CLAS-NOTE 2001-004

Operation of the TOF Laser Calibration System

Kyungmo Kim, Kevin Giovanetti, Wooyoung Kim, Elton S. Smith

February 22, 2001

Contents

Abstract

The laser system has been installed in Hall B, which is used to calibrate the time-of-flight (TOF) counters. The procedure has been completely automated and monitored, so it can be operated by a single CODA configuration, TOF_LASER. The system consists of four setups which include a pulsed nitrogen laser, filter and mask, and fiber bundles which distribute light to all scintillation counters on a given support structure. Each laser can be powered and readied by remote control; the positions of the filter and mask are set by stepping motors within remote process.

1 Introduction

The CLAS time-of-flight (TOF) measurement system which consists of 342 scintillator counters can be calibrated by the data obtained through laser light, cosmic rays and particles created by the interaction of the electron beam and with the target. In particular, the time-walk correction parameters rely on the pulse hight obtained by the laser system which delivers light pulses with varying intensity to the center of each counter.[1] In this note, we will describe the electronics to control the laser system, and then detail the software that performs the calibration sequence.

2 Description

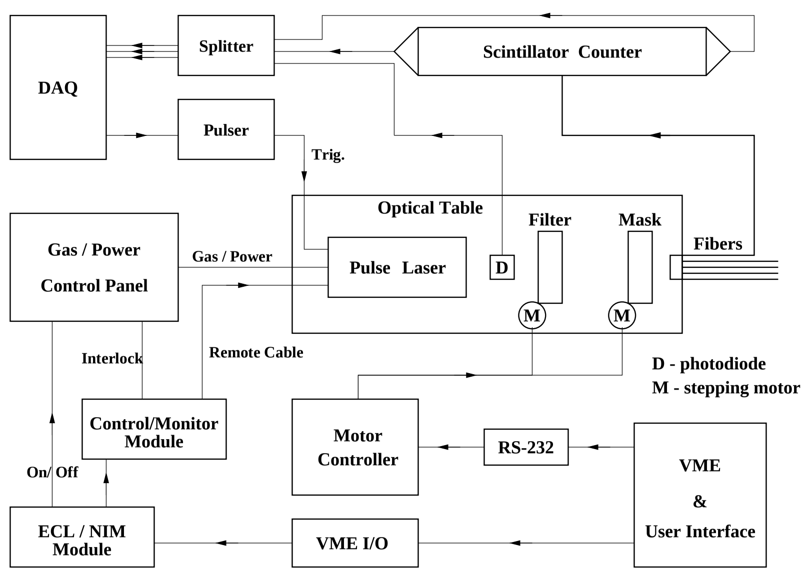

Figure 1: Schematic diagram of the laser calibration setup.

The laser system consists of four optical tables, each table contains a nitrogen laser [1] operating at 337 nm, a neutral-density filter to vary the amount of light to each fiber, a movable plate, or "mask," with holes to select which fibers are illuminated, a photo-diode circuit, and other optical elements. The system uses standard NIM, VME and CAMAC modules and some specially built modules to control the lasers, filters and masks remotely. Figure 1 gives a general overview of the calibration setup.

[1] LN203C, Laser Photonics Scientific

2.1 Laser Control

The laser can be powered on/off, enabled/disabled, and triggered remotely. The flow of nitrogen to the laser can also be remotely controlled with a solenoid valve. The gas flow rate is set locally by a flow meter which has been adjusted to provide sufficient nitrogen for all standard laser triggering rates. A pressure-sensing switch on the nitrogen line has been included in the laser's interlock circuit to prevent the laser from operating without nitrogen gas. A control/monitor module and a control panel were designed and built to provide these laser control functions as well as to provide feedback on the laser status (see Figures 1 and 2). A front-end processor in a VME crate reads laser status and controls laser functions through a VME ECL input/output module.[2] An ECL-TTL translation module [3] converts these signals to the TTL signals required by the control/monitor module and the control panel. A delay gate generator[4] is also used for the enable and disable. To calibrate TOF scintillators additional logic is required to generate DAQ triggers and to synchronize the operation of the four lasers. A diagram of the system is shown in Figures 1 and 2.

2.2 Motor Control - Filter and Mask

Light emitted from the laser is split and a small fraction (∼4%) is directed to a photodiode. The photodiode signal is used as the time reference for calibration. Most of the light passes through a neutral density filter, which is used to attenuate the light. The filter provides a range of intensity values suitable for measuring the time-walk correction of each pmt signal. Two overlapping continuously varying light attenuation plates are moved simultaneously but in opposite directions. The laser beam passes through the central overlap region and is attenuated by both plates. The plates can be moved continuously from one non-overlapping position (no attenuation) to the second non-overlapping position with the plate position reversed. This configuration provides a continuous variation of the beam while maintaining a uniform attenuation over the beam spot. The movement of the plates is controlled by a stepping motor. The setting of the motor determines the intensity of transmitted light.

[2] Shudi Gu and Ed Jastrzembski, Jlab.
[3] Logic level translator, Phillips Scientific NIM 726.
[4] Quad gate and delay generator, Phillips Scientific NIM 794.

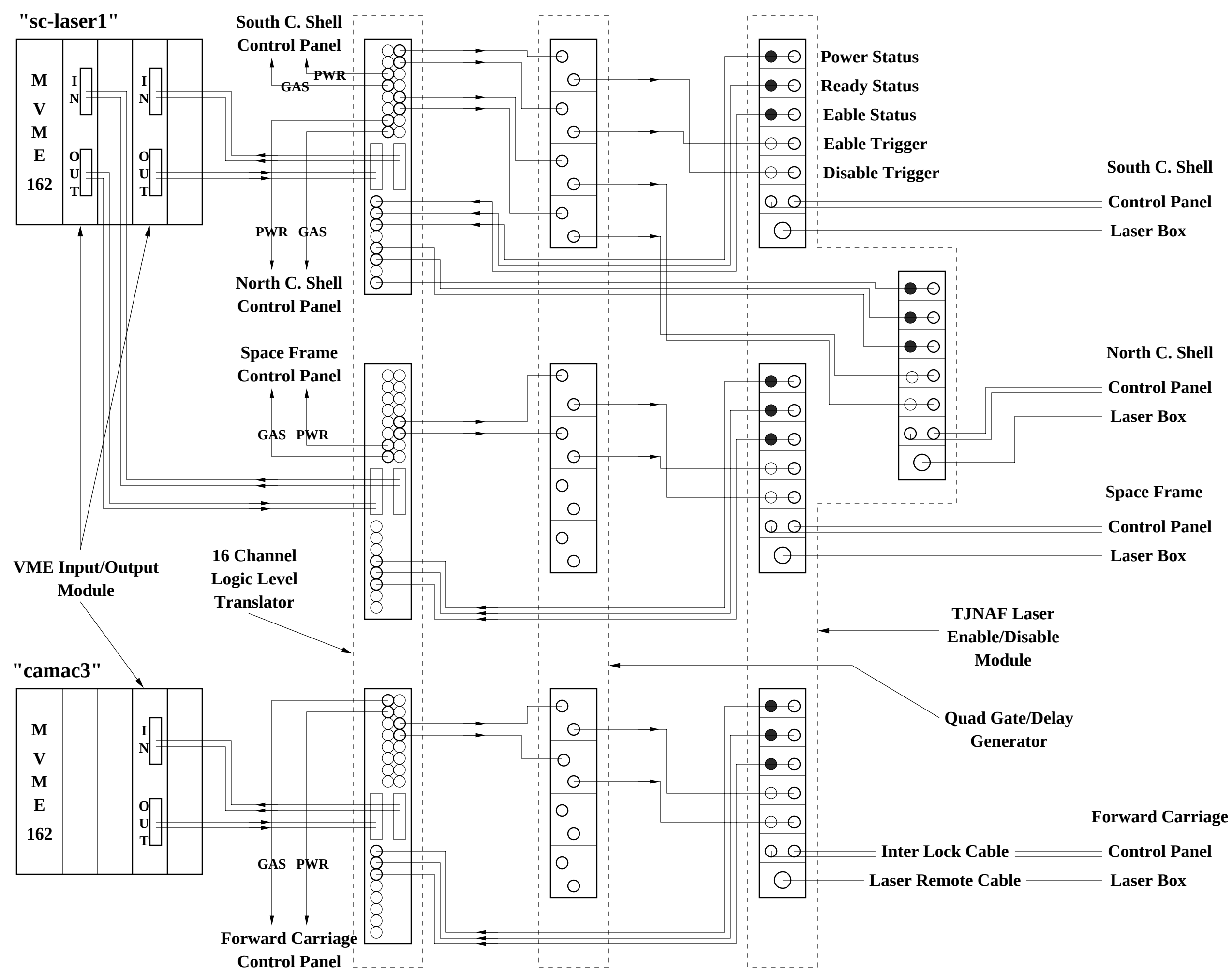

Figure 2: Schematic of Laser Control

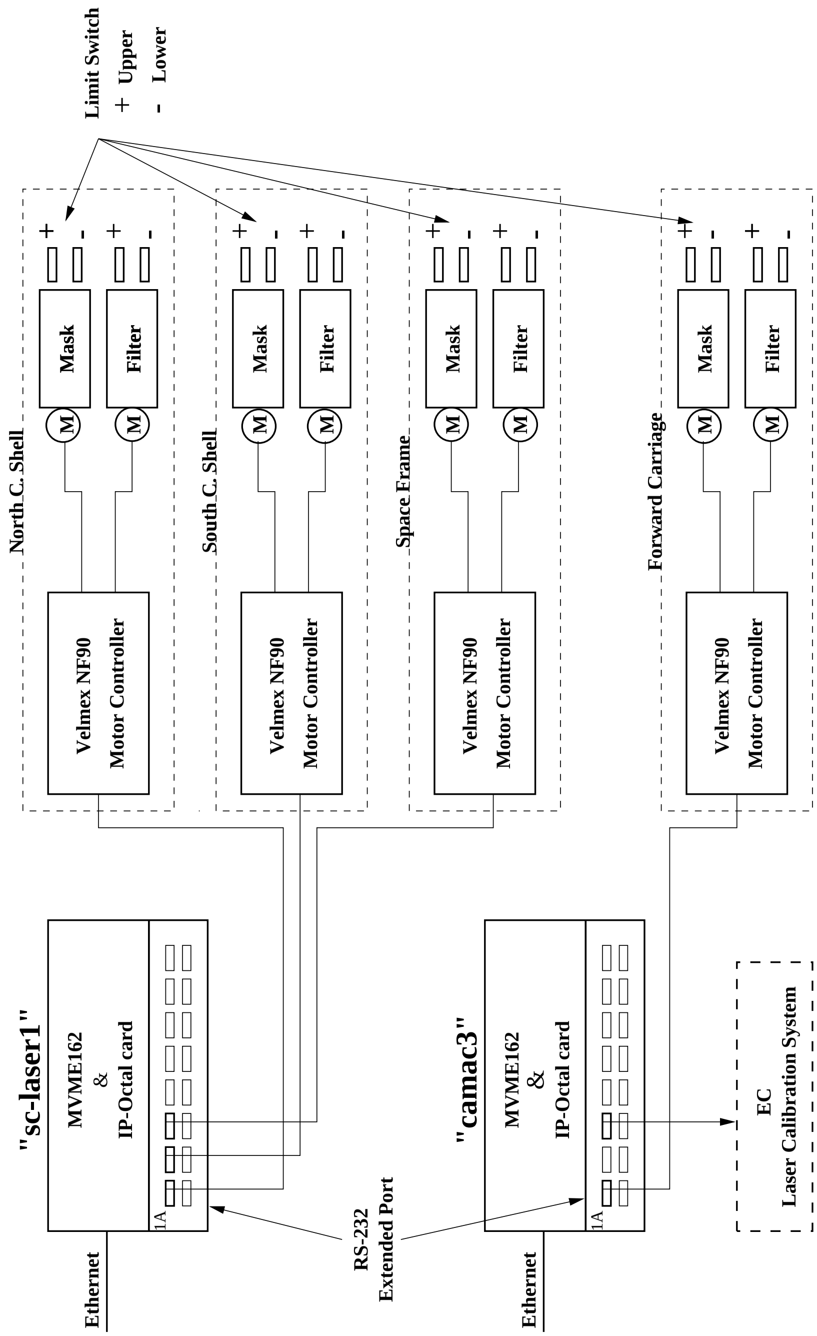

Figure 3: Schematic of Motor Control

After the filter, the light is diffused and then passes through a hole pattern in a metal plate, or "mask," which selects which fibers are illuminated. By moving the mask perpendicular to the incident laser beam, different twelve hole patterns can be positioned to illuminate various combinations of a fibers in fiber bundle that transmits the light to each TOF scintillator. The stepping motors of the filter and mask are operated by motor controller[5] connected to a VME crate via a RS-232 serial line, as shown in Figure 3.

3 Operation

The elements of the laser system, including the laser, filter and mask, are controlled via C library routines which are loaded directly into the VME controllers. In the TOF laser system there are two single board computers [6] which serve as VME controllers: "sc-laser1" and "camac3". The sc-laser1 covers the TOF laser system of the South and North Clam Shell and the Space Frame. The camac3 controls the laser system of TOF and EC in Forward Carriage. This VME controllers have the standard Hall B configuration and are maintained by Data Acquisition Group. The operating systems for the VME computers, VxWorks[7] offers a convenient method to control VME modules. The computers can be downloaded with precompiled procedures that perform VME I/O and control. These procedures can be executed as commands from other systems over the Internet. DAQ software implements this distributed functionality with routines, DPcommands, that can be included in remote programs. The commands can be also directly executed when logged into the VME computers. An example of the direct command execution is shown below:

```
clon00:clasrun> telnet sc-laser1
Trying 129.57.167.207...
Connected to sc-laser1.jlab.org.
Escape character is '^]'.

sc-laser1> open_port(4)
value = 0 = 0x0
sc-laser1> logout
                Connection closed by foreign host.
clon00:clasrun>
```

[5]Three-axis Stepping Motor Controller, NF90 series, Velmex Inc.
[6]MVME162, Motorola, Inc.
[7]VxWorks 5.2, Wind River Systems, Inc.

This architecture allows the development of simple robust procedures that can be combined remotely into complex tasks. The procedures were thoroughly tested via remote login to the VME computers. The more complex management of the TOF calibration system is provide by c programs that combine these VxWorks commands to initiate and then check laser and filter operation. To perform routine calibrations a program, TOF_laser.c, was written that initializes lasers and filters and then steps the system through a series of predefined configurations. The program is started by the DAQ and communication between the DAQ and TOF_laser allows program synchronization and TOF calibration status to be entered into the data stream. The progress of the calibration is displayed by a JAVA GUI. This status GUI operates independently. It receives messages from the DAQ system and the TOF_laser.c program through the network and displays this information in a convenient format. The status display is also started by the DAQ program.

The independence of the programs is an important feature. Data recording should not be impacted by the failure of a non-critical program that only provides status information for the experimenters. A server, the RT server, is available for broadcasting and receiving messages. The various programs broadcast information through the server. Any program can register to receive this information. Using this mechanism of information transfer means that the JAVA GUI does not need to be running in order to complete a calibration. Any number of programs can request information from the server. Therefore several monitoring programs can run simultaneously. Along with the broadcast information, some DAQ program and system status is ascertained using DPcommands.

3.1 C routines

The VME-C routines are divided into two parts; a part to control the laser and a part to control the motor. The laser control, as specified in the file laser.c, includes several routines to setup and control the laser operation and read its status. These routines read and write to VME registers. By setting bits in the registers one initiates a control function. By reading the register one can monitor the status (see Figure 2). The VME crate, "sc-laser1", has two VME I/O modules. The VME crate, "camac3", has one I/O module. The modules have configurable VME bus addresses that can be changed by dip-switches in the modules. The current address settings can be found in the program module laser.c. Addresses for both VME crates are shown below.

```
1st module on camac3
                        VMEbus Base Addr     [addr = ec]
```

```
                   INPUT PORT REGISTER  [addr = 6 ]
                   OUTPUT PORT REGISTER [addr = 2 ]
1st module on sc-laser1
                   VMEbus Base Addr     [addr = ed]
                   INPUT PORT REGISTER  [addr = 8 ]
                   OUTPUT PORT REGISTER [addr = 6 ]
2nd module on sc-laser1
                   VMEbus Base Addr     [addr = ee]
                   INPUT PORT REGISTER  [addr = 8 ]
                   OUTPUT PORT REGISTER [addr = 6 ]
```

The program module motor.c is used to control the motors for the filters and masks. The motors are driven by Velmex motor controllers which are connected to the VME controller via RS-232 ports. Eight additional RS232 connections have been added to VME crates sc-laser1 and camac3. The program module motor.c must be able to open and close the RS-232 ports as well as send and receive ASCII strings. The following shows the part of the motor.c code which specifies the RS232 port names used by the VME crates. The names refer to specific locations of the RS232 extension hardware (see Figure 3).

```
int open_port(int id)
{
  char *SerialPortName[] = { "",
                             "/tyIP/8",   /* for camac3 */
                             "/tyIP/0",   /* for sc-laser1 */
                             "/tyIP/1",
                             "/tyIP/2"
                           };
                        .
                        .
                        .
```

Port communication with the motor controller relies on standard I/O operations supported by VxWorks; write(), read(), etc.. Details can be found in VxWorks manual[3].

Commands (program modules) are normally downloaded when the VME computer is rebooted. All of the VME controller software is located in the directory tree ($CLON_VXWORKS for object code and $CLON_SOURCE/laser/sc/src/c_cntl and subdirectories for source code, see below). The reboot script for each VME computer can be found in the subdirectory boot_scripts, see below. The modules mentioned above are stored in different subdirectory (code, see below).

```
VME boot scripts: $CLON_VXWORKS/boot_scripts/
modules located:  $CLON_VXWORKS/code/
where
CLON_VXWORKS=/usr/local/clas/release/1.0/vxworks
```

3.2 Automatic process, TOF_laser

The program TOF_laser steps through a TOF calibration sequence using the simple commands defined in the program modules laser.c and motor.c discussed above. The program first proceed through an initialization step. A configuration file is read to determine which lasers will be used. The program loops through a setup loop where power and gas flow are initiated for each laser, the motor controllers are initialized, and the system status is checked. (Lasers require a five minute warm up before indicating ready.) Flags are set based on the system checks. In order to proceed the following conditions must be met:

- One laser must be ready.
- The associated filter motor must be ready.
- DAQ must be correctly configured (TOF_LASER).
- DAQ must be ready (active state).
- Trigger module for DAQ must be ready (bit 10 set).
- Must be connected to RT server.

When conditions are met the program steps the system through the corresponding calibration sequence read from the configuration file. The laser is triggered and the filter motor is set to loop several times through the full range of attenuation. The status of the motor is regularly checked. When the motor completes the required number of loops the code marks the sequence as completed and returns to the setup loop. Should an error occur the sequence is not marked as completed and the program returns to the setup loop. The program continues until all requested calibration sequences are complete. All systems are turned off upon successful completion. A simplified flow chart Figure 5 illustrates the program flow.

During the various cycles the program sends out status information using RT server broadcasts. In particular it sends status information which is inserted into the data stream.

Normal completion of the calibration will be indicated by the JAVA GUI. The experimenter would then end the run. If hardware errors do not allow for the completion of the requested calibration, the TOF_laser job must be manually stopped. There are expert programs available that can be used to check status and turn systems off by hand.

3.3 Monitoring JAVA Window

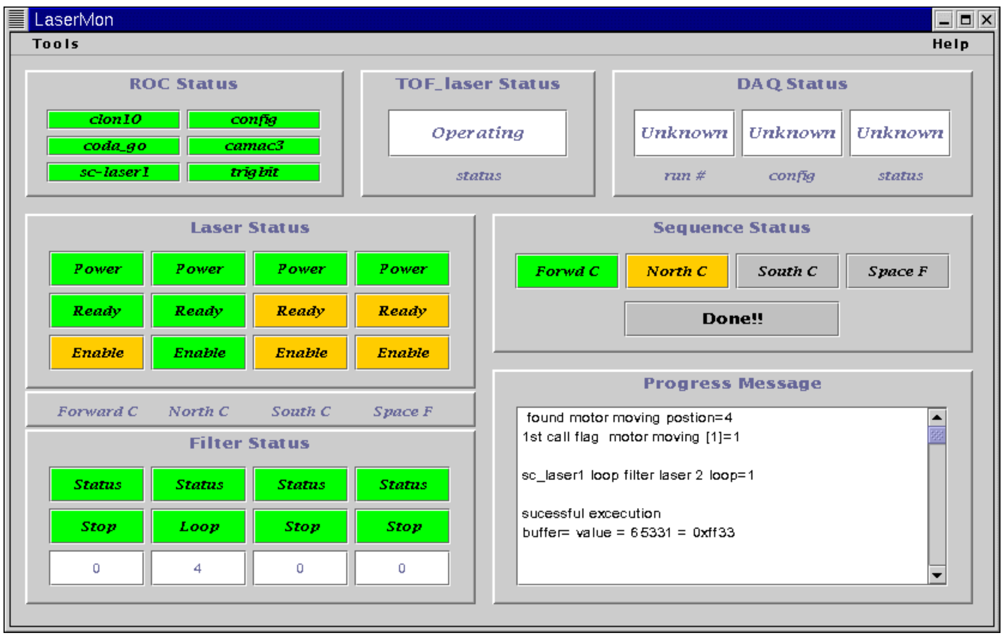

Figure 4: monitoring JAVA window.

The monitoring window written in JAVA is made to check the status of the laser system during TOF_ laser sequence. The program, TOF_laser, sends messages about the status using the RT-Server. The monitoring program gets the messages and displays status on the JAVA window.

Figure 4 shows the JAVA window. The section, "Laser Status", on the JAVA window shows the status of each laser. The "Filter Status" displays the program looping through a series of filter settings for each laser. We note that these reports are time delayed relative to the actual motion of the motors and are mainly diagnostic. The section, "Sequence Status" displays the status of each calibration sequence. The "green light" indicates those sequences which are already done, the "yellow light" means the sequence is underway. Each status button on the GUI may be clicked in order to receive additional information about use of the button.

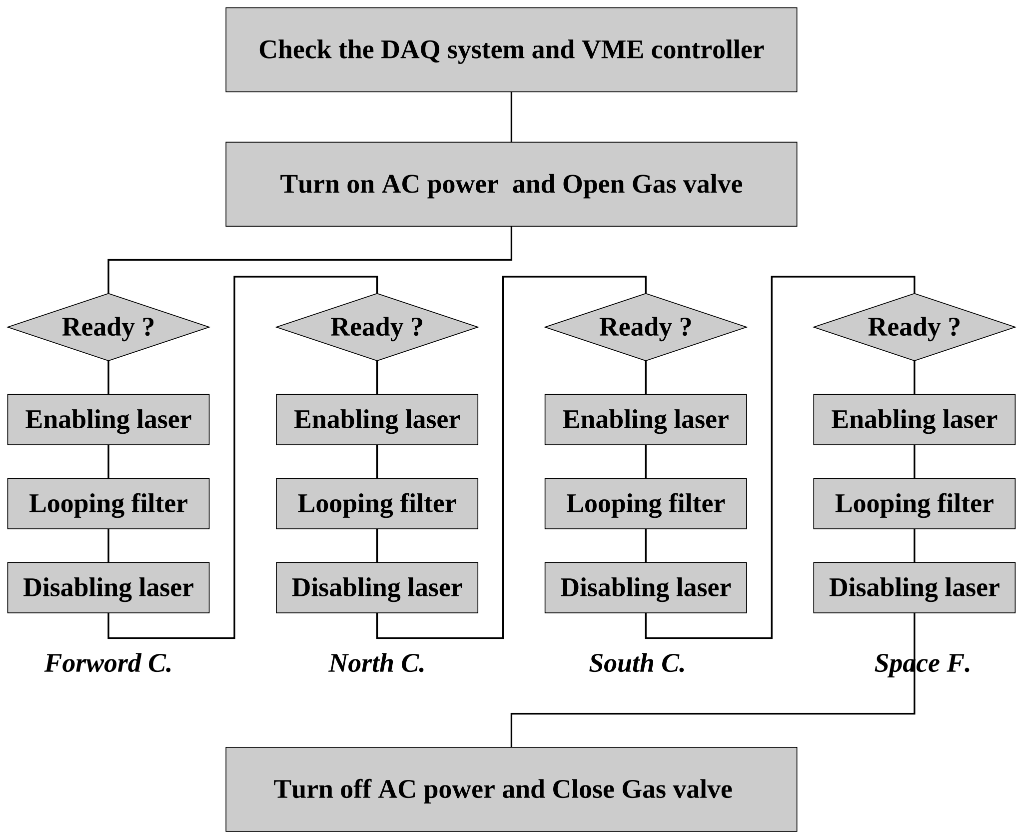

Figure 5: The sequence of the laser calibration process.

This monitoring GUI was developed in JAVA 1.2. The JAVA GUI includes the use of the Swing API. Two files LaserMon.java and LaserMonGui.java contain the required JAVA class definitions. The DAQ starts the program from $CLON_BIN. Therefore a current version of all class files needs to be loaded into this directory. All of this software has been added to the data acquisition software package. This will serve as the official version of the code. The DAQ software package has directory trees for a development version of the code and a current version of the code. A summary of the current version directory structure is shown below.

```
JAVA source: $CLON_SOURCE/laser/sc/gui/
JAVA executable:  $CLON_BIN/
where
CLON_SOURCE=/usr/local/clas/release/current/source
```

```
CLON_BIN=/usr/local/clas/release/1.0/bin
```

New versions of the code are kept in a development area. The structure parallels that of the current release. The source code can be found, for example in the following directory.

```
JAVA source: /usr/local/clas/devel/source/laser/sc/gui
```

4 Conclusion

An automated control system has been setup to work in conjunction with the CODA data acquisition in order to perform TOF calibrations. The code sets-up and controls the lasers and filters, monitors progress, sends status to the data stream, and displays status in a convenient format. The system has been tested and performs simply and reliably.

A Subroutine Guide

These routines are in laser.c and motor.c. The files are located in $CLON_SOURCE/vxworks/laser/ on the clon machines.

```
NAME        pwr()
SYNOPSIS    void pwr (
                  int id,      /* ID of a laser */
                  int x        /* turn ON(1), or turn OFF(0)
                  )
NAME        gas()
SYNOPSIS    void gas (
                  int id,      /* ID of a laser */
                  int x        /* turn ON(1), or turn OFF(0)
                  )
NAME        ena()
SYNOPSIS    void gas (
                  int id,      /* ID of a laser */
                  int x        /* Enable the laser(1), or Disable(0)
                  )
NAME        input()
SYNOPSIS    void input (
                    int id     /* ID of a laser */
                    )
  DESCRIPTION   This routine checkes the value of VME I/O input
                register. The value offers the information on the
                status of the laser.

NAME        open_port()
SYNOPSIS    int open_port (
                       int id   /* ID of a laser */
                       )
RETURN      OK(0), or if on error(-1)

NAME        close_port()
SYNOPSIS    void close_port (
                         int id  /* ID of a laser */
                         )
NAME        send_command()
SYNOPSIS    void send_command (
                           int id1, /* ID of a laser */
                           int id2, /*  1: filter motor */
```

```
                                     /*  2: mask motor */
                                     /*  3: filter motor speed */
                                     /*  4: mask motor speed */
                            int step /* a number of step */
                            )
  DESCRIPTION   This routine sends the command to move motor or
                change speed of motor to the velmex motor controller.

NAME     read_buffer()
SYNOPSIS void read_buffer (
                          int id  /* ID of a laser */
                          )
  DESCRIPTION   This routine reads the data from the velmex controller.

NAME     loop_filter()
SYNOPSIS loop_filter (
                     int id  /* ID of a laser */
                     )
  DESCRIPTION   This routine makes the filter move on the loop.
```

B Configuration File

The configuration file is presently in
$CLON_SOURCE/laser/sc/src/c_cntl/config/

```
This is the intialization and control file for laser running

 this is the comment block
 everyting enetered until the BEGIN_INIT is ignored

 laser 1 = Forward Carriage
 laser 2 = North Carriage
 laser 3 = South Carriage
 laser 4 = Space Frame

 only the first word or number is read per line
 start at the right margin and leave a white space when complete
 entries must be made in the exact order as found here


all data to be read in must start as the first char of a line

BEGIN_INIT   /* 1 note: 1=on, 0=off
0            /* 2 debug: set this to one
TOF_LASER    /* 3 configuration
10           /* 4 trigger bit
active     booted /* 5 coda state to wait for
booted       /* 6 camac3 status
booted       /* 7 sc_laser1 status
10.0         /* 8 laser frequency  (not implemented)
500          /* 9 number of pulses (-1 = inf not implemented)
10           /*10 time in seconds between initialization sweeps
1            /*11 Forward carriage laser used
1            /*12 North Clam shell laser used
1            /*13 South Clam shell laser used
1            /*14 Space frame laser used
1            /*15 use camac3 for Forward carriage laser
1            /*16 use sc_laser1 for NC,SC,SF lasers
END_INIT

For the procedure the lines begining with a # are ignored.
```

Do not leave any blank lines in the lists.

```
BEGIN_PROCEDURE
##### 1
1            /*seq 1 laser number (FC)
1            /*seq 1 filter 1 (only 1 filter available)
30           /*seq 1 speed
2            /*seq 1 number of loops
#####  2
2            /*seq 2 laser number (NC)
1            /*seq 2 filter 1 (only 1 filter available)
30           /*seq 2 speed
2            /*seq 2 number of loops
#####  3
3            /*seq 3 laser number (SC)
1            /*seq 3 filter 1 (only 1 filter available)
30           /*seq 3 speed
2            /*seq 3 number of loops
#####  4
4            /*seq 4 laser number (SP)
1            /*seq 4 filter 1 (only 1 filter available)
30           /*seq 4 speed
2            /*seq 4 number of loops
END_PROCEDURE
```

References

[1] E.S. Smith et al., Nucl. Instr. and Meth. 432(1999) 292.

[2] K.L. Giovanetti et al., "Detailed Report on the Design and Operation of the Calibration for EC", CLAS-NOTE-99-006, June 3, 1999.

[3] Vxworks Programmer's Guide, Wind River Systems, Inc.