



---

## IRIS 2 Spectrograph Task

### Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Instrument Electronics.	3
1.2	Software	3
<b>2</b>	<b>Software Interface</b>	<b>4</b>
2.1	DRAMA Actions	4
2.2	DRAMA Parameters for devices	6
2.3	Other DRAMA Parameters provided by this task	6
2.4	Environment Variables	8
2.5	Device Specific Simulation	8
2.6	VxWorks Machine and System Startup.	8
2.7	Logging System	8
<b>3</b>	<b>Task Design</b>	<b>10</b>
3.1	Overview of task	10
3.1.1	<i>Main module</i> .....	10
3.1.2	<i>Stepped Devices</i> .....	10
3.1.3	<i>Wheels</i> .....	11
3.1.4	<i>Message Encoder</i> .....	11
3.1.5	<i>Input Handler</i> .....	11
3.1.6	<i>Stepper Motor Controllers</i> .....	11
3.1.7	<i>Electromagnet System</i> .....	11
3.2	Locking	11
3.3	Files	12
<b>4</b>	<b>Source Code and Building</b>	<b>12</b>
<b>A</b>	<b>Controller board interface</b>	<b>13</b>
A.1	Microcode Source Files etc.	13
A.2	Device numbers	13
A.3	Microcode functions	13
A.4	I/O allocation	14
A.4.1	<i>Outputs</i> .....	14
A.4.2	<i>Inputs</i> .....	14
A.4.3	<i>Electro-Magnet Control</i> .....	15
<b>B</b>	<b>Engineering Interface</b>	<b>16</b>
B.1	Starting Up.	16
B.1.1	<i>Start-up Options</i> .....	16
B.2	General Operation.	17
B.3	Commands and Status	18
B.3.1	<i>Wheel Mechanisms</i> .....	18

B.3.2	<i>Focus Mechanism</i> .....	19
B.3.3	<i>Locks</i> .....	19
B.3.4	<i>Telemetry</i> .....	20
B.3.5	<i>Electromagnets</i> .....	20
B.3.6	<i>Other Commands</i> .....	20
B.3.6.1	TEST RUN .....	20
B.3.6.2	AUTOFOCUS .....	20
B.3.6.3	POLL/KICK POLL .....	20
B.3.6.4	EXIT TASK .....	20
B.3.6.5	EXIT GUI .....	20
B.3.6.6	Reset .....	20
B.3.6.7	LEDs On/LEDs Off .....	20
B.3.6.8	UNLOCK .....	20
B.4	Testing Using Scripts.	20
B.5	Engineering GUI Source.	21
<b>References</b>		<b>22</b>

## Revisions:

- 08-Dec-2000: Version 1.1, last prior to Revision list.
- 07-May-2001: Version 1.2, TJF, Added Download action. Add the Files section that describes the various files.
- 20-Jun-2001: Version 2.0. TJF. Noted FINDHOME argument to FOCUS command. Added ELECTRO\_SET and ELECTRO\_GET actions. Added \_SIM parameters for each mechanism. Added INITIALISED parameter. Added mention of logging system and Log File directory argument to INITIALISE action. Added details of RESET command.
- 05-Jun-2002: Version 3.0. TJF. General update after commissioning. Drop the RESETCARD action, now done by "RESET HARD". Add LEDS\_ON, LEDS\_OFF and TELEM\_CHECK actions. Update parameter list. Drop section on main GUI – now part of IRIS 2 documentation. Drop sections on PC based simulator and test jog – no longer needed. Add section and source code and building. Put into standard ASD template.

## 1 Introduction

This document describes the IRIS 2 Dewar Control Software. The purpose of this software is to control the spectrograph wheels and detector focus, and to monitor the state of the Dewars for the IRIS2 instrument. Reference 3 provides more details about the instrument itself.

### 1.1 Instrument Electronics.

The instrument electronics is fully described in reference 4. There are four wheels and a focus mechanism controlled by stepper motors. A LED and optical sensor scheme is used to encode the defined positions of the four wheels whilst limit switches and a home switch are used to determine the position of the focus mechanism.

Most of the electronics is controlled via a “Radio-Spares 440” stepper motor controller (stock no. 440-098). This programmable device provides for simultaneous control of two stepper motors axes and provides eight output bits and either input bits. Purpose built electronics is provided to implement a multiplexing scheme that allows for control of up to either stepper motors. The scheme also allows for sensing of Dewar telemetry, reading of the encoder values and sensing of fault conditions. The RS440 controller is operated via RS232 from a VME system running the VxWorks operating system.

Additional electronics is used to control the wheel Detent mechanisms. From the electronics viewpoint, this is a set of electromagnets. The Parallel port of the VME system is used to control the electromagnets.

### 1.2 Software

A DRAMA<sup>1</sup> task named “SPECTRO” provides a networked and interlocked interface to the IRIS 2 Dewar electronics. This task is the main subject of this document. In normal operations, the SPECTRO task is started on the IRIS 2 VME system by the IRIS 2 Observer software. The IRIS 2 Observer software then initialises and runs the task as required.

The RS440 controller is programmable. Various functions have been developed in RS440 microcode and are downloaded at Initialisation/Reset to extend the functionality of the controller.

An engineering interface is provided to allow debugging of the hardware and software. This is documented in section B.

Various simulation modes are provided. The basic simulation mode allows the SPECTRO task to be run on a Unix or VxWorks machine without any hardware. Hardware responses are simulated or presumed. This mode allows for debugging of the software. A second mode allows replies from the RS440 controller to be entered

---

<sup>1</sup> The AAO’s distributed instrumentation control system see-  
<http://www.aao.gov.au/drama/html/dramaintro.html>.

by hand (this mode has not been tested recently). A third mode of simulation involves the hardware being generally presumed to be operating, but individual mechanisms may be simulated – to allow operation of a partially working Dewar.

## 2 Software Interface

This section describes the SPECTRO task via its DRAMA interface

### 2.1 DRAMA Actions

The following Actions (commands) will be accepted by the SPECTRO task:

COMMAND	DESCRIPTION
INITIALISE [ <i>logdir</i> ]	<p>Initialise this task and home all the wheels. The argument if any is the name of a directory where the log file will be written.</p> <p>This operation is equivalent to a “RESET FULL” – see below.</p>
RESET [ <i>type</i> ]	<p>Reset the task. The argument, if any, should be one of SOFT/FULL/HARD.</p> <p>The default reset mode is SOFT, which causes the software to be re-initialized and the mechanisms to be homed (unless the AUTOHOME flag is false).</p> <p>If the FULL flag is specified, then the RS440 microcode is also downloaded to the RS440 controller, before the home operation. This download will clear any microcode previously downloaded.</p> <p>If the HARD flag is specified, then the stepper motor control card will also be hard reset before a FULL reset is commenced. A Hard reset uses a scheme which provides a complete reset of the stepper motor controller, stopping immediately any motion in progress.</p>
SLIT <i>args</i>	Move aperture/slit wheel
FILTER <i>args</i>	Move filter wheel
COLDSTOP <i>args</i>	Move cold stop wheel
GRISM <i>args</i>	Move grism wheel
FOCUS <i>steps/fh</i>	Move the detector to a new focal offset or specify FINDHOME to cause the home position to be found again.
CONFIGURE <i>cfg_args</i>	Configure all devices in the spectrograph. Cfg_args must contain a single argument for each of the above device actions.
AUTOFOCUS	Wait until all current wheel movements have completed, and then move to the focal offset found in a lookup table based on the positions of the filter, coldstop and grism wheels.

COMMAND	DESCRIPTION
RSCMD <i>string</i>	Send a command directly to an RS Stepper Motor Controller board. String is the command, and must not contain spaces.
DOWNLOAD <i>file</i>	Download the specified file to the RS Stepper Motor Controller board. String is the name of the file, with a default file type of “.rs440”. This file must be assessable to the VxWorks machine on which the SPECTRO task is running. Depending on the contents of this file, the original microcode may be overridden, impacting the ability of this program to control the Dewar.
POLL	Initiate a periodic poll to read the dewar temperature, pressure, and any other diagnostics.
EXIT	Terminate this task.
ELECTRO_SET <i>bit</i>	Used to test the electromagnet system. The argument is an integer giving the bit number (0-7) of a electromagnet bit to set. If the argument is not supplied, the current value is returned. Note that only two electromagnets may be active at any time and this command will prevent any more being made active. Also note that you should not leave electromagnets active for more then a few minutes, as they may heat up the Dewar.
ELECTRO_CLEAR <i>bit</i>	Used to test the electromagnet system. The argument is an integer giving the bit number of a electromagnet bit to clear. If the argument is not supplied, the current value is returned.
TELEM_CHECK	Check the Dewar Telemetry. This is normally run by the POLL action.
LEDS_ON	Turn the Dewar LEDs on. Normally the LEDS are turned on an off as required to minimize stray light in the Dewar. This action allows them to be turned on continuously for debugging purposes.
LEDS_OFF	Turn the Dewar LEDS off.
LOG_LEVEL <i>levels</i>	Set the task logging level. See section 2.6 for more details of logging and the argument..
SIMULATE_LEVEL <i>level</i>	Set the task simulation level. The argument should be either NONE or FULL and is used to set the SIMULATE_LEVEL parameter value.

**Notes:**

1. *args* may be one of the following:
  - (a) STEP *s* - Move wheel to step *s*.
  - (b) FINDHOME - Move wheel to home position as indicated by wheel encoders.
  - (c) CHECKPOS - Turn on LEDs and update parameters with encoded wheel position.
  - (d) ANGLE *a* - Rotate wheel to angle *a*. (Angle is in degrees)
  - (e) *n* - Move wheel to encoded position *n* ( $1 \leq n \leq 14$ ).

- (f) *str* - Move wheel to encoded position represented by the string *str*. This string must match an entry in the wheel config file for this wheel
2. *step* is the requested step number for the device.
  3. *fh* represents the string FINDHOME.

## 2.2 DRAMA Parameters for devices

A number of DRAMA parameters are created and set by the spectrograph task for each device.

The names of the parameters are formed by adding a suffix to the device name. The following suffixes are used:

- *\_ST* - The current step number of the device. (integer)
- *\_PN* - The current position number of the device. (integer)
- *\_PS* - The current position of the device as a string. (string). These strings come from the wheel configuration file, or may be values such as “-Unknown-“, “--Homing-“, “--Checking-“ or “-Moving-“, depending on the current state of the software.
- *\_AN* - The current angle of the device. (real)
- *\_SIM* - Indicates if this device is being simulated. (int). 0 for not simulating, 1 for simulating.

DEVICE	PARAMETERS
SLIT	SLIT_ST SLIT_PN SLIT_PS SLIT_AN SLIT_SIM
FILTER	FILTER_ST FILTER_PN FILTER_PS FILTER_AN FILTER_SIM
COLDSTOP	COLDSTOP_ST COLDSTOP_PN COLDSTOP_PS COLDSTOP_AN COLDSTOP_SIM
GRISM	GRISM_ST GRISM_PN GRISM_PS GRISM_AN GRISM_SIM
FOCUS	FOCUS_ST FOCUS_SIM

## 2.3 Other DRAMA Parameters provided by this task

In addition to the above parameters, the following parameters are provided.

PARAMETER	TYPE	DESCRIPTION
RESPONSE_LOCK	Integer	Lock indicator for RS232 channel
DRIVE_LOCK_0	Integer	Lock indicator for drive address on controller board 0
DRIVE_LOCK_1	Integer	Lock indicator for drive address on controller board 1
DEVICES_ACTIVE	Integer	The number of devices currently active
POLL_PARAMETER	Real	Poll period (Default period = 10 seconds)
TELEM_BIT_0	Integer	Main Dewar Telemetry bit 0 – Main Dewar Over Pressure
TELEM_BIT_1	Integer	Main Dewar Telemetry bit 1 – Main Dewar Over Temp
TELEM_BIT_2	Integer	Main Dewar Telemetry bit 2 – Main Dewar Under Temp
TELEM_BIT_3	Integer	Main Dewar Telemetry bit 3 - Main Dewar Over Heat
TELEM_BIT_4	Integer	Fore Dewar Telemetry bit 0 – Fore Dewar Over Pressure
TELEM_BIT_5	Integer	Fore Dewar Telemetry bit 1 – Fore Dewar Over Temp
TELEM_BIT_6	Integer	Fore Dewar Telemetry bit 2 – Fore Dewar Under Temp
TELEM_BIT_7	Integer	Fore Dewar Telemetry bit 3 - Fore Dewar Over Heat
TELEM_TEST	Integer	Used when simulating only. The value of this parameter is copied to the telemetry bits each time the TELEM_CHECK action is run. It allows changing telemetry conditions to be simulated.
TELEM_FD_OK	Integer	Set true if the telemetry of the main Dewar is ok.
TELEM_MD_OK	Integer	Set true if the telemetry of the fore Dewar is ok.
TELEM_IGNORE	Integer	If true, then ignore telemetry when working out if it is safe to move a wheel. If false, then you cannot move a wheel if the temperature of the Dewar is too high.
ELECTRO_MAGNETS	Integer	The bit mask representing the current electromagnet setting.
INITIALISED	Integer	Indicates if the task has been initialised. 0 = No, 1 = Yes.
AUTOHOME	Integer	If set true (non-zero) during Initialization or reset, then the motors are homed.
SEARCH_RANGE	Integer	If when moving a wheel to a flagged position, the position is not found, then a search is made +/- this number of steps from the specified position.

PARAMETER	TYPE	DESCRIPTION
LEAVE_LEDS_ON	Integer	If true (non-zero) then the LEDs are left turned on after reading the position sensors. This parameter is normally set true by the LEDS_ON action and cleared by the LEDS_OFF action.
SIMULATE_LEVEL	String	The simulation level. This is read at Initialisation/Reset to determine the simulation level. If empty, then the simulation level is read from the SPECTRO_SIM environment variable. Otherwise, it should be NONE or FULL. The default is not to simulate and after Initialisation/Reset, this parameter will contain the actual simulation value.

## 2.4 Environment Variables

The following environment variables are used by the software.

**SPECTRO\_SIM** - Set to 1 to enable full simulation mode. Only used during INITIALISE/RESET actions. See the SIMULATE\_LEVEL parameter above for more information.

**SPECTRO\_DIR** - Set to directory that contains wheel config files etc. See section 3.3. Read only during Initialise/Reset.

**SPECTRO\_LOG\_LEVEL** – sets the default logging level. See section 2.6 below.

## 2.5 Device Specific Simulation

Each device can be individually simulated by creating a file within the SPECTRO\_DIR directory. See section 3.3 for more details.

## 2.6 VxWorks Machine and System Startup.

The SPECTRO task normally runs on the VxWorks machine known as “aatvme7”. On boot-up, this VxWorks machine will run the script /instsoft/iris2/App1/Startup/aatvme7, which is responsible for loading drivers etc. This script will then run /instsoft/iris2/App1/aatvme7, which is responsible for loading DRAMA and the Spectro task into memory. The DRAMA networking is then started using port 12345.

The IRIS 2 Observing software is actually responsible for running the program via DRAMA networking facilities.

## 2.7 Logging System

Logging is implemented using the “GitLogger” object, which can log various details of a DRAMA task. The SYSTEM name used is “SPECTRO”, so the log file will be named SPECTRO-`{date}`.log. The log file directory is optionally specified as the first argument to INITIALISE.

The LOG\_LEVEL action is used to control logging with the default value set by the SPECTRO\_LOG\_LEVEL environment variable. The following logging levels are available

Level	Meaning
STARTUP	Log Startup/Shutdown/Reset messages
ERRORS	Log Error messages
ACTENT	Log entry to an action handler routine.
ACTEXIT	Log exit from an action handler routine.
INST	Log instrument communications.
DEBUG	Not Used
USER1	Not Used
USER2	Not Used
USER3	Not Used
USER4	Not Used
SCREEN	Logging to screen.

We set with the LOG\_LEVEL actions, the argument is a list of logging levels, separated by spaces or comma. The specified logging levels are turned on. Previously enabled logging levels remain in force. To turn a level off, prefix it by 'NO'. Multiple levels can be enabled with each call, separate the words by spaces or comma's. To enable a screen bit, suffix the level name with '-S'.

The level SCREEN is a special case. When supplied, the corresponding screen bits for all other levels are also enabled. When 'NOSCREEN' is supplied, the level is enabled but the corresponding screen bit is disabled, unless a level is suffixed by '-S'.

Levels may be abbreviated to the smallest unique name.

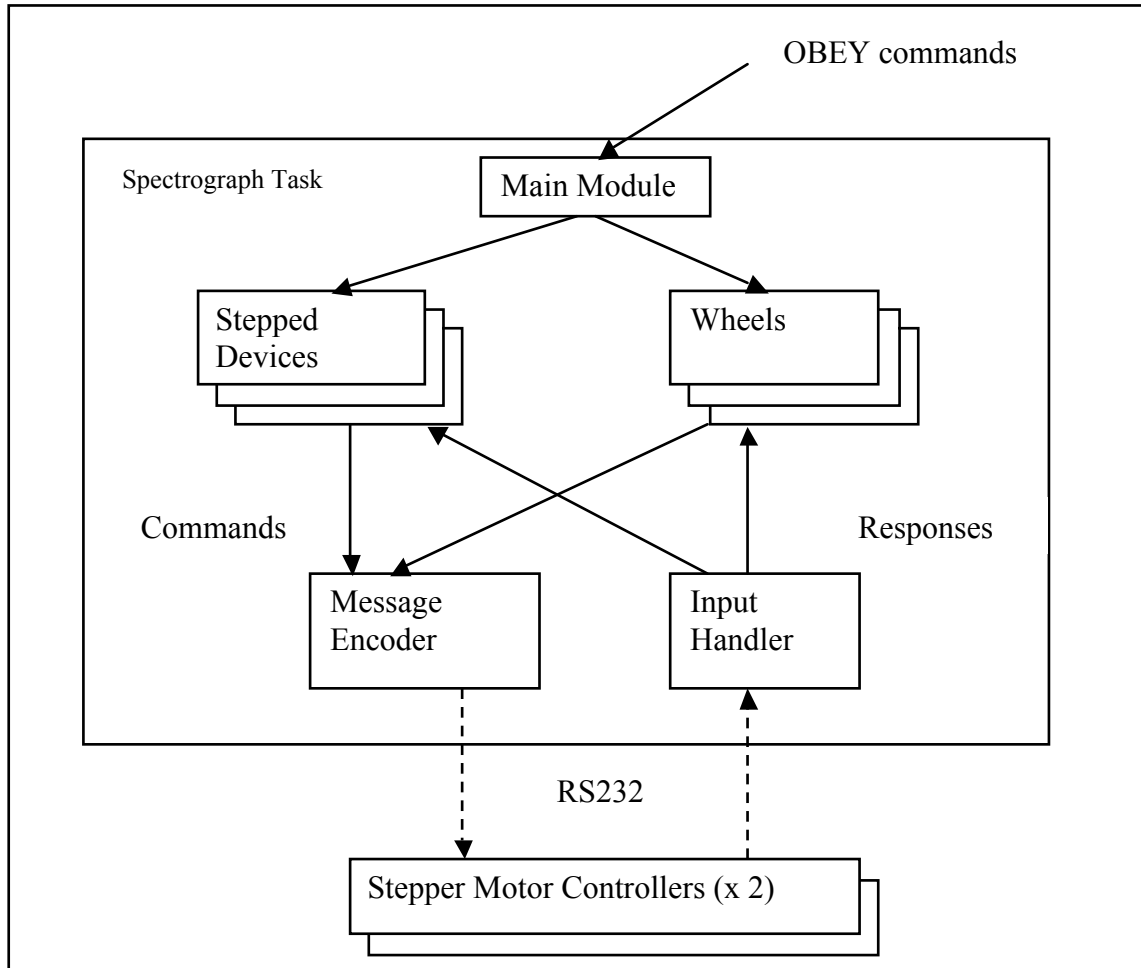
See "GitLogger" for full details.

Under VxWorks – GitLogger uses a low priority background thread to flush the log file every few seconds – ensuring it is up to date.

### 3 Task Design

#### 3.1 Overview of task

The task consists of a main module which receives DRAMA OBEY commands, and C++ objects representing each of the devices controlled in the spectrograph.



##### 3.1.1 Main module

The main module creates all the device objects (Stepped Devices, and Wheels) and then calls `DitsAltInLoop()`. `DitsAltInLoop()` processes all the DRAMA OBEY messages and input from the RS232 stream.

Each OBEY command requesting a physical re-configuration of the spectrograph results in the appropriate action routine being executed in the software object corresponding to that device. The action routine sends an RS232 command along the serial line to the appropriate stepper motor controller and then puts itself to sleep. When a response is received on the RS232 stream, the input handler routine triggers the sleeping action routine. The action then sets the appropriate DRAMA parameters and completes.

##### 3.1.2 Stepped Devices

This set of objects represent the devices that the user positions using a step number. Currently, the only device of this type is the Detector focus.

### 3.1.3 Wheels

These objects represent devices that the user positions using a position number (an integer between 1 and 16), a string which maps to a position number, a step number, or an angle. The mappings between strings and position numbers are stored in a wheel configuration (\*.cfg) file.

### 3.1.4 Message Encoder

This module contains routines to generate RS232 messages on the serial link between the spectrograph task and the stepper motor controllers.

### 3.1.5 Input Handler

This routine handles all input from the RS232 stream. For each RS232 response message, it triggers the appropriate action routine using DitsSignal().

### 3.1.6 Stepper Motor Controllers

The RS controller boards are programmable in a C-like language (see ref [5]). New routines can be downloaded via the RS232 line. The enhanced EPROM software (RS stock no. 718-830) is used facilitate the development of these routines.

The C++ class rs440io is responsible for accessing the stepper motor controllers.

### 3.1.7 Electromagnet System

The four worm driven wheel assemblies contain an electromagnet which when energized, pulls the worm drive away from the wheel. This ensures there is no wedge effect between the worm and the gear i.e. the worm drives only on one face of the gear. This provides clearance and reduces the load on the motor.

The electromagnets are each controlled from separate data bits driven by the MVME147 single board computer centronics parallel printer port. This port is accessed using the parallel port driver.

The C++ class electroMag is responsible for controlling this system.

## 3.2 Locking

Locking is needed since only a single RS232 channel connects both stepper motor controller boards to the VxWorks machine and up to four pairs of motors are multiplexed from each axis of each controller board.

Two levels of locking are used. These are:

1. Drive locking; and
2. Response locking.

Drive locking is at the level of a controller board. Each board is connected to four drives, each of which can be connected to up to two motors (one per axis). Drive locking ensures that the drive address can not be changed while either of the motors for that drive are active.

Response locking is at the level of the RS232 link to the controller boards. This locking is needed since each command issued to a controller board will usually result in a response. If multiple commands were allowed to be issued at once to control different devices then there would be no way to determine which command a response was related to. There is only one response lock.

### 3.3 Files

Various files are used to configure the system. This should be placed in the directory pointed to by the SPECTRO\_DIR environment variable (created automatically by the dramastart command).

The following files are used

File	Usage
Devices.cfg	Used to enable each supported device. For each device (COLDSTOP/FILTER/GRISM/SLIT/FOCUS) it indicates the mechanism type, controller number, device number and limits (if any) applied. This file can be modified to disable devices if required, but this is no longer required – see {devname}.sim below for an alternative approach. See the file itself for full details.
COLDSTOP.cfg	Gives the names, step positions etc for the COLDSTOP wheel. Only read if “devices.cfg” enables this wheel. See the file for details of the format.
FILTER.cfg	As per COLDSTOP.cfg for the FILTER wheel.
GRISM.cfg	As per COLDSTOP.cfg for the GRISM wheel
SLIT.cfg	As per COLDSTOP.cfg for the SLIT wheel
Prog.rs440	The default module to be downloaded to the RS Stepper Motor control. This contains the various routines used to operate the devices.
{devname}.sim	Where devname is one of FOCUS/ COLDSTOP/ FILTER/ GRISM/ SLIT. The contents of these files are ignored. If the file exists then the specified device will be simulated, even when task level simulation is disabled. This allows a particular device to be disabled whilst control tasks continue to see the same parameter and command set being available. These values are re-read on each Initialise/Reset operation.

## 4 Source Code and Building

The source code for this program is found in the SCCS library aatssy:/instsoft/iris2/spectro/SCCS. It is maintained and build using normal AAO standards – see reference 6.

## A Controller board interface

This appendix describes the routines that are downloaded to the RS stepper motor controller boards at startup. These routines form the interface through which the spectrograph task running on the VxWorks machine will control and monitor the stepper motors.

### A.1 Microcode Source Files etc.

In normal operations, the file “prog.rs440” is downloaded during Initialisation/ Hard Reset and defines the various procedures etc. used. It should be noted that the RS440 controller has little memory and it may not be possible to extend the microcode at all. It should also be noted that during downloading, comments and excess white space are deleted, since memory would otherwise be used to store these in the controller.

A second microcode source file – findpoint.rs440, is available. This is used to find the location of positions on the wheels. It is normally downloaded after initialisation using the “DOWNLOAD” action. The “FindPoint()” function is then used to perform the operation in question. See findpoint.rs440 for details.

### A.2 Device numbers

For simplicity of table lookups, a single number is used to refer to each device. This device number is formed by multiplying the drive number (0..3) by two and adding it to the axis number (0..1). Thus, even and odd numbered devices are on different axes. Using this numbering scheme, only devices  $n$  and  $n+1$ , where  $n$  is even, may be driven simultaneously.



### A.3 Microcode functions

Function/Procedure	Description
<b>LedsOn()</b>	Turn position sensing LEDs on.
<b>LedsOff()</b>	Turn position sensing LEDs off
<b>Ax(a)</b>	Set Axis Select bit. ( $a=0$ or $1$ )
<b>Drv(dr)</b>	Set Drive Select bits for drive $dr$ . ( $dr$ in range $0..3$ )
<b>En(a)</b>	Turn Motor Enable bit on for axis $a$
<b>Dis(a)</b>	Turn Motor Enable bit off for axis $a$ .
<b>HOn(a)</b>	Turn homing bit on for axis $a$ .
<b>HOff(a)</b>	Turn homing bit off for axis $a$

Function/Procedure	Description
<b>FindHome(d)</b>	Move device <i>d</i> to home position and zero step count. (This procedure enables LEDs and motor during the move. Both are disabled afterwards). This procedure does not complete until the device has been homed.
<b>FocHome(d)</b>	Similar to FindHome() but used to home a focus mechanism.
<b>Goto(d,n)</b>	Initiate move of device <i>d</i> to step <i>n</i> . NOTE: The motor for the appropriate axis ( <i>d</i> &1) must be enabled before this procedure is called, and must remain enabled until the move has completed.
<b>IsMoving (d)</b>	Return least significant bit = 1 if device <i>d</i> is moving, 0 otherwise. Divide the result by 2 to get the current step number for the device (this allows IsMoving() to be used to monitor the motion and position of a device)
<b>ReadPos(d)</b>	Return sensed position for device <i>d</i> . (Values 1 thru 14 are valid, 0 is invalid, and 15 indicates an error condition). To avoid latching problems, the appropriate motor enable bit should also be off. NOTE: this function sets the axis select and drive select bits.
<b>ReadTelem(a)</b>	read telemetry nibble a

#### A.4 I/O allocation

##### A.4.1 Outputs

BIT	LABEL	NAME	EXTRA NOTES
0	PLE	Pos LED Enable	False/True selects telemetry/position.
1	ASL	Axis Select	False/True selects X/Y data bits.
2	DA0	Drive Address 0	Must not change while motor enabled.
3	DA1	Drive Address 1	Must not change while motor enabled.
4	XME	X Motor Enable	When true, X data bits set to 15 if PLE true.
5	YME	Y Motor Enable	When true, Y data bits set to 15 if PLE true.
6	XMH	X Motor Home	Stops X motor when RD true.
7	YMH	Y Motor Home	Stops Y motor when RD true.

##### A.4.2 Inputs

BIT	LABEL	NAME	EXTRA NOTES
0	0DB	0 Data Bit	When LED Enable low, telemetry bit 0
1	1DB	1 Data Bit	When LED Enable low, telemetry bit 1
2	2DB	2 Data Bit	When LED Enable low, telemetry bit 2

3	3DB	3 Data Bit	When LED Enable low, telemetry bit 3
4	RD	Reference Detent	Latched when MH true. Only valid if PLE true.
5	DFM	Drive Fault Motor	0 if motor supply lost
6	DFP	Drive Fault Position	0 if fault (valid if PLE=true)
7	DFL	Drive Fault Limit	0 if fault (valid if ME=true, and drive has limits).

#### A.4.3 Electro-Magnet Control

Bit	Usage
0	Slit Wheel Detent.
1	Filter Wheel Detent
2	Coldstop Wheel Detent
3	Grism Wheel Detent
4 .. 8	Unused.

## B Engineering Interface

An engineering user interface has been written in Tcl/Tk that provides low-level access to the IRIS2 motor control system.

### B.1 Starting Up.

To start the engineering interface, execute the command “iris2eng” from the AATOBS or IRIS2TES account. Note that if IRIS 2 is already being run in any form on the machine in question, then you must use the same account.

The engineering interface will automatically try to find the SPECTRO task on machine aatvme<sup>72</sup>. If not found to be running, it is started automatically. If the task is started or if found running but has not been initialised, then the interface is presented as in Figure 1. Figure 2 attempts gives a summary of each component of this interface.

In normal operations, it is sufficient to invoke the “INITIALISE” button at this point. This will initialise the SPECTRO task, homing all the motors etc. If this completes successfully, the INITIALISE button is removed from the interface.

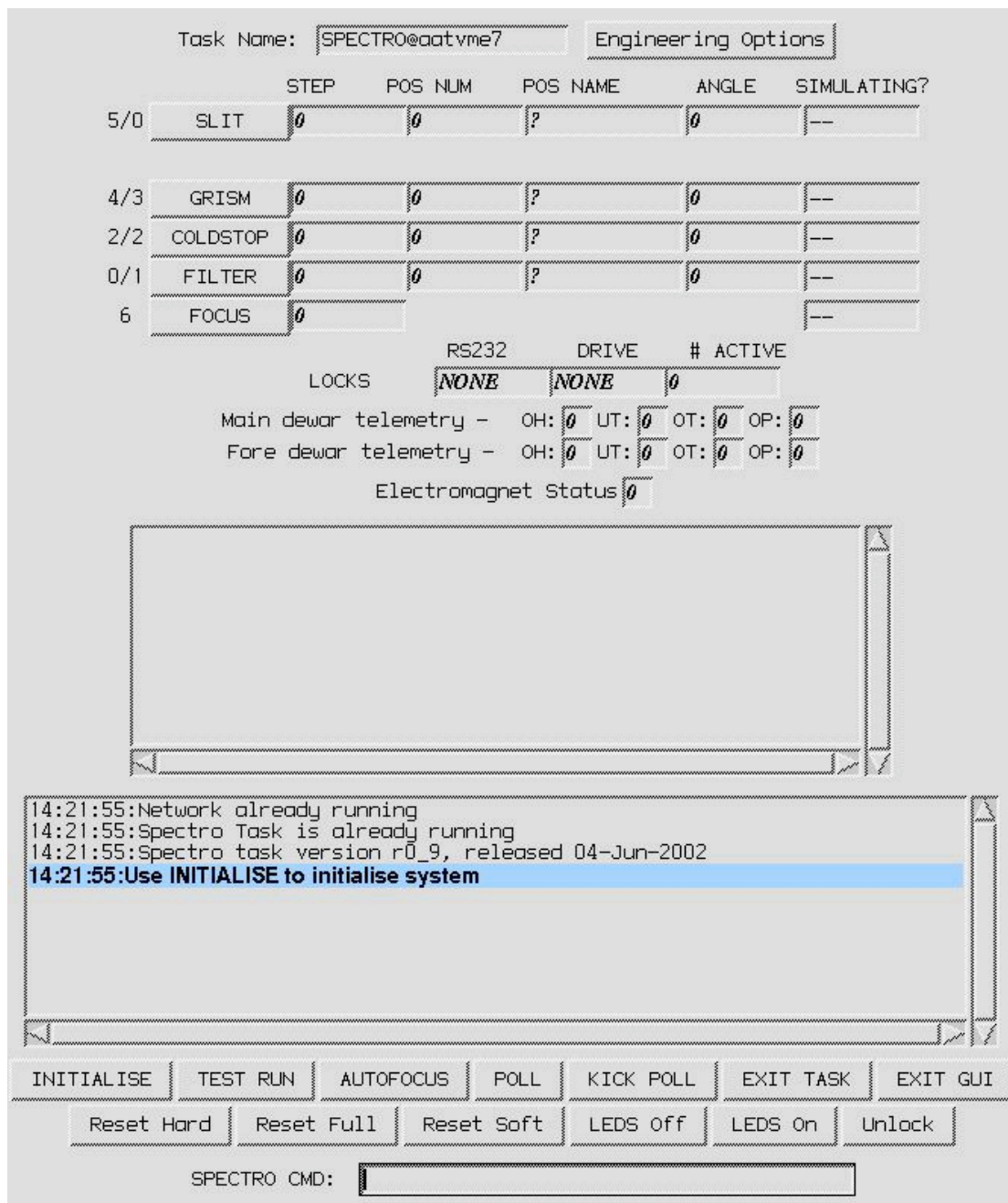
#### B.1.1 Start-up Options

The “Engineering Options” menu provides several options that are used during Intialise/Reset. These are related to particular parameters, see section 2.3. The options are described below.

Option	Usage	Parameter
Full Simulation	If this check button is set, then the task is initialised with full simulation. Otherwise it is initialised in the normal fashion – controlling the hardware. Normally clear.	SIMULATE_LEVEL
Auto Home	If this check button is set, then Initialise/Reset commands will cause the motors to be homed. Normally set.	AUTOHOME
Ignore Telemetry	If this check button is set, then Telemetry is not checked prior to moving the motors. Normally this should be clear (but at the moment it is normally set).	TELEM_IGNORE

You should set these appropriately before using Initialise/Reset.

<sup>72</sup> Note this implies that if you wish to run in simulation under UNIX, you must start the SPECTRO task running on UNIX before starting the engineering interface. You can do this with the command \$SPECTRO\_DEV/spectro &



**Figure 1: Engineering Interface Pre Initialisation**

## B.2 General Operation.

The “Command Sent” log shows the list of commands sent to the SPECTRO task, whilst the “Message Log” shows all replies, errors etc. Red is used to highlight errors and blue to highlight other messages of significance.

For the various drop-down menus, whilst they dropped down clicking on the “dashed line” will cause the menu to separate. This makes many operations easier. The menu can then be closed using the window manager close window facility (window manager specific, but normally found in the window decorations).

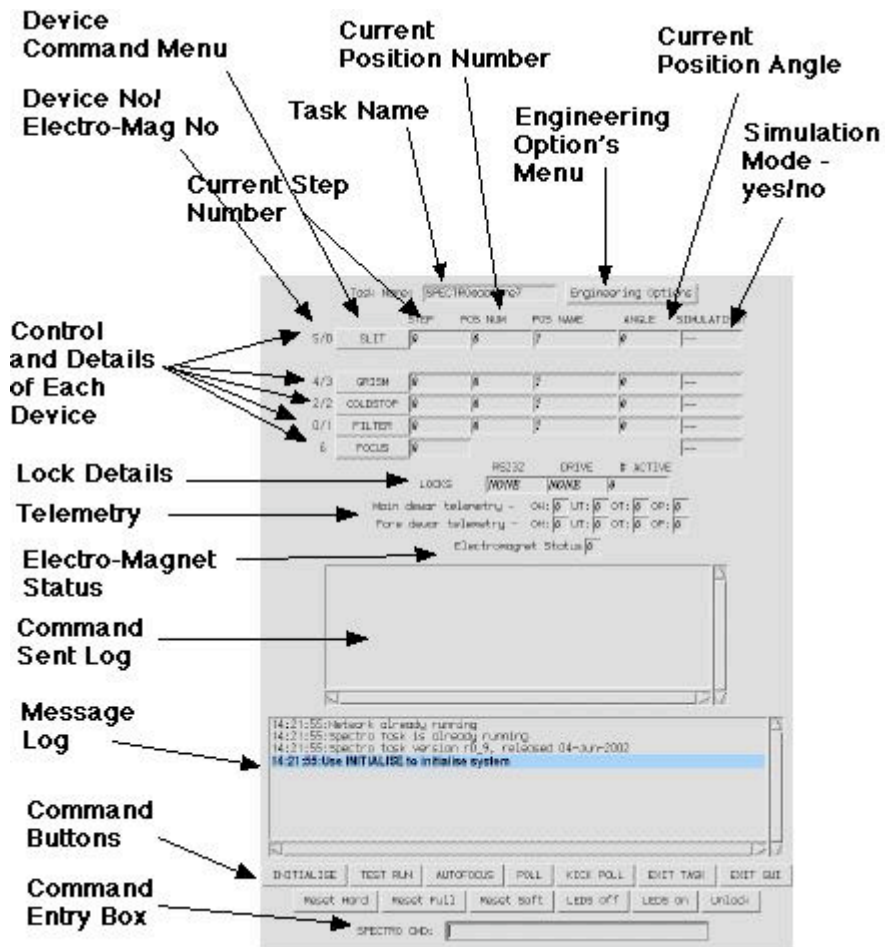


Figure 2: Engineering Interface Details

### B.3 Commands and Status

The engineering interface contains various menus, status display boxes and command buttons to allow control of the instrument. Additionally, you may enter a command in the “Command Entry Box” at the bottom of the window. Hitting return here causes the command to be sent. This allows any other commands not provided elsewhere to be entered.

#### B.3.1 Wheel Mechanisms

Each wheel has a menu and a set of displays giving the current step number, current position (flag) number, name of current flagged position, current angle and an indication if the device is simulating. Figure 3 shows these details for the SLIT mechanism just after initialisation, when simulating. Note the “5/0” which starts the line – this indicates that the slit is mechanism 5 and uses electromagnet bit 0. The “SLIT” button can be used to produce the menu of Slit commands – see below. In this case the Slit wheel is at step number 0, which is position, known as “CLEAR” at angle 0. The wheel is being simulated.

	STEP	POS NUM	POS NAME	ANGLE	SIMULATING?
5/0	SLIT	0	CLEAR	0	yes

**Figure 3: Slit Mechanism Details**

For each wheel, the corresponding menu provides commands to go to each valid position. For the slit, there are 7 valid positions (unfortunately, these positions are not available by name at this stage).

Additional commands are available to abort moves, explicitly find home, Cycle the mechanism through each available position, Abort a Cycle and to operate the Detent Magnets.

To go a step position, you must use the command entry box to enter a command of the form

SLIT STEP n

Where n is the step number.

### B.3.2 Focus Mechanism

The focus mechanism does not have defined positions, simply operating within a range of about +/- 1000. Hence a much reduced status display and fewer commands.

Commands are available to move to -600/0/600 and -1000/1000, the later with checking of limits turned off (to check the full mechanical range). Abort Move and Find Home are also available.

To move the FOCUS to any other position, you must use the Comment Entry Dialog, entering a command link "FOCUS n" where n is the step number.

### B.3.3 Locks

The engineering interface display indicates if the RS232 communications is locked, if any drive is locked or if any devices are active. See section 3.2 for more details on locking. The "Unlock" command at the bottom right of the engineering interface can be used to explicitly clear the locks – if an error has occurred.

Goto Position 1	
Goto Position 2	
Goto Position 3	
Goto Position 4	
Goto Position 5	
Goto Position 6	
Goto Position 7	
-----	
Abort Move	
Find Home	
-----	
Cycle	
Abort Cycle	
-----	
Magnet On	
Magnet Off	

**Figure 4: The Focus Menu**

Move to -600	
Move to 0	
Move to 600	
-----	
Move to -1000, Lmt Chk Off	
Move to 1000, Lmt Chk off	
-----	
Abort Move	
Find Home	

**Figure 5: Focus Menu**

### **B.3.4 Telemetry**

The engineering interface displays the four telemetry bits for each Dewar.

### **B.3.5 Electromagnets**

The status of electromagnets used to control the wheel Detent mechanisms are displayed as a hexadecimal number on the engineering interface. The corresponding bit for each wheel is the second number just to the left of the menu button for each wheel. For example, in Figure 1 the electromagnet bit for the slit wheel is bit 0, the bit for the Grism wheel is bit 3. When if both these are on, you would expect the status to be 5. You can explicitly set or clear an electromagnet using the `ELECTRO_SET` and `ELECTRO_CLEAR` commands in the command entry box. The argument is the bit number of the magnet to set/clear.

### **B.3.6 Other Commands**

Various other commands are provided through the engineering interface.

#### **B.3.6.1 TEST RUN**

This command is used to test the operation of all mechanisms. It executes a script that randomly selects a mechanism and then randomly selects a position for that mechanism and then sends that mechanism to the selected position. This runs continuously. The command button is changed to an “Abort Test” button to allow you to stop the test.

#### **B.3.6.2 AUTOFOCUS**

Performs the AUTOFOCUS operation.

#### **B.3.6.3 POLL/KICK POLL**

Polling (of telemetry for example) is normally started automatically by the task which loads/initialisation/resets the SPECTRO task. This includes the engineering interface. These commands allow you to start/stop polling if desired.

#### **B.3.6.4 EXIT TASK**

Sends “EXIT” to the SPECTRO task.

#### **B.3.6.5 EXIT GUI**

Exits the engineering interface, but leaves the SPECTRO task running.

#### **B.3.6.6 Reset**

Each of the three RESET commands are available.

#### **B.3.6.7 LEDs On/LEDs Off**

These allow you to explicitly turn the LEDS on or off.

#### **B.3.6.8 UNLOCK**

Allows you to explicitly clear the system locks.

### **B.4 Testing Using Scripts.**

Due to the use of DRAMA, it is possible to write various scripts to test the instrument, if required. For example, the following is from a DRAMA TCL script named “test1.tcl” can be run with “dtcl -t test1.tcl”

```
for { set i 1 } { $i <= 10 } { incr i } {
  obey SPECTRO@aatvme7 SLIT [args 4] -wait
  puts "Have moved to position 4"
  after 1000
  obey SPECTRO@aatvme7 SLIT [args 1] -wait
  puts "Have moved to position 1"
  after 1000
  obey SPECTRO@aatvme7 SLIT [args 7] -wait
  puts "Have moved to position 7"
  after 1000
  puts "Completed loop $i"
}
```

For more details of what you can do with DRAMA TCL scripts, see reference 7.

### **B.5 Engineering GUI Source.**

The engineering interface source code is a simple Tcl/Tk script intended to be run with the DRAMA “dtk” program. The source is located in the same directory as the SPECTRO task and is called **eng\_gui.tcl**. It is released into the SPECTRO\_DIR directory.

## References

- [1] Tony Farrell (AAO), 1996, *DRAMA – The Anglo-Australian Observatory's distributed instrumentation control system*.  
<http://www.aao.gov.au/drama/html/dramaintro.html>.
- [2] Tony Farrell (AAO), 18-Jan-94, *Generic Instrumentation Task Specification*.  
[http://www.aao.gov.au/drama/doc/ps/git\\_spec\\_9.ps](http://www.aao.gov.au/drama/doc/ps/git_spec_9.ps).
- [3] Chris Tinney (AAO), 6/12/98, *IRIS2 System and Technical Specification, version 3.0*.
- [4] Brian Hingley (AAO), Aug/Sep 1998, *IRIS2 Electronics Specification*.
- [5] RS Components, Nov 1993, *Programmable Stepper Motor Control Board (2 axis) (stock no. 440-098)*.
- [6] Tony Farrell (AAO), Anglo-Australian Observatory Programming Standards.  
<http://aaossi.aao.gov.au/softdocs/AAOProgrammingStandards.pdf>.
- [7] Jeremy Bailey, Tony Farrell (AAO), 15-Nov-1997, DTCL - DRAMA Tcl Interface, [http://www.aao.gov.au/drama/doc/ps/dtcl\\_12.ps](http://www.aao.gov.au/drama/doc/ps/dtcl_12.ps).