

String 18 Operations in Detail

John Jacobsen
jacobsen@rust.lbl.gov



Lawrence Berkeley National Laboratory

Version 1.5
March 8, 2002

In an effort to figure out the software requirements for and interactions between the various String 18 software components, I wrote the following outline of operations which describes the steps required to take meaningful data. Azriel, Kael and Doug provided helpful commentary.

The design presented here reflects the philosophy that, when a piece of hardware is powered up or restarted, the required software will automatically start running. An exception to this rule is the executive program "domexec," and various other testing programs meant to run interactively. I would like to avoid the need for domexec to start programs explicitly using exec or ssh.

This document addresses the normal mode of operation of the string, and does not (yet) cover all pathologies which may arise.

Contents

Software Nomenclature.....	1
Communication Channels, Enumerated.....	1
String 18 Phases of Operation.....	2
DOMCOM Software Diagram.....	5

Software Nomenclature

string18 (lowercase): PC used for String 18 control. AKA string18.spole.gov.
String 18: AMANDA string of DOMs.

The primary programs used to run String18 in its normal mode of operation are:

domexec: The master executive for the DAQ

EBTrig: The string trigger/event builder

RAPCal: Programs in the DOMCOM PCs which calculate and apply RAP calibrations to uncalibrated data stream from domexec before transmission to EBTrig.

domserver: The multithreaded application in the DOMCOM PCs. Includes threads for the transmission of DOM data to RAPCal, the execution of time calibrations, and for the retrieval of PMT data and slow control info from the DOMs.

In addition, the following programs are useful for testing or configuration of the string:

domtalk: talks to the DOM in “boot mode”

domcom: allows one to power a DOM on or off, or load an FPGA in a DOMCOM board

domtest: allows one to directly issue messages to the DOM application, for testing and debugging of the DOMs.

domlogger: similar to domtest but running in batch mode, collecting large amounts of data from the DOMs.

Communication Channels, Enumerated

The following channels of information are relevant. Each has its own brand of “message,” so the term message is vague. The diagram at the end of the document may be helpful for the visualization of the system as a whole.

- CC 1. domserver message thread to DOM application (DOM communications channel).
- CC 2. executive to domserver control thread [port 4200] (DOMCOM control channel).
- CC 3. executive to trigger/event builder [port 4201] (EBTrig) (event builder control channel)
- CC 4. domserver data thread to RAPCal [port 4020-4027] (uncalibrated data channel)
- CC 5. RAPCal to event builder [port undefined] (calibrated data channel)

In addition to the above channels, which are needed for normal operation of the string, the following channels are needed for testing, configuring, or backwards compatibility with existing software:

- CC 6. domtalk to domserver character thread to DOM boot program (raw character channel).
Older/existing Perl programs use the raw character channel as well, encoding the messages directly rather than going through Channel 2. This channel lives on each DOMCOM PC, ports 4000-4007.
- CC 7. syncserver message channel (used by the programs domcom, domtest, domlogger). This channel lives at Port 3666.
- CC 8. FastDOMMsg channel (used by pre-domserver Perl programs, which sent message “summaries” to a program “messageserver” running in the DOM; this “messageserver” then sent the fully-packetized messages on Channel 1 to the DOMs). This channel lives on each DOMCOM PC, ports 4010-4017.

String 18 Phases of Operation

Items flagged in **Bold** still need to be completed.

1. HARDWARE POWER-UP AND BOOT

Power on each of 5 DOMCOM PCs (tbdaq-1 through -5). Linux boots. Init script “tbrc” starts 8 RAPCal programs and one domserver program, and loads the kernel device driver for the DOMCOM boards. Domserver starts the following threads:

- ? Data source (x 8) (for RAPCal)
- ? Master Control (for Executive)
- ? Channel Control (x 8) (one thread to control each channel)
- ? Syncserver (deprecated, for backward compatibility w/ Perl scripts)
- ? Message I/O (x 8) (deprecated, for backward compatibility w/ Perl scripts)
- ? Character I/O (x 8) (for talking w/ DOMs in boot mode, and for messaging w/ older Perl scripts)
- ? Web info (for getting status of system through a Web browser)

String18 PC power-on: Linux boots. **Init script “ebrc” starts EBTrig and RAPCal daemons. RAPCal establishes communication with domserver’s data source thread on CC 4.**

Domserver loads default FPGA in DOMCOM boards.

Domserver opens/initializes the DOMCOM device driver.

A startup script on tbdaq-5 loads the clock distribution system FPGA through the jamplayer (currently done by hand).

HP Power supply on - DOMs boot applications and load FPGAs (this takes ~ 100 seconds).

2. EXECUTIVE STARTUP

When data taking is to begin, the executive is run by the “dom” account on string18, by typing “domexec.”

Executive reads database of DOMs. Database was produced by domtest and stores DOMCOM addresses, HV settings, **local coincidence settings**, etc.

Executive connects to EBTrig (CC 3).

Executive tells EBTrig which addresses to use for incoming data.

Executive connects to all 5 domserver’s control threads (CC 2).

Executive tells all DOMCOM FPGAs to synchronize their clocks to the next 1pps signal from the GPS clock; domserver reports the correct values to RAPCal.

3. DETECTOR INITIALIZATION

Once the executive is started up, it can initialize the DOMCOM hardware and DOMs to prepare for data taking. At the highest level, this consists of messages from domexec to domserver (CC 2).

- ? Make sure DOMs are booted into application
- ? **Make sure DOM FPGAs are loaded and identical for all DOMs.**
- ? **Start fast communications:**
 1. domserver’s message I/O thread tells DOM to change speed
 2. DOM changes speed
 3. domserver’s syncserver thread tells DOMCOM FPGA to change speed
 4. domserver’s message thread issues test message to make sure it worked
 5. domserver’s message thread tells DOM if it got the correct reply.
 6. (The double-message at the end is required so that both parties know that the new speed is in force - what happens if fast communications doesn’t work?)
- ? **Set DOM local coincidence FPGA registers (DOM dependent)**
- ? **Turn on and check DOM high voltage**
- ? **Set DOM SPE discriminators**
- ? **Make sure clock distribution system has stabilized (Jerry?)**
- ? **Set ATWD trigger masks**

? Time calibration initialization: set appropriate FPGA registers in DOM; set appropriate DAC.

4. BEGIN RUN

Run initialization

Exec tells EBTrig (via CC 3) the following:

- ? what run number and file name base to use
- ? how big to make each file in the run
- ? trigger configuration info (according to Kael) -
 1. N hits in space window of M contiguous DOMs
 2. Time window (nsec)
 3. H, hit threshold above which you only check the time coincidence and not the space coincidence. (This allows you to always trigger when you get 6 hits, say, in the time window, irrespective of their locations on the string. There is physical motivation for this triggering.)

Exec tells domserver control thread via CC 2 to perform N time calibrations. Control thread sends the result to RAPCal via the data source thread using CC 4. RAPCal sends the result to EBTrig on CC 5.

Exec tells domserver control thread to enter “normal” data collection mode.

Control thread loop (the frequency of each of these items is as yet undefined).

Get PMT data from DOM:

- ? domserver control thread decides to retrieve some data.
- ? domserver message thread -(CC 1)-> DOM: give me some data.
- ? DOM reads out lookback memory & sends the data on CC1;
- ? domserver data source thread reports the data to RAPCal on CC 4;
- ? RAPCal reports time-calibrated data to EBTrig on CC 5 (Azriel wants the ability to get the raw data too).

Perform “normal running” time calibration:

- ? Domserver message thread -(CC 1)-> DOM: start time calibration sequence.
- ? Domserver syncserver thread disables DOMCOM to DOM communications on CC 1.
- ? Domserver syncserver thread: Issues time tick.
- ? DOM sends upgoing time tick.
- ? Domserver syncserver thread reads out surface ADC waveform and time stamps
- ? Domserver message thread -(CC 1)-> DOM: give me ADC waveform in DOM and time stamps
- ? Domserver syncserver thread enables DOMCOM to DOM communications on CC 1.
- ? Domserver data source thread reports surface/DOM waveforms and surface/DOM time stamps to RAPCal on CC 4.
- ? RAPCal uses this data however it likes! (To correctly time-calibrate data for EBTrig).

Executive **verifies status** of domserver program and EBTrig, via periodic messages to each. Also, DOM slow control information such as high voltages, temperature, DAC settings, power supply voltages, etc. are collected. These quantities or summaries thereof are handed off to some as-yet-undefined monitoring facility.

5. STOP RUN

Do a final time calibration. Domserver sends results to RAPCals, along with an end-of-run marker.

Exec tells EBTrig to end the run, via a message over the control socket.

domserver, RAPCals keep running and just wait.

NOTE NOTE NOTE

As of this writing, the plan is that domexec doesn't have to be running after the run has started. Domexec is run only to start a run, to stop a run, or to query the status of each channel and the system as a whole.

6. DETECTOR SHUTDOWN

Turn off high voltage. (exec -CC 2-> domserver -CC1-> DOM).

Power DOMs off as well? If so, domexec -CC 2-> domserver's syncserver thread to set the appropriate FPGA registers.

7. CLOSEOUT

This should be a rare occurrence -- normally, you want to leave things powered on. To power off, issue Linux shutdown command to shut down DOMCOM PCs and string18. Power off these PCs and the HP power supply.

DOMCOM Software Diagram

Items in Grey are included in the domserver program.

