The LINUX L0 Code Manual

I. General Operations:

Normal operation is identical to the normal situation with no changes for the shift operator. In particular:

1. Runs are started and stopped with run control
2. The order of debugging is the same:
   - Press “stop run”
   - Issue “reboot”
   - Cycle the power on the L0 crate

II. Basic Differences:

The code runs on a VME Cpu as before, though the board is faster than the old vxworks board (x10), has 2 cpu's, and runs linux. There are the following differences:

1. The name of the CPU is “xvme01” (which will likely change in the near future). In the log files this name shows up as the first tag rather than the vxworks cpu's “L0”.

2. On “reboot” the CPU does not reboot. Only the process reboots. The main result of this is that the reboot of L0 is nearly instantaneous.

3. The board boots through the standard LINUX network boot sequence of

   DHCP ---> TFTP –-> PXE ---> LINUX

   The boot setup is on daqman:

   /etc/dhcpd.conf // to specify the location of the boot files

   /RTS/tftpboot/pxelinux.cfg/AC100E01 // specifies boot parameters and linux distro location

   /RTS/tftpboot/eth-linux-sl5.4-x86_74-test // linux distro

4. To log in you do not use “grab.” You use “ssh root@xvme01”. The password is currently the standard DAQ root password (which I/Tonko/Jack/Michael & Wayne) all know. Contact one of us to get it if you needed it. I plan to change this to allow an automatic login from the startrg machine, but I haven't done this yet.

5. The standard vxworks utilities are not present. In particular the “m command” is not available. On the other hand the standard linux utilites do exist (such as ps & top). However see the “debugging” section.

III. Debugging

1. Normally it should not be necessary to stop or start the process by hand. However to do so, the name of the process is “l0Mother”. To stop it, type “killall l0Mother” at the linux prompt. To restart type “l0Mother”.

2. The L0 processes are tied to a single processor, while the linux processes use the other processor. You will see the following “top” output:

```
[root@xvme01 ~]# top -p `ps -C l0Task -o pid=` -b
top - 11:46:27 up 2 days, 21:40,  2 users,  load average: 6.34, 4.49, 3.52
Tasks:   5 total,   1 running,   4 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.0%us,  0.0%sy,  0.0%ni, 99.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 99.9%us,  0.0%sy,  0.0%ni,  0.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   4021088k total,   399672k used,  3621416k free,    37100k buffers
Swap:        0k total,        0k used,        0k free,   186476k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
11610 root     -51   0 95900 8876 1460 R 100.0  0.2   4139:01 l0Task
11614 root      RT   0 95900 8876 1460 S  0.0  0.2   0:00.00 l0Task
11615 root     -86   0 95900 8876 1460 S  0.0  0.2   0:00.00 l0Task
11616 root     -86   0 95900 8876 1460 S  0.0  0.2   0:00.10 l0Task
11617 root     -86   0 95900 8876 1460 S  0.0  0.2   0:00.00 l0Task
```

In particular look at the CPU usage. You should see “cpu0” at 0.0, and “cpu1” at 99.0%. Also, note the negative priorities of the l0Task threads. This is the correct priority list.

3. It should not happen, but I have provoked situations where the CPU gets locked up and it is impossible to log on while debugging. In this case, there is a utility which can be run from startrg or daqman to restart the process:

> /RTS/exe/daqman/bin/LINUX/x86_64/l0Kick kill

which will kill and restart the l0 process.

4. To peek/poke VME registers use the “poke” command:

```
[root@xvme01 ~]# poke 0x1f000000
1f000000:  abcd0401-
1f000004:  00000007-
1f000008:  da130221-
1f00000c:  00000000-
```

This operates exactly the same as the vxworks “m 0x1f000000, 4” command. You can change the value by entering a number. You can read the next value by hitting return. Hitting “.” exits.