Continuum Laser Operation

Eric L. Michelsen 4/5/2006 10:37:00

Here's what I know about laser programming so far. The laser has a programming model you must understand to configure and store programs. I have determined its characteristics, register model, and configuration state machine by 2 days of experiment.

Register Model

The laser has 3 parameters you can set:

- Repetition rate (10-20 Hz, in 1 Hz increments): flash lamp firing rate
- Q-switch delay (1 to 1000 μs, in 1 μs increments): delay between flash lamp firing, and tuning the cavity to resonance, which extracts the energy from the population inversion of electrons
- Pulse division ratio (1 to 10, in increments of 1, and "external trigger"): number of flashes for each output of the laser (i.e., for each cavity dump). This allows you to generate laser pulses at a submultiple of the flash lamp repetition rate.

The laser has 16 "programs" which record values for each of these parameters. A program is simply a set of 3 values, one for each parameter. There are 16 programs, numbered 1-16. Programs 1-3 are fixed at the factory, and cannot be changes. Programs 4-16 are "user" programs, and can be set to any valid values. See diagram below. All programs are non-volatile, and retained across power cycles.

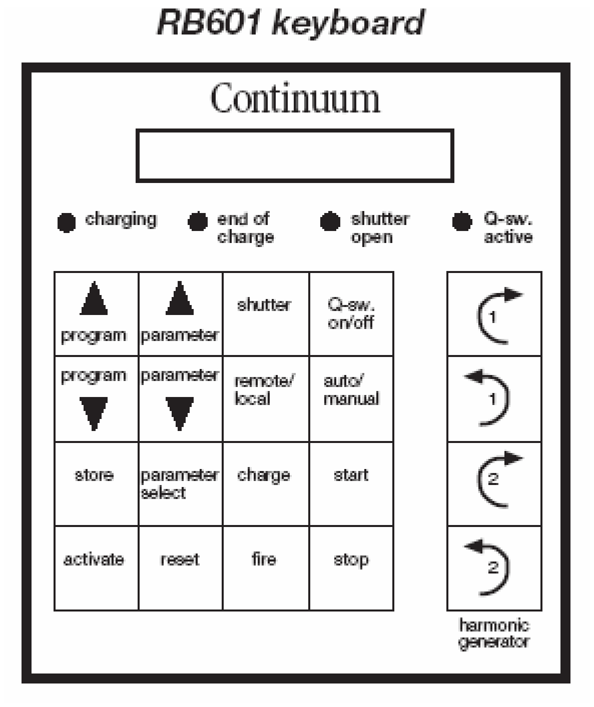
At every moment, the laser has a set of "active" parameters, which are the ones currently running the hardware. Note that through a quirk, it is possible that the active parameters are not those of any of the 16 programs. More on this later.

Besides the 16 programs, there are 2 other registers that hold laser parameters: I call them the "display register" and the "active register." The display register holds temporary values while you are changing them. The active register holds the values the laser is actually using for operation. Besides the 3 laser parameters, the display and active registers also hold the alleged program number with which the parameters are associated. The parameters may or may not agree with what is actually currently stored in those programs.

Thus, there are 18 registers in the laser: 16 programs, a display register, and an Active register.

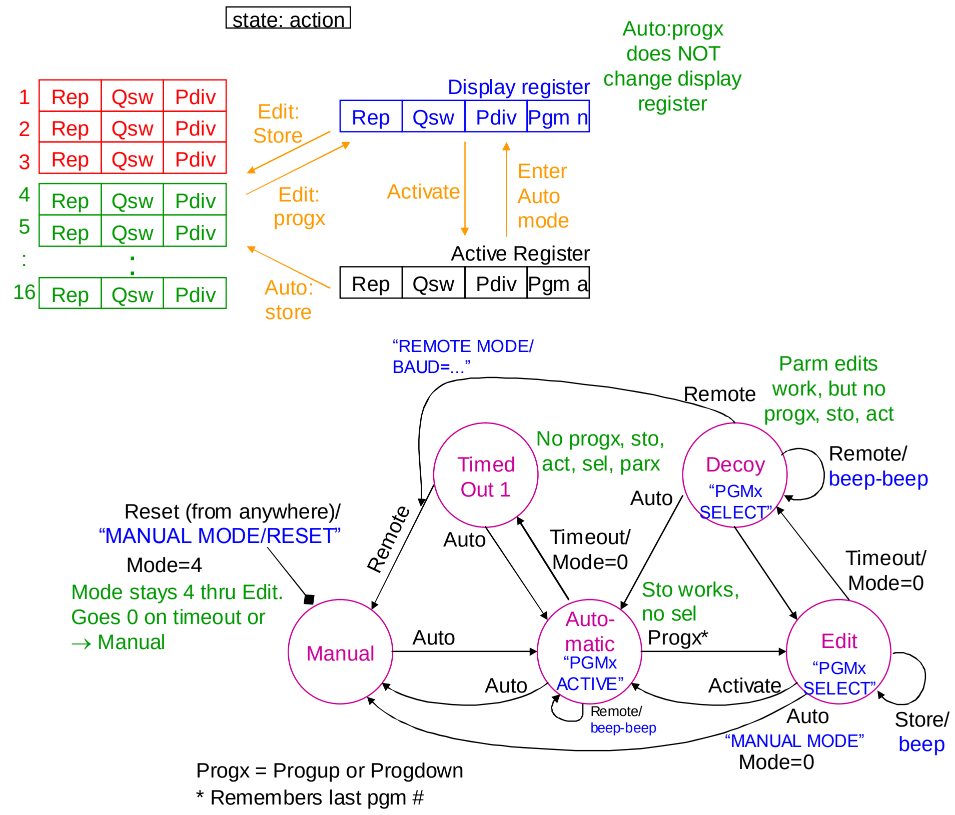
Programming Sequence

You push buttons on the control box to program the laser:



The state machine's behavior is weird, inconsistent, and unreasonable, so don't try to make too much sense of it. See diagram below [Make it bigger?].

Laser Programming: Storage and State Machine



In the following description, square brackets indicate [buttons] on the control box. The laser resets into the "Manual" state. To program it, push [Auto/Manual] (hereafter abbreviated [Auto]). Do not confuse the "Automatic" state with the [Auto] button. From here you can [Start] the laser, or [Stop] it.

To change programs or the active parameters, press [Program up] or [Program down] (hereafter abbreviated [Progx]). The control box shows the display register, along with nn, the last selected program number: "PGMnn/SELECT". If the display register was edited in the past, but not [Store]d, then the display register will be different than the currently stored program parameters. Pressing [Progx] increments or decrements nn, and copies program nn to the display register. You can edit the 3 parameters with the [Parameter select] button, and the [Parameter up] and [Parameter down] buttons. Note that going from 1 µs Q-switch delay to 1000 µs takes a long time.

When you're done with your edits, pressing [Store] copies the display register to Program nn. Pressing [Activate] copies the display register to the active register, and the parameters become the active parameters. Note that you can [Activate] without [Store]ing, and then the active parameters are not those of any stored program.

Timeouts

The most annoying part of the human/laser interface is the timeouts. When editing program parameters ("Edit" state), if you pause for ~5 seconds, the system "times out," and enters "Decoy" state. It gives no notice of the timeout, though, so you can't ever be sure when it happens. Worse, you can still edit parameters after a timeout, so things *look* normal, but the laser ignores any [Store] or [Activate] buttons. The only way out of "Decoy" state is [Auto], which takes you back to "Auto" state. Then [Progx] takes you back to "Edit" state.

When in “Auto” state, there’s another timeout of ~5 sec. It is also silent, so you can’t tell exactly when it happens. After timeout, you can’t enter [Progx]. Again, the way back to “Auto” state is [Auto].

If you get confused (how could that happen?), pressing [Auto] in “Auto” state takes you to “Manual” state, and the display shows “Manual Mode.” At least you can tell when this happens from the display.

Remote Programming Interface

The designated remote access serial interface is quite bogus, and I doubt anyone has ever used it successfully. The serial port is a typical (asynchronous) DTE, but on a DB-25 *female* socket (Most DTEs are male). Asynchronous parameters are fixed at 8N1 (8 data bits, no parity, 1 stop bit). It requires a null-modem cable to talk directly to a PC.

It supplies RTS, and requires CTS to transmit. However, it does not appear to use RTS to regulate data into it (no flow control). The manual (Rev C, p 3-26) says ‘If no host computer is interfaced to the RS232 connector, the CU601C will “time out” after 5 seconds, beep and do a “warm reset”.’ So far as we can tell, there is no timeout. However, it is possible that the timeout only applies if the laser is running.

The remote programming interface is primarily just a remote button pusher. Each button on the control box has a code, and sending that code into the serial interface is equivalent to pushing that button on the control box. The state machine (including bogus timeouts) is identical. The control box display follows the activity of the remote interface. In addition to buttons, there are 2 “inquiry” commands: CU status, and Program Status, which are “documented” (and I use the word loosely) on pp 2-16 and 2-17 of the Powerlite laser manual, Rev C. They are totally undocumented in the Leopard manual.

Characters cannot be sent at line speed, and there appears to be no buffering at all. It seems we must put delay between each character. 150 ms fails miserably (misses ½ the characters). 175 ms seems to work; Linux probably rounds this up to 180 ms. We should target around 2x minimum delay (~360 ms) for reliability.

Since the control box display is not available on the Remote Programming Interface, programming can only be done by dead reckoning from a Manual Mode Reset state. There is no way to read the current mode (Auto/Manual), or know the current Active Program number. You can read the Active parameters, though.

The following sequence allows reading, changing, and activating any program by dead reckoning:

[Reset]	Forces manual mode
[Auto]	Enters “Automatic” state
[Progx]	Enters “Edit” state
[Program down] 16 times	Forces into Program 1, because the laser pegs at Program 1.
[Program up] nn-1 times	Selects to program nn
[Activate]	Activates this program, because you can only read the active parameters
[Program Status]	Reads active parameters
[Parameter Select]	Selects Repetition rate

...

You can now navigate through the 3 parameters, change them, store them, and activate them.

Keyboard Box

Since the designated remote interface is so limited, we hacked the private interface between the keyboard box (which Continuum calls the “remote box”) and the CU. The interface is RS-232. Ohming out the cable from inside the local control box: it connects to J1 on the board, with the square pad identifying pin 1. Two wires come from pin 5. The cable terminates in a proprietary DB-9 plug, which goes directly into the CU DB-9 socket. The wires in the cable are

J1 Pin	Cable Wire Color	DB-9 Male Pin	Function
1	red	1	+12V power
2	white	4	RS-232 Box to CU
3	brown	3, 6, 9	Ground
4	blue	2	RS-232 CU to Box
5	green and black	3, 6, 9	Ground

Opening the DB-9 shell, I see there is no connection to pins 5, 7, or 8. I cannot see the color of the wires, because they are covered in heat-shrink tubing. Inside the CU, pin 9 is frame ground, so signals should reference pins 3 or 6, not pin 9. This is confirmed on p3-30 of the Leopard manual 996-0254 Rev C, and I verified it visually by opening the CU.

Scoping J1 on the box reveals pin 2 is Box-to-CU data, using RS-232, 1200 baud, 7 data bits, even parity. The number of stop bits is indeterminate, but almost certainly 1. However, the data from CU-to-box is 8N, not 7E.

The board has a 7805 voltage regulator, and an 80C51 microcontroller, apparently UV erasable (we could feel the window under the sticker) .

The box is a dumb keyboard/display unit; the CU controls everything and stores all the programs. The box sends each key press to the CU as a character, as described for the remote interface. The code is just row/column encoded into the H/L nybbles of each byte. For example, STORE is 0x31.

On power up, the Box generates its own “Remote Box OK” message; it is not sent from the CU

The CU sends a binary code of bytes commanding the box to do display things, and sometimes beep. There seem to be several additional codes that we don’t understand, but they don’t seem important. The box has a 2 line by 20 character display. We have hacked out the codes that control the display (all bytes in hex):

Code (hex)	Argument	Description
00	1 byte	Unknown, but only appears near the beginning of power up as 00 7F. This may be some kind of protocol id or capabilities mask.
02	variable ASCII string	Write display text to the current cursor position. String terminated by 03 (ASCII CANCEL character)
04	none	clear the display
05	1 byte	Position cursor: the 0x40 bit = 0 for line 1, =1 for line 2; the low 5 bits specify the horizontal position, starting at 0.
07	1 byte	01 = start beep. 00 = stop beep.
08	1 byte	Unknown. Rare, usually seen as 08 01, which happens after RESET or STOP.
09	none	appears to be a keep alive
11	1 byte	Q-switch active LED (unknown color, because it’s always off): 00/01 = off/on
12	1 byte	Shutter open LED (green): 00/01 = off/on

13	1 byte	End of Charge LED (green): 00/01 = off/on
14	1 byte	Charging LED (red): 00/01 = off/on
15	1 byte	unknown. The argument is almost always 00.

A test program processed a log file of the CU->Box codes, and successfully reproduced the display screen. Perhaps some of the codes return display characters to the CU? Perhaps some are simply diagnostics, and have no essential function.

The system powers up in Local (local keypad) input state. A human must push the Remote button to enable remote serial port access. We'd like the system to be controllable remotely directly from power up, because it's a big burden for operators to have to push a button. (Other systems, e.g. the bolometer, have the same problem.)

We get around this by replacing the Box with our own computer and adapter cable, and spoofing as a local keypad. We don't use Remote Mode or the remote interface at all.

CU601 Control Unit

Inside the CU, pin 9 is frame ground, so signals should reference pins 3 or 6, not pin 9. This is confirmed on p3-30 of the Leopard manual 996-0254 Rev C, and Eric M verified it visually by opening the CU. The schematic also correctly identifies pins 7 & 8 as unused. We use pin 8 as a control input to our own relay to allow us to "virtually" cycle the keyswitch.

We use an adapter to take a standard RS-574 (DB-9 PC serial port) plug into a plug for the CU601. We drive the relay on pin 8 with the PC DTR lead. +12 V or so is DTR ON; –12 V is DTR OFF. We put a diode in series with the relay coil so it activates only when DTR is ON.

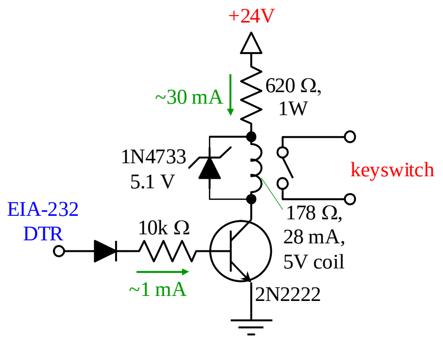
RS-574 to CU601 adaptor

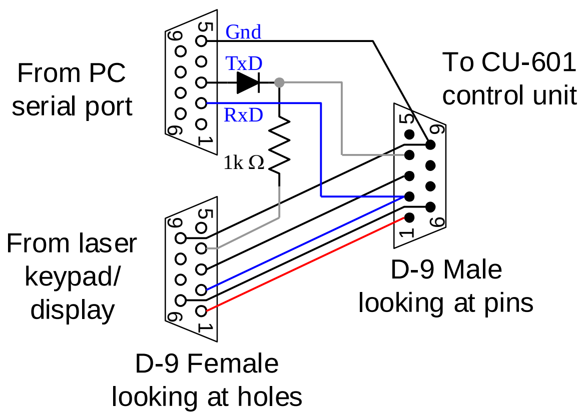
Female DB-9 RS-574 Pin #	**Signal Name**	**Signal Description**	**Male DB-9 Pin # for CU601**
1	CD	Carrier Detect	
2	RXD	Receive Data	2 (data from CU)
3	TXD	Transmit Data	4 (data to CU)
4	DTR	Data Terminal Ready	8 (relay control)
5	GND	Signal Ground / Common	3, 6 (Grounds)
6	DSR	Data Set Ready	
7	RTS	Request To Send	
8	CTS	Clear To Send	
9	RI	Ring Indicator	

External Laser Firing

The laser can fire under external control in either MANUAL or AUTO laser modes. A DC signal on the charge line does indeed cause charging immediately after each firing.

Test Circuits





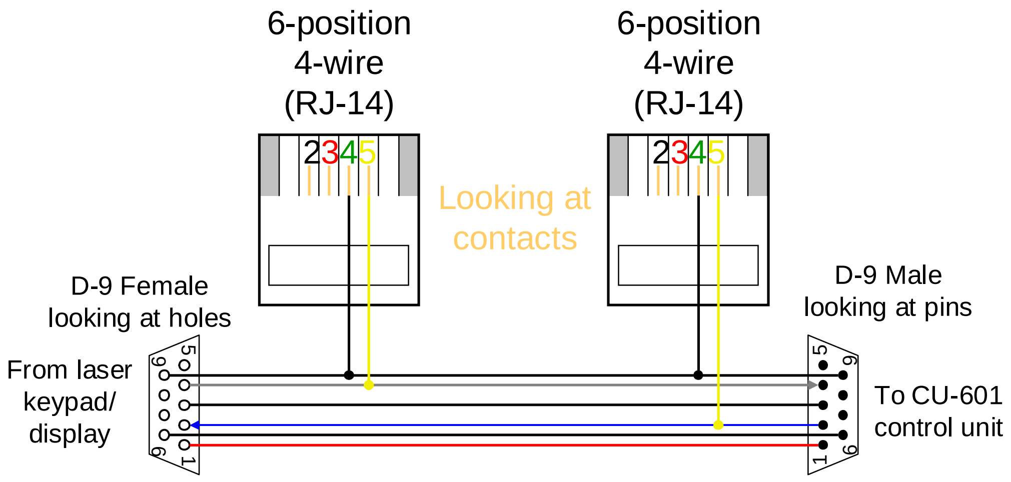
The left circuit closes the keyswitch contacts when the EIA-232 input goes high. These contacts are in parallel with the keyswitch on the front panel of the CU-601 control unit.

The right circuit is the cable adaptor that allows a PC serial port to inject characters into the control stream to theCU-601, while keeping the laser's own keyboard (display) box plugged in. Note that the keyboard box sends keep alives every 5 seconds or so, which can occasionally corrupt an injected character, so you have to watch out for this.

The circuit works as follows: EIA-232 outputs have a typical impedance of 150 Ω. EIA-232 inputs have an impedance of 5 kΩ. Adding the 1 kΩ resister doesn't hurt anything, but allows the PC serial port to "outdrive" the CU-601, when going positive. EIA-232 Tx leads idle at negative voltage (mark, or logic 1). They transit positive (logic 0) only when sending a character. Thus the diode/resistor combination acts as a logic AND gate, allowing either serial port to send characters. If they both send at the same time, both characters are corrupted.

Keyboard/CU Snooper Cable

This cable allows us to see traffic in both directions between the Keypad and CU, so we can better emulate the keypad.


6-position 4-wire (RJ-14)
2345
Looking at contacts
6-position 4-wire (RJ-14)
2345
D-9 Female looking at holes
From laser keypad/ display
D-9 Male looking at pins
To CU-601 control unit

DB-9 Female Pin	**Cable Wire Color**	**6 Position (RJ-x) Plugs**	**Keyboard Function**
1	red		+12V power
4	white	Plug 1 pin 5	RS-232 Box to CU
3, 6, 9	brown	Plugs 1 & 2 pin 4	Ground
2	blue	Plug 2 pin 5	RS-232 CU to Box
3, 6, 9	green and black		Ground

Note that the serial settings are 1200 bps 8N1, though (as mentioned earlier) the keyboard sends even parity.

Open Issues

What is "Mode" in the program status report? It is 4 immediately after reset. After the first Activate or timeout, it goes to 0. Those are the only 2 states I've seen, though there may be some associated with the laser actually running, which I cannot do yet. Is this a bit mask representing the status lights on the control box?