Using the LI-6400

Portable Photosynthesis System

Version 5

LI-COR Biosciences

NOTICE

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

LI-COR MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. LI-COR shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Publication Number 9806-122
Printing History:
1st Printing July, 1998 - OPEN Software version 3.2
2nd Printing May, 1999 - OPEN Software version 3.3
3rd Printing Sept, 2000 - OPEN Software version 3.4
4th Printing Sept, 2001 - OPEN Software version 4.0
5th Printing July, 2002 - OPEN Software version 5.0
6th Printing Mar, 2003 - OPEN Software version 5.1
7th Printing June 2004 - OPEN Software version 5.2
8th Printing June 2005 - OPEN Software version 5.3

Macintosh® is a registered trademark of Apple Computer, 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-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

Part III: Working With Files

Part IV: Configuration Issues

Part V: Maintenance & Troubleshooting

18 Calibration Issues

19 Maintenance & Service

Part VII: Accessories

Contents

Welcome to the LI-6400

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-

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. Old units are upgradable, so 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-6400.

[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 27 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 5 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.
  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 4.0 of Adobe® Acrobat® Reader is required. You can download this software for no charge from www.adobe.com)

Version 5 Summary

There are a number of interface changes that came with OPEN 5.0, 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.

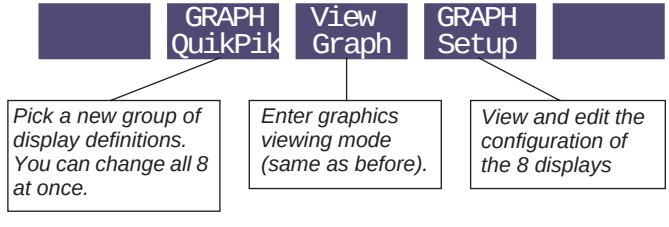
Version 5 runs on new hardware; there are 128 Mbytes of dynamic memory, and 64 Mbytes of flash memory (for files). The processor is faster (used to be 16 MHz, 16 bit - now it's 200 MHz, 32 bit), and the underlying operating system is now LINUX. Overall operation is much snappier, with minimal delays. And, no more coffee breaks while OPEN loads - the infamous bar chart now goes by in 5 seconds.

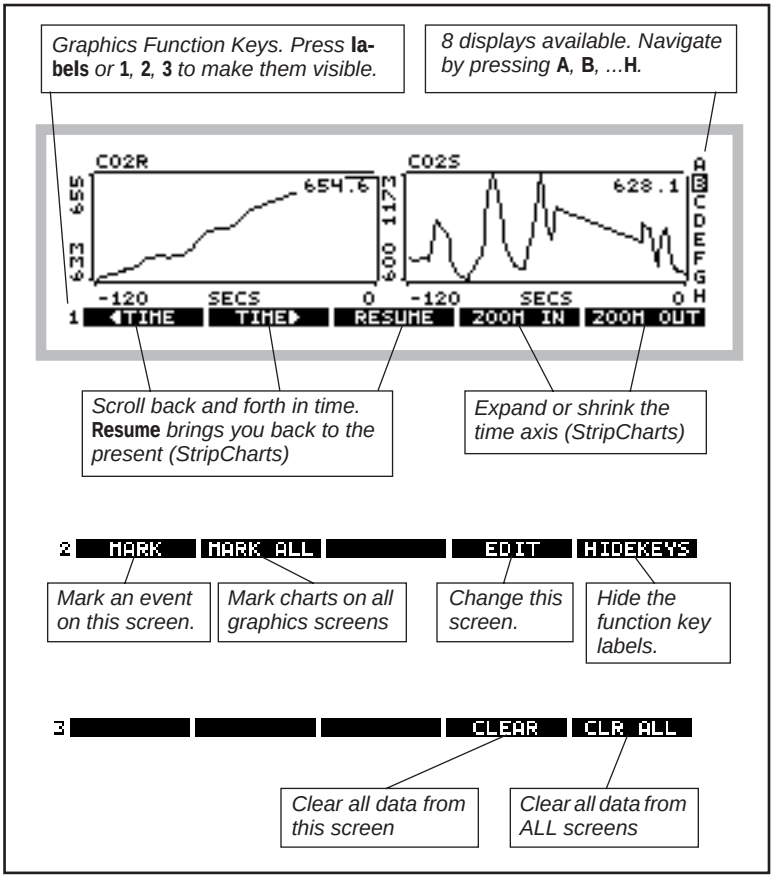
Real Time Graphics

There are now 8 independent graphics displays for showing real time graphics in New Measurements mode, instead of just one. Each display can hold 1, 2, or 3 independent strip charts or XY plots, as before. However, the scaling options are improved, and strip charts retain 10 minutes (or more - it's user definable) worth of data, allowing you to scroll back in time, or zoom in and out (i.e. change time intervals). A typical real time graphics display is illustrated on the next page.

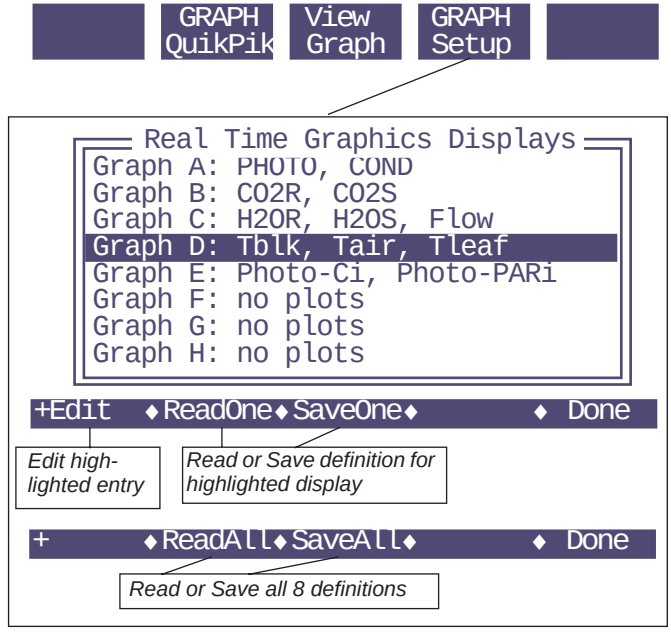
Notice that there are 3 levels of function keys available while viewing the graph for control and manipulation. Also, notice the A, B, C... indicator on the right hand side. This shows which display is being viewed. Press the letters **A** through **H** to jump directly to a particular display. You can also use the arrow keys ↑ or ↓ to step through them sequentially.

In New Measurements mode, the level 4 function keys still control real time graphics, but now they look like this:

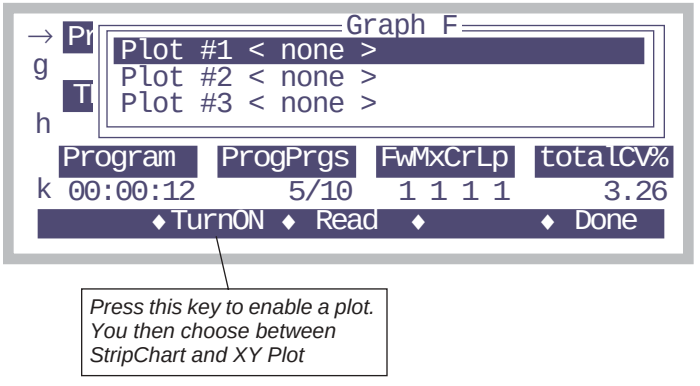


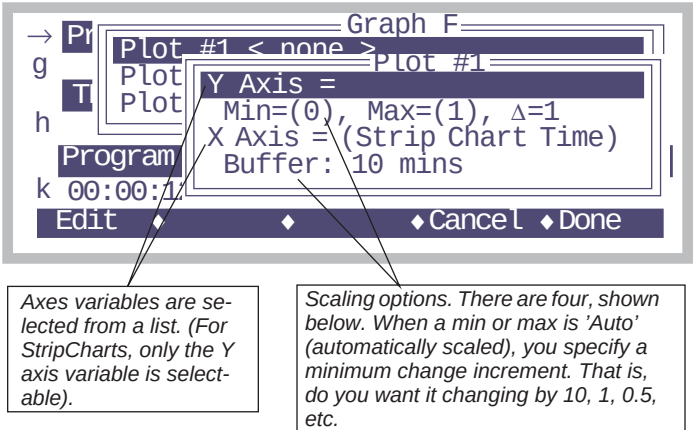

Graphics Function Keys. Press labels or 1, 2, 3 to make them visible.
8 displays available. Navigate by pressing A, B, ...H.
CO2R
654.6
655
633
CO2S
628.1
1173
600
A
B
C
D
E
F
G
H
-120 SECS 0 -120 SECS 0
1 ◄TIME TIME► RESUME ZOOM IN ZOOM OUT
Scroll back and forth in time. Resume brings you back to the present (StripCharts)
Expand or shrink the time axis (StripCharts)
2 MARK MARK ALL EDIT HIDEKEYS
Mark an event on this screen.
Mark charts on all graphics screens
Change this screen.
Hide the function key labels.
3 CLEAR CLR ALL
Clear all data from this screen
Clear all data from ALL screens

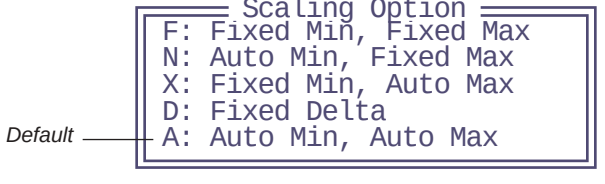
The **GRAPH Setup** key allows the 8 real time graphics displays to be configured individually, or all at once.



Editing a particular graphics display brings up an editor that is illustrated on the next page. This is also the editor accessed by pressing the EDIT function key while viewing a real time graphics display.


Graph F
Plot #1 < none >
Plot #2 < none >
Plot #3 < none >
Program
ProgPrgs
FwMxCrLp
totalCV%
00:00:12
5/10
1 1 1 1
3.26
♦TurnON ♦ Read ♦ ♦ Done
Press this key to enable a plot. You then choose between StripChart and XY Plot

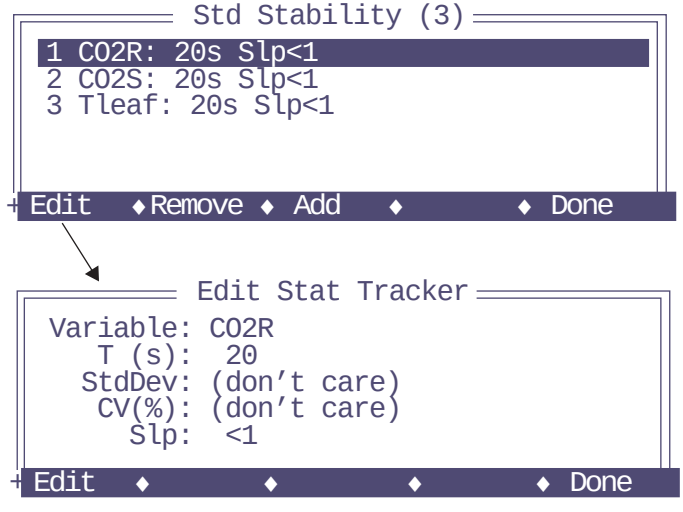

Graph F
Plot #1
Y Axis =
Min=(0), Max=(1), Δ=1
X Axis = (Strip Chart Time)
Buffer: 10 mins
Edit ♦ ♦ ♦Cancel ♦Done
Axes variables are selected from a list. (For StripCharts, only the Y axis variable is selectable).
Scaling options. There are four, shown below. When a min or max is 'Auto' (automatically scaled), you specify a minimum change increment. That is, do you want it changing by 10, 1, 0.5, etc.
For StripCharts, you can pick how many minutes of data to retain.


Scaling Option
F: Fixed Min, Fixed Max
N: Auto Min, Fixed Max
X: Fixed Min, Auto Max
D: Fixed Delta
A: Auto Min, Auto Max
Default

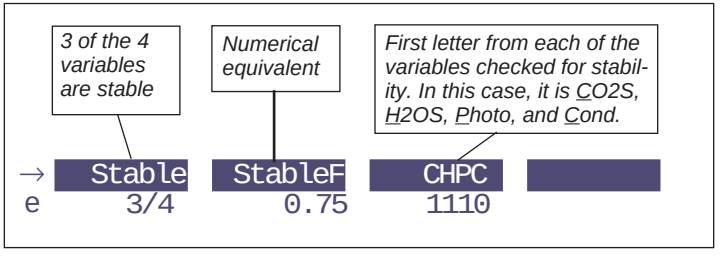
Stability Testing

System stability was formerly determined by the coefficients or variation of flow and the IRGA differentials. In Version 5, it is completely user definable. You can pick a list of variables (system or user) and the stability criteria for each, including rate of change, standard deviation, and/or coefficient of variation.

In New Measurements mode, there is a function key on level 5 named **Define Stability**. This accesses a list of the current stability criteria, which you can edit, store, read, etc. Stability criteria can be stored as part of a system configuration.



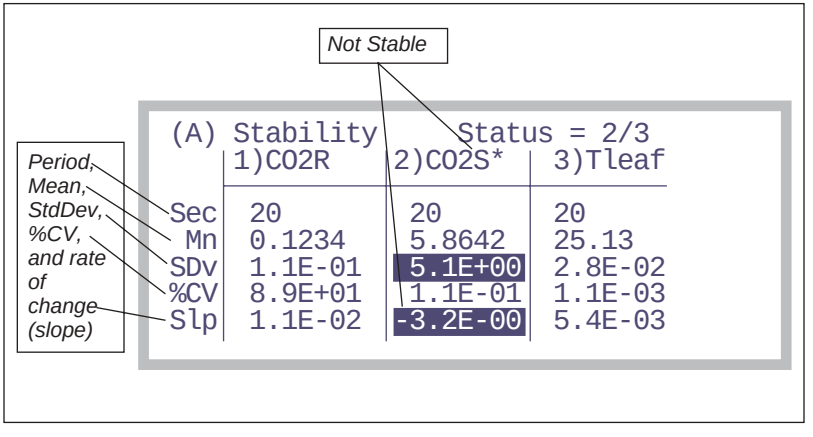
You can monitor some variables in New Measurements mode that indicate the state of system stability. In addition to this, statistical details can be viewed in a diagnostic screen, described in the next section.



Diagnostic Screens

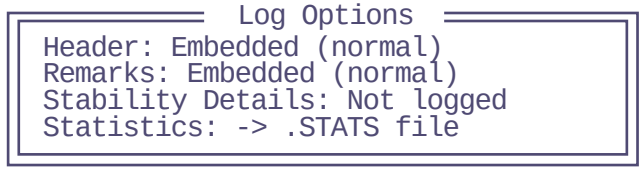
In New Measurements mode, there is a new set of screens called Diagnostic Screens, that provide some behind-the-scenes information that is not normally available for viewing. To get there, press the left square bracket key ([). There are 7 of diagnostic screens, labelled A through G, and you can navigate by typing **A** through **G**. When done, press **[** or **escape** to leave.

Perhaps the most useful diagnostic screen is *A*, from which you can view all of the system stability information (discussed above).



Log Options

There is a dialog named Log Options, available from the Logging Control Dialog, or from the New Measurements screen (**f4** level 5). (The key in either location is shown only if there is no log file open. You can't change your log options in the middle of logging.)



Header

The choice is "Embedded (normal)" or "Separate .HDR file". The latter case puts all the header entries (which normally go before the column labels) into a separate file. This file has the same name as your log file, but with .HDR appended.

Remarks

The choice is "Embedded (normal)" or "Separate .REM file". The latter case puts all remarks (including fluorescence actions) into a separate file (same name as log file, but with .REM appended), instead of interspersing them in the data records.

Stability Details

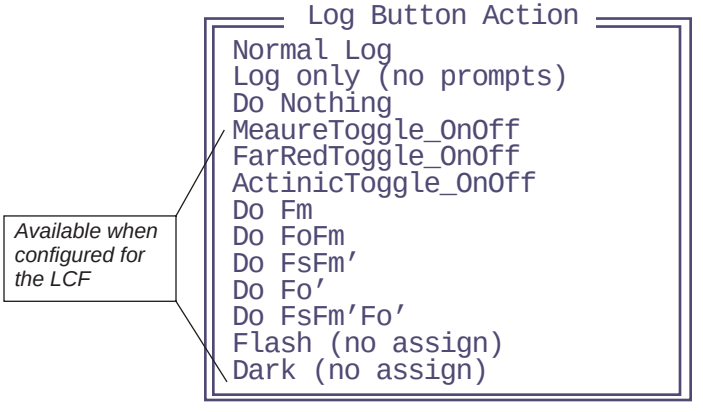
The choice is "Not logged", or "Logged". Stability details refers to the list of variables that you are using to define system stability. Setting this option to "Logged" will cause extra data columns to be appended in your log file. For each variable in the stability list, you'll get a sample count, mean, standard deviation, coefficient of variation, and the rate of change. Each has a column label, so you'll know what they are.

Statistics

Choices are "None (normal)" or "-> .STATS file". The latter case will cause statistics (mean and standard deviation over a user-entered time period) to be computed for each variable in your log list. This information is logged in a separate file (same name as log file, but with .STATS appended).

User Definable Log Button

What the log button does when pressed (while in New Measurements mode) is now user definable. **f5** level 5 lets you select possible actions from a menu of choices:



The Filer

The file system has changed, and no longer requires periodic defragmentation. Purging files moves them to a trash directory, and when the trash is emptied, the space is recovered.

The Filer function keys are shown below:

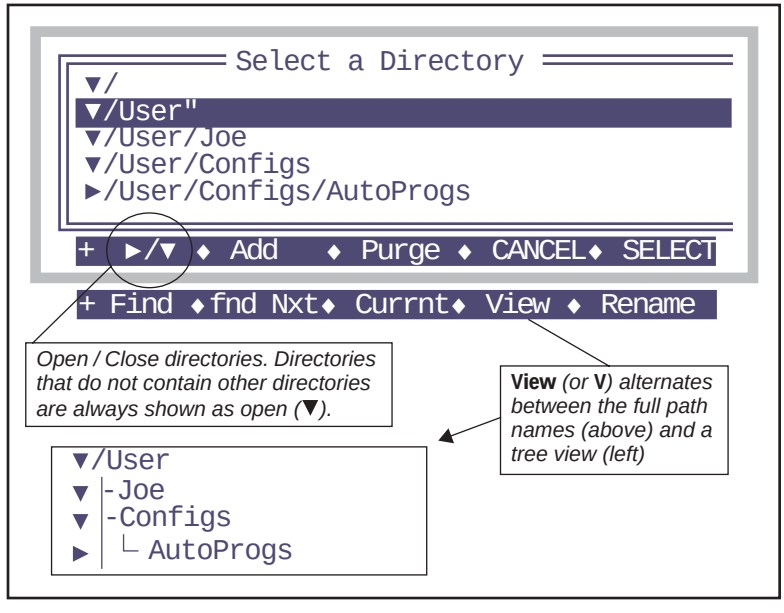
```
+ Dir   ♦ fiLter♦ Sort  ♦ Tag   ♦ purGe
+ Copy  ♦ Move  ♦dUplct ♦ Rename♦ Print
+ View  ♦ Bview ♦ Edit  ♦ grapH ♦Working
+ space ♦ trAsh ♦eXecute♦ Find  ♦re-scaN
```

Changes are:

fiLter Short-cut key changed from F to L

Bview	Binary view. For viewing binary files, or text files. Shows hex codes for each byte of the file.
space	Gets the space on both file system partitions.
trAsh	Empty the trash.
Find	Find (global search) a file name.

The dialog for choosing directories has a slightly different look. It can show directories in a tree format or traditional full name format. Also, directories can be expanded or collapsed.



Version 5.2 Summary

Version 5.2 brought additional changes:

Backlight ON at startup

A new program has been added to the folder "/Sys/Autostart" named BackLightON. (All files in this folder execute when LPL starts running at power

up.) BackLightON is a very short program that simply turns on the display backlight (if present).

```
:FCT main
{
  1 BACKLIGHT
}
```

If you do not wish for this to happen, edit BackLightON, and change the 1 to a 0. Or, simply move the file to the trash.

The backlight can be toggled on and off at any time by pressing **ctrl shift home** simultaneously.

New System Variable: TotalCV

New system variable *TotalCV* (id number -74) has been added in version 5.2. It is on display line 'e' by default.

TotalCV is the sum of the CVs (%) of all stability variables presently defined. For more on stability variables, see **Stability Variables** on page 14-11.

New System Constant: Oxygen%

Another new system variable (actually a user-entered constant) is *Oxygen%* (id number -52). This value defaults to 21, but can be adjusted by you by putting it into the prompts list (**Prompt Control** on page 9-15). You can then be prompted for it in New Measurements mode (**f5** level 3).

The purpose of *Oxygen%* is to account for O_2 concentration in the computation of *CO2R* and *CO2S*, the reference and sample CO_2 concentrations, and on *H2OR* and *H2OS*, the reference and sample water vapor concentrations.

The impact of O_2 on the CO_2 measurement is a band broadening effect that fits the theory fairly well (See **Carbon Dioxide Concentrations** on page 14-6, and **Band Broadening Corrections** on page 14-23). For water vapor, with its much more saturated bands, the effect is entirely empirically determined. See **Water Concentrations** on page 14-5.

New Configuration Parameter: LogOption=

The Log Options (**Log Options** on page 9-9) can now be made part of a configuration file, and changes to them (done in New Measurements mode or from Logging Control) will trigger the CONFIG change flag in Open's Main Screen.

Fan Control in IRGA Zero and Span

Chamber fan control has been added to the IRGA zero and span routines. This has been done to allow these routines to be used with large chambers which are attached directly to the IRGA sample cell. In this configuration, the chamber fan is the connection between the chamber's volume and the IRGA's cell. Turning the fan off allows the cell to measure incoming concentration with little influence from the chamber.

Typing 'f' will increment the fan speed through "Fast", "Slow", and "off". See **CO2 and H2O Analyzers** on page 18-3.

Soil Respiration Enhancements

The soil respiration chamber configuration has several improvements.

Logging Rate

Prior to version 5.2, raw readings were logged at 2 Hz, and an internal buffer of 50 pairs (rate of change and concentration) was filling up in about 30 seconds, leading to early termination of each cycle in some low flux / large delta cases. In 5.2, logging occurs once every two seconds, and the buffer size is much larger, so early termination should not be a problem.

All Settings in One Place

Prior to 5.2, the soil respiration configuration had settings accessed via several different function keys. Now, you can access them all from one place: **f5** level 7

Storable Settings

In addition to being in one place, these settings are now storable. This feature is also accessed via **f5** level 7.

These settings are saved (by default) in a new directory named /User/Configs/SoilParams. The default file is named /User/Configs/SoilParams/Std Settings, as defined in the default soil respiration configuration file:

```
SoilParams= "Std Settings"
```

If you use Save or Open in the soil settings editor to change this file, or if you make a change to one or more of these settings and don't update the file, the CONFIG flag will appear in the main screen, and the Config Status display will show the reason: an unsaved file, or a different file name.

New Configuration: Custom Chamber - Closed

A new configuration builder is available in the Installation menu, named "Custom Chamber - Closed". The fundamental assumption here is that flux measurements are to be made in a closed (non-flow through) chamber based on a time rate of change of CO_2. The program will guide you through a series of questions, and build an appropriate configuration based on your responses. See **Custom Chamber - Closed** on page 16-20

New Configuration Parameter: MatchType=

OPEN 5.2 adds the option of an alternative matching scheme, useful for turf and soil chambers operating in open mode. See **Matching Without the Match Valve** on page 16-34.

Version 5.3 Summary

MultiLevel Flash Option for the LCF

Version 5.3 adds support for doing multilevel saturation flashes with the 6400-40 LCF. This technique allows the fluorescence at saturation to be measured without actually achieving saturation. This type of flash has three different intensity levels, and the saturated value of *F* is computed by plotting *F* against 1/*Q*, and finding the 1/*Q* intercept. See **Saturating Flashes** on page 27-22.

Interfacing Options

Version 5.3 add remote control capability, using a computer connected to the RS-232 port, or over the internet. See **Control via Terminal Software (LTerm / iTerm)** on page 11-16.

Version 5.3 makes it very simple to incorporate data coming in the RS-232 port into user variables in OPEN. See **Importing RS-232 Data** on page 15-14.

Incoming data can also be treated as commands to be compiled and executed. This allows other devices to exchange data with - as well as have some control over - the LI-6400. See **Control via LPL Commands** on page 11-20.

Minor Changes

Version 5.3 has a number of minor changes, including

- **Tab is the default**
  The tab character is now the default delimiter in log files, instead of a comma. This can be changed with a configuration command, if you wish.

- **Logged Data can be simultaneously echoed to the Comm Port**
  Prior to version 5.3, you could log to a file *or* to the Comm port. Now you can do both. See **Using the Comm Port** on page 11-4.

Part I

The Basics

System Description

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

1 System Description

This chapter acquaints you with the LI-6400's operating principle, major components, and equations.

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).

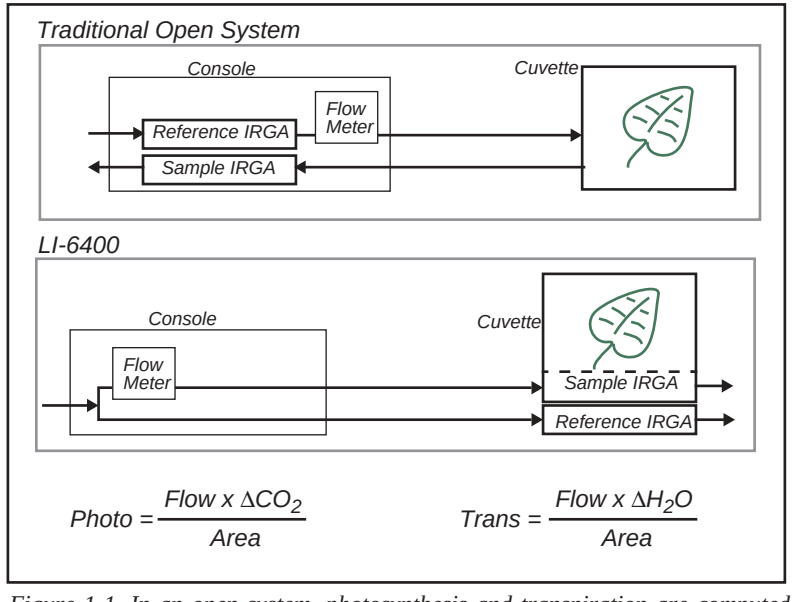


Figure 1-1. In an open system, photosynthesis and transpiration are computed from the differences in CO_2 and H_2O between in-chamber conditions and pre-chamber conditions. The equations are given in ***Equation Summary*** *on page 1-7.*

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

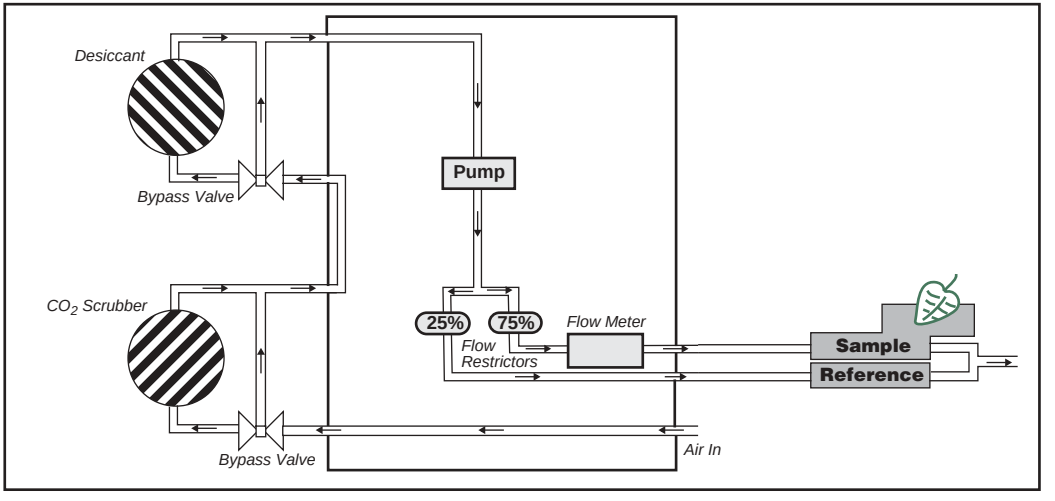
diffusivity to CO_2 of all the gasket material we have tested, but it's not perfect. Diffusion effects are measurable at low flow rates when there is a large concentration difference between chamber and ambient. See **Diffusion Leaks** on page 4-43 for more details.

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 4-48.

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

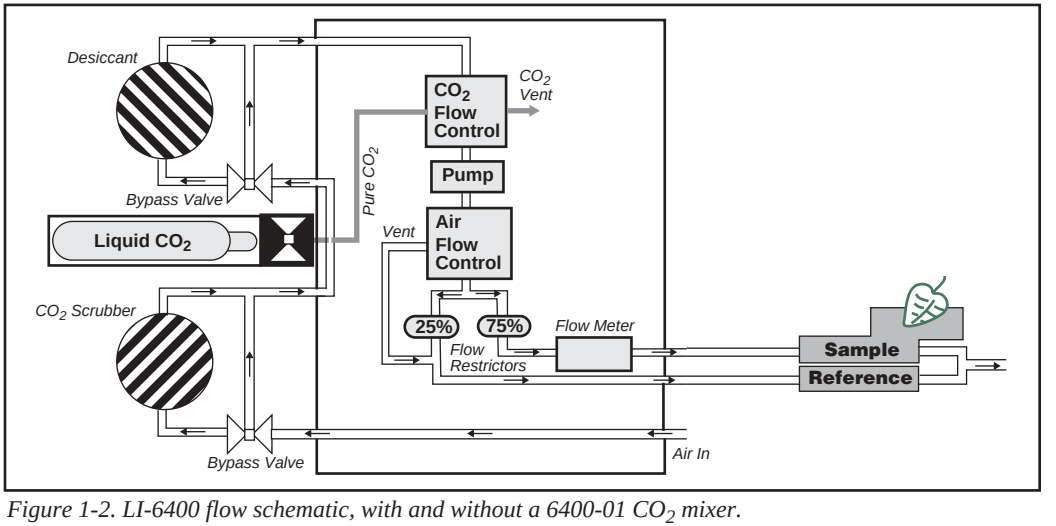


Figure 1-2. LI-6400 flow schematic, with and without a 6400-01 CO_2 mixer.

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.

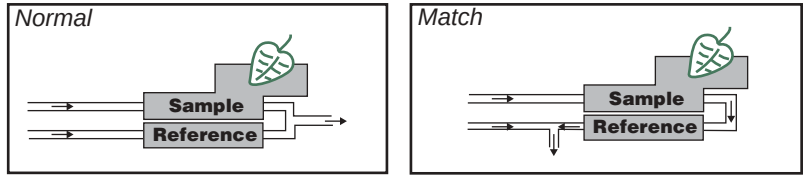


Figure 1-3. In Match Mode, leaf chamber air is measured by both sample and reference IRGAs, allowing the sample IRGA to be adjusted to match the reference IRGA.

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 µmol 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 µmol 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

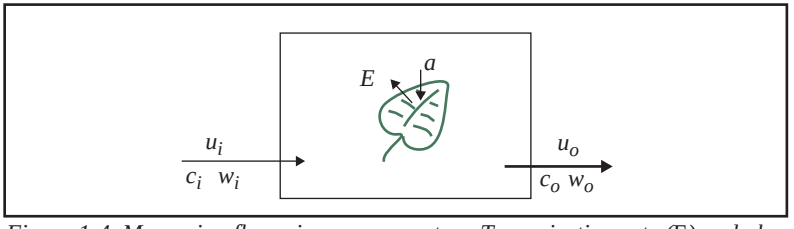


*Figure 1-4. Measuring fluxes in an open system. Transpiration rate (*E*) and photosynthetic rate (*a*) change the water and CO_2 concentrations of air as it passes through the chamber. Transpiration also causes the exit flow* u_o *to be greater than the incoming flow rate (*u_i*).*

$$sE = u_o w_o - u_i w_i \qquad (1\text{-}1)$$

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

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

we can write

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

---

[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

$$E = \frac{u_i(w_o - w_i)}{s(1 - w_o)} \quad (1\text{-}4)$$

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

$$\begin{aligned} u_i &= F/10^6 \\ w_i &= W_r/10^3 \\ w_o &= W_s/10^3 \\ s &= S/10^4 \end{aligned} \quad (1\text{-}5)$$

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

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

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

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

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)

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

The function *e*(T) is saturation vapor pressure (kPa) at temperature *T* (C). The formula used by the LI-6400 is (14-22) on page 14-10.

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.

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

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

$$sa = u_i c_i - u_o c_o \qquad (1\text{-}11)$$

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

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

which rearranges to

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

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

$$\begin{aligned} c_i &= C_r / 10^6 \\ c_o &= C_s / 10^6 \\ a &= A / 10^6 \end{aligned} \qquad (1\text{-}14)$$

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

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

Why does transpiration 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

$$C_i = \frac{\left(g_{tc} - \frac{E}{2}\right)C_s - A}{g_{tc} + \frac{E}{2}} \qquad (1\text{-}16)$$

where g_{tc} is the total conductance to CO_2, and is given by

$$g_{tc} = \frac{1}{\frac{1.6}{g_{sw}} + \frac{1.37k_f}{g_{bw}}} \qquad (1\text{-}17)$$

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

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

w_i = 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-6400 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:

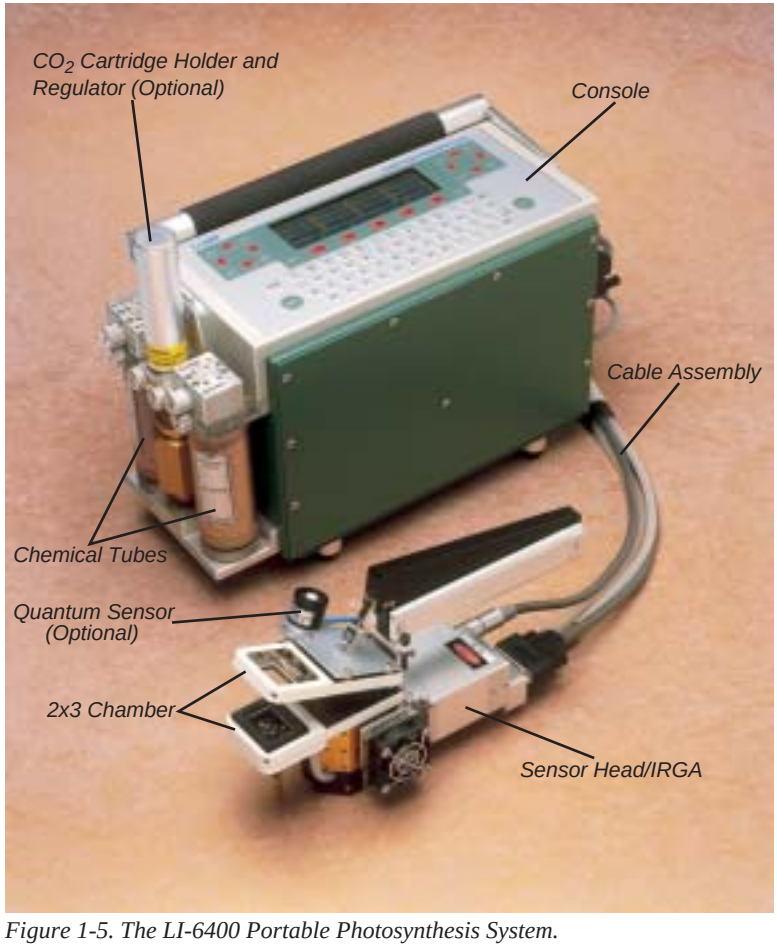


Figure 1-5. The LI-6400 Portable Photosynthesis System.

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.

RS-232C Cable

Part number 9975-016 has a 9 pin to 9 pin cable, and a separate 9 to 25 pin adapter. Figure 11-1 on page 11-3 illustrates how to use it.

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.

6400-55 CD

The CD contains a hyper-linked, electronic version (.pdf file) of this manual, plus a number of Windows® and Macintosh® support programs for tasks such as (re-)installing instrument software, running the LI-6400 from a computer, and doing data file transfers. Note: New versions of this 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 CO_2 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 8-7 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.)

6400-06 PAM 2000 Adaptor

This chamber top is designed for 8 mm fluorometer probes such as is used on the PAM 2000 (Heinz Walz GmbH, Effeltrich, Germany) and Hansatech Instruments, Ltd (Norfolk, England). It can be used with the standard 2x3 cm opaque chamber bottom, or the 6400-08 Clear Bottom Chamber. A GaAsP light sensor is included.

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.

6400-10 MiniPAM Adaptor
This is a 2x3 cm chamber top that is designed to hold a 2 mm fiber probe, such as is used by the MiniPAM fluorescence system (Heinz Walz GmbH, Effeltrich, Germany). A GaAsP light sensor is included.

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.

6400-14 Opti-Sciences Adapter
This is a 2x3 cm chamber top that is designed to accommodate the 10 mm trifurcated fiber probe used by Opti-Sciences fluorometers (Opti-Sciences, Tyngsboro, MA). A GaAsP light sensor is included.

6400-15 Arabidopsis 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).

6400-40 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 pressing **ctrl** + **home**.

2

Assembling the LI-6400

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 chemical 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.

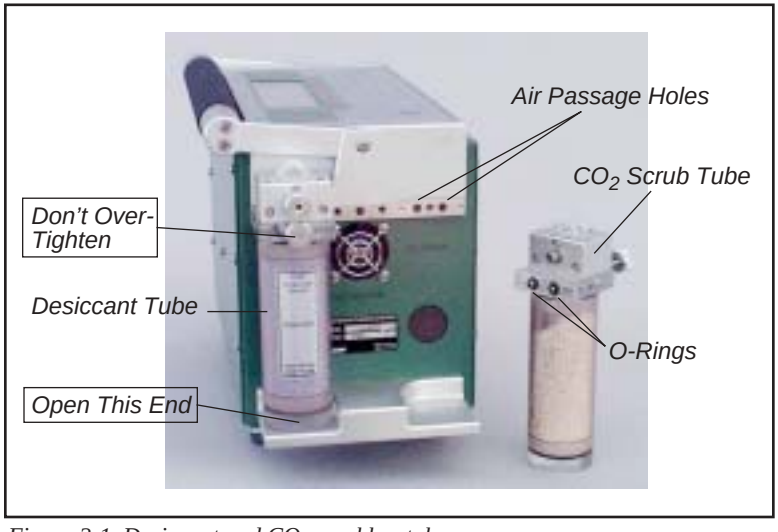


Figure 2-1. Desiccant and CO_2 scrubber tubes.

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 can not 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 4-48.

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.

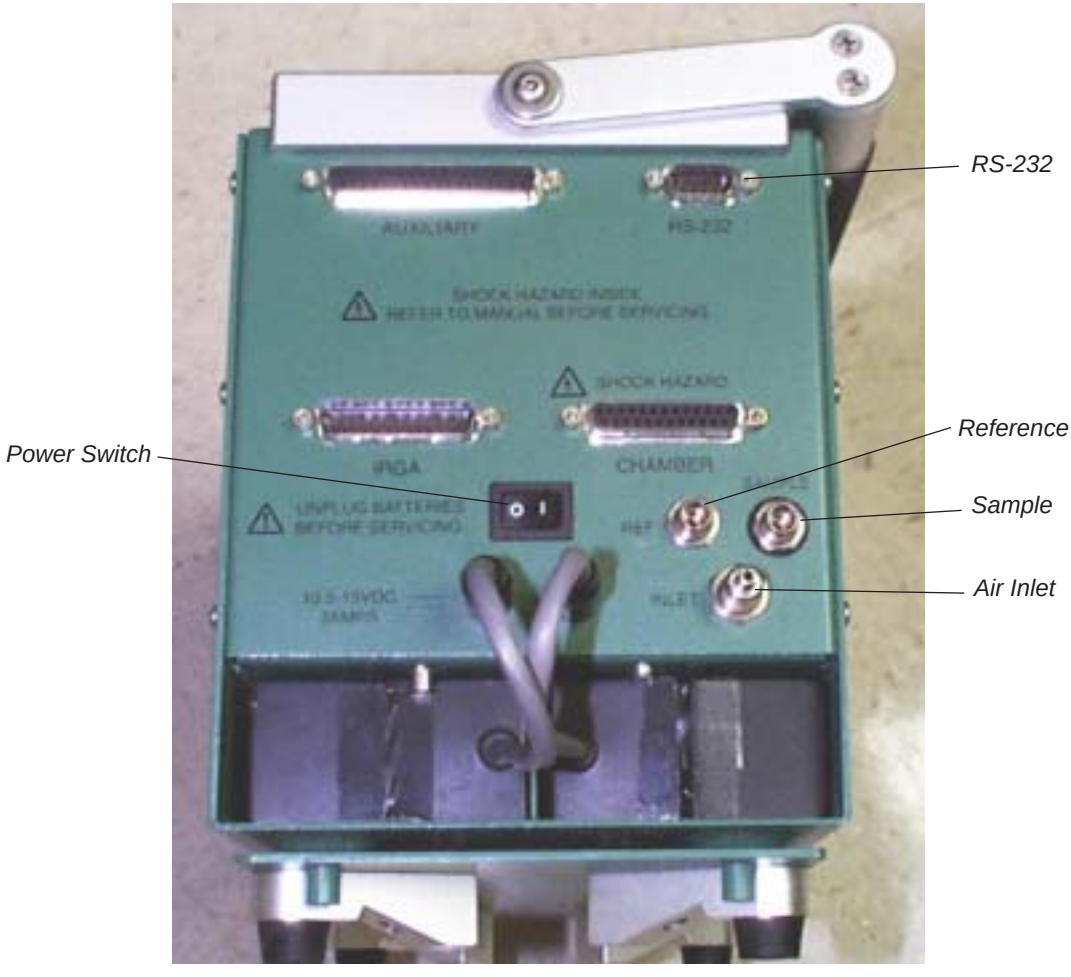


Figure 2-2. Console tubing and cable connections.

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.

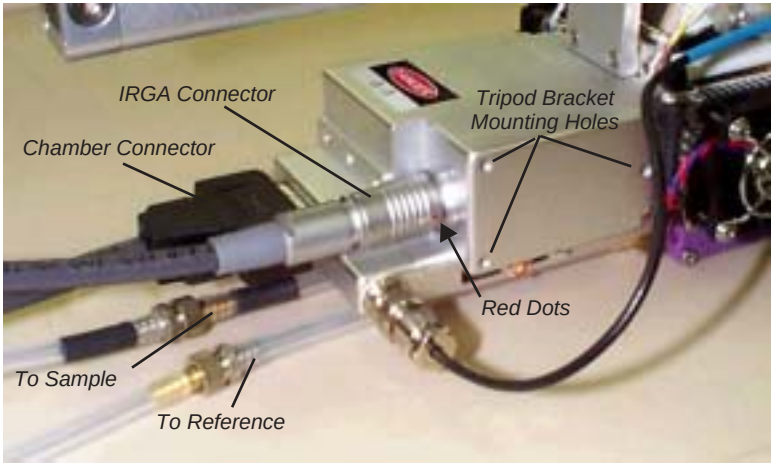


Figure 2-3. Electrical connectors, air hoses, and tripod bracket mounting holes on the sensor head / IRGA.

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.

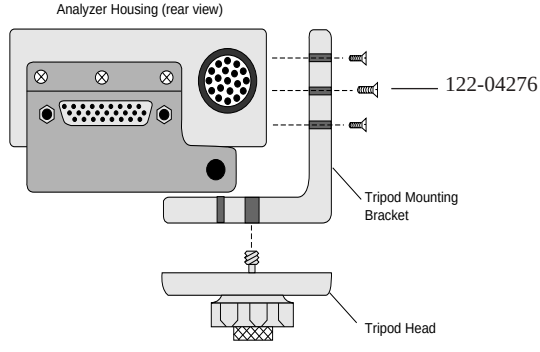


Figure 2-4. Mounting holes for the tripod bracket are found on the right side of the sensor head.

6400-01 CO_2 Injector Installation

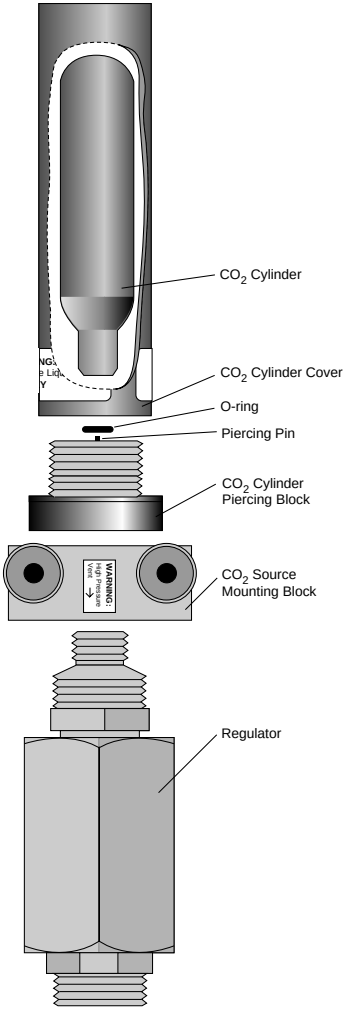


Figure 2-5. 9964-026 External CO_2 Source Assembly

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.

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.

Using CO_2 Cartridges

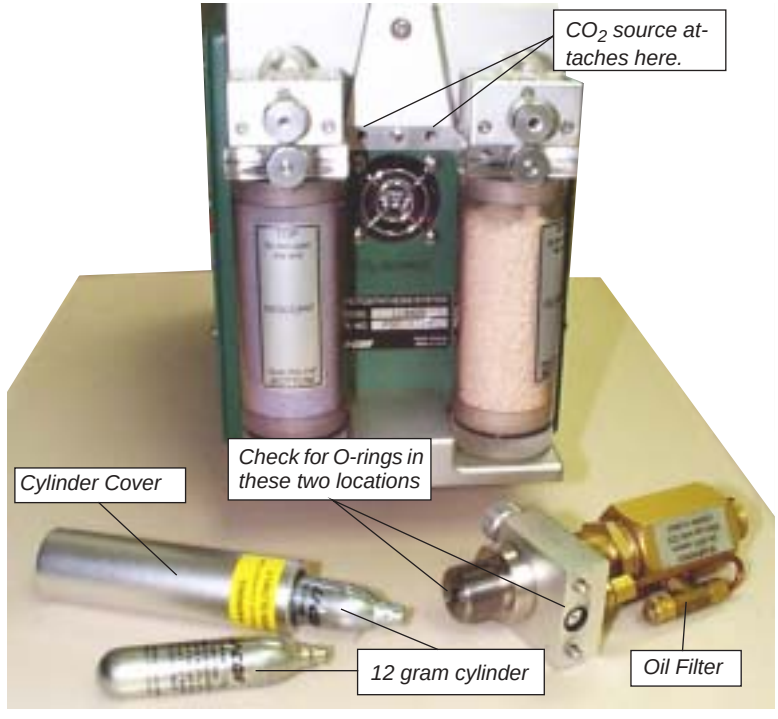


Figure 2-6. Location of external CO_2 source assembly.

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.

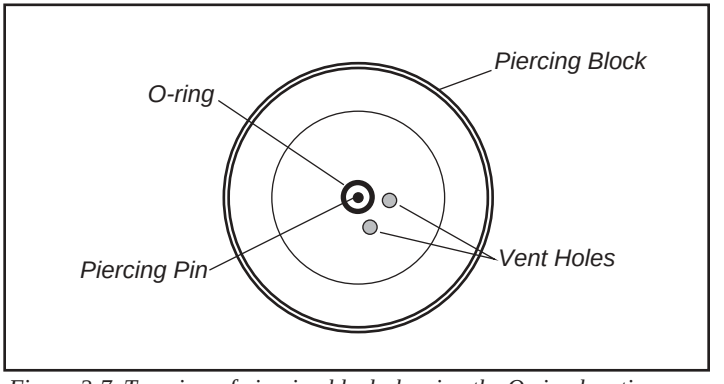


Figure 2-7. Top view of piercing block showing the O-ring location.

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 much 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.

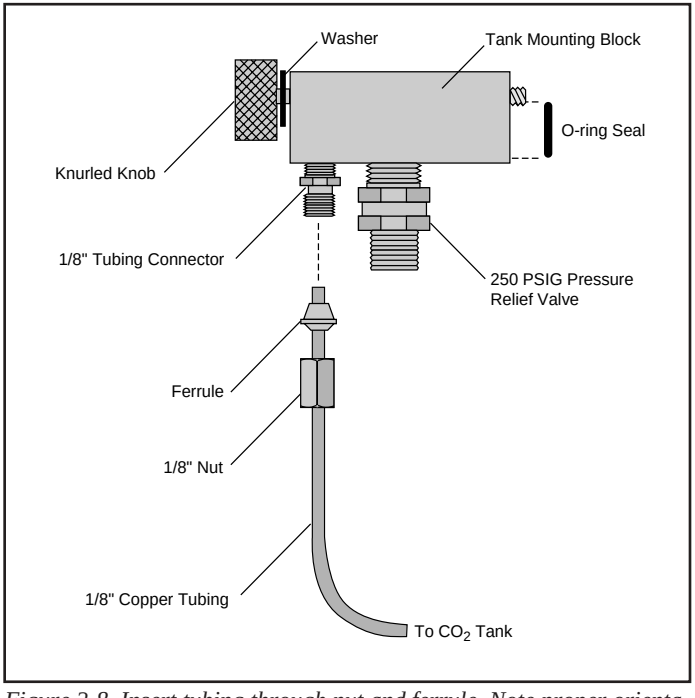


Figure 2-8. Insert tubing through nut and ferrule. Note proper orientation of ferrule.

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.

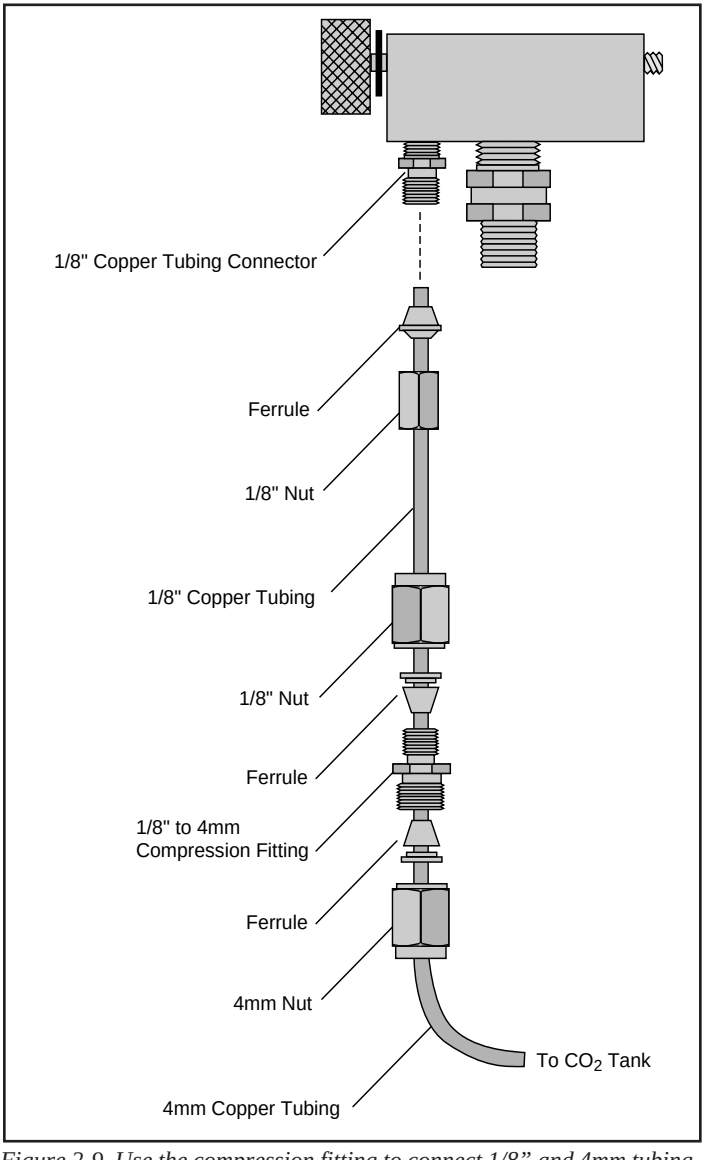


Figure 2-9. Use the compression fitting to connect 1/8" and 4mm tubing.

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 **The Installation Menu** on page 16-5.

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 (don't wiggle side to side) with a pair of long nose pliers (or your fingernails) gripping the connector (Figure 2-11).

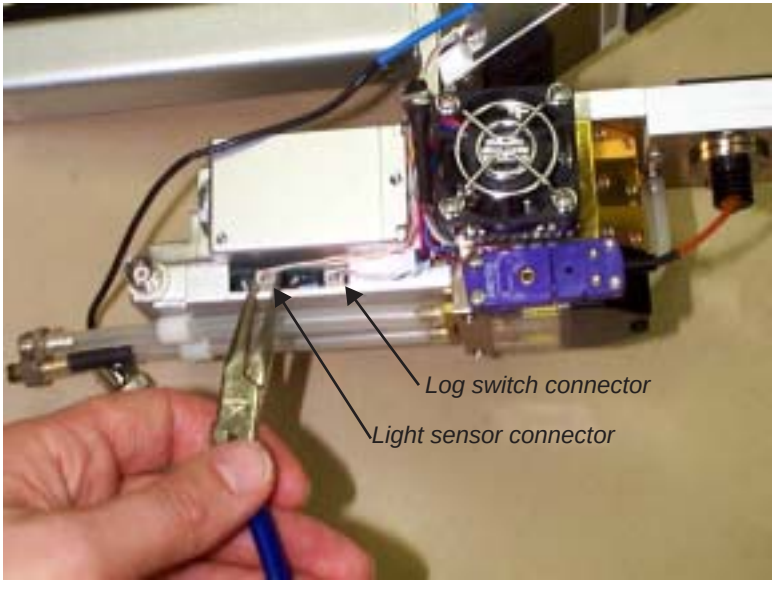


Figure 2-11. Disconnecting the in-chamber PAR sensor connector.

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).

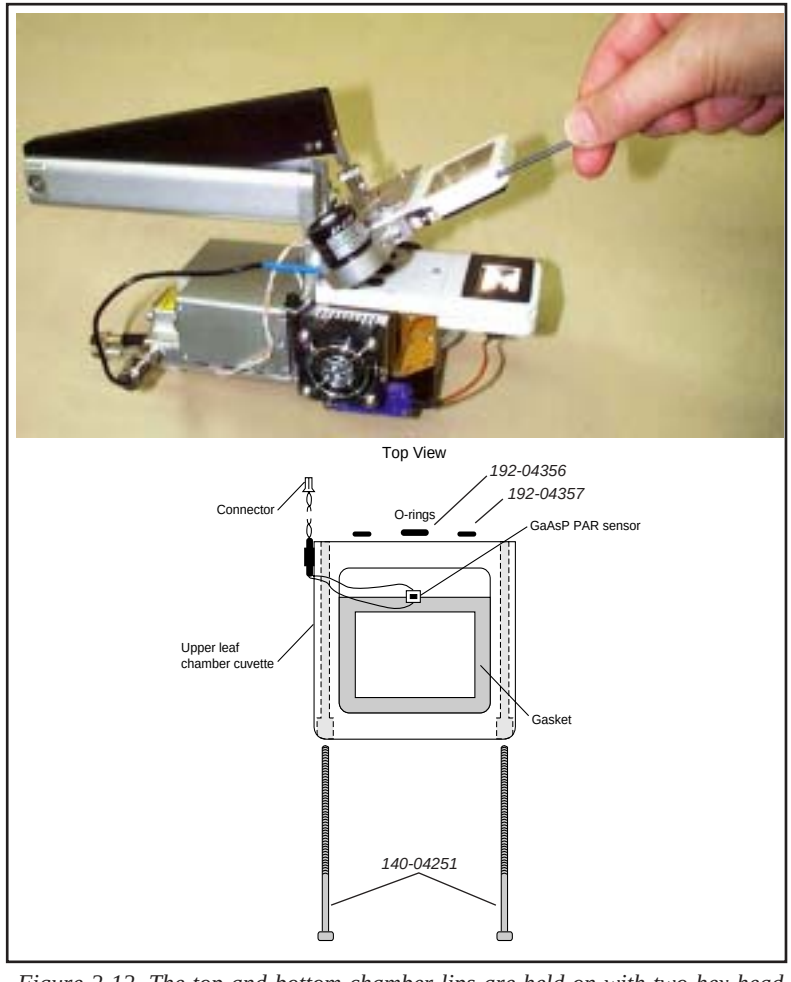


Figure 2-12. The top and bottom chamber lips are held on with two hex head bolts. Use the 3/32 inch hex key to loosen and tighten them.

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.

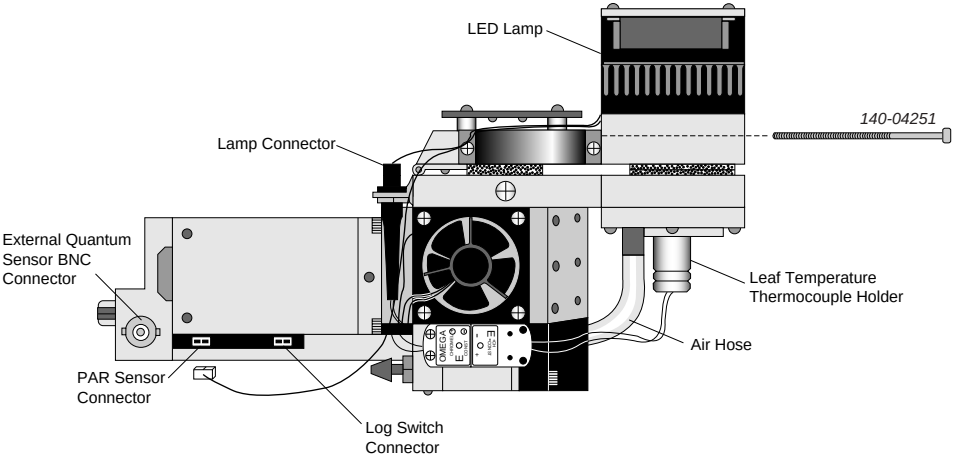


Figure 2-13. Attach the lamp and PAR sensor connectors.

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 using the Installation Menu.

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 5-17). 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.

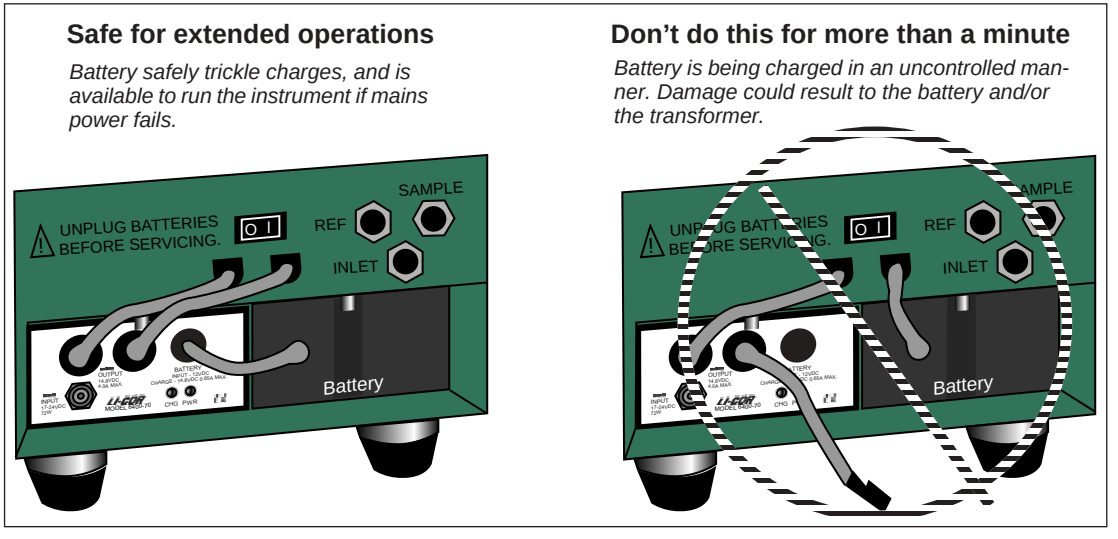


Figure 2-15. Avoid extended (more than a minute) operations with a battery and one of the AC module's plugs connected to the console. In this configuration, you will be providing uncontrolled charging of the battery, which could damage the battery and/or the transformer.

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 not 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 Welcome 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 5.0 and above

Version 5.0 and above requires the new digital board, which began shipping with serial number PSC-1214. Any LI-6400 can be upgraded to this board. The upgrade is part number 6400-915. To determine which board an LI-6400 has, see Figure 2-16.

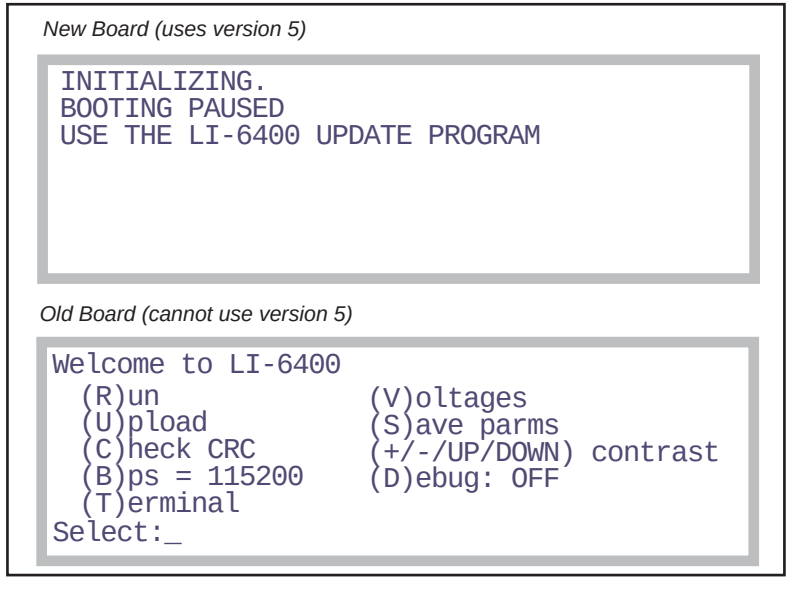


Figure 2-16. To determine which digital board an LI-6400 has, power up with the **escape** *key held down, and see which display is shown.*

[2] Or fix enhancements and add bugs…

Version 4

Version 4.x requires the 6MB flash memory board, which has been standard on all units from about serial number PSC-350 until the shift was made to the new digital board. Any old-board unit can be upgraded to this memory board. (Part number 6400-906.).

Version 3

What's left is version 3.x, which can be installed on any old-board LI-6400.

Note: The 6400-40 Leaf Chamber Fluorometer requires at least Version 4 (old board) or Version 5 (new board).

What To Do

The latest system software (3.x, 4.x, or 5.x) is available on the 6400-55 CD, and also from our web site (www.licor.com). The version 5 installer requires Windows®, while the Version 3 or 4 can be done with Macintosh® or Windows® computers.

1 **Obtain the system software**
System software is on the CD, or can be downloaded from our web site.

2 **Power the LI-6400 on while pressing escape.**
This will access the Boot Screen (Figure 2-16 on page 2-22).

3 **Connect the computer to the LI-6400**
Connection instructions are on page 11-2.

4 **Run the installation program**
Two pieces of software are installed: LPL and the /Sys directory. LPL is the operating system that supports the programs that run on the LI-6400, and the /Sys directory contains those programs. The installation should not affect any other file system information, other than updating the factory default files (listed in Table 16-3 on page 16-19) in the /User/Configs directory.

3

Guided Tours

Learning how to make it work

3 Guided Tours

The purposes of this chapter are to teach 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. 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. 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**.

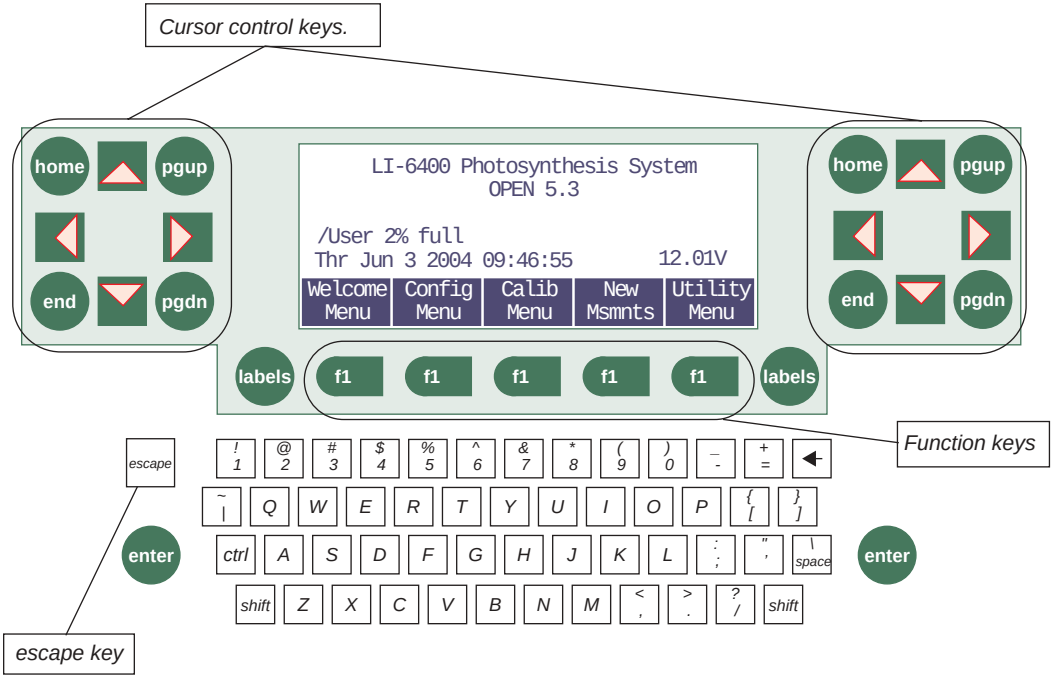


Figure 3-1. The LI-6400 keypad. The cursor control keys, labels, and enter are paired to facilitate access with either hand.

Tour #1: OPEN Overview

Running OPEN

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

1 Turn the LI-6400 ON

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

```
INITIALIZING.
```

with one, two, and finally three dots. You should then see:

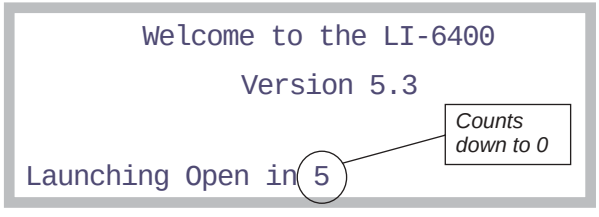


Figure 3-2. OPEN's Autostart countdown is 5 seconds long. Press **enter** *to skip it, or press* **escape** *to prevent OPEN from loading.*

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). This takes about 5 seconds.

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

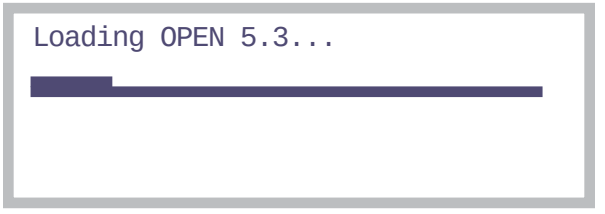


Figure 3-3. OPEN's bar chart is displayed while loading.

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 no one has been creating configuration files on this instrument, then you won't be asked to do this.)

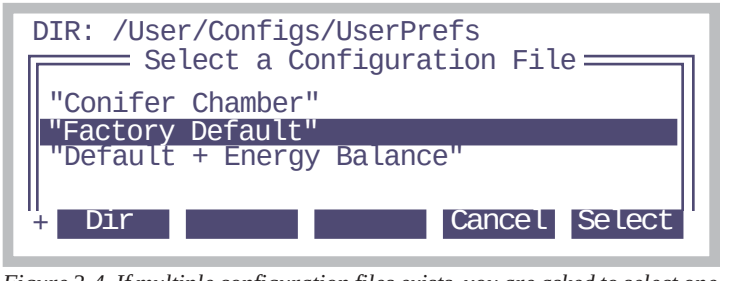


Figure 3-4. If multiple configuration files exists, you are asked to select one when OPEN first runs.

A configuration file contains settings and values used by OPEN. Except for the one named Factory Default, they are user created. Configuration files are easy to create and modify, and you can read about them in Chapter 16.

For now, however, select Factory Default. It should already be highlighted, so just press **enter**.

4 **Power up the IRGAs.**

After a few more seconds, and several more messages, you'll be asked

Press **Y** when you have both electrical cables between the console and the sensor 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; these 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 measurements, 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 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 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.

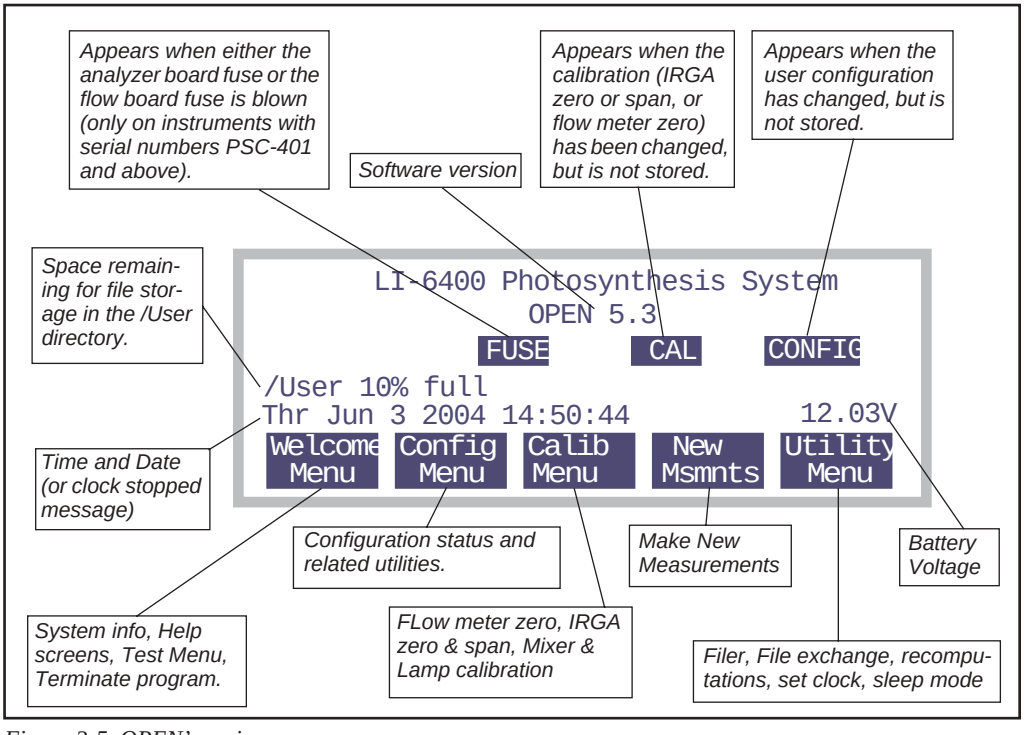


Figure 3-5. OPEN's main screen.

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 upgraded with a new back plane 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 instrument is capable of producing this alert message, check for "`Fuse Aware`" in the "`About this Unit`" screen, found in the Welcome Menu, as shown in Figure 3-6.)

CAL

This message appears if you have changed the IRGA zero and/or span setting, or else the flow meter zero, and have not stored the new setting(s). To store them, press **f3** (**Calib Menu**), select "`View,Store Zeros & Spans`", then press **f1** (**Store**). For details, see **View, Store Zeros & Spans** on page 18-19.

Config

The instrument's configuration has been changed, but is not stored. To save this configuration, press **f2** (**Config Menu**), select "`Config Status`", then press **f2** (**Save**) or **f3** (**saveAs**). For details, see **Config Status** on page 16-15.

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 20-6.

The Welcome Menu

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

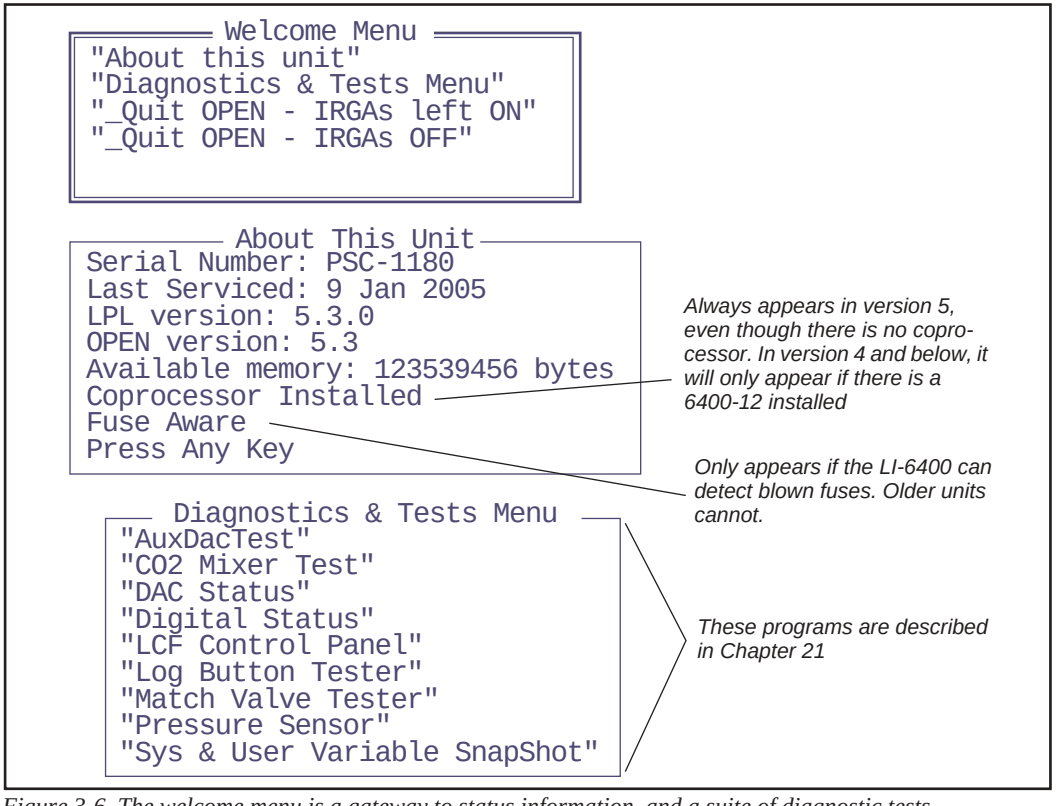


Figure 3-6. The welcome menu is a gateway to status information, and a suite of diagnostic tests.

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

4 **Select "Diagnostics & Tests Menu"**
A number of system test programs are grouped in this menu. Note that these programs are documented in Chapter 21. Right now, we'll try out a simple one.

5 **Select "Log Button Tester"**
Highlight this item in the test menu, and press **enter**.

6 **Try the button**
The log button is located on the handle of the chamber/IRGA. Press the button, and watch the display indicate switch status with

```
Log Button is down
```

or

```
Log Button is up
```

until you press **escape** to quit.

7 **Return to the Welcome Menu**
Press **escape**.

8 **A word about exiting OPEN**
The bottom two entries in the Welcome Menu will terminate OPEN. If you "Quit Open - IRGAs Off", the program will turn off the flow control board and the analyzer board. If you "Quit Open - IRGAs left ON", not surprisingly, the boards are left on. There is normally no need to do either of these, however; when you have finished using the LI-6400, you can just press the power switch while in OPEN's main screen.

9 **Return to the OPEN's Main Screen**
Press **escape** again.

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 3-69.

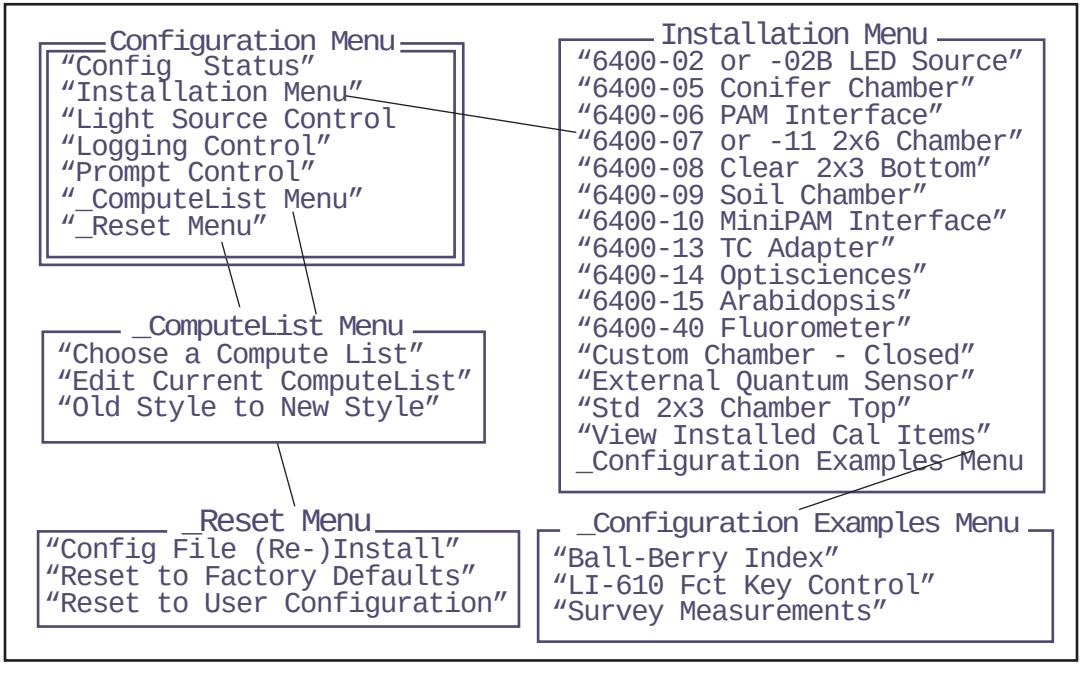


Figure 3-7. The Config Menu

Press **f2** and explore in the Config Menu if you like. Otherwise, we'll move on to the Calibration 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.

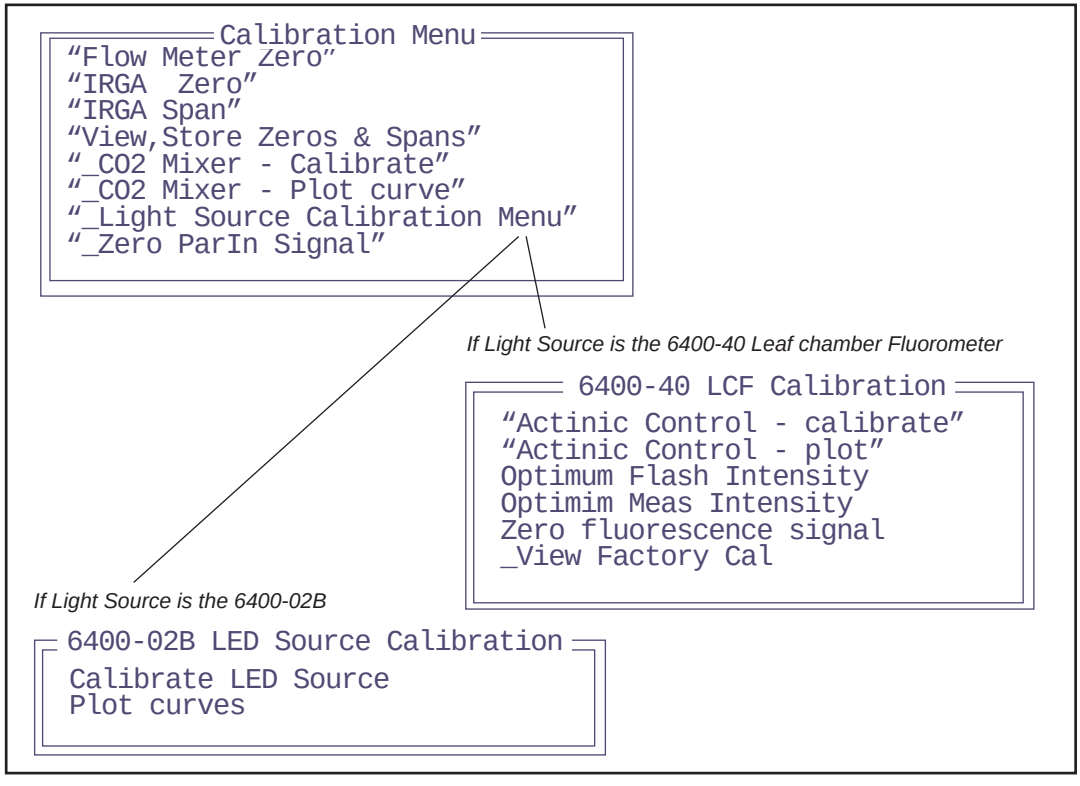


Figure 3-8. The Calib Menu. The light source calibration sub menu depends on what the light source ("Light Source Control" in the Config Menu) is currently set to.

Chapter 18 discusses calibrations and the programs in this menu, but for now we'll just do a couple of simple tasks: zero the flow meter, and view the CO_2 mixer calibration curve.

Zero the flow meter

For a more complete explanation of what zeroing the flow meter is all about, see page 18-18.

1 **Select "Flow meter zero"**
Highlight "Flow Meter Zero" and press **enter**.

2 **Wait and watch**
The pump and fan are automatically turned off, thereby eliminating all sources of air movement through the flow meter. The display will show a countdown for 10 seconds, along with the flow meter's signal (mV). At the end of 10 seconds, an internal adjustment is made so that the flow meter's signal is zero (or very close to it) (Figure 18-7 on page 18-18).

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

■ **Plot the mixer calibration**
Whether or not your LI-6400 is equipped with a CO_2 mixer, it will have a calibration curve for one. This curve relates the control signal used to provide mixer set points to the resulting CO_2 concentration. See **6400-01 CO2 Mixer** on page 18-21 for more details.

1 **Select "_CO2 Mixer - Plot curve"**
Highlight it, and press **enter**. You should see a graph something like Figure 3-9.

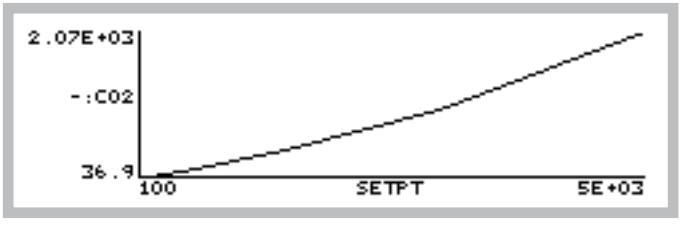


Figure 3-9. A typical CO_2 mixer calibration curve, relating set point signal (mV) to CO_2 concentration (µmol mol^{-1}).

2 **View the data points**
To stop viewing a graph, you would normally press **escape**. If you would like to see the values of the plotted data, however, press **V** (for View) instead. (This shortcut works for any plot done by GraphIt, a built in plotting package. GraphIt is introduced in Tour #4.) The data values are shown in a list that you can scroll.

3 **Return to Calib Menu**
When you are done viewing the data values, press **escape**.

The Utility Menu

The Utility Menu (Figure 3-10) has a collection of useful things that you'll occasionally need to do while using OPEN.

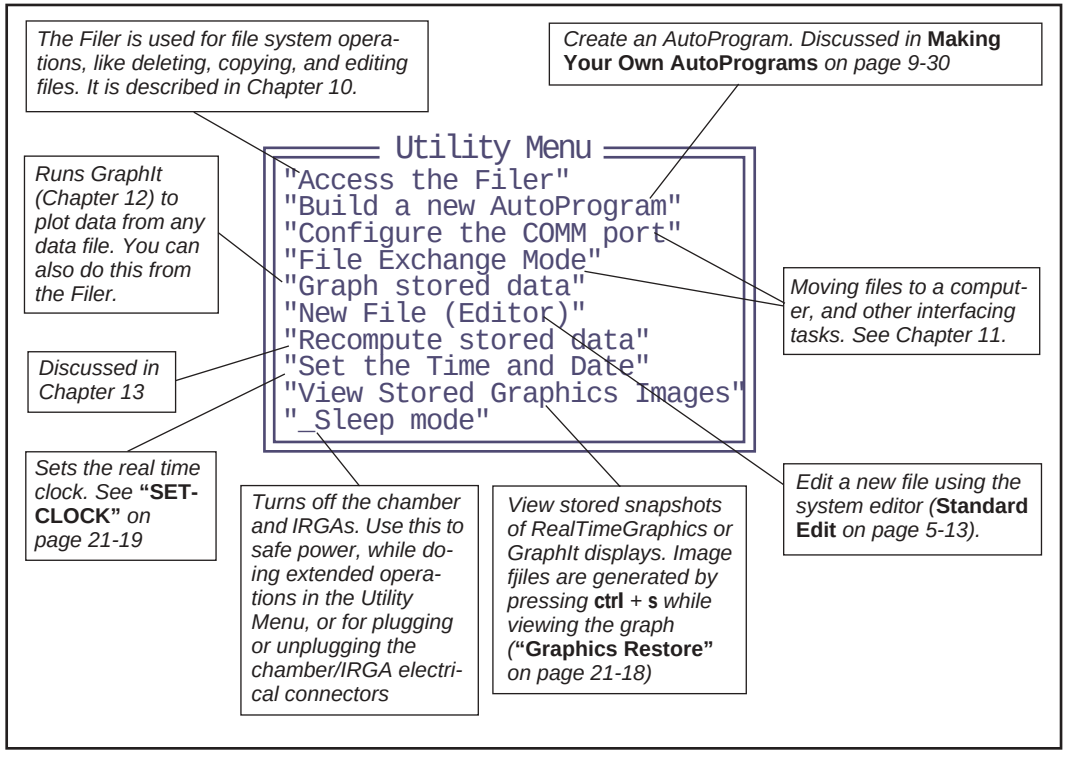


Figure 3-10. The Utility Menu

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 it's 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, download data, and other ancillary operations.

The New Measurements screen (Figure 3-11) uses three rows of variables; each row has highlighted labels above the values. A row of function key labels appears at the bottom.

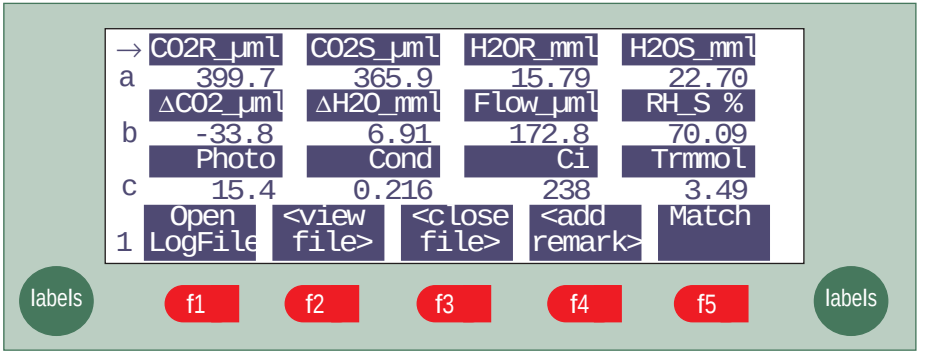


Figure 3-11. New Measurements screen text display.

New Measurements mode has two other display modes for providing you information. In addition to this text mode, there is Graphics mode and Diagnostics mode. Figure 3-12 summarizes how to switch between these modes

.

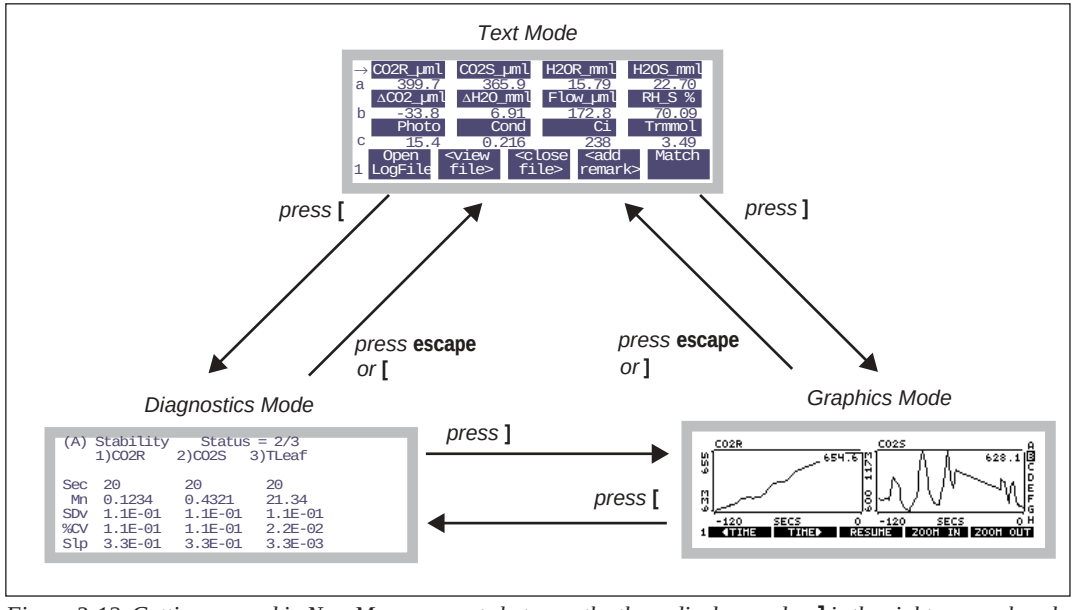


Figure 3-12. Getting around in New Measurements between the three display modes. **]** *is the right square bracket key, and* **[** *is the left square bracket key.*

We'll spend the most time discussing text mode, but before we do, however, 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 describes more details then. Details of all three display modes are in Chapter 6.

Function Keys

Text mode has 7 sets of function key definitions (or 10 with the Leaf Chamber Fluorometer); 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. The current function key level number is displayed in the bottom left hand corner of the display.

■ **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 3-13. Note that the level 7 key labels are all blank in the default configuration, and are available for your own use (described in **Example #1: Using New Measurement's Fct Keys** on page 26-10).

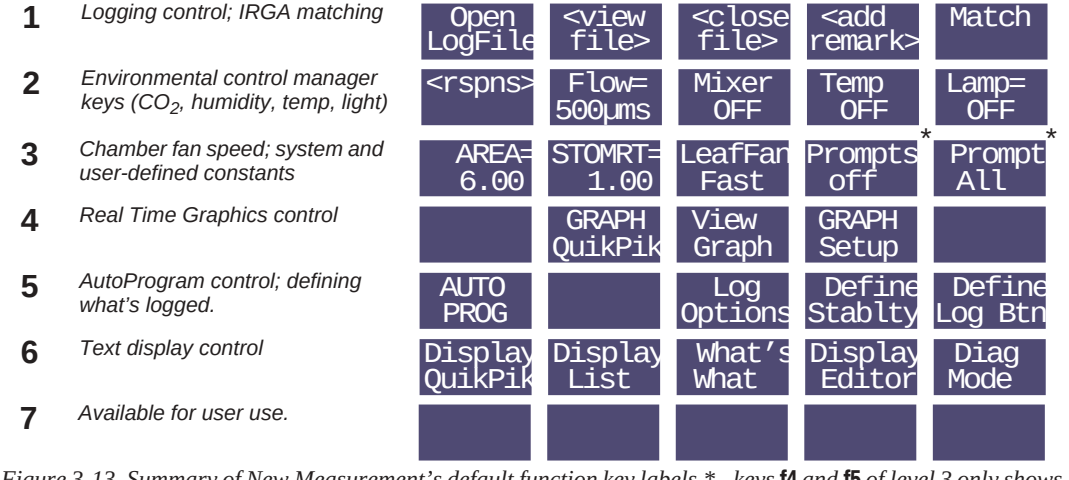


Figure 3-13. Summary of New Measurement's default function key labels. - keys* **f4** *and* **f5** *of level 3 only shows these labels when user constants are defined (described in* ***Prompt Control*** *on page 9-15).*

Example: Changing the chamber fan speed

1 Access the chamber fan control function key
Press **3**. The function key labels will change to

2 Access the control display
Press **f3**. The fan speed control box will pop up on the display.

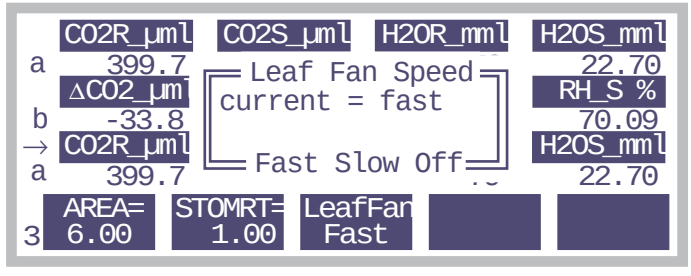


Figure 3-14. Controlling the fan speed. Press **F**, **S**, *or* **O** *for fast, slow, or off.*

3 Turn the fan off
Press **O** (the letter), and the fan will stop. If you are listening, you should hear a drop in the noise level.

4 Turn the fan back on (fast)
Press **f3**, then **f**, and the fan will resume running.

Text Display

Twelve variables are displayed in Figure 3-11 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 3-15.

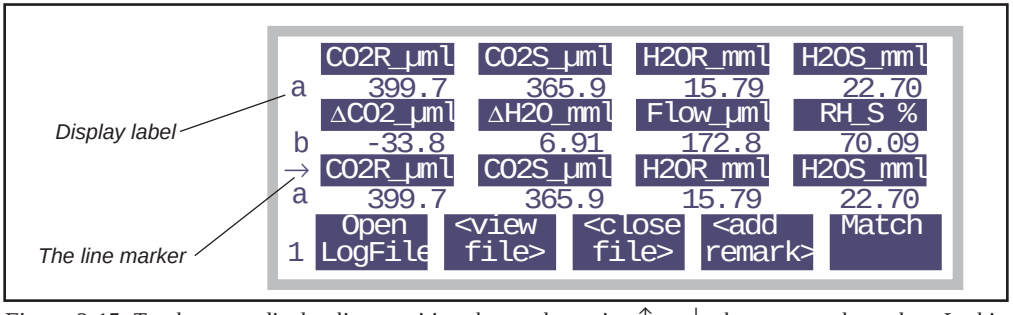


Figure 3-15. To change a display line, position the marker using ↑ or ↓, then press a letter key. In this figure, we've put the marker on the bottom line and pressed **A**.

3 **Alternatively, press ← or →**

Pressing the horizontal arrow keys will scroll the change line through all the possible displays. But it's a lot faster to use the letter shortcuts.

4 **Go exploring!**

Table 3-1 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 (what variables are shown, and where) to suit nearly any taste. It's all explained in 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**, **pagup**, **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**, **pagup**, **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.

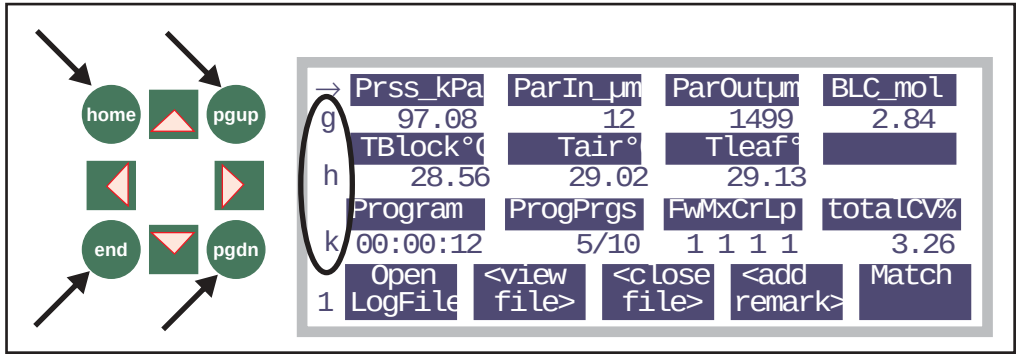


Figure 3-16. Define a display group by holding the **ctrl** *key down and pressing any of the 4 indicated group keys:* **home**, **pgup**, **pgdn**, *and* **end** *(on either side of the display). To change all three lines to a previously defined group, simply press that group key.*

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**.

Display group information is stored with display definitions. See **Display Editor** on page 6-6.

Table 3-1. The variables and how they are grouped for viewing in the default display configuration. For more information about these variables, refer to Table 14-8 on page 14-19.

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	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_%	Dew point temp in the reference cell (C)
	Td_S_%	Dew point temp in the sample cell (C)

Table 3-1. (Continued)The variables and how they are grouped for viewing in the default display configuration. For more information about these variables, refer to Table 14-8 on page 14-19.

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})
	BLC_mol	Total boundary layer conductance for the leaf (includes stomatal ratio) (mol 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)
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

Graphics Displays

New Measurements mode also provides a method to monitor these variables graphically. To access the graphics displays, press **4**, then **f3 (View Graphs)**. (There is also a short cut: **]** the right square backet key.) The display should show something like Figure 3-17.

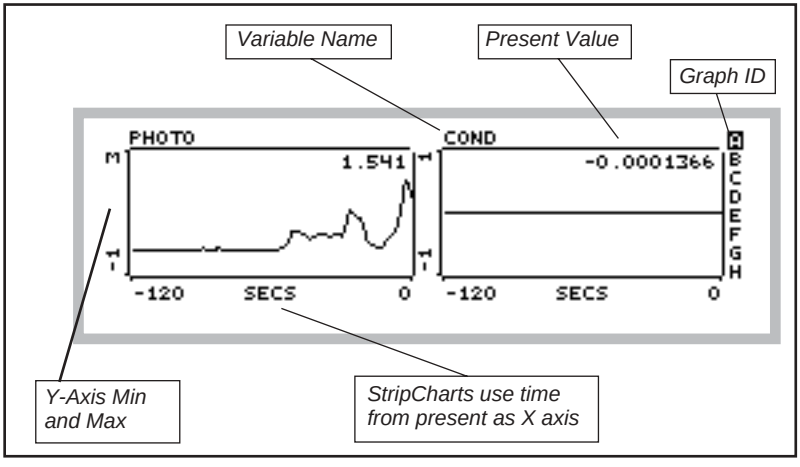


Figure 3-17. The default Real Time Graphics (RTG) display A in New Measurements mode shows photosynthesis and conductance.

There are 8 graphics displays available in New Measurements mode, and they are designated by the letters *A* through *H*. Figure 3-17 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 3-18) -one variable plotted against another using discrete points, usually representing logged observations.

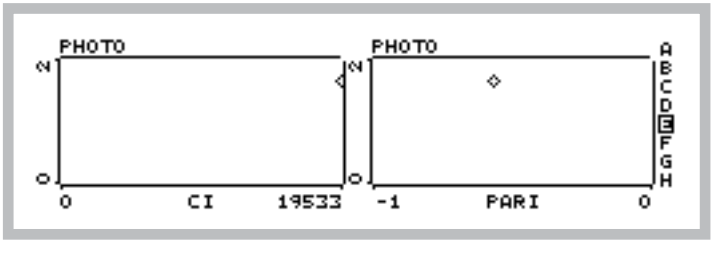


Figure 3-18. An XYPlot. Present value is shown by the diamond. Logged data (there are none here) are shown with a +.

Here are two navigational rules:

- **To change Graphics 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.

- **To exit Graphics Mode**
  This will take you back to the text displays we were looking at earlier.

If no plots are defined for a particular graphics display, it will appear as in Figure 3-19.

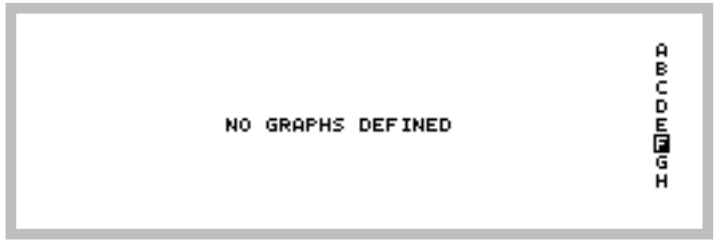


Figure 3-19. An undefined graphics display.

Graphics Function Keys

There are two 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.

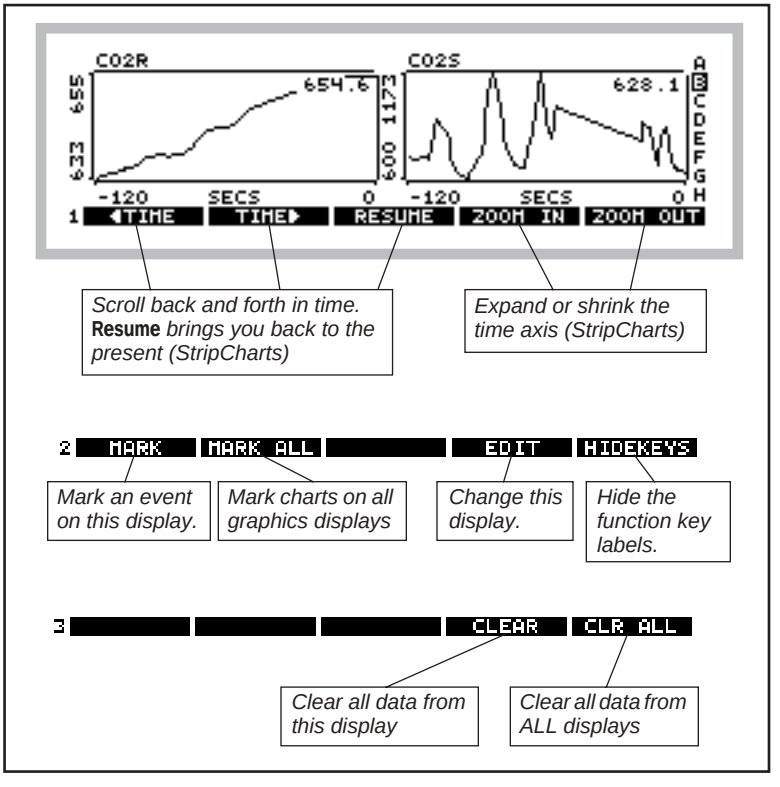


Figure 3-20. Real Time Graphics has three levels of function keys. Use **labels**, *or* **1**, **2**, *and* **3** *to change them.*

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, doesn't turn them on for any other.

2 **There are 3 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.

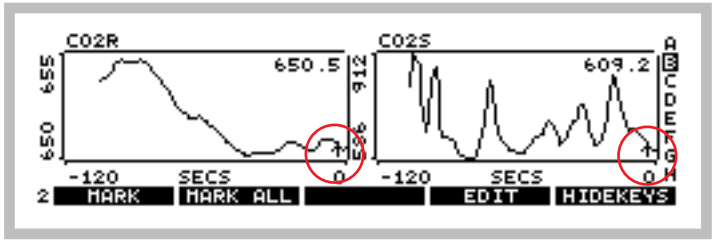


Figure 3-21. **MARK** *puts a small marker on StripCharts. This also happens when data is logged.*

As you might expect, **MARK ALL** marks all the 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 3-21 are showing 2 minutes 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) se**veral times (Figure 3-22). 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 contines to be collected and saved. The axis time labels still reflect time since the present.

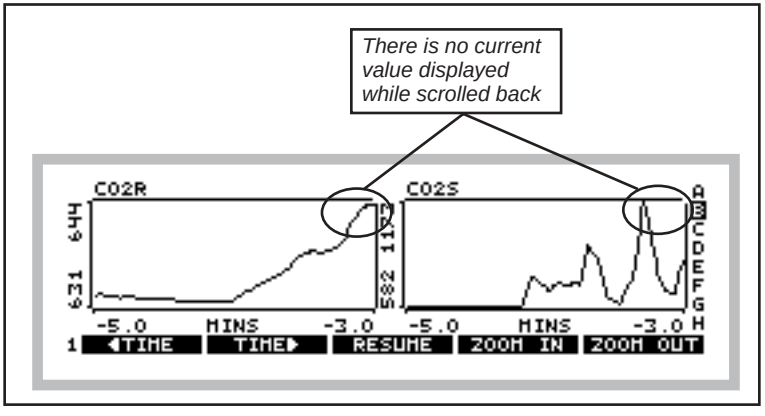


Figure 3-22. Scroll back in time. Here we are looking at data between 3 and 5 minutes ago.

To resume normal plotting, you can press **f3** (**RESUME**), or scroll back to the present with **f2** (**TIME►**), or simply don't press any keys 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 3-23).

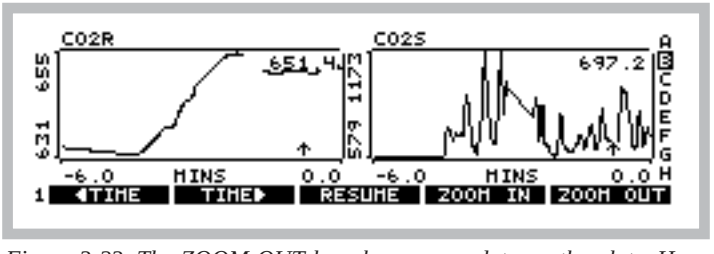


Figure 3-23. The ZOOM OUT key shows more data on the plots. Here we've gone to 6 minutes.

Now, press **ZOOM IN** several times to return to 2 minute display width.

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 3-24). Now, pressing a time control key will operate on only that plot.

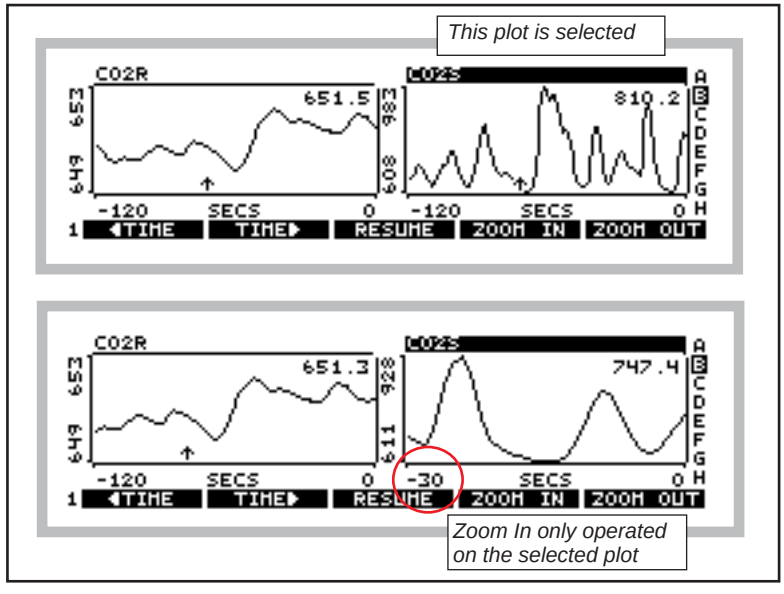


Figure 3-24. Select plots using ← or →. When no plot is selected, the time control keys work on all plots in the display.

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 3-25).

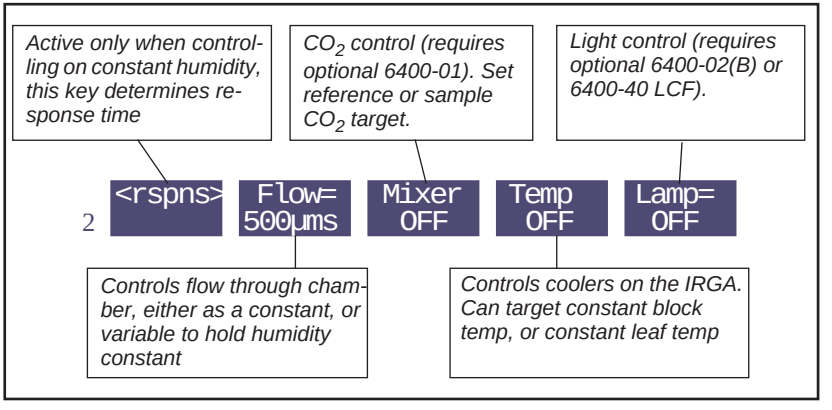


Figure 3-25. Summary of the environmental control function keys.

The four control areas are flow/humidity, CO_2, temperature, and light[2]. 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

[2] There's actually a fifth control: wind speed, via the fan. The fan control is on level 3, but we usually keep the fan speed on fast.

(Figure 3-25). Use the adjusting nut to make the chamber seal snugly, but not too tight.

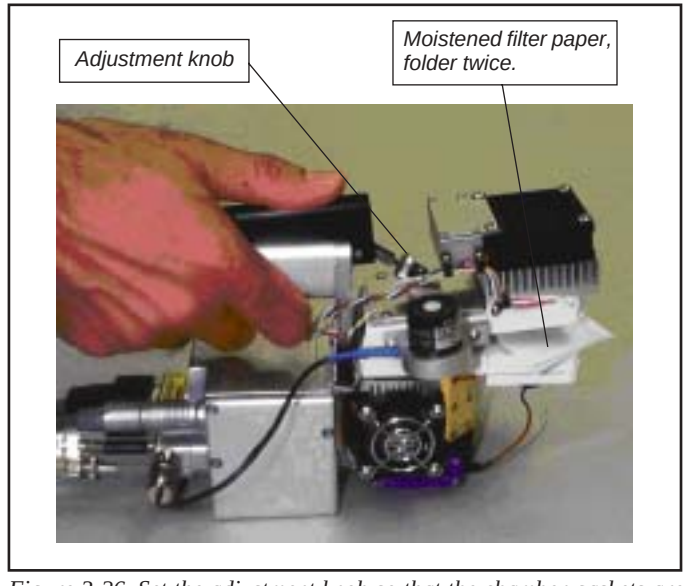


Figure 3-26. Set the adjustment knob so that the chamber gaskets are slightly compressed when closed with no leaf. Then, open the chamber, tighten one more half turn, and close onto the leaf or filter paper.

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 it's travel, quit. It's 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**, **F** (for fixed flow), **700**, then **enter** (Figure 3-27).

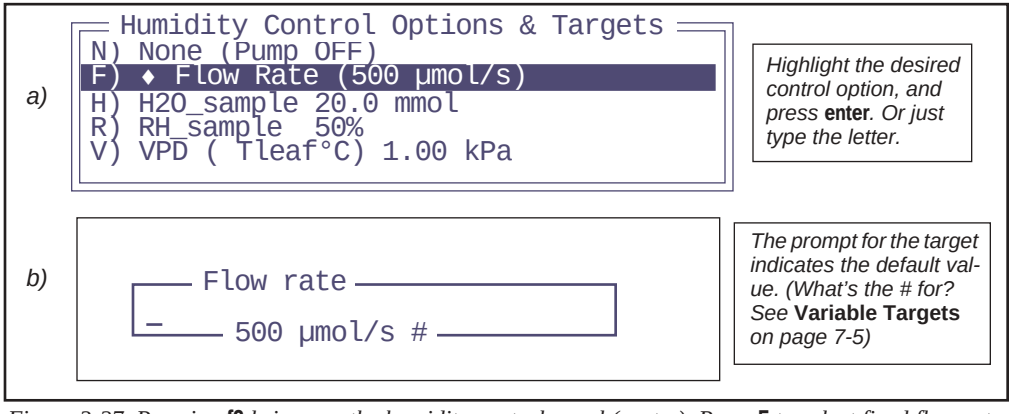


Figure 3-27. Pressing **f2** *brings up the humidity control panel (part a). Press* **F** *to select fixed flow rate, then enter the target value (part b).*

4 Note the reference and sample water vapor channels

If it's 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 µmol s^{-1}.
Press **F2** then type **100**. (Notice the short cut: when you aren't changing control *modes*, you can just start typing the target value after you press the function key. In our case, we didn't need to press **F** again to specify fixed flow mode.)

8 Note the sample and reference humidities
The reference value hasn't 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[3], and b) the reference 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.

- 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 3-52 on page 3-57.)

[3] 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).

Fixed Humidity Operation

One of the powerful features of the LI-6400 is it's 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). 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, let's now do some constant humidity operations. We'll start in fixed flow mode to see what range of humidities are achievable.

1 Set flow to 400

Press **2**, **f2**, then **400 enter**.

2 Put desiccant mid-range

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, because this is the region where the knob is the loosest.

3 Note the sample water mole fraction

The *H2OS_mml* (display line *a*) value is probably about 20 mmol mol^{-1}.

4 Switch to humidity control

Press **f2**, **H**, then **20 enter** (or whatever value you had in the last step).

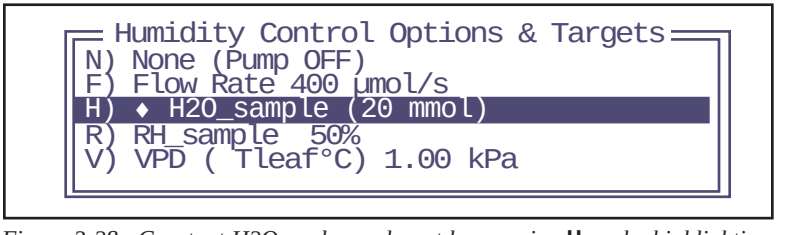


Figure 3-28. Constant H2O mode can be set by pressing **H**, *or by highlighting the H line and pressing* **enter**.

5 Note the function key labels

In fixed humidity operation, **f1** level 2 becomes active (Figure 3-29). We'll demonstrate this key a bit later. Also, the **f2** label should reflect the new type of control, and the target.

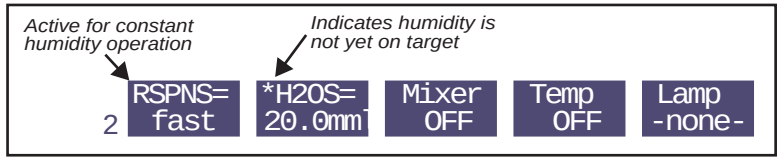


Figure 3-29. When controlling a constant humidity, **f1** *becomes active, and controls the response time of the feedback circuit.* **f2** *indicates constant mole fraction control, and the target value.*

6 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 3-29) will disappear, indicating that the water mole fraction is on target and stable.

7 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. 22 mmol mol^{-1}). The flow rate should fall, and eventually settle on a new value that maintains this higher humidity.

8 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. 18 mmol mol^{-1}). The flow rate will eventually settle on a higher value that maintains this lower humidity.

9 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. 15 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.

10 **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 4 (e.g. 25, since we used 20). 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.

11 **Return to the original target**

We'll 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 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 handful of 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 20-7. When such a situation occurs, a warning message is displayed on the middle label line on the display, as illustrated in Figure 3-30.

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.

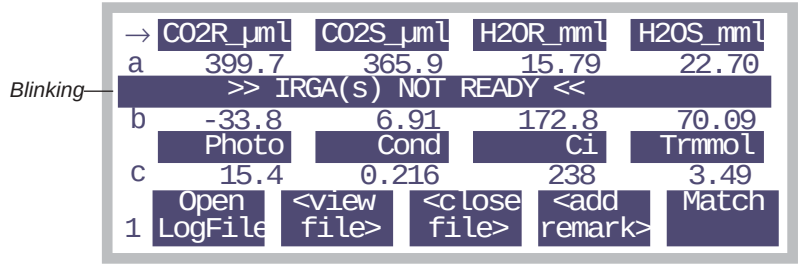


Figure 3-30. New Measurement Mode's warning messages appear on the 3rd line of the display.

Dynamic Response of Humidity Control

We'll 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'll be using is fully described in **Real Time Graphics** on page 6-11. 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:

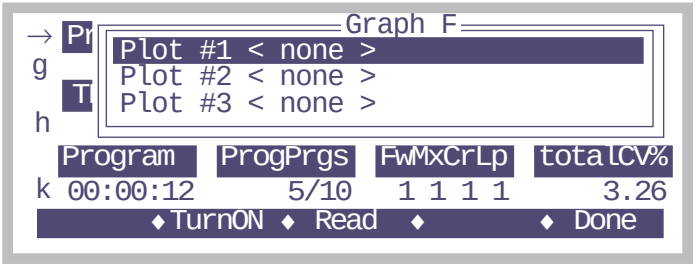


Figure 3-31. Editing Graphics Display F.

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 3-32). The top line, "Y Axis =" shows the variable to be plotted, and presently there is none defined.

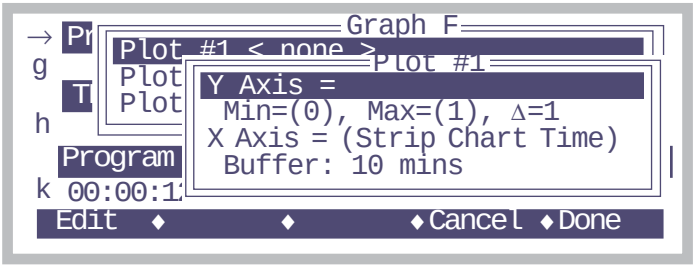


Figure 3-32. StripChart plot editor

5 Select RH_S

With the top line (`Y Axis =`) line highlighted, press **f1** (**Edit**). A list of variables will appear (Figure 3-33).

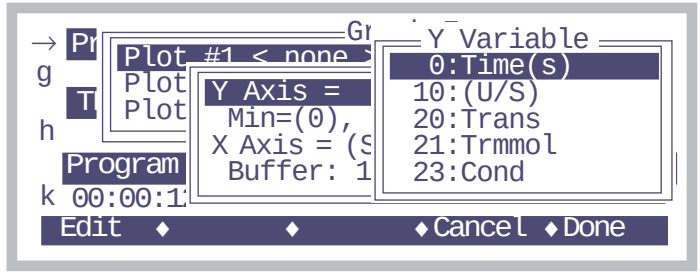


Figure 3-33. Selecting a variable to plot.

Page down (press **pgdn**) several times until you get to the line that says

```
-15:RH_S
```

and select it by pressing **f5** (**Select**) or **enter**.

Then press **f5** (**Done**) to end the plot definition. The display should appear as in Figure 3-34.

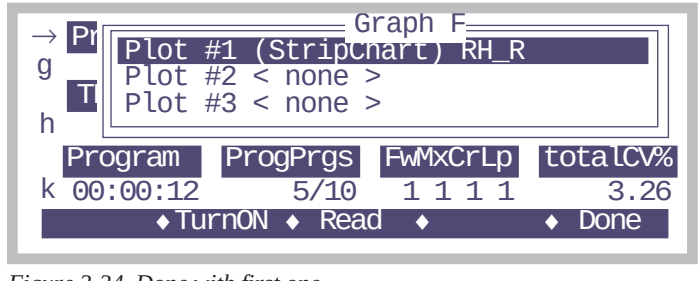


Figure 3-34. Done with first one

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**).

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).

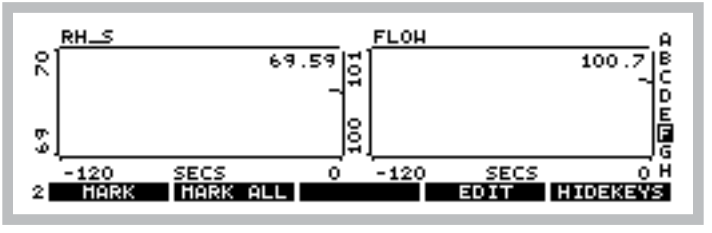


Figure 3-35. The finished product.

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

■ **Part B: Do the Test**

We'll 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 **2 f2 F 400 enter**. Note the RH value (*RH_S_%*, line *b*). We'll use this value (to the nearest 1 or 2) for the target in the next step.

2 **Constant RH**

Switch to fixed RH mode, enter the target value(**2**, **f2**, **R**, <value> **enter**).

3 **View the Graph**

Press **]**.

4 **Turn the desiccant to full bypass**

Moister 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.

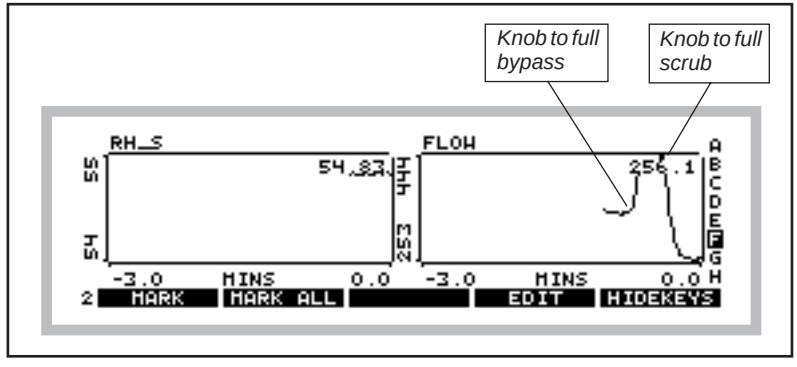


Figure 3-36. Flow increased when desiccant was put on full bypass, then dropped when desiccant was put on full scrub.

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 **f1** to drop the response from *fast* to *medium*. Resume watching the graph by **]**.

8 Full bypass, wait 10 or 15 seconds, then full scrub

The graph will look like Figure 3-37.

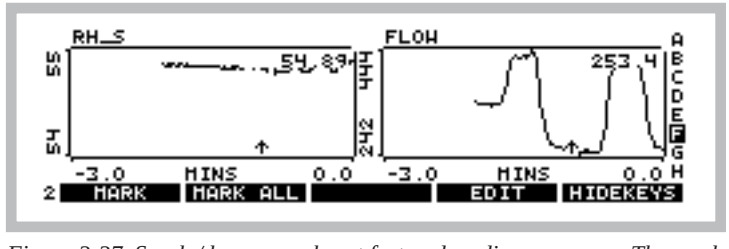


Figure 3-37. Scrub / bypass cycles at fast and medium response. The mark shows when the response was changed from fast to medium.

9 **Once more at slow response.**
Press **MARK**, (mark the graph) then **] f1** (change to slow) then **]** (back to graphics). After you've scrubbed and bypassed, you should see something like Figure 3-38.

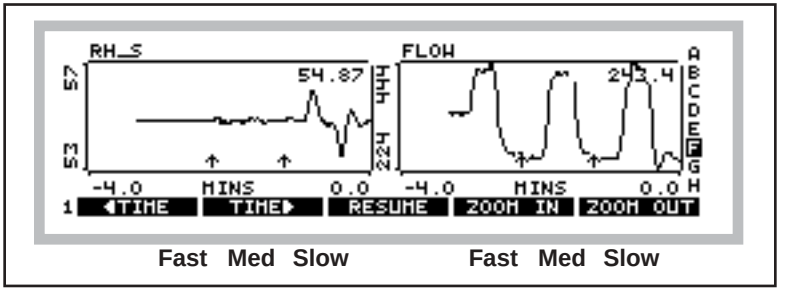


Figure 3-38. Results from the dynamic response test. Note that on fast response, the humidity is the most stable, and the flow rate the most unstable. On slow response, the flow rate is most stable, but the humidity is least stable, reacting slowly to the changing incoming humidity.

We'll 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 **The RSPNS Key** on page 7-12.)

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

CO_2 Control - Without a 6400-01

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**, **F**, **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 it's not zero, it's because the sample and reference IRGAs aren't reading the same thing. Matching, discussed on page 4-33, will take care of that, but we'll 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'll 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'll need a buffer volume to make stable measurements. See **Air Supply Considerations** on page 4-48
- The desiccant buffers CO_2.

CO_2 Control - With a 6400-01

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**, **F**, **500 enter**.

2 Turn on mixer and set a 400 µmol mol^{-1} reference target.
Press **f3 R 400 enter**. Set the soda lime to full scrub.

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

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 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 18-21) can improve this performance.

5 Once *CO2R_µml* is stable, change to 20 µmol mol^{-1}
Press **f3 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 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 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}.
- Mixer adjusts quicker lowering the concentration, than raising it.

Experiment #6 Dynamic Response of the CO_2 Mixer

We'll need StripCharts for CO2R and CO2S (where we're headed is Figure 3-39 on page 3-45). 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 F 500 enter**.

3 Set the mixer for a 400 µmol mol^{-1} reference target

Press **f3 R 400 enter**. Watch the graph (**]**).

4 After 1 minute, change reference target to 900 µmol mol^{-1}.

When the *CO2S* curve flattens out, press **] 2 f3 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 **] 2 f3 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.

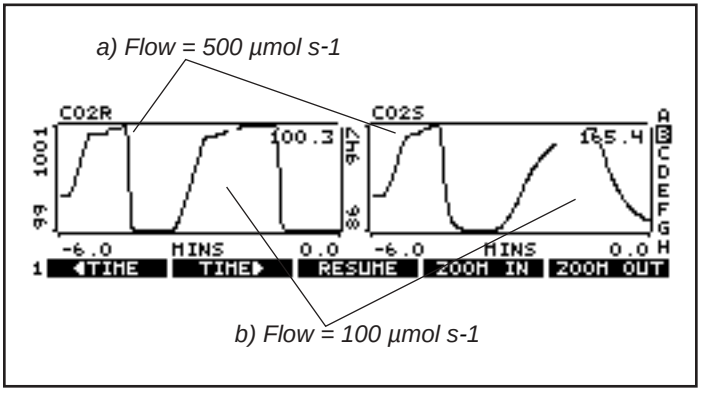


Figure 3-39. Controlling on reference CO_2 concentration, with a closed, empty chamber. Note the slower response of the sample cell (due to its larger volume) is aggravated at lower flow rates. Note also the faster response when lowering CO_2 than when raising it.

Points to Remember

• The reference cell responds quicker than the sample cell, especially at low flow rates.

• 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 7-16 has details, but the next two experiments illustrate the differences.

We'll 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.

Experiment #7 Controlling on Block Temperature

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

1 Determine the ambient temperature.
Display line *h* of the default display map has block, air, and leaf temperatures. Note the block temperature.

2 Set block target to 3° cooler than current value
Press **2 f4 B** <*value* - 3> **enter** (where *value* is the block temperature from Step 1). 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 degrees warmer than ambient
Press **f4** <*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.

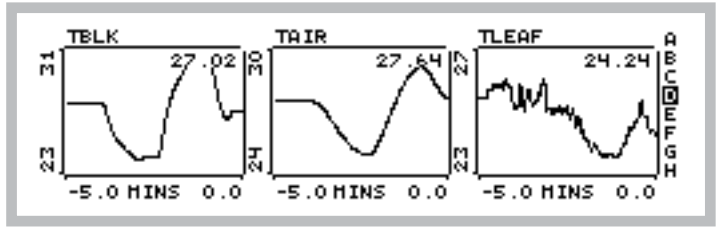


Figure 3-40. Results of Experiment 7.

Points to Remember

- Controlling on block temperature is slow but steady.
- Limit of control is generally within 7 degrees 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'll 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 degree cooler than it's present value

Press **2 f4 L** <*value* - 1> **enter**.

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 **2 f4** <*value* + 1> **enter**. Then **]** to view the strip charts (Figure 3-41b).

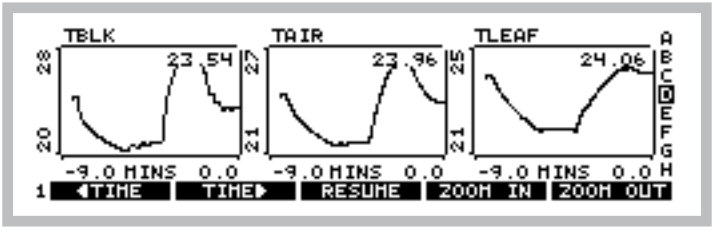


Figure 3-41. Results of Experiment 8.

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 a block temperature control, since the mechanism is to control the temperature or the air that is blowing by the leaf.

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,[4] then do the following:

■ **To configure OPEN for the LED source:**

1 **Go to the Light Source Control**
Press **escape** to leave New Measurements mode. In OPEN's main screen, press **f2** (**Config Menu**), then select the entry "Light Source Control" (described on page 8-4) using the arrow keys, and press **enter**.

2 **Press f1 (Pick Source), and select a light source**
A menu of possible light sources appears. Choose the appropriate 6400-02, 6200-02B, or 6400-40 entry[5], and press **enter**.

3 **Return to New Measurements Mode**
Press **f5** (**Done**), to return to the Config Menu, then **escape** to return to OPEN's main screen, then **f3** (**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[6].

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 1000 µmol m^{-2} s^{-1}.**
(If using a 6400-02 or -02b:) Press **2 f5 Q 1000 enter**.
(If using a 6400-40 LCF:) Press **2 f5 P 1000 enter**.

---

4. The control key label, **f5** level 2, will show "-none-" if OPEN is not configured for a light source.
5. If there are none, then follow the instructions on page 16-6 for the light source or page 27-10 for the LCF.
6. Leaf reflectance plays a big role, but it is stable while the leaf is in the chamber, so we don't worry about it.

4 **Watch PARIN**
The *ParIn_µm* value will go to a value fairly close to the target, and then a few seconds later shift to the exact target value.

5 **Open the chamber**
When you open the chamber the *ParIn_µm* value will drop about 10% due to the sudden decrease in reflectance.

6 **Watch ParIn adjust**
After several seconds, the *ParIn_µm* 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 isn't 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 descibed on page 18-26 (for the light source) or page 27-61 for the LCF.

Control Summary

Even though the four control areas that we've 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: Logging Data

This topic is discussed in detail in Chapter 9, but here we show the basics of how to record experimental data in a file.

Logging Data Manually

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

Experiment #1 **Log a data set manually:**

1 **Press 1 (if necessary) to bring up the logging control keys**

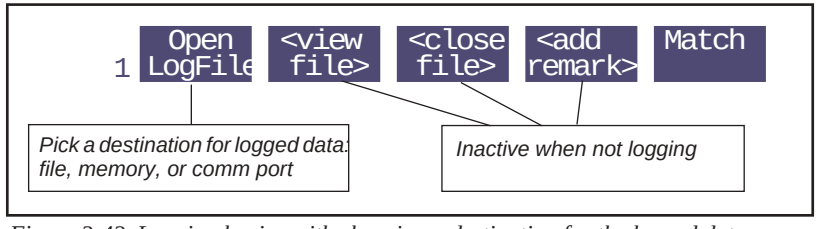


Figure 3-42. Logging begins with choosing a destination for the logged data.

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

3 **Name the destination file**
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.

Press **labels** to access the editing keys, then **f1** (**DelLn**) to clear the default name, then type **Experiment 1**, and press **enter**.

There are some illegal characters for file names (such as ':' and '/'), but this dialog won't 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.

4 **Enter initial remarks.**
You'll be shown the remark entering box (Figure 3-43). Type a remark if you wish, and press **enter**.

Figure 3-43. 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.*

5 **Redo Experiment #1 on page 3-29.**
At each equilibrium value (that is, at the end of steps 4, 6, 8, and 9 of Experiment #1), press **1** then **f1** to record data at those moments.

While the log file is open, the level 1 **f1** function key label (Figure 3-44) will show the number of observations logged.

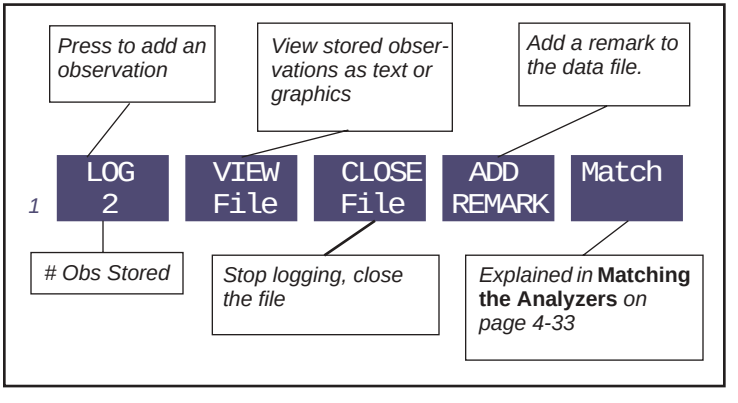


Figure 3-44. Level 1 function key labels when logging to a file.

6 **Graph the data**
Earlier we showed how to view strip charts of data in real time. Now we'll show how to graph data you've 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 or starting over is necessary.

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

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.

1 Access GraphIt

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

After a few seconds, you will see something like Figure 3-45. 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.

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.

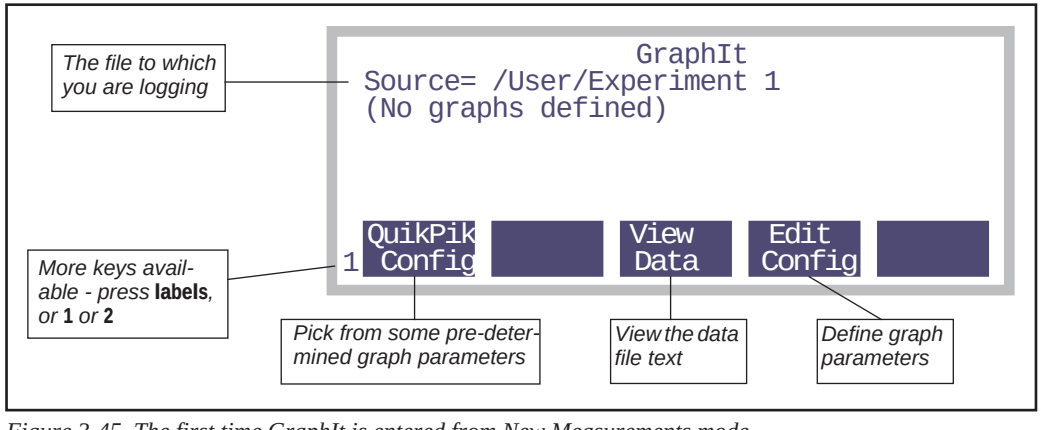


Figure 3-45. The first time GraphIt is entered from New Measurements mode.

2 **View the data file as stored**
Press **f3**, then **F** (Figure 3-46).

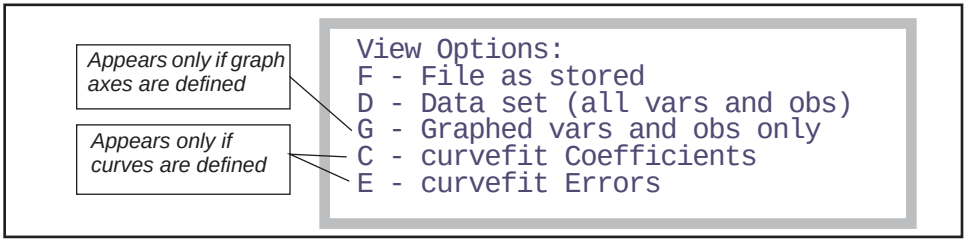


Figure 3-46. GraphIt's View File screen. provides options to view the file or subsets thereof, or to view curve fit coefficients and errors.

After pressing **F**, you should see something like Figure 3-47a. If you press **pgdn**, you'll see more of the file (Figure 3-47b).

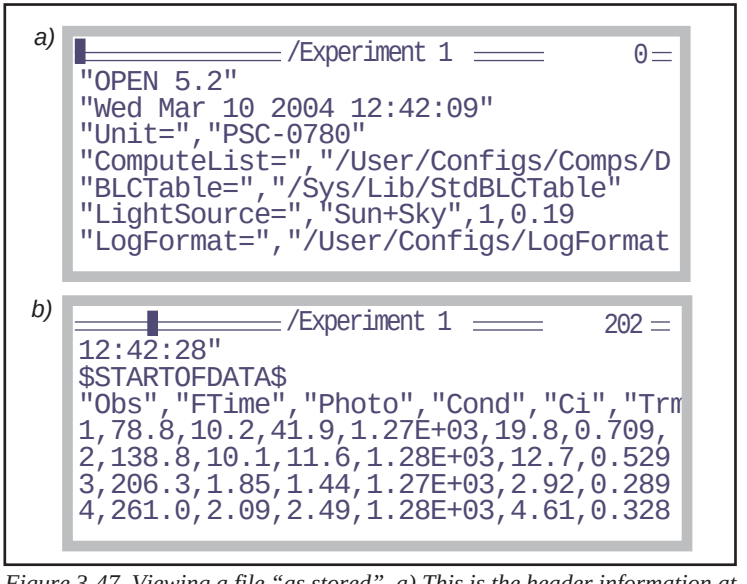


Figure 3-47. Viewing a file "as stored". a) This is the header information at the top of the file. b) After pressing **pgdn**, *the rest of the file is shown. An explanation of the top line banner in each figure is given in Figure 5-1 on page 5-3.*

If you see little square boxes ■ instead of commas on the display, those are tab characters. Version 5.3 changed the default delimiter in files from com-

mas to tabs, but you can set it as you wish with the configuration command LogDelimiter=. (See **Modifying Config Files** on page 16-13).

If you could see all of the file at once, it would look like Figure 3-48. If this seems an inconvenient way to view the data, you'll like Step 3.

```
"OPEN 5.2"
"Sat Jun 26 2004 12:42:09"
"Unit=","PSC-0780"
"ComputeList=","/User/Configs/Comps/Default"
"BLCTable=","/Sys/Lib/StdBLCTable"
"LightSource=","Sun+Sky",1,0.19
"LogFormat=","/User/Configs/LogFormats/Std Output"
"LogCodes="-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,-72,-23
"PromptList=","/User/Configs/Prompts/Default (none)"
"Stability= (CO2S 15s SLP<1)(H2OS 15s SLP<1)(Flow 15s SLP<1)"
"12:42:28"
$STARTOFDATA$
"Obs","FTime","Photo","Cond","Ci","Trmmol","VpdL","Area","StmRat","BLCond","Tair","Tleaf","TBlk","CO2R","CO2S",...
                                    ..."H2OR","H2OS","RH_R","RH_S","Flow","PARi","PARo","Press","CsMch","HsMch","StableF","Status"
1,78.8,10.2,41.9,1.27E+03,19.8,0.709,6,1,2.84,26.36,19.81,26.47,1323.0,1292.3,-0.125,16.570,-0.35,46.75,699.4,0,11,97.21,0,0,  111105
2,138.8,10.1,11.6,1.28E+03,12.7,0.529,6,1,2.84,26.40,20.44,26.51,1316.1,1293.4,8.678,19.358,24.42,54.48,700.2,0,10,97.21,0,0,  111105
3,206.3,1.85,1.44,1.27E+03,2.92,0.289,6,1,2.84,26.49,22.81,26.58,1316.5,1283.1,8.745,25.716,24.48,71.98,100.4,0,11,97.2,0,0,  111105
```

Figure 3-48. Experiment 1 as stored. The label line has been broken in the middle (…)to fit the figure.

3 View the data file in columns

Press **escape** to stop viewing the file "as stored", then press **D** (for Data Set). You will see a more legible display (Figure 3-49).

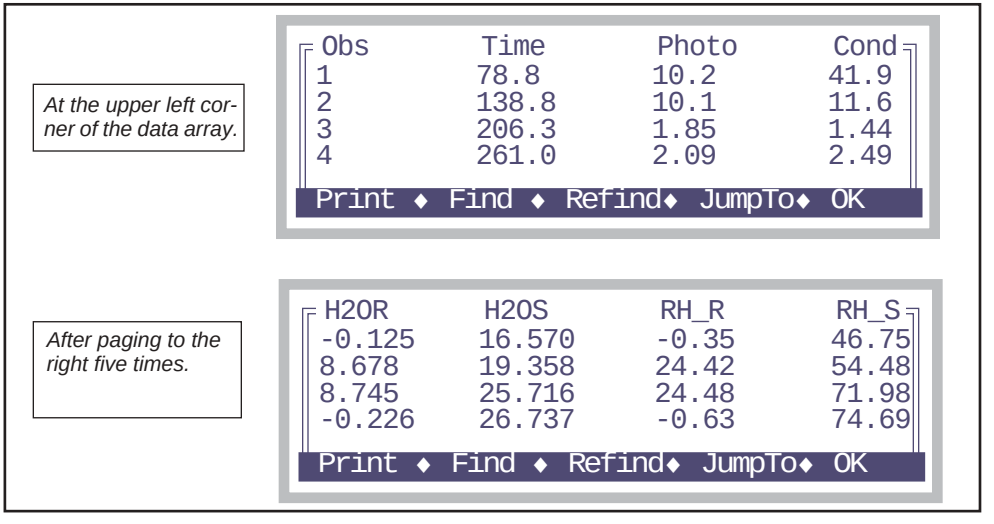


Figure 3-49. Viewing the data set in column format. Only the data is shown. Use **shift** → *and* **shift** ← *to page left and right, and* **ctrl** → *and* **ctrl** ←*to move one column at a time.*

4 Define a plot

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

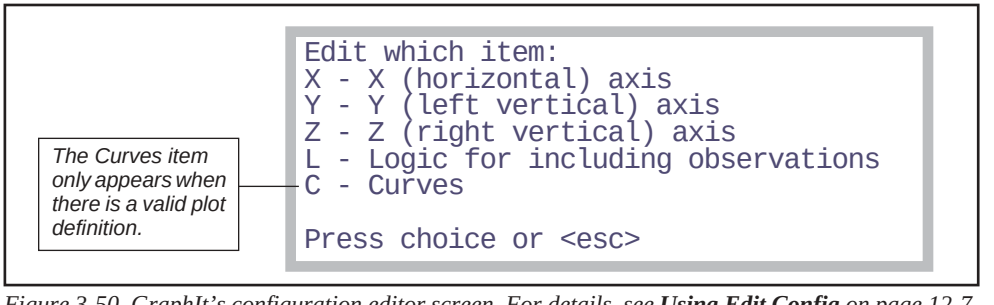


Figure 3-50. GraphIt's configuration editor screen. For details, see ***Using Edit Config*** *on page 12-7.*

5 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**.

You'll be asked for the minimum and then the maximum of the axis (Figure 3-51). (If you had to press **V** to change the variable, you'll have to press **R** now to set the range.)

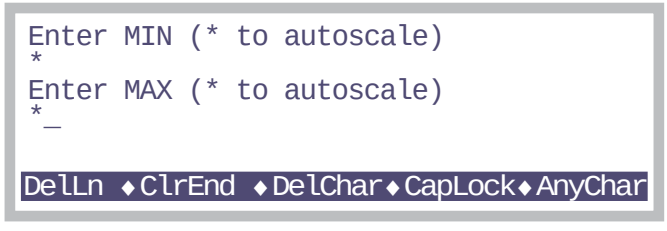


*Figure 3-51. Range information is obtained in two prompts. Enter a * for either the min or the max to let that value be adjusted according to the data being plotted.*

When you are done, if the display does not return to the menu shown in Figure 3-50, you'll have to press **escape**.

6 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 5, press **V**). Highlight RH_S, and press **enter**.

Now press **escape** to signal no more Y variables (you'll 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 min and max are not automatically asked, press **R** to do this).

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

7 **Plot the graph**

If it didn't plot automatically upon leaving the configuration editor, then press **f2** to plot the graph (Figure 3-52).

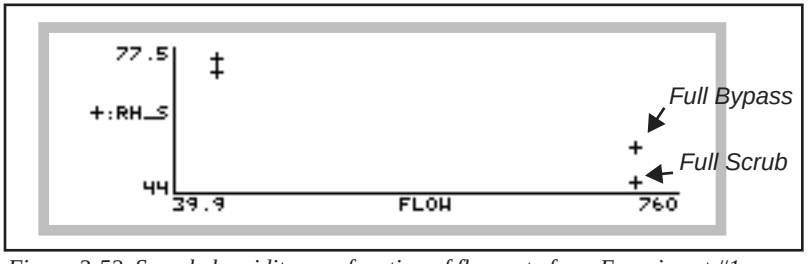


Figure 3-52. Sample humidity as a function of flow rate from Experiment #1.

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 3-53).

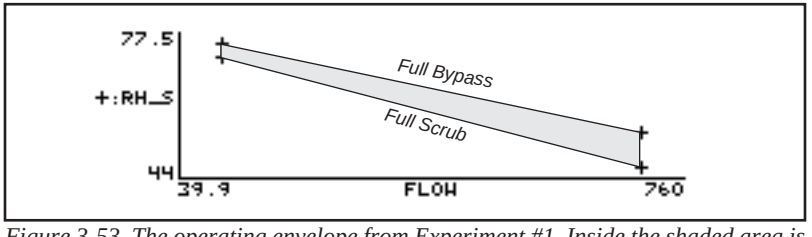


Figure 3-53. The operating envelope from Experiment #1. Inside the shaded area is achievable, outside is not

8 **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.)

9 **Quick! Plot something else!**

From GraphIt's main screen, press **f1** (**QuikPik Config**). You'll 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'll get a very uninspiring plot (Figure 3-54) but it illustrates how to quickly change plot definitions.

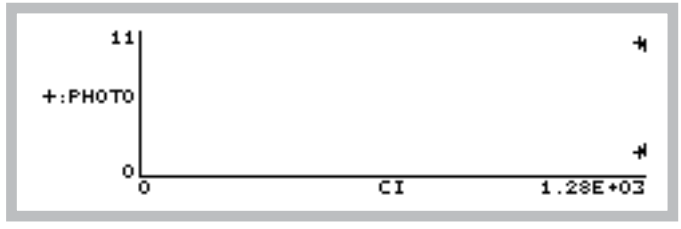


Figure 3-54. A-Ci data from Experiment #1.

10 **Replot RH and Flow…**
Press **escape**, then **f1 (QuikPik Config)**, and select "RH & Flow", to redraw to our first plot.

11 **…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 3-55).

Pt#	Flow	RH_S
1	699.4	46.75
2	700.2	54.48
3	100.4	71.98
4	99.9	74.69

Figure 3-55. Pressing V after drawing a plot in GraphIt will generate a list of the data points plotted.

Press **escape** to exit from this.

12 **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 AutoProgram is 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 9-20), but you can modify these and add to the collection.

Experiment #3 Log Data At Regular Intervals

We'll demonstrate AutoPrograms with one that logs data at regular intervals.

1 Open a data file

Press **1** (if necessary), then **f1**, to name and open a data file.

2 Pick the AutoProgram "AutoLog"

Press **5** then **f1**. A menu of AutoPrograms will appear. Select the one named *AutoLog*, and press **enter** (Figure 3-56).

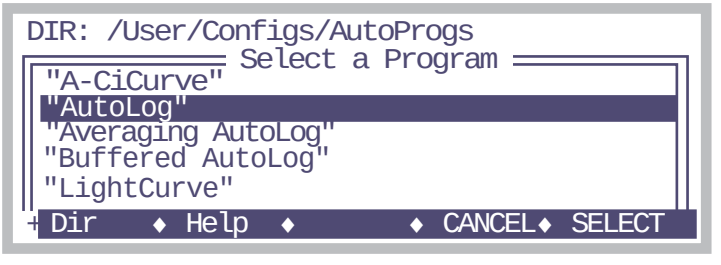


Figure 3-56. Selecting an AutoProgram.

3 Answer the questions

You'll be asked the following three questions in succession:

```
Log every __ secs:
Run for __ minutes:
Auto match every __ minutes (0 = none):
```

To which you should answer 15, 3, and 0 respectively (pressing **enter** after each reply). This will log every 15 seconds, add 10 observations to the data file, and do no matching.

4 Watch

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

Press **K** to watch the AutoProgram status line (Figure 3-57).

Figure 3-57. 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 *AutoLog*, however, *Program* shows the time remaining in the whole program.

5 Terminate it

To stop an AutoProgram before it is done, press **escape**, then **A** (Figure 3-58).

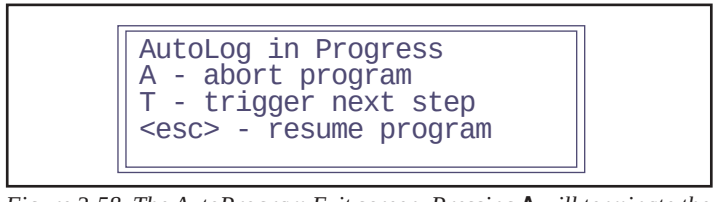


Figure 3-58. The AutoProgram Exit screen. Pressing **A** *will terminate the AutoPrograms,* **T** *will trigger the next step in the AutoProgram, and* **escape** *will let the AutoProgram resume*

Notice the *T* option in Figure 3-58. Normally, this triggers the next step in an AutoProgram. With *AutoLog*, however, it too will terminate the program. (If you want to log early with *AutoLog*, just press the **Log** key.)

6 Close the data file.

Press **1** then **f3** 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.

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

We wouldn't ask that if the answer weren't 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

1 Press f4 level 5.
You'll see a screen similar to Figure 3-59, listing the current stability criteria.

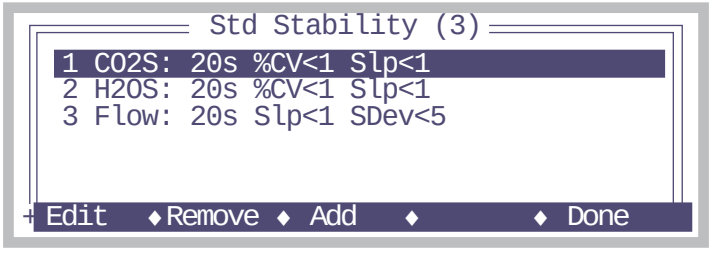


Figure 3-59. Defining stability, accessed by f4 level 5.

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, press **Remove** (**f2**) until there are none. This is not normal operating procedure, but it gives us a clean slate to work with.

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'll 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 degree 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'll specify a some time period, such as 10 or 20 seconds, over which OPEN will keep running statistics; at any instant, 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 Ssp). Table 3-2 illustrates our "tour stability criteria", designed to avoid being fooled by a fluctuating signal.

Table 3-2. 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 Tblk to the list

Press the **Add** function key (**f3**). A list of variables will appear. Select `-1:CO2R`, and press **enter**. Press **Add** again, and this time select `-2:CO2S`. Do

it a third time and **pgdn** to select -10:Tleaf. Your list should now look like Figure 3-60.

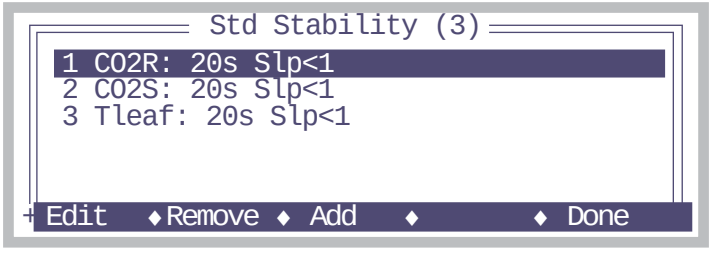


Figure 3-60. After clearing the list, and adding PHOTO and COND.

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

5 Set the rate of change thresholds

Highlight CO2R, and press **Edit** or **enter** (Figure 3-61). 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.

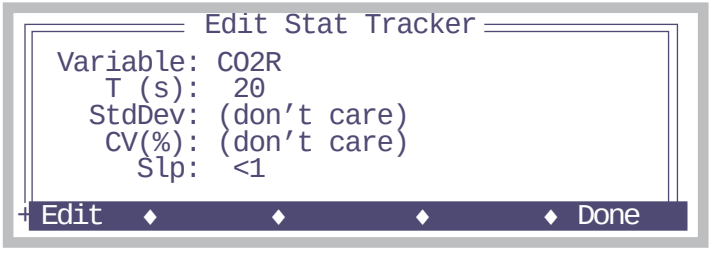


Figure 3-61. Editing an entry in the stability list. T is time period for the running computations, StdDev is standard deviation, CV(%) coefficient of variation, as percent, and Slp is slope (rate of change with time).

Scroll down to the "Slp:" line, and press **Edit** or **enter**. You'll be asked

```
Check Rate of Change?
```

Press **Y**. You'll then be prompted for the threshold. Enter 1.0. Now edit the StdDev: line in a similar fashion following the values from Table 3-2 on page 3-62. Then press **OK** (**f5**).

Now edit CO2S and Tleaf. When done, your list should look like Figure 3-62.

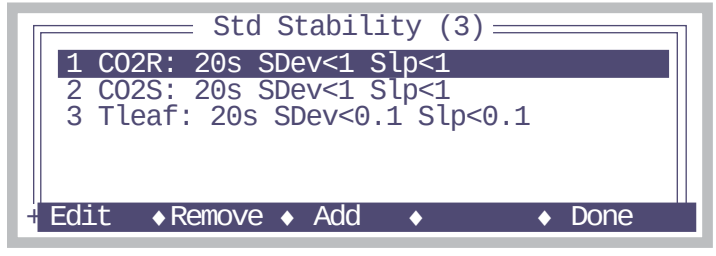


Figure 3-62. Editing an entry in the stability list.

6 Save your list (optional)

Press **labels** to bring up the second level of function keys associated with the stability editor, and press **Save** (**f2**).

A Standard File Dialog will open for storing the stability definition (Figure 3-63). Press **labels**, then **f1** (**DelLn**) to clear the name entry field, and type in **test** (or any name of your choosing), and press **enter** or **SELECT** (**f5**).

Figure 3-63. Storing a stability definition.

7 Exit the editor

Press **Done** (**f5**) and return to New Measurements mode.

In the next part of the tour, we'll 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 3-64, one of the three is stable.

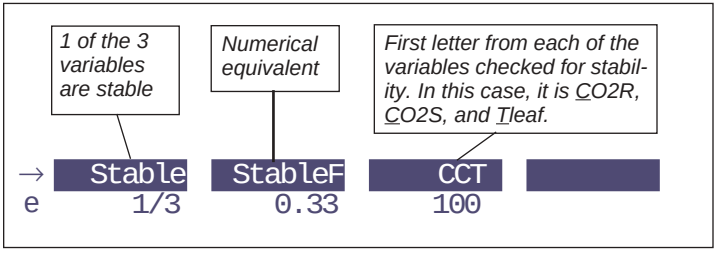


Figure 3-64. Display line e shows three stability indicators. Stable *shows the number of stable variables, and the total number checked.* StableF *is the decimal equivalent of that fraction. The third quantity* CCT *(in this case) indicates which variables are stable or not.*

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'd 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 **6**, then **f5**, labelled **Diag Mode**). You should see something like Figure 3-65. (If not, press **A**, and it will come into view.)

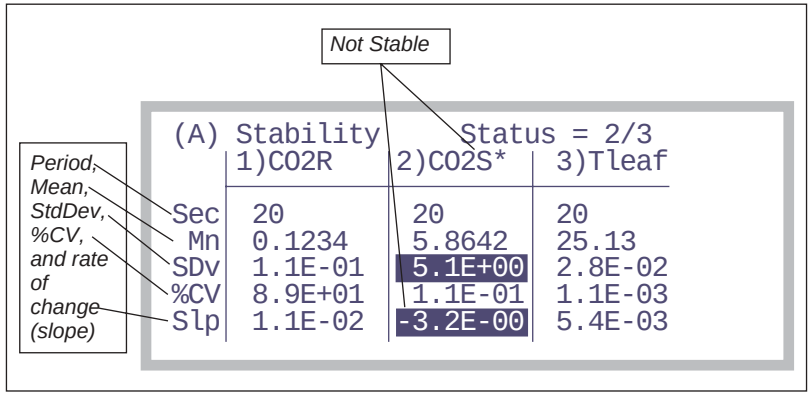


Figure 3-65. Diagnostic display A monitors the details of the stability computations. If there are more than 4 quantities in the definition, ← and → will scroll the columns, while **home** *and* **end** *jump to the left and right limits.*

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 criterion. 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 6-19.

Stability and AutoPrograms

A number of the default AutoPrograms can make use of this stability feature. The programs that do will always prompt you for 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.

Programs that use this feature will give you a chance to adjust your stability criterion prior to starting. You can also change your criterion as much as you like while the AutoProgram is running.

There is one more aspect of stability: How can you log the system's stability details along with your normal data? The answer is next.

Log Options

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

Experiment #6 Log Stability Variable Statistics

1 Access Log Options

Log Options is f3 level 5. When you press it, you'll see something like Figure 3-66.

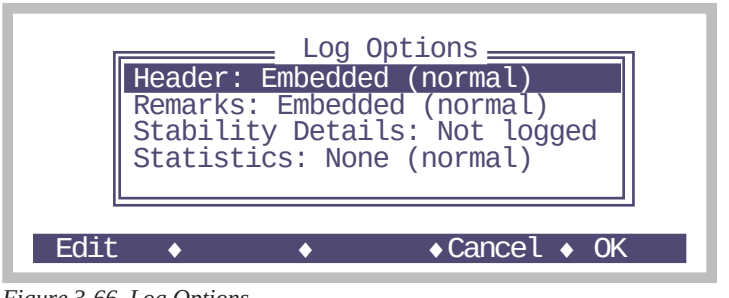


Figure 3-66. Log Options.

The four settings 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 9-9, but for now, focus on the third item, `Stability Details`. Setting this to `Logged` (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.

2 Enable Stability Details

Highlight `Stability Details`, and press **Edit**. The line should change to `Stability Details: Logged`.

3 Exit Log Options

Press **OK** 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.

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       Time      Photo      Cond
1         2.4       -1.67E-6   -1.07
2         5.4       -2.75E-6    2.22
2         6.4       -3.33E-6    2.07
```

Now scroll to the right by pressing **shift** + →. (You can get to the end of the line quickly by pressing **shift** + **end**). Eventually, you'll see

```
Status    N(CO2R)  MN(CO2R)  SD(CO2R)
111115    3        455.34    0.1345
111115    5        456.44    0.1366
111115    6        456.47    0.1256
```

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.

Tour #5: Configuration Introduction

This tour will acquaint you with some basic concepts about managing configurations in OPEN. These concepts are simple and worth mastering, since that can save you from a lot of work and frustration.

If you'd like 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 Make Several Configuration Changes

1 Start with Factory Default

To implement this, go to the Config Menu, select "_Reset Menu", then select "Reset to User Configuration". If there are no other configurations stored, you will find yourself back in the _Reset Menu after a few seconds. If there are other configurations stored, you will be asked to select one (Figure 3-67). If so, then select "Factory Default".

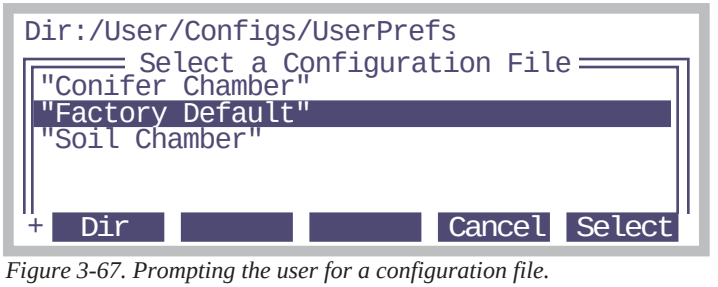


Figure 3-67. Prompting the user for a configuration file.

2 Check the Config Status

Press **escape** to return to the Config Menu, then select the top item, "Config Status". This program will show you all the settings in the current configuration. You should be looking at an empty list (Figure 3-68).

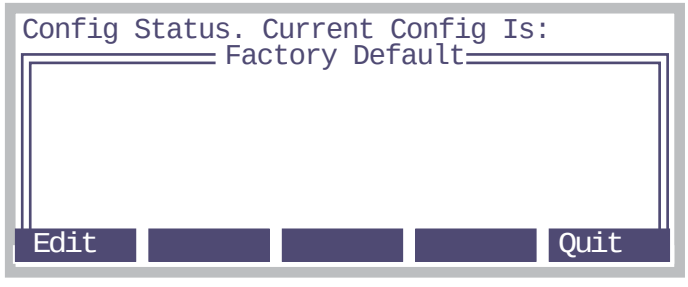


Figure 3-68. Config Status showing the factory default configuration.

If it seems the factory default is nothing, you're right, and here's the first lesson: Configuration files only contain changes to some default configuration. So, an empty configuration file means no changes.

Now, let's go 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 Change the stability criteria

Press **Define Stablty** (**f4** level 5). Your list probably has *CO2S*, *H2OS*, and *Flow*. Add to it *CO2R* and *H2OR*. (Use the **Add** key, **f3**).

Before we leave, let's store this new stability list as a new file. Press **labels** to access the second level of function keys, the press **Store** (**f2**). Name the file Std Stability + Refs, and press **enter**.

Then press **Done** to return to New Measurements.

6 Add a graph

The last change we'll make is to add a new graphics screen. Press **]** to view the graphs, then find a screen that has nothing defined (probably F).

Press **2** to bring up the 2nd level of function keys, then **EDIT** (**f4**) to edit this screen. **TurnOn** plot #1, make it a **Strip**Chart, and **Edit** the variable to be anything you want (scroll up and pick *Trans* (transpiration), for example). Then press **f5** three times (**Done**, **Done**, **HideKeys**), and **]** to return to New Measurements mode.

Notice that we changed the graphics configuration, but didn't store it. This was intentional, to illustrate something that's coming up. (Note the effort here to build dramatic tension...)

7 Return to OPEN's Main Screen

Press **escape** to leave New Measurements mode. There will be a message waiting for you: the Config flag (Figure 3-69).

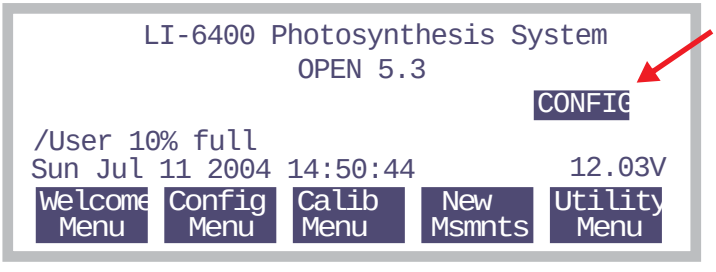


Figure 3-69. The CONFIG flag in OPEN's main screen.

This flag alerts you to that fact that your configuration is changed and not saved. Continue reading, and we'll see how to deal with this condition.

Experiment #2 Save Configuration Changes

1 Use Config Status to view the changes

Go to the Config Menu, then select "Config Status". We no longer have an empty list (Figure 3-70):

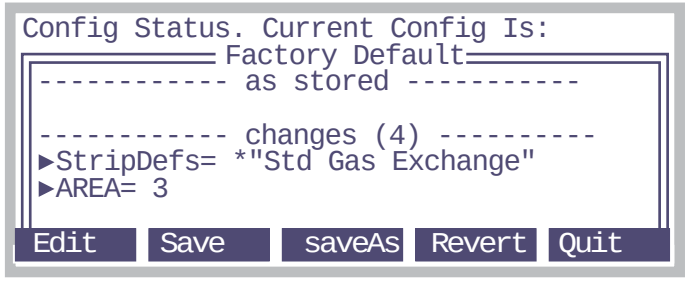
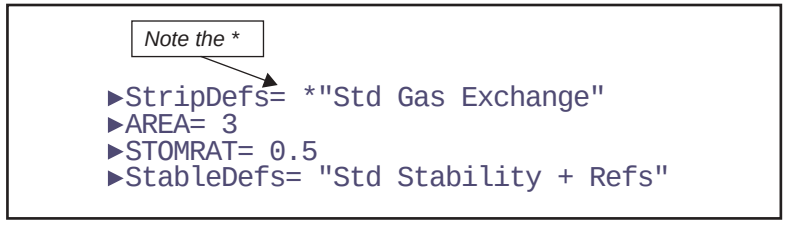


Figure 3-70. Config Status showing some changes.

Scroll down to see all four changes listed.



Now we're starting to see what sort of things are in a Configuration File. Configuration files hold a list of settings, or commands. The *AREA*= and *STOMRAT*= are fairly obvious - they specify the value of area and Stomatal ratio. The real time graphics setting (*StripDefs*=) and stability setting (*StableDefs*=) don't contain values, but instead reference file names.

Remember that we stored the stability change as a new file, but not the graphics. In the changes shown to us by Config Status, the unsaved graphics file is marked with a *.

At this point, our overall configuration (Factory Defaults) has changed and is unsaved. Here are our options:

- **Do nothing**
When we power off, all of our changes will be gone. The stability file "Std Stability + Refs" will still exist, but to use it we'd have to go to the stability list editor, and read it in.

- **Save**
Pressing the **Save** key (**f2**) will change the "Factory Defaults" file by adding these 4 changes.

- **Save As**
If we don't want to mess up "Factory Defaults", we can press **saveAs** (**f3**) and name the file. Then, the next time we power on, or reset to user defaults, that file will be one of our choices. (We'll do this one in a minute...)

- **Revert**
The Revert key will re-read the current configuration (Factory Defaults) and implement it. All of our changes will go away, except for the new stability file we saved.

"But what about our unsaved graphics file?", you ask. (That's a good question, and you should be commended for thinking of it.) Anytime you press **Save** or **saveAs** to save a configuration file, if there are any unsaved files in the list (labelled *), then you will be given a chance to save each of those files, and under a new name, if you wish.

Let's try it.

2 **Save the configuration as a new file**
Press **saveAs**. You'll be prompted as in Figure 3-71:

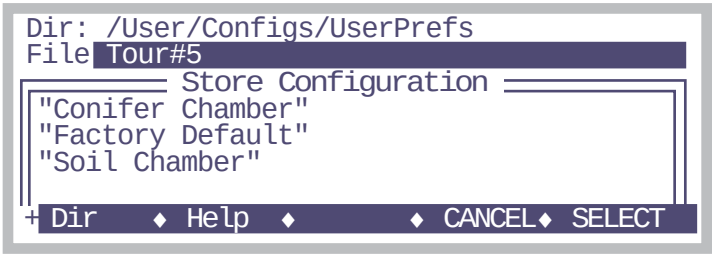
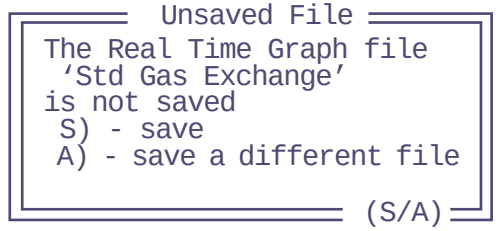


Figure 3-71. saving the configuration as a new file.

Enter **Tour#5** as the file name (press **labels**, then **DelLn** - that's delete line, **f1** - to clear the old name), and press **enter**.

3 **Save your unsaved files**
You'll then get the following message



Press **S** to save it with the old name.

If we had multiple unsaved files, we'd get prompted like this for each one.

4 **Back to Main Screen**
Press escape to return to the main screen, and you'll see the CONFIG flag is gone. We are now in a safe position to power off, if we wanted, and nothing would be lost.

5 **Once more to Config Status**
Go back to Config Status one more time. You'll see there are 3 items instead of 4 listed. What happened to the *StripDefs*= entry?

```
AREA= 3
STOMRAT= 0.5
StableDefs= "Std Stability + Refs"
```

We changed the graphics file Std Gas Exchange, but didn't rename it. Since the name is unchanged, the entry doesn't appear in the list. If we had stored it as a different name back in Step 3 as "Junk", then this entry would appear in our configuration file as

```
StripDefs= "Junk"
```

Press **escape** to return to the Config Menu.

6 **Try out the new configuration**

We have just stored a new configuration named Tour#5. Let's see if we can find it. Select "_Reset Menu", then "Reset to User Configs". You should see Tour#5 in the list (Figure 3-72).

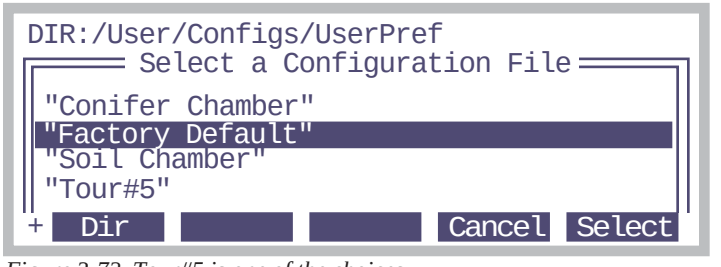


Figure 3-72. Tour#5 is one of the choices.

7 **Change back to factory config**

Select "Factory Default". Then go to New Measurements mode and verify that leaf area is 6 (level 3 function keys) and stomatal ratio is 1. Then check the stability definition (level 5) and see that it in it's original condition.

Now view the graphs. Note that the one we added, screen F, is still there. That's because we changed the definition of the file Std Gas Exchange.

8 **Change back to Tour #5 Configuration**

Go back to the Reset menu, and change configurations to Tour#5. Now go to New Measurements, and verify that the changes from factory default are present.

We are almost done with the tour. The only piece of unfinished business is our changed graphics file. Suppose we wanted to get it back the way it was. Is there a quick way? Keep reading.

Experiment #3 Reinstalling Configuration Files

1 Undo the graphics file change

Go to the reset menu again, only this time select "Config File (Re-)Install". You'll see a screen like Figure 3-73.

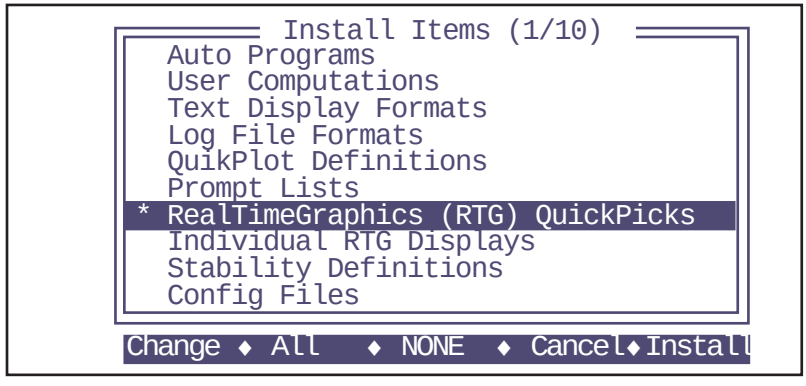


Figure 3-73. Tag the graphics definitions for re-installation.

This program lets you select various categories of files that you might want to reinstall. (It copies them from their home in the /Sys/Open/Defaults folder, to where you use them in the various folders in /User/Configs. In our case, we only want to change RealTimeGraphics definitions, so move the cursor to that line, and press **Change** (**f1**) to tag it. Then press **Install** (**f5**) to do the work.

2 Change to factory default

Now, reset to the factory default, and verify that the original graphics definitions are back.

Equations, Displays, LogLists

One of the strengths of the LI-6400 its adaptability. You can modify its behavior quite a bit to get it to do what you want. One of those areas is in user defined equations. We illustrate that with a step-by-step example of adding a water use efficiency computation. (User defined equations are dealt with in detail in Chapter 15.) Along the way, we'll see how to modify what's displayed and what's logged.

Experiment #4 Add Water Use Efficiency

There are basically three steps needed to do this: 1) Define this variable, 2) Add it to the display, so we can see it, and 3) Add it to the list of things being logged, so we can store it.

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

$$W = \frac{A \times 10^{-6}}{E} \times 100 . \qquad (3\text{-}1)$$

OPEN's user-defined computations, like *PHOTO* and *Cond*, are defined in a file called the ComputeList. We need to access it and change it.

1 Edit the ComputeList

Starting from OPEN's main screen, press **Config Menu** (**F2**), then select "_ComputeList Menu". Then, highlight "Edit the Current ComputeList" and press enter. You'll be editing a file similar to Figure 3-74.

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)"

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20 * 1E3"

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, Tleaf_c, condBL_mol, press_kPa, h2o_2_mm)"

##30 "Photo" "Photosynthesis (umol/m2/s)"
" -#10 * co2_diff_um - co2_2_um * #20"

##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / condBL_mol)"

##36 "Ci" "Intercellular CO2 (umol/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 "

##25 "VpdL" "Leaf VPD (es(Tleaf) - eair)"
" SatVap(Tleaf_c) - eAir_2_kPa"

##27 "VpdA" "Air VPD (es(tair) - eair)"
" satVapTair_kPa - eAir_2_kPa"
```

Figure 3-74. A typical compute list.

The compute list is simply a list of user-defined quantities. Here's a quick summary of the rules: Each item starts out with a ## followed by an ID number. Then there's a label, a description, and a formula. Each of these last three things is between quotes ("like this").

2 Decide where to insert the new variable.
Since we need both photosynthetic rate and transpiration for this equation, insert it after those entries. Move the cursor down below photosynthesis and press **enter** a few times to give you space to work. (see Figure 3-76).

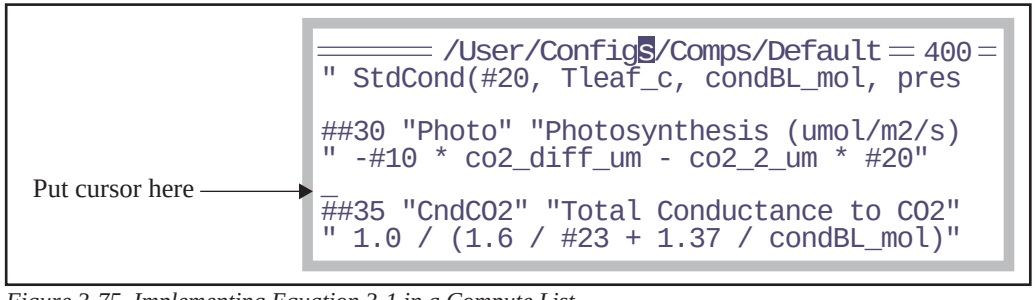


Figure 3-75. Implementing Equation 3-1 in a Compute List.

3 Make the changes to the file
See Figure 3-76.

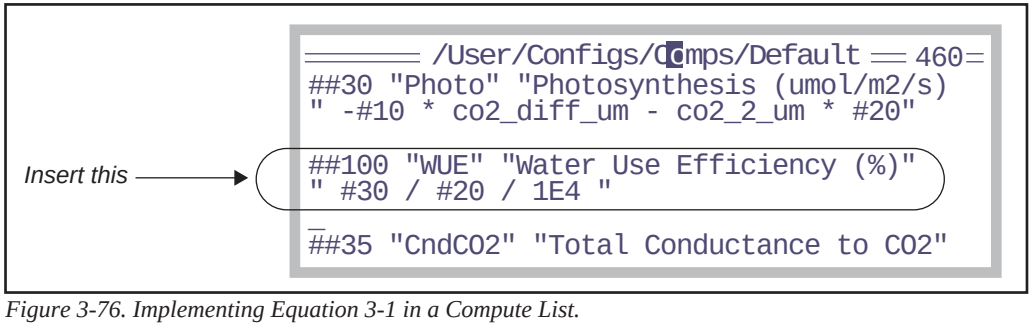


Figure 3-76. Implementing Equation 3-1 in a Compute List.

The label will be *WUE*. The description comes next, but isn't important (if you want to skip it, you still need the quotes (" "). The formula is simple: #30 means photosynthesis (ID #30 right above it), and the #20 means transpiration. Leaves spaces between, and don't forget the quotes.

4 Save under a new name
Press **escape**, then **S**, and name the file "Default + WUE", for example.

5 **Implement the new ComputeList**
While still in the editor's exit menu, press **Q**, then **Y** when asked `"Implement this file?"`.

If you've made some typo, there will be an error reported. If that happens, fix it. Otherwise, this part is done.

Press **escape** to get back to the Config Menu. Now we're ready to add try out *WUE*.

6 **Add WUE to the New Measurements display.**
Go to New Measurements, and press **Display Editor** (**f4** level 6).

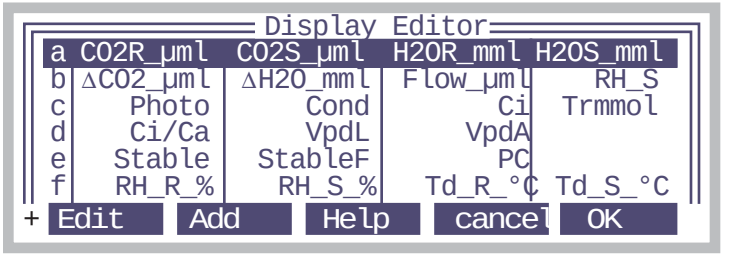
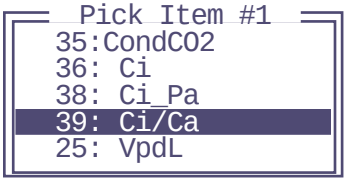


Figure 3-77. The Display Editor, described on page 6-6.

There's room on line *d*, so we'll put it there. Press **D** to jump to that line (or ↓ a few times to scroll to that line), and press **Edit** (**f1**). You will be prompted for each item in line *d*.



When you edit a display line, you are prompted for items until the line is full. Usually, this is 4 items. Press **enter** three times to keep the first three items unchanged. For the fourth item, scroll up to *WUE* (#100, located below *Photo*, near the top of the list), and then press **enter**. (If you don't want to fill a line, press **escape** when you are finished picking items of interest.)

Now we need to save the display. Press **labels**, then **SaveAs**. Name the file `Std Display + WUE`.

Now leave the Display Editor by pressing **OK**. Bring up line *d* on the display, and view *WUE*.

	Ci/Ca	VpdL	VpdA	WUE
d	0.553	1.74	1.78	1.84

Is it being computed correctly?

7 Add WUE to the Log List

In the Config Menu, select "Logging Control" Its screen will look like Figure 3-78.

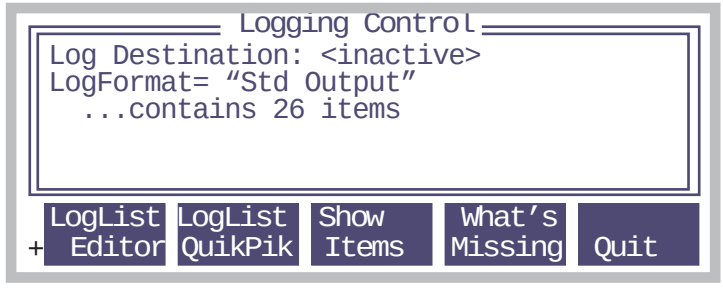


Figure 3-78. The Logging Control screen, described on page 9-8.

Press **LogList Editor** (**f1**). This will show the list of variables to be logged, in the order in which they are written (Figure 3-79).

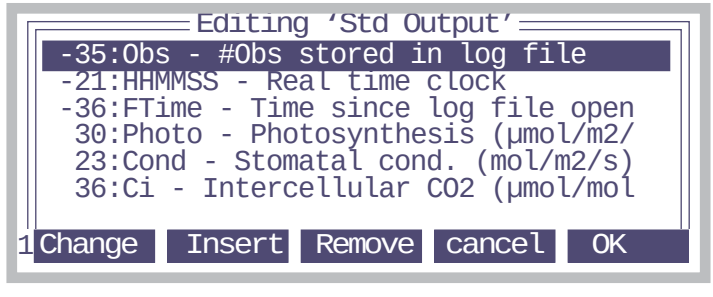


Figure 3-79. The LogList Editor, described in detail on page 9-12.

Insert the cursor in the list where you want to add *WUE*, and press **Insert**. Then, pick *WUE* from the list of variables (it will be near the top), and press **enter** (Figure 3-80).

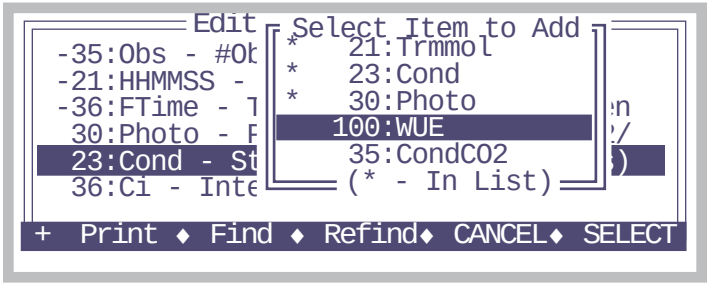


Figure 3-80. Adding WUE to the list.

Now we need to save this log list. Press **labels**, then **SaveAs** (**f5**). Name the file Std Output + WUE.

Next we'll add *WUE* to the display, so we can see it in New Measurements mode.

8 **Leave New Measurements**

Go back to OPEN's main screen. The CONFIG indicator will be showing (Figure 3-69 on page 3-71). This is a reminder that we haven't saved our configuration yet.

9 **Save your configuration**

Go to Config Status in the Config Menu, and press **saveAs**. Name the file Water Use Efficiency, or simply WUE, if you don't like to type.

Now, anytime you power up or do a reset, you'll have this configuration available to choose.

Using The Installation Menu

The Installation Menu lives in the Config Menu. It's normal use is to help you set up configurations for optional accessories that you may have. However, it is also a powerful tool for some other jobs, as we illustrate here.

Experiment #5 **Configure for Needles**

Suppose you wish to measure pine needles instead of broadleaves. This brings up several configuration issues:

- **Leaf Area**
  You probably won't be using 6 cm^2 for leaf area, so at the very least, you'll have to remember to set the area to your best estimate of what the needle area really is. It's easy to enter a value: there's a function key in New Measurements mode for that.[7]

- **Boundary Layer Conductance**
  The default configuration uses a lookup table based on leaf area and fan speed, but that table is appropriate for broadleaves, not needles.

- **Leaf Temperature**
  It's very difficult to get the thermocouple to make good contact with a needle. You'll be better off using the leaf temperature thermocouple to measure chamber air temperature, and compute needle temperature based on an energy balance.

How do you get all of this to happen? The simplest method is to use the Installation Menu:

(Part A) Build An Energy Balance Configuration File

1 **Select "Installation Menu"**
Highlight it, and press **enter**.

2 **Select the appropriate chamber**
Let's make this configuration for the standard 2x3 chamber (clear top, opaque bottom), so select "Std 2x3 Chamber Top", and press **enter**.

3 **Is the calibration entered? Yes**
You will be shown a list of installed chamber tops (anything having a serial number that begins GA- or GB-), and asked if the one you want is in the list. At the very least, there will be a GA- entry for your standard 2x3 chamber top, so just press **Y**.

4 **Build a configuration file? Yes**
You will be asked if you wish to build a configuration file, and of course we do, so press **Y** again.

5 **Select a chamber bottom**
You'll get a menu of the two chamber bottoms that fit the 2x3 chamber: the clear bottom, and the opaque bottom. Highlight the "Std 2x3 Opaque Bottom" entry, and press **enter**.

---

[7] On the other hand, determining needle area is *not* so easy.

6 **Pick the light source**
Highlight the "Sun+Sky" entry, and press **enter**.

7 **Pick the chamber top (if more than one)**
If you have more than one chamber top (that is, a GA- or GB- calibration) installed, you'll be asked to pick which one you'll be using. (This step is skipped if there's only one calibration to choose from, or if you picked an LED light source for Step 6.)

8 **Press E for Energy Balance**
When prompted

```
Leaf Temperature:
Measured or Energy Balance? (M/E)
```

press **E**.

9 **Press N for Needles**
When prompted

```
Enter leaf type
Needles or Broadleaves (N/B)?
```

press **N**.

10 **Admire your work**
You'll next be shown the configuration file, which should appear as in Figure 3-81.

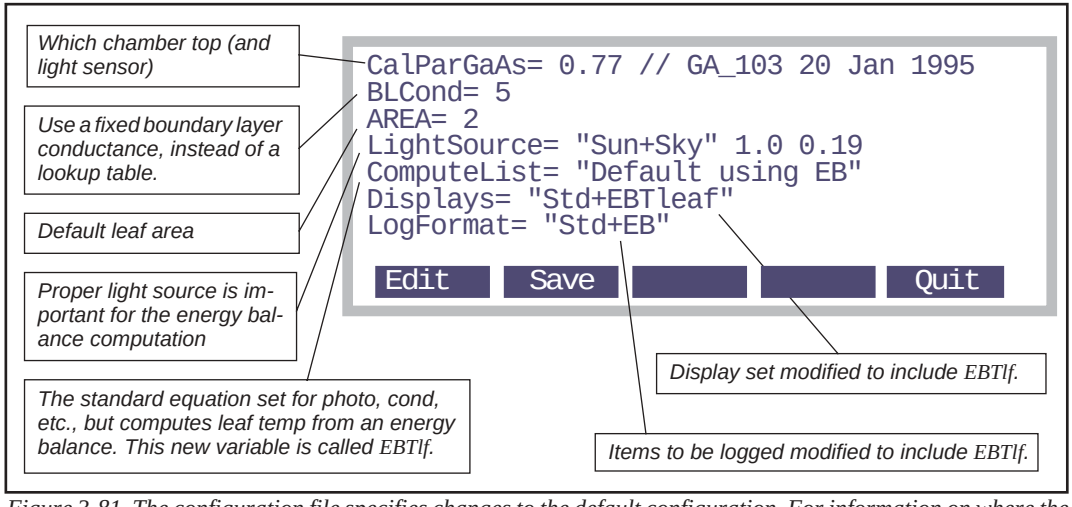


Figure 3-81. The configuration file specifies changes to the default configuration. For information on where the boundary layer and area settings come from, refer to ***Installation Menu: Behind the Scenes*** *on page 16-10.*

11 Name the file

Press **f2** to save the file. The default name for this file will be

```
2x3 Opaque Needles EB
```

Modify the name if you like (press **labels** to access the line editing function keys), then press **enter**.

12 Quit

Press **f5**.

13 No more configurations

Press **N** when asked if you wish to build another configuration.

14 **Read the message**
A message (Figure 3-82) indicating what you have to do to actually implement this new configuration is shown. We'll do this next. Press **enter** or **escape** to clear it.

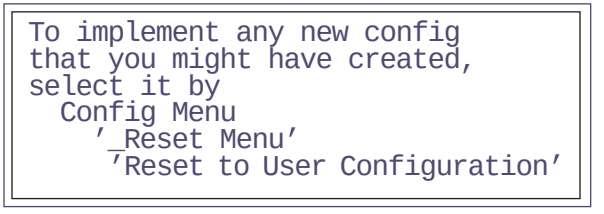


Figure 3-82. Building a configuration file doesn't actually implement it. To do that, you select it when OPEN first runs, or else do a reset and select it then.

15 **Return to the Config Menu**
Press **escape** until you are back in the Config Menu.

(Part B) Implement The New Energy Balance Configuration File

1 **Select the Reset Menu**
Highlight "_Reset Menu", and press **enter**.

2 **Select "Reset to User Configuration"**
Highlight that entry, and press **enter**.

3 **Select the configuration**
After a few seconds, you'll be shown a list of configuration files (there might only be "Factory Default" and "2x3 Opaque Needles EB"). Highlight the latter, and press **enter**.

4 **View the new variable *EBTlf***
Once the configuration has finished loading, press **escape** until you get to OPEN's Main Screen, then go to New Measurements mode, and press **H**. The computed leaf temperature will be labelled *EBTlf*, and the variable *Tleaf*°*C* will be chamber air temperature. (When using this configuration, be sure to pull the thermocouple down a bit so it does not contact the leaf, but measures air temperature in the chamber instead. This will differ from the standard *Tair_C*, since that is air temperature in the sample cell of the IRGA, not the leaf chamber.)

If you now wish to reset to the default configuration, proceed to the Reset Menu, and select "Reset to Factory Default".

Cleaning up

We'll finish our configuration tour by resetting the LI-6400 back to the factory configuration (or, to a configuration of your choice). Access the Reset Menu (in the Config Menu), and select either "Reset to Factory Default", or "Reset to User Configuration". If you do the latter, then select "Factory Default" from the list of configurations, it's equivalent to doing the former.

Tour #6: LPL

If you hold the opinion that computers are a necessary evil, and try to avoid them as much as possible, then feel free to skip this tour and proceed to Chapter 4.

The LPL Screen

LPL stands for LI-COR Programming Language, and its description starts in Chapter 22.

The user interface to LPL is the LPL Screen (Figure 3-83), which allows you to edit files, access the Filer, enter File Exchange Mode, run LPL programs, or execute LPL commands from the keyboard. For details, see **The LPL Screen** on page 5-19.

For this tour, we'll perform a series of tasks to demonstrate what can be done from this level.

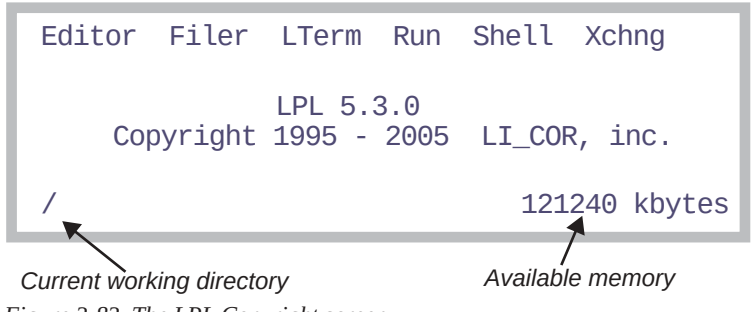


Figure 3-83. The LPL Copyright screen.

View Data Files from LPL

1 **Access the Filer**
Press **F**.

2 **Highlight the file to be viewed.**
Press **D** and select the /User directory, and then highlight a data file.

3 **View the File**
Press **V**, or press **labels** a couple of time, then press **f1** (**View**). The file will be read and shown to you. You can scroll around in it, then press **escape** to leave it.

4 **Plot the data**
Highlight the file to be plotted, and press **H**.

5 **Press escape to leave GraphIt**
Graph something if you wish, but we're moving on.

6 **Return to the LPL Screen**
Press **escape** until you get back (Figure 3-83).

> **Time-Saving Hint:**
> Some file related tasks (plotting, viewing, downloading, etc.) do not require that OPEN be loaded; you can do them from the LPL screen.

■ **Write and Run an LPL Program**
The LPL program you'll see the most is OPEN. The file system contains a number of other programs, however, that don't require OPEN. What we'll do now is have you write a simple program, then run it.

1 **Press E to access the Editor**
This editor is described in **Standard Edit** on page 5-13.

2 **No file name**
You'll be asked

```
Edit What File: /Sys/Autostart/Welcome
```

Press **f1** (**DelLn**) to clear the file name, and press **enter**. You'll be editing a new (empty) file.

3 **Enter a program**
Type the following

```
:fct main {
"Hello there" print getkey }
```

4 Run the program
Press **escape** to view the exit menu, followed by **X**, to run the file you just typed.

```
Hello there
```

will be printed, and the LI-6400 will wait for you to press a key. Press **enter** and you'll return to Standard Edit's exit menu.

5 Save the program
From the exit menu, press **S**, and use the Standard File Dialog to name the file hello, and put it in the directory /User. To specify the directory user (if it's not already selected), press **f1** (**Dir**) and pick it.

6 Return to the LPL Screen
From the exit menu, press **Q**.

7 Run your program again
Press **R**, and enter

```
/User/hello
```

Your program should run again. (Remember, the file system is case sensitive. It won't find Hello if you named it hello.) Press **enter** to end it.

Run another program
Let's run a slightly more useful program from the LPL screen.

1 Press R
You'll be asked for a file name, and the default will be the last one you entered.

2 Use the wild card feature
Press DelLn to clear the file name, and enter

```
/Sys/Utility/*
```

This will provide a list of all files in the /Sys/Utility directory. You can select one to run from the list (Figure 3-84). By the way, these programs are described in **/Sys/Utility Programs** on page 21-12.

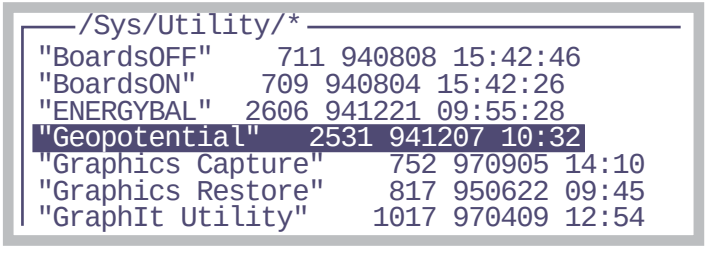


Figure 3-84. Selecting the program Geopotential from a menu generated by entering a wild card for the file name.

3 **Select** "Geopotential"

This program displays the pressure sensor's reading, and computes geopotential height. (It is documented on page 21-17.) It can also be run from OPEN's Tests & Diagnostics Menu, found in the Welcome Menu (Figure 3-6 on page 3-9).

While this program is running, lift the console. You should be able to see the indicated height above sea level (in an idealized atmosphere) change.

4 **Quit**

Press **escape** to exit the program, and return to the LPL Screen.

We're done. If you would like to pursue programming in LPL, you'll find a tutorial and reference in this manual, starting in Chapter 22.

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 4-48).

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 LightSource Control (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 20-14 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 off)[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 18-18). 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 18-4. 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_2O 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 calibration program described in **6400-01 CO2 Mixer** on page 18-21. 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 (see **6400-02(B) LED Source** on page 18-26, or **"Actinic Control - Calibrate"** on page 27-61 for the LCF). 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 18-20).

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_2O

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_2O, 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 6-25).

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 9-7 and **Log Options** on page 9-9.

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 3-29. 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 20-10 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 4-43.)

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 4-55.)

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 *ΔCO2* and *ΔH2O* should each have also increased by a factor of 4.

Question #2: Did you observe a 4-fold increase in *ΔCO2*? How about ΔH2O? If you didn't, why not? (Answer on page 4-55.)

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 4-55.)

Question #4: How might you differentiate real stomatal changes from water sorption effects? (Answer on page 4-55.)

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 4-55.)

Question #6: Will the sample cell CO_2 (*CO2S_µml*) increase or decrease during this step? Why? (Answer on page 4-55.)

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 4-55.)

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 7-10.)

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 7-10.)

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 4-55.)

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 7-10 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 4-56.)

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 4-56.)

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 that 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 4-56.)

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 4-56.)

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 warming scenarios are 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 4-56.)

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 4-56.)

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 be interfering with your measurements. How could you do it? (Answer on page 4-56.)

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 light control menu (**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 4-48. 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 4-51) 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 4-40). 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 9-15) 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. You might wish to install the "Survey Measurements" example configuration (Config Menu → Installation Menu → Examples Menu), and modify that to suit your needs. 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 eternal 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 9-25) and "TimedLamp" (described on page 9-27). "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 "`Enter the desired light values?`", edit the list as needed, and press **enter**.

When asked "`Enter min 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 "Enter max 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| < ppm", enter 15.

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 4-43.

Step-By-Step

Here's how to make an automatic CO_2 response curve. It uses the AutoProgram "A-CiCurve", described on page 9-23.

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.

Flow: Fixed at 300 µmol s^{-1}.

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 "Enter min 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 "Enter max 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.

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:

$$A \propto C_r - C_s \qquad (4\text{-}1)$$

If C_r = 360 µmol mol^{-1} and C_s = 330 µmol mol^{-1}, and there is a 1 µmol 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 µmol 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 14-5 and page 14-6) 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: OPEN 5.2 added the option of an alternative method of matching, useful for soil and turf chambers. See **Matching Without the Match Valve** on page 16-34.

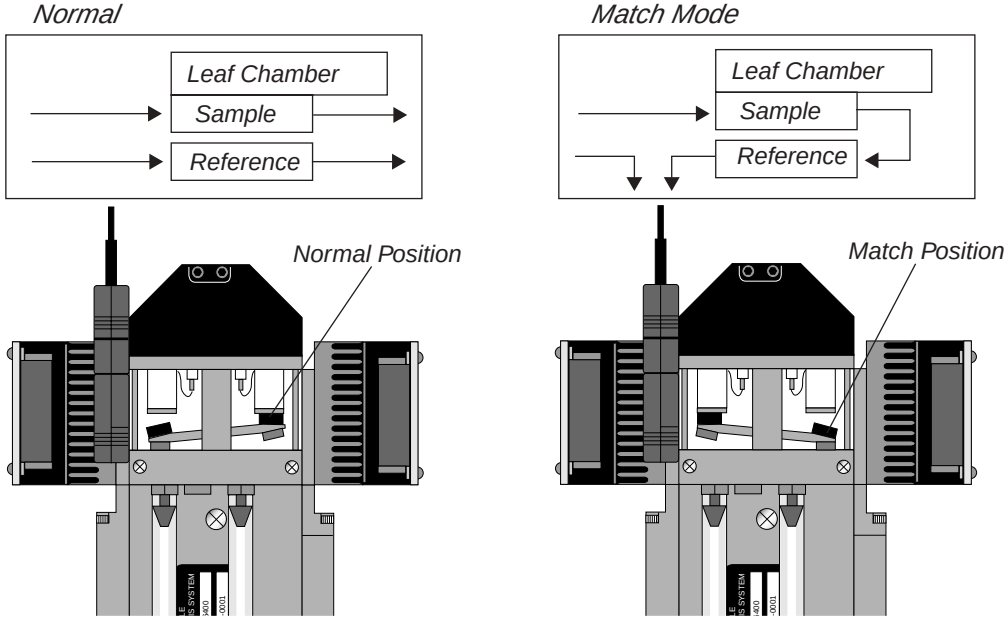


Figure 4-2. The match valve puts exhaust air from the sample cell into the reference cell, allowing both cells to be matched without altering conditions in the leaf chamber.

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}.

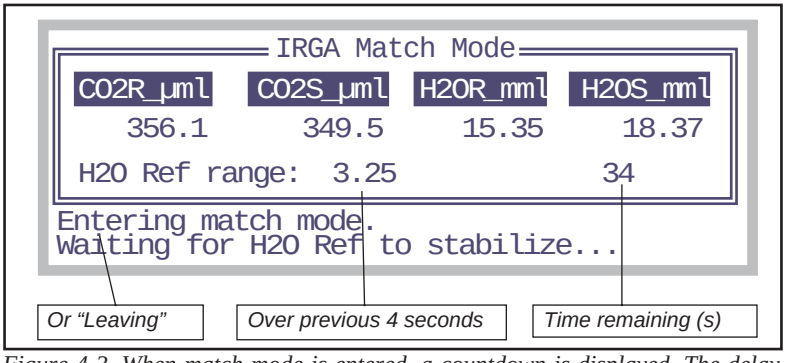
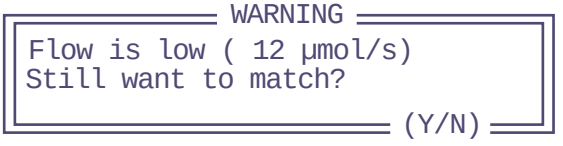


Figure 4-3. When match mode is entered, a countdown is displayed. The delay gives the reference cell time to flush out.

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



giving you a chance to cancel match mode. If you choose to enter, the countdown will likely last the full 45 seconds.

Note that the entry and exit delays can be cut short by pressing **escape** during the countdown.

What Happens in Match Mode

Once in match mode (Figure 4-4), 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.

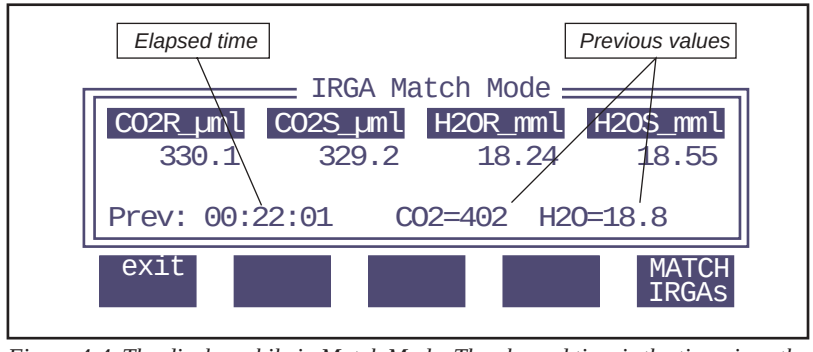


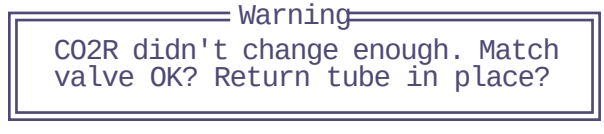
Figure 4-4. The display while in Match Mode. The elapsed time is the time since the previous match, and the previous values are the concentrations at which the last match occurred.

Messages in Match Mode

Match mode has some messages for alerting you to possible problems:

"CO2R Didn't Change"

After the H2O reference reading has stabilized, if the message



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 4-5).

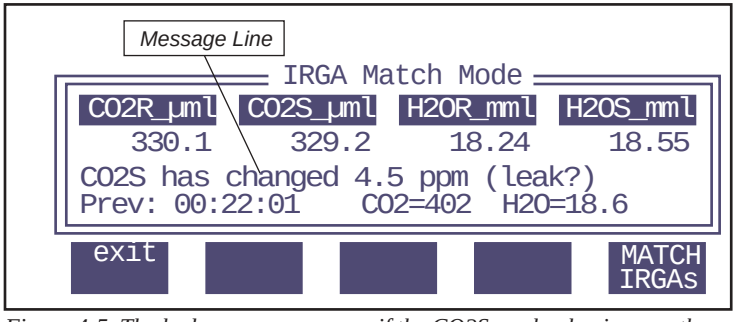


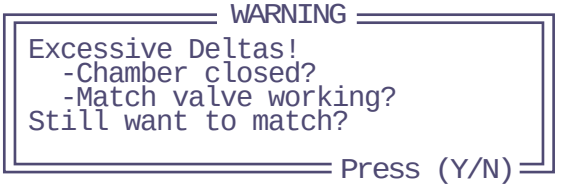
Figure 4-5. The leak message appears if the CO2S_µml value is more than 3 µmol mol^{-1} from the value it had when match mode was entered.

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 20-23 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 IRGAs exceeds 10 µmol mol^{-1} for CO_2 or 1 mmol mol^{-1} for H_2O, a warning will be displayed:



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 20-23 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. More typically, however, the IRGAs may not be so well zeroed and spanned, so matching will be a function of concentration.
- **Periodically**
  How often? That depends most on temperature changes, so it's hard to prescribe 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.

Logging Match Adjustments

The adjustment factors that are computed when a match occurs can be stored each time you log data (Figure 4-6).

"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 4-6. 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 it 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"
```

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 4-36), 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 4-37), this message is logged, but the match still occurs.

```
"14:20:33 Warning! Large Deltas in match: ΔCO2=n,
ΔH2O=n"
```

Modifying the Match Display

While in match mode, press **labels**, then **Edit Display** (**f3**), and use an editor (described in **Display Editor** on page 6-6) to pick the variables you wish to view. Changes will stay in effect until power off. To make the changes permanent, store (when exiting the editor) the configuration in the file "/User/Configs/MatchDisplays".

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 6-25). 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 it 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, that just looking at the only the rate of change could be risky. If you have several variables, then the chances of getting fooled go down.

Logging Statistics

The Log Options dialog includes something that provides information on system stability at the time of logging: The "Statistics" entry (Figure 4-7).

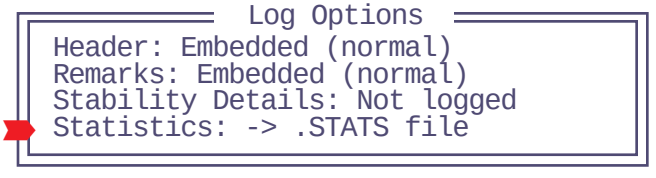


Figure 4-7. The Log Options Dialog

This feature, when set to "-> .STATS file" creates a separate file that contains statistics on each floating point variable logged in your normal 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 4-8. 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.

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 6-11). 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 "AvgTime=" command, which specifies the running average time for all sensors. By default, it is 4 seconds, but you can raise or lower that number if you wish. Chapter 16 discusses this and other configuration commands.

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, how-

ever, 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 synthet-

ic 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 Arabidoposis Chamber.

A diffusion model is presented in Figure 4-9, 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 $(c_a - c_o)$.)

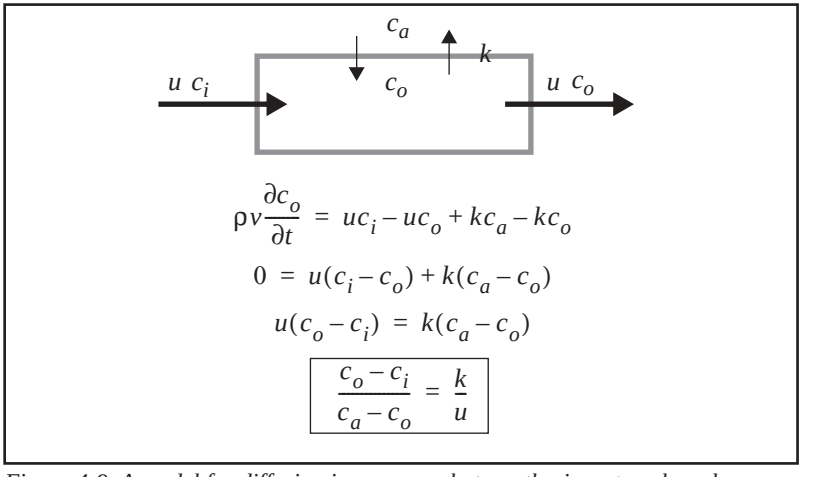


Figure 4-9. A model for diffusion in an open photosynthesis system, based on conservation of mass. The walls of the system have CO_2 leak rate k *(mol s^{-1}).* u *is bulk flow rate (mol s^{-1}), and* c_i, c_o, *and* c_a *are incoming, outgoing, and ambient CO_2 concentrations (mol mol^{-1}). In steady 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 4-10.

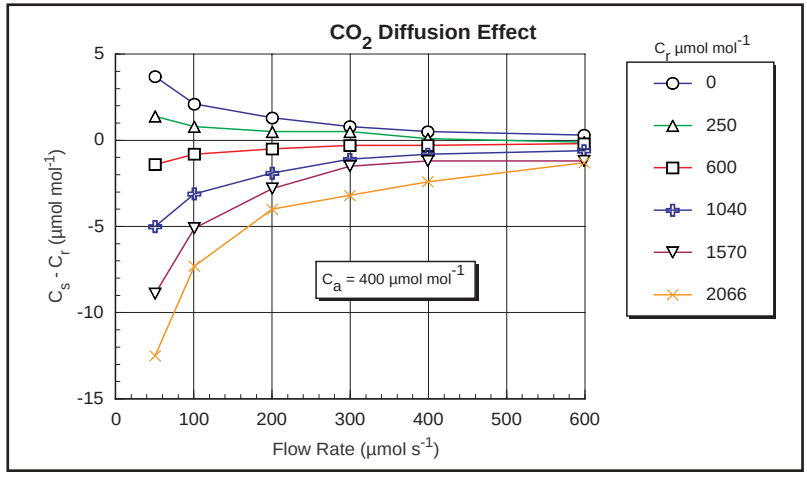


Figure 4-10. CO_2 diffusion into a closed, empty LI-6400 chamber with black neoprene gaskets as a function of flow rate, for various reference concentrations C_r. C_a (ambient CO_2) was 400 µmol mol^{-1}. Diffusion effects are highest at low flow rates and large gradients. The lines in the figure connect data of common C_r.

The diffusion effect $(C_s - C_r)$ plotted in Figure 4-10 is equivalent to the $(c_o - c_i)$ term in the model in Figure 4-9. 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 4-11). 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.

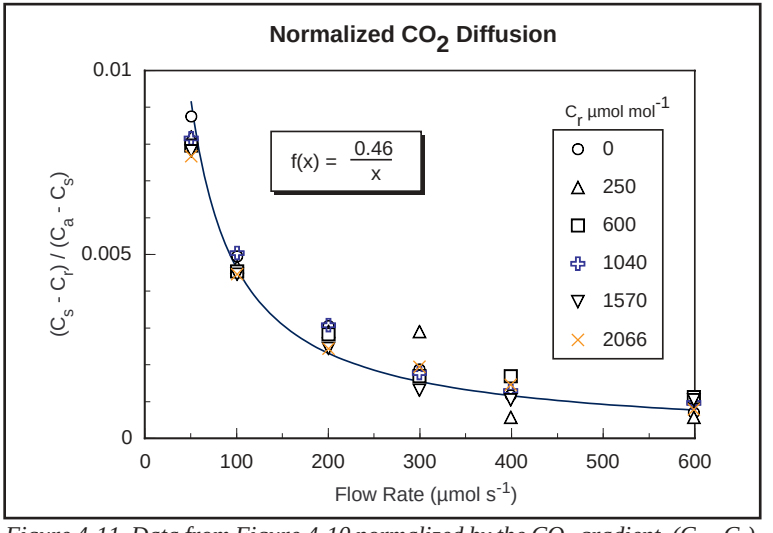


Figure 4-11. Data from Figure 4-10 normalized by the CO_2 gradient, (C_a - C_s).

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	$\frac{F(C_r - C_s)}{100S}$	$\frac{300(380-340)}{100\times 6} = 20$	$\frac{200(50-48)}{100\times 6} = 0.67$
Transpiration Correction	$C_s E$	$340 \times 0.005 = 1.7$	$48 \times 0.007 = 0.33$
Diffusion Correction	$\frac{k(C_a - C_s)}{100S}$	$\frac{0.4(370-340)}{100\times 6} = 0.02$	$\frac{0.4(370-48)}{100\times 6} = 0.21$
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.

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 damped, and can be stable enough to use for gas exchange purposes (Figure 4-12).

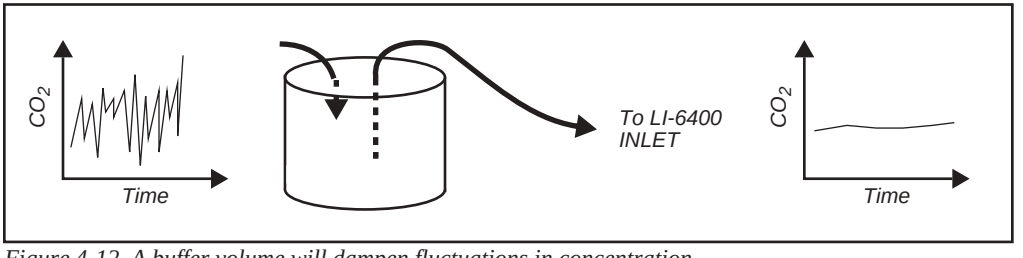


Figure 4-12. A buffer volume will dampen fluctuations in concentration.

Acceptable volumes depend on the magnitude of the fluctuations that need to be damped, 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 4-13 illustrates how to do this using a T arrangement.

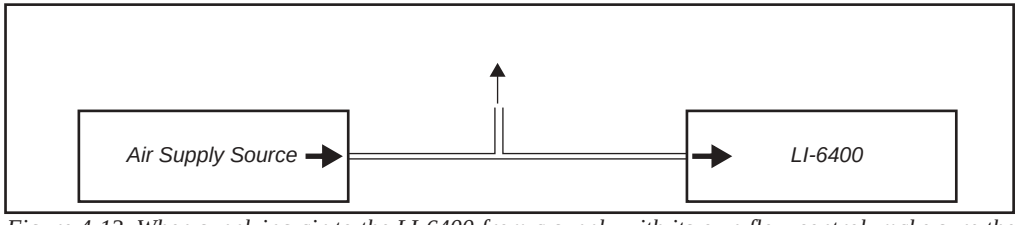


Figure 4-13. When supplying air to the LI-6400 from a supply with its own flow control, make sure the supplied air flow exceeds that required by the LI-6400, and that there is excess flow coming out the T. Make sure the open leg of the T is long enough to prevent diffusion from contaminating the air to the LI-6400.

What's the Light Source?

Make sure the "LightSource=" configuration command matches as closely as possible the actual light source you are using (**The Light Source Control Utility** 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 Sensor Considerations** 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 4-43.

- **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 4-14). 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 4-14 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 moist-

ened (and clumpy) pellets. Avoid adding too much water; 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.

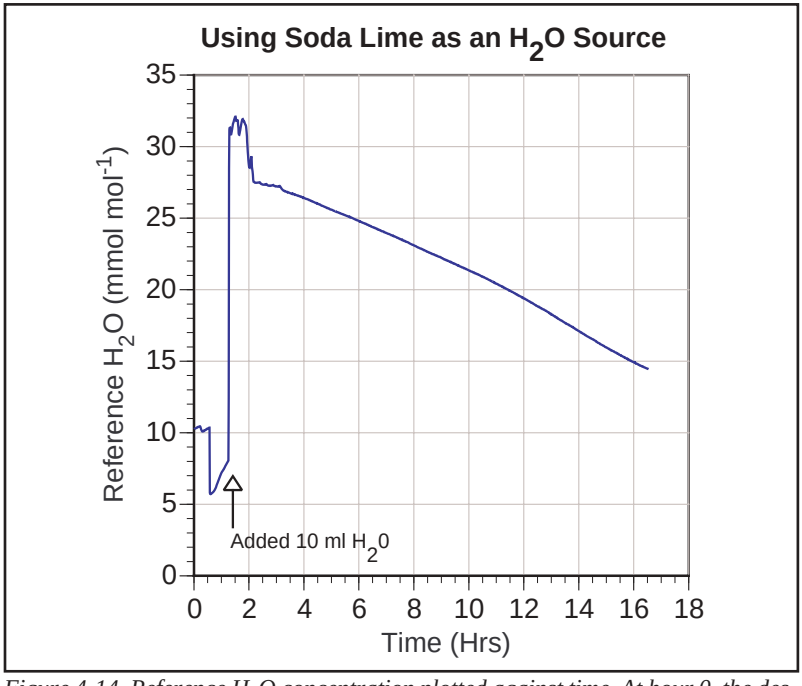


Figure 4-14. Reference H_2O concentration plotted against time. At hour 0, the desiccant and soda lime tubes were fully bypassed, and ambient humidity was measured. After 30 minutes, the soda lime was turned full on, and the water concentration dropped, then began to climb back toward the ambient value. 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 4-15). The excess flow from this device is dumped into the reference leg, providing faster response times in the reference IRGA.

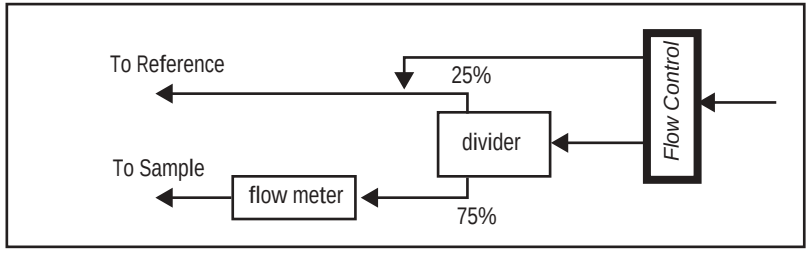


Figure 4-15. Flow schematic for units with a 6400-01 CO_2 mixer. Flow control is done with a diverter valve, and the excess flow is dumped into the reference flow.

(The complete schematic is in Figure 20-12 on page 20-41). However, when the flow control unit is routing most 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 4-16), 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.

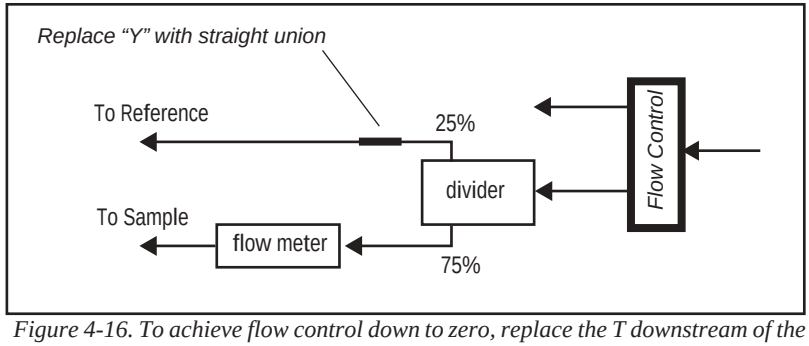


Figure 4-16. To achieve flow control down to zero, replace the T downstream of the diverter with a straight union.

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 takes 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 4-51, for example.

Part II

Useful Details

5

Standard Tools

Menus, Editors, and File Dialogs

5 Standard Tools

As you use the LI-6400, you will encounter a few interface tools again and again. The three most common are 1) a dialog for selecting a file, 2) a menu for selecting one of several choices, and 3) a single line editor for entering a remark or numeric value. This chapter fully describes these tools, along with some other ones that you may not see so often.

Standard Menu

Menus have two basic uses (Figure 5-1):

1. **Picking one of a number of possibilities**
   OPEN's Welcome 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.

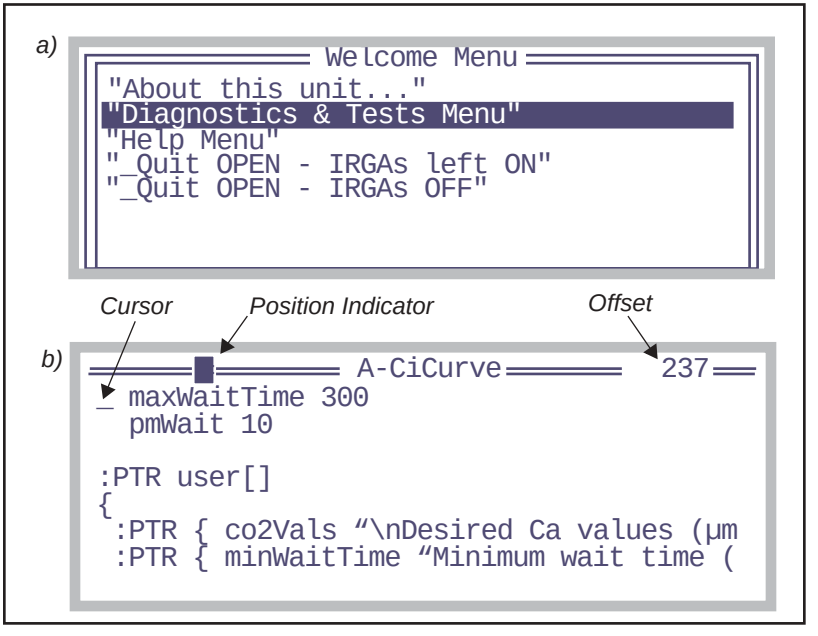


Figure 5-1. Two examples of Standard Menu in action. a) a list of selections with a highlighted bar that the user can move up or down with the arrow keys. b) viewing uneditable text. In this example, there is a banner that includes the file name. To the right is the cursor location, expressed as bytes 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 5-2 illustrates the function keys, and Table 5-1 describes their meaning.

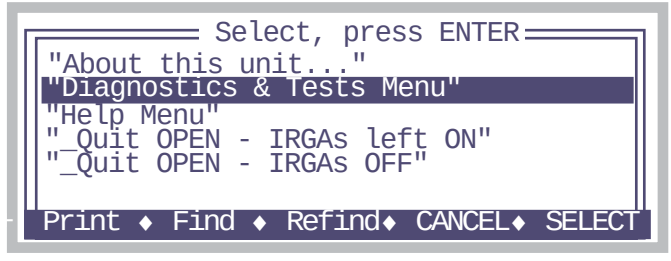


Figure 5-2. Pressing **labels** *causes the function key labels to appear until the next keystroke.*

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).

Table 5-2. (Continued)Standard Menu cursor key definitions

Key	Action
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 5-3. 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 5-4).

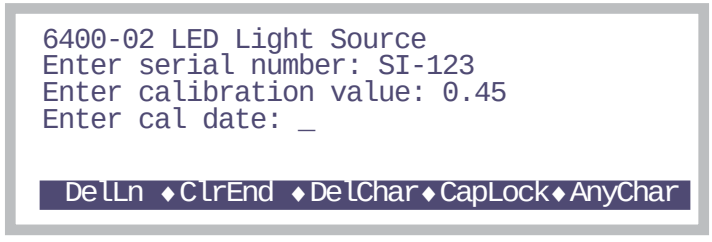


Figure 5-4. The succession of questions asked when installing a light source (Installation Menu) use the Standard Line Editor, but without visible windows.

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 Edit 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 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)

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 5-5) allows you to view all the characters in the character set, and select the one you want (Figure 5-5). The character codes are viewable in decimal or hex. Pressing **D** and **H** toggle 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.

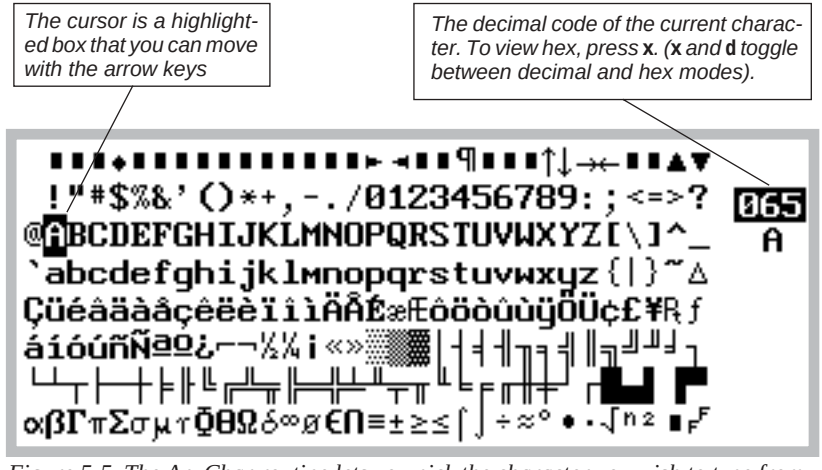


Figure 5-5. The AnyChar routine lets you pick the character you wish to type from a map.

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 5-6 shows the dialog boxes for a) selecting existing files (usually for reading), b) and for selecting new or existing files (usually for writing).

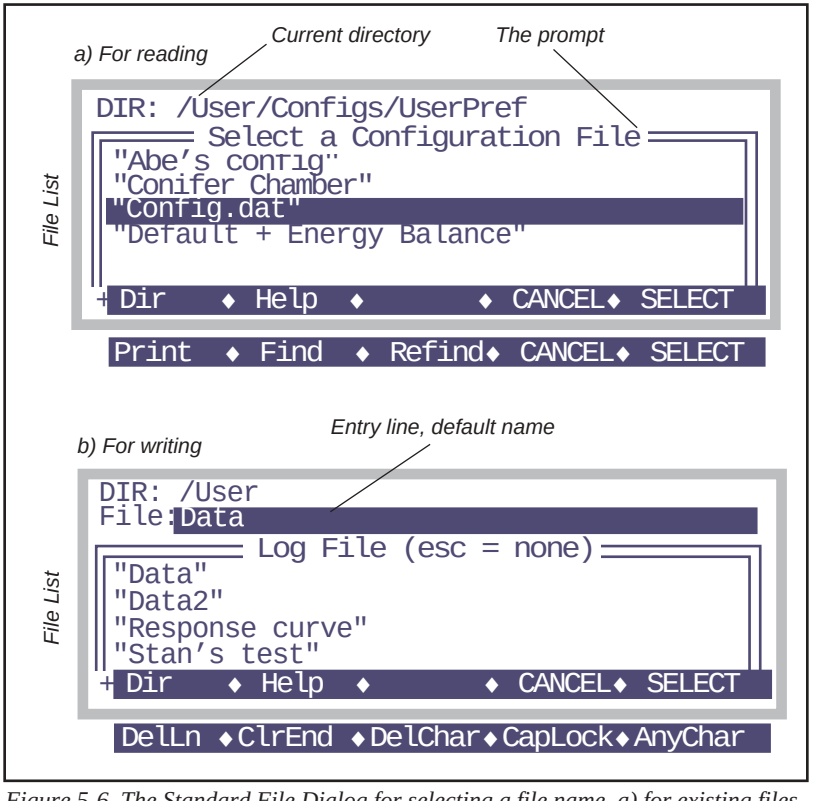


Figure 5-6. The Standard File Dialog for selecting a file name. a) for existing files. b) for new or existing files.

For Reading

When selecting a file for reading (Figure 5-6a), the dialog is essentially a menu. There are two levels of function key definitions (Figure 5-7), and **labels** is used to toggle between the two. Note that this dialog is simply a variation of the Standard Menu; an extra layer of function key definitions has been added.

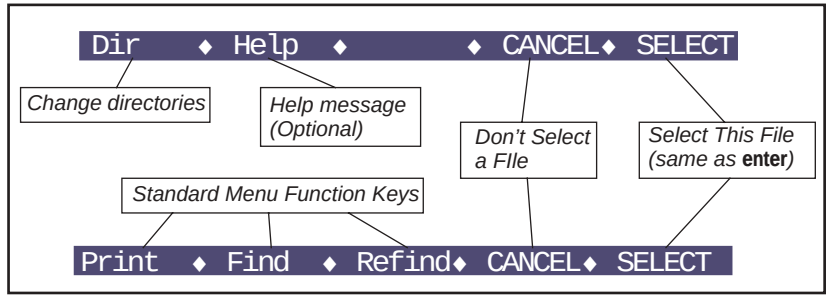


Figure 5-7. Function keys for Standard FIle Dialogs for existing files.

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).

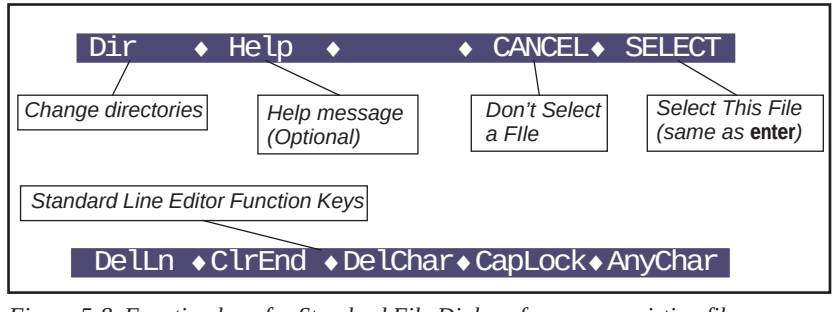


Figure 5-8. Function keys for Standard File Dialogs for new or existing files.

Note that the file list is scrollable with the ↑ ↓ **pgup pgdn**; even though you are not using that list like a menu, you are still able to view all of the file names in that directory.

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**.

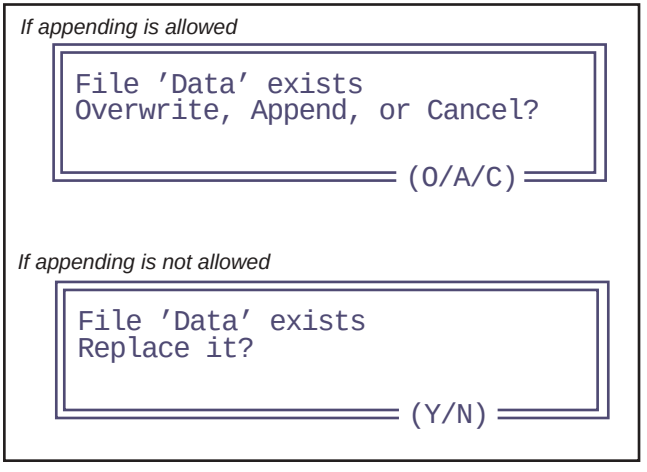


Figure 5-9. When an existing file is specified (that is different from the default file name), you will be notified with one of these prompts. The message depends on whether appending is permitted.

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 5-10.

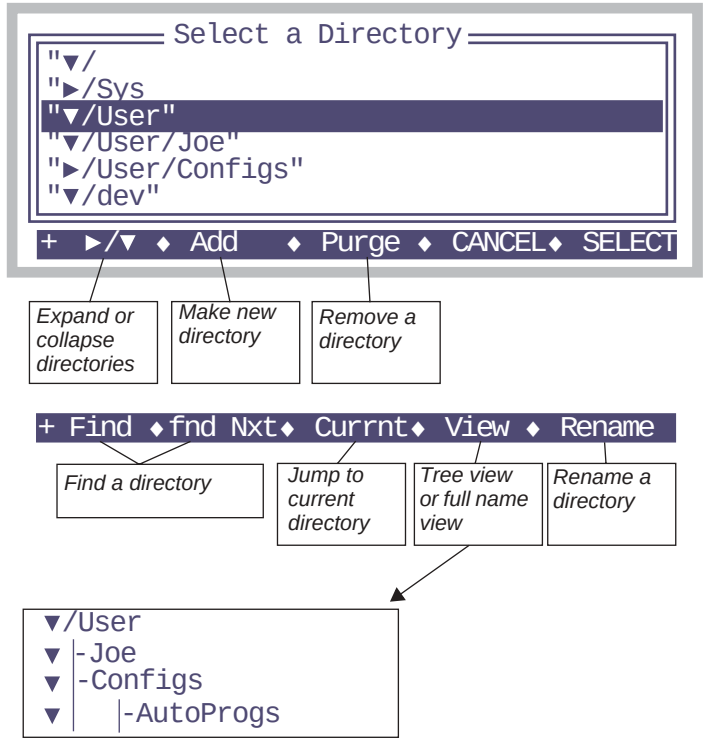


Figure 5-10. Selecting a directory in a Standard File Dialog is done from a Standard Menu accessed by pressing the **Dir** *function key.*

As an example, when you open a data file in New Measurements mode, you will see the Standard File Dialog in Figure 5-6b. 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 `"New File (Editor)"` 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 5-11).

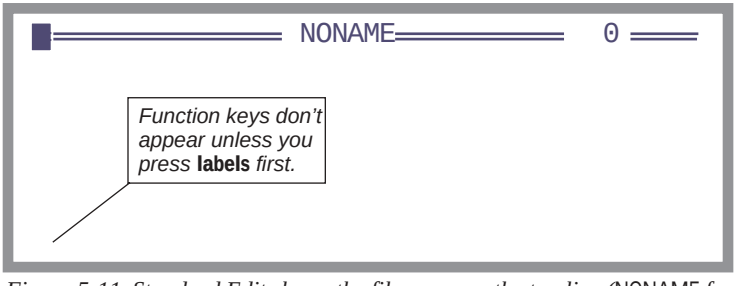


*Figure 5-11. Standard Edit shows the file name on the top line (*NONAME *for new files).*

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.

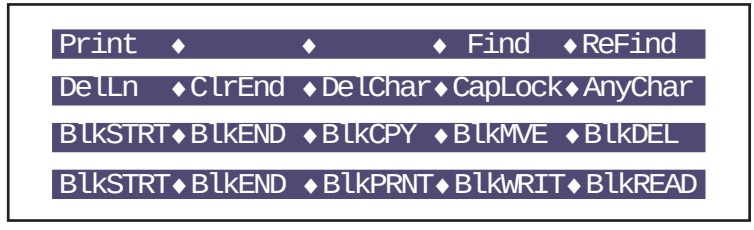


Figure 5-12. The four level of function keys defined in Standard Edit.

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.
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 5-4).

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.

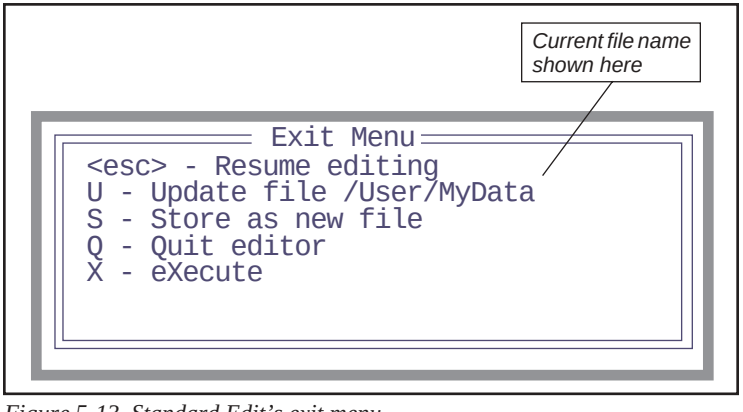


Figure 5-13. Standard Edit's exit menu.

Table 5-6. Standard Edit's exit menu options

Press	Action
escape	Resume editing.
U	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.
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	Execute the contents of the edit buffer as an LPL program. This makes sense only if the edit buffer is an LPL program, of course.

Hot Keys

Most keys are context dependent, but there are some special combinations (Table 5-7) that will *always* work regardless of what else is going on.

Table 5-7. Hot key combinations.

Press...	To Do...
ctrl shift ↑	Darken display contrast.
ctrl shift ↓	Lighten display contrast
ctrl shift home	Toggles display backlight (if installed).
ctrl shift →	Turn graphics mode on. If already on, turns text mode off.
ctrl shift ←	Turn text mode on. If already on, turns graphics mode off.
ctrl escape	Abort the current application.
ctrl shift escape	Reboot, and display the boot screen.

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.

The Boot Screen

The purpose of the Boot Screen is to update software. See **Installing System Software** on page 2-22.

Version 5.0 and above

```
INITIALIZING.
BOOTING PAUSED
USE THE LI-6400 UPDATE PROGRAM
```

Prior to Version 5.0

```
Welcome to LI-6400
  (R)un              (V)oltages
  (U)pload           (S)save parms
  (C)heck CRC        (+/-/UP/DOWN) contrast
  (B)ps = 115200     (D)ebug: OFF
  (T)erminal
Select:_
```

Figure 5-14. The Boot Screen is version dependent.

You can prevent the boot program from launching the LPL operating system by powering on with **escape** held down. If you do this, the display will show the Boot Screen

You can also get to the Boot Screen at any time by pressing

shift ctrl escape

This is quite drastic, since it restarts the computer. With version 5, this will appear to take about 5 seconds, and you'll need to keep the **escape** key pressed.

The LPL Screen

The LPL screen (Figure 5-15) represents the user interface to the LPL operating system, much like the prompt

```
C:>
```

in DOS indicates that you are at the operating system level.

You normally do not see this screen, because the Autostart feature (page 5-22) automatically launches OPEN.

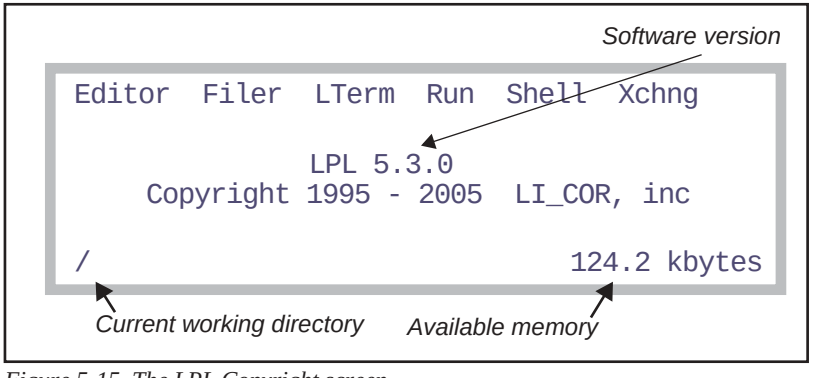


Figure 5-15. The LPL Copyright screen.

Table 5-8 summarizes the LPL Screen's options.

Table 5-8. 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
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

LTerm is discussed in **Control via Terminal Software (LTerm / iTerm)** on page 11-16.

The Shell Program

Pressing S launches the file "/Sys/Lib/StdShell". When run, this program will appear as in Figure 5-16.

Figure 5-16. 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)
```

A Stop Watch

Figure 5-17 illustrates a method to use StdShell to make a stop watch.

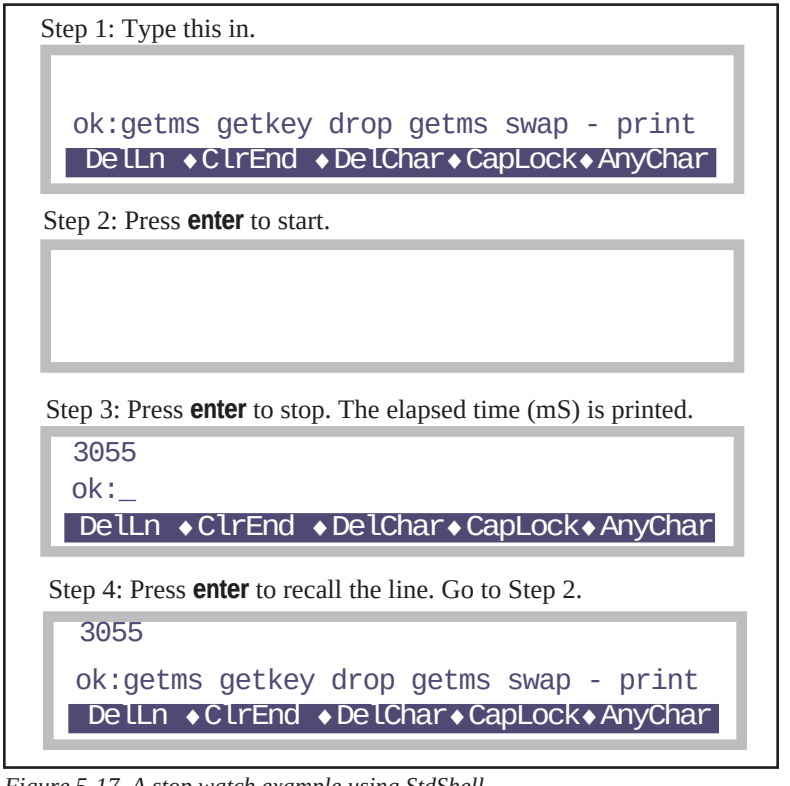


Figure 5-17. A stop watch example using StdShell.

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

Sets the display contrast to a reasonable value if it is full on or off.

BackLightON

Turns on the display backlight, if present.

CheckParm0

Checks to see if there is a /dev/parm0 file. See **“Warning: File ‘/dev/parm0’ is missing”** on page 20-4.

CheckParm1

Checks to see if there is a /dev/parm1 file. See **“Warning: File ‘/dev/parm1’ is missing”** on page 20-5.

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

6 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.

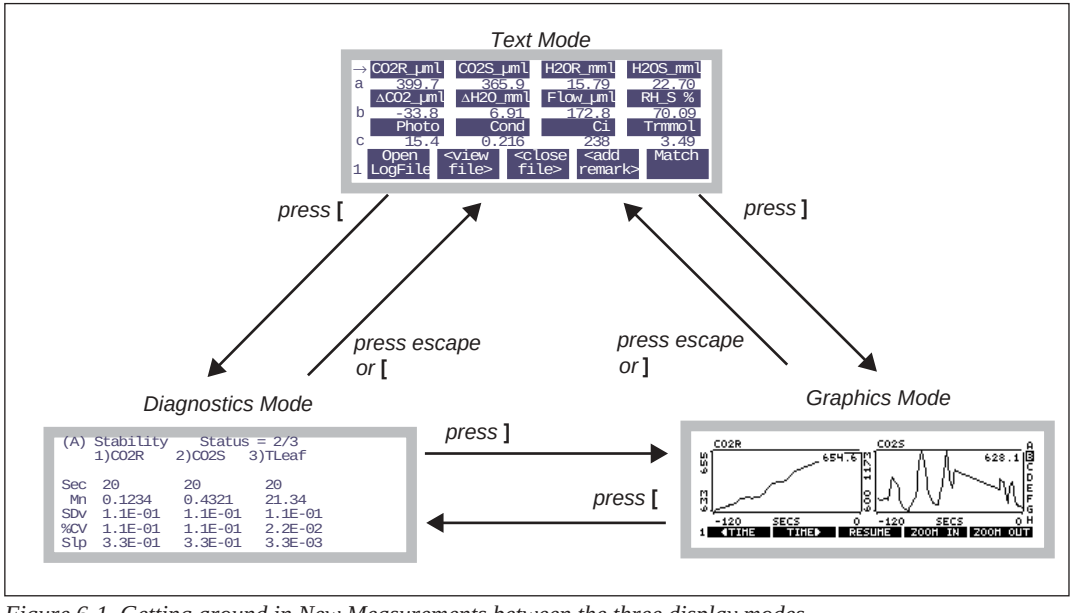


Figure 6-1. Getting around in New Measurements between the three display modes.

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 LCF 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 LogFile** 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 <rspns> | Flow= 500µms | Mixer OFF | Temp OFF | Lamp= OFF	
Rspns	Sets the response of the humidity control Only active when doing humidity control. See **Constant Humidity Operation** on page 7-9.
Flow=	Sets flow control mode. See **Humidity Control** on page 7-7.
Mixer=	Sets CO2 mixer control. See **CO2 Control** on page 7-13.
Temp=	Sets chamber cooler control. See **Temperature Control** on page 7-16.
Lamp=	Sets lamp control. See **Light Control** on page 7-18.
3 AREA= 6.00 | STOMRT= 1.00 | LeafFan Fast | Prompts off | Prompt All	
AREA=	Sets leaf area.

Table 6-1. Summary of LCF function keys, New Measurements Mode (Continued)

Label	Description
STOMRT=	Sets stomatal ratio.
LeafFan	Chamber fan control.
Prompts	Sets prompts behavior (if defined). See **Prompts and Remarks** on page 9-15.
Prompt All	Prompt for all defined prompts. See **Prompts and Remarks** on page 9-15.
4 [ ] [GRAPH QuikPik] [View Graph] [GRAPH Setup] [ ]	
GRAPH QuickPik	Setup a group of real time graphics screens. See **Real Time Graphics** on page 6-11.
View Graph	View graphics screens. Same as pressing **]**.
GRAPH Setup	Setup one or more graphics screens. See **Real Time Graphics** on page 6-11.
5 [AUTO PROG] [ ] [Log Options] [Define Stablty] [Define Log Btn]	
AUTO PROG	Launch an AutoProgram. See **AutoPrograms** on page 9-20.
Log Options	Log Options are discussed in **Log Options** on page 9-9.
Define Stablty	Stability is discussed in **Stability Indicators** on page 6-25.
Define Log Btn	Define the log button behavior. See **User Definable Log Button** on page 9-7.
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 LCF 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.

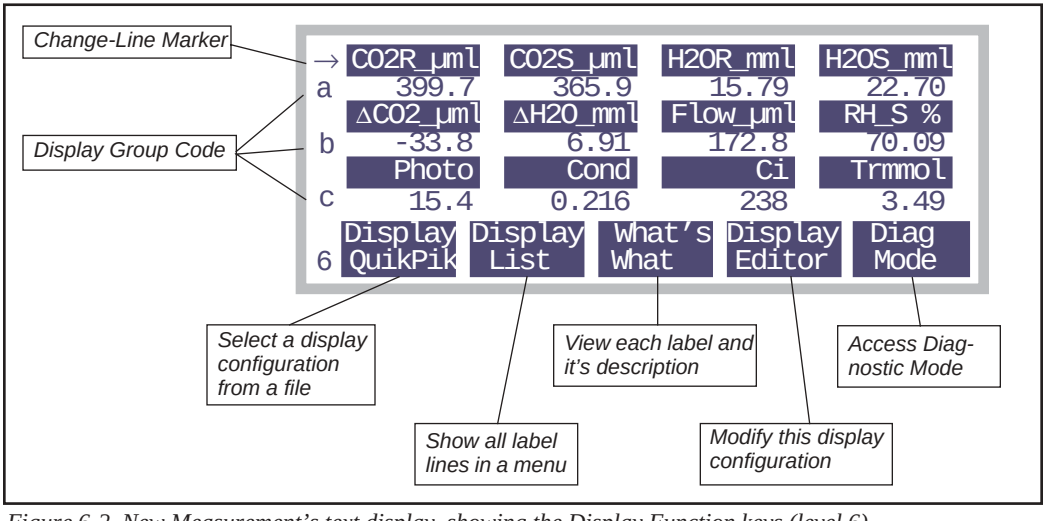


Figure 6-2. New Measurement's text display, showing the Display Function keys (level 6)

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 6-19. 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.

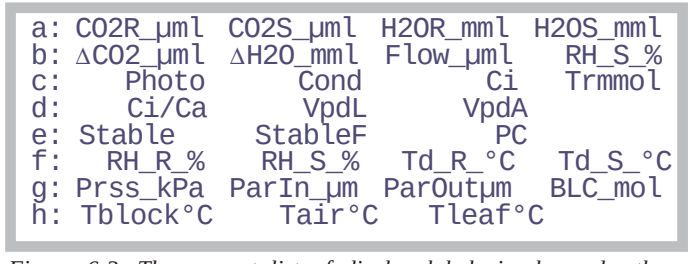


Figure 6-3. The current list of display labels is shown by the Display List *function,* **f2**, *level 6*

This is an instance of Standard Menu (page 5-2), 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.

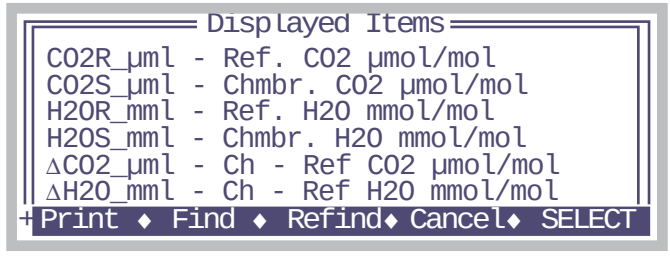


Figure 6-4. Descriptions of displayed items is shown by the What's What *function,* **f3**, *level 6*

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.

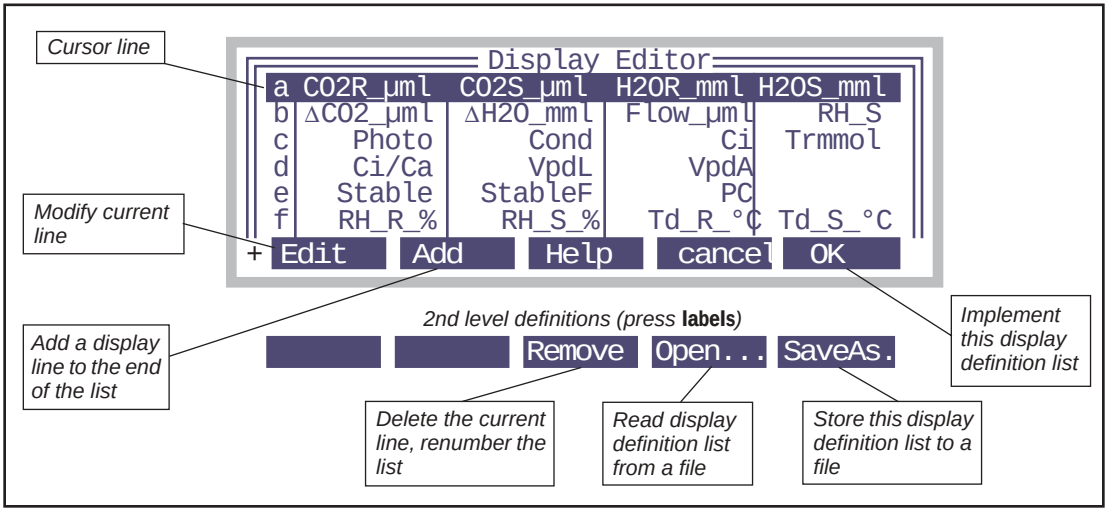


Figure 6-5. The Display Editor allows you to modify the current display list, store it, or retrieve a new one.

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** (or whatever the highest line label is).

To modify what appears on a line (**Edit**), or to add a new line (**Add**), you select each item to appear in the line from a menu (Figure 6-6). Usually (but not always) there is room for 4 items per line. If you want fewer items than what there is room for, press **escape** rather than selecting the next item.

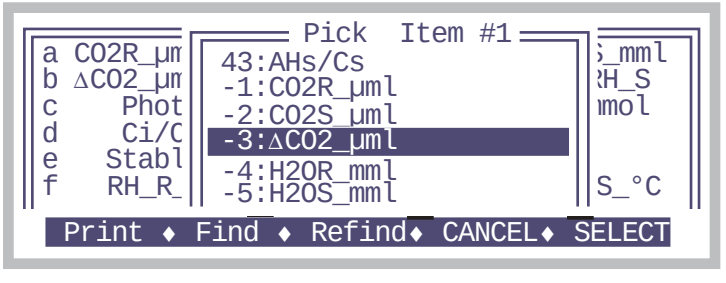


Figure 6-6. When Editing or Adding a display format line, you are prompted for each item in turn, selecting it from a menu of variables.

The structure of the menu used to select items in the display format list is illustrated in Figure 6-7. The system variables are a fixed set (**OPEN's System Variables** on page 14-1), while the user variables are determined by the current Compute List (**Defining User Variables** on page 15-1).

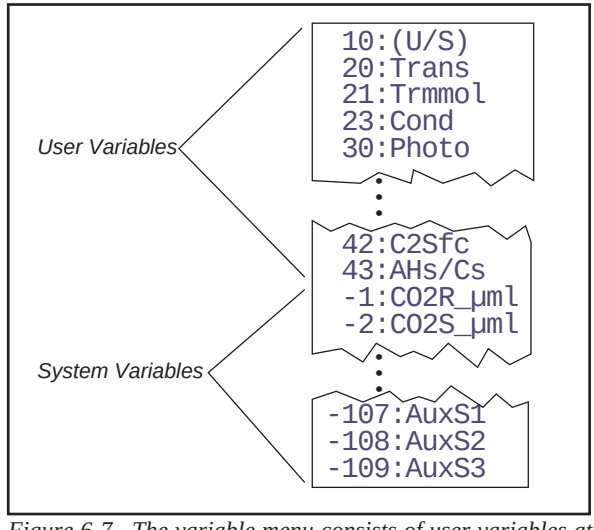


Figure 6-7. The variable menu consists of user variables at the top, followed by the system variables.

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**, **pagup**, **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**, **pagup**, **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.

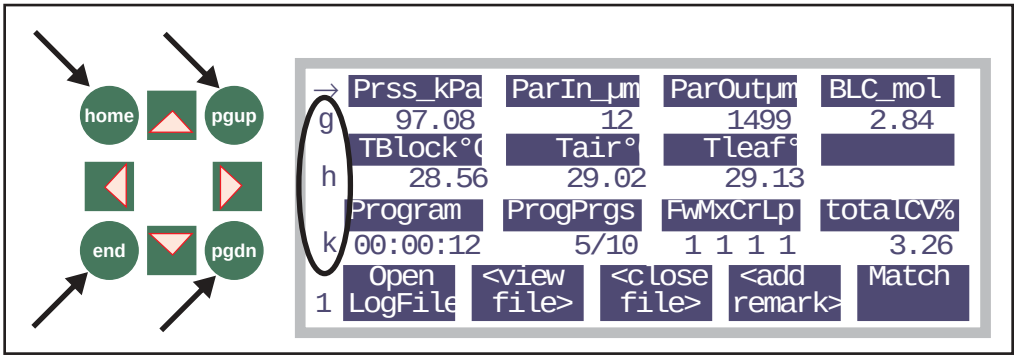


Figure 6-8. Define a display group by holding the **ctrl** *key down and pressing any of the 4 indicated group keys:* **home**, **pgup**, **pgdn**, *and* **end** *(on either side of the display). To change all three lines to a previously defined group, simply press that group key.*

Display Files

Display files are generally stored in the /User/Configs/Displays directory. The format of a display format file is illustrated in Figure 6-9.

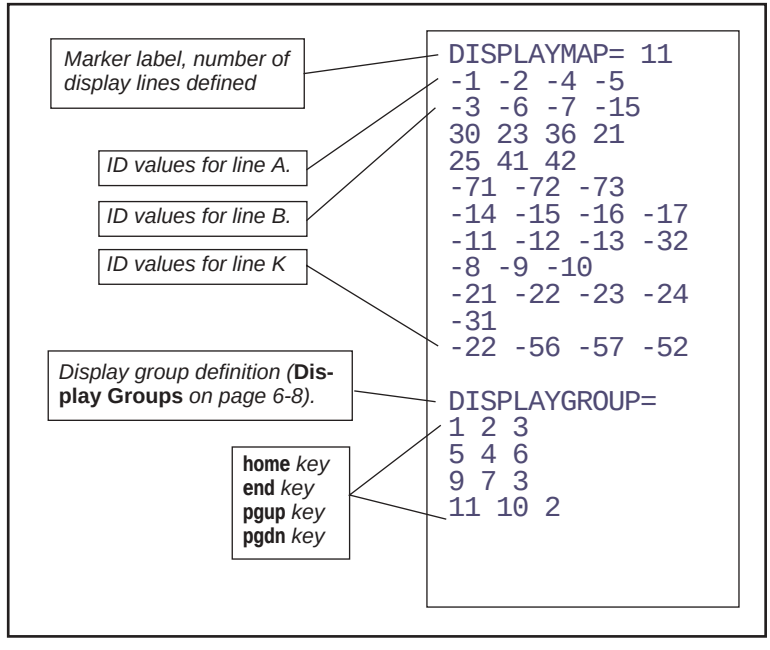


Figure 6-9. Display format files have a marker (DISPLAYMAP=), followed by the number of display lines to be defined. Following this are the ID values of each item. System variables have ID values <0, while user items have positive ID values.

Real Time Graphics

Real Time Graphics (RTG) provides a graphical method of monitoring recent activity in New Measurements mode (Figure 6-15). 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 6-10).

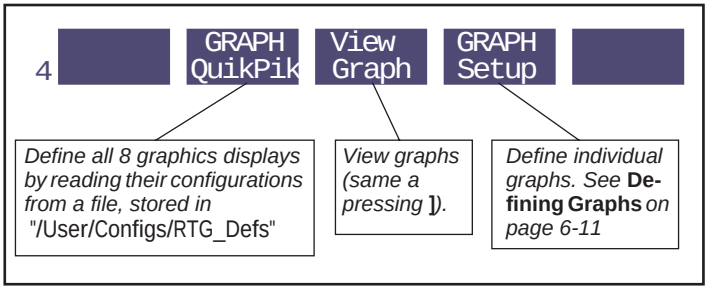


Figure 6-10. The graphics control function keys.

Defining Graphs

Graph Setup (**f4** level 4) brings up a list of the 8 RTG screens (Figure 6-11), providing an overview of how they are configured. From this screen, you can read or save an individual RTG configuration (**ReadOne** and **SaveOne**), or read or save them all as a group (**ReadAll**, **SaveAll**). (**ReadAll** does the same thing as **Graph QuikPik**, except it doesn't put you in graphics view mode.) Individual definitions (**ReadOne** and **SaveOne**) are stored in "/User/Configs/RTG_Defs/Graphs", while the group definitions (**ReadAll**, **SaveAll**, and **Graph QuikPik**) use "/User/Configs/RTG_Defs". File formats are described in **RTG Configuration Files** on page 6-17.

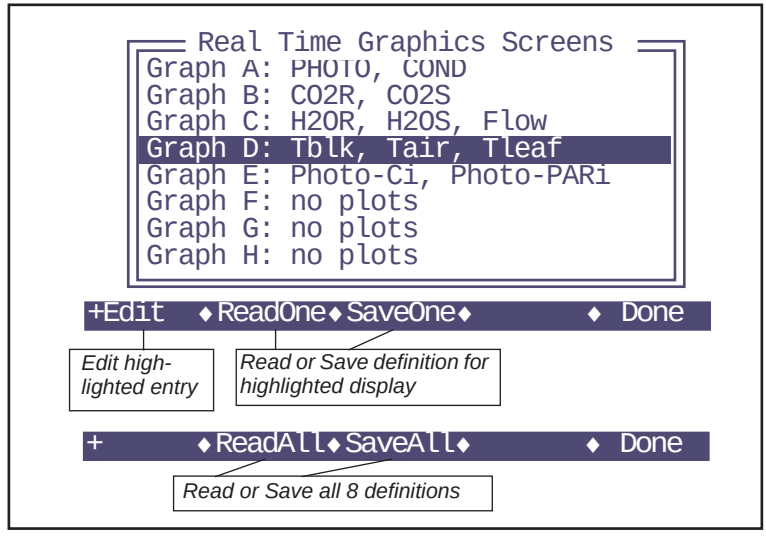


Figure 6-11. The Graph Setup Screen.

To edit a graph, highlight the desired entry and press **Edit** to bring up the Graph Editing Dialog (Figure 6-12).

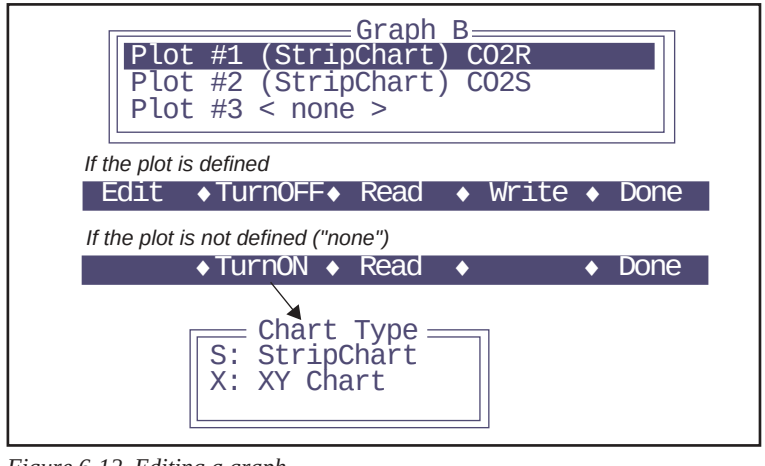


Figure 6-12. Editing a graph.

Up to three plots on a screen can be defined. Individual plot definitions can be stored or retrieved (**Read** and **Write**). **Edit** lets you adjust a particular definition (Figure 6-13). **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 6-15 on page 6-15), but you can specify how much data is to be retained. The default is 10 minutes.

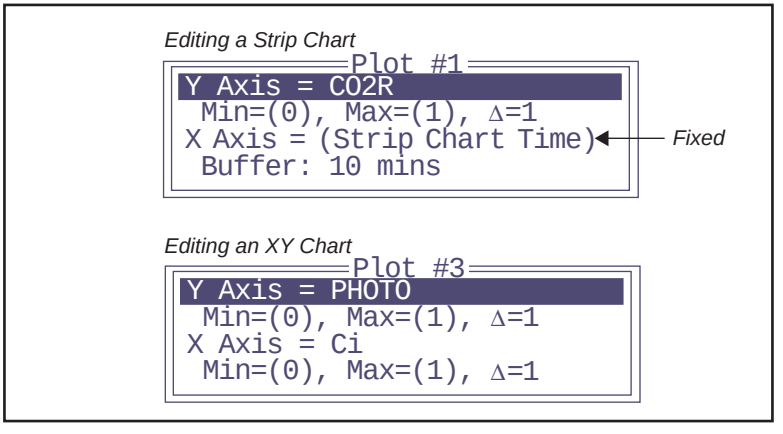


Figure 6-13. Editing a plot definition

There are five scaling options available (Figure 6-14):

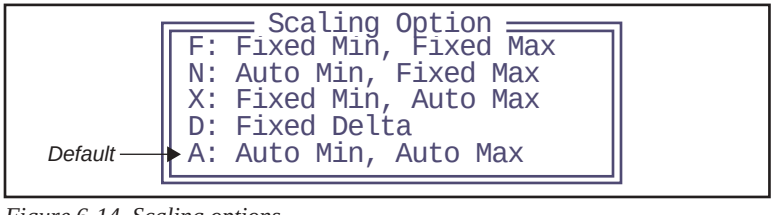


Figure 6-14. Scaling options

- **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, Δ= 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

Version 5 offers dynamic control of RTG displays via three levels of function keys (Figure 6-15).

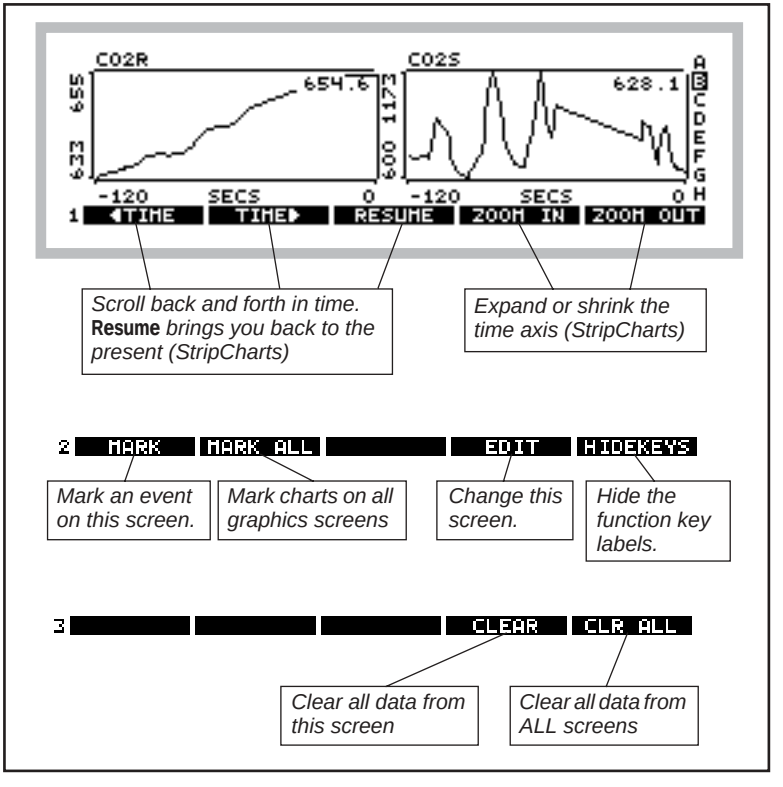


Figure 6-15. The Real Time Graphics function keys

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▶** bring 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 6-16). Pressing that key again will select the next plot, and so on, until none is selected.

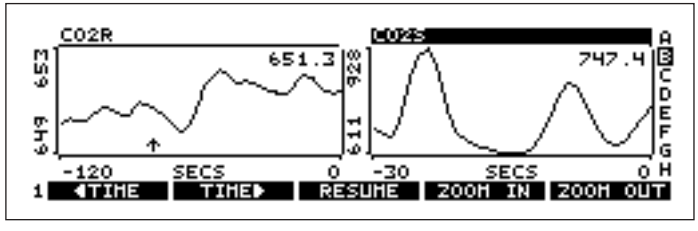


Figure 6-16. A selected plot has its title highlighted

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

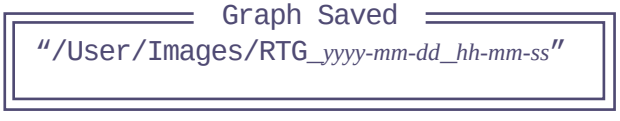
A graphics screen can be edited directly from graphics mode by pressing **EDIT** on level 2. This will access the editor shown in Figure 6-12 on page 6-12.

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 6-17 on page 6-18). The group definitions in "/User/Configs/RTG_Defs" are collections of eight <GRAPH>...</GRAPH> sets.

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**.



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 tranfer them to another computer, you should treat them as binary files, not text (**Text vs. Binary Files** on page 11-9).

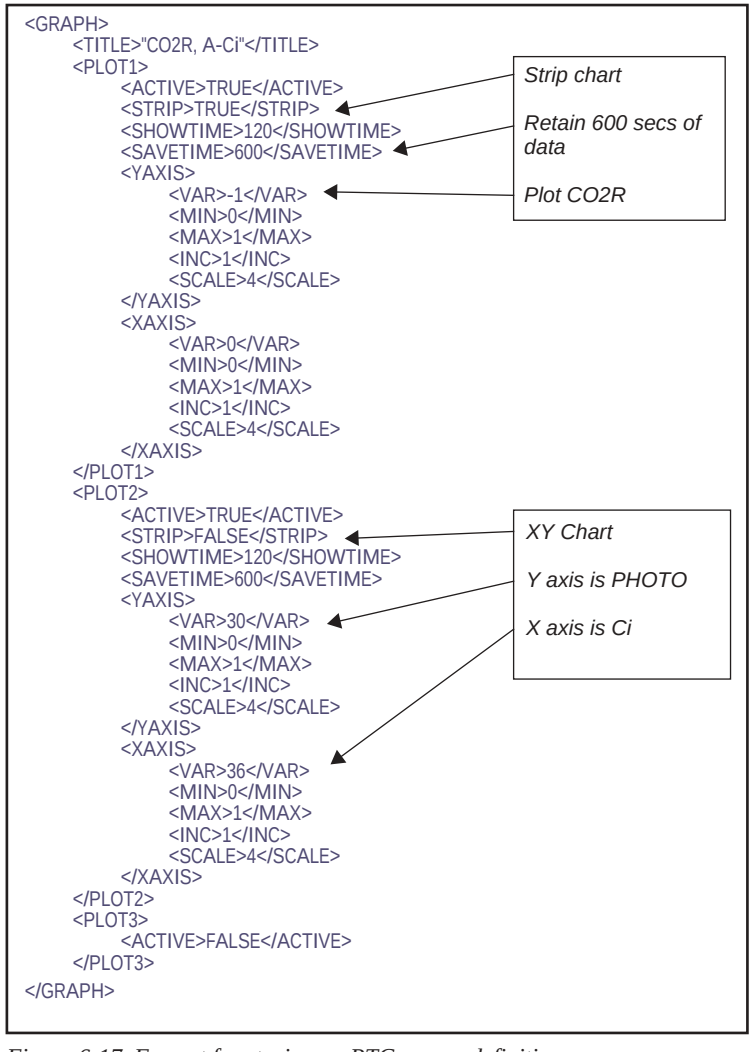


Figure 6-17. Format for storing an RTG screen definition.

Diagnostics

The third mode for viewing real time data in New Measurements node is Diagnostics Mode. Diagnostics mode is entered by pressing **[** from text or graphics mode, or by pressing **Diag Mode** (**f6** level 6 from text mode).

There are seven screens, labelled A through G.

A: Stability

This display (Figure 6-18) is probably the most useful during normal operations, as it shows the current statitics on the list of variables that you have in your stability list (described in **Stability Indicators** on page 4-40).

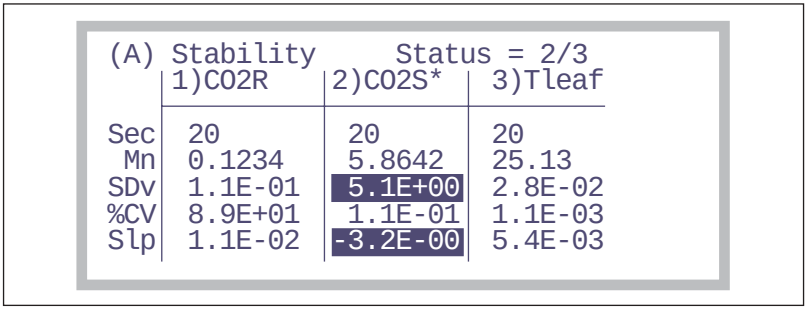


Figure 6-18. Diagnostic display A monitors the details of the stability computations. If there are more than 4 quantities in the definition, ← and → will scroll the columns, while **home** *and* **end** *jump to the left and right limits.*

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 6-19). There's not much here of interest unless you are troubleshooting.

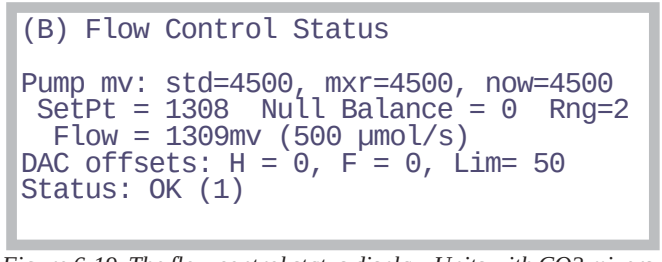


*Figure 6-19. The flow control status display. Units with CO2 mixers have two set points for the pump speed: one with the mixer (*mxr=*) and one when it's off (*std=*).* Now= *shows the current pump D/A output setting.* SetPt= *is the flow or humidity control set point. For fixed flow, this is the flow 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 6-20) shows if the mixer board is powered, and if the solenoid including the mixer in the flow circuit is activated.

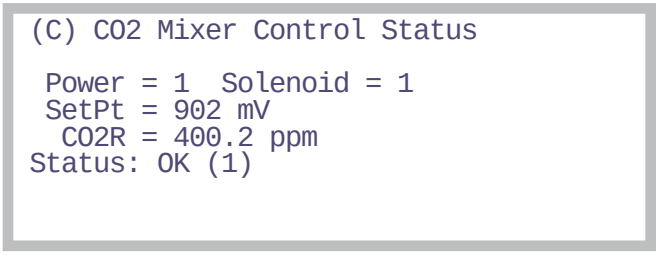


Figure 6-20. CO_2 Mixer Control Status screen. Power= should always be 1 (if a mixer is installed), but Solenoid= will be 1 only when actually doing CO_2 control. SetPt= *is the mixer's target, and* CO2R= *is the result.*

D: Temperature Control Status

The temperature control status screen is shown in Figure 6-21.

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 6-21. Temperature Control Status screen

E: Lamp Control Status

The Lamp Control status screen is shown in Figure 6-22.

With 6400-02(B) LED Source

```
(E) Lamp Control Status

Power = 1
SetPt = 3912 mV
ParIn = 1772 µmol/m2/s (-2954.0 mV)
  Cal = -.60   Offset = 0 mV
```

With 6400-40 Leaf Chamber Fluorometer

```
(E) LCF Actinic Status

Power  ResSetmV  BlueSetmV  FarSetmV
    1       307        312       -100
BlueCal=-.95  RedCal=-1.68  Cal=-1.57
ParIn = 497 µmol/m2/s (-317.0 mV)
%Blue= 9.2,   Offset = 0 mV
```

With no light source attached

```
(E) ParIn Status

ParIn = 650 µmol/m2/s (825.0 mV)
   Cal= 0.79,   Offset = 2 mV
```

Figure 6-22. 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 18-25). 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.*

F: IRGA Diagnostic Screen

The IRGA Diagnostic Display (Figure 6-23) shows raw millivolt signals, computed concentrations, and the temperatures and pressure that also go into the calculations. (Figure 6-23).

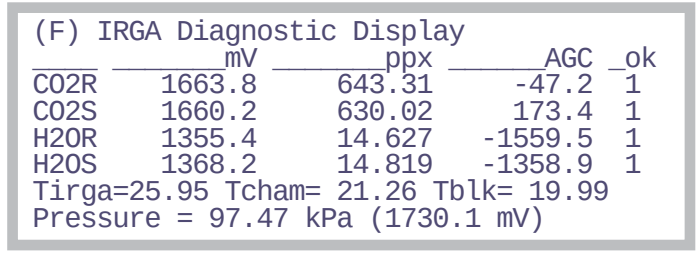


Figure 6-23. IRGA Diagnostic Screen. ppx means ppm or ppt - that is, µmol mol^{-1} for CO_2, or mmol mol^{-1} for H_20

This is a valuable screen for troubleshooting. See page 20-15.

G: System Diagnostic Screen

The System Diagnostic Screen is shown in Figure 6-24.

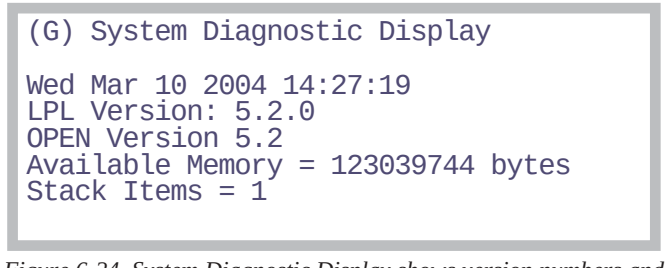


Figure 6-24. System Diagnostic Display shows version numbers and available memory.

Stability Indicators

Version 5 introduces a powerful technique for determining system stability. 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. (For computational details, see **Stability Variables** on page 14-11.)

Defining Stability

The Stability Dialog (Figure 6-25) 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 "<don't care>" is shown.

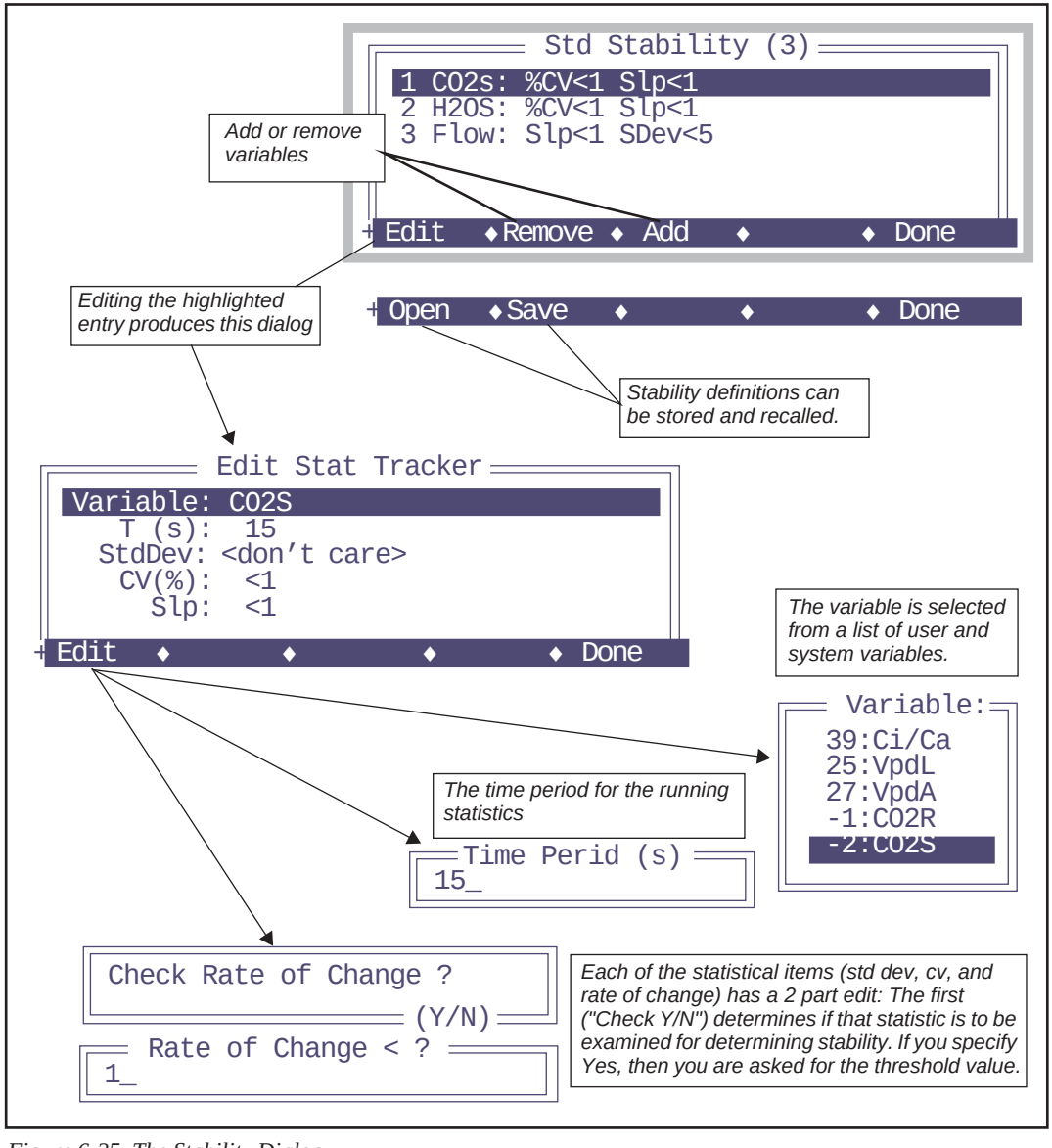


Figure 6-25. The Stability Dialog.

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 6-26).

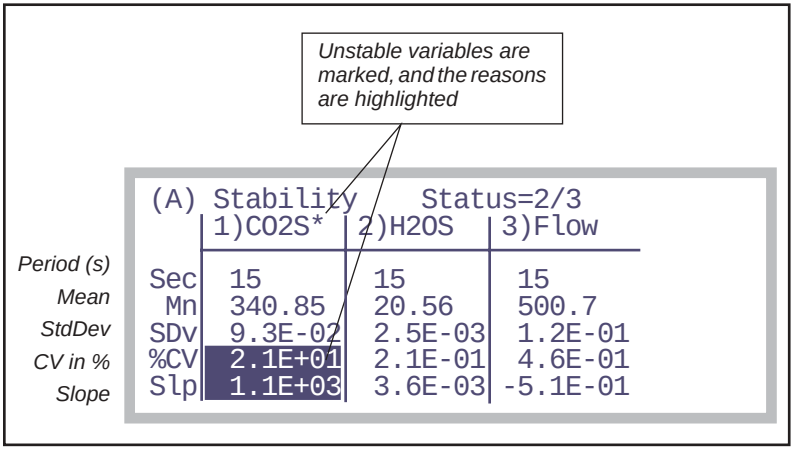


Figure 6-26. The stability diagnostic display.

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 ←.

Stability Definition Files

Stability definitions can be stored and retrieved from the file system, using the **Open** and **Save** function keys in the Stability Dialog. They normally live in the /User/Configs/StableDefs directory. These files' names can also be the "value" of the stability configuration command (**StableDefs=** on page 16-40).

A typical file is shown in Figure 6-27.

30 = Photo
20 seconds
Don't check %CV
Do check slope
Don't check std dev

23 = Cond
20 seconds
Don't check %CV
Do check slope
Don't check std dev

```
<STAT TRACKER>
      <TITLE>"Std Stability"</TITLE>
      <ITEM>
            <ID>30</ID>
            <SIZE>20</SIZE>
            <PCV_CHECK>FALSE</PCV_CHECK>
            <PCV_VALUE>1</PCV_VALUE>
            <SLP_CHECK>TRUE</SLP_CHECK>
            <SLP_VALUE>0.5</SLP_VALUE>
            <SDV_CHECK>FALSE</SDV_CHECK>
            <SDV_VALUE>1</SDV_VALUE>
      </ITEM>
      <ITEM>
            <ID>23</ID>
            <SIZE>20</SIZE>
            <PCV_CHECK>FALSE</PCV_CHECK>
            <PCV_VALUE>1</PCV_VALUE>
            <SLP_CHECK>TRUE</SLP_CHECK>
            <SLP_VALUE>0.01</SLP_VALUE>
            <SDV_CHECK>FALSE</SDV_CHECK>
            <SDV_VALUE>1</SDV_VALUE>
      </ITEM>
</STAT TRACKER>
```

Figure 6-27. Listing of a typical stability definition file.

Keyboard Summary

In addition to **Hot Keys** on page 5-16, the following control keys are available in New Measurements mode..

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.
0..9	(Text and Graphics) Selects function key level.
home	(Text) Implement a Display Group (page 6-8) key.
end	
pgup	
pgdn	
shift ctrl home	(Text) Define a Display Group key.
shift ctrl end	
shift ctrl pgup	
shift ctrl pgdn	
↑ ↓	(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 display to /User/Images

7 Environmental Control

How OPEN controls chamber conditions

7 Environmental Control

One of the things that makes the LI-6400 a powerful system is it's 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 3-29, 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) provide a means of controlling humidity (using flow rate), CO_2 (with optional 6400-01 CO_2 Mixer), temperature, and light (with optional 6400-02 or -02B LED Source, or 6400-40 LCF).

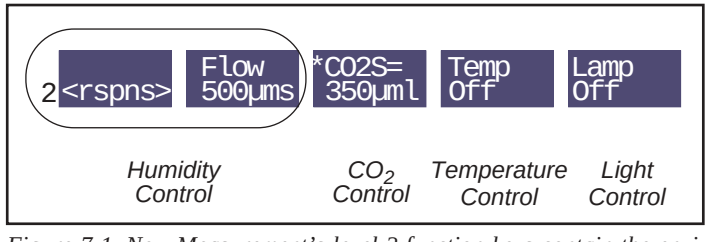


Figure 7-1. New Measurement's level 2 function keys contain the environmental controls.

Each of the four 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 menu of the control options, listed with present targets (if applicable) for each option. The flow/humidity key (**F2**), for example, brings up a 5-option menu (Figure 7-2).

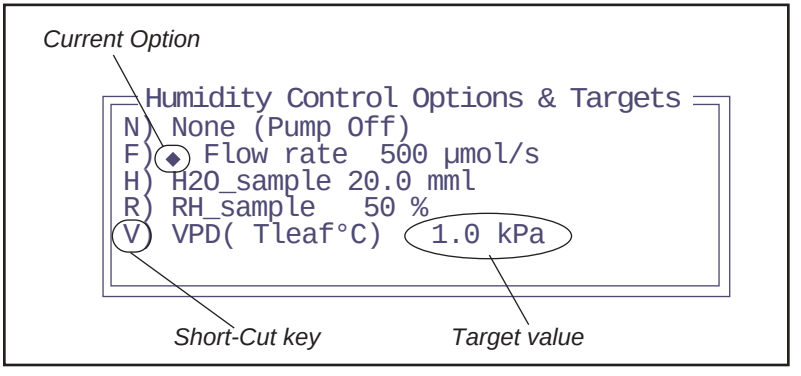


Figure 7-2. The elements of a typical control screen: each control option has a short cut key and its default target value; the currently selected option is marked with a solid diamond (♦).

Typing a numeric value (even before the option menu appears) automatically selects the current option and brings up the target value entry box, which allows the user to change the target value (Figure 7-3).

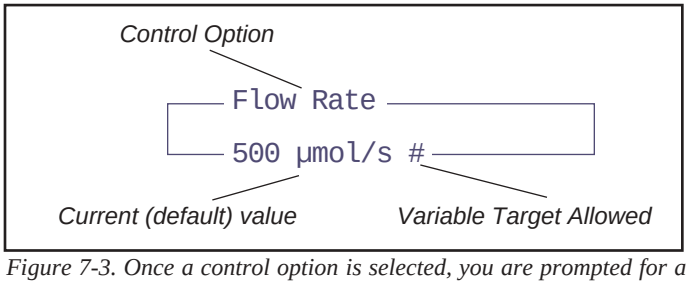


Figure 7-3. Once a control option is selected, you are prompted for a target value.

A blank entry (just press **enter**) will retain the current value. Variable targets are discussed on page 7-5.

Behind The Scenes

Once the user has specified a control and target, the control manager sets the appropriate hardware according to the control option and target. The system then begins a regime of periodically checking the stability status and fine tuning the controls 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. While the control manager is active, ISRs execute at regular intervals.

The LED light source, for example, 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 Manger

An inactive control manager means that the fine tuning necessary for active target tracking is not being done. 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 time periods. 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 temper-

[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 of the hardware circuitry controlling it.

ature. Specifics can be found in the ISR descriptions for each control described below.

Variable Targets

Any control mode whose target entry box includes a pound sign (#) after the default label can have a variable target, rather than 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
```

(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 provided by user channel #1001 named *xtrnRH*. Once this is defined appropriately in the Compute List File (Chapter 15), we can press **F2** to select the flow/humidity control, press **R** for RH_sample control, and type **#1001 enter** (Figure 7-4). Or, you could just enter **#** and pick *xtrnRH* from the menu.

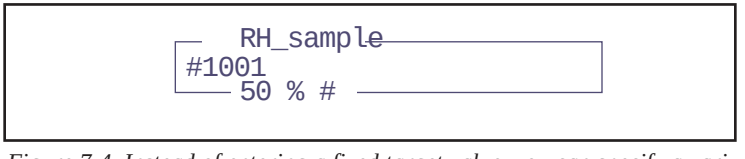


Figure 7-4. Instead of entering a fixed target value, you can specify a variable, system or user. The target value comes from this variable, and is updated every 30 seconds.

Then, the flow/humidity control function key will show the humidity target as the latest value of the variable we specified.

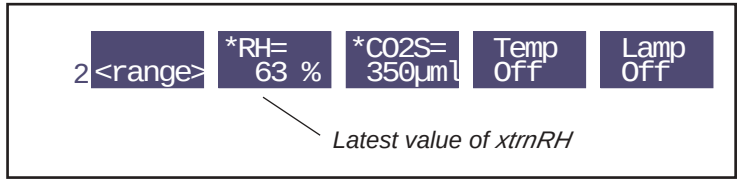


Figure 7-5. When a control has a variable target, the value is updated every 30 seconds. This value is shown on the control key label.

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. When the flow/humidity control function key is pressed again, the control menu will show the variable name and target value (Figure 7-6).

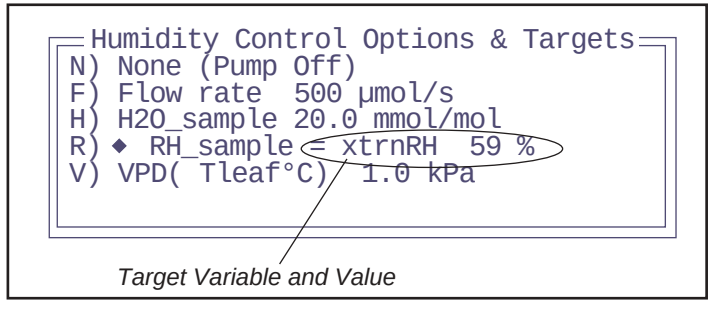


Figure 7-6. When a control has a variable target, the variable name and the current value are shown in the control screen.

and if we select it again, the target default will indicate the variable target (Figure 7-7).

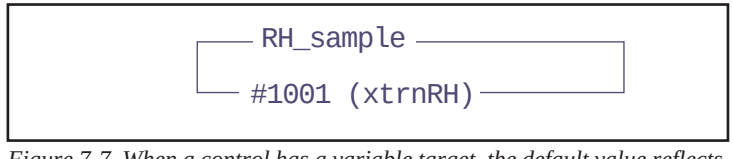


Figure 7-7. When a control has a variable target, the default value reflects the variables ID value, and label.

Pressing **enter** with no entry will retain the target variable. Entering a value will change back to fixed target mode.

Variable targets are a powerful tool, but do have a possible limitation when combined with AutoPrograms. See **AutoPrograms and the Control Manager** on page 25-22.

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 desiccant is set by the adjust 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 4-51).

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 7-8).

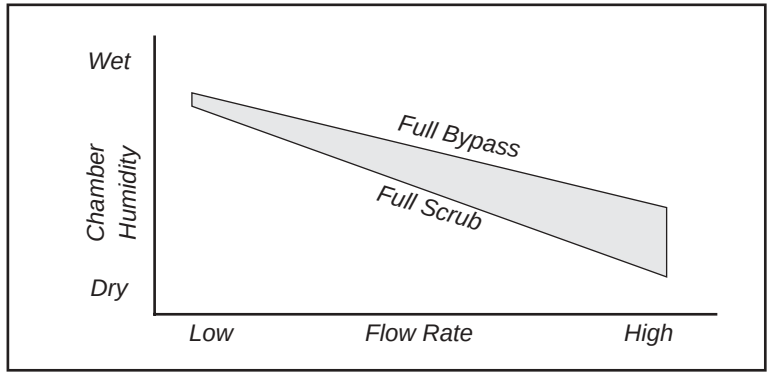


Figure 7-8. The operating envelope is determined by the transpiration from the leaf. To maintain any given humidity, a combination of flow rate and desiccant knob settings might achieve it. Typically, the knob remains fixed, and the flow is adjusted automatically.

Control Options

When **f2** in level 2 is pressed in New Measurements Mode, the following control options are presented:

```
Humidity Control Options
N) None (Pump Off)
F) Flow rate  500 µmol/s
H) H2O_sample 20.0 mml
R) RH_sample   50 %
V) VPD( Tleaf°C)  1.0 kPa
```

Figure 7-9. The humidity control screen presents one fixed flow option, and several constant humidity options.

Table 7-1. Flow/Humidity control menu options

Option	Description
N) None	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.

Table 7-1. (Continued)Flow/Humidity control menu options

Option	Description
F) Flow rate	Maintains a fixed flow rate through the chamber. This is a good default option.
H) H2O_sample	Maintains a constant water mole fraction in the sample cell. This is useful during response curves for light or CO_2.
R) RH_sample	Maintains a constant relative humidity in the sample cell.
V) VPD	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 F option (fixed flow rate), at a mid range flow rate (400 µmol 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't get the humidity high enough at 500 µmol 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 make the differentials that you are trying to measure (CO_2 and H_2O) small. 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 µmol s^{-1} (without a CO_2 mixer installed) or 50 µmol s^{-1} (with a mixer installed).

Constant Humidity Operation

The H, R, and V options *actively regulate* the flow rate to maintain a constant humidity (either 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 and/or low transpiration rates, or an empty leaf chamber, can make these 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 F option (fixed flow rate)**
Manually find the flow rate / desiccant tube setting that provides the desired humidity or vapor pressure deficit.

2 **Switch over to the H, R, or V 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.

H) Constant 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 signal is also a function of pressure, temperature, and how the IRGA is zeroed and matched, then there needs to be periodic updates of this target signal, even when the target mole fraction doesn't change.

R) Constant RH

The constant RH control is a simple extension of the H mode. The actual control is the same, it's just that more frequent updates are needed, since one more variable is now in the mix: temperature. If temperatures are changing rapidly (such as when the temperature controllers are operating at full capacity), 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).

V) Constant VPD

VPD stands for vapor pressure deficit, which is the difference in vapor pressure between what could be *there* versus what is actually in the air. Where is *there*? Well, you have at least two options: the sample cell IRGA, or the substomatal cavity of the leaf. In the case of the former,

$$VPD = e(T_a) - e_s \qquad (7\text{-}1)$$

and in the case of the latter,

$$VPD = e(T_l) - e_s \qquad (7\text{-}2)$$

where $e(T)$ is the saturation vapor pressure function (Equation (14-22) on page 14-10), T_a is sample cell air temperature, T_l is leaf temperature, and e_s is vapor pressure in the sample cell (Equation (14-19) on page 14-10). 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, including incident radiation, transpiration rate, etc.

When you select the V option for vapor pressure deficit control, you are asked for a target value, followed by a temperature (Figure 7-10).

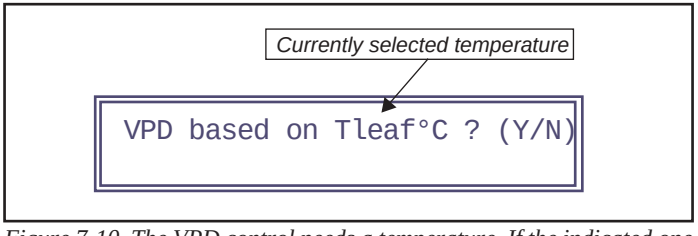


Figure 7-10. The VPD control needs a temperature. If the indicated one is what you want, press **Y**. *Otherwise, press* **N** *to change it.*

If you press **N** to change the temperature, you are shown a menu of all user defined variables, and two system variables: *Tleaf°C* and *Tair°C*. (Figure 7-11).

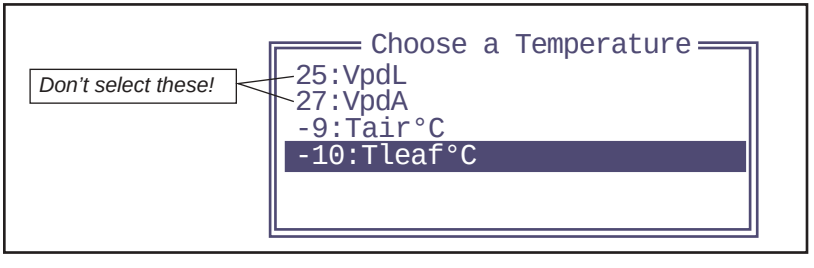


Figure 7-11. When asked to select a temperature for the VPD calculation, be sure what you select is a temperature. Don't select a VPD! The system does its own VPD computation based on the value of the (temperature) variable you select.

The reason the menu in Figure 7-11 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. Remember, you not selecting a VPD value, you are selecting a *temperature* for the system to use for its own VPD calculation.

The RSPNS Key

When one of the constant humidity options (H, R, or V) is active, the **RSPNS** (response) key (**f1** level 2) is also active.

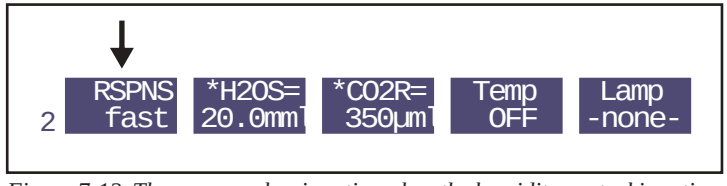


Figure 7-12. The response key is active when the humidity control is active. The choices are fast, med, *and* slow. Fast *is the default.*

The label will cycle between *fast*, *med*, and *slow* as you press the **RSPNS** key. This determines the rapidity of response of the control circuit (see Figure 3-38 on page 3-41). 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 the lock in rapidly on the target humidity as you go from leaf to leaf.The *med* 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'd like stable flow readings and stable humidities, and aren't 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.

The **RSPNS** key applies ONLY to the constant humidity control circuit.

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:

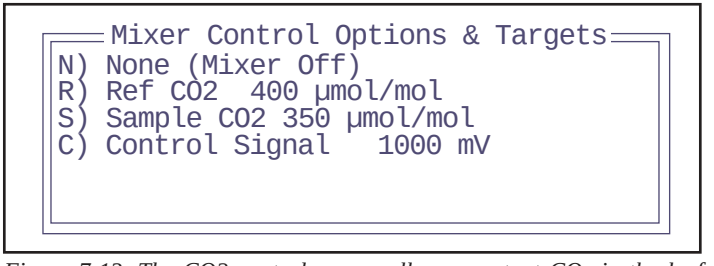


Figure 7-13. The CO2 control screen allows constant CO_2 in the leaf chamber, or a constant incoming CO_2.

Table 7-2. CO_2 control options

Option	Description
N) None	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.)
R) Ref CO2	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.
S) Sample CO2	Maintains a constant CO_2 concentration in the sample cell and chamber.
C) Control signal	The mixer control signal is set and held a target value. This is a diagnostic tool.

Constant Reference Option

Usually the preferred option is R (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,[2] so maintaining it is usually no problem. If it does drift, the software will attempt to bring it back on target.

Constant Sample Option

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 with the S option:

1 **Start out with the R option**

Specify a target 20 or 30 µmol mol^{-1} *above* what you want in the chamber, and wait for the system to stabilize.

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 looses control, you can always drop back to the R option and bring things back under control.

Unlike the constant humidity control options, which depend on the leaf providing lots of water, the S) 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[3]).

---

[2] Temperature and pressure changes, bad soda lime, or the soda lime tube not on full scrub.

[3] 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.

Interaction with Humidity Control

There is 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 S option) to sit back and wait until things stabilize, before attempting adjustments.

The result is that the S CO_2 option will work with the H, R, or V humidity options, but expect longer system equilibration times.

Constant Control Signal Option

The C) 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, but you'll 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, flow rate changes, long term drift of the controller, etc.).

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 "_CO2 Mixer - Plot curve" from the Calib Menu. A typical plot is shown in Figure 18-13 on page 18-24.

The CO_2 control software uses this calibration to come up with a first guess when you've specified some target CO_2 concentration. If you find (when operating in constant reference concentration mode) that the first guesses don't seem very good, you can generate a new set of calibration points for it to use, described in **6400-01 CO2 Mixer** on page 18-21.

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:

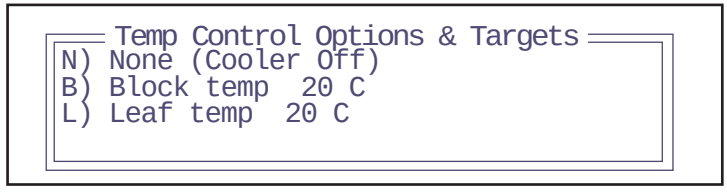


Figure 7-14. The temperature control screen provides for block or leaf temperature control

Table 7-3. Temperature control options

Option	Description
N) None	Turns off the chamber coolers.
B) Block Temp	Maintains a constant block temperature. Good default option.
L) Leaf Temp	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 changing 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[4].

Condensation?

It is possible (and 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 20-7.

[4] For example, using the leaf temperature thermocouple to measuring air temperature and calculating leaf temperature from an energy balance.

Light Control

This control is available only when the system is configured for an LED source. See **The Light Source Control Utility** on page 8-4.

When **f5** (level 2) is pressed in New Measurements Mode, the following control options are presented:

With the 6400-02 or -02B

```
Lamp Control Options & Targets
N) None (Lamp Off)
Q) Quantum flux  500 µmol/m2/s
C) Control signal  1000 mV
T) Track ParOutµm
```

With the 6400-40 LCF

```
Lamp Options & Targets
N) None (Lamp Off)
P) PQuantum 500 µm (10% Blue)
F) FQuantum 500 µm (100 Blue)
C) Control mV R=0, B=1000
T) Track ParOutµm
```

Figure 7-15. The lamp control screen provides constant quantum flux, or a tracking option.

Table 7-4. Light control options for the 6400-02 or -02B

Option	Description
N) None	Lamp off
Q) Quantum Flux	Maintains a constant quantum flux within the chamber, as measured by the lamp's light sensor.
C) Control signal	Set light to fixed control value.
T) Track ParOutµm	Tracks external quantum sensor with a 3 second update rate.

Table 7-5. Light control options for the 6400-40 LCF

Option	Description
N) None	Turns off red and blue actinic LEDs
P) PQuantum	Proportional quantum. Two target values: Total PAR (red + blue) in µmol m^{-2} s^{-1}, and proportion that should be blue, in%.  Keep in mind the maximum blue intensity is 150 or 200 µmol m^{-2} s^{-1}, so may not achieve the requested percentage.
F) FQuantum	Fixed quantum. Two target values: Total (red + blue) µmol m^{-2} s^{-1}, and blue µmol m^{-2} s^{-1}. 
C) Control signal	Two target values, red (mV) and blue (mV) 
T) Track ParOutµm	Tracks external quantum sensor (or other channel). Every 3 seconds, a new target is determined. You are prompted for the %Blue faction to use, and which channel to track.

6400-02 Options

Quantum Flux

This is the usual option for controlling the a 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 ParOut

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.

6400-40 Options

See **The LCF as a Light Source** on page 27-19.

8 Light Sensor Considerations

The most important parameter is the hardest to measure

8 Light Sensor Considerations

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.

---

[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.

There is a third type of light sensor that is used in the LI-6400. It is a silicon diode that monitors and controls the optional 6400-02 or -02B LED Light Source. 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:

- **LED 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 (14-15) through (14-17), starting on page 14-8.

The Light Source Control Utility

The Config Menu contains a "Light Source Control" entry, which brings up a screen (Figure 8-1) showing the currently configured light source and related constants.

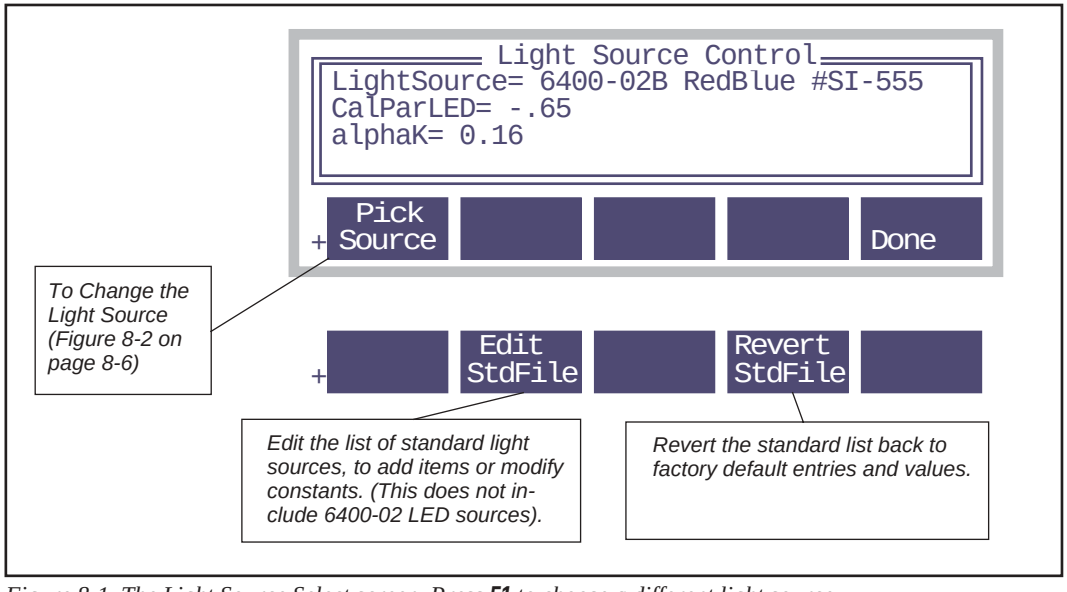


*Figure 8-1. The Light Source Select screen. Press **F1** to choose a different light source.*

6400-02s

When the light source is a 6400-02 or -02B (as in Figure 8-1), the related items shown are *CalParLED* (the calibration value of the light source), and *alphaK* (used only for energy balance computations to convert µmol m^{-2} s^{-1} to W m^{-2}, as described in Chapter 17.)

6400-40 LCFs

If you select the 6400-40 as a light source, you will be asked

If you press **L**, then the New Measurements function keys allowing control of the fluorescence related parts of the LCF will not be shown, and you can use the LCF simply as a red-blue light source. You will have independent control over the red and blue.

When the light source is a 6400-40 LCF, the related items shown are the serial number and calibration factors (red, blue, and fluorescence zero offset) it read from the unit, and the standard *alphaK* value.

```
LightSource= 6400-40 Fluorometer
"LCF-0116" Cals= -1.98, -2.05, -1986
alphaK = 0.16
```

All Others

When the light source is something other than a 6400-02 or -02B, the related items are *alphaK*, *CalParGaAs* (the calibration of the light sensor in the chamber top being used), and *actinity* (a correction factor for spectral variations from solar).

```
LightSource= Sun+Sky
CalParGaAs = 0.82
actinity = 1
alphaK = 0.19
```

Pick Source

When you press **Pick Source** (**F1**) in the Light Source Control screen, the LightSource Selection Menu (Figure 8-2) will appear.

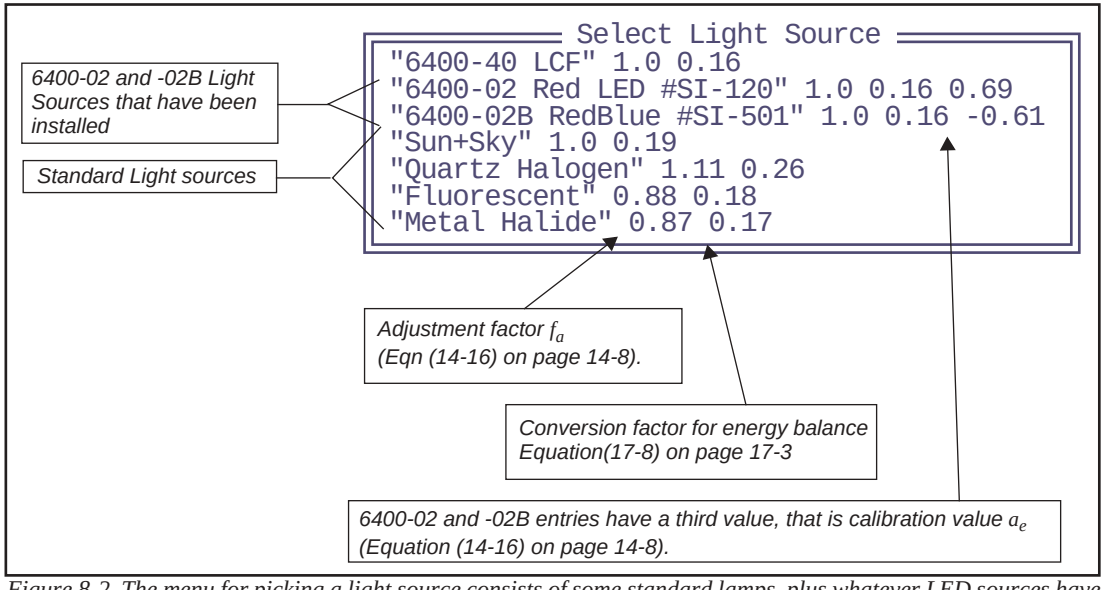


Figure 8-2. The menu for picking a light source consists of some standard lamps, plus whatever LED sources have been installed. If an 6400-40 fluorometer has been installed, there will be an entry for that as well.

Edit StdFile.

This option in the Light Source Control screen opens the file /User/Configs/LightSources for editing. You would do this if you wished to add your own entries, or modify existing entries. Note that this is NOT the method of modifying the LED source entries. They are added via the Installation Menu.

Revert StdFile

The revert file option changes the file /User/Configs/LightSources back to it's default state. The default version of this file is stored in /Sys/Lib/LightSources.

6400-02 and -02B Light Sources

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 8-3). 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 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 18-26.

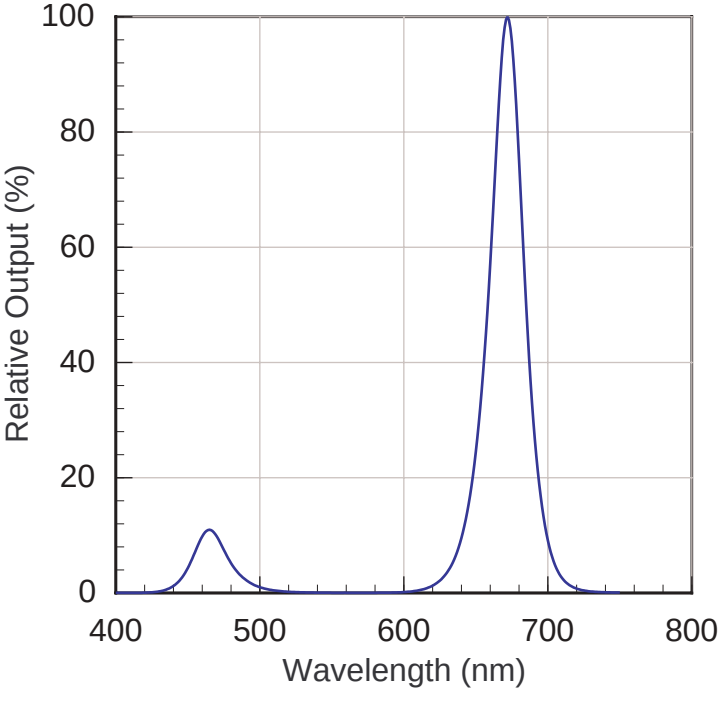


Figure 8-3. Typical output of a 6400-02B LED source at 25°C. (A 6400-02 source has the same red peak, but no blue peak.)

[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.

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 continuously, 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.

6400-40 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 8-4;

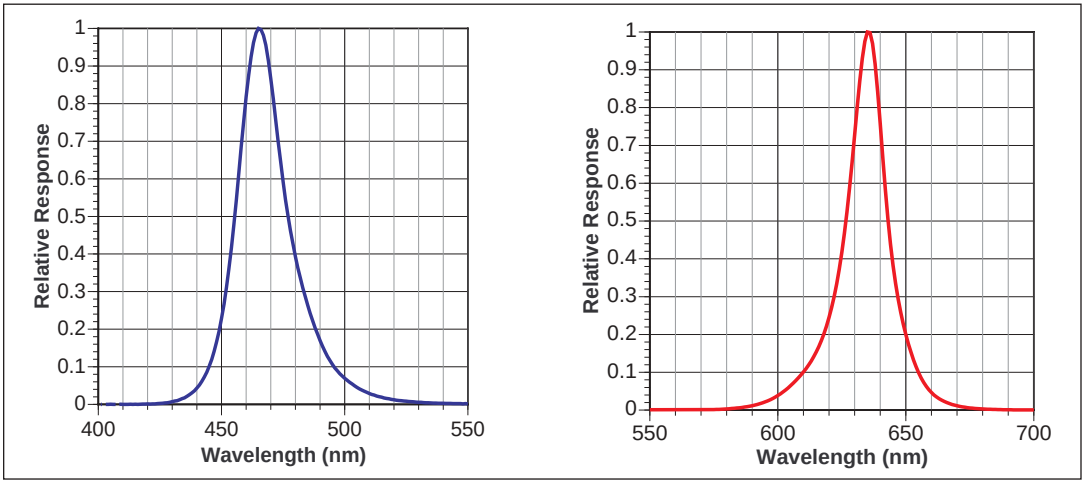


Figure 8-4. Relative spectral outputs of the red and blue LEDs used in the 6400-40 Leaf Chamber Fluorometer.

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 is available to the software from the current LED calibration curves (generated by the user doing is calibration menu item described in **"Actinic Control - Calibrate"** on page 27-61). 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 8-5 on page 8-10, 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 18-30).

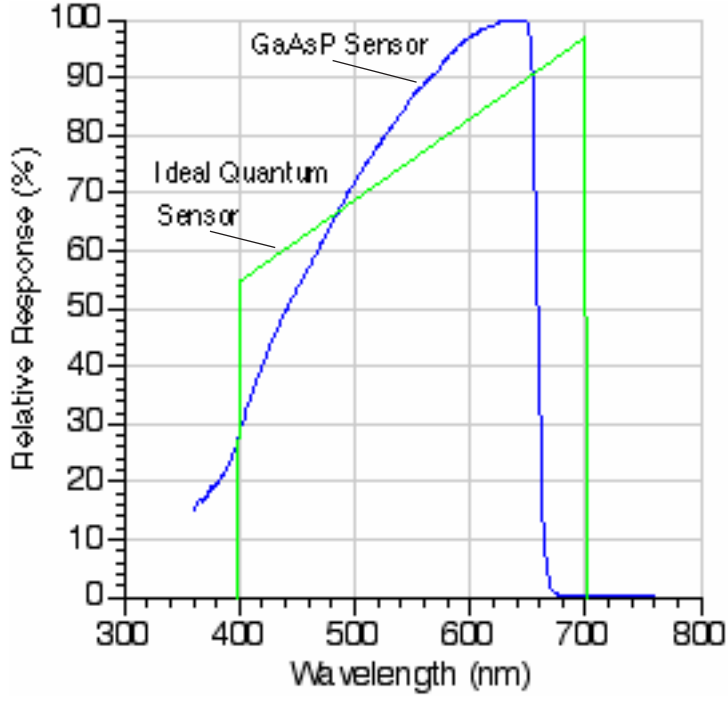


Figure 8-5. Spectral response of GaAsP sensor and ideal quantum response.

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. There 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.

9 Data Logging

Storing 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 a introduction to this topic, see **Tour #4: Logging Data** on page 3-50

Basic Concepts

The LI-6400, in its default configuration, is an open system. As such, it shows a continuous stream of measurements and calculations. You can monitor light levels and photosynthetic rates, for example, while changing those light levels. Over the course of, say, 5 minutes, the LI-6400 will have made hundreds of measurements. The question is, which of these will you want to retain for future use?

Person A might only want two sets of data: one when conditions were stable at light level X, and another 5 minutes later at light level Y. Person B, however, might be studying the dynamics of the change from X to Y, and want to record 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 port in real time. Files can be 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 instant in time. A file can contain 1 or 100 or more such observations. Or no observations. The observations can be irregularly spaced, recorded manually and whimsically, or evenly spaced, recorded automatically based on time. They can even be recorded automatically when the program determines that stability has been achieved.

What does it look like?

A sample file with logged data is shown in Figure 9-1, in very small print.

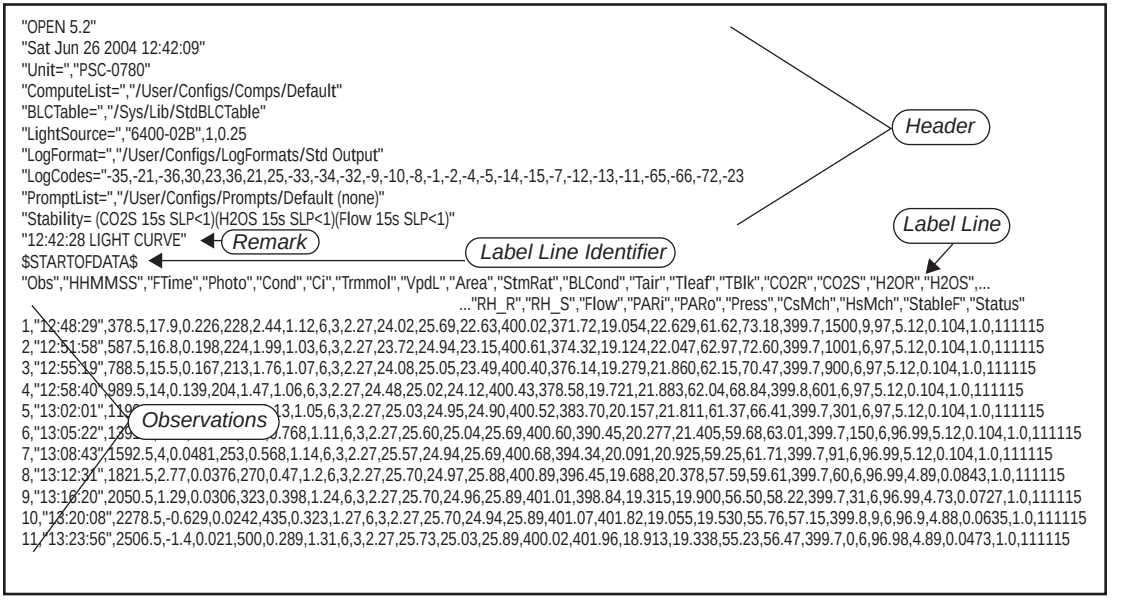


Figure 9-1. Sample data file, in comma delimited format.

At the start of the file is the header. The header consists of a series of lines identifying some configurations associated with the data. The data consist of a line of labels identifying each column, and rows of observations. The delimiter in these files can be commas, as in Figure 9-1, or tabs, or spaced into justified columns. This latter option makes the file more readable for humans, but at the price of using more memory for storage. Note that regardless of how the file is stored, GraphIt (described in Chapter 12) allows the data to be viewed in columns.

Getting Started

Logging is done in New Measurements mode, and is initiated with the function keys in level 1.

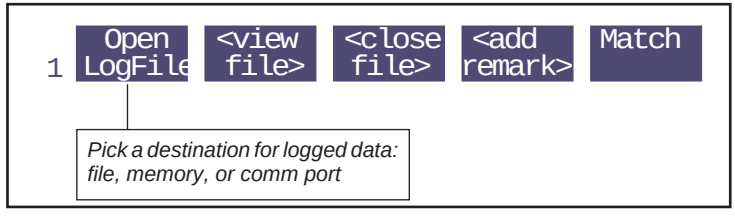


Figure 9-2. Logging is initiated and controlled from New Measurements' function key level 1.

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).

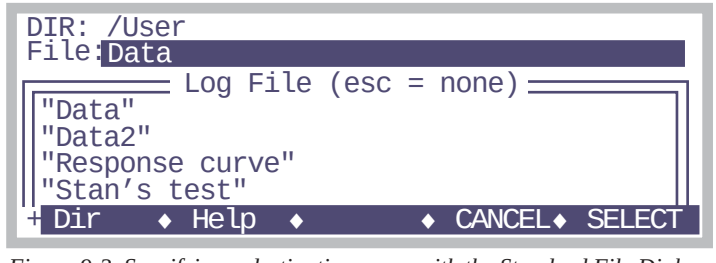


Figure 9-3. Specifying a destination name with the Standard File Dialog. If you wish to log to the Comm port, just press **CANCEL** *or* **escape**.

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.

Figure 9-4. The alternate destination prompt. Press **Y** *for Comm port, or* **escape** *to not log anywhere.*

If you wish to log to a file and to the Comm port simultaneously, use the "Echo logged data" feature (Utility Menu -> Configure the Comm Port. See Figure 11-2 on page 11-4).

Logging Remarks

Once the destination is established, you'll be shown the box for entering remarks (Figure 9-5). These initial remarks are recorded on the line just prior to the label line. See Figure 9-1 on page 9-3.

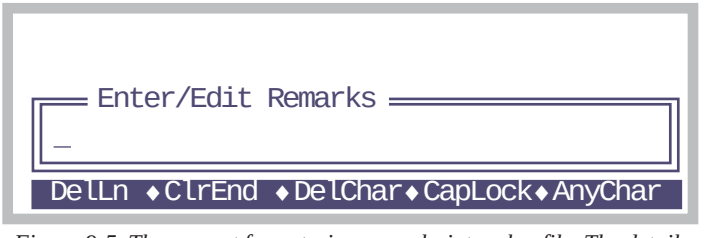


Figure 9-5. The prompt for entering remarks into a log file. The details of this type of dialog box are described in ***Standard Line Editor*** *on page 5-5.*

Remarks can also be entered at any time the file is open, by pressing **Add_Remark** (**F4** level 1), and will take the form of a quoted string on it's own line in the file. The start of a remarks line contains the time (HH:MM:SS) the remark was entered (Figure 9-6).

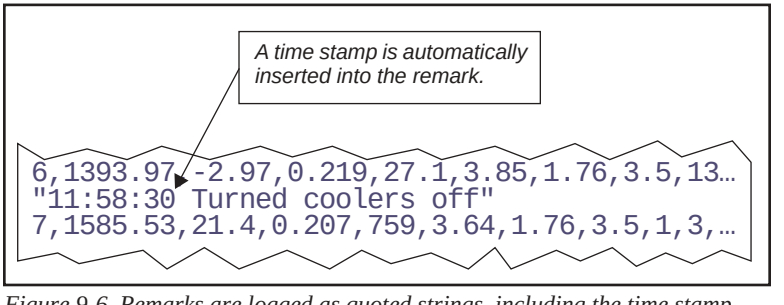


Figure 9-6. Remarks are logged as quoted strings, including the time stamp.

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 9-15.

Logging Observations

Once the destination and initial remarks are established, the function keys (Figure 9-7) will show (in the label for **f1**) the number of observations logged.

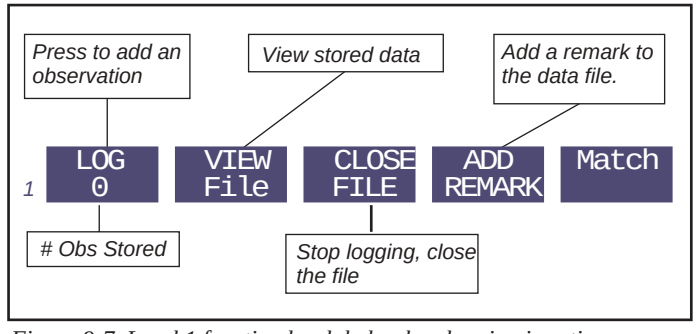


Figure 9-7. Level 1 function key labels when logging is active.

Observations can be added to the logged data by pressing the Log Key or the Log Button. The log button is located on the left hand side of the sensor head handle. Press and hold the log button until an audible tone sounds (within 1 second), 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. Press **Define Log Btn** (**f5** level 5) to bring up a menu of possibilities (Figure 9-8).

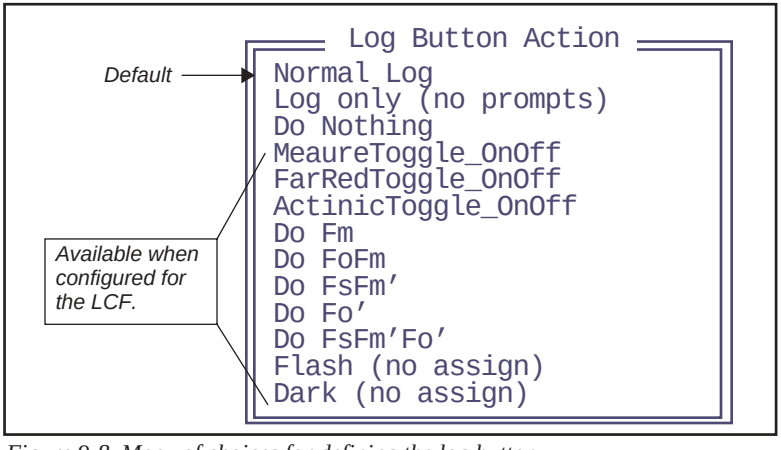


Figure 9-8. Menu of choices for defining the log button.

The difference between "Normal Log" and "Log only" is that user prompts (described on **Prompts and Remarks** on page 9-15) are bypassed in the latter case.

The fluorescence options are described in Table 27-20 on page 27-73, but essentially correspond to many of the LCF control function keys. If prompts are set to "On Log" (see Figure 9-16 on page 9-15), then the fluorescence commands that include logging ("Do..") will also trigger prompts, when executed via the log button.

Determining What is Logged

The quantities logged and their arrangement are determined by the LogList, which is a list of system and user ID numbers to be logged. Logging Control (in the Config Menu) provides an editor for making changes to this list.

Logging Control

The Logging Control screen (Figure 9-9) shows the format file in use (the LogFormat= configuration command), and some status information.

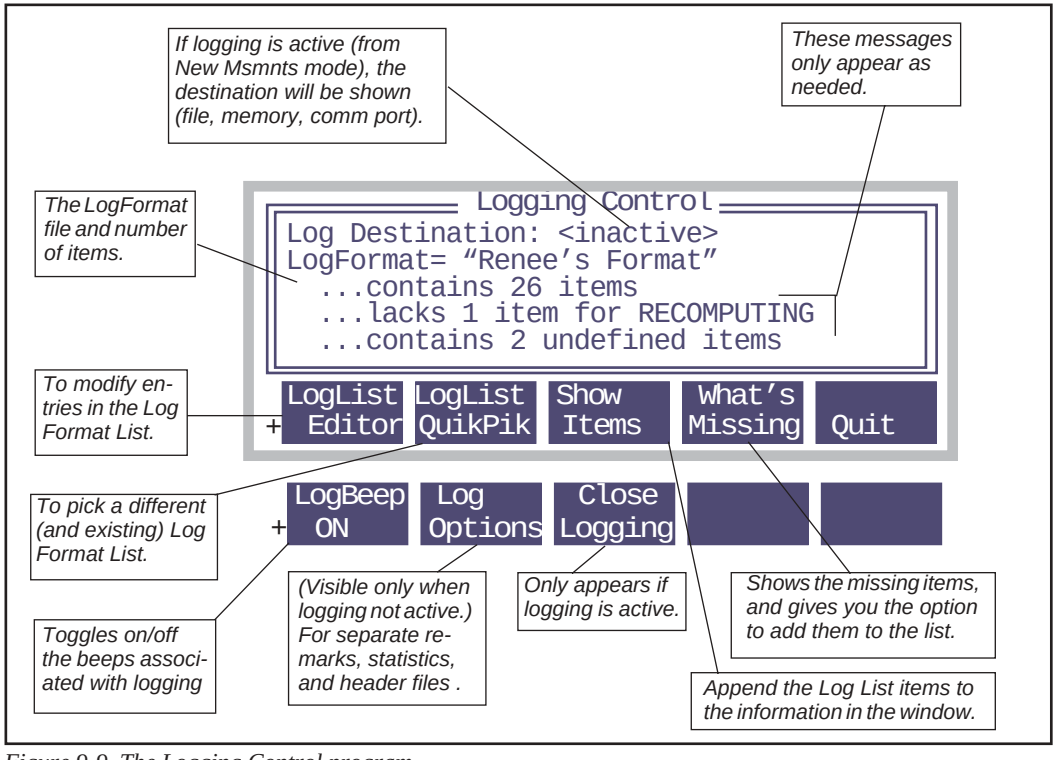


Figure 9-9. The Logging Control program.

Status information can include:

…contains <n> items
The number of items in the Log Format list.

…lacks <n> items for RECOMPUTE
Items that would be needed to recompute the data file are missing. When a Compute List is compiled, OPEN makes a list of all the system variables that you've referenced (directly or indirectly). Logging Control looks for these items in your log list. If you have some missing, it shouldn't concern you unless you want to recompute later.

…lacks <n> PromptList items

Items in the PromptList are not being logged. To remedy either of the "...lacks" messages, press **F4** (**What's Missing**) to see a list of the offending items (with an option to insert them into the log list), or else use the LogList Editor. Or QuikPik another Log List.

…contains <n> undefined items

These will be labelled as "???". To take care of undefined items, either switch to the appropriate ComputeList that defines these items, or else use the LogList Editor to remove the items.

Log Options

The **Log Options** key (visible only when logging is not active) brings up a menu of options (Figure 9-10). (This is also accessible from New Measurements mode, level 5.)

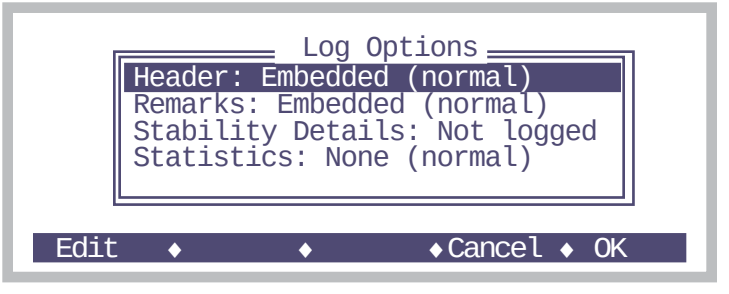


Figure 9-10. The Log Options menu.

Use the **Edit** key to change any of the settings. **OK** keeps them (they apply to subsequent data files), and **Cancel** undoes any changes.

Header

The choice is "Embedded", or "Separate .HDR file". If you select a separate .HDR file, then your data file will not have any of the normal header information above the line labels. This will instead be put into separate file (Figure 9-11).

"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 5.2"
"Fri Jun  4 2004 10:08:46"
"Unit=","PSC-0780"
"ComputeList=","/User/Configs/Comps/Default"
"BLCTable=","/Sys/Lib/StdBLCTable"
"LightSource=","Sun+Sky",1,0.25
"LogFormat=","/User/Configs/LogFormats/Std Output"
"LogCodes="-35,-36,30,23,36,21,25,-33,-34,-32,-9,-10,-8,-1,-2,-4,-5,-14,-15,-7,-12,...
"PromptList=","/User/Configs/Prompts/Default (none)"
"10:08:46 Stability: (Photo 20s SLP<0.5) (Cond 20s SLP<0.01)"
```

Figure 9-11. A data file, and its separate .HDR file.

Remarks

The choice is "Embedded" or "Separate .REM file". If you select 'Separate .REM file", all remarks will go into a .REM file. This can really clean up a fluorescence file, which contains remarks from flash or dark pulse events.

Stability Details

The choice is "Not logged" or "Logged". If you choose "Logged", your data file will contain extra columns of data. The mean, standard deviation, CV, and slope of each variable in the stability list will be appended to each line.

For example, if *Photo* and *Cond* are in the stability list, then the data file column headers will appear as shown in Figure 9-12.

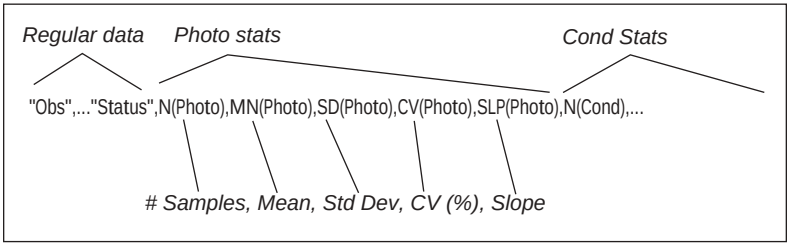


Figure 9-12. Stability details (in this case for PHOTO and COND) are appended to the normal data record. There are five extra columns for each variable in the stability list.

Statistics

The choice is "None" or "→ .Stats file". When set to "→ .STATS file" creates a separate file that contains statistics on each floating point variable logged in your normal 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 9-13. Example of a .STATS file. Each relevent floating point variable logged in the data file has a corresponding mean (MN) and standard deviation (SD) logged in the stats file.

LogList Editor

The LogList Editor (Figure 9-14) allows you to edit the entire list. Items in the list are flagged if they are required for recomputation (according to the current Compute List). You may remove any of these items, however. Thus, if you wish to only log 3 items per observation to save a lot of room, you may. Also, as an aid to building log lists, the master list (the menu of possibilities from which you choose when Changing or Inserting) indicates whether or not an item is already in the Log List.

The LogList Editor is a dialog, so if you leave by pressing **cancel**, all changes are undone. If you leave by pressing **OK**, you will be given the option of storing your changes. Your changes remain in effect, however, regardless of whether or not you store them.

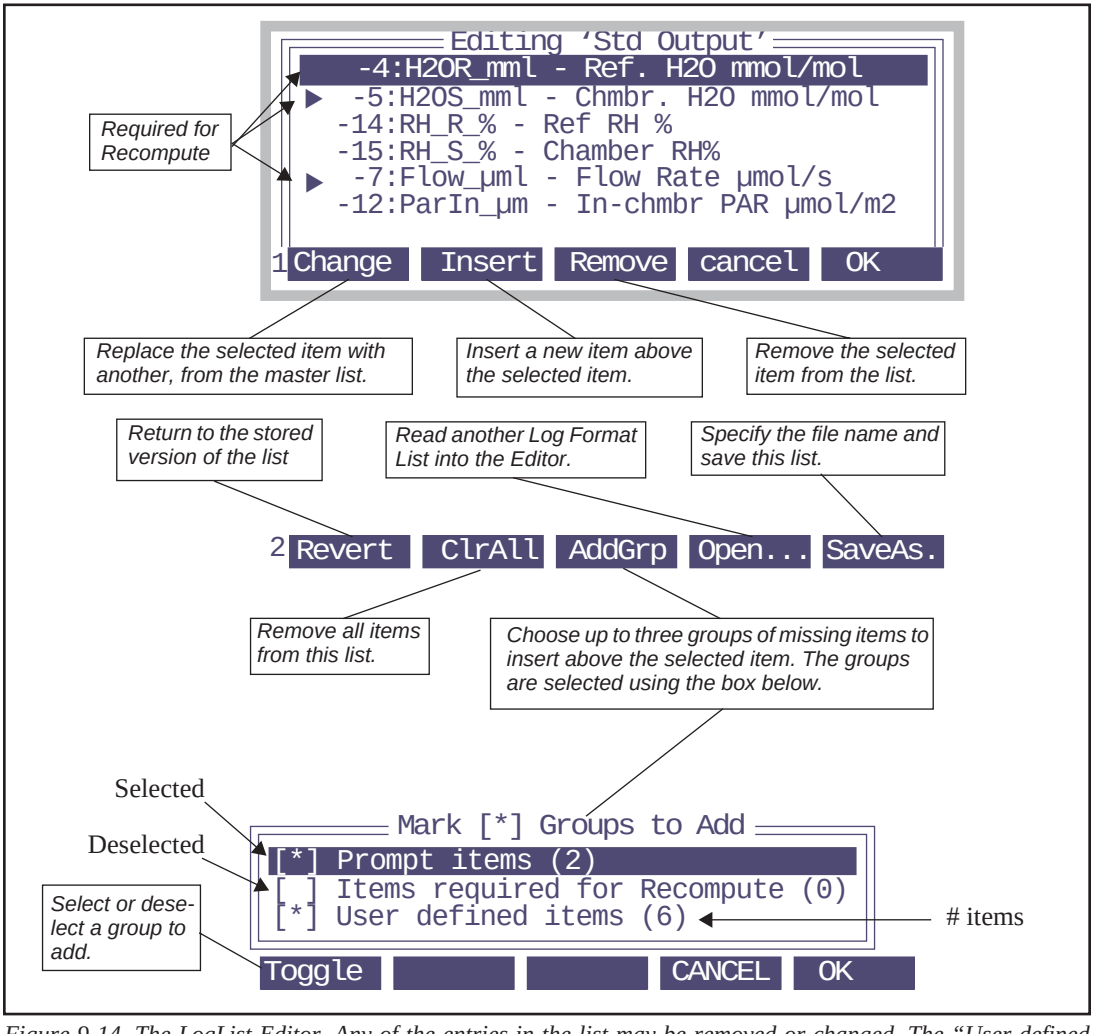


Figure 9-14. The LogList Editor. Any of the entries in the list may be removed or changed. The "User defined items" in the Group Add menu are any user defined (Compute List) items that aren't already in the Log List.

LogList Files

LogList files are generally stored in the directory /User/Configs/LogFormats directory. The format of these files is shown in Figure 9-15.

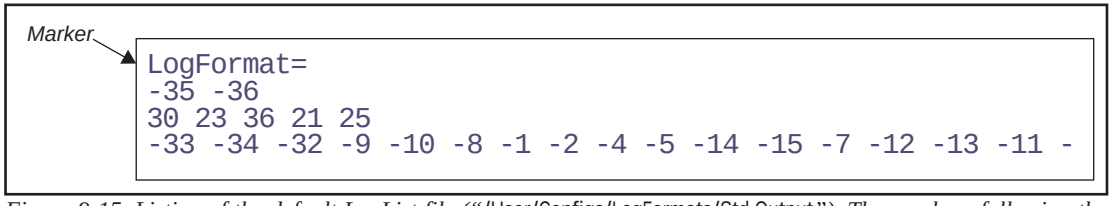


Figure 9-15. Listing of the default LogList file ("/User/Configs/LogFormats/Std Output"). The numbers following the marker indicate which items (system or user variables) are to be logged. These numbers can be on any number of lines; it does not matter how they are distributed.

Log list files contain the marking label "LogFormat=", followed by a list of ID numbers, on any number of lines. Positive variable ID numbers indicate user variables (Chapter 14), and negative IDs indicate system variables (Chapter 15).

Logging Times and Dates

The default LogList includes an observation time in HH:MM:SS format, and also the number of seconds that has elapsed since the file was opened. Table 9-1 lists all the system variables that contain time and date information. To include any of these in your log file, simply access the Log Editor, and insert them where you want them to be.

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 file 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	19970811

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. Thus, for example, "10:40:17" internally becomes 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

Frequently, it is convenient to log constants (such as leaf#, plot#) and remarks (treatment codes, etc.) as columns in the data file. OPEN's Prompt Control allows you to define these types of fields. Once defined, you can cause the program to prompt you for these values each time you press Log, or anytime you press the AutoPrompt function key (**F5** level 3).

Prompt Control

The Prompt Control screen (Figure 9-16), accessed from the Config Menu, provides a method of defining and managing user constants and remarks.

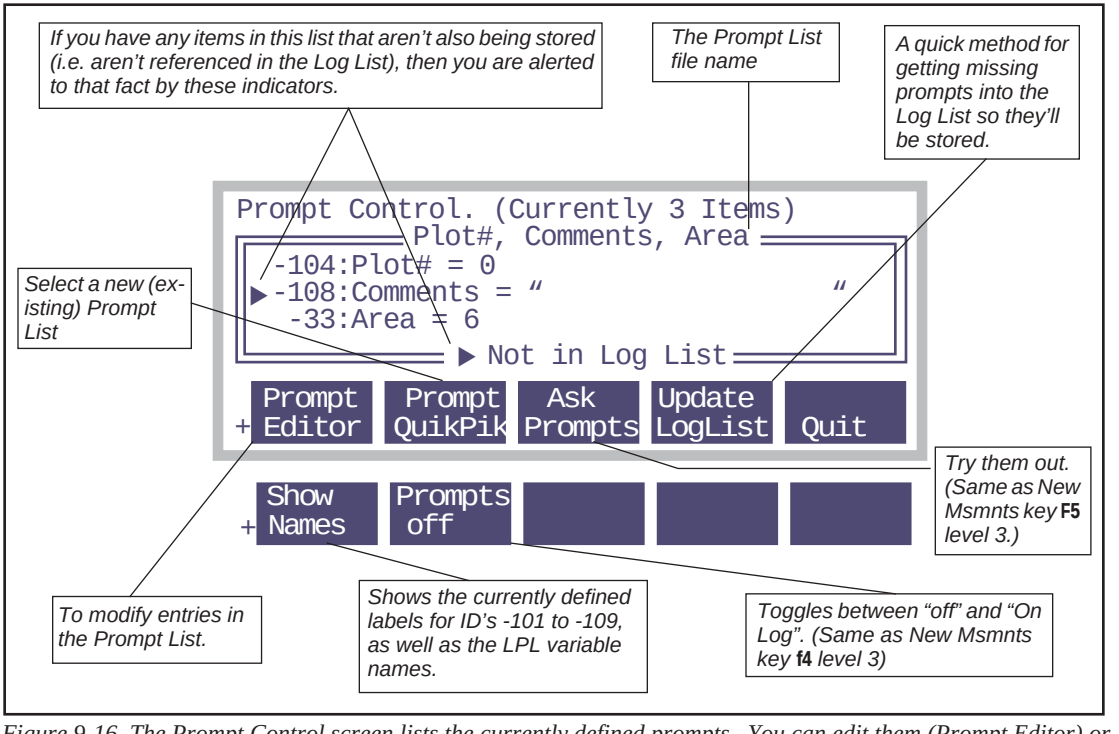


*Figure 9-16. The Prompt Control screen lists the currently defined prompts,. You can edit them (Prompt Editor) or select a new list. (**Prompt Quikpik**). The prompts are used when **f5** level 3 is pressed in New Measurements mode, or when **f3** is pressed here (**Ask Prompts**).*

The prompt configuration shown in Figure 9-16 will prompt for “Plot#” (an integer), “Comments” (a 16 character maximum length 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 **Prompts On** (f4 level 3)

There are two configuration commands (Chapter 16) that pertain to prompting:

```
PromptList= <file name>
Prompts= <off / ON LOG>
```

The PromptList file is simply a list of variable ID numbers that you wish to be prompted for. These ID numbers can be user defined constants, or any of the system constants described in Table 9-2. The Prompts= command (introduced in OPEN 3.2) determines whether prompts are asked automatically with each log.

Prompt List files are usually stored in the directory /User/Configs/PromptList/.

If you install the configuration example “Survey Measurements”, then go to Prompt Control and QuikPik the file “Plot#, Comments, Area”, the display will appear as shown in Figure 9-16 on page 9-15. The Prompt Control screen will indicate which if any prompts are missing from the active Log List.

To add missing items to the Log List, you can press **Update Loglist** (**F4**), or else go to the Logging Control screen, and update from there.

The Prompt List Editor is illustrated in Figure 9-17. This editor functions in a similar manner to the Log List editor, but with some small differences, including a **Relabel** key (**F2** level 2) that operates on ID values -101 to -109, the system constants with user defined labels.

Using Prompts

When prompts are defined, you can cause OPEN to ask you for values for them from New Measurements mode by pressing **F5** level 3. To be prompted automatically when you manually log data (as opposed to logging with an AutoProgram), set **f4** level 3 to “Prompt on Log”.

Interaction With Real Time Graphics

When you press the log button with “Prompt on Log” and RTG is active and displaying graphs, what happens? With OPEN 3.01 and below, the graphs DO NOT go away, and it’s not clear what is happening. You can press **ctrl**

shift ←← to turn off graphics mode, and turn on text mode, so you can see the prompts. With OPEN 3.2 and above, this happens automatically, so there is no problem.

Timing relative to logging

When you manually log, here is the sequence of events:

1. Latest measurements are captured
2. If "Prompt on Log", prompts are presented, user responses recorded.
3. Computations done.
4. Record added to log file.
5. Beep (if enabled).

Thus, if you are prompting for values that are used in computations, the computations done for the logged data will use the latest values that you have entered.

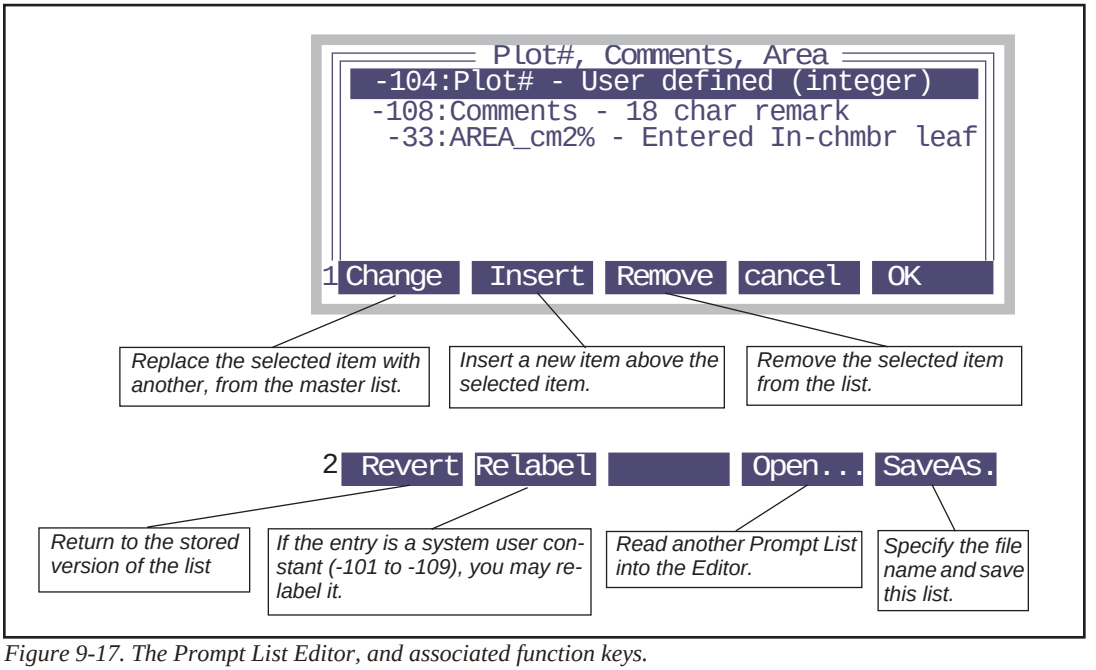


Figure 9-17. The Prompt List Editor, and associated function keys.

System Variables for Prompts

Table 9-2 lists the system variables that are available for use with prompts.

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
-86	Fo	Minimal fluorescence[b]	flr_fo
-88	Fm	Maximal fluorescence	flr_fm
-101	AuxF1[c]	Double precision floating point value.	auxF1
-102	AuxF2		auxF2
-103	AuxF3		auxF3
-104	AuxN1	Long integer (-2147483648 to +2147483647)	auxN1
-105	AuxN2		auxN2
-106	AuxN3		auxN3
-107	AuxS1	8 character string	auxS1
-108	AuxS2	18 character string	auxS2
-109	AuxS3	38 character string	auxS3

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 it's name, not its label.
b.Available when configured for fluorescence.
c.Items -101 through -109 have user defined labels. What's shown here are default labels.

System variables -101 through -109 have user defined labels. Thus, for example, system variable number -104 has a default label of "AuxN1", and in Figure 9-17 on page 9-17, it has been changed to "Plot#". These system variables are also different from the others in that their values aren't used for anything, and don't change unless you change them. They are really system constants, not variables.

"Which Ones Do I Use?"

Suppose you wish to store some auxiliary, user entered information with your data. How do you determine which of these system constants to use?

If what you want to store involves letters or words, then use the strings. If it's numeric, and can be an integer (10, -1634, etc.), then use variables -104 through -106. If it's floating point (5.124, -7E-12, etc.), then use variables -101 through -103. You may use whatever combination of these items that you need. And, of course, whichever ones you do use, you can and should rename to be more meaningful than "AuxF1".

If you wish to select data records for plotting based on the value of these constants, then use the numeric ones. The logical selection criterion of GraphIt (described in Chapter 12) works for numeric fields, not string fields.

"What If I Need More?"
You can add as many constants as you like by modifying your Compute List. See Chapter 15. Remarks (strings) can also be defined in Compute List Files.

Prompts and AutoPrograms

To trigger the prompt list from an AutoProgram, use the command

```
UcnAskAll
```

Chapter 25 discusses AutoProgram programming. If you are making an AutoProgram using the AutoProgram Builder (described on page 9-30), the menu selection "G) Prompt for user constants" will accomplish this task.

Prompt List Files

Prompt List files are generally stored in the directory /User/Configs/Prompts directory. The format of these files is shown in Figure 9-18.

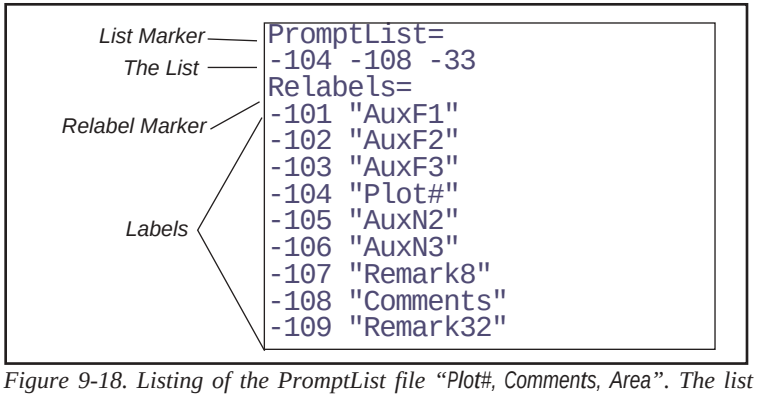


Figure 9-18. Listing of the PromptList file "Plot#, Comments, Area". The list contains the ID's of values to be prompted for, and the label list contains the names of all 10 system variables that can be renamed.

AutoPrograms

One mechanism by which the LI-6400 can operate automatically is the AutoProgram.

What are AutoPrograms?

AutoPrograms are small LPL application programs designed to run on top[1] 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. See Chapter 25.

Table 9-3. Standard AutoPrograms

Name	Description
"A-CiCurve"	Controls CO_2 mixer, logs data based on stability.
"AutoLog"	Logs instantaneous data at regular intervals.
"LightCurve"	Controls LED source, logs data based on stability.
"Remote Control"	Interprets and executes commands sent from a remote terminal or computer.
"Timed Lamp"	User selects LED source values, logging frequencies, and time intervals. Useful for sunfleck simulations.

[1] 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 prompt you for some input data. The default values will be the values that you entered the last time that AutoProgram was launched.

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 9-19).

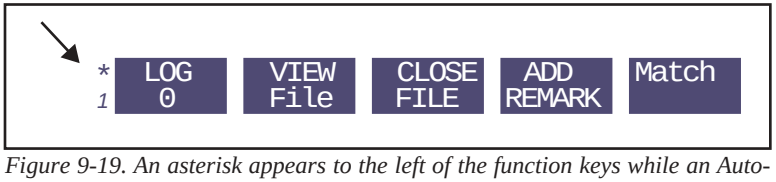


Figure 9-19. An asterisk appears to the left of the function keys while an AutoProgram is active.

- **You can't leave New Measurements Mode**
  If you attempt to exit New Measurements Mode, you will be presented with the AutoProgram Exit screen:

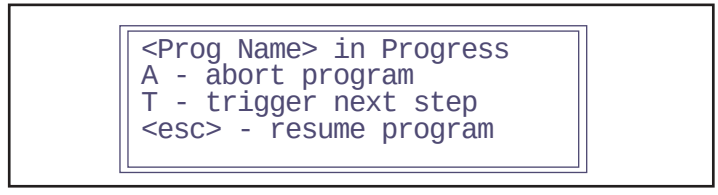


Figure 9-20. The AutoProgram Exit screen. Pressing **A** *will terminate the AutoPrograms,* **T** *will trigger the next step in the AutoProgram, and* **escape** *will let the AutoProgram resume*

- **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 9-21).

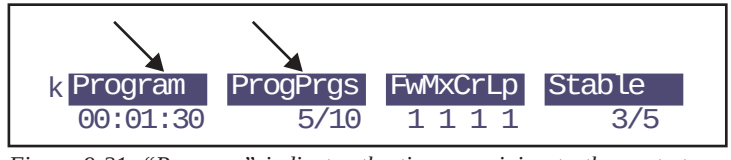


Figure 9-21. "Program" indicates the time remaining to the next step, while "ProgPrgs" indicates the program is presently working on the 5th of 10 steps.

The "Program" indicator shows the time remaining until the next step.

- **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 terminate it early, or trigger the next step before it would otherwise occur. To do either, press **escape** to access the AutoProgram Exit screen (Figure 9-20 on page 9-22), and press **A** or **T**.

AutoProgram Descriptions

OPEN includes a suite of default Autoprograms, described below:

"A-CiCurve"

Controls the 6400-01 CO_2 mixer, and logs data when stability is reached.

A-CiCurve will prompt for the following:

1 **Desired C_a values (µmol/mol)**

Enter the CO_2 values you wish to achieve. Type the values, separated by spaces, such as

```
400 300 200 100 50 400 600 800
```

Note: these values can be reference CO_2, sample CO_2, or control (milli-)voltages (for any of the three operating modes of the mixer). This AutoProgram does not change the mode of the mixer control, only the value. Usually for A-Ci curves, it is best to operate in constant reference mode (R), or constant command signal mode (C).

2 **Minimum wait time (secs)**

The time between a change in the mixer's target, and when the program will begin to check stability to see if an observation can be logged. No logging will occur during this period.

3 **Maximum wait time (secs)**

The time period (following the minimum wait time) during which logging could occur, if stability is reached. If stability is never reached, logging will occur at the end of this time.

4 **Matching**

You are prompted

```
Match if |ΔCO2| <
```

which allows you to enter a CO_2 differential threshold that will cause matching to occur if the absolute value of the sample - reference IRGA is smaller. If you wanted to match automatically on every observation, enter a large number (100, 1000, etc.). If you never want to match automatically during the AutoProgram, enter 0. Typically, you might want to use 15 or 20 µmol mol^{-1} as the threshold.

If matching occurs (for either software version), it occurs immediately after the minimum wait time. Following automatic matching (which typically takes about 30 seconds), there will be a short delay (10 seconds), then the maximum wait time and stability checking is entered. Thus, automatic matching tends to increase the total time between logging events by about 40 seconds.

5 **Stability...OK?**
You have a chance to check or adjust your stability definition. (This is not your last chance; you also can do it anytime while the AutoProgram is running.)

Two Styles for "Program"

The *Program* field (Figure 9-21 on page 9-22) has two styles with the *A-Ci-Curve* AutoProgram (and also *LightCurve*). The delimiter used is a semicolon, or asterisk, as explained in Table 9-4.

Table 9-4. Delimiters used in the Program field for *A-CiCurve* and *LightCurve*.

Style	Meaning
00;01;30	(In minimum wait time) 1 minute 30 seconds remains until the program starts checking for stability to see if it can log data.
00*01*30	(Checking stability) As soon as the stability meets the criterion, it will log. Otherwise, logging will occur in 1 minute and 30 seconds.

"AutoLog"

AutoLog is designed to log instantaneous data at regular intervals.

■ **AutoLog prompts for the following:**

1 **Log every __ secs:**
Enter the desired period between logged observations. This can be any increment of 0.5 seconds.

2 **Run for __ minutes:**
This determines how long *AutoLog* will run. If you manually log data while AutoLog is running, it does *not* affect when *AutoLog* quits.

3 **Match every __ minutes (0 = none):**
Enter the desired interval between matching, or 0 for no matching.

While *AutoLog* is running, the *Program* field (Figure 9-21 on page 9-22) will show the time remaining for the program, *not* the time until the next observation is logged (as was the case for earlier versions of AutoLog).

AutoLog has another difference from normal AutoPrograms: Trigger (the T option in Figure 9-20 on page 9-22) will terminate the program, instead of logging early. If you want to log early, just press **Log**.

"LightCurve"

Controls the 6400-02 or -02B LED Source, and logs data when stability is reached.

■ ***LightCurve* will prompt for the following:**

1 Desired lamp settings (µmol/m2/s)
Enter the PAR values you wish to achieve. Type the values, separated by spaces, such as

```
1500 1000 800 600 400 200 50
```

2 Minimum wait time (secs)
The time between a change in the lamp's target, and when the program will begin to check stability to see if an observation can be logged. No logging will occur during this period.

3 Maximum wait time (secs)
The time period (following the minimum wait time) during which logging could occur, if stability is reached. If stability is never reached, logging will occur at the end of this time.

4 Matching
You are prompted

```
Match if |ΔCO2| <
```

which allows you to enter a CO_2 differential threshold that will cause matching to occur if the absolute value of the sample - reference IRGA is smaller. If you wanted to match automatically on every observation, enter a large number (100, 1000, etc.). If you never want to match automatically during the AutoProgram, enter 0. Typically, you might want to use 15 or 20 µmol mol^{-1} as the threshold.

If matching occurs (for either software version), it occurs immediately after the minimum wait time. Following automatic matching (which typically

takes about 30 seconds), there will be a short delay (10 seconds), then the maximum wait time and stability checking is entered. Thus, automatic matching tends to increase the total time between logging events by about 40 seconds.

5 **Stability...OK?**
You have a chance to check or adjust your stability definition. (This is not your last chance; you also can do it anytime while the AutoProgram is running.)

While *LightCurve* is running, the "Program" field will show time in two styles, as described in **Two Styles for "Program"** on page 9-24.

"Remote Control"

Remote Control is an AutoProgram than doesn't log or control or do much of anything. However, as it's name implies, it does allow an external device (computer or terminal) to control the LI-6400 via it's comm port. Here's how it works: AutoPrograms run on top of OPEN, and this particular one does nothing except set up for incoming communications, which are captured in a circular queue, and then goes off and does New Measurements mode. A line feed (ascii 10) serves as a magic character; when one is received, it triggers an interrupt, in which the message in the queue is compiled, executed, and disposed. If you send it valid LPL commands and functions defined by OPEN, they will be executed by the LI-6400, just like they would be if they were commands in an AutoProgram running on the instrument. **Useful AutoProgram Commands** on page 25-12 and **Low Level Control Tools** on page 25-22 list some of the commands that will work for Remote Control. For example, to make the LI-6400 log a data record, send

```
LPLog
```

followed by a line feed. Note that case doesn't matter, so LpLog is LPlog is lpLog, etc. Other examples are shown in Table 9-5.

Table 9-5. Sample commands to send to the Remote Control AutoProgram. Each should be followed by a line feed.

Example Command	Description
`1000 LPSetLamp`	Set the lamp to 1000 µmol m^{-2} s^{-1}
`50 4 FlowSetNewTarget`	Control at 50% RH
`LPMatch`	Match the IRGAs

If you want logged data to be sent back to the computer, simply specify the comm port as the destination when you open the log file (press escape when prompted for the file name, then press C when given a choice between logging to memory or logging to the comm port).

"TimedLamp"

TimedLamp allows you to program timed changes in the 6400-02 or -02B LED source. 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. *TimedLamp* lets you do this. You can also specify the logging frequency for each lamp level. You may, for example, want to record data less frequently during the initial stability period, and very frequently when the light changes.

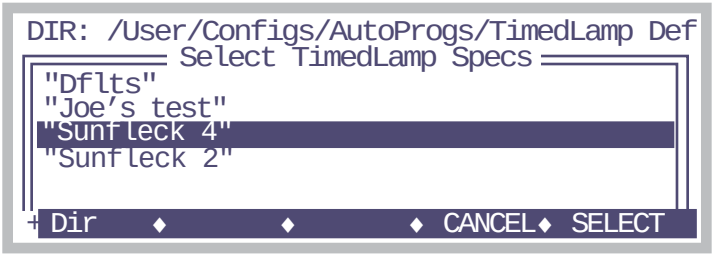
TimedLamp prompts

1 Get Data from Keyboard or File

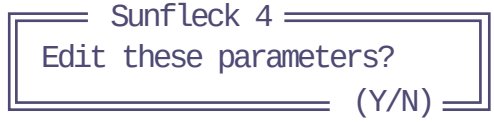
Press **K** to enter the program's set points from the keyboard (if you do this, go to Step 3). Or, press **F** to use a previously entered set of points (if you do this, go to Step 2).

2 Select TimedLamp's Steps

You are asked to select a file from those in the directory "/User/Configs/AutoProgs/TimedLamp Defaults".



Once a file is selected, you are given a chance of viewing or editing the file's contents before starting the program:



If you press **Y**, go to Step 3. If you press **N**, the program will begin running immediately.

3 **Time(s) Lamp(µmol) LogInt(s)**

You a shown a window in which you can enter or edit all the desired program steps (Figure 9-22). You can have up to 20 steps - the window scrolls. (If you need more, see page 25-8).

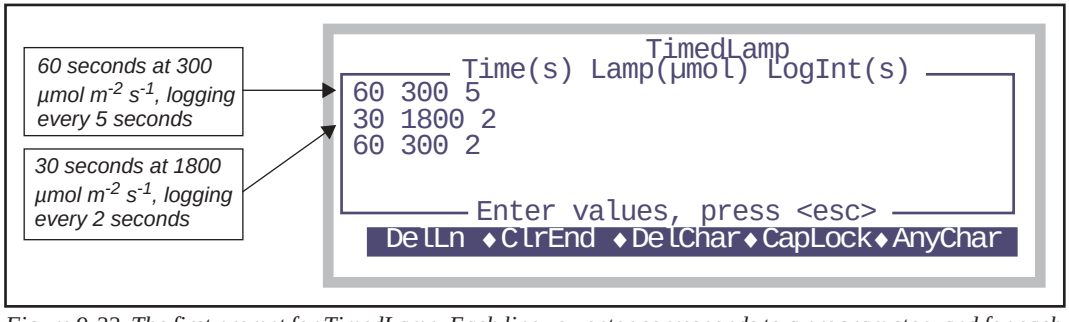


Figure 9-22. The first prompt for TimedLamp. Each line you enter corresponds to a program step, and for each step, specify the time period (secs), light level (µmol m^{-2} s^{-1}), and logging interval (s). When the list is the way you like it, press **escape** *to go on.*

4 **Options**

Once you leave the editing window, you are presented with some options:

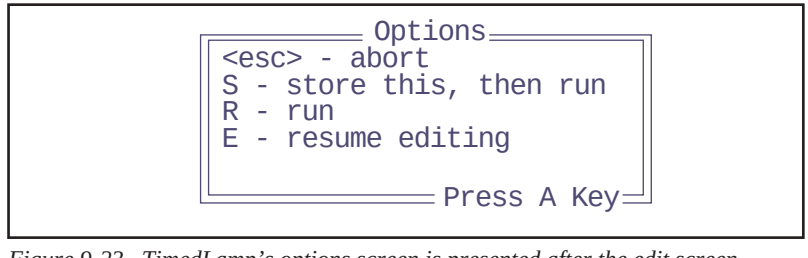


Figure 9-23. TimedLamp's options screen is presented after the edit screen.

R and **S** both run the program, but **S** stores the set points for possible use the next time *TimedLamp* runs (Figure 9-24). **escape** aborts *TimedLamp*, and **E** goes back to the edit window.

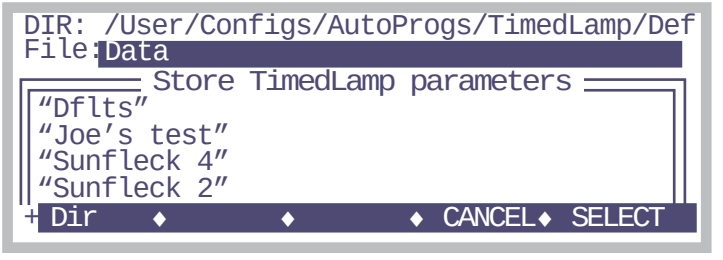


Figure 9-24. TimedLamp's set points can be stored for use next time.

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: "Build 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, once it's built, and further customizing it.)

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 1: 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
```

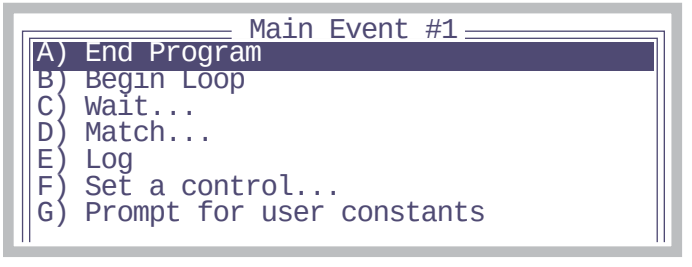
```
User enters wait time
LOOP over flow rates
  Set flow rate
  Wait some time
  Log
END LOOP
```

Before using the AutoProgram builder, it is important to have this sort of outline in front of you.

To Build This AutoProgram

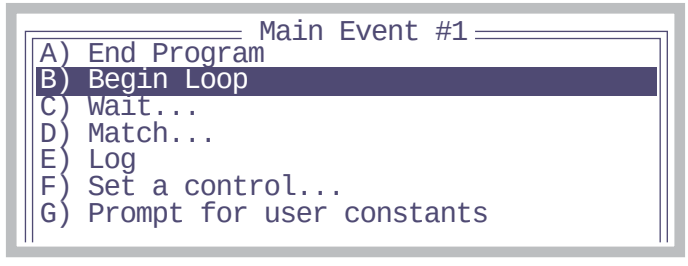
1 Launch the AutoProgram Builder

Select the entry entitled "Build a New AutoProgram" in the Utility Menu. It takes about 10 seconds to load and run. Press **enter** when finally prompted with "Press <enter> to start".

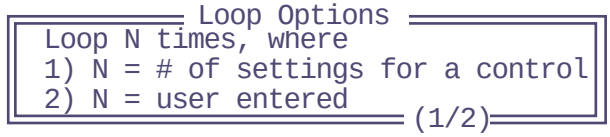


Since the first thing on our outline involves prompting the user for information, you might think the first thing we want to do would involve choice G "Prompt for user constants". Well, you're wrong. When we get done with entering program steps, we'll get to choose how the user inputs wait times and flow rates and any other constant that pertains to this program (Step 11 on page 9-35). So for now, let's just ignore the user entry steps, and get straight to the loop.

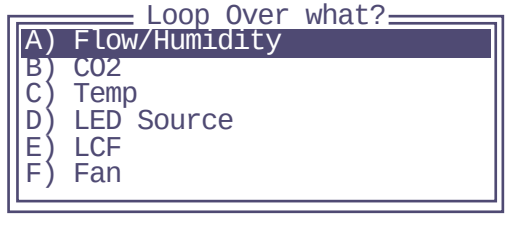
2 **Select "Begin Loop" from the menu**
Highlight "B) Begin Loop" and press **enter**, or else simply press **B**.



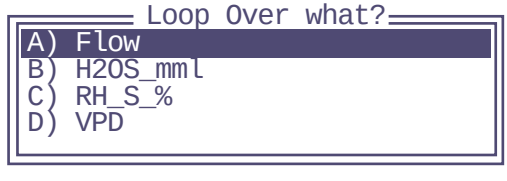
3 **Make it a Control Loop**
Press **1** to select a control loop.



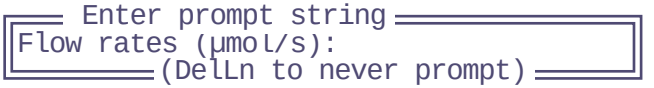
4 **Specify the type of Control Loop: Fixed Flow Rate**
When the control loop options are shown, select "A) Flow/Humidity" and press **enter**, or simply press **A**.



Follow that with what type of flow control: Flow (A) again.



5 **Edit the prompt string for the flow control loop**
You'll be given the chance to edit the prompting string for the flow rates. Just

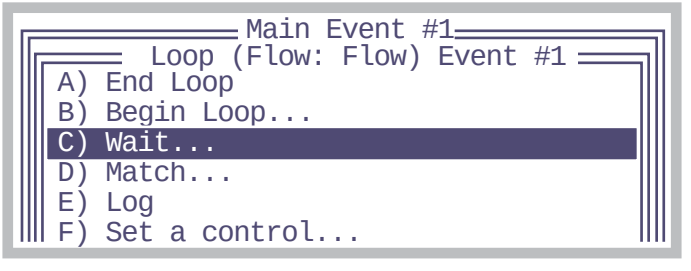


press **enter** if the prompt is OK. (You would press **DelLn** (**f1**) to clear this or any prompt if you want a fixed value that is never prompted for.)

6 **Edit the default values for the flow control loop**
You'll be given the chance to set the default values for the flow rates. Adjust

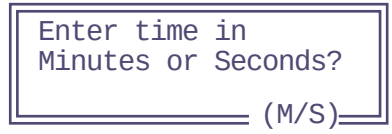
them as you wish, and press **enter** when done.

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**.

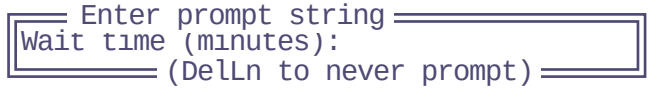


You'll be asked to specify what sort of wait you want. Press **A** to select the fixed time wait.

You'll be asked how you wish to specify this wait time. Press **M** to select minutes.

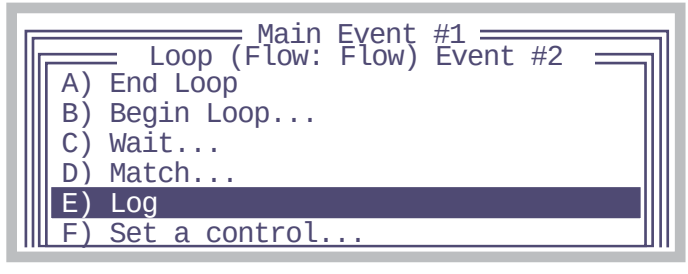


You'll be asked to edit the time delay prompt. Notice that the default units reflect your earlier choice of minutes or seconds. Press **enter**.

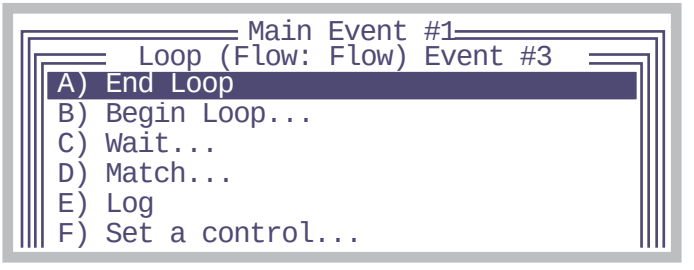


You'll be asked to specify the default wait time value. Press **enter** when done.

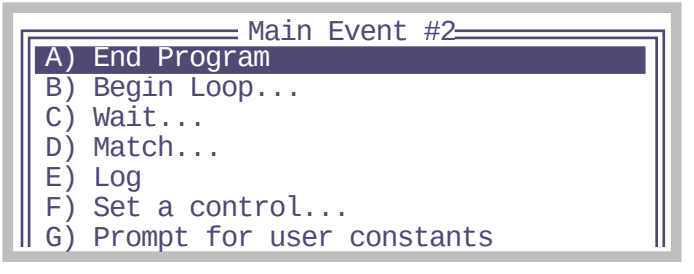
8 **Specify the second action in the flow control loop: Logging.**
The second action is logging, so highlight "E) Log" and press **enter**, or simply press **E**.



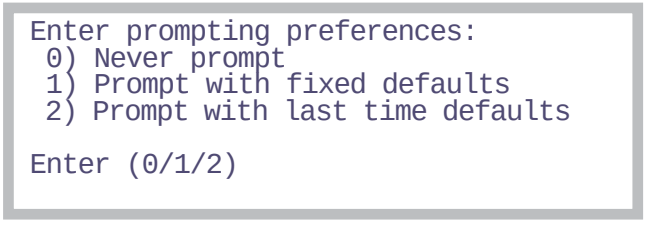
9 **End the control loop**
Highlight "A) End Loop" and press **enter**, or simply press **A**.



10 **End the program**
We're now out of the flow control loop, and since there is nothing more for the program to do, we'll end that by highlighting "A) End Program" and pressing **enter**, or else pressing **A**.



11 **Specify your prompting preference**
There are three options: 0)You can have the program never prompt you, but always use the values you've 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**.



12 Store the AutoProgram

You are given the opportunity to store the AutoProgram.

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 BaCaEAA, 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. When you exit the editor (Standard Editor), you can save the file.

Example 2: A-Ci Curves at Various Light Levels

This example builds an AutoProgram that does a family of A-Ci curves, each one at a different light level. The program will have this structure:

```
Prompt for inputs
LOOP over light level
   Set light level
   Wait fixed time
   LOOP over reference CO2
      Set reference CO2
      Wait for stability
      Match if deltas are small
      Log
   END LOOP
END LOOP
```

To Build This AutoProgram

1 Run the AutoProgram Builder
Select "`Build an new AutoProgram`" in the Utility Menu.

2 Make a control loop for Light (BJ)
Press **B** to select "`B) Begin Loop...`". Press **1** to specify a control loop, then press **D** for "`D) LED Source`", followed by **B** for "`B) Quantum flux`". Press **enter** twice to accept the prompt string and default values.

3 Select a post-light wait (CA)
We'll use a fixed time wait, so press **C** to select "`C) Wait...`", followed by **A** to select "`A) Wait: Fixed Time`". Specify Minutes, and modify the prompt to read "`Post-light wait time (minutes)`". Make the default value 5.

4 Make a control loop for CO_2 (B1E)
Make the inner loop by pressing **B** for "`B) Begin Loop...`", followed by **1** for a control loop, then **B** for "`B) CO2`", then **A** for "`A) CO2R`". Set the prompt string and default values as you wish.

5 Select a stability-based wait (CBA)
Press **C** to select "`C) Wait...`", followed by **B** for "`B) Wait: Min, Max, Stability`". Use Minutes for times, and modify the prompts and initial values for the min, and max values as you wish.

6 Match if the Δ's are small (DD)
For the second event in the CO_2 loop, press **D** for "`D) Match...`" followed by another **D** for "`D) Match if |ΔCO2|<x OR |ΔH2O|<y`". Keep the default prompts, but make the default values 10 for ΔCO_2, and 2 for ΔH_2O.

7 Select Log (E)
For the 3rd CO2 loop event, press **E** for "`E) Log`" next.

8 End Loop, End Loop, and End Program (AAA)
Finish by pressing **A** three times, to exit both loops and end the program.

9 Set Prompt Preferences, and Store.
Press **2** for the prompt preference (use last time defaults), then **Y** to store the program. The default name will be "`BbdCaBbaCbDdEAAA`", but you can change it to something more intelligible like "A-Ci Multi Light".

The AutoProgram Builder Reference

The major options of the AutoProgram Builder are shown in Figure 9-25.

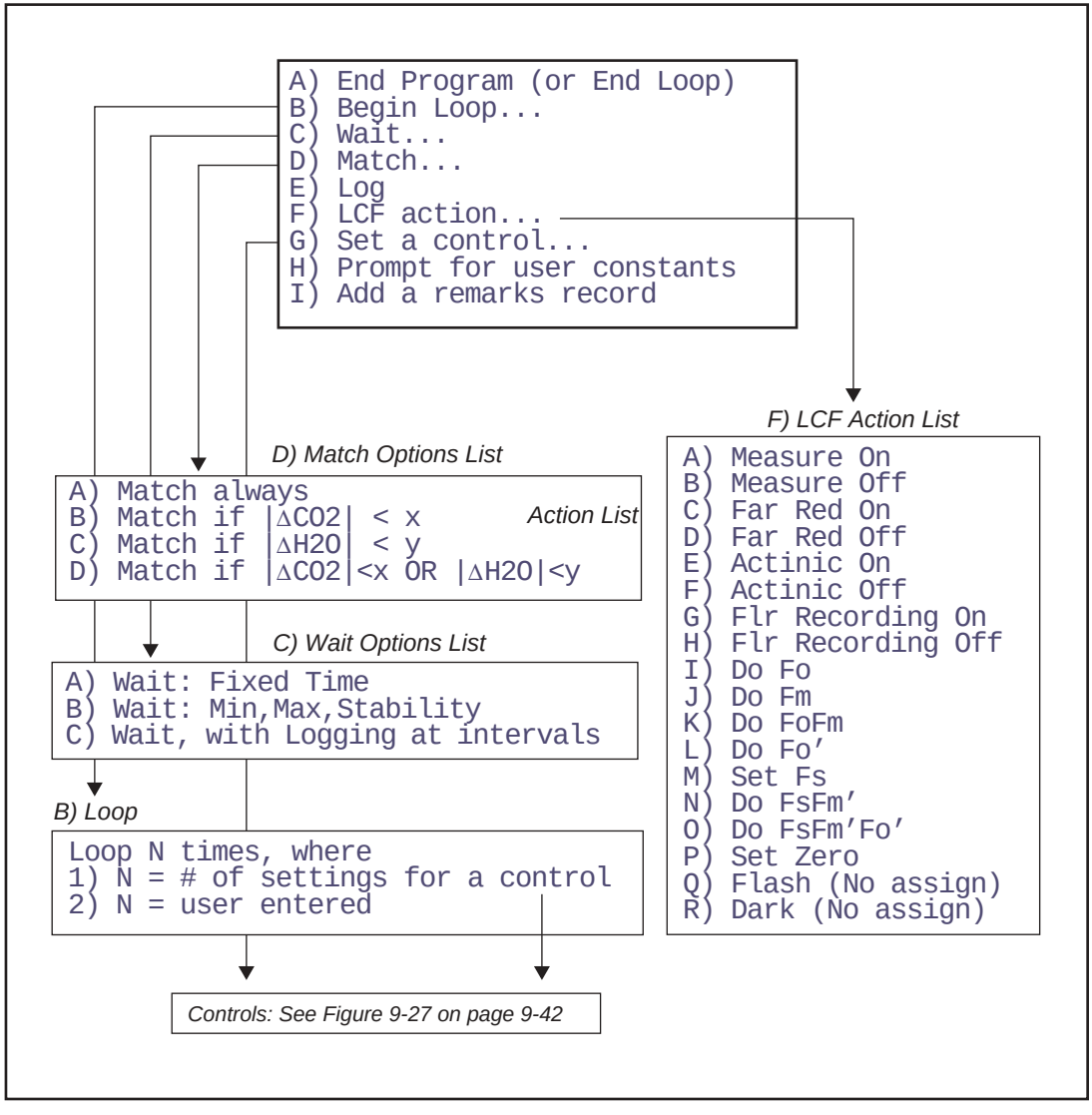


Figure 9-25. AutoProgram Builder's major options. The Type 1 Loop uses multiple control values, while the "Set a control…" event uses one control value. The Type 2 Loop sets no control values - it is a fixed count loop.

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 9-27 on page 9-42, 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.

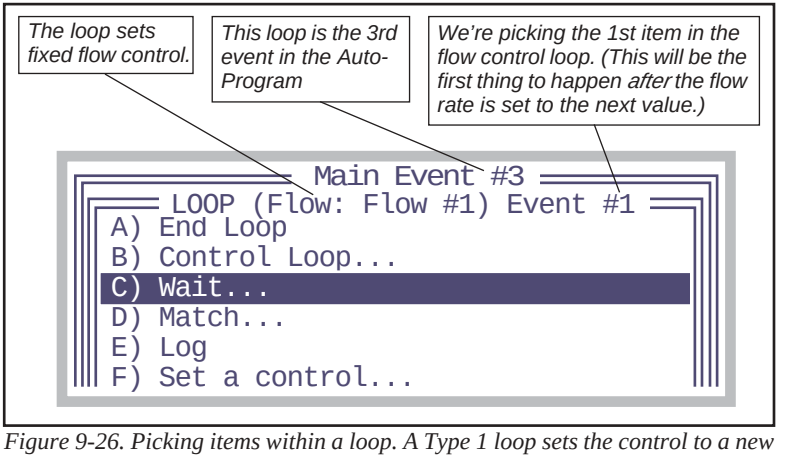


Figure 9-26. Picking items within a loop. A Type 1 loop sets the control to a new value automatically; you have to select everything else that you wish to have happen after that, however.

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, and 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 9-27 on page 9-42. For example, if you were building an light curve AutoProgram, 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 (**User Consts**) 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.)

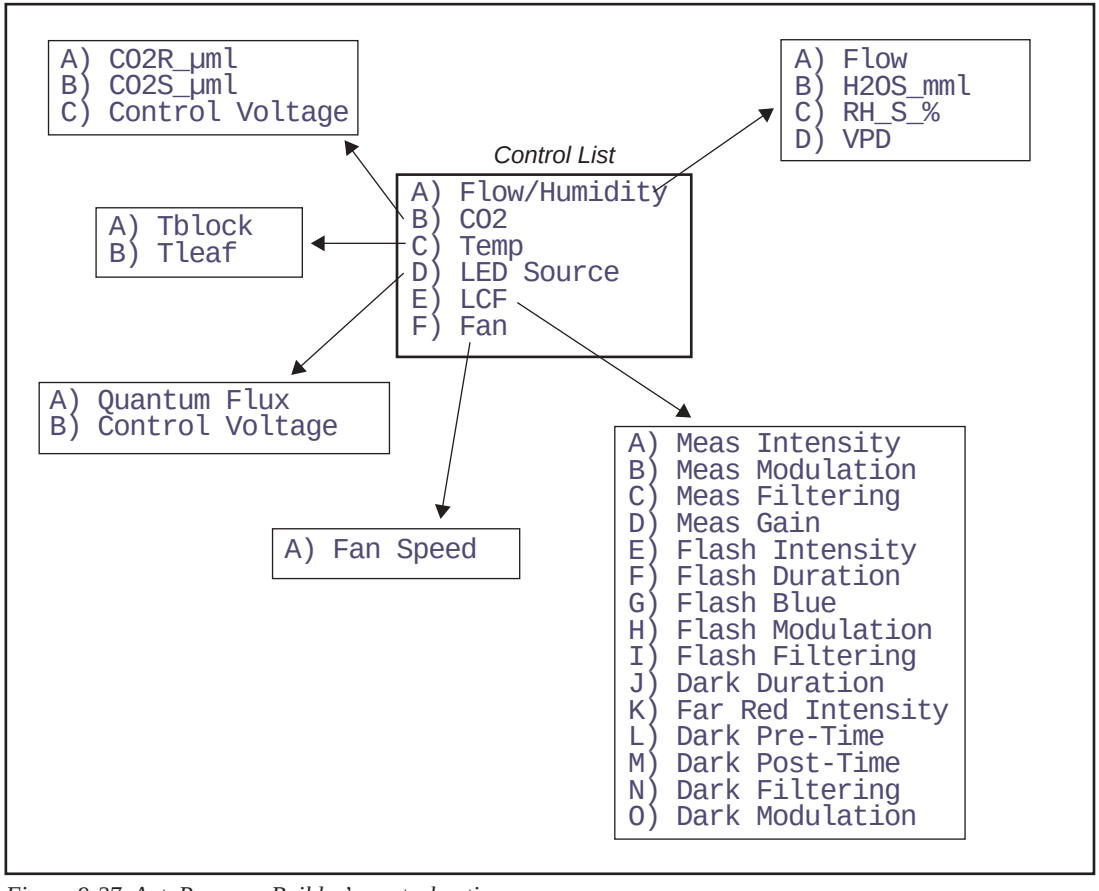


Figure 9-27. AutoProgram Builder's control options

Part III

Working With Files

The LPL File System

10

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* (in version 5) 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, 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 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/parm0 - factory calibration coefficients, /dev/parm1 - user calibration coefficients (zero and span, for example), .lcd - the display last setting for the display contrast, .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.

Partitions

The version 5 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.

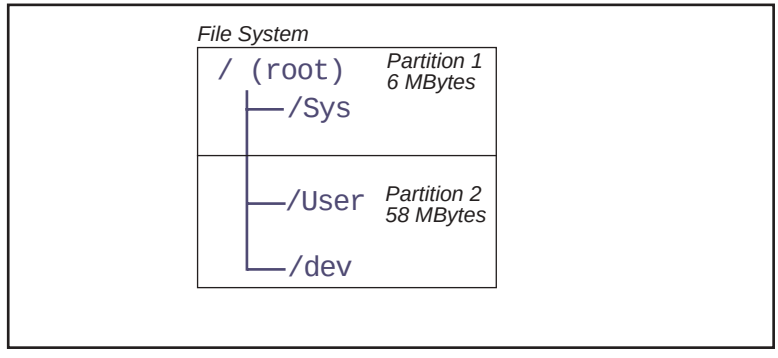


Figure 10-1. The file system has 2 partitions.

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", 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.

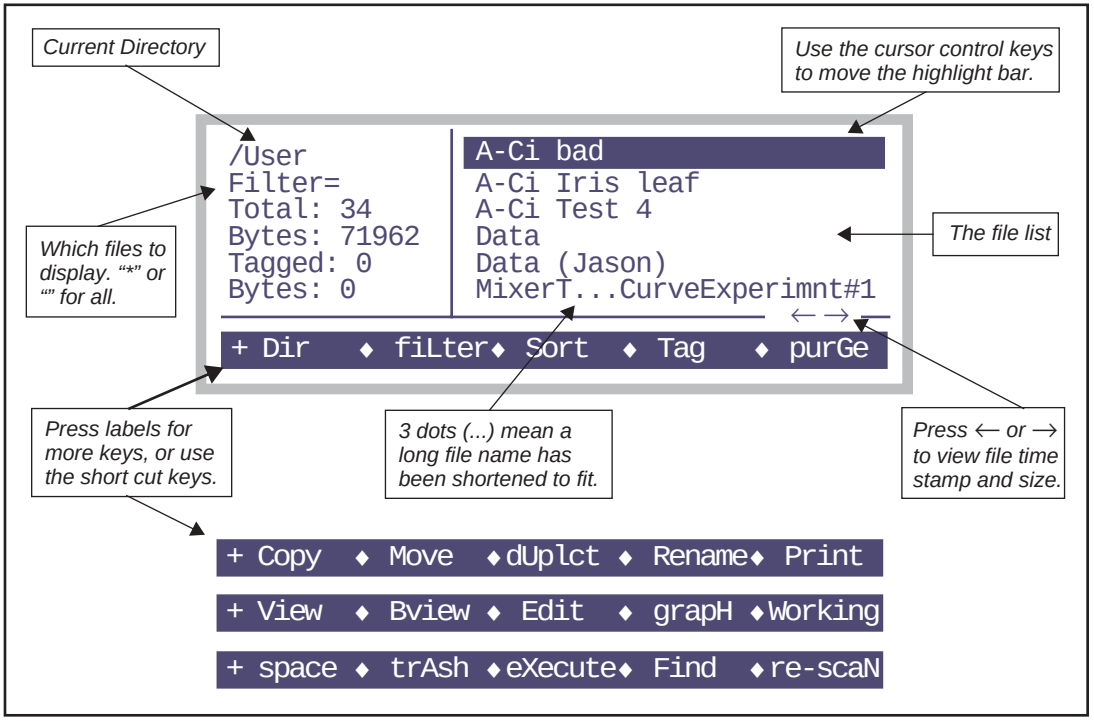


Figure 10-2. The Filer's main screen.

The Function Keys

To navigate through the Filer's function keys, press **labels** until the desired key appears. 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] This capitalized key short cut is not necessarily true for other screens in Open.

Table 10-1. The Filer's main function key labels and short cuts.

Label	Short Cut	Description
Dir	D	Select a directory. See **Directory Dialog** on page 10-7.
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 10-16.
Copy	C	Copy tagged files to a another directory. See **Copying and Moving Files** on page 10-16.
Move	M	Same as Copy, except the original is purged.
dUplct	U	Duplicate all tagged files. The new files will have "copy" appended to their name. See **Duplicating Files** on page 10-17.
Rename	R	Rename the highlighted file. See **Renaming Files** on page 10-17.
Print	P	Print all tagged files. See **Printing Files** on page 10-18.
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.
Working	W	Set the current directory to be the default once the Filer is exited.
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 short cuts.

Label	Short Cut	Description
eXecute	X	Run the highlighted file (if it's an LPL program). See **Executing Programs** on page 10-18
Find	F	Find matching file names in the entire file system. **Viewing a Subset of Files** on page 10-11.
re-scaN	N	Re-build the list of directories, and the list of files.

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.

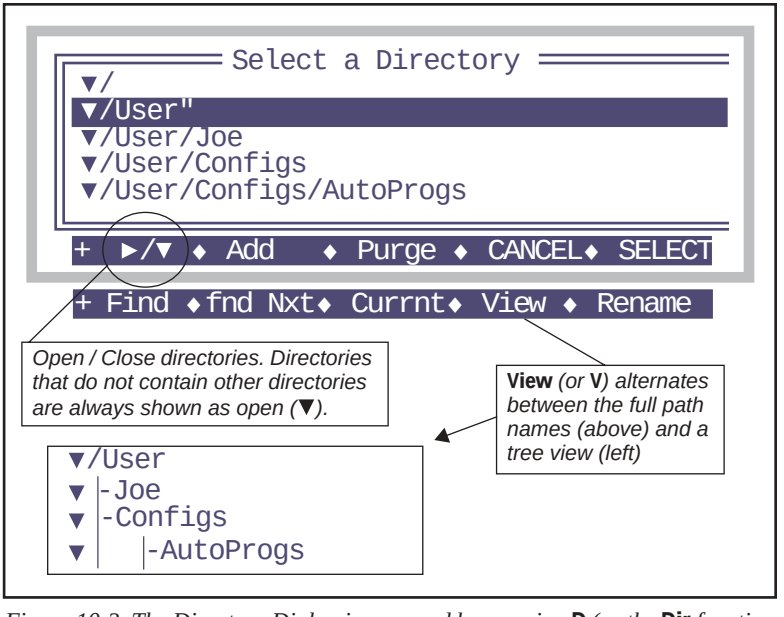


Figure 10-3. The Directory Dialog is accessed by pressing **D** *(or the* **Dir** *function key) from the main menu. The function keys are described in Table 10-2.*

In addition to allowing you to select the directory whose files you wish to view, 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
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**	Finds the next occurrence of the target string.
Currnt	**C**	Highlights 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

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.

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. 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.

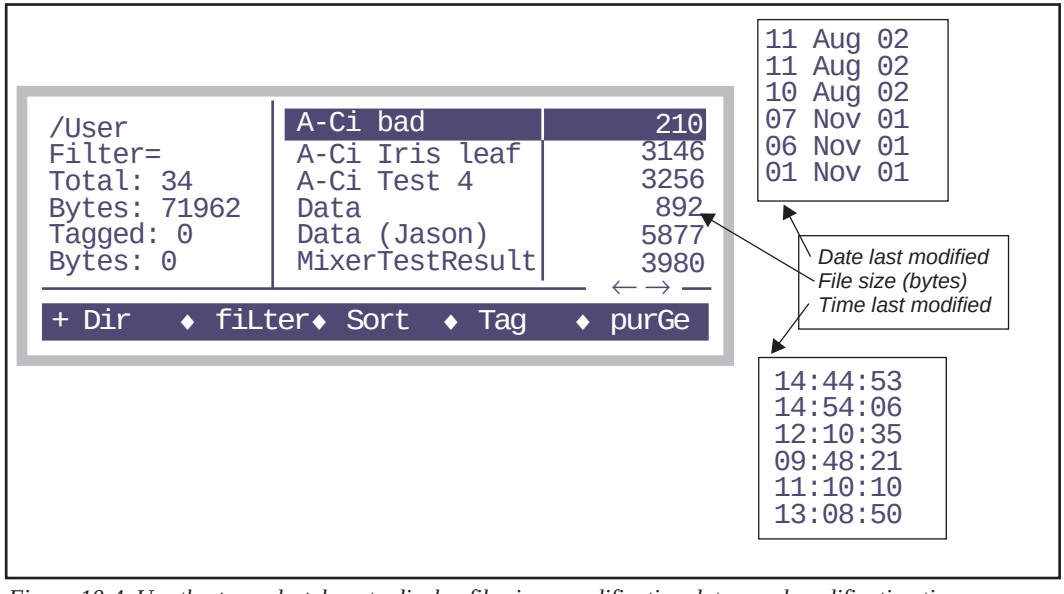


Figure 10-4. Use the ← and → keys to display file sizes, modification dates, and modification times.

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)

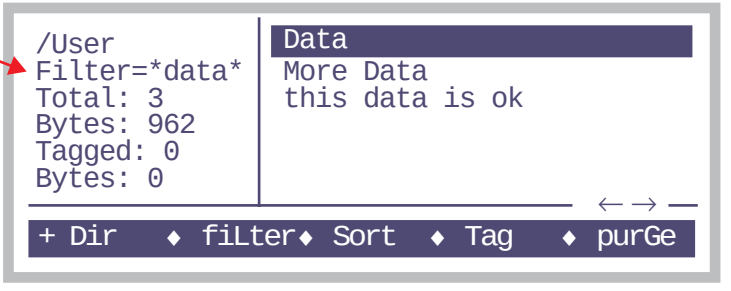


*Figure 10-5. The Filter determines which files are shown. A blank, or *, will show all files. The characters* or ? are wild cards.*

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.

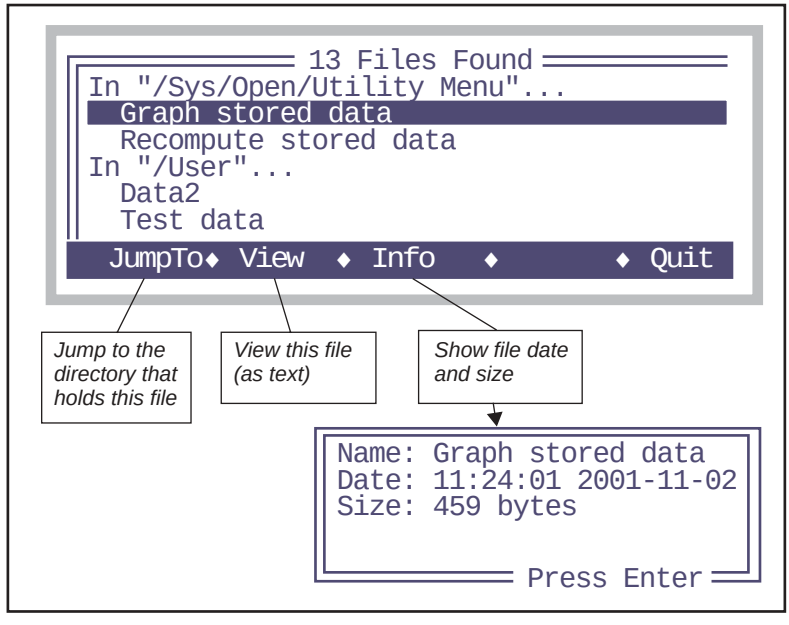


*Figure 10-6. Results of Find. The target in this example was "*data*", so any file containing the word 'data' is listed.*

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.

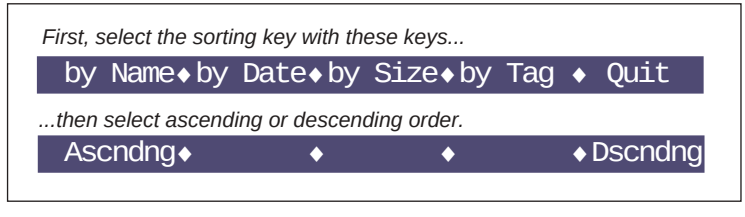


Figure 10-7. Function keys for sorting. You can also press **N**, **D**, **S**, **T**, *or* **Q** *for the sort options, and* **A** *or* **D** *for the direction choice.*

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 5-2 for details. To edit a file, press **Edit** (or **E**) instead. Refer to **Standard Edit** on page 5-13 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.

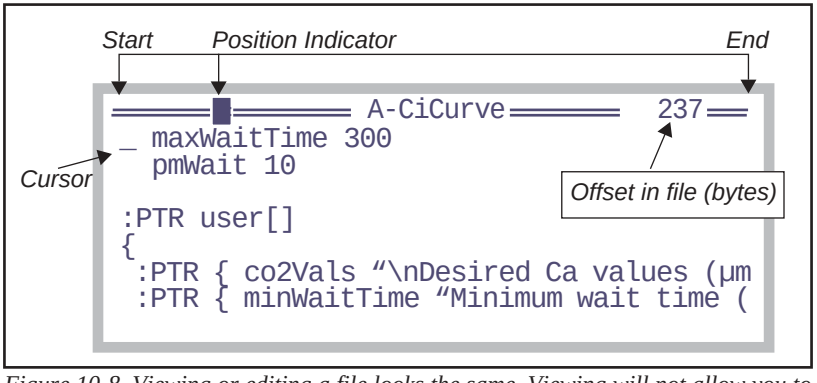


Figure 10-8. Viewing or editing a file looks the same. Viewing will not allow you to make changes, however.

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).

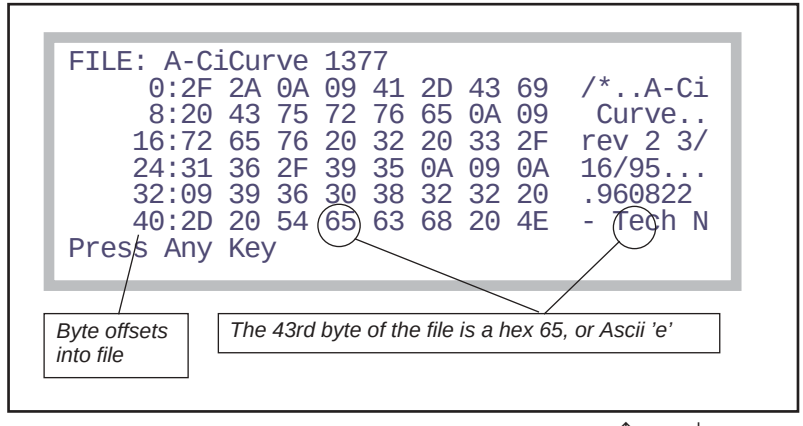


Figure 10-9. Binary file view. Use **home**, **end**, **pgup**, **pgdn**, ↑ *and* ↓ *to move through the file. To jump to any arbitrary byte in the file, press* **J** *and enter the offset value.*

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**.

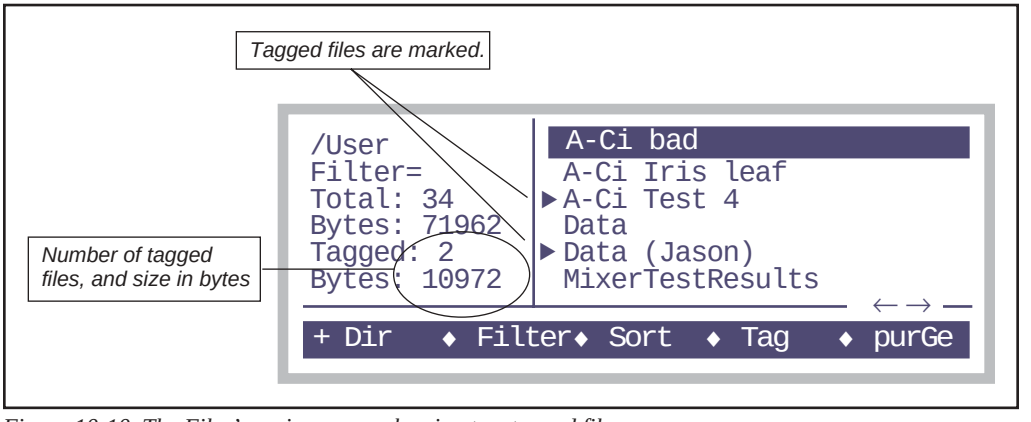


Figure 10-10. The Filer's main screen, showing two tagged files.

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.

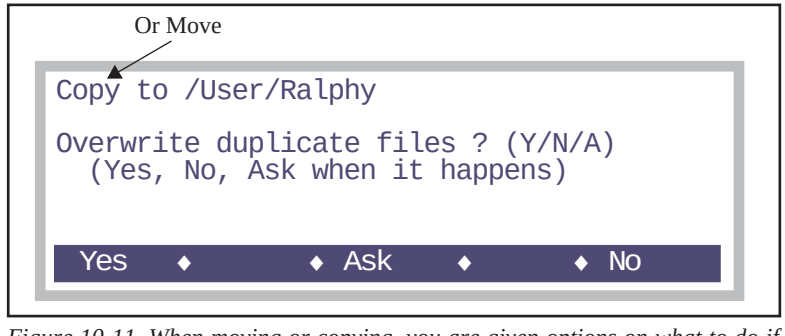


Figure 10-11. When moving or copying, you are given options on what to do if there is a duplicate file in the destination directory.

There is a utility program (/Sys/Utilites/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 21-19.

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.

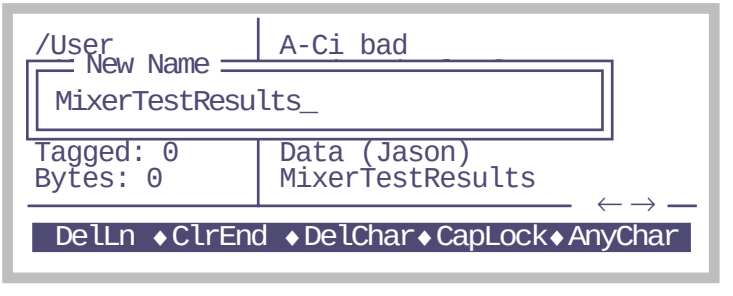


Figure 10-12. Renaming the highlighted file. Standard Line Edit is used (page 5-5).

Printing Files

Downloading files is discussed in Chapter 11. One 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. Typically, data files are stored as comma delimited. This is determined by the LogDelimiter= configuration setting (**LogDelimiter=** on page 16-39). 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 (regardless of how the file was stored), 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 **eXecute** (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 21-12.

Housekeeping

Space Available

The space available on both partitions 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.

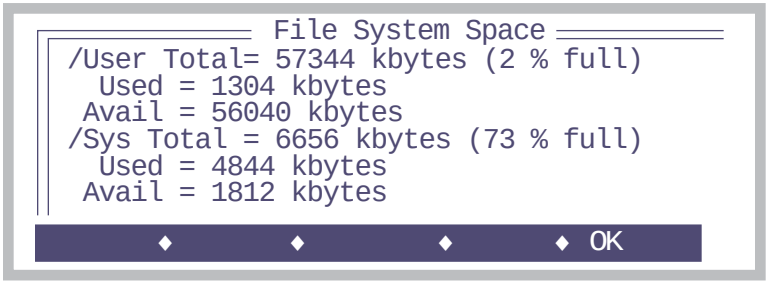


Figure 10-14. The available space display. /Sys and /User represent to two disk partitions.

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 2 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 for associated with the file list you are viewing.

Connecting to a Computer

Retrieving your data, and remote control

Connecting to a Computer

Eventually you'll 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, but the first step is getting the LI-6400 connected to your computer.

Connecting the LI-6400 to a Computer

The LI-6400 is shipped with a communications cable that has a 9 pin female D connector on each end. The part number is 9975-016. 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 PC. If your PC has a 25 pin male RS-232 connector, use the 9 to 25 pin adapter (Figure 11-1)

If your computer has USB ports instead of a serial port, then you must use a Serial/USB adapter. USB/Serial adapters are available from LI-COR (part number 392-07713) as well as other sources (e.g. www.keyspan.com).

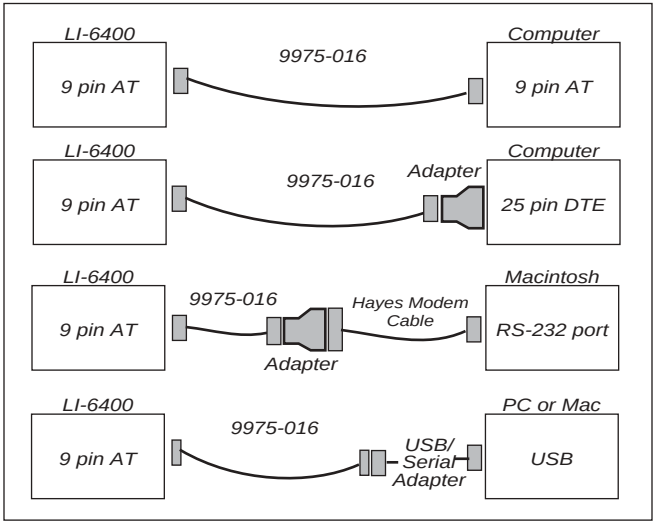


Figure 11-1. Typical cable connections using the 9975-016 Cable Kit, with 9-25 pin adapter.

Using the Comm Port

OPEN's Utility Menu has two entries useful for interfacing with computers: "`Configure the COMM port`", and "`File Exchange Mode`".

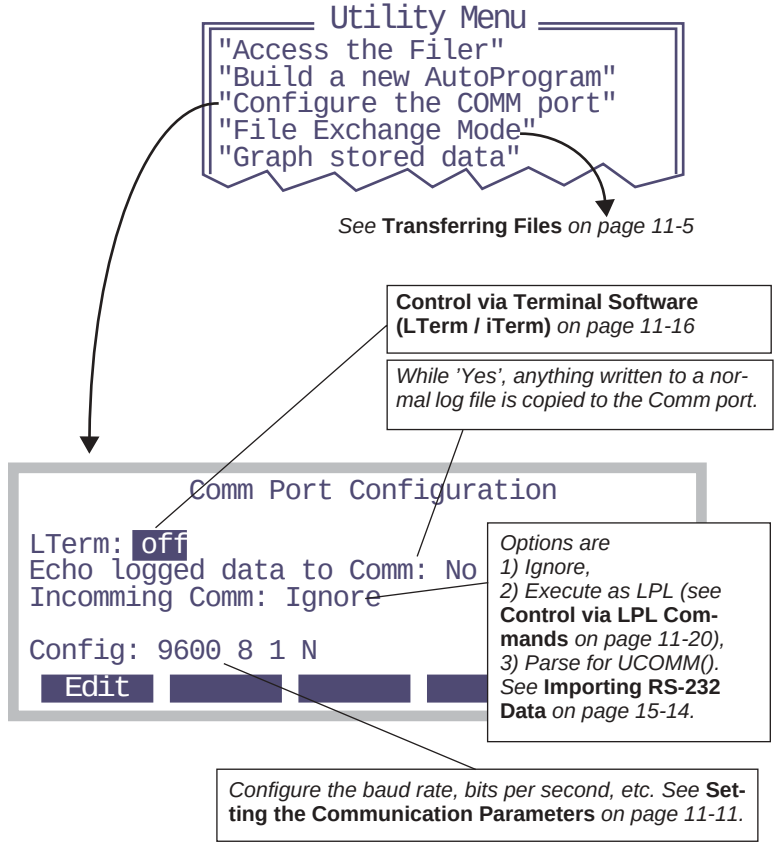


Figure 11-2. Most RS-232 tasks begin in the Utility Menu.

Transferring Files

Files can be moved to and from the LI-6400 with File Exchange Mode, which is described below. It can also be done with one of the remote control options (**Control via Terminal Software (LTerm / iTerm)** on page 11-16.), or with a generic data capture program (**Using a Data Capture Program** on page 11-10).

File Exchange Mode

The simplest way to move files back and forth is to put the LI-6400 into File Exchange Mode, and run a file exchange (FX) program on your computer. There are two ways to get the LI-6400 into File Exchange mode:

1 **From the LPL Copyright Screen**
Press **X.**

2 **While OPEN is running**
Access the Utility Menu, and select "File Exchange Mode".

Note: When LTerm (**Control via Terminal Software (LTerm / iTerm)** on page 11-16) is active, File Exchange Mode is unnecessary, and is disabled.

FX for Windows: LI6400FileEx

LI6400FileEx is a Windows program that interacts with the LI-6400 while it is in File Exchange Mode. An installer program is located on the 6400-55 CD. The latest version can also be downloaded from www.licor.com.

The installation program will create an entry named LI6400_5.3 in the Programs menu. Within that, you will find LI6400FileEx. There will also be a shortcut on your desktop that you can double-click.

When first run, LI6400FileEx will show empty windows for the LI-6400's directory tree and file list. If you are using Comm port 1, press

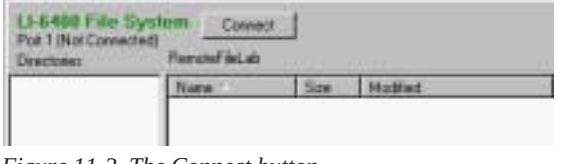


Figure 11-3. The Connect button.

Connect. Otherwise, select Prefs from the file menu, then select the correct Comm port, and press OK. Then click the Connect button.

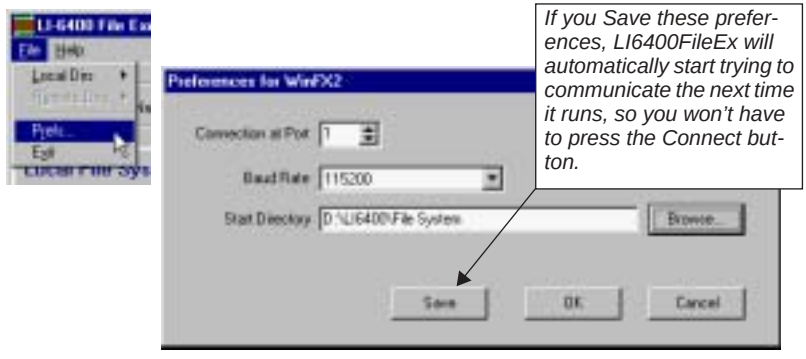


Figure 11-4. Setting the Comm port.

The program provides a view of the file system of your PC, and also the LI-6400. You can expand or contract the directory view by clicking on the node boxes that have a + or - in them (Figure 11-5).

Moving files is as easy as opening the desired directory, and clicking on the desired files, and dragging them to the other file list or directory. Complete documentation is available under the Help menu.

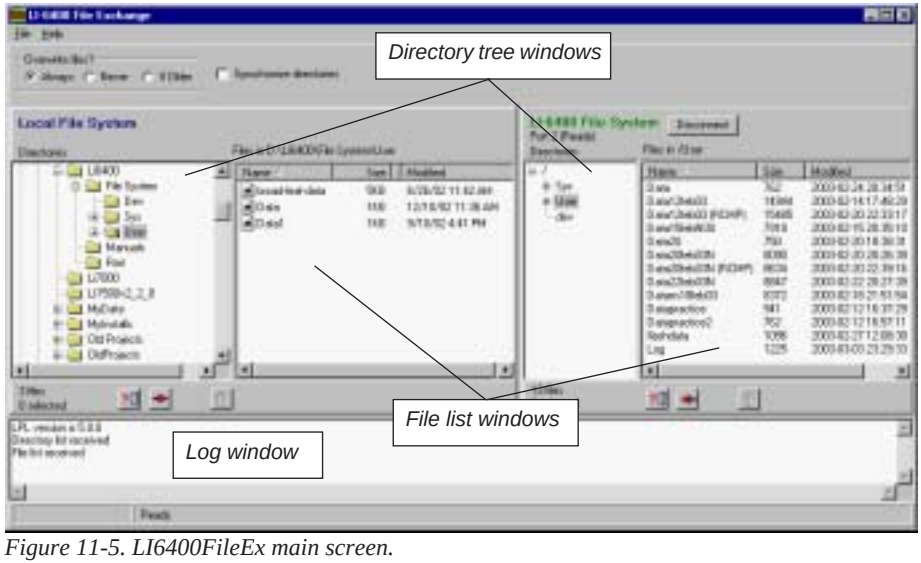


Figure 11-5. LI6400FileEx main screen.

FX for Macintosh: FX

FX for Macintosh, as of this writing (version 1.6), only works on operating systems 9.x and below.

FX for Macintosh is available on the 6400 CD, and from www.licor.com. The installation program will create a directory on your hard disk named FX 1.6, which contains the application. If you have a 680x0 processor, this application is named FX 1.6. On a PowerPC, it is named FX 1.6 PowerPC.

To transfer files using FX on a Macintosh

1 Connect the cables
Connect the one end of the 9975-016 cable to the LI-6400. Connect the other end to either a Serial/USB adapter or a Hayes compatible modem cable (Figure 11-1 on page 11-3).

2 Launch FX
Double click the FX application icon.

3 Choose the Port
Click Ports in the menu bar. A list of serial ports will appear. Select the one you are using.

4 Establish communications
Put the LI-6400 into File Exchange Mode (page 11-5), and click on the Connect button. Once the connection is established, you will be able to navigate through the LI-6400 file system (Figure 11-6).

5 Tag items to be copied
Tag files (or entire directories) that you wish copied.

6 Set the direction arrow
Make sure the direction arrow is aimed in the desired direction.

7 Click Send
The Send dialog allows you to specify text or binary (or autodetect), what to do with duplicate file names, among other things.

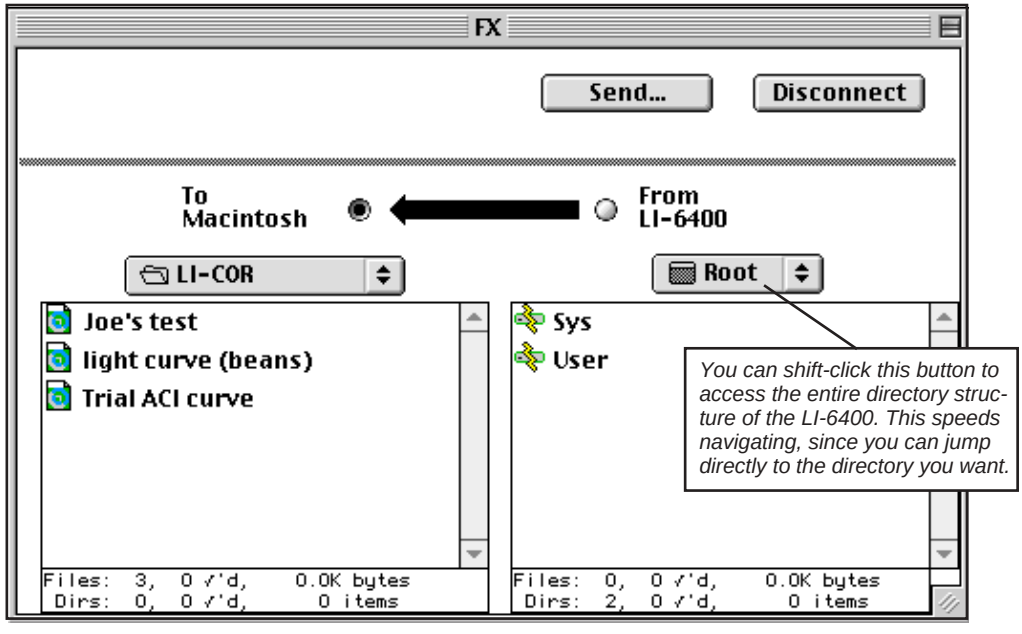


Figure 11-6. FX/Macintosh shows files and directories on both the LI-6400 and the Macintosh. Direction of file transfer is set by the arrow direction.

Text vs. Binary Files

Most of the data and all of the LPL program files that you will ever encounter on the LI-6400 are text files. The exception is graphics images (see **RTG Image Files** on page 6-17 and **Storing and Retrieving Graphics Images** on page 12-20). We call non-text files binary files. The difference between how the two types are transferred comes down to end-of-line sequences in text files.

The preferred end-of-line sequence in text files is platform specific:

Table 11-1. Preferred end-of-line sequences for text files.

Platform	End of Line
LI-6400	<lf>
Windows	<cr><lf>
Macintosh	<cr>

When an FX application moves text files, it will convert end-of-line sequences from or to the preferred native environment, depending on the direction of transfer. (This is the reason that file size may differ after text files are moved by an FX program from the LI-6400 to a computer.)

When moving binary files, you don't want any end-of-line conversions going on. Therefore, to move binary files:

- **FX/Mac**
  The dialog box that comes up when you press Send... contains an option for picking Text, Binary, or AutoDetect. Pick Binary (or AutoDetect).

- **LI6400FileEx**
  The program auto-detects binary files, and transfers then accordingly.

File Name Considerations

Macintosh

When using FX/Mac, you won't have to worry about file name incompatibilities if you avoid these characters

```
: ; / \ ? *
```

and keep your file names under 32 characters in length. When naming files on the LI-6400, the Standard File Dialog will prevent you from entering any of these problem characters. If you use one of these characters when naming a file on the Macintosh that is to be transferred to the LI-6400, that character will be changed to an underscore at the time of transfer. If an LPL file name is too long for the Macintosh (32 characters or more), it will be shortened, and if it contains a colon (:), which is the directory delimiter on the Macintosh, it will be converted to an underscore. Note that these changes are not remembered, so if you begin with a file on the LI-6400 named

```
abc:def:ghi
```

and transfer it to the Macintosh, it will be named

```
abc_def_ghi
```

If you transfer it back to the LI-6400, it will still be named

```
abc_def_ghi
```

Case

The LI-6400 version 5 software uses a case-sensitive file system. Windows and Macintosh does not. This generally is not a problem, but may lead to some surprises. For example, you may have two files on the instrument whose names only differ by case. Data and DATA, for example. If you transfer these to your computer, the second one transferred may overwrite the first, if you are allowing file overwriting, or not be transferred at all.

Using a Data Capture Program

Any generic serial communications program can be used to capture LI-6400 data files. One such program is HyperTerminal, which is bundled with Windows operating systems. This method has three basic steps:

1 **Set the communications parameters**
Described below.

2 **Open a destination file**
Once you've configured the LI-6400's communications parameters to match those used by your data capture program, you'll 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 10-18).

Setting the Communication Parameters

The baud rate, data bits, stop bits, parity, and hardware handshaking can be set by selecting "Configure the COMM port" in the Utility menu. Alternatively, run the program "/Sys/Utility/SetComm" from the Filer (Figure 11-7).

Figure 11-7. The program SetComm allows you to view and edit the communications settings.

Table 11-2 shows the acceptable values of these parameters, and Table 11-3 illustrates some sample configurations.

Table 11-2. 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 11-3. 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

Remote Control

There are three methods to remotely control an LI-6400:

- **Simulation Software**
  The computer acts like an LI-6400, because it is running a special version of the LI-6400's software built for the computer's OS. Meanwhile, the real LI-6400 runs a slave program, and cannot be controlled from its front panel. This method is described below, and can be used on any LI-6400, regardless of its software version.

- **Terminal Software**
  The computer acts like a terminal to the LI-6400, providing all of the normal front-panel capability locally or even over the internet. Meanwhile, the LI-6400 continues to function normally. File transfers in the background are possible with this method, which is described in **Control via Terminal Software (LTerm / iTerm)** on page 11-16. The LI-6400 must have version 5.3 (or above) software.

- **Send LPL Commands**
  The LI-6400 (version 5.3 or above) can be made to compile and execute incoming LPL commands while OPEN is running. This method lets the instrument operate normally, but 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 the value of leaf areas, and trigger fluorescence and logging events. This method is described in **Control via LPL Commands** on page 11-20.

Control via Simulation Software

The computer runs a program that simulates an LI-6400. It can function with or without an actual instrument. If you connect to an LI-6400, the instrument runs in a "slave mode", getting actual readings from the IRGA and sensors, but all the computations are done on the computer. All the files (data, autoprogram, etc.) also reside on the computer.

This technique can be used to operate the latest software (e.g. 5.3) on the computer, while the LI-6400 can have any version (3.x, 4.x, 5.x) of software. Thus, a 3.x version of LI-6400 can operate with a 6400-40 LCF, which otherwise requires the host instrument to have version 4.x or above.

One implication of using a simulator is that control must be done from the computer. While the instrument is being controlled, there is nothing you can do on its keyboard or see on its front panel. Another implication involves file synchronization. Because the simulator is only aware of local (your comput-

er) files, you'll want to be sure that the calibration data files (/dev/parm0 and /dev/parm1) on your computer match those in your instrument.

Simulation software is available for both Macintosh and Windows.

LI6400Sim (Windows)

An installer for the Windows LI-6400 simulator is on the 6400-55 CD. The latest version is available at www.licor.com. A manual (.pdf file) for the program is on the CD, and is also installed with the program and viewed via Help in the program's menu bar.

To control an LI-6400 using LI6400Sim_5.3, do the following:

1 **Run the application**
The installer puts a shortcut to LI6400Sim_5.3 on the desktop, and an entry in the Start menu under Programs -> LI6400_5.3.

2 **Connect your PC to the LI-6400**
See Figure 11-4 on page 11-6.

3 **Put the LI-6400 into File Exchange Mode**
Described in **File Exchange Mode** on page 11-5.

4 Access the Remote panel

Open the Control window, and access the Remote Panel (Figure 11-8). Set the "Using COM" value to match the comm port you are using.

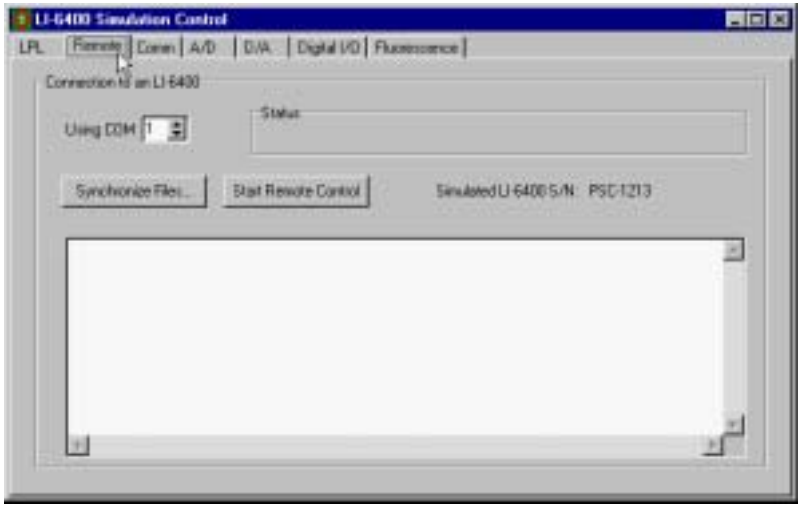


Figure 11-8. The Remote Control Panel.

5 (Optional) Synchronize files

If the "Simulated LI-6400" serial number doesn't match the real unit's serial number, you may want to synchronize files. See the LI6400Sim manual for more details.

6 Start Remote Control

Click on the "Start Remote Control" button. The application will download a server program to the instrument, if it is already there, and run it.

Wait until the Remote Control Status indicates Active.

7 Power on the Simulated Console

Click Start in the Console window.

LPL 5.3 PPC (Macintosh)

An installer for the Macintosh LI-6400 simulator (PowerPC processors only) is on the 6400-55 CD. The latest version is available at www.licor.com. The program is documented in a text file found in the folder (LPL/Mac 5.3) that is installed.

To control an LI-6400 using LPL 5.3 PPC, do the following:

1 **Run the application**
It will be in the folder named LPL/Mac 5.3.

2 **When the countdown starts, click on or press escape**
Don't let the simulation run, but press escape, leaving you in the simulator's LPL screen.

3 **Connect to the instrument**
Described in **Connecting the LI-6400 to a Computer** on page 11-2

4 **Put instrument in file exchange mode**
See Figure 11-4 on page 11-6.

5 **Open Remote Device window**
Click on Remote in the menu bar, and select Use Remote Device. Check "Get remote device's calibration data" if this is the first time you have controlled this particular instrument, or if you have controlled some other instrument the last time.

Click the **UseRemote** button.

The application will download a server program if necessary. Wait until the status says "active".

6 **Run OPEN on the simulator**
Press R in the simulator's LPL screen, to run a file. The file you want to run is /Sys/Autostart/Welcome, which will be the default, so just press **Return**.

Control via Terminal Software (LTerm / iTerm)

LPL version 5.3 introduces a powerful new method for remote control of the LI-6400. The local component of this is named **LTerm**, and the internet component is named **iTerm**. When **LTerm** is enabled on an LI-6400, the instrument can be controlled, and its display viewed, from a computer via the Comm port. The computer runs an **LTerm**-compatible program that communicates in the background with the LI-6400; all the while, the LI-6400 continues to operate normally, and can continue to be controlled from its own keyboard.

iTerm extends this control to anywhere in the world via the internet (Figure 11-9). Two programs with **iTerm** capability connect through a special server (li6400.licor.com), and pass information back and forth. The remote internet connection requires no knowledge of any network configuration parameters and is completely firewall transparent. The local computer passes along the iTerm information to and from the LI-6400.

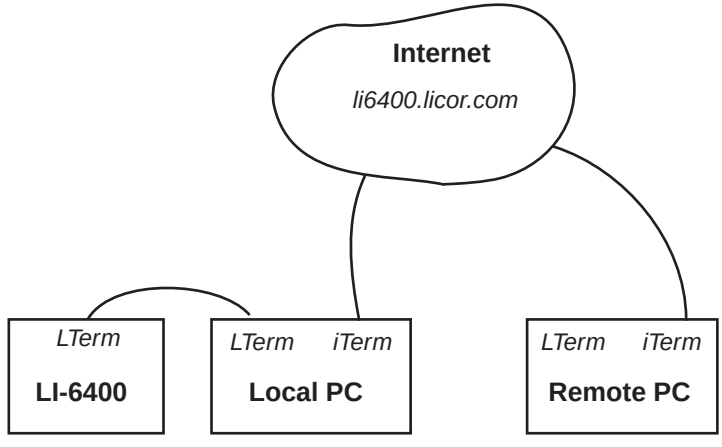


Figure 11-9. The LI-6400 can be viewed and controlled from its front panel and from a local computer and from a second computer via the internet.

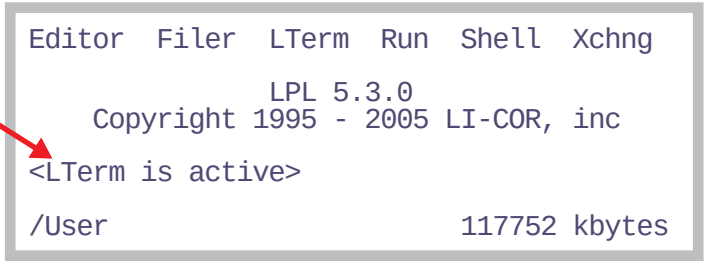
LI6400Term (Windows)

There is a Windows application named LI6400Term that supports LTerm and iTerm, and it is available on the 6400-55 CD. The latest version is available from www.licor.com. Once installed, begin controlling by these three steps, which can be done in any order.

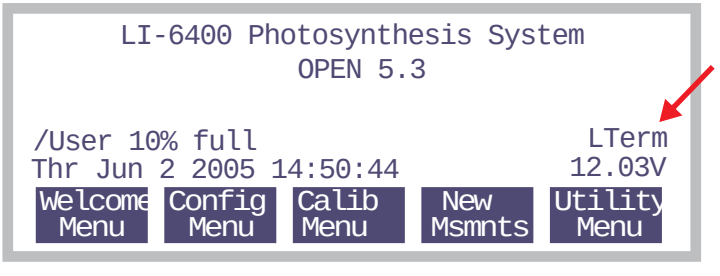
1 **Connect the Computer to the LI-6400**
Described in **Connecting the LI-6400 to a Computer** on page 11-2.

2 **Start LTerm on the LI-6400**
Pressing the **L** key from Open's main screen (Figure 11-10). When LTerm is active, the label LTerm will appear in the lower right part of the display. LTerm can also be turned on and off the LPL screen, and from Open's Utility Menu.

LPL Screen



OPEN's Main Screen



OPEN's Utility Menu (Configure the Comm Port)

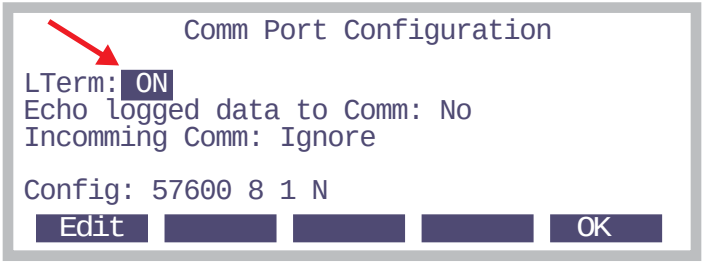


Figure 11-10. From either the main screen of OPEN (above) or the LPL screen (left), LTerm can be toggled on and off with the **L** *key. It can also be accessed from OPEN's Utility Menu*

3 Enable LTerm on the computer

Launch LI6400Term, click Windows in the program's menu bar, and select Connection. In the Local tab, select the appropriate comm port, and click Start (Figure 11-11).

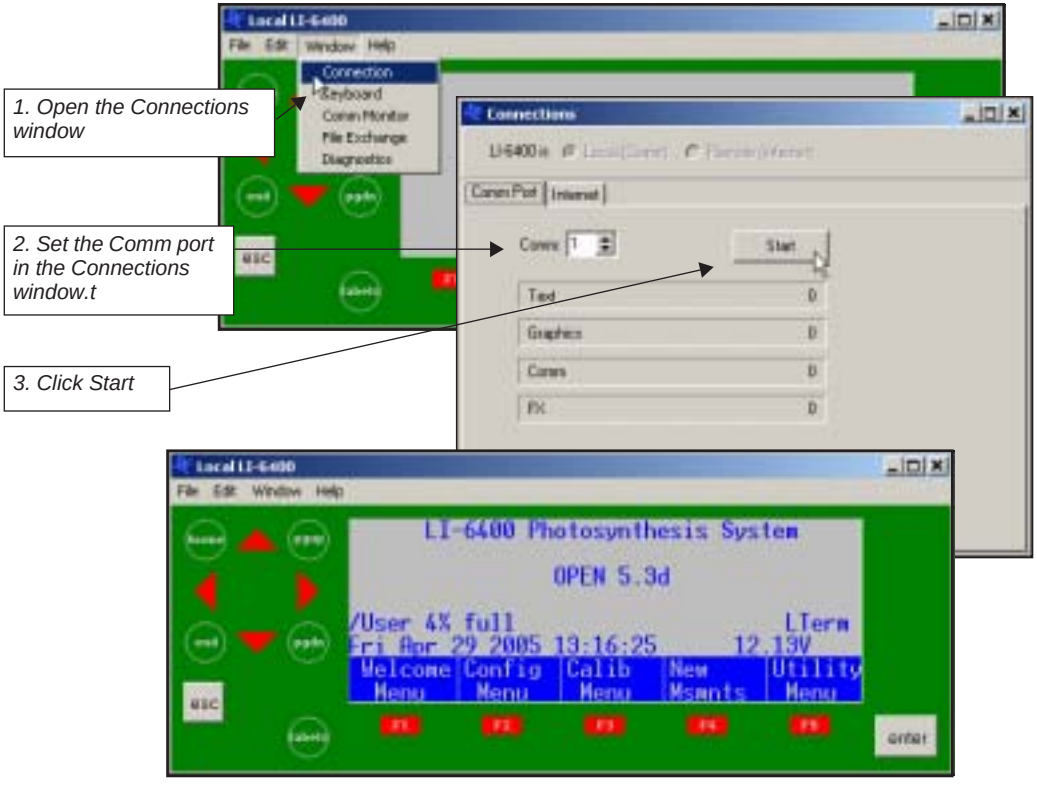


Figure 11-11. The Windows program LI6400Term with a local (LTerm) connection to an LI-6400.

At this point, you will see the LI-6400 display on LI6400Term's Console window, and be able to control the instrument from either the front panel or the computer's keyboard.

Further details about the operation of LI6400Term, including internet connections and file transfers, can be found in its manual, available under Help in the program's menu bar.

Effect of LTerm-iTerm on Normal Operation

For the user looking at the console of the LI-6400, LTerm and iTerm have only two visible effects on normal operation of the instrument (aside from the fact that the instrument is responding to "unseen" commands and keystrokes):

1 File Exchange Mode is disabled while LTerm is active

When using LTerm, file exchange mode is unnecessary, since files can be moved in the background. If you try to enter file exchange mode, you will instead see the LTerm Monitor display (Figure 11-12.), which can be exited by pressing **escape**.

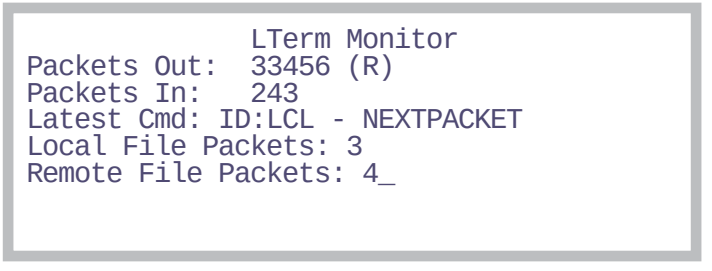


Figure 11-12. LTerm Status mode is shown instead of File Exchange mode while LTerm is active.

2 The Comm configuration cannot be changed while LTerm is active

From OPEN's Utility Menu, the "Configure the Comm port" item will show the Comm port's configuration, but won't let you change it while LTerm is active. Note that LTerm can be toggled on and off from this screen.

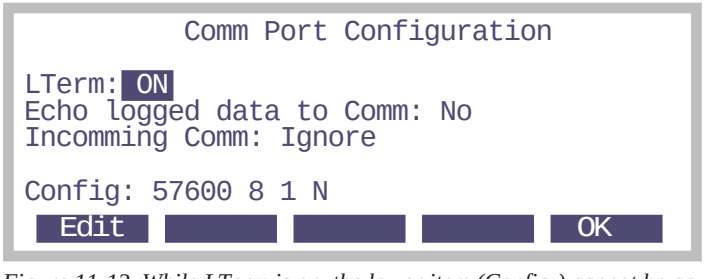


Figure 11-13. While LTerm is on, the lower item (Config:) cannot be accessed and changed.

There is a little utility program ("/Sys/Utility/SetComm") that is used by other programs to set the Comm port. If you run it, or cause it to run, while LTerm

is active, it will show the configuration, but not let you change it (Figure 11-14).

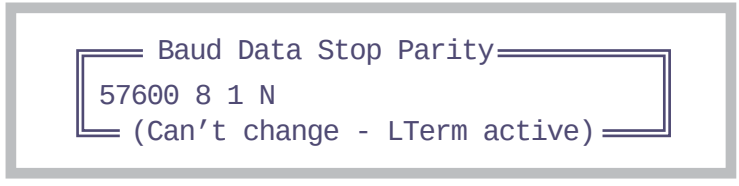


Figure 11-14. The program SetComm will not allow config changes while LTerm is active.

The Comm port configuration can always be changed directly through LPL code (keyword **COMMCONFIG** on page 24-26), but that will potentially disable LTerm until it is turned off and back on again.

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. Perhaps you want to get an occasional value of CO_2 or photosynthesis, or you want the computer to tell the instrument to start or stop an autoprogram. Is there a way to communicate short of LI6400Term or LI6400Sim? There has to be, or this chapter would end right here.

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 11-15).

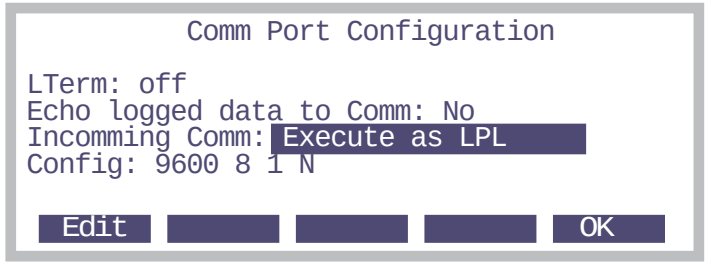


Figure 11-15. The Comm Port Configuration screen, from OPEN's Utility Menu, set for compiling and executing incoming LPL Commands.

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 14-8 on page 14-19 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 5 digits with no other label, you can send

```
area_cm2 "%1.5f\n" comm print
```

to receive

```
1.23456
```

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 your setting a value may not 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 11-23.

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 =".

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 25-17 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 27-73, 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 LI6400Term or LI6400Sim.

- **Use LI6400Term and an LI-6400**

  Establish the LTerm connection (**Start LTerm on the LI-6400** on page 11-17). Then, enable the incoming LPL option (Figure 11-15 on page 11-20) 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 11-16).

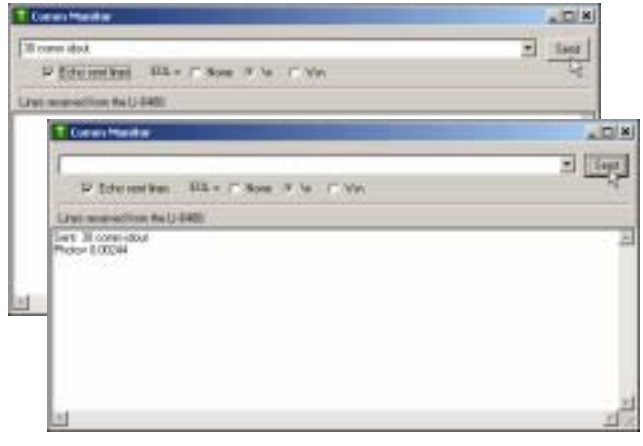


Figure 11-16. Using the Comm Monitor window of LI6400Term to test LPL commands.

- **Use LI6400Sim by itself**
  Using the Windows simulator, enable the incoming LPL option (Figure 11-15 on page 11-20) 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 11-17).

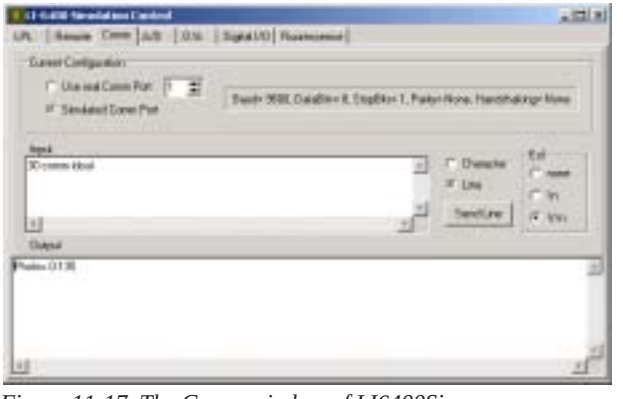


Figure 11-17. The Comm window of LI6400Sim.

12

GraphIt

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 labelled columns, plot selected variables, and do polynomial curve fitting.

An introductory tour of GraphIt can be found in **Viewing Stored Data** on page 3-52.

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 Stored Data" 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 Stored Data" 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.

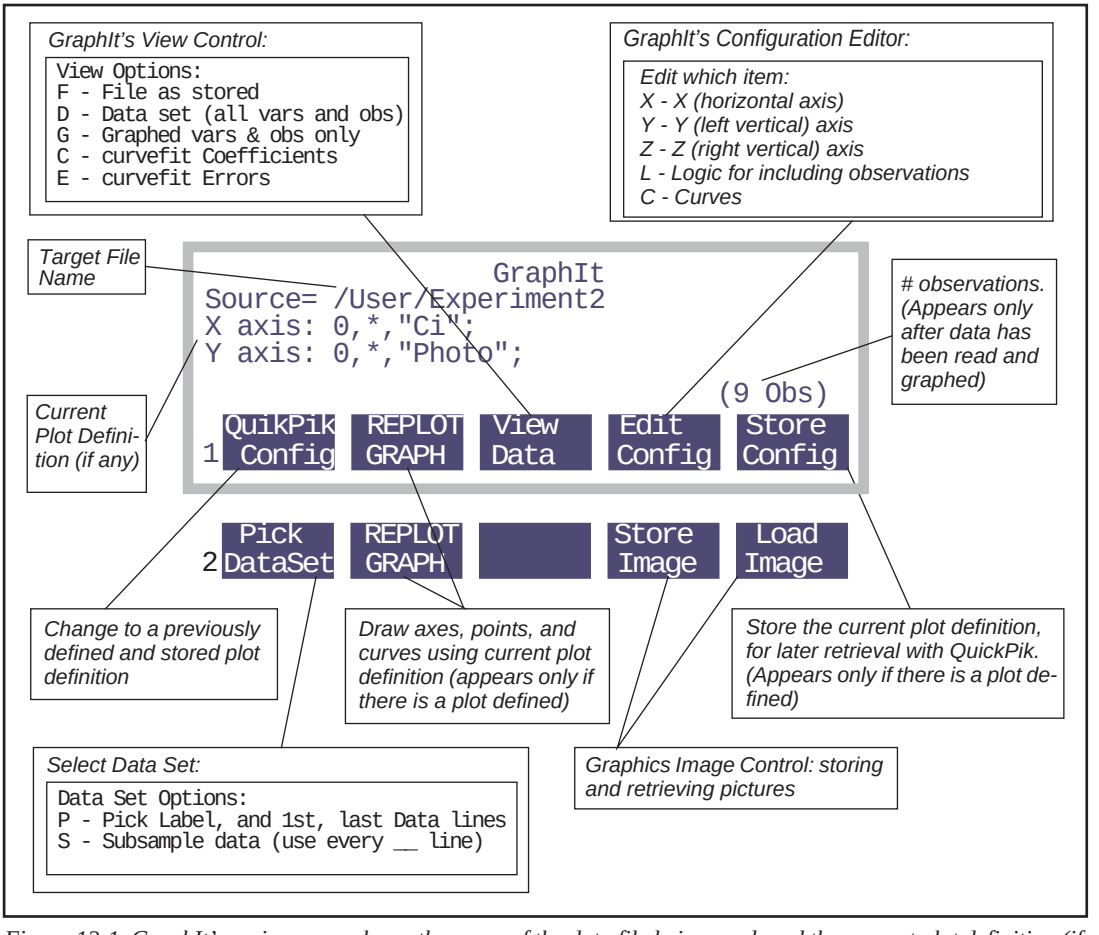


Figure 12-1. GraphIt's main screen shows the name of the data file being used, and the current plot definition (if any). There are 2 levels of function keys defined, and one can toggle between them by pressing **labels** *or* **1** *or* **2**. *Descriptions of GraphIt's functionality begin with* ***Defining 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 1 row (line) is 1 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.

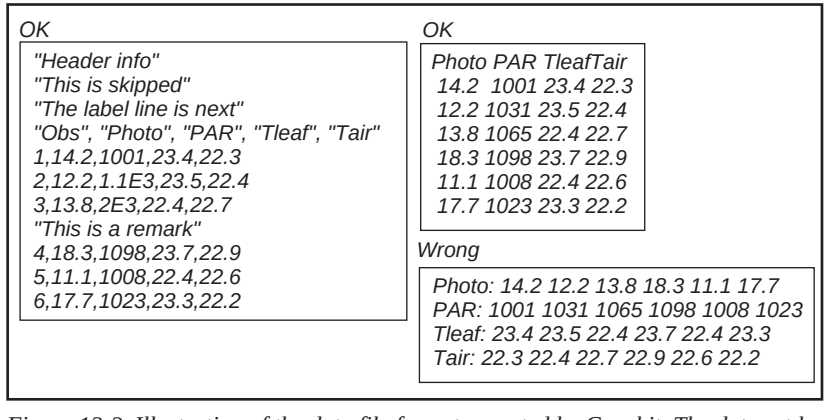


Figure 12-2. Illustration of the data file format expected by Graphit. The data set labelled "Problem" is just that because the first thing that appears in the data line is a double quote. (OPEN 3.2 and above handles this.)

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 1 to 5 variable names (Figure 12-3).

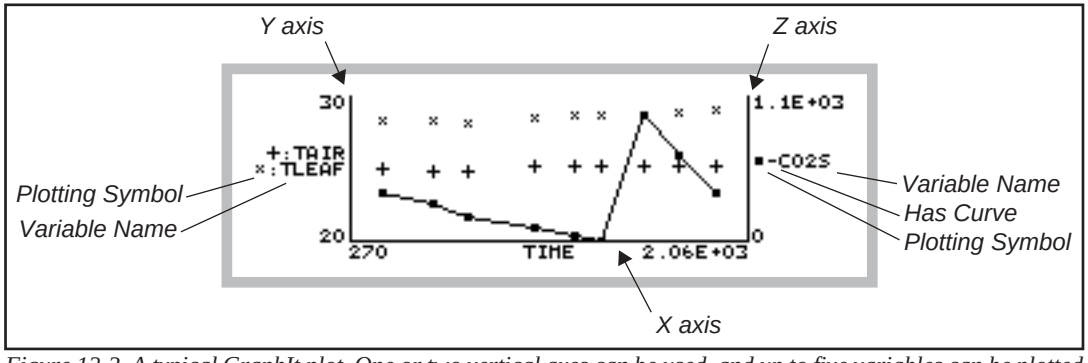


Figure 12-3. A typical GraphIt plot. One or two vertical axes can be used, and up to five variables can be plotted on each axis. Curve fitting options consist of dot-to-dot (as shown), or polynomials.

There are two methods of defining a plot: the quick method is to select a previously defined and stored definition from a menu (press **QuikPik Config**). The general method is to use the plot configuration editor (press **Edit Config**).

Using QuikPik Config

This is quite simple, as the following example illustrates.

■ **To configure for an A-Ci curve using QuikPik Config:**

1 **Press QuikPik Config**
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!).

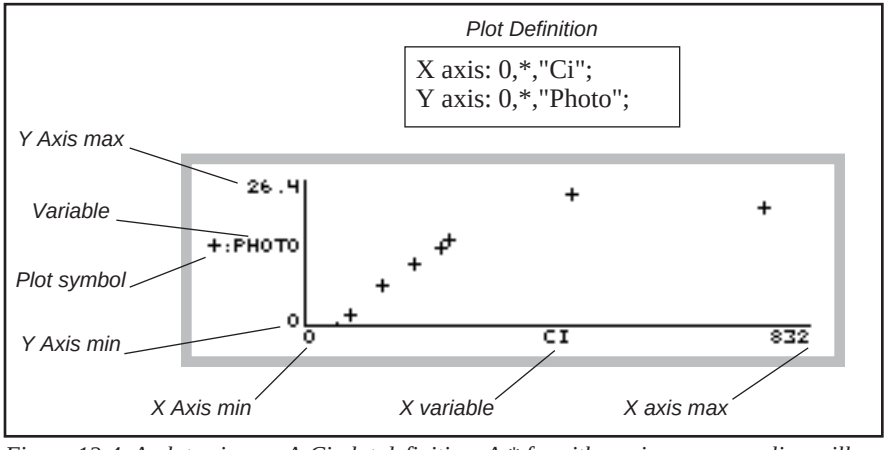


*Figure 12-4. A plot using an A-Ci plot definition. A * for either min or max scaling will result in that min or max value being computed based on the data.*

Using Edit Config

To define a graph, or to modify the current configuration, press **Edit Config**. The edit menu will appear:

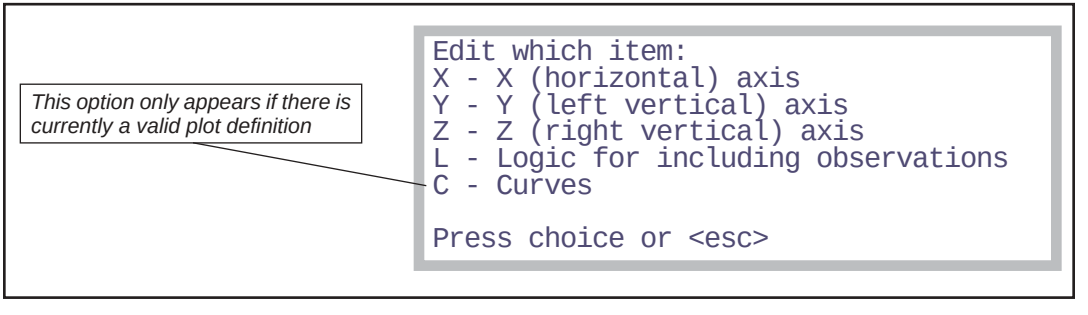


Figure 12-5. GraphIt's configuration editor's main window. Note: you can by-pass this window by typing **X**, **Y**, **Z**, **L**, *or* **C** *directly from GraphIt's main screen.*

Editing Axis Definitions

Pressing **X**, **Y**, or **Z** will allow you to edit the definition of that axis. The display will show

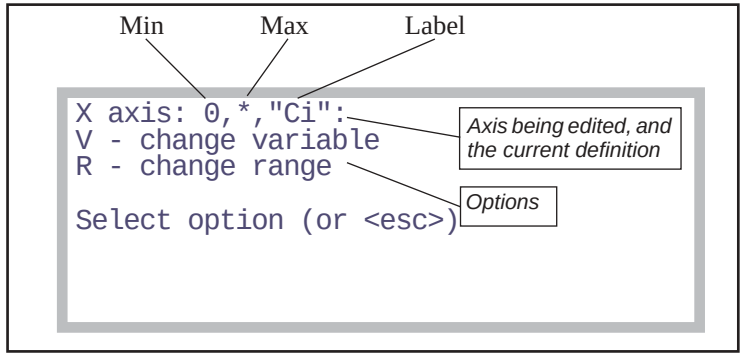


Figure 12-6. Editing an axis has two parts, that can be done independently: selecting the variable(s), and selecting the scaling.

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 1 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.

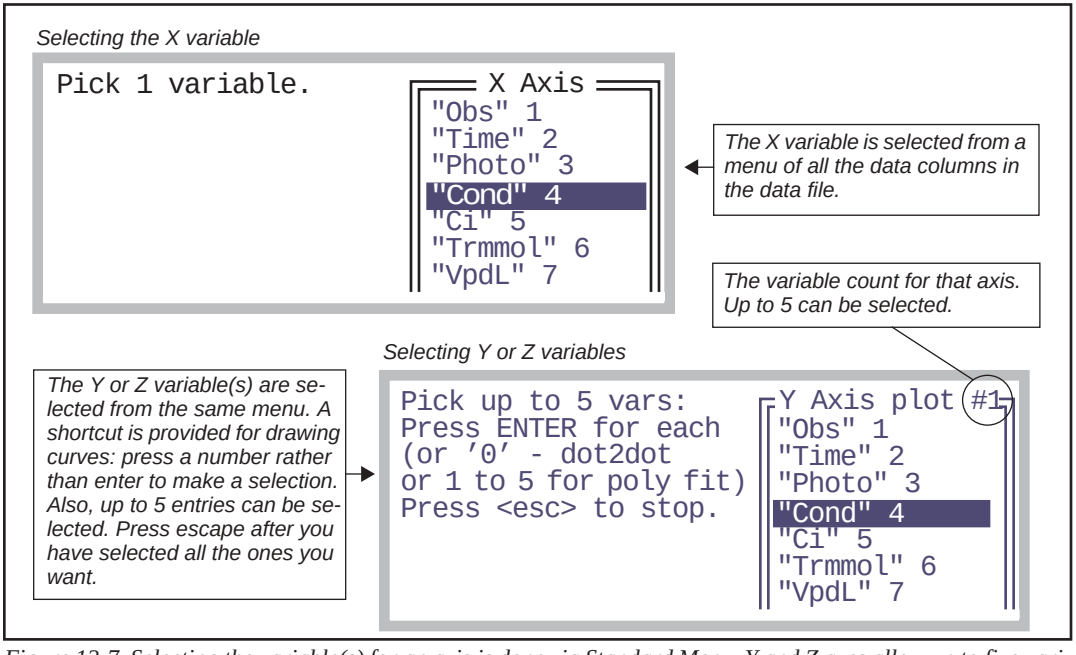


Figure 12-7. Selecting the variable(s) for an axis is done via Standard Menu. Y and Z axes allow up to five variables to be included, and provide a short-cut for doing curves. When you've selected all the variables you want, press **escape**.

- **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.

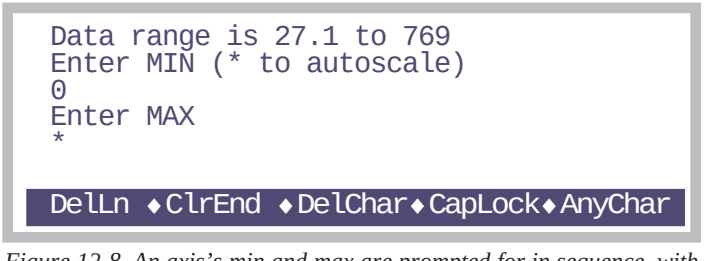


Figure 12-8. An axis's min and max are prompted for in sequence, with the default value or response displayed.

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 **Store Config** function key (**f5** level 1) allows you to store the current plot definition for later recall using the **QuikPik Config** function key (**f1** level 1). Definitions should be stored in the directory /User/Configs/PlotDefs. The directory and file name are obtained from you by GraphIt 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 nth 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 "**Data Set Pick**" (**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).

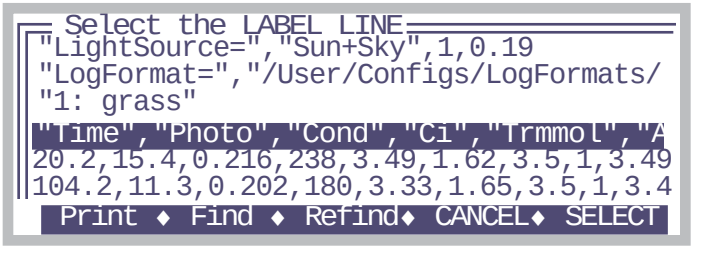


Figure 12-9. Prompting for the Label Line. Put the highlighted bar on the label line, and press **enter**. *When selecting the data lines, the prompts will be "Select the FIRST DATA LINE" and "Select the LAST DATA LINE"*

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 µmol m^{-2} s^{-1}.

Inclusion logic is defined by pressing "**Edit Config**" (**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.

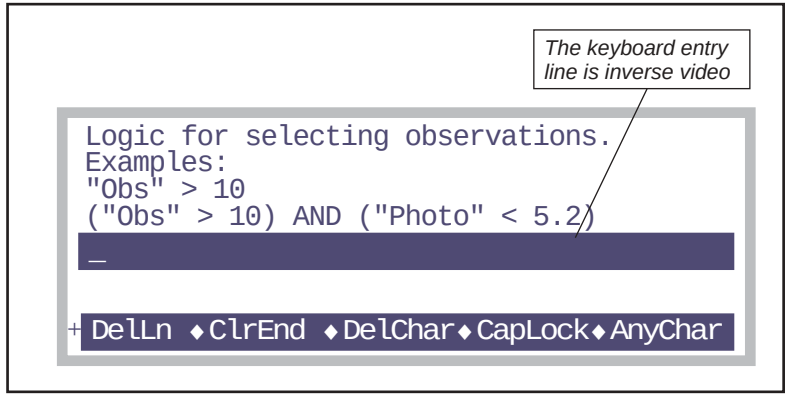


Figure 12-10. Entering selection logic. Put data set variable names in quotes.

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.

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,
<= >=	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, so, 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 (of no use at all) 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 1 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 4 values is sufficient data. Therefore, be aware, that if your data set includes a labels line, then it will likely be considered as data.

[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.

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 Config**, then **C** (short cut: just press **C**). If only 1 Y or Z variable is defined, the curve is simply selected from a menu (Figure 12-11).

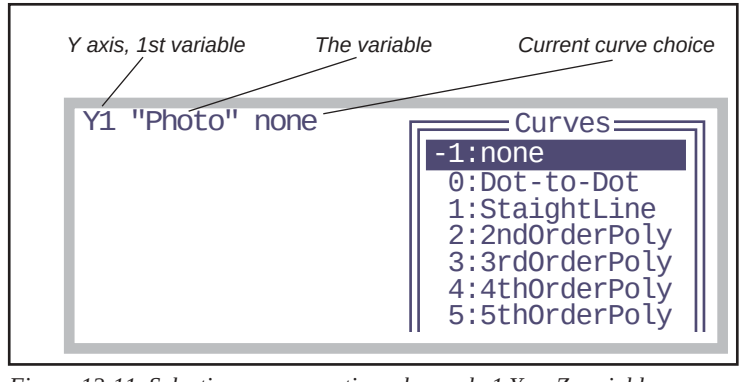


Figure 12-11. Selecting a curve option when only 1 Y or Z variable.

If multiple Y and/or Z variables are defined, then it is a two part process (Figure 12-12).

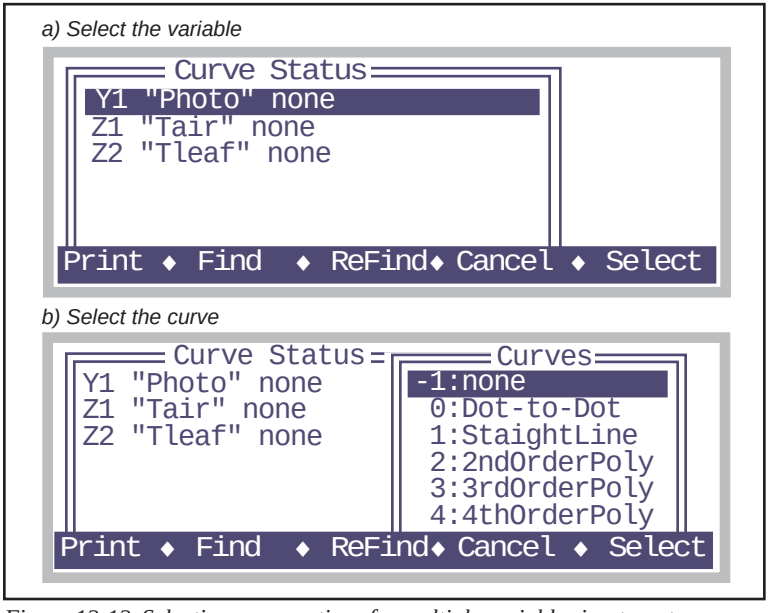


Figure 12-12. Selecting curve options for multiple variables is a two step process: a) pick the variable, and b) pick the curve.

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 **QuikPik Config** or **Edit Config** to set up a plot of photosynthetic rate as a function of intercellular CO_2.

2 Select a straight line curve

Press **Edit Config**, then **C**, and select a straight line as the curve type.

3 Ignore the higher Ci values

While in the config 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.

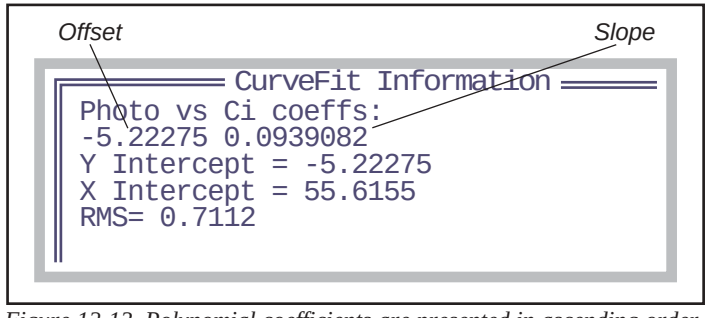


Figure 12-13. Polynomial coefficients are presented in ascending order of power. X intercept(s) are computed for 1st and 2nd order polynomials.

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 become 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 Config** (**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.

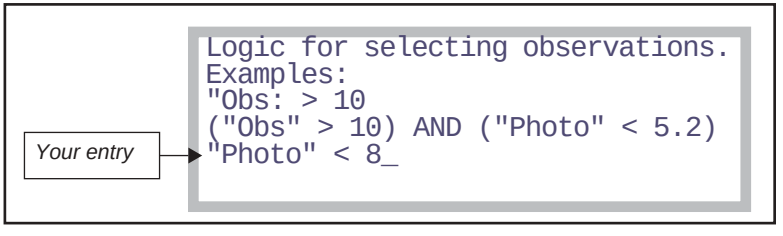


Figure 12-14. A Ci Curve init slope edit config logic screen. Enter the maximum photosynthesis rate on the linear part of the curve.

4 Fit a Curve

To fit a curve to this reduced data set, press **Edit Config** and select **C** for curves. Choose "1: Straight line" from the list, and press **enter**. When you exit the Edit Config menu, 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
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.

D - Data set (all vars and obs)

This option views the currently selected data set in column format (Figure 12-16). The observations reflect the first and last, and 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).

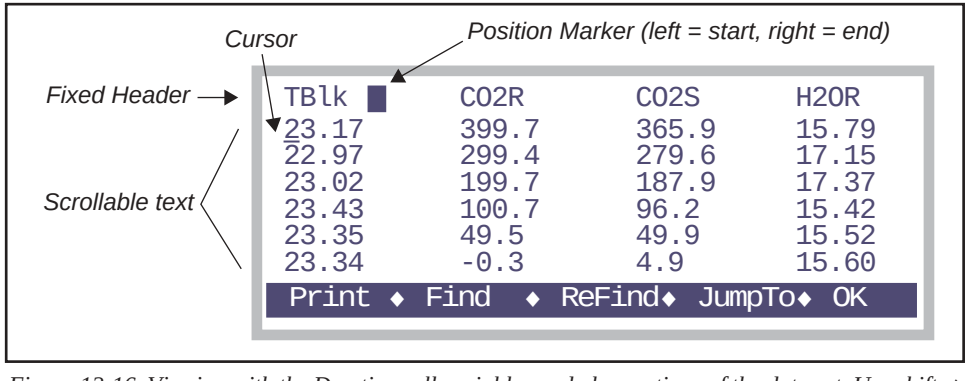


Figure 12-16. Viewing with the D option: all variables and observations of the data set. Use shift→ and shift←to scroll left and right 4 columns at a time. An inverse box in the fixed header indicates relative position in the data of the cursor.

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.

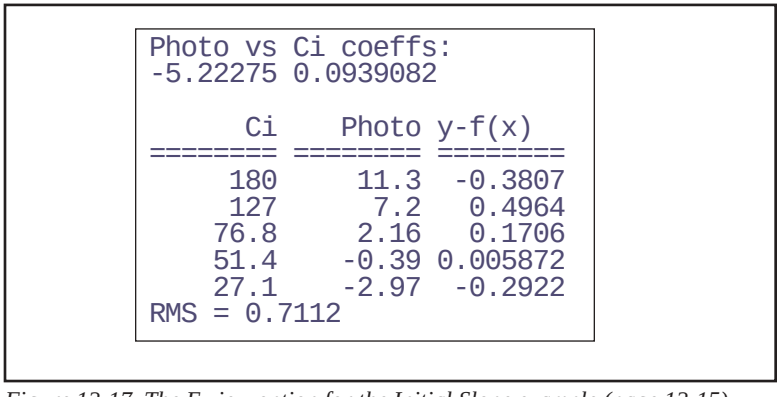


Figure 12-17. The E view option for the Initial Slope example (page 12-15).

As with the C option, you are given the option of storing the information to disk when you exit from viewing it.

PlotDef File Format

Figure 12-18 illustrates the format of a plot definition file. These files are expected in, but not confined to, /User/Configs/PlotDefs

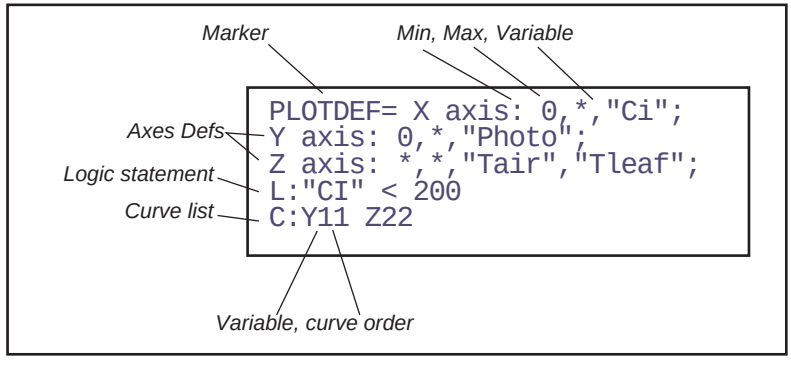
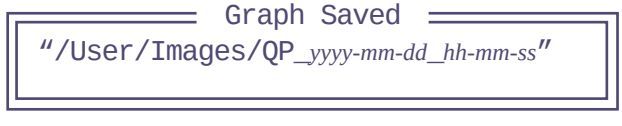


Figure 12-18. An example plot definition. All entries are optional, except the Marker (plotdef=).

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 an FX (Chapter 11), be sure to treat them as binary files (**Text vs. Binary Files** on page 11-9).

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:



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 done 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 12-19).

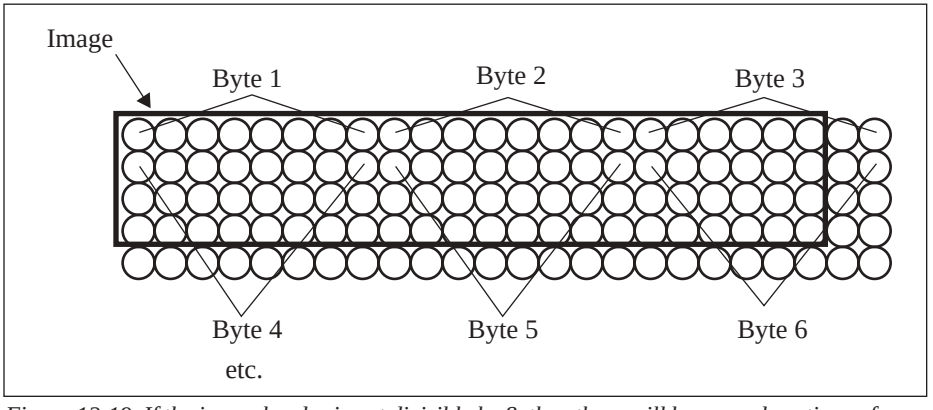


Figure 12-19. If the image border is not divisible by 8, then there will be unused portions of some of the bytes. Note that the image length and width must be preserved, or else the image will be shuffled.

13

Recomputing Data Files

How to recompute data files

13 Recomputing Data Files

Sooner or later, you'll find that you've collected some data and had something set incorrectly, such as leaf area, and need to recompute the file. This chapter describes how to accomplish this task.

Note: A convenient method of recomputing data is to use the Windows program LI6400Sim. This simulator provides a more familiar spreadsheet environment for recomputing data files. LI6400Sim is available on the 6400-55 CD, and from www.licor.com.

Reasons to Recompute

- **To Change Leaf Area**
  This is the most common reason for recomputing data. However, it's not just leaf area that can be changed, but any of the user entered constants (stomatal ratio, boundary layer conductance, etc.).

- **To Change What's Stored**
  You can select a subset of a data file, such as only three variables, to be output to a different file.

- **To Change What's Computed**
  You can add quantities to be computed by changing the Compute List file. Obviously, the computed quantities should depend only on things that are available in the original file.

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

Go to OPEN's Utility Menu, highlight "Recompute stored data", and press **enter** (Figure 13-1).

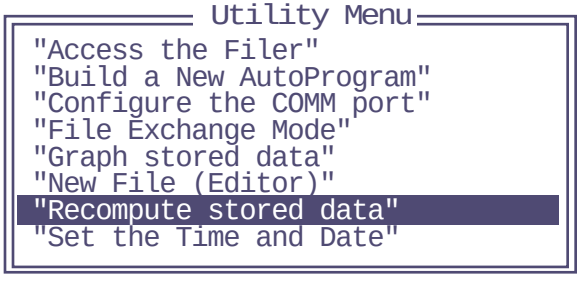


Figure 13-1. OPEN's recomputation utility is found in its Utility Menu

2 Specify the source file

Highlight "SourceFile:", and press **f1** (**edit**) (Figure 13-2).

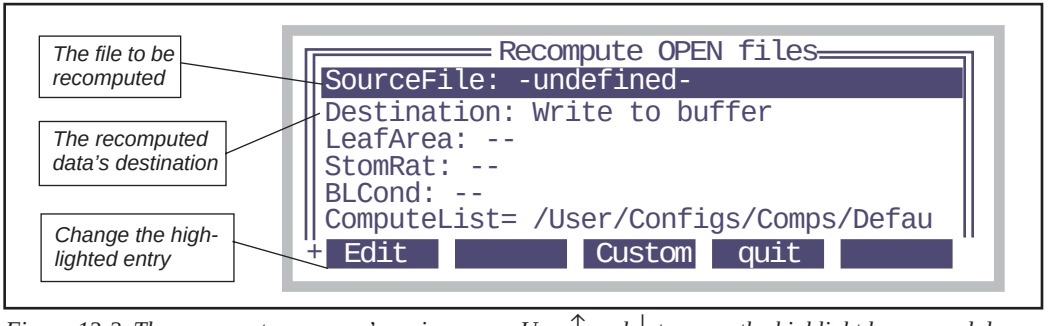


Figure 13-2. The recompute program's main screen. Use ↑ and ↓ to move the highlight bar up and down, and press edit to change the highlighted entry.

The Standard File Dialog will appear. Highlight the file "MyData", and press **enter** (Figure 13-3).

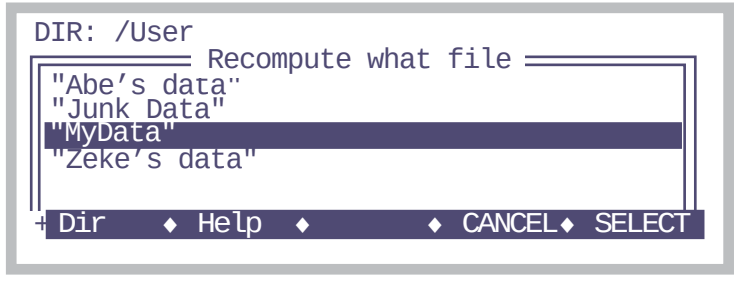


Figure 13-3. The Standard File Dialog is used to select the source file.

3 Select ComputeList and LogFormat Files?

If the current compute list file and/or log format file differ from the ones used in MyData (or whatever your file is named), then you will be asked to select between the two. This is discussed in **Source File Considerations** on page 13-8.

If the current compute list and log format are the same as the ones used in MyData, you are not asked this.

4 Select the destination

Highlight "Destination:" and press **Edit** until the line says

```
Destination: Write to file
```

The choice is between a "File" and "Buffer" (memory). "File" is generally the better choice, unless you aren't sure you really want to store the result. When you have destination set to a file, you are prompted for that file name when you actually start the recompute, which we'll do in a few more steps.

5 Specify a change in leaf area

Highlight the "LeafArea:" line, and press **Edit** (Figure 13-4). If all of the observations in the file should have the same leaf area, select "One global value". If you wish to be prompted as the file recomputes for each new leaf area, then select "Prompt for new values".

For our example, we'll highlight "One global value" and press **enter**. We are then immediately asked for that value, and we'll enter 4.4.

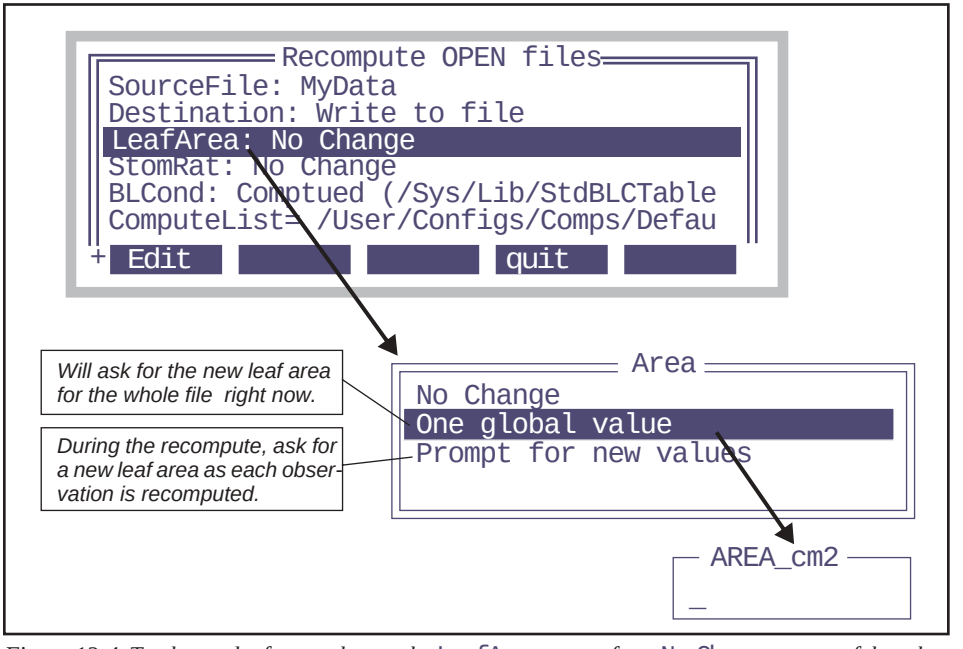


Figure 13-4. To change leaf area, change the `LeafArea:` *entry from* `No Change` *to one of the other choices.*

6 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.

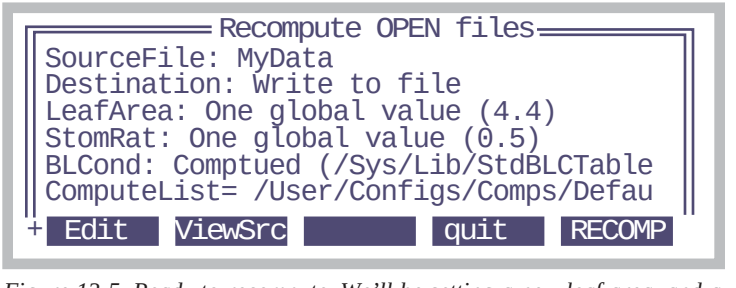


Figure 13-5. Ready to recompute. We'll be setting a new leaf area, and a new stomatal ratio. The recomputed data will be writen to an as yet unnamed file.

7 Recompute the file

Press **f5** (**RECOMP**). Because the destination is a file, you will be asked to name this file (Figure 13-6).

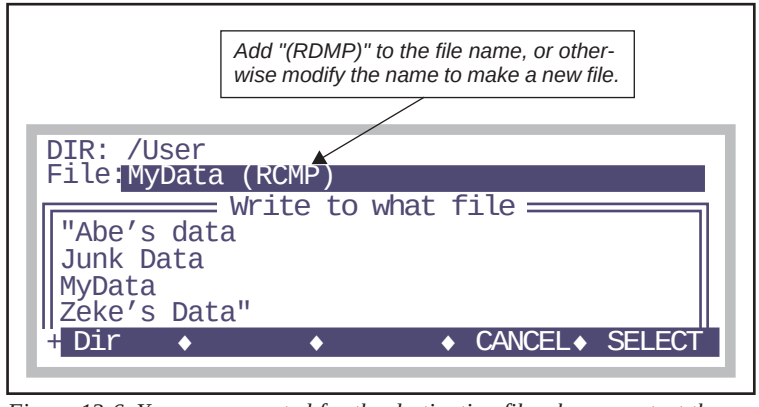


*Figure 13-6. You are prompted for the destination file when you start the recomputation, if the Destination: entry is set to "*Write to a file*".*

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 13-14.

8 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.

The Details

Recompute's Main Screen

The complete list of items to be found in Recompute's main screen is shown in Table 13-1.

Table 13-1. Recompute's Main Screen entries.

Main Screen Entry	Discussion
SourceFile:	The file to be recomputed.
Destination:	Toggles between `"Buffer"` and `"File"`. Select `"File"`, or you may run out of memory unless the file is very small.
LeafArea:	Has three possible settings: *No Change* *One Global Value (n)* *Prompt for new values* The global value applies to all observations in the file. Prompt for new values if each observation has a potentially different area.
StomRat:	Has three possible settings: *No Change* *One Global Value (n)* *Prompt for new values* The global value applies to all observations in the file. Prompt for new values if each observation has a potentially different stomatal ratio.
BLCond:	Has five possible settings: *No Change* *Computed (file name)* *One global value (1-sided)* *Prompt for new values* *No BLC correction* If you are changing leaf area or stomatal ratio, then set this to *Computed*. The global value and prompted values should be a 1-sided value, which is subsequently adjusted for stomatal ratio.
ComputeList=	Specify the name of the compute list file to use.
LogFormat=	Specify the name of the log format file to use.
PromptList=	Specify the name of the prompt list file to use.
LightSource=	Specify the light source. The relevant item here is not so much the source, as the conversion factor for converting the ParIn value to absorbed W/m2. This is relevant only to energy balance computations.
AlphaK=	If none of the light source entries has the AlphaK you need, then you can enter it here explicitly.

The main menu allows parameters that are frequently changed parameters to be changed. Nearly any value can be modified as well, and is explained in **Customizing Recompute** on page 13-12.

Source File Considerations

When you open a file and log data to it using OPEN, the data file contains a header[1] that includes some configuration information (Figure 13-7). This header lists information such as which compute list (described in Chapter 15) and which log format (**LogList Files** on page 9-14) was used. It may be that at the time you are recomputing the file, different compute list or log format files are in effect. Or, you may want to impose a different one when you do the recompute. The program provides quite a bit of flexibility in this regard.

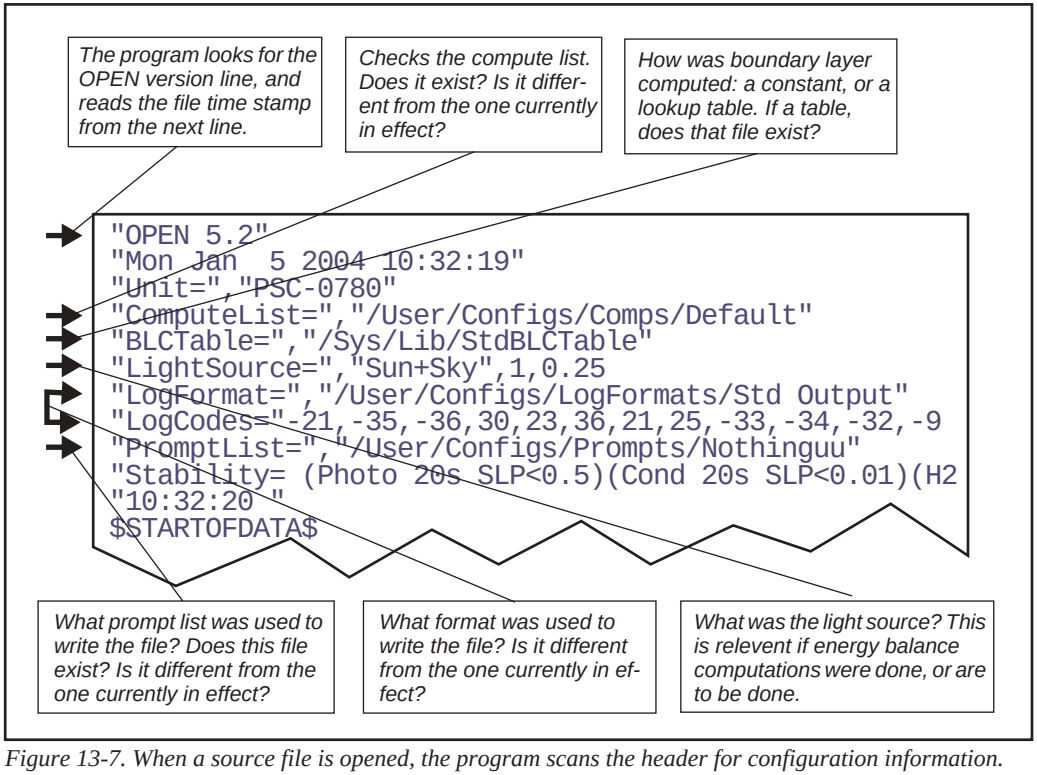


Figure 13-7. When a source file is opened, the program scans the header for configuration information.

[1] 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).

Compute List Files

The compute list file specifies the user defined values and how they are computed, and is part of OPEN data file header information (Figure 13-7). If there is a difference between the one used for the file, and the one currently in effect, you are prompted to select which to use (Figure 13-8). If the one used cannot be found, you are alerted to this (Figure 13-9), and the current one is used. Either way, you can always set the compute list before recomputing by highlighting the `"ComputeList:"` entry in Recompute's main screen, and pressing **enter**.

Log Format Information

The log format file determines what variables are written to the destination file. The original format comes from the LogCodes= entry. (If that is not there[2], the program will get the original information from the LogFormat= file.) The the original format file and the current format file are different, you are alerted to that fact (Figure 13-9), and the current one remains in effect. If the original one and current one differ in name, you are asked to select which to use (Figure 13-8). You can also set the log format list prior to recomputing by highlighting the LogFormat= entry in Recompute's main screen, and pressing **enter**.

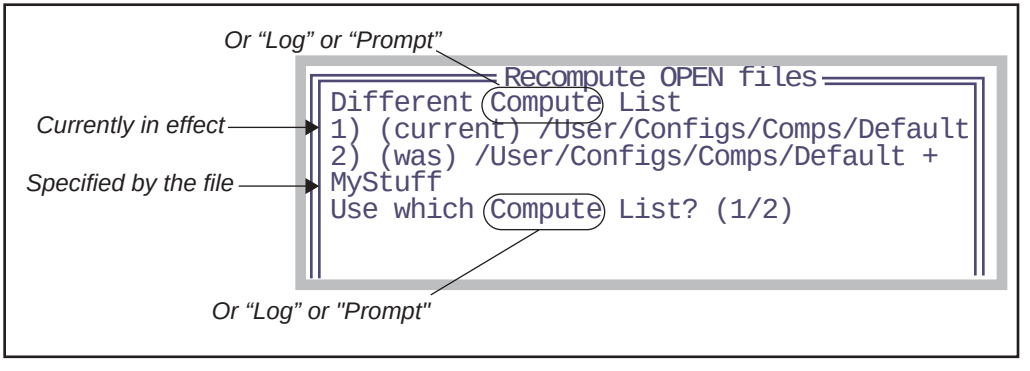


Figure 13-8. If the compute list, log format, or prompt list file differs between the one currently in effect, and the one used originally, you are prompted to select between the two.

2. This would be the case if you loaded got a pre-version 5 data file onto the instrument to recompute it

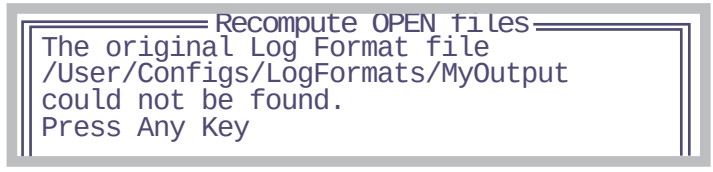


Figure 13-9. If an original compute list, log format, or prompt list file is missing, you are notified, and the current file (the one currently in effect) is used.

Building the Address List

The address list is the program's list of items that were originally written to the file. This information comes from the "LogCodes=" entry in the file header. This contains the system and user ID's of the data columns.

If for some reason the "LogCodes=" line isn't found, then the "LogFormat=" entry in the header is used to get the original log file list. (This was the main approach prior to version 5.) The potential problem with this approach is that this log format file may have been changed subsequent to the data file's creation. To get around this problem, the program uses the label line. If the label line cannot be found, you will be asked to manually identify it (Figure 13-10).

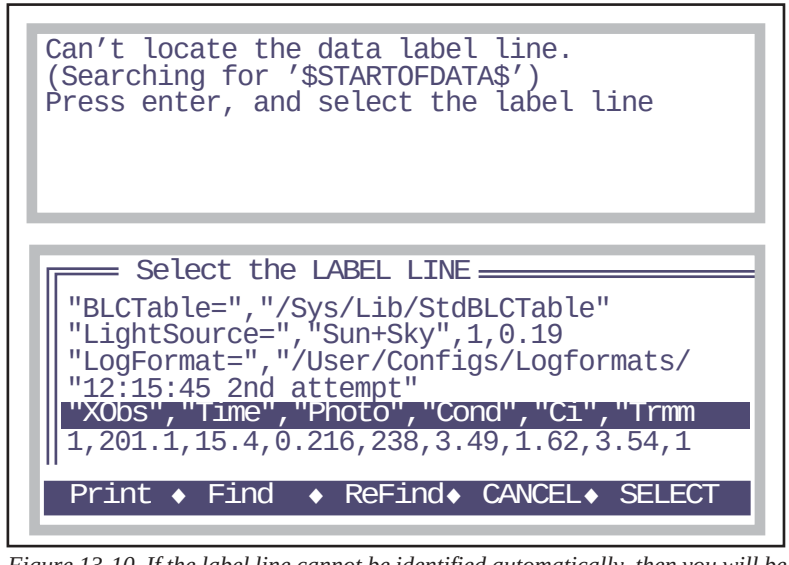


Figure 13-10. If the label line cannot be identified automatically, then you will be asked to do it manually.

The program reconciles the LogFormat file and the label line. If the log format file still exists and is unchanged since the data file was made, then the list of labels in the label line should match the labels made using the internal ID numbers. If the label line in the data file was edited, for example, or if the original log format file is not in effect, or can't be found, then the program has to try to figure out from the label which of the presently defined variables (if any) each data column corresponds to. If the program fails to match the label with any known variable, you are alerted to the problem (Figure 13-11).

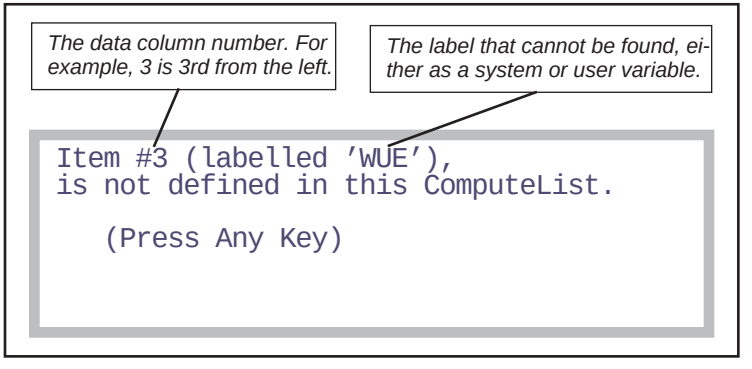


Figure 13-11. If one of the data columns has a label that cannot be found, the user is alerted with a message such as this.

If the original prompt list file is not available, it actually doesn't matter, because the program will rename any system variable (-101 to -109) that you logged to match the label it had in your source file.

Destination File Considerations

File or Memory

You have a choice of writing to memory, or writing directly to a file. If you choose the latter, do NOT overwrite the original file.

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 13-12). If you recompute multiple times, you'll get multiple headers.

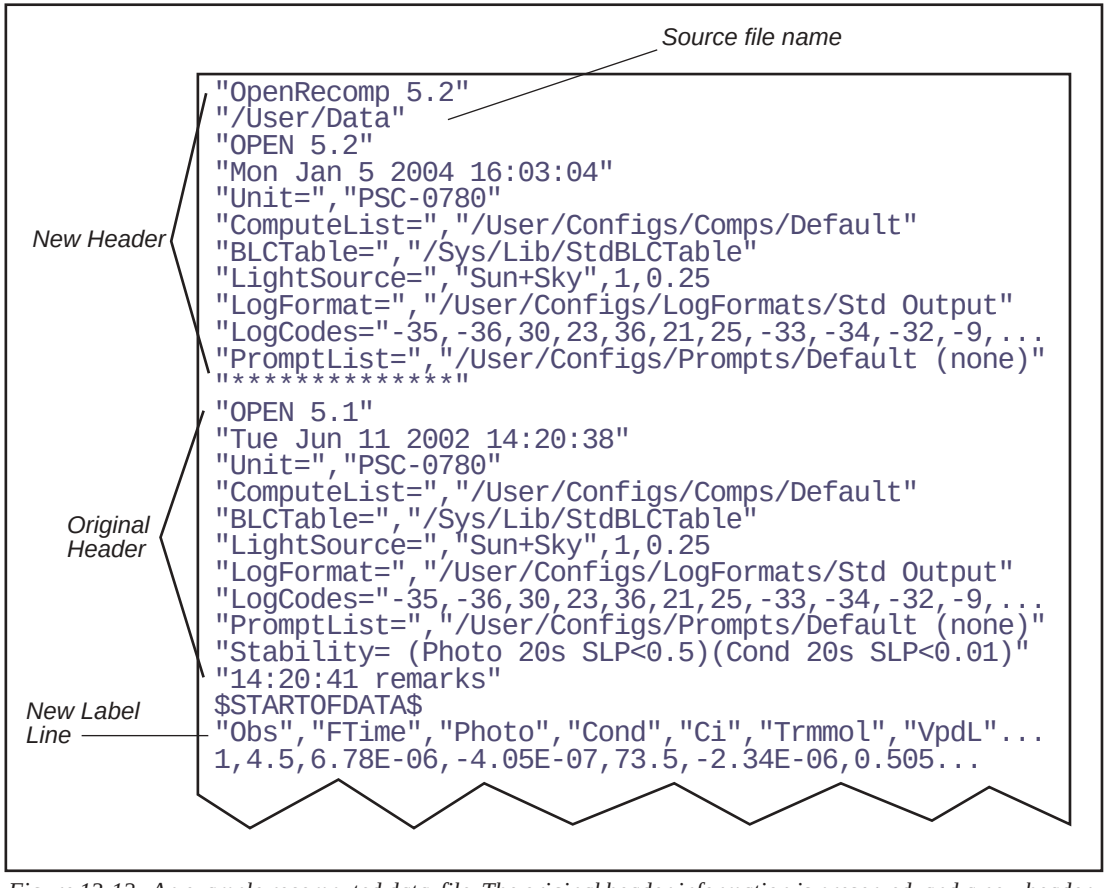


Figure 13-12. An example recomputed data file. The original header information is preserved, and a new header written before it containing the configuration used for the recomputation. The label line will reflect the log format file used for the recompute.

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 shown in Figure 13-13. 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.

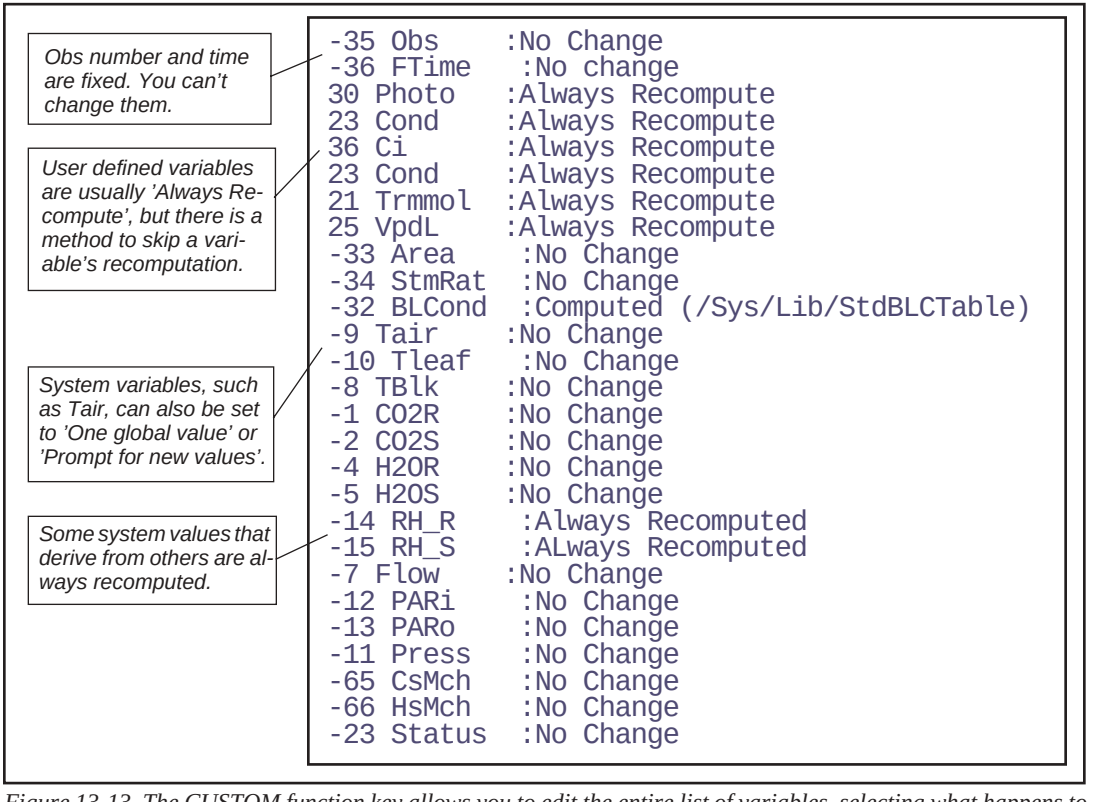


Figure 13-13. The CUSTOM function key allows you to edit the entire list of variables, selecting what happens to each (with some limitations, of course).

Hints

Suppressing a Variable's Recomputation

Suppose you have a user defined variable that is based on a mV signal from a sensor, but that mV signal was not stored in the file. You would not want to recompute that variable, because all the new values would be nonsense, since they would be based on that channel at the moment, not the signals recorded during the original measurement. It is the ComputeList that controls whether or not a user defined variable can be skipped over when recomputing. See **The ASSIGN() function** on page 15-10.

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.

- **Changing log lists during logging**
  When a data file is opened, the header information is written immediately, including the label line. If the log list is then modified while the file remains open, there is potential for a mismatch between label line and observations. See also **Log Format Information** on page 13-9.

- **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 will go until the end of 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...xxxxxxxxx.....
```

It probably means if found a remark after 5 observations, and then after another three observations, if found another data set appended on, with 4 observations. Or, someone logged nine remarks one after the other.

Part IV

Configuration Issues

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**) will 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 14-8 on page 14-19 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.

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 14-12.

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 the configuration command[2]

```
CalPress= <ap0> <ap1>
```

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

$$T_b = \frac{V_b}{100} \qquad (14\text{-}3)$$

---

1. These ID numbers refer to the system ID number. See Table 14-8 on page 14-19.
2. Configuration commands are discussed in Chapter 16.

IRGA Temperature

The IRGAs 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 (signal in mV is V_x), and use this in a software correction for zero drift, described below in (14-7) and (14-11).

ID: -58

$$T_x = \frac{1}{5.3375\times10^{-6} V_x + 0.0090167} - 22.696 \qquad (14\text{-}4)$$

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

$$W_r = \left[f\left((V_{wr} + z_w)G_{wr}\frac{101.3}{P}\right)\left[\frac{273 + \frac{T_b + T_a}{2}}{273}\right] + W_{mr}\right]F_{O_2} \qquad (14\text{-}5)$$

ID: -5

$$W_s = \left[f\left((V_{ws} + z_w)G_{ws}\frac{101.3}{P}\right)\left[\frac{273 + T_a}{273}\right] + W_{ms}\right]F_{O_2} \qquad (14\text{-}6)$$

where the function *f(x)* is a 3rd order polynomial whose three coefficients $a_{w1}...a_{w3}$ (the offset term a_{w0} is always 0) are specified by the configuration command

`CalH2O= <`a_{w1}`> <`a_{w2}`> <`a_{w3}`>`

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 18-14).

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}).

$$z_w = (T'_{xw} - T_x)S_w \qquad (14\text{-}7)$$

S_w is the zero shift calibration term for water. It, and the corresponding value S_c for CO_2 are specified by the configuration command

`CalZero= <`S_c`> <`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)[3].

$$F_{o_2} = 0.96 + \frac{0.04}{21} X_o \qquad (14\text{-}8)$$

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

$$C_r = g\left(\frac{(V_{cr} + z_c)G_{cr}}{B_r}\frac{101.3}{P}\right)\left[\frac{273 + \frac{T_b + T_a}{2}}{273}\right]B_r + C_{mr} \qquad (14\text{-}9)$$

[3] J.A.Bunce, 2002. Sensitivity of infrared water vapor analyzers to oxygen concentration and errors in stomatal conductance. *Photosynthesis Research* 71:273-276.

ID: -2

$$C_s = g\left(\frac{(V_{cs} + z_c)G_{cs}}{B_s}\frac{101.3}{P}\right)\left[\frac{273 + T_a}{273}\right]B_s + C_{ms} \qquad (14\text{-}10)$$

where the function *g(x)* is a 5th order polynomial whose five coefficients $a_{c1}...a_{c5}$ (the offset term a_{c0} is 0) are specified by the configuration command

```
CalCO2= <ac1> <ac2> <ac3> <ac4> <ac5>
```

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 18-11).

The zero drift correction term z_c for CO_2 is analogous to that for water (14-7).

$$z_c = (T'_{xc} - T_x)S_c \qquad (14\text{-}11)$$

The band broadening correction terms B_r and B_s for water vapor and oxygen in the IRGA cell, enabled by the configuration command

```
BndBrdCorr= <yes or no>,
```

are computed from

$$B_r = 1.0 + 0.0005\,W_r - 0.001X_o$$
$$B_s = 1.0 + 0.0005\,W_s - 0.001X_o \qquad (14\text{-}12)$$

This formulation for the water band broadening correction is derived in **Band Broadening Corrections** on page 14-23.

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_l (C) by

ID: -10

$$T_l = T_b + \frac{V_l}{100} \quad (14\text{-}13)$$

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

$$F = a_f V_f \quad (14\text{-}14)$$

The calibration factor a_f is specified by the configuration command

```
CalFlow= <af>
```

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

$$Q_c = a_{qc}(V_{qc} - V_{qo}) \quad (14\text{-}15)$$

The value of the offset term V_{qo} is 0 at power on, or whatever was read from the file "/User/Configs/ParInZero", but can be changed by the user by performing the routine **Zeroing the ParIn Signal** on page 18-25. The value of a_{qc} depends upon three configuration commands:

```
LightSource= <name> <fa> <B>
CalParGaAs= <ag>
CalParLED= <ae>
```

Specifically,

$$a_{qc} = \begin{cases} a_e & \text{if LightSource is 6400 LED} \\ a_g f_a & \text{if LightSource is anything else} \end{cases} \quad (14\text{-}16)$$

The actinity correction factor f_a for the GaAsP sensor allows the "same" calibration factor a_g to be used for various light sources. For further details about the *LightSource=* configuration command, refer to **LightSource=** on page

16-39.

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

$$Q_x = \frac{10}{a_{qx}} V_{qx} \qquad (14\text{-}17)$$

where the calibration factor a_{qx} ($\frac{\mu A}{1000\ \mu mol\ m^{-2}\ s^{-1}}$)is specified by the configuration command

`CalParOut= <`a_{qx}`>`

Computed Variables

Computed variables don't 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

$$e_r = \frac{W_r P}{1000} \qquad (14\text{-}18)$$

while in the sample cell, the vapor pressure e_s (kPa) is

ID: -54

$$e_s = \frac{W_s P}{1000} \quad (14\text{-}19)$$

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

$$h_r = \frac{e_r}{e(T_a)} 100 \quad (14\text{-}20)$$

ID: -15

$$h_s = \frac{e_s}{e(T_a)} 100 \quad (14\text{-}21)$$

Saturation Vapor Pressure

The saturation vapor pressure function *e()* used by OPEN is from Buck (1981)[4]:

$$e(T) = 0.61365 e^{\frac{17.502T}{240.97 + T}} \quad (14\text{-}22)$$

where the argument *T* is in degrees C. The LPL implementation of function *e()* is named *SatVap*. (Table 15-3 on page 15-16).

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

$$T_{dr} = d(e_r) \quad (14\text{-}23)$$

ID: -17

$$T_{ds} = d(e_s) \quad (14\text{-}24)$$

The dew point function *d()* is Equation (14-22) solved for temperature *T*, along with a check to keep bad things from happening should vapor pressure be less than or equal to 0.

---

[4] Buck, A.L. (1981) New equations for computing vapor pressure and enhancement factor. J. Appl. Meteor. 20:1527-1532.

$$d(e) = \begin{cases} -99.9 & \text{if e} < 0.01 \\ \dfrac{240.97z}{17.502 - z} & \text{if e} > 0.01 \end{cases} \quad (14\text{-}25)$$

where the argument *e* is vapor pressure in kPa, and $z = \ln\left(\frac{e}{0.61365}\right)$. The LPL implementation of function *d()* is named *DewPoint* (Table 15-3 on page 15-16).

Stability Variables

Any number of quantities can be included in the system stability test (**Stability Indicators** on page 4-40). 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), and *dF/dt* (ID:-98).

For *n* observations of quantity *x*, the formulae for mean, standard deviation, and coefficient of variation are given below:

Mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (14\text{-}26)$$

Standard Deviation

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad (14\text{-}27)$$

Coeff Variation (%)

$$c = 100(s/\bar{x}) \quad (14\text{-}28)$$

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 loose adequate resolution (0.5 seconds) after about 92 days. This means you need to power off at least every 3 months or so, or else the real time graphics and fluorescence events (flash, dark pulse) will begin to loose 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 14-1. CO_2 and H_2O IRGA status.

Value	Meaning	What is Sensed			
		CO_2 IRGA		H_2O IRGA	
		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 20-7.

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 14-2. 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 14-3. 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 20-28).

ID: -28

"FLOW"

The flow control hardware can indicate four values:

Table 14-4. 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 a big target value, such as 2000. 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. This

occurs most often when using the 6400-01 CO_2 Mixer, which means the flow control is done with a diverter valve.

ID: -30

"FAN"

The sample and leaf chamber cell mixing fan can have three values:

Table 14-5. 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 6 digit numerical composite of the status flags.

Table 14-6. 6 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 14-1 illustrates a piece of LPL code that does one thing if the IR-

GAs are OK, and another if they are not.

```
StatusCO2 StatusH2O OR IF
  /* Not ok */
...

ELSE
  /* both ok */
...

THEN
```

Figure 14-1. 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 25-12).

Table 14-7. 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 25-12).

ID: -57

"FwMxCrLp"

This four digit (it's actually an 8 character string) value provides a control manager status indicator (Figure 14-2). 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.)

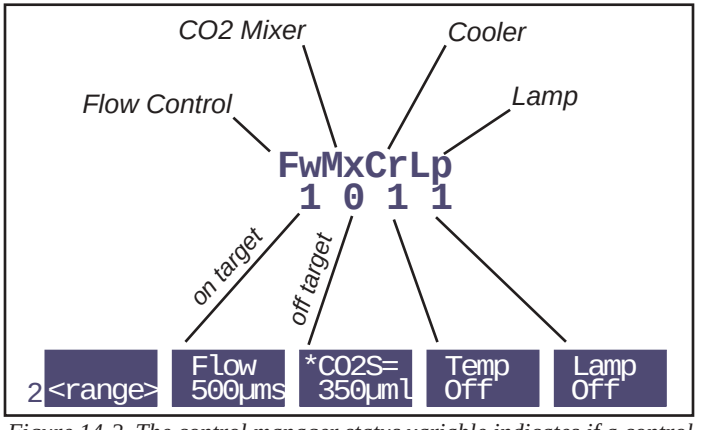


Figure 14-2. The control manager status variable indicates if a control is on target and stable.

A '0' in the FwMxCrLp display corresponds to an asterisk on the related control manager function key.

Boundary Layer Variables

Boundary layer conductance is computed based on the configuration command BLCond=, which can be either a file name, or else a number.

BLCond = <numeric value>

The numeric value is taken to be the one-sided boundary layer conductance to water, g_{bw} which is a system variable (ID: -55). The effective boundary layer conductance g'_{bw}, which takes into account the ratio K (ID: -34) of stomata on one side of the leaf to the other, is

ID: -32

$$g'_{bw} = \frac{g_{bw}(K+1)^2}{K^2+1} \qquad (14\text{-}29)$$

K is determined by the StomRat= configuration command. Since boundary layer conductance is also a function of wind speed and leaf area, this fixed value option is generally used when not measuring broadleaves.

BLCond= <file name>

The file specified is data for a lookup table, that accounts for the leaf area and the fan speed in computing the boundary layer conductance of broadleaves. The structure of the file in question is illustrated in Figure 14-3. In fact, this file (/Sys/Lib/StdBLCTable) is the default setting for BLCond=.

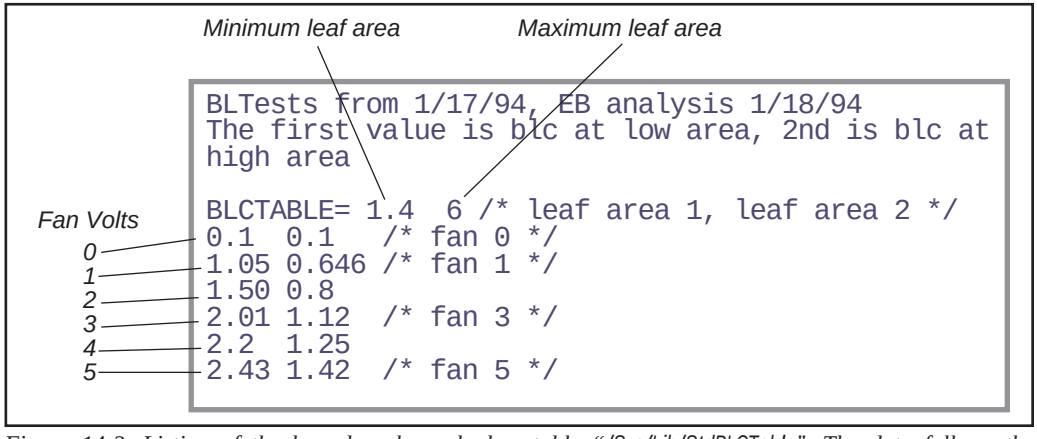


Figure 14-3. Listing of the boundary layer lookup table "/Sys/Lib/StdBLCTable". The data follows the BLCTABLE= string, and two values are expected in that line: the minimum leaf area, and the maximum leaf area. Following this comes 6 lines, one for each fan speed voltage (0 to 5). On each line is a pair 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 boundary layer values from the table are used to compute g_{bw}:

ID: -55

$$g_{bw} = \text{Table}(s, v_f) \qquad (14\text{-}30)$$

where s (ID: -33) is leaf area (cm^2) and v_f is fan voltage (0 to 5). g'_{bw} is then computed from (14-29). The Table() function is a linear interpolation based on leaf area, using the lookup table boundary layer values for the appropriate fan voltage. 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 warmer 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 21-14.

List of Open 5.3 System Variables

Table 14-8 lists the system variables provided by OPEN. Any of these can be displayed, logged, plotted, etc., as well as used in computations and AutoPrograms.

Table 14-8. 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 14-12
-1	CO2R_µml CO2R	Reference CO_2 µmol mol^{-1}	*co2_1_um*	C_r Eqn (14-9), pg 14-6
-2	CO2S_µml CO2S	Sample CO_2 µmol mol^{-1}	*co2_2_um*	C_s Eqn (14-10), pg 14-7
-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 14-5
-5	H2OS_mml H2OS	Sample H2O mmol mol^{-1}	*h2o_2_mm*	W_s Eqn (14-6), pg 14-5
-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 (14-14), pg 14-8
-8	Tblock°C Tblk	IRGA Block Temp C	*tblk_c*	T_b Eqn (14-3), pg 14-4
-9	Tair°C Tair	Chamber Air Temp C	*tcham_c*	T_a Eqn (14-2), pg 14-4
-10	Tleaf°C Tleaf	Leaf Temp C, measured with the thermocouple	*tleaf_c*	T_l Eqn (14-13), pg 14-8
-11	Prss_kPa Press	Atm Press kPa	*press_kpa*	P Eqn (14-1), pg 14-4
-12	ParIn_µm PARi	In-chamber PAR µmol m^{-2} s^{-1}	*parIn_um*	Q_c Eqn (14-15), pg 14-8
-13	ParOutµm PARo	External PAR µmol m^{-2} s^{-1}	*parOut_um*	Q_x Eqn (14-17), pg 14-9
-14	RH_R_% RH_R	Reference RH%	*rhIn*	h_r Eqn (14-20), pg 14-10

Table 14-8. (Continued) System Variables - Labels and Variable Names

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-15	RH_S_% RH_S	Sample RH%	*rhOut*	h_s Eqn (14-21), pg 14-10
-16	Td_R_°C TdR	Reference Dew Point °C	*tdIn*	T_{dr} Eqn (14-23), pg 14-10
-17	Td_S_°C TdS	Sample Dew Point °C	*tdOut*	T_{ds} Eqn (14-24), pg 14-10
-21	HH:MM:SS HHMMSS	Real time clock	*clocktime*	page 14-12
-22	Program	Auto Program time status	*lpTimeStat*	page 14-16
-23	CHPWMF Status	Numerical status code	*statusWord*	page 14-15
-24	Battery	Battery voltage	*battery_v*	Reported in Volts
-25	CO2	CO2 IRGA status	*stat_co2*	page 14-13
-26	H2O	H2O IRGA status	*stat_h2o*	page 14-13
-27	PUMP	Pump status	*stat_pump*	page 14-13
-28	FLOW	Flow Control status	*stat_flow*	page 14-14
-29	MIXR	CO2 Mixer status	*stat_inj*	page 14-14
-30	FAN	Chamber fan status	*stat_fan*	page 14-15
-31	CO2...FAN	The status line	*statLineVar*	page 14-12
-32	BLC_mol BLCond	Total Boundary Layer Conductance (mol m^{-2} s^{-1})	*condBL_mol*	g'_{bw} Eqn (14-29), pg 14-17
-33	AREA_cm2 Area	In-chamber leaf area cm^{-2}	*area_cm2*	page 16-36
-34	STMRATIO StmRat	Stomatal ratio estimate	*stom_rat*	page 16-40
-35	Obs	# Obs stored in log file	*obsInPad*	page 14-12
-36	FTime	Time since logging started (s)	*obsTime*	page 14-12
-37	CO2R_mv	Ref CO2 IRGA mV	*co2_1_mv*	V_{cr} Eqn (14-9), pg 14-6
-38	CO2S_mv	Sample CO2 IRGA mV	*co2_2_mv*	V_{cs} Eqn (14-10), pg 14-7
-39	H2OR_mv	Ref H2O IRGA mV	*h2o_1_mv*	V_{wr} Eqn (14-5), pg 14-5
-40	H2OS_mv	Sample H2O IRGA mV	*h2o_2_mv*	V_{ws} Eqn (14-6), pg 14-5

Table 14-8. (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 14-4
-42	Tair_mv	Sample air temp mV	*tcham_mv*	V_a Eqn (14-2), pg 14-4
-43	Tleaf_mv	Leaf temp mV	*tleaf_mv*	V_l Eqn (14-13), pg 14-8
-44	flow_mv	Flow meter mV	*flow_mv*	V_f Eqn (14-14), pg 14-8
-45	press_mv	Pressure mV	*pressure_mv*	V_p Eqn (14-1), pg 14-4
-46	parIn_mv	In-chamber PAR mV	*parIn_mv*	V_{qc} Eqn (14-15), pg 14-8
-47	parOutmV	External PAR mV	*parOut_mv*	V_{qx} Eqn (14-17), pg 14-9
-48	CRagc_mv	Ref CO2 IRGA AGC mV	*agc_c1_mv*	**AGC voltages** on page 20-16
-49	CSagc_mv	Chamber CO2 IRGA AGC mV	*agc_c2_mv*	
-50	HRagc_mv	Ref H2O IRGA AGC mV	*agc_h1_mv*	
-51	HSagc_mv	Chamber H2O IRGA AGC mV	*agc_h2_mv*	
-52	Oxygen%	Oxygen concentration (%)	*oxyPct*	x_o Eqn (14-8), pg 14-6 and Eqn (14-12), pg 14-7
-53	vapR_kPa	Ref vapor press kPa	*eAir_1_kPa*	e_r Eqn (14-18), pg 14-9
-54	vapS_kPa	Sample vapor press kPa	*eAir_2_kPa*	e_s Eqn (14-19), pg 14-10
-55	BLC1_mol	1-sided Boundary Layer Conductance (mol m^{-2} s^{-1})	*condBL_one*	g_{bw} Eqn (14-29), pg 14-17
-56	ProgPrgs	AutoProgram Progress	*lpProgress*	page 14-16
-57	FwMxCrLp	Control Panel Stability: Flow Mixer Cooler Lamp	*cpStable*	page 14-16
-58	Tirga°C Tirga	IRGA Temp C	*tIrga_c*	T_x Eqn (14-4), pg 14-5
-59	Tirga_mv	IRGA Temp mV	*tIrga_mv*	V_x Eqn (14-4), pg 14-5
-60	uc_20_mV	User Channel 20 mV	*chan20_mv*	See **Analog Input Channels** on page 26-17
-61	uc_21_mV	User Channel 21 mV	*chan21_mv*	
-62	uc_22_mV	User Channel 22 mV	*chan22_mv*	
-63	uc_23_mV	User Channel 23 mV	*chan23_mv*	

Table 14-8. (Continued) System Variables - Labels and Variable Names

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-64	DecHour	Decimal time of day	*decHour*	page 14-12
-65	CsMch	Sample CO_2 offset µmol mol^{-1}	*co2_2_offset*	C_{ms} Eqn (14-10), pg 14-7
-66	HsMch	Sample H_2O offset mmol mol^{-1}	*h2o_2_offset*	W_{ms} Eqn (14-6), pg 14-5
-67	CrMch	Ref CO_2 offset µmol mol^{-1}	*co2_1_offset*	C_{mr} Eqn (14-9), pg 14-6
-68	HrMch	Ref H_2O offset mmol mol^{-1}	*h2o_1_offset*	W_{mr} Eqn (14-5), pg 14-5
-69	DOY	Day of the year (0…366)	*clockDOY*	page 14-12
-70	YYYYMMDD	Date code (integer)	*clockDate*	page 14-12
-71	Stable	Stable / Total	*StableStat*	**Stability Indicators** on page 4-40.
-72	StableF	Stable / Total as fraction	*fractStable*	
-73	<letters>	Status as string. (e.g. "1101")	*StableFlags*	
-74	TotalCV	Sum of stab. variables' CV	*totalCV*	
-80	F	Fluorescence signal (zero subtracted)	*flr_f*	**Display Summary** on page 27-29
-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*	

Table 14-8. (Continued) System Variables - Labels and Variable Names

ID	Display Label Log Label[a]	Description	Variable Name	Definition or Reference
-90	Fs	Steady state fluorescence	*flr_s*	**Display Summary** on page 27-29
-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 Fs last set	*parIn_fs*	
-97	FlrCV_%	CV (in percent) of F	*flrRms*	page 14-11
-98	dF/dt	Rate of change of F (per minute)	*flrSlope*	
-101	AuxF1[b]	User defined, float	auxF1	**System Variables for Prompts** on page 9-18
-102	AuxF2	User defined, float	auxF2	
-103	AuxF3	User defined, float	auxF3	
-104	AuxN1	User defined, integer	auxN1	
-105	AuxN2	User defined, integer	auxN2	
-106	AuxN3	User defined, integer	auxN3	
-107	AuxS1	User defined, string (8)	auxS1	
-108	AuxS2	User defined, string (18)	auxS2	
-109	AuxS3	User defined, string (38)	auxS3	

a.Same as Display Label, if none shown
b.Default. -101 thu -109 have user defined labels.

Band Broadening Corrections

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),

$$\frac{A}{P} = g(u/P) \qquad (14\text{-}31)$$

where A is total band absorption, P is total pressure (kPa), u is absorber amount (mol m^{-2}) = ρL ; 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

$$\begin{aligned} \rho &= \frac{p}{RT} \\ &= \frac{XP}{RT} \end{aligned} \qquad (14\text{-}32)$$

where p is absorber partial pressure and X is absorber mole fraction (mol absorber / mol air). Therefore,

$$\frac{u}{P} = \frac{XL}{RT} \qquad (14\text{-}33)$$

Substituting (14-33) into (14-31) and incorporating the constants L and R into a new function h gives,

$$\frac{A}{P} = h(\frac{X}{T}) \qquad (14\text{-}34)$$

In principle, (14-34) can be solved for mole fraction, giving

$$X = h^{-1}(\frac{A}{P})T \qquad (14\text{-}35)$$

Since LI-COR gas analyzers produce an output voltage that is proportional to absorptance,

$$V = KA \qquad (14\text{-}36)$$

substituting (14-36) into (14-35) yields

$$C = f(V\frac{P_0}{P})\frac{T}{T_0} \qquad (14\text{-}37)$$

where C is the CO_2 mole fraction in µmol mol^{-1}, and the constant K is included in the calibration function f; P_0 = 101.3 kPa, T_0 =273 K. Equation (14-37) 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

$$P_e = P_{N_2} + \sum \alpha_i P_i + b_{CO_2} \qquad (14\text{-}38)$$

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 14-9. 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 14-9. Typical foreign gas and self-broadening coefficients for the CO_2 4.26 µm absorption band (Burch et. al. 1962).

Gas	Broadening Coefficient	Comment
N_2	1.00	Foreign Gasses
O_2	0.81	

Table 14-9. (Continued) Typical foreign gas and self-broadening coefficients for the CO_2 4.26 µm absorption band (Burch et. al. 1962).

Gas	Broadening Coefficient	Comment
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,

$$P_e = P[1 + (\alpha_w - 1)w + \sum(\alpha_i - 1)x_i] \qquad (14\text{-}39)$$

x_{CO_2} is typically small (3.5×10^{-4}) for gas exchange measurement conditions and is neglected. If we consider oxygen, water vapor, and nitrogen, then (14-39) becomes

$$\begin{aligned} P_e &= P[1 + (\alpha_w - 1)w + (\alpha_o - 1)x_o] \\ &= P\,c(w,x_o) \end{aligned} \qquad (14\text{-}40)$$

A calibration equation similar to (14-36) that includes the pressure broadening effects of variable water vapor can be obtained by substituting P_e for P in (14-31) and (14-32), and carrying through the subsequent steps to give

$$C = c(w,x_o)F\left[\frac{VP_0}{Pc(w,x_o)}\right]\frac{T}{T_o} \qquad (14\text{-}41)$$

Finding an appropriate values for α_w and α_o for use in (14-41) 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. Thus, Equation (14-12) comes from

$$\begin{aligned} c(w,x_o) &= 1 + (1.5-1)w + (0.9-1)x_o \\ &= 1 + 0.5w - 0.1x_o \\ &= 1 + 0.0005W - 0.001X_o \end{aligned} \qquad (14\text{-}42)$$

where W has units of mmol mol^{-1}, w is mol mol^{-1}, X_o is %, and x_o is mol mol^{-1}.

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.

15

Defining User Variables

Equations, Constants, and Remarks

15 Defining User Variables

OPEN provides a method of extending the quantities computed beyond the set of system variables described in the previous chapter. The user can define variables (or constants) that can be viewed, logged and plotted, just like the system variables can. This chapter explains how.

An introduction to this topic can be found in **Add Water Use Efficiency** on page 3-76.

The ComputeList File

A ComputeList is simply a list of the variables and constants that you wish to define. ComputeLists take the form of a file. These files are generally stored in the directory "/User/Configs/Comps".

Where Do They Come From?

There is a default ComputeList that implements the equations for photosynthesis, conductance, and other useful quantities listed in **Equation Summary** on page 1-7. This file is named "/User/Configs/Comps/Default". There are two actions that will cause other ComputeList files to appear in this directory.

- **Creating a configuration from the Installation Menu**
  Some items have special ComputeLists associated with them, such as the 6400-09 Soil Flux Chamber, and the 6400-05 Conifer Chamber, so installing those items will cause a new file to appear in the /User/Configs/Comps directory. Also, just building a configuration for a more conventional chamber will, if you choose to use energy balance for leaf temperature, create an energy balance version of "Default" named "Default using EB" or "Clear bottom EB".

- **Editing a ComputeList file, and saving under a new name**
  You can directly make a ComputeList file by editing an existing one (or starting from scratch), and storing under a new name. You might do this to add a computation that is not included in the default, or to change an equation, should you not like the formula we used in the default.

ComputeList files can be specified as part of a configuration by using the

```
ComputeList=
```

configuration command (Chapter 16).

Editing ComputeList Files

One of the entries in the Config Menu is "_ComputeList Menu", and it is shown in Figure 15-1.

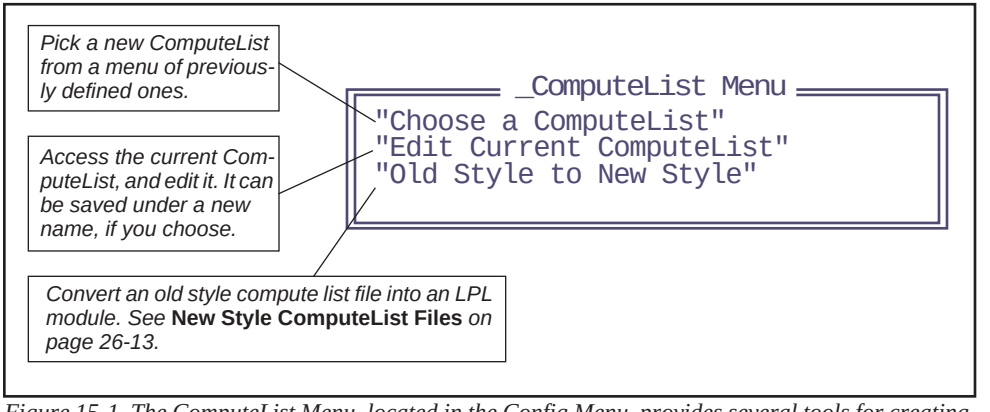


Figure 15-1. The ComputeList Menu, located in the Config Menu, provides several tools for creating, editing, and implementing ComputeLists.

To view and edit the currently active ComputeList, select the "Edit Current ComputeList" entry (no surprise there) and press **enter**. If the currently active ComputeList is the default one, you will see the file shown in Figure 15-10 on page 15-18.

Once you understand the format that ComputeList files should use (next section), you can make the necessary changes to the file, and then store it by pressing **escape** to access the editor's exit menu (described in **The Exit Menu** on page 5-15) and press **S** to store it as a new file (or **U** to store it without changing the name).

ComputeList File Format

The file containing the list of user computations must adhere to a particular format. We illustrate by showing the first few variables of the default ComputeList (Figure 15-2). The complete listing is in **The Default ComputeList** on page 15-18.

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)"

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20 * 1E3"

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, Tleaf_c, condBL_mol, press_kPa, h2o_2_mm)"
```

Figure 15-2. A sample of a Compute List file, including the equations for transpiration and conductance.

In general, the format for each item defined in a ComputeList is

```
##<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 ID number is quite arbitrary, but it must be an integer between 1 and 32767. The ID numbers must be unique in the file, so if you are adding a new variable, give it a unique number. These numbers are used by OPEN to identify the variable in display format files, real time graphics plot definitions, log lists, etc.

There are no consequences of duplicating a user variable number other than confusion, because only the first occurrence of a particular variable ID will actually be used; the duplicate(s) will be computed but inaccessible.

Format Code

Immediately following the ID number is the optional format code. The default format for displaying and logging user computations is to use 3 significant digits, 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. A format specifier consists of the letter F or G followed by a digit. 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 15-1).

Table 15-1. π, 1000π, and π/1000 expressed using the various format codes

Code	π	1000π	π/1000	Code	π	1000π	π/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.

If, for example, you wish to see conductance shown with 4 significant digits instead of 3, you could do this by appending a G4 to the ID number, as shown below:

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)"

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20 * 1E3"

##23G4 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, Tleaf_c, condBL_mol, press_kPa, h2o_2_mm)"
```

Figure 15-3. Conductance definition modified to display 4 significant digits instead of 3.

The Label String

The first of the three quoted strings following the ID/Format sequence is the label, which must be 8 characters or less. The variable's label is used to identify the variable in text, graphics, and file output.

The Description String

The second quoted string is a short description. This description is used in some menus, such as when selecting variables in the LogList editor.

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 14-8 on page 14-19), and other user variables (referred to by number, so #20 in the equation for conductance in Figure 15-3 refers to transpiration rate, for example). Table 15-2 provides some examples.

Table 15-2. Sample user variable definitions, illustrating the sorts of things that can make up the equation part.

Example Definitions	Discussion
`##10 "(U/S)" "flow:area: ratio"` `" flow_um/area_cm2/100.0 "`	*Typical numerical definition. It refers to two system variables by their name.*
`##10 "(U/S)" "flow:area: ratio"` `" $ flow_um area_cm2 100.0 / / "`	*Same thing, but in post-fix notation. The $ signals the change, and there must be spaces between each item.*
`##21 "Trmmol" "transpiration (mmol/m2/s)"` `" #20 * 1E3"`	Multiples an earlier user variable, #20, by 1000.
`##101 "Plot#" "xxx" "UCON(0)"`	Defines a floating point user constant. Nothing is computed here, but this value can be used in other definitions by referencing it as #101. The default value of the constant is 0 (the argument of UCON).
`##9872 "Comments" "whatever" "UREM(20)"`	Defines a remark field of up to 20 characters. This is not a numeric quantity, shouldn't be used in computations.

`##30 "Photo" "Photosynthesis (umol/m2/s)"`
`"-#10*co2_diff_um - co2_2_um*#20`
`AOSET(#30 * 5000.0 / 30.0, 3)`
	After computing photosynthesis, the D/A converter on port 3 is set to reflect the photo value, where 5000mV would correspond to 30 µmol m^{-2} s^{-1}.

`##1001 "TxAir" "External air temp"`
`"NOASSIGN`
`Assign( &u1001, chan21_mv / -10) "`
	This variable will have the option of being "skipped" during a recompute.

`##1953 "Crazy" "sample multi-line"`
`"NOASSIGN`
`IF (#25 > 0)`
`u1953 = 10.23 / #25`
`ELSE`
`u1953 = 0`
`THEN "`
	Example of a multi-line definition. Suppress the automatic assignment with NOASSIGN in the first row, then write the desired code in subsequent lines, and do the variable assignment yourself.

Including Files

The ## marker can also be used to include multiple files. The format is

```
##"<file name>"
```

For example, the standard compute list file for the Leaf Chamber Fluorometer looks like this:

```
##"/user/configs/comps/Default"
##"/user/configs/comps/FlrOnly"
```

It simply joins two files that contain normal ## definitions. The first file is the default compute list (shown below in **The Default ComputeList** on page 15-18). To that is appended a further set of user definitions for fluorometry.

Wherever the ## include occurs in the file is where the included file is inserted. If must occur when a ## is valid, however. You can't, for example, include a file as part of the equation in a ## definition.

Writing and Modifying ComputeList Files

There are several options for editing ComputeList Files:

- **"Edit Current ComputeList"**
  This entry is in the "_ComputeList Menu", found in the Config Menu. Selecting this will access the system editor (**Standard Edit** on page 5-13) using the current ComputeList File.

  If you make changes to this file and save it (without changing the file's name), upon exit the updated file will be compiled and implemented.

  If you have stored the changes under a new file name, upon exit you will be asked

  ```
  Implement file <name>? (Y/N)
  ```

  If you press **Y**, the new ComputeList is read and implemented, and your configuration will have changed.

- **Use the system editor.**
  Use the Filer, or example, to edit a ComputeList file, and store it under a new name. To actually use the new file, you either "Choose a ComputeList" in the "_ComputeList Menu" to pick it, or use the Config Editor, add a "ComputeList=" line, and edit it.

- **Write one on your own computer**
  Save it as a text file, transfer it to /User/Configs/Comps, then implement it using the procedure described above.

Some Important Rules

Order of Definition

When user variables are computed, they are computed in order of occurrence in the file. Therefore, if #448 uses #633 in its definition, make sure #633 comes first in the ComputeList.

What can I put in the equation?

Equations can include:

- **Numerical System Variables**
  See Table 14-8 on page 14-19 for a list of variable names.
- **Other User Variables**
  These are referenced by ID number, preceded by a single # or (starting in OPEN 3.2) the letter *u*. For example, #101 or u101.
- **LPL Functions and Operators**
  Table 15-3 on page 15-16 provides a list of useful functions and operators.
- **UREM() or UCON()**
  These specify the variable to be user input, and not computed. This is described in **User Defined Constants and Remarks** on page 15-13.
- **The NOASSIGN marker**
  When OPEN processes a ComputeList, it generates code for a function that will execute every time user variables are computed. Each user variable that you define causes a variable named *unnn* to be created (where *nnn* 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 would appear as in Figure 15-4.

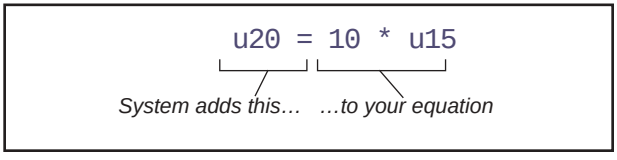


Figure 15-4. When building the code for each user variable, the system makes the equation by inserting the variable name and an = sign before your equation.

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. (I can think of no good reason to do this, but it illustrates the point.) 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 post-fix 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 the ComputeList is processed. (Test compiling is when it lists each variable followed by an "ok" or an error.) 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, and your entire ComputeList is not accepted.

- **The ASSIGN() function**

The Assign() function allows user variables to be computed during New Measurements mode, but allows them to be skipped during recomputations. For example, suppose we are measuring external air temperature with a 6400-13 Thermocouple Adapter. The air temperature channel is defined something like this:

```
##1001 "TxAir" "External air"
"NOASSIGN
 Assign( &u1001, chan21_mv / -10) "
```

Notice that the NOASSIGN keeps "U1001 =" from being prefixed to the equation, but then we assign *u1001* to the value by using *Assign*(). The first argument of Assign() must be the address of the thing being assigned (&*u1001* means the address of *u1001*), and the second argument is the value. What good is that? Well, you can log *TxAir* by itself, and not have to log chan21_mv. If you recompute, *TxAir* will not get recomputed (unless you ask it to be), so its value won't be changed. And you probably wouldn't want its value changed.

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 15-5. If you need comments, put them between ## definitions.

Compilation Errors

If you make a mistake defining a variable, you may get a message to that effect when it is compiled. While a ComputeList 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)"
" #20 * xyz"
```

The symbol *xyz* will not be found, so when *Trmmol* is compiled, an error will be reported (Figure 15-6).

[1] This doesn't happen when using :include (**Open's Hooks** on page 26-7) or NOASSIGN.

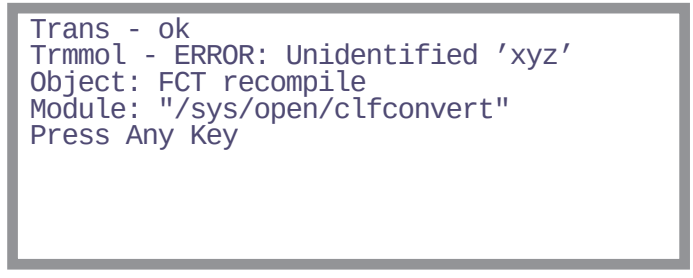


Figure 15-6. Example error message, when the unknown variable xyz *is put into the defining equation for* Trmmol.

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 ComputeList to fail. The message will look like Figure 15-7. No user defined quantities will be available. Either edit the ComputeList and fix the problem, or select another ComputeList.

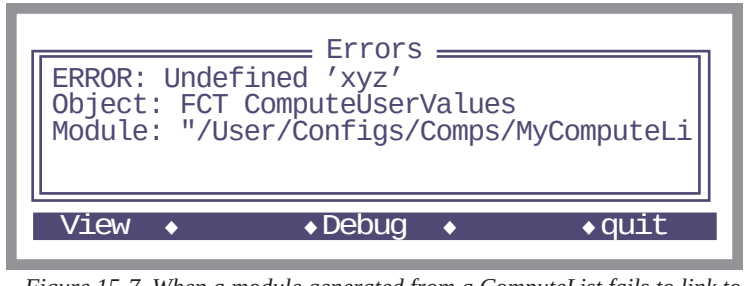


Figure 15-7. When a module generated from a ComputeList fails to link to open, a message such as this is shown. No user defined variables will be available.

User Defined Constants and Remarks

In addition to defining equations, the ComputeList File can be used to make extra user defined constants and remarks. Recall that the system provides some predefined ones (that you can rename) for your use (see **Prompts and Remarks** on page 9-15). The techniques described in this section would be used to add extra ones.

User Constants

A numerical user constant can be defined by using the function

```
UCON(0)
```

in the equation definition string. For example, Figure 15-8 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 15-8. 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.

User Remarks

If you wish a constant to contain letters and punctuation, in addition to numbers, then you’ll need a string remark. User remarks are defined just like user constants, except instead of UCON, use

```
UREM(n)
```

where *n* is the desired maximum string length of the remark.

Using Constants and Remarks

To include user defined prompts and remarks in the prompt list, use the Prompt List Editor, and add them, once they are defined (see **Prompts and**

Remarks on page 9-15). Otherwise, they can be treated just like user variables (they are user variables) in that you can log them, display them, plot them, etc.

Importing RS-232 Data

The macro function

```
UCOMM(n)
```

will retrieve the n^{th} item from the latest incoming line of RS-232 data the instrument has received while in New Measurements Mode. 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).

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
```

would be considered to have two items available; UCOMM(1) would retrieve "123", and UCOMM(2) would retrieve "45.67".

3 **Double quotes override delimiter characters**
The record

```
10.56 "123,45.67",The End
```

would be considered to have four items: UCOMM(1) would get "10.56", UCOMM(2) gets "123,45.67", UCOMM(3) gets "The", and UCOMM(4) gets "End". If an item is quoted, the quotes are stripped from the returned string.

An Example

Suppose we have a hypothetical GPS device that outputs a stream of data regularly, and wish to use the 2nd (longitude), 3rd (latitude) and 5th (status)

items in each record as variables in the LI-6400. Further, suppose the data records look 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 displayed 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 meaningful 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 15-9.

```
##876F5 "Long." "Longitude" "UCOMM(2)"

##875F5 "Lat." "Latitude" "UCOMM(3)"

##874S8 "XStat" "Status of the device"
"UCOMM(5)"
```

Figure 15-9. 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, maximum length of 8.

That's all you have to do, other than add the three variables to your display format and log file format.

UCOMM() and CommGet()

UCOMM() is a macro for use with the "old style" compute lists. The real function that does the work is named CommGet().

What UCOMM() does

The UCOMM() macro does a number of things for you automatically, and has some side-effects:

1 **The variable becomes "No Assigned"**

The variable using UCOMM will be treated as if non-assigned (see **The NOASSIGN marker** on page 15-8). This means that if you recompute a data page using a compute list with UCOMM in it, those variables will not be changed.

2 **Anything else in the definition is ignored.**
For example, you cannot do the following to multiply the 3rd comm item by 1000:

```
UCOMM(3) * 1000.0   /* won't work */
```

If you want to do this, you'll need to create another user variable to perform the multiplication and hold the result. That is,

```
##123 "X3" "3rd comm item" "UCOMM(3)"
##124 "X3Scaled" "now adjust it" "#123 * 1000.0"
```

3 **UCOMM(n) invokes CommGet().**
The actual function that does the work is *CommGet*(*x*, *y*), where *x* is the address of the destination variable, and *y* is the number of the item in the data record. Thus, the line

```
##875F5 "Lat." "Latitude" "UCOMM(3)"
```

actually gets changed into

```
##875F5 "Lat." "Latitude"
"NOASSIGN
CommGet(&u875, 3)"
```

4 **"Parse incoming Comm Data" is automatically set**
When CommGet() is called when New Measurements mode is active, the LI-6400 will enable the incoming comm parsing feature automatically, if necessary. This feature is normally enabled/disabled from "Configure the Comm Port" in OPEN's Utility Menu.

Useful Variables and Functions

Table 14-8 on page 14-19 contains variable names for the System Variables. Table 15-3 gives some useful functions.

Table 15-3. Some Useful Functions and Operators

Function	Description
+	Add
-	Subtract
*	Multiply

Table 15-3. (Continued) Some Useful Functions and Operators

Function	Description
/	Divide
^	Raise to a power
LOG(x)	natural log of x
LGT(x)	log base 10
CHS(x)	changes sign of x
ABS(x)	absolute value of x
SQRT(x)	square root of x
EXP(x)	exponential.
SatVap(T)	Returns saturation vapor pressure (kPa) T - temp (C)
StdCond(E, Tf, Blc, P, W)	Returns stomatal conductance (mol m^{-2} s^{-1}) E - transpiration (mol m^{-2} s^{-1}) Tf - leaf temp (C) Blc - boundary layer cond (mol m^{-2} s^{-1}) P - atmos pressure (kPa) W - ambient water vapor (mmol mol^{-1})
DewPoint(e)	Returns the dewpoint temperature (C) e - vapor pressure (kPa)
EB_DeltaT(Rn, Tw, Ta, Blc, E)	Returns Tleaf - Tair from energy balance. E - transpiration (mol m^{-2} s^{-1}) Blc - boundary layer cond (mol m^{-2} s^{-1}) Ta - Air temp (C) Tw - Chamber wall temp (C) Rn - Net radiation (W m^{-2})

The Default ComputeList

The default ComputeList is shown in Figure 15-10

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)"

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20 * 1E3"

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, Tleaf_c, condBL_mol, press_kPa, h2o_2_mm)"

##30 "Photo" "Photosynthesis (umol/m2/s)"
" -#10 * co2_diff_um - co2_2_um * #20"

##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / condBL_mol)"

##36 "Ci" "Intercellular CO2 (umol/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 "

##25 "VpdL" "Leaf VPD (es(Tleaf) - eair)"
" SatVap(Tleaf_c) - eAir_2_kPa"

##27 "VpdA" "Air VPD (es(tair) - eair)"
" satVapTair_kPa - eAir_2_kPa"
```

Figure 15-10. Listing of file /User/Configs/Comps/Default

Old Style vs. New Style

The discussion up to this point, especially concerning the format of ComputeList files, involves the “Old Style”. Until OPEN version 3.0, the Old Style was the only style. The need for increased flexibility, to support devices such as the Soil Flux chamber, gave rise to an alternative format ComputeList, which is termed the “New Style”.

Essentially, the new style ComputeList is one that is directly written in LPL, the LI-6400’s programming language (see Chapter 22). When an old style ComputeList is used, OPEN converts it into a new style, then links it. If you use a new style to start with, this step is eliminated.

One easy way to take advantage of the customization potential of the new style is to use the conversion program “Old Style to New Style”.

Further details are provided in **New Style ComputeList Files** on page 26-13.

Configuration Basics

16

What you need to know to get by

16 Configuration Basics

In the early days of the LI-6400, there was but one way to do any configuration change: go to the Config Editor. While this tool is still available as a last resort, later versions of software have introduced various configuration "helpers", such as Logging Control (page 9-8), Prompt Control (page 9-15), and Light Source Control (page 8-4).

The purpose of this chapter is to explain OPEN's configuration scheme, and how the various configuration helpers fit into it. We also describe another helper, the Installation Menu. There is a guided tour in an earlier chapter (see page 3-69), but we give the details here. Finally, we document the general purpose configuration tool, the Config Editor.

A Definition of Terms

Configuring OPEN for making the measurements that you want covers a variety of issues, including the proper calibration constants for the sensors being used, the mix of variables you want stored, and the particular equation set that you deem appropriate, to name but three. All of these are handled by one vehicle: the configuration file.

Configuration Files

When OPEN is first run, one of the things that happens is the selection and implementation of a *configuration file*. This may happen automatically, or may involve user assistance. If multiple configuration files exist, the user is asked to select one from a menu (Figure 16-1). The default configuration file is named Factory Default, and if that is the only one found, it is used without first prompting the user to select it.

So what, then, is a Configuration File? It is simply a collection of configuration commands that specify changes to the default settings. While there are about 30 possible configuration commands, usually only a few are found in a configuration file.

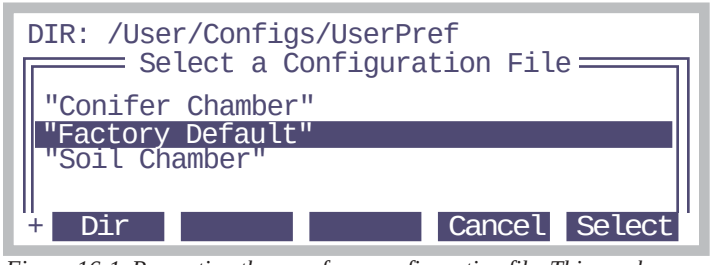


Figure 16-1. Prompting the user for a configuration file. This can happen at power up, or when Resetting to User Configuration *(from the Reset Menu, found in the Config Menu).*

Configuration Commands

Three sample configuration commands are listed in Figure 16-2.

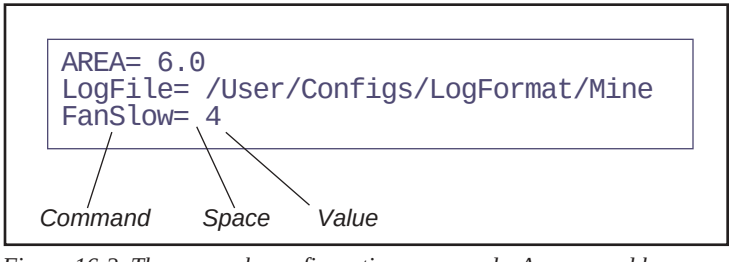


Figure 16-2. Three sample configuration commands. A command has an = as the last character, and is followed by a space, then the value.

A configuration command consists of a string terminated by =, followed by a space. The number or text that comes next is the value of the command. The complete list of configuration commands, and what they mean, is given in **Configuration Command Summary** on page 16-36.

Master and User Files

OPEN has stored away in the file system a *master file* of configuration commands, and these comprise the factory default settings[1]. Whenever OPEN implements a configuration, this master file is first read and implemented.

[1] In case you're interested, this master "file" is actually two files: "/Sys/Open/Config Items 1", and "/dev/parm0".

Following that, a *user file* is read and implemented. If the user file is empty (a legitimate possibility, and in fact the file Factory Default is empty), then the resulting configuration is the factory default. Usually, though, the user file consists of a handful of configuration commands that, because it is read last, override the corresponding settings of the master list. Figure 16-3 illustrates the relationship between the master and user files.

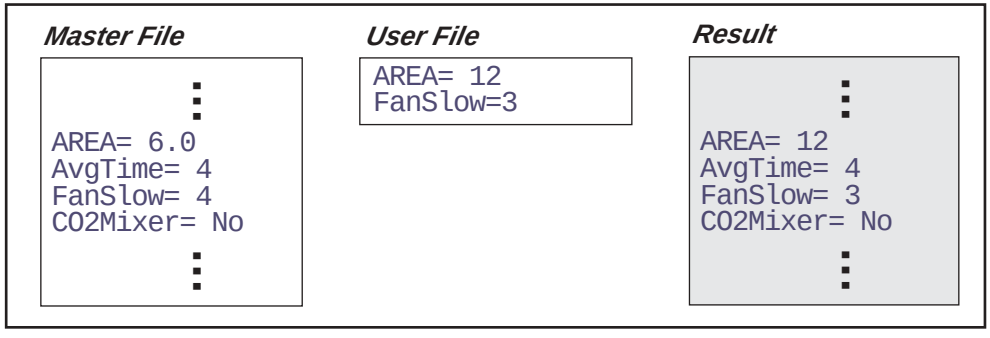


Figure 16-3. The master list is implemented first, then the user list. User list commands thus override master list settings.

The complete master file is a combination of the files /dev/parm0, and the file "/Sys/Open/Config Items 1", shown below.

```
AREA= 6
StomRat= 1
LightSource= "Sun+Sky" 1.0  0.19
AvgTime= 4
Prompts= off
UserChan= 0 0 0
BndBrdCorr= YES
FanSlow= 4
LogDelimiter= 44 (comma, for IBM)

ComputeList= "Default"
Configs= "Factory Default"
Displays= "Std Display"
StripDefs= "Std Gas Exchange"
StableDefs= "Std Stability"
LogFormat= "Std Output"
PromptList= "Default (none)"
FlrParams= "Std Settings"
LogFile= "Data"
BLCond= "/Sys/Lib/StdBLCTable"
AutoProgs= "_"
PlotDefs= "_"
UserDir= "/User"
```

Figure 16-4. Listing of "/Sys/Open/Config Items 1".

Making Configuration Files

The Installation Menu

The simplest method of building a configuration file appropriate for some configuration is to use the Installation Menu (Figure 16-5). Anytime you wish to add the calibration of an accessory, such as an external quantum sensor, chamber light sensor, LED light source, etc., or generate a configuration file that makes use of a different chamber or other accessory, then select the item in question from the Installation Menu, and follow the ensuing directions.

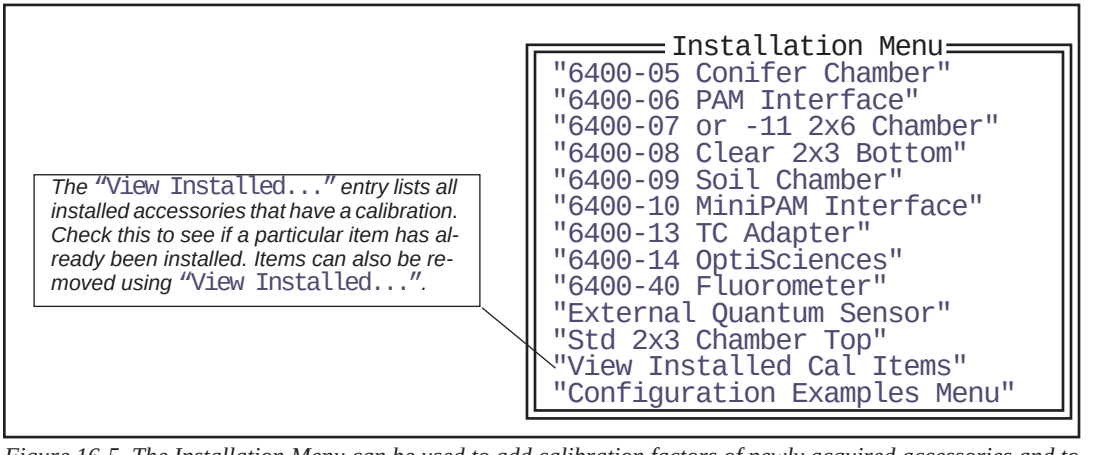


Figure 16-5. The Installation Menu can be used to add calibration factors of newly acquired accessories and to generate or modify a configuration to use that accessory.

The Installation Menu is a powerful tool that can save you time and grief. Suppose, for example, you were going to use a clear bottomed 2x6 chamber, and wanted to configure the LI-6400 accordingly. If you were some sort of a LI-6400 guru, you might be able to go to the Config Editor and build one from scratch[2], keeping in mind the following issues: leaf temperature is no longer measured, so you need an energy balance computation; it's a 2x6 chamber, so the boundary layer conductance is different; if you're measuring needles instead of broadleaves, the boundary layer is different again. However, a more sane person could just go to the Installation Menu, select "6400-08 Clear 2x3 Bottom", answer some questions, and produce a proper configuration, without having to know any arcane details.

[2] This guru wouldn't try. He'd just use the Installation Menu.

Two Examples

We present two examples of using the Installation Menu to build configuration files.

Example: 6400-02B LED Source

Suppose you already have a 6400-02 red LED source, but you have acquired a 6400-02B source as well. This example will show you how to install it, and easily switch configurations from one to the other.

1 Select "6400-02 or -02B LED Source"

This is in the `"Installation Menu"`, which is in the Config menu.

2 Is the calibration already installed?

You'll be shown a list of installed 6400-02 and -02B calibration factors (Figure 16-6). If the one you are using is shown, press **Y** and go to step 4. Otherwise, press **N** to indicate that it is not shown, then **Y** to add it now.

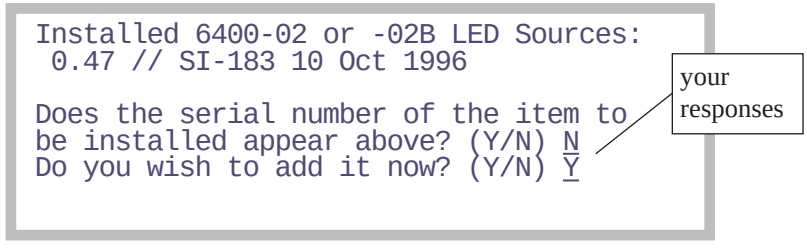


Figure 16-6. The LED source installation program shows the currently installed sources, and gives you the chance to add to the list.

3 Add the calibration information

You'll need the calibration sheet (Figure 16-7) for the next three questions:

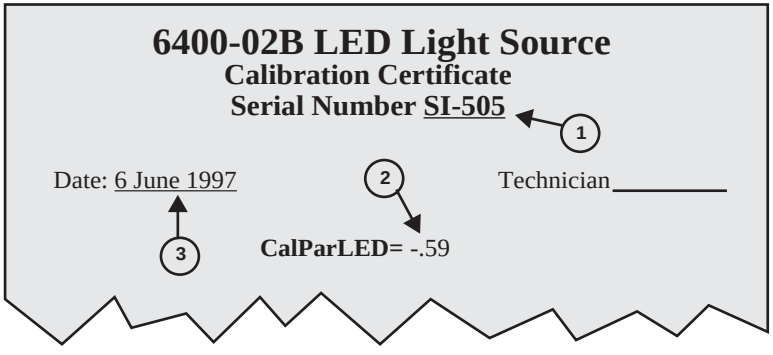


Figure 16-7. The required calibration information is available from the light source's calibration sheet.

This information is entered using three prompts:

```
6400-02B or -02B LED Light Source
Enter serial number: SI-505
Enter calibration value: -0.59
Enter cal date: 6 June 1997
```

When prompted

```
"Is this OK",
```

confirm the information entered with a **Y** or **N**. (If you press **N**, you will be asked for the three items in step 3 again.)

Once a 6400-02 or -02B calibration is installed, you can readily change any existing configuration to use it, by using the Light Source Control (page 8-4). However, you may wish to build a configuration for it, which you can do with the Installation Menu. Continuing our example…

4 Build the configuration file

Press Y when asked

```
Build a configuration file? (Y/N)
```

5 Specify the chamber bottom

You'll be asked to specify which chamber bottom is being used (either the standard 2x3 chamber bottom or the clear bottomed 6400-08 2x3).

6 Specify the light source

You'll be asked to select a light source from a menu (Figure 16-8) which will contain all installed LED sources, in addition to some standard sources.

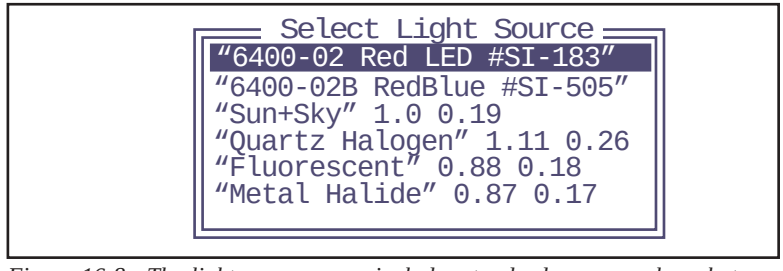


Figure 16-8. The light source menu includes standard sources, plus whatever 6400-02 and -02B sources have been installed.

7 Leaf Temp: Measured or Energy Balance (M/E)?

Specify how the leaf temperature is to be made (if you selected the clear bottomed chamber, it won't ask this)

```
Leaf Temperature:
Measured or Energy Balance ? (M/E) E
```

If the chamber has a clear bottom or is the conifer chamber, an energy balance is always used, since there is no leaf temperature thermocouple present. Otherwise, you have a choice between measuring leaf temperature directly, or using the thermocouple to measure air temperature and computing the leaf temperature with an energy balance. The latter is useful for non-flat plant material, such as needles. The response here determines a number of configuration commands, including "ComputeList=", "LogFormat=", and "Display="

8 Leaf Type: Needles or Broadleaves (N/B)?

The response to the question

```
Enter leaf type
Needles or Broadleaves (N/B)?
```

determines if the boundary layer conductance will be computed from a look-up table (broadleaves), or if a fixed value is used (needles). The "BLCond=" configuration command is set by this response.

9 **View the config file**

You are then shown what the config file will look like (Figure 16-9).

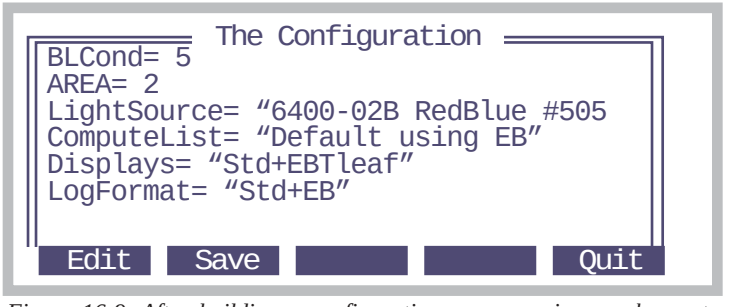


Figure 16-9. After building a configuration, you are given a chance to view, edit, and save it.

10 **Store the Config File**

Press **Store** (**f2**) to store the configuration. A Standard File Dialog (described in **Standard File Dialog** on page 5-9) will appear with a default name which captures the essence of the configuration (Figure 16-10).

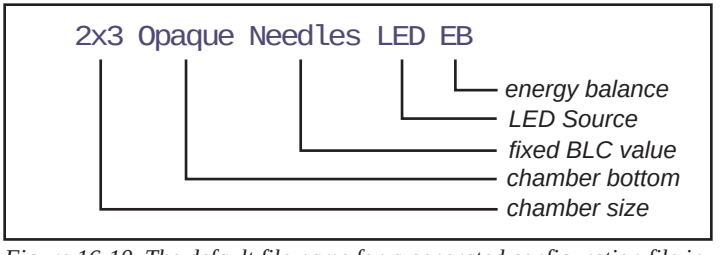


Figure 16-10. The default file name for a generated configuration file includes several details. You can use it, modify it, or invent your own name.

You may wish to modify the default name to specify that it is for the Red/Blue LED Source.

To implement the configuration, select "Reset to User Config" from the Reset Menu in the Config Menu.

■ **Example: 6400-09 Soil Chamber**

The 6400-09 has no calibration factors to install, so the installation is a simple 2-step process:

1 **Select “6400-09 Soil Chamber”**
From OPEN’s main screen, press **Config Menu** (**F2**), then select the Installation menu.

2 **View andStore the file**
Modify the default “Soil Chamber” name if you desire.

The installation program will then copy some other support files from /sys to /user, and you are done.

None of the configurations that can be generated by the Installation Menu actually take effect until you select them after a) power on, or b) you select "Reset To User Configuration", found in "_Reset Menu", in the Config Menu.

Installation Menu: Behind the Scenes

What actually happens when you use the Installation Menu to install a new sensor, or build a new configuration file?

Installing a Sensor

Any item that has a calibration associated with it (chamber tops, LED sources, external PAR sensors) needs to be “installed”, which simply means that there needs to be an entry in a certain file that contains all the calibration information. That file is “/dev/parm0”, and its contents make up part of the master

list illustrated in Figure 16-3 on page 16-4. A typical "/dev/parm0" is shown in Figure 16-11.

```
// Calibration Data G103950109
CO2Mixer= YES
CalCO2= 0.21732 2.1807E-05 1.9303E-08 -3.2863E-12 3.272E-16
CalH2O= 0.0063802 1.9083E-06 -2.4303E-11
CalZero= -4.0 -2.3
CalFlow= 0.35307
CalPress= 88.522 0.005458
CalParGaAs= 0.77 // GA-103 20 Jan 1995
// CalParGaAs= 0.70 // GB-244 07 Jul 1996
CalParLED= 0.48 // SI-120 10 Jun 1995
CalParLED= -0.54 // SI-501 27 Jul 1997
CalParOut= 5.78 // Q13436 27 Aug 1990
ThisUnit= "PSC-0103"
Serviced= "9 Jan 1995"
```

Figure 16-11. Listing of a typical "/dev/parm0". This file contains calibration data for all sensors. When you install sensors using the Installation Menu, the data you enter winds up here. When there are multiple sensors of one type, the "extra" ones can either appear as comments (line starts with //), or come first in the list (the last one will take precedence).

What Calibrations Are Installed?

The Installation Menu entry "View Installed Cal Items" scans "/dev/parm0" for the following entries (commented or not):

```
CalParGaAs=
CalParLED=
CalParOut=
```

and presents the results in a list.

To Determine What Calibrations Are Installed:

1 **Access the Installation Menu**
Highlight it, and press **enter**.

2 **Select "View Installed Cal Items"**
Press **end** then ↑ to highlight it, and press **enter**.

3 View the list

You'll see a list that looks something like Figure 16-12. It will include the three classes of light sensor calibrations.

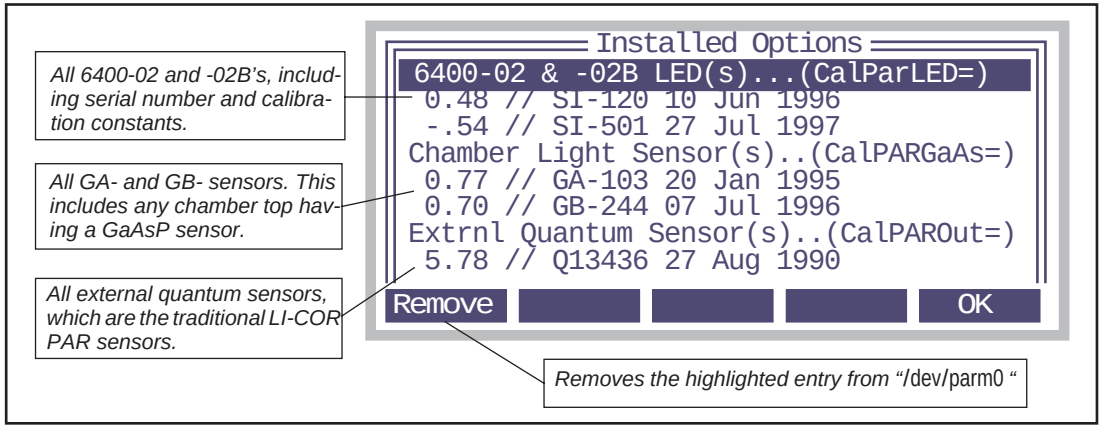


Figure 16-12. A summary of light sensor related calibrations is available by selecting "View Installed Cal Items" in the Installation Menu.

4 Return to the Config Menu

Press escape until you get there.

Editing /dev/parm0

When all else fails, sometimes you just have to get in there and adjust something directly. You can. Access the Filer, locate the /dev directory, highlight the file parm0, and press E (for edit). You'll be using **Standard Edit** on page 5-13.

Using Energy Balance

When you build a configuration file using the Installation Menu, and are asked if leaf temperature is to be measured or computed, selecting the latter will cause an energy balance to be implemented. Energy Balances are discussed in Chapter 17, and their implementation via the ComputeList is discussed in Chapter 15. There are three standard energy balance files that might get used:

```
Clear Bottom EB
Conifer Chamber EB
Default using EB
```

The configuration builder sets the ComputeList= configuration command to the appropriate file, based on what chamber you are using, and/or how you answered the energy balance question. The 6400-05 Conifer chamber, and the clear bottom chambers, always use an energy balance.

Boundary Layer Conductance

When the Installation Menu's configuration builder asks you about broadleaves or conifers, it is determining what default area to use, and how to handle boundary layer conductance calculations. If you are measuring broadleaves, then the configuration command BLCond= will be set to one of these two lookup tables:

```
StdBLCTable
BLCTable_2x6
```

Both are stored in the "/Sys/Lib" directory. If you specified conifers, a fixed value of 5 is used instead of the lookup table. (If you have a better value, you can use the Config Editor and enter it.)

Modifying Config Files

The general purpose tool for building or editing a user configuration file is the Config Editor (page 16-16). Frequently, however, it is not necessary to resort to this to accomplish what you want. The Logging Control screen (page 9-8), Prompt Control screen (page 9-15), and LightSource Control screen (page 8-4) are examples of ways to affect the user config file without using the Config Editor.

Table 16-1. Methods of modifying config files

Action	Affected Configuration Command
AREA (**F1** level 3) New Msmnts Mode	**AREA=**
STOMRT (**F2** level 3) New Msmnts Mode	**StomRat=**
Display QuikPik (**F2** level 6) New Msmnts Mode	**Displays=**
SaveAs or **OK** in Display Editor	
Pick Source (**F1**) in Light Source Control	**LightSource=**
LogList QuikPik (**F2**) in Logging Control	**LogFormat=**
SaveAs or **OK** in LogList Editor	
SaveAs or **OK** in PromptList Editor	**PromptList=**

Table 16-1. (Continued) Methods of modifying config files

Action	Affected Configuration Command
"Choose a Compute List" in ComputeList Menu	**ComputeList=**
GRAPH QuikPik (**F2** level 4) New Msmnts Mode	**StripDefs=**
ReadAll, **SaveAll** in Real Time Graphics Editor	
SaveAs or **OK** in FlrEditor	**FlrParam=**
Flr QuikPik (**F1** level 8) New Msmnts Mode	
Open or **Save** in Stability Editor	**StableDefs=**
SaveAs or **OK** in Config Editor	Any command

An indication that the current configuration has been changed, but is not stored, is the presence of the CONFIG flag in OPEN's main screen (Figure 16-13). An unstored configuration will be lost at power off, so if you wish to keep the changes for the next session, or to store the changed configuration as a new configuration file, then this can be done with the Config Status, described on page 16-15.

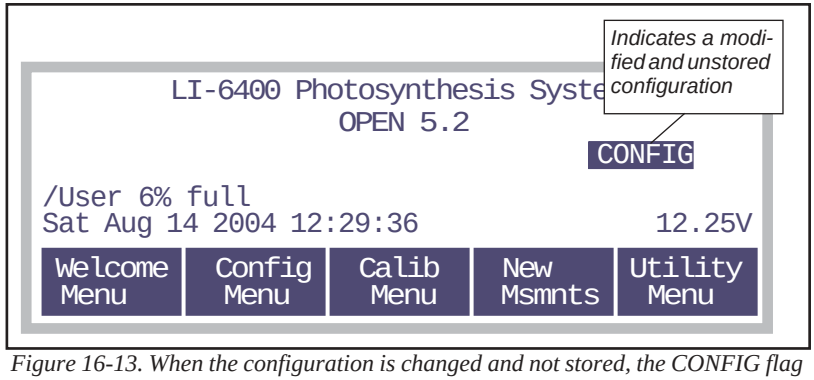
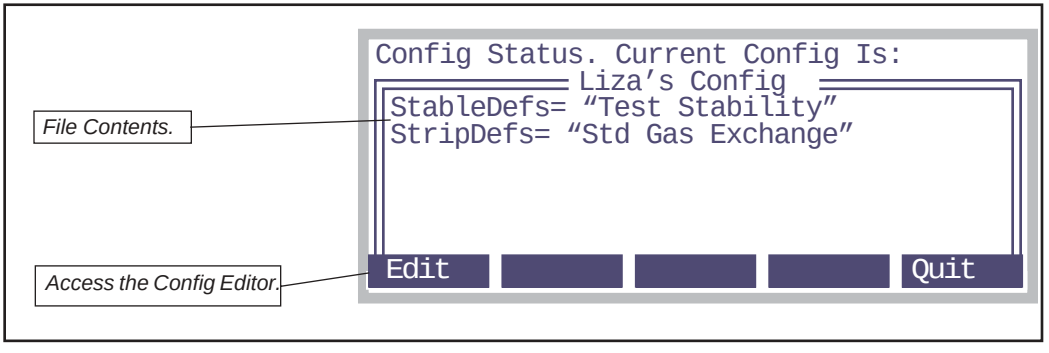


Figure 16-13. When the configuration is changed and not stored, the CONFIG flag will appear in OPEN's main screen. Too see what the change is, and to store the configuration, go to Config Status in the Config Menu.

Config Status

"Config Status" in the Config Menu shows a screen that indicates the current user config file name, and what changes, if any, have been made since this file was last stored (Figure 16-14).

If the current configuration file is saved:



If the current configuration file is NOT saved:

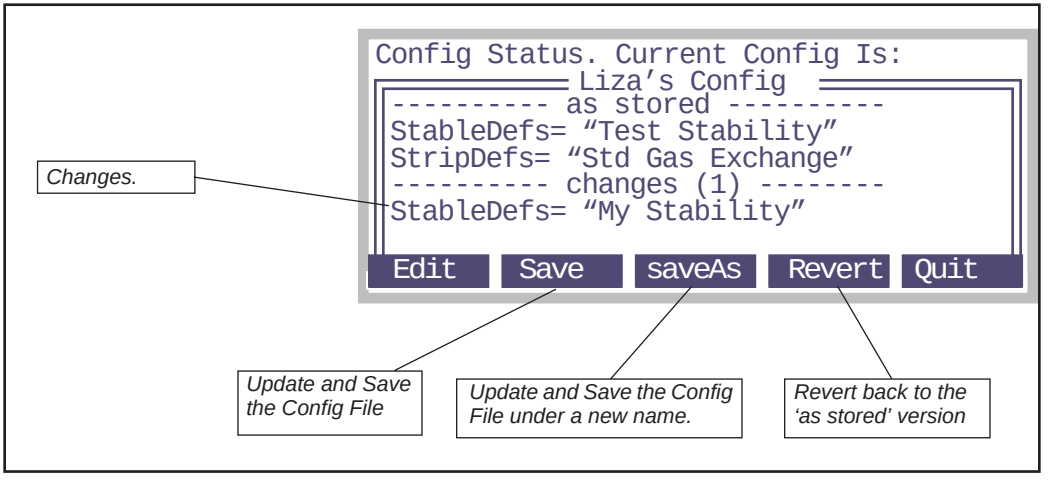


Figure 16-14. The Config Status screen shows the user config file, and any changes.

Note in the example (Figure 16-14), there is one change. The *StableDefs*= file name has changed.

There is another reason a configuration can be considered unsaved. Suppose, in the example in Figure 16-14, that the *StableDefs*= file is still the original,

"Test Stability", but Liza changes her stability definition and doesn't store the changes. In that case, the configuration status will show an asterisk by the file name (Figure 16-15).

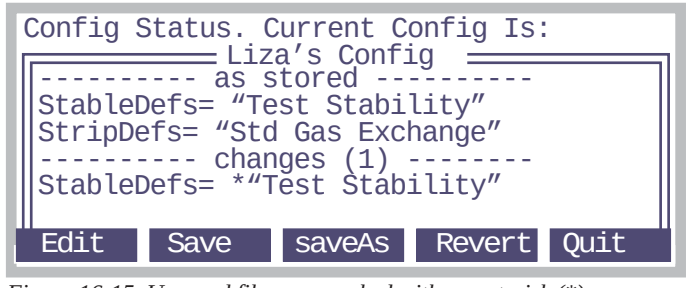


Figure 16-15. Unsaved files are marked with an asterisk (*).

When you save the overall configuration, by pressing **Save** or **saveAs**, you will be given the opportunity to save any unsaved file (marked with a *). You will be prompted for each as illustrated in Figure 16-16.

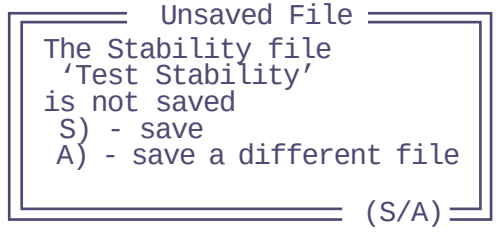


Figure 16-16. You will be prompted for saving each unsaved file, when saving a configuration.

Config Editor

The Config Editor is essentially a dialog box, in which the user is shown the current user configuration items in a list. The user can make modifications, create new configuration files, recall and edit other configuration files, and implement a different configuration.

The Configuration Editor is driven by function keys, whose definitions are

given in Table 16-2.

Table 16-2. The Config Editor's function keys

Key	Action
Add	If you don't see an item in the config file, you can add it with this key. The items of the master file are shown to you as a menu, and you may select the desired item to be added to your config file. Once it is added, you can then edit it if necessary.
Edit	Change the highlighted entry in the config file. What actually happens depends on the item being edited. **Edit** might access a menu of possibilities (e.g. LogDelimiter=), or prompt for a new value (e.g. AREA=), or access the Standard File Dialog box (e.g. Displays=).
Remove	Remove the highlighted entry. This is functionally equivalent to (and more efficient than) setting it to the factory default value.
Disable	Disables a config command without removing it by turning it into a comment. (Config comments start out with two slashes - //).
Enable	Removes the slashes from a commented config command.
SaveAs.	Store the file under a new name. You are prompted for a file name using the Standard File Dialog box, and if a file is specified, the config file is stored as that file name.
Open...	The user is prompted for a file name using the Standard File Dialog box, and if a file is selected, the file is read in to be edited, replacing the current config file.
Cancel	Abandon the config editor, and make no configuration changes.
OK	Implement the current contents of the config editor, and update the file.

A complete list of configuration commands is given below in **Configuration Command Summary**.

When the Config Editor is exited using **Cancel**, no configuration changes are made. When the Config Editor is exited using **OK**, the an exit screen (Figure 16-17) will appear if editing changes have been made to the config file. You can save the file, save the configuration as a new file, or abandon

the whole thing.

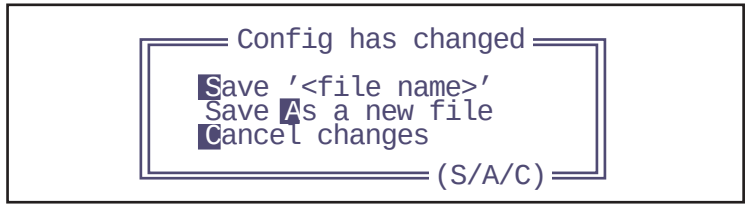


Figure 16-17. The exit options when leaving the Config Editor after changing a configuration.

Pressing **S** or **A** will implement the configuration file; the only difference is what name the configuration file will have. Pressing **C** will be the same as exiting via Cancel.

The Reset Menu

"Config File (Re-)Install"

This program will set some or all of the default configuration files back to factory defaults. It can be used to undo changes you might have made to these files. After the opening message, the program shows a list, as in Figure 16-6.

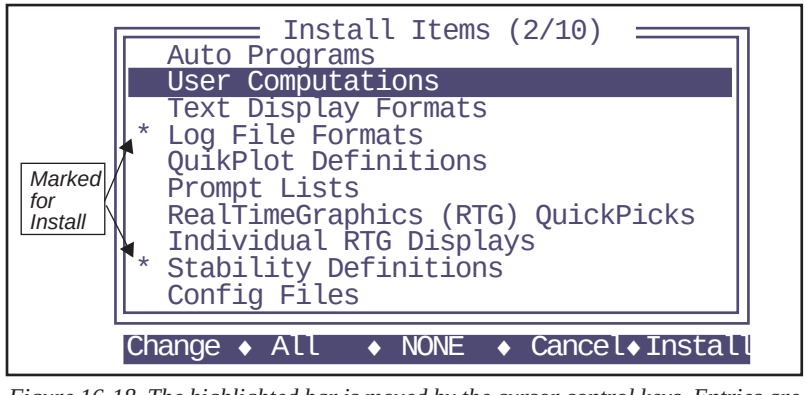


Figure 16-18. The highlighted bar is moved by the cursor control keys. Entries are marked (*) by pressing Change. To install the marked categories, press Install.

Press **Change** to mark or unmark the highlighted entry.

The files involved are listed in table Table 16-3.

Table 16-3. Files installed by Config File Reinstall.

Category	Destination directory	Files to be overwritten
AutoPrograms	/User/Configs/AutoProgs	AutoLog LightCurve A-CiCurve TimedLamp Remote Control
User Computations	/User/Configs/Comps	Default Default using EB Clear Bottom EB Conifer Chamber EB
Display Formats	/User/Configs/Displays	Std Display Std+EBTleaf Diagnostic
Log File Formats	/User/Configs/LogFormats	Std Output Std+EB
QuikPlot Definitions	/User/Configs/PlotDefs	Photo, Cond vs Obs Light Curve A Ci Curve
Prompt Lists	/User/Configs/Prompts	Default (none)
RTG Definitions	/User/Configs/RTG_Defs	Std Gas Exchange
Individual RTG Screens	/User/Configs/RTG_Defs/Graphs	PHOTO-Ci Tblk, Tair, Tleaf H2OR, H2OS, Flow CO2R, CO2S PHOTO, COND
Stability Definitions	/User/Configs/StableDefs	Std Stability
Config Files	/User/Configs/UserPrefs	Factory Default

"Reset to Factory Defaults"

This program is designed to put the LI-6400 into a "known" state quickly. It simply implements the master list without any user list. It is functionally equivalent to "Reset to User Config", and selecting "Factory Default" (provided "Factory Default" is an empty file).

The user is prompted for one thing:

```
Re-install config files? (Y/N).
```

Pressing **Y** will reinstall the configuration files. Pressing **N** will skip that, and pressing **escape** will abort.

"Reset to User Configuration

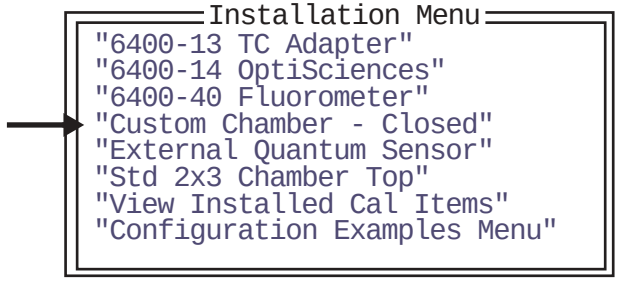
This program allows the user to select a configuration (provided there is more than just "Factory Defaults" in the "Configs/UserPrefs" directory. After implementing the master list, the user is prompted to select a configuration (Figure 16-1 on page 16-3), and it is then implemented.

Custom Chamber - Closed

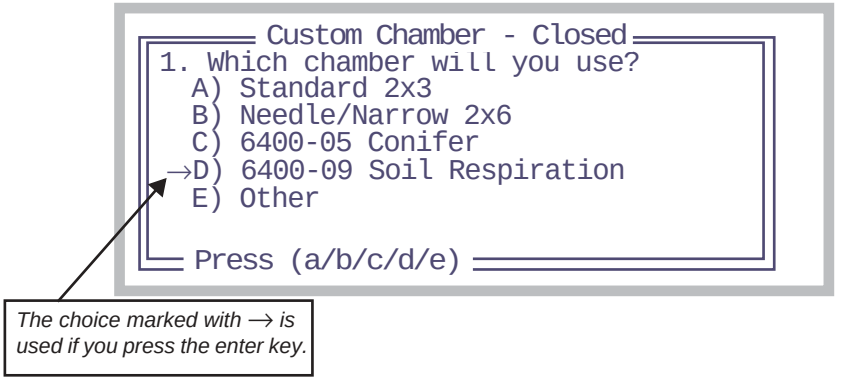
One of the items in the Installation menu is "Custom Chamber - Closed", a configuration builder for making a closed system for flux measurements based on a time rate of change of CO_2.

Installation

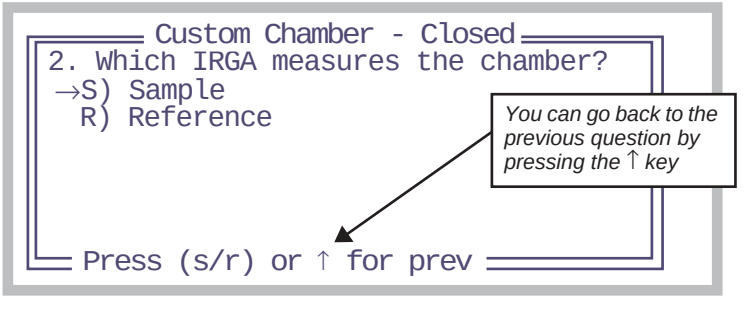
The Installation Menu has an entry named 'Custom Chamber - Closed" that will create a configuration based on your responses to a series of questons.



1 **"Which chamber will you use?"**
The purpose of this question is to get the default volume and area. If you pick **E**, you will be prompted for these values. Note that you can change (and store) volume and area later - this just gets the initial default values.



2 **"Which IRGA measures the chamber?"**
The answer here is likely Sample, but if your chamber is connected to the reference side, pick R.



3 **"Chamber air temp measured by...?"**
A closed system requires measuring the air temperature in the chamber. 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 Figure 16-32).

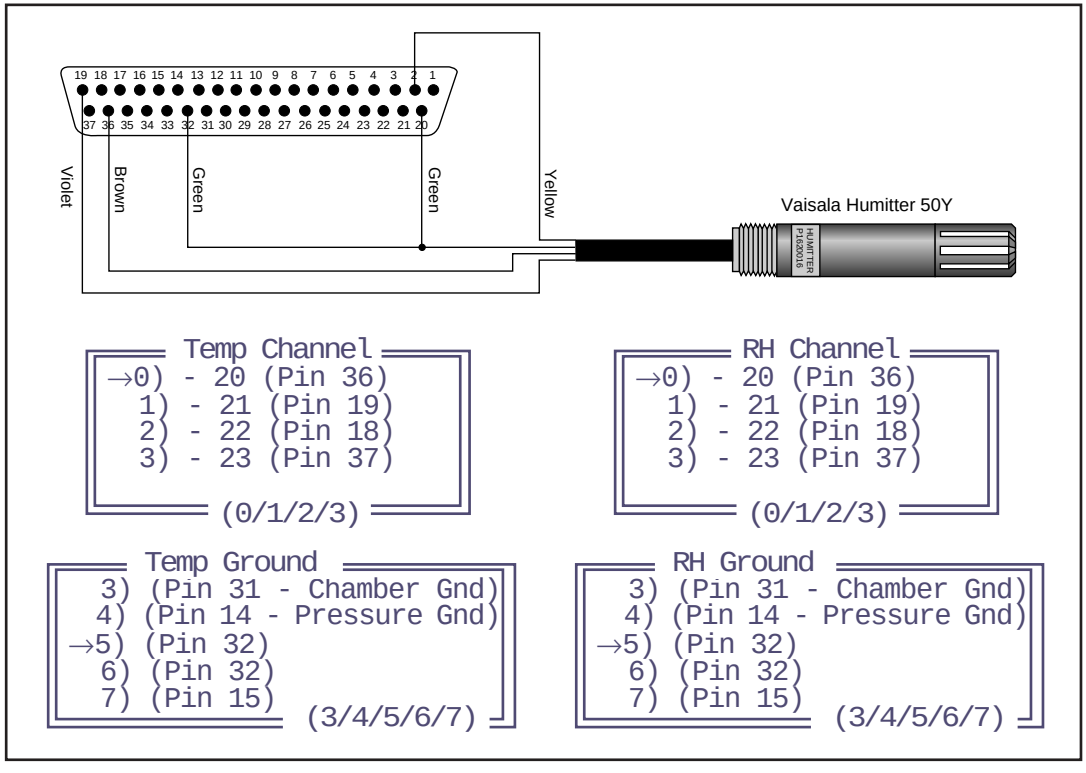


Figure 16-19. Using a Vaisala Humitter 50Y for chamber temperature.

D) If something else is connected to the console auxiliary port, you will be prompted for the channel and ground for Temperature (Figure 16-32, but 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 16-20.

```
Enter the Transform for Temp
DegC= V * 0.1 - 40
(Use 'V' for raw Volts)

Enter the Transform for RH
DegC= V * 0.1
(Use 'V' for raw Volts)
```

Figure 16-20. Entering the transform. Use upper case V for the raw signal.

4 **"Do you want Tsoil measured with the 6400-09 Soil Temp probe?"**
If you are using the 6400-09TC soil temperature probe, pick Yes.

5 **"Do you want an ON/OFF FCT key control for fans..."**
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). You would wire the mating connector (part number 310-04300) to use the 2 pins (Figure 16-21) that normally supply power to the LED light source fan. (*Do not use the pins that supply power to the LEDs*).

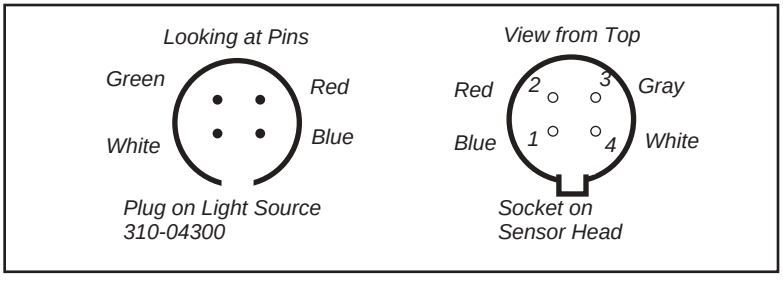


Figure 16-21. The pins that go into holes 1 and 2 are the ones to use for wiring a fan. Blue (1) is ground, Red (2) is power.

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 without 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 **"Do you wish to have the Pump ON for a normal measurement?"**
If you respond No to this question, your configuration will have the following features:

- **Default flow control state will be Pump OFF.**

- **"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 you can quit, or go back to previous questions to check your answers. If you are ready to continue, press Y.

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.

The name will be a file name, so if you enter any of the following seven deadly characters, they will be converted to an underscore (_):

```
" : * ? ; \ /
```

The following files will be created (where XXX is what you entered in Step 8):

```
/User/Configs/Comps/XXX.LPL
/User/Configs/CustomClosed/XXX
/User/Configs/Displays/XXX
/User/Configs/LogFormats/XXX
/User/Configs/Prompts/XXX
/User/Configs/RTG_Defs/XXX
/User/Configs/UserPrefs/XXX
```

The directory /User/Configs/CustomClosed is new, and is used to store settings (volume, area, etc.) for this configuration.

Using The Closed Configuration

The following guide assumes the closed system configuration was named "Closed Flux System".

Once "Custom Chamber - Closed" has been run, your named configuration file will appear in the list of available configurations that appears when OPEN first runs, and when you change User Configurations (Config Menu -> _Reset Menu -> _Reset to User Configuration).

The important part of this configuration is defining the file "/User/Configs/Comps/Closed Flux System.LPL" as the active ComputeList file. The other parts of the Closed Flux System configuration define the display format, log

format, etc. The main screen (Figure 16-22) indicates the version of the Closed System it is using, and the version of OPEN that it is running on.

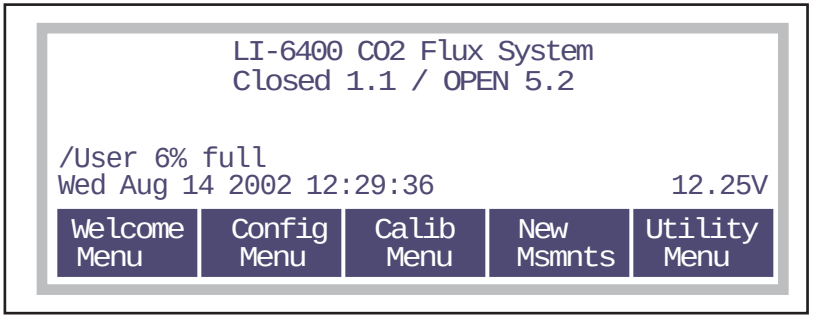


Figure 16-22. A configuration generated by Custom Chamber - Closed.

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 16-23).

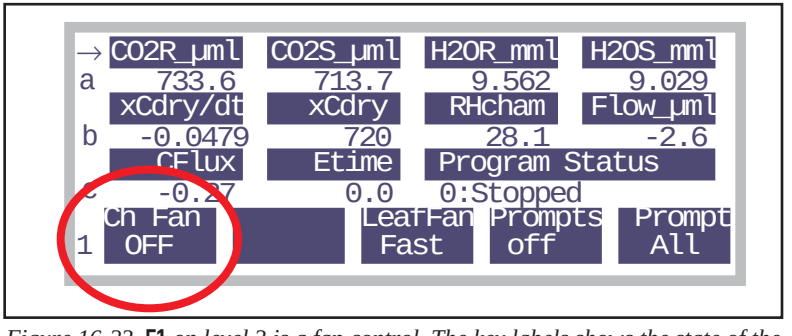


Figure 16-23. **F1** *on level 3 is a fan control. The key labels shows the state of the chamber fan (assumed attach to the light source control connector.)*

The main control for the Flux System is on function key level 7

(Figure 16-24).

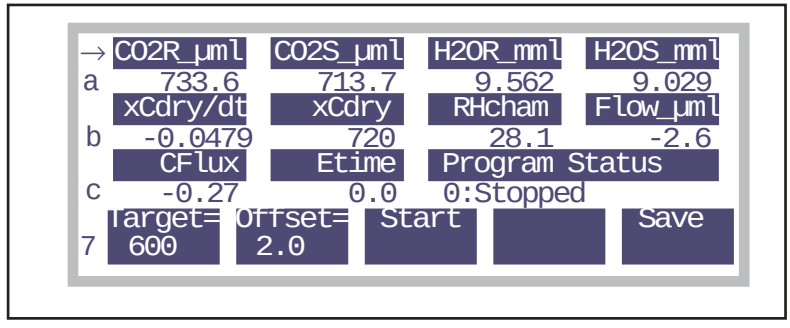


Figure 16-24. Controls for Closed Mode are on function key level 7.

The **Target=** key (**f1**) prompts for the target CO_2 concentration (for the final regression), the total measurement time (s), and the logging interval (s).

The **Offset=** key (**f2**) prompts for Offset (e.g. collar height on which the chamber sits), system volume (total volume without collar), and chamber area. The relation between total volume and these three parameters is shown below

$$V_{tot} = V_{sys} + A_{cham}h \qquad (16\text{-}1)$$

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 **Save** key (**f5**) allows you to retain the current values of all of the items prompted for in keys 1 and 2 for the next time power on defaults. If the present values differ from the defaults, an asterisk will appear as part of the **Save** label.

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**)

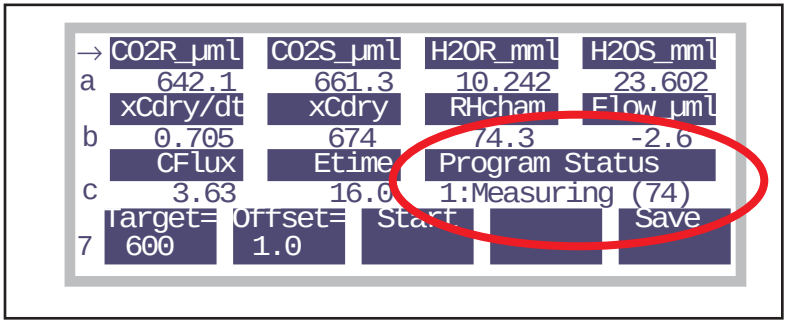


Figure 16-25. Program Status and ETime (elapsed time) indicate the state of a measurement.

Once the measurement is done, the display will appear as in Figure 16-26).

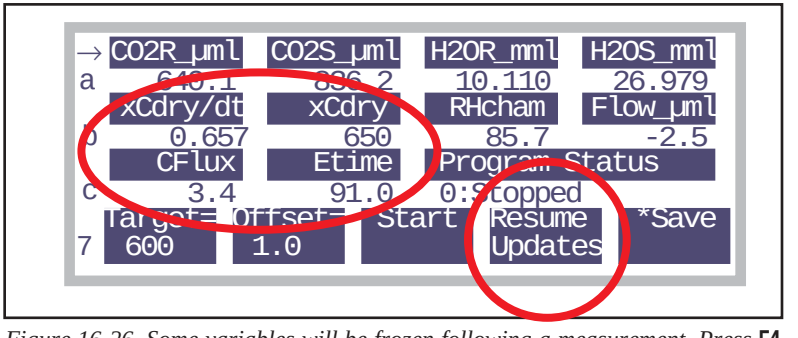


Figure 16-26. Some variables will be frozen following a measurement. Press **F4** *to resume updates for them.*

The some of the displayed variables (Table 16-5 on page 16-29) 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 16-4 on page 16-28.

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 16-4. Real time measurement variables defined by Closed.LPL

Label	Symbol	User ID	Description
Dilution	D	416	$D = 1 - \frac{W}{1000}$, where W is the sample cell H_2O concentration (H2OS)
Cdry	C_{dry}	412	$C_{dry} = \frac{C}{D}$, where C is the sample cell CO_2 concentration (CO2S)
TCham	T_c	411	Chamber temperature. Defaults to the channel normally used for leaf temp (Tleaf).
Density	ρ	413	Density of the air: $\rho = \frac{1000P}{8.314(T_c + 273.15)}$
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 16-5. 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	$\overline{\frac{dC_{dry}}{dt}}$	405	Mode 1: Averaged $\frac{dC_{dry}}{\overline{dt}}$ Mode 2: Regressed to C_{target}
CFlux	F_c	401	CO_2 flux computed from the above variables: $F_c = \frac{\bar{\rho}\bar{D}V_{tot}}{100A_{cham}}\frac{\overline{dC_{dry}}}{\overline{dt}}$

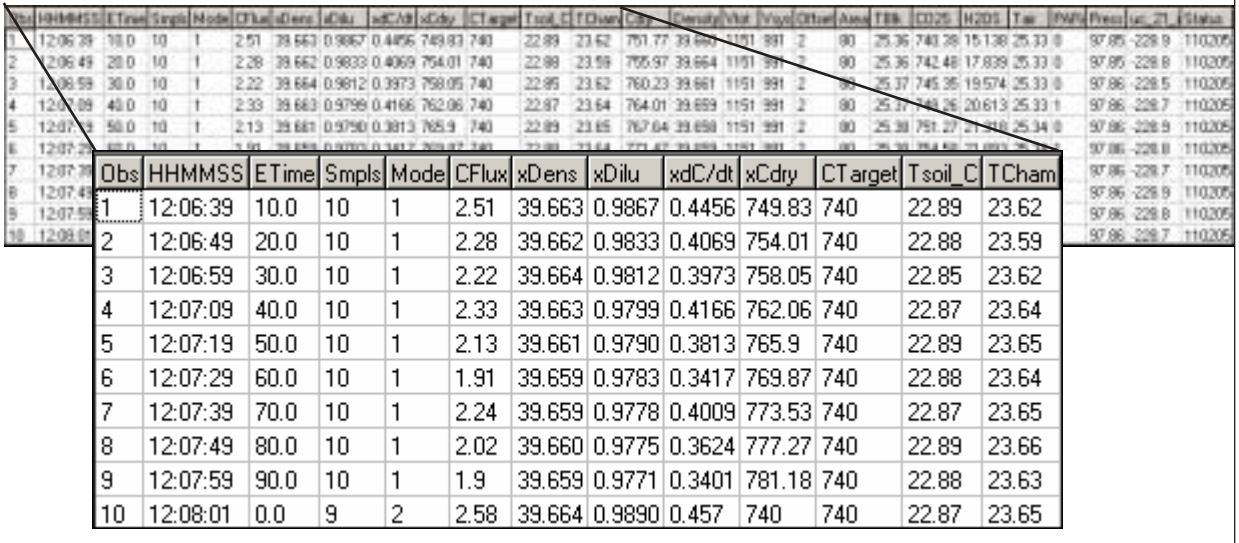
During a measurement (Mode = 1), the Table 16-5 variables come from 10 second running statistics. Each time a reading is logged (every *LogInt*), the current value of $\overline{D}$, $\overline{C}_{dry}$, $\overline{\frac{dC}{dt}_{dry}}$, and $\overline{\rho}$ are added to an internal vector.

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 16-27 and 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 5.2"
"Fri Apr 30 2004 12:06:11"
"Unit=","PSC-999"
"ComputeList=","/User/Configs/Comps/Closed Flux System.LPL"
"BLCTable=","/Sys/Lib/StdBLCTable"
"LightSource=","Sun+Sky",1,0.25
"LogFormat=","/User/Configs/LogFormats/Closed Flux System"
"LogCodes=",-35,-21,414,431,430,401,403,404,405,406,435,410,411,412,413,420,…
"PromptList=","/User/Configs/Prompts/Closed Flux System"
"Stability= (CO2S 15s SLP<1)(H2OS 15s SLP<1)(Flow 15s SLP<1)"
"12:06:12 "
$STARTOFDATA$
"Obs","HHMMSS","ETime","Smpls","Mode","CFlux","xDens","xDilu","xdC/dt","xCdry",…
1,"12:06:39",10.0,10,1,2.51,39.663,0.9867,0.4456,749.83,740,22.89,23.62,751.77,…
2,"12:06:49",20.0,10,1,2.28,39.662,0.9833,0.4069,754.01,740,22.88,23.59,755.97,…
3,"12:06:59",30.0,10,1,2.22,39.664,0.9812,0.3973,758.05,740,22.85,23.62,760.23,…
4,"12:07:09",40.0,10,1,2.33,39.663,0.9799,0.4166,762.06,740,22.87,23.64,764.01,…
5,"12:07:19",50.0,10,1,2.13,39.661,0.9790,0.3813,765.9,740,22.89,23.65,767.64,…
6,"12:07:29",60.0,10,1,1.91,39.659,0.9783,0.3417,769.87,740,22.88,23.64,771.47,…
7,"12:07:39",70.0,10,1,2.24,39.659,0.9778,0.4009,773.53,740,22.87,23.65,775.31,…
8,"12:07:49",80.0,10,1,2.02,39.660,0.9775,0.3624,777.27,740,22.89,23.66,778.94,…
9,"12:07:59",90.0,10,1,1.9,39.659,0.9771,0.3401,781.18,740,22.88,23.63,782.80,…
10,"12:08:01",0.0,9,2,2.58,39.664,0.9890,0.457,740,740,22.87,23.65,783.46,39.661,…
```

Figure 16-27. Sample data file



Obs	HHMMSS	ETime	Smpls	Mode	CFlux	xDens	xDilu	xdC/dt	xCdry	CTarget	Tsoil_C	TCham
1	12:06:39	10.0	10	1	2.51	39.663	0.9867	0.4456	749.83	740	22.89	23.62
2	12:06:49	20.0	10	1	2.28	39.662	0.9833	0.4069	754.01	740	22.88	23.59
3	12:06:59	30.0	10	1	2.22	39.664	0.9812	0.3973	758.05	740	22.85	23.62
4	12:07:09	40.0	10	1	2.33	39.663	0.9799	0.4166	762.06	740	22.87	23.64
5	12:07:19	50.0	10	1	2.13	39.661	0.9790	0.3813	765.9	740	22.89	23.65
6	12:07:29	60.0	10	1	1.91	39.659	0.9783	0.3417	769.87	740	22.88	23.64
7	12:07:39	70.0	10	1	2.24	39.659	0.9778	0.4009	773.53	740	22.87	23.65
8	12:07:49	80.0	10	1	2.02	39.660	0.9775	0.3624	777.27	740	22.89	23.66
9	12:07:59	90.0	10	1	1.9	39.659	0.9771	0.3401	781.18	740	22.88	23.63
10	12:08:01	0.0	9	2	2.58	39.664	0.9890	0.457	740	740	22.87	23.65

Figure 16-28. Spreadsheet view of the sample data in Figure 16-27.

Note there are 9 *Mode* 1 records, followed by 1 *Mode* 2 record. Each *Mode* 1 record shows *Smpls* of 10[3]. 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 740 ppm (*Ctarget*),

[3] 1 Hz sampling and 10 second period. Normally, Smpls is 20, since the instrument collects data at 2 Hz. This particular data set was collected with LI6400Sim, which collects data at 1 Hz.

and the final value of *xCdry* reflects this.

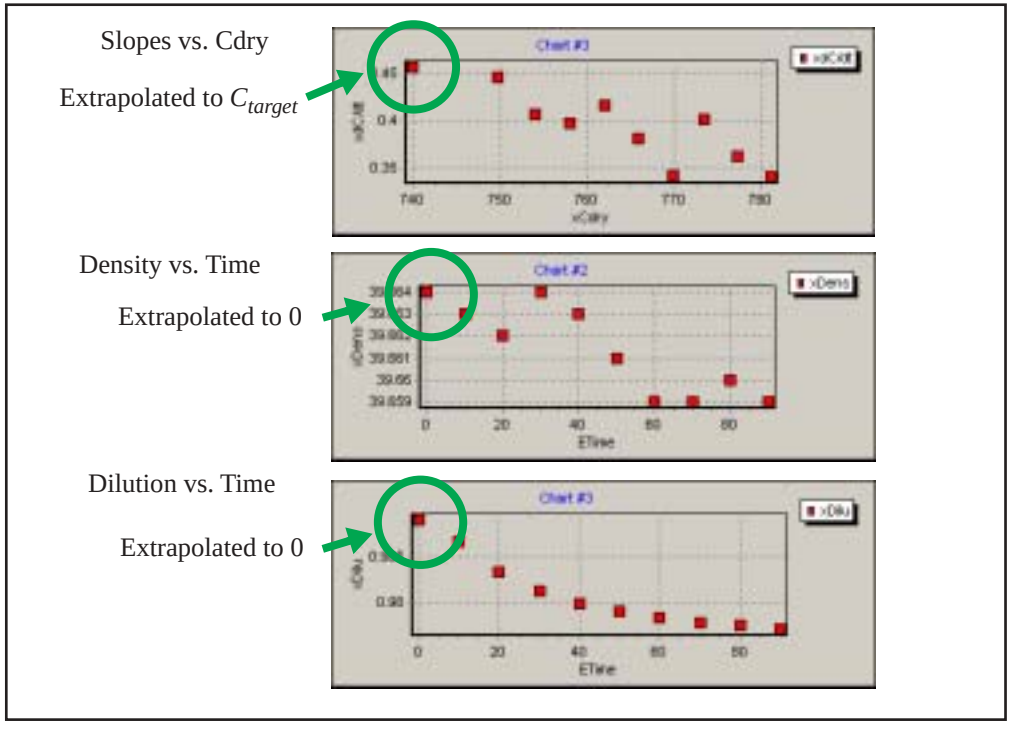


Figure 16-29. Plot of $\overline{\frac{dC}{dt}_{dry}}$, $\bar{\rho}$, *and* $\bar{D}$ *from Figure 16-5. The circled points are the Mode 2 values.*

The plot of CFlux is shown in Figure 16-30.

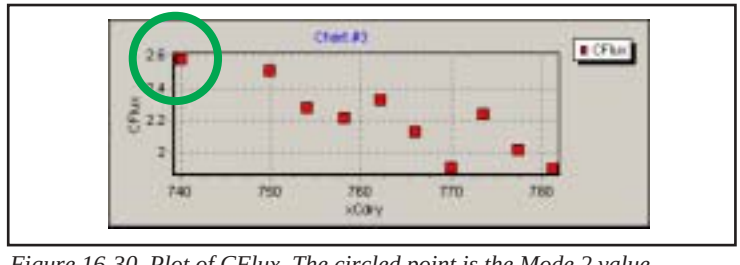


Figure 16-30. Plot of CFlux. The circled point is the Mode 2 value.

Theory

Starting from the LI-6400 soil flux equation (Equation (28-6) on page 28-41)

$$f_c = \frac{\rho v}{s}\left(\frac{dc}{dt} + \frac{c}{1-w}\frac{dw}{dt}\right) \quad (16\text{-}2)$$

we see that CO_2 flux (mol m^{-2} s^{-1}) is related to the rate of change of CO_2 c (mol CO_2 mol^{-1} s^{-1}) and the rate of change of water vapor w (mol H_2O mol^{-1} s^{-1}). (Volume v and area s are in m^3 and m^2, respectively, and density ρ is in mol m^{-3}.) 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), 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

$$c_{dry} = \frac{c}{1-w} \quad (16\text{-}3)$$

and note that

$$\begin{aligned} \frac{dc}{dt}dry &= \frac{1}{1-w}\frac{dc}{dt} + \frac{c}{(1-w)^2}\frac{dw}{dt} \\ &= \frac{1}{1-w}\left(\frac{dc}{dt} + \frac{c}{(1-w)}\frac{dw}{dt}\right) \quad (16\text{-}4) \\ (1-w)\frac{dc}{dt}dry &= \frac{dc}{dt} + \frac{c}{(1-w)}\frac{dw}{dt} \end{aligned}$$

We then substitute this result into Equation (16-2) and arrive at

$$f_c = \frac{\rho v(1-w)}{s}\frac{dc}{dt}dry \quad (16\text{-}5)$$

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_2O mol^{-1}, and CO_2 C in µmol CO_2 mol^{-1} is

$$F_c = \frac{\rho V\left(1 - \frac{W}{1000}\right)dC_{dry}}{100S\,dt} \qquad (16\text{-}6)$$

where

$$\rho = \frac{1000P}{R(T_{cham} + 273.15)} \qquad (16\text{-}7)$$

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

$$C_{dry} = \frac{C}{1 - \frac{W}{1000}} \qquad (16\text{-}8)$$

Matching Without the Match Valve

Suppose you wish to make CO_2 flux measurements in open mode. Since the chamber sits on a porous surface (turf or soil), it will be impossible 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[4] thereby isolating the sample cell 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. But, the thing that keeps you from just turning the fan off and doing a normal match is that you don't want the match valve to move, since that cuts off the reference cell from its normal flow. What's needed is an alternative match mode that leaves the match valve alone, and turns the fan off instead.

OPEN 5.2 added this, and you enable it by adding a line to the configuration

[4] 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.

file. Here is a step by step guide:

Adding fan-based matching to a configuration

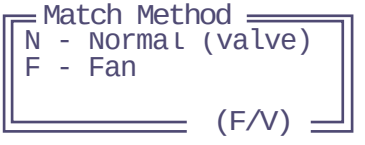
1 Select your starting configuration

2 Access the Config Editor
From OPEN's main screen, go to the Config Menu, then "Config Status", then press **Edit** (**f1**).

3 Add the MatchType= line
In the Config Editor, press **Add** (**f2**). **pgdn** a couple of times, and select the following line from the list::

```
MatchType= normal
```

4 Set MatchType=
Once it has been added, highlight the line, and press **Edit** (**f1**). Then press **F** to choose fan matching.



5 Save
Press **ok**, then **S** or **A** (for Save or saveAs).

Using Fan-based Matching

The MatchType= 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 a functioning match valve, regardless of the MatchType= setting.

In normal matching, the reference side changes, since it starts seeing 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 done on the sample side.

The match mthod is controlled by a single variable, named *matchType*. When *matchType* = 2, fan-based matching is in effect, and when *matchType* = 1,

normal matching is in effect. Thus, you can also set match type programmatically, in an AutoProgram, for example.

The match mode display indicates which type of match is in effect (Figure 16-31)

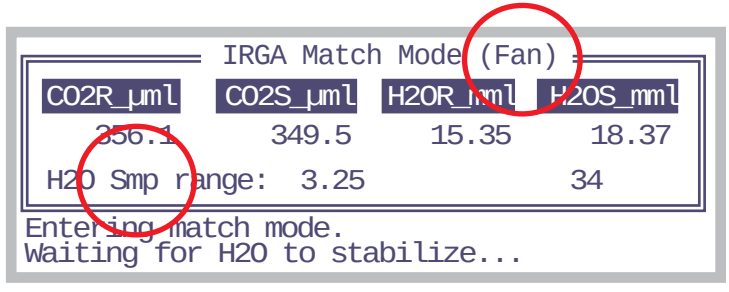


Figure 16-31. Entering match mode when matching via the fan instead of the valve.

Configuration Command Summary

The following commands can be used in a configuration file. For the most up to date list valid for whatever version of OPEN you are running, press the **Help** function key in the Configuration Menu. There may be changes described there that occurred after this writing.

AREA=

Sets the leaf area parameter. Note that this can be set in New Measurements mode as well. Leaf area is a system variable (ID number -33).

AutoProgs=

Default auto program. The default value is "_". This does not specify a file, so the first file in "/user/Configs/AutoProgs" will be highlighted the first time the AutoProg menu is accessed (New Measurements mode, **f1** level 5).

AvgTime=

Running average time (seconds) for IRGA measurements. The running average time for all other channels is 1 second. The default IRGA signal average time is 4 seconds. This is for noise reduction.

BLCond=

Specify either a fixed value for the one-sided boundary layer conductance to use, or else the file name of the lookup table to use. See **Boundary Layer Variables** on page 14-17 for the equations. The default setting

```
BLCond= "/Sys/Lib/StdBLCTable"
```

is appropriate for broadleaves in the 2x3 chamber, and allows the effective boundary layer conductance to be computed automatically as a function of leaf area and fan speed. When using a 2x6 chamber (6400-07 or 6400-11), then use

```
BLCond= "/Sys/Lib/BLCTable_2x6"
```

If you are using conifers, then a fixed value is more appropriate:

```
BLCond= 5
```

BndBrdCorr=
Use band broadening corrections (page 14-23) for water vapor when computing CO_2 concentrations? A Yes/No toggle. Default is Yes, and there is probably no good reason why it should be changed.

CalCO2=
CO2 IRGA cal coefficients, for Equations (14-9) and (14-10) on page 14-6. These coefficients for a 5th order polynomial with no 0-order term are generated at the factory, and there is probably no good reason for you to change them. (If you are thinking about interchanging IRGA/sensor heads, don't.)

CalH2O=
H2O IRGA cal coefficients for Equations (14-5) and (14-6) on page 14-5. These are coefficients for a 3rd order polynomial with no 0-order term.

CalFlow=
Flow meter calibration multiplier for the linear flow meter, Equation (14-14) on page 14-8.

CalPress=
Pressure sensor calibration offset and multiplier for the linear pressure sensor, Equation (14-1) on page 14-4.

CalParGaAs=
GaAsP sensor calibration factor a_g. See **In-Chamber PAR** on page 14-8.

CalParOut=
External PAR calibration factor a_{qx}, in Equation (14-17) on page 14-9.

CalParLED=

LED source sensor calibration factor a_e in Equation (14-16) on page 14-8. Note that since version 3.0 of OPEN, a_e is specified in the calibration file using the CalParLED= command, but typically is specified in a user config file as part of the LightSource= command. Thus, there is no longer any reason for CalParLED= to appear in a user configuration file.

If for some reason, a configuration file contains a LightSource= command that specifies an LED source with one value of a_e, and also a CalParLED= that specifies another value of a_e, the one that takes precedent is whichever one is last in the file.

CalZero=

The two values which follow specify IRGA zero drift with temperature for CO_2 and H_2O respectively. The values correspond to S_c and S_w in Equations (14-11) and (14-7) starting on page 14-5.

ComputeList=

File containing user equations (Chapter 15). The user equations are not actually recompiled unless the file name or the file modification date has changed since power on. If no directory is specified, then "/User/Configs/Comps" is assumed.

CO2Mixer=

Is the CO2 mixer installed? This is also a Yes/No toggle, and should be set to reflect whether the mixer is installed or not; it has nothing to do with whether you are actually using the mixer.

Displays=

The display format file. If no directory is specified, then "/User/Configs/Displays" is assumed.

FanSlow=

Volts for slow fan speed. 0 is off, 5 is full speed. We recommend 4 for the slow value, so that's the default value.

FlrParams=

Specifies the initial fluorometer settings for fluorescence measurements, saturation flashes, and dark pulses. Groups of settings can be stored in and retrieved from the directory /User/Configs/FlrParams (see **Changing Control Parameters** on page 27-36).

LightSource=

Specifies the light source in use. When edited in the Config Editor, this entry is selected from a menu of possibilities (Figure 16-8 on page 16-8). This command requires multiple values. For non-LED sources:

LightSource= <name> f_a B

where f_a is an actinity correction factor (Equation (14-16) on page 14-8), and B is used to convert incident PAR to absorbed energy (Table 17-1 on page 17-4) for energy balance calculations. If the source is an LED source (a 6400-02 or -02B, or 6400-40 LCF), then a third parameter a_e is expected:

LightSource= "6400-02…" f_a B a_e

where a_e is the light source's calibration factor (Equation (14-16) on page 14-8). This quantity has historically been specified with the CalParLED= command, but since OPEN 3.0 and the introduction of the Light Source Control utility, it can also be included in the LightSource= command.

LogDelimiter=

Data field delimiter to use in log files. The editor for this item lists a menu, the contents of which are stored in the file "/Sys/Lib/Delims ".

LogFile=

Initial default log file name. Once the user opens a log file, his choice becomes the default for the next time a file is opened.

LogFormat=

The logList file, identifying the variables to be stored in a log file. If no directory is specified, "/User/Configs/LogFormat" is assumed.

LogOptions=

Specifies the Log Option settings (**Log Options** on page 9-9).

MatchType=

Specifies valve- or fan- based matching (**Matching Without the Match Valve** on page 16-34). If the text following *MatchType*= contains the letters FAN, then fan-based matching will be done. Otherwise, New measurement's match routine will use the match valve, which is the normal condition.

PATCH=

This is a catch-all tool. Whatever comes after the = is compiled (using LPL's COMPILE keyword) and run. The usual use for this is to set variables from

the configuration menu that don't have a normal interface. Sample uses of Patch= are shown in Figure 16-32.

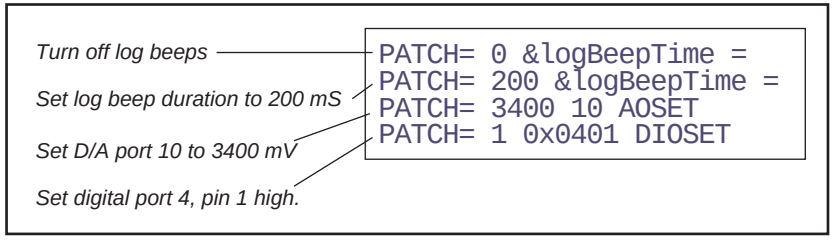


Figure 16-32. Some example uses of the Patch= command

PlotDefs=
Default plot axes definitions. If no directory is specified, /User/Configs/PlotDefs is assumed.

PromptList=
The prompt list file, which contains the list of system and user constants to be prompted for. See **Prompts and Remarks** on page 9-15.

Prompts=
Toggles between "off" and "ON LOG". Determines if the prompt list (if any) will be prompted for when data is manually logged.

SoilParams=
Specifies the soil respiration chamber settings file.

StableDefs=
The stability definition. This is what is set via **Define Stablty** (**f4** level 5) in New Measurements mode.

StripDefs=
The configurations for the 8 real time graphics screens in New Measurements mode.

StomRat=
Sets the stomatal ratio parameter K (Equation (14-29) on page 14-17). Note that this can be set in New Measurements mode as well (**f2** level 3).

UserChan=
The three values following the command are signal channel (20 through 23), ground channel (3 through 7), and resolution (0 = low, 1 = high). See **Analog Input Channels** on page 26-17.

//

This marks a comment. Any line starting with // is ignored. Note that the **Enable** and **Disable** function keys in the Config Editor comment and uncomment lines in this manner.

17

Using an Energy Balance

Computing what you can't measure

17

Using an Energy Balance

Sometimes it is not convenient or even possible to measure leaf temperature with thermocouple. LI-6400 chambers that do not measure leaf temperature directly are the 6400-05 Conifer Chamber, the 6400-07 Needle Chamber, and the 6400-08 Clear Bottomed 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, solar and thermal ($R = R_{solar} + R_{thermal}$). The solar part we will estimate by converting the incoming PAR reading in the chamber R_Q (µmol m^{-2} s^{-1}) to total visible and near IR energy using an empirical (measurable with a spectroradiometer) conversion factor k, and using the leaf absorptance α averaged over the spectrum of the light source.

$$R_{solar} = \alpha k R_Q \qquad (17\text{-}2)$$

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}:

$$L = 44100E \quad (17\text{-}4)$$

The sensible heat flux is a function of the leaf - air temperature difference, the specific heat capacity of the air c_P (J mol^{-1} K^{-1}), and the one-sided boundary layer conductance of the leaf g_b (mol m^{-2} s^{-1})

$$Q = 2c_p g_b (T_l - T_a) \quad (17\text{-}5)$$

The energy balance equation now becomes

$$\alpha k R_Q + 2\varepsilon\sigma(T_w + 273)^4 - 2\varepsilon\sigma(T_l + 273)^4 = 44100E + 2c_p g_b (T_l - T_a) \quad (17\text{-}6)$$

If we let $\Delta T \equiv T_l - T_a$, and note that for small ΔT

$$(T_l + 273)^4 \approx (T_a + 273)^4 + 4(T_a + 273)^3 \Delta T \quad (17\text{-}7)$$

then we can solve (17-6) for ΔT

$$\Delta T = \frac{\alpha k R_Q + 2\varepsilon\sigma\left[(T_w + 273)^4 - (T_a + 273)^4\right] - 44100E}{2c_p g_b + 8\varepsilon\sigma(T_a + 273)^3} \quad (17\text{-}8)$$

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).

Table 17-1. Radiation absorption coefficients for various light sources[a]

Light Source	k	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	0.16
6400-02B LED (Red + Blue)	0.19	0.84	0.15
Metal Halide	0.28	0.61	0.17

a.Measured with an LI-1800 spectroradiometer. Leaf spectral measurements were made on a green ash leaf using an 1800-12 Integrating Sphere.

.

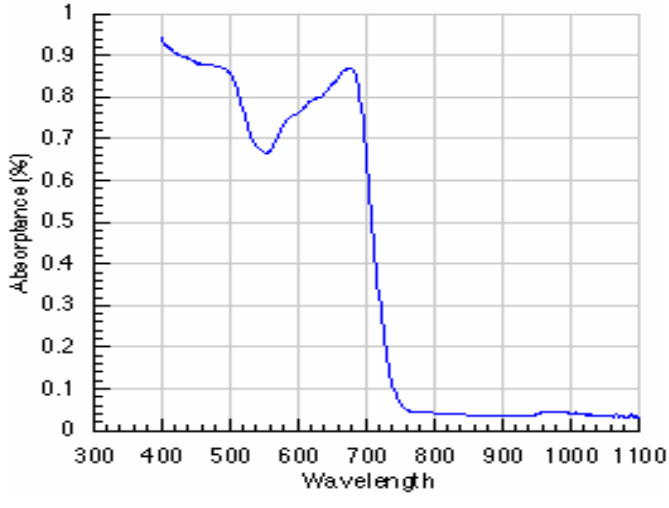


Figure 17-1. Leaf absortance spectrum used in the computations of Table 17-1.

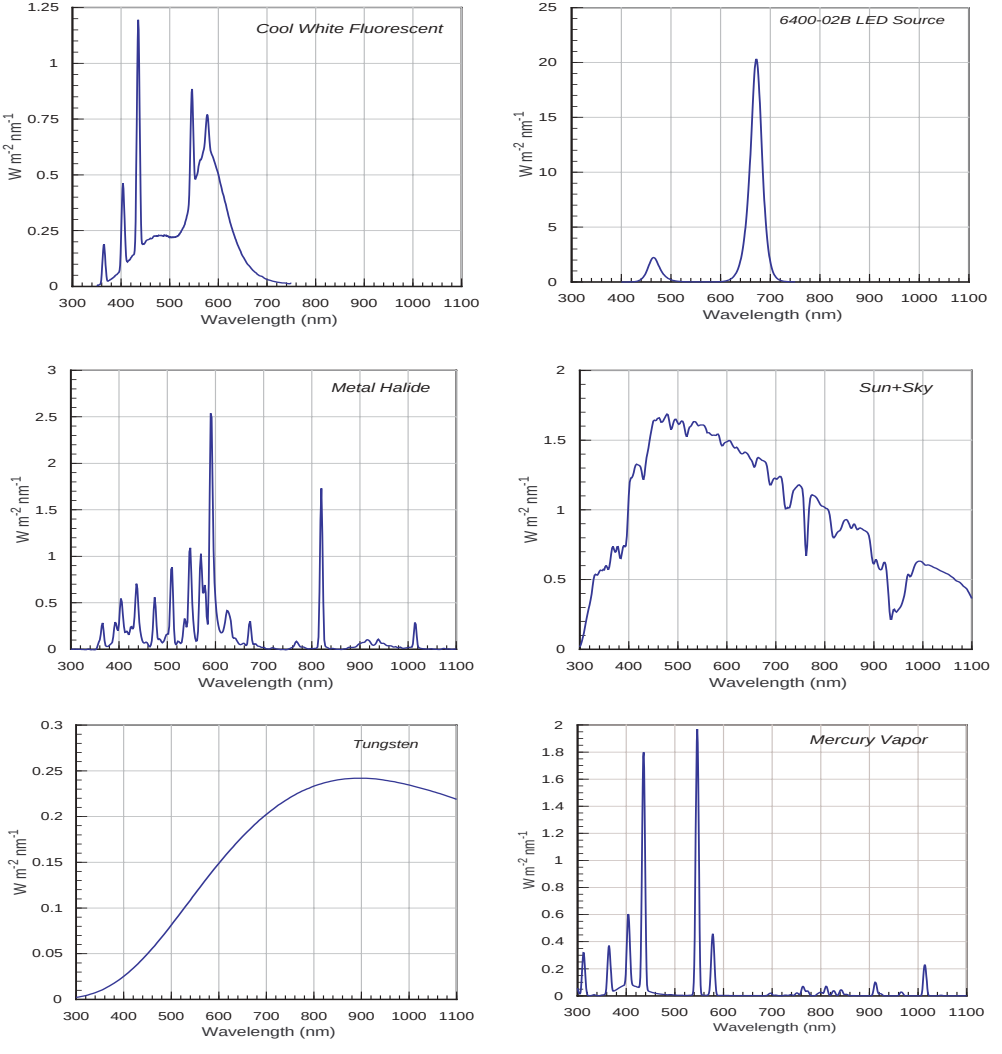


Figure 17-2. Spectra used to compute results in Table 17-1 on page 17-4. Vertical scales have units of W m^{-2} nm^{-1} in all cases.

Using Energy Balance in OPEN

How To Do It

Do the following to implement computing leaf temperature by energy balance.

1 **Use the Installation Menu**
It's in the Config Menu.

2 **Select the chamber to be used**
From the items in the Installation Menu, select the chamber you will be using.

3 **Select Energy Balance**
When asked if leaf temperature will be measured or computed, select the latter.

4 **Save the configuration, and change to it**
Name the configuration, save it, and use the reset menu to implement it.

5 **If there is a leaf temp thermocouple, use it for air temp**
If you are using a chamber that uses a leaf temperature thermocouple, pull it down so that it does not contact the leaf, but use it for getting air temperature close to the leaf. The 6400-05 Conifer Chamber, 6400-07 Needle Chamber, and 6400-08 Clear Bottomed chamber all have thermocouples expressly for measuring chamber air temperature.

The Details

Energy balance computations are implemented via the ComputeList (Chapter 15). There are three factory-provided compute lists that use energy balance. When you use the Installation Menu to build a configuration, it will use one of these compute lists. One of them, "Default using EB", is shown in Figure 17-3 on page 17-7.

```
------ Energy Balance Leaf Temp Versions of default compute list -----

##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)"

---- Use TCham_c for wall temp, and leaf thermocouple for air temp
##100F2 "EBdT" "Energy Bal. Tleaf-Tair (C)"
" EB_DeltaT( parIn_um * alphaK, tCham_c, tLeaf_C, condBL_one, #20) "

##101F2 "EBTlf" "Energy Bal. Tleaf (C)"
" tLeaf_c + #100 "

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20 * 1E3"

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, #101, condBL_mol, press_kPa, h2o_2_mm)"

##30 "Photo" "Photosynthesis (umol/m2/s)"
" -#10 * co2_diff_um - co2_2_um * #20"

##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / condBL_mol)"

##36 "Ci" "Intercellular CO2 (umol/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 "

##25 "VpdL" "Leaf VPD (es(Tleaf) - eair)"
" SatVap(#101) - eAir_2_kPa"

##27 "VpdA" "Air VPD (es(tair) - eair)"
" satVapTair_kPa - eAir_2_kPa"
```

Figure 17-3. Listing of "Default using EB". The energy balance leaf temperature is used in Cond and VpdL.

Each of these compute lists add two new user variables, *EBdT* and *EBTlf*. The first uses the LPL function *EB_DeltaT()* to compute the leaf-air temperature difference. This function implements Equation (17-8), takes five parameters, and returns the leaf-air temperature difference (Table 15-3 on page 15-16).

```
EB_DeltaT( parIn_um * alphaK, tCham_c, tLeaf_C,
condBL_one, #20)
```

Absorbed Energy

The first parameter is the energy absorbed by the leaf. The global variable *alphaK* is the product of the two multipliers (α and *k*) in (17-2). This product is given in Table 17-1 on page 17-4, and is set via the Lightsource= configuration command (page 16-39). The variable *parIn_um* is the current reading of the in-chamber light sensor.

If the chamber has a clear bottom, then the incident PAR would be larger than measured by the uplooking chamber light sensor. For this reason, in the ComputeList file "Clear Bottom EB", the call to *EB_DeltaT* looks a bit different:

```
EB_DeltaT( parIn_um * alphaK * 1.2,tCham_c,tLeaf_C,condBL_one,#20)
```

The *parIn_um* value is increased 20% to account for the clear bottom.

The third variation on the theme is provided by "Conifer Chamber EB", which has no in-chamber PAR sensor, so the external one must be used. Accordingly, the call to *EB_DeltaT* looks like this:

```
EB_DeltaT( parOut_um * alphaK, tCham_c, tLeaf_C, condBL_one, #20)
```

Wall Temperature

The second parameter needed for the *EB_DeltaT* function is wall temperature. The LI-6400 doesn't have such a sensor per se, but the air temperature thermistor 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.

Air Temperature

The third parameter is the air temperature next to the leaf. The LI-6400 has sensors for this only in the 6400-05, -07, and -08 chambers, so in the rest of the chambers we use the leaf thermocouple to do this job (variable *tLeaf_C*), 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.

Boundary Layer Conductance

The fourth parameter is the one-sided boundary layer conductance, which is available as global variable *condBL_one*.

Transpiration Rate

The final parameter is transpiration rate, which is user variable number 20.

If these assumptions are inadequate for your application, then you should modify the ComputeList accordingly (and rename it to keep it separate).

Recomputation Considerations

The recomputation option available in OPEN's Utility Menu allows you to change any of the following: user equations, output formats, absorbed radiation coefficients, area, boundary layer conductance, and stomatal ratio. Note, however, that if you use the leaf temperature thermocouple to measure air temperature (as we recommend), then you can't go back and recompute *without* using the energy balance, because you will not have a measured leaf temperature.

Each of OPEN's log files contains header information telling what user equations, light source, output format, etc. were used for that file. Thus, you can easily tell after the fact what mode you were in when looking at old data.

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 (17-6). 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 the filter paper is very wet, and allows us to write

$$E = 2g_b\left(\frac{e(T_l) - e_s}{P}\right) \qquad (17\text{-}9)$$

where the function *e(T)* computes saturation vapor pressure (Equation (14-22) on page 14-10), T_l is the filer paper temperature, and e_s is the vapor pressure surrounding the paper (Equation (14-19) on page 14-10). Solving for one-sided boundary layer conductance g_b,

$$g_b = \frac{EP}{2(e(T_l) - e_s)} \qquad (17\text{-}10)$$

and substituting that expression, along with $R_Q = 0$ into (17-6) leaves us with

$$0 + 2\varepsilon\sigma(T_w + 273)^4 - 2\varepsilon\sigma(T_l + 273)^4 = \qquad (17\text{-}11)$$
$$44100E + 2c_p\left(\frac{EP}{2(e(T_l) - e_s)}\right)(T_l - T_a)$$

Solving (17-10) for E yields

$$E = \frac{2\varepsilon\sigma((T_w + 273)^4 - (T_l + 273)^4)}{44100 + \frac{c_p P(T_l - T_a)}{e(T_l) - e_s}} \qquad (17\text{-}12)$$

Everything in Equation (17-12) is known except leaf (paper) temperature T_l. If T_l can be found by iteration, then boundary layer conductance can be determined from (17-10). An LPL program is provided with OPEN that does implements this solution, and it is described in **"ENERGYBAL"** on page 21-14. Its primary use is for doing boundary layer conductance determinations.

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.

Part V

Maintenance & Troubleshooting

Calibration Issues

18

How much is good data worth?

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 18-26).
PARi (no LED)	GaAsP photodiode		Measure adjustment factor for light sources not already in the list. (See **Generating a Calibration Correction** on page 18-30).
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 18-21).

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 14-5), and *g(x)* for CO_2 (Equations (14-9) and (14-10) on page 14-6). 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. These measurements are made at a series of temperatures (typically 15, 30, and 45C), 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 entered in the instrument as the values following the *CalCO2=* and *CalH2O=* configurations commands (see Figure 16-13 on page 16-14). The calibration also provides data on the IRGAs' drift with temperature, and this information is provided via the *CalZero=* calibration command.

User Calibration (Zero and Span)

User calibration actions consist of checking and/or adjusting the zero and span. There are two parts: "zero", which is checking the reading with dry, CO_2-free air in the cell, and "span", which is checking the reading with a known concentration in the cell. A zero adjustment shifts the entire response curve a fixed amount. For example, if an IRGA reads 2 µmol mol^{-1} with CO_2-free air in the cell, and you do a zero adjustment to make it read 0, then you have changed the response by -2 µmol mol^{-1} at *every* concentration. By contrast, a span adjustment affects the sensitivity of the response. For example, suppose an IRGA reads 990 µmol mol^{-1} when there is 1000 µmol mol^{-1} in the cell. If you adjust the span to correct this, you will have increased the response by 10 µmol mol^{-1} at 1000, but at 500 µmol mol^{-1}, the response will be increased by only 5 µmol mol^{-1}, and at 2000 µmol mol^{-1}, it will be about 20. (The numbers are not quite right, since the span adjustment is linear with IRGA signal, not concentration.) At 0 concentration (or rather, at zero voltage), a span adjustment has no effect at all.

How Often?

When OPEN first runs, it reads the IRGA and flow meter zero information (and IRGA span information) from a file[1], and applies it. This information gets into these files by your storing it (as described in **View, Store Zeros & Spans** on page 18-19). In theory, you shouldn't have to re-zero every time you turn the instrument on, especially if the temperature is essentially unchanged from the last time. However, if nothing else, zeroing and spanning serves as a diagnostic on part of the system.

Your own experience should be your guide, but our experience is this:

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.

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. Just leave it alone.

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 18-7).

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. "/dev/parm1", if you are curious.

1 Select "IRGA Zero" in the calibration menu

You be greeted with a couple of prompt screens, followed by the main screen (Figure 18-1):

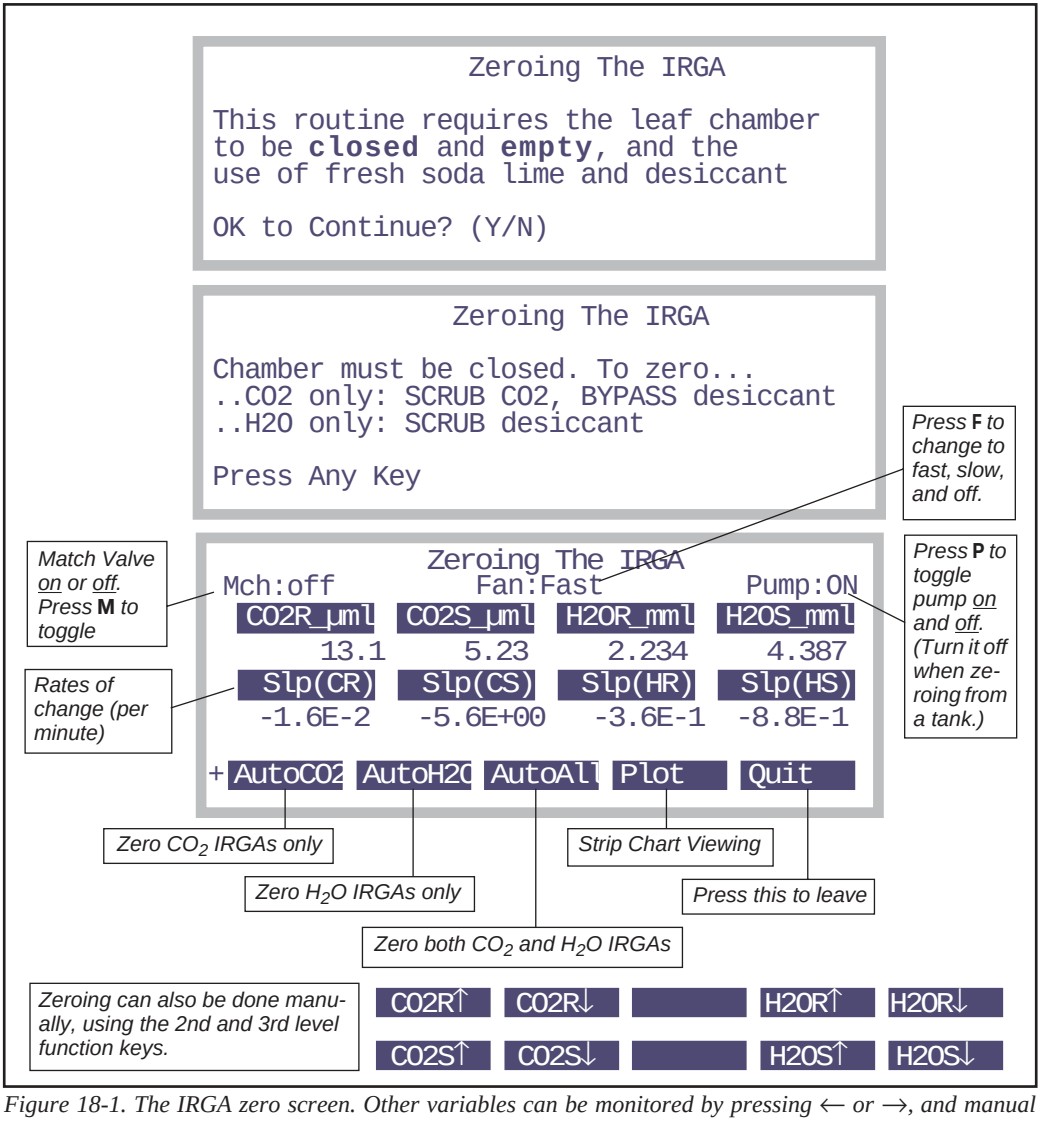


Figure 18-1. The IRGA zero screen. Other variables can be monitored by pressing ← or →, and manual zeroing of individual IRGAs is available by pressing **labels**.

2 **Soda Lime: full scrub Desiccant: full bypass**

Figure 18-1 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 18-2). To return to the normal text display, press **escape** or **QUIT**). It should take less than 2 minutes to get reasonably stable readings.

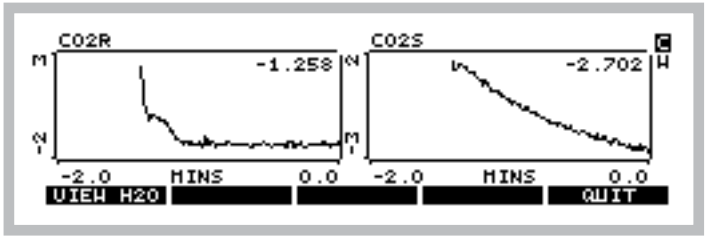


Figure 18-2. Pressing **plot** *(***f4***) brings up strip charts charts for H_2O and CO_2. Switch between the two by pressing* **C** *and* **H**, *or* **f1**.

4 **If re-zeroing is necessary, press AutoCo2 (f1).**

If both CO_2 values are within 5 µmol mol^{-1} of 0, you can skip this if you wish[2]. If you do the press **AutoCo2**, 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

[2] 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[3] or so, and…

6 **Press AutoH2O (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**

8 **Save it**
Select "View, Store Zeros & Spans" now, if you are done calibrating.

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 0.5 l 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

[3] There's a way to do this faster. See **Manually Zeroing the IRGAs** on page 18-10.

reference IRGAs simultaneously. Both methods are illustrated in Figure 18-5 on page 18-13. The preferred method is to use a “Y” connector.

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 ex-

plained under step 5 on page 18-6.

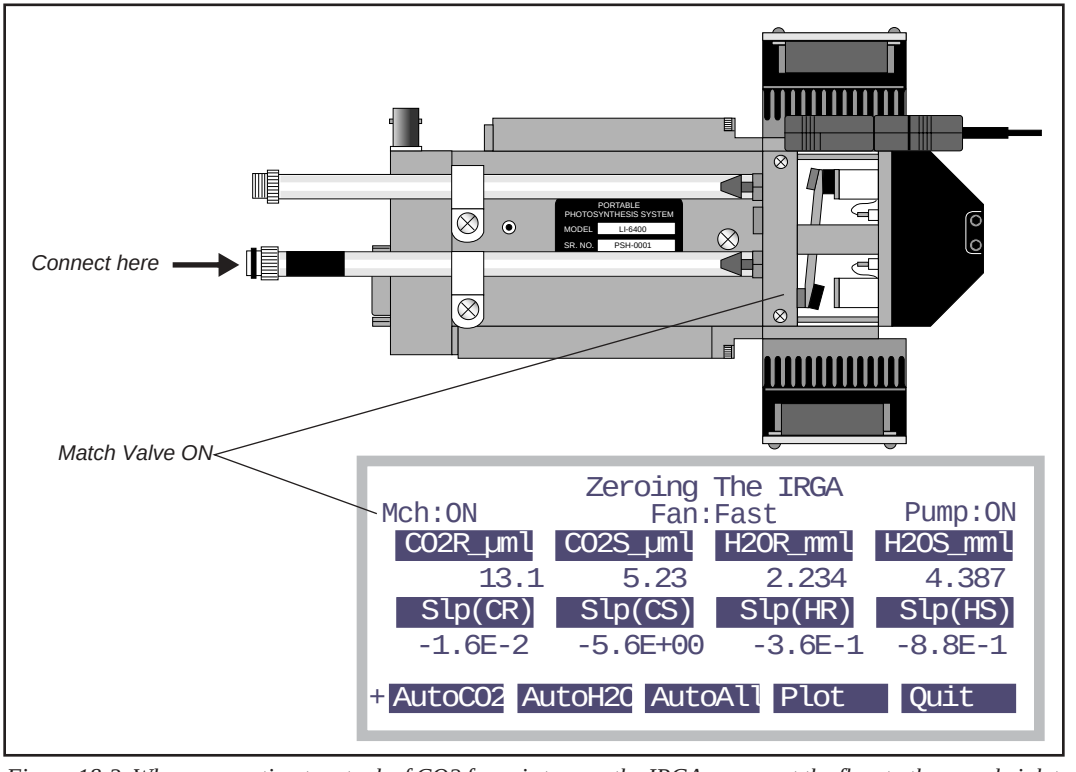


Figure 18-3. When connecting to a tank of CO2 free air to zero the IRGAs, connect the flow to the sample inlet, and have the match valve turned on. The chamber needs to be closed, as well.

6 **When stable, press AutoAll**
Since the air source is dry and CO_2 free, you can zero both IRGAs simultaneously.

7 **Save it**
Select "View, Store Zeros & Spans" now, if you are done calibrating.

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 (Figure 18-8 on page 18-19).

- **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 using `"View,Store Zero & Span"` (page 18-19). It may be that you had very non-zero air when you zeroed, or else the IRGA has a problem.

Manually Zeroing the IRGAs

Normally, when the IRGA zeroing program is used, the IRGAs are automatically zeroed by pressing **AutoCO2** or **AutoH2O**. It is possible to zero any one of the four IRGAs separately, and manually (Figure 18-4).

One occasion when this can be a useful tool is in zeroing the water vapor sample cell. If both cells are connected with a "Y" (as in Figure 18-5 B), have the match valve off, and zero as described above when the *reference* cells are stable. Then turn the match valve on, and adjust the H_2O sample cell to match

the H2O reference value.

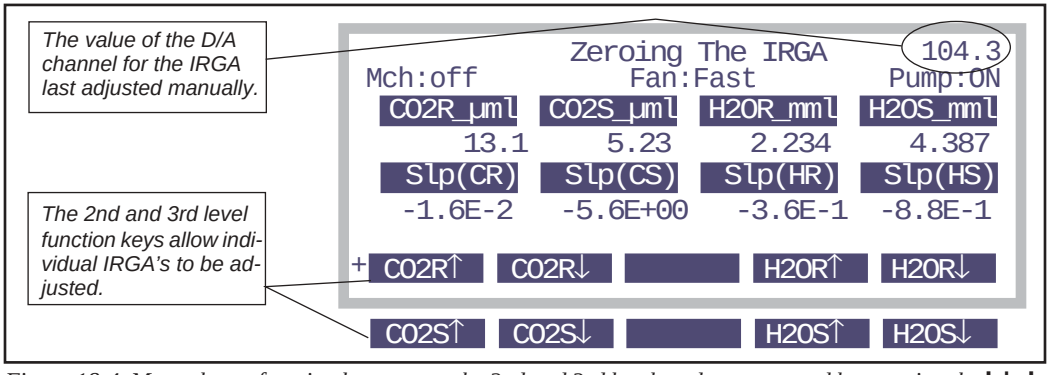


Figure 18-4. Manual zero function keys are on the 2nd and 3rd level, and are accessed by pressing the **labels** *key. When one of the manual zeroing function keys is pressed, the value of the D/A channel that controls that IRGA's zero is displayed to the right of the Match Status. This is also the value displayed in the "View,Store Zeros&Spans" program in the Zero column (Figure 18-8 on page 18-19).*

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 correctly calibrated gas analyzer. The concentration should be at or above where you work most of the time, so 400 or 500 µmol mol^{-1} is fine[4].

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 (14-9) and (14-10) on page 14-6).

Checking/Setting the CO_2 Span

Look at Figure 18-5 on page 18-13.

1 Select "IRGA span" from the Calib Menu

2 Set the flow from the tank.
See the comments under step 1 on page 18-7.

[4] At the factory, we use our 1500 µmol mol^{-1} tank for setting spans.

3 Attach to the IRGA.
You have a choice here, as shown by Figure 18-5: 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 *CO2Rµml* is highlighted, you are adjusting the CO_2 reference IRGA's span factor. The sample IRGA is *CO2Sµml*.

The span values should be close to 1.0, as described in **What Can Go Wrong Setting the Span** on page 18-16).

5 Quit and Save, if done
Press **escape**, and select "View, Store Zeros & Spans" now, if you are done calibrating.

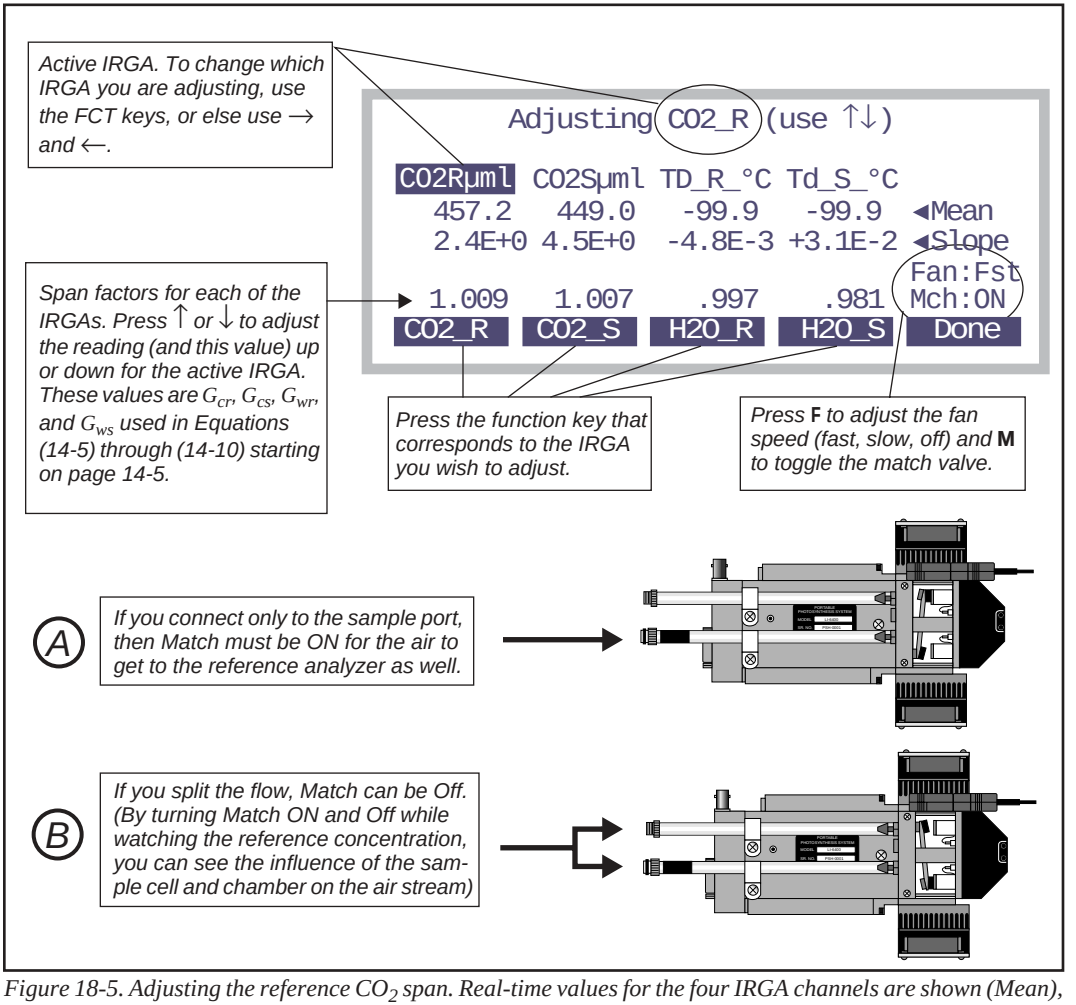


Figure 18-5. Adjusting the reference CO_2 span. Real-time values for the four IRGA channels are shown (Mean), as well as their rates of change (Slope). The span factors (should always be near 1.0) are used as multipliers of the IRGA voltages. Pressing ↑↓ increases or decreases the span factor for the 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 14-5).

To set the H_2O Span

1 Setup the LI-610 for an appropriate dew point

Subtract about 5C 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 5C "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 a 0.5 l min^{-1} flow from the LI-610.

3 Attach to the IRGA

You have the same choice, as shown in Figure 18-5 on page 18-13. 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.

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 18-6).

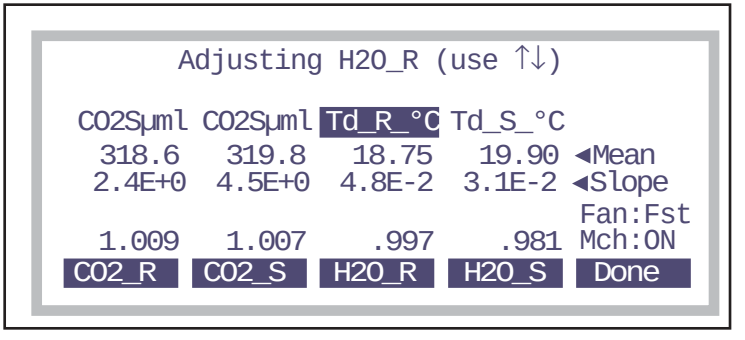


Figure 18-6. Adjusting the H_2O span. Adjust until the displayed dewpoint value matches what the LI-610 is set to.

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 left hand (reference) *Td_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 to read that value.

(Do you see the trick here? We've just 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.)

10 **Quit and Save, if done**
Press **escape**, and select "View, Store Zeros & Spans" now, if you are done calibrating.

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)[5]. The farther out of this range you go in attempting to make the analyzer read the correct concentration, the more likely it is that something is wrong:

- **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 sent to you.

  (H_2O) Is there water in the condensor? Are you asking for a target that is at or above the room temperature? (You won't get that dew point, of course, 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 correctly?**
  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 doesn't permit listing all the reasons why that is a bad idea.)

---

[5] 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.

- **"IRGAs Not Ready"?**
  If this message is flashing on your display, then there are more pressing problems to be addressed (see **"IRGAs Not Ready" Message** on page 20-17), and you certainly shouldn't be setting the span.

- **IRGAs not responding**
  See **IRGA(s) Unresponsive** on page 20-18.

Storing the Zeros and Spans

If the zero and/or span adjustments needed to be made, and you are happy with the results, then store them by

1 **Select** `"View, Store Zeros & Spans"` **in the Calib menu.**
It takes a few seconds to load. (This program is described on page 18-19)

2 **Press Store (F1)**
This will check the flow meter zero, the IRGA zeros, and the IRGA spans to see if they are different that what is stored. You will be prompted

```
"Store (Y/N)"
```

for each item if there is a difference.

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 series of mass flow controllers. (The flow controllers are, 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 (14-14) on page 14-8. This value is shown on the LI-6400 calibration sheet, and is stored in the instrument, as the object of the CalFlow= configuration command.

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 will be shut off, and the system will begin a 10 second count down (Figure 18-7). After ten seconds, the flow meter signal should read within 1 mV of zero. Press **OK**.

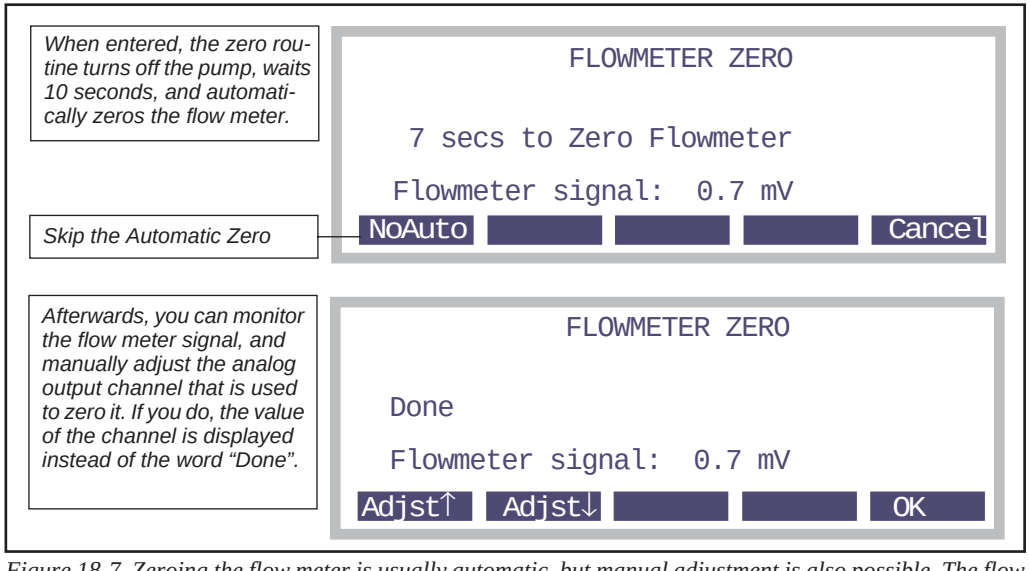


Figure 18-7. Zeroing the flow meter is usually automatic, but manual adjustment is also possible. The flow meter zero is controlled electronically by an analog output channel, whose value is displayed in the Zero column of the View, Store Zero & Span program (page 18-19).

The flow meter's zero has a slight temperature sensitivity, but the drift is typically less than 1 µmol s^{-1} per 10 degrees C. The flow meter zero is controlled by an analog output channel, and the value is displayed in the "View, Store Zeros & Spans" program (page 18-19) under the Zeros column. The value shouldn't change much from day to day, but checking it and setting it is so easy, it's worth the effort to make sure.

View, Store Zeros & Spans

This entry in the Calib Menu launches a program that allows you to view the current zero and span settings for the IRGAs, and the current flow meter zero. You can also revert these values back to the power on or factory settings.

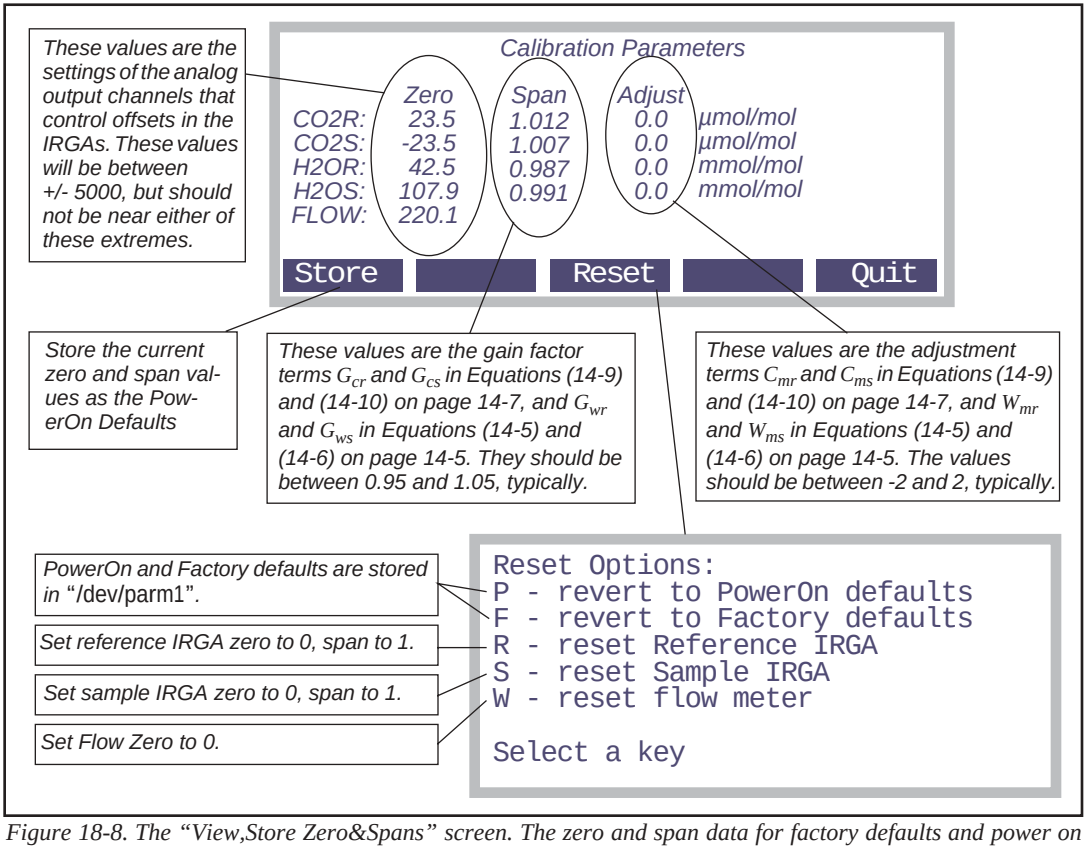


Figure 18-8. The "View,Store Zero&Spans" screen. The zero and span data for factory defaults and power on defaults is stored in the file "/dev/parm1", along with a number of other settings.

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, for example, 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 18-9). 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.

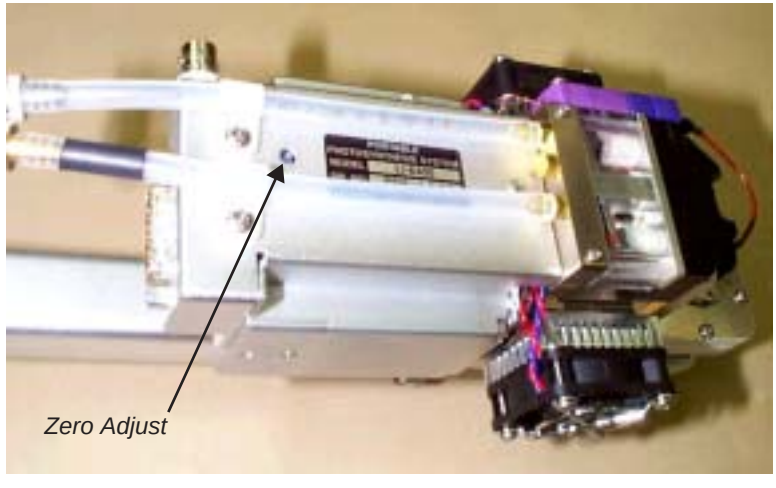


Figure 18-9. Location of the leaf temperature zero adjustment screw.

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. There is, however, a calibration that can be done to relate command voltage to resulting CO2 concentration, and that 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 `"_CO2 Mixer - Calibrate"`. When prompted

```
OK to continue? (Y/N)
```

press **Y**.

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 18-10). "Stable" means 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.

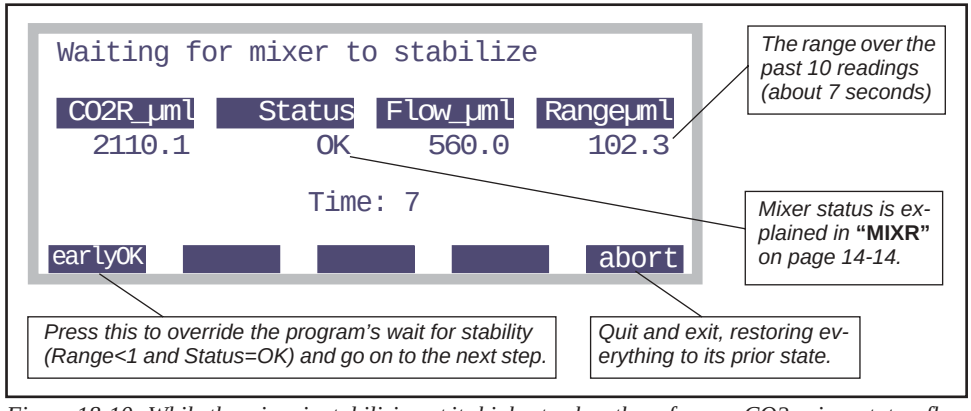


Figure 18-10. While the mixer is stabilizing at its highest value, the reference CO2, mixer status, flow rate, and range of reference CO2 values over the past 10 readings are displayed.

2 OK the upper limit

Once the mixer is stable at its maximum value, this value is presented with an option to change it (by adjusting pump speed):

```
Max value is about 2200 ppm.
Is this OK (Press N to adjust) (Y/N)
```

(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**.

If you have OPEN 3.2 or above, go to Step 3.

If you have OPEN 3.01 or below, you will be prompted to enter the desired upper limit.

```
Enter desired max CO2 (ppm) _
```

The pump speed is adjusted based on the ratio of what the upper limit of CO_2 was measured to be, and what you want it to be. Type in some value, such as 2500, and press **enter**.

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.

3 Adjust the pump speed manually
A routine is entered (Figure 18-11) that lets you adjust the pump speed with the function keys to achieve the desired upper limit.

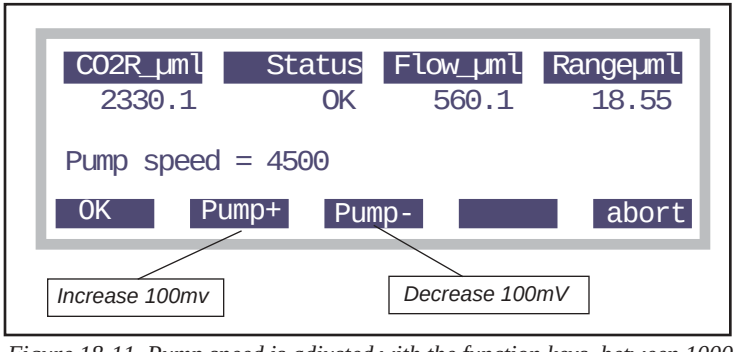


Figure 18-11. Pump speed is adjusted with the function keys, between 1000 mV and 5000 mV. Default pump speed is 4500 mV.

The pump's speed control is limited to between 1000 and 5000 mV. The strategy is to adjust the pump speed, then wait for the CO_2 to stabilize, then readjust the pump speed, etc. Allow several seconds for the CO_2 concentration to equilibrate after a pump speed change. Pressing **OK** takes you back to step 2.

4 Automatic mixer calibration

The program loops through 8 set points, from 5000 mV down to 0 mV, and records the CO_2 concentration for each (Figure 18-12).

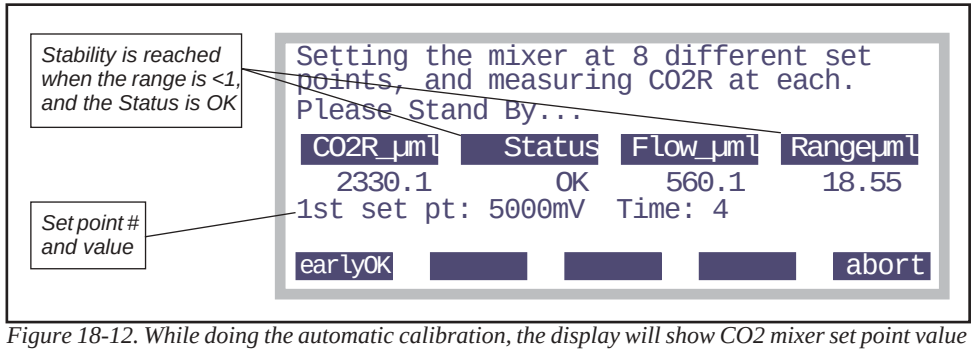


Figure 18-12. While doing the automatic calibration, the display will show CO2 mixer set point value and resulting CO2 concentration.

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. (The last set point, 0 mV, ignores the Status). Pressing **f1** (**earlyOK**) causes the point to be accepted, and it moves to the next set point.

5 View the calibration curve

Once the calibration is done, the data is printed and

```
Plot this (Y/N)
```

is asked. Press **Y**. You should see something like Figure 18-13. (You can also see this plot at any later time, by selecting "_CO2 Mixer - Plot Curve" from the Calib Menu.)

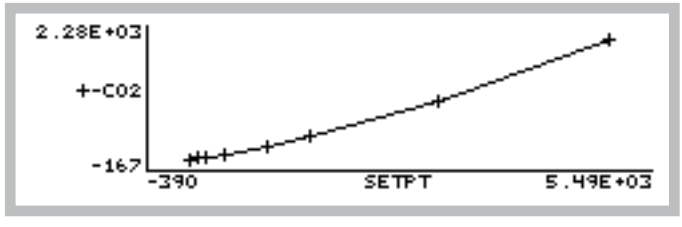


Figure 18-13. Typical plot of a CO2 mixers calibration curve. The range of CO_2 concentrations is typically 40 to 2200 µmol mol^{-1}.

If the user accepts the calibration, it is stored in /User/Configs/Mixer, along with

the pump voltage. A typical mixer file is shown in Figure 18-14

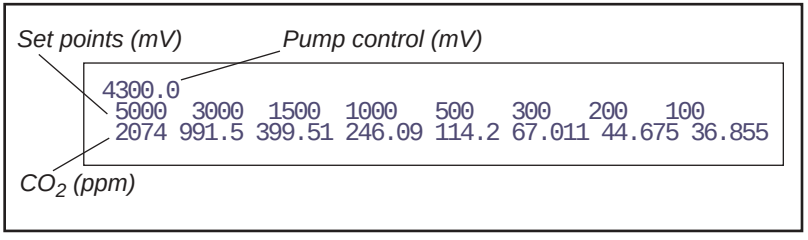


Figure 18-14. The file "/User/Configs/Mixer" contains the pump voltage to be used while the mixer is on, and the calibration data, relating command voltage to resulting CO_2 concentration.

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 program to do this is located at the bottom of the Calib Menu, and is named `"_Zero ParIn Signal"`. When run, it looks like Figure 18-15.

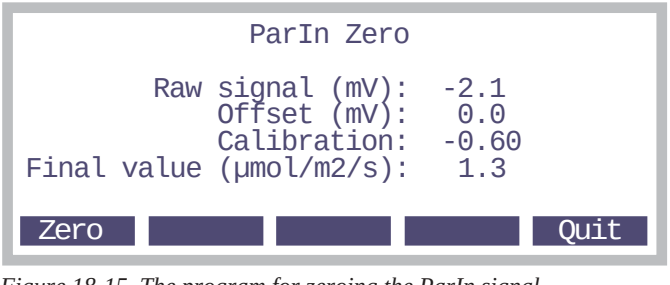


Figure 18-15. The program for zeroing the ParIn signal.

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 (14-15) on page 14-8.)

The offset value is retained in the file "/User/Configs/ParInZero". The offset value can be viewed in New Measurements mode in Diganostic Screen E (page 6-23).

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 LEDs 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 LED's 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.

- **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 light source calibration program. Select the "_Light Source Calibration" entry in the Calib Menu, followed by "Calibrate LED Source" This program is fairly automatic, once you get past the opening screen (Figure 18-16).

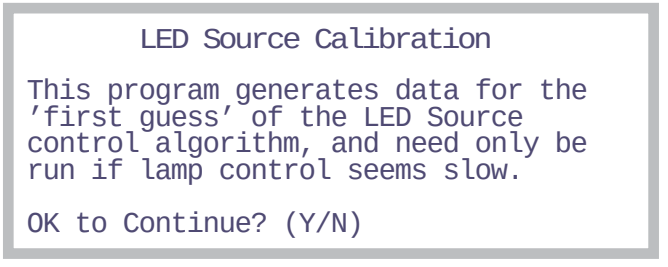


Figure 18-16. The LED Source Calibration's opening screen provides a fail-safe in case you blunder into it while doing some other experiment that you don't want interrupted.

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 18-17).

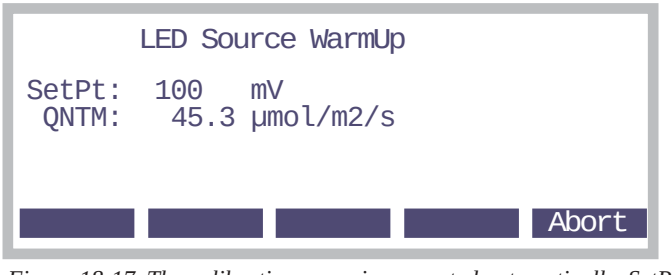


Figure 18-17. The calibration curve is generated automatically. SetPt is the command signal, and QNTM is the resulting PAR (µmol m^{-2} s^{-1}).

When done, the program will show the data and provide you with an opportunity to plot it (Figure 18-18). You can also view the current LED source calibration later by selecting "_LED Source - Plot Curve" (OPEN 3.4 and below) or "_Light Source Calibration Menu" then "Plot curve" (OPEN

4.0 and above).

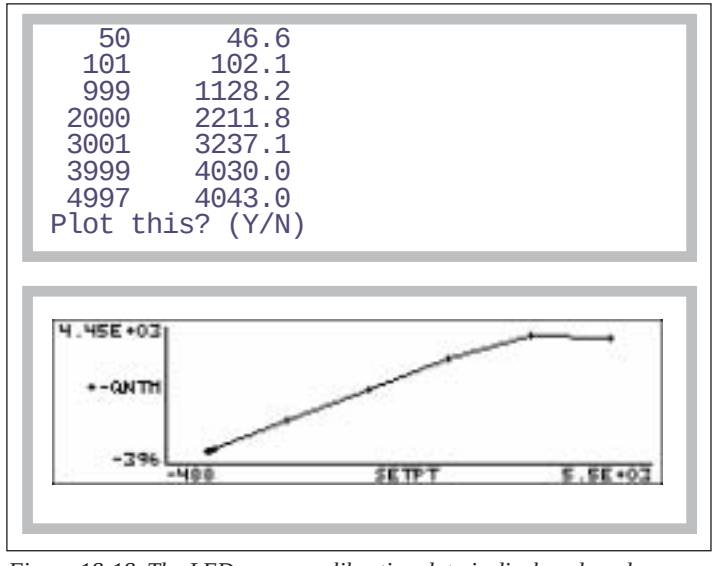


Figure 18-18. The LED source calibration data is displayed, and you are given a chance to plot the graph.

To actually implement this calibration data, you must respond by pressing **Y** when asked (Figure 18-19).

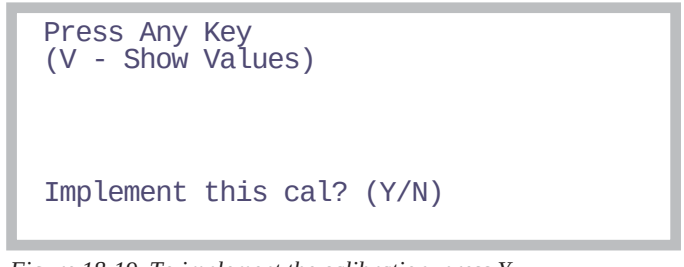


Figure 18-19. To implement the calibration, press Y.

The calibration data for the LED source is always stored in file "/User/Con-

figs/Lamp". The file format is illustrated in Figure 18-20.

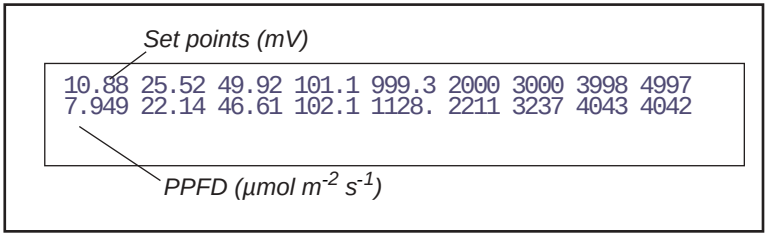


Figure 18-20. Typical "/User/Configs/Lamp" file. The lowest set point is 10 mV in OPEN 3.01 and below. With OPEN 3.2, it changed to 50 mV.

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, however, is that the red and blue LEDs are independently controlled, and thus each has its own calibration. See **"Actinic Control - Calibrate"** on page 27-61 for details.

Fluorescence Calibration

The fluorometer part of the LCF only requires zeroing. See **"Zero Fluorescence Signal"** on page 27-66.

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 8-5 on page 8-10). 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 (14-16) on page 14-8.

Generating a Calibration Correction

A calibration correction factor (f_a in Equation (14-16) on page 14-8) 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, however, 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

$$f_a = \frac{Q_x}{Q_c} \qquad (18\text{-}1)$$

7 **Enter the new value**
Select `"Light Source Control"` in the Config Menu, press **labels**, then press **Edit StdFile** (**F2**). Add your new correction factor to the list, as shown in Figure 18-21. You'll need a name in double quotes, followed by the f_a value, followed by a guess at the absorbed energy conversion factor (use 0.2 unless you know better).

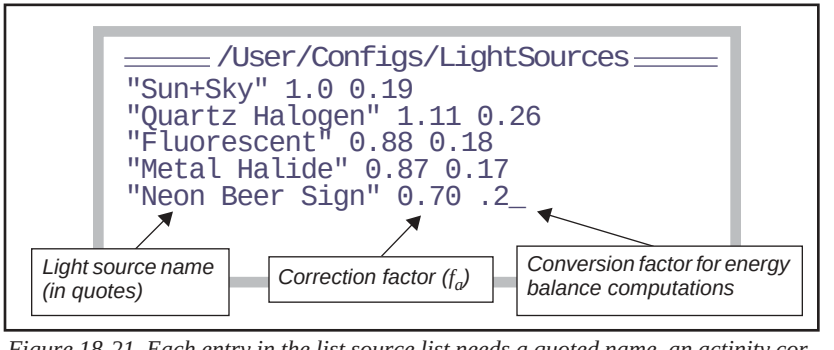


Figure 18-21. Each entry in the list source list needs a quoted name, an actinity correction factor, and a conversion factor (used only for energy balance computations).

Note: We do not recommend using an LI-190 quantum sensor to calibrate, or even accurately measure, the LED light source output because the drop off in quantum sensor sensitivity near 700nm (Figure 8-5 on page 8-10) is similar

to the long wave edge of the LED emission spectrum (Figure 8-3 on page 8-7 or Figure 17-2 on page 17-5). Small changes in these sharp slopes due to temperature, unit to unit variation, or simply over time can lead to large measurement uncertainties.

Calibration and Configuration File Summary

/dev Files

Calibration information resides in a directory named /dev. Two of the files there are named parm0 and parm1.

"/dev/parm0"

This file contains all factory calibration information for the IRGAs and light sensors. Typically, information is added or removed from this file by using the Installation Menu, in the Config Menu. It can, however, be edited directly via the Filer (**Standard Edit** on page 5-13). A typical "/dev/parm0" is shown in Figure 18-22.

```
// this is a remark
CO2Mixer= YES
CalCO2= 0.21732 2.1807E-05 1.9303E-08 -3.2863E-12
3.272E-16
CalH2O= 0.0063802 1.9083E-06 -2.4303E-11
CalZero= -4.0 -2.3
CalFlow= 0.35307
CalPress= 88.522 0.005458
CalParGaAs= 0.77 // GA-103 20 Jan 1995
CalParOut= 5.78 // Q13436 27 Aug 1990
ThisUnit= "PSC-0103"
Serviced= "9 Jan 1995"
```

Figure 18-22. A typical "dev/parm0" file.

"/dev/parm1"

This file contains user and factory zero and span settings, as well as some factory-determined calibration values for the instrument's A/D and D/A hardware. Zero and span settings are written to this file as described in **Storing the Zeros and Spans** on page 18-17. It can be directly edited by using the LPL editor (**Standard Edit** on page 5-13). A typical "/dev/parm1" is shown in Figure 18-23.

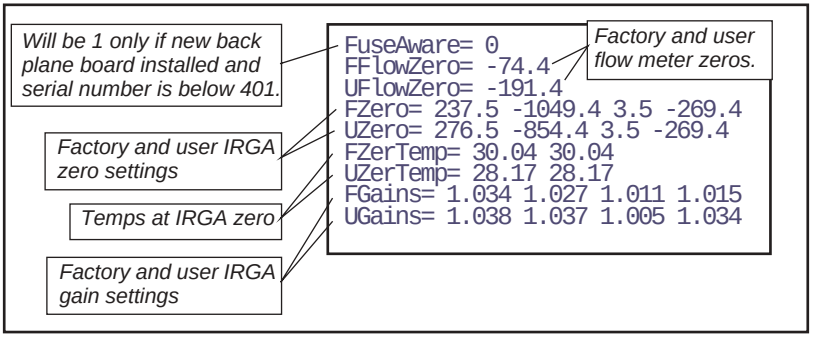


Figure 18-23. A typical example of "/dev/parm1".

The Configs directory

In the /User/Configs directory there are several files that contain calibration or configuration related information. Typical contents of these files can be seen by looking at the relevent sections of Figure 21-5 on page 21-11.

"/User/Configs/history"

This file contains a history of changes made to "/dev/parm1". A typical history file is shown in Figure 18-24. Each time calibration changes are saved (**View, Store Zeros & Spans** on page 18-19), the changes made are added to the history file, along with the time and date.

```
Thr Jun  6 2002 10:25:17
UFlowZero=507.8

Thr Jun  6 2002 10:25:18
UZero=195.3 0.0 0.0 0.0
UZerTemp=20.00 20.00

Fri Jun  7 2002 09:03:13
UZero=195.3 -39.1 39.1 -39.1
UZerTemp=20.00 20.00

Fri Jun  7 2002 09:03:14U
Gains=0.972 0.982 0.966 0.969

Fri Jun  7 2002 09:38:45
UFlowZero=468.8

Mon Jun 10 2002 13:51:53
UZero=195.3 0.0 39.1 0.0
UZerTemp=20.00 20.00

Mon Jun 10 2002 13:51:53
UGains=0.971 0.983 0.964 0.959
```

Figure 18-24. A typical history file.

"/User/Configs/lamp"

This file contains the latest LED source calibration, and is generated by Calib Menu, then "_Light Source Calibration Menu", then "Calibrate LED Source". A typical file is shown in Figure 18-25 . A typical plot is illustrated in Figure 18-20 on page 18-29.

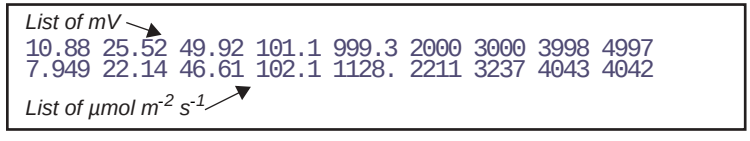


Figure 18-25. A typical lamp file

"/User/Configs/mixer"

This file contains the latest CO_2 mixer calibration, generated by executing "_CO2 Mixer - Calibrate" in the Calib Menu. A typical file is shown in Figure 18-26, and a typical plot is illustrated in Figure 18-14 on page 18-25.

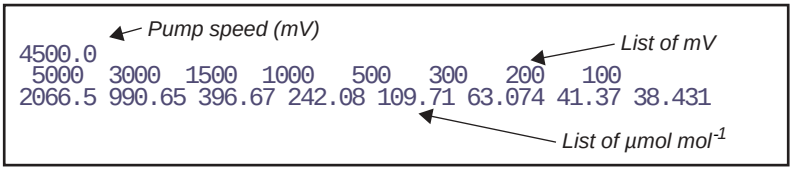


Figure 18-26. Typical mixer calibration file.

"/User/Configs/fluor"

This file contains the latest fluorometer actinic calibration, and is generated by Calib Menu, then "_Light Source Calibration Menu", then "Actinic control - calibrate".

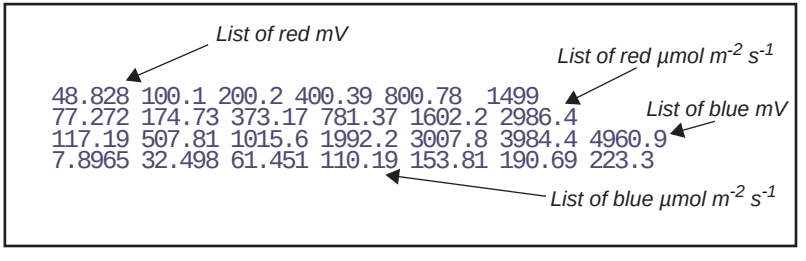


Figure 18-27. Typical LCF actinic calibration file.

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.

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 (as you view the figure) as you attempt to loosen the screw.

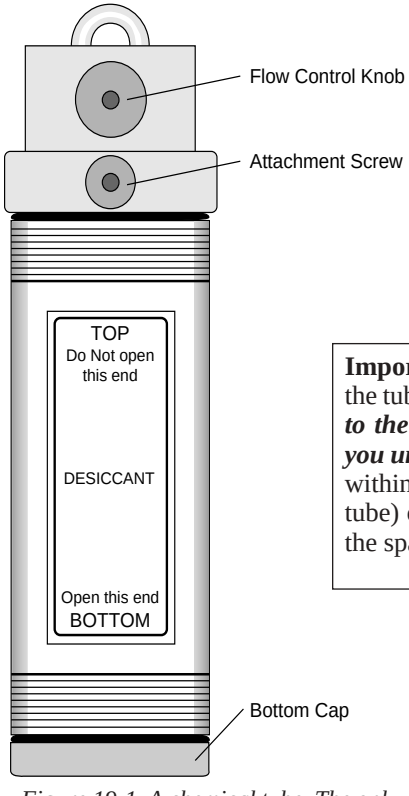


Figure 19-1. A chemical tube. The only difference between the soda lime tube and the desiccant tube is the label.

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.

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.

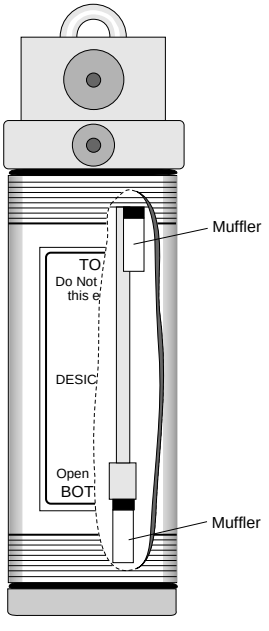


Figure 19-2. Air mufflers in soda lime and desiccant tubes.

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 20-14.)

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 it or 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.

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 of 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 over heated 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 just slightly snug, 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.

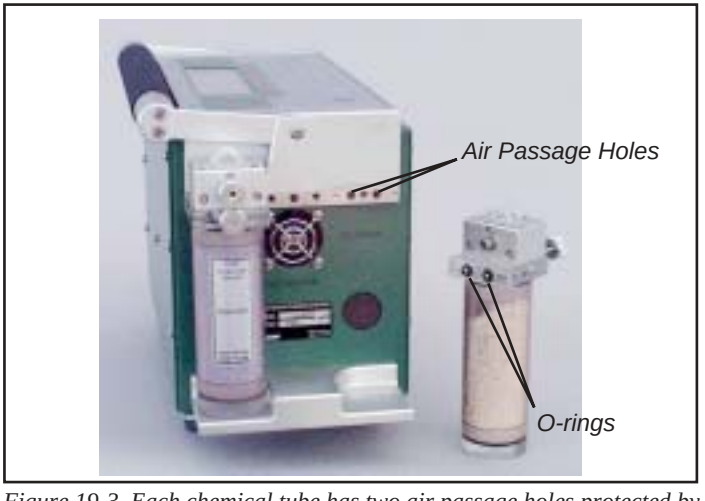


Figure 19-3. Each chemical tube has two air passage holes protected by O-rings.

Chemical Tube Flow Control

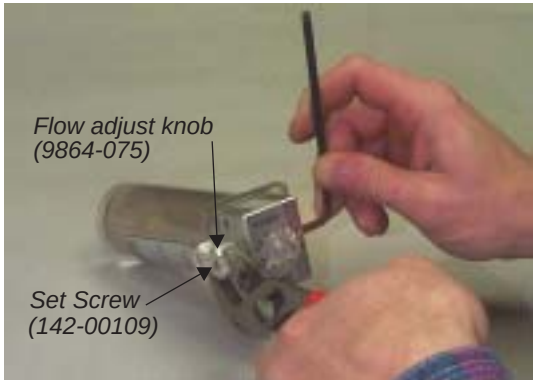
The usual reason to take apart the flow control assemblies on the soda lime or desiccant tube is that one of the small flow tubes becomes pinched, or some debris clogs up the tubing, or gets in the small air passage ways.

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

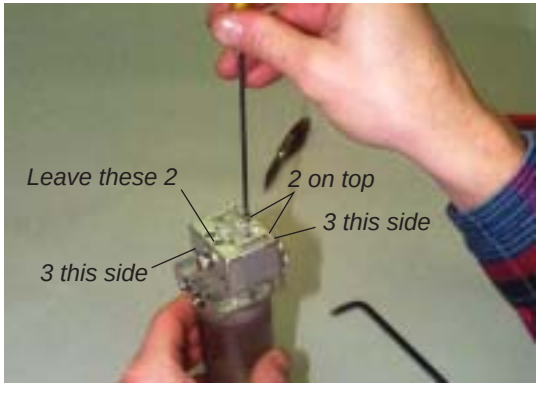


Use a 5/32 hex key to hold the flow adjust bolt, and loosen the flow adjust knob with channel lock pliers. (Turn the nut toward SCRUB to loosen.) Remove the nut and white washer that is beneath it.

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.)

Figure 19-4. Removing the flow adjust knob.

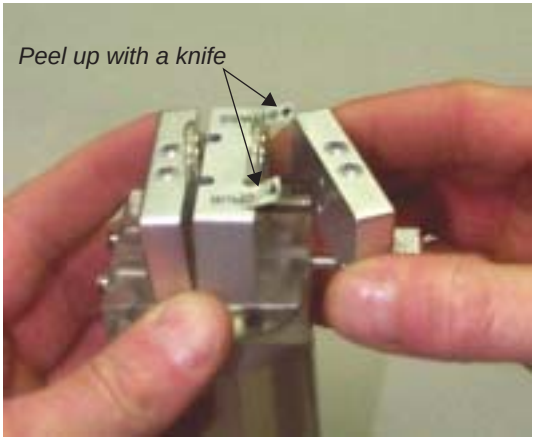
3 Remove 8 screws



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.

Figure 19-5. Removing 8 screws from the flow adjust assembly.

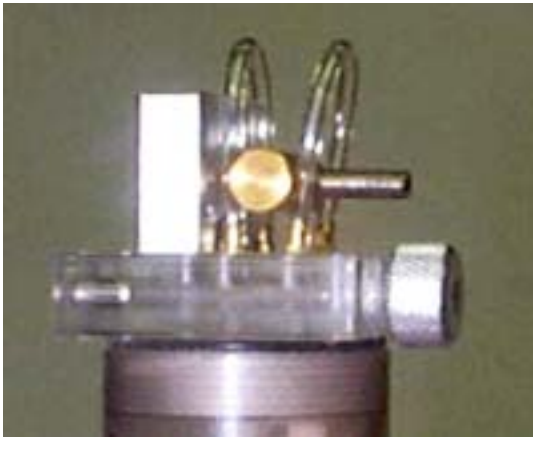
4 Remove the outer and center pieces



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 2 of the 3 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" inside diameter, and 1/8" outside diameter.

Figure 19-7. The flow adjust assembly works by pinching one or the other of 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 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 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, however.)

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).

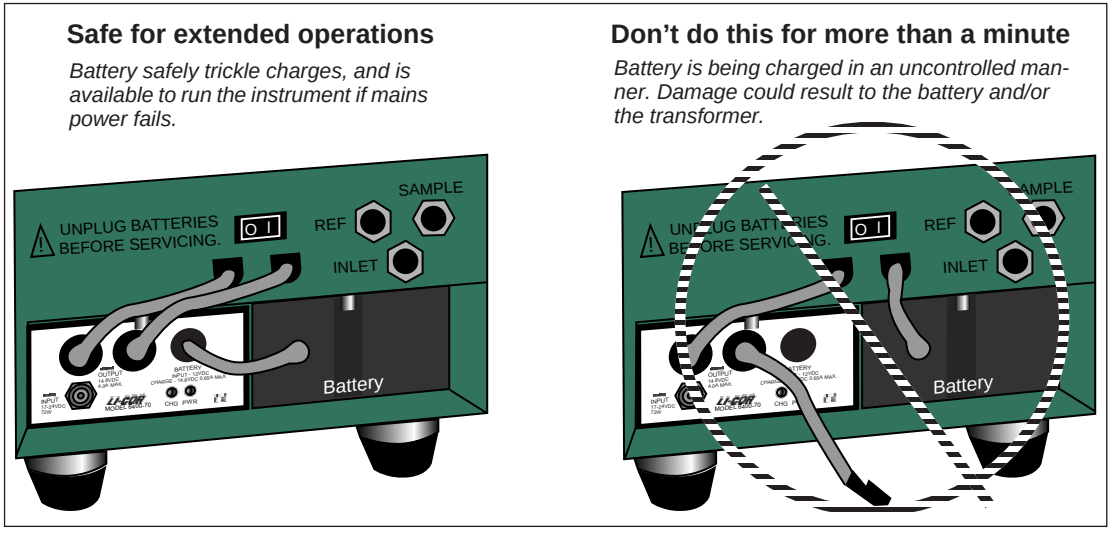


Figure 19-8. Avoid extended (more than a minute) operations with a battery and one of the AC module's plugs connected to the console. In this configuration, you will be providing uncontrolled charging of the battery, which could damage the battery and/or the transformer.

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 (127-00007) (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 20-12 on page 20-41 or Figure 20-16 on page 20-46 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.

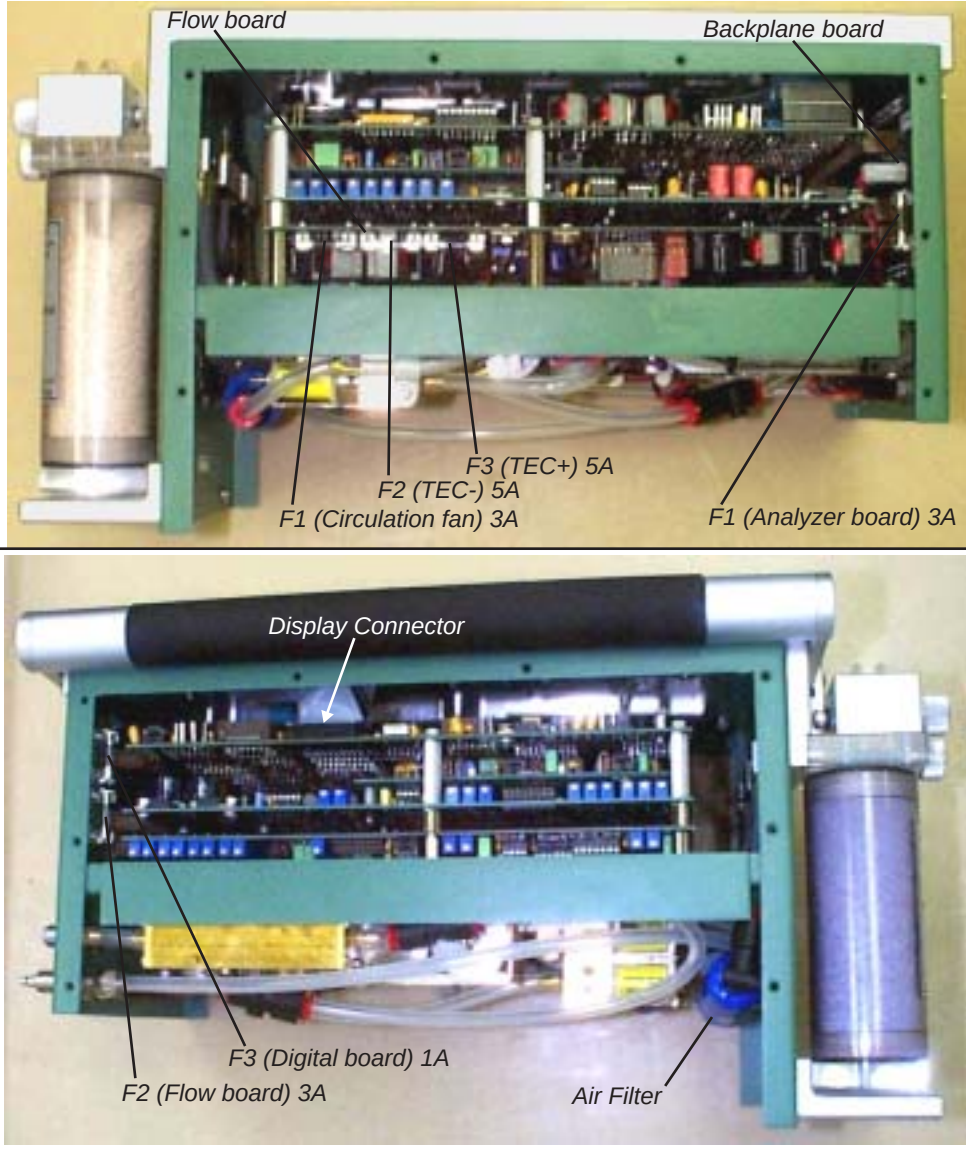


Figure 19-9. Location of console fuses. The backplane board fuses protect the analyzer board, flow board, and digital boards. The flow board fuses protect the thermoelectric coolers and fan.

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).

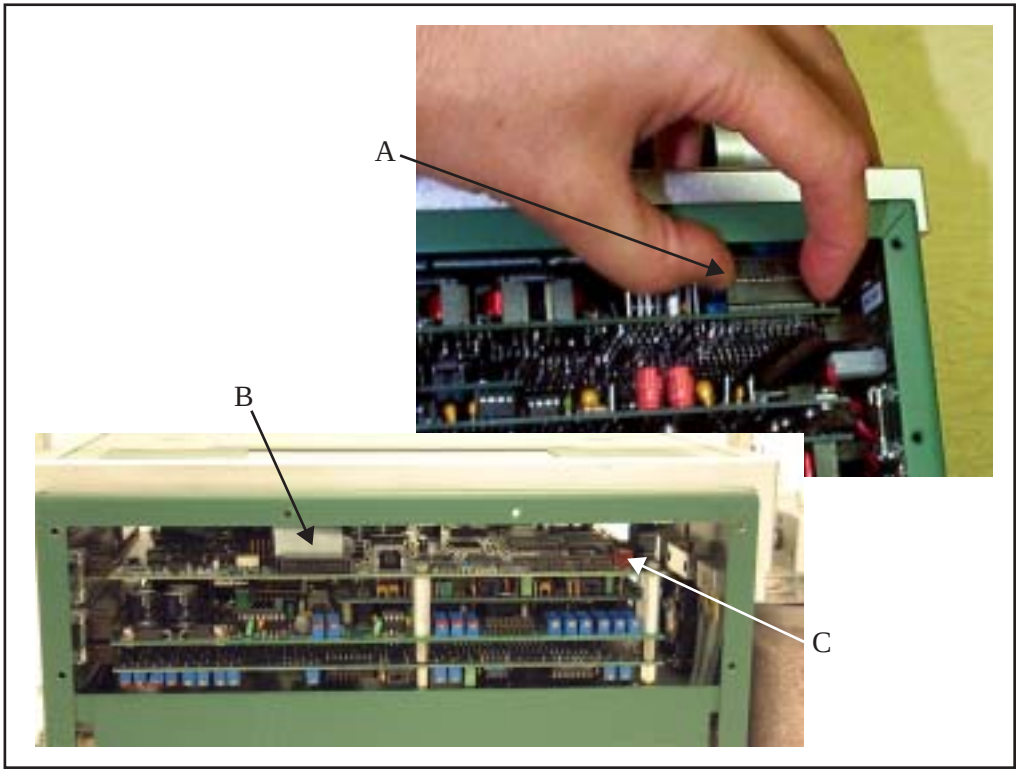


Figure 19-10. The keyboard (A), display (B), and backlight (C) connectors.

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).

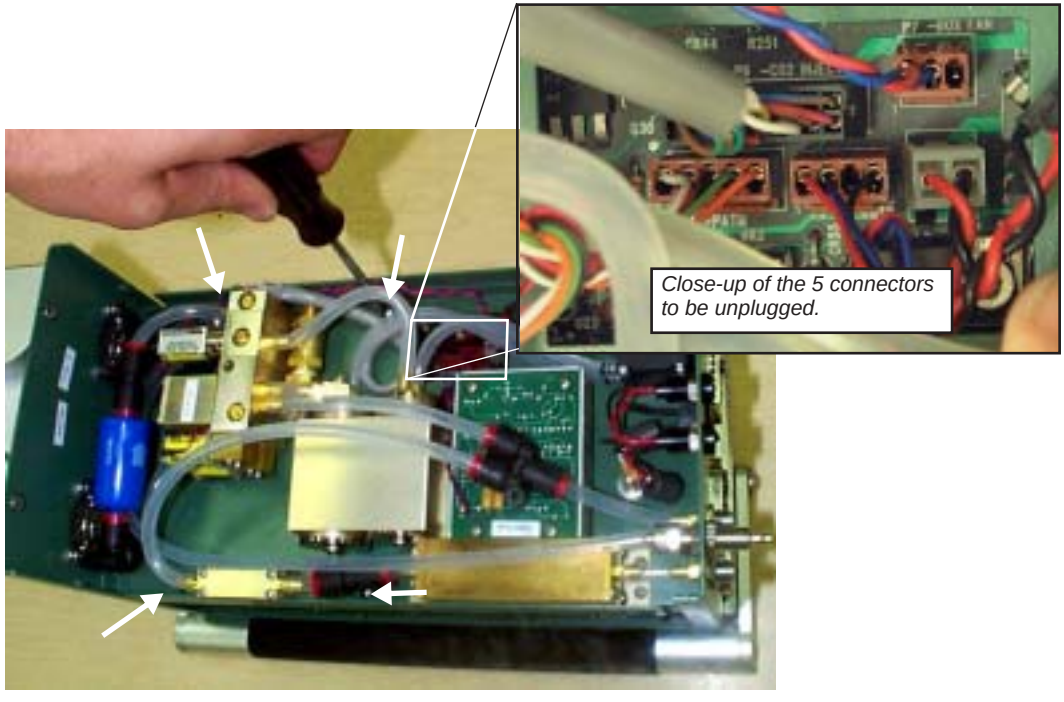


Figure 19-11. The underside of the console, showing the connector well from which all connectors should be unplugged, and the 4 screws that should be removed that hold the board assembly in place.

5 **Remove the board assembly retaining screws**
There are 4 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).

Figure 19-12. Unplug the board assembly from the back plane board by pulling (and rocking side to side gently) on the board stand-offs.

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.

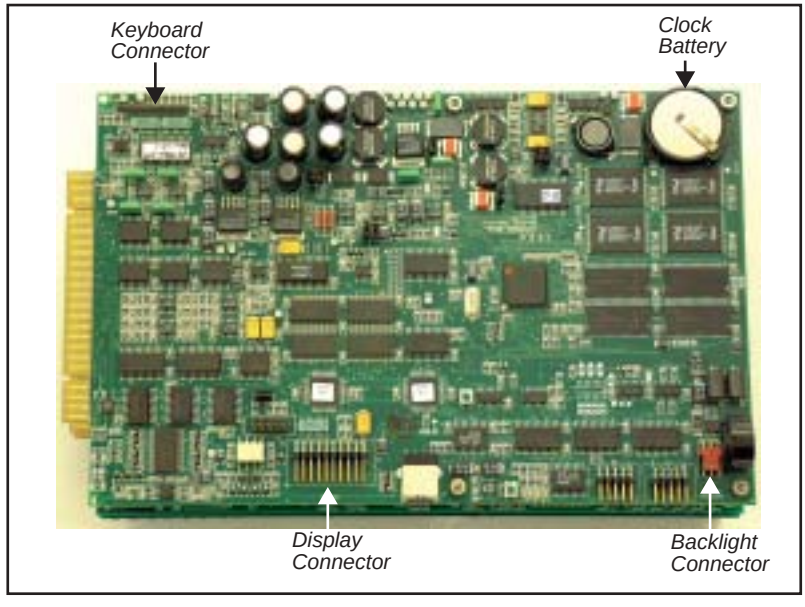


Figure 19-13. The PC boards out of the console. Digital board is the one shown.

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.

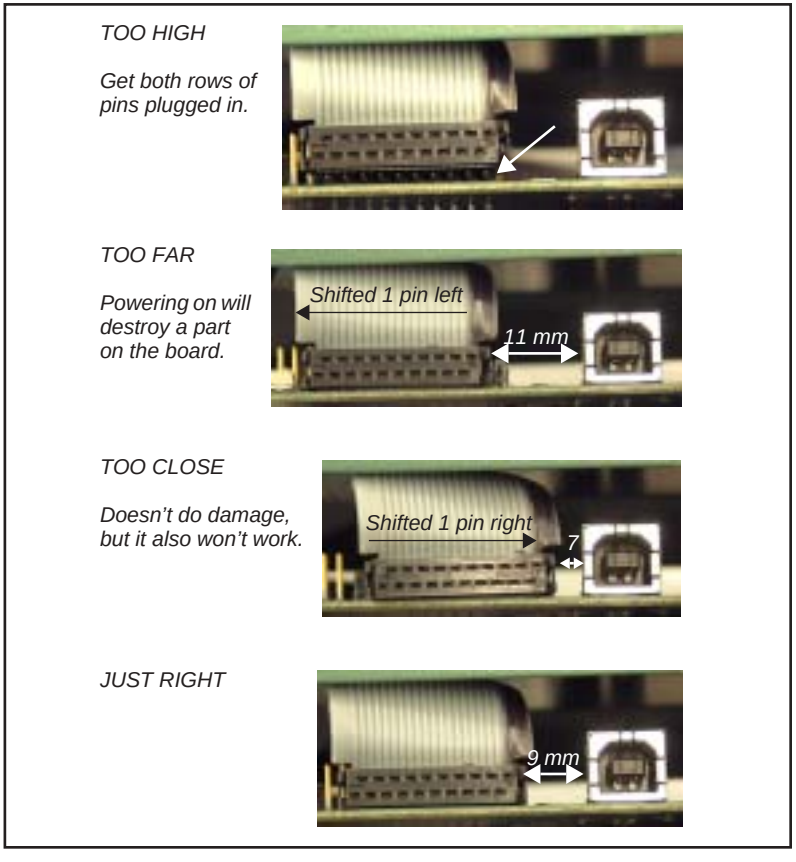


Figure 19-14. Be especially careful to get the display connector centered. If you don't, power on might cause damage.

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.

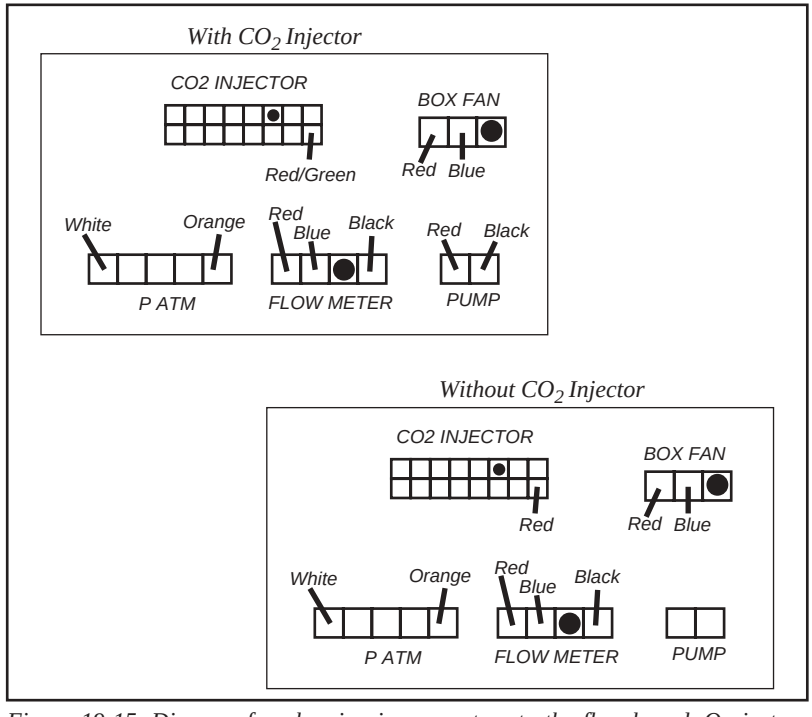


Figure 19-15. Diagram for plugging in connectors to the flow board. On instruments that do not have a CO_2 injector, the pump connector is left empty. The pump plugs in to the CO_2 injector connector, instead.

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 20-6 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.

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).

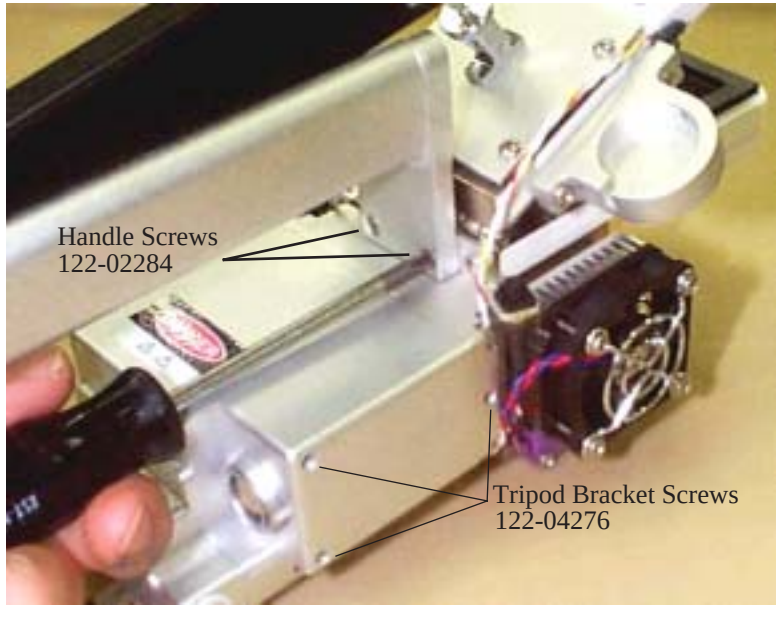


Figure 19-17. Keep the handle screws snug. Older units have three screws.

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.

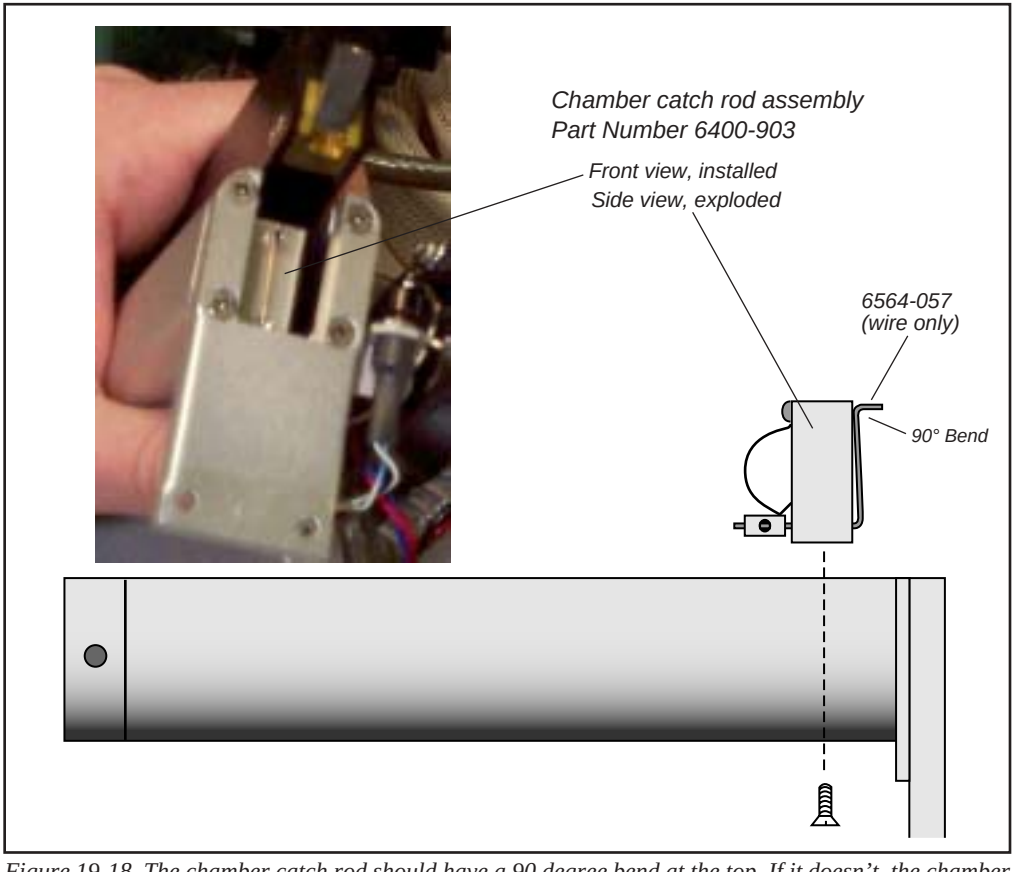


Figure 19-18. The chamber catch rod should have a 90 degree bend at the top. If it doesn't, the chamber will not latch correctly. The catch rod can be re-bent with a pair of needle nosed pliers, if necessary. Should the wire break, you can replace it (6564-057) or replace the entire assembly (6400-903).

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

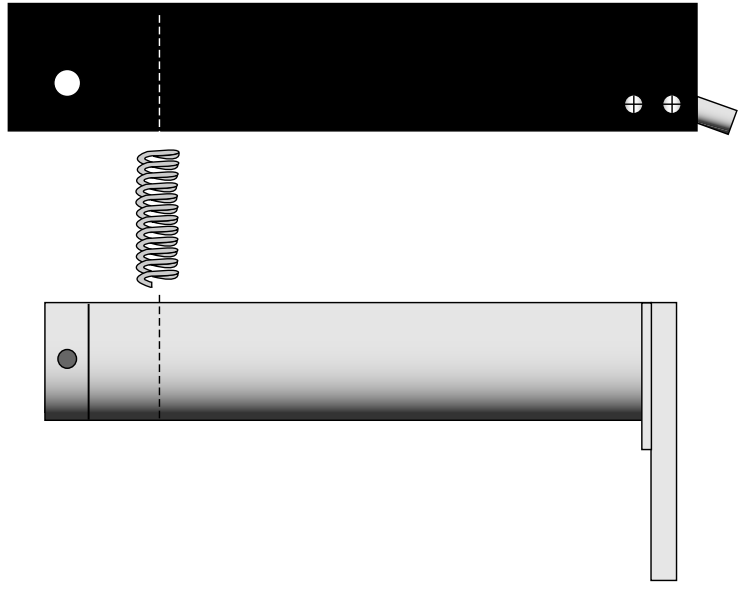


Figure 19-19. The latch return spring is in the rear of the handle. To access, remove the bolt in the rear that serves as a hinge.

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).

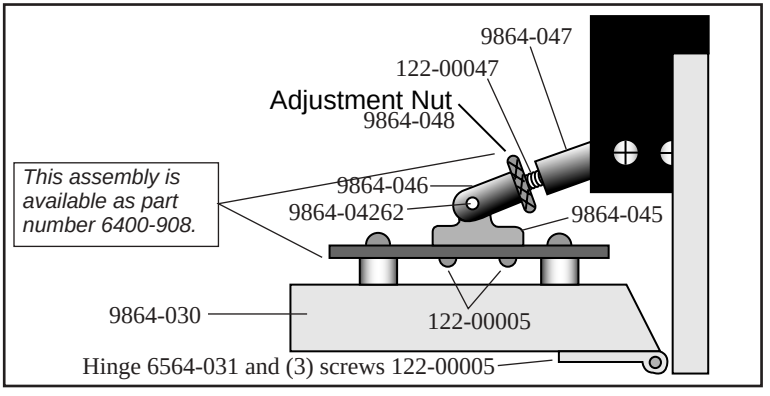


Figure 19-20. Unscrew leaf chamber adjustment nut.

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 handle to rest at the side of the sensor head with the log button attached.

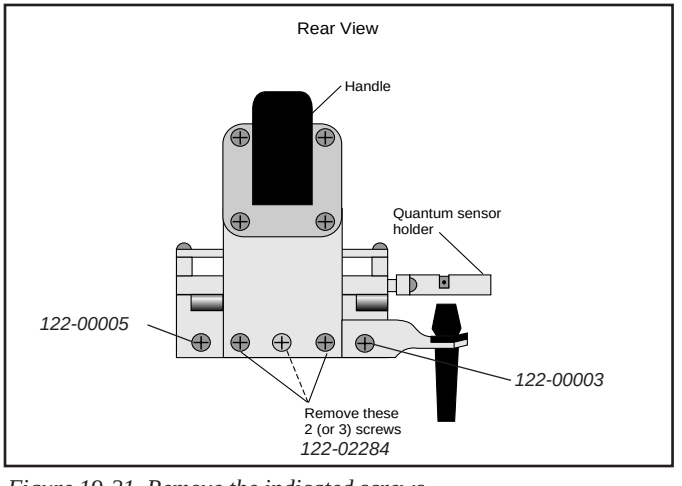


Figure 19-21. Remove the indicated screws.

Leaf Temperature Thermocouple

Thermocouple Maintenance

The thermocouple circuit should be zeroed periodically. The procedure is described in **Zeroing the Leaf Temperature Thermocouple** on page 18-20.

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].

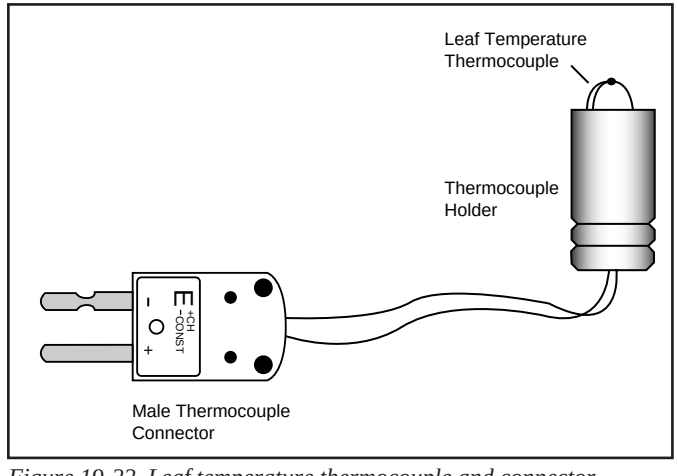


Figure 19-22. Leaf temperature thermocouple and connector.

■ **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 to 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.

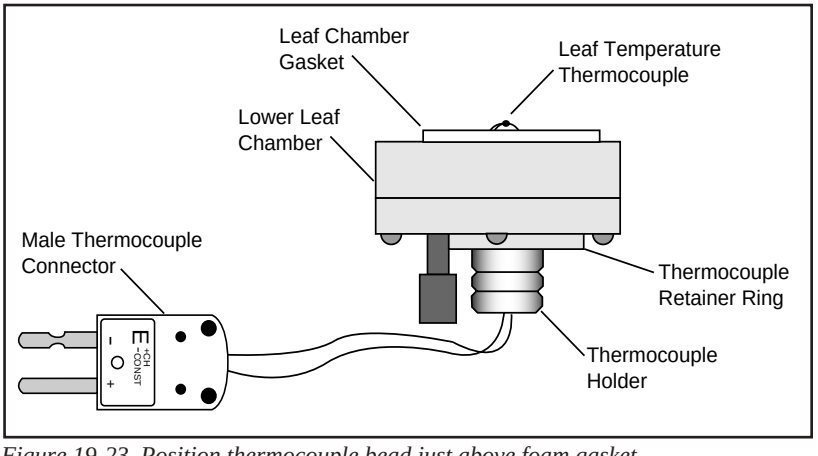


Figure 19-23. Position thermocouple bead just above foam gasket.

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, and the 6400-11 top. You will need to replace 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.

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 tight air 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 your first cut as close to the edge as possible.

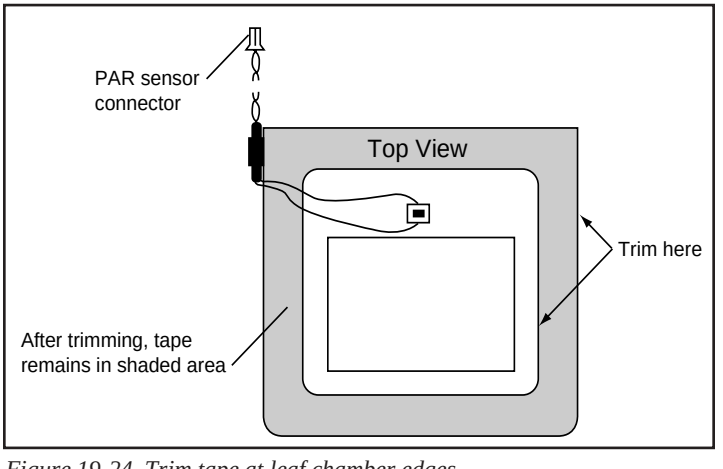


Figure 19-24. Trim tape at leaf chamber edges

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 20-35 for a discussion of how to determine in 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 neoprene 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

In OPEN's Welcome menu there is an entry entitled "Tests and Diagnostics", which leads to a menu containing "Match Valve Tester". This program allows you to "manually" (via the function keys) control the match valve.

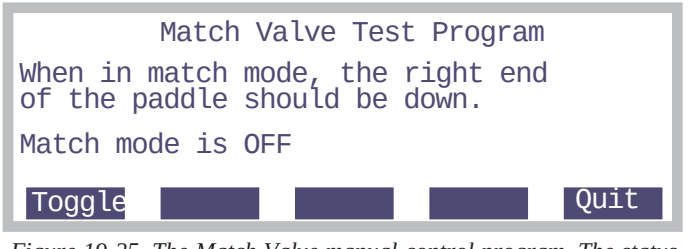


Figure 19-25. The Match Valve manual control program. The status line indicates where the valve should be, not necessarily where it actually is.

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.

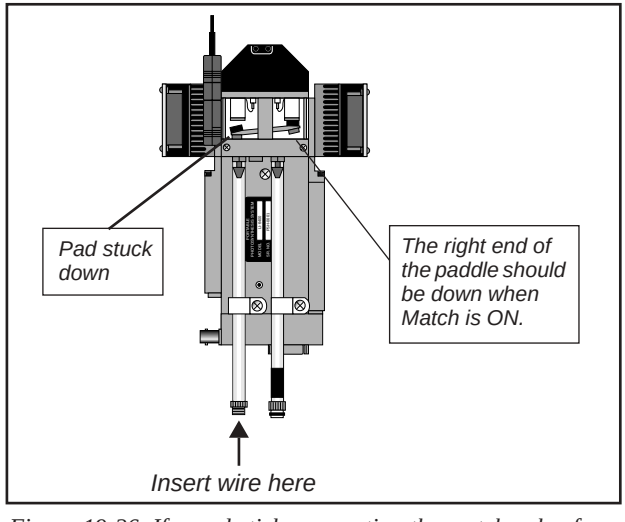


Figure 19-26. If a pad sticks preventing the match valve from operating, insert a thin wire as shown and pop the pad free.

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 4 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*.

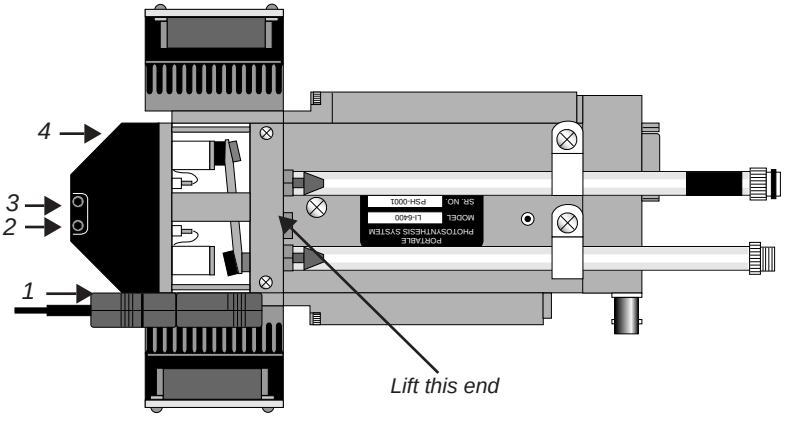


Figure 19-27. Remove the match valve cover by removing four screws, and lifting the cover from the indicated end.

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 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 build 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.

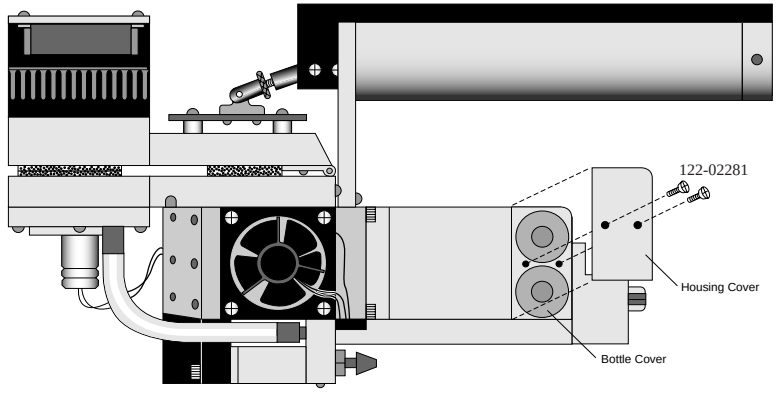


Figure 19-28. Remove the two screws that secure the housing cover.

2 Prepare new bottles

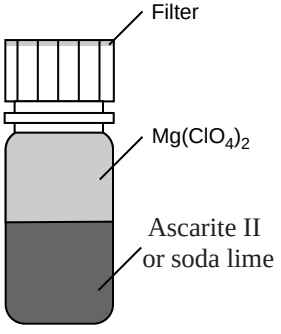


Figure 19-29. Fill the bottles with the CO_2 scrubber, followed by magnesium perchlorate.

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 dry 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-

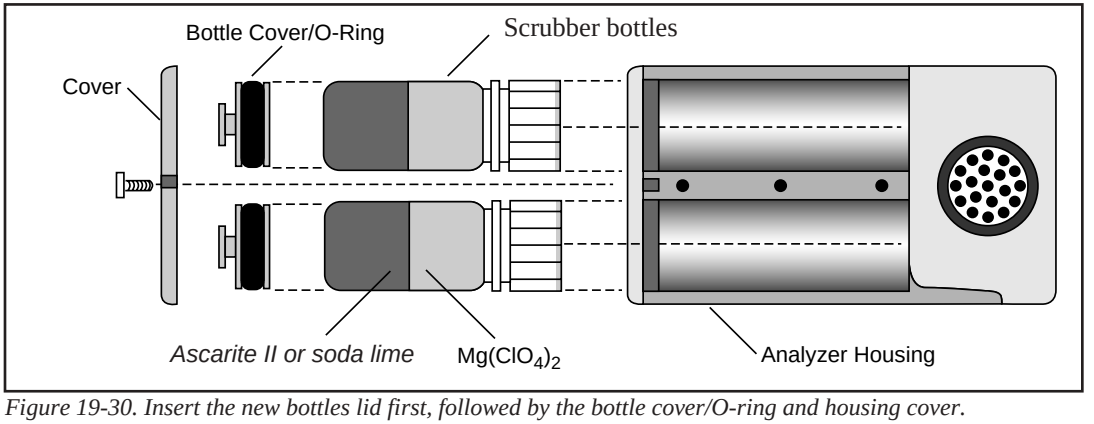


Figure 19-30. Insert the new bottles lid first, followed by the bottle cover/O-ring and housing cover.

mercial suppliers. One type that works well is marketed under the name *Dehydrite*, 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 20-17).

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 6 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 not 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 2 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.

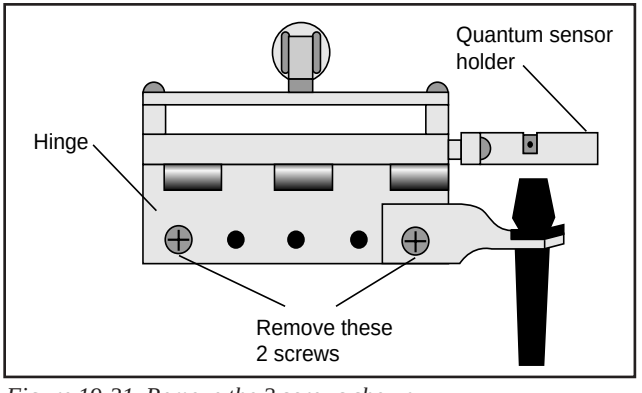


Figure 19-31. Remove the 2 screws shown.

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 8 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.

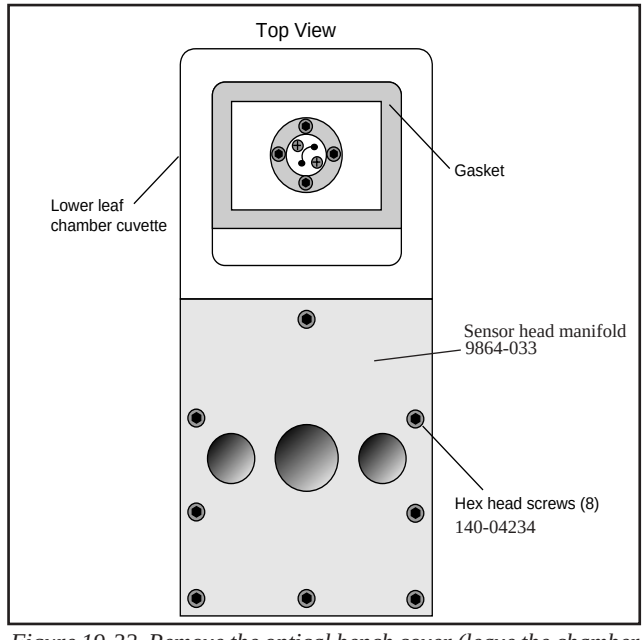


Figure 19-32. Remove the optical bench cover (leave the chamber bottom attached) by removing the 8 screws in the manifold.

4 Clean the windows

Moisten a cotton swab and swab the two optical windows for the sample cell (Figure 19-33).

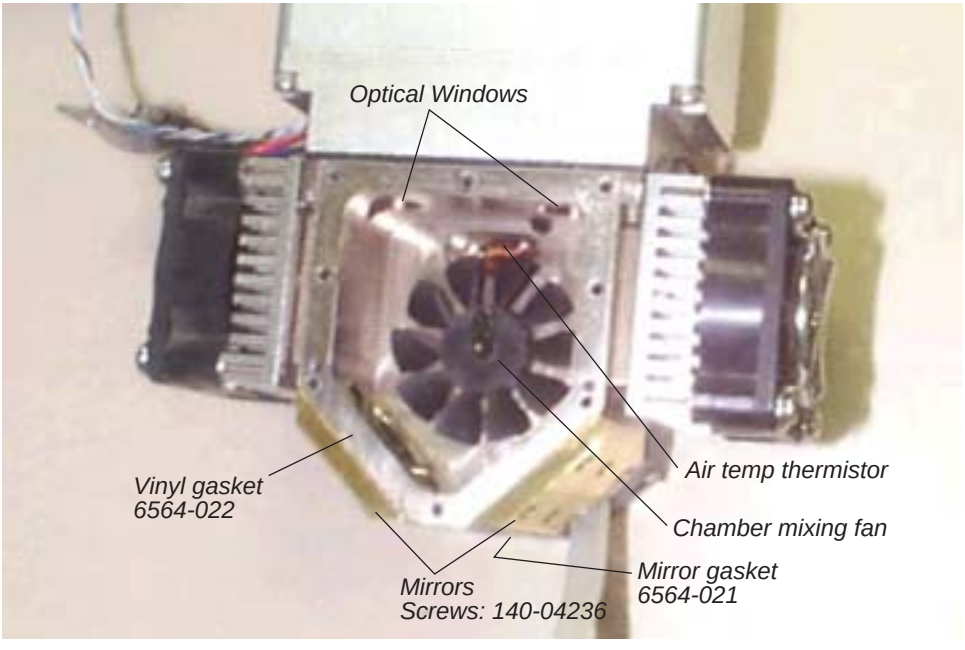


Figure 19-33. Clean the two optical windows.

If you haven't done so already, you can also remove and clean the two gold plated optical mirrors by unscrewing the 6 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 8 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

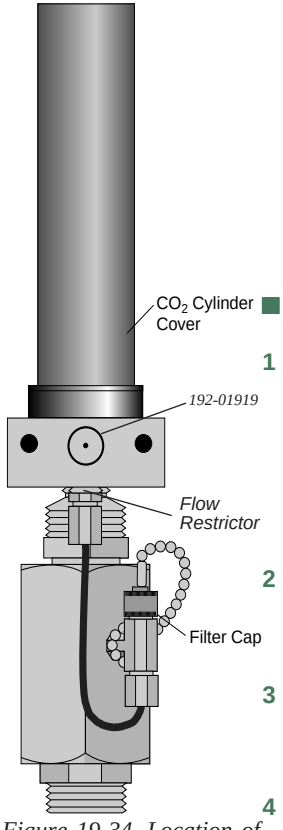


Figure 19-34. Location of oil filter cap.

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.

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, 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 can not 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.

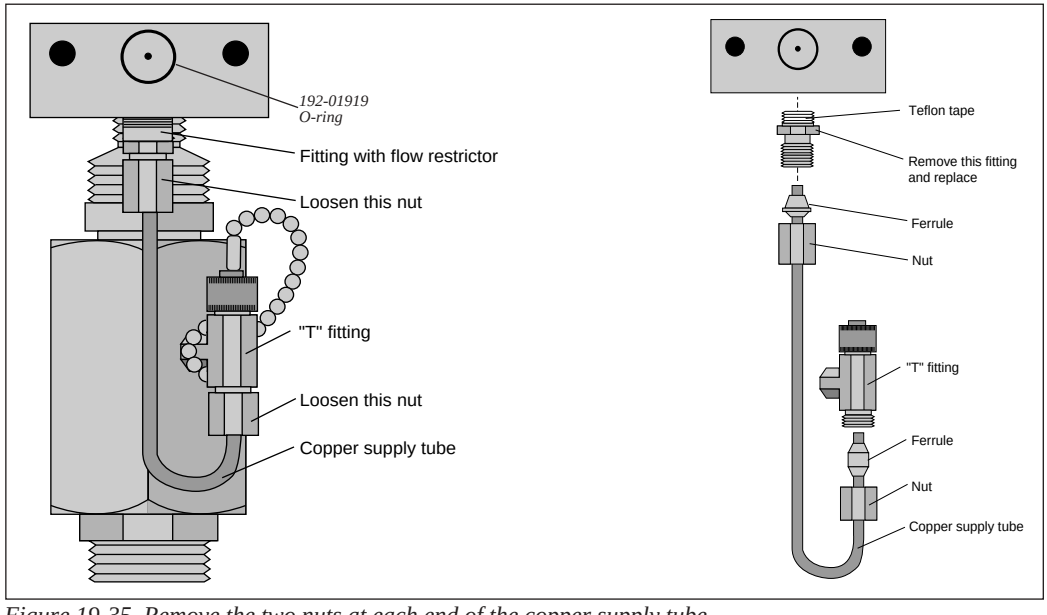


Figure 19-35. Remove the two nuts at each end of the copper supply tube.

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 RTN (Return Tracking Number).**

- **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 don't normally delete files from a customer's /user disk, and we do normally save a copy of the /user disk before we start working on the instrument. 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: LED source. Includes 5 black neoprene gaskets for the lower chamber, and 5 3-hole gaskets.	6400-33
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
Lower half of std 2x3 chamber	9964-029
Upper half of 2x6 chambers	9964-052

Items for Chambers(Continued)	Part Number
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

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
Connector screws for chamber/IRGA cables	314-04913
Chamber handle screws	122-02284
Tripod bracket screws	122-04276
Internal scrubber bottles (2), with chemicals	6400-950

Troubleshooting

20

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 10 second wait while the displays shows

```
INITIALIZING...
```

with 1, 2, then 3 dots. This is followed by the version 5 welcome screen with a 5 second countdown (Figure 20-1).

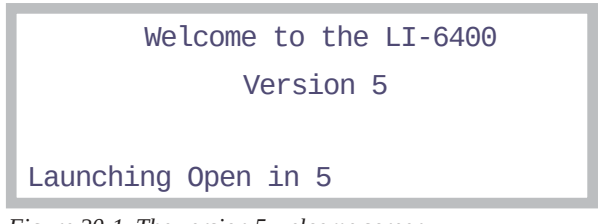


Figure 20-1. The version 5 welcome screen.

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 5-22). Try re-installing system software (**Installing System Software** on page 2-22).

Continually Blows Fuses

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, based on what fuse is blowing:

- **Flow board fuse**
  Items in this circuit are the pump, chamber mixing fan (see **The Mixing Fan** on page 20-34), and 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.

Warning Messages

There are several programs in the /Sys/Autostart folder which run in alphabetical order at power on. Two of them check the state of the factory and user calibration files, and can generate these messages:

"Warning: File '/dev/parm0' is missing"

/dev/parm0 contains factory calibration information. A typical file is shown in Figure 18-22 on page 18-32. If this file is missing, you get this message, and an option to do something about it:

```
Press 'Y' if you wish to correct this
now, or 'N' if you wish to ignore it."
```

If you press **Y**, a generic file will be created. This file will have typical calibration values for the standard sensors. You are then given a chance to edit this file, using a copy of your calibration sheet (which contains your specific /dev/parm0 as of the last time the instrument left the factory).

```
A generic calibration file has been
created. Locate the /dev/parm0 infor-
mation on your cal sheet, and edit this
file accordingly.

Do this now ? (Y/N)
```

If you opt for doing this now, you will get to edit the file and make your changes. When done, press **escape**, then **U** (for update), then **Q** (for quit).

If you don't adjust this file now, each time you power up you'll be warned that you are using a generic calibration file:

"Operating with a generic /dev/parm0 file"
See the above discussion.

"Warning: File '/dev/parm1' is missing"
/dev/parm1 contains the user calibration data (zero and spans). If this file goes away, re-zero your flow meter (**Zeroing the Flow meter** on page 18-18) and recalibrate the IRGAs (**User Calibration (Zero and Span)** on page 18-3). Then store your calibration (**Storing the Zeros and Spans** on page 18-17), which will update /dev/parm1.

"Unknown config command"
This message will appear when OPEN is scanning a configuration file and encounters a configuration command it doesn't recognize. Configurations are discussed in Chapter 16, and a complete list of commands is given in **Configuration Command Summary** on page 16-36.

The program will display the offending command between single quotes, such as

```
'/CO2Mixer='
```

(The problem here is the single slash. A commented-out command should have a double slash //.) If you have recently been editing a configuration file,

and the problem command is something you entered, then you'll need to edit that file and fix the problem. You can either use the Config Editor (page 16-16), or simply edit the file with Standard Edit from the Filer (highlight the file, and press **E**).

Note that configuration commands need a space after the '=', but not one before. See **Configuration Commands** on page 16-3.

If the offending command is not in your configuration file, and it appears to be a calibration-related command, then you might try looking for the problem in /dev/parm0.

Real Time Clock Problems

Doesn't Keep Time When Power Off

The first thing to check is the clock's battery (Figure 20-2). Measure its voltage relative to ground, as shown; it should be 2.7V or greater.

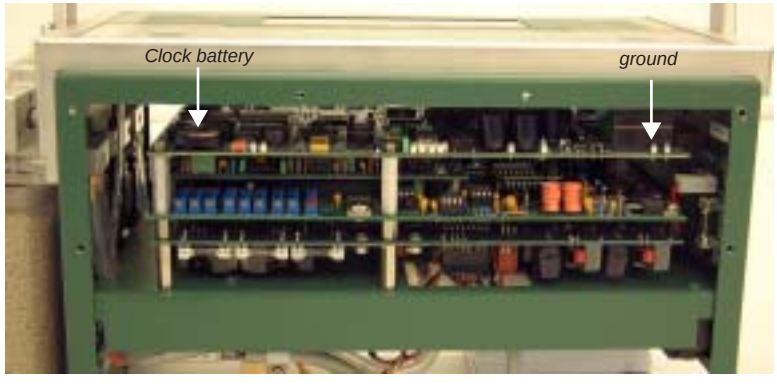


Figure 20-2. Location of the clock battery.

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 20-17.

"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 (14-22) on page 14-10). 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 **f3** (level 3), labelled **FAN**. Note that the instrument cannot sense if the fan is actually moving or not; the message is a response to the switch (digital output) setting. 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 may wish 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 4-51).
- **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 not 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.

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 ΔCO2 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 µmol 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 µmol 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 µmol 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 4-50 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 µmol 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 20-26).

Not using the CO_2 Mixer: A larger buffer volume is probably called for. See **Air Supply Considerations** on page 4-48. 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 20-18.

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 20-39.

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 14-17). 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 (1-16) on page 1-10). 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 18-20)? 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 20-16 on page 20-46.

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 18-21.

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 20-13 on page 20-43.) 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 20-43.

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 20-3). In New Measurements mode, press **[**, followed by **F** to see it.

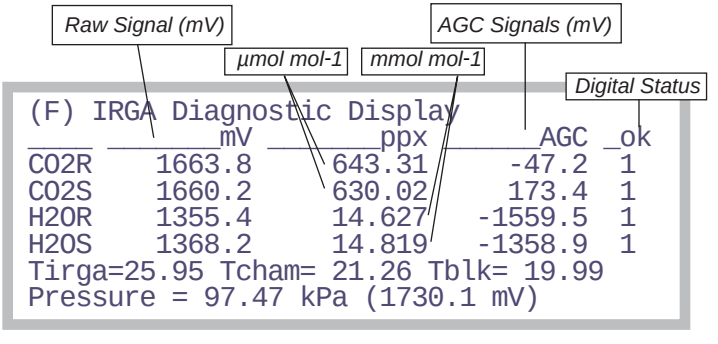


Figure 20-3. IRGA Diagnostic Screen.

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 numbers 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

What should the raw values be? Typically, near 0 at 0 ppm, about 1000 mV at 370 µmol m^{-2} s^{-1}, and near 4500 mV for 3000 µmol m^{-2} s^{-1}. It is a non-linear relationship. For H_2O, 0 mV is 0, 1000 mV is about 10 mmol m^{-2} s^{-1}.

AGC voltages

The AGC signals are shown the IRGA Diagnostic Screen (Figure 20-3 on page 20-15) and also in the text diagnostic display (Figure 20-15 on page 20-45) on level *h*. 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 14-13). 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 20-21). 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 20-16. 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 "Show, View Zero and Spans" 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 4-40). Then you can monitor the running standard deviations of those signals.

[1] 4 second averaging, at 350 µmol mol^{-1} CO_2, and 20 mmol mol^{-1} for H_2O.

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 20-25.

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 20-39. 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 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 20-35.

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 20-39.

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 20-16).

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 20-45), 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 20-15.) 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 14-5 and 14-6.)

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 20-18.

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 **View, Store Zeros & Spans** on page 18-19.) Try resetting them to the factory defaults, or follow the zero and span setting procedures starting on page 18-4. 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 O** (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 20-39).

- **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 O** [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 20-17), a bad cable, or a stalled chopper motor.

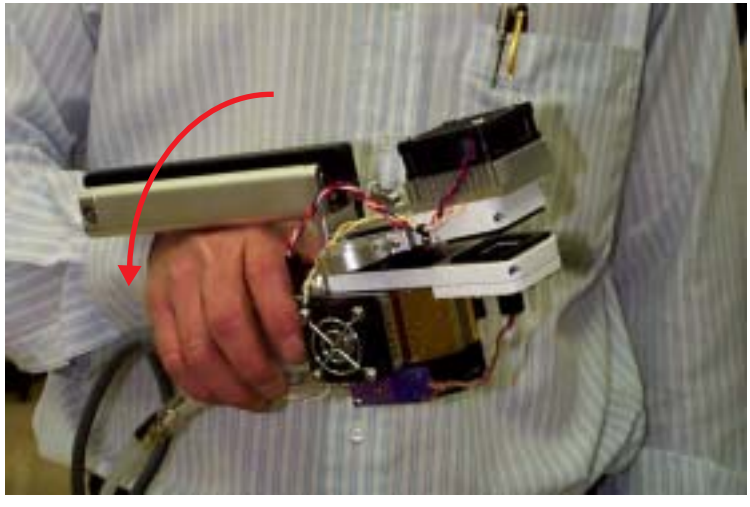


Figure 20-4. Grasp the IRGA as shown, then flick (abruptly turn) your wrist to try to start a stalled chopper motor.

■ **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 20-4). 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 20-39.

"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 $\mu mol\ mol^{-1}$ for CO_2, and 1 $mmol\ mol^{-1}$ for H_2O.

- **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 20-5).

"CO2R Didn't Change" Message

(OPEN 3.2) 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 µmol 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.

Match Valve Doesn't Move

Stuck match valve? See **Match Valve Maintenance** on page 19-29.

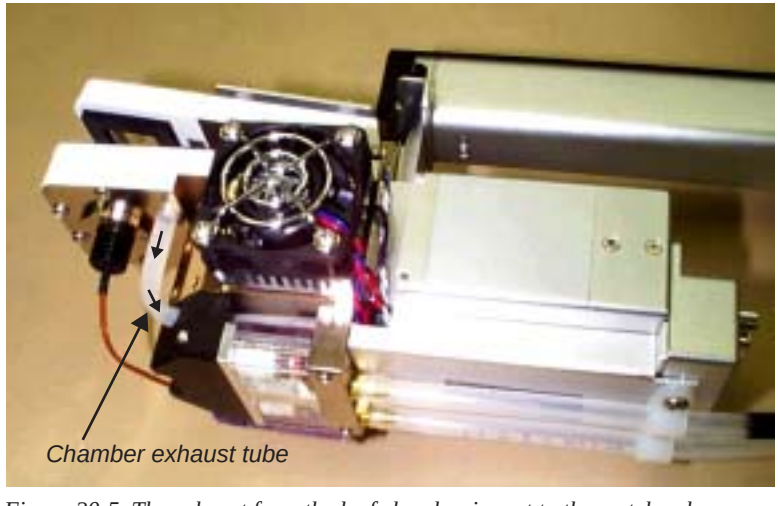


Figure 20-5. The exhaust from the leaf chamber is sent to the match valve.

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 20-6). 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.

Figure 20-6. 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 20-18 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 20-25.

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 20-25. 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 18-21), to improve the mixer's "first guesses".

Can't Achieve Low Values

Typically, the minimum value achieved by the mixer is 30 to 50 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?**
In new measurements mode, press **2**, then **f2**, then **F**, then enter 1000, to request a fixed flow of 1000 µmol s^{-1}. Then monitor flow rate (display line *b*). If the value is considerably lower than 700 µmol s^{-1}, turn the mixer off. If that dramatically increases the flow rate, then the "problem" is that the mixer calibration has specified a reduced flow rate to achieve higher CO_2 concentrations, with the side affect of increasing the minimum concentration. Redo the mixer calibration (**Calibrating the CO2 Mixer** on page 18-21).

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 **Backwards valve**
(If someone has been working on the mixer...) If the valve labelled "Bottom valve" in Figure 20-7 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 20-25.

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 20-7 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.

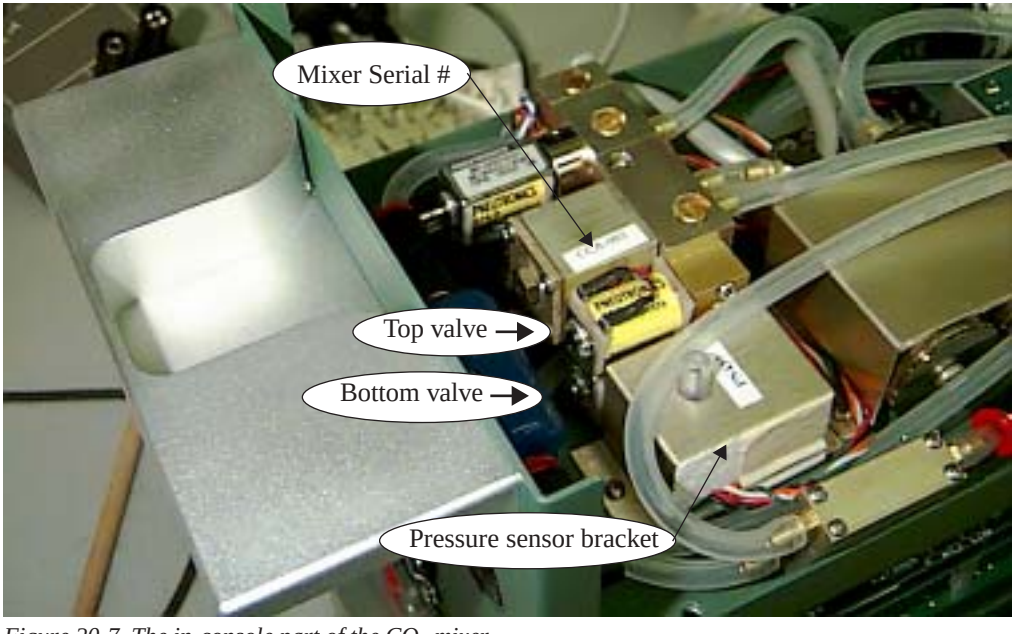


Figure 20-7. The in-console part of the CO_2 mixer.

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 20-42). 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 20-6 on page 20-26.

- **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 Light Source Control program, found in the Config Menu, and specify the LED source. If your LED source is not there, then use the Installation Menu to add it.

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 20-8). Make sure it is not pushed too far in, as it can get into the fan and prevent the blades from turning.

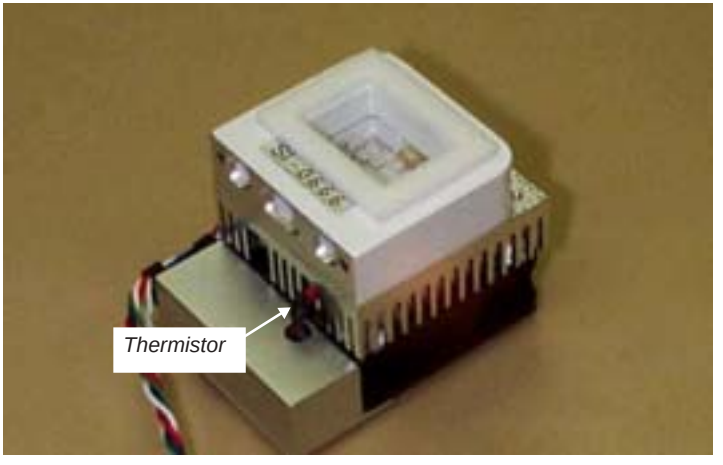


Figure 20-8. The temperature sensor in 6400-02B's.

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 20-9) 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.*

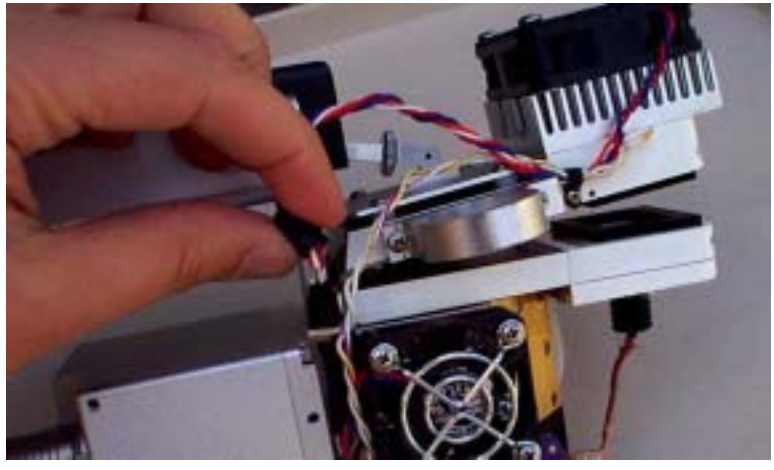


Figure 20-9. Unscrew the lamp connector to access the four wires. Caution - High Voltage.

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 20-47).

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 8-8). 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.

6400-40 Leaf Chamber Fluorometer

See **LCF Troubleshooting** on page 27-68.

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 **f3** level 3.)

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 20-47) 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 won't 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**

- **For OPEN version 3.x and 4.x software:**
  Setup the display graphing to show 2 graphs, Tblk and Tair in degrees C. Set the Y axis to show a range of 1 degree C. Stripchart time can be set to 120 or 180 seconds. You may wish to store this configuration so it can be chosen easily from the quick pick list.

- **For OPEN version 5.x software:**
  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.

In OPEN 3.x and 4.x, the graph's Y axis is fixed at whatever you have specified. If an offset causes a one degree C change in the measured signal, you will see the data displayed go off of the screen momentarily. Then it will start to show you the graph again. With OPEN 5.x, 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 be manifested 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 20-47); 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 20-10) and the wires that come out from beneath it. Something may be pinched.

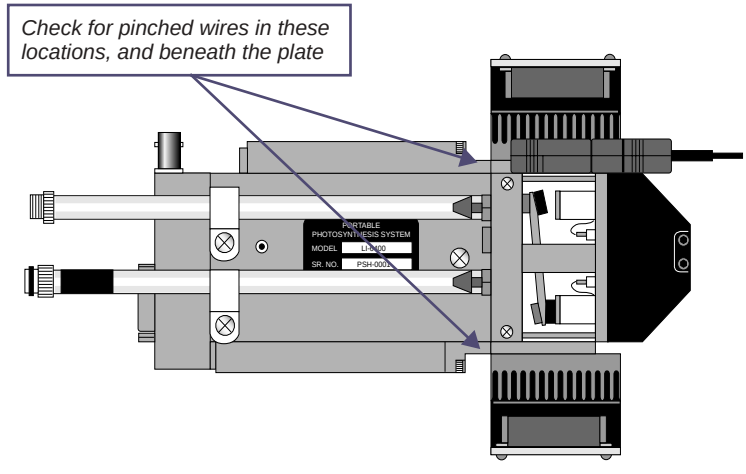


Figure 20-10. Check this metal cover to be sure that it is not pinching any of the wires that come out from beneath it, especially near the match valve.

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 20-35.

■ **Can't Zero Leaf Temperature**
See **Noise caused by the Fan Motor** on page 20-35.

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 20-11.

- **Gaskets**
  Are the centered? Are there a tiny wrinkle beneath that acts like a channel? Don't forget the 3-hole rear gasket.

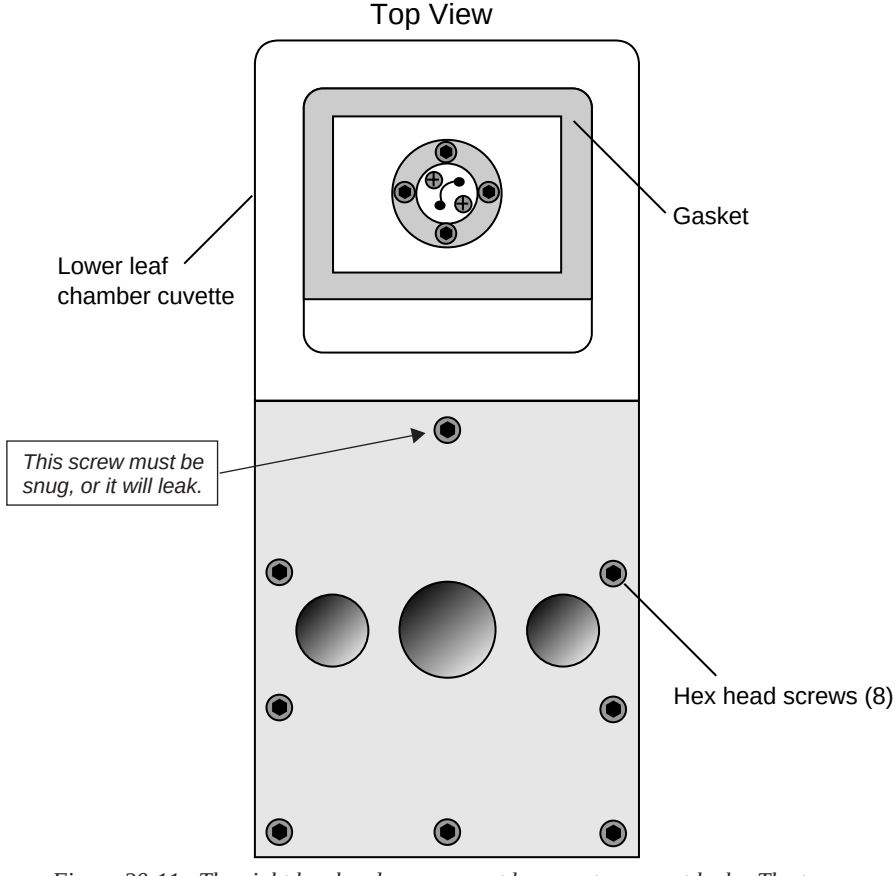


Figure 20-11. The eight hex head screws must be snug to prevent leaks. The top screw is extra important, however, because it passes through one of the air passages in the manifold, and there is just a metal-to-metal seal beneath the screw head.

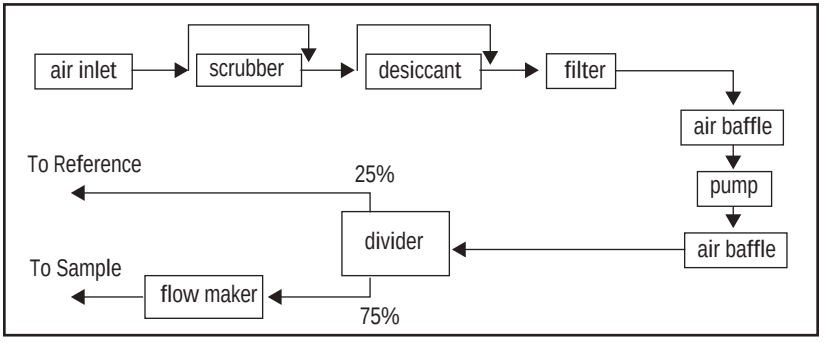
Hose Leaks

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.

Console Leaks

It is helpful to have a simple flow schematic (Figure 20-12) 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

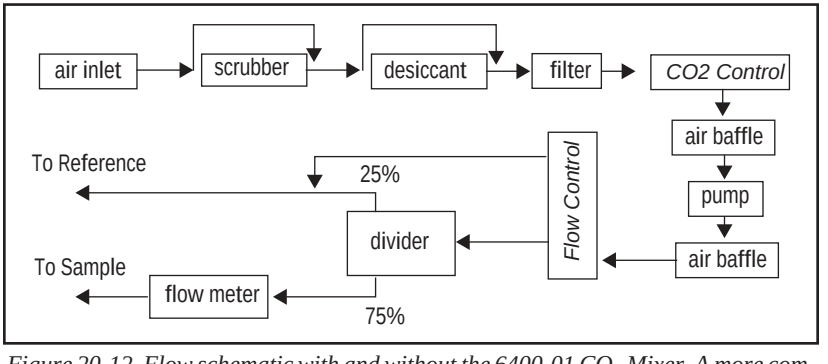


Figure 20-12. Flow schematic with and without the 6400-01 CO_2 Mixer. A more complete schematic is found in Figure 20-16 on page 20-46.

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[3] 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 additional 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

[3] Actually, if it does rotate freely, it could be leaking.

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 leaves a mark on the end of the hose; if this mark is 1/4" from the end, the hose was inserted correctly.

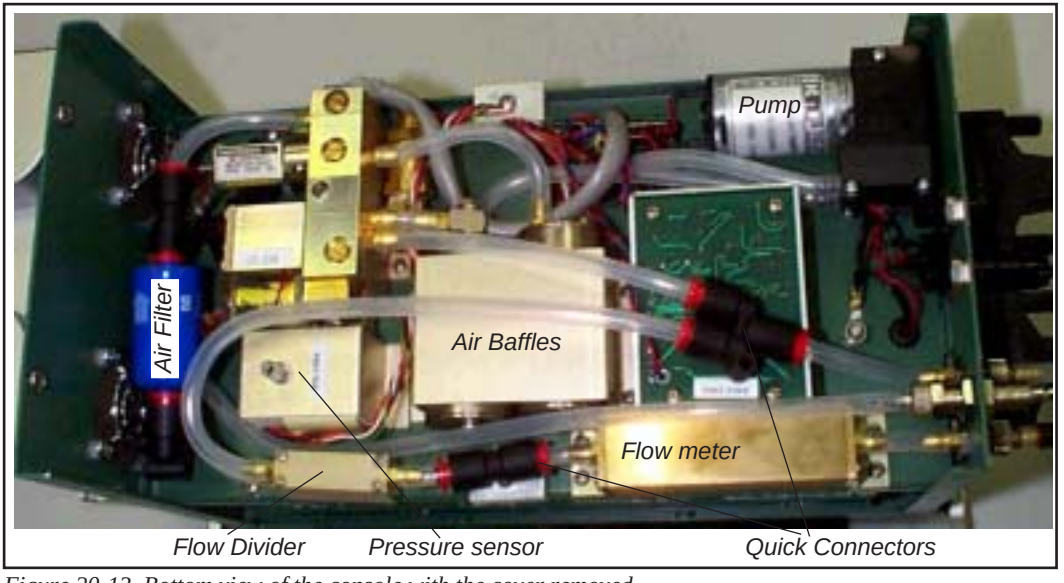


Figure 20-13. Bottom view of the console with the cover removed.

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 20-14) 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 restrictor), put a pencil mark on the outside of the two halves so that it can be reassembled correctly.

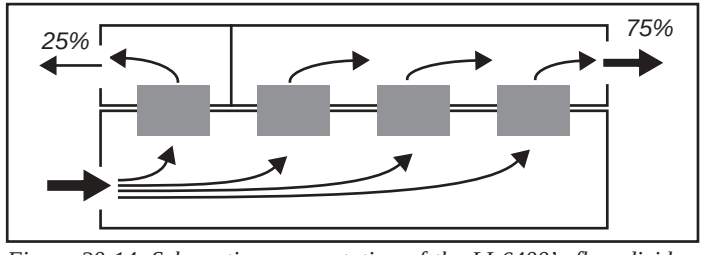


Figure 20-14. Schematic representation of the LI-6400's flow divider, that partitions flow to the sample and reference cells of the IRGA.

Soil Chamber Problems

See **Troubleshooting the Soil Chamber** on page 28-35.

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 20-15). This display is frequently 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.

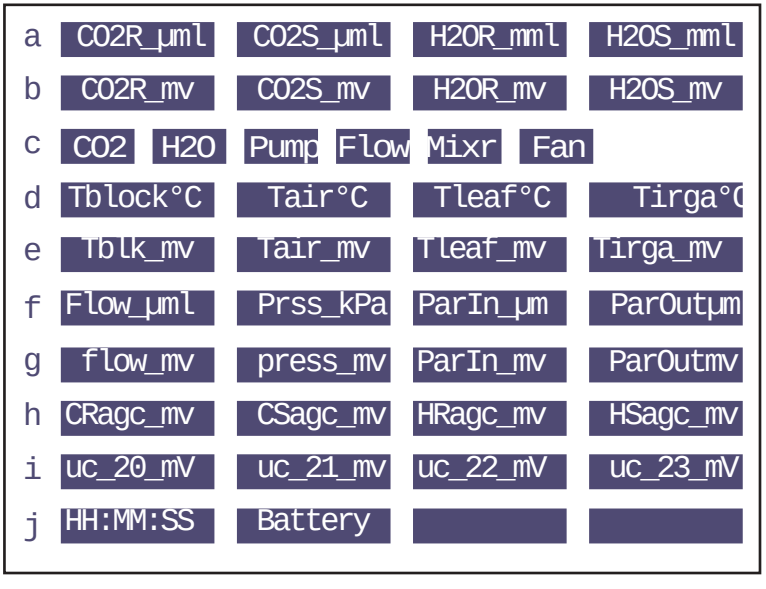


Figure 20-15. The diagnostic display map for New Measurements mode.

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 20-16 provides a schematic of the flow components of the LI-6400.

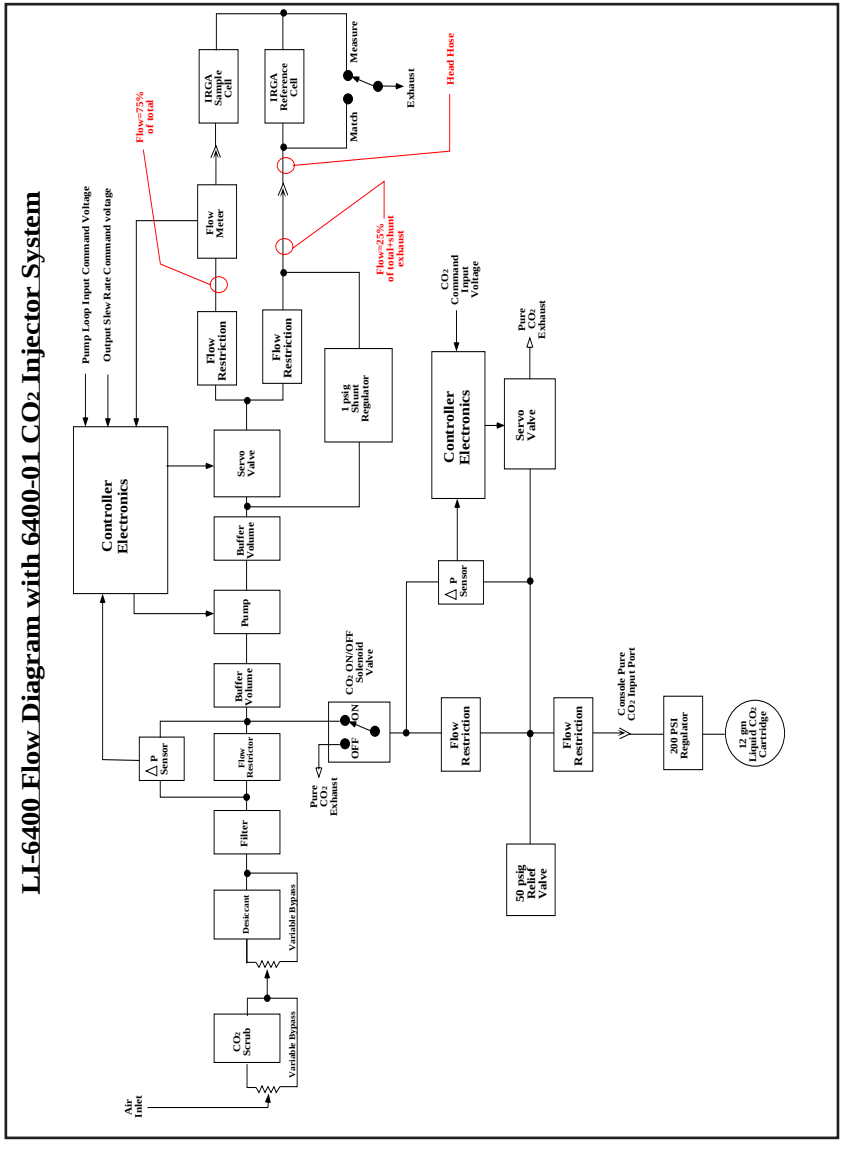


Figure 20-16. LI-6400 system flow schematic, with CO_2 mixer option.

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.

21

Diagnostics and Utilities

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 Welcome Menu. OPEN need not be running for these to work, unless otherwise stated.

"AuxDacTest"

This program tests 4 auxilliary DAC outputs (channels 10, 17, 18, and 19) and 4 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.

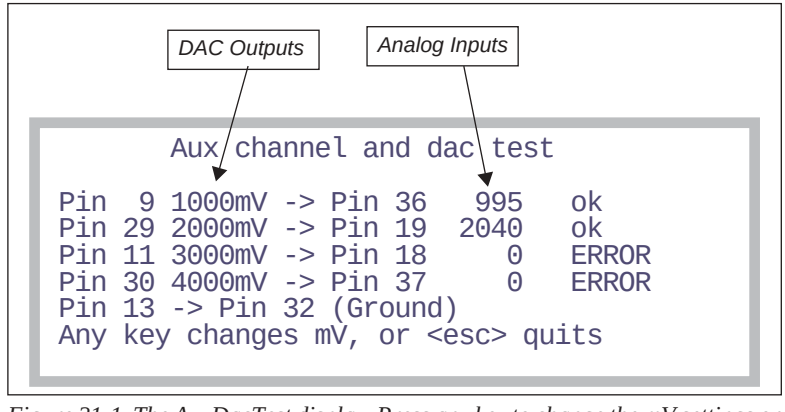


Figure 21-1. The AuxDacTest display. Press any key to change the mV settings on the DACs.. The change cycles between 1-2-3-4 V, 4-3-2-1 V, and 0-0-0-0 V. If the DAC output and the analog input are within 50mV, then the status will show "ok". Otherwise, it will show "ERROR".

"CO2 Mixer Test"

(Requires OPEN). 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 points can be entered from the keyboard, or from a previously defined file, or a standard test can be run.

```
Select Test
  S - Std test
  K - from Keyboard
  F - from file
```

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**

2 **Range of values over the past 5 seconds is less than 1 µmol mol^{-1}.**

Standard Test

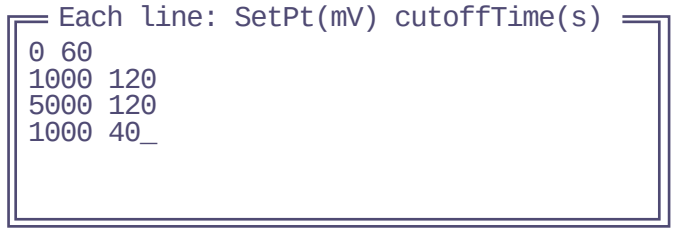
The standard test specifies the following combination of set points and maximum wait times:

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
3000	90		

From Keyboard

Enter a series of set points and cut off times in the window provided.

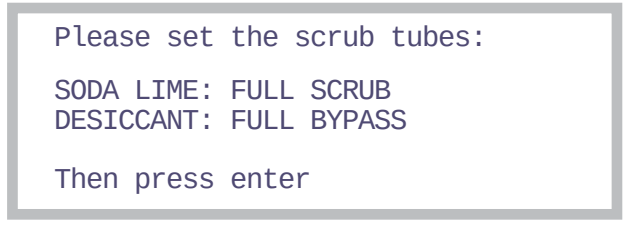


After entering the data, press escape. You will have an opportunity to store your entries in a file for use again, by selecting the "F - from File" option.

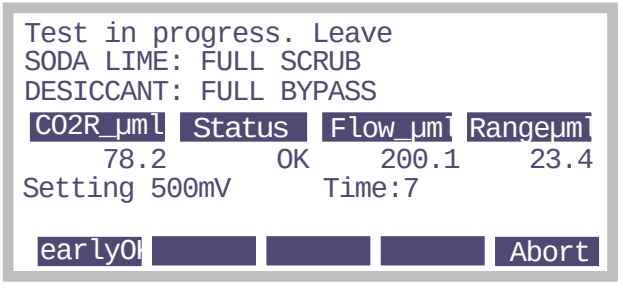
From File

Any file that has pairs of numbers that can be interpreted as set point and wait time can be used, and the file is selected using the Standard File dialog.

Once the input data is established, you are prompted to prepare the chemical tubes.

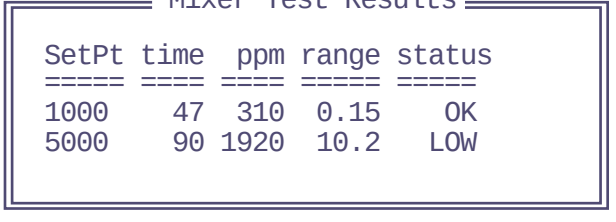


Once you press **enter**, the test begins, and the display will remind you to leave the chemical tubes as set.



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. After viewing the results, press



escape. You are given the option of storing the results.

"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 the user. See **Analog Output Channels** on page 26-18.

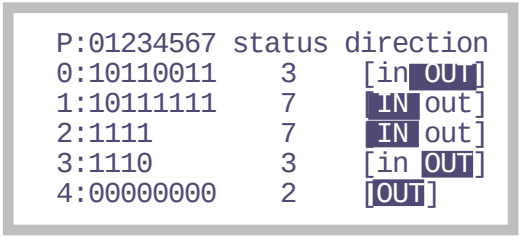
DAC Status

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	
6	4491	Pump
7	4978	Leaf Fan
8	3.4	Cooling Fan
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 21-2. The list produced by the DAC Status program shows each D/A channel and its current value. The channels described by pin nn *are available for user use. For range and resolution information on these spare channels, see* ***Analog Output Channels*** *on page 26-18. (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.



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								Status[a]
	0	1	2	3	4	5	6	7	
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]*	*Input/Output*
1	*Pump status*	*Mixer Hi*	*Mixer Lo*	*Flow Hi*	*Flow Lo*	*Log Button*	*37Pin #4*	*37Pin #22*	*Input/Output*
2	*H2O Ref*	*H2O Smp*	*CO2 Ref*	*CO2 Smp*					*Input/Output*
3	*Pump*	*IRGA board*	*Flow board*	*CO2 solenoid*					*Input/Output*
4[c]	*37Pin #23*	*37Pin #5*	*37Pin #24*	*37Pin #6*	*37Pin #25*	*37Pin #7*	*37Pin #26*	*37Pin #8*	*Output*

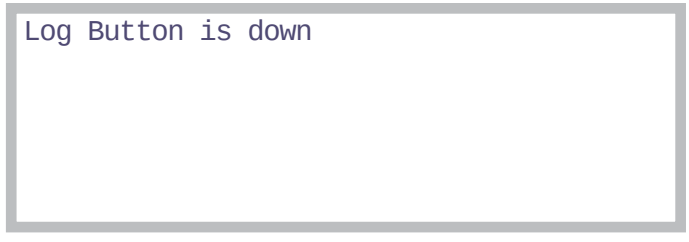
a.In version 5, no port can be switched from between input and output.
b.This signal is also available on pin 21 of the 37 pin external connector (5V = match off, 0V = match on)
c.All of the pins of port 4 are available for user use. See **Digital Output** on page 26-20.

"LCF Control Panel"

This program is for the 6400-40 Leaf Chamber Fluorometer, and is described in **"LCF Control Panel" Program** on page 27-68.

"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.



"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.

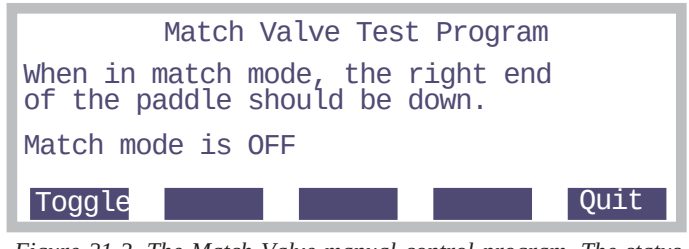


Figure 21-3. The Match Valve manual control program. The status line indicates where the valve should be, not necessarily where it actually is.

"Pressure Sensor"

The pressure sensor test program merely runs the program **"Geopotential"** on page 21-17.

"Sys & User Variable SnapShot"

This program generates a list of all user and system variables, along with their present value. It can be a diagnostic tool.

```
 ID# Label      PresentValue
10 (U/S)       0.832
20 Trans       -4.06E-05
21 Trmmol      -0.0406
23 Cond        -0.00168
30 PHOTO       3.21
35 CndCO2      -0.00105
```

Press escape when done viewing the list (a complete example is shown in Figure 21-4 and Figure 21-5). You will be given an opportunity to store the list in a file.

```
Store this ? (Y/N)
```

Pressing Y will bring up the standard file dialog. The default name is "/user/SnapShot Data".

```
 10 (U/S)      0.834
 20 Trans      0.00458
 21 Trmmol     4.58
 23 Cond       0.28
 30 Photo      -203
 35 CndCO2     0.161
 36 Ci         1.5E+03
 38 Ci_Pa      145
 39 Ci/Ca      5.69
 25 VpdL       1.7
 27 VpdA       1.73
  0 Time       879.5
 -1 CO2R_Êml   21.00
 -2 CO2S_Êml   263.12
 -3  CO2_Êml   242.12
 -4 H2OR_mml   9.338
 -5 H2OS_mml   14.750
 -6  H2O_mml   5.411
 -7 Flow_Êml   500.3
 -8 Tblock_C   24.80
 -9 Tair_C     24.91
-10 Tleaf_C    24.76
-11 Prss_kPa   97.07
-12 ParIn_Êm   -0
-13 ParOutÊm   0
-14 RH_R_%     28.67
-15 RH_S_%     45.28
-16 Td_R__C    5.49
-17 Td_S__C    12.26
-21 HH:MM:SS   "17:02:36"
-22 Program    "-none-   "
-23 CHPWMF     111105
-24 Battery    11.90
-25 CO2        "OK"
-26 H2O        "OK"
-27 PUMP       "OK"
-28 FLOW       "OK"
-29 MIXR       "off"
-30 FAN        "Fast"
-31 CO2fi H2OfiPumpfiFlowfiMixrfi Fan  "  OK
OK   OK   OK  off Fast"
-32 BLC_mol    2.84
-33 AREA_cm2   6
-34 STMRATIO   1
-35 Obs        0
-36 FTime      50.4
-37 CO2R_mv    19.0
-38 CO2S_mv    828.8
-39 H2OR_mv    992.6
-40 H2OS_mv    1440.2
-41 Tblk_mv    2481.0
-42 Tair_mv    2491.1
-43 Tleaf_mv   6.9
-44 flow_mv    1309.4
-45 press_mv   1657.4
```

```
-46 parIn_mv   -0.2
-47 parOutmV   -0.1
-48 CRagc_mv   -368.6
-49 CSagc_mv   370.1
-50 HRagc_mv   -1779.2
-51 HSagc_mv   -1184.2
-53 vapR_kPa   0.91
-54 vapS_kPa   1.43
-55 BLC1_mol   1.42
-56 ProgPrgs   "        "
-57 FwMxCrLp   " 1 1 1 0"
-58 Tirga_C    24.81
-59 Tirga_mv   2247.9
-60 uc_20_mV   0.0
-61 uc_21_mV   0.0
-62 uc_22_mV   0.0
-63 uc_23_mV   0.0
-64 DecHour    17.04333
-65 CsMch      0
-66 HsMch      0
-67 CrMch      0
-68 HrMch      0
-69 DOY        195
-70 YYYYMMDD   20020714
-71 Stable     "0/3"
-72 StableF    0.0
-73 CHF        "000"
-80 F          0.0
-81 %Blue      0
-82 FlrMax_    0.0
-83 FPeak_Êm   0
-84 FCnt       0
-85 Fzero      -3000.0
-86 Fo         0.0
-87 Fo'        0.0
-88 Fm         0.0
-89 Fm'        0.0
-90 Fs         0.0
-91 FlrEvent   "---      "
-92 M:Int  kHz  Hz  Gn  "  "
-93 F:Dur Int Blu  kHz  Hz  "  "
-94 D:Dur Far Bfr Aft  kHz  Hz  "      "
-95 FlrMin     0.0
-96 ParIn@Fs   0.0
-97 FlrCV_%    0.00
 -98 dF/dt     0
-101 AuxF1     0
-102 AuxF2     0
-103 AuxF3     0
-104 AuxN1     0
-105 AuxN2     0
-106 AuxN3     0
-107 Remark8   "        "
-108 Remark18  "                  "
-109 Remark38  "                     "
```

Figure 21-4. Part I. Sample data from the "Sys & User Variable Snapshot" program, shown in two columns.

```
"*** Other variables ***
fuse_mv = 4992.043945
d2aSlopH = 0.000000
d2aSlopF = 7.327393
d2aSlopLimit = 50.000000

*** /dev/parm0 ***
// Calibration Data: G780991213
CO2Mixer= YES
CalCO2= 0.2137 4.1871E-05 8.7851E-09 -
3.8E-13 6.3366E-17
CalH2O= 0.0061259 2.2982E-06 5.3246E-12
CalZero= -2.7 -2.4
CalFlow= 0.38199
CalPress= 88.022 0.005461
CalParGaAs= 0.82 // GA-780 28 Sep 1999
CalParOut= 7.05 // Q27343 23 Nov 1999
CalParLED= -0.81 // SI-912 17 Aug 1999
CalParLED= -0.77 // SI-1176
CalParLED= -0.60 // SI-999 today
ThisUnit= "PSC-0780"
Serviced= "13 Dec 1999"

*** /dev/parm1 ***
FuseAware=0
FFlowZero=0.0
UFlowZero=468.8
FZero=0.0 0.0 0.0 0.0
UZero=195.3 0.0 39.1 0.0
FZerTemp=20.00 20.00
UZerTemp=20.00 20.00
FGains=1.000 1.000 1.000 1.000
UGains=0.971 0.983 0.964 0.959

*** /User/Configs/history ***

Thr Jun  6 2002 10:25:17
UFlowZero=507.8

Thr Jun  6 2002 10:25:18
UZero=195.3 0.0 0.0 0.0
UZerTemp=20.00 20.00

Fri Jun  7 2002 09:03:13
UZero=195.3 -39.1 39.1 -39.1
UZerTemp=20.00 20.00

Fri Jun  7 2002 09:03:14U
Gains=0.972 0.982 0.966 0.969

Fri Jun  7 2002 09:38:45
UFlowZero=468.8

Mon Jun 10 2002 13:51:53
UZero=195.3 0.0 39.1 0.0
UZerTemp=20.00 20.00

Mon Jun 10 2002 13:51:53
UGains=0.971 0.983 0.964 0.959

*** /User/Configs/lamp ***
48.828 100.1  1001 3000.5 4997.6
24.914 59.805 725.47 2097.3 3030.8

*** /User/Configs/mixer ***
4500.000000
 5000  3000  1500  1000   500   300   200
100
2066.5 990.65 396.67 242.08 109.71 63.074
41.37 38.431

*** /User/Configs/fluor ***
48.828 100.1 200.2 400.39 800.78  1499
77.272 174.73 373.17 781.37 1602.2 2986.4
117.19 507.81 1015.6 1992.2 3007.8 3984.4
4960.9
7.8965 32.498 61.451 110.19 153.81 190.69
223.3

*** /User/Configs/MatchDisplays ***
ISPLAYMAP= 2
-1 -2 -4 -5
-3 -6 -9 -16
```

Figure 21-5. Part II. The rest of the snap shot data.

/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)
```

"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)
```

"Control Panel"

This program allows direct access to most of the analog and digital outputs of the LI-6400 (Figure 21-6). The complete list is presented in Table 21-3.

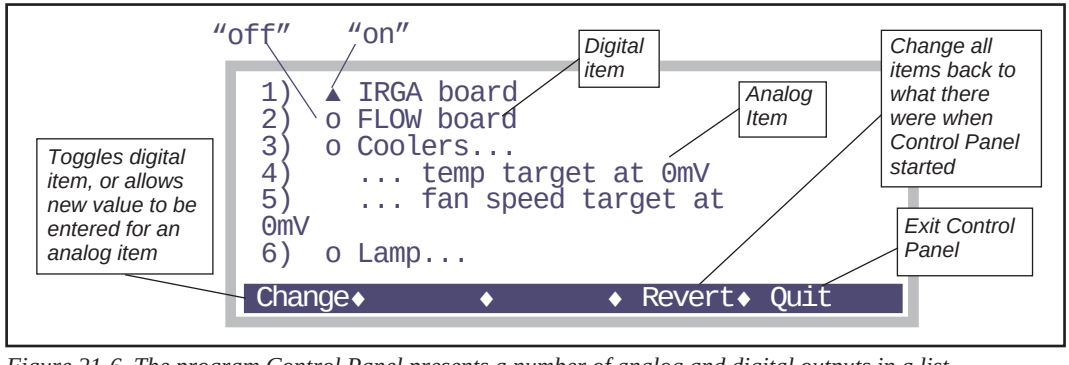


Figure 21-6. The program Control Panel presents a number of analog and digital outputs in a list.

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.
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)	Digital outputs available on the 37 pin console connector. LPL address shown in (). See **Digital Output** on page 26-20. Off is (o), on is (▲).
17	User Digital: Pin 5 (0x0401)	
18	User Digital: Pin 24 (0x0402)	
19	User Digital: Pin 6 (0x0403)	
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)	Analog outputs available on the 37 pin connector. LPL address shown in (). See **Analog Output Channels** on page 26-18.
25	Analog Output: Pin 27 (9)	
26	Analog Output: Pin 9 (10)	
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 14-17.

```
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
```

Figure 21-7. `"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. The listing for this program is shown below.

```
/*******************

  Energy Balance Analysis

*******************/

:FCT EBmain
{
  0 :FLOAT p_kpa
  0 :FLOAT tair
  0 :FLOAT vap_kpa
  0 :FLOAT trans_mmol

  LOOP
    "Enter: P_kPa vap_kPa Trans_mmol Tair\n" PRINT
    &tair &trans_mmol &vap_kPa &p_kPa "%f%*c%f%*c%f%*c%f" KBD
ENTER
```

```
    4 <> IF RETURN THEN

    p_kpa vap_kPa trans_mmol 1000 / tair
    EnergyBalanceAnalysis
    "BLC = %1.3G Tleaf = %1.3G\n\n" PRINT
  ENDLOOP
}

:FLOAT
  _EBtrans_mol 0 /* trans. mol/m2/s */
  _EBtair  0     /* air temp C */
  _EBvap_kPa 0   /* vapor press kPa */
  _EBp_kPa 0     /* barometric press kPa */

  _EBepsilon 0.95 /* emissivity */
  _EBsigma 5.67E-8 /* Steffan-Boltzman */
  _EBcp 29.3      /* heap cap air J/mol/K */
  _EBbbTair 0    /* black body rad @ tair */

:FCT
EnergyBalanceAnalysis
{
  /* press, vap, trans, tair */
  /* tleaf, blc */
  &_EBtair =
  &_EBtrans_mol =
  &_EBvap_kPa =
  &_EBp_kpa =

  0 :FLOAT t2
  0 :FLOAT lnRatio
  0 :FLOAT tleaf
  0 :FLOAT blc_mol
$
  _EBbbTair = EBBlackBody(_EBtair)

  lnRatio = LOG(_EBvap_kPa / 0.61365)
  tleaf = 240.97 * lnRatio / (17.502 - lnRatio) + 0.5
  tleaf = EBIterate(&EBQDenom, tleaf)
  tleaf = EBIterate(&EBEnergyBal, tleaf)
  blc_mol =  _EBtrans_mol * _EBp_kPa / (EBSatVap(tleaf) -
_EBvap_kPa)
  RETURN(tleaf, blc_mol)
}

EBIterate
{
  /* &fct, FirstGuessT */
  /* final T */

  :FLOAT t1
  :PTR fct
  0 :FLOAT x
  0 :FLOAT t2
$
  LOOP
    x = fct(t1)
    t2 = t1 - 0.01 * x  / (fct(t1 + 0.01) - x)
    IF (ABS(t1 - t2) < 0.02)
      RETURN (t2)
```

```
    THEN
    t1 = t2;
  ENDLOOP
}

EBEnergyBal
{
  /* temp (C) */
  /* computed temp */
  :FLOAT t
  $
 RETURN ((_EBbbTair - EBBlackBody(t))/EBQDenom(t) - _EBtrans_mol)
}

EBCpStuff
{
  /* t (C) */
  /* */
  _EBtair - _EBp_kPa * 0.93 * _EBcp *
}

EBBlackBody
{
  /* t (C) */
  /* W/m2 */
  273.2 + 4 ^ _EBsigma * _EBepsilon * 2.0 *
}

EBSatVap
{
  /* t (C) */
  /* kPa */
  :FLOAT t

  $ RETURN(0.61365 * EXP(17.502 * t / (240.97 + t)))
}

EBLambdaFn
{
  /* t (C) */
  /* latent heat af vapor, W/m2 */

  42.91 * 45060.0 +
}

EBQDenom
{
  /* t (C) */
  :FLOAT t
  $ RETURN (EBLambdaFn(t) + EBCpStuff(t) / EBVpd(t))
}

EBVpd
{
  /* t (C) */
  /* guess */
  EBSatVap _EBvap_kPa -
}
```

"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, of course. 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 you do. Press **escape** to terminate the program.

```
            Pressure:   99.27 kPa
Geopotential Height:     610 ft
                         186 m
```

"Graphics Capture"

Note: This program is hanging around for historical reasons. It really is no longer necessary, since you can generate a graphics image by pressing **ctrl** + **s** anytime you are viewing a graph. The images are saved in the /User/Images directory, and given a name that reflects the time and date when you pressed **ctrl** + **s**. See Figure 21-8 on page 21-18.

Graphics Capture will save the current state of the graphics display to a file. The file is named using the Standard File Dialog (page 5-9). Files created in this manner can be re-loaded to the graphics display via the program "Graphic Restore", described next. Graphics image files can also be downloaded to computer, although to do anything with them there, you will need to write a program to decode the data (see **Storing and Retrieving Graphics Images** on page 12-20).

```
DIR: /User/images
File:a-ci curve 3
        Store Graphics to...
"a-ci curve 1"
"a-ci curve 2"

+ Dir   ◆ Help  ◆       ◆ CANCEL◆ SELECT
```

"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**, or "Graphics Capture", described above.

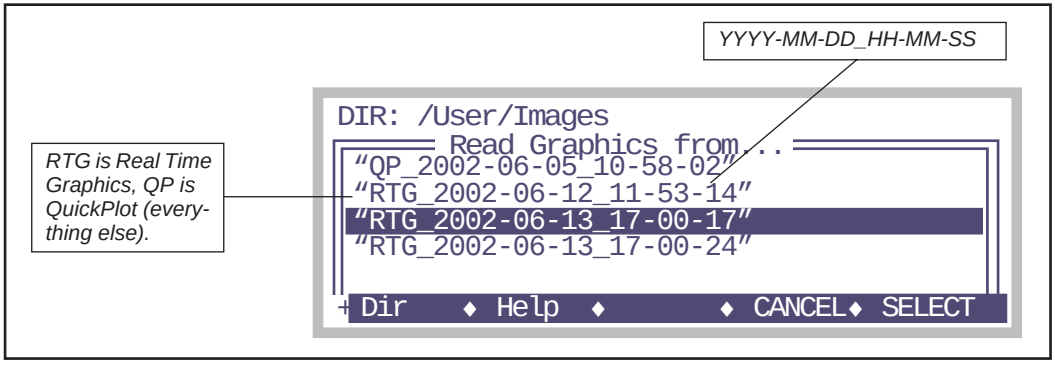
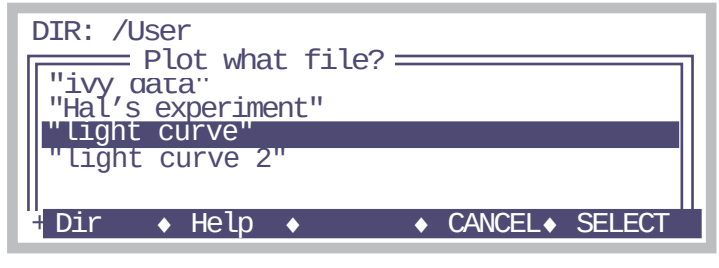


Figure 21-8. Graphics images stored by pressing ctrl + s while viewing a graph are stored in /User/Images with a filename that reflects the source (Real Time Graphics or something else) and the date and time.

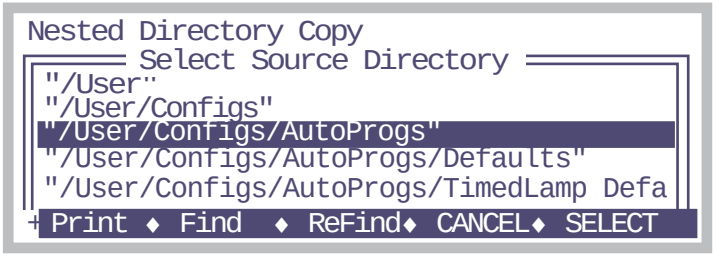
"Graphit Utility"

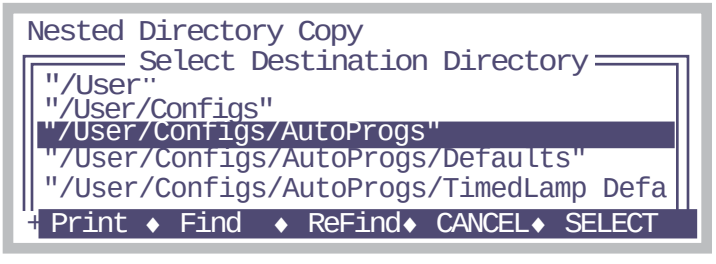
This program uses the Standard File Dialog (page 5-9) to select a file to be graphed. GraphIt (Chapter 12) is run for the selected file.



"Nested Directory Copy"

This program prompts you to select a source directory, and a destination directory. It then copies the all files and subdirectories that are in the source directory to the destination directory.

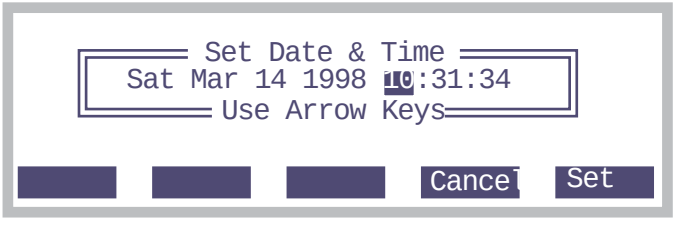




Pressing **escape** for either prompt will abort the program.

"SETCLOCK"

Sets the system time and date. 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.

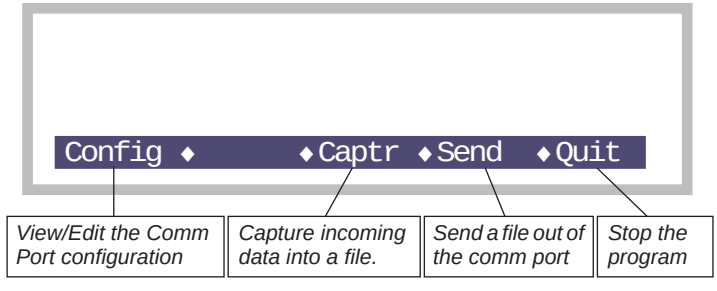


"SETCOMM"

Sets the system's comm port configuration. This program is discussed in **Setting the Communication Parameters** on page 11-11.

"Simple Terminal"

This program sends all keystrokes out of the comm port, and displays all comm port incoming data on the display.



In file capture mode, 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.

In the file send mode, the display will show the number of bytes transmitted.

"Verify Calibration"

Note: This program is here for historical reasons. You can accomplish the same thing by editing the file /dev/parm0 from the Filer. (Prior to version 5, that was not possible.)

This program allows you to edit the file that contains sensor calibration information. A typical example is shown below. Normally, the user interacts with this data by using the Installation Menu, but this program serves as a general purpose alternative. Changes made to this file do not take affect until you re-run OPEN, or change configurations (the "_Reset Menu").

```
// this is a remark
CO2Mixer= YES
CalCO2= 0.21732 2.1807E-05 1.9303E-08 -3.2863E-12
3.272E-16
CalH2O= 0.0063802 1.9083E-06 -2.4303E-11
CalZero= -4.0 -2.3
CalFlow= 0.35307
CalPress= 88.522 0.005458
CalParGaAs= 0.77 // GA-103 20 Jan 1995
CalParOut= 5.78 // Q13436 27 Aug 1990
ThisUnit= "PSC-0103"
Serviced= "9 Jan 1995"
```

Part VI

Programming

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. I		*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. Parenthesis, 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 inside a **:FCT** definition serves a slightly different purpose, and has a different syntax, than an **:INT** that appears outside.

Local Arrays

Local arrays can be declared in a couple of different ways (Figure 22-5).

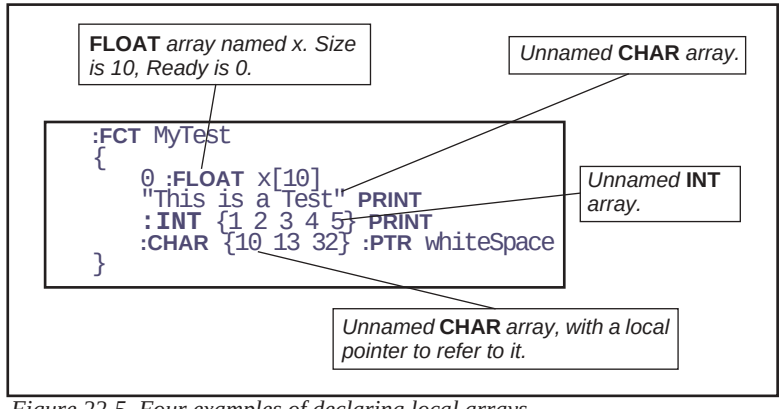


Figure 22-5. Four examples of declaring local arrays.

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 **PTR**s, 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 21-17). 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

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 ready 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** nevers 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 23-39) 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 address of the 5th element of array *x* on the stack, and sets the value 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 type 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 Two Object Transforms (e.g. **+**), or Single Object Transforms (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 values 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 target 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).

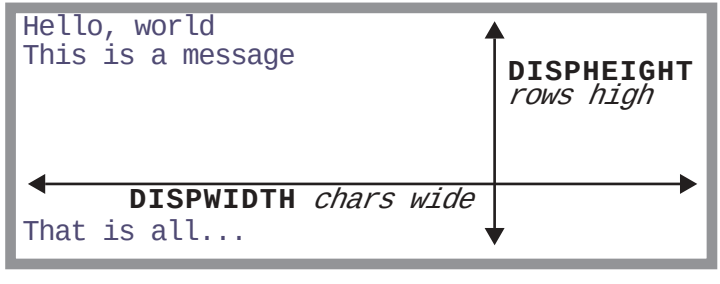


Figure 23-11. Illustration of DISPHEIGHT and DISPWIDTH.

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(relatve) (Figure 23-12).

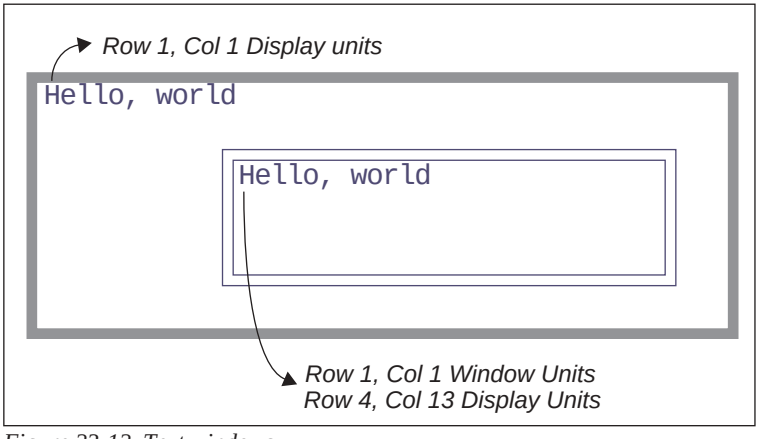


Figure 23-12. Text windows

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
}
```

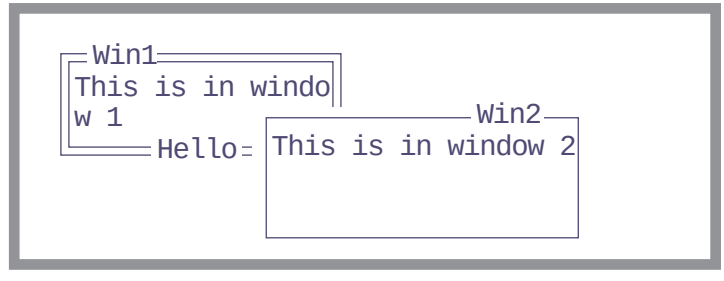


Figure 23-13. Illustration of **WINDOW** *keyword., with resulting display*

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
1 */
       w2[8] { } /* Holds attributes for
window 2 */
:PTR activeWindow 0
:FCT Main
{
    0 :INT flop
    CLEAR
    "Type something (<esc> quits)" PRINT

/* Make window 1. */
    1 3 20 7 2 WINDOW
/* Normal.*/
    0 SETATTR
/* Full height cursor.*/
    2 SETCURSOR
/* Save attributes for window  1 */
    w1 GETWINDOW

/* Set normal attrib for border. */
    0 SETATTR
/* Make window 2.*/
    22 3 40 7 2 WINDOW
/* Inverse. */
    1 SETATTR
/* No cursor.*/
    0 SETCURSOR
/* Save attributes for window 2.*/
    w2 GETWINDOW

    LOOP
        GETKEY :INT k
        k 0x1b == BREAKIF
        &flop NOT VAL IF
            w1 &activeWIndow =
        ELSE
            w2 &activeWindow =
        THEN
        activeWindow PUTWINDOW
        k "%c" PRINT
        activeWindow GETWINDOW
    ENDLOOP
    0 SETATTR
    1 SETCURSOR
    1 1 DISPWIDTH DISPHEIGHT 0 WINDOW
}
```

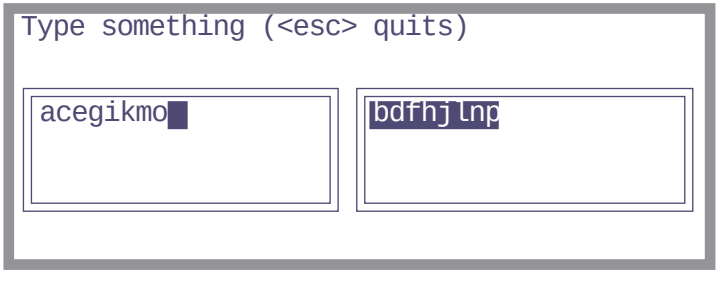


Figure 23-14. Window switching example. Every other character typed goes into the left window, and the other characters go into the right window.

If you run the program illustrated in Figure 23-14 and type “abcdefghijklmnop”, 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 **GETTEXT**/**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, **GETDISP** 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** ontrols 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 23-39. 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 key-stroke*
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	*0x2102*

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**

To illustrate, consider the AutoProgram "Remote Control", whose listing is shown in Figure 23-17.

```
/*
    Remote

    970506 - Remote control autoprogram
*/

:CHAR
      progName[] "REMOTE"
:PTR buff 0

:FCT main
{
    progName LPSetName
    progName "\f%s" PRINT

    /* Configure the comm port
    */
    "9600 8 1 n" COMMCONFIG

    /* Open a data comm queue
    */
    1000 OPEN_QUE IF RETURN THEN
    &buff =

    /* Set up incoming data capture
    */
    COMM buff XFER
    10 &GotIt ONCOMM

    LPPrep
    LOOP
        10000 LPMeasure
        lpAbort BREAKIF
    ENDLOOP

    /* Cleanup
    */
    LPCleanup
    buff CLOSE
    OFFCOMM
}

/* Gets called on receipt of line feed
*/
:CHAR line[500] ""
:FCT
GotIt
{
    line 0 SETREADY
    line "%(.)c\n" buff ENTER DROP
    line COMPILE NOT IF
        :PTR fct
        fct
        &fct FREE
    THEN
}
```

*Figure 23-17. Listing of the AutoProgram "Remote Control". The program sits endlessly in New Measurements mode (*LPMeasure*), and incoming data from the comm port is collected in a buffer (***COMM** buff **XFER**). Every time a line feed is received, however, that data is compiled and executed (**FCT** GotIt). This allows a remote device to control the LI-6400. The sequence* 10 &GotIt **ONCOMM** *sets this up.*

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 23-18 illustrates.

```
:FCT Main
{
    /*  Illustrate use of function keys
    */

    CLEAR

    0 ' ' 1 2 0 5 MAKESOFT
    /* 0 - plain text for label
delimiter
      ' ' - the delimiter is a space
       1 - inverse labels
       2 - labels 2 lines high
      0 - don't save background beneath
the labels
       5 - total of 5 function keys
allowed
    */

/* Fct Key 1 */
    "PRINT  TIME" &PrintTime 1 ONSOFT
/* Fct Key 2 */
    "PRINT  DATE" &PrintDate 2 ONSOFT
/* Fct Key 3 */
    "PRINT Ctime" &PrintCTime 3 ONSOFT
/* Fct Key 5 */
    "       quit" &Quit 5 ONSOFT
    SHOWSOFT

    IDLE

    FREESOFT
}

/* When FctKey 1 pressed */
PrintTime
{
    1 1 POSXY
    GETTDS TIME "Time is %2d:%2d:%2d"
PRINT
}

/* When Fct Key 2 pressed */
PrintDate
{
    1 2 POSXY
    GETTDS DATE "Year: %d Month:%d
Day:%d" PRINT
}

/* When Fct Key 3 pressed */
PrintCTime
{
    1 3 POSXY
    GETTDS CTIME
}
/* When Fct Key 5 pressed */
Quit { HALT }
```

Figure 23-18. 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
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 **ENTER** 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 23-19, 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 23-19. 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 24-29 on page 24-65).*
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 23-20. 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 23-21, 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 23-21. 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*
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*
FGETWP	*Get write protect status of a file or directory*
FLIST	*Get a list of files and/or directories*

Table 23-22. File System Keywords

Keyword	Description
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 23-22.

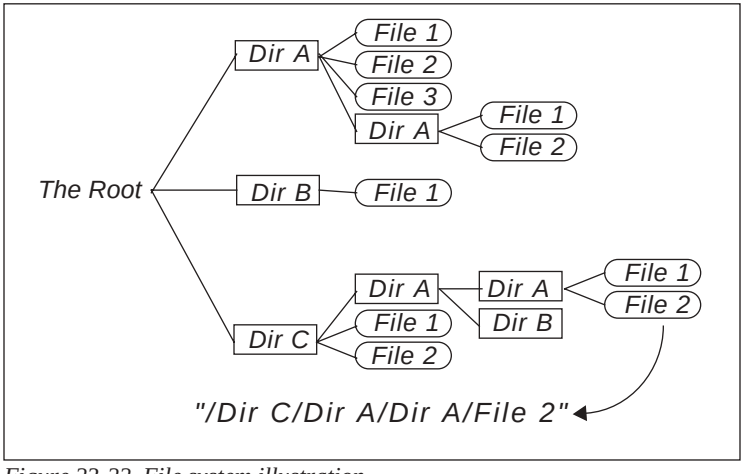


Figure 23-22. File system illustration.

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 23-39). Files can be opened with various combinations of attributes: write access, read access, and write-append. Am example program is shown in Figure 23-23.

```
: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 23-23. 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 Filer implementation only - removing a file from a running LPL program by using the **FERASE** keyword does not 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
```

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 23-23. 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 23-24 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
}
```

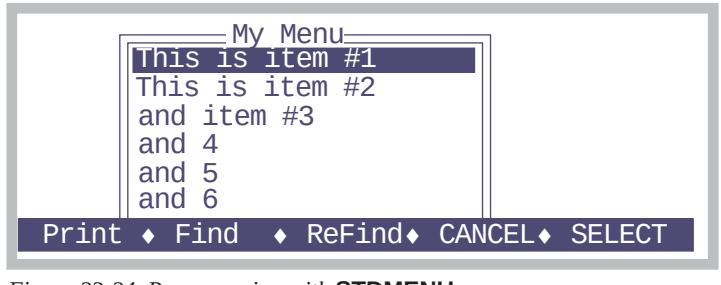


Figure 23-24. Programming with **STDMENU**.

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 5-4.

What was selected?

Something very important is missing in Figure 23-24: 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 23-25 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 23-25. 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 23-26. 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, **SETTARGET**, **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 23-27. Note the inclusion of the file "/Sys/Lib/CEDefs", which contains useful tools for these operations.

```
/**********************

    DisplayEditor (partial)

    Called from OPEN

**********************/
:IFNDEF CERefresh
    :INCLUDE "/sys/lib/CEDefs"
:ENDIF

:PTR menu 0 /* our data path */
       edPtr 0 /* custom data struct */

:FCT Main
{
/* Make a path for the menu contents */
    1000 OPEN_BUFF IF RETURN THEN
    &menu =

/* Make a window. */
    1 1 DISPWIDTH DISPHEIGHT 2 2
        "Display Editor" WINDOW

/* Fill the content path.*/
    menu DEBuildMenu

/* Create a custom menu structure.*/
    menu EDOPEN IF RETURN THEN
    &edPtr =

    &DEKeys ONKBD
    DESoftKeys
/* Turn on the menu bar.*/
    edPtr CEToggleMenuBar

    IDLE

    ResetWindow
    edPtr EDCLOSE
    menu CLOSE
}
/* Write the initial contents of the menu
to the path passed to this function. */

DEBuildMenu
{
    :PTR menu
    ...
}

DEKeys
{
    GETKEY :INT k

    k _Esc == IF DEQuitTest RETURN THEN

    k 127 < IF  /* is it ascii? */
        &k LWC DROP
/* If 'a' thru 'z', jump to that format
line.*/
        k 'a' >= k 'z' <= AND IF
            edPtr CEHomeToggle
            k 'a' - 1 + workingFmtList
READY MIN
            edPtr CEJumpToLine
            edPtr CEToggleMenuBar
        THEN
        ELSE
/* If it's non-ASCII, pass it through to
EDKEY */
            edPtr CEToggleMenuBar
            k edPtr EDKEY
            edPtr CEToggleMenuBar
        THEN
    }

    DESoftKeys
    {
        0 ' ' 1 1 0 10 MAKESOFT

/* We'll just illustrate Fct Key 1 */
        " Edit" &DEEditLine 1 ONSOFT
        ...
        SHOWSOFT
    }
```

Figure 23-27. 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 23-24.

Keyword	Description
LBNEW	*Create a list box*
LBFREE	*Destroy a list box*
LBSTART	*Pick the starting cursor line*
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*

As an example of using a ListBox, a listing of the program "/Sys/Open/LogOptionsEditor" is presented below. This is described in **Log Options** on page 9-9.

```
/*

    LogOptionsEditor

*/

:PTR options[]
{
    :FCT { _lop_Header NOT IF "Header: Embedded (normal)" ELSE
"Header: Separate .HDR file" THEN }
```

```
    :FCT { _lop_Remarks NOT IF "Remarks: Embedded (normal)"
ELSE "Remarks: Separate .REM file" THEN }
    :FCT { _lop_Stability NOT IF "Stability Details: Not
logged" ELSE "Stability Details: Logged" THEN }
    :FCT { _lop_Stats NOT IF "Statistics: None (normal)" ELSE
ShowStats THEN }
}
:PTR edit[]
{
    :FCT { &_lop_Header NOT DROP }
    :FCT { &_lop_Remarks NOT DROP }
    :FCT { &_lop_Stability NOT DROP }
    :FCT { &_lop_Stats  NOT VAL IF SetupStats THEN }
}
:INT _lop_Header 0
     _lop_Remarks 0
     _lop_Stats 0
     _lop_Period 15
     _lop_Stability 0

:FCT main
{
    lop_Header &_lop_Header =
    lop_Remarks &_lop_Remarks =
    lop_Stats &_lop_Stats =
    lop_Period &_lop_Period =
    lop_Stability &_lop_Stability =

    0 5 1 LBNEW :PTR b

    0xff01 1 0x0d "Edit" b LBDEFKEY
    0xff00 4 0x1b "Cancel" b LBDEFKEY
    0xff01 5 'D' "OK" b LBDEFKEY

    3 2 38 7 2 2 "Log Options" b LBWINDOW

    LOOP
        1 :INT i
        b LBCLEAR
        options READY NLOOP
            options i PICK CALL b LBAPPEND
            &i 1 + DROP
        ENDLOOP

        b LBEXECUTE :INT k
        k 0x1b == BREAKIF
        k 'D' == IF
            _lop_Header &lop_Header =
            _lop_Remarks &lop_Remarks =
            _lop_Stats &lop_Stats =
            _lop_Period &lop_Period =
            _lop_Stability &lop_Stability =
            BREAK
        THEN

        edit b LBSEL 1 + PICK CALL

    ENDLOOP
    b LBERASE
    b LBFREE
```

```
}

ShowStats
{
    _lop_period "%d sec Stats: -> .STATS file" to_string
}

SetupStats
{
    &StdFloatAssign
    "Enter Period"
    "%d secs"
    &_lop_period
    AskNewFloat
}
```

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 23-25. RTG keywords

Keyword	Description
RTGNEW	*Create a new 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 varaible 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 relevent 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 23-26. 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 23-26. 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 pixel. The Pixel Hardware Coordinates (PHC) has pixel (0,0) as the lower left pixel in the physical display window. In Pixel Window Coordinates (PWC), pixel (0,0) is the lower left pixel in the display window (a subset of the physical display). Finally, there are User Coordinates (UC), 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 23-28 illustrates the use of some of these graphics key words for scaling.

```
:FLOAT scale[4] {0 1.1 360 -1.1}

:FCT Main
{
/* Reset the scaling. */
    GINIT
/* Turn off text, turn on graphics,
clear the graphics display. */
    1 GRAPH 0 ALPHA
    GCLEAR


/* Get the dimensions of the graphics
display. */
    GWIDTH :INT pixelsWide
    GHEIGHT :INT pixelsHigh

    DEG
/* Scale the plotting area.*/
    scale GSCALE

/* Draw a sine curve */
    DrawACurve

/* New window in the lower left 1/3 of
the display */
    0 pixelsHigh 3 / pixelsWide 3 / 0
GWINDOW


/* Re-draw sine curve */
    DrawACurve

/* Admire the results */
    GETKEY
    1 ALPHA 0 GRAPH
}

DrawACurve
{
/* Frame the graphics window */
    1 scale GBOX

    0 :INT angle
/* Move to 0, sin(0) */
    0 0 GMOVE
/* Draw a sine curve */
    36 NLOOP
        angle DUP SIN GDRAW
        &angle 10 + DROP
    ENDLOOP
}
```

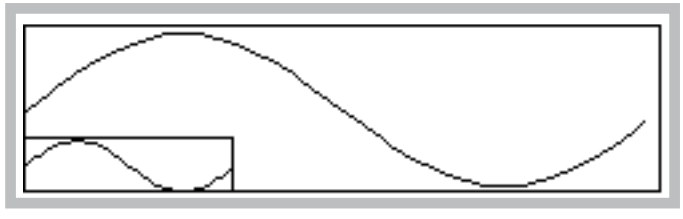


Figure 23-28. Graphics windows illustration, and the program m that generated it.

Drawing and Labelling

Single pixel access is available through **GPGET** and **GPPUT**. More useful, higher level routines are available, however. Figure 23-28 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 23-29). 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.

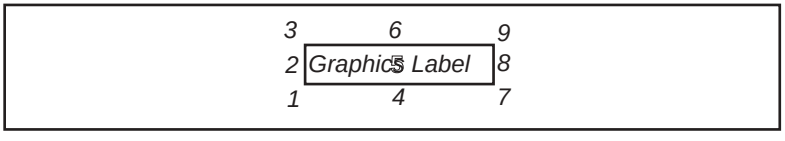


Figure 23-29. **GLORG** *locations for a graphics label relative to pen location.*

```
: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 23-30. Labelling in graphics mode.

The program in Figure 23-30 produces a labelled cosine curve (Figure 23-31).

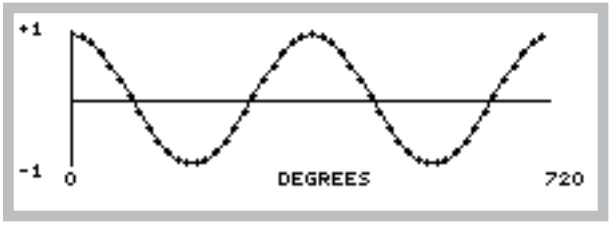


Figure 23-31. Labelled cosine curve

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 23-27.

Table 23-27. 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 23-32.

```
:INT rect1[] {0 30 40 8}
    add[] {1 0 1 0}
    rect2[] {10 50 30 10}
    mode1 0
    mode2 0
:PTR mode[] {"0 NONE" "1 OR" "2 EOR"
                "3 AND" "4 NOT"}
:FCT Main
{
    1 GRAPH 0 ALPHA
    GWIDTH 5 / :INT delta
    add delta * DROP
    GINIT GCLEAR
    1 :INT i
    5 NLOOP
        mode i PICK DoBoxes
        &mode2 1 + DROP
        &i 1 +
    ENDLOOP
    0 &mode2 = DROP
    GETKEY DROP
    0 GRAPH 1 ALPHA
}
DoBoxes
{
    :PTR label

    mode1 GMODE

/* Example 1 uses this */
    2 rect1 GBOX
/* Example 2 uses this */
/* 1 rect2 GBOX */

    mode2 GMODE

/* Example 1 uses this */
    1 rect2 GBOX
/* Example 2 uses this */
/*  2 rect1 GBOX */

    rect2 1 PICK VAL 30 GMOVE
    label GLABEL

    0 GMODE
    rect2 1 PICK VAL 4 GMOVE
    label GLABEL
    rect1 add + DROP
    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 23-32. 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 21-17.

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 23-28. 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 21-20) illustrates the use of **COMMSTATUS** and **COMMCONFIG**. Its listing is shown in Figure 23-33.

```
/*
  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 23-33. 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 23-29. 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 channel 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 group, 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 a lot 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 23-34 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 23-34. Complete A/D example

Zero and Span

Figure 23-34 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 23-30), and 8 reference channels (Table 23-31). Note that the library file "/Sys/Lib/StdAnalogIn" provides variable names for these channels, which should be used rather than the numbers.

Table 23-30. 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 23-30. 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 23-31. 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 23-32. 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 23-35 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 */
 uparrow 0x2600
 dnarrow 0x2800
 escape 0x1b

:FCT main
{
 1 0x0302 DIOSET /* flow board on */
 1 0x0005 DIOSET /* lamp on */
 &Keys ONKBD

 CLEAR "Press up and down arrows\n"
PRINT
 "(escape to quit)" PRINT
 0 ChangeLamp
 IDLE
 OFFKBD
}
Keys
{
 GETKEY :INT k
 k escape == IF HALT THEN
 k uparrow == IF
  delta ChangeLamp
 THEN
 k dnarrow == IF
  delta CHS ChangeLamp
 THEN
}

ChangeLamp
{
 :INT delta

 lamp AOVAL delta + lamp AOSET

 1 5 POSXY
 lamp AOVAL "Lamp at %4d mV" PRINT
}
```

Figure 23-35. 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 23-33.

Table 23-33. 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 23-34).

Collections of pins in a port can be dealt with in groups, using **DIOGETPORT** 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 23-34. 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/Output*
1	*Pin 22*	*Pin 4*	*Log Button*	*Flow Lo*	*Flow Hi*	*Mixer Lo*	*Mixer Hi*	*Pump status*	*Input/Output*
2					*CO2 Sample*	*CO2 Ref*	*H2O Sample*	*H2O Ref*	*Input/Output*
3					*CO2 solenoid*	*Flow board*	*IRGA board*	*Pump*	*Input/Output*
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.

Application Tools

LPL provides tools for running LPL programs from other LPL programs.

Table 23-35. 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 **:INCLUDE**. 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.

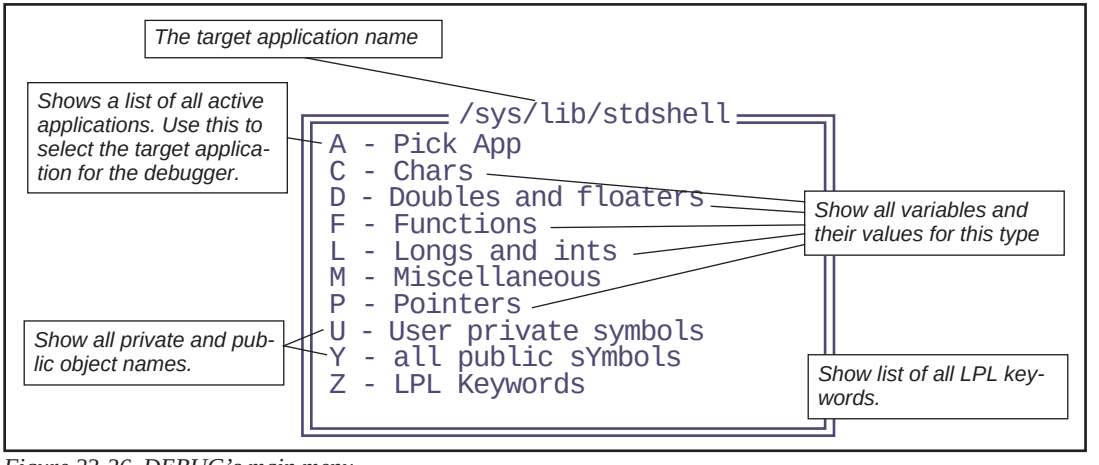


Figure 23-36. DEBUG's main menu.

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 23-37) 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.

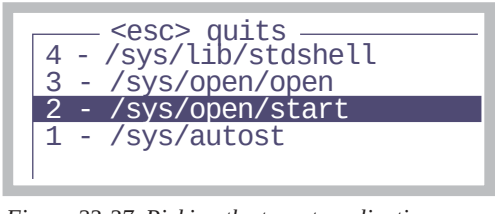


Figure 23-37. Picking the target application.

Select /sys/open/start.

3 **View the functions**
Press **F** to view the function list. Press **escape** when done.

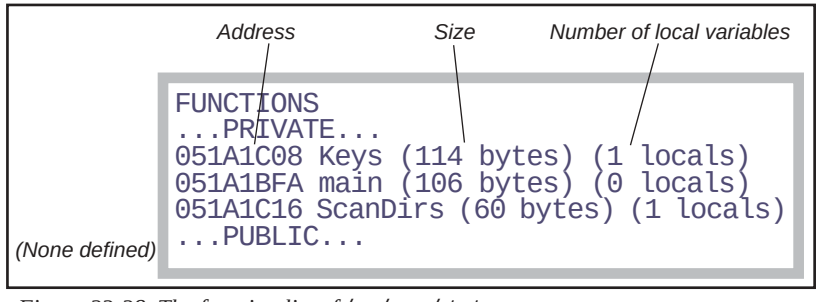


Figure 23-38. The function list of /sys/open/start.

4 **Press M to view the miscellaneous items**
The list of things in this menu is as follows:

Table 23-36. 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.

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, including **StatTracking** on page 23-88, and **Real Time Graphics** on page 23-59.

Table 23-37.

Keyword	Description
NLNEW	*Create a new namelist*
NLFREE	*Free a name list*
NLADD	*Add an item*
NLFIND	*Find an item*
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 23-38.

Keyword	Description
STNEW	*Create a new StatTracker.*
STADD	*Add a variable to the list*
STREMOVE	*Remove a variable from the list*
STSIZE	*How many variables are inthe 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*
STREAD	*Read a configuration*
STWRITE	*Write a configuration*
STGET	*Retrieve a value*
STSET	*Set a value*
STBUILD	*Create a string of values or labels*
STFIND	*Find a variable in the list*
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 23-39. 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 5-17: 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 23-40.

Keyword	Description
VREFGET	*Retrieve the reference voltages*
VREFSET	*Set the reference voltages*

See also Table 24-42 on page 87.

```
<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 23-39. 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

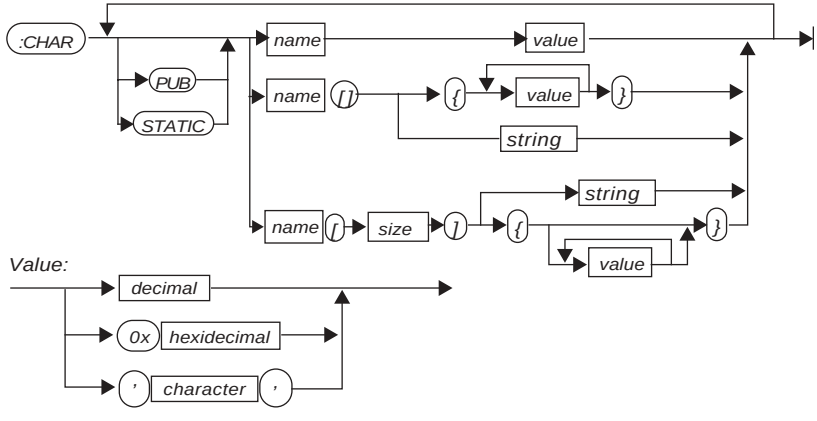
24 LPL Reference

This chapter contains a reference for LPL keywords and compiler directories.

Syntax Summaries

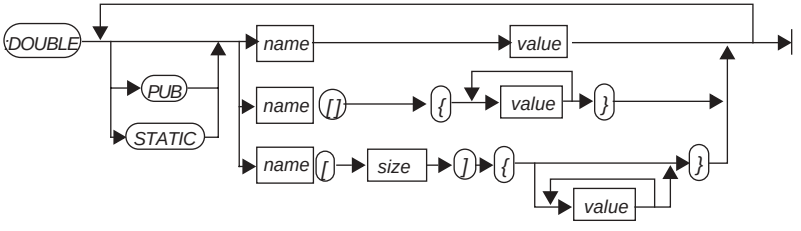
LPL Type Declarations

:CHAR **Character declaration**



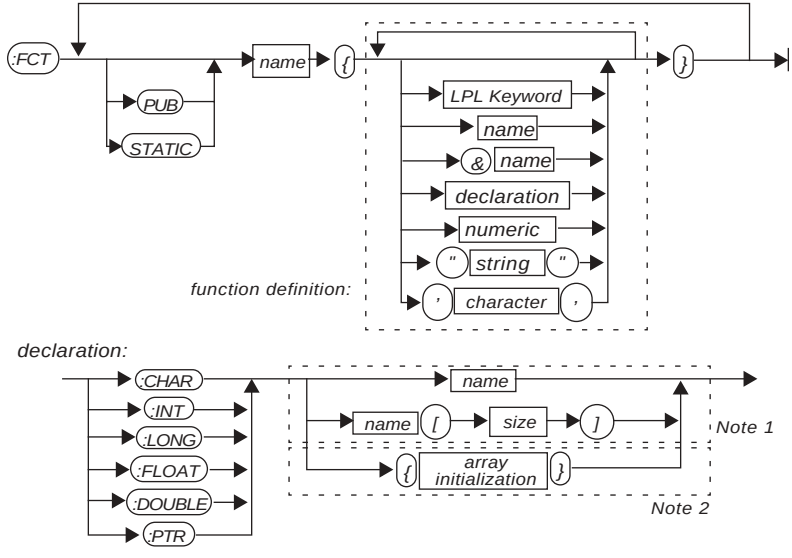
Item	Description	Range
name	*valid LPL name*	-
size	*integer*	*1…32767*
decimal	*Any combination of the chars 0…9*	-
hexidecimal	*Any combination of 0…9 plus A…F*	-
character	*Any character*	-
string	*Delimiter char is first encountered*	

:DOUBLE **High precision floating point declaration**



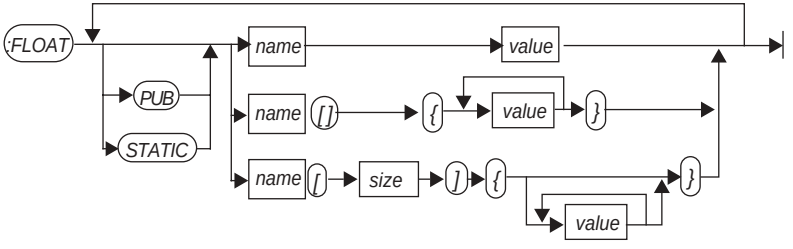
Item	Description	Range
name	*valid LPL name*	-
size	*integer*	*1...32767*
value	*integer or floating point*	-

:FCT **Function declaration**



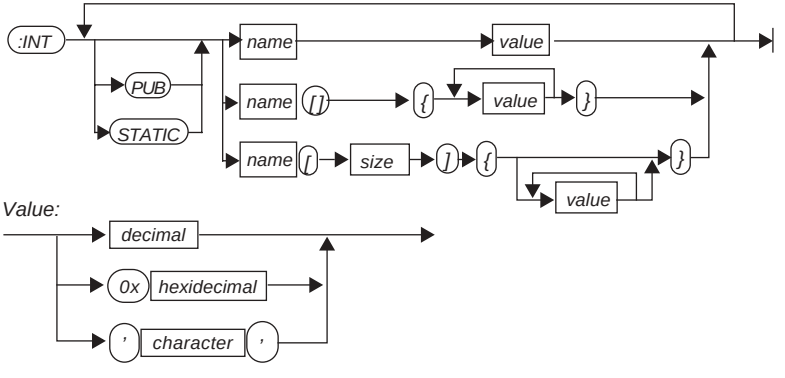
Item	Description	Range
name	*valid LPL name*	*-*
size	*integer*	*1…32767*
array initialization	*specify each array element in the un-named array*	*.*
Note 1	*This operation consumes an item from the stack.*	
Note 2	*This operation consumes nothing from the stack, and leaves the address (of the unnamed 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*	-

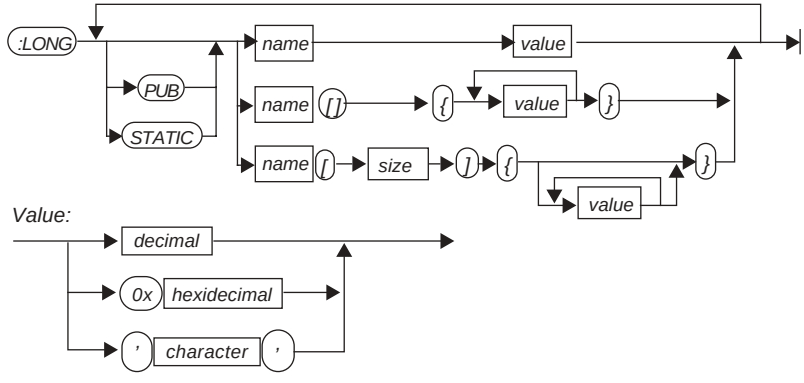
:INT **Short integer declaration**



Item	Description	Range
name	*valid LPL name*	-
size	*integer*	*1…32767*
decimal	-	-
hexidecimal	-	-

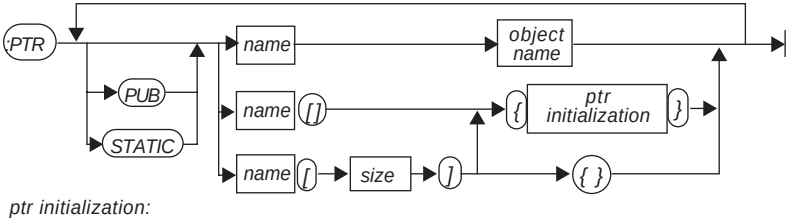
Item	Description	Range
character	-	-

:LONG **Long integer declaration**

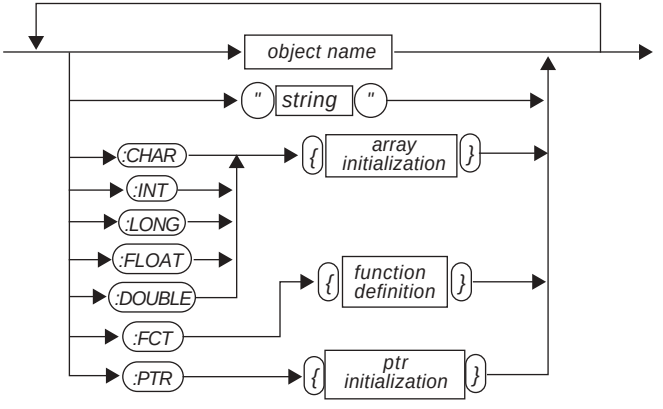


Item	Description	Range
name	*valid LPL name*	-
size	*integer*	*1…32767*
decimal	-	-
hexidecimal	-	-
character	-	-

:PTR **Pointer declaration**



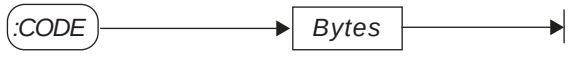
ptr initialization:



Item	Description	Range
name, object name	*valid LPL name*	-
size	*integer*	*1…32767*
array initialization	*specify each array element*	
function definition		

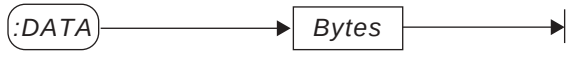
Compiler Declarations

:CODE **Declares the code space for this application**



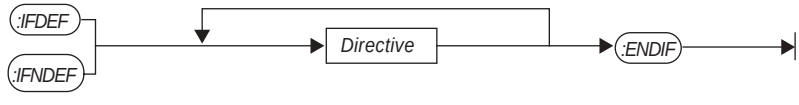
Item	Description	Range
Bytes	*Creates a code segment.*	*1 to 64000*

:DATA **Declares data space for this application**



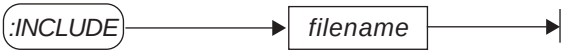
Item	Description	Range
Bytes	*Creates a data segment.*	*1 to 64000*

:IFDEF
:IFNDEF **Compiler logic**



Item	Description	Range
Directive	*LPL compiler directive or declaration*	*-*

:INCLUDE **Include a file**



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.*

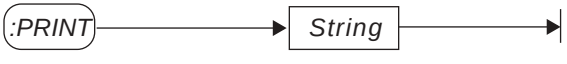
:LOCAL **Create space for local objects**

Item	Description	Range
Bytes	*The number of bytes to use for local objects. Default = 1000*	*1 to 64000*

:NAMES **Make space for variable names**

Item	Description	Range
Bytes	*The number of bytes to make a name segment. Multiple segments are allowed.*	*1 to 64000*

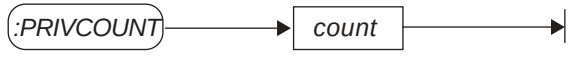
:PRINT **Print a message to the display**



Item	Description	Range
String	*The first non-whitespace character following the :PRINT is used as the string delimiter.*	*-*

:PRIVCOUNT

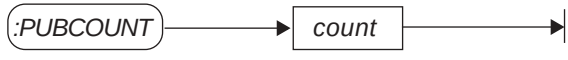
Declare number of private objects



Item	Description	Range
count	*The number of private objects that this module defines.*	*1 to 4096*

:PUBCOUNT

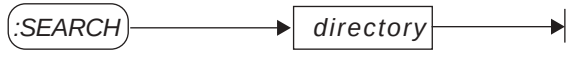
Allocate space for N public items



Item	Description	Range
count	*The number of public objects that this module defines.*	*1 to 4096*

:SEARCH

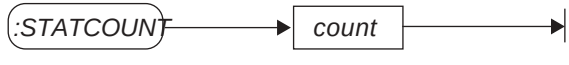
Add a directory to the list to be searched



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.*

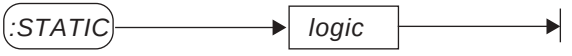
:STATCOUNT

Declare number of private objects



Item	Description	Range
count	*The number of static objects that this module defines.*	*1 to 4096*

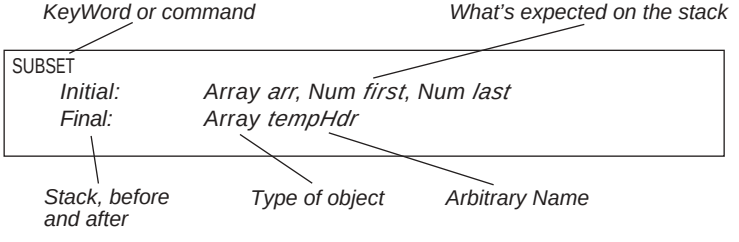
:STATIC **Enable / Disable making all subsequent declarations to be static.**



Item	Description	Range
logic	*A number. If non-zero, then it's considered true. If zero, then it's considered false.*	*integer*

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 one 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*

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.

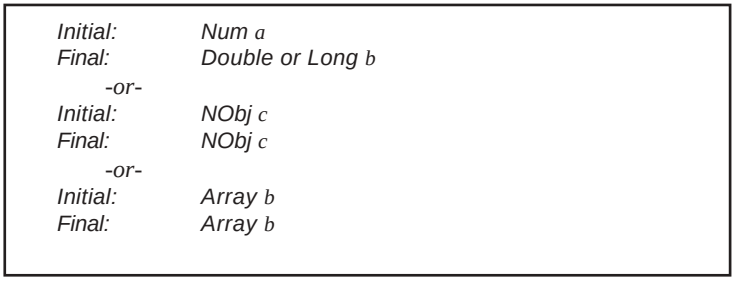


Figure 24-1. Single value transform stack requirements

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 requirements are shown in Figure 24-2

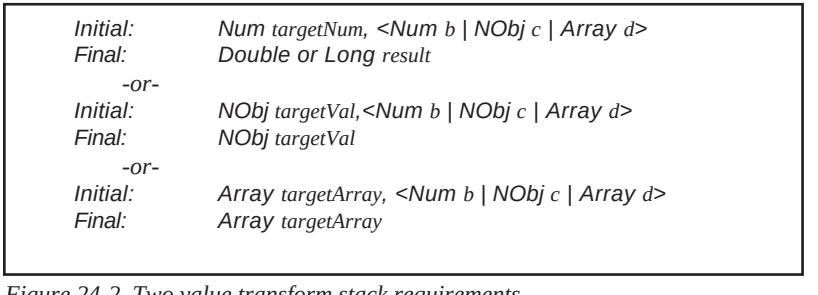


Figure 24-2. Two value transform stack requirements

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 24-4 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 24-13)*

- **Subtract values, arrays, or combinations of the two**

Initial / Final: (see **Two Value Transforms** *on page 24-13)*

***** **Multiply values, arrays, or combinations of the two**

Initial / Final: (see **Two Value Transforms** *on page 24-13)*

/ **Divide values, arrays, or combinations**

Initial / Final: (see **Two Value Transforms** *on page 24-13)*

^ **Raise to a power**

Initial / Final: (see **Two Value Transforms** *on page 24-13)*

= **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.*
`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 24-15)*

Do not use this operator to compare arrays (use `COMPARE` for that).

<> **Logical negative compare**

Initial / Final: (see **Two Value Logical Transforms** *on page 24-15)*

> **Logical greater than**
Initial / Final: (see **Two Value Logical Transforms** *on page 24-15)*

>= **Logical greater than or equals**
Initial / Final: (see **Two Value Logical Transforms** *on page 24-15)*

< **Logical less than**
Initial / Final: (see **Two Value Logical Transforms** *on page 24-15)*

<= **Logical less than or equals**
Initial / Final: (see **Two Value Logical Transforms** *on page 24-15)*

ABS **Absolute value of a value or array**
Initial / Final: (see **Single Value Transforms** *on page 24-13)*

ACOS **Inverse cosine**
Initial / Final: (see **Single Value Transforms** *on page 24-13)*

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: -
Final: *A2dPtr a, Long 0*
-or-
Final: *Long 1*

Same as AIOPEN, but returns a pointer.
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 24-15)*

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 24-13)*

Related Keywords: DEG, RAD

ATAN **Inverse tangent**

Initial / Final: (see **Single Value Transforms** *on page 24-13)*

Related Keywords: DEG, RAD

ATAN2 **Inverse tangent given two sides of a right triangle**

Initial / Final: (See **Two Value Transforms** *on page 24-13)*

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 24-15.)*

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 24-13.)*

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 24-15.)*

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 24-15. 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.

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 24-13.)*

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

For a discussion of the COMM_ keywords, see **Serial Communications** on page 23-69.

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 24-14. 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, 4:DSR)*

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 24-15. 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 24-13.)*

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 COMPILE 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*
Final: *Long error (0=ok, -1=failed)*

Note that you cannot create a directory in the root on the LI-6400, since items in the root are determined by hardware.

DIRSAVE **Force an update of the flash file system**

Initial: -
Final: -

This function is generally not required.

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 23-54.

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 24-16
Related Keywords: EDOPEN.

Table 24-16. 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

Table 24-16. (Continued)EDCTL Codes

Hex	Description	Hex	Description
0x34	*Move to start of next word*	*0x39*	*Page down*
0x40	*Jump to the beginning*	*0x43*	*Jump to end of current line*
0x41	*Jump to the start of the last line*	*0x44*	*Jump to top of window*
0x42	*Home on current line*	*0x45*	*Jump to bottom 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*
0x60	*Define/Search for target*	*0x61*	*Find next target*
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 24-17.

Table 24-17. 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 24-18. 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 24-19*
Width	*Optional. Maximum number of many characters to consume while satisfying the format directive, not counting leading whitespace.*
Type	*See Table 24-20*

Table 24-19. 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 24-20. Format Type Codes for ENTER

Type	Description
d or D	*Integer*

Table 24-20. (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 24-21 contains numerous examples.

Table 24-21. 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 24-13.)*

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 24-22. 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 available 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 system 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 24-23).

Table 24-23. 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

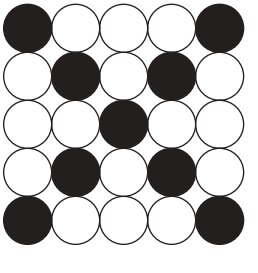
Initial: -
Final: Long *pixelsHigh*

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 24-4).



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

Figure 24-4. 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**

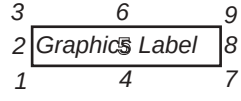
Initial: *Logic onOff (1=key defs carried over, 0=key defs don't carry)*
Final: -

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



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 not 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 26-22.

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

LBEXECUTE **Begin user interaction with a list box**

Initial: ListBox *b*
Final: Long *exitKey*

exitKey is the keycode for the key that triggered leaving the list box.

LBCLEAR **Clear the contents of the list box**

Initial: ListBox *b*
Final: -

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 24-34). Two common ones are cancel (0xff00) and accept (0xff01).

LBERASE **Blank the function key labels of a list box**

Initial: ListBox *b*
Final: -

LBFREE **Free a list box**

Initial: ListBox *b*
Final: -

Related Keywords:LBNEW

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.

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 24-13.)*

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

LOG **Natural log**

Initial / Final: (See **Single Value Transforms** *on page 24-13.)*

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 24-13.)*

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 23-37.

Table 24-24. MAKESOFT parameters

de	Delimiter enhancement Table 24-37 on page 24-77
dc	Delimiter character.
le	Label enhancement. Table 24-37 on page 24-77
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 24-25. 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 24-13.)*

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: *Num mode (=0), [Num pos1, Text lab1, [Num pos2, Text lab2]] Text messge*
Final: -
-or-
Initial: *Num mode (=1), [Num pos1, Text lab1, [Num pos2, Text lab2]] Text messge*
Final: *LONG keyCode*
-or-
Initial: *Text keys, Num 2, [Num pos1, Text lab1, [Num pos2, Text lab2]] Text messge*
Final: *LONG keyCode*

Mode 0:does not pause execution to wait for the user. The message goes up, and execution continues. There is no automatic cleanup.
Mode 1:waits for the user to press any key after the message is displayed. When this happens, the message and frame go away, and the underlying display is restored. If no frame labels are defined, “Press Any Key” will appear on the lower frame.
Mode 2:also waits for a user keystroke, except that the allowed keys are specified. **escape** is always assumed to be an allowed key.

MIN **Returns min value**

Initial / Final: (See **Two Value Transforms** *on page 24-13.)*

Related Keywords:MAX

MOD **Returns the remainder of A/B**

Initial / Final: (See **Two Value Logical Transforms** *on page 24-15.)*

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 "tuning" 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 longlab, 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
longLab - the long 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 *n*
Final: -

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 24-13.)*

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 23-39.

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 24-26. For example, “wa” is write-append.
Related Keywords: CLOSE

Table 24-26. 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 24-27. 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 24-15.)*
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 24-28, and Table 24-29.

Table 24-28. PATHSTAT Code Values

code	value
1	*Path type: See Table 24-29*
2	*Path size*
3	*Path fill*
4	*Path empty*

Table 24-29. 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 24-30. 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 24-31
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 24-32 on page 24-68

Table 24-31. 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 24-32. 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 24-32. (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 24-33. 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 24-34).

Table 24-34. PRINT Default Formats

Stack Object	Format Used	Stack Object	Format Used
LONG	*%ld*	*DOUBLE Addr*	*%lG*
DOUBLE	*%lG*	*CArray*	*%s*

Table 24-34. 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, DArray xArray, DArray 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 24-35. See **Path Filters** on page 23-43.

Table 24-35. Default File Filters

Platform	ConvertOUT	ConvertIN
LI-6400	*"e"*	*"e"*
DOS	*"e"*	*"E"*
Macintosh	*"e"*	*"n"*

Related Keywords: SETDFCIN, SETDFCOUT, GETDFCIN, GETDFCOUT

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: -

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.

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 24-36. The path pointers are changed only if *return* is not 0.

Table 24-36. 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 24-37.
Related Keywords: GETWINDOW, WINDOW

Table 24-37. 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 24-13.)*

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 24-13.)*

Related Keywords: MATHERR

STADD **Add a variable to the StatTracker list**

Initial: *Num period, Num id, STTR owner*
Final: *Long index*
-or-
Final: *Long -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.

STBUILD **Retrieve a string of labels or values for all StatTracker variables**

Initial: *Text dest, Num delim, Num code, STTR list*
Final: *-*

code specifies what you want, and are given in Table 24-38 on page 24-80.

delim is the character to be used for the delimiter.

Table 24-38. Codes for STBUILD

code	Meaning
0x40	*Mean labels*
0x41	*Min labels*
0x42	*Std Dev labels*
0x43	*Std Error labels*
0x45	*Slope labels*

STDEDIT **Invokes the system editor using the current text window**

Initial: *[Text filename], Text toEdit*
Final: *-*

If *filename* is present, pressing **escape** will bring up the exit menu.
Related Keywords: STDMENU, STDLINE, EDOPEN

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 24-39 on page 24-81) 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 24-39. 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

STEDIT **Allow the user to edit a StatTracker**

Initial: *[Text fileName], STTR list*
Final: *Long 1 (changed) or 0 (no changes)*

The optional *fileName* is the default file name for storing the StatTracker.

STFIND **Find the location of a variable in a StatTracker**

Initial: *Num id, STTR list*
Final: *Long index or -1 (not found)*

STFREE **Free a StatTracker**

Initial: *STTR list*
Final: *-*

Related Keywords:STNEW

STGET **Retrieve a value from a StatTracker**

Initial: *Num code, Num index, STTR list*
Final: *(depends on code)*

Values of code shown in Table 24-40.

Table 24-40. 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 Std Error?*	
		0x25	*Check Slope?*	
0x10	*Mean*			Double
0x11	*Minimum*			
0x12	*Maximum*			
0x13	*Std dev*	*0x33*	*Std Dev Limit*	
0x14	*Std error*	*0x34*	*Std Error 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**

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.

STREMOVE **Remove a variable from a StatTracker**

Initial: *Num index, STTR list*
Final: -

index is the 0-based index of the item to be removed. To remove all items, set *index* to -1 (or anything < 0).

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 24-41.

Table 24-41. 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**

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 24-13.)*

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 24-13.)*

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 24-42. 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*
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 24-42 on page 24-87.

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 24-5.

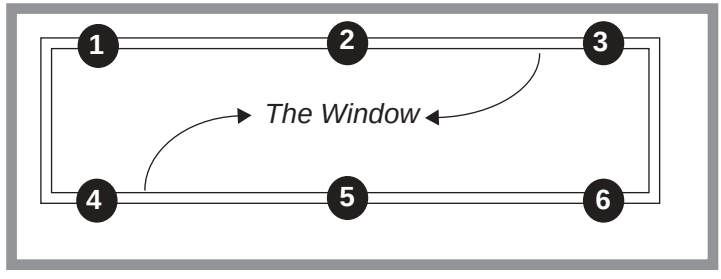


Figure 24-5. Window label position codes

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 24-43. XFER Combinations

From \ 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

XREF **Generate a cross reference for an application**

Initial: Text *filename,* Path *dest*
Final: -

AutoProgramming Reference

25

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 9-30.

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 25-13). 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

"AutoLog"

"AutoLog" is described on page 9-24.

```
/*
      Auto logging

      970502 - Remembers last time defaults
   */

   :CHAR defaultFile[] "AutoLog dflts"
        progName[] "AUTOLOG"

   :FLOAT logInterval 5
         matchInterval 0
         totalTime 10

   :INT quitAfter 0
      added 0

   :LONG t0 0
         m0 0

   :PTR user[]
   {
      :PTR { logInterval "\nLog every __ secs: " }
      :PTR { totalTime "Run for ___ minutes: " }
      :PTR { matchInterval "Match every ___ minutes
   (0=none): "}
   }

   :FCT main
   {
      progName LPSetName

      progName "\f%s" PRINT

      defaultFile user LPPrompts2 IF RETURN THEN
```

This array holds the list of things to be prompted for, with a call to LPPrompts2, below.

```
      totalTime 60 * logInterval / &quitAfter =
      &matchInterval 60 * DROP

      LPPrep
      GETMS DUP &t0 = &m0 =

      &ReadingHook &HookPostComps =
      quitAfter 0 LPSetProgress

      totalTime 60 * 1 + LPMeasure

      &Noop &HookPostComps =
   }

   ReadingHook
   {
      GETMS :LONG now
      now t0 - 1000.0 / logInterval >= IF
         LogOneObsNoComp
         &added 1 + DROP
         quitAfter added LPSetProgress
         now &t0 =
      THEN

      matchInterval IF
         now m0 - 1000.0 / matchInterval >= IF
            LPMatch
            now &m0 =
         THEN
      THEN
   }
```

ReadingHook will be called each time new readings are available.

Undo the hook.

If it's time to log again.

If it's time to match again.

"LightCurve"

```
/*
 Simple Light Curve, with stability checking

 960822 - Tech Note 14 modification, wait time defaults
 970502 - remembers last time defaults
 980309 - delta-based matching
 020524 - version 5 stability
*/

:CHAR defaultFile[] "LightCurve Dflts"
:FLOAT lampVals[20] { 2000 1500 1000 500 200 100 50 20 0 }
  stable 1.0
  matchIf 20
```

"Light Curve" is described on page 9-25.

These default values are used if there is no last-time-defaults file found.

```
:INT  minWaitTime 120
  maxWaitTime 200
  pmWait 10

:PTR user[] {
   :PTR { lampVals "Desired lamp settings (µmol/m2/s)\n"}
   :PTR { minWaitTime "Minimum wait time (secs) "}
   :PTR { maxWaitTime "Maximum wait time (secs) "}
   :PTR { matchIf "Match if |ΔFCO2| less than (ppm) " }
   }
:FCT main
{
 /* Set program name
 */
 "Light Curve" DUP LPSetName
 "\f%s\n" PRINT
```

Prompt for user input.

```
 defaultFile user LPPrompts2 IF RETURN THEN

 LPPrep /* Must be done */

 /* Loop over the lamp values
 */
 1 :INT i
```

Loop over lamp values.

```
 lampVals READY LPRegLoop NLOOP LPLoopStat

  lampVals i PICK VAL LPSetLamp /* set lamp */
```

Wait for min time, then for stability.

```
  stable minWaitTime maxWaitTime LPMeasureTilStable
   lpAbort BREAKIF
```

Match (fct is below).
Log.

```
  MatchFct
  LPLog    /* log the data */
  &i 1 + DROP   /* increment */
 ENDLOOP LPDeregLoop

 LPCleanup /* must be done */
}

MatchFct
{
 co2_diff_um ABS matchIf < IF
  LPMatch
  10 LPMeasure
 THEN
}
```

"A-CiCurve"

```
/*
```

This program is described on page 9-23.

```
A-Ci Curve
rev 2 3/16/95

960822 - Tech Note 14 modification
970502 - remembers last time defaults
980309 - delta-based matching
*/

:CHAR defaultFile[] "A-CiCurve Dflts"
:FLOAT co2Vals[20] {400 300 200 100 50 400 400 600}
 stable 1.0
 matchIf 20

:INT  minWaitTime 60
 maxWaitTime 300
 pmWait 10

:PTR user[]
{
 :PTR { co2Vals "\nDesired Ca values (Êmol/mol)\n" }
 :PTR { minWaitTime "Minimum wait time (secs) " }
 :PTR { maxWaitTime "Maximum wait time (secs) " }
 :PTR { matchIf "Match if |\x7FCO2| less than (ppm) " }
}

:FCT main
{
 0 :FLOAT waitTime
 LPIsTargetB :INT target

 /* Get rid of the non-serious....
 */
```

If no mixer is installed, or if it is not presently on, kick them out.

```
 mixerAvail NOT IF
  1 "Sorry.  Need a CO2 Mixer for this." MESSBOX DROP
  RETURN
 THEN
 IsCO2MixerOn NOT IF
  1 "Turn on the CO2 Mixer and get it
stabilized before running this." MESSBOX DROP RETURN
 THEN

 "A-Ci Curve" DUP LPSetName
 "\f%s\n" PRINT

 MixerGetTarget SWAP DROP :INT theTarget

 defaultFile user LPPrompts2 IF RETURN THEN

 1 :INT i
 LPPrep
 target LPMixerTarget

 co2Vals READY LPRegLoop NLOOP LPLoopStat
```

If asking for a zero value, turn the mixer off.

```
  co2Vals i PICK VAL DUP IF
   theTarget
  ELSE
   0 1
  THEN
  MixerSetNewTarget

  stable minWaitTime maxWaitTime LPMeasureTilStable
   lpAbort BREAKIF

  MatchFct
  LPLog
  &i 1 + DROP
 ENDLOOP LPDeregLoop

 LPCleanup
}

MatchFct
{
 co2_diff_um ABS matchIf < IF
  LPMatch
  10 LPMeasure
 THEN
}
```

“Timed Lamp”

This program is described on page 9-27. (This is version 3.2 shown here.)

```
/*

 Timed lamp control

 970502 - Bundled with OPEN 3.0
 980515 - Updated for 3.2

*/
```

Maximum steps is 60 / 3. To increase this, change the number 60 to something bigger.

```
:INT wndw[4] {1 2 39 7}
 values[60] { }
 changed 0

:LONG lastTime 0
  logMS 0

:CHAR defaults[40] “Dflts”
   defaultDir[80] “AutoProgs/TimedLamp Defaults/”
   fullSpec[80] ””

   myName[] “TimedLamp”

:FCT main
{
```

```
 FLUSH CLEAR
 myName LPSetName
 15 1 POSXY  myName PRINT

 500 OPEN_BUFF IF RETURN THEN
 :PTR buff

 buff PickParams IF
  buff CLOSE
  RETURN
 THEN

 0 0 buff SETEMPTY
 values "%(*)d" buff ENTER DROP

 values READY 3 / :INT n
 n NOT IF RETURN THEN

 LPPrep
 SetComp
 SetUpdate
 1 :INT i
 &CustomTic LPTicFct
 n NLOOP
  values i PICK VAL :INT time
  values i 1 + PICK VAL LPSetLamp
  values i 2 + PICK VAL 1000 * 50 - &logMS =
  &i 3 + DROP
  n i 3 / LPSetProgress
  time LPMeasure lpAbort BREAKIF
 ENDLOOP
 buff CLOSE

 LPCleanup
 ResetComp
 ResetUpdate
}
```

The array values is loaded up with the numbers in the buffer. The number of items in the array divided by 3 is the number of loops. Each loop has three values: time, lamp value, and log interval.

User types in parameters using Standard Edit.

```
FromKeyboard
{
 :PTR buff
 0 :INT rtn

 wndw 1 1 "Time(s) Lamp(Êmol) LogInt(s)" 5 "Enter values, press
<esc>" WINDOW
 LOOP
  buff STDEDIT
  "\x1bSRE" 2 2 "Options" 6 "Press A Key"
  "<esc> - abort\nS - store this, then run\nR - run\nE - resume
editing\n"
  MESSBOX UPC :INT key
  _Esc key == IF
   1 &rtn = BREAK
  THEN
```

```
  'S' key == IF
   buff StoreDefaults
   BREAK
  THEN
  'R' key == BREAKIF
 ENDLOOP
 1 1 DISPWIDTH DISPHEIGHT 0 WINDOW CLEAR
 rtn RETURN
}
```

Parameters come from a file.

```
FromFile
{
 :PTR buff

 0 fullSpec "Select TimedLamp Specs" OPEN_FILE_ASK IF 1 RETURN
THEN
 :PTR f
 buff RESET
 f buff XFER
 f CLOSE

 defaults 0 SETREADY
 defaultDir 0 SETREADY
 defaultDir defaults fullSpec FPARSE

 "YN" 2 2 defaults 6 "(Y/N)" "Edit these parameters?" MESSBOX 'Y'
== IF
  buff FromKeyboard RETURN
 THEN

 0 RETURN
}

/* Returns 0 for abort, 1 for ok
*/
```

User selects how to enter input parameters.

```
PickParams
{
 :PTR buff
 buff RESET

 CheckDir
 "KF" 2 2 "Timed Lamp Program" "\nGet data...\n  K - from
Keyboard\n  F - from File"
 MESSBOX :INT k
 k 0x1b == IF 1 RETURN THEN
 k 'K' == IF
  buff FromKeyboard
  RETURN
 THEN
 k 'F' == IF
  buff FromFile
  RETURN
 THEN
}
```

Does the default directory exist?

```
CheckDir
{
 theConfigDir fullSpec =
 "/" fullSpec APP
 defaultDir fullSpec APP

 fullSpec FTYPE 0 == IF
  fullSpec DIRMAKE DROP
 THEN

 defaults fullSpec APP
}
```

This routine runs every second, while the AutoProgram runs.

```
CustomTic
{
 GETMS DUP :LONG now
 lastTime - logMS > IF
  LPLog
  now &lastTime =
 THEN
 LogProgEverySec
}

ReadDefaults
{
 :PTR dest

 fullSpec FTYPE 1 == IF
  "r" fullSpec OPEN_FILE NOT IF
   :PTR file
   file dest XFER
   file CLOSE
  THEN
 THEN
}

StoreDefaults
{
 :PTR source

 0 0 source SETEMPTY
 1 fullSpec "Store TimedLamp parameters" OPEN_FILE_ASK NOT IF
  :PTR dest
  source dest XFER
  dest CLOSE
 THEN
}
```

compEvery is described in Table 26-3 on page 26-5.
hiResUpdateTime is described in Table 26-2 on page 26-4.

```
:FCT SetComp { 0 &compEvery = }
 ResetComp { 2 &compEvery = }
:FCT SetUpdate { 1 &hiResUpdateTime = A2dProgram 1 LPMeasure}
 ResetUpdate { .75 &hiresupdatetime = A2dProgram }
```

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 24-11.

Standard Commands

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 another 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 fo 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 LPPrompts, 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.

LED Source Control

These functions expect the 6400-02 or -02B LED Source, or 6400-40 LCF to be installed. Further commands specific to the LCF are in

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 typeCode*
Final: -

The units of *value* depend on *typeCode* (Table 25-1).For the 6400-40, which has two

Table 25-1. Lamp control typeCodes (for -02 or -02B)

typeCode	6400-02 or -02B	6400-40 LCF
1	Lamp off	Lamp off
2	Constant PAR (µmol m^{-2} s^{-1})	PAR, with proportional blue
3	Constant control signal (mV)	PAR, with fixed blue
4	Tracks external quantum	Constant control signal (mV)
5		Tracks external quantum.

targets for most of the control options, *value* corresponds to the first target. To set the second, one of the following variables must be explicitly changed: *blueTarget_um*, *blueTarget_mV*, and *blueTarget_pct*.

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 25-1.

LampSetTargetVal **Sets the current target value without changing control type.**

Initial: *Num value*
Final: -

Control type remains the same. Only the target changes.

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)

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.

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 25-3).

Table 25-3. 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 25-3.

FlowSetTargetVal **Set new target value independent of control mode**

Initial: *Num value*
Final: -

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 25-4).

Table 25-4. 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 25-4).

CoolSetTargetVal **Set the cooling target (independent of control mode)**

Initial: *value>*

Final: -

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).

Leaf Chamber Fluorometer Control

These functions pertain to the 6400-40 LCF. They require that the file "/sys/lib/FlrAP" be included in the autoprogram.

LPFlrSetup **Prompts for standard fluorescence prompts**

Initial: -

Final: *Long Logic (0=ok, 1=user pressed escape)*

See **Standard Fluorescence Prompts** on page 27-56 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)*

LPFlrCleamup **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 — **Turns on the flow control board.**

Initial: -
Final: -

This board must be on to control any of the analog or digital inputs or outputs. Uses digital output *dio_flowbd* (0x0302).

AbdOff — **Turns off the flow control board.**

Initial: -
Final: -

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 25-20.

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 25-17. For higher level fluorescence functions, see **Programming Commands** on page 27-73.

Actinic_on **Turns actinic on**

Initial: -

Final: -

Uses the current actinic definition and targets. See **LampSetNewTarget** on page 25-17.

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 returnd 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: -

Unsets digital output *dio_match* (0x0007).

IrgaMatchOff **Turn off the IRGA matching valve.**

Initial: -

Final: -

Sets 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 the last chapter in the manual implies that the things that have come before, especially involving programming in LPL, should not be totally foreign to you.

Using PATCH=

A simple method of modifying OPEN is through the Configuration Editor's Patch= command. This method is best suited for modifying a system variable that you otherwise do not have access to. The section **New Style ComputeList Files** on page 26-13 lists a number of these variables and what they control.

How to add a Patch= command to a configuration file

1 **Access the Config Editor**
In the Config Menu, select `"Config Status"`, then press **Edit** (**f1**).

2 **Add a dummy line, or select a blank line**
Put the cursor where you want the new line, press **Add**, and select the `"AREA= 6"` entry.

3 **Make the line a comment**
Highlight the new line, an press **labels** then **Disable** (**f1**).

4 **Edit the commented line**
Press **labels**, then **Edit** (**f1**). Press **DelLn** (**f1**), and type

```
Patch=
```

followed by a space, and the command(s) you wish to do. For example, if you wish to set the digital output on pin 7 (0x0405) high, type

```
Patch= 1 0x0405 DIOSET
```

Press **enter** to end the editing.

5 Remove the comments

Press **labels**, then **Enable** (**f2**), and the comment (//) will be removed from the front of your patch command.

Table 26-1 lists a number of examples of patch command uses.

Table 26-1. Samples of Patch= commands, and what they do.

Patch=	Description
`3500 3 AOSET`[a]	*Sets analog output 3 (pin 12) to 3500 mV*
`0 &logBeepTime =`	*Turn off log beeping*
`200 &logBeepTime =`	*Make log beeps 0.2 seconds long*
`1 0x0401 DIOSET`	*Set digital output (pin 5) low.*
`1 &compEvery =`	*User values computed (and display updated) every other time new readings are available.*
`"115200 8 2 N" COMMCONFIG`	*Sets comm port configuration*

a.The entire configuration line would be `Patch= 3500 3 AOSET`.

There are limits to what you can do with this method, however. You can only change a variable that is already defined (and PUB), and it only happens when the configuration file is actually processed. The section **Open's Hooks** on page 26-7 describes more powerful method for modifying OPEN.

Useful Variables

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*	.75	Update time. New readings available every __ seconds. (Low and Hi resolution channels both use this period.)
hiResCount	*INT*	16	Number of raw readings per update time.
lowResCount	*INT*	4	
hiResAvgTime	*FLOAT*	2[b]	Running average periods (secs).
lowResAvgTime	*FLOAT*	1	

a.Don't change it if it's shaded!
b.Also set by the AvgTime= configuration command.

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.*
compEvery	*INT*	2	*Determines how often user computations are done.* *0 = every time new measurements are available,* *1 = every other time* *2 = every 3rd time, etc.*
matchStableLimit	*FLOAT*	2	*CO_2 range limit (ppm) for determining "stability" when auto-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 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 extended control to users via a series of hooks. What is a hook? Suppose that you want the LI-6400 to print a CO_2 value to the RS-232 port (or any other task) every 1 second while in New Measurements mode. Happily, there are some things that OPEN has to do every second, and so there is a function that is called with that frequency. Happier still, you can substitute your own function for it. This ability to substitute at strategic points is what hooks provide.

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 (* = New in version 5.2)

Pointer Name	When Called	Normally Points to	Looks for Name
*HookConfigInit *	When a configuration changes, just after the comnpute list is compiled.	:FCT Nothing { }	UserConfigInit
HookMainDisplay	Called in Open's main screen to put the top three lines on the display.	StdMainDisplay	UserMainDisplay
HookCalibMenu	Called from Open's main screen in response to pressing the Calib Menu FCT key.	StdCalibMenu	UserCalibMenu
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
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.	:FCT CompPrep	UserPreComps
HookPostComps	Called just after the User Computations.	: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 falue 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 *UserN-*

MEnter 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 20-7*
WDC_IRGA	**"IRGAs Not Ready"** *on page 20-7*
WDC_Humidity	**"High Humidity Alert"** *on page 20-7*
WDC_Pump	**"Pump is Off"** *on page 20-8*
WDC_Fan	**"Chamber Fan is Off"** *on page 20-8*
WDC_Flow	**"Flow is Too Low"** *on page 20-8, and "Flow is low" in Match Mode (page 4-35).*
WDC_Light	**"Negative PAR! LightSource? Cal?"** *on page 20-9*

Example #1: Using New Measurement's Fct Keys

Figure 26-1 shows the module "LI-610 Fct Key", and illustrates how hooks can be used to define function keys in Open's New Measurements mode. The task at hand is to provide a function key in New Measurements mode that will set a D/A channel for controlling an LI-610 Dew Point Generator. The function key will be on level 7 (which normally has 5 blank keys).

Note: Do not power both the LI-610 and the LI-6400 from A/C power while controlling the LI-610 via LI-6400 DAC output. This configuration can create a ground loop, making the target voltage for the LI-610 unstable. If either unit is powered from battery, however, a ground loop will be prevented.

You can try this example by selecting **Config Menu** → "Installation Menu" → "Configuration Examples Menu" → "LI-610 Fct Key Control". This will install a new configuration, and if you switch to it ("Reset to User Config"), you'll see the new key in New Measurements mode, level 7.

```
/*
  Module to control an LI-610
  with fct key 1, level 7 in
  New Measurements mode.
*/

/* --------- internal stuff ----------- */
:STATIC 1
:FLOAT targetDP 20.0

/* Uses D/A port number 3
   That's pins 12 and 13 (signal and ground)
   on the 37 pin console connector.
*/
:INT   d2aPort 3

/*  The variable ukey1 is the keynumber
  for fct key 1, level 7
  Similarly, ukey2, ukey3, ukey4, and ukey5
  define the others
*/
:FCT Def610Key
{
  targetDP "Target= %g" to_string
&Change610Target ukey1 ONSOFT
}

Change610Target
{
  &StdFloatAssign "610 Target Temp (C)" "%g"
&targetDP AskNewFloat
  Def610Key
  SHOWSOFT
  SetThe610
}

SetThe610
{
  targetDP 100 * d2aPort AOSET
}

/* --------- external stuff ------------- */

:STATIC 0
:FCT
/* HOOK: Called when defining New Measurements
fct keys
*/
DefUserKeys
{
  Def610Key
}

/* HOOK: Called when new measure mode entered
*/
UserNMStartup
{
  SetThe610
  StdNewMeasureStartup
}
```

Figure 26-1. The file "/User/Configs/Modules/LI-610 Fct Key". It uses two hooks by defining the functions DefUserKeys, *and* UserNMStartUp.

Discussion

We accomplish this task by using two hooks: *DefUserKeys* is the most important, since that is how we'll get our key defined. The other hook, *UserNMStartup*, is only used to set the LI-610 to the target temperature when New Measurements mode is first entered. If we didn't do this, the LI-610 would not be set until the user presses the function key to set a target the first time (and perhaps that is desirable, but we're doing it for illustration purposes). At any rate, the two external functions at the end of the module are our hooks.

```
DefUserKeys
{
  Def610Key
}


UserNMStartup
{
  SetThe610
  StdNewMeasureStartup
}
```

Notice that the *UserNMStartup* function call our function *SetThe610*, and then calls it's normal function, *StdNewMeasureStartup* (see Table 26-5 on page 26-8). That's important, because we don't normally want *StdNewMeasureStartup* to be skipped.[1]

The function *Def610Key* uses ONSOFT to define the our key.

```
:FCT Def610Key
{
  targetDP "Target= %g" to_string &Change610Target ukey1
ONSOFT
}
```

The function *to_string* is a handy tool for building a temporary string, and is defined in /sys/open/open.utl. Since ONSOFT requires a function key number, we use *ukey1*, rather than 31 (**f1** level 7 is 31, **f2** is 33, etc.). It's safest to use the variables (defined in /sys/open/open.msr) *ukey1*, *ukey2*, *ukey3*, *ukey4*, and *ukey5* when defining New Measurements "empty" keys, since the actual key numbers may change in future releases, but the named constants will always be correct.

How a module is used

Once you have a module written, a good place to keep it is in /User/Configs/Modules. Then, anytime you want to add that functionality to a

---

[1] The soil flux configuration, on the other hand, uses this hook and purposefully skips *StdNewMeasureStartup*.

ComputeList, simply add an "include" line to the top of the ComputeList file. In our example, the ComputeList that is active for the LI-610 configuration has a line near the top of it that says

```
:include "/User/Configs/Modules/LI-610 Fct Key"
```

You could add that line to any ComputeList file, and thereby include the LI-610 function key feature.

Example #2: Periodic Actions

As a second example of using OPEN's hooks, we'll make a module that transmits some data out the comm port at regular intervals. Let's suppose we want CO_2 reference and sample, and H_2O reference and sample, and we want them output as often as we have new readings. And, this needs to work whether or not an Autoprogram is running, or we are logging, etc.

The hook we want is *UserNewReadings,* since that is called each time we have new measured values. The only other thing we need is the variable names of the four quantities (see Table 14-8 on page 14-19). The module is shown in Figure 26-2.

```
/*

    IRGA data -> comm port

*/
:PTR list[] { co2_1_um co2_2_um h2o_1_mm h2o_2_mm }
:CHAR format[] "%6.1f %6.1f %6.2f %6.2f\n"
:FCT UserNewReadings
{
    list format COMM PRINT
}
```

Figure 26-2. Module to transmit IRGA values out the comm port. Data is sent as it becomes available.

To use this, store it in the /User/Configs/Modules directory, and include it in whatever ComputeList you wish. A useful addition to this module would be a function key for turning this transmission on and off. We leave that as an exercise for the reader, as math texts are prone to say.

New Style ComputeList Files

ComputeLists (Chapter 15) are a collection of definitions of variables that the user can define. This definition file (example, Figure 26-3) is converted into a file written in LPL that can be linked to OPEN (Figure 26-4 on page 26-14).

ComputeLists can be written in either format. The “old style” (Figure 26-3) or the “new style” (Figure 26-4). The advantage of the new style is that you have direct control over everything, so the customization possibilities are endless.

To help you take advantage of this style, there is a utility for generating a new style ComputeList from an old style; it is the `“Old Style to New Style”` entry found in the `"ComputeList Menu"` entry in the Config Menu. It uses a Standard File Dialog box to prompt you to pick an old style compute list, and then generates a new style compute list from it. The new style compute list is shown to you, and you can choose to store it or not.

```
##10 "(U/S)" "flow:area ratio"
" flow_um / area_cm2 / 100.0"

##20 "Trans" "Transpiration (mol/m2/s)"
" #10 * (h2o_2_mm - h2o_1_mm) / (1000.0 -
h2o_2_mm)"

##21 "Trmmol" "Transpiration (mmol/m2/s)" " #20
* 1E3"

##23 "Cond" "Stomatal cond. (mol/m2/s)"
" StdCond(#20, Tleaf_c, condBL_mol, press_kPa,
h2o_2_mm)"

##30 "Photo" "Photosynthesis (umol/m2/s)"
" -#10 * co2_diff_um - co2_2_um * #20"

##35 "CndCO2" "Total Conductance to CO2"
" 1.0 / (1.6 / #23 + 1.37 / condBL_mol)"

##36 "Ci" "Intercellular CO2 (umol/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 "

##25 "VpdL" "Leaf VPD (es(Tleaf) - eair)"
" SatVap(Tleaf_c) - eAir_2_kPa"

##27 "VpdA" "Air VPD (es(tair) - eair)"
" satVapTair_kPa - eAir_2_kPa"
```

Figure 26-3. Listing of file /User/Configs/Comps/Default

Declaration Section

List of user variables, and attributes

Computation Section

```
/*
  Converted from

  /user/configs/comps/Default

  Mon Dec 29 1997 14:41:11
*/

:STATIC 1
:CHAR s8[] "%8s"
:DOUBLE u10 0
u20 0
u21 0
u23 0
u30 0
u35 0
u36 0
u38 0
u39 0
u25 0
u27 0
:STATIC 0

:PTR userList[]
{
   :PTR { 10 "(U/S)" s8  u10 g83 "flow:area ratio" 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 { 23 "Cond" s8  u23 g83 "Stomatal cond. (mol/m2/s)" 0 g13  }
   :PTR { 30 "Photo" s8  u30 g83 "Photosynthesis (umol/m2/s)" 0 g13  }
   :PTR { 35 "CndCO2" s8  u35 g83 "Total Conductance to CO2" 0 g13  }
   :PTR { 36 "Ci" s8  u36 g83 "Intercellular CO2 (umol/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 { 25 "VpdL" s8  u25 g83 "Leaf VPD (es(Tleaf) - eair)" 0 g13  }
   :PTR { 27 "VpdA" s8  u27 g83 "Air VPD (es(tair) - eair)" 0 g13  }
}

:FCT ComputeUserValues
{ $
  u10 =  flow_um / area_cm2 / 100.0
  u20 =  u10 * (h2o_2_mm - h2o_1_mm) / (1000.0 - h2o_2_mm)
  u21 =  u20 * 1000
  u23 =  StdCond(u20, Tleaf_c, condBL_mol, press_kPa, h2o_2_mm)
  u30 =  -u10 * co2_diff_um - co2_2_um * u20
  u35 =  1.0 / (1.6 / u23 + 1.37 / condBL_mol)
  u36 =  ((u35 - u20/2) * co2_2_um - u30) / (u35 + u20/2)
  u38 =  u36 * press_kPa * 1E-3
  u39 =  u36 / co2_2_um
  u25 =  SatVap(Tleaf_c) - eAir_2_kPa
  u27 =  satVapTair_kPa - eAir_2_kPa
}
```

Figure 26-4. The ComputeList "Default", converted to New Style.

The new style 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. (These are the items with ID numbers greater than zero that appear at the top of the selection list that you see when building displays or Log Lists, for example). Each item in the pointer array is itself a pointer that must contain 8 items (Figure 26-5), and optionally a 9th item (Table 26-7).

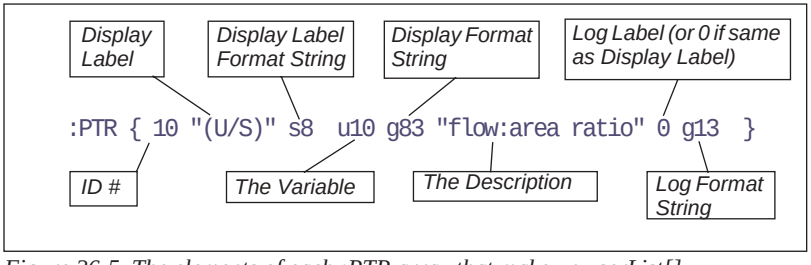


Figure 26-5. The elements of each :PTR array that make up userList[].

Table 26-7. Descriptions Of The Elements In Each *userList[]* Entry

#	Name	Description
1	*ID#*	*Any integer 1...32767*
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 26-7. (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, should this variable ever be included in a Log List. If you wish to use the display label for this, just enter a numeric value, like 0, instead of a string.*
8	*Log Format String*	*The format you wish used 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 "%1.1s".*
9	*(Optional)*	*Can be nothing, or 1, or 2:* *1. A user defined constant, that should be included automatically in the active Prompt List.* *2. A user defined constant that should not be 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 with a similar :PTR array). If you refer to Table 15-1 on page 15-5, you will see some display formats that can be specified in the old style Compute List. The actual format string used for display and logging is shown in Table 26-8 below.

Table 26-8. Relation Between Old Style Format Codes and Actual Format Strings[a]

Old Style Format Code	Display Format Used		Logging Format Used	
	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.

Using Spare Channels

Analog Input Channels

Four spare analog input channels are available for use (Table 26-9):

Table 26-9. User define analog input channels.

LPL ID#	Pin # (37 Pin Connector)	Range (V)	Resolution (mV)
20	36	-5 to +5	0.27 (low) 0.06 (high)
21	19		
22	18		
23	37		

Table 26-10. 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	

Measurements are referenced to a ground signal that is software selectable (Table 26-10). All of the LI-6400's analog inputs are over voltage protected to ±35V, with or without power.

Connections

To wire an analog input, select a signal pin (Table 26-9) and a ground pin (Table 26-10).

Software

OPEN takes care of the details of programming for the spare analog input channels. All you have to do is add the *UserChan*= configuration command to the configuration file, and edit it so that it reflects the signal and ground channels being used by the sensor. Choose the resolution (high or low). A channel measured with high resolution is sampled at the same rate as the gas analyzers, and is subject to the *AvgTime*= configuration command. Signals measured with low resolution have a 1 minute average time.

For a general discussion of analog input programming, see **Analog Measurements** on page 23-71.

Analog Output Channels

Six digital to analog output channels are available for use (Table 26-11).

Table 26-11. Spare D/A channels.

LPL ID#	Pin # (37 Pin Connector)	Range (V)	Resolution (mV)	Current (mA)
3	12	-5 to +5	2.44	±5
9	27	0 to 5	19.5	+5 / -2
10	9			
17	29	-5 to +5	39	
18	11			
19	30			

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.) There are a number of ways to update an analog output signal, and a few examples are listed below. Which method you use depends on how often you wish to have the signal updated.

- **Once at the start of OPEN**
  This is probably most easily done by inserting a PATCH= command into the configuration file. This will execute every time that configuration is implemented. For example, suppose you need to provide 4 volts to some device as part of a particular configuration. If you are using channel 10 for this, you would add

  ```
  PATCH= 4000 10 AOSET
  ```

  to your configuration file. That will set analog output channel 10 to 4000 mV.

- **Only when the user presses a function key and enters a new value**
  This technique involves defining function keys in New Measurements mode. An example of doing this can be seen by installing the `"LI-610 Fct Key`

Control" example found in the "Configuration Examples Menu" in the "Installation Menu" of the Config Menu.

As a result of doing this installation, a module named "/User/Configs/Modules/LI-610 Fct Key" will be created (Figure 26-1 on page 26-10). You can modify this module as needed to suit your purposes, and store it under a new name. To add your function key to any configuration, add the line

```
:include "/User/Configs/Modules/<name>"
```

to the ComputeList, where <*name*> is the name of your modified module.

- **Every time user variables are computed**
  A very easy way to update an analog output channel is to do it each time user variables are computed (approximately every 2 or 3 seconds). For example, if you want to have an analog output that is proportional to photosynthetic rate, edit the ComputeList definition of photosynthesis to be

```
##30 "Photo" "Photosynthesis (umol/m2/s)"
"-#10*co2_diff_um - co2_2_um*#20
AOSET(#30 * 5000.0 / 30.0, 3)
```

  This uses D/A channel 3, and scales the signal so that 5V is 30 µmol m^{-2} s^{-1}.

- **Whenever you feel like it (no programming required)**
  A simple method of setting a D/A channel is to do the AOSET manually from the keyboard. From OPEN's main screen, press **K** to run the LPL Shell program. At the ok: prompt, type the command and press **enter**. For example,

```
ok:2450 3 AOSET
```

  will set channel 3 to 2450 mV. Press **escape** to leave this mode.

- **In an AutoProgram**
  If the D/A is to be controlled only when a particular AutoProgram is running, then that can simplify the programming. The control can be set at the start of the AutoProgram, or, if frequent updates are necessary, the control can be updated once each second during the AutoProgram. See **New Style ComputeList Files** on page 26-13.

Digital Output

Several digital channels are available for use (Table 26-12):

Table 26-12. 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, and when it is set low, it will go to whatever voltage you have on the other side of the resistor (Figure 26-6.

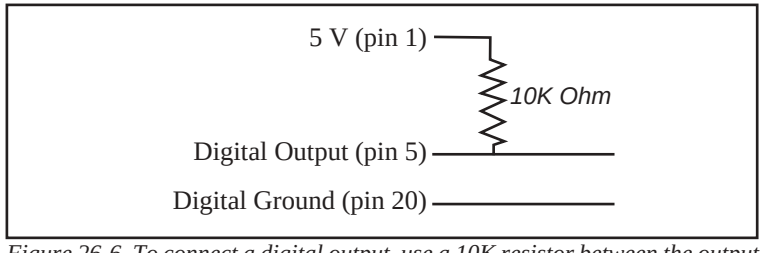


Figure 26-6. To connect a digital output, use a 10K resistor between the output pin and the 5V supply, and measure the voltage between the output pin and digital ground. When the output is set (high), you'll measure 0 volts between pins 5 and 20, and when not set (low), you'll measure 5 volts.

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).

Digital Input

Two digital input channels are available, besides the log switch. (Table 26-12):

Table 26-13. 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 26-7 illustrates a method of connecting a switch.

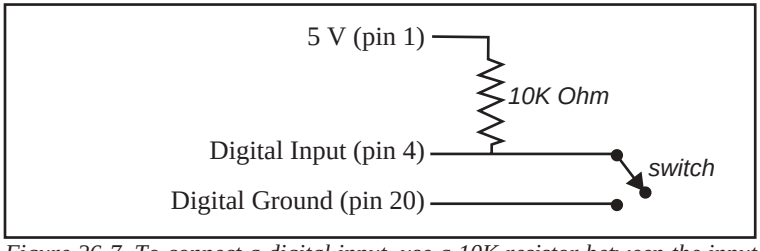


Figure 26-7. To connect a digital input, use a 10K resistor between the input pin and the 5V supply. When the switch is open, the digital input will be high, and when closed, the input will be low.

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. (For higher frequency pulse counting, see **High Speed Counter** on page 23-81.)

The question of where one does the programming arises: it depends on how often you need to check the digital input. You might want to include this in a ComputeList, or an AutoProgram, or do it every second in New Measure-

ments mode.

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 26-8 illustrates a method of connecting a switch.

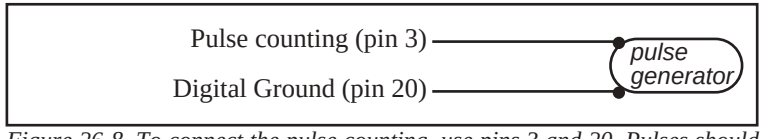


Figure 26-8. To connect the pulse counting, use pins 3 and 20. Pulses should be between 0 and (5 < x < 8) Volts.

Software

Use 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 26-14. 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 26-14. 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 26-22*
4		Digital Input 0x0106	**Digital Input** *on page 26-21*
5	F	Digital Output 0x0401	**Digital Output** *on page 26-20*
6	F	Digital Output 0x0403	
7	F	Digital Output 0x0405	
8	F	Digital Output 0x0407	
9	F	Analog Output 10 (0 to 5V, 8 bit)	**Analog Output Channels** *on page 26-18*
10		Analog Output 8 (0 to 5V, 8 bit)	
11	F,T	Analog Output 18 (-5 to +5V, 8 bit)	
12		Analog Output 3 (-5 to +5V, 12 bit)	
13	T	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 26-17*
16		Sample H2O raw analog output	
17		Analog Input Ground 7	**Analog Input Channels** *on page 26-17*
18		Analog Input 22	
19	T	Analog Input 21	
20	F	Digital Ground	Use with digital I/O
21		Digital Output 0x0007	Match mode
22	F	Digital Input 0x0107	**Digital Input** *on page 26-21*
23	F	Digital Output 0x0400	See **Digital Output** *on page 26-20*
24	F	Digital Output 0x0402	
25	F	Digital Output 0x0404	
26	F	Digital Output 0x0406	
27		Analog Output 9 (0 to 5V, 8 bit)	**Analog Output Channels** *on page 26-18*
28		Analog Output 16	Flow meter zero
29	F	Analog Output 17 (-5 to +5V, 8 bit)	**Analog Output Channels** *on page 26-18*
30	F	Analog Output 19 (-5 to +5V, 8 bit)	
31		Analog Input Ground 3	Used by leaf chamber
32	F	Analog Input Ground 5	**Analog Input Channels** *on page 26-17*
33		Sample CO2 raw analog output	
34		Reference H2O analog output	
35		Reference CO2 raw analog output	
36	F	Analog Input 20, ±5V	**Analog Input Channels** *on page 26-17*
37	F	Analog Input 23, ±5V	

a.F = Used by 6400-40 LCF. T = Used by 6400-13 T/C Adapter.

Part VII

Accessories

Leaf Chamber Fluorometer

27

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 27-18 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 27-41.

Getting Started

If you are new to fluorescence, read through **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, follow through **Installing the LCF** on page 27-10.

2 **Make sure the LCF is working properly**
Work through **Basic Functionality Test** on page 27-71.

3 **Calibrate the light source**
This is described in **"Actinic Control - Calibrate"** on page 27-61.

Also note these important sections:

- **Operational Summary on page 27-18**
A summary of the controls, displays, and utilities available when configured for the LCF.

- **Changing Control Parameters on page 27-36**
A discussion of the various LCF settings, what they mean, and what are right for you.

- **Basic Experiments on page 27-41**
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) photosynthetic photochemistry. This relationship can be expressed as:

$$F + H + P = 1 \qquad (27\text{-}1)$$

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 as 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:

$$F_m + H_m + 0 = 1 \qquad (27\text{-}2)$$

so

$$H_m = 1 - F_m \qquad (27\text{-}3)$$

If we also assume that the ratio of heat to fluorescence de-excitation does not change

$$\frac{H}{F} = \frac{H_m}{F_m} \qquad (27\text{-}4)$$

then it follows that:

$$H = \frac{F(1 - F_m)}{F_m} \qquad (27\text{-}5)$$

By making two measurements of F, one 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).

$$P = 1 - F - H$$
$$= 1 - F - \frac{F(1 - F_m)}{F_m} \qquad (27\text{-}6)$$
$$= \frac{F_m - F}{F_m}$$

If the non-saturating light condition is total darkness, and the leaf is completely adapted to that darkness, Equation (27-6) takes the form[1]

ID: 4202[2]

$$P_{dark} = \frac{F_m - F_o}{F_m} = \frac{F_v}{F_m} \qquad (27\text{-}7)$$

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 most plants, F_v / F_m is about 0.8.

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

$$P_{light} = \frac{F_m' - F_s}{F_m'} = \frac{\Delta F}{F_m'} = \Phi_{PSII} \qquad (27\text{-}8)$$

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 $\Delta F / F_m$'.

A similar relationship that is sometimes used is

ID: 4211

$$\frac{F_v'}{F_m'} = \frac{F_m' - F_o'}{F_m'} \qquad (27\text{-}9)$$

---

[1] We use the nomenclature put forth by VanKooten and Snel (1990). (References are listed in **Chlorophyll Fluorescence References** on page 27-82.)

[2] ID values refer to user defined variable numbers. See **Fluorescence ComputeList** on page 27-76.

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

$$\Phi_{CO_2} = \frac{A - A_{dark}}{I\alpha_{leaf}} \qquad (27\text{-}10)$$

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

$$ETR = \left(\frac{F_m' - F_s}{F_m'}\right) f I \alpha_{leaf} \qquad (27\text{-}11)$$

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 C4.

Photochemical quenching q_P of fluorescence can be computed from

ID: 4216

$$q_P = \frac{F_m' - F_s}{F_m' - F_o'} \qquad (27\text{-}12)$$

Photochemical quenching includes photosynthesis and photorespiration, and tends to be largest 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

$$q_N = \frac{F_m - F_m'}{F_m - F_o'} \qquad (27\text{-}13)$$

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 sometimes reported is

ID: 4221

$$NPQ = \frac{F_m - F_m'}{F_m'} \qquad (27\text{-}14)$$

Some researchers prefer to use F_o rather than F_o' in the calculation of q_p and q_N.

ID; 4231

$$q_P\big|_{F_o} = \frac{F_m' - F_s}{F_m' - F_o} \qquad (27\text{-}15)$$

ID: 4232

$$q_N\big|_{F_o} = \frac{F_m - F_m'}{F_m - F_o} \qquad (27\text{-}16)$$

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 **LogList Editor** on page 9-12.

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 27-1). Here is an outline of how it works:

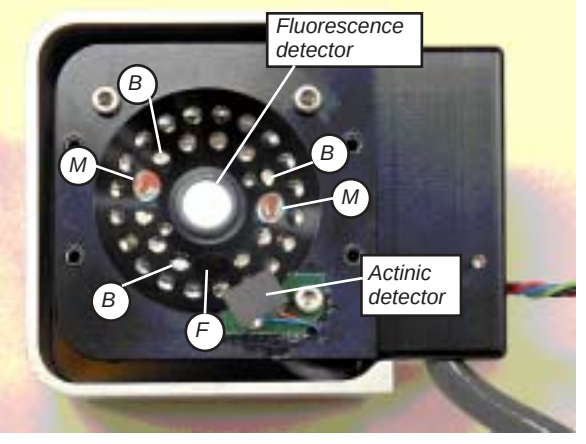


Figure 27-1. View of the LEDs of the LCF, showing the two red modulated measuring beam LEDs (M), the three blue actinic LEDs (B), and the far red LED (F). The remaining LEDs are red, and are used for actinic and saturating flashes.

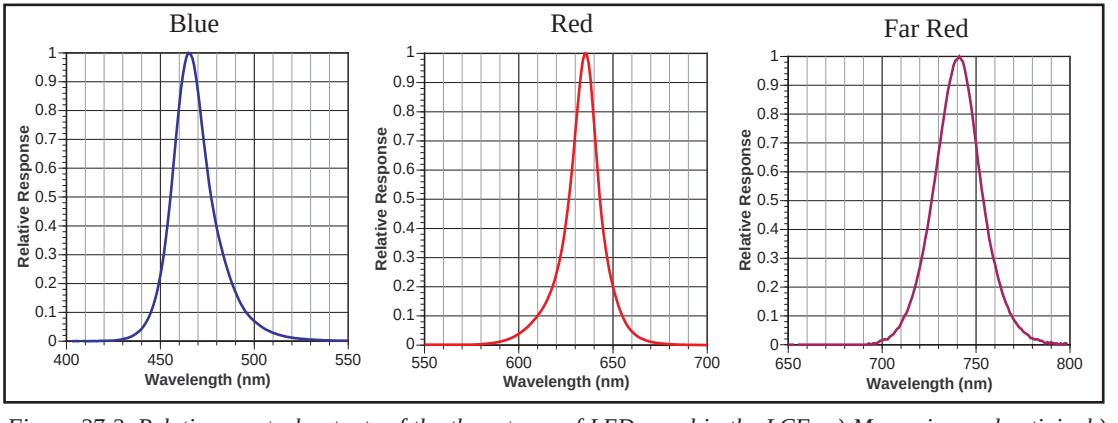


Figure 27-2. Relative spectral outputs of the three types of LEDs used in the LCF: a) Measuring and actinic, b) actinic and saturation flashes, c) far red.

Measuring Fluorescence

The LCF uses two red LEDs (center wavelength about 630 nm - see Figure 27-2 on page 27-7) 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 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 27-7 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 final fluorescence value is relatively noisy due to the limited number of points in the average. 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 to work with, and the resulting 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 - that 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 µmol m^{-2} s^{-1}) by using 27 red LEDs (center wavelength: 630 nm. See Figure 27-2 on page 27-7). PAR levels during saturating flashes are monitored 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

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 740 - see Figure 27-2c). 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 4 LEDs are illuminated. Also, just before and after a flash, the 4 status LEDs briefly flicker.

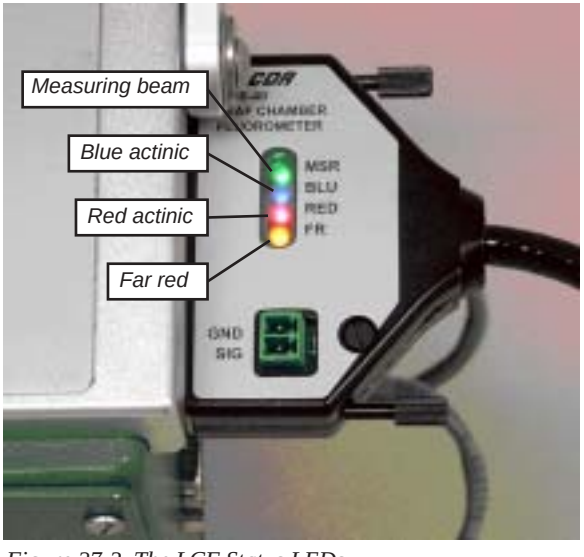


Figure 27-3. The LCF Status LEDs

Installing the LCF

The 6400-40 LCF requires one of the following:

- **Version 5 (or above) software**

-or-

- **Version 4 software AND the 6 MB memory board**
To determine which memory board is installed in a pre-version 5 LI-6400, access the Filer, press **K** to access the disk keys, then press **S**. Pick any of the disks (sys, user, etc.), and press **enter**. The display will show used and available space.

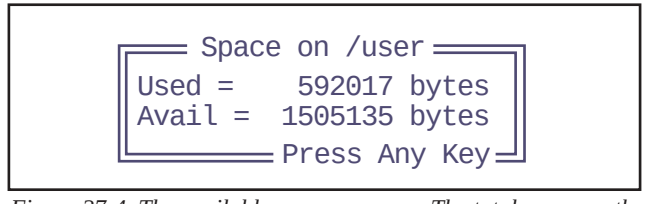


Figure 27-4. The available space message. The total space on the disk is the sum of these two numbers. When a 6MB memory board is installed, the used *plus* avail *space for one disk will sum to 2 MB.*

If *Used + Available* sums to about 2 million, as in Figure 27-4, you have the proper board for OPEN 4.0. If *Used + Avail* sums to about 524,000, you have the small memory board, and will need to upgrade. You have two choices:

1. Update to version 5. Order part number 64500-915. (This is preferable, since you'll be getting a 200 MHz processor, in addition to 128 MBytes RAM, and 64 MBytes flash, so operation is much faster.)

2. Upgrade to the larger memory board. Order part number 6400-906.).

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 27-5a). However, if you are going to be using the LCF more or less all the time, you may want to consider inserting the LCF cable into the bundle

(Figure 27-5b). This latter approach takes a few minutes of work, but is more convenient in the long run.

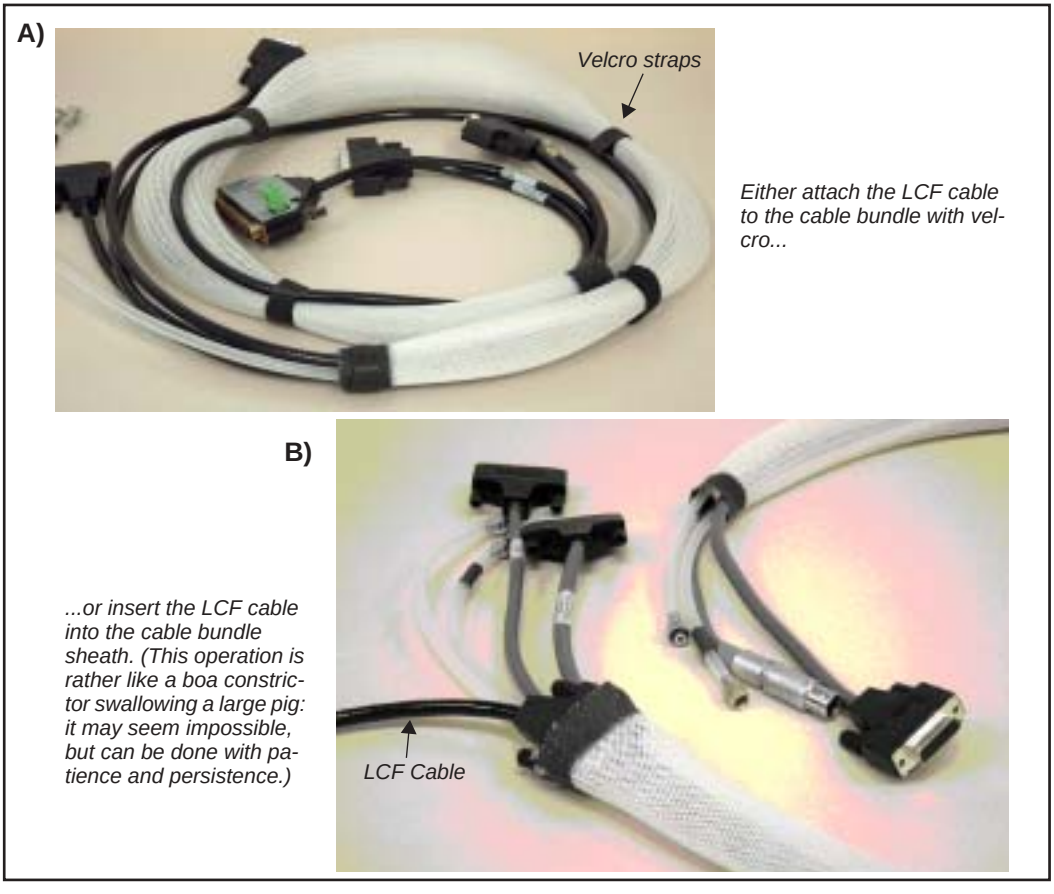


Figure 27-5. The LCF cable can be lashed to the LI-6400 cable bundle with velcro (A), or inserted into the bundle (B). If you choose to do the latter, insert the smaller connector into the console end of the bundle (the end with two 25 pin connectors) so it will come out at the sensor head end.

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 27-6).

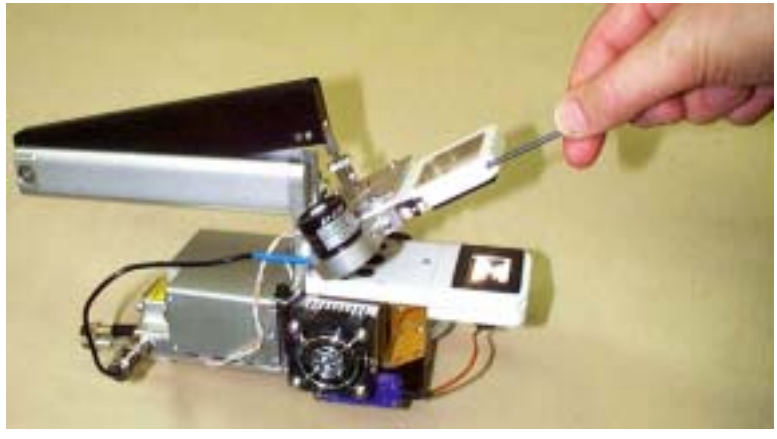


Figure 27-6. Remove the chamber top and bottom using a 3/32" hex key

3 Attach the lower part of the LCF

Make sure the 3 o-rings are in place on the LCF bottom chamber, and attach it to the sensor head. Install the leaf thermocouple (Figure 27-7)

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.

.

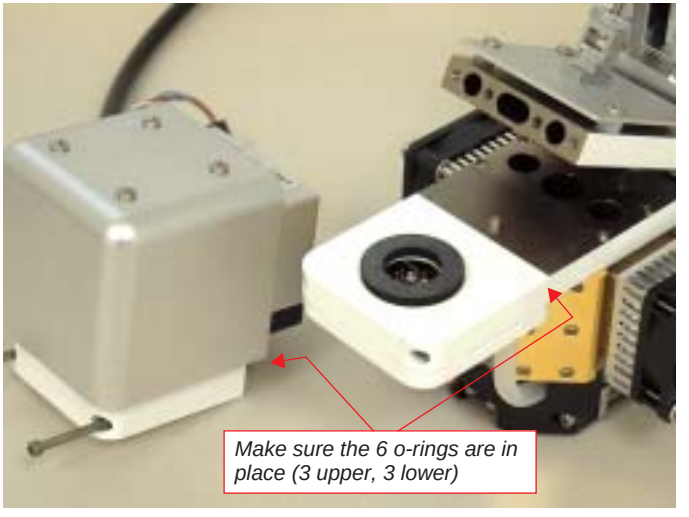


Figure 27-7. The LCF lower chamber attached, and the upper chamber ready.

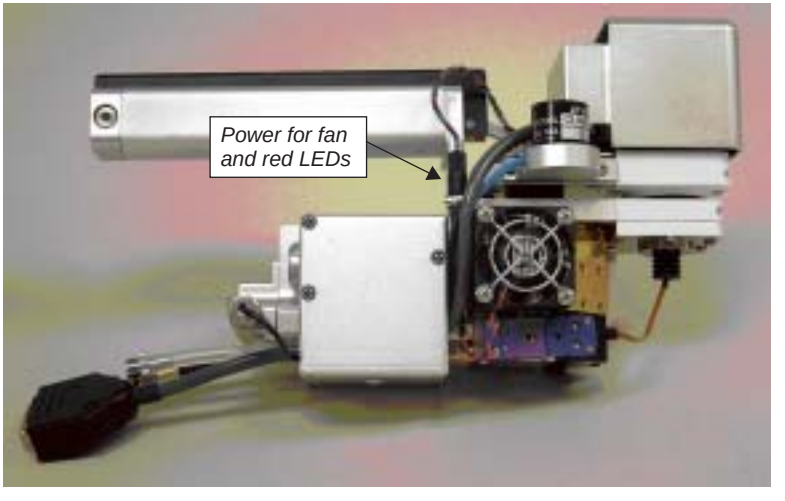


Figure 27-8. The LCF attached to the sensor head. The main cable can be routed behind the quantum sensor, and through the tripod mount (remove to do this).

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 the Installation Menu

With OPEN running (with any configuration), press **f2** (**Config Menu**) from the main screen. Then highlight "Installation Menu" and press **enter**.

2 Select "6400-40 Fluorometer"

Page down to this entry, and press **enter**. After a few seconds, the initial display will appear.

```
6400-40 Fluorometer Installer

This program will
  a) create a config file
  b) install some support files

Press <enter> to continue
```

Figure 27-9. The initial display of the 6400-40 Installation Program. Press **enter** *to continue, or* **escape** *to abort.*

Press **enter**.

3 Broadleaves or Needles?

You will then be asked to choose between broadleaves and needles (Figure 27-10). If you select broadleaves, the leaf temperature thermocouple will be used as a direct measure of leaf temperature, and the boundary layer conductance table for the LCF (**LCF Boundary Layer Conductance** on page 27-78) will be used to compute boundary layer as a function of leaf area.

If you choose needles, then boundary layer will be assumed constant, and leaf temperature will be computed from an energy balance equation. You will also need to pull the leaf temperature thermocouple down just a bit so that it measured air temperature near to the needles. The computed leaf temperature will be displayed as a variable named *EBTlf* on display line *h*. The variable named *Tleaf* next to it will be the air temp near the needles. For more details on the energy balance method, see Chapter 17, **Using an Energy Balance**.

```
Select leaf type:
  Broadleaves (measured leaf temp
    and variable BLC).
  Needles (leaf temp from enery
    balance and fixed BLC).

Pick Broadleaves or Needles (B/N)_
```

Figure 27-10. The broadleaf / needle choice selects an appropriate method for determining boundary layer conductance and leaf temperature.

4 View the configuration file

The installer program will next let you view/edit the default configuration file (Figure 27-11).

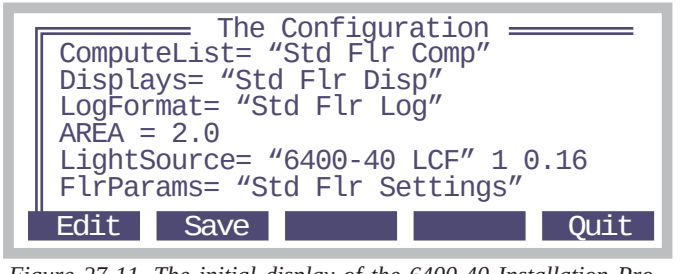


Figure 27-11. The initial display of the 6400-40 Installation Program. For an explanation of these items, see ***Configuration File Details*** *on page 27-72.*

Press **Save (f2)**.

5 Name the configuration file

You will be asked to name the configuration file. You can use the default name, or modify it as you wish. (If you choose a configuration for needles, the default configuration name will be "Fluorometer EB" rather than the "Default Fluorometer" shown in Figure 27-12).

Press **enter** or **f5** when done.

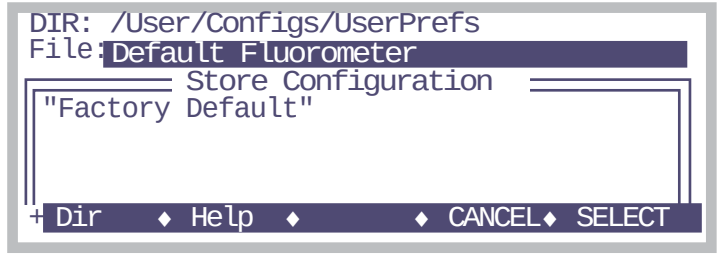


Figure 27-12. Naming and storing the configuration file uses the Standard File Dialog (described on page 5-9).

The file name chosen will be the name you see when presented with a list of configurations to choose on power up, or when resetting to a user configuration (Figure 27-13).

Once the configuration is stored, you'll then return to the LCF configuration dialog (Figure 27-11).

6 Exit the configuration

Press **Quit** (**f5**). You'll see a message telling you how to implement the configuration just created (we'll do this in the next step). Press **enter** to return to the Installation Menu, then **escape** to return to the Config Menu.

7 Select the LCF Configuration

To implement the LCF configuration, go to the Reset Menu (the last entry in the Config Menu). Then select "Reset to user config", and select the entry "Default Fluorometer" (or "Fluorometer EB", or whatever you named it) from the list of configurations (Figure 27-13).

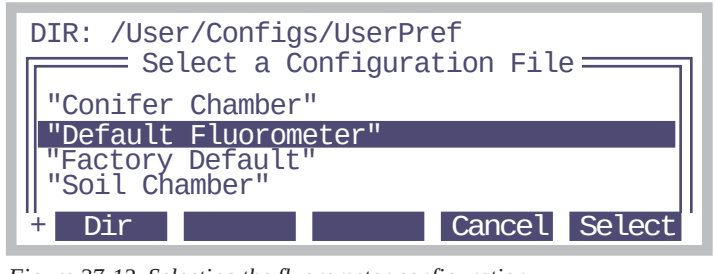


Figure 27-13. Selecting the fluorometer configuration.

To create a light source calibration file

1 **Create and Select an LCF Calibration**
What you just completed starting on page 27-14.

2 **Do the calibration**
See **Calibration Issues** on page 27-59.

Operational Summary

Figure 27-14 summarizes the changes to the New Measurements Mode displays and function keys associated with the LCF.

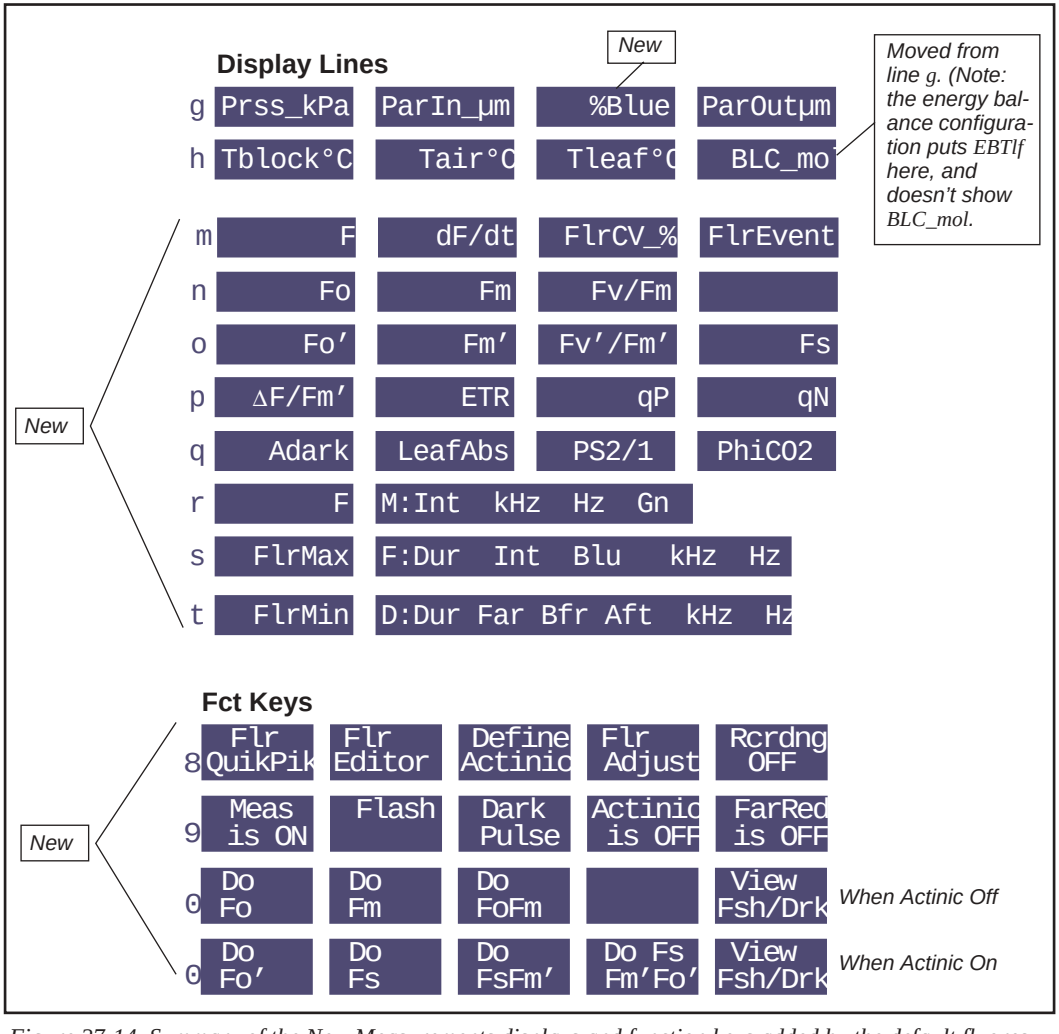


Figure 27-14. Summary of the New Measurements displays and function keys added by the default fluorescence configuration. Display explanations are in Table 27-3 on page 27-29, and the new function keys are described in Table 27-4 on page 27-32

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. In version 5 only, if you specify the 6400-40 as the light source (via **The Light Source Control Utility** 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 as a 6400-02 Red/Blue Source. 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.

The light source control panel in New Measurements (accessed by **f5** level 2) also works for the LCF. 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 standard control panel for the light source (described in **Light Control** on page 7-18) looks slightly different when the LCF is selected as the light source (Figure 27-15).

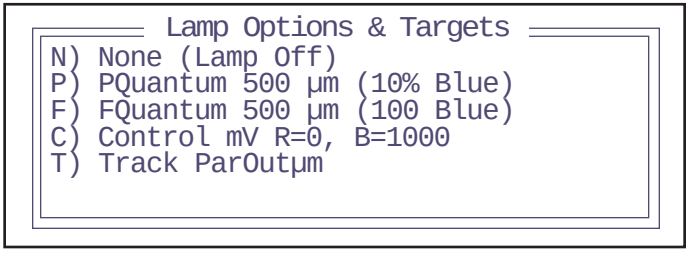


Figure 27-15. The lamp control panel (f5 level 2) for the LCF.

The 5 options presented in the control panel are described in Table 27-1.

Table 27-1. LCF actinic control options

Option	Description
N) None	Turns off red and blue actinic LEDs
P) PQuantum	Proportional quantum. Two target values: Total PAR (red + blue) in µmol m^{-2} s^{-1}, and proportion that should be blue, in%.  Keep in mind the maximum blue intensity is 150 or 200 µmol m^{-2} s^{-1}, so may not achieve the requested percentage.
F) FQuantum	Fixed quantum. Two target values: Total (red + blue) µmol m^{-2} s^{-1}, and blue µmol m^{-2} s^{-1}. 
C) Control signal	Two target values, red (mV) and blue (mV) 
T) Track ParOutµm	Tracks external quantum sensor (or other channel). Every 3 seconds, a new target is determined. You are prompted for the %Blue faction to use, and which channel to track.

Three of the control options (P, F, and C) have two target values. One can enter both of these values, separated by a space or comma, to set both targets. If only one number 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 27-2).

Table 27-2. Entering 2 value targets

Entry	Total (P and F), or Red (C)	Blue Target
500, 50	500	50
800 100	800	100

Table 27-2. Entering 2 value targets

Entry	Total (P and F), or Red (C)	Blue Target
1000	1000	unchanged
,80	unchanged	80

The Tracking option (T) gets the target value 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 and tracks that variable. Tracking mode sets the light source in proportional blue mode, and the user specifies the desired blue fraction.

Measuring Fluorescence

When the fluorescence measuring beam is turned on (**f1** level 9), the raw fluorescence signal can be viewed on display line *m*, under the label *F*.

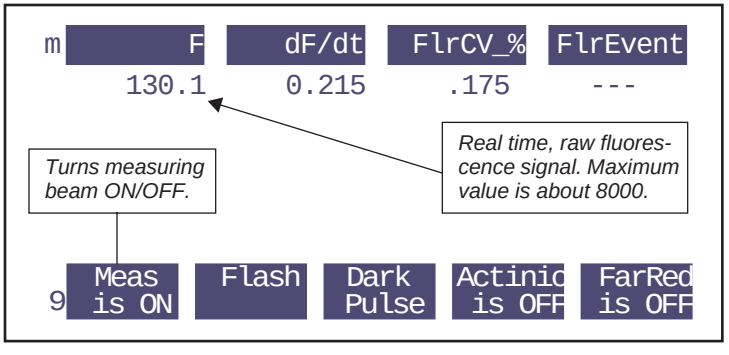


Figure 27-16. Basic fluorescence monitoring and control

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 27-66). 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 5.3 and above) support two types of saturating flash: single and multiple intensity (Figure 27-17). A **single** intensity 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 27-63). The **multiple** intensity flash uses three different intensities during the flash, and computes the fluorescence at saturation from the y-intercept of a plot of fluorescence vs. 1/intensity.

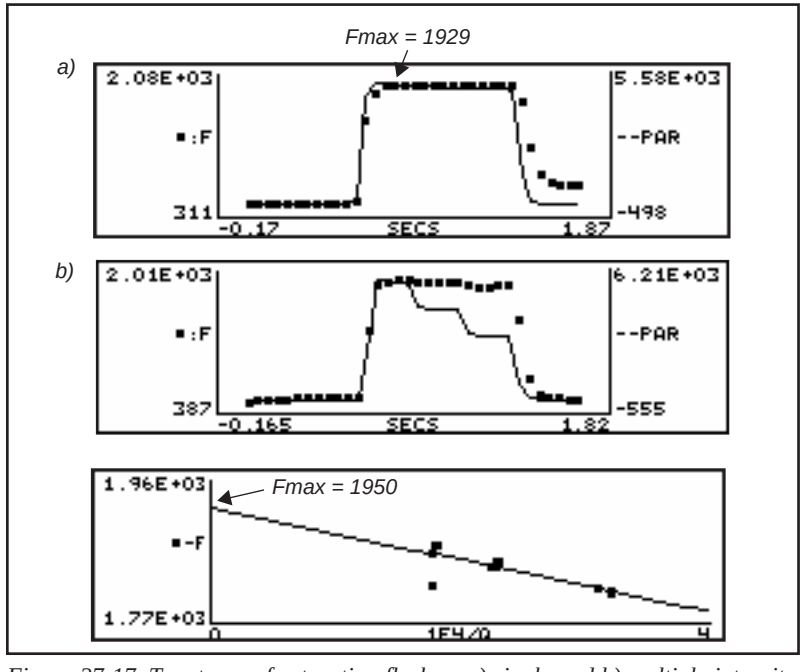


Figure 27-17. Two types of saturating flashes: a) single and b) multiple intensity with regression analysis.

There are several ways to trigger a saturating flash (Figure 27-18). 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.

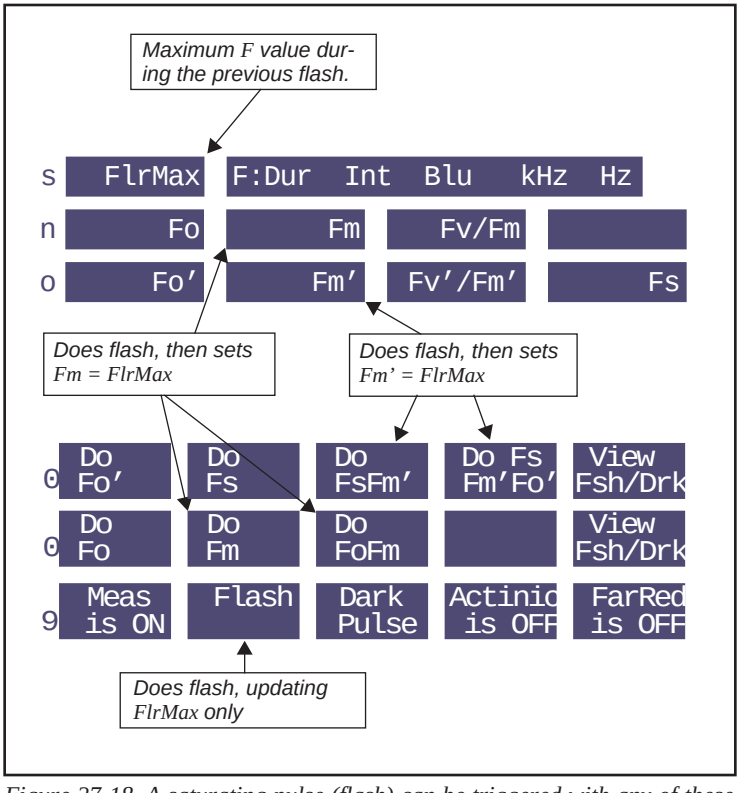


Figure 27-18. A saturating pulse (flash) can be triggered with any of these three function keys. They only differ in how the maximal fluorescence value is used.

Dark Pulses

A "dark pulse" (brief dark period) is designed to measure F_o' on a light adapted leaf. Figure 27-19 illustrates the timing parameters.

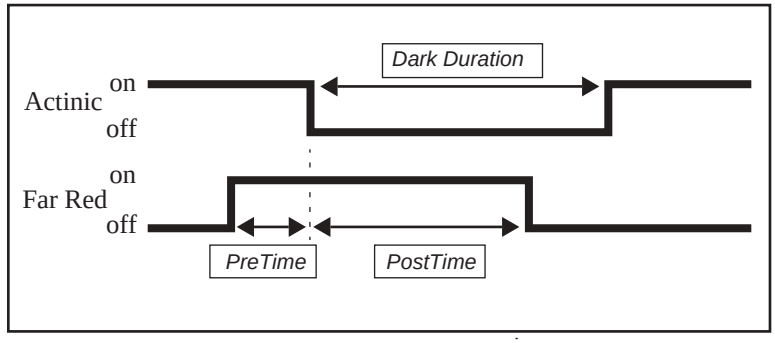


Figure 27-19. Illustration of the timing parameters for a dark pulse. The far red LED comes on PreTime *seconds before the actinic light goes off, and remains on for* PostTime *seconds after. The actinic is off for* DarkDuration *seconds.*

Figure 27-20 illustrates how to trigger a dark pulse.

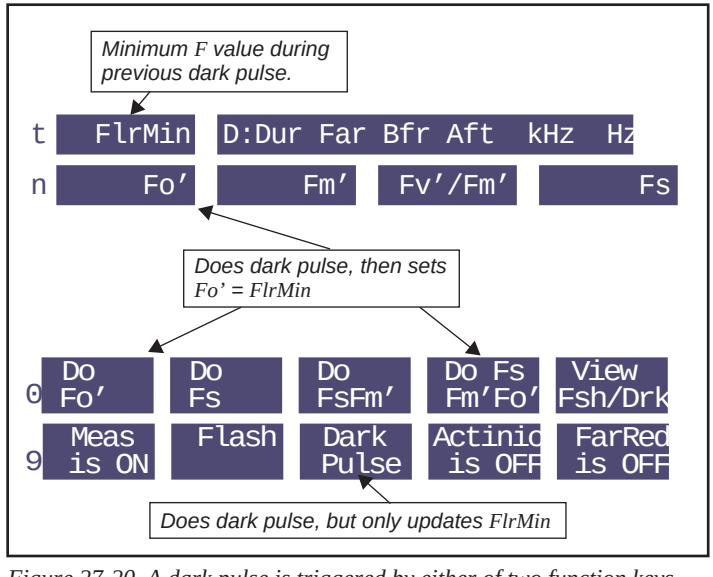


Figure 27-20. A dark pulse is triggered by either of two function keys.

Viewing Flash and Dark Pulse Details

View Fsh/Drk (**f5** level 0) allows you to view the latest flash or dark pulse (Figure 27-21). Press **View Flash** to see the fluorescence trace during the last flash, or **View Dark** to see the fluorescence trace during the last dark pulse.

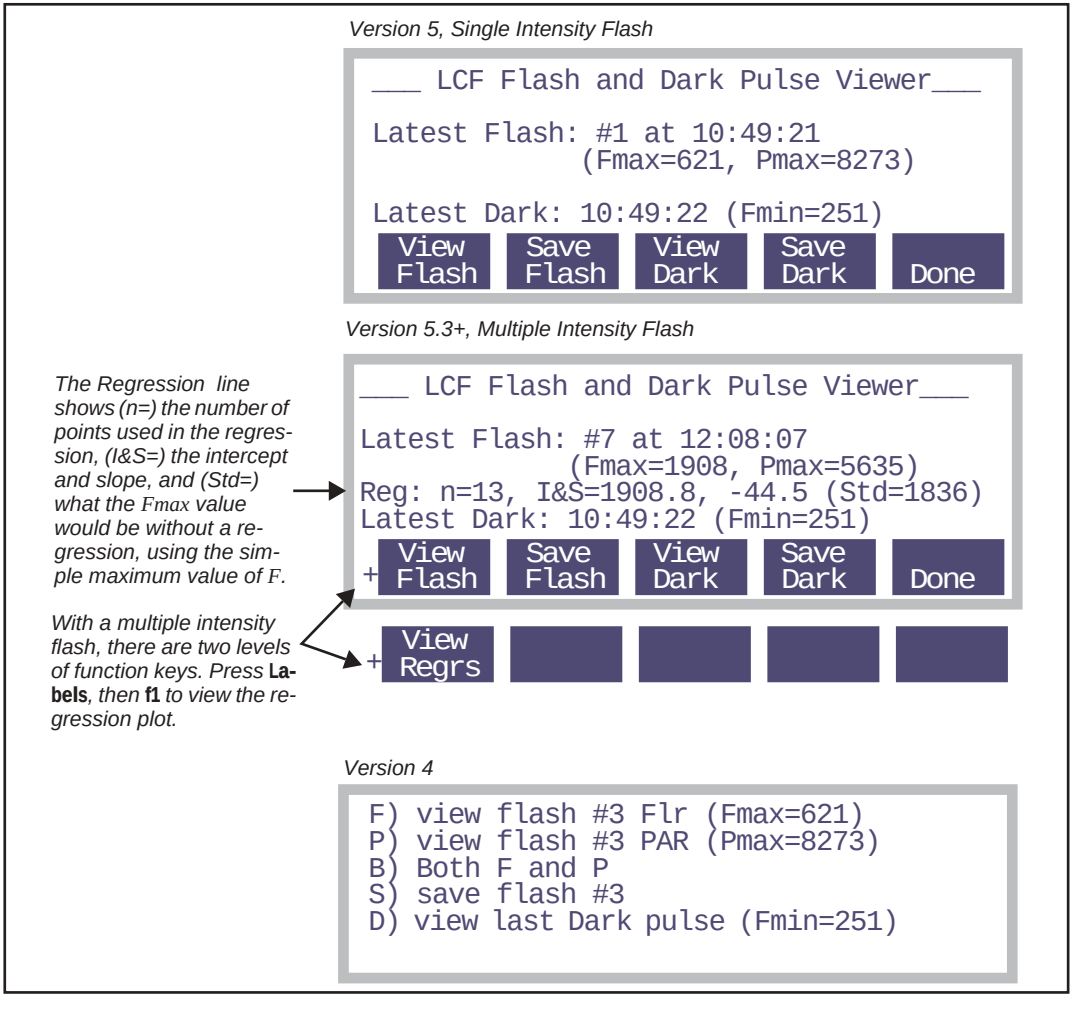


Figure 27-21. The View Flash/Dark screen for the two software versions.

The most recent flash and/or dark data can be saved to a file. The file is named automatically for you, and resides in the directory "/User/LCF". File names consist of the flash number and date, such as

```
#1 2002-03-27 10:12:39
DARK 2002-03-27 10:12:42
```

Sample flash files are shown plotted and as text saved in a file in Figure 27-22 (single intensity) and Figure 27-23 (multiple intensity).

Viewing Stored Flash Files.

After a flash (or dark) file has been stored, it can be viewed at a later time by using the "_View Flash File" program in the LCF Calibration Menu. This program will prompt you for a flash file using the Standard File Dialog box, and let you plot Flr, PAR, or both (as in Figure 27-22). It operates in a loop, so it is easy to view a series of flash data files.

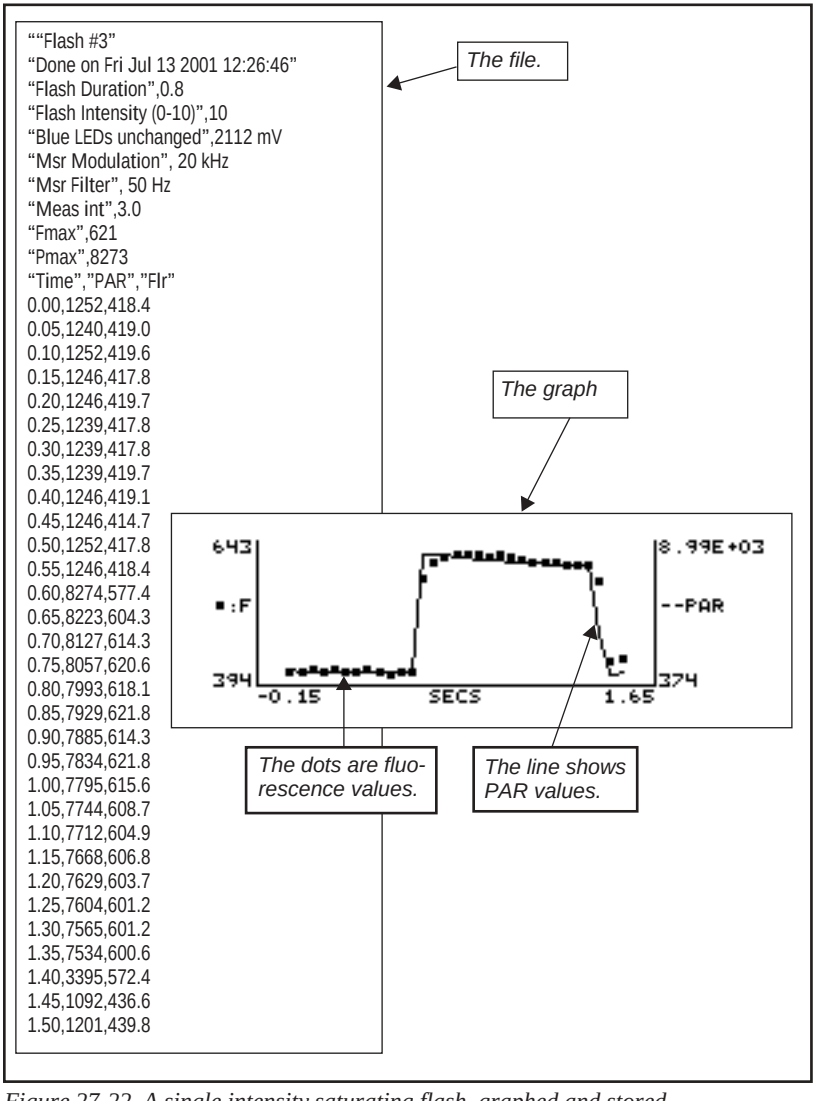


Figure 27-22. A single intensity saturating flash, graphed and stored.

"Flash #7"
"Done on Wed May 18 2005 12:08:07"
"Flash Type""Multiple"
"Flash Duration (s)"0.8
"Flash Intensities (0-10)"9 7 5
"Blue LEDs""NoChange"
"Msr Modulation(kHz)"20
"Msr Filter(Hz)"50
"Meas int"3.0

The subset of data used in the regression

"F"	1809.8	1828.6	1836.3	1829.3	1806.8	1807.7	1810.8	1811.7	1803.6	1764	1762.8	1765.8	1767.2
"Q"	5635	5604.38	5574.98	5231.98	4395.3	4375.7	4368.35	4362.23	4026.58	3144.57	3127.43	3128.65	3126.2
"1E4/Q"	1.77462	1.78432	1.79373	1.91132	2.27516	2.28535	2.28919	2.29241	2.4835	3.18008	3.19752	3.19627	
3.19877													

"Int&Slope"1908.8 -44.5315
"Simple Fmax"1836.3

The slope and intercept of the regression

Fmax without regression (single flash technique)

"Time"	"PAR"	"Flr"
0.00	9	610.2
0.05	10	614.0
0.10	7	625.6
0.15	9	634.8
0.20	7	643.2
0.25	10	649.2
0.30	9	656.0
0.35	10	660.7
0.40	10	666.2
0.45	9	671.3
0.50	10	674.7
0.55	70	693.5
0.60	4895	1749.3
0.65	5635	1809.8
0.70	5604	1828.6
0.75	5575	1836.3
0.80	5232	1829.3
0.85	4395	1806.8
0.90	4376	1807.7
0.95	4368	1810.8
1.00	4362	1811.7
1.05	4027	1803.6
1.10	3145	1764.0
1.15	3127	1762.8
1.20	3129	1765.8
1.25	3126	1767.2
1.30	2552	1727.7
1.35	445	1121.0
1.40	109	668.2
1.45	50	597.0
1.50	29	580.6
1.55	22	572.3
1.60	18	567.6
1.65	17	564.0

All of the data

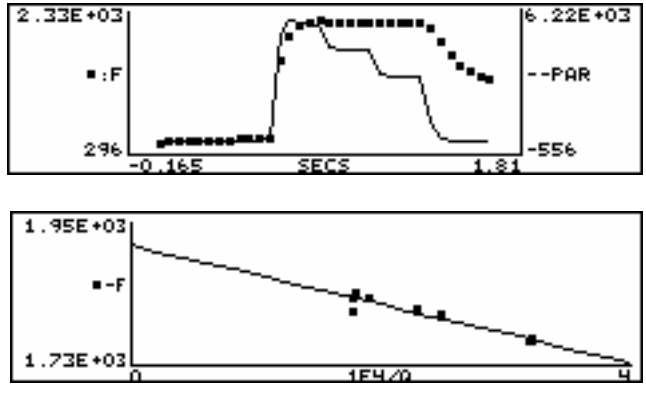


Figure 27-23. A multiple intensity flash, graphed and stored.

Display Summary

Table 27-3 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	m, o, p
pgdn	g, h, d

Table 27-3. 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
m	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 20 readings.	-98
FlrEvent	Indicates the last action performed or variable that was set.	-91
n	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	Variable to maximal fluorescence (dark). Equation (27-7) on page 27-4.	4202
o	Fo′ Fm′ Fv′/Fm′ Fs	

[3] See **Display Groups** on page 3-19.

Table 27-3. New Measurement displays added by "Std Flr Disp" (Continued)

Label	Description	[a]ID #
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 (27-9) on page 27-4.	
Fs	Steady-state fluorescence. Set by pressing **Do FsFm'** or **Do FsFm'Fo'** (**f3** or **f4** level 0).	4211
p PhiPS2 ETR qP qN		
PhiPS2	PS II efficiency. Equation (27-8) on page 27-4.	4212
ETR	Electron transport rate. Equation (27-11) on page 27-5.	4223
qP	Photochemical quenching. Equation (27-12) on page 27-5.	4216
qN	Non-photochemical quenching. Equation (27-13) on page 27-5.	4220
q Adark LeafAbs PS2/1 PhiCO2		
Adark	Dark photosynthetic rate (µmol CO_2 m^{-2} s^{-1}). (Sign convention: a value less than 0 indicates respiration). Set by pressing **Prompt All** (**f5** level 3).	4213
Leaf Abs	Leaf Absorptance. Computed based on the fraction of blue light, and two user-entered absorptances: one for blue, and one for red. Equation (27-17) on page 27-77.	4207
PS2/1	Photosystem distribution factor. Set by pressing **Prompt All** (**f5** level 3). This is *f* in Equation (27-11) on page 27-5.	4222
PhiCO2	Equation (27-10) on page 27-5.	4215
r F M:int kHz Hz Gn		
F	Fluorescence intensity. Measured continually.	-80
M: Int … Gn	Measuring beam parameters: intensity, modulation, filter, gain. See Table 27-6 on page 27-39.	-92
s FlrMax F:Dur Int Blu kHz Hz		
FlrMax	Maximal fluorescence obtained during the previous saturating flash.	-82

Table 27-3. New Measurement displays added by "Std Flr Disp" (Continued)

Label	Description	[a]ID #
F: Dur … Hz	Saturation flash parameters (Table 27-7 on page 27-39): duration, intensity, blue, modulation, filter.	-93
t 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 27-8 on page 27-40): 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 27-66.	-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
BlueAbs	Leaf absorptance at 460 nm. User entered constant.	4205
RedAbs	Leaf absorptance at 640 nm. User entered constant.	4206
PARAbs	Absorbed PAR. ParIn_µm * LeafAbs.	4208
NPQ	Alternative non-photochemical quenching. Equation (27-14) on page 27-6.	4221

a.ID values < 0 are system variables, and ID values > 0 are user variables. See **List of Open 5.3 System Variables** on page 14-19, and **Fluorescence ComputeList** on page 27-76.

Function Key Summary

Table 27-4 summarizes the New Measurements function keys enabled when the light source is set to “6400-40 LCF”.

Table 27-4. Summary of LCF function keys, New Measurements Mode

Label	Description
	
Flr QuikPik	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 27-36.
Define Actinic	Set actinic light, without turning on the LED’s. See **The LCF as a Light Source** on page 27-19.
FlrAdjust	Allows measurement, flash, and dark pulse settings to be changed on the fly, without interrupting measurements. See **Flr Adjust** on page 27-36.
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 27-34.
	
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.
 (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 27-4. 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 27-25.
0 Do Fo | Do Fm | Do FoFm | | View Fsh/Drk (Actinic Off)	
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 9-7.).
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 27-79.

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 27-5.

Table 27-5. 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	"14:10:32 Fs=1010"
Saturating flash	"14:20:32 Flash #8, 5580 um, Fmax = 1215 (0.8 7 NoC 20 50)[a]
Dark pulse	"16:18:40 Dark, Fmin=231 (3 8 1 1 1 0.5)"[b]

a."0.8 7 NoC 20 50" is a summary of the flash settings. See Table 27-6 on page 27-39.
b."3 8 1 1 1 0.5" is a summary of the dark pulse settings. See Table 27-6 on page 27-39.

An example of what the resulting log file looks like can be found in Figure 27-52 on page 27-81.

These remarks can be routed to a separate file. See **Log Options** on page 9-9.

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 saturating flashes and parts of dark events, when the spacing is 0.05 seconds. 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.

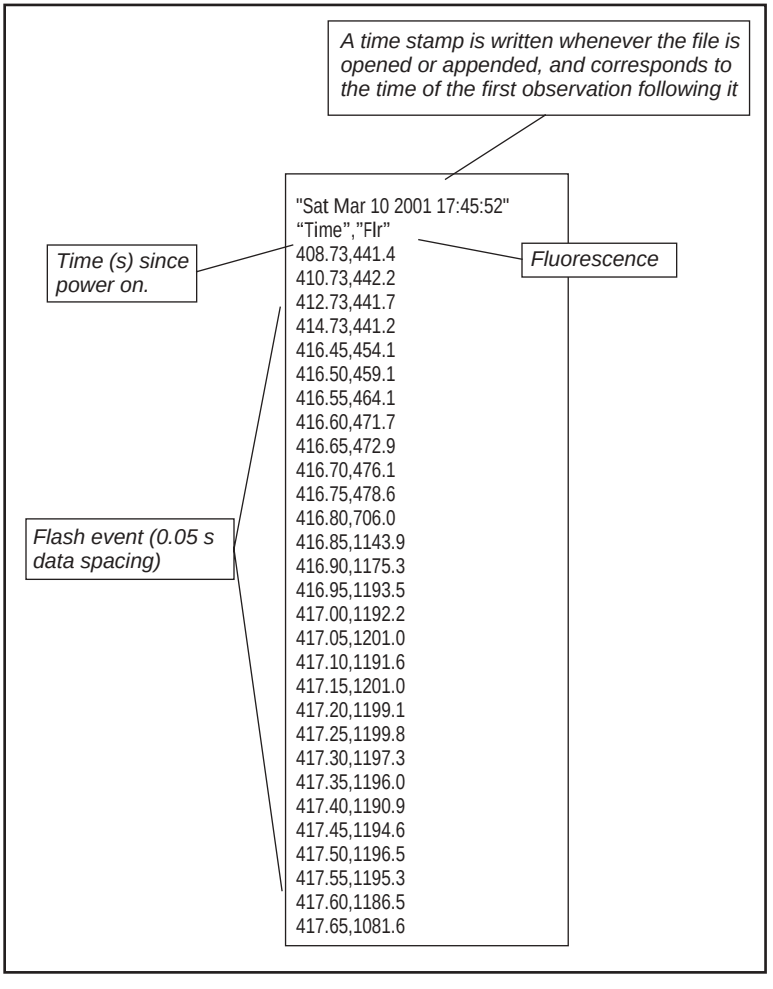


Figure 27-24. Example of a Fluorescence Recording file.

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 27-6 on page 27-39. 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 timing and measuring parameters are user defined, and illustrated in Figure 27-19 on page 27-24.

There are several ways to adjust these control parameters from the keyboard[4]:

Flr Editor

The Flr Editor (Figure 27-25 on page 27-37) is accessed via **f2** level 8. It allows any of the three groups of parameters to be edited, all in one dialog box. No changes take affect 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.

Flr QuikPik

Flr QuikPik is accessed via **f1** level 8. It presents a list of fluorescence parameter files (created by you using the Flr Editor), and prompts you to select one.

Flr Adjust

Flr Adjust (Figure 27-26 on page 27-38) is accessed via **f4** level 8. The advantage of Flr Adjust is that it allows fluorescence parameters to be adjusted without interrupting data collection and measurements. The disadvantage is that the response to keystrokes is sluggish because of the ongoing data collection and computations. HINT: Before entering Flr Adjust, put display line *m* on the top line, so you can immediately see the effect of changes to measuring

[4] To adjust them from a program, see **Leaf Chamber Fluorometer Control** on page 25-20 and **LCF Control Functions** on page 25-26.

parameters. Also, if you have real time graphics showing *F* as a function of time, you can switch back and forth from that display with one keystroke, **Charts** (**f4**).

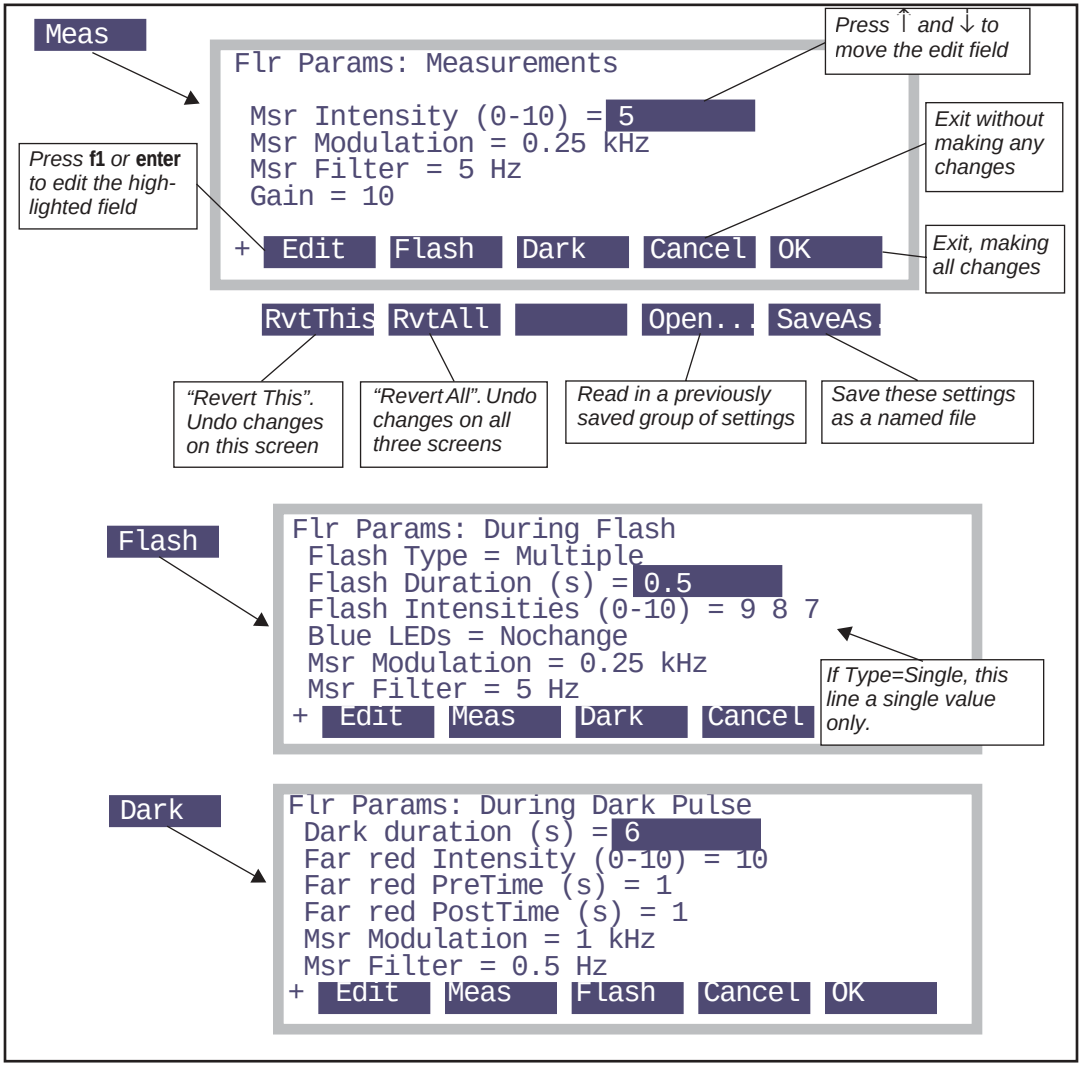


Figure 27-25. The Flr Editor, accessed by pressing **f2** *level 8. The editor is a dialog with three screens. Press* **Meas**, **Flash**, *and* **Dark** *to move between the screens.*

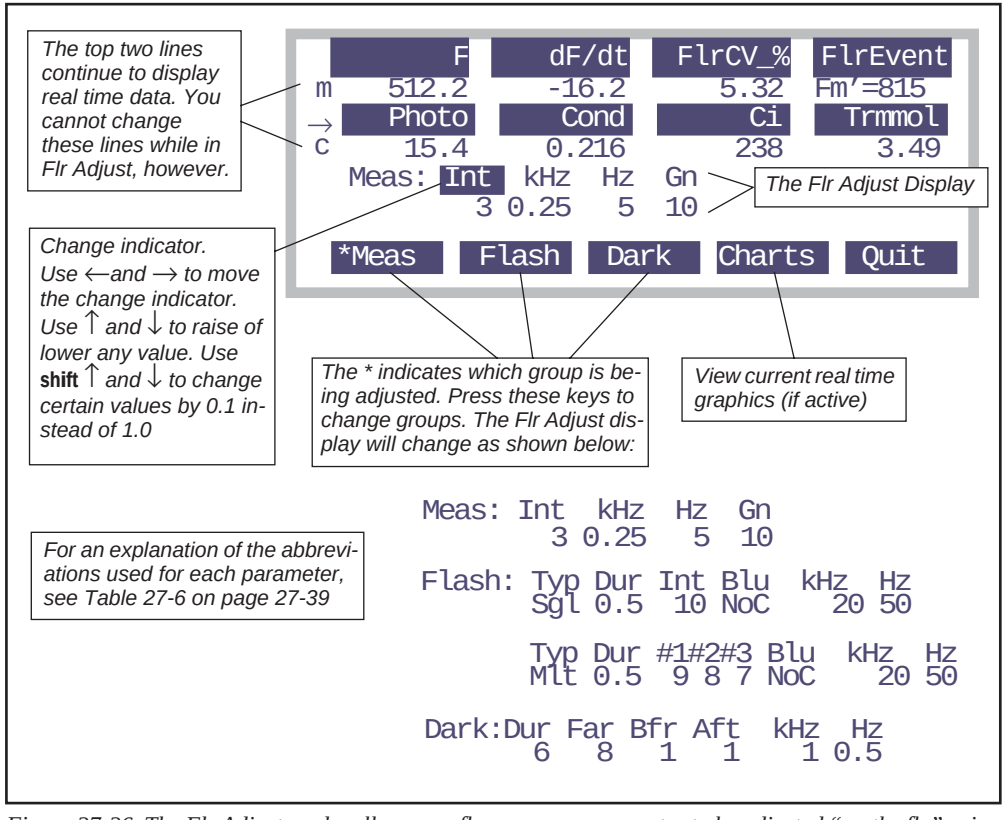


Figure 27-26. The Flr Adjust mode, allows any fluorescence parameter to be adjusted "on the fly" using only the arrow keys ←↑↓→, without interrupting normal New Measurements operations. Pressing **shift** *+ ↑or↓increments the selected field by 0.1 rather than 1.0.*

Table 27-6. Fluorescence Measurement Parameters

Editor Label (Adjust Abbreviation)	Description
Msr Intensity (0-10) (Int)	Intensity of the 2 blue measuring LEDs. Typical value is 5. You can entering floating point values (e.g. 3.3) if you wish - they don't have to be integers. Effective resolution is 0.1.
Msr Modulation (kHz)	Modulation used normally (i.e. not in a saturating flash or a dark pulse). (0.25, 1, 10, or 20 kHz).
Msr 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 1.
Gain (Gn)	Gain factor for the fluorescence signal (10, 20, 50, or 100). Use 10.

Table 27-7. Fluorescence Saturating Pulse (Flash) Parameters

Editor Label (Adjust Abbreviation)	Description
Flash Type (Typ)	Single or Multiple intensity. See **Saturating Flashes** on page 27-22.
Flash Duration(s) (Dur)	Length of the saturating flash. Keep it between 0.2 and 2.0 seconds.
Flash Intensity (0-10) (Int)	Saturating flash intensity. Typically, 10 will be 6000 or more umoles. (This is measured, and displayed as "FPeak_µm" on display line *u*.)
Flash Intensities (0-10) (#1#2#3)	The three intensity values for a multilevel flash.
Blue LEDs (Blu)	Do what with the blue actinic LEDs during the saturating flash? One of: "Off" "NoChange" "MaxOn" "Off" turns the blue off, "NoChange" leaves the blues alone, and "MaxOn" turns them on full for the flash.
Msr Modulation (kHz)	Fluorescence modulation to use during the flash. 20 kHz is typical.
Msr Filter (Hz)	Averaging to do during the flash. 20 Hz, typically.

Table 27-8. Fluorescence Dark Pulse Parameters

Editor Label (Adjust Abbreviation)	Description
Dark duration (s) (Dur)	Length of dark interval. Typically < 5 secs.
Far red intensity (0-10) (Far	Intensity of far red. (Setting of 10 is typically about 5 µmol m^{-2} s^{-1}).
Far red Pretime (s) (Bfr)	Far red LED timing. See Figure 27-19 on page 27-24.
Far red Posttime (s) (Aft)	
Msr modulation (kHz)	Fluorescence modulation to use during the dark pulse.
Msr Filter (Hz)	Averaging to use during the dark pulse. 1 Hz is typical.

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 27-71.

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

Fv/Fm 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.

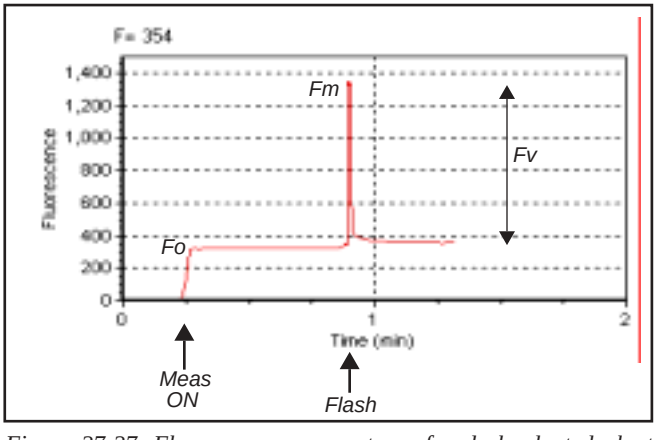


Figure 27-27. Fluorescence parameters of a dark-adapted plant upon application of a saturation flash.

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 27-64.
	Modulation	0.25	Always use 0.25 kHz for dark adapted leaves
	Filter	1	
	Gain	10	
Flash	Duration	0.8	0.5 to 0.8 is typical
	Intensity	7	
	Blue	NoChange	
	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 *m*. 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 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 *Fo*, *Fm*, and *Fv* / *Fm* on display line *n*. Is Fv/Fm 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 Flash** (**f1**) to view the flash (Figure 27-21 on page 27-25). The detailed data collected at 20 Hz during the flash will be plotted versus time. Looking at the details of the flash makes it is 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 27-28. Note that there is an automated way of determining the appropriate flash setting. See **"Optimum Flash Intensity"** on page 27-63.

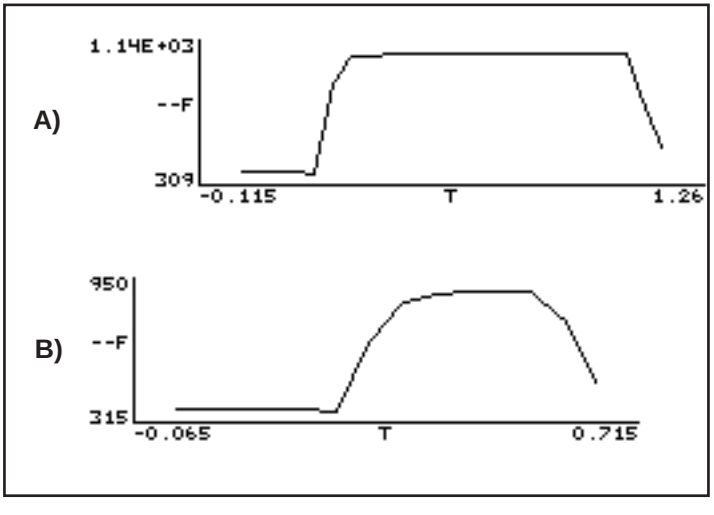


Figure 27-28. A) Example of a good saturation flash: flash length and intensity are appropriate for this leaf material (philodendron dark-adapted for 20 minutes). B) Example of a poor saturation flash: flash length was too short and intensity too small.

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 (also called ΔF/Fm') is the fraction of absorbed PSII photons that are used in photochemistry, and is measured with a light adapted leaf (Equation (27-8) on page 27-4). 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 27-29). See also Genty et al. (1989).

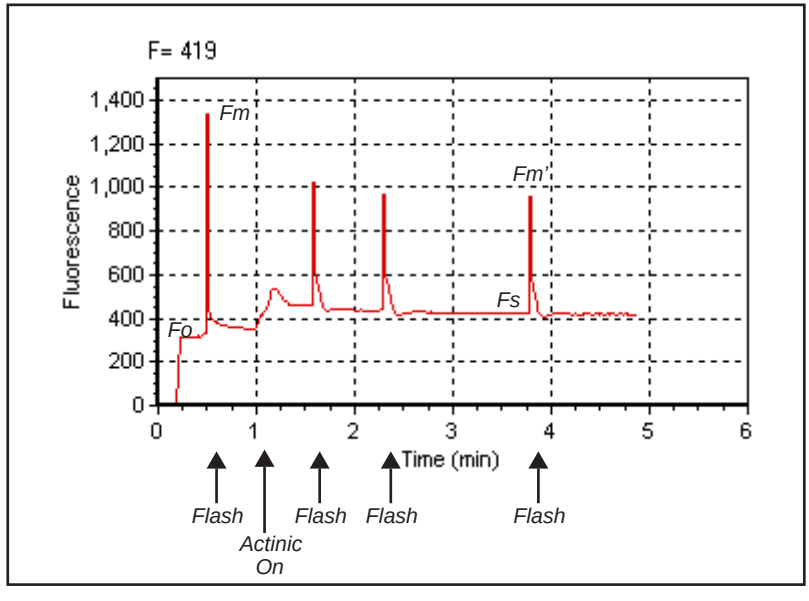


Figure 27-29. Fluorescence parameters of a light-adapted plant upon application of a saturation flash.

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.

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	Not so important for light adapted leaves
	Modulation	20 kHz	10 or 20 should be fine
	Filter	1	
	Gain	10	
Flash	Duration	0.8	0.5 to 0.8 is typical
	Intensity	8	See **"Optimum Flash Intensity"** on page 27-63.
	Blue	NoChange	
	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 *m*) 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). *Fs* and *Fm'* can be viewed on display line *o*, and *PhiPS2* on display line *p*. How does *PhiPS2* compare with the

Fv/Fm 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 27-28 on page 27-43, 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 re-evaluated 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 $\Delta F / F_m'$ 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 F_v / F_m and Quenching Coefficients

Three other useful fluorescence parameters will be explored in this experiment. F_v' / F_m' (Equation (27-9) on page 27-4) 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 (q_N) quenching (Equations (27-12) and (27-13) on page 27-5). Many disciplines use these parameters, but the latter two are particularly useful as quantifiers in stress physiology research.

All three of these parameters require Fo', 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 27-30 on page 27-47) 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 second method to determine *Fo'*. See Figure 27-19 on page 27-24 for an illustration of the dark pulse timing parameters.

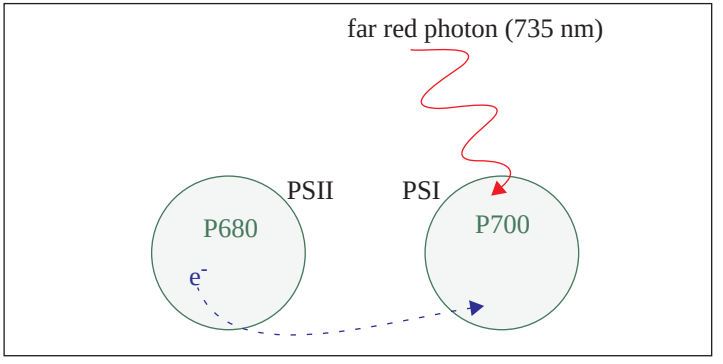


Figure 27-30. PSI is preferentially excited by far-red light which drives electron transport of PSI, and thus drains electrons from PSII. This is a method of rapid equilibration for determining Fo'.

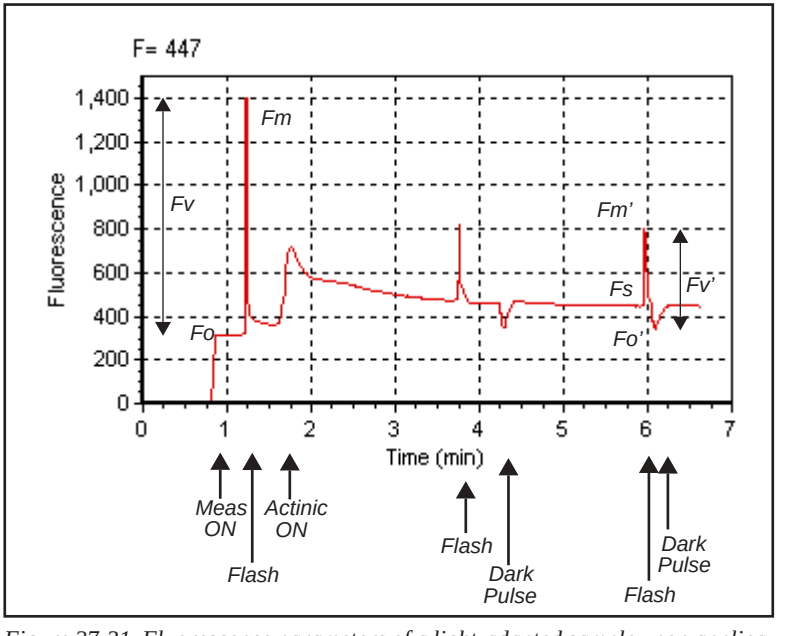


Figure 27-31. Fluorescence parameters of a light-adapted sample upon application of a saturating flash, followed by a dark period with far-red light.

1 Select some plants

We'll need both dark and light-adapted leaves for this exercise.

2 Set LCF settings as in Experiment #1

In addition to the settings used before (Table 27-9 on page 27-42), this exercise also requires the calculation of F_o and F_o', which requires configuring the dark pulse parameters (Table 27-11) to measure *Fo'*.

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	1 sec.	Far red turns off 1 second after actinic goes off.
	Modulation	0.25 kHz	As with Fo, 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 *m*) becomes stable. (e.g., wait until | *dF/dt* | < 5.)

4 Measure Fv / Fm

Press **Do FoFm** (**f1** level 0). *Fo* (display line *n*) is immediately set to the current value of *F*. Then a saturating flash is done, and *Fm* is set to the maximum value during the flash. Following the flash, the data is logged. Check the *Fv / Fm* value (display line *n*) 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 *Fs*, do a saturating flash and set *Fm'*, and then a dark pulse to set *Fo'*. When it is done

(it will take about 12 seconds), view the flash details and check the new *Fs*, *Fm'*, and *Fo'* parameters in group *o*. The light-adapted flashes should resemble plot A in Figure 27-28 on page 27-43.

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 27-32).

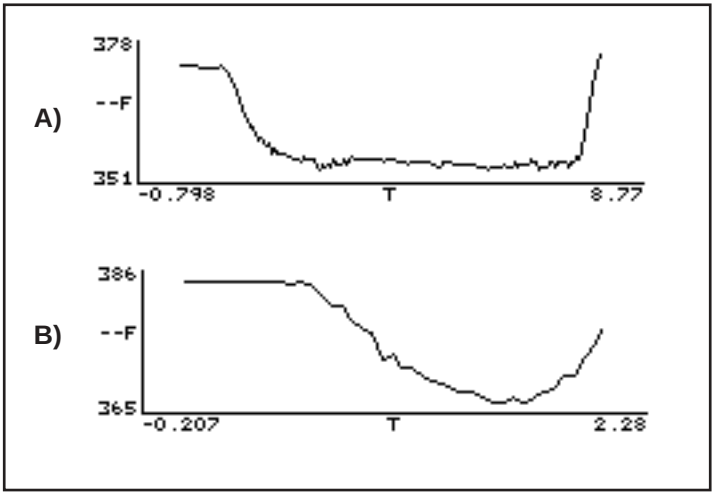


Figure 27-32. A) Example of a good dark curve: far red intensity and time were appropriate for the leaf material (notice the curve flattened before the actinic light was turned on). B) Example of a poor dark curve: far red intensity is not bright enough and time is too short to get a flat curve.

8 Compare F_v / F_m with F_v' / F_m'

Look at the calculated F_v' / F_m' value (display line *o*). Question: How should this value compare with the F_v / F_m values collected in Experiment #1? Answer: Fv' / Fm' should be < Fv / Fm, since Fv / Fm is the maximum light harvesting efficiency, and Fv' / Fm' is the actual light harvesting efficiency at some light level, reduced by competing processes for light energy.

9 Examine the quenching coefficients

The quenching coefficients, *qP* and *qN*, are located in display group *p*, and are described in Equations (27-12) and (27-13) on page 27-5. *qP* and *qN* range in value between 0 and 1. For a truly dark adapted plant, *qP* = 1 and *qN* = 0 at the time *Fv/Fm* is determined. As light increases, these coefficients tend

to move in opposite directions (see, for example, Figure 27-33 on page 27-52.)

10 Repeat steps 3-7 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 use AutoPrograms. If you are not familiar with these, the basics are discussed in **AutoPrograms** on page 9-20.

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, 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).

Setup 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 6-11.

3 Prepare chamber environment

CO_2 - If there is a mixer installed, control at ambient values in the sample cell (e.g. 360 µmol mol^{-1}).

Light - Set for a normal mid-day value (**f3** level 8), but leave Actinic turned off (**f1** level 9).

Flow - Fixed at 500 µmol s^{-1}.

4 Prepare LCF settings
Set as determined in experiments 1 and 2 with dark adapted material.

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 "Flr Kinetic"
Autoprograms are launched by pressing **Auto Prog** (**f1** level 5). Select the AutoProgram named "Flr Kinetic". Once you've named the data file, you will be asked a series of questions (Table 27-12 on page 27-51).

Three of the four fluorescence AutoPrograms provided ("Flr Kinetic", "Flr ACi-Curve", and "Flr Light Curve") have a standard set of fluorescence related prompts. These are indicated by the shaded entries in Table 27-12, and show how you should respond to all the prompts for this experiment.

Table 27-12. Fluorescence prompts, and responses for Flr Kinetic.

Prompt	Enter	Discussion
Dark adapt before starting (Y/N)	Y	
Dark time before Fo (min):	20	If your leaf is already dark adapted, enter 0.1.
Light time after Fo (min):	0.1	This is short, because we don't want to give the leaf time to light adapt.
Measure dark photo at Fo (Y/N)	Y	
Measure Fo' with Fs & Fm' (Y/N)	Y	Fo' is needed for the quenching coefficients
Enable Flr Recording before Fo (Y/N)	Y or N	Your choice. If you aren't sure, press N
Turn off Flr Recording when done (Y/N)		
Loop N Times	5	
Time between flashes (minutes)	3	
Match if |ΔCO2| less than (ppm):	0	*No matching*

7 Watch
After starting the program, watch the real time graphics display (**f3** level 4, or else **]**). Given enough time, a steady-state fluorescence level, Fs, and steady-state photosynthesis rate should develop. What happens to *qP* and *qN* 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.)

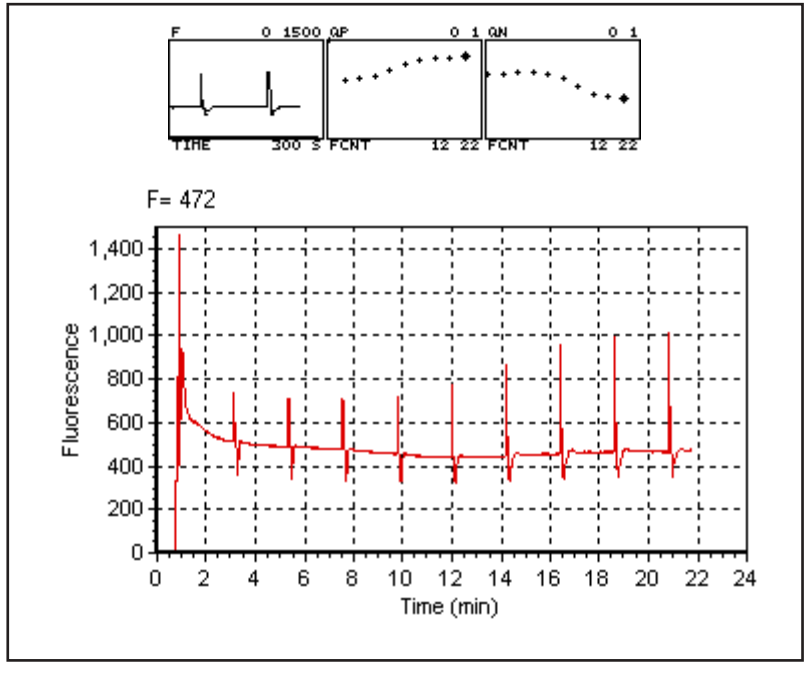


*Figure 27-33. Sample results from Experiment 4. The upper graphs are the real time graphics images produced on the LI-6400 console, and show the last 300 seconds of F, and QP and QN as a function of flash count (FCNT). The lower plot showing the trace of F for the entire experiment illustrates a use 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 (27-8) and (27-10) on page 27-5).

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 mea-

surement - of assimilation in the dark (*Adark*). We also need absorbed PAR, which involves knowing incident PAR and leaf absorptivity (see **Leaf Absorptance** on page 27-77).

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. If you do not do this, the measured relationship between Φ_{PSII} and Φ_{CO_2} will likely not be linear. Using a C_4 plant avoids all of this.

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 "Flr Light Curve" 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_2at 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:

Adark: Photosynthesis rate in the dark. Use -1 µmol m^{-2} s^{-1}.
BlueAbs: Leaf absorptance in the blue. Use 0.9 (see Table 27-21 on page 27-78).
RedAbs: Leaf absorptance in the red. Use 0.84.
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 *m*, (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, change the flow control to constant water mole fraction using the current sample cell concentration as the target. Also, change the CO_2 control to target the sample cell, using the current sample cell value as the target value.

5 Set up real time graphics for the light curve

QuikPik (**f2** level 4) the real time graphics file "A-Q, PhiPS2-PhiCO2".

6 Launch the AutoProgram

Choose the program "Flr Light Curve" from the AutoProgram menu (**f1** level 5). Name the file, then answer the prompts as indicated in Table 27-13.

Table 27-13. Fluorescence prompts, and responses for Flr Light Curve.

Prompt	Enter	Discussion
Dark adapt before starting (Y/N)	N	
Fo value		Doesn't matter.
Fm value		Doesn't matter
Dark photo rate	-1	Use -1, or your measured value.
Measure Fo' with Fs & Fm' (Y/N)	N	Fo' won't be needed
Turn on Flr Recording (Y/N)	Y or N	Your choice. If you aren't sure, press N
Turn off Flr Recording when done (Y/N)		
Actinic Control: <shows current target>	Y or N	*If the current target is not "PQntm, 10% blue", then press N and change it.*
Desired lamp settings (µmol/m2/s)	2000 1600 1200 800 400	
Minimum wait time (min)	3	
Maximum wait time (min)	10	
Match if |ΔCO2| less than (ppm):	10	
Current stability uses *n* variables. OK?	Y/N	*See* **Defining Stability** *on page 6-25*

7 Watch the light curve develop

The real time graphics should be something like that shown in Figure 27-34.

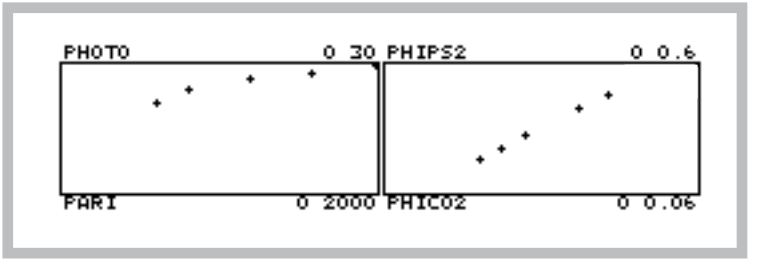


Figure 27-34. Typical light curve real time graphics. The program does light from high to low, so the light curve (left) will develop from right to left, while the PhiPS2-PhiCO2 plot (right) will develop from left to right.

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 **Graph Quikpik** (**f1**) and choose "Light Curve". The light curve (*Photo* vs. *parIn_µm*) will be drawn from the data (Figure 27-35).

Figure 27-35. Light curve for Experiment 5, using GraphIt.

Now, plot PhiPS2 against PhiCO2. The quickest way to do this is to press **Graph Quikpik** (**f1**) and pick "PhiPS2 vs PhiCO2". This will plot the points, and fit a straight line through them (Figure 27-36). 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 27-82.

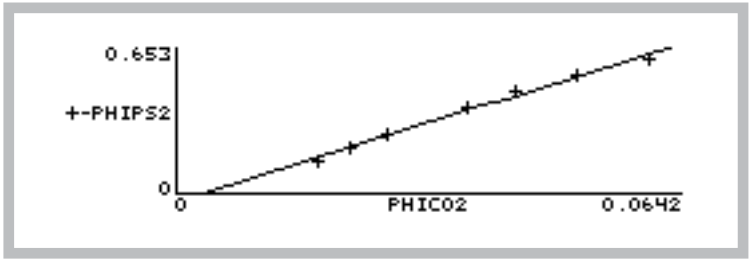


Figure 27-36. PhiPS2 vs. PhiCO2 from Experiment 5.

To view the slope and intercept, press **View Data** (**f3**), then **C** for Curvefit coefficients. For the data shown in Figure 27-36, 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.

Fluorescence AutoPrograms

Standard Fluorescence Prompts

These prompts are part of three fluorescence AutoPrograms, and constitute the first group of questions asked.

Table 27-14. Standard Fluorescence Prompts for AutoPrograms.

Prompt	Discussion
Dark adapt before starting (Y/N)	If Y, program will wait, measure Fo and Fm, set the actinic light, and wait again for light adaptation. If N, this is skipped.

Table 27-14. Standard Fluorescence Prompts for AutoPrograms.

Prompt			Discussion
If dark adapt = Y	Dark time before Fo (min):		Wait time before setting Fo and applying a saturating pulse to measure Fm.
	Light time after Fo (min):		Acclimation time after actinic is turned on.
	Measure dark photo at Fo (Y/N)		If Y, then right before Fo is set, the value of Photo is used to set Adark.
	If measure dark photo= N	Dark photo rate:	Enter the value to be used for dark photosynthesis.
	Enable Flr Recording before Fo (Y/N)		"Y" will turn Flr Recording on just before the Fo measurement.
If dark adapt = N	Fo value:		If not doing the dark adaptation routine, then you are prompted with the current values of Fo, Fm, and Dark Photo, and given a chance to set them to whatever values you wish.
	Fm value:		
	Dark photo rate:		
	If Flr Recording is Off	Turn on Flr Recording (Y/N)	In case you want it on, and forgot to do it.
Measure Fo' with Fs & Fm' (Y/N)			Y = Do FsFm'Fo' before each log N = Do FsFm' before each log
If Flr Recording will be on at the end	Turn off Flr Recording when done (Y/N)		

Flr A-Ci Curve

"Flr A-Ci Curve" performs an A-Ci curve, similar to that described in **"A-Ci-Curve"** on page 9-23. Each time logging is done, however, it is preceded by a fluorescence measurement. The prompts are shown in Table 27-15.

Table 27-15. Flr A-Ci Curve prompts.

Prompt	Discussion
Standard Fluorescence Prompts	*See Table 27-14 on page 27-56.*
Desired Ca values (µmol/mol)	
Minimum wait time (mins)	
Maximum wait time (mins)	

Table 27-15. Flr A-Ci Curve prompts.(Continued)

Prompt	Discussion
Match if |ΔCO2| less than (ppm):	Enter 0 to skip matching.
Current stability uses *n* variables. OK?	*See* **Defining Stability** *on page 6-25*

Flr Kinetic

"Flr Kinetic" simply loops N times. Each loop consists of a waiting period, followed by an optional match, followed by an fluorescence event, either Do FsFm' or Do FsFm'Fo'. The prompts are shown in Table 27-16.

Table 27-16. Flr Kinetic prompts.

Prompt	Discussion
Standard Fluorescence Prompts	See Table 27-14 on page 27-56.
Loop N Times	Number of times
Time between flashes (minutes)	Wait time each loop
Match if |ΔCO2| less than (ppm):	Enter 0 to skip matching.

Flr Light Curve

"Flr Light Curve" performs an light curve with a fluorescence measurement. The prompts are shown in Table 27-17.

Table 27-17. Flr Light Curve prompts.

Prompt	Discussion
Standard Fluorescence Prompts	See Table 27-14 on page 27-56.
Desired lamp values (µmol/m2/s)	
Minimum wait time (mins)	
Maximum wait time (mins)	
Match if |ΔCO2| less than (ppm):	Enter 0 to skip matching.
Current stability uses *n* variables. OK?	*See* **Defining Stability** *on page 6-25*

Flr Loop

"Flr Loop" is a simple program designed to do repetitive fluorescence measurements. The prompts are shown in Table 27-18.

Table 27-18. Flr Loop prompts.

Prompt	Discussion
How many events?	
Time between events (secs)	
Pick Flr event 1) Fs Fm' 2) Fs Fm' Fo' 2) Fo' 4) Fo Fm Press (1/2/3/4)	Press 1, 2, 3, or 4 to pick the type of measurement to be done each time.

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 PS I. 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 µmol m^{-2} 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 27-2 on page 27-7). 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 is typically very small (0.02 and 0.2 µmol m^{-2} s^{-1} - see Figure 27-37) because of the modulation. If the modulation is turned off, and the measuring beam setting turned up to 10, typical output is about 90

µmol m^{-2} s^{-1}. However, the internal light sensor will usually *not* measure this accurately, since it is not calibrated for those two LEDs.

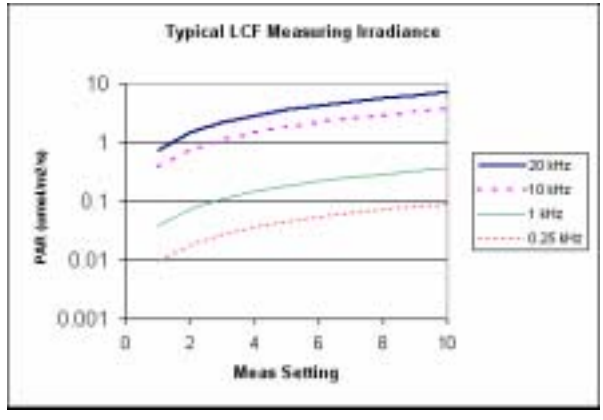


Figure 27-37. Typical measuring beam quantum irradiance values for the 6400-40 LCF, as a function of measuring beam setting and modulation rate. These data were measured with an LI-1800 Spectroradiometer and an integrating sphere.

LCF Calibration Menu Programs

Fluorometer calibration routines are accessed via "_Light Source Calibration Menu" found in the Calib Menu. When the LCF is installed, selecting this entry will bring up the following submenu:

```
6400-40 LCF Calibration
"Actinic Control - calibrate"
"Actinic Control - plot"
Optimum Flash Intensity
Optimim Meas Intensity
Zero fluorescence signal
_View Factory Cal
_View Flash Files
```

Figure 27-38. The LCF calibration menu

"Actinic Control - Calibrate"

This calibration routine will generate the relation between red and blue control voltages and resulting quantum fluxes in the chamber. These relations 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 (Figure 27-38).

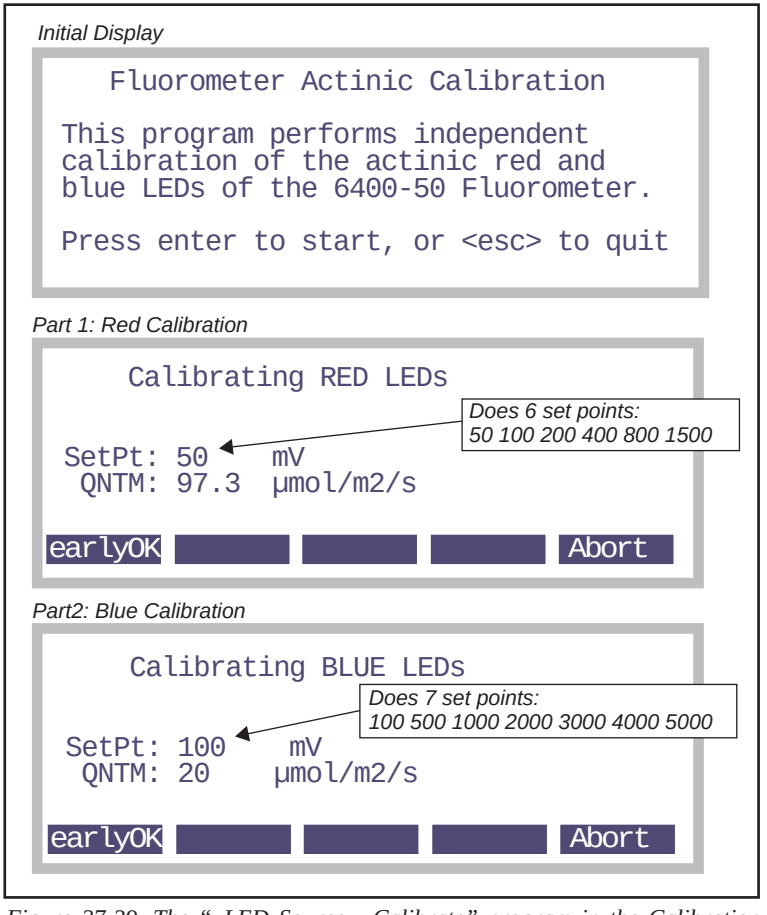


Figure 27-39. The "_LED Source - Calibrate" program in the Calibration Menu.

When it is done, the data points are printed, and you are given an opportunity to view plots of the curves (Figure 27-40). You are then asked whether or not to implement this calibration:

```
Implement this cal? (Y/N)
```

If you press **Y**, The data is stored in file "/user/configs/fluor", and becomes the active calibration.

"Actinic Control - Plot"

This routine plots the currently active relation between red and blue control signals and resulting in-chamber quantum flux. Typical curves are shown in Figure 27-40.

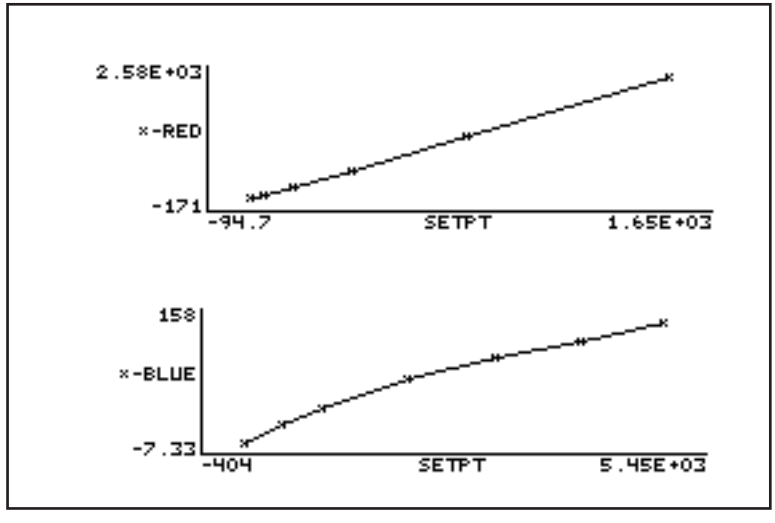


Figure 27-40. Red and Blue control signal calibration curves.

Press **escape** when done viewing each curve. To view the actual values in either curve, press **V** instead of **escape**.

```
Pt#   SETPT   Blue
1   83.92  0.2463
2   513.2   15.85
3   981.5   39.91
4    1996    68.7
5    3011   82.21
6    3986   92.91
7    4962   108.2
```

Figure 27-41. View the data points by pressing V when viewing the graph.

"Optimum Flash Intensity"

This program is designed to be an aid in determining the best flash intensity to use (Figure 27-42). 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**.

```
Enter intensities:
7 8 9 10
Recovery time (minutes): 3
Store each flash? (Y/N)
Store results at end? (Y/N)
```

Figure 27-42. 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 27-43.

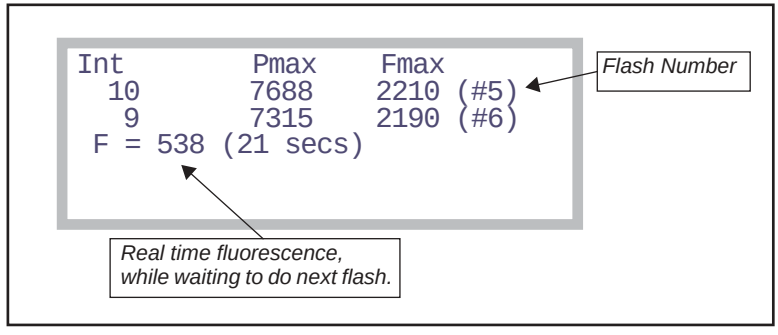


Figure 27-43. Optimum Flash Setting during operation.

After doing the list of flash intensities you asked for, 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 27-44.

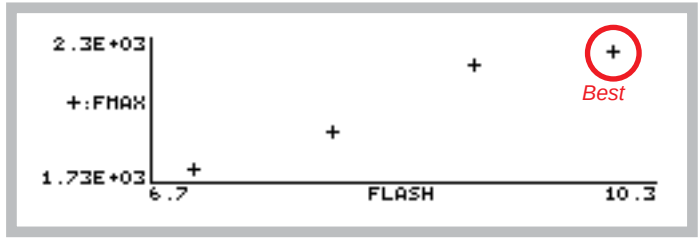


Figure 27-44. Plot of results from the Optimum Flash Intensity program.

"Optimum Meas Intensity"

This program is designed to help you find the proper measurement intensity for determining Fo. That is, **the intensity that is as large as possible without inducing photosynthesis**.

You are prompted for a series of measuring beam intensities (Figure 27-45) 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 fluorescence 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 beam 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 27-45. 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 27-46). During the recovery period, *Meas* will show "off", and the time remaining.

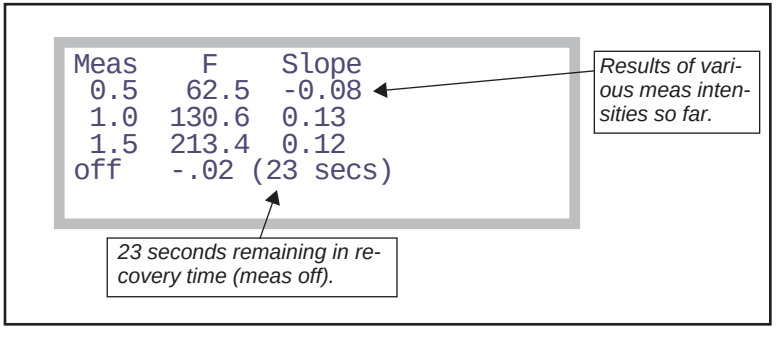


Figure 27-46. Optimum Meas Setting during operation.

At the end, dF/dt as a function of measuring intensity is plotted for you (Figure 27-47).

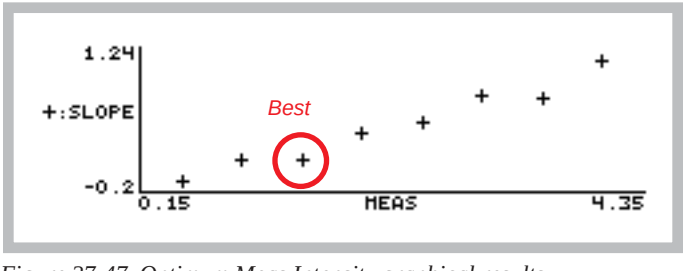


Figure 27-47. Optimum Meas Intensity graphical results.

"Zero Fluorescence Signal"

This routine is used to zero the LCF fluorescence signal. The chamber does NOT have to be empty to do this.

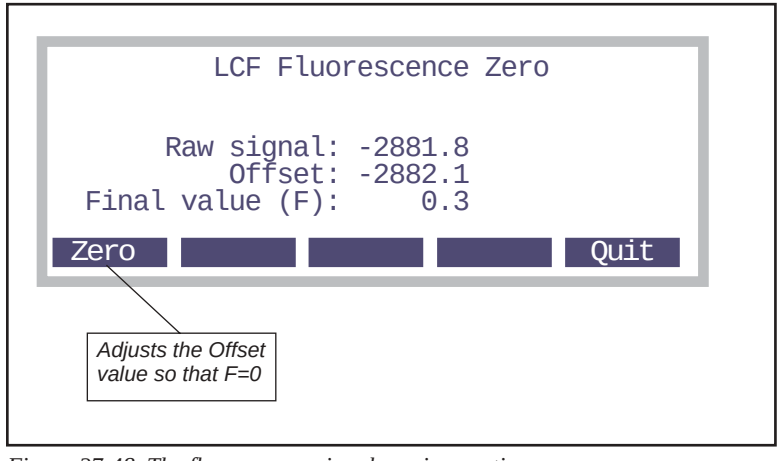


Figure 27-48. The fluorescence signal zeroing routine.

"_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.

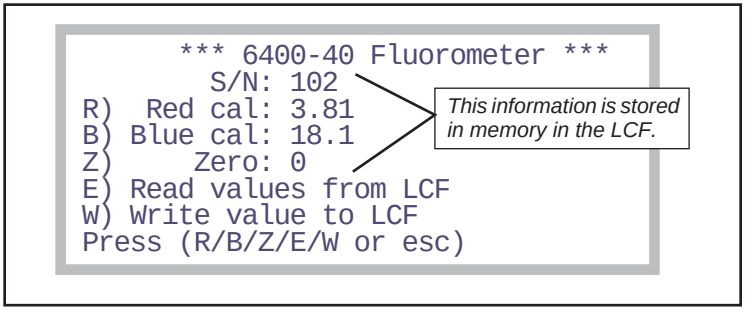


Figure 27-49. The information that is stored in the LCF can be viewed and edited with this routine. Appropriate key strokes are described in Table 27-19.

Table 27-19. _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="
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 27-66.)
E	Re-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.

"_View Flash Files"

This program allows you to re-plot stored flash files. It operates in a loop, and lets you pick flash files that were stored during new measurements mode or during the Optimum Flash Intensity program described on page 27-63. See **Viewing Flash and Dark Pulse Details** on page 27-25.

LCF Troubleshooting

"LCF Control Panel" Program

A convenient place to do troubleshooting on the LCF is the "LCF Control Panel" program, located in the "Diagnostics &^ Tests Menu" in the Welcome Menu.

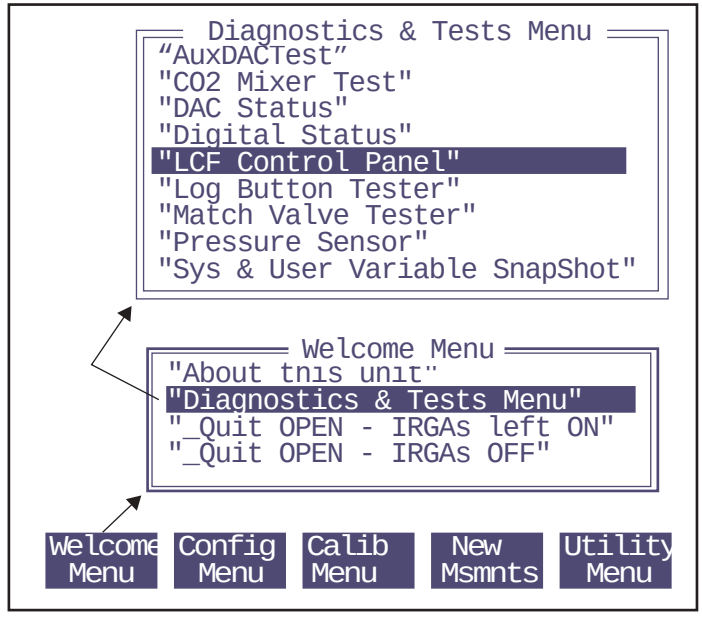


Figure 27-50. Navigating to the LCF Control Panel program from OPEN's main screen function keys.

This program presents a screen shown in Figure 27-51. Real time values of fluorescence and PAR, as measured by the LCF detectors, are displayed. At the bottom of the display are the current settings of all the basic controls. The values of these settings are adjusted by pressing the up and down arrow keys. The field that is being changed is highlighted. To move to a different field, press the left or right arrow.

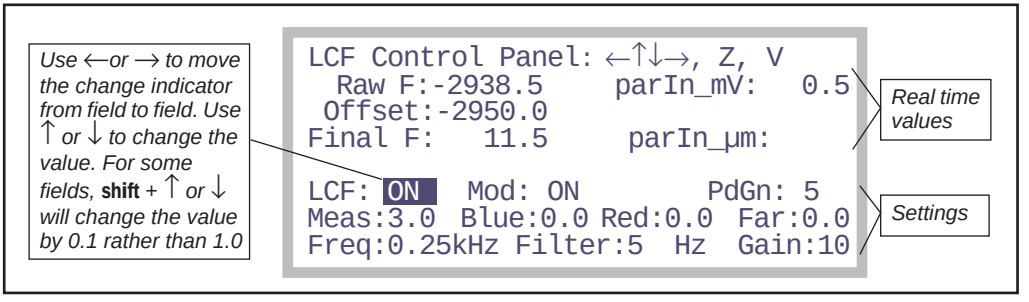


Figure 27-51. The LCF Control Panel program. The top lines show real time values of fluorescence and the parIn_µm values. The bottom three lines are the settings fields. The highlighted value can be changed by pressing ↑ or ↓. For the Meas, Blue, Red, and Far field, **shift** *+ ↑ or ↓ changes the settings 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 27-66.*

"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 27-66) 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 blue 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 about 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"[5] 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.

---

[5] 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.

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**
Turn the Reds to 1 and see that they 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.

LCF Reference

Configuration File Details

The LCF configuration specifics are:

- **LightSource= "6400-40 LCF"**
When the light source is "6400-40 LCF" (as opposed to "Sun+Sky", or "6400-02B RedBlue", for example), three new levels of function keys are added to New Measurements mode for controlling the fluorometer. These are described in **Function Key Summary** on page 27-32.

  In addition, the Calibration Menu entry "_Light Source Calibration Menu"[6] will bring up a submenu of calibration routines that pertain specifically to the 6400-LCF. Details are given in **Display Summary** on page 27-29.

- **ComputeList= "Default FLR.LPL"**
This compute list adds a number of fluorescence computations. See **Logging Considerations** on page 27-33.

[6] This is new in OPEN 4.0, and replaces the entries "_LED Source - Calibrate" and "_LED Source - Plot curve"

- **Displays= "Std Flr Disp"**
  This display definition adds a number of display lines to the standard set. The quantities displayed are a combination of user variables defined in "Default FLR.LPL", and fluorescence system variables defined by OPEN 4.0. (For a discussion of system and user variables, see **OPEN's System Variables** on page 14-1, and **Defining User Variables** on page 15-1.)

- **LogFormat= "Std Flr Log"**
  Defines the items to be recorded when logging data. See **Logging Considerations** on page 27-33.

- **BLCond= "/sys/lib/BLCTable_LCF"**
  The boundary layer conductances of the LCF are a little higher than the standard chamber, and this information is contained in the file /sys/lib/BLCTable_LCF. See **LCF Boundary Layer Conductance** on page 27-78.

- **AREA=2.0**
  If the circular aperture is not covered with leaf material, use **Area=** (**f1** level 3) in new measurements mode then enter the correct leaf area.

- **FlrParams= "Std Flr Settings"**
  Specifies the initial fluorometer settings for fluorescence measurements, saturation flashes, and dark "pulses". Groups of settings can be stored in and retrieved from the directory /user/configs/FlrParams (see **Changing Control Parameters** on page 27-36). The file "Std Flr Settings" is there by default, but you can define and name others.

Programming Commands

The commands listed in Table 27-20 can be used in LPL programs (such as AutoPrograms) to perform fluorescence related tasks. For control commands, see **Leaf Chamber Fluorometer Control** on page 25-20 and **LCF Control Functions** on page 25-26.

Table 27-20. 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

Table 27-20. Fluorescence Commands(Continued)

Command	Description	Fct Key Equivalent
DoFm	Do a saturating flash, set Fm, compute user variables, and Log	f2 level 0 (when actinic off)
DoFmp	Do a saturating flash, set Fm , and Log	f0 level 0 (when actinic on)
DoFoFm	Set Fo, then DoFm	f3 level 0 (when actinic off)
DoFsFmp	Set Fs, do a saturating flash, set Fm , compute user variables, and Log	f3 level 0 (when actinic on)
DoFop	Do a dark pulse, set Fo , compute user variables, and Log	f1 level 0 (when actinic on)
DoFsFmpFop	Set Fs, saturating flash, set Fm , dark pulse, set Fo , compute and Log	f4 level 0 (when actinic on)
SetZero	Uses the current fluorescence signal as a future offset.	-none-
SetFs	Set Fs to current value of F	f2 level 0 (when actinic on)
SetFo	Set Fo to current value of F	f0 level 0 (when actinic off)
SetFm	Set Fm to max F value of latest flash	These have no fct key equivalents, but are performed as part of some fct key routines
SetFm_Prime	Set Fm to max F value of latest flash	
SetFo_Prime	Set Fo to min F 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	

Table 27-20. Fluorescence Commands(Continued)

Command	Description	Fct Key Equivalent
DoFm	Do a saturating flash, set Fm, compute user variables, and Log	f2 level 0 (when actinic off)
DoFmp	Do a saturating flash, set Fm , and Log	f0 level 0 (when actinic on)
DoFoFm	Set Fo, then DoFm	f3 level 0 (when actinic off)
DoFsFmp	Set Fs, do a saturating flash, set Fm , compute user variables, and Log	f3 level 0 (when actinic on)
DoFop	Do a dark pulse, set Fo , compute user variables, and Log	f1 level 0 (when actinic on)
DoFsFmpFop	Set Fs, saturating flash, set Fm , dark pulse, set Fo , compute and Log	f4 level 0 (when actinic on)
SetZero	Uses the current fluorescence signal as a future offset.	-none-
SetFs	Set Fs to current value of F	f2 level 0 (when actinic on)
SetFo	Set Fo to current value of F	f0 level 0 (when actinic off)
SetFm	Set Fm to max F value of latest flash	These have no fct key equivalents, but are performed as part of some fct key routines
SetFm_Prime	Set Fm to max F value of latest flash	
SetFo_Prime	Set Fo to min F 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	

Table 27-20. Fluorescence Commands(Continued)

Command	Description	Fct Key Equivalent
FlrRecordingOn	Open fluorescence recording (uses current flr recording file name, and appends)	-none-
FlrRecordingOff	Close fluorescence recording file	f5 level 8
FlrRecordingAsk	Prompt user to pick a new file for fluorescence recording, and if a file exists, will append.	

Fluorescence ComputeList

The default LCF compute list is named "Std Flr Comp", and is listed below

```
##"/user/configs/comps/Default"
##"/user/configs/comps/FlrOnly"
```

It essentially links two other compute lists together (this ability is new to OPEN 4.0). The first file, "Default", is listed in Figure 15-10 on page 15-18. 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 27-76) 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

$$\alpha = \frac{\alpha_{blue} B + \alpha_{red}(100 - B)}{100} \qquad (27\text{-}17)$$

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 27-21 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.

$$\alpha = \frac{\int_{\lambda_1}^{\lambda_2} S(\lambda)\alpha(\lambda)d\lambda}{\int_{\lambda_1}^{\lambda_2} S(\lambda)d\lambda} \qquad (27\text{-}18)$$

Table 27-21. 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 (27-11) on page 27-5) and Φ_{CO_2} (Equation (27-10) on page 27-5).

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 14-3 on page 14-18.

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 *m*. 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 **Fs is set.**
System variable *Fs* (display line *o*) is set to the current value of *F*.

2 **Normal events stop**
"Normal events" refers to the updating of new raw readings and system variables every second, and computing user variables every 2 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 *Fm'* (display line *o*) 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 27-34, 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 ex-

change 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 27-52 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 Fm. 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 9-9.

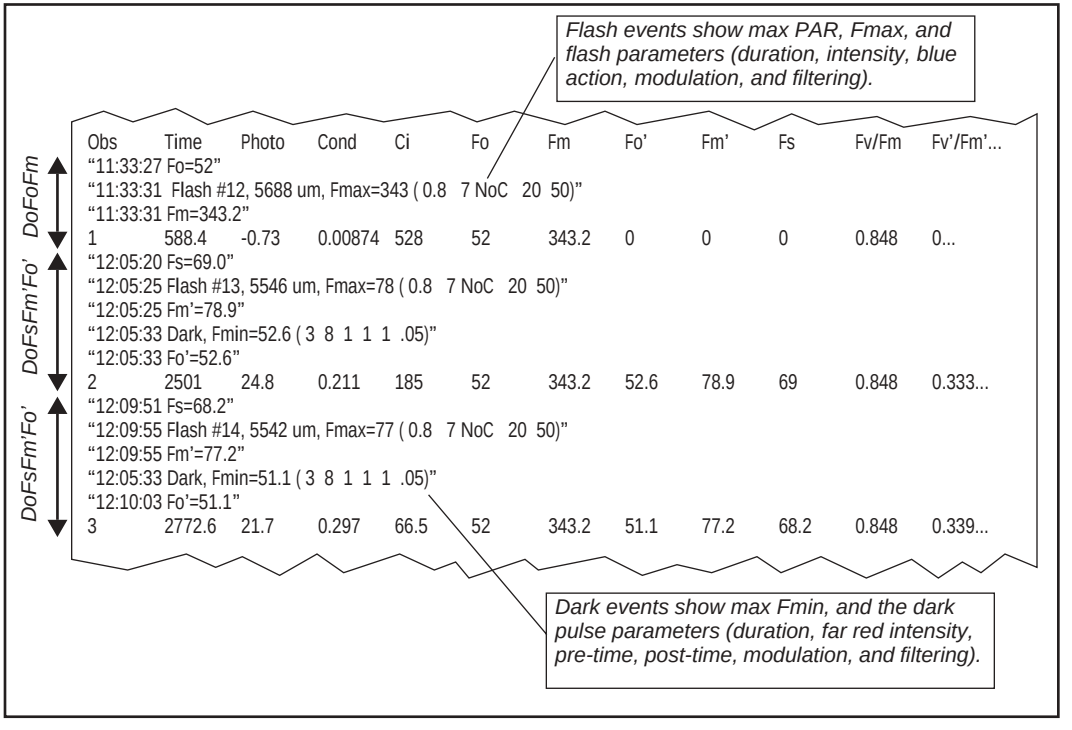


Figure 27-52. Sample lines from a data file showing remarks generated by fluorescent measurement events, and data records. At 11:33:27, a DoFoFm *was performed, causing* Fo *to be set. A saturating flash was completed at 11:33:31, and the value of* Fm *set accordingly. A data record was then logged, using gas exchange data from 11:33:27. Observations 2 and 3 were each triggered by a* DoFsFm'Fo' *event.*

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.

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. Biol. 42: 313-49.

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 Research 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 underestimate 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).

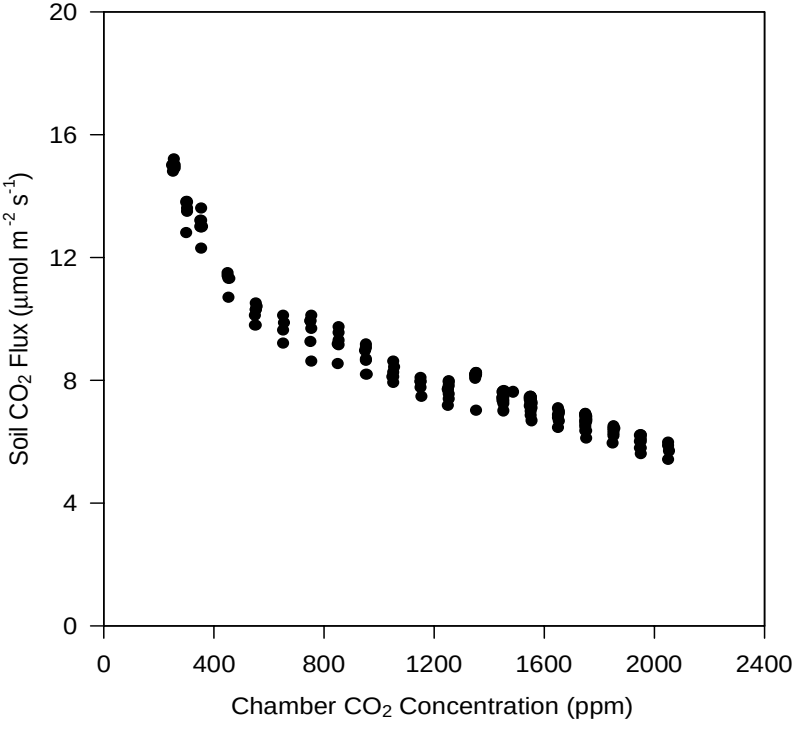


Figure 28-1. Soil CO_2 flux depends on the chamber CO_2 concentration.

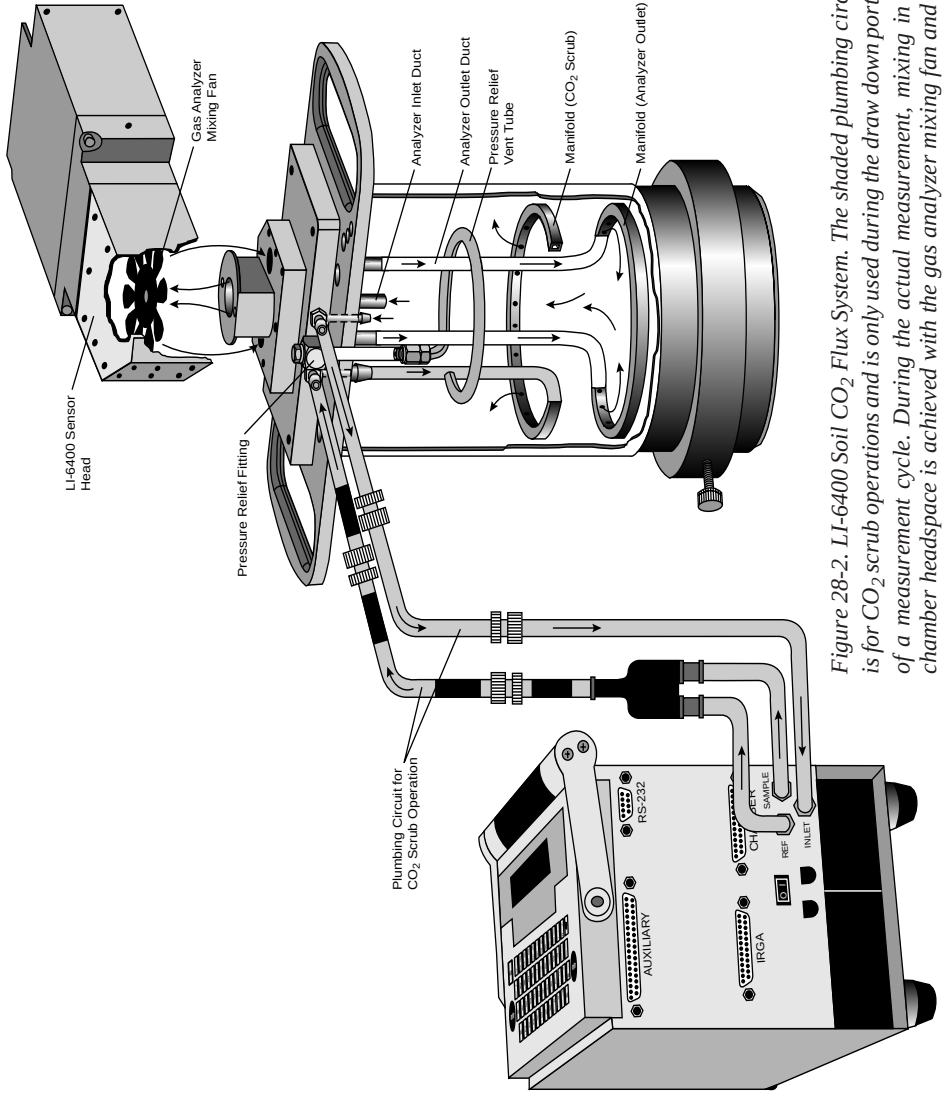


Figure 28-2. LI-6400 Soil CO_2 Flux System. The shaded plumbing circuit is for CO_2 scrub operations and is only used during the draw down portion of a measurement cycle. During the actual measurement, mixing in the chamber headspace is achieved with the gas analyzer mixing fan and the associated plumbing.

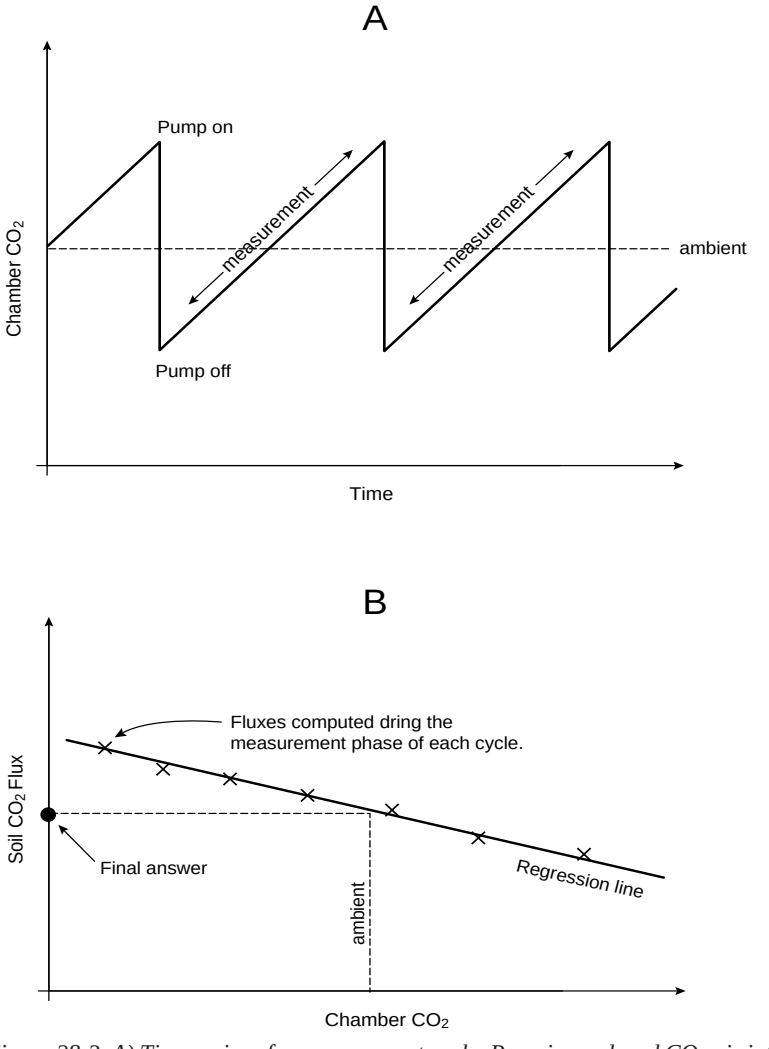


Figure 28-3. A) Time series of a measurement cycle. Pumping reduced CO_2 air into the chamber brings the CO_2 below ambient. After the pump turns off, CO_2 rises due to soil CO_2 efflux. During this phase, soil CO_2 flux is computed, and data for regressing flux as a function of CO_2 is generated. B) 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

1 **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

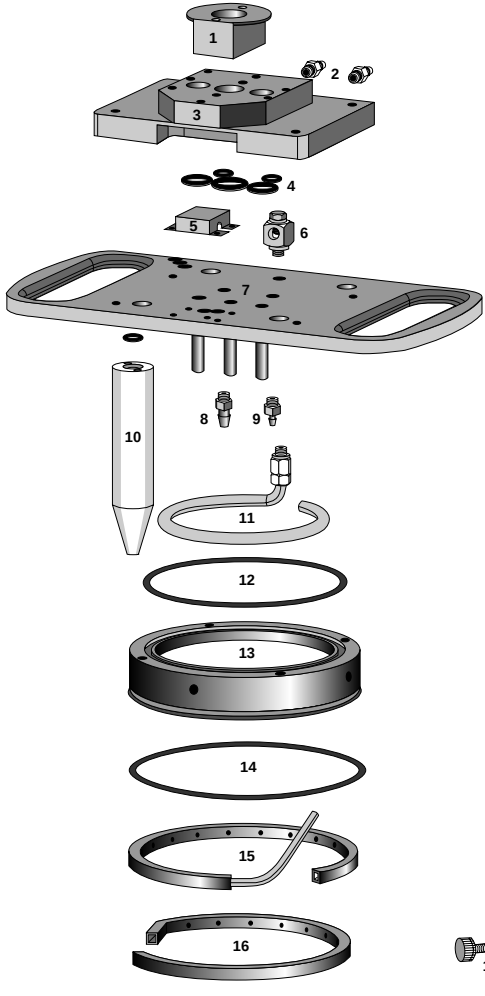
Healy, Richard W., R.G. Striegl, T.F. Russell, G.L. Hutchinson, and G.P. Livingston, 1996. Numerical Evaluation of Static-Chamber Measurements of Soil-Atmosphere Gas Exchange: Identification of Physical Processes. Soil Sci. Soc. Am. J. 60:740-747.

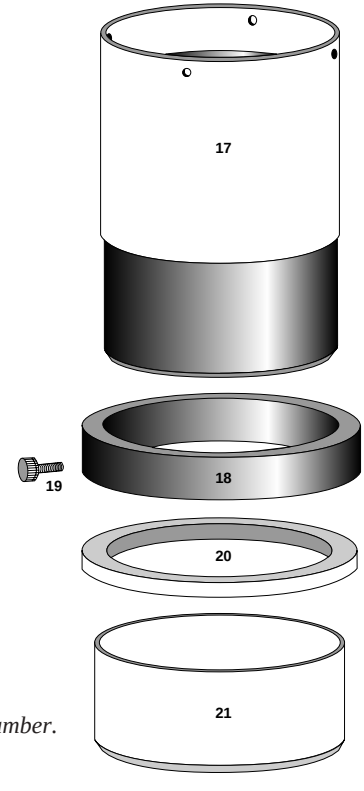
Attaching the Soil Chamber

General Description

Figure 28-4 shows 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





Parts List

#	Description		Part #
1	Fan Inlet Duct		
2	Hose Barbs		300-02547
3	Soil Chamber Adapter Manifold		
4	O-rings	Small	192-02597 (2)
		Larger	192-00225 (2)
		Largest	192-02889 (1)
5	Radiation Shield		6564-171
6	Pressure Relief Fitting		300-02561
7	Mounting Plate		
8	Hose Barb (to Air Supply Manifold #15)		300-02547
9	Hose Barb (open)		300-00567
10	Soil 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 PVC soil collar)		6560-229
21	PVC Soil Collar (optional)		6560-228

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.

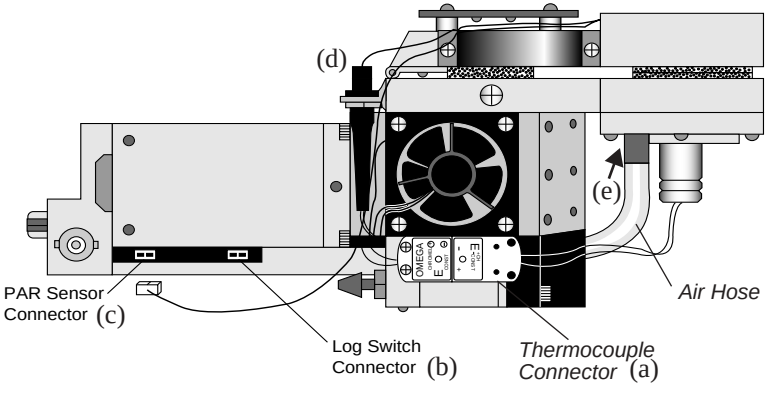


Figure 28-6. Preparing to remove the standard leaf chamber.

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 not 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 2-5.

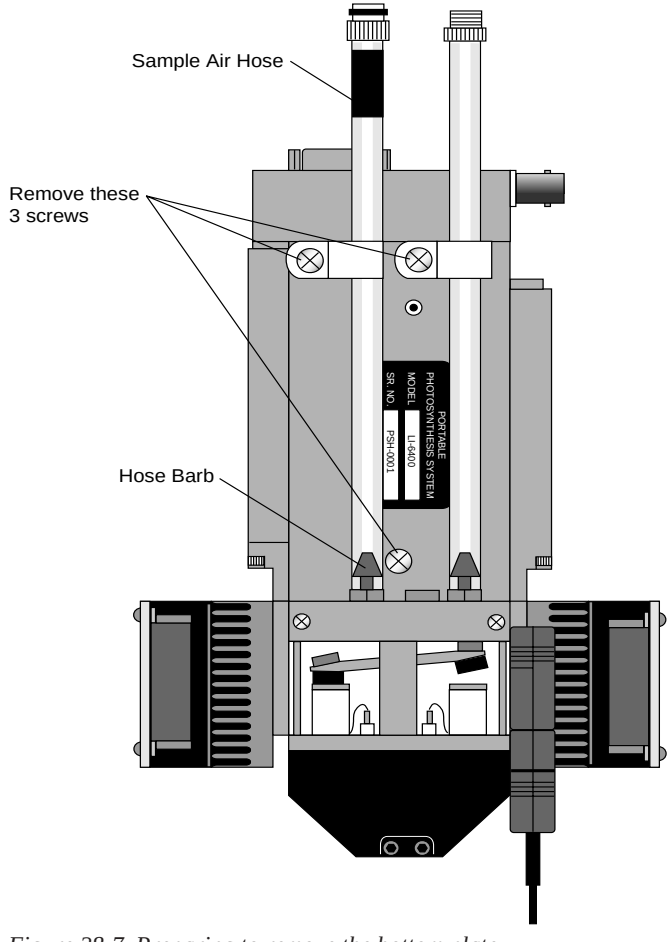


Figure 28-7. Preparing to remove the bottom plate.

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).

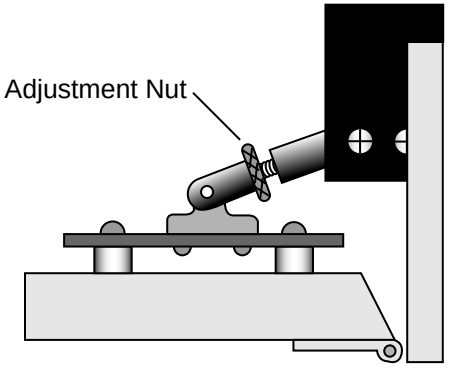


Figure 28-8. Turn the adjustment nut clockwise to remove.

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 2 screws (3 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.

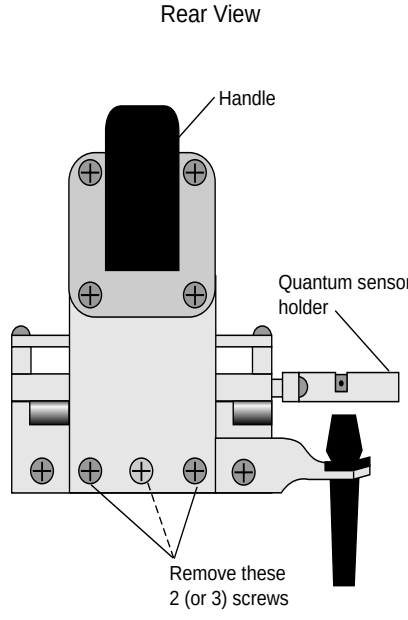


Figure 28-9. Remove the screws on the back side of the handle.

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).

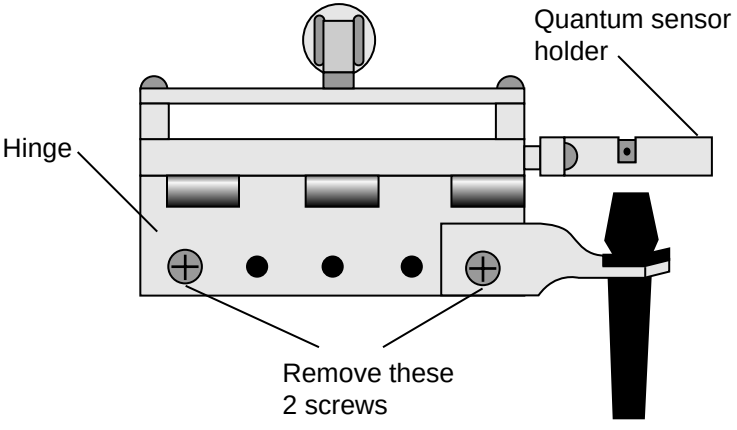


Figure 28-10. Remove the two screws from the handle hinge.

b) Remove one fan shroud screw and attach the lamp connector (Figure 28-11).

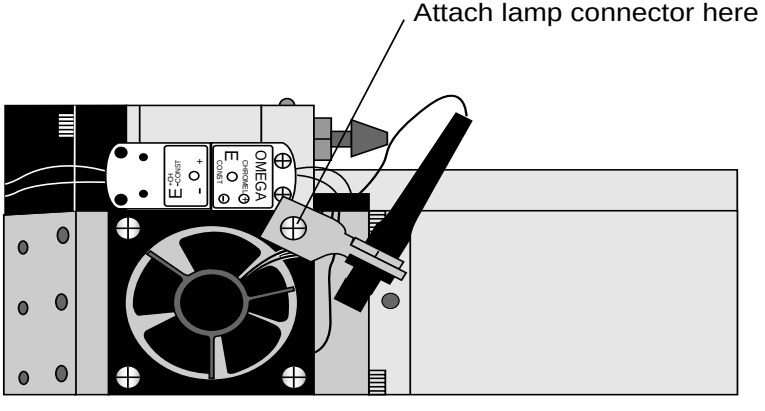


Figure 28-11. Attach the lamp connector to the fan shroud.

6 Remove the lower half of the chamber

There are 8 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.

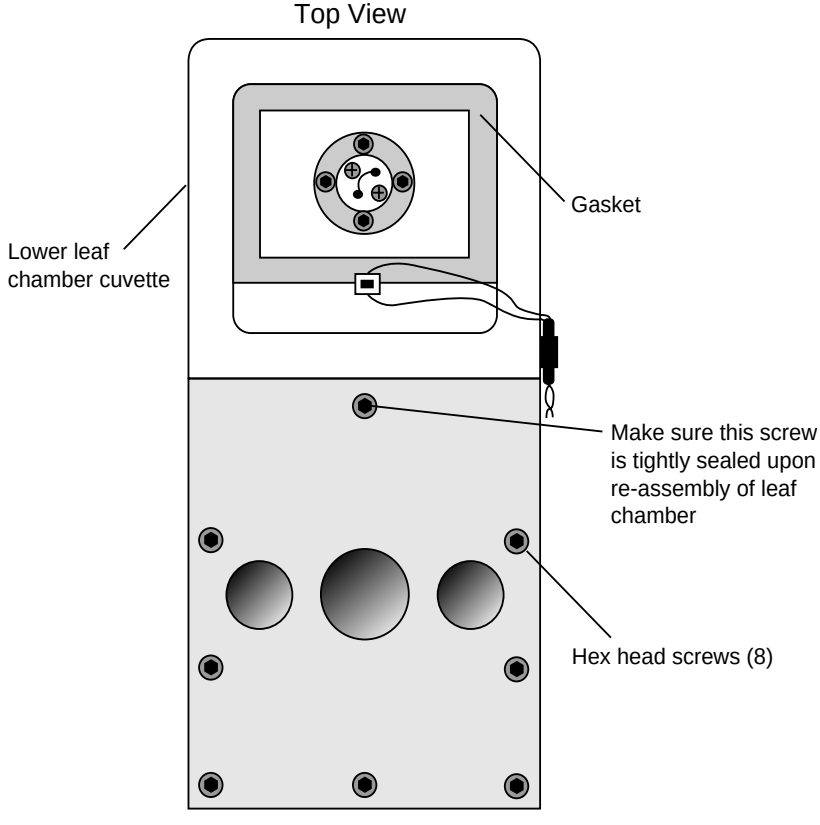


Figure 28-12. Remove the 8 hex head screws.

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*.

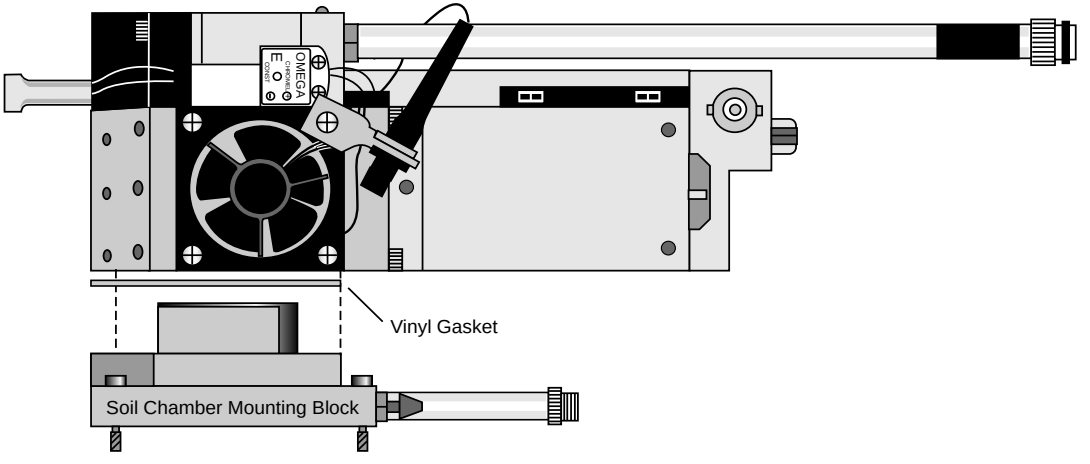


Figure 28-13. Attach the mounting block to the sensor head.

8 Check the O-rings

Make sure all O-rings are properly positioned, as shown in Figure 28-14.

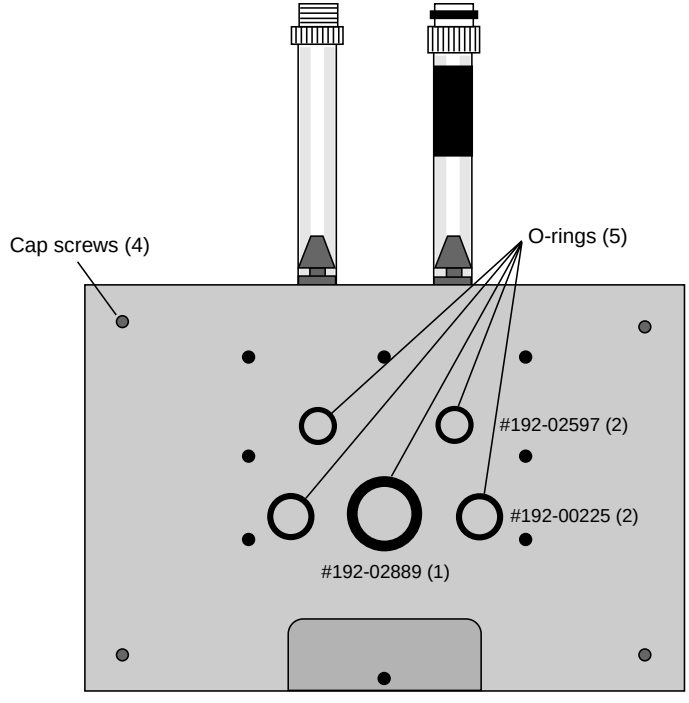


Figure 28-14. Location of the O-rings and cap screws.

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).

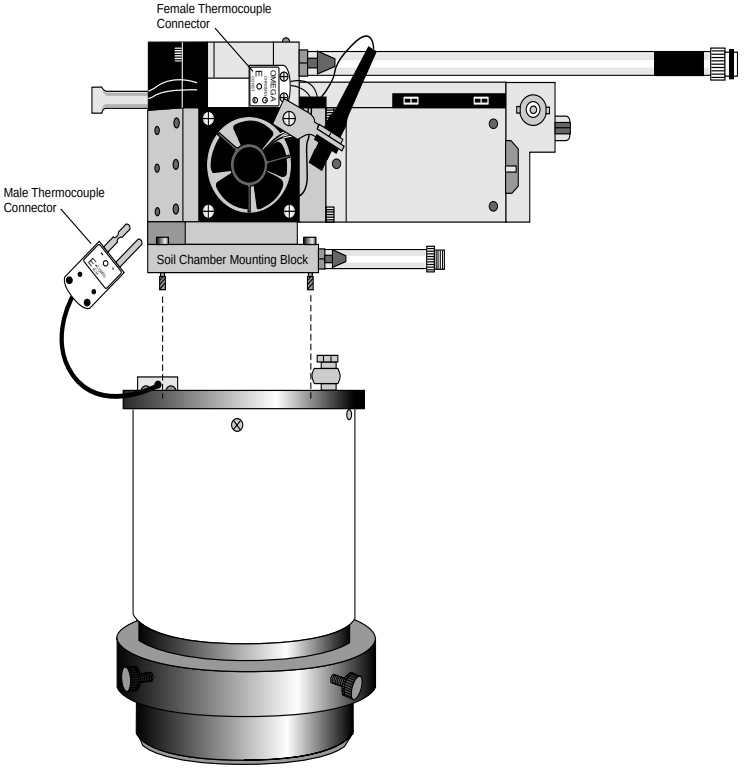


Figure 28-15. Attach the 6400-09 body to the mounting block.

10 **Connect 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.

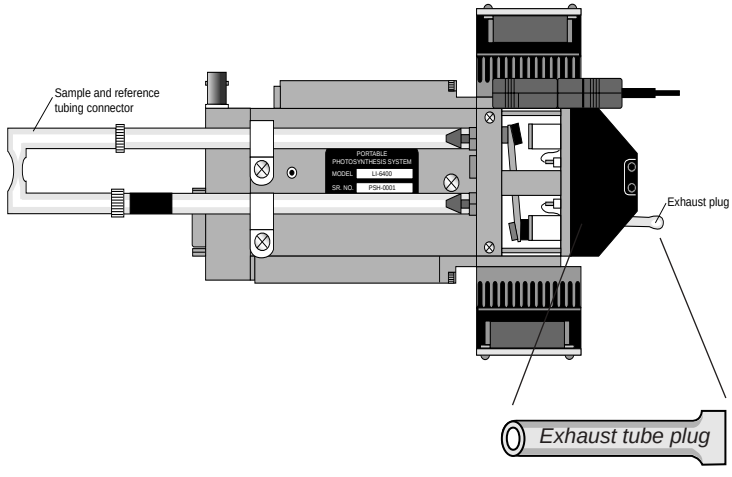


Figure 28-16. Insert exhaust plug, and sample and reference tube junction as shown.

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.

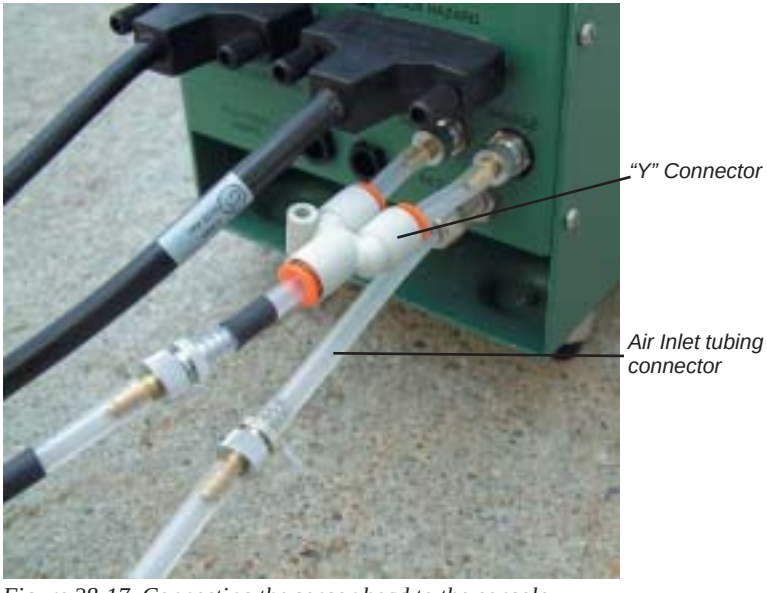


Figure 28-17. Connecting the sensor head to the console.

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 **12. 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.

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 -> _Reset Menu -> Reset to User Configuration).

Creating a Soil Chamber Configuration

1 **Access the Config Menu**
Press **f2** from OPEN's main screen.

2 **Select "Installation Menu"**

3 **Select "6400-09 Soil Chamber"**
You will be shown the opening screen (Figure 28-18).

```
Soil CO2 Efflux Chamber Installer

This program will
  a) create a config file
  b) install some support files

Press <enter> to continue_
```

Figure 28-18. The first screen of the soil chamber installer.

Press **enter**.

4 **View and store the config file**
The second screen will be the configuration file itself (Figure 28-19).

Press **Save** (**f2**) to bring up the Standard File Dialog to save the file. You may keep the default name (Soil Chamber), or change it as you wish.

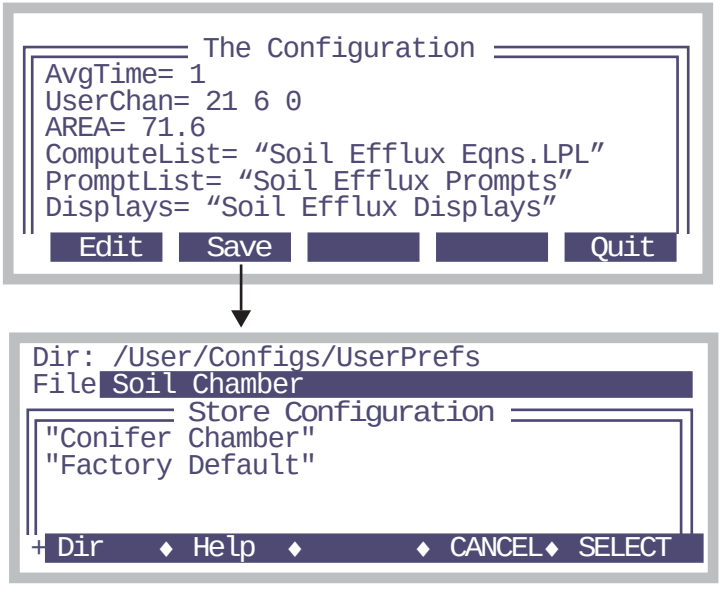


Figure 28-19. saving the configuration file for the soil chamber.

5 Exit the Installer Program

Press **Quit** (**f5**). A list of files will scroll by on the display, too quickly to read. These are the names of files being copied into the /User/Configs directory for use by the soil configuration. Finally, the standard post-install message will appear (Figure 28-20)

```
To implement any new config
that you might have created,
select it by
  Config Menu
    '_Reset Menu'
     'Reset to User Configuration'
```

Figure 28-20. Building a configuration file doesn't actually implement it. To do that, you select it when OPEN first runs, or else do a reset and select it then.

Press **enter** to return to the Installation Menu, followed by **escape** to return to the Config Menu.

Implementing the Soil Chamber Configuration

If OPEN is already running, follow these steps. If you are starting from power on, all you need is Step 4 below.

1 **Access the Config Menu.**
f2 from OPEN's Main Screen.

2 **Select "_Reset Menu"**

3 **Select "Reset to User Configuration"**

4 **Pick the Soil Chamber Configuration**
A list of all the configuration on your instrument is presented. Select "Soil Chamber" (or whatever you named it when you created it), and press **enter** or **Select** (**f5**).

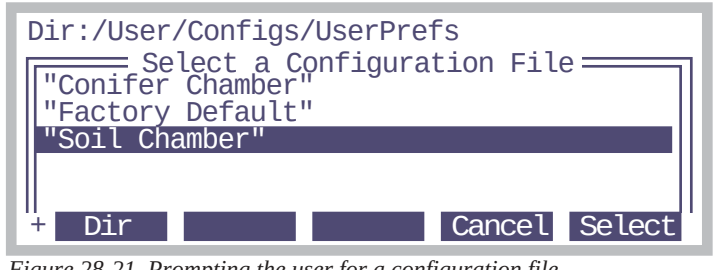


Figure 28-21. Prompting the user for a configuration file.

5 **Return to OPEN's Main Screen**
Press **escape** twice to get there.

OPEN's Main Screen

When configured for soil flux measurements, OPEN's Main Screen appears as in Figure 28-22. The only change from normal is the title.

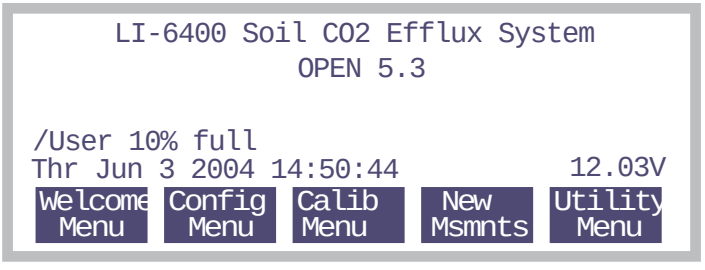


Figure 28-22. OPEN's Main Screen.

The Calibration Menu

While configured for soil flux measurements, the flow meter and reference IRGA are not used, so the Calibration Menu is a bit simplified (Figure 28-23)

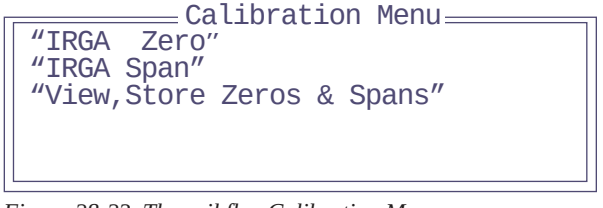


Figure 28-23. The soil flux Calibration Menu

Also, the zeroing and spanning routines are a bit different, and some re-plumbing is necessary. This is explained later in **Zeroing the IRGAs** on page 28-38.

New Measurement Mode

New Measurements mode has some new variables and function keys (Table 28-24 on page 28-24).

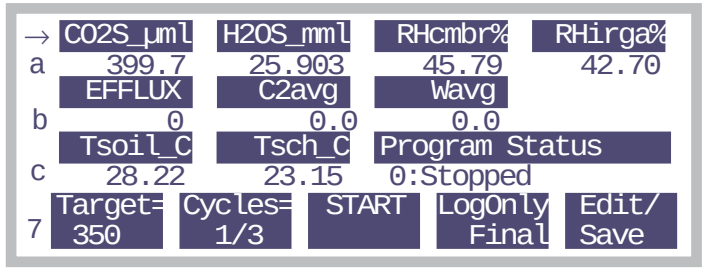


Figure 28-24. New Measurements with the soil flux configuration. Function key level 7 has most of the controls.

The function keys are described below, and the new variables are discussed in Table 28-1 on page 28-28.

Function Keys

Figure 28-25 shows the function keys in New Measurements while configured for soil CO_2 flux. The important group of keys is on level 7.

OPEN Version 5.2 and above

1	*No changes*	Open LogFile	<view file>	<close file>	<add remark>	Match
2	*f1-f3 disabled*				Temp OFF	Lamp -none-
3	*f1 sets a different variable* *f2 is disabled*	AREA= 71.6		LeafFan Fast	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	LogOnly Final	Edit/ Save

Differences for versions prior to 5.2

2	*f5 also disabled*				Temp OFF	
3	*f2 has auxiliary parameters*	AREA= 71.6	Aux Op Params	LeafFa Fast	Prompt off	Promp All
7	*f5 is different*	Target= 350	Cycles= 1/3	START	LogOnly Final	Depth= 0.0

Figure 28-25. The soil flux measurements are controlled from level 7.

The main difference in the soil configuration for versions prior to 5.2 is that the auxiliary parameters are on **f2** level 3, instead of **f5** level 7.

The following function keys are new - or have different meanings - for the soil configuration:

Target= (**f1** level 7,). Sets Target and Delta. 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 and a delta of 20, the measurement will occur while the CO_2 rises from 340 to 380. Once above 380, the pump will pull the CO_2 back down to 340 (minus any extra draw down ddMargin) for another cycle.

Cycles= (**f2** level 7). Sets NumCycls, 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.)

Log… (**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 28-26.

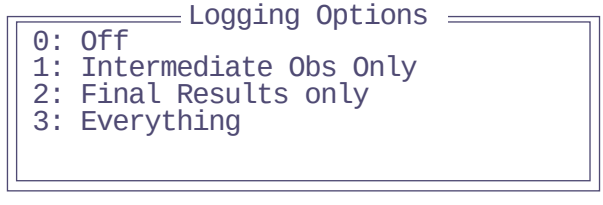


*Figure 28-26. Logging options (**f4** level 7)*

During the measurement cycle, the LI-6400 computes an intermediate observation (Mode = 3) every 2 seconds (2.5 seconds for versions 3.x and 4.x). 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 (Target + Delta), a final result (Mode = 4) record is computed by regressing the rates of change of CO_2 (dc'/dt) against the mean CO_2 concentration

(C2avg), and computing the CO_2 efflux rate a(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/Save (**f5** level 7).This key provides access to all the soil flux parameters, and allows them to be viewed, edited, stored and retrieved. The full list appears in Figure 28-27.

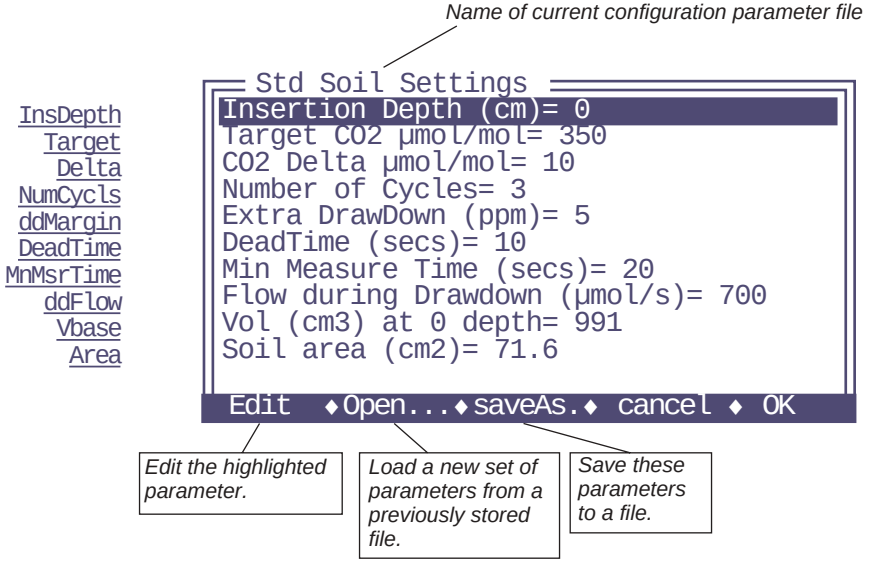


Figure 28-27. The **Edit/Save** *dialog (***f5** *level 7, version 5.2 and above only).*

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 28-1).

Table 28-1. User defined variables and constants for the Soil Configuration

ID #	Label	Description	How to Change[a]		LPL Variable name
			5.2 and up	5.1 and below	
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, computed by the software: *Vtot* = Vbase - Area * InsDepth			*soilChamSysVol*
305	InsDepth	Insertion depth (cm). Distance from chamber edge to the top of the soil surface. See Figure 28-28 on page 28-32.	**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 (cm3) at 0 depth.	**f5** level 7	Note[b]	*soilChamBaseVol*
308	Area	Soil area (cm2)	**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. Mode=4: Number of intermediate observations used for the final result.			*numSamps*

Table 28-1. User defined variables and constants for the Soil Configuration(Continued)

ID #	Label	Description	How to Change[a]		LPL Variable name
			5.2 and up	5.1 and below	
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.	**f5** level 7	**f2** level 3	*ddSafetyMargin*
316	DeadTime	Dead Time (secs). Minimum time between when the pump turns off, and when measurements begin. See Figure 28-29 on page 28-33, part #2.			*postPumpDeadSecs*
317	MnMsrTime	Min Measure Time (secs). Minimum length of the Mode 3 part of a measurement. See Figure 28-29 on page 28-33, part #3.			*minMeasureTime*
318	NumCycls	Number of cycles to do. 1 cycle is drawdown (Mode=1) through final result (Mode=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	Mode=3: Flux for intermediate observation Mode=4: Flux for final result. See Equation (28-11) on page 28-41.			*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	Mode=3: Meaningless Mode=4: Slope of dc'/dt vs. *C2avg*			*dcdtSlope*
331	R(c)b	Mode=3: Meaningless Mode=4: Intercept of dc'/dt vs. *C2avg*.			*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 the advantages of not using collars are those of versatility and spontaneity in choosing measurement locations, 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 cm2 (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 CO_2 Flux Chamber.

3 Set Target and Delta

In New Measurements mode, press **f1** level 7 to set the Target and Delta values.Use the ambient CO_2 concentration determined in Step 1 as the Target, and choose a Delta appropriate for your site. For low rates, the Delta 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, *InsDepth* will be a positive value, such as 1 or 2 cm, depending on the soil type and the Stop ring position (Figure 28-28A). With soil collars (Figure 28-28B), 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.

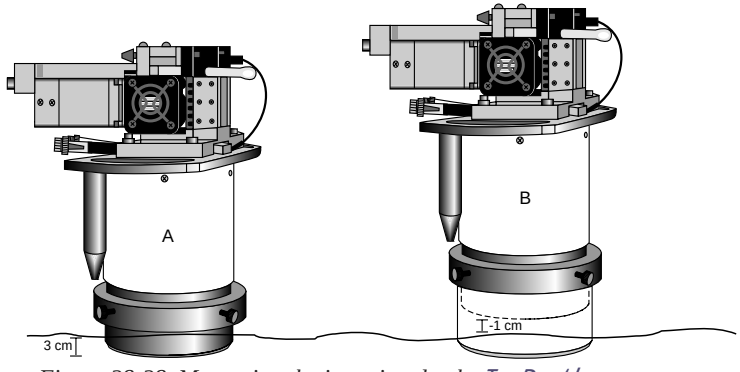


Figure 28-28. Measuring the insertion depth, InsDepth

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?

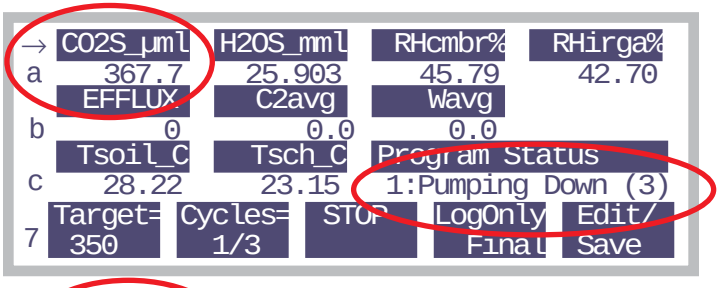
If you wish to be prompted for some things (location, Area, InsDepth, etc.) automatically at the start of each measurement, set that up by:
a) Prompt Control in the Config Menu to set up the item list.
b) **f4** level 3 to turn Prompts ON.

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.

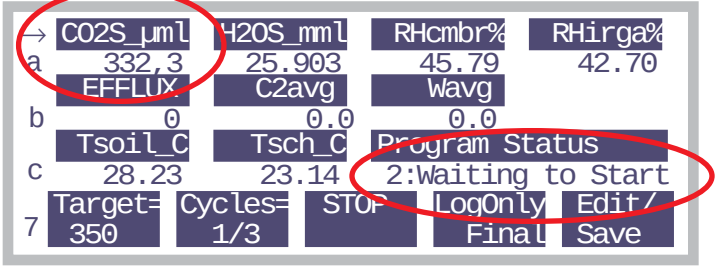
The measurement cycle will begin (Figure 28-29).

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.



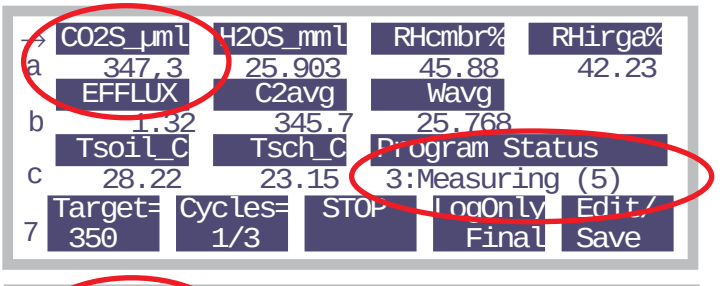
2. CO2S_µml is now low enough, so the pump turns off. We are waiting for the value of CO2S_µml to rise above Target - Delta, so the measurement can begin.

There is a minimum time for this wait, set by the parameter DeadTime.



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.



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.

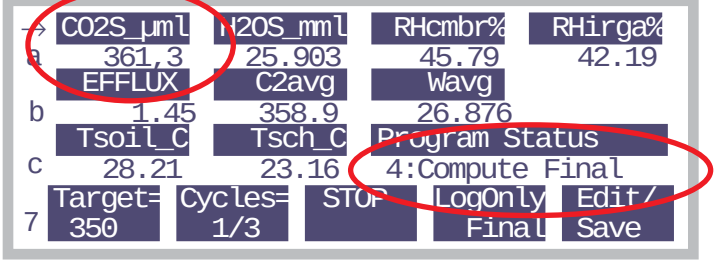
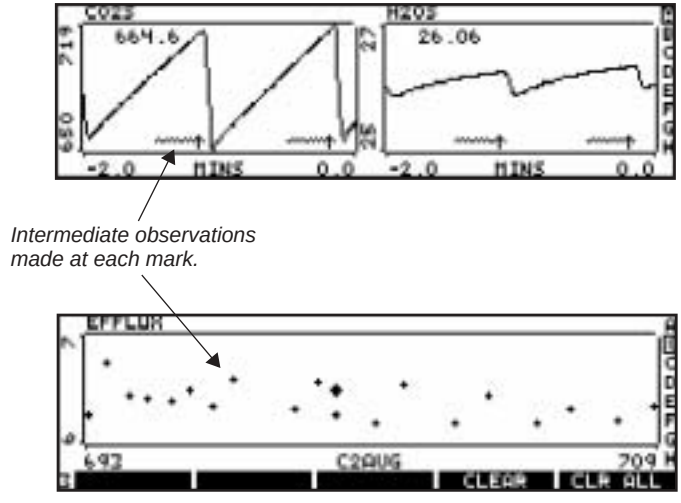


Figure 28-29. The four stages in a measurement cycle.

Real Time Graphics

The default soil chamber configuration has some Real Time Graphs (**Real Time Graphics** on page 6-11) defined that are useful (Figure 28-30). 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 28-30. 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 9-20), and it will set the targets and begin the measurements automatically to cover the range you specified.

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 (Target - Delta), 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 (Target + Delta), 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 4 parameters (accessible by **f5** level 7, or on old software, **Aux Params**, **f2** level 3) that influence the pump's behavior.

  ddMargin ("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.
  DeadTime ("Dead Time (secs)") prevents measurements from starting too soon after the pump turns off.
  MnMsrTime ("Min Measure Time (secs)") is the minimum measurements time, and prevents the pump from turning on again too soon.
  ddFlow ("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 fully 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 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.

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 28-2. 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)

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 2 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 section 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).

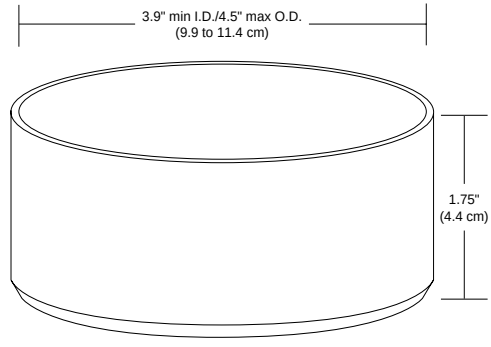


Figure 28-31. Soil collar dimensions.

Zeroing the IRGAs

In order to zero the sample analyzer, it is necessary to re-plumb the system.

Plumbing changes for zeroing the IRGA:

1 Move the "To Chamber" air line from the chamber to the IRGA

The "to chamber" air line has black heat shrink on it. This needs to be moved from the connector that goes into the chamber to it's normal (leaf chamber configuration) location on the sensor head (Figure 28-32).

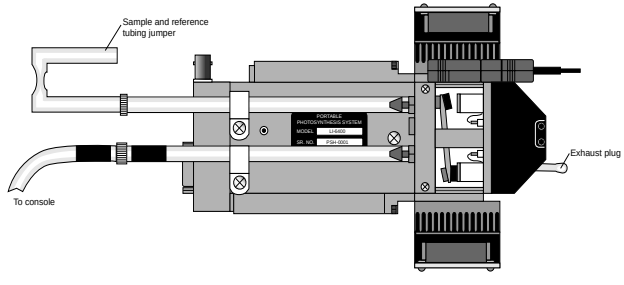


Figure 28-32. The "to chamber" line attached to the sensor head, for zeroing purposes.

2 Cover the open end of the soil chamber

Either put the plastic cap back in place, or set it down on a smooth surface. The goal is to prevent higher concentration CO_2 (such as from your breath) from finding its way into the chamber, and diffusing into the IRGA. During

zeroing, the system will have the fan shut off, so air from the soil chamber will not be actively exchanged with air in the optical path of the sample analyzer. The system is now plumbed for zeroing.

3 **Soda lime and desiccant on full scrub**
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.

Setting the IRGA Span

1 **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 **Cover the open end of the soil chamber**
This is necessary for the sample IRGA. Either put the plastic cap back in place, or set the chamber down on a smooth surface to prevent large CO2 concentrations from finding their way into the chamber, and diffusing into the IRGA.

During spanning, the system will have the fan shut off (regardless of where you have the fan speed set), so air from the soil chamber will not be actively exchanged with the air in the optical path of the sample analyzer.

The system is now ready for checking the span.

Equation Derivation

The mass balance of CO_2 for the 6400-09 Soil Flux Chamber (Figure 28-33) is given by

$$CO_2In = Strorage + CO_2Out$$

$$sf_c = \rho v\frac{\partial c}{\partial t} + uc \qquad (28\text{-}1)$$

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.

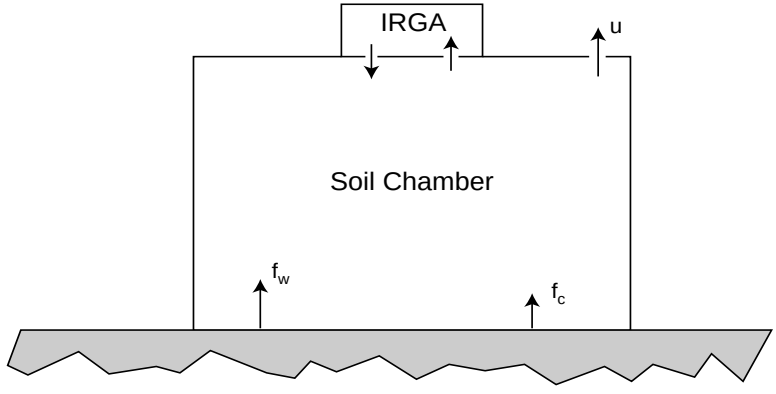


Figure 28-33. Schematic of the soil chamber. Soil evaporation f_w and soil CO2 flux f_c add mass to the chamber air, which is balanced by flow u (mixture of air, water, and CO_2) out of the chamber.

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_20 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 the equation implemented in the LI-6400.

$$F_c = \frac{kPV}{S(T+273)}\left(\frac{\partial C}{\partial t} + \frac{C}{(1000-W)}\frac{\partial W}{\partial t}\right) \qquad (28\text{-}11)$$

where k = 10 / 8.314 = 1.2028.

Table 28-3. 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_µml	-2
W	H2O concentration	mmol H_2O mol^{-1}	H2OS_mml	-5
$\frac{kP}{(T_c + 273)}\left(\frac{\partial C}{\partial t} + \frac{C}{(1000 - W)}\frac{\partial W}{\partial t}\right)$			dc'/dt	306

Partial Listing of Soil Efflux Eqns.LPL

The compute list file for the soil configuration is "/User/Configs/Comps/Soil Efflux Eqns.LPL". It is a lengthy file, defining a lot of the special behavior necessary for the closed system measurement protocol. Two sections are shown here: variable declarations (Figure 28-34) and the actual flux computations (Figure 28-35 on page 28-44).

```
/* -- external -- */
:STATIC 0
:FLOAT
  PUB soilChamBaseVol 991.0 /* cm3 */
  PUB soilInsDepth  0.0     /* cm */
  PUB targetCO2     350
  PUB deltaCO2     10
  PUB ddFlow       700
  PUB ddSafetyMargin  5 /* ppm */
  PUB soilArea    71.6  /* cm2 */
:INT
  PUB numCycles   3
  PUB postPumpDeadSecs 10
  PUB minMeasureTime  20

:FLOAT
  /* computed */
  soilChamSysVol  991.0
  soilEfflux    0
  dcdtSlope    0
  dcdtOffset    0
  co2Slope    0
  h2oSlope    0
  dcdt      0
  co2Mean      0
  h2oMean      0
  rhsc      0
  tSoil      0
  logInt_Secs    2
  prevLog_Secs  0

  PUB soilSTPeriod 10

:INT  numSamps    0
  opMode      0
  PUB  cycleNum   1
```

Figure 28-34. Variable declarations, about midway through the file.

```
: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 tLeaf_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
{
  0 :FLOAT density
  0 :FLOAT va
  $
  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
  THEN
  soilEfflux = dcdt * va
}
```

Figure 28-35. Flux computations at the end of the file. For an explanation of the format, see ***New Style ComputeList Files*** *on page 26-13.*

Hooks used by Soil Efflux Eqns.LPL

The soil configuration makes use of a number of the hooks provided by OPEN (**Open's Hooks** on page 26-7). These are shown below.

UserMainDisplay

This hook is called when printing the OPEN's main screen. We use it to replace the title.

```
/* --- hooks --- */

/* Main screen hook
*/
UserMainDisplay
{
  openRevID "    LI-6400 Soil CO2 Efflux System

              OPEN %s\n" PRINT
}
```

UserCalibMenu

This intercepts the Calib Menu from the Main Screen, substituting a different directory.

```
/* Calib menu hook
*/
UserCalibMenu
{
  "Soil Calib Menu" "/Sys/Closed/Soil Cal Menu"
RunReadFromDirectory
}
```

UserPreComps, UserPostComps, UserNewReadings

These hooks are used to carry out processing and mode switching.

```
/* Before and after user comps hooks
*/
UserPreComps { CompPrep PreFct }
UserPostComps { PostFct }

/* Called when new readings are ready
*/
UserNewReadings
{
  SoilActionFct
}
```

DefUserKeys
This hook is used for all the function key modifications.

```
/* Called when defining New Measurements fct keys
*/
DefUserKeys
{
  /* Disable some keys
  */
  matchKey OFFSOFT
  area_key OFFSOFT
  stomrat_key OFFSOFT
  range_key OFFSOFT
  lamp_ctrl_key OFFSOFT

  DefAreaKey
  DefTargetKey
  DefCyclesKey
  DefStartKey
  DefLogWhatKey
  DefEditSaveKey

}
```

UserEverySec
Called each time the clock tics (every second).

```
/* The new EverySec function
*/
UserEverySec
{
  UpdateClock
  opMode IF
    &modeTimeBase_Secs 1 + VAL srBase "%s (%d)"
to_string srState =
  THEN
}
```

UserWatchDog
Called every 10 seconds, this hook adjusts what warnings are displayed.

```
/* The new watch dog function
*/
UserWatchDog
{
  WMFuse NOT IF
    WMIrgas NOT IF
```

```
      warningMess 0 SETREADY
      rhout 95 > IF
        CoolGetTarget SWAP DROP 1 == IF
          tCham_c 3 + 2 CoolSetNewTarget
        ELSE
              "    >> HIGH HUMIDITY ALERT <<"
warningMess =
        THEN
      THEN
    THEN
    THEN
    DisplayWarning
}
```

UserConfigInit
This is called when the soil config is selected for use.

```
/* On selection
*/
UserConfigInit
{
  -3600 18 AOSET

  _ReadSoilParams
  iIdleMode SetMode

}
```

UserNMEnter, UserNMStartUp, UserNMExit
These hooks are called when entering and leaving New Measurements Mode.

```
UserNMEnter
{
  statTrackerStatus NOT IF
    OpenVarNameList openSysID STNEW &SoilST =
    soilSTPeriod -2 SoilST STADD
    soilSTPeriod -5 SoilST STADD
    1 &statTrackerStatus =

  THEN
  hiResUpdateTime SoilST STRESET

  5 ChFanSet
  IsCo2MixerOn IF
    Co2MixerOff
  THEN
  0 1 FlowSetNewTarget
  stdPump_mv &pump_mv =
```

```
}


/* Called when new measure mode entered
*/
UserNMStartup
{
  /* shut down some controls
     (comment these lines out
     to keep these controls active)
  */

  0 &allowFlowControl =
  0 &allowMixerControl =
  0 &allowLampControl =

}

UserNMExit
{
  iIdleMode SetMode

  statTrackerStatus IF
    SoilST STFREE
    0 &statTrackerStatus =
  THEN

  1 RETURN /* allow exit */
}
```

Soil Chamber Specifications

System Volume (0 insertion depth): 991 cm3

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 T 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")

Weight: 1.8 kg (3.75 lbs.)

*Specifications subject to change without notice.

INDEX

Part Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z

Index