CLAS-NOTE 2003-014
CVS: docs/clas_notes/tof_package

New Operating Software Package for the TOF Laser Calibration System

Serguei A. Pozdniakov
Institute of Theoretical and Experimental Physics
Moscow, Russia

August 22, 2003

Abstract

This document describes new software package for TOF Laser Calibration System, which includes User Interface, High Level Interface and Low Level Interface. New UNIX master based on completely revised new UNIX library gives the opportunity to calibrate the Time-of-Flight (TOF) counters in full and flexible manner, this procedure is completely automated and clearly monitored. Set of new scripts allows experts to control and operate filters, masks and lasers remotely at any time from any CLON clusters. Low level interface has become more compact and more reliable.

Contents

1 Introduction

The detail description of the Hall B TOF laser calibration system can be found in "Operation of the TOF Laser Calibration System" [1]. Briefly, this system consists of four optical tables. Each table has specific hardware setup which includes a pulsed nitrogen laser, filter, mask, and set of 24 fiber bundles [1, 2]. Laser light attenuated by filter and partially intercepted by mask is distributing by chosen fiber bundles to the currently investigating scintillation counters on a given support structure (Forward Carriage, North Clam Shell, South Clam Shell and Space Frame). The calibration procedure suppose to be completely automated and operated by a single CODA configuration, TOF_LASER. Filters and masks are set by Velmex stepping motors and could be operated remotely. Sequential choice of different combinations of filters and masks gives the opportunity to calibrate all scintillation counters in full manner.

Since the first version of the software program for TOF laser calibration had been written a lot of up-to-date changes (upgrades) have been made. Most of them are interdependent:

- upgrade to the latest version of VxWorks operating system
- upgrade to the newest and fastest CPU
- installation of the new serial board VMIC 6016

At the same time first version of the TOF laser calibration software was not full and very reliable:

- masks were never used
- low level interface to operate masks was not written
- communication problem were very frequently
- monitoring was not very convenient due to communication problems

All these problems and modifications of system and hardware lead to necessity of new software package for TOF laser calibration system, which now has new operating logic more compact and more reliable. This document describes three main part of this software package User Interface, High Level Interface and Low Level Interface.

2 User Interface

In this Section all necessary steps for successful TOF laser calibration will be described. This information will be enough for any advanced user to prepare, manage and make correct TOF laser calibration.

The calibration procedure assumes not only the right sequence of laser firing and predictable moving of masks and filters, it is also expect proper data taking. For this purpose CLAS Data Acquisition System (DAQ) will run specific CODA configuration, TOF_LASER. Implementation of the TOF laser calibration into CODA has been done by including three top level scripts in generic run configuration file (see Section 2.1).

Flexibility of the TOF laser calibration sequence lies in the TOF configuration file 'configuration.txt' which format will be explained in Section 2.2.

Then some useful scripts and programs will be presented in Section 2.3. These tools helps to operate filters, masks and lasers remotely and get their statuses at any time.

2.1 Generic run configuration file

CODA run configuration file selected during 'Download' transition. This file determines what actions will be done during different transitions for the current run and what CODA setting will be up. Below is the name of the file with the full path:

$CLON_PARMS/trigger/config/calib/tof_laser.cfg

The whole file is presented in Section A. Some essential part of this file will be discussed here.

Normally laser requires approximately five minutes to warm up, so it is better to turn laser on as early as possible. The following lines in 'tof_laser.cfg' turn gas and power on for all lasers in Hall B during 'Prestart' transition:

```
<prestart>
TOF_laser_on
</prestart>
```

'TOF_laser_on' could have the input parameters. If somehow, it is necessary to calibrate only North and South Clam Shells, the upper lines will look like:

```
<prestart>
TOF_laser_on 2 3
</prestart>
```

'TOF_laser_start' script opens the additional Xterm on the workspace where Run Control is running and starts the TOF laser UNIX master in the opened Xterm. TOF laser UNIX master ('TOF_laser') starts the calibration procedure and shows all steps of calibration in real time mode. If any essential error occurred the calibration will be stopped and adequate error message will be printed. The next lines activate 'TOF_laser_start' during 'Go' transition:

```
<go>
TOF_laser_start
</go>
```

After calibration will be done it is necessary to turn all lasers off. This realized by starting 'TOF_laser_off' during 'End' transition:

```
<end>
TOF_laser_off
</end>
```

Besides of including these three TOF laser scripts in run configuration file it is necessary to predefine some CODA parameters and activate some actions. The essential are starting the pulse-generator which will be pulsing lasers and enabling the TS trigger bit number 10 which corresponds to this pulse-generator. These actions are activated by including next lines in 'tof_laser.cfg':

```
...
<l1enable>
<!--  1   2   3   4   5   6   7   8   9  10  11  12   -->
<!-- === === === === === === === === === === === ===  -->
      0   0   0   0   0   0   0   0   0   1   0   0
</l1enable>
...
...
<pulser>
pulser_start 100 15 -1
</pulser>
...
```

2.2 TOF configuration file

TOF configuration file 'configuration.txt' allows to be very flexible in calibration procedure. TOF laser UNIX master reconfigures sequence of current calibration in accordance with settings of the parameters inside of this file. TOF laser UNIX master reads this file at the beginning of its operation from:

$CLON_PARMS/TOF_config

The original source of 'configuration.txt' are in directory:

$CLON_SOURCE/laser/sc/s

The example of TOF configuration file is shown in Section B. The format description of this file is presented within this file itself in the commented blocks.

TOF configuration file has two main section 'INIT block' and 'PROCEDURE block'. 'INIT' keeps expected CODA setting and determines what lasers will be used. 'PROCEDURE block' has full information about masks, filters, their speeds and their sequence of moving.

The mask is wide and long metallic plate placed between laser and set of fiber bundles. Several different hole patterns along this plate can be positioned to choose various combinations of fiber bundles. Normally after the initialization process all plates and filters are set to the extreme 'negative' positions. So all lasers are wide open and all fiber bundles could distribute light to the whole TOF detector. It is possible to calibrate TOF counters partially to avoid crosstalk and to separate electronics channels by choosing different predefined position of mask.

In example of TOF configuration file 'mask' is set to zero for all lasers. It means that masks will stay unmovable during calibration procedure. But one can use different number (up to 12) of mask positions, for a example it could be 3 for laser#1:

```
3     mask    1         /* number of masks (mask#1 positions)
0     mpos    1  0      /* position#0 of mask#1
1000  mpos    1  1      /* position#1 of mask#1
1200  mpos    1  2      /* position#2 of mask#1
1600  mpos    1  3      /* position#3 of mask#1
```

This means that four mask loops (from 0 to 3) will be done, mask position ('mpos') will be changed sequentially: 0, 1000, 1200, 1600. New mask positions could be added easily just by adding new lines with appropriate values, meanings and indexes.

2.3 Useful scripts and programs

Besides of the TOF laser UNIX master there are several scripts and programs which can be useful for remote operating and control of the laser calibration system. All of them are available from any CLON clusters. Their executable codes are in '$CLON_BIN' directory and their sources are in

'$CLON_SOURCE/laser/sc' directory. In accordance with their functionality scripts can be divided in two parts, operation scripts:

```
TOF_laser_on            - correct turn lasers on
TOF_laser_off           - correct turn lasers off
TOF_laser_enable        - enable lasers (if they are ready)
TOF_laser_disable       - disable lasers
TOF_move_mask           - move mask for specified laser
TOF_move_filter         - move filters for specified laser
```

and control scripts:

```
TOF_laser_status        - get lasers statuses
TOF_get_mask_position   - get mask position for specified laser
TOF_get_filter_position - get filter position for specif. laser
```

Control scripts could be executed at any time and by anybody. Operation scripts have to be executed only by experts or by advanced users and NOT during TOF calibration runs. Input parameters for some scripts are optional, for some of them are necessary. Options 'h', '-h' or 'help' will give the synopsis of the script.

3 High Level Interface

The whole software package for TOF Laser Calibration System was written in C language. Sources for UNIX part of the package are in directory:

$CLON_SOURCE/laser/sc/s

executable codes are installed in:

$CLON_BIN

This Section is mostly for experts, but it could be helpful for anybody who wants to know more about logic of TOF laser UNIX master (see Section 3.1) and communication interface between UNIX part of calibration and VxWorks part (in Section 3.2).

3.1 TOF laser UNIX master

'TOF_laser.c' is the main program for TOF laser calibration. Logic of this program became more simple and straightforward. At the same time it is easy to read and to understand. Almost half of the program is comments, which explain not only the logic of 'TOF_laser.c' itself, but also the logic of

included routines. To avoid many words of logic explanation here the whole C code of 'TOF_laser.c' is presented in Section C. The secret of the simplicity of TOF laser UNIX master logic lies in the completely new UNIX library for TOF laser calibration, which is presented in the next Section.

3.2 TOF laser UNIX library

TOF laser system uses standard NIM, VME, CAMAC modules and special hardware controlling lasers, masks and filters. During calibration procedure TOF laser UNIX master is doing a lot of conversation with hardware electronics, mostly with standard VME controllers using VxWorks operating system and with Velmex stepping motor controllers connected to VME via new serial board VMIC 6016. Low level library providing control of these hardware electronics will be discussed in Section 4.

All communication between UNIX and VxWorks now collected in one place 'TOF_laser_library.c' and all other routines use this communication interface. Mainly it is four functions using DP protocol:

```
int   command_init         ()
int   command_execute      (char *roc, char *cmd)
char *command_get_result   ()
void  command_print_result ()
```

First of them initializes DP protocol, second executes VxWokrs command ('cmd') in chosen VME crate ('roc' is a name of VME crate), third gets result of DP command execution and stores it in global variable, and the fourth one prints the result of DP command.

For this moment TOF laser system has two VME crates 'camac3' and 'sc-laser1', four serial port channels for Velmex stepping motor controllers and four lasers. To avoid multiplicity of variables and to have strict correspondence between low level electronics a few arrays of global variables were implemented in TOF library:

```
/* rocname vs laser_number */
char *rocname[]={"",
                 "camac3","sc-laser1","sc-laser1","sc-laser1"};

/* laser_number vs serial port channel number */
int  lsrN[]={0,1,2,3,4};

/* serial port channel number vs laser_number */
int  chnlN[]={0,1,2,3,4};
```

So now it is very easy to track any modifications in hardware redesign and to keep them correct in software part.

'TOF_laser_library.c' also includes almost all essential routines, list of them are in Section D. Some meaningful routines are in separate files:

```
bits.c            - bit operations
init_checks.c     - read configuration.txt
system_check.c    - check statuses of CODA and TOF system
system_setup.c    - setup lasers, masks and filters
system_shutdown.c - close motor ports and turn OFF lasers
```

All routines have short but sufficient descriptions in commented blocks in the same files, almost all of them are equivalents of VxWorks library routines and have the same synopsis.

4 Low Level Interface

Low level library (controlling lasers, masks and filters) and drivers for new VME serial board VMIC 6016 are loaded directly into the VME controllers by adding the next lines in 'camac3' and 'sc-laser1' boot_scripts:

```
#adds support for VMIC 6016 serial board
ld < drv6016
ld < install6016

# load TOF
ld < laser_ppc.o
ld < motor_ppc.o
```

The library codes and support for VMIC 6016 are in directory:

$CLON_VXWORKS/code

and the sources are in:

$CLON_SOURCE/vxworks/laser

One laser has one mask and one filter, both of them (mask and filter) are controlled by one Velmex stepping motor. Laser and motor have different way of controlling. Laser operations go through VME ECL Input/Output module, it is mainly address operations and processes of reading and writing to VME registers. So all laser routines base on the next four:

```
void set_address ()
void hexbin      (unsigned short h)
char *output     (int laser_number)
char *input      (int laser_number)
```

The motors are driven by Velmex motor controllers which are connected to VME via different channels of VME serial board VMIC 6016. Correct motor operation bases on stable communication with Velmex controller. It means proper opening and closing serial port channel, sending valid command to Velmex and recognizing response from Velmex. All this done by the next five routines, other motor routines using these five:

```
int  open_port       (int channel)
int  close_port      (int channel)
int  writeBuffer     (int channel, char *buff)
int  readBuffer      (int channel, char *buff, int nbytes)
int  decodBufferChar (char *str, int ch)
```

Function prototypes for all low level routines are collected in 'laser.h' in '$CLON_SOURCE/vxworks/laser' directory and are presented in Section E.

5 Conclusion

The described software package has been tested. It has been already used for mask calibration [3]. Several TOF laser calibration runs have been successfully done, but still without using mask. Unfortunately the process of mask calibration still not done in the full manner. Some test TOF laser calibration runs with random set of mask positions were also done. All above show reliable work of new package.

At this moment we have TOF laser UNIX master. This leads to a lot of network communications during TOF laser calibration, that makes the main procedure slow and fragile. One of the way to improve robustness of calibration procedure is to replace UNIX master by VxWorks master. It could be done only if all controlling electronics will be collected in one VME crate. Of course that expects a lot of work, new cabling, new hardware upgrade, but TOF laser software package even now almost ready for VxWorks master because of low level routines have the same synopsis and functionality in both UNIX and VxWorks libraries.

A CODA run configuration file

```
<!-- tof_laser.cfg -->

<!-- TOF Laser calibration configuration file -->


<trigger>


<prestart>
TOF_laser_on
</prestart>


<go>
TOF_laser_start
</go>


<end>
TOF_laser_off
</end>


<l1trig ignore="yes"/>

<l2trig ignore="yes"/>


<tsprog>
default
</tsprog>


<l1enable>
<!--  1   2   3   4   5   6   7   8   9  10  11  12   -->
<!-- === === === === === === === === === === === ===  -->
      0   0   0   0   0   0   0   0   0   1   0   0
</l1enable>
```

```
<prescale>
<!--   1    2    3    4    5    6    7    8   -->
<!--  ===  ===  ===  ===  ===  ===  ===  ===  -->
       0    0    0    0    0    0    0    0
</prescale>


<rawbanks>
s_pmt_raw_true
</rawbanks>


<scalers>
s_exclude_scaler
</scalers>


<pulser>
pulser_start 100 15 -1
</pulser>


<ccpretrig ignore="yes"/>

<ecpretrig ignore="yes"/>

<scpretrig ignore="yes"/>

<cctdc ignore="yes"/>

<ectdc ignore="yes"/>

<sctdc ignore="yes"/>

<lactdc ignore="yes"/>

<photon ignore="yes"/>


</trigger>
```

B TOF configuration file

```
#   configuration.txt
#
# This is the initialization and control file for
# TOF laser running
#
# This is a comment block.
# Everything before BEGIN_INIT is ignored.
#
# In INIT block:
#   first field is a value,
#   second field is a meaning,
#   all rest fields are comments.
# Any line can be commented by putting "#" at the beginning
# of line, in this case commented element will keep default
# value, which preassigned to him in init_checks.h
#

BEGIN_INIT
1          debug            /* 0=OFF, 1=ON (extra prints)
TOF_LASER  configuration    /* configuration
10         bit_number       /* trigger bit
active     coda_state       /* expected CODA state
1          laser#1          /* Forward Carriage \  laser usage:
1          laser#2          /* North Clam Shell  > ------------
1          laser#3          /* South Clam Shell /    1 = YES
1          laser#4          /* Space Frame     /     0 = NO
END_INIT

#
# In PROCEDURE block:
#   first field is a value,
#   second field is a meaning,
#   third field is a meaning index (laser id),
#   fourth field is a second meaning index (only for mpos),
#   all rest fields are comments.
# Any line can be commented by putting "#" at the beginning
# of line, in this case commented element will keep default
# value, which preassigned to him in init_checks.h
#
```

```
BEGIN_PROCEDURE
####  laser = 1  ################################
0     mask    1         /* number of masks (mask#1 positions)
0     mpos    1  0      /* position#0 of mask#1
1000  mpos    1  1      /* position#1 of mask#1
1500  mpos    1  2      /* position#2 of mask#1
200   mspeed  1         /* mask speed
30    fspeed  1         /* filter speed
2     floops  1         /* number of filter loops
####  laser = 2  ################################
0     mask    2         /* number of masks (mask#2 positions)
0     mpos    2  0      /* position#0 of mask#2
2000  mpos    2  1      /* position#1 of mask#2
4000  mpos    2  2      /* position#2 of mask#2
200   mspeed  2         /* mask speed
30    fspeed  2         /* filter speed
2     floops  2         /* number of filter loops
####  laser = 3  ################################
0     mask    3         /* number of masks (mask#3 positions)
0     mpos    3  0      /* position#0 of mask#3
2000  mpos    3  1      /* position#1 of mask#3
4000  mpos    3  2      /* position#2 of mask#3
200   mspeed  3         /* mask speed
30    fspeed  3         /* filter speed
2     floops  3         /* number of filter loops
####  laser = 4  ################################
0     mask    4         /* number of masks (mask#4 positions)
0     mpos    4  0      /* position#0 of mask#4
2000  mpos    4  1      /* position#1 of mask#4
1000  mpos    4  2      /* position#2 of mask#4
3000  mpos    4  3      /* position#3 of mask#4
200   mspeed  4         /* mask speed
30    fspeed  4         /* filter speed
2     floops  4         /* number of filter loops
################################################
END_PROCEDURE
```

C TOF_laser.c

```
/* TOF_laser.c - TOF laser UNIX master.
 *
 * Logic of this code:
 *
 * 1. Get initial values for the checks and procedure:
 *      - read configuration file configuration.txt,
 *        get and set initial values;
 *      - check rationality of continuation.
 * 2. System setup:
 *      - turn ON (gas ON and power ON) all selected lasers
 *        to the point from where they can be easily enabled;
 *      - disable all lasers, only 1 laser will be enabled
 *        and used at an appropriate moment;
 *      - initialize masks and filters (set them to the CCW
 *        limits and nullify mask's and filter's motors);
 *      - set masks and filters speeds.
 * 3. System check:
 *      - get and check statuses for all the lasers;
 *      - get and check trigger bit setting;
 *      - get and check coda configuration;
 *      - get coda state;
 *      - check and return lasers ready status and coda state.
 * 4. Main execution:
 *      - main loop over all selected lasers;
 *      - inside laser_loop loop over all chosen mask
 *        positions selected in configuration.txt;
 *      - inside mask_loop for each mask positions activate
 *        n-times filter_loop, n get form configuration.txt;
 *      - for each step in mask_loop:
 *          . pause the run,
 *          . move mask at new position,
 *          . resume the run,
 *          . enable appropriate laser,
 *          . activate filter_loop n-times,
 *          . disable current laser.
 * 5. Shutdown:
 *      - close motor ports;
 *      - turn OFF all lasers.
 */
```

```
#define INIT 1
#define TIMEOUT 600

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <strings.h>
#include "laser.h"
#include "init_checks.h"

extern int  chnlN[];
extern char *rocname[];
extern int  current_mpos[];

char prog_msg[1024];     /* message buffer; */

int
main(int argc, char *argv[])
{
  time_t tp;
  int  tmpval;
  int  lsri, mski, fltri, i;
  int  step;


  /*****!!!!!!!!   Init, Setup and Check   !!!!!!!!*****/

  /* Get time */
  tp=time(NULL);
  sprintf(prog_msg,"\n TOF_laser \n\n Start time = %s \n",
          ctime(&tp));
  ss_prog(prog_msg);

  /* Get initial values for the checks and procedure:  */
  /* read configuration.txt                            */
  ss_prog("Read configuration.txt");
  tmpval = init_checks(debug_value);
  sprintf(prog_msg,"init_checks() return status = %d \n",
          tmpval);
  ss_prog(prog_msg);
```

```
/* Check rationality of continuation */
if(!use_laser[1] && !use_laser[2] &&
   !use_laser[3] && !use_laser[4]) {
  ss_prog("All use_laser[i]=0,  Program stopped !!! \n");
  return(-1); }

/* Begin system setup */
ss_prog("Begin system setup");
tmpval = system_setup(debug_value);
sprintf(prog_msg,"system_setup() return status = %d \n",
        tmpval);
ss_prog(prog_msg);
if(tmpval != 0) {
  ss_prog("Some error occurred.
           Fix problem and restart program. \n");
  return(-1); }

/* Begin system check */
ss_prog("Begin system check");
tmpval=10;
while(tmpval != 0) {
  printf(" wait a few second ");
  for(i=0;i<10;i++) {printf(".");fflush(stdout);sleep(1);}
  printf("\n");
  tmpval = system_check(debug_value);
  sprintf(prog_msg,"system_check() return status = %d \n",
          tmpval);
  ss_prog(prog_msg);
  if(tmpval < 0) {
    ss_prog("Some error occurred.
             Fix problem and restart program. \n");
    return(-1); }
  if((time(NULL) - tp) > TIMEOUT) {
    sprintf(prog_msg,
    "SYSTEM_CHECK_TIMEOUT of %d seconds passed.\n",TIMEOUT);
    ss_prog(prog_msg);
    ss_prog("TIMEOUT EXIT: System still not ready
                           for Main Execution. \n");
    return(-1); } }
```

```
  /*****!!!!!!!!   Execution   !!!!!!!!*****/

  ss_prog("Main Execution");

  for(lsri=1; lsri<=4; lsri++) {

    if(use_laser[lsri] == 1) {

      for(mski=0; mski<=seq_mask[lsri]; mski++) {

ss_prog("  ------------------------------------------");
sprintf(prog_msg,"  main loop for laser#%d
            mask's position = %d", lsri, seq_mpos[lsri][mski]);
ss_prog(prog_msg);

/* pause the run */
if(coda_pause() == 0) {
  if(debug_value) ss_prog("  run paused"); }
ELSE_Error("clastrig2");

/* move mask at new position */
step = seq_mpos[lsri][mski] - current_mpos[lsri];
if(step != 0) {
  ss_prog("  move mask at new position \n");
  if(move_mask_ctrl(chnlN[lsri], step) < 0) {
    ss_prog("\n Error: Can not set mask properly \n");
    return(-1); } }

/* resume the run */
if(coda_resume() == 0) {
  if(debug_value) ss_prog("  run resumed"); }
ELSE_Error("clastrig2");

/* enable laser */
if(TOF_laser_enable(lsri) == 0) {
  if(debug_value) {
    sprintf(prog_msg,"  laser#%d enabled", lsri);
    ss_prog(prog_msg); } }
ELSE_Error(rocname[lsri]);
```

```
/* activate filter loop */
for(fltri=1; fltri<=seq_floops[lsri]; fltri++) {
  ss_prog("  =============");
  sprintf(prog_msg,"  filter loop#%d \n", fltri);
  ss_prog(prog_msg);
  if(loop_filter_ctrl(chnlN[lsri]) != 0) {
    ss_prog("\n Error: Can not loop filter properly \n");
    return(-1); } }

/* disable laser */
if(TOF_laser_disable(lsri) == 0) {
  if(debug_value) {
    sprintf(prog_msg,"  laser#%d disabled", lsri);
    ss_prog(prog_msg); } }
ELSE_Error(rocname[lsri]);

      } /* end of "for(mski=1;mski<=seq_mask[lsri];mski++)" */

    } /* end of "if(use_laser[lsri] == 1)" */

  } /* end of "for(lsri=1; lsri<=4; lsri++)" */

  ss_prog("main execution done \n");


  /*****!!!!!!!!   Shutdown   !!!!!!!!*****/

  ss_prog("Begin system shutdown");

  tmpval = system_shutdown(debug_value);
  sprintf(prog_msg,"system_shutdown() return status = %d \n",
          tmpval);
  ss_prog(prog_msg);
  if(tmpval != 0) {
    ss_prog("Some error occurred. Do shutdown manually. \n");
    return(-1); }

  return(0);
}
```

D UNIX laser.h

```
/*   laser.h - for UNIX   */

.....

/* function prototypes */

int   command_init         ();
int   command_execute      (char *roc, char *cmd);
char *command_get_result   ();
void  command_print_result ();

/*****  equivalents of VxWorks laser.c  *****/
char *output (int laser_number);
char *input  (int laser_number);
int  gas_ON  (int laser_number);
int  gas_OFF (int laser_number);
int  pwr_ON  (int laser_number);
int  pwr_OFF (int laser_number);
int  TOF_laser_enable  (int laser_number);
int  TOF_laser_disable (int laser_number);

/*****  equivalents of VxWorks motor.c  *****/
int  open_port           (int channel);
int  close_port          (int channel);
int  nullify_motors      (int channel);
int  set_mask_speed      (int channel, int speed);
int  set_filter_speed    (int channel, int speed);
int  get_mask_position   (int channel);
int  get_filter_position (int channel);
int  move_mask           (int channel, int step);
int  move_mask_ctrl      (int channel, int step);
int  move_filter         (int channel, int step);
int  loop_filter         (int channel);
int  loop_filter_ctrl    (int channel);
int  init_set            (int channel);
int  init_ctrl           (int channel);

.....
```

E VxWorks laser.h

```
/*   laser.h - for VxWorks   */

.....

/* function prototypes */

int  writeBuffer     (int channel, char *buff);
int  readBuffer      (int channel, char *buff, int nbytes);
int  decodBufferChar (char *str, int ch);
void set_address     ();
void hexbin          (unsigned short h);

/*****  laser.c  *****/
char *output (int laser_number);
char *input  (int laser_number);
int  gas_ON  (int laser_number);
int  gas_OFF (int laser_number);
int  pwr_ON  (int laser_number);
int  pwr_OFF (int laser_number);
int  TOF_laser_enable  (int laser_number);
int  TOF_laser_disable (int laser_number);

/*****  motor.c  *****/
int  open_port           (int channel);
int  close_port          (int channel);
int  nullify_motors      (int channel);
int  set_mask_speed      (int channel, int speed);
int  set_filter_speed    (int channel, int speed);
int  get_mask_position   (int channel);
int  get_filter_position (int channel);
int  move_mask           (int channel, int step);
int  move_mask_ctrl      (int channel, int step);
int  move_filter         (int channel, int step);
int  loop_filter         (int channel);
int  loop_filter_ctrl    (int channel);
int  init_set            (int channel);
int  init_ctrl           (int channel);

.....
```

References

[1] K. Kim et al., "Operation of the TOF Laser Calibration System", CLAS-NOTE 2001-004, February 22, 2001.

[2] E.S. Smith et.al., "The time-of-flight system for CLAS", Nucl. Inst. and Meth. A432, 265 (1999).

[3] Kiho Chu et.al., "Mask Calibration of the TOF Laser System", CLAS-NOTE 2003-013, February 12, 2003.

[4] E.S. Smith et.al., "CLAS TOF Online Manual", CLAS-NOTE 2001-019, October 29, 2001.

[5] CLAS TOF Group, "Calibration of the CLAS TOF System", CLAS-NOTE 1999-011, November 1999.

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