

Gigabit Ethernet Driver

S. Durkin

Getting Software:

Files to Download from <http://www.physics.ohio-state.edu/~durkin/software:>

```
ddu_gigabit_schar.tar.gz
schar_xxxx.tar.gz
```

Software Installation:

LINUX Interface Software Installation

- a. `cd /home/fast/fast_daq/driver`
- b. `ln -s ddu_gigabit_schar ddu`
- c. `cd ddu`
- d. `gcc -c ddu.c`

Gigabit Ethernet Driver Installation

We have provided you with a D-LINK Gigabit ethernet card. It will run 64 bit/ 66 MHz, 64/33, 32/64, or 32/33 pci.

Now the board must be configured for point to point ethernet (no outside users). To do this in control-panel set the ethernet address for the board to 192.168.0.3. Do not activate the board.

You must now make the character device in /dev. Execute the command: `mknod /dev/schar c 42 0`. You should also type: `chmod 777 /dev/schar` to give users permission to access the device.

The file `schar_xxxx.tar` contains the driver for the ethernet card. Unpack the driver, and as root run `ethreset` to load driver. You will have to specify the correct driver, either `ethreset_dgelin` (D-LINK) or `ethreset_acenic` (3COM, others...). You may have to modify this script changing `eth1` to the specific interface (e.g. `eth0,eth1,eth2,eth3`). Two drivers are loaded: either `acenic.o` and `eth_hook.o` or alternatively `dgelin.o` and `eth_hook.o`. You may also have to recompile the drivers. To check the drivers are loaded correctly type: `/sbin/lsmmod` and you should see the two drivers are loaded. You can also monitor `/proc/sys/dev/schar/0` (see below for details).

Running `/sbin/ifconfig` you should also see something like:

```
eth1      Link encap:Ethernet  HWaddr 00:50:BA:F3:B3:9E
          inet addr:192.168.0.3  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING NOARP PROMISC MULTICAST  MTU:9000 Metric:1
          RX packets:2002 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:5 Base address:0xe400 Memory:febfec00-febfed00
```

This means the driver is loaded correctly.

Driver Description:

The gigabit ethernet driver consists of the standard gigabit ethernet driver with a hook added to intercept reads before they are sent to the linux kernel. The only change to the `acenic.c` and `dgelin.c` drivers is that the call to `netif_rx` has been replaced by a call to the hook routine `netif_rx_hook`.

The character driver that contains `netif_rx_hook` is called `eth_hook.c`. It stores the incoming ethernet packets into a large ring buffer allocated in memory. The size of the memory allocated is given by the lines:

```
#define MMT_BUF_SIZE 131072
#define PNT_RING_SIZE 80
```

Here 131072 is the largest block of continuous memory one is allowed to allocate inside kernel routines. I have allocated 80 such blocks, yielding $80 * 131072 = 8$ Mbytes of memory. Incoming packets are automatically stored into this large ring buffer regardless of whether a read is taking place or not. To look at the present state of the loaded driver one can type: `cat /proc/sys/dev/schar/0`. On the screen one will see:

```
gigabit driver

written          1697385
read             1697385
nleft            0
ndumped          711868
bytes            121538162
page_ring        34
page_ring_r      34
```

Here written refers to the number of ethernet packets written to the ring buffer, read refers to the number of packets read by the external program, nleft is the number of packets left to read, ndumped is the number of packets dumped (see below), and bytes is the total number of bytes written to the ring buffer. The entries page_ring and page_ring_r refer to the ring block presently being accessed by the packet write and read respectively.

Checking the Software is Installed Correctly:

First of all the link light on the gigabit ethernet card should be lit. If not the card driver is not being installed correctly. In this case check /sbin/ifconfig, and /sbin/lsmmod for problems.

Generate a trigger for the DAQMB (pulse,inject,...). When the data is received by the gigabit ethernet card the light should flash. Typing /sbin/ifconfig you should see the number of packets received has increased. Typing cat /proc/sys/dev/schar/0 you should see the number of packets written has increased.

Using the interface routines you should now be able to read the events out of the ring buffer. If the open in init_ddu(); succeeds you will be able to read data. If not check /dev/schar exists, has major number 42 and minor number 0, and has user execute permission set.

Interface Routines

The driver interface routines called by the daq program consist of four main routines

```
Initializing: init_ddu();
Resetting: reset_ddu();
Reading: read_ddu();
Ending: close_ddu();
```

After initializing and example of reading data is shown below.

Receiving data:

```
count=-1;
buf2=(char **)malloc(2);
count=read_ddu(buf2,dp);
if(count<1){
    printf("No data from DDU\n");
    return count;
}

buf=(unsigned short *)*buf2;
buf=buf+2; /* ignore packet information from driver */
```

In addition to these four main routines there are five routines to modify the behavior of the driver.

Dumping Mode: `enabledump_ddu();`
`disabledump_ddu();`

Block Modes: `enableblock_ddu();`
`disableblock_ddu();`
`endblockdump_ddu();`

Dumping Mode:

Running with a continuous trigger the ring buffer in standard mode will finally overwrite data before it has been read. Dumping mode informs the driver to fill up the ring and then dump data until reading frees up room in the ring buffer.

Block Mode:

The routine `ddu_read.c` can be set to either wait until data is available before returning (blocking mode) or return immediately even if there is not data (nonblocking). In the latter case a “no data “ error will be generated. The blocking mode is useful when receiving data from an external source. In blocking mode the read will wait for ever for new data to arrive. The routine `endblockdump_ddu` called from an external process will terminate the read gracefully if there is no data. The read will exit with a “end block dump” error.