Contents

Classes

SDIAutoParallel

No Doc

Inherits from: **XDIBASE**
Class Data:

(long)	id	(string)	status	(float)	wavelength
(double)	start_time	(float)	param	(int)	step
(int)	nominal	(int)	leg1	(int)	leg2
(int)	leg3	(int)	curr_leg	(int)	param_pos
(ptr)	ref_image	(int)	get_ref_flag	(string)	obj_num
(structure)	geometry	(int)	need_frame	(int)	need_timer
(int)	auto	(structure)	palette	(obj)	manager

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdiautoparallel__define.pro

---

METHODS:

(function) INIT

Method Documentation:
No Doc
Arguments:

`data=data`: No Doc

`restore_struc=restore_struc`: No Doc

Example Call:

```
result = SDIAutoParallel->init(  data=data,
                                 restore_struc=restore_struc)
```

(pro) CLEANUP

Method Documentation:
No Doc
Arguments:

`log`: No Doc

Example Call:

```
SDIAutoParallel->cleanup,  log
```

(pro) FRAME_EVENT

Method Documentation:
No Doc
Arguments:

`image`: No Doc

`channel`: No Doc

Example Call:

```
SDIAutoParallel->frame_event,  image,
                               channel
```

(function) GET_SETTINGS

Method Documentation:
No Doc
Takes no arguments
Example Call:

```
result = SDIAutoParallel->get_settings(  )
```

(pro) START_PARALLEL

Method Documentation:
No Doc
Arguments:

`event`: No Doc

Example Call:

```
SDIAutoParallel->start_parallel,  event
```

(pro) STOP_PARALLEL

Method Documentation:
No Doc
Arguments:

`event`: No Doc

Example Call:

```
SDIAutoParallel->stop_parallel, event
```

SDIEtalonScanner

The EtalonScanner plugin lets you continuously scan the etalon over one order of interference at a given wavelength, and optionally pause during a scan.

Inherits from: **XDIBASE**
Class Data:

(long)	id	(string)	status	(float)	wavelength
(double)	start_time	(int)	nchann	(string)	obj_num
(structure)	geometry	(int)	need_frame	(int)	need_timer
(int)	auto	(structure)	palette	(obj)	manager

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdietalonscanner__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Initialize the EtalonScanner.
Arguments:

`data=data`: Misc data

`restore_struc=restore_struc`: Restored settings

Example Call:

```
result = SDIEtalonScanner->init(  data=data,
                                  restore_struc=restore_struc)
```

(pro) CLEANUP

Method Documentation:
Cleanup, stop any current scans.
Arguments:

`log`: No Doc

Example Call:

```
SDIEtalonScanner->cleanup,  log
```

(pro) FRAME_EVENT

Method Documentation:
A new frame has been recieved. Update leg diagrams, decide if we need to start a new scan.
Arguments:

`image`: The new camera frame

`channel`: The current scan channel

Example Call:

```
SDIEtalonScanner->frame_event,  image,
                                channel
```

(function) GET_SETTINGS

Method Documentation:
Select settings to save.
Takes no arguments
Example Call:

```
result = SDIEtalonScanner->get_settings(  )
```

(pro) PAUSE_SCAN

Method Documentation:
Pause the current scan.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonScanner->pause_scan,  event
```

(pro) SET_WAVELENGTH

Method Documentation:
Set the wavelength for scanning.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonScanner->set_wavelength, event
```

(pro) START_SCAN

Method Documentation:
Start a scan.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonScanner->start_scan, event
```

(pro) STOP_SCAN

Method Documentation:
Stop the current scan (will restart from beginning on next 'start')
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonScanner->stop_scan, event
```

SDIEtalonSpacer

The EtalonSpacer plugin allows you to adjust the etalon plate separation at each leg. You can control each leg individually, or adjust paralellism along two orthogonal axes.

Inherits from: **XDIBASE**
Class Data:

(long)	id	(string)	status	(int)	step
(int)	leg1	(int)	leg2	(int)	leg3
(string)	obj_num	(structure)	geometry	(int)	need_frame
(int)	need_timer	(int)	auto	(structure)	palette

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdietalonspacer__define.pro

---

METHODS:

(function) INIT

Method Documentation:
EtalonSpacer initialization.
Arguments:

`data=data`: Misc data

`restore_struc=restore_struc`: Restored settings

Example Call:

```
result = SDIEtalonSpacer->init(  data=data,
                                 restore_struc=restore_struc)
```

(pro) ADJUST_LEGS_EVENT

Method Documentation:
An event from the widget sloders representing leg voltages.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonSpacer->adjust_legs_event,  event
```

(pro) CLEANUP

Method Documentation:
Cleanup - nothing to do
Arguments:

`log`: No Doc

Example Call:

```
SDIEtalonSpacer->cleanup,  log
```

(function) GET_SETTINGS

Method Documentation:
Get settings for saving.
Takes no arguments
Example Call:

```
result = SDIEtalonSpacer->get_settings(  )
```

(pro) STEP_CHANGE

Method Documentation:
Change the size of the tilt adjustment.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonSpacer->step_change,  event
```

(pro) TILT

Method Documentation:
A tilt event, for adjusting along the two orthogonal axes.
Arguments:

`event`: Widget event

Example Call:

```
SDIEtalonSpacer->tilt,  event
```

SDIPhaseMapper

The Phasemapper plugin records 'phase maps' which encode the scan channel at which a spectrum recorded at the phasemap wavelength peaks for every pixel in the camera frame.

Inherits from: **XDIBASE**
Class Data:

(long)	id	(int)	nscans	(int)	current_scan
(int)	scanning	(int)	nchann	(float)	wavelength
(int)	channel	(ptr)	image	(ptr)	phasemap
(int)	xdim	(int)	ydim	(ptr)	p
(ptr)	q	(ptr)	px	(ptr)	qx
(int)	source_order	(float)	source_lambda	(ptr)	source_pmap
(int)	current_source	(float)	gain	(float)	exptime
(float)	smooth_window	(string)	obj_num	(structure)	geometry
(int)	need_frame	(int)	need_timer	(int)	auto
(structure)	palette	(obj)	manager	(obj)	console

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdiphasemapper__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Phasemapper initialization.
Arguments:

`restore_struc=restore_struc`: Misc data

`data=data`: Restored settings

Example Call:

result = `SDIPhaseMapper->init(` `restore_struc=restore_struc, data=data`)

(function) AUTO_START

Method Documentation:
Auto start the Phasemapper - called whn running in auto mode, and plugin is started from a scheduled command.
Arguments:

`args`: String of arguments passed from the schedule file

Example Call:

result = `SDIPhaseMapper->auto_start(` `args`)

(pro) CLEANUP

Method Documentation:
Cleanup, close any active scans.
Arguments:

`log`: No Doc

Example Call:

```
SDIPhaseMapper->cleanup,  log
```

(pro) FRAME_EVENT

Method Documentation:
Frame event - update the Fourier summations for every pixel, if scan is finished, finalize and unwrap the phasemap, and save it.
Arguments:

`image`: Latest frame from the camera

`channel`: Current scan channel

Example Call:

```
SDIPhaseMapper->frame_event,  image,
                              channel
```

(function) GET_SETTINGS

Method Documentation:
Get settings to save.
Takes no arguments
Example Call:

```
result = SDIPhaseMapper->get_settings(  )
```

(pro) SET_INTERP

Method Documentation:
When using more than one wavelength to generate a phasemap, we set the order of the cal sources (the numbers corresponding to positions of the calibration source selector switch) and the wavelengths those sources correspond to. The info from both phasemaps is store in such a way as to allow the spectral plugin to interpolate between the phasemaps at the two wavelengths.
Arguments:

```
SDIPhaseMapper->set_interp, event
```

`event`: Widget event

Example Call:

(pro) SET_NUM_SCANS

Method Documentation:
Set the number of scans to co-add.
Arguments:

`event`: Widget event

Example Call:

```
SDIPhaseMapper->set_num_scans, event
```

(pro) SET_SMOOTH_WINDOW

Method Documentation:
Set the width of the smoothing window, applied after phasemap is unwrapped.
Arguments:

`event`: Widget event

Example Call:

```
SDIPhaseMapper->set_smooth_window, event
```

(pro) START_SCAN

Method Documentation:
Start scanning.
Arguments:

`event`: Widget event

Example Call:

```
SDIPhaseMapper->start_scan, event
```

(pro) STOP_SCAN

Method Documentation:

Stop the current scan.

Arguments:

`event`: Widget event

Example Call:

```
SDIPhaseMapper->stop_scan, event
```

SDISharpness

No Doc

Inherits from: **XDIBASE**
Class Data:

(long)	id	(float)	sbuffer	(float)	history
(int)	count	(int)	bcount	(float)	best
(int)	leg1_best	(int)	leg2_best	(int)	leg3_best
(int)	xcen	(int)	ycen	(int)	xdim
(int)	ydim	(string)	obj_num	(structure)	geometry
(int)	need_frame	(int)	need_timer	(int)	auto
(structure)	palette	(obj)	manager	(obj)	console

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdisharpness__define.pro

---

METHODS:

(function) INIT

Method Documentation:
No Doc
Arguments:

`restore_struc=restore_struc`: No Doc

`data=data`: No Doc

Example Call:

```
result = SDISharpness->init(  restore_struc=restore_struc,
                              data=data)
```

(pro) CLEANUP

Method Documentation:
No Doc
Arguments:

`log`: No Doc

Example Call:

```
SDISharpness->cleanup,  log
```

(pro) FRAME_EVENT

Method Documentation:
No Doc
Arguments:

`image`: No Doc

`channel`: No Doc

`scan`: No Doc

Example Call:

```
SDISharpness->frame_event,  image,
                            channel,
                            scan
```

(pro) GET_CENTER

Method Documentation:
No Doc
Arguments:

`event`: No Doc

Example Call:

```
SDISharpness->get_center,  event
```

(function) GET_SETTINGS

Method Documentation:
No Doc
Takes no arguments
Example Call:

```
result = SDISharpness->get_settings(  )
```

SDISpectrum

The Spectrum plugin acquires spectra and saves them to netcdf files. It requires a wavelength, a filename to save to and a zonemap file to use for dividing up the field of view. Some things are hard coded in this which should probably be made configurable (see the frame_event method). After each complete exposure, this plugin will attempt to send back a snapshot of the most recent acquisition to an ftp server for real-time data processing (the snapshot is sent back if the logging.ftp_snapshot field is populated in the settings file, see XDIConsole::spectrum_snapshot for the details of this).

Inherits from: **XDIBASE**
Class Data:

(long)	id	(int)	scanning	(int)	nchann
(int)	xdim	(int)	ydim	(int)	save_file_id
(ptr)	spectra	(ptr)	last_spectra	(ptr)	zonemap
(ptr)	zonemap_boundaries	(ptr)	phasemap	(float)	signal_noise_history
(float)	channel_background_history	(float)	scan_background_history	(ptr)	zone_centers
(int)	nzones	(string)	dll	(int)	nscans
(int)	file_id	(string)	zone_settings	(float)	wavelength
(float)	a	(float)	b	(float)	c
(double)	scan_start_time	(string)	spec_path	(int)	nrings
(string)	file_name_format	(string)	filename	(ptr)	rads
(ptr)	secs	(ptr)	accumulated_image	(int)	finalize_flag
(string)	insprof_filename	(float)	etalon_gap	(string)	obj_num
(structure)	geometry	(int)	need_frame	(int)	need_timer
(int)	auto	(structure)	palette	(obj)	manager

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdispectrum__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Spectrum initializer, make sure we have a filename format and a zone map.
Arguments:

`restore_struc=restore_struc`: Restored settings

`data=data`: Misc data from the console

`zone_settings=zone_settings`: A zone settings file name

`file_name_format=file_name_format`: Format for generating the netcdf file name

Example Call:

```
result = SDISpectrum->init( restore_struc=restore_struc,
                            data=data,
                            zone_settings=zone_settings,
                            file_name_format=file_name_format)
```

(function) AUTO_START

Method Documentation:
Auto-start method, called when running in auto-mode.
Arguments:

`args`: String array of arguments from the schedule file

Example Call:

result = `SDISpectrum->auto_start( args)`

(pro) CLEANUP

Method Documentation:
Cleanup - stop any active scans, close the netcdf file, free pointers.
Arguments:

`log`: No Doc

Example Call:

`SDISpectrum->cleanup, log`

(pro) FINALIZE_SCAN

Method Documentation:
Called when a user clicks on the "Finalize" button, to indicate that an exposure should be finished after the next scan, regardless of signal-to-noise, etc.
Arguments:

`event`: Widget event

Example Call:

`SDISpectrum->finalize_scan, event`

(pro) FIT_SPECTRA

Method Documentation:
Fit spectra and create skymaps of peak position and temperature, and display them. This function was introduced to diagnose Mawson camera problems, and has stuck around since it may be generally useful.
Arguments:

`event`: Widget event

Example Call:

```
SDISpectrum->fit_spectra,  event
```

(pro) FRAME_EVENT

Method Documentation:
Frame event where the spectral information from the latest camera image is extracted. The primary purpose of this function is to call "uUpdateSpectra" in the SDI_External dll, which updates the current spectral information based on the latest camera frame. This function also checks to see if exposures are finished, sends real-time data snapshots to the console for ftp-ing, accumulates the background 'allsky' image, and updates the display of spectra and signal/noise history.
Arguments:

`image`: Latest camera image

`channel`: Current scan channel

Example Call:

```
SDISpectrum->frame_event,  image,
                           channel
```

(function) GET_SETTINGS

Method Documentation:
Get settings to save.
Takes no arguments
Example Call:

```
result = SDISpectrum->get_settings(  )
```

(pro) INITIALIZER

Method Documentation:
Initialize plugin variables, prepare the phase map, create a zone map.
Takes no arguments
Example Call:

```
SDISpectrum->initializer
```

(pro) SET_PHASEMAP

Method Documentation:
Set-up the phasemap, that is, retrieve phase map parameters from the console, interpolate to the spectrum wavelength, and wrap the phase map.
Arguments:

`failed`: OUT: flag to indicate failure, not currently used (returns 0)

Example Call:

```
SDISpectrum->set_phasemap, failed
```

(pro) START_SCAN

Method Documentation:
Start scanning.
Arguments:

`event`: Widget event

Example Call:

```
SDISpectrum->start_scan, event
```

(pro) STOP_SCAN

Method Documentation:
Stop a currently active scan.
Arguments:

`event`: Widget event

Example Call:

```
SDISpectrum->stop_scan, event
```

SDIStepsPerOrder

The StepsPerOrder plugin is used to calculate the size of the 'voltage' increment that needs to be applied to each etalon leg at each channel in a scan such that a full scan corresponds to a unit change in interference order.

Inherits from: **XDIBASE**
Class Data:

(long)	id	(ptr)	corr	(int)	num_chords
(int)	curr_chord	(int)	scanning	(int)	nchann
(int)	start_volt_offset	(int)	stop_volt_offset	(float)	volt_step_size
(obj)	scan_obj	(int)	curr_chann	(int)	last_chann
(ptr)	image	(ptr)	ref_image	(int)	xdim
(int)	ydim	(int)	counter	(int)	last_counter
(ptr)	chord_hist	(float)	wavelength	(int)	record_value
(string)	record_file	(float)	gain	(float)	exptime
(string)	obj_num	(structure)	geometry	(int)	need_frame
(int)	need_timer	(int)	auto	(structure)	palette

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdistepsperorder__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Initialize the StepsPerOrder plugin.
Arguments:

`restore_struc=restore_struc`: Restored settings

`data=data`: Misc data from the console

Example Call:

```
result = SDIStepsPerOrder->init(  restore_struc=restore_struc,
                                  data=data)
```

(function) AUTO_START

Method Documentation:
Auto-start called when running in auto-mode.
Arguments:

`args`: String array of arguments from the schedule file

Example Call:

```
result = SDIStepsPerOrder->auto_start(  args)
```

(pro) CLEANUP

Method Documentation:
Cleanup - free pointers, stop any active scan.
Arguments:
`log`: No Doc

Example Call:

```
SDIStepsPerOrder->cleanup,  log
```

(pro) FRAME_EVENT

Method Documentation:
Process the latest camera frame: bascially calculate the correlation between the current camera image and a reference image, store this value in a vector. If finished scanning, fit the vector of correlation values to find the peak, and calculate the steps/order value based on the position of that peak and the number of channels in a scan.
Arguments:
`image`: Latest camera image
`channel`: Current scan channel

Example Call:

```
SDIStepsPerOrder->frame_event,  image,
                                channel
```

(function) GET_SETTINGS

Method Documentation:
Get settings to save.
Takes no arguments
Example Call:

```
result = SDIStepsPerOrder->get_settings(  )
```

(pro) START_SCAN

Method Documentation:
Start scanning, set-up variables.
Arguments:
`event`: Widget event

Example Call:

```
SDIStepsPerOrder->start_scan,  event
```

(pro) STOP_SCAN

Method Documentation:
Stop the current scan, no steps/order value will be saved.
Arguments:

`event`: Widget event

Example Call:

```
SDIStepsPerOrder->stop_scan,  event
```

(pro) TOGGLE_RECORD

Method Documentation:
Toggle on/off the option to record steps/order values to a dedicated log file. This option is located under the file menu of the plugin, and will be rememberd for this plugin.
Arguments:

`event`: Widget event

Example Call:

```
SDIStepsPerOrder->Toggle_Record,  event
```

SDIVidshow

The Vidshow plugin displays the latest camera images as they are recorded.

Inherits from: **XDIBASE**
Class Data:

(long)	id	(int)	inst	(float)	exp_time
(int)	xdim	(int)	ydim	(int)	scale
(float)	scale_fac	(int)	crosshairs	(int)	crosshairs_point
(int)	grid	(int)	color_table	(long)	framecount
(double)	tstrt	(int)	mask_quadrants	(string)	obj_num
(structure)	geometry	(int)	need_frame	(int)	need_timer
(int)	auto	(structure)	palette	(obj)	manager

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/sdividshow__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Initialize the Vidshow plugin.
Arguments:

`restore_struc=restore_struc`: Restored settings

`data=data`: Misc data from the console

Example Call:

```
result = SDIVidshow->init( restore_struc=restore_struc,
                           data=data)
```

(pro) CLEANUP

Method Documentation:
Cleanup - nothing to do.
Arguments:

`log`: No Doc

Example Call:

```
SDIVidshow->cleanup, log
```

(pro) FIT_WINDOW

Method Documentation:
Resize the window to fit the native resolution of the camera image, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->fit_window,  event
```

(pro) FRAME_EVENT

Method Documentation:
Receive a new camera frame, scale it and display.
Arguments:

`image`: Latest camera image

`channel`: Current scan channel

Example Call:

```
SDIVidshow->frame_event,  image,
                          channel
```

(function) GET_SETTINGS

Method Documentation:
Get settings to save.
Takes no arguments
Example Call:

```
result = SDIVidshow->get_settings(  )
```

(pro) MASK_QUADRANTS

Method Documentation:
Mask out most of the four quadrants of the image, leaving only a small 'cross' of the image left to display, helps for slow connections, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->mask_quadrants, event
```

(pro) SCALING

Method Documentation:
Toggle between using the manual scale factor and auto scaling, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->scaling, event
```

(pro) SET_COLOR_TABLE

Method Documentation:
Set the color table, called when user selects this option from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->set_color_table, event
```

(pro) SET_CROSSHAIRS

Method Documentation:
Toggle on/off diaplying the crosshairs, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->set_crosshairs, event
```

(pro) SET_CROSSHAIRS_POINT

Method Documentation:
Set where the crosshairs intersect (x, y), called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->set_crosshairs_point, event
```

(pro) SET_GRID

Method Documentation:
Toggle on/off displaying a grid overlay, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->set_grid, event
```

(pro) SET_SCALE

Method Documentation:
Set a manual scale value applied to image prior to display, called from the menu.
Arguments:

`event`: Widget event

Example Call:

```
SDIVidshow->set_scale, event
```

XDIBase

This class defined basic properties all plugins inherit, like geometry, references to the console and widget manager, flags like need_timer and need_frame, etc. For a plugin to work, it must inherit from XDIBase.

Inherits from: **None**
Class Data:

(string)	obj_num	(structure)	geometry	(int)	need_frame
(int)	need_timer	(int)	auto	(structure)	palette

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/xdibase__define.pro

XDIConsole

XDIConsole is the main routine for SDI control. See the software manual for details.

Inherits from: **XDIBASE**
Class Data:

(structure)	etalon	(structure)	camera	(structure)	header
(structure)	logging	(structure)	misc	(structure)	runtime
(structure)	buffer	(string)	obj_num	(structure)	geometry
(int)	need_frame	(int)	need_timer	(int)	auto
(structure)	palette	(obj)	manager	(obj)	console

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/xdiconsole__define.pro

---

METHODS:

(function) INIT

Method Documentation:
The console initialization routine. See the SDI software manual for a description of what this function does.
Arguments:

`schedule=schedule`: The schedule file name

`mode=mode`: Mode to run in - "auto" or "manual" (default)

`settings=settings`: The console settings file (required)

`start_line=start_line`: Optional start line in the schedule file

Example Call:

```
result = XDIConsole->init(  schedule=schedule,
                            mode=mode,
                            settings=settings,
                            start_line=start_line)
```

(pro) CAM_COOLER

Method Documentation:
Called when the user clicks on the Cooler menu option. Opens up a widget for controlling camera temperature set point.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->cam_cooler,  event
```

(pro) CAM_COOLER_EVENT

Method Documentation:
Event handler for the camera cooler widget.
Arguments:
`event`: Widget event

Example Call:

```
XDIConsole->cam_cooler_event, event
```

(pro) CAM_EXPTIME

Method Documentation:
Set the camera exposure time.
Arguments:
`event`: Widget event
`new_time=new_time`: Use this to supply the new time, instead of asking for it

Example Call:

```
XDIConsole->cam_exptime, event,
                         new_time=new_time
```

(pro) CAM_GAIN

Method Documentation:
Set the camera EM gain.
Arguments:
`event`: Widget event
`new_gain=new_gain`: Use this to supply the new gain, instead of asking for it

Example Call:

```
XDIConsole->cam_gain, event,
                      new_gain=new_gain
```

(pro) CAM_INITIALIZE

Method Documentation:
Initialize the camera.
Arguments:
`event`: Widget event

Example Call:

```
XDIConsole->cam_initialize, event
```

(pro) CAM_SHUTDOWN

Method Documentation:
Shutdown the camera. If cooler is running, will flag that we need to wait for the cam temp to reach a safe level before doing a final shutdown.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->cam_shutdown, event
```

(pro) CAM_SHUTTERCLOSE

Method Documentation:
Close the camera shutter.
Arguments:

`event`: Widget event

`shutdown=shutdown`: Flag to indicate we are shutting down the camera

Example Call:

```
XDIConsole->cam_shutterclose, event,
                              shutdown=shutdown
```

(pro) CAM_SHUTTEROPEN

Method Documentation:
Open the camera shutter
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->cam_shutteropen, event
```

(pro) CAM_STATUS

Method Documentation:
Retrieve the current camera status.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->cam_status, event
```

(pro) CAM_TEMP

Method Documentation:
Called to retrieve the current camera temperature. In normal camera running mode (run till abort) this will not retrieve the temperature unless the calling widget sets a field called `force` equal to 1, which forces an abort acquisition.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->cam_temp, event
```

(pro) CLEANUP

Method Documentation:
Cleanup after the console. Call instrument-specific cleanup routine.
Takes no arguments
Example Call:

```
XDIConsole->cleanup
```

(pro) CLOSE_MPORT

Method Documentation:
Close the mirror port.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->close_mport, event
```

(pro) EDIT_PORTS

Method Documentation:
Edit the structure that defines what the com ports for each device are.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->edit_ports, event
```

(pro) EDIT_SETTINGS

Method Documentation:
Launch the console settings editor `edit_console_settings`.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->edit_settings, event
```

(pro) EDITOR_CLOSED

Method Documentation:
Called when the editor, launched from the console, is closed. This applies the new settings.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->editor_closed, event
```

(pro) END_AUTO_OBJECT

Method Documentation:
Plugins which are running in auto-mode can call this method to indicate that they have finished their current task, and another plugin can be made active. Plugins that don't stick around (like the phasemapper) can also indicate that they should be destroyed. stepsperorder plugins.
Arguments:

`id`: Widget id

`ref`: Ovject reference

`kill=kill`: Destroy the plugin

Example Call:

```
XDIConsole->end_auto_object,  id,
                              ref,
                              kill=kill
```

(pro) EVENT_HANDLER

Method Documentation:
Widget events get re-routed from the sdi_main.pro to here. If the event is a tiemr event, the timer_event method in those plugins which are registered to receive timer events (which includes the console itself) is called. For other events (for example a user clicks a button in a plugin) they are sent to their appropriate plugin.

Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->Event_Handler,  event
```

(pro) EXECUTE_SCHEDULE

Method Documentation:
The implementation of schedule file commands are placed in this function. If new schedule commands are added, their actions should be placed in this method.
Takes no arguments
Example Call:

```
XDIConsole->execute_schedule
```

(pro) FILE_CHANGE_SCHED

Method Documentation:
Open up a dialog to select a new schedule file. Sets the current schedule_line to 0.

Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->file_change_sched, event
```

(pro) FILE_RE_INITIALIZE

Method Documentation:
Call the instrument-specific initialise routine.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->file_re_initialize, event
```

(pro) FILE_SHOW

Method Documentation:
Print out a list (to the console log) of active plugins.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->file_show, event
```

(pro) FILE_SHOW_SCHED

Method Documentation:
Open up notepad to show the current schedule file if one is set.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->file_show_sched, event
```

(function) FORCE_IMAGE_UPDATE

Method Documentation:
Force the camera grab a new image (sometimes used when acquiring reference images).
Takes no arguments
Example Call:

---

result = XDIConsole->force_image_update( )

(pro) GET_CAMERA_TEMP

Method Documentation:
Fills up some variables with the current values of cam_temp, temp_state, cooler_temp.
Arguments:

temp: OUT: camera temp currently stored in settings

temp_state: OUT: camera temp state currently stored in settings

set_point: OUT: camera cooler temp set point currently stored in settings

Example Call:

```
XDIConsole->get_camera_temp,  temp,
                              temp_state,
                              set_point
```

(function) GET_DEFAULT_PATH

Method Documentation:
Return the default settings path (for plugin settings files).
Takes no arguments
Example Call:

result = XDIConsole->get_default_path( )

(function) GET_DLL_NAME

Method Documentation:
Get the name of the SDI_External dll.
Takes no arguments
Example Call:

result = XDIConsole->get_dll_name( )

(function) GET_ETALON_INFO

Method Documentation:
Return the etalon structure.
Takes no arguments
Example Call:

---

result = XDIConsole->get_etalon_info( )

(function) GET_HEADER_INFO

Method Documentation:
Return the header structure.
Takes no arguments
Example Call:

result = XDIConsole->get_header_info( )

(function) GET_IMAGE

Method Documentation:
Return the processed camera image currently stored in the console buffer.
Arguments:

image: No idea why this argument is here

Example Call:

result = XDIConsole->get_image(image)

(function) GET_LOGGING_INFO

Method Documentation:
Return the logging structure.
Takes no arguments
Example Call:

result = XDIConsole->get_logging_info( )

(function) GET_PALETTE

Method Documentation:
Return the palette.
Takes no arguments
Example Call:

result = XDIConsole->get_palette( )

(function) GET_PHASE_MAP_PATH

Method Documentation:
Return the current phasemap path (where a copy of each phasemap is saved to).
Takes no arguments
Example Call:

result = `XDIConsole->get_phase_map_path( )`

(pro) GET_PHASEMAP

Method Documentation:
Get the phase map info.
Arguments:

`phasemap_base`: OUT: phasemap base

`phasemap_grad`: OUT: phasemap gradient

`phasemap_lambda`: OUT: wavelength of phasemap base

Example Call:

```
XDIConsole->get_phasemap, phasemap_base,
                          phasemap_grad,
                          phasemap_lambda
```

(function) GET_PORT_MAP

Method Documentation:
Return the port map structure.
Takes no arguments
Example Call:

result = `XDIConsole->get_port_map( )`

(function) GET_RAW_IMAGE

Method Documentation:
Return the raw camera image currently stored in the console buffer.
Arguments:

`image`: No idea why this argument is here

Example Call:

---

result = XDIConsole->get_raw_image(image)

(function) GET_SNR_PER_SCAN

Method Documentation:
Get the current value of snr per scan.
Takes no arguments
Example Call:

result = XDIConsole->get_snr_per_scan( )

(pro) GET_SOURCE_MAP

Method Documentation:
Get the structure which defines the mapping between source position and wavelength.
Arguments:

smap: OUT: current source map

Example Call:

XDIConsole->get_source_map, smap

(function) GET_SPEC_SAVE_INFO

Method Documentation:
Spectrum plugins call this when creating new netcdf files.
Arguments:

nrings: Number of rings in the zonemap

Example Call:

result = XDIConsole->get_spec_save_info(nrings)

(function) GET_SPECTRA_PATH

Method Documentation:
Return the path where spectrum data stored.
Takes no arguments
Example Call:

result = XDIConsole->get_spectra_path( )

(function) GET_TIME_NAME_FORMAT

Method Documentation:
Get the format string used to create netcdf file names in the spectral plugins.
Takes no arguments
Example Call:

result = `XDIConsole->get_time_name_format( )`

(function) GET_ZONE_SET_PATH

Method Documentation:
Return the path where zone map settings files are stored.
Takes no arguments
Example Call:

result = `XDIConsole->get_zone_set_path( )`

(pro) IMAGE_CAPTURE

Method Documentation:
Plugins can use this method to take captures of their draw widgets and have them saved to the `screen_capture_path` field of the `misc` structure in the console settings. Images can be saved as jpeg or png. The widget using this function needs to define a uval structure with the following with the following fields: tag:"image_capture", type:"jpg" or "png", id:[array of tv ids], name:[array of string names, same size as id array]. Events from widgets with a uval.tag of "image_capture" will always be routed to here, instead of to the plugin as would usually occur.
Arguments:

`event`: Widget event

Example Call:

`XDIConsole->image_capture, event`

(pro) KILL_HANDLER

Method Documentation:
Widget destruction events are re-routed from the sdi_main.pro handler to here. This function checks to see if we are destroying the whole hierarchy (if the user closed the console) or just a single plugin. Before destroying a plugin, this function checks to see if that plugin requires any of its settings to be saved, and if so, gets the widget manager object to save those settings. If the whole hierarchy is being destroyed, this function attempts to shut down the camera. If cooling is on, a flag is set which tells the console to wait

for the temperature to go above a safe temperature (0 degrees C i think) before actually terminating. This check is done inside the timer_event method.

Arguments:

`id`: Widget id

`kill_widget=kill_widget`: Flag to indicate widget is to be destroyed

Example Call:

```
XDIConsole->Kill_Handler,  id,
                           kill_widget=kill_widget
```

(pro) LOAD_SETTINGS

Method Documentation:

Load console settings from a settings file.

Arguments:

`event`: Widget event

`filename=filename`: Filename to load from

`error=error`: OUT: error code

`first_call=first_call`: Set if this is the first time settings are being loaded (i.e. in init)

Example Call:

```
XDIConsole->load_settings,  event,
                            filename=filename,
                            error=error,
                            first_call=first_call
```

(pro) LOG

Method Documentation:

Called by widgets when they want to log events. These get logged to a log file, and optionally output to the display.

Arguments:

`entry`: String containing the log message

`sender`: String identifying the sender of the message

`display_entry=display_entry`: Set this if the message is to be displayed to the console log window

Example Call:

```
XDIConsole->log,  entry,
                  sender,
                  display_entry=display_entry
```

(pro) MODE_SWITCH

Method Documentation:
This is called when the user toggles between auto and manual mode from the console menu.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mode_switch, event
```

(pro) MOT_DRIVE_CAL

Method Documentation:
Home the mirror motor to the calibration viewing position. Calls instrument-specific file.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mot_drive_cal, event
```

(pro) MOT_DRIVE_SKY

Method Documentation:
Drive the mirror motor to the sky viewing position. Calls instrument-specific file.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mot_drive_sky, event
```

(pro) MOT_HOME_CAL

Method Documentation:
Home the mirror motor to the calibration viewing position. Calls instrument-specific file.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mot_home_cal, event
```

(pro) MOT_HOME_SKY

Method Documentation:
Home the mirror motor to the sky viewing position. Calls instrument-specific file.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mot_home_sky,  event
```

(pro) MOT_SEL_CAL

Method Documentation:
Select a new calibration source, or home it. Calls instrument-specific file.
Arguments:

`event`: Widget event

`set_source=set_source`: Supply the new source number instead of asking for it

Example Call:

```
XDIConsole->mot_sel_cal,  event,
                          set_source=set_source
```

(pro) MOT_SEL_FILTER

Method Documentation:
Select a new filter. Calls instrument-specific file.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->mot_sel_filter,  event
```

(pro) OPEN_MPORT

Method Documentation:
Open the mirror port.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->open_mport, event
```

(pro) REFRESH_SPEC_PMAPS

Method Documentation:
I don't think this has ever been used, but it is meant to force all active spectral plugins to refresh their phasemaps, if for example a phase map refresh was called during observations.
Takes no arguments
Example Call:

```
XDIConsole->refresh_spec_pmaps
```

(pro) SAVE_CURRENT_SETTINGS

Method Documentation:
Save current settings file.
Arguments:

`filename=filename`: Filename to save to

Example Call:

```
XDIConsole->save_current_settings, filename=filename
```

(pro) SCAN_ETALON

Method Documentation:
Interface for starting, stopping and pausing etalon scans.
Arguments:

`caller`: String identifying who is calling this function
`start_scan=start_scan`: Flag to start a new scan
`stop_scan=stop_scan`: Flag to stop a scan
`pause_scan=pause_scan`: Flag to pause a scan
`cont_scan=cont_scan`: Flag to continue a paused scan
`start_volt_offset=start_volt_offset`: For manual scans, the start offset
`stop_volt_offset=stop_volt_offset`: For manual scans, the stop offset
`volt_step_size=volt_step_size`: For manual scans, the volt step size
`status=status`: OUT: result of the call
`reference=reference`: OUT: a reference image at zero offset
`get_ref=get_ref`: Flag to indicate that we want a reference image (need to also supply reference keyword)
`wavelength=wavelength`: Wavelength to scan at
`force_start=force_start`: Force a scan to start even if already scanning

Example Call:

```
XDIConsole->scan_etalon,  caller,
                          start_scan=start_scan,
                          stop_scan=stop_scan,
                          pause_scan=pause_scan,
                          cont_scan=cont_scan,
                          start_volt_offset=start_volt_offset,
                          stop_volt_offset=stop_volt_offset,
                          volt_step_size=volt_step_size,
                          status=status,
                          reference=reference,
                          get_ref=get_ref,
                          wavelength=wavelength,
                          force_start=force_start
```

(pro) SEE_CALIBRATION

Method Documentation:
Show the phase map.
Arguments:

`event`: Widget event

Example Call:

```
XDIConsole->see_calibration,  event
```

(pro) SET_CENTER

Method Documentation:
Set the camera image center pixels.
Arguments:

`xcen`: X center

`ycen`: Y center

Example Call:

```
XDIConsole->set_center,  xcen,
                         ycen
```

(pro) SET_NM_PER_STEP

Method Documentation:
Set a new value for the steps per order.
Arguments:

`nm_per_step`: New nm per step value

```
XDIConsole->set_nm_per_step,  nm_per_step
```

Example Call:

(pro) SET_PHASEMAP

Method Documentation:
Set new phasemap information (multiple info is required for interpolating).
Arguments:

`phasemap_base`: Phasemap recorded at the lower wavelength

`phasemap_grad`: 'Gradient' used when interpolating

`phasemap_lambda`: Wavelength at which the base phasemap was recorded (the smaller of the two lambdas)

Example Call:

```
XDIConsole->set_phasemap,  phasemap_base,
                           phasemap_grad,
                           phasemap_lambda
```

(pro) SET_SNR_PER_SCAN

Method Documentation:
Set a new value for snr/scan,
Arguments:

`snr`: New snr value

Example Call:

```
XDIConsole->set_snr_per_scan,  snr
```

(pro) SET_SOURCE_MAP

Method Documentation:
Set the structure which defines the mapping between source position and wavelength.
Arguments:

`smap`: New source map

Example Call:

```
XDIConsole->set_source_map,  smap
```

(pro) SHUTDOWN_SPEX

Method Documentation:
This is called by Spectrum plugins if they detect that something has gone wrong with the laser. It shuts down all Spectrum plugins. The calling plugin then restarts the SDI software.
Takes no arguments
Example Call:

```
XDIConsole->shutdown_spex
```

(pro) SPECTRUM_SNAPSHOT

Method Documentation:
FTP a data snapshot provided by a spectrum plugin back to an SFTP server using PSFTP. The server and login info is store in `logging.ftp_snapshot`, for example: "137.111.22.333 -l username -pw password here".
Arguments:

`snapshot`: The data snapshot

Example Call:

```
XDIConsole->spectrum_snapshot,  snapshot
```

(pro) START_PLUGIN

Method Documentation:
When a user clicks on a plugin in the menu or a schedule command requires a plugin to be created this method is called. It is responsible for creating the plugin/object, registering it with the widget manager
Arguments:

`event`: No Doc

`args=args`: No Doc

`new_obj=new_obj`: No Doc

Example Call:

```
XDIConsole->start_plugin,  event,
                           args=args,
                           new_obj=new_obj
```

(pro) TIMER_EVENT

Method Documentation:
Timer events are processed here, this involves checking the camera for new images, updating solar elevation angle etc, incrementing the scan channel if a new image arrived, passing new images onto to registered plugins, and checking to see if a new schedule command is required.
Takes no arguments
Example Call:

```
XDIConsole->timer_event
```

(pro) UPDATE_CAMERA

Method Documentation:
Update the camera with the current set of values stored in the `camera` structure of the console settings. If you want to add new camera commands, do so here, and make sure to include the new command anywhere that this function is called.
Arguments:

`commands`: A string array of commands

`results`: OUT: a string array of results from the commands

Example Call:

```
XDIConsole->update_camera, commands,
                           results
```

(pro) UPDATE_LEGS

Method Documentation:
Update the etalon legs (plate separation).
Arguments:

`leg1=leg1`: Optional leg 1 value

`leg2=leg2`: Optional leg 2 value

`leg3=leg3`: Optional leg 3 value

`legs=legs`: Update all legs using their current values

Example Call:

```
XDIConsole->update_legs, leg1=leg1,
                         leg2=leg2,
                         leg3=leg3,
                         legs=legs
```

XDILog

The Log class manages writing log output, both to the console log window and to a text file.

Inherits from: **None**
Class Data:

(string)	log	(long)	log_window	(string)	prog_name
(string)	log_path	(int)	show_log	(string)	curdate

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/XDILog__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Initialize the log.
Arguments:

`log_window=log_window`: Widget window for the log, optional

`show_log=show_log`: Show the log

`prog_name=prog_name`: The name of the plugin, used for naming log files

`log_path=log_path`: Path to store the log files

`log_append=log_append`: Append to existing logs

`enabled=enabled`: Is logging enabled?

`header=header`: A header for the log file, used when creating a new log

Example Call:

```
result = XDILog->init(  log_window=log_window,
                        show_log=show_log,
                        prog_name=prog_name,
                        log_path=log_path,
                        log_append=log_append,
                        enabled=enabled,
                        header=header)
```

(pro) REFRESH

Method Documentation:
Refresh the log window with the current log contents.
Takes no arguments
Example Call:

```
XDILog->refresh
```

(pro) UPDATE

Method Documentation:
Add an entry to the log, prepending a date/time string.
Arguments:

`entry`: String entry to add to the log

Example Call:

```
XDILog->update,  entry
```

XDIWidgetReg

This class manages plugins, by storing their object references and names in a linked list and managing that list.

Inherits from: **None**
Class Data:

(long)	id	(string)	type	(obj)	ref
(int)	store	(int)	need_timer	(int)	need_frame

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/xdiwidgetreg__define.pro

---

METHODS:

(function) INIT

Method Documentation:
Initialize the plugin list with the id and object reference of the console.
Arguments:

`ref=ref`: Console object reference

`id=id`: Console widget id

Example Call:

```
result = XDIWidgetReg->init(  ref=ref,
                              id=id)
```

(function) COUNT_OBJECTS

Method Documentation:
Count the number of plugins, and return the count.
Takes no arguments
Example Call:

```
result = XDIWidgetReg->count_objects(  )
```

(pro) DELETE_INSTANCE

Method Documentation:
Remove a plugin from the list.
Arguments:

`id`: Widget id of the plugin to remove

```
XDIWidgetReg->delete_instance,  id
```

Example Call:

(function) GENERATE_LIST

Method Documentation:
Generate a structure whose fields are arrays, one element for each plugin, containing the plugin info.
Takes no arguments
Example Call:

```
result = XDIWidgetReg->generate_list(  )
```

(function) MATCH_REGISTER_FRAME

Method Documentation:
From a widget id, return the need_frame field of the plugin.
Arguments:

`id`: Widget if of the plugin's main window

Example Call:

```
result = XDIWidgetReg->match_register_frame(  id)
```

(function) MATCH_REGISTER_FROM_TYPE

Method Documentation:
From a plugin type, return the object reference of the plugin.
Arguments:

`type`: String type of the plugin

Example Call:

```
result = XDIWidgetReg->match_register_from_type(  type)
```

---

result = XDIWidgetReg->match_register_ref(id)

(function) MATCH_REGISTER_REF

Method Documentation:
From a widget id, return the object reference of the plugin.
Arguments:

id: Widget if of the plugin's main window

Example Call:

(function) MATCH_REGISTER_STORE

Method Documentation:
Given a widget id, return the value of the store field for the corresponding plugin.
Arguments:

id: Widget id of the plugin's main window

Example Call:

result = XDIWidgetReg->match_register_store(id)

(function) MATCH_REGISTER_TIMER

Method Documentation:
From a widget id, return the need_frame field of the plugin.
Arguments:

id: Widget if of the plugin's main window

Example Call:

result = XDIWidgetReg->match_register_timer(id)

(function) MATCH_REGISTER_TYPE

Method Documentation:
From a widget id, return the plugin type.
Arguments:

id: Widget id of the plugin's main window

Example Call:

result = XDIWidgetReg->match_register_type(id)

(pro) PRINT_REGISTER

Method Documentation:
Print out info about the list of plugins.
Takes no arguments
Example Call:

```
XDIWidgetReg->print_register
```

(pro) REGISTER

Method Documentation:
Add a plugin to the list.
Arguments:

`id`: Widget id of the plugin's main window

`ref`: Object reference for this instance of the plugin

`type`: Plugin type (string name)

`store`: Flag to indicate whether or not to save plugin settings

`timer`: Flag to indicate this plugin needs to recieve timer events

`frame`: Flag to indicate this plugin needs to recieve frame events

Example Call:

```
XDIWidgetReg->register,  id,
                         ref,
                         type,
                         store,
                         timer,
                         frame
```

(pro) SAVE_SETTINGS

Method Documentation:
This implements the ability of plugins to save their settings, to be restored next time they are opened.
Arguments:

`path`: The settings save path

`id`: The widget id of the plugin's main window

`owner`: String name of the plugin

`ref`: Object reference to the plugin

Example Call:

```
XDIWidgetReg->save_settings,  path,
                              id,
                              owner,
                              ref
```

(pro) SET_CONTROL

Method Documentation:
I have no idea what this does, and it does not appear to be called anywhere in the SDI code base, so it is probably a holdover from an early version.

Arguments:

`id`: No Doc

`ref`: No Doc

`control`: No Doc

Example Call:

```
XDIWidgetReg->set_control,  id,
                            ref,
                            control
```

Functions

(function) GET_ERROR

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_error.pro
Function Documentation:
Return an ANDOR error string given an error code.
Arguments:

`err_code`: Error code

(function) GET_NAMES

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_names.pro
Function Documentation:
From a full path list of plugins, return only the plugin names
Arguments:

`path_list`: Vector of plugin full path names

(function) ACE_FILTER_INTERFACE

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/ace_filter_interface.pro
Function Documentation:
Sends commands to an ACE filter wheel (used only at Poker I guess, since com ports are hard coded here.
Arguments:

`command=command`: Command to send

(function) DRIVE_MOTOR

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/drive_motor.pro
Function Documentation:
Wrapper for controlling Fualhaber motors. Open/close ports, enable/disable motor, get status, set position, drive to position, set speed/accel, drive in a direction in small increments until blocked (i.e. when homing the mirror motor) etc.
Arguments:

`port`: Com port of the motor
`dll_name`: SDI_External dll name (full path)
`direction=direction`: String direction ("forwards" or "backwards") to drive until blocked
`gohix=gohix`: Drive to nearest hall index
`goix=goix`:
`drive_to=drive_to`: Drive to absolute position
`control=control`: String control command (see function body)
`readpos=readpos`: Read the motor position (returned from the function)
`speed=speed`: Set the speed
`accel=accel`: Set the acceleration
`verbatim=verbatim`: Send a string command verbatim to the motor, appending a carriage return
`home_max_spin_time=home_max_spin_time`: Max time to spin (for every small increment) when homing
`timeout=timeout`: Timeout in seconds

(function) GET_PATHS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_paths.pro
Function Documentation:
No Doc
Takes no arguments

(function) GET_SUN_ELEVATION

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_sun_elevation.pro
Function Documentation:
Get the current sun elevation for a given latitude and longitude.
Arguments:

`lat`: Geographic latitude
`lon`: Geographic longitude

(function) PHASEMAP_UNWRAP

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/phasemap_unwrap.pro
Function Documentation:
'Unwrap' a phasemap produced by the SDIPhasemapper plugin.
Arguments:

`xcen`: Nominal x center

`ycen`: Nominal y center

`radial_chunk`: Size of the chunk over which to average the phase (value of 50 is used in phasemapper)

`channels`: Number of channels in the scan

`threshold`: Value of 80 is used by the phasemapper

`wavelength`: The wavelength at which the phasemap was recorded

`phasemap`: The actual phasemap 2D array

`show=show`: Show the unwrap as it occurs

`tv_id=tv_id`: Id of the tv window for showing the unwrap

`dims=dims`: Dimensions of the tv window for drawing

(function) ZONEMAPPER

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/zonemapper.pro
Function Documentation:
Creates a zone map, a 2D array of numbers indicating the zone number of each pixel.
Arguments:

`nx`: X dimension

`ny`: Y dimension

`cent`: 2-element vector containing x and y center pixels

`rads`: Vector containing the radius of each ring

`secs`: Vector containing the number of sectors in each ring

`nums`: This should be set to 0, it is not needed

`show=show`: Show the resulting zonemap

`outang=outang`: OUT: return the 'azimuth' of each zone

`outrad=outrad`: OUT: return the radius of each zone

Procedures

(pro) GET_EPHEMERIS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_ephemeris.pro
Procedure Documentation:
No Doc
Arguments:

`save_name=save_name`: No Doc

`safe_sea=safe_sea`: No Doc

`lat=lat`: No Doc

`lon=lon`: No Doc

`timeres=timeres`: No Doc

`start_stop_times=start_stop_times`: No Doc

`get_sea=get_sea`: No Doc

(pro) HANDLE_ERROR

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/SDI_Main.pro
Procedure Documentation:
Error handler.
Arguments:

`error`: Error recieved

(pro) HANDLE_EVENT

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/SDI_Main.pro
Procedure Documentation:
Handle widget events. These are rerouted to the console's event handler.
Arguments:

`event`: Widget event structure

(pro) KILL_ENTRY

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/SDI_Main.pro
Procedure Documentation:
Handle widget destroy events. These are rerouted to the consoles kill handler.
Arguments:

`id`: Widget id

(pro) MARKS_PALETTE

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/load_pal.pro
Procedure Documentation:
No Doc
Takes no arguments

(pro) SDI_MAIN

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/SDI_Main.pro
Procedure Documentation:
SDI entry point, called with a settings file, optional schedule and optional mode.
Arguments:

`settings=settings`: Settings file (required)

`schedule=schedule`: Schedule file (required if mode is "auto")

`mode=mode`: String mode, "auto" or "manual", defaults to "manual"

(pro) TREE_CLEANUP

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
If this editor was created by the SDI console, alert it that we have closed.
Arguments:

`id`: Widget id

(pro) TREE_EVENT

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Handle events generated by the tree widget.
Arguments:

`event`: Widget event structure

(pro) WRITE_SPECTRA_NETCDF

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/write_spectra_netcdf.pro
Procedure Documentation:
Wrapper for creating and writing to NETCDF files, used by the Spectrum plugin to save spectral data.
Arguments:

`ncdid`: File id to write to, 0 if opening a new file

`spectra`: The array of spectra (nzones X nchannels

`start_time`: The time at which the exposure started

`end_time`: The time at which the exposure finished

`nscans`: The number of scans in the exposure

`acc_im`: The accumulated allsky image for the exposure

`create=create`: Set this to create a new file

`fname=fname`: Filename of the file

`return_id=return_id`: When creating a new file, the netcdf id is returned

`header=header`: Header info, when creating a new file

`data=data`: Misc data, see function body

`reopen=reopen`: Reopen a file and append to it, for example after a shutdown

`update=update`: Open for writing, see function body

(pro) COMMS_WRAPPER

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/comms_wrapper.pro
Procedure Documentation:
No Doc
Arguments:
`port`: No Doc
`dll_name`: No Doc
`type=type`: No Doc
: No Doc

(pro) CONSOLE_CRASH_ROUTINE

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/crash_routines.pro
Procedure Documentation:
Check to see if the console 'crash' file is present. If it is, it is likely that the SDI console has stopped running, and this gets logged.
Arguments:
`log_file`: The filename to send/append log output to

(pro) CONSOLE_MAKE_CRASH_FILE

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/crash_routines.pro
Procedure Documentation:
Create the console 'crash' file.
Arguments:
`crash_file`: Filename for the crash file

(pro) CRASH_ROUTINES

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/crash_routines.pro
Procedure Documentation:
This gets called by a Windows scheduled script, and checks to see if a crash file is present (the console should delete this file, so if it is present, the console has likely crashed), and if so it logs a crash. If not ,it recreates the file.
Takes no arguments

(pro) DEFINE_VARIABLES

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Create the SDI variables/structures.
Arguments:

`var_holder`: Variables will be returned in this structure

(pro) DRIVE_MOTOR_WAIT_FOR_POSITION

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/drive_motor.pro
Procedure Documentation:
Wait for a position reached notification from the motor (a 'p' character). A timeout can be provided to prevent waiting forever.
Arguments:

`port`: Com port for the motor

`dll_name`: Name of the SDI_External dll

`com`: String 'com' type, e.g. "moxa"

`max_wait_time=max_wait_time`: Max time to wait in seconds

`errcode=errcode`: Returned error code

(pro) EDIT_CONSOLE_SETTINGS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Entry point for the console settings editor. Can be called ddirectly from IDL command line, or from the SDI console.
Arguments:

`filename=filename`: Pass in a filename to load upon startup

`leader=leader`: Widget leader, when called from the console

`console=console`: The console object reference, if started from the console

---

(pro) EDIT_LOAD_SETTINGS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Load a settings file from disk.
Arguments:

`filename=filename`: Filename to load

(pro) EDIT_PORT_SETTINGS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Create an xvaredit dialog for editing the port structure.
Takes no arguments

(pro) EDIT_SAVE_SETTINGS

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/edit_console_settings.pro
Procedure Documentation:
Save the current settings.
Arguments:

`filename=filename`: Filename to save to

`nosplash=nosplash`: Optionally hide the "File saved" dialog

(pro) GET_JD0_SEC

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/get_jd0_sec.pro
Procedure Documentation:
Get the current julian date and the seconds into the day.
Arguments:

`jd0`: OUT: Julian date at midnight I think...

`sec`: OUT: Seconds into the julian day

(pro) LOAD_PAL

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/load_pal.pro
Procedure Documentation:
No Doc
Arguments:

`culz`: No Doc

`idl_table=itbl`: No Doc

`bright=brt`: No Doc

`proportion=prp`: No Doc

(pro) PAL_SUBSAMP

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/load_pal.pro
Procedure Documentation:
No Doc
Arguments:

`idxlo`: No Doc

`idxhi`: No Doc

`sred`: No Doc

`sgrn`: No Doc

`sblu`: No Doc

`brt`: No Doc

`satval`: No Doc

`sign`: No Doc

(pro) RESTART_MOXA

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/restart_moxa.pro
Procedure Documentation:
Restart the MOXA USB hub, using pstools (TODO: is this used? Paths are hard coded...)
Takes no arguments

(pro) SCHEDULE_READER

Defined in file:
C:/cal/Operations/SDI_Instruments/common/idl/core/schedule_reader.pro
Procedure Documentation:
Query an SDI schedule file for the next command.
Arguments:

`schedule_file`: Schedule file name

`schedule_line`: The current schedule line

`xcomm`: OUT: string command

`xargs`: OUT: string array of arguments

`lat`: Geographic latitude

`lon`: Geographic longitude

`console_ref`: Object reference for the console

`refresh_nm_per_step=refresh_nm_per_step`: Look for a nm per step refresh command (special syntax)

`refresh_phasemap=refresh_phasemap`: Look for a phasemap refresh command (special syntax)
