FIORD Software for LRIS Motor Control

John Cromer

July 6, 1993

Contents

0.1 Introduction

This document describes the FIORD software used to control the LRIS mechanical stages and reference lamps. The functions that comprise this software are those that are called from `cset()` and `cread()` in response to the **`modify`** and **`show`** instrument commands. This software executes on the instrument workstation and communicates with **`infoman`** and the VME systems via the **`traffic`** process. This document is intended to give the instrument specialist, the systems engineer and programmer an insight into the hows and whys of the the host software for motor control. See the appropriate reference manuals for more information on **`traffic`**, **`cread`** and **`cset`**, or **`show`** and **`modify`**. See the Function Descriptions document produced with **`wflman`** for details of the operation of each function.

0.2 Source Code and Nomenclature

FIORD functions for LRIS motor control are found in the the directory
`/kroot/kss/lris/fiord/lris`. In general, each module is named `f_`*keyword*`.c` where *keyword* is the FITS command keyword that is to be modified or shown. In each module are two functions: `output_`*keyword*`()` and `input_`*keyword*`()`. These are the functions that are called by **`modify`** and **`show`** respectively. Any auxiliary functions required by the input or output functions are also included in the same module.

0.3 General Mechanisms

In general LRIS motor control FIORD functions have only a few requirements. A typical output function accepts a new value for a stage position from the calling program, may or may not convert this value to encoder or motor units, creates a music message containing the request and the new position, sends the message to the VME system or to **`infoman`**, and accepts and returns the reply. The general algorithm is:

1. Convert sync flag into a boolean value.
2. Perform a simple range check on the new stage value.
3. Convert from user units to stage units if necessary.
4. Create the music message with the message ID, stage ID and new position.
5. Transmit the music message to the destination task, **`lserv`** on the VME crate, or **`infoman`**.
6. Return a sequence number if this is a no-wait command.
7. Wait for the first response from the destination task.
8. Return an error if first response was an error.

9. Wait for and receive the second response from the destination task.
10. Return an error if second response was an error.

Note in step 1, the "sync flag" is an integer used to coordinate the execution of the FIORD function with the replies it receives from other tasks. The function may wait for a reply, not wait for a reply, or wait for a specific signal or music message. Because of the serial nature of the LRIS design, all LRIS FIORD functions operate in the wait-for-response mode.

The input functions do the inverse of the output functions. They create and send a music message request to the VME system or to **infoman**, accept a reply, decode the reply and convert it to user units if necessary before passing it back to the calling program. The algorithm is:

1. Create music message with message ID and stage ID.
2. Transmit the music message to the destination task (**lserv** on the VME crate, or **infoman**.
3. Wait for and accept the response for the destination task
4. Unpack the returned value from the response.
5. Convert the value from motor or encoder units to user units.
6. Pass the new value back to the calling program.

Exceptions to the above algorithms are noted in the notes for individual stage functions. For the most detail, see the individual function descriptions in the document created from the source code headers with **wflman**.

0.4 Software for each Stage

The FIORD software associated with each stage is discussed below.

0.4.1 The Grating Subsystem

The grating turret subsystem consists of the grating turret and its detent mechanism, and each of 5 grating tilt mechanisms with each's associated motor-driven brakes.

The Grating Turret

The grating turret FIORD function `output_grating()` sends a message to the VME crate requesting the the turret to move to 1 of 5 positions, selecting a new grating. The input function, `input_grating()` sends a message requesting the current turret position, 1-5. Both use the completely standard algorithms described above. No units conversions are done. To understand how this request translates to a combined turret and turret detent move see the *VME Software for LRIS Motor Control* manual.

The grating turret position can also be selected by name using the GRANAME keyword. The functions **`output_graname()`** and **`input_graname()`** are the FIORD functions used for this. The algorithm for **`output_graname()`** is:

1. Call **`infoman`** to get the entry for the selected name.
2. Convert the name to the grating turret position number.
3. Call **`output_grating()`** to send the request to move the turret.
4. Update **`infoman`** with the new grating position name.

Grating Tilt by Angle

The keyword GRANGLE is used to tilt the current grating in the optical port. The functions **`output_grangle()`** and **`input_grangle()`** are the FIORD functions in module **`f_grangle.c`**. Both functions follow the general algorithm outlined in section 0.2 except as noted below.

Conversion from user coordinates (degrees) to stage coordinates (encoder steps) is done with a simple scale multiplication with a zero offset. Encoder steps are transmitted to and received from the VME system.

The **`output_grangle()`** function broadcasts the new grating tilt position after the request is sent to the VME system. It broadcasts the position again after the request response has been received. Ask Al Conrad why it's broadcast twice and why the grating tilt but not the grating turret position.

Grating Tilt by Wavelength

The keyword WAVELEN is used to tilt the current grating to the desired transmision wavelength in Angstroms. The FIORD functions are **`output_wavelen()`** and **`input_wavelen()`** in module **`f_wavelen.c`**.

The conversion from desired wavelength to encoder steps is a rather complicated procedure. The general algorithm for **`output_wavelen()`** is:

1. Call **`input_grating()`** to get the current grating ID.
2. Call **`get_grating_info()`** which contacts **`infoman`** with the grating ID to get the grating parameters.
3. Call **`wave_to_angle()`** to convert the wavelength to grating tilt angle.
4. Convert the angle to encoder steps.
5. Convert the sync flag to boolean.

6. Create the music message.
7. Transmit the music message to the VME system.
8. Wait for and receive the first response from the VME system.
9. Wait for and receive the final response from the VME system.

The heart of the procedure is the conversion from wavelength in Angstroms to grating tilt. The function **`wave_to_angle()`** does this.

The grating angle formula with the LRIS geometry gives for the wavelength:

$$\lambda = \frac{10^7}{nd}[\sin(\alpha + \beta) + \sin(\alpha + \beta - \phi)] \tag{0.1}$$

where:

λ is the wavelength in Angstroms.

α is the collimator angle — collimator to the LRIS y axis.

β is the grating angle — LRIS y axis to the grating normal.

ϕ is the total LRIS beam angle — collimator to red camera.

n is the grating order.

d is the grating resolution in groves per mm.

Each of the angles above has as its vertex the center of the grating surface. All of the parameters except β, the grating angle and λ, the wavelength are constants for each grating. The equation is easily solved for λ given β. Solving for β given λ requires a numerical procedure. The function **`wavelen_to_angle()`** uses a Newton–Rapheson procedure for this purpose. Any reasonable text on numerical analysis should be consulted as a reference. After the numerical procedure completes, another correction is applied to β of the following form:

$$\Delta\beta = a + b\beta$$

where a and b are empirically determined constants. The final angle that is passed back to the calling function is $\beta + \Delta\beta$. The values of constants a and b are defined in the in-line code of **`wave_to_angle()`**, as this was a recent modification to the original code.

As stated above the other parameters which are constant for each grating, ϕ, α, n, and d, are stored in the **`infoman`** database and are read with the function **`get_grating_info()`**. This function requires a grating number as input and returns an array with the grating information in it. The algorithm is:

1. Switch on the grating number to select the correct `icode` for `infoman`.
2. Call `input_float_map()` to get the information from `infoman`.
3. Return the `info` array with the grating information to the calling program.

The grating information is stored in `infoman` in floating point maps and may be accessed and changed via the keywords G1MAP, G2MAP, G3MAP, G4MAP and G5MAP, for gratings 1–5 respectively. For example

```
show -s lris g2map

                  g2map =
     resolution (lines/mm) 300.000000
      beam angle (degrees) 44.200001
      coll angle (degrees) 6.400000
                     order 1.000000
    apix (angstroms/pixel) 2.700000
         blaze (angstroms) 5000.000000
```

shows the contents of the `infoman` table for grating #2. Similarly the `modify` command can be used to change the values in these arrays. The functions used to maintain this information are contained in the module `map.c` and were lifted from the HIRES software. For more information on these mapping functions see the document *FIORD Software for HIRES Motor Control* by Al Conrad and Dean Tucker.

The function `input_wavelen()` displays the current value of the grating angle in terms of the central wavelength. The algorithm is essentially the reverse of `output_wavelen()`:

1. Call `input_grating()` to determine which grating is in the optical path.
2. Package the music message requesting the current grating angle.
3. Send the message to the motor control VME system.
4. Wait for and receive the response from the VME system.
5. Unpack the grating angle encoder position from the received message.
6. Convert the angle to wavelength via equation (0.1) above.
7. Return wavelength to the calling program.

Two additional functions are included in the `f_wavelen.c` module in support of the conversion of angle to wavelength. These are `fiord_sin()` and `fiord_cos()`. These calculate the sin and cos of angles (in radians) each using the appropriate 8-term Taylor series. These functions are used rather than those from the Sun math library because the math library cannot link with position-independent code. The FIORD library is position independent.

Finally, for moving the grating tilt by wavelength for use with multislits, the functions `output_mswave()` and `input_mswave()` are used in support of the MSWAVE keyword. These functions are identical to `output_wavelen()` and `input_wavelength()` except that the grating angle is modified by an angular offset. The offset is added to the grating angle before its conversion to encoder position in `output_mswave()` and subtracted from the angle before its conversion to wavelength in `input_mswave()`.

0.4.2 Filter Selector/Changer

Red camera filters may be changed by name using the keyword REDFILT, or by number using the keyword REDFNUM. The functions `output_redfilt()` and `input_redfilt()` in module `f_redfilt.c` are the FIORD routines for the keyword REDFILT, `output_redfnum()` and `input_redfnum` in module `f_redfnum.c` for REDFNUM. The functions are completely standard and follow the general mechanisms and algorithms described above. For more details see the Functions Descriptions document created from the source code headers with `wflman`.

0.4.3 Slitmask Selector/Changer

Slitmasks may be changed by name using the keyword SLITNAME, or by number using the keyword SLITMASK. The functions `output_slitname()` and `input_slitname()` in module f_slitname.c are the FIORD routines for the keyword SLITNAME, `output_slitmask()` and `input_slitmask` in module `f_slitmask.c` for SLITMASK. The functions are completely standard and follow the general mechanisms and algorithms described above. For more details see the Functions Descriptions document created from the source code headers with `wflman`.

0.4.4 Red Camera Focus

The red camera focus is changed with the keyword REDFOCUS. The FIORD routines `output_redfocus()` and `input_redfocus()` in module `f_redfocus.c` support this keyword. These functions are completely standard and use the general mechanisms and algorithms described above. The conversion from focus position in microns to and from encoder steps is given by the following fomulae:

$$e = (f - fmin)scale + offset$$

$$f = frace - offsetscale + fmin$$

where:

e is the encoder position in encoder steps.

f is the focus position in microns.

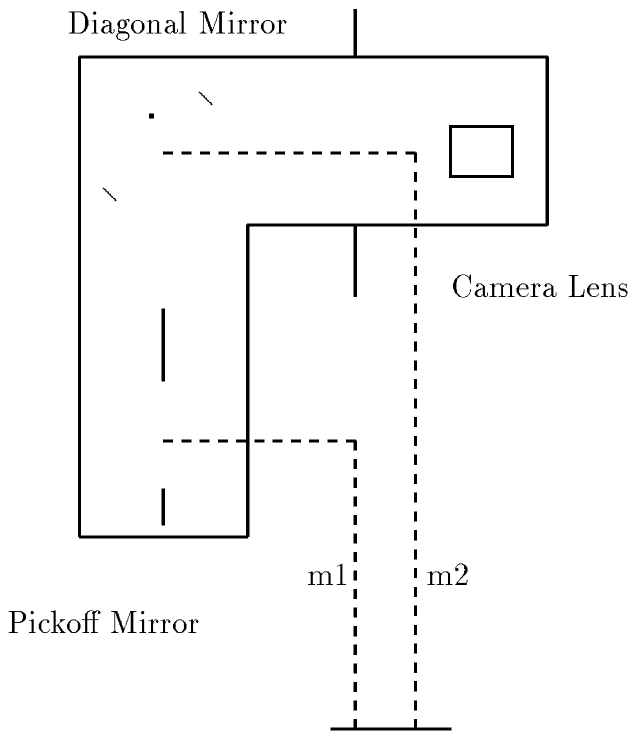
fmin is the minimum possible focus position in microns.

scale is the conversion factor in encoder steps per micron.

offset is the encoder reading at the minimum position.

0.4.5 Offset Guider

Refer to the following figure in the discussion of the offset guider.



The Offset Guider

The offset guider position is selected using the keyword TV1FPOS. The functions `output_tv1fpos()` and `input_tv1fpos()` in module in `f_tv1fpos.c` implement the `modify` and `show` commands for TV1FPOS. Two motors are involved in moving the offset guider. One, labeled M2, moves the position of the entire stage. The other, labeled M1, moves the pickoff mirror with respect to the stage in order to change the guider focus. Unfortunately, this last move changes the pickoff mirror position also, obviously. When an offset guider move is requested the software must calculate moves for M1 and M2. The algorithm for `output_tv1fpos()` is:

1. Check the requested move for range error.
2. Call `calc_tv1_epos()` to get the destination positions for M1 and M2.
3. Create the music message with the two stage IDs and the two destination encoder positions.
4. Send the message to the motor control VME system.
5. Wait for and receive the 1st response from the VME system.
6. Wait for and receive the final response from the VME system.

The key is the calculation of the two encoder positions for M1 and M2 done by the function `calc_tv1_epos()`. Here is the algorithm:

1. Calculate M2's position as a function of M1 in mm.

$$m2 = a_0 + a_1 m1 + a_2 m1^2$$

2. Calculate M2's displacement with respect to its home position.

$$\Delta m2 = m2 - m2home$$

3. Calculate the destination separation between M1 and M2

$$\Delta m1m2 = m2 - m1$$

4. Calculate M1's displacement with respect to M2.

$$\Delta m1 = \Delta m1m2home - \Delta m1m2$$

5. Convert the destination displacements of M1 and M2 to encoder steps.

$$e1 = m1\_offset + \Delta m1 \cdot m1\_scale$$

$$e2 = m2\_offset - \Delta m2 \cdot m2\_scale$$

where:

$m1$ is the position of the pickoff mirror in mm desired by the user. The distance from the LRIS origin to the pickoff mirror center.

$m2$ is the destination position in mm of the M2 stage actually the distance from the LRIS origin to the diagonal mirror.

a_n are the quadratic coefficients.

$m2\_home$ is the home position in mm of the M2 stage.

$\Delta m1m2$ is the destination separation between M1 and M2.

$\Delta m1$ is the destination position of M1 with respect to M2.

$\Delta m1m2home$ is the separation in mm between M1 and M2 at their home positions.

$m1\_offset$ is the home position in encoder steps of M1.

$m2\_offset$ is the home position in encoder steps of M2.

$m1\_scale$ is the conversion constant in encoder steps per mm for M1.

$m2\_scale$ is the conversion constant in encoder steps per mm for M2.

$e1$ is the destination encoder position for M1.

$e2$ is the destination encoder position for M2.

The destination encoder positions for M1 and M2 are sent back to the calling program, in this case `output_tv1fpos()`.

The `input_tv1fpos()` function performs the inverse of `output_tv1fpos()`. The algorithm is:

1. Create the music message with the stage IDs of M1 and M2.
2. Send the message to the motor control VME system.
3. Wait for and receive the response from the VME system.
4. Unpack the encoder positions of M1 and M2 from the response message.
5. Calculate the positions of M1 and M2 in mm.
6. Return the position of M1 to the calling program.

The following formulae are used to calculate the M1 and M2 positions in mm from their respective encoder positions are:

$$m2 = m2\_home - \frac{e2 - m2\_offset}{m2\_scale}$$

$$m1 = m2 - (\Delta m1m2home - \frac{e1 - m1\_offset}{m1\_scale})$$

where the symbols have the same definitions as above.

0.4.6 Guider Filters

Both the offset guider (designated `tv1`) and the slitviewing guider (designated `tv2`) have 4-position filter wheels that are accessed with the keywords TV1FILT and TV2FILT. The corresponding FIORD functions are `output_tv1filt()`, `input_tv1filt()`, `output_tv2filt()` and `input_tv2filt()`. These functions are completely standard and use the general algorithms and mechanisms discussed in section 0.3.

0.4.7 Trapdoor

The trapdoor is opened and closed using the keyword TRAPDOOR. The FIORD functions are `output_trapdoor()` and `input_trapdoor()` in module `f_trapdoor.c`. These functions are standard and use the general algorithms and mechanisms in section 0.3.

0.4.8 Reference Lamps

The FIORD functions supporting the LRIS reference lamps are based on the macro style of routines used by the HIRES FIORD software for functions of a repetitive nature. All functions are defined in the module `f_lamps.c`. The key FIORD output function is `switch_lamp()`. This function follows the standard FIORD output function algorithm. The "new position" in this case is a boolean toggle switch. If it is non-zero, the lamp will be turned on, if zero, the lamp will be turned off.

There are four operating lamps on the LRIS at this time, controlled via the keywords LAMP1, LAMP2, LAMP3 and LAMP4. Macros defined in the C preprocessor in `f_lamps.c` use `switch_lamp()` described above to create the four individual FIORD output functions: `switch_lamp1()`, `switch_lamp2()`, `switch_lamp3()` and `switch_lamp4()`. For more details see the Function Description document generated from the source code headers by `wflman`.

There is only one FIORD input function for the reference lamps. It is called `input_lamps()` and is also found in the `f_lamps.c` module. It sends the request to the VME crate and receives a music message containing the status of all lamps. The array containing the lamp status values is passed back to the calling program. The algorithm used here is the standard one described in section 0.3.

Appendix A

Encoder Stage Scales

Scale Factors for LRIS Encoder Stages	
Stage Name	Scale Factor
Grating Tilt	886.0 encoder steps/degree
Red Camera Focus	25.802 encoder steps/micron
Offset Guider M1	3149.61 encoder steps/mm
Offset Guider M2	3149.61 encoder steps/mm