

About “tb,” the DOMCOM Device Driver

John Jacobsen
jacobsen@rust.lbl.gov



Lawrence Berkeley National Laboratory

Version 1.2
March 10, 2002

Look for updated versions of this document at

http://rust.lbl.gov/~jacobsen/docs/tb_driver.pdf and
http://rust.lbl.gov/~jacobsen/docs/tb_driver.doc

Table of Contents

What the Driver Does	2
How the Driver Works	2
Device files, major and minor numbers	2
Interrupts and circular buffers	2
Read and write	3
Diagnostics	3
Installing and Running the Driver	3
How to check out the driver code	3
How to compile the driver	3
How to install the driver	3
Troubleshooting	4
Additional Documentation	4

What the Driver Does

The “tb” device driver provides access to the Test Boards / DOMCom boards, hardware which control communications with the Digital Optical Modules (DOMs) over ~2km of twisted pair cable. FPGA logic in the DOMCOM boards provides input and output FIFOs connected to UARTs in the DOMCOM boards. The UARTs in turn drive the communications circuits. By writing to and reading from FPGA registers mapped to port addresses (e.g. 0x0300), the tb driver allows Linux programs to communicate with the DOMs.

The driver is meant to be used in conjunction with a program (dataserver or domserver) which connects the devices to network sockets to make them available on the network. In other words, data written to the socket over the network gets sent to the DOM, and data received from the DOM gets sent to the remote program over the socket. An independent program, syncserver, is used for timing calibration. It issues specialized commands to the DOMCOM FPGAs to generate timing pulses to the DOMs, and reads out digitized return pulses from the DOMs. Syncserver is currently independent of the tb device driver¹.

How the Driver Works

The tb driver is a kernel module. That means first of all that it runs with kernel privileges (to allow it access to the low level hardware, and to run quickly “at interrupt time” when needed). It also means that it can be loaded into the kernel dynamically, after the system has booted, and unloaded later at any time.

The driver can be thought of as having two parts - an interrupt handler, to allow for the fast buffering of incoming data from the DOMCOM board, and a system interface, which consists of open, close, read and write functions that allow the device to be treated like a file.

There are eight possible DOMCOM board IDs (0-7), configurable on the hardware by setting DIP switches. Each DOMCOM board is addressed through the ISA bus based on its DOMCOM board ID.

Device files, major and minor numbers

The device files for the driver are /dev/tb0, ... , /dev/tb7. The major number is 88 (defined in driver/tb.h) and the minor number is the DOMCOM board ID. When a user program opens one of the device files, it causes the driver to initialize buffer space for that DOMCOM board, enables interrupts on that board, and enables the hardware UART (so that communications are routed through the UART on the DOMCOM board, rather than through the RS-232 connection to the optional terminal server hardware). It also clears the input FIFO in case any garbage data is waiting to be read out.

Each of the device files /dev/tb* can only be opened by one process at a time.

Interrupts and circular buffers

Interrupt request (IRQ) number 10 is used. This can be changed in portio/portio.h. If changed, the DOMCOM PC BIOS must be changed to handle “legacy ISA” for that interrupt line². When this interrupt is issued by the FPGA of one of the DOMCOM boards, the interrupt handler is called, and all the DOMCOM boards with data are read out into circular buffers (1024 kB for each DOMCOM board). To

¹ This is mostly for historical reasons, and also because the communications functions provided by the tb driver are largely independent of the time calibration functions provided by syncserver.

² See “DOMCOM PC Installation Instructions” in the Additional Documentation section.

guard against race conditions, these circular buffers have pointers for “producer” (interrupt handler) and “consumer” (read function) ends. They also have a counter which indicates how many times the buffer has been wrapped (i.e., how many times the pointer has gone off the end and put at the first byte) for both the producer and consumer. This guarantees that any overflows of the buffer are caught.

Read and write

After a user program opens one of the files `/dev/tb[0,...,7]`, it can then `read()` from it or `write()` to it. The `read()` system function causes the driver function `read_tb()` to be called, which pulls the appropriate number of bytes from the circular buffer of the corresponding DOMCOM board. Similarly, `write()` causes `write_tb()` to be called, which writes data to the correct FPGA registers so that data appears in the output FIFO of the desired DOMCOM board.

Only non-blocking reads are currently allowed (many devices cause a reading application to halt execution, or “block,” until data is available to be read, but since some programs need to read from both the connecting socket and the hardware, this functionality is generally not wanted). `select()` or `poll()` are needed to determine if data is available to be read. Write also won’t block but may return only partially completed if the DOMCOM board’s output FIFO is full. In that case, a subsequent call to write will be needed. NOTE: if calling write repeatedly, put a `usleep` of ~10 msec in; if you simply busy-wait by calling `write()` repeatedly, CPU time will be wasted.

Diagnostics

When opening or closing the DOMCOM board device, or when errors occur, the tb driver causes messages to be written to the system log (`/var/log/messages`). A window on the DOMCOM board control PC running “`tail -f /var/log/messages`” will show these messages as they are generated. Additional diagnostic messages are also available in the driver, but you will have to change the code for those statements, substituting “`printk`” for “`pprintk`” and recompiling (see below).

Installing and Running the Driver

How to check out the driver code

The driver code is part of the DOM software archive (domsoft), organized using the CVS system. The subdirectory “driver” contains the bulk of the device driver code, although the “portio” subdirectory contains hardware information used by both the driver and the programs `syncserver` / `dataserver` and `domserver`. The “init” subdirectory contains a startup script for `/etc/init.d` which installs the driver at boot time.

How to compile the driver

Change to the driver subdirectory (`cd domsoft/driver`). Compile by typing “`make`”.

How to install the driver

The driver should already be installed on the DOMCOM PCs at LBNL and at the Pole. The `tbrc` startup script runs automatically at system boot time to install the release version of the driver (in `/usr/local/dom/driver`), and also to run `dataserver` and `syncserver` (or `domserver`, in newer versions of the software).

To install the driver by hand: become root. Change to the driver subdirectory. Make (should do nothing if already compiled). Install in release directory by typing "make install". If you want to load the driver by hand in the currently running kernel, "insmod tb". To remove the driver, "rmmod tb". You may have to do this first if the driver is already running. If any process is using the driver (i.e. has one or more of /dev/tb* open), you won't be able to remove the driver.

To see if the driver is installed, "lsmod".

Troubleshooting

The best way to test the driver is to start trying to talk through it. Install it, start dataserver and syncserver (or domserver) running on the DOMCOM PC (we'll use tbdaq-6.lbl.gov for this example), and start a tail of /var/log/messages. Run domtalk to that PC (tbdaq-6) and the appropriate port (4002 for DOMCOM ID 2), and power-cycle the DOMs. If it's running, you should see the DOM boot prompt appear:

```
Welcome to DOMBOOT release (CRC Enabled) of 15-Nov-1999
CRC table initialized...
Timeout value set to 30 seconds...
```

After the return key is pressed, the domboot prompt should appear:

```
Domboot 1.16 DOM 16 Enter command (? for menu):
```

You can also look at the system log (/var/log/messages) and see some of the driver output, both when the kernel module is installed, and when a connection is established by domtalk.

If there are problems, the following checklist should also help make sure everything is in place:

- ? DOM cabled properly to Test Board?
- ? Power supply connected, and voltage set to 80 V?
- ? Device driver loaded? (lsmod to check)
- ? Dataserver and syncserver running, or domserver?
- ? DOMTalk running, connected to correct server and port?
- ? DOMCOM board FPGA loaded?

If these things are in place, you should see the domboot prompt in domtalk when you cycle the power on the DOMs (if the DOM is in application mode, power-cycling will return to boot mode and show the boot prompt).

Additional Documentation

See **DOMCOM PC Installation Instructions** for information on the installation and configuration of the DOMCOM PCs.

See **String 18 Operating Instructions** for more information about how to do various DOM and DOMCOM related tasks.

These documents are available on the author's Web site, <http://rust.lbl.gov/~jacobsen>.

See also K. H. Sulanke's DOMCOM FPGA API document (ask_sulanke@ifh.de).