import is accessed via **f1** level 8. It presents two sources of fluorescence parameters: any that have been previously exported (stored in **/User/Configs/FlrParams**), and any system configuration files that happen to include Flr parameters (stored in /User/Configs/UserPrefs) (Figure 27-10).

Figure 27-30. You can import from two sources: previously exported Flr Settings, or from system configurations that included Flr parameters.

Flr Adjust

Flr Adjust (Figure 27-31 on page 27-37) is accessed via **f4** level 8. The advantage of Flr Adjust is that it allows measuring fluorescence parameters to be adjusted without interrupting data collection and measurements. HINT: Before entering Flr Adjust, put display line *n* on the top line, so you can immediately see the effect of changes to measuring parameters. Also, if you have real time graphics showing *F* as a function of time, you can view that display with one keystroke, **Charts** (**f4**). To return to the Flr Adjust display from graphics, press **escape**.

*Figure 27-31. The Flr Adjust mode, allows measurement parameters to be adjusted "on the fly" using only the arrow keys ←↑↓→, without interrupting normal New Measurements operations. Pressing **shift** + ↑or↓ increments the selected field by 0.1 rather than 1.0.*

Table 27-5. Fluorescence Measurement Parameters

Node (Adjust Abbrev)	Description
intensity (Int)	Intensity (0 to 10) of the 2 red measuring LEDs. Typical value is 3. You can enter floating point values (e.g. 3.3) if you wish - they don't have to be integers. Effective resolution is 0.1.
rate (kHz)	Modulation used normally (i.e. not in a saturating flash or a dark pulse). (0.25, 1, 10, or 20 kHz). Always use 0.25 kHz for dark-adapted leaves.
filter (Hz)	Averaging done on the measurement signal normally (i.e. not during saturating flash or dark pulse). Specify bandwidth from one of the following (0.5, 1, 5, 10, 20, 50, 100, or 200 Hz). Typical setting is 5.
gain (Gn)	Gain factor for the fluorescence signal (10, 20, 50, or 100). Use 10.

Table 27-6. Fluorescence Saturating Pulse (Flash) Parameters

Editor Label	Description
type	Rectangular or MultiPhase flash. See **Saturating Flashes** on page 27-21.
duration	Length of the saturating flash if Rectangular. Keep it between 0.3 and 2.0 seconds.
ramp	Percent ramp if MultiPhase flash.
phase1 phase2 phase3	Timing (ms) of the three phases of the MultiPhase flash.
target	Saturating flash intensity. Typically, 10 will be 6000 or more μmol m^{-2} s^{-1}.
rate	Fluorescence modulation to use during the flash. Always use 20 kHz.
filter	Averaging to do during the flash. 20 Hz, typically.
autosave	Saves the flash to a file in the /User/LCF folder.

Table 27-7. Fluorescence Dark Pulse Parameters

Editor Label	Description
duration	Length of dark interval. Typically < 5 secs.
farred	Intensity of far red. (Setting of 10 is typically about 5 μmol m^{-2} s^{-1}).
pre	Far red LED timing. See Figure 27-22 on page 27-23.
post	
rate	Fluorescence modulation to use during the dark pulse.
filter	Averaging to use during the dark pulse. 1 Hz is typical.
autosave	Save the dark pulse results to a file in the /User/LCF folder.

Table 27-8. Other Fluorescence Configuration Parameters

Editor Label	Description
Calib[a]	
unit	LCF Serial Number
red	red actinic calibration factor
blue	blue actinic calibration factor
zero	offset
Constants[b]	
red_abs	*RedAbs* id: 4206
blue_abs	*BlueAbs* id: 4207
a_dark	*Adark* id: 4213
ps_ratio	*PS2/1* id: 4222
Stats	
Period	Time period (s) used for *dF/dt* (-98), *FlrCV_%* (-97), and *Fmean* (-115).

a. Items under Calib are read-only. To edit them, see **_View Factory Cal** on page 27-81, or **LCF Control Panel** on page 27-82.
b. See **Fluorescence ComputeList** on page 27-87.

Using the MultiPhase Flash

Background Information

What is the MultiPhase Flash Protocol?

Maximal fluorescence (F_m') is measured in order to estimate the effective quantum efficiency of Photosystem II (Φ_{PSII}) and the rate of electron transport from chlorophyll fluorescence measurements (Genty et al., 1989). F_m' is usually measured with a single saturating, multi-turnover flash to reduce the primary electron acceptor of PSII (Q_A, RF method, Figure 27-32). Multi-turnover means that Q_A is reduced and oxidized multiple times until the transport chain of Q_A to plastoquinone (PQ) is in a reduced state. In many conditions, especially in high-light adapted field plants, it is difficult to achieve full reduction of the Q_A-PQ pool with the RF method, which results in an under-estimation of Fm', Φ_{PSII}, and ETR (Earl and Ennahli, 2004; Markgraf and Berry, 1990). This problem can be avoided with the multiphase flash (MPF) protocol (Figure 27-32) (Loriaux et al., 2006).

Figure 27-32. Rectangular flash (RF) method: a saturating multiturnover flash (Q) of 400 to 1200 ms duration. Multiphase flash (MPF) method: (1) high, nearly saturating Q for approximately 250 ms to reduce QA-PQ pool; (2) ramp of declining Q for about 500 ms; (3) return to the initial high Q for approximately 250 ms to check for flash-induced non-photochemical quenching (qN).

Figure 27-33. Fm' values from phase 2 of the MPF method are regressed against 1E4/Q and extrapolated to estimate the maximal fluorescence at infinite flash intensity.

Configuring for a MultiPhase Flash

Access the saturating flash parameters in the Flr Editor, **F2** level 8 (Figure 27-34). Once in the editor, navigate to the flash parameters.

A) Rectanglular

B) MultiPhase

Figure 27-34. Setting the flash parameters for the two types, Rectanglular and MultiPhase

Flash Type determines whether the saturating flash is rectangular or multiphase. When set as multiphase (Figure 27-34 B), *ramp* is the percentage drop in the maximum saturating intensity during phase 2 of the flash. A good starting ramp value is typically 15 to 40% (Figure 27-35).

The length of the saturating flash is determined via the *duration* (rectangular, Figure 27-34 A) or the *phase1 ... phase3* settings (multiphase, Figure 27-34 B). Information on setting the multiphase timing is in **Making Measurements** on page 27-43.

Target is the maximum light intensity of a rectangular flash, or the first and third phases of a multiphase flash. *Rate* is the measuring light modulation frequency during the flash, and should always be set to 20 kHz. *Filter* is the signal averaging during a flash; a good starting value is 50 Hz. *Autosave* allows the saturating flash file to be automatically saved.

Figure 27-35. Relationship between % ramp depth and maximal fluorescence values for sunflower under 600, 1500, and 2000 µmol m^{-2} s^{-1} actinic intensities. Value are normalized to the maximum of each light series. Values are staggered in the horizontal slightly to discern the error bars.

Making Measurements

1   **Measure a rectangular flash on an example leaf**
In order to set-up the multiphase saturating flash parameters, you must first perform a normal, rectangular flash on a representative, example leaf. Check the shape of this flash for the following characteristics:

A) Square PPFD: In order to check the next two characteristics, the incident PPFD must have a square shape. Start with a flash intensity of 9 or less. At a starting intensity of 10, the LCF does not have any further capacity to boost the light intensity to make it square. Non-square and square flash examples are displayed in Figure 27-36. The lines (PPFD values) should be reasonably horizontal in the 10% and 1% views. If the flash PPFD is not very square, then run the Square Flash Calibration routine (**Square Flash Calibration** on page 27-79). If the flash PPFD is still not square after calibration, then try a lower flash Target.

With no (or poor)
square calibration

With proper
square calibration

*Figure 27-36. Comparison of unsquared (left) and squared (right) flashes. Each is shown with a normal (top) view (press **G**), a view zoomed to the top 10% (press **T**), and a view zoomed in to 1% (press **O**). Arrows indicate what would be the minim time for Phase 1 in a MultiPhase Flash.*

B) QA-PQ pool reduction plateau (Phase 1 timing): With a square PPFD flash, there is a minimum time required to achieve a fluorescence plateau, or a quasi-filling of the QA-PQ pool. This minimum time determines the phase 1 timing of the multiphase flash, and varies with species and light history. A good starting value is 300 milliseconds, but the minimum time required should be checked for each type of leaf being measured in a study. The minimum time required for phase 1 in the two flashes shown in Figure 27-36 is indicated with arrows.

C) NPQ quenching (Phase 1-3 duration): With a square PPFD flash, check the maximum time to ensure fluorescence is not being quenched too soon by non-photochemical quenching. The fluorescence signal should remain at its maximum till the end of the flash. If the flash is too long, quenching will reduce the fluorescence signal.

For example, looking again at Figure 27-36, the squared flash on the right

shows no evidence of quenching. The one on the left, however, starts to drop toward the end, but it is not clear if that is due to quenching, or to the dropping light in the non-squared flash (Hence, the problem with non-square flashes). A leaf that will require a lengthy Phase 1 is shown in Figure 27-37. Clearly, after 1 second, the fluorescence has not reached a stable maximum.

Full scale

Ten%

Figure 27-37. Two views of a slow leaf. The ten% view shows that after 1 second, it had not reached a maximum.

An example of a flash that is too long is shown in Figure 27-38. Notice the drop in the fluorescence signal after 400 ms as indicated with the vertical lines. The phase timing of the multiphase flash on this leaf should be adjusted so the entire flash duration does not exceed 400 ms.

Using a graphics cursor to measure the time interval, we find the plateau lasts 400 ms.

Figure 27-38. Selecting maximum times for a flash. The entire flash is 1 second, but the stable plateau is only 400 ms.

To learn how to use the graphics cursors, see **Measuring Graphs** on page 12-21.

2   Set the multiphase flash timing
Adjust the flash timing for phases 1 through 3 as determined from Step 1. These are set in the Flr Editor menu, **f2** level 8. Change the flash *type* to multiphase, and expand the ramp settings in order to adjust the timing of each phase (Figure 27-34 on page 27-42).

3   Apply and analyze an example multiphase flash
Once a square flash is performed on an example leaf and the timing has been adjusted, perform a multiphase flash on the example leaf. Look at the shape of the flash, as well as the flash details. The first phase should have a maximum fluorescence plateau before the second phase ramp begins. The ramp should be sufficiently large to have plenty of points to fit the regression. Finally, the last phase should agree well with the regression (*FC_DeltaF* or *Check* < 0) and have the same or lower maximum fluorescence as the first phase.

Figure 27-39 on page 27-47 shows data from three MPFs, two poor and one good.

a) Phase 1 too short: The first example shows an MPF in which the first phase is too short. The phase 1 fluorescence did not reach a good maximum before the second phase ramp began. In addition, the third phase has a slightly higher maximum fluorescence than the first phase, which is also an indication that the first phase was too short.

b) Phase 1 too long: The second example shows an MPF in which the first two phases are too long. The phase 3 data are much lower than phase 1. The large negative *FC_DeltaF* is another indicator the first two phases' timing needs to be shortened.

c) Good MPF: The first phase has a nice plateau before the second phase began, and the third phase's maximum fluorescence agreed well with the first phase. The small *FC_DeltaF* confirms that the phase timing is appropriate.

a) Phase 1 too short - FC_DeltaF= 0.6

b) Phase 1 too long - FC_DeltaF= 6.5

c) Good flash- FC_DeltaF= 2.9

Figure 27-39. Three multiphase saturating flash examples. (a) Phase 1 too long. (b) Phase 1 and 2 are too long. (c) Good MPF. See text for discussion.

LI6400XTerm and the LCF

The terminal program LI6400XTerm, which allows you to control the LI-6400 from your computer, has some very useful tools for interacting with the 6400-40 LCF. You can have all ten levels the New Measurements function keys on a palette; you can view continuous trace of your fluorescence curve, and analyze and compare flashes and dark pulses.

If you have not installed the LI-6400 support software on your computer, see **Support Software** on page 11-2. If you are not familiar with using LI6400XTerm, see **Using LI6400XTerm** on page 11-30. The rest of this section assumes you have gotten the program running, and are connected to your LI-6400, which is configured for the LCF (Figure 27-40).

Figure 27-40. LI6400XTerm connected to an LI-6400 configured for the LCF.

Function Key Palette

This feature is not unique to the fluorescence configuration, but we emphasize it here since this is the configuration that has the most New Measurements mode function keys associated with it. To bring up the function key palette, click on the Fcts tool button in LI6400XTerm's menu bar (Figure 27-41).

The buttons on the palette reflect the active function keys on the LI-6400 at any moment in time. The buttons are active: to do the task, just click the button.

Figure 27-41. Opening the function key pallette.

Real Time Fluorescence Trace

Open the Fluorescence Monitor window (Figure 27-42), and select the Fluor tab.

Figure 27-42. The real time fluorescence chart is in the Flr tab of the Fluorescence Monitor window.

The trace of fluorescence will be active if the Chart Active box is checked, and fluorescence measurements are being made. Unlike the real time graphics on-board the LI-6400, this trace will remain active outside of New measurements mode, if you are in OPEN's main screen, or in a routine that has the A/D converter active (i.e. a routine that uses real time measurements).

Flash and Dark Events

To view and analyze fluorescence events, use the Flash Events or Dark Events tabs (Figure 27-43).

Figure 27-43. The Flr Flash and Dark Pulse event tabs in the Fluorescence Monitor Window.

There are four ways to get flash and dark pulse files into this window:

1   Capture them live
While LI6400XTerm is connected to an LI-6400, any flash event that occurs will be captured provided the corresponding event Capture check box is checked.

2   Re-send them from the LI-6400
When the Capture check box checked, enter the event view on the LI-6400 (**f5** level 0, **View Fsh/Drk**), select the flash or dark pulse event to send, and press **ctrl + c.** It will be resent, but will not appear on the live trace Figure 27-42.

3   Read them directly from the LI-6400XT
(LI-6400XT only). If the XTs file system is visible to your computer (**Connecting with Ethernet** on page 11-7), you can use the read button, and navigate to /lpl/User/Lcf, and select the file(s) you wish to look at. This option is available even if the terminal is not connected to the LI-6400XT.

4   Copy the files to the computer, then read them
The read button of course can be used to read any flash files you may have already moved from /User/LCF to your computer. Again, the read button does not require the terminal to be connected to LI-6400.

Basic Experiments

The following experiments will guide you through a range of LCF operations. The first three experiments deal strictly with fluorescence, while the last two combine fluorescence with gas exchange.

Before performing the first three experiments, verify the basic functionality of the LCF as described in **Basic Functionality Test** on page 27-84.

Fluorescence Experiments

These experiments cover some basic fluorescence measurements, such as determining quantum efficiencies for dark adapted and light adapted leaves, and serve as a good introduction to fluorometry with the LCF.

Experiment #1 **Determination of F_v / F_m**

F_v/F_m is an estimate of the maximum quantum efficiency of PSII reaction centers. This ratio is calculated from two parameters: F_o and F_m. F_o is the fluorescence level of a dark-adapted plant with all PSII primary acceptors 'open' (Q_A fully oxidized). F_m is the maximal fluorescence level achieved upon application of a saturating flash of light, such that all primary acceptors 'close' (Q_A fully reduced). Variable fluorescence, F_v, is the difference between F_o and F_m. The variable to maximal fluorescence ratio is normally between 0.75 and 0.85, depending on leaf health, age and preconditioning.

Figure 27-44. Fluorescence parameters of a dark-adapted plant upon application of a saturation flash.

1   **Dark-adapt the leaves**
The best technique for thoroughly dark adapting leaves is to leave them in complete darkness overnight. For illustration purposes, however, it may be adequate to use plants dark-adapted for at least 20 minutes prior to measurements of F_v / F_m.

You may also wish to use the 9964-091 dark adapting clips.

2   **Set the appropriate LCF settings**
Check to make sure the LCF settings are appropriately set (**f2** or **f4**, level 8), as in Table 27-9.

Table 27-9. Suggested settings for determining F_v / F_m.

Parameter		Suggested Value	Comments
Meas	Intensity	1	If too high, it can drive photosynthesis. See also **Optimum Meas Intensity** on page 27-78.
	Modulation	0.25	Always use 0.25 kHz for dark adapted leaves
	Filter	1	
	Gain	10	
Flash	Type	Rectangular	
	Duration	0.8	0.5 to 0.8 is typical
	Intensity	7	
	Modulation	20	Always use 20 kHz
	Filter	50	

3   **Logging?**
If you wish, open and name a log file (**f1** level 1).

4   **Clamp the first leaf in the dark LCF**
The measuring light should be on (**f1** level 9) and the actinic and far red off (**f4** and **f5** level 9). Monitor the fluorescence signal F on display line *n*. It should stabilize within a few seconds. Watch dF/dt on that same display line to indicate stability. Typically, when the absolute value of $dF/dt < 5$, F can be considered to be stable.

If F doesn't stabilize, the measuring intensity may need to be lowered. It is important that the modulation rate be low (0.25 kHz). The intensity of the

measuring light needs to be set high enough for a measurable fluorescence signal, but not so high as to excite PSII and drive photosynthesis.

5   Do FoFm

After F has stabilized, press **Do FoFm** (**f3**, level 0). The status LEDs on the 37-pin connector will flash before and after the saturation flash, and a beep will sound when the data is logged (if you've opened a log file). Note the values of F_o, F_m, and F_v / F_m on display line n. Is F_v/F_m reasonable (0.75 to 0.85)? If it is not, it may be due to inadequate dark adaptation, or inappropriate fluorescence measurement settings.

6   View the flash details

Press **View fsh/drk** (**f5** level 0) followed by **view_Graph** (**f1**) to view the flash. The detailed data collected at 20 Hz during the flash will be plotted versus time. Looking at the details of the flash makes it easier to determine if the settings were appropriate, and if the leaf material was adequately dark-adapted. Examples of "good" and "suspect" flashes are shown in Figure 27-45. Note that there is an automated way of determining the appropriate flash setting. See **Optimum Flash Intensity** on page 27-77.

Figure 27-45. A dark-adapted philodendron measured with two different flash settings. A) Example of a good saturation flash: flash length and intensity are appropriate for this leaf material. B) Example of a poor saturation flash: flash length was too short and intensity too small. F_v/F_m was underestimated by 5%.

7   Repeat with additional samples

Repeat these measurements and try to compare results of different plants and healthy versus stressed plants (e.g. water or temperature-stressed).

■   **Question: Does stress increase or decrease F_v / F_m?**

PSII is sensitive to environmental stresses such as temperature, drought and radiation. Stresses that affect PSII efficiency will cause a decrease in F_v / F_m. As with other plant measurements, there are many plant and environmental factors that affect fluorescence results, including leaf age, health, and environmental conditioning.

Experiment #2   Determination of PSII efficiency

PhiPS2 (ϕ_{PSII} also called $\Delta F/F_m$') is the fraction of absorbed PSII photons that are used in photochemistry, and is measured with a light adapted leaf (Equation (27-9) on page 27-5). It is calculated from F_s and F_m', where F_s is steady-state fluorescence and F_m' is the maximum fluorescence from a light-adapted sample upon application of a saturation flash (Figure 27-46). See also Genty et al. (1989).

Figure 27-46. Fluorescence parameters of a light-adapted plant upon application of a saturation flash.

1   Select light-adapted leaves

For this exercise, select a light-adapted plant, that has had at least 20 minutes of acclimation at the desired light level. For the first part of this experiment, you'll need several leaves that are about the same age and have similar illumination. At the end, you may want to try some leaves that have been at other light levels (e.g. shade vs. sun leaves).

2   Set the proper LCF settings

Use **Flr Editor** (**f2** level 8) or **Flr Adjust** (**f4** level 8) to set the measuring and flash settings (Table 27-10). Also, adjust the actinic intensity to the leaves' ambient level and turn it on (**f5** level 2).

Table 27-10. Suggested settings for determining $\Delta F / F_m$.

Parameter		Suggested Value	Comments
Meas	Intensity	5	This setting can be higher than for dark-adapted leaves, because we don't have to worry about the light becoming actinic.
	Modulation	20 kHz	10 or 20 should be fine. Again, it can be higher than for dark-adapted leaves.
	Filter	1	
	Gain	10	
Flash	Type	Rectangular	
	Duration	0.8	0.5 to 0.8 is typical
	Intensity	8	See **Optimum Flash Intensity** on page 27-77.
	Modulation	20 kHz	Always use 20 for a flash
	Filter	50	

3   Open a log file

This is optional.

4   Clamp onto the first leaf and equilibrate

Wait until F (display line n) comes to steady-state. Watch dF/dt as in the first exercise. If you are using an actinic light intensity that is different than what the leaf was adapted to, you may need to wait 20 minutes or more until the leaf is acclimated to the new light level.

5   DoFsFm'

Once at steady-state, trigger a saturating flash to reduce any oxidized primary acceptors by pressing **Do Fs Fm'** (**f3** level 0). F_s and F_m' can be viewed on display line p, and *PhiPS2* on display line q. How does *PhiPS2* compare with the F_v/F_m measured in experiment #1? (It should be lower, as is explained below.)

6   View the flash details

View the flash via **f5** level 0. The light-adapted flash should look similar to curve A in Figure 27-45 on page 27-55, but have smaller amplitude. More signal noise may also be evident. If the flash plot looks more like curve B, then the Flr Editor settings should be adjusted for your plant material.

7   Repeat with other leaves

Repeat steps 3 to 5, measuring leaves of similar age and light history.

8   For further study: try different light values

Compare PSII quantum yields of leaves adapted to high (2,000 μmol m^{-2} s^{-1}) and low (100 μmol m^{-2} s^{-1}) light levels.

■   **How does ΔF / F$_m$' differ between leaves adapted to high vs. low light?**

PSII quantum yields are usually high under low light conditions because a large proportion of the absorbed light is used in photochemistry. High light adapted plants tend to have low $\Delta F / F_m$' values because a higher proportion of the absorbed energy is dissipated through non-photochemical processes.

Experiment #3   Determine Fv /Fm and Quenching Coefficients

Three other useful fluorescence parameters will be explored in this experiment. F_v' / F_m' from Equation (27-10) on page 27-6 represents the efficiency of energy harvesting by oxidized (open) PSII reaction centers in the light. Two competing processes that quench (decrease) the level of chlorophyll fluorescence in the light are referred to as photochemical (q_P) and non-photochemical (*NPQ*) quenching (Equations (27-13) and (27-14) on page 27-7). Many disciplines use these parameters, but the latter two are particularly useful as quantifiers in stress physiology research.

All three of these parameters require F_o', the minimal fluorescence (in the dark) of a light-adapted leaf. How can this be determined? One method would be to allow the sample to dark-adapt and wait until all PSII centers oxidize (usually 20 minutes or more). A more expedient method (Figure 27-47 on page 27-59) would be to use far-red light to preferentially excite PSI and force electrons to drain from PSII. Only a few seconds of far-red time are needed for this to occur. The LCF provides a "dark pulse" routine which uses this sec-

ond method to determine F_o'. See Figure 27-22 on page 27-23 for an illustration of the dark pulse timing parameters.

Figure 27-47. PSI is preferentially excited by far-red light which drives electron transport of PSI, and thus drains electrons from PSII. This is a method of rapid equilibration for determining F_o'.

Figure 27-48. Fluorescence parameters of a light-adapted sample upon application of a saturating flash, followed by a dark period with far-red light.

1   Select some plants

We'll need both dark-adapted and light-adapted leaves for this exercise.

2   Set LCF settings as in Experiment #1

In addition to the dark-adapted settings used before (Table 27-9 on page 27-54), this exercise also requires configuring the dark pulse parameters (Table 27-11) to measure F_o'.

Table 27-11. Suggested settings for determining $\Delta F / F_m$.

Parameter		Suggested Value	Comments
Dark	Duration	6 secs	Actinic off for 6 seconds.
	Far Red Intensity	8	Full scale (10) typically provides about 6 or 7 μmol m^{-2} s^{-1}.
	Pre-time	1 sec.	Far red turns on 1 second before actinic off.
	Post-time	4 sec.	Far red turns off 4 seconds after actinic goes off.
	Modulation	0.25 kHz	As with F_o, a low modulation frequency is desired.
	Filter	1 Hz	

3   Clamp onto the first leaf

With the actinic off and measuring light on, wait until F (display line n) becomes stable. (e.g., wait until $| dF/dt | < 5$.)

4   Measure Fv / Fm

Press **Do FoFm** (**f3** level 0). F_o (display line o) is immediately set to the current value of F. Then a saturating flash is done, and F_m is set to the maximum value during the flash. Following the flash, the data is logged. Check the F_v / F_m value (display line o) and make sure it is reasonable. Finally, view the flash details (**f5** level 0).

5   Turn on the actinic light and equilibrate

Set the actinic level to the average mid-day PAR value for your plant material. Let the plant adapt to the new light level for about an hour before going to the next step. To determine when the plant is adapted to the new light level, look for stability in F.

6   DoFsFm'Fo'
Press **Do Fs Fm' Fo'** (**f4** level 0) to trigger the following sequence: set F_s, do a saturating flash and set F_m', and then a dark pulse to set F_o'. When it is done (it will take about 12 seconds), view the flash details and check the new F_s, F_m', and F_o' parameters in group o. The light-adapted flashes should resemble plot A in Figure 27-45 on page 27-55.

7   View the dark pulse details
Assess whether the dark pulse settings were appropriate by checking to see if the fluorescence signal leveled off during the pulse (see Figure 27-49).

A) $$\frac{F'_m - F'_o}{F'_m} = \frac{800 - 352}{800} = 0.56$$

B) $$\frac{F'_m - F'_o}{F'_m} = \frac{800 - 366}{800} = 0.54$$

Figure 27-49. A light-adapted bean leaf measured with two different dark pulse settings. A) Example of a good dark curve: far red intensity and time were appropriate for the leaf material (notice the curve flattened before the actinic light was turned on). B) Example of a poor dark pulse: far red intensity is not bright enough and time is too short to get a flat curve. This underestimates F'_v/F'_m by about 4%.

8   Compare F_v / F_m with F_v' / F_m'
Look at the calculated F_v' / F_m' value (display line p). Question: How should this value compare with the F_v / F_m values collected in Experiment #1? Answer: F_v' / F_m' should be $< F_v / F_m$, since F_v / F_m is the maximum light harvesting efficiency, and F_v' / F_m' is the actual light harvesting efficiency at some light level, which is reduced by competing processes for light energy.

9   Examine the quenching coefficients
The quenching coefficients, q_P and q_N, are located in display group p, and are described in Equations (27-13) and (27-14) on page 27-7. q_P and q_N range in

value between 0 and 1. For a truly dark adapted plant, $q_P = 1$ and $q_N = 0$ at the time F_v/F_m is determined. As light increases, these coefficients tend to move in opposite directions (see, for example, Figure 27-52 on page 27-65.)

10   Repeat step #3 to #7 on additional leaf samples
Compare the results with leaves adapted to high and low light conditions like the previous exercise. Question: Would you expect low or high light-adapted leaves to have more efficient PSII light-harvesting reaction centers? Answer: Normally, low light-adapted leaves will be more efficient at using the incident light. At low light intensities there is less excess PAR, allowing most plants to be more efficient at light harvesting.

Fluorescence and Gas Exchange Experiments

The next two experiments combine gas exchange and fluorescence. Make sure the IRGAs are zeroed and matched, and the instrument is ready to measure gas exchange. If you are not familiar with gas exchange measurements with the LI-6400, refer to **Preparation Check Lists** on page 4-2. This experiment uses AutoPrograms. If you are not familiar with these, the basics are discussed in **AutoPrograms** on page 9-31.

Experiment #4   Kinetic Experiment
The goal of this experiment is to take a fully dark-adapted plant and measure the progress of gas exchange, electron transport, and fluorescence quenching, while illuminating a leaf with high light.

1   Dark-adapt plants
It is best to use plants that have been dark-adapted overnight. At the very least, dark-adapt them for 20 minutes.

2   Set up real time graphics
We'll want to plot the quenching coefficients against flash count. That is, QP (ID 4216) vs. FCnt (ID -84), and QN (ID 4220) vs. FCnt. Configure the real time graphics by pressing **GRAPH Setup** (**f4** level 4).

Set up an unused graphics screen with two XY Plots (QP vs. FCnt, and QN vs. FCnt). Use the default autoscale option. For details doing this, see **Real Time Graphics** on page 6-14.

3   **Prepare chamber environment**

CO_2 - If there is a mixer installed, control at ambient (e.g. 390 μmol mol^{-1}) values in the sample cell.

Light - Set for a normal midday value (**f3** level 8), but leave Actinic turned off (**f1** level 9).

Flow - Fixed at 300 μmol s^{-1}.

4   **Prepare LCF settings**

Set for dark-adapted material, as determined in experiments 1 and 2.

5   **Clamp onto the first leaf**

Turn the measuring beam on and monitor F until it stabilizes. If F climbs steadily, the measuring beam intensity is probably too high.

6   **Run the autoprogram "AutoLog2"**

Autoprograms are launched by pressing **Auto Prog** (**f1** level 5). Select the one named "AutoLog2". Once you've named the data file, you will be shown a setup dialog (Figure 27-50).

```
══ (1/3) AutoLog2 Setup═══════
→▶Flr Actions
 ▶Summary: Every 30s for 00:02:00
 ▶Log Opts: beep ctrl


+   ▼/▶   ◆       ◆       ◆cancel ◆ START
```

Figure 27-50. The setup dialog for AutoLog2. When configured for fluorometry, there will be a Flr Actions *node at the top.*

Set the values according for Figure 27-51.

```
═ (1/18) AutoLog2 Setup══════════════
→▼Flr Actions: DarkAdapt +
  ▼Dark adapt before starting= yes
   ├ Dark time (min) before FoFm= 20 ─────── If your leaf is already dark adapted, enter 0.1
   ├ Light time (min) after FoFm= 0.1
   └ Measure dark photo at Fo= yes
  ├ Save each flash= yes
  ├ Save each dark pulse= yes ───────── Your choice on these.
  └ Flr Recording= "Off always"
 ▼Summary: Every 180s for 00:30:00
  ▼Duration (mins)= 30
   ▼Wait for= time ───────── After the dark adapting, perform a log (and
    ├ Log interval (s)= 180          Fs Fm' Fo') every 3 minutes for 30 minutes.
    └ Action= Log w/ Fs Fm' Fo'
 ▼Log Opts: beep
  ├ Beep: on ───────── The Log Options are up to you.
  ▶Statistics: off
  ├ Control changes: no
  └ Echo to Comm: no
```

Figure 27-51. Configuring AutoLog2 for Experiment #4.

7   Watch

After starting the program (**f5 Start**), watch the real time graphics display (**f3** level 4, or else **]**). Given enough time, a steady-state fluorescence level, F_s, and steady-state photosynthesis rate should develop. What happens to q_P and q_N over time, after the actinic light turns on?

How long does it take for gas exchange to come to steady-state? (Compare this time to the time for the fluorescence parameters come to steady-state.)

*Figure 27-52. Sample results from Experiment 4. The upper graphs are the real time graphics images produced on the LI-6400 console, and show the last 300 seconds of F, and QP and QN as a function of flash count (FCNT). The lower plot showing the trace of F for the entire experiment illustrates a use of fluorescence recording (**f5** level 8).*

Experiment #5   **Light Response Curve**
There can be many possible objectives in doing light response curves (see **Light Response Curves** on page 4-24). In this experiment, we will focus on two parameters, *PhiPS2* and *PhiCO2* (Φ_{PSII} and Φ_{CO_2} in Equations (27-9) and (27-11) on page 27-6).

Φ_{PSII} is the quantum yield of PSII calculated from fluorescence, while Φ_{CO_2} is the quantum yield calculated from CO_2 assimilation. In order to calculate this, we need to know total assimilation, which comes from measured (net) CO_2 assimilation (*Photo*) in the light, and an assumption (or prior measurement) of assimilation in the dark (*Adark*). We also need absorbed PAR, which involves knowing incident PAR and leaf absorptivity (see **Leaf Absorptance** on page 27-88).

Note: If using a C_3 species for this experiment, it is best to proceed under non-photorespiratory conditions. That is, low oxygen. (For the plant, not you). This can be achieved by connecting a tank of 2% or less oxygen to the LI-6400 inlet, using an appropriate regulator, "T" fitting, and flow meter, to provide adequate flow for the pump, and a place for the excess flow from the tank to be vented (Figure 27-53). If you do not do this, the measured relationship between Φ_{PSII} and Φ_{CO_2} will likely not be linear. Don't forget to change the *OxyPct* value in the LI-6400. Using a C_4 plant avoids all of this.

Figure 27-53. Supplying low oxygen for C_3 work.

In this exercise, we will start with a light-adapted plant and gradually work toward higher quantum efficiencies and yields by decreasing the incident light. We will use the AutoProgram "Light Curve2" to accomplish this.

1   **Set the environmental controls and LCF.**
Pick a light adapted leaf, and set the environmental controls as follows:

CO_2: (**f3** level 2). Start with controlling on reference CO_2 at 20 μmol mol^{-1} above ambient.

Humidity/Flow: (**f2** level 2) Start with fixed flow at 500 μmol s^{-1} and bypass most of the desiccant.

Light: 2000 μmol mol^{-1}, with 10% blue.

Temperature: Block temperature set to ambient.

LCF: Use settings determined in the previous exercises.

2   Set the fluorescence constants
Press **Prompt All** (**f5** level 3 in New Measurements mode). Here are some suggested values:

BlueAbs: Leaf absorptance in the blue. Use 0.92 (see Table 27-15 on page 27-89).

RedAbs: Leaf absorptance in the red. Use 0.87.

Adark: Photosynthesis rate in the dark. Use -1 μmol m^{-2} s^{-1}.

PS2/1: Fraction of photons that go to PSII. Use 0.5.

3   Set up real time graphics to wait for stability.
Wait for stable, flat lines in the conductance, photosynthesis, and fluorescence graphs. Also, display lines *e* and *n*, (or **[** then **C** for the diagnostic mode stability display) can be checked for gas exchange and fluorescence stability parameters.

4   Change to automatic control
Once the leaf has stabilized, check the value of *H2OS* (line *a*) and change the flow control (**f2** level 2) to constant water mole fraction using the current *H2OS* value as the target. Also, change the CO_2 control (**f3** level 2) to target the sample cell, using the current sample cell value (*CO2S* line *a*) as the target.

5   View the real time graphics for the fluorescence light curve
Press] and view the real time fluorescence at **f**, and the *PhiPS2-PhiCO2* plot at **g**.

6   Configure the AutoProgram

Choose the program "LightCurve2" from the AutoProgram menu (**f1** level 5). Name the file, then configure as indicated in Figure 27-54.

```
═══ (1/4) LightCurve2 Setup═══
→▶Flr Actions
  ▶Summary: 5 SetPts for Qntm
  ▶Stability Definition: 3 items
  ▶Log Opts: beep ctrl


+    ▼/▶   ♦       ♦        ♦cancel ♦ START
```

```
═══ (1/34) LightCurve2 Setup═══
→▼Flr Actions
  ▼Dark adapt before starting= no
   ├ Dark photo rate= -1
   ├ Fo value= 0
   └ Fm value= 0
   ├ Save each flash= no
   ├ Save each dark pulse= no
   └ Flr Recording= "Off always"
 ▼Summary: 7 SetPts for Qntm
   ├ Lamp control= PAR
   ▼SetPts: 7 total, 1st= 2000 10 1
    ▼Stability wait= 120 to 200 s
     ├ Minimum (secs)= 120
     └ Maximum (secs)= 200
    ▼Match before log= "If one of..."
     ├ ...Elapsed time (min) > 20
     ├ ...|CO2 change| (ppm) > 100
     ├ ...|ΔCO2| (ppm)< 10
     ├ Post-match recovery min (s)= 10
     └ Post-match recovery max (s)= 300
   └ Log= Log w/ Fs Fm'
 ▼Stability Definition: 4 items
  ▼items= "Flr Stability"
   ├ items[1]= Photo (30) 20 Slp<0.5
   ├ items[2]= Cond (23) 20 Slp<0.01
   ├ items[3]= H2OS (-5) 20 Slp<1
   └ items[4]= F (-80) 20 Slp<1
 ▼Log Opts: beep ctrl
  ├ Beep: on
  ▶Statistics: off
  ├ Control changes: yes
  └ Echo to Comm: no
```

Your choice on these

Pick 7 points, from 2000 down to 200.

```
═══3 Values/Line: Total, Blue, 0 or 1(%)═══
1600, 10, 1
1200, 10, 1
800, 10, 1
600, 10, 1
400, 10, 1
200, 10, 1
♦DelLn  ♦ClrEnd ♦DelChar♦Cancel ♦OK
```

"10, 1" means "10% blue"

"10, 0" would mean "10 μmol m^{-2} s^{-1} blue"

At each log, we'll do an Fs Fm'

Your choice on these

Figure 27-54. Configuring LightCurve2 for Experiment #5.

If you wish to save this setup before starting, press the **labels** key to get the second level of fct keys, and press **saveAs**. You could name the parameters Experiment#5, for example. If you wanted to run this again at a later date, you would load these parameters from LightCurve2's setup screen by pressing **Open...** and picking the Experiment#5 file.

7   Watch the light curve develop

Press **Start** to launch the program, and watch the light curve develop. The real time graphics should be something like that shown in Figure 27-55.

Figure 27-55. Typical light curve real time graphics. The program does light from high to low, so the light curve (left) will develop from right to left, while the PhiPS2-PhiCO2 plot (right) will develop from left to right.

8   **Graph data and calculate values**
After the last light level, press **View File**, (**f2** level 1) to access GraphIt for your open data file. Press **Import Config** (**f1**) and choose "Light Curve". The light curve (*Photo* vs. *parIn_μm*) will be drawn from the data (Figure 27-56).

Figure 27-56. Light curve for Experiment 5, using GraphIt.

Now, plot *PhiPS2* against *PhiCO2*. The quickest way to do this is to press **Import Config** (**f1**) and pick "PhiPS2 vs PhiCO2". This will plot the points, and fit a straight line through them (Figure 27-57). The relationship should be linear, with an intercept near zero and a slope > 8. For more information on this ratio refer to **Chlorophyll Fluorescence References** on page 27-93.

Figure 27-57. PhiPS2 vs. PhiCO2 from Experiment 5.

To view the slope and intercept, press **View Data** (**f3**), then **C** for Curvefit co-efficients. For the data shown in Figure 27-57, the intercept (Y-axis) is -0.0234, and the slope is 11.0. (For details on finding slopes and doing curve fits with GraphIt, see **Curve Fitting** on page 12-14). When you press **escape**, you'll be asked if you want to add the curvefit information to your data file.

Fluorescence AutoPrograms

Flr Actions Node

The Flr Actions node is included in the setup of the following AutoPrograms when OPEN is configured for the LCF: A-CiCurve2, AutoLog2, CO2Curve_MultipleLight, LightCurve2, LightCurve_MultipleCO2. Flr Action's nodes are described in Table 27-12.

Table 27-12. Standard Fluorescence Prompts for AutoPrograms.

Prompt			Discussion
Dark adapt before starting (yes / no)			If Y, program will wait, measure F_o and F_m, set the actinic light, and wait again for light adaptation. If N, this is skipped.
If dark adapt = yes	Dark time (min) before FoFm		Wait time in minutes before setting F_o and applying a saturating pulse to measure F_m.
	Light time (min) after FoFm		Acclimation time in minutes after actinic is turned on.
	Measure dark photo at Fo (yes / no)		If Y, then right before F_o is set, the value of *Photo* is used to set *Adark*.
	If measure dark photo= no	Dark photo rate	Dark photosynthesis *Adark* (μmol m^{-2} s^{-1})
	Enable Flr Recording before Fo (Y/N)		"Y" will turn Flr Recording on just before the F_o measurement.
If dark adapt = no	Fo value		The value for F_o
	Fm value		The value for F_m
	Dark photo rate		The value (μmol m^{-2} s^{-1}) to be used for dark photosynthesis.
Flr recording	Off always		
	On always		Starts after dark adapt, stays on at end of program
	On at start, Off at end		Starts after dark adapt, turns off at end of program
Save each flash (yes / no)			
Save each dark pulse (yes / no)			

Flr Loop2

FlrLoop2 is a simple program designed to do repetitive fluorescence measurements. The setup dialog is shown in Figure 27-58.

Figure 27-58. FlrLoop2 setup dialog.

There are three settings specific to this program:

1   **How many events?**
This determines how many times you loop through the program.

2   **Time between events (secs)**
This determines how long the program waits each time before doing its action.

3   **Action**
Editing this node toggles through the following four choices:

1: Fo' - a dark pulse
2: Fo Fm - (assumes light off) assign F_o, then does a flash
3: Fs Fm' - assigns F_s, then does a flash
4: Fs Fm' Fo' - assigns F_s, then does a flash, followed by a dark pulse

Calibration Issues

The LCF's Light Sensor

The LCF has independently controlled red and blue LEDs for actinic light to drive photosynthesis. There are also two red LEDs for measuring fluorescence, and a far red LED for driving PSI. The internal light sensor in the LCF sees - directly or indirectly - all of these LEDs. The factory calibration of this sensor to the actinic LEDs consists of generating two factors (one for red, one for blue) that relate actual light output to sensor signal. Actual light output is measured with an integrating sphere and an LI-1800 Spectroradiometer. No calibration factor is generated for the measuring beam, or for the far red LED.

The far red LED has a definite impact on the internal light sensor. Typically, a setting of 10 on the far red LED (and all other LEDs off) will cause the light sensor to read 30 or more. However, the actual quantum flux is usually about $10 \, \mu\text{mol m}^{-2} \, \text{s}^{-1}$ at a setting of 10, so don't trust the internal sensor to measure this accurately. Note that this irradiance is not photosynthetically active, since most of the output is above 700 nm (Figure 27-4 on page 27-8). A plot of *ParIn_μm* during a dark pulse will thus show a bump caused by the far red LED, if it is used.

The measuring beam intensity is typically very small (0.02 and 0.2 $\mu\text{mol m}^{-2} \, \text{s}^{-1}$ - see Figure 27-59) because of the modulation. If the modulation is turned off, and the measuring beam setting turned up to 10, typical output is about 90 $\mu\text{mol m}^{-2} \, \text{s}^{-1}$. However, the internal light sensor will usually *not* measure this accurately, since it is not calibrated for those two LEDs.

Figure 27-59. Typical measuring beam quantum irradiance values for the 6400-40 LCF, as a function of measuring beam setting and modulation rate. These data were measured with an LI-1800 Spectroradiometer and an integrating sphere.

LCF Calibration Menu Programs

Fluorometer calibration routines are found in "`Calib Menu|LCF Source`", but only when the <open> <light> <source> node specifies the 6400-40 LCF.

Figure 27-60. The LCF calibration menu

Calibrate...

This calibration routine (Figure 27-62 on page 27-76) will generate the relation between red and blue control voltages and resulting quantum fluxes in the chamber. These relationships are used to compute first guesses for actinic control, as well as for subsequent computations of *%Blue* (viewed on display line *g*). Once started, the routine runs through a range of control settings for the red and blue LEDs separately.

If you implement the results (step 7 in Figure 27-62), the calibration history will be updated (Figure 27-61).

Figure 27-61. Results from Calib Menu|LCF Source|Calibrate *will show up in the calibration history (*Calib Menu|View Settings|View History*).*

Plot

This routine plots the currently active relation between red and blue control signals and resulting in-chamber quantum flux. That is, it does steps 5 and 6 in Figure 27-62.

1. Initial display.

```
     Fluorometer Actinic Calibration

This program performs independent
calibrations of the actinic red and
blue LEDs of the 6400-40 Flourometer.

Press enter to start, or <esc> to quit
```

2. Does 6 set points for red: 50, 100, 200, 400, 800, and 1500.

```
     Calibrating RED LEDs


SetPt:      49 mV
 QNTM:   21.86 µmol/m2/s    _

earlyOK                              Abort
```

3. Does 7 set points for blue: 100, 500, 1000, 2000, 3000, 4000, and 5000.

```
     Calibrating BLUE LEDs


SetPt:     508 mV
 QNTM:   36.24 µmol/m2/s    _

earlyOK                              Abort
```

4. Press **Y**

```
SetPt     Red     SetPt  Blue
   50     21.8      100      9.0
  100     58.5      500     36.3
  200    154.9     1000     67.0
  400    393.6     2000    118.9
  800    906.5     3000    165.9
 1500   1798.3     4000    206.1
Plot this? (Y/N)_
```

5. Press **escape**

6. Press **escape**

7. Press **Y**

```
Press Any Key
(V - show values)




Implement this cal? (Y/N)_
```

Figure 27-62. *The* Calib Menu | LCF Source | Calibrate *program*.

Optimum Flash Intensity

This program is designed to be an aid in determining the best flash intensity to use (Figure 27-63). You are prompted to enter a range of flash intensities, and a recovery time after each flash. Run this program while clamped onto a representative, light-adapted leaf, and the program will, at the end, present you with a graph of Fmax plotted against flash intensity. **Ideally, the optimum flash intensity is the lowest intensity value that yields the largest fluorescence value**. If the highest intensity (10) causes the highest fluorescence yield, then your leaf is a good candidate for the MultiPhase Flash protocol (page 27-40).

```
Enter intensities:
7 8 9 10
Recovery time (minutes): 3
Store each flash? (Y/N)
Store results at end? (Y/N)
```

Figure 27-63. Prompts for the Optimum Flash Intensity program. If you wish to view the details of each flash, press **Y** *for "Store each flash?".*

Once the program is running, the display will look something like Figure 27-64.

Figure 27-64. Optimum Flash Setting during operation.

After doing the list of flash intensities, you are prompted "Plot This (Y/N)". A **Y** response will provide a plot of *Fmax* as a function of flash intensity, as

shown in Figure 27-65.

Figure 27-65. Plot of results from the Optimum Flash Intensity program.

Optimum Meas Intensity
This program is designed to help you find the proper measurement intensity
for determining F_o. That is, **the intensity that is as large as possible without
inducing photosynthesis**.

You are prompted for a series of measuring beam intensities (Figure 27-66)
to try. The time at each intensity is entered, as well as the recovery (measuring
beam off) time. During the "time at each intensity", the program collects flu-
orescence data F, then determines dF/dt (labelled "slope"). At the end, you
are provided with a plot of these slopes as a function of measuring intensity.

```
Enter intensities:
.5 1 1.5 2 2.5
Time at each intensity (s): 15
Recovery time (s): 60
Store results at end? (Y/N)
```

*Figure 27-66. The prompts for the Optimum Meas
Intensity program.*

Once the program is running, the display shows measuring beam intensity
(*Meas*), mean F, and dF/dt (*Slope*) (Figure 27-67). During the recovery peri-
od, *Meas* will show "off", and the time remaining.

```
Meas     F     Slope
 0.5    62.5  -0.08  ←          Results of vari-
 1.0   130.6   0.13             ous meas inten-
 1.5   213.4   0.12             sities so far.
off    -.02 (23 secs)
```

23 seconds remaining in recovery time (meas off).

Figure 27-67. Optimum Meas Setting during operation.

At the end, *dF/dt* as a function of measuring intensity is plotted for you (Figure 27-68).

Figure 27-68. Optimum Meas Intensity graphical results.

Square Flash Calibration

This automated routine is an important part of using the MPF method. In order to appropriately set the MPF parameters, an example rectangular square flash (RF) must be performed first (see **Making Measurements** on page 27-43). Getting nice square flashes out of the LCF requires calibration. To do that, clamp onto an example leaf (or use an empty or open chamber) and run the Square Flash Calibration in the LCF Calibration Menu (Figure 27-69).

```
┌─Calib Menu═══════════════════════════┐
│  ▼LCF Source                         │
│      Calibrate...                    │
│      Plot...                         │
│      Optimum Flash Intensity...      │
│      Optimum Meas Intensity...       │
│      Square Flash Calibration...     │
│  ▼/▶   ♦        ♦      ♦ EXIT ♦ SELECT│
└──────────────────────────────────────┘
```

Opening screen.
```
SquareFlash Shape Calibration Routine

This program calibrates the LCF for
squared rectangular flashes, allowing
proper Phase3 intensities for
MultiPhase flashes.

OK to proceed? (Y/N)
```

From here on it is automatic...
```
This takes several minutes...
Iteration #1:
dQ/dt=-236.94
dQ/dt=-235.40
AvgDecaySum=-0.0272415
Iteration #2:
```

...eventually displaying this when done.

Press **Y**.
```
Targets= 9.0 8.0 7.0 6.0 5.0
Factors= 1.092 0.876 0.696 0.566 0.431
TCoeffs= 0.000000 0.111432 -0.107965 0.0
53423
ACoeffs= 1.069204 -0.000226
RCoeffs= 108.362131 0.028990 0.000035
Keep? (Y/N)_
```

Figure 27-69. Square Flash Calibration. This is an automated routine that is executed from the Calib Menu. This should be run at the beginning of the day and periodically throughout the day if the temperature changes.

The reason[6] this calibration is necessary is that LED efficiency drops with heating. When the LEDs in the LCF are turned on very brightly for a flash event, their output begins to fall almost immediately upon reaching their target brightness. We compensate for this by attempting to increase the current with time to the LEDs just enough to balance this. Unfortunately, the required "shape" of the control signal is not determined in real time, but must be predicted beforehand. Thus, the Square Flash Calibration routine runs the LCF through a variety of flashes of varying duration and intensity, and determines some empirical relationships that it can use later to provide reasonably square rectangular flashes, and reasonably square phase 1 and phase 3 portions of multi-phase flashes.

_____

[6.]You may skip this paragraph, unless you like arcane details.

Zero Fluorescence Signal

This routine is used to zero the LCF fluorescence signal. The chamber does NOT have to be empty to do this.

Adjusts the Offset value so that F=0

```
          LCF  Fluorescence  Zero

        Raw signal:-2928.7
            Offset:-2928.0
  Final value (F):  -0.7
       Zero                                Quit
```

Figure 27-70. The fluorescence signal zeroing routine.

_View Factory Cal

Each LCF has a unique calibration that relates the in-chamber light sensor's signal to quantum flux. In fact, there are two calibration factors: one for the red LEDs, and another for the blue LEDs. They are measured at the factory, and reside in the memory of the LCF. This allows LCFs and consoles to be interchanged freely, without having to keep track of calibrations. The calibration stays with the LCF.

The "_View Factory Calibration" program allows you to view these values and even change them, although such changes are not normally recommended for the user.

```
      *** 6400-40 Fluorometer ***
          S/N: LCF-0304
  R)  Red cal: -2.63
  B) Blue cal: -1.85
  Z)     Zero: -2928
  E) Read values from LCF
  W) Write value  to LCF
  Press (R/B/Z/E/W or esc)_
```

Figure 27-71. The information that is stored in the LCF can be viewed and edited with this routine. Appropriate key strokes are described in Table 27-13.

Table 27-13. _View Factory Calibration key commands

Key	Description
R	Change the red cal constant. User is prompted with "Red="
B	Change the blue cal constant. User is prompted with "Blue="

Table 27-13. (Continued)_View Factory Calibration key commands

Key	Description
C	Change the zero value. User is prompted with "Zero=". (This value is normally set by running the "Zero Fluorescence Signal" program, described on page 27-81.)
E	Read the red, blue, and zero values from the LCF flash memory.
W	Write the current red, blue, and zero values to the LCF flash memory.

LCF Troubleshooting

LCF Control Panel

A convenient place to do troubleshooting on the LCF is from the LCF Control Panel (`Home Menu|Diagnostics & Tests|LCF Control Panel`), shown in Figure 27-72.

Figure 27-72. The LCF Control Panel program. Use ← or → to move the change indicator from field to field. Use ↑ or ↓ to change the value. For some fields, shift + ↑ or ↓ will change the value by 0.1 rather than 1.0

The top lines show real time values of fluorescence and the *parIn_μm* values. The bottom three lines are the settings fields. The highlighted value can be changed by pressing ↑ or ↓. For the Meas, Blue, Red, and Far field, **shift** with ↑ or ↓ changes the settings by 0.1 instead of 1.0. Pressing **Z** will set the offset value so that *Final F* will be 0. This can only be done when *Mod* is ON, and *Meas* is set to 0. Pressing **V** will access the program _*View Factory Cal* on page 27-81.

Raw F

The raw fluorescence reading of the LCF. This value will be between -5000 and +5000, and is the mV signal on analog input channel 20.

Offset

The offset value for zeroing the fluorescence signal. This is typically -2900 or so. This value is subtracted from the *Raw F* reading to yield the *Final F* value.

Final F

This is the value presented in New Measurements as *F*, the basic fluorescence signal. It typically ranges from 0 to 8000.

ParIn_mV

The mV signal of the LCF quantum sensor. This is analog input channel 23. It ranges from 0 to -5000 mV. (That is why both the red and blue calibration constants are < 0. See **ParIn_μm** below.)

ParIn_μm

The PAR as measured by the LCF quantum sensor. The calibration factors used to convert *parIn_mV* to this value are stored in the LCF sensor head. To see these values, press **V**, and the "_View Factory Cal" program (described on page 27-81) will be accessed.

LCF: ON/OFF

This control turns the LCF on and off. The LCF fan turns on and off with this control as well. If you want to interchange an LCF, use this control to turn it off before disconnecting. After reconnecting, use this control to turn it back on.

Mod: ON/OFF

This control turns the modulation on the measuring beam on and off. When off (and the measuring beam intensity set > 0), the two red measuring LEDs will appear significantly brighter than when modulation is on.

PdGn: 1 or 5

The photodiode gain for the parIn_mv measurement. Normally, this is set to 5. During a flash, OPEN automatically sets it to 1 to prevent overranging the signal. When you toggle this back and forth, the *parIn_mv* and *parIn_μm* values should change by a factor of 5.

Meas: 0 - 10

This controls the intensity of the measuring beam. The higher this is the brighter the two measuring LEDs will appear. If measuring a fluorescing sample, the values of FlrRaw and Final F will be proportional to this setting.

Blue: 0 - 10

The intensity of the blue actinic LEDs. At 10 (and with Red = 0), the *parIn_μm* value should be 100 or more.

Red: 0 - 10

The intensity of the red actinic LEDs. Typically by 4 the *parIn_μm* reading will be above 2000 μmol m^{-2} s^{-1}. You can run it up to 10, but don't leave it there for more than a few seconds. This isn't the best way to test maximum output, as the output will decrease steadily as the LEDs heat up. Leaving the red LEDs at 10 for an extended period will shorten their useful life.

Far red: 0 - 10

The intensity of the far red LED. You can see it, although it looks a bit dim. At full output (10), the typical quantum flux is <10 μmol m^{-2} s^{-1}.

Freq: 0.25, 1, 10, 20 kHz

Select the frequency of modulation of the measuring beam.

Filter: 0.5, 1, 5, 10, 20, 50, 100, 200 Hz

Select the bandwidth (averaging) for the raw fluorescence signal. The higher the number, the faster the response to changes, but at a price of increased noise.

Gain: 10, 20, 50, 100

The gain used for the raw fluorescence signal. 10 is normally used. At 20, the raw fluorescence signals will be about twice as large. So will the noise.

Basic Functionality Test

The LCF Control Panel program described above can be used for a simple functionality test.

1 LCF ON/OFF

Toggle the LCF on and off. When off, the fan should stop. When on, the fan should start.

2 Mod ON/OFF

With the LCF: ON, Mod: ON, and Meas: 3.0, the green MSR status LED should be on. Now look at the two red measuring LEDs in the LCF as you

toggle Mod ON and OFF. The two LEDs should get bright when Mod is OFF, and be dimmer when Mod is ON. The green MSR status LED will remain on.

3   **Freq settings**

With Mod ON, cycle through the Freq settings. You should see the brightness on the two measuring LEDs change. The higher the frequency, the brighter the LEDs.

4   **Zero the fluorescence signal**

Turn Mod ON and set Meas to 0. The signal (*Final F*) should go to zero (+/- 4). Press **Z** to make it do that, if necessary.

5   **Check the fluorescence response**

With Meas on 3, Mod ON, Filter on 0.5, and Gain on 10, clamp the chamber onto the pink fluorescence "standard"[7] in the LCF spares kit. The readings should be roughly 300 or 400. The peak to peak noise of the signal should be about 2 or 3. If there is no response to the standard, or to any leaf, then the fluorometer is not working.

6   **Check digital filtering**

Change the Filter to 200 (press down arrow to jump there from 0.5). The peak to peak noise will increase by a factor of 10. Set the filter back to 0.5.

7   **Blue Check**

Change Blue from 0 up to 2. Check to see that all three blue actinic LEDs in the LCF are on. Now increase it up to 10. The *parIn_μm* value should be about 150 or 200. Turn them back to 0.

8   **Red Check**

Set Red to 1 and see that the red LEDs come on. (They will either be all on or all off.) Turn them off. (Warning: there's nothing keeping you from turning the reds up to 10. They'll be very bright and painful to view directly. Leaving them on that high for more than a few seconds will shorten their lifetime.)

9   **Far red check**

Turn the far red up to 10. Look and see that it is illuminated. Turn it off.

---

[7]·This isn't a standard in any quantitative sense. It is just something (10 pieces of paper laminated in plastic) that provides a fairly strong fluorescence signal, similar to a that of a leaf.

LCF Reference

Programming Commands

The commands listed in Table 27-14 can be used in LPL programs (such as AutoPrograms) to perform fluorescence related tasks. For control commands, see **Leaf Chamber Fluorometer Control** on page 25-24 and **LCF Control Functions** on page 25-34.

Table 27-14. Fluorescence Commands

Command	Description	Fct Key Equivalent
FMeas_On	Turn on measuring beam	f1 level 9
FMeas_Off	Turn off measuring beam	
DoFlash	Do a saturating flash	f2 level 9
DoDark	Do a dark pulse	f3 level 9
DoFm	Do a saturating flash, set F_m, compute user variables, and Log	f2 level 0 (when actinic off)
DoFmp	Do a saturating flash, set F_m', and Log	f0 level 0 (when actinic on)
DoFoFm	Set F_o, then DoFm	f3 level 0 (when actinic off)
DoFsFmp	Set F_s, do a saturating flash, set F_m', compute user variables, and Log	f3 level 0 (when actinic on)
DoFop	Do a dark pulse, set F_o', compute user variables, and Log	f1 level 0 (when actinic on)
DoFsFmpFop	Set F_s, saturating flash, set F_m', dark pulse, set F_o', compute and Log	f4 level 0 (when actinic on)
SetZero	Uses the current fluorescence signal as the offset for future readings.	-none-
SetFs	Set F_s to current value of *Fmean*	f2 level 0 (when actinic on)

Table 27-14. Fluorescence Commands(Continued)

Command	Description	Fct Key Equivalent
SetFo	Set F_o to current value of *Fmean*	f0 level 0 (when actinic off)
SetFm	Set F_m to F_{max} value of latest flash	These have no fct key equivalents, but are performed as part of some fct key routines
SetFm_Prime	Set F_m' to F_{max} value of latest flash	
SetFo_Prime	Set F_o' to F_{min} value during latest dark pulse	
Actinic_On	Turn on actinic, using current mode and targets.	f4 level 9
Actinic_Off	Turn off actinic.	
FarRed_On	Turn on far red LED using current target (farRedTarget).	f5 level 0
FarRed_Off	Turn off far red LED	
FlrRecordingOn	Open fluorescence recording (uses current flr recording file name, and appends)	-none-
FlrRecordingOff	Close fluorescence recording file	
FlrRecordingAsk	Prompt user to pick a new file for fluorescence recording, and if a file exists, will append.	f5 level 8

Fluorescence ComputeList

The default LCF compute list is named "Std Flr Comp_6.2", and is listed below

```
##"/User/Configs/Comps/StdComps_6.2"
##"/User/Configs/comps/FlrOnly"
```

It essentially links two other compute lists together. The first file, StdComps_6.2, is listed in **The Default ComputeList** on page 15-37. The second file Flr Only adds the fluorescence variables, and it is listed below.

```
##4201F1 "Fv" "Fm-Fo"
" flr_m - flr_o "

##4202F3 "Fv/Fm" "Fv/Fm"
" #4201 / flr_m "
```

```
##4205F3 "BlueAbs" "Leaf abs in blue"
" UCON(0.85) "

##4206F3 "RedAbs" "Leaf abs in red"
" UCON(0.85) "

##4207F3 "LeafAbs" "Leaf absorptance"
" (bluePct * #4205 + (100 - bluePct) * #4206)/100 "

##4208F1 "PARabs" "Absorbed PAR" " parIn_um * #4207 "

##4210F1 "Fv'" "Fm' - Fo'"
" flr_mp - flr_op "

##4211F3 "Fv'/Fm'" "Fv'/Fm'"
" #4210 / flr_mp  "

##4212F3 "PhiPS2" "(Fm'-Fs)/Fm'"
" (flr_mp - flr_s)/flr_mp "

##4213 "Adark" "Dark photo value"
" UCON(-1) "

##4215F3 "PhiCO2" "(A-Ao)/(aQ)"
" (#30 - #4213)/(#4208) "

##4216F3 "qP" "(Fm'-Fs)/(Fm'-Fo')"
" (flr_mp - flr_s)/(flr_mp - flr_op) "

##4220F3 "qN" "(Fm-Fm')/(Fm-Fo')"
" (flr_m - flr_mp)/(flr_m - flr_op) "

##4221F3 "NPQ" "(Fm-Fm')/Fm'"
" (flr_m - flr_mp)/flr_mp "

##4222 "PS2/1" "Photosystem Distribution Factor"
" UCON(.5) "

##4223F3 "ETR" "Electron Transport Rate"
" #4212 * #4222 * #4207 * parIn_fs"

##4231F3 "qP_Fo" "(Fm'-Fs)/(Fm'-Fo)"
" (flr_mp - flr_s)/(flr_mp - flr_o) "

##4232F3 "qN_Fo" "(Fm-Fm')/(Fm-Fo)"
" (flr_m - flr_mp)/(flr_m - flr_o) "
```

Leaf Absorptance

The standard fluorescence ComputeList (**Fluorescence ComputeList** on page 27-87) allows leaf absorptance to be a function of the fraction of blue light. Therefore, two user constants are defined (*RedAbs* and *BlueAbs*), and the *LeafAbs* variable computed from

$$\alpha = \frac{\alpha_{blue}B + \alpha_{red}(100 - B)}{100} \qquad (27\text{-}18)$$

where α is effective leaf absorptance, α_{blue} and α_{red} are absorptances in the blue and red, and *B* is the percentage (0-100) of incident light is that is blue. Table 27-15 shows some a sampling of absorptances. The blue tends to be a bit higher than the red. The values in the table are computed by integrating the product of the spectral irradiance $S(\lambda)$ of the LED with the spectral absorptivity of the leaf $\alpha(\lambda)$, and dividing by the integrated spectral irradiance.

$$\alpha = \frac{\int_{\lambda_1}^{\lambda_2} S(\lambda)\alpha(\lambda)d\lambda}{\int_{\lambda_1}^{\lambda_2} S(\lambda)d\lambda} \qquad (27-19)$$

Table 27-15. Leaf absorptances in blue and red for a few species, measured with an LI-1800 spectroradiometer.

Species	α_{blue}	α_{red}
Maize	0.90	0.85
Bean	0.91	0.83
Jasmine	0.92	0.87
Orange	0.94	0.93

The leaf absorptance variable is used in computing electron transport rate ETR (Equation (27-12) on page 27-6) and Φ_{CO_2} (Equation (27-11) on page 27-6).

To set the values of *RedAbs* and *BlueAbs*, do one of the following:

• `Config Menu|View/edit`, navigate to the node <FlrSettings> <Constants>, or

• Press **Sys&User Consts** (**f3** level 3) in New Measurements mode, or

• Press **Flr Editor** (**f2** level 8) in New Measurements mode.

LCF Boundary Layer Conductance
The boundary layer information for the LCF is contained in the file "/Sys/Lib/BlcTable_LCF", which is listed below:

```
LCF, broadleaves. 7/19/01
```

```
BLCTABLE= 0.48 2
0.58 0.14
3.73 1.92
4.65 2.29
5.36 2.94
5.89 3.39
6.46 3.72
```

For an explanation of the BLCTable format, see Figure 14-5 on page 14-22.

Simultaneous Gas Exchange and Fluorescence

The phrase 'simultaneous gas exchange and fluorescence measurements' raises a question: What is happening to the gas exchange measurements while fluorescence measurement events (saturating flash and/or dark pulse) are occurring? Obviously, a flash or several seconds of darkness is not going to leave steady state photosynthesis unaffected, so how are these possible interactions accommodated?

The fundamental fluorescence measurement is the raw, real-time signal (mV) coming from the LCF. This is labelled F, and can be found on display line n. It is measured simultaneously with the gas analyzers, temperatures, and other signals that go into the gas exchange measurements and computations. When a fluorescence event comes along, all gas exchange measurements cease for the duration of the event, while the fluorescence measurement (and the *parIn_µm* value) continues. At the end of the event, one final computation is done, and if logging is active, all of the gas exchange and fluorescence computations are written to the log file.

■ **Example: Do Fs Fm′ (f3 level 0)**
This keystroke (or the corresponding command in an AutoProgram) is supposed to capture the steady state fluorescence value, the state of the gas exchange parameters, do a saturating flash, and add a data record to the log file (if it is open) that captures all of this information in one line. How does it happen? Here is the sequence of events:

1 **F_s is set.**
System variable F_s (display line p) is set to the current value of F.

2 **Normal events stop**
"Normal events" refers to the updating of new raw readings, system variables, and user variables every 0.5 seconds. With the exception of raw fluorescence and the parIn_mV value, no raw readings will update until the end of **Do Fs Fm′**.

3   Saturating Flash
The system variable F_m' (display line p) is set to the maximum value of F during the flash.

4   Compute and Log
If a log file is open (not **Fluorescence Recording** on page 27-33, but normal data logging), user computations are done prior to the data being written to the file. This effectively updates all fluorescence-related quantities, such as *PhiPS2*. Every item in the compute list is recomputed, including the gas exchange related values. They will not change, however, since the latest gas exchange related readings haven't changed since Step 2.

5   Normal events resume.
Regular readings and computations resume.

Figure 27-73 shows a sample data file in which data was logged by fluorescence events. In this case, the first observation was made on a dark adapted leaf, and was triggered by **DoFoFm**, which captures the minimal fluorescence value, does a saturating flash, and assigns the maximal fluorescence to F_m. Note that each of these actions puts one or more remarks into the data file indicating what happened. The gas exchange data (e.g. the photosynthetic rate of -0.73) reflect what was happening just prior to the fluorescence event.

Hint: The LCF generates a lot of remarks. If you do not want these intermingled with your data, they can be routed to a separate file. See **Log Options** on page 9-14.

Rectangular Flashes

DoFoFm

DoFsFm'Fo'

DoFsFm'Fo'

"Obs " "HHMMSS " "FTime " "EBal? " "Photo " "Cond " "Ci " "FCnt " "DCnt " "Fo " "Fm " "Fo '""Fm '""Fs " "Fv/Fm " "Fv'/Fm…
"10:51:31 Fo=519"
"10:51:31 RF#1 5749 um, Fmax=1510 Int=9 Mod=20 Filter=50"
"10:51:32 Fm=1510"
1 "10:51:32 "49.5 0 -18.7 -0.0481 -301 1 0 519.8 1510.5 0.0 0.0 0.0 0.656 0.000 0.000 -1 0.870 0.920 0 0.870 -8.991 0.000…
"10:51:46 Fs=1284 (3 0.25 5 10)"
"10:51:46 RF#2 5833 um, Fmax=1510 Int=9 Mod=20 Filter=50"
"10:51:48 Fm'=1510"
0 "10:51:56 Dark#1 Fmin=479 Dur=6 FarRed=8 Pre=1 Post=4 Mod=0.25 Filter=5"
"10:51:56 Fo'=479"
2 "10:51:56 "65.0 0 -24.7 -0.0389 -708 2 1 519.8 1510.5 479.4 1510.9 1284.4 0.656 0.683 0.150 -1 0.870 0.920 54 0.897…
"10:52:15 Fs=1245 Msr=3 Mod=0.25 Filter=5 Gain=10"
"10:52:15 MPF#3 5825 um, Fmax=1360 Int=1404 +/- 6 Slp=-11 +/- 3 Rmp=20 P1=300 P1=300 P3=300 Int=9 Mod=20 Filter=50
"10:52:16 Fm'=1404"
"10:52:24 Dark#2 Fmin=464 Dur=6 FarRed=8 Pre=1 Post=4 Mod=0.25 Filter=5"
"10:52:24 Fo'=464"
3 "10:52:24 "93.0 0 19.6 -0.0313 1.26E+03 3 2 519.8 1510.5 464.1 1404.9 1245.4 0.656 0.670 0.114 -1 0.870 0.920 54…

MultiPhase Flash

Dark pulses

The remark for an MPF flash shows *Fmax*, the "simple" maximum during phase 1, and *Int*, the intercept of the regression from phase 2. The *Int* value becomes the F_m' value, as reported by the next remark, Fm'=1404.

Figure 27-73. Sample lines from a data file showing remarks generated by fluorescence measurement events, and data records. At 10:51:31, a DoFoFm was performed, causing F_o to be set. A saturating flash was completed at 10:51:31, and the value of Fm set accordingly. A data record was then logged using gas exchange data from the moment Fo was set. Observations 2 and 3 were each triggered by a DoFsFm'Fo' event, with a MultiPhase flash used for observation 3.

Chlorophyll Fluorescence References

Bjorkman, O., and B. Demmig-Adams. 1995. Regulation of photosynthetic light energy capture, conversion, and dissipation in leaves of higher plants. p.17-47. In E. D. Schulze and M. M. Caldwell (ed.), *Ecophysiology of Photosynthesis*. Springer Verlag, Berlin.

Bolhar-Nordenkampf, H. R., and G. Oquist. 1993. Chlorophyll fluorescence as a tool in photosynthesis research. p. 193-206. In D. O. Hall et al. (ed.), *Photosynthesis and Production in a Changing Environment: a Field and Laboratory Manual*. Chapman & Hall, London.

Earl, H. J. and S. Ennahli. 2004. Estimating photosynthetic electron transport via chlorophyll fluorometry without photosystem II light saturation. *Photosynth. Res.* 82:177-186.

Earl, H.J. and M. Tollenaar. 1998. Relationship between thylakoid electron transport and photosynthetic uptake in leaves of three maize (*Zea mays* L.) hybrids. *Photosynth. Res.* 58:245-257.

Genty, B., J-M. Briantais, and N. R. Baker. 1989. The relationship between the quantum yield of photosynthetic electron transport and quenching of chlorophyll fluorescence. *Biochimica et Biophysica Acta*. 990:87-92.

Krause, G. H. and E. Weis. 1991. Chlorophyll fluorescence and photosynthesis: the basics. *Annu. Rev. Plant Physiol. Plant Mol. Bio*l. 42: 313-49.

Loriaux, S. D., R. A. Burns, J. M. Welles, D. K. McDermitt, and B. Genty. 2006. Determination of maximal chlorophyll fluorescence using a multiphase single flash of sub-saturating intensity. Poster Presentation. August, 2006. American Society of Plant Biologists Annual Meetings, Boston, MA.

Markgraf, T. and J. Berry. 1990. Measurement of photochemical and non-photochemical quenching: correction for turnover of PS2 during steady-state photosynthesis. In: M. Baltscheffsky (ed.), *Curr. Res. Photosynth*. IV:279-282.

Schreiber, U., W. Bilger, and C. Neubauer. 1994. Chlorophyll fluorescence as a nonintrusive indicator for rapid assessment of in vivo photosynthesis. p. 49-70. In: E. d. Schulze and M. M. Caldwell (ed.), *Ecophysiology of Photosynthesis*. Vol. 100. Springer, Berlin.

Schreiber, U., W. Bilger, H. Hormann, and C. Neubauer. 1998. Chlorophyll fluorescence as a diagnostic tool: basics and some aspects of practical relevance. p. 320-336. In: A. S. Raghavendra (ed.), *Photosynthesis- A Comprehensive Treatise*. Cambridge University Press, Cambridge, UK.

van Kooten, O. and Snel, J. F. H. 1990. The use of chlorophyll fluorescence nomenclature in plant stress physiology. *Photosynthesis Res*. 25:127-150.

Soil CO$_2$ Flux Chamber

28

Using the 6400-09 Soil Chamber

Introduction

Considerations

Soil carbon dioxide is primarily produced by root respiration, decay of organic matter, and activity of microbes. Rainwater can have direct effects as well, by displacing gas in soil pore spaces (enhancing CO_2 flux at the surface), and by interacting with limestone soils. Also, rainwater itself carries some dissolved CO_2 that can be released in the soil.

Thus, soil CO_2 flux is dependent on soil temperature, organic content, moisture content and precipitation, and has a great deal of spatial variability. Soil CO_2 flux is also extremely sensitive to pressure fluctuations. An unvented chamber will induce significant pressure increases just by closing. Soil water evaporation and heating of the air in the chamber head space also induces pressure increases in an unvented chamber. The 6400-09 Soil CO_2 Flux Chamber is vented so that pressures inside and outside the chamber are in a dynamic equilibrium.

Soil CO_2 flux measured using a chamber system is dependent on the CO_2 concentration in the measurement chamber. This is illustrated in Figure 28-1, which shows typical variations in measured soil CO_2 flux when the chamber headspace CO_2 concentration was allowed to rise. Healy et. al. (1996) used analytical and numerical models of gas diffusion to evaluate chamber headspace concentration influence on estimates of soil CO_2 flux. They found that chamber-induced perturbations of soil-gas concentration gradients could result in substantial underestimation of soil CO2 flux (6 to 34% for a 30 minute measurement).

The LI-6400 Soil CO_2 Flux System has been designed to minimize perturbation in the soil-gas concentration gradient. Before starting the measurement, ambient CO_2 concentration at the soil surface is measured. Once the chamber is installed, the CO_2 scrubber is used to draw the CO_2 in the closed system down below the ambient concentration. The scrubber is turned off, and soil CO_2 flux causes the CO_2 concentration in the chamber headspace to rise (Figure 28-2). Data are logged while the CO_2 concentration rises through the ambient level. The software then computes the flux appropriate for the ambient concentration. This measurement cycle repeats for as many iterations as you select (Figure 28-3).

Figure 28-1. Soil CO$_2$ flux depends on the chamber CO$_2$ concentration.

Figure 28-2. LI-6400 Soil CO₂ Flux System. The shaded plumbing circuit is for CO₂ scrub operations and is only used during the draw down portion of a measurement cycle. During the actual measurement, mixing in the chamber headspace is achieved with the gas analyzer mixing fan and the associated plumbing.

A

B

Figure 28-3. A) Time series of a measurement cycle. Pumping reduced CO_2 air into the chamber brings the CO_2 below ambient. After the pump turns off, CO_2 rises due to soil CO_2 efflux. During this phase, soil CO_2 flux is computed, and data for regressing flux as a function of CO_2 is generated. B) At the end of the measurement cycle, the final flux value is computed by regressing flux vs. CO_2, and computing the flux that corresponds to the target (ambient) concentration value.

Precautions

- **Sun**
  Keep the soil chamber shaded as much as possible to minimize heating.

- **Wind**
  If measurements are made on bare soil with no canopy, variation in the measured flux can occur due to dynamic pressure fluctuations at the pressure vent outlet caused by wind effects. The vent on the 6400-09 is shielded to minimize direct wind effects, but you may wish to shield the entire chamber from the wind.

- **Rain**
  If a thin upper layer of soil becomes saturated from short intense rainfall, a surface gas seal can form that causes CO_2 concentration to increase below the saturated layer. A burst of CO_2 may be released when the sharp edge of the chamber is inserted, causing excessively high flux measurements when in actuality the undisturbed flux is very small. Better measurements can usually be done with a collar (after the initial installation disturbance) if care is taken not to disturb the collar when setting the chamber onto it.

Reference

Healy, Richard W., R.G. Striegl, T.F. Russell, G.L. Hutchinson, and G.P. Livingston, 1996. Numerical Evaluation of Static-Chamber Measurements of Soil-Atmosphere Gas Exchange: Identification of Physical Processes. Soil Sci. Soc. Am. J. 60:740-747.

Attaching the Soil Chamber

General Description

Figure 28-4 shows the 6400-09 Soil Chamber attached to the LI-6400 sensor head, and Figure 28-5 is an exploded diagram showing the parts of the Soil Chamber, along with a detailed parts list should you need to order individual pieces.

Figure 28-4. 6400-09 Soil CO$_2$ Flux Chamber

Soil CO2 Flux Chamber

Attaching the Soil Chamber

#	Description		Part #
1	Fan Inlet Duct		9964-032
2	Hose Barbs		300-02547
3	Soil Chamber Adapter Manifold		9864-174
4	O-rings	Small	192-02597
		Medium	192-00225
		Large	192-02889
5	Radiation Shield		6564-171
6	Pressure Relief Fitting		300-02561
7	Mounting Plate		
8	Hose Barb (to #15)		300-02547
9	Hose Barb (open)		300-00567
10	Soil Temperature Probe Holder		9860-223
11	Pressure Relief Vent Tube		6560-232
12	O-ring		192-04095
13	Mounting Ring		9860-225
14	O-ring		192-04096
15	Air Supply Manifold (from pump)		
16	Air Supply Manifold (from IRGA)		
17	Chamber Body		9860-226
18	Adjustable Stop Ring		9860-227
19	Set Screw		
	Screw		140-04103
	Knob		236-03742
20	Foam Gasket (for use with soil collar)		6560-229
21	PVC Soil Collar (optional)		6560-228
	Plastic cap (not shown)		620-04126

Figure 28-5. Exploded view of the 6400-09 Soil Chamber.

Attaching the Soil Chamber to the Sensor Head

The following steps take you through the removal of the standard leaf chamber from the sensor head, and the attaching of the soil chamber.

■   **Attaching the soil chamber**

1   Remove the standard leaf chamber
Refer to Figure 28-6.

a) Remove the male end of the leaf temperature thermocouple connector by pulling straight out.
b) Unplug the log switch connector.
c) Unplug the light sensor connector.
d) Unplug the LED power connector (if LED source is attached).
e) Pull the air hose from the underside of the leaf chamber.

Figure 28-6. Preparing to remove the standard leaf chamber.

2   The log switch
The log switch is not used with the soil chamber. If the log switch wires are threaded underneath the bottom cover of the sensor head, this cover must be removed to free the log switch (Step 3). (When it is time to put the standard chamber back on, you can save yourself some work and <u>not</u> re-thread the log switch wires beneath this cover.)

If the log switch wires are not threaded beneath the cover, skip to Step 4.

3   Remove the bottom cover (if necessary to free the log switch)
a) Turn the sensor head over and remove the 3 Phillips head screws as shown in Figure 28-7.

Sample Air Hose

Remove these 3 screws

Hose Barb

PORTABLE
PHOTOSYNTHESIS SYSTEM
MODEL   LI-6400
SR. NO.   PSH-0001

Figure 28-7. Preparing to remove the bottom plate.

b) Remove the hose barbs, if necessary. You may be able to slide the cover out from underneath the hose barbs; be careful not to damage the PC board under the cover. If you remove the hose barbs, note the position of the sample and reference air hoses; the sample hose is wrapped with a piece of black shrink wrap.

c) Free the wires.

d) Re-assemble the sensor head bottom cover. Be very careful not to pinch any wires when replacing the cover.

4   **Remove the handle assembly:**
a) Unlatch the handle, and unscrew the knurled leaf chamber adjustment nut (turn clockwise) until it is free of the handle (Figure 28-8).

Adjustment Nut

Figure 28-8. Turn the adjustment nut clockwise to remove.

b) With the handle latching mechanism in the closed position, wrap tape or string around the handle (where your hand would normally be) so that it will stay together. Failure to do so may result in the rear spring coming out. Leave the handle secured in this manner.

c) Remove the two screws (three on some instruments) on the back side of the handle, as shown in Figure 28-9, using a #1 Phillips head screwdriver. Be careful not to lose the spacer that is between the handle mounting plate and the hinge.

Read View

Handle

Quantum sensor holder

Remove these
2 (or 3) screws

Figure 28-9. Remove the screws on the back side of the handle.

5   **Remove the upper half of the chamber**

a) Remove the 2 screws from the hinge on the rear of the upper half of the leaf chamber (Figure 28-10).

Quantum sensor holder

Hinge

Remove these 2 screws

Figure 28-10. Remove the two screws from the handle hinge.

b) Remove one fan shroud screw and attach the lamp connector (Figure 28-11).

Attach lamp connector here

Figure 28-11. Attach the lamp connector to the fan shroud.

6 Remove the lower half of the chamber

There are eight hex head cap screws on the optical bench cover, as shown in Figure 28-12. Remove the cap screws with a 5/64" hex key (in the spares kit). The lower half of the leaf chamber can now be removed.

Top View

Figure 28-12. Remove the 8 hex head screws.

7   Attach the soil chamber mounting block

Use the 8 hex head cap screws from the previous step. The proper orientation of the mounting block is shown in Figure 28-13. Note the thin vinyl gasket on the top surface of the optical bench. This gasket is reusable; it should adhere to the optical bench, but if it becomes detached, be sure to reposition it before attaching the mounting block. Tighten the 8 screws carefully and evenly. *Note that the screw nearest the leaf chamber forms a metal-to-metal seal in the air pathway, and must be tight upon re-assembly of the standard leaf chamber.*

Vinyl gasket 6564-022

Soil Chamber Mounting Block

Figure 28-13. Attach the mounting block to the sensor head.

8   Check the O-rings
Make sure all O-rings are properly positioned, as shown in Figure 28-14.

Figure 28-14. Location of the O-rings and cap screws.

9   Attach the chamber body to the mounting block

Attach the 6400-09 body to the sensor head/mounting block assembly using the 4 cap screws (use the 5/64" hex key included), located on each corner of the mounting block (Figure 28-15).

Female thermocouple connector

Male thermocouple connector

Soil Chamber Mounting Block

Figure 28-15. Attach the 6400-09 body to the mounting block.

10   Connect the air temperature thermocouple.

There is a thermocouple in the soil chamber that measures air temperature. Its connector plugs into the connector normally used for leaf temperature on the sensor head.

11   Join the sample and reference tubes

Connect the tubes together on the sensor head with the "U" shaped piece of tubing, in the 6400-09 replacement parts kit (Figure 28-16). Also, insert the exhaust tube plug onto the match valve's chamber port. The main purpose of

these items is to keep dirt out of the (now unused) match valve and air lines going to the IRGAs.

Figure 28-16. Insert exhaust plug, and sample and reference tube junction as shown.

12   Connect the air supply hoses

The two long air hoses in the LI-6400 cable bundle connect as follows: On the chamber end, connect to the tubes coming from the chamber connection block (shown in Figure 28-14 on page 28-16) *instead* of to their normal place on the sensor head (i.e. the tubes shown in Figure 28-16). The tubes with the black shrink wrap connect to each other.

On the console end, connect as shown in Figure 28-17. The tubing with black shrink wrap connects to a "Y" connector (in the spares kit), and on to the sample and reference ports. The second hose leading from the soil chamber is connected to the Air Inlet port on the console with a short adapter tube (also in the spares kit).

A schematic of the proper overall plumbing is shown in Figure 28-2 on page 28-4.

Figure 28-17. Connecting the sensor head to the console.

NOTE: Make sure the two short pieces of tubing are *FULLY* seated into the two legs of the Y connector. If one of them is not, it will leak, creating problems when pulling the chamber CO_2 concentration below ambient during the drawdown phase of the measurement.

13 **Connect the soil temperature probe to the LI-6400 console**
The 6400-13 thermocouple adapter assembly is in the spares kit. The adapter plugs into the 37 pin auxiliary port on the LI-6400, and the soil temperature thermocouple plugs into the adapter.

Assembly is now complete.

Soil CO2 Flux Chamber
Software

Software

Configuring OPEN for Soil Measurements

To make soil CO_2 flux measurements, a configuration file must first be created. After that, anytime you wish to use the soil chamber, select this configuration file from the list of configurations presented at power up, or when you change configurations (Config Menu -> Open...).

■ **Creating a Soil Chamber Configuration**
Follow the steps in Figure 28-18.

1. Select "New..." in the Config Menu.

```
┌══Config Menu═══════════════════
│   New...
│   Open...
│   Save as...
│   View/edit...
│
│   ▼/▶   ♦        ♦        ♦ EXIT  ♦ SELECT
```

2. Select "6400-09 Soil Chamber".

```
┌══Make a NEW Config for...═══════
│   ▶Light Sources
│   ▶Clear 2x3 Tops
│   ▶Larger Chambers
│   ▼Other Chambers
│       6400-09 Soil Chamber...
│       6400-15 Extended Reach Chamber...
│   ▼/▶   ♦        ♦        ♦ EXIT  ♦ SELECT
```

3. Press **N**.

```
Config Modifications Ready. Pick one:

N) implement them as a New config:
     named 'SoilChamber_6.1.xml'
C) apply them to the Current config
V) View the modifications

<escape> - discard and quit_
```

4. Press **enter**. (Modify the name first, if you like)

```
Dir: /User/Configs/UserPrefs
File:SoilChamber_6.1.xml_
Filter: ver 6 config files (*.xml)
════════════Save this config═══════
│ Default+LED.xml
│ Default+RGB.xml
│ FactoryDefault_6.1.xml
+ Dir  ♦        ♦ Filter♦ CANCEL♦ SELECT
```

5. **escape** back to the main screen. You are ready to operate.

```
   LI-6400XT Soil CO2 Efflux System

           OPEN 6.1aa

/User 26% full
Fri Oct 17 2008 09:18:46      13.64V
┌───────┬────────┬───────┬───────┬────────
│ Home  │ Config │ Calib │ New   │ Utility
│ Menu  │ Menu   │ Menu  │ Msmnts│ Menu
```

Figure 28-18. Creating a soil chamber configuration.

■ **Selecting An Existing Soil Chamber Configuration**
Once a soil chamber configuration has been created, you can select it from the list of configurations that are presented at power up, or when "Open..." is selected from the Config Menu (Figure 28-19).

```
Dir: /User/Configs/UserPrefs
═════════Select a new configuration═════════
  Default+LED.xml
  Default+RGB.xml
  FactoryDefault_6.1.xml
  SoilChamber_6.1.xml

+ Dir    ◆        ◆        ◆ CANCEL◆ SELECT
```

Figure 28-19. Prompting the user for a configuration file.

OPEN's Main Screen

When configured for soil flux measurements, OPEN's Main Screen appears as in Figure 28-20. The only change from normal is the title.

```
   LI-6400XT Soil CO2 Efflux System

            OPEN 6.1aa

/User 26% full
Fri Oct 17 2008 09:18:46        13.64V
  Home   Config  Calib  New    Utility
  Menu    Menu   Menu  Msmnts   Menu
```

Figure 28-20. OPEN's Main Screen.

The Calibration Menu

While configured for soil flux measurements, the flow meter and reference IRGA are not used, so the Calibration Menu is slightly changed (Figure 28-21).

```
═Calib Menu═══════════════════════════════
  ▸View Settings...
   Flow meter zero...
  ▾IRGA
     IRGA zero (Soil Chamber)
     IRGA Span (Soil Chamber)
  ▸CO2 Mixer
  ▾/▸   ◆        ◆        ◆ EXIT  ◆ SELECT
```

Figure 28-21. The soil flux Calibration Menu

Also, the zeroing and spanning routines are a bit different, and some re-plumbing is necessary. This is explained later in **IRGA Calibration** on page 28-42.

New Measurement Mode

New Measurements mode has some new variables and function keys (Table 28-22 on page 28-22).

Figure 28-22. New Measurements with the soil flux configuration. Function key level 7 has most of the controls.

The function keys are described below, and the new variables are discussed in Table 28-2 on page 28-27.

Function Keys

Figure 28-23 shows the function keys in New Measurements while configured for soil CO_2 flux. The important group of keys is on level 7.

1	No changes	Open LogFile	<view file>	<close file>	<add remark>	
2	f2, f3, f5 disabled	LeafFan Fast			Temp OFF	
3	f1 sets a different variable f2 is disabled	AREA= 81.6		Sys&Usr Consts	Prompts off	Prompt All
4	No changes		GRAPH QuikPik	View Graph	GRAPH Setup	
5	No changes	AUTO PROG		Log Options	Define Stablty	Define Log Btn
6	No Changes	Display QuikPik	Display List	What's What	Display Editor	Diag Mode
7	Soil Flux Control Keys	Target= 350	Cycles= 1/3	START	Logging All	Edit Params

Figure 28-23. The soil flux measurements are controlled from level 7.

The following function keys are new - or have different meanings - for the soil configuration:

Target= (**f1** level 7,). Sets <u>Target</u> and <u>Delta</u>. The target is the CO_2 concentration at which you want the measurement taken, and the delta defines the operating window around that target. For example, if you specify a target of 360 ppm and a delta of 20 ppm, the measurement will occur while the CO_2 rises from 340 to 380 ppm. Once above 380 ppm, the pump will pull the CO_2 back down to 340 ppm (minus any extra draw down <u>ddMargin</u>) for another cycle).

Cycles= (**f2** level 7). Sets <u>NumCycls</u>, the number of repetitions to perform. The key label shows the current repetition number, and the maximum number of repetitions (e.g. 1/3 is the first of three).

START (**f3** level 7). Starts/stops the measurement cycle. (While a measurement is in progress, the label will be **Stop**.) If no log file is open when **Start** is pressed, you will be prompted for one. If a file is open, you will be prompted "Append to the current log file? (Y/N)". If you respond **N**, you are prompted for a new log file name.

It is possible to do a measurement without logging to any file. Simply press **escape** when asked for the file name, and press **N** for the subsequent prompt "Log to COMM port? (Y/N)". (The other way to accomplish this is to turn off logging, as described next.)

Logging… (**f4** level 7). Selects what is stored in the log file (if a log file is open) during a measurement. The dialog box is shown in Figure 28-24.

*Figure 28-24. Logging options (**f4** level 7).*

During the measurement cycle, the LI-6400 computes an intermediate observation (<u>Mode</u> = 3) every 2 seconds (see #310 in Table 28-2 on page 28-27). This observation includes a CO_2 efflux value based on a rate of change of CO_2 with time over the previous 10 seconds, which corresponds to 20 samples. (For versions 3.x and 4.x, it is 10 samples over 7.5 seconds.) In addition to the rate of change of CO_2, the intermediate observation includes the mean CO_2 for that interval. At the end of the cycle, when CO_2 has reached the upper limit (<u>Target</u> + <u>Delta</u>), a final result (<u>Mode</u> = 4) record is computed by re-

gressing the rates of change of CO_2 (`dc'/dt`) against the mean CO_2 concentration (`C2avg`), and computing the CO_2 efflux rate (`EFFLUX`) appropriate for the target concentration. Typically, you only need to store the final result record (option 2), but the other options are available.

Edit Params (**f5** level 7).This key provides access to all the soil flux parameters, and allows them to be viewed, edited, stored and retrieved (Figure 28-25).

*Figure 28-25. The **Edit Params** screen.*

The entire list looks like this:

label= Default settings	*A label for your reference*
insert_depth= 0	*ID #305,* InsDepth
target= 375	*ID #313,* Target
delta= 10	*ID #314,* Delta
numCycles= 3	*ID #318* NumCycls
▼ log	*For Mode, see #310 in Table 28-2 on page 28-27*
intermediateObs= yes	*Log mode 3 observations?*
finalFluxes= yes	*Log mode 4 observations?*
▼ settings	
deadTime= 10	*ID #316,* DeadTime
minMeasTime= 20	*ID #317,* MnMsrTime
extra_ppm= 5	*ID #315,* ddMargin
flow= 700	*ID #319,* ddFlow
▼ chamber	
soil_area= 81.6	*ID #308,* Area
base_vol= 991	*ID #307,* Vbase

To edit any of the parameters, place the cursor on that line, and press **f2** (**Edit**).

Figure 28-26. Changing a setting.

If the value of any setting is different from its value as last stored with the configuration file, it appears with an asterisk (Figure 28-27).

Figure 28-27. Changed values (un-stored) are indicated by *.

The **Undo** key (**f4**) allows any or all un-stored values to be reverted back to their stored condition (Figure 28-28).

*Figure 28-28. The **Undo** key in action.*

The **Import** key (**f3**) allows soil settings to be imported from any stored configuration file that has them, or from soil config files that may exist from older versions, if you've upgraded from OPEN 6.0.x (Figure 28-29).

Parameters from an older version of OPEN.

System configurations containing soil parameters.

Figure 28-29. Importing a set of soil parameters.

Soil CO2 Flux Chamber

Software

Display Map

The default soil configuration has the display arranged as shown in Table 28-1.

Table 28-1. The default soil display map for New Measurements mode.

Group	Label	Description
A	CO2S_μml	Sample cell CO_2 (μmol CO_2 mol^{-1})
	H2OS_mml	Sample cell H_2O (mmol H_2O mol^{-1})
	RHcmbr%	RH in the soil chamber (#321)
	RHirga%	RH in the IRGA (#323)
B	EFFLUX	Soil Efflux (#320)
	C2avg	Average CO2 (#302)
	Wavg	Average water concentration (#303)
C	Tsoil_C	Soil temperature (#322)
	Tsch_C	Soil chamber temperature (#324)
	Program Status	State of the measurement (#31)
D	Vtot	Total volume (computed) (#304)
	InsDepth	Insertion depth (#305)
	Target	Target concentration (#313)
	Delta	CO_2 range (#314)
E	Press_kPa	Atmospheric pressure (kPa).
	Flow_μml	Flow rate (μmol s^{-1})
	dC/ct	Rate of change of CO_2 (#300)
	dW/dt	Rate of change of H_2O (#301)
F	Tblock°C	Block temperature
	Tair°C	Temperature in the IRGA sample cell
	Tsch_C	Temperature in the soil chamber (#324)
	Tsoil_C	Temperature of the soil temperature probe (#322)
G	HH:MM:SS	Real time clock
	Program	Shows AutoProgram status
	CHPWMF	Status word (summary of line J)
	Battery	Battery voltage

Table 28-1. (Continued)The default soil display map for New Measurements mode.

Group	Label	Description
H	CO2	Status of CO_2 IRGAs
	H2O	Status of H_2O IRGAs
	Pump	Status of pump
	Flow	Status of Flow controller
	Mixr	Status of CO_2 mixer
	Fan	Speed of chamber fan
I	CRagc_mv	Reference CO_2 AGC (automatic gain control) signal, in mV
	CSagc_mv	Sample CO_2 AGC signal
	HRagc_mv	Reference H_2O AGC signal
	HSagc_mv	Sample H_2O AGC signal

User Variables

The soil CO_2 flux configuration makes use of a compute list file named "/User/Configs/Comps/Soil Efflux Eqns.LPL", which defines a number of user variables and constants (Table 28-2).

Table 28-2. User defined variables and constants for the Soil Configuration .

ID #	Label	Description	How to Change[a] 5.2 and up	How to Change[a] 5.1 and below	LPL Variable name
300	dC/ct	Mode=3: Rate of change of CO_2 Mode=4: Meaningless (last *Mode*=3 value)			*co2Slope*
301	dW/dt	Mode=3: Rate of change of H_2O Mode=4: Meaningless (last *Mode*=3 value)			*h2oSlope*
302	C2avg	Mode=3: Average CO_2 Mode=4: The target CO2.			*co2Mean*
303	Wavg	Mode=3: Average H_2O Mode=4: Meaningless (last *Mode*=3 value)			*h2oMean*
304	Vtot	Total Volume (including IRGA), computed by the software: *Vtot* = Vbase - Area * InsDepth			*soilChamSysVol*

Table 28-2. User defined variables and constants for the Soil Configuration(Continued).

ID #	Label	Description	How to Change[a] 5.2 and up	How to Change[a] 5.1 and below	LPL Variable name
305	InsDepth	Insertion depth (cm). Distance from chamber edge to the top of the soil surface. See Figure 28-30 on page 28-31.	f5 level 7	f5 level 5	*soilInsDepth*
306	dc'/dt	Mode=3: Dilution corrected CO_2 density Mode=4: Dilution corrected density regressed to target concentration.			*dcdt*
307	Vbase	Volume (cm^3) at 0 depth. Includes IRGA.	f5 level 7	Note[b]	*soilCham-BaseVol*
308	Area	Soil area (cm^2)	f1 level 3 f5 level 7	f1 level 3	*soilArea*
310	Mode	0 - Measurement not active 1 - Pump is on, drawing CO_2 down 2 - Pump off, waiting to start. (CO_2 still too low) 3 - Computing intermediate observations 4 - Final result			*opMode*
311	Smpls	Mode=3: Number of samples used in the intermediate observation that was logged. Mode=4: Number of intermediate observations of of dc'/dt used for the final regression.			*numSamps*
312	Program Status	Mode=0: "0:Stopped" Mode=1: "1. Pumping Down" Mode=2: "2:Waiting to start" Mode=3: "3:Measuring" Mode=4: "4:Compute Final"			*srState*
313	Target	Target CO_2 μmol/mol. Computations and logging are performed while the CO_2 is rising from Target - Delta to Target + Delta.	f1 level 7 f5 level 7	f1 level 7	*targetCO2*
314	Delta	CO_2 Delta μmol/mol.			*deltaCO2*
315	ddMargin	Extra Drawdown (ppm). Mode 1 (drawdown) ends when the CO2 concentration drops below Target - Delta - ddMargin.			*ddSafetyMar-gin*
316	DeadTime	Dead Time (secs). Minimum time between when the pump turns off, and when measurements begin. See Figure 28-31 on page 28-33, part #2.	f5 level 7	f2 level 3	*postPump-DeadSecs*
317	MnMsrTime	Min Measure Time (secs). Minimum length of the Mode 3 part of a measurement. See Figure 28-31 on page 28-33, part #3.			*minMeasure-Time*

Table 28-2. User defined variables and constants for the Soil Configuration(Continued).

ID #	Label	Description	How to Change[a] 5.2 and up	How to Change[a] 5.1 and below	LPL Variable name
318	NumCycls	Number of cycles to do. 1 cycle is drawdown (<u>Mode</u>=1) through final result (<u>Mode</u>=4).	**f2** level 7 **f5** level 7	**f2** level 7	*numCycles*
319	ddFlow	Flow during draw down (Mode=1).	**f5** level 7	**f2** level 3	*ddFlow*
320	EFFLUX	<u>Mode</u>=3: Flux for intermediate observation <u>Mode</u>=4: Flux for final result. See Equation (28-11) on page 28-46.			*soilEfflux*
321	RHcmbr%	Relative humidity in the soil chamber			*rhsc*
322	Tsoil_C	Temperature of the external soil probe			*tSoil*
323	RHirga%	Relative humidity in the IRGA			*rhOut*
324	Tsch_C	Air temperature in the soil chamber.			*tLeaf_c*
330	R(C)m	<u>Mode</u>=3: Meaningless <u>Mode</u>=4: Slope of <u>dc'/dt</u> vs. <u>C2avg</u>			*dcdtSlope*
331	R(c)b	<u>Mode</u>=3: Meaningless <u>Mode</u>=4: Intercept of <u>dc'/dt</u> vs. <u>C2avg</u>.			*dcdtOffset*

a.Any of the constants can be added to the prompt list, and edited via **f5** level 3 (**Prompt All**).

b.To modify this variable, add it to the prompt list.

Making Measurements

The procedure below lists the steps required to make a soil surface CO_2 flux measurement. It is assumed that the LI-6400 sensor head has already been attached to the chamber (**Attaching the Soil Chamber to the Sensor Head** on page 28-9), and that the system has been configured for use with the chamber (**Configuring OPEN for Soil Measurements** on page 28-20).

There are two different methods of making measurements. The 6400-09 can be inserted directly into the soil for measurements, or it can be used with soil collars that are inserted into the soil.

Usually, it is preferable to use collars, since inserting a ring (collar or chamber) down into the soil will create artificial flux rates for (potentially) hours afterwards. Collars, once installed, avoid this, allowing the possibility of repeated measurements (chamber on - several cycles - chamber off) at one location. Direct insertion of the chamber, on the other hand, is essentially a destructive, one-time measurement. Perhaps the only advantage of not using collars are those of versatility and spontaneity in choosing measurement lo-

cations, and the avoidance of long-term microclimate changes that are likely with long-term collars.

Measuring With Soil Collars

Soil collars should be installed several hours to one day before making a measurement. You can test to see if the flux has stabilized by making a measurement immediately after installing the collar, and then make subsequent measurements over time. Note, however, that the soil surface CO_2 flux depends on the time of day, and the diurnal cycle can be quite large.

Care must also be taken not to let the bottom edge of the chamber disturb the soil surface within the collar. However, the chamber edge should be as close to the soil surface as practical so that air flow within the chamber produces mixing near the soil surface. Adjust the stop ring to position the chamber near the soil surface. Use a foam gasket between the bottom of the stop ring and the top of the soil collar to minimize leaks between the collar and the chamber.

The soil area value should be set to 80 cm^2 (or whatever is appropriate based on the collar diameter).

Measuring Without Soil Collars

The chamber should be installed on the soil surface by pressing gently and firmly straight down on the mounting plate without rotation. Rotating the chamber may disturb the soil surface by creating a gap around the inside of the chamber, allowing CO_2 in the soil to escape. The soil surface should not be disturbed at all immediately before the measurement. If the surface must be cleared or smoothed before measurements can be made, it should be done prior to the measurement; preferably hours for minor alterations and a day for severe alteration.

Making Measurements

■ **Before you start**

1 **Position the Air Supply Manifold**
Move the lower air supply manifold up or down inside the chamber body so that it is 1-2 cm above the soil surface, regardless of whether or not soil collars are being used for the measurement. This will ensure proper mixing of air coming from the IRGA.

2   Check Hose Connections
Make sure the plumbing is connected as shown in Figure 28-2 on page 28-4, and especially that the two short pieces of tubing in the Y connector are fully seated (Figure 28-17 on page 28-19).

■   **The Measurement Sequence**

1   Determine the CO_2 concentration of the air near the soil surface.
To do this, lay the chamber on its side and monitor soil chamber CO_2 concentration (*CO2S_µml*). You may want to fan ambient air into the chamber (no exhaling, please) if there is little or no wind.

2   Install the 6400-09 at the measurement location.
Insert the soil temperature probe to an appropriate depth (typically 5 to 10 cm) near the Soil Chamber.

3   Set Target and Delta
In New Measurements mode, press **f1** level 7 to set the <u>Target</u> and <u>Delta</u> values. Use the ambient CO_2 concentration determined in Step 1 as the <u>Target</u>, and choose a <u>Delta</u> appropriate for your site. For low rates, the <u>Delta</u> should be 5 or 10 ppm. For higher rates, the delta will have to be increased.

4   Enter the insertion depth of the chamber
If inserting directly into the soil, <u>*InsDepth*</u> will be a positive value, such as 1 or 2 cm, depending on the soil type and the Stop ring position (Figure 28-30A). With soil collars (Figure 28-30B), this will be a negative number that is equal to the distance in cm between the bottom edge of the soil chamber and the soil surface.

Figure 28-30. Measuring the insertion depth, <u>InsDepth</u>.

To enter the insertion depth value, press **f5** level 7.

5   **Enter the number of cycles**
The number (NumCycls) of cycles the instrument should perform at any given location is entered via **f2** level 7.

6   **Select what you want stored in the log file**
Press **f4** level 7 and select whether you want to save final computed values, intermediate instantaneous observations, or both.

7   **Set prompts?**
If you wish to be prompted for some things (location, Area, InsDepth, etc.) automatically at the start of each measurement, set that up by:
a) Prompt Control in the Config Menu to set up the item list.
b) **f4** level 3 to turn Prompts to OnLog.

8   **Start the measurement**
Press Start (**f3** level 7). If you have Prompts ON, you will get those now. Then, you will be prompted to enter a name for the log file if one is not already open. If you don't wish to create a file, press **escape** instead, then press **escape** again when asked about logging to the Comm port.

The measurement cycle will begin (Figure 28-31).

1. The pump turns on, and *CO2S_µml* will decrease. Program Status shows the number of seconds the pump has been on. This will continue until *CO2S_µml* drops below Target - Delta - *ddMargin*.

CO2S_µml	H2OS_mml	RHcmbr%	RHirga%
a 373.1	25.172	89.14	86.20
EFFLUX	C2avg	Wavg	
b 0	380.0	26.13	
Tsoil_C	Tsch_C	Program Status	
c 21.04	22.69	1:Pumping Down (6)	
Target=	Cycles=	STOP	Logging	Edit
7 380	1/3		ALL	Params

2. *CO2S_µml* is now low enough, so the pump turns off. We are waiting for the value of *CO2S_µml* to rise above Target - Delta, so the measurement can begin.

There is a minimum time for this wait, set by the parameter *DeadTime*.

CO2S_µml	H2OS_mml	RHcmbr%	RHirga%
a 363.6	25.021	88.12	85.58
EFFLUX	C2avg	Wavg	
b 0	380.0	26.13	
Tsoil_C	Tsch_C	Program Status	
c 21.02	22.78	2:Waiting to start	
Target=	Cycles=	STOP	Logging	Edit
7 380	1/3		ALL	Params

3. Collecting Data. This will go on until *CO2S_µml* rises above Target +Delta and the timer in Program Status gets above the minimum time (MnMsrTime).

The EFFLUX shown here is for the previous 10 seconds.

CO2S_µml	H2OS_mml	RHcmbr%	RHirga%
a 375.1	25.745	90.49	87.92
EFFLUX	C2avg	Wavg	
b 0	0.0	0.00	
Tsoil_C	Tsch_C	Program Status	
c 21.02	22.82	3:Measuring (4)	
Target=	Cycles=	STOP	Logging	Edit
7 380	1/3		ALL	Params

4. This is shown just momentarily while the software computes and logs the final result record.

The Cycles key label will increment (2/3), and the process continues again with step 1 above.

The EFFLUX value shown here (and until mode=3 again) is the final result value.

CO2S_µml	H2OS_mml	RHcmbr%	RHirga%
a 395.6	26.382	92.73	89.98
EFFLUX	C2avg	Wavg	
b 6.05	389.2	26.22	
Tsoil_C	Tsch_C	Program Status	
c 21.05	22.82	4:Compute Final	
Target=	Cycles=	STOP	Logging	Edit
7 380	1/3		ALL	Params

Figure 28-31. The four stages in a measurement cycle.

Real Time Graphics

The default soil chamber configuration has some Real Time Graphs (**Real Time Graphics** on page 6-14) defined that are useful (Figure 28-32). You can monitor the time series of CO_2, and the relationship between flux and CO_2 concentration, among other things.

Intermediate observations made at each mark.

Figure 28-32. Sample of the Real Time Graphs for soil flux measurements.

AutoPrograms

The typical AutoPrograms (such as AutoLog) are not necessary with the soil chamber, since the measuring routine is built into the events triggered by pressing **Start** (**f3** level 7). However, there is one program -"Soil Efflux vs CO2" - that is added to the programs stored in /User/Configs/AutoProgs when the soil CO_2 flux configuration is created. This program lets you specify a range of target values to be measured automatically. You launch this program like any other (see **AutoPrograms** on page 9-31), and it will set the targets and begin the measurements automatically to cover the range you specified.

Data Files

Text Version

A soil chamber data file is shown in Figure 28-33.

```
"OPEN 6.1"
"Fri Oct 17 2008 12:50:09"
<open><version>"6.1aa"</version></open>
<open><configfile>"/User/Configs/UserPrefs/SoilChamber_6.1.xml"</configfile></open>
<open><light><source>"Sun+Sky"</source><par_in>1 <sensor>"GA-853"<cal>0.8 <actinity>1.00</actinity></cal></sensor>…
<open><comps><file>"/User/Configs/Comps/StdSoilEqns_6.1.LPL"<header>""</header><extras></extras></file><energybal>…
<open><prompts><onlog>off</onlog><items>"Soil Efflux Prompts"<items[1]><id>-104 </id><label>"Plot#"</label><desc>""</desc>…
<open><stability><items>"None"<items[1]><id>-2 </id><size>15 </size><pcv><onoff>0 </onoff><value>1 </value></pcv>…
<open><log><format>"StdSoilLogList_6.0"<items>{ -35 -21 -36 -104 310 311 320 302 303 306 304 321 322 323 …
<open><a2d><avgtime>1.0 secs</avgtime><userchans><ch20>"Off"</ch20><ch21>"ON, gnd=6, res=0"</ch21><ch22>…
<open><matching><type>1 </type><disp>'a'</disp><settings><CO2Limit>10.0</CO2Limit><H2OLimit>1.0</H2OLimit>…
<SoilSettings><label>"Default settings"</label><insert_depth>0 </insert_depth><target>375 </target><delta>10 </delta>…
<li6400><factory><unit>"PSC-853"</unit><serviced>"17 Feb 2003"</serviced><fuseaware>0 </fuseaware><co2mixer>…
<li6400><user><flow_zero>1058.2</flow_zero><irga_zero><co2>-190.3 -970.7<at>25.88 </at></co2><h2o>-502.5 4.8<at>23.3…
"Const= "305 "InsDpth "0
"Const= "308 "Area "81.6
"Const= "313 "Target "375
"Const= "314 "Delta "10
"Const= "315 "ddMargin "5
"Const= "316 "DeadTime "10
"Const= "317 "MnMsrTme "20
"Const= "318 "NumCycls "3
"Const= "319 "ddFlow "700
"Const= "307 "Vbase "991
"11:23:04 "
$STARTOFDATA$
"Obs" "HHMMSS" "FTime" "Plot#" "Mode" "Smpls" "EFFLUX" "C2avg" "Wavg" "dc'/dt" "Vtot" "RHcmbr%" "Tsoil_C" "RHirga%"…
1 "11:24:39 "103.0 0 3 8 9.47 381.5 21.55 0.7794 991 78.31 -226.21 77.57 22.59 1.8774 0.19812 0 0 22.53 385.76 …
2 "11:24:41 "105.0 0 3 12 9.71 383.5 21.74 0.7997 991 79.60 -226.17 78.74 22.57 1.9311 0.18973 0 0 22.54 389.65 …
3 "11:24:43 "107.0 0 3 16 9.28 385.3 21.91 0.7645 991 80.38 -226.32 79.58 22.59 1.8484 0.17464 0 0 22.54 392.64 …
4 "11:24:45 "109.0 0 3 20 9.04 387.0 22.06 0.7443 991 81.43 -226.21 80.57 22.58 1.8019 0.1632 0 0 22.55 396.45 …
5 "11:24:47 "111.0 0 3 20 8.79 390.6 22.37 0.7238 991 82.14 -226.29 81.33 22.59 1.7572 0.1449 0 0 22.55 399.53 …
6 "11:24:49 "113.0 0 3 20 8.51 394.1 22.65 0.7007 991 82.98 -226.26 82.11 22.58 1.7052 0.12916 0 0 22.55 403.48 …
7 "11:24:51 "115.0 0 3 20 8.61 397.5 22.90 0.7086 991 83.40 -226.31 82.61 22.61 1.729 0.11836 0 0 22.56 406.10 …
8 "11:24:53 "117.0 0 3 20 8.41 400.9 23.12 0.6926 991 84.01 -226.36 83.22 22.61 1.6937 0.10583 0 0 22.56 409.57 …
9 "11:24:55 "119.0 0 3 20 8.13 404.3 23.32 0.6694 991 84.51 -226.37 83.75 22.62 1.6407 0.092781 0 0 22.56 412.71 …
10 "11:24:56 "119.5 0 4 33 9.93 375.0 23.32 0.8175 991 84.64 -226.42 83.88 22.62 1.6407 0.092781 -0.0051959 2.7659 …
11 "11:25:19 "143.0 0 3 8 8.46 374.6 23.34 0.6965 991 84.41 -226.67 83.02 22.53 1.7095 0.096334 0 0 22.60 378.44 …
…
30 "11:26:15 "198.5 0 4 33 8.39 375.0 24.92 0.6911 991 89.80 21.22 88.14 22.57 1.5279 0.049291 -0.0036618 2.0643 …
"Const= "313 "Target "380
"Const= "314 "Delta "10
31 "11:27:02 "246.5 0 3 8 7.74 380.0 24.37 0.6376 991 87.49 21.06 85.76 22.62 1.5599 0.10051 0 0 22.78 383.55 …
…
```

Note the <SoilSettings> node in the header.

These items are declared as "headers" in the log format list, so appear as rows, instead of as columns. If any header values are prompted for, or otherwise change, while the file is open, the new value(s) are shown as rows when it happens.

Figure 28-33. A sample text data file.

The header of the file contains all the configuration and calibration settings, and also the <SoilSettings> data. If you examine a text file from the console of the LI-6400 using GraphIt, the header information is easy to navigate (Figure 28-34).

In the Filer, highlight the file, and press **H**

Press **f3**, (**View Data**).

If you press **F**...

...or **H**...
(Expand the nodes to explore the tree)

...or **D**

Figure 28-34. Viewing a soil data file on the LI-6400 console.

Excel Version

The Excel version of a soil data file is shown in Figure 28-35. These files have the equations built in. For example, if you wish to change insertion depth after the fact, just edit cell A8, and all the volumes and flux values will be updated accordingly.

	A	B	C	D	E	F	G	H	I	J
1	OPEN 6.1									
2	Fri Oct 17 2008 11:23:04									
3	Unit=	PSC-853								
4	Config=	/User/Configs/UserPrefs/SoilChamber_6.1.xml								
5	Remark=									
6										
7	InsDpth	Area	Target	Delta	ddMargin	DeadTime	MnMsrTme	NumCycls	ddFlow	Vbase
8	0	80.6	375	10	5	10	20	3	700	991
9										
10	Obs	HHMMSS	FTime	Plot#	Mode	Smpls	EFFLUX	C2avg	Wavg	dc'/dt	Vto
11	in	in	in	in	in	in	out	in	in	out	out
12	1	11:24:39	103	0	3	8	9.58327707	381.522278	21.5547066	0.77942697
13	2	11:24:41	105	0	3	12	9.83260689	383.46875	21.7411938	0.79970546
14	3	11:24:43	107	0	3	16	9.39951641	385.266815	21.9060421	0.76448135
15	4	11:24:45	109	0	3	20	9.15098214	387.017273	22.0573864	0.74426757
16	5	11:24:47	111	0	3	20	8.89875216	390.607727	22.3716259	0.7237532
17	6	11:24:49	113	0	3	20	8.61547049	394.111969	22.6490631	0.70071334
18	7	11:24:51	115	0	3	20	8.71250178	397.549286	22.8956566	0.70860509
19	8	11:24:53	117	0	3	20	8.51593242	400.913666	23.1175137	0.69261771
20	9	11:24:55	119	0	3	20	8.23099923	404.256653	23.3188438	0.66944353
21	10	11:24:56	119.5	0	4	33	10.0511595	375	23.3188438	0.81748078
22	11	11:25:19	143	0	3	8	8.56411623	374.62561	23.3392887	0.6965366
23	12	11:25:21	145	0	3	12	8.59147221	376.337158	23.4406776	0.69876151
24	13	11:25:23	147	0	3	16	8.82788034	378.122406	23.5584354	0.71798906
25	14	11:25:25	149	0	3	20	8.6851227	379.825134	23.6612396	0.70637829
26	15	11:25:27	151	0	3	20	8.56170677	383.24176	23.8496189	0.69634063
27	16	11:25:29	153	0	3	20	8.22599739	386.555389	24.0261326	0.66903672
28	17	11:25:31	155	0	3	20	8.00624503	389.875092	24.1867752	0.65116382
29	18	11:25:33	157	0	3	20	7.96074641	393.065155	24.3155499	0.64746333
30	19	11:25:35	159	0	3	20	7.78563502	396.208771	24.4310818	0.63322117
31	20	11:25:36	159.5	0	4	33	8.7873114	375	24.4310818	0.7146895
32	21	11:25:58	182	0	3	8	8.42208545	376.729492	24.0428581	0.68498495
33	22	11:26:00	184	0	3	12	8.27822539	378.371704	24.1301327	0.67328453
34	23	11:26:02	186	0	3	16	8.38371519	380.066559	24.2147198	0.68186422
35	24	11:26:04	188	0	3	20	8.19436333	381.667542	24.2915268	0.66646386
36	25	11:26:06	190	0	3	20	8.09213286	384.939789	24.4460278	0.65814925
37	26	11:26:08	192	0	3	20	7.86017585	388.115326	24.5826263	0.63928373
38	27	11:26:10	194	0	3	20	7.6711193	391.272034	24.7080898	0.62390738
39	28	11:26:12	196	0	3	20	7.61348517	394.315033	24.8190041	0.61921988
40	29	11:26:14	198	0	3	20	7.58992492	397.43277	24.9215031	0.61730368
41	30	11:26:15	198.5	0	4	33	8.49721025	375	24.9215031	0.691095
42										
43	Target	Delta								
44	380	10								
45	31	11:27:02	246.5	0	3	8	7.83930053	379.967896	24.3723297	0.6375859
46	32	11:27:04	248.5	0	3	12	8.03396952	381.582458	24.4682503	0.65341871

Figure 28-35. An Excel version of a soil chamber data file.

Soil CO2 Flux Chamber
Data Files

Making Sense of *Smpls*

Let's focus on the first cycle in the data file shown on the Excel sheet example (Figure 28-36). The first nine observations are *mode*=3, which means they are taken as the CO_2 is rising. Notice the *Smpls* column, which is the number of samples (`Smpls`). It contains 8, 12, 16, 20, 20, etc. Then, for the *mode*=4 observation (final flux value), it is 33.

	A	B	C	D	E	F	G	H	I	J
1	OPEN 6.1									
2	Fri Oct 17 2008 11:23:04									
3	Unit=	PSC-853								
4	Config=	/User/Configs/UserPrefs/SoilChamber_6.1.xml								
5	Remark=									
6										
7	InsDpth	Area	Target	Delta	ddMargin	DeadTime	MnMsrTme	NumCycls	ddFlow	Vbase
8	0	80.6	375	10	5	10	20	3	700	991
9										
10	Obs	HHMMSS	FTime	Plot#	Mode	Smpls	EFFLUX	C2avg	Wavg	dc'/dt
11	in	in	in	in	in	in	out	in	in	out
12	1	11:24:39	103	0	3	8	9.58327707	381.522278	21.5547066	0.77942697
13	2	11:24:41	105	0	3	12	9.83260689	383.46875	21.7411938	0.79970546
14	3	11:24:43	107	0	3	16	9.39951641	385.266815	21.9060421	0.76448135
15	4	11:24:45	109	0	3	20	9.15098214	387.017273	22.0573864	0.74426757
16	5	11:24:47	111	0	3	20	8.89875216	390.607727	22.3716259	0.7237532
17	6	11:24:49	113	0	3	20	8.61547049	394.111969	22.6490631	0.70071334
18	7	11:24:51	115	0	3	20	8.71250178	397.549286	22.8956566	0.70860509
19	8	11:24:53	117	0	3	20	8.51593242	400.913666	23.1175137	0.69261771
20	9	11:24:55	119	0	3	20	8.23099923	404.256653	23.3188438	0.66944353
21	10	11:24:56	119.5	0	4	33	10.0511595	375	23.3188438	0.81748078

Figure 28-36. One soil efflux cycle.

The LI-6400 makes a new set of readings two times per second. When it starts recording data (*mode*=3), it begins the 10-second running stats. It will log a *mode* 3 observation whenever (a) it has at least eight samples in its running stats buffer, and (b) it has been at least 2 seconds since the last logged reading. Thus, the first four mode 3 observations have 8, 12, 16, and 20 samples. It stays at 20 after that due to the buffer size: twice per second a new sample pushes the oldest one out.

The number of observations of `dc'/dt` and `C2avg` that go into the final regression is given by the *Smpls* column in the *mode*=4 record (line 21). Two observations are added each second, once the minimum (8) samples have been recorded. So, we would appear to have gotten one for the first observation (line 12), and four more for each of the next eight observations (lines 13 through 20), for a total of 33.

Troubleshooting the Soil Chamber

Pump Doesn't Run

When configured for the 6400-09 Soil Flux Chamber, the pump turns on and off automatically during the measurement cycles. If the sample cell CO_2 concentration is above the lower window value (<u>Target</u> - <u>Delta</u>), the pump should come on when you start a measurement and stay on until *CO2S_μml* falls below that value. When *CO2S_μml* rises above the upper window value (<u>Target</u> + <u>Delta</u>), the pump should turn back on, unless the number of requested cycles has been completed. If this is not happening, it could be one of the following:

- **Fuse**
The flow board fuse protects the pump.

- **Parameter settings**
There are four parameters (accessible by **f5** level 7, or on old software, **Aux Params**, **f2** level 3) that influence the pump's behavior.

    <u>ddMargin</u>(**"Extra Draw Down (ppm)"**) can be used to intentionally overshoot the lower limit, providing extra time for things to stabilize before the next measurement cycle.
    <u>DeadTime</u>(**"Dead Time (secs)"**) prevents measurements from starting too soon after the pump turns off.
    <u>MnMsrTime</u>(**"Min Measure Time (secs)"**) is the minimum measurement time, which prevents the pump from turning on again too soon.
    <u>ddFlow</u>(**"Flow during DrawDown"**) lets you set the approximate flow rate when the pump is on.

CO_2 Seems Unresponsive

Verify the operation of the IRGA mixing fan. If it isn't working, the sample cell of the IRGA will never "see" the chamber air.

CO_2 Doesn't Draw Down

Between measurement cycles, the pump should turn on, and draw the chamber CO_2 concentration down. Make sure that at least some of the flow is going through the soda lime tube. Note that leaks through the porous soil mean that there is a minimum CO_2 concentration that you can achieve, based on the soil CO_2 flux rate, the pump flow rate, and how much you are scrubbing the air.

You can adjust the flow rate (**f5** level 7, or on old software, **Aux Params**, **f2** level 3) during draw down.

If the CO_2 cannot be drawn down, even with full flow going though the soda lime, try capping the end of the chamber with the white plastic cap. If CO_2 drops when the chamber is capped, but not when it is on the soil surface, then perhaps you are trying to achieve too low a concentration, or have the flow rate too low, or are not scrubbing enough.

If CO_2 doesn't drop adequately even with the chamber capped, it is likely due to one of the following causes:

- **Chamber fan not functioning**
Verify the operation of the chamber fan by turning it on and off (**f3** level 3) and listening for the noise change.

- **Flow blockage**
Check for obstructions in the return flow hose barb.

- **Leak**
It takes two holes in a closed loop to make a leak. With the soil chamber attached to the sensor head in a closed loop, one (big) hole is the chamber itself (porous soil, pressure vent tube, etc.).

  If there is a leak somewhere between the pump and the chamber, then air can escape the loop there, and is made up by bringing outside air through the chamber pressure relief port. The primary suspect is the Y connector combining the sample and reference air streams (Figure 28-17 on page 28-19). Make sure all three pieces of Bev-a-line tubing are <u>fully</u> seated into this connector (push in until it stops, then push an *additional* 1/4 inch).

  If the leak is between the soda lime tube on the console, and the pump, then outside air will be sucked in and mixed with the scrubbed air. A good way to test if this is happening is to temporarily route the return flow to the chamber directly to the sample cell of the IRGA, and see if this flow is fully scrubbed when the soda lime tube is on full scrub.

High Humidity Headaches
Chamber air is only potentially dried during drawdown. To maximize the drying that occurs during drawdown, set the desiccant knob to full scrub, and back-off on the soda lime scrubbing, so the CO_2 is reduced more slowly.

Note that the system will only give high humidity alerts if the RH in the IRGA

(not the chamber) exceeds 95%. If this happens, it will also automatically turn on the sensor head Peltier coolers to warm the IRGAs a bit to try and avoid condensation. You may want to manually increase the target temperature if high humidity in the IRGA remains a problem.

Maintenance

Spare Parts Kit

This kit contains some common replacement parts for the 6400-09. If you need to re-order any individual parts, please refer to the part numbers shown in Table 5-1 below. More part numbers are shown in Figure 28-2 on page 28-4.

Table 28-3. Soil CO2 Flux Chamber Spare Parts List.

Part #	Description
6000-09TC	Soil temperature probe
6400-13	Thermocouple adapter assembly
9964-054	Replacement parts kit
6560-228	Soil collars[a]
9960-112	Gasket kit (foam gaskets and O-rings)
620-04126	Large plastic cap for soil chamber

a.Soil collars can be easily made from polyvinyl chloride (PVC) tubing. Instructions are given below.

Soil Temperature Probe

The soil temperature probe cable insulation may have a tendency to work loose from the thermocouple connector shell. If this happens, open the connector (remove two screws) and stretch the cable insulation back into the shell, and reassemble.

The soil temperature probe can be ordered from LI-COR under part #6000-09TC, or directly from Omega Engineering Inc. (Stamford, CT) under part #MHP-CXSS-316U-6-SMP-M-NP.

Making Soil Collars

Soil collars can be easily constructed from thin-walled polyvinyl chloride (PVC) pipe (i.e., sewer and drain pipe). The tubing must have an inside diameter of 3.930" (10 cm) or larger [maximum 4.65" (11.8 cm) O.D.]. Cut a sec-

tion approximately 1.75" (4.4 cm) long or longer, depending on your soil type and experiment, and bevel one edge with a grinding wheel so that it can be pressed into the soil. Soil collars are also available from LI-COR at a nominal cost under part #6560-228 (1 each).

Figure 28-37. Soil collar dimensions.

IRGA Calibration

The normal calibration procedure is described on **CO2 and H2O Analyzers** on page 18-10. With the soil chamber in place, however, it introduces some issues. In order to zero or span, it is necessary to re-plumb the system. That is, the air supply, whether scrubbed air from the console for zeroing, or a span tank for spanning, must go directly to the sample IRGA (Figure 28-38).

Figure 28-38. Attach air supply (from console or tank) directory to the sample IRGA when calibrating.

The second thing to do is prevent air from mixing between the soil chamber and the sample cell itself. One approach is to cover the end of the soil chamber with the plastic cap, and have the fan turned off to prevent exchange be-

tween the two volumes. This works pretty well as long as there is not high CO_2 trapped in or floating around the chamber (i.e. work alone, and don't breathe too much). Alternatively, you can loosen the four cap screws (Figure 28-15 on page 28-17) and place a piece of cellophane tape over the three holes that go to the IRGA sample cell volume.

■ **To Zero the IRGA**

1 Connect the source
If the console will be the source of the scrubbed air, then connect the "to chamber" air line (has black heat shrink on it) to the sensor head (Figure 28-38). Turn the soda lime tube on full scrub when zeroing for CO_2, and the desiccant tube on full scrub when zeroing for H_2O.

2 Run the Zero Program
The Calib Menu is slightly modified when configured for the soil chamber (Figure 28-39). Pick the IRGA zero program.

Figure 28-39. The Calib Menu configured for soil.

You'll be asked if the plumbing is ready (Figure 28-40).

*Figure 28-40. If you press **H** at the opening prompt, you'll get a guide to scroll through.*

The soil zeroing program is much like the normal one (**Setting the CO2 and H2O Zero** on page 18-11), only this one is limited to the sample cell, and it

has the chamber fan turned off.

Figure 28-41. The soil chamber IRGA zeroing program.

■ **To Set the IRGA Span**

1 **Connect the span gas directly to the sample analyzer inlet**
Do not connect to the console. NOTE: If you also want to check the span of the reference analyzer, connect the gas directly to it when you are ready. Do NOT connect via the sample cell with the match valve on.

2 **Run the Span program in the Calib Menu**
You'll get the same opening screen as the zeroing program (Figure 28-41 on page 28-44). After that, it runs the normal spanning program (**Setting the CO2 Span** on page 18-17, and **Setting the H2O Span** on page 18-20).

Equation Derivation

The mass balance of CO_2 for the 6400-09 Soil Flux Chamber (Figure 28-42) is given by

$$CO_2 In = Strorage + CO_2 Out$$

$$sf_c = \rho v \frac{\partial c}{\partial t} + uc$$

(28-1)

where s is the soil surface area (m^2) enclosed by the chamber, v is the volume (m^3) of the chamber and IRGA, f_c is the flux of CO_2 coming from the soil surface (mol CO_2 m^{-2} s^{-1}), ρ is the density of the air (mol m^{-3}), c is the CO_2 concentration (mol CO_2 mol^{-1}), and u is the flow rate (mol s^{-1}) of escaping air

from the system, largely due to soil evaporation into the system.

Figure 28-42. Schematic of the soil chamber. Soil evaporation f_w and soil CO2 flux f_c add mass to the chamber air, which is balanced by flow u (mixture of air, water, and CO_2) out of the chamber.

The mass balance of water vapor is given by

$$WaterIn = Strorage + WaterOut$$

$$sf_w = \rho v \frac{\partial w}{\partial t} + uw \qquad (28\text{-}2)$$

where f_w is the flux of H_2O coming from the soil surface (mol H_2O m^{-2} s^{-1}), and w is the water vapor concentration (mol $H_2$0 mol^{-1}).

If we assume that evaporation is the sole cause of the leakage, then

$$u = sf_w \qquad (28\text{-}3)$$

and we can write

$$sf_w = \rho v \frac{\partial w}{\partial t} + sf_w w$$

$$= \frac{\rho v}{(1-w)} \frac{\partial w}{\partial t} \qquad (28\text{-}4)$$

Substituting this for u in Equation (28-1) leads to

$$sf_c = \rho v \frac{\partial c}{\partial t} + \frac{\rho v c}{(1-w)} \frac{\partial w}{\partial t} \qquad (28\text{-}5)$$

Collecting terms leads to

$$f_c = \frac{\rho v}{s}\left(\frac{\partial c}{\partial t} + \frac{c}{(1-w)}\frac{\partial w}{\partial t}\right) \qquad (28\text{-}6)$$

Equation (28-6) takes a slightly different form as implemented in the LI-6400, since the measured and entered parameters have different units, and density must be computed from temperature and pressure. The terms are defined in Table 6-1.

$$c\left(\frac{\text{mol } H_2O}{\text{mol } air}\right) = C\left(\frac{\text{mol } H_2O}{\text{mol } air}\right) \times 10^{-6}\left(\frac{mol}{\mu mol}\right) \qquad (28\text{-}7)$$

$$w\left(\frac{\text{mol } H_2O}{\text{mol } air}\right) = W\left(\frac{m\text{mol } H_2O}{\text{mol } air}\right) \times 10^{-3}\left(\frac{mol}{mmol}\right) \qquad (28\text{-}8)$$

$$\rho\left(\frac{mol}{m^3}\right) = \frac{P(kPa) \times 10^3\left(\frac{N/m^2}{kPa}\right)}{8.314\left(\frac{Nm}{\text{mol } K}\right)(T_c + 273)(K)} \qquad (28\text{-}9)$$

$$\frac{v}{s}\left(\frac{m^3}{m^2}\right) = \frac{V\left(cm^3\right) \times 10^{-6}\left(\frac{m^3}{cm^3}\right)}{S\left(cm^2\right) \times 10^{-4}\left(\frac{m^2}{cm^2}\right)} \qquad (28\text{-}10)$$

Substituting Equations (28-6) through (28-10) into Equation (28-5) yields

$$F_c = \frac{kPV}{S(T+273)}\left(\frac{\partial C}{\partial t} + \frac{C}{(1000-W)}\frac{\partial W}{\partial t}\right), \qquad (28\text{-}11)$$

where $k = 10 \,/\, 8.314 = 1.2028$. This is the equation implemented in the

LI-6400.

Table 28-4. Symbols used in the LI-6400 Soil Flux Equation.

Symbol	Description	Units	Screen Label	ID#
F_c	Flux of CO2	μmol CO_2 m^{-2} s^{-1}	`EFFLUX`	320
P	Atmospheric pressure	kPa	`Prss_kPa`	-11
V	Total system volume	cm^3	`Vtot`	304
S	Soil area	cm^2	`Area`	308
T_c	Soil chamber air temp	C	`Tsch_C`	324
C	CO2 concentration	μmol CO_2 mol^{-1}	`CO2S_`μ`ml`	-2
W	H2O concentration	mmol H_2O mol^{-1}	`H2OS_mml`	-5
$\dfrac{kP}{(T_c + 273)}\left(\dfrac{\partial C}{\partial t} + \dfrac{C}{(1000 - W)}\dfrac{\partial W}{\partial t}\right)$			`dc'/dt`	306

Implementation

The file "/Sys/Configs/Comps/StdSoilEqns_6.1.LPL" contains the implementation of the flux equations. The relevant part is shown below. For an explanation of the format used, see **Format for Modules** on page 15-30.

```
…
INT PUB SoilSaveList[] { 305 313 314 318 315 316 317 319 307 308}

:DOUBLE Tsch_C 0
    satVapTsch 0

:PTR userList[]
{
  :PTR { 300 "dC/dt" s8 co2Slope df3g "dC/dt (Êmol/s)" sad lf5g }
  :PTR { 301 "dW/dt"  s8 h2oSlope df3g "dW/dt (mmol/s)" sad  lf5g }
  :PTR { 302 "C2avg" s8 co2Mean df1f "Mean CO2 (Êmol/s)" sad lf1f }
  :PTR { 303 "Wavg"  s8 h2oMean df2f "Mean H2O (mmol/s)" sad lf2f }
  :PTR { 304 "Vtot"  s8 soilChamSysVol df4g "Total volume (cm3)" sad lf4g  }
  :PTR { 305 "InsDpth" s8 soilInsDepth df3g "Insertion Depth (cm)" sad lf3g 2 }
  :PTR { 306 "dc'/dt" s8 dcdt df3g "rate of change of co2 density" sad lf4g }
  :PTR { 307 "Vbase" s8 soilChamBaseVol df4g "Vol (cm3) at 0 depth" sad lf4g 2 }
  :PTR { 308 "Area" s8 soilArea df4g "Soil area (cm2)" sad lf4g 2 }

  :PTR { 310 "Mode"  s8 opMode  df8d "0-4" sad lf1d  }
  :PTR { 311 "Smpls" s8 numSamps df8d "# obs of slopes or efflux rates" sad lf1d }
  :PTR { 312 "Program Status" "%-18s" srState "%-18s" "Status" "PgSts" "%-18s" }
  :PTR { 313 "Target" s8 targetCO2 df8d "Target CO2 Êmol/mol" sad lf1d 2 }
  :PTR { 314 "Delta" s8 deltaCO2 df8d "CO2 Delta Êmol/mol" sad lf1d 2 }
```

```
:PTR { 315 "ddMargin" s8 ddSafetyMargin df8d "Extra DrawDown (ppm)" sad lf1d 2 }
:PTR { 316 "DeadTime" s8 postPumpDeadSecs df8d "DeadTime (secs)" sad lf1d 2 }
:PTR { 317 "MnMsrTme" s8 minMeasureTime df8d "Min Measure Time (secs)" sad lf1d 2 }
:PTR { 318 "NumCycls" s8 numCycles df8d "Number of Cycles" sad lf1d 2 }
:PTR { 319 "ddFlow" s8 ddFlow df8d "Flow during DrawDown (Êmol/s)" sad lf1d 2}

:PTR { 320 "EFFLUX" s8 soilEfflux df3g "CO2 Efflux Êmol/m2/s" sad lf3g }
:PTR { 321 "RHcmbr%" s8 rhsc df2f "RH in soil chamber %" sad lf2f }
:PTR { 322 "Tsoil_C" s8 tSoil df2f "Soil Temp C" sad lf2f}
:PTR { 323 "RHirga%" s8 rhOut df2f "IRGA RH" sad lf2f }
:PTR { 324 "Tsch_C" s8 Tsch_C df2f "Soil Chamber Air Temp C" sad lf2f }

:PTR { 330 "R(C)m" s8 dcdtSlope df3g "Slope of dc'/dt vs CO2" sad lf5g }
:PTR { 331 "R(C)b" s8 dcdtOffset df3g "Offset of dc'/dt vs CO2" sad lf5g }

}

/* Called from recomp program, or open, every time user vals are to be computed
*/
:FCT ComputeUserValues
{
    $1
    tsch_C = tLeaf_c
    satVapTsch = 0.61365 * EXP(17.502 * tsch_C / (240.97 + tsch_C))

    rhsc = 100.0 * eAir_2_kPa / satVapTsch
    tSoil = chan21_mv / -10.0
    soilChamSysVol = soilChamBaseVol - soilArea * soilInsDepth

    $0 /* post fix to make the dcdt formula excel-friendly  (the if statement) */

    opMode 3 == IF
        press_kPa 1.2028 * tsch_C 273 + / h2oSlope 1000.0 h2oMean - / co2Mean * co2Slope + *
    ELSE
        opMode 4 == IF
            dcdtSlope co2Mean * dcdtOffset +
        ELSE
        0
        THEN
    THEN
    &dcdt =

    $1
    soilEfflux = dcdt * soilChamSysVol / soilArea
}
```

Soil Chamber Specifications

System Volume (0 insertion depth): 991 cm^3

Soil Area Exposed: 71.6 cm^2 (11.1 in^2)
80.0 cm2 (12.4 in^2) with supplied PVC soil collar

Diameter: 9.55 cm (3.76 in.)

Air Temperature Thermocouple:
Type E: Range: ± 50 °C of reference junction
Reference Junction: Optical housing block thermistor
Accuracy: ± 10% of temperature difference between air and reference junctions with the amplifier zeroed

Soil Temperature Probe:
Type E: Ambient Temperature Range: 0 to 50 °C
Soil Temperature Range: ± 30 °C from ambient within the range of -20 °C to 60 °C.
Accuracy: ± 1.5 °C, 0 to 50 °C

Size: 16.50 H x 19.80 W x 10.20 D cm. (6.5 x 7.8 x 4.0 in.)

Weight: 1.8 kg (3.75 lbs.)

*Specifications subject to change without notice.

License Agreements

A

The fine print

License Agreements

Apache

T*he majority of the source code in the mDNSResponder project is licensed under the terms of the Apache License, Version 2.0, available from:*

http://www.apache.org/licenses/LICENSE-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

a. You must give any other recipients of the Work or Derivative Works a copy of this License; and

b. You must cause any modified files to carry prominent notices stating that You changed the files; and

c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

d. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and

do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from

the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

> a) You must cause the modified files to carry prominent notices   stating that you changed the files and the date of any change.

> b) You must cause any work that you distribute or publish, that in   whole or in part contains or is derived from the Program or any   part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

> c) If the modified program normally reads commands interactively   when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer   to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent

license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY

MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIA-
BLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL
OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE
THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR
A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN
IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest

possible use to the public, the best way to achieve this is to make it free software which every-
one can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of
each source file to most effectively convey the exclusion of warranty; and each file should have
at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms
> of the GNU General Public License as published by the Free Software Foundation;
> either version 2 of the License, or   (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY
> WARRANTY; without even the implied warranty of MERCHANTABILITY or FIT-
> NESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
> more details.
>
> You should have received a copy of the GNU General Public License along with this
> program; if not, write to the Free Software   Foundation, Inc., 59 Temple Place, Suite
> 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive
mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show
> w'.
> This is free software, and you are welcome to redistribute it under certain conditions;
> type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the
General Public License. Of course, the commands you use may be called something other than
`show w' and `show c'; they could even be

mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your

school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program

`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

GNU Lesser General Public License

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

 Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files

to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

 In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPY-ING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does

not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the

Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your

obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE

OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIA-BLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF

SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICU-LAR PURPOSE. See the GNU   Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweak-ing knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

Boost Software License

Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Apple Public Source License

Version 2.0 - August 6, 2003

Please read this License carefully before downloading this software. By downloading or using this software, you are agreeing to be bound by the terms of this License. If you do not or cannot agree to the terms of this License, please do not download or use the software.

1. General; Definitions. This License applies to any program or other work which Apple Computer, Inc. ("Apple") makes publicly available and which contains a notice placed by Apple identifying such program or work as "Original Code" and stating that it is subject to the terms of this Apple Public Source License version 2.0 ("License"). As used in this License:

1.1 "Applicable Patent Rights" mean: (a) in the case where Apple is the grantor of rights, (i) claims of patents that are now or hereafter acquired, owned by or assigned to Apple and (ii) that cover subject matter contained in the Original Code, but only to the extent necessary to use, reproduce and/or distribute the Original Code without infringement; and (b) in the case where You are the grantor of rights, (i) claims of patents that are now or hereafter acquired, owned by or assigned to You and (ii) that cover subject matter in Your Modifications, taken alone or in combination with Original Code.

1.2 "Contributor" means any person or entity that creates or contributes to the creation of Modifications.

1.3 "Covered Code" means the Original Code, Modifications, the combination of Original Code and any Modifications, and/or any respective portions thereof.

1.4 "Externally Deploy" means: (a) to sublicense, distribute or otherwise make Covered Code available, directly or indirectly, to anyone other than You; and/or (b) to use Covered Code, alone or as part of a Larger Work, in any way to provide a service, including but not limited to delivery of content, through electronic communication

with a client other than You.

1.5 "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.6 "Modifications" mean any addition to, deletion from, and/or change to, the substance and/or structure of the Original Code, any previous Modifications, the combination of Original Code and any previous Modifications, and/or any respective portions thereof. When code is released as a series of files, a Modification is: (a) any addition to or deletion from the contents of a file containing Covered Code; and/or (b) any new file or other representation of computer program statements that contains any part of Covered Code.

1.7 "Original Code" means (a) the Source Code of a program or other work as originally made available by Apple under this License, including the Source Code of any updates or upgrades to such programs or works made available by Apple under this License, and that has been expressly identified by Apple as such in the header file(s) of such work; and (b) the object code compiled from such Source Code and originally made available by Apple under this License

1.8 "Source Code" means the human readable form of a program or other work that is suitable for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an executable (object code).

1.9 "You" or "Your" means an individual or a legal entity exercising rights under this License. For legal entities, "You" or "Your" includes any entity which controls, is controlled by, or is under common control with, You, where "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.

2. Permitted Uses; Conditions & Restrictions. Subject to the terms and conditions of this License, Apple hereby grants You, effective on the date You accept this License and download the Original Code, a world-wide, royalty-free, non-exclusive license, to the extent of Apple's Applicable Patent Rights and copyrights covering the Original Code, to do the following:

2.1 Unmodified Code. You may use, reproduce, display, perform, internally distribute within Your organization, and Externally Deploy verbatim, unmodified copies of the Original Code, for commercial or non-commercial purposes, provided that in each instance:

> (a) You must retain and reproduce in all copies of Original Code the copyright and other proprietary notices and disclaimers of Apple as they appear in the Original Code, and keep intact all notices in the Original Code that refer to this License; and

> (b) You must include a copy of this License with every copy of Source Code of Covered Code and documentation You distribute or Externally Deploy, and You may not offer or impose any terms on such Source Code that alter or restrict this License or the recipients' rights hereunder, except as permitted under Section 6.

2.2 Modified Code. You may modify Covered Code and use, reproduce, display, perform, internally distribute within Your organization, and Externally Deploy Your Modifications and Cov-

ered Code, for commercial or non-commercial purposes, provided that in each instance You also meet all of these conditions:

> (a) You must satisfy all the conditions of Section 2.1 with respect to the Source Code of the Covered Code;

> (b) You must duplicate, to the extent it does not already exist, the notice in Exhibit A in each file of the Source Code of all Your Modifications, and cause the modified files to carry prominent notices stating that You changed the files and the date of any change; and

> (c) If You Externally Deploy Your Modifications, You must make Source Code of all Your Externally Deployed Modifications either available to those to whom You have Externally Deployed Your Modifications, or publicly available. Source Code of Your Externally Deployed Modifications must be released under the terms set forth in this License, including the license grants set forth in Section 3 below, for as long as you Externally Deploy the Covered Code or twelve (12) months from the date of initial External Deployment, whichever is longer. You should preferably distribute the Source Code of Your Externally Deployed Modifications electronically (e.g. download from a web site).

2.3 Distribution of Executable Versions. In addition, if You Externally Deploy Covered Code (Original Code and/or Modifications) in object code, executable form only, You must include a prominent notice, in the code itself as well as in related documentation, stating that Source Code of the Covered Code is available under the terms of this License with information on how and where to obtain such Source Code.

2.4 Third Party Rights. You expressly acknowledge and agree that although Apple and each Contributor grants the licenses to their respective portions of the Covered Code set forth herein, no assurances are provided by Apple or any Contributor that the Covered Code does not infringe the patent or other intellectual property rights of any other entity. Apple and each Contributor disclaim any liability to You for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, You hereby assume sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow You to distribute the Covered Code, it is Your responsibility to acquire that license before distributing the Covered Code.

3. Your Grants. In consideration of, and as a condition to, the licenses granted to You under this License, You hereby grant to any person or entity receiving or distributing Covered Code under this License a non-exclusive, royalty-free, perpetual, irrevocable license, under Your Applicable Patent Rights and other intellectual property rights (other than patent) owned or controlled by You, to use, reproduce, display, perform, modify, sublicense, distribute and Externally Deploy Your Modifications of the same scope and extent as Apple's licenses under Sections 2.1 and 2.2 above.

4. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In each such instance, You must make sure the requirements of this License are fulfilled for the Covered Code or any portion thereof.

5. Limitations on Patent License. Except as expressly stated in Section 2, no other patent rights, express or implied, are granted by Apple herein. Modifications and/or Larger Works may require additional patent licenses from Apple which Apple may grant in its sole discretion.

6. Additional Terms. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations and/or other rights consistent with the scope of the license granted herein ("Additional Terms") to one or more recipients of Covered Code. However, You may do so only on Your own behalf and as Your sole responsibility, and not on behalf of Apple or any Contributor. You must obtain the recipient's agreement that any such Additional Terms are offered by You alone, and You hereby agree to indemnify, defend and hold Apple and every Contributor harmless for any liability incurred by or claims asserted against Apple or such Contributor by reason of any such Additional Terms.

7. Versions of the License. Apple may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Once Original Code has been published under a particular version of this License, You may continue to use it under the terms of that version. You may also choose to use such Original Code under the terms of any subsequent version of this License published by Apple. No one other than Apple has the right to modify the terms applicable to Covered Code created under this License.

8. NO WARRANTY OR SUPPORT. The Covered Code may contain in whole or in part prerelease, untested, or not fully tested works. The Covered Code may contain errors that could cause failures or loss of data, and may be incomplete or contain inaccuracies. You expressly acknowledge and agree that use of the Covered Code, or any portion thereof, is at Your sole and entire risk. THE COVERED CODE IS PROVIDED "AS IS" AND WITHOUT WARRANTY, UPGRADES OR SUPPORT OF ANY KIND AND APPLE AND APPLE'S LICENSOR(S) (COLLECTIVELY REFERRED TO AS "APPLE" FOR THE PURPOSES OF SECTIONS 8 AND 9) AND ALL CONTRIBUTORS EXPRESSLY DISCLAIM ALL WARRANTIES AND/OR CONDITIONS, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. APPLE AND EACH CONTRIBUTOR DOES NOT WARRANT AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE COVERED CODE, THAT THE FUNCTIONS CONTAINED IN THE COVERED CODE WILL MEET YOUR REQUIREMENTS, THAT THE OPERATION OF THE COVERED CODE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE COVERED CODE WILL BE CORRECTED. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY APPLE, AN APPLE AUTHORIZED REPRESENTATIVE OR ANY CONTRIBUTOR SHALL CREATE A WARRANTY. You acknowledge that the Covered Code is not intended for use in the operation of nuclear facilities, aircraft navigation, communication systems, or air traffic control machines in which case the failure of the Covered Code could lead to death, personal injury, or severe physical or environmental damage.

9. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT SHALL APPLE OR ANY CONTRIBUTOR BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO THIS LICENSE OR YOUR USE OR INABILITY TO USE THE COVERED CODE, OR ANY PORTION THEREOF, WHETHER UNDER A THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCTS LIABILITY OR OTHERWISE, EVEN IF APPLE OR SUCH CONTRIBUTOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND NOTWITHSTANDING THE FAILURE OF ESSENTIAL PURPOSE OF ANY REMEDY. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF LIABILITY OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS

LIMITATION MAY NOT APPLY TO YOU. In no event shall Apple's total liability to You for all damages (other than as may be required by applicable law) under this License exceed the amount of fifty dollars ($50.00).

10. Trademarks. This License does not grant any rights to use the trademarks or trade names "Apple", "Apple Computer", "Mac", "Mac OS", "QuickTime", "QuickTime Streaming Server" or any other trademarks, service marks, logos or trade names belonging to Apple (collectively "Apple Marks") or to any trademark, service mark, logo or trade name belonging to any Contributor. You agree not to use any Apple Marks in or as part of the name of products derived from the Original Code or to endorse or promote products derived from the Original Code other than as expressly permitted by and in strict compliance at all times with Apple's third party trademark usage guidelines which are posted at

 http://www.apple.com/legal/guidelinesfor3rdparties.html.

11. Ownership. Subject to the licenses granted under this License, each Contributor retains all rights, title and interest in and to any Modifications made by such Contributor. Apple retains all rights, title and interest in and to the Original Code and any Modifications made by or on behalf of Apple ("Apple Modifications"), and such Apple Modifications will not be automatically subject to this License. Apple may, at its sole discretion, choose to license such Apple Modifications under this License, or on different terms from those contained in this License or may choose not to license them at all.

12. Termination.

12.1 Termination. This License and the rights granted hereunder will terminate:

(a) automatically without notice from Apple if You fail to comply with any term(s) of this License and fail to cure such breach within 30 days of becoming aware of such breach;

(b) immediately in the event of the circumstances described in Section 13.5(b); or

(c) automatically without notice from Apple if You, at any time during the term of this License, commence an action for patent infringement against Apple; provided that Apple did not first commence an action for patent infringement against You in that instance.

12.2 Effect of Termination. Upon termination, You agree to immediately stop any further use, reproduction, modification, sublicensing and distribution of the Covered Code. All sublicenses to the Covered Code which have been properly granted prior to termination shall survive any termination of this License. Provisions which, by their nature, should remain in effect beyond the termination of this License shall survive, including but not limited to Sections 3, 5, 8, 9, 10, 11, 12.2 and 13. No party will be liable to any other for compensation, indemnity or damages of any sort solely as a result of terminating this License in accordance with its terms, and termination of this

License will be without prejudice to any other right or remedy of any party.

13. Miscellaneous.

13.1 Government End Users. The Covered Code is a "commercial item" as defined in FAR 2.101. Government software and technical data rights in the Covered Code include only those rights customarily provided to the public as defined in this License. This customary commercial license in technical data and software is provided in accordance with FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for Department of Defense purchases, DFAR

252.227-7015 (Technical Data -- Commercial Items) and 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation). Accordingly, all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

13.2 Relationship of Parties. This License will not be construed as creating an agency, partnership, joint venture or any other form of legal association between or among You, Apple or any Contributor, and You will not represent to the contrary, whether expressly, by implication, appearance or otherwise.

13.3 Independent Development. Nothing in this License will impair Apple's right to acquire, license, develop, have others develop for it, market and/or distribute technology or products that perform the same or similar functions as, or otherwise compete with, Modifications, Larger Works, technology or products that You may develop, produce, market or distribute.

13.4 Waiver; Construction. Failure by Apple or any Contributor to enforce any provision of this License will not be deemed a waiver of future enforcement of that or any other provision. Any law or regulation which provides that the language of a contract shall be construed against the drafter will not apply to this License.

13.5 Severability. (a) If for any reason a court of competent jurisdiction finds any provision of this License, or portion thereof, to be unenforceable, that provision of the License will be enforced to the maximum extent permissible so as to effect the economic benefits and intent of the parties, and the remainder of this License will continue in full force and effect. (b) Notwithstanding the foregoing, if applicable law prohibits or restricts You from fully and/or specifically complying with Sections 2 and/or 3 or prevents the enforceability of either of those Sections, this License will immediately terminate and You must immediately discontinue any use of the Covered Code and destroy all copies of it that are in your possession or control.

13.6 Dispute Resolution. Any litigation or other dispute resolution between You and Apple relating to this License shall take place in the Northern District of California, and You and Apple hereby consent to the personal jurisdiction of, and venue in, the state and federal courts within that District with respect to this License. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded.

13.7 Entire Agreement; Governing Law. This License constitutes the entire agreement between the parties with respect to the subject matter hereof. This License shall be governed by the laws of the United States and the State of California, except that body of California law concerning conflicts of law. Where You are located in the province of Quebec, Canada, the following clause applies: The parties hereby confirm that they have requested that this License and all related documents be drafted in English. Lesparties ont exig que le prsent contrat et tous les documents connexes soient rdigs en anglais.

EXHIBIT A.

"Portions Copyright (c) 1999-2003 Apple Computer, Inc. All Rights Reserved. This file contains Original Code and/or Modifications of Original Code as defined in and that are subject to the Apple Public Source License Version 2.0 (the 'License'). You may not use this file except in compliance with the License. Please obtain a copy of the License at http://www.opensource.apple.com/apsl/ and read it before using this file.

The Original Code and all software distributed under the License are distributed on an 'AS IS' basis, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, AND

APPLE HEREBY DISCLAIMS ALL SUCH WARRANTIES, INCLUDING WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT OR NON-INFRINGEMENT. Please see the License for the specific language governing rights and limitations under the License."

OpenSSH

This file is part of the OpenSSH software.

The licences which components of this software fall under are as follows. First, we will summarize and say that all components are under a BSD licence, or a licence more free than that.

OpenSSH contains no GPL code.

1)

```
* Copyright (c) 1995 Tatu Ylonen <ylo@cs.hut.fi>, Espoo, Finland
*                   All rights reserved
*
* As far as I am concerned, the code I have written for this software
* can be used freely for any purpose.  Any derived versions of this
* software must be clearly marked as such, and if the derived work is
* incompatible with the protocol description in the RFC file, it must be
* called by a name other than "ssh" or "Secure Shell".

* However, I am not implying to give any licenses to any patents or
* copyrights held by third parties, and the software includes parts that
* are not under my direct control.  As far as I know, all included
* source code is used in accordance with the relevant license agreements
* and can be used freely for any purpose (the GNU license being the most
* restrictive); see below for details.
```

   [However, none of that term is relevant at this point in time. All of these restrictively licenced software components which he talks about have been removed from OpenSSH, i.e.,

```
- RSA is no longer included, found in the OpenSSL library
- IDEA is no longer included, its use is deprecated
- DES is now external, in the OpenSSL library
- GMP is no longer used, and instead we call BN code from OpenSSL
- Zlib is now external, in a library
- The make-ssh-known-hosts script is no longer included
- TSS has been removed
- MD5 is now external, in the OpenSSL library
- RC4 support has been replaced with ARC4 support from OpenSSL
- Blowfish is now external, in the OpenSSL library
```

Note that any information and cryptographic algorithms used in this software are publicly available on the Internet and at any major bookstore, scientific library, and patent office worldwide. More information can be found e.g. at "http://www.cs.hut.fi/crypto".

The legal status of this program is some combination of all these permissions and restrictions. Use only at your own responsibility. You will be responsible for any legal consequences yourself; I am not making any claims whether possessing or using this is legal or not in your country, and I am not taking any responsibility on your behalf.

NO WARRANTY

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WAR-
RANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS
AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PAR-
TICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE
OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE,
YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORREC-
TION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRIT-
ING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO
YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CON-
SEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR
A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN
IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

2) The 32-bit CRC compensation attack detector in deattack.c was contributed by CORE SDI
S.A. under a BSD-style license.

```
* Cryptographic attack detector for ssh - source code
*
* Copyright (c) 1998 CORE SDI S.A., Buenos Aires, Argentina.
*
* All rights reserved. Redistribution and use in source and binary
* forms, with or without modification, are permitted provided that
* this copyright notice is retained.
*
* THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS
* OR IMPLIED WARRANTIES ARE DISCLAIMED. IN NO EVENT
* SHALL CORE SDI S.A. BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL
* DAMAGES RESULTING FROM  THE USE OR MISUSE OF THIS
* SOFTWARE.
*
* Ariel Futoransky <futo@core-sdi.com>
* <http://www.core-sdi.com>
```

3) ssh-keyscan was contributed by David Mazieres under a BSD-style license.

```
* Copyright 1995, 1996 by David Mazieres <dm@lcs.mit.edu>.
*
* Modification and redistribution in source and binary forms is
* permitted provided that due credit is given to the author and the
* OpenBSD project by leaving this copyright notice intact.
```

4) The Rijndael implementation by Vincent Rijmen, Antoon Bosselaers and Paulo Barreto is in
the public domain and distributed with the following license:

```
* @version 3.0 (December 2000)
*
```

```
* Optimised ANSI C code for the Rijndael cipher (now AES)
*
* @author Vincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
* @author Antoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
* @author Paulo Barreto <paulo.barreto@terra.com.br>
*
* This code is hereby placed in the public domain.
*
* THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
* NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT-
* ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL THE AUTHORS OR
* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
*  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCURE-
* MENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLI-
* GENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
* USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
```

5) One component of the ssh source code is under a 3-clause BSD license, held by the University of California, since we pulled these parts from original Berkeley code.

```
* Copyright (c) 1983, 1990, 1992, 1993, 1995
*     The Regents of the University of California.  All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
* 3. Neither the name of the University nor the names of its contributors
*    may be used to endorse or promote products derived from this software
*    without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE REGENTS AND
* CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND
* FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
* IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
* BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION)  HOWEVER CAUSED AND ON ANY THEORY
* OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
* SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
```

6) Remaining components of the software are provided under a standard 2-term BSD licence with the following names as copyright holders:

Markus Friedl
Theo de Raadt
Niels Provos
Dug Song
Aaron Campbell
Damien Miller
Kevin Steves
Daniel Kouril
Wesley Griffin
Per Allansson
Nils Nordman
Simon Wilkinson

Portable OpenSSH additionally includes code from the following copyright holders, also under the 2-term BSD license:

Ben Lindstrom
Tim Rice
Andre Lucas
Chris Adams
Corinna Vinschen
Cray Inc.
Denis Parker
Gert Doering
Jakob Schlyter
Jason Downs
Juha Yrjl
Michael Stone
Networks Associates Technology, Inc.
Solar Designer
Todd C. Miller
Wayne Schroeder
William Jones
Darren Tucker
Sun Microsystems
The SCO Group
Daniel Walsh

```
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the above copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.
*
* THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
* NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANT-
* ABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DIS-
* CLAIMED.  IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
* ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
* OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMIT-
* ED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTER-
* RUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
```

* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
* TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
* ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
* ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8) Portable OpenSSH contains the following additional licenses:

a) md5crypt.c, md5crypt.h

* "THE BEER-WARE LICENSE" (Revision 42):
* <phk@login.dknet.dk> wrote this file. As long as you retain this
* notice you can do whatever you want with this stuff. If we meet
* some day, and you think this stuff is worth it, you can buy me a
* beer in return.   Poul-Henning Kamp

b) snprintf replacement

* Copyright Patrick Powell 1995
* This code is based on code written by Patrick Powell
* (papowell@astart.com) It may be used for any purpose as long as this
* notice remains intact on all source code distributions

c) Compatibility code (openbsd-compat)

Apart from the previously mentioned licenses, various pieces of code
in the openbsd-compat/ subdirectory are licensed as follows:

Some code is licensed under a 3-term BSD license, to the following   copyright
holders:

Todd C. Miller
Theo de Raadt
Damien Miller
Eric P. Allman
The Regents of the University of California
Constantin S. Svintsoff

* Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:

* 1. Redistributions of source code must retain the above copyright
*   notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in the
*   documentation and/or other materials provided with the distribution.
* 3. Neither the name of the University nor the names of its contributors
*   may be used to endorse or promote products derived from this software
*   without specific prior written permission.
* THIS SOFTWARE IS PROVIDED BY THE REGENTS AND
* CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED
* WARRANTIES, INCLUDING, BUT NOT LIMITED, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND
* FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS
* OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCURE-
* MENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
* LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

```
* OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
* THE POSSIBILITY OF SUCH DAMAGE.
```

Some code is licensed under an ISC-style license, to the following copyright holders:

Internet Software Consortium.
Todd C. Miller
Reyk Floeter
Chad Mynhier

```
* Permission to use, copy, modify, and distribute this software for any
* purpose with or without fee is hereby granted, provided that the above
* copyright notice and this permission notice appear in all copies.
*
* THE SOFTWARE IS PROVIDED "AS IS" AND TODD C. MILLER
* DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
* SOFTWARE INCLUDING ALL IMPLIED WARRANTIES
* OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL
* TODD C. MILLER BE LIABLE FOR ANY SPECIAL, DIRECT,
* INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY
* DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE,
* DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
* NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
* OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
* THIS SOFTWARE.
```

Some code is licensed under a MIT-style license to the following copyright holders:

Free Software Foundation, Inc.

```
* Permission is hereby granted, free of charge, to any person obtaining a
* copy of this software and associated documentation files (the
* "Software"), to deal in the Software without restriction, including
* without limitation the rights to use, copy, modify, merge, publish,
* distribute, distribute with modifications, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included
* in all copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY
* O F ANY KIND, EXPRESS  OR IMPLIED, INCLUDING BUT NOT
* LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND
* NONINFRINGEMENT.   IN NO EVENT SHALL THE ABOVE
* COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
* DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
* CONTRACT, TORT OR
* OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
* WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
* THE SOFTWARE.
*
* Except as contained in this notice, the name(s) of the above copyright
* holders shall not be used in advertising or otherwise to promote the
* sale, use or other dealings in this Software without prior written
* authorization.
*************************************************************
```

OpenSSL

LICENSE ISSUES

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

```
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL
 * PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
```

* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
* ==
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*   notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*   notice, this list of conditions and the following disclaimer in the
*   documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software

```
*    must display the following acknowledgement:
*    "This product includes cryptographic software written by
*     Eric Young (eay@cryptsoft.com)"
*    The word 'cryptographic' can be left out if the rouines from the library
*    being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
*    the apps directory (application code) you must include an acknowledgement:
*    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS
* OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
* PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR
* OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed.  i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/
```

Free License Agreement

INDEX

....

Part Numbers

A

Index

"View / Edit Accessory Cals" menu item 18-8
"View/edit..." menu item 16-11
"View Factory Cal" menu item 27-81
"View" fct key 10-13
"View File" fct key 12-2
"View Fsh/Drk" fct key 27-24
"view Graph" fct key 27-24
"View History" menu item 18-5
"view Regrs" fct key 27-24
"View Stored Images" menu item 21-15
"ViewSrc" fct key 13-3
vinyl gasket 19-37
von Caemmerer, S. 1-7
"VPD=" fct key 7-11
"VpdA" label 3-21
"VpdL" label 3-21

W

wall temperature 17-2
warning messages
    battery 5-18
    New Measurements 20-5
water corrections
    band broadening 14-27
    dilution 1-10
water use efficiency example 3-100
Weis, E. 27-93
Welles, J. M. 27-93
"What's What" fct key 6-6
white gaskets. See gaskets, polyethylene
Whole Plant Chamber
    configuration 16-52
    description
wild card characters 10-11
Williams, D. 14-33

Windows
    apps 11-2
WinSCP 11-14
Wolfe, W.L. 14-33

Y

"YYYYMMDD" label 14-15

Z

zero
    definition 18-10
    flow meter 18-23
    leaf temperature 18-24
    manual IRGA 18-16
    setting CO2 and H2O 18-11
zero drift
    IRGA 14-7, 14-9
"Zero flr. signal" menu item 27-81
Zissis, G.J. 14-33
"ZOOM IN" fct key 6-19
"ZOOM OUT" fct key 6-19
"zoom topOne%" fct key 27-24
"zoom topTen%" fct key 27-24