Part I

Overview and User Guide

Chapter 1

PlaceHolder for future documentation

Part II

Reference Documentation

Chapter 1

DEIMOS Keyword Documentation

1.1 Overview

This section provides detailed documentation of table design and field type and usage for all the schema (proposed and production) designed for DEIMOS.

1.2 Dictionary of DEIMOS Keywords

1.2.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

1.2.2 Active Counters

NAXIS (NOST1) `smallint I3 %d` *[axes]* (0.0 to 999.0) *The value field shall contain a non-negative integer no greater than 999, representing the number of axes in an ordinary data array. A value of zero signifies that no data follow the header in the HDU.*

NUMAMPS (KECK1CCD) r `smallint I6 %d` *[amplifiers]* (1.0 to 99.0) *Number of CCD amps in use. Set via AMPLIST.*

1.2.3 Regular Memes

deimosA (KTL) `table table` (EXTNAME = 'deimosA') *Defines the KTL service called deimosA– really the same thing as a FIORD table, and should eventually be used to generate same.* deimosA consists of elements:

ACPOWER (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *AC power to the instrument. Place(s) where you can monitor this signal (ups line fail* I): At pin 15 on cable/J9 as shown on drawing el-3113-1d – corresponds to signal chan/16 on Galil controller 0. At pin 6 on cable/JV1 as shown on drawing el-3102-1w – corresponds to signal ()/() on Galil controller 0.* Uses map KTLboolean to convert or format failed_okay:

failed_okay (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.* Valid values of failed_okay are:

- failed : System failed.
- okay : System okay.

AIRSUPLY (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Compressed air supply alarm. Place(s) where you can monitor this signal (air pressure is OK* I): At pin 15 on cable/J3 as shown on drawing el-3036-1d – corresponds to signal chan/16 on Galil controller 1.* Uses map KTLboolean to convert or format failed_okay:

failed_okay (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.* Valid values of failed_okay are:

- failed : System failed.
- okay : System okay.

BARENFAN (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Instrument (barrel) electronics cooling fans. The barrel electronics cooling monitor keyword BARENMON must be set to MANUAL to prevent the monitor from controlling the fan state. Place(s) where you can monitor this signal (fans on* (barrel) O): At pin 33 on cable/J1 as shown on drawing el-3034-1d – corresponds to signal chan/7 on Galil controller 1. At pin 16 on panel/1 as shown on drawing el-3004-1l – corresponds to signal chan/7 on Galil controller 1.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

BARENMON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Instrument (barrel) electronics enclosure temperature control monitor.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

BARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument (barrel) electronics enclosure temperature in raw A/D converter counts.*

BARENTRG (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument (barrel) electronics enclosure target temperature in raw A/D converter counts.*

BARENVAL (DEIMOS2) rw `real E13.6 map` *[degC]* (-50.0 to 50.0) *Instrument (barrel) electronics enclosure temperature in degrees celsius. Place(s) where you can monitor this signal (enclosure temperature (barrel) I): At pin 1 on conn/J5 as shown on drawing el-3020-1d – corresponds to signal analog/1 on Galil controller 1. At pin 18 on termbd/A as shown on drawing el-3026-1d – corresponds to signal analog/1 on Galil controller 1.* Uses map Linear to convert or format BARENRAW:

BARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument (barrel) electronics enclosure temperature in raw A/D converter counts.*

BARGLFLO (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Coolant (glycol) flow status for the instrument (barrel) electronics cooling system. Place(s) where you can monitor this signal (not yet supported I): At pin -1 on none/null as shown on drawing el-0000-0a – corresponds to signal analog/-1 on Galil controller 1.* Uses map KTLboolean to convert or format failed_okay:

failed_okay (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.* Valid values of failed_okay are:

- failed : System failed.
- okay : System okay.

BARGLRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Coolant (glycol) temperature in raw A/D converter counts for the instrument (barrel) electronics.*

BARGLVAL (DEIMOS2) r `real E13.6 map` *[degC]* (-50.0 to 50.0) *Coolant (glycol) temperature in degrees celsius for the instrument (barrel) electronics. Place(s) where you can monitor this signal (glycol temperature (barrel) I): At pin 2 on cable/J5 as shown on drawing el-3020-1d – corresponds to signal analog/2 on Galil controller 1. At pin 19 on termbd/A as shown on drawing el-3026-1d – corresponds to signal analog/2 on Galil controller 1.* Uses map Linear to convert or format BARGLRAW:

BARGLRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Coolant (glycol) temperature in raw A/D converter counts for the instrument (barrel) electronics.*

BARXTRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *External (dome) temperature in raw A/D converter counts for the instrument (barrel) electronics cooling system.*

BARXTVAL (DEIMOS2) r `real E13.6 map` *[degC]* (-50.0 to 50.0) *External (dome) temperature in degrees celsius for the barrel electronics cooling system. Place(s) where you can monitor this signal (dome temperature (barrel) I): At pin 3 on conn/J5 as shown on drawing el-3020-1d – corresponds to signal analog/3 on Galil controller 1. At pin 20 on termbd/A as shown on drawing el-3026-1d – corresponds to signal analog/3 on Galil controller 1.* Uses map Linear to convert or format BARXTRAW:

BARXTRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *External (dome) temperature in raw A/D converter counts for the instrument (barrel) electronics cooling system.*

CARENFAN (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Carriage electronics cooling fans. The carriage electronics cooling monitor keyword CARENMON must be set to MANUAL to prevent the monitor from controlling the fan state. Place(s) where you can monitor this signal (fans on* (barrel) O): At pin 43 on cable/J5* *as shown on drawing el-3111-1d – corresponds to signal chan/2 on Galil controller 0. At pin 2 on termbd/TBVD as shown on drawing el-3107-1l – corresponds to signal chan/7 on Galil controller 1.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

CARENMON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Carriage electronics enclosure temperature control monitor.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

CARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage electronics enclosure temperature in raw A/D converter counts.*

CARENTRG (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage electronics enclosure target temperature in raw A/D converter counts.*

CARENVAL (DEIMOS2) rw `real E13.6 map` *[degC]* (-50.0 to 50.0) *Carriage electronics enclosure temperature in degrees celsius. Place(s) where you can monitor this signal (enclosure temperature (cradle) I): At pin 1 on conn/J5 as shown on drawing el-3030-1d – corresponds to signal analog/1 on Galil controller 0. At pin 18 on termbd/() as shown on drawing el-3031-1d – corresponds to signal analog/1 on Galil controller 0.* Uses map Linear to convert or format CARENRAW:

CARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage electronics enclosure temperature in raw A/D converter counts.*

CARGLFLO (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Coolant (glycol) flow status for the carriage electronics cooling system. Place(s) where you can monitor this signal (coolflow flow is OK (cradle)*

I): At pin 17 on cable/J9 as shown on drawing el-3113-1d – corresponds to signal chan/15 on Galil controller 0. At pin 32 on panel/3 as shown on drawing el-3103-1l – corresponds to signal chan/15 on Galil controller 0. Uses map KTLboolean to convert or format failed_okay:

failed_okay (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.* Valid values of failed_okay are:

- failed : System failed.
- okay : System okay.

CARGLRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Coolant (glycol) temperature in raw A/D converter counts for the carriage electronics.*

CARGLVAL (DEIMOS2) r `real E13.6 map` *[degC]* (-50.0 to 50.0) *Coolant (glycol) temperature in degrees celsius for the carriage electronics. Place(s) where you can monitor this signal (glycol temperature (cradle) I): At pin 2 on cable/J5 as shown on drawing el-3030-1d – corresponds to signal analog/2 on Galil controller 0. At pin 19 on termbd/() as shown on drawing el-3031-1d – corresponds to signal analog/2 on Galil controller 0.* Uses map Linear to convert or format CARGLRAW:

CARGLRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Coolant (glycol) temperature in raw A/D converter counts for the carriage electronics.*

CARXTRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *External (dome) temperature in raw A/D converter counts for the carriage electronics cooling system.*

CARXTVAL (DEIMOS2) r `real E13.6 map` *[degC]* (-50.0 to 50.0) *External (dome) temperature in degrees celsius for the carriage electronics cooling system. Place(s) where you can monitor this signal (dome temperature (at cradle) I): At pin 3 on conn/J5 as shown on drawing el-3030-1d – corresponds to signal analog/3 on Galil controller 0. At pin 20 on termbd/() as shown on drawing el-3031-1d – corresponds to signal analog/3 on Galil controller 0.* Uses map Linear to convert or format CARXTRAW:

CARXTRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *External (dome) temperature in raw A/D converter counts for the carriage electronics cooling system.*

ccd (KTLinst) `bundle bundle` *Defines a keyword bundle for CCD subsystem keywords. The bundle will be included in a named KTL service for Lick/Keck instruments – really the same thing as a FIORD table, and will be used to generate same.* ccd consists of elements:

ABORTEX (KECK1CCD) w `char(4) A %d` (0.0 to 1.0) *Set true to abort exposure during integration. NOTE: currently cannot abort durring erase or readout.* Uses map KTLboolean to convert or format write_true:

write_true (SharedMemeMap) w `tinyint L1 %d` (0.0 to 1.0) *Generic map meme for KTL keywords that are write-only and can only be written True.* Valid values of write_true are:

- true :

AMPLIST (KECK1CCD) rw `varchar(68) A %s%s...` *An array describing the readout of a CCD with multiple amplifiers. It has 4 elements, one for each possible amp on the CCDs. Amps numbered 1 to 4 (1=top left,2 = top right,3= bottom left,4 = bottom right), only 1 and 2 valid for two amp devices.* Valid values of AMPLIST are:

- (1,0,0,0) : upper left amp
- (2,0,0,0) : upper right amp
- (2,1,0,0) : both top amps

AUTOERAS (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *Set true to enable erase prior to integration.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

AUTOFILL (KECK1CCD) rw `int I11 %d` *[enum]* *Not yet implimented.*

AUTOREAD (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *Set true to enable read out at end of integration.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

AUTOSHUT (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *Set to true to open shutter at start of integration.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

BINNING (KECK1CCD) rw `varchar(68) A %s%s...` *[pixel (CCD)] The amount of on-chip binning performed along each readout axis of a CCD.* Consists of a list of NAXIS of:

BinSize (KECK1CCD) rw `smallint I6 %d` *[pixel (CCD)]* (1.0 to 8.0) *The amount of on-chip binning performed along one readout axis of a CCD.*

CCDGAIN (KECK1CCD) rw `char(4) A %d` (0.0 to 1.0) *1 = high gain, 0 = low gain. High gain is 2x low gain.* Uses map KTLboolean to convert or format low_high:

low_high (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (low, high) keywords.* Valid values of low_high are:

- low :
- high :

CCDROTYP (DEIMOSdash) rw `varchar(68) A %s` *A general classification of the windowing pattern which will be used to readout a mosaic detector such as DEIMOS.*

CSHUTTER (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *Manual operation of the CCD shutter. Values are 'open' and 'closed'.* Uses map KTLboolean to convert or format close_open:

close_open (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSE/OPEN) keywords.* Valid values of close_open are:

- close : Close the door or hatch
- open : Open the door or hatch

DATALINE (KECK1CCD) r `int I11 %d` *[lines]* (0.0 to 8192.0) *Number of imaging lines (rows) in an image. In the FITS image this will match the y-size of the WINDOW keyword in* *binned pixels. Although KTL believes it readable and writable, there is no code in the crate for this.*

DATAPIX (KECK1CCD) r `int I11 %d` *[pixel]* (0.0 to 8192.0) *Number of binned data pixels (not pre/postscan) per amplifier on the CCD. This value is unaffected by WINDOW, it simply reflects CCD geometry and binning. Although KTL believes it readable and writable, there is no code in the crate for this.*

DEWARID (KECK1CCD) r `int I11 %d` *[dewar id]* (0.0 to 31.0) *Returns the value of the dewar id jumper block.*

DWINDOW (KECK1CCD) rw `varchar(68) A %s` *Defined for Keck HIRES, but never fully implemented. The desired readout window expressed in unbinned pixels. This would specify a readout window unaffected by changes in binning.* Is really a:

window0 (KECK1CCD) `tuple tuple` *This bundle describes a single readout sub-raster of one CCD. It was originally defined with the intent of use with a mosaic of CCDs, but this has never been exercised. It uses zero-indexed, unbinned pixels.* window0 consists of elements:

WINchip (KECK1CCD) `smallint I5 %d` (0.0 to 10000.0) *To which chip in the mosaic does this sub-raster apply? Note that this keyword has never been used with a mosaic and that the semantics of this element are ill-defined.*

WINxlen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the serial-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINxstart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the serial-readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.*

WINylen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the parallel-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINystart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the parallel-*

readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.

DWRMANF (KECK1CCD) r `varchar(68) A %d` (0.0 to 1.0) *Set to true to manually initiate a dewar LN2 fill.* Valid values of DWRMANF are:

- fill requested :
- no fill request :

DWRN2LV (KECK1CCD) r `real F12.8 %s` *[percent full]* (0.0 to 100.0) *Liquid nitrogen level in dewar.* Uses map DWRN2LVM to convert or format DWRN2LVR:

DWRN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Dewar liquid nitrogen level expressed as a 'raw' value from an analog-to-digital converter.*

DWRN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar liquid nitrogen level value (see DWRN2LVR) into a percentage full (see DWRN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

DWRN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Dewar liquid nitrogen level expressed as a 'raw' value from an analog-to-digital converter.*

ELAPTIME (KECK1CCD) r `int I11 %d` *[s]* (0.0 to 10000.0) *Elapsed time of the exposure.*

ERASECNT (KECK1CCD) rw `int I11 %d` *[erase cycles]* (1.0 to 100.0)
Number of erase cycles to perform prior to start of exposure.

ERASEIP (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *TRUE while CCD is erasing.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

ERASLINE (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *A convention for saving overscan rows of CCD readout adopted from practice at Caltech. If TRUE, overscan rows are averaged into a single row appended as the first row following actual image pixel data; if FALSE, the rows are simply appended.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

EXPOSE (KECK1CCD) w `char(4) A %d` (0.0 to 1.0) *Set TRUE to start an exposure.* Uses map KTLboolean to convert or format write_true:

write_true (SharedMemeMap) w `tinyint L1 %d` (0.0 to 1.0) *Generic map meme for KTL keywords that are write-only and can only be written True.* Valid values of write_true are:

- true :

EXPOSIP (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *TRUE while an exposure is in progress.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

FRAMENO (KECK1DAS) rw `int I11 %d` (0.0 to 9999.0) *Frame number of the next (or currently in-progress) exposure. This is appended after OUTFILE when constructing the name of the output disk file.*

KEEPPREP (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *Should prescan pixels be retained when the image is displayed and saved as FITS? Prescan pixels are typically artifacts of a serial shift register with several pixels preceding those transferred from the parallel array. Usually for engineering only.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

MPPMODE (KECK1CCD) rw `char(4) A %d` (0.0 to 1.0) *TRUE if CCD is being read out in MPP mode.* Uses map KTLboolean to convert or format low_high:

low_high (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (low, high) keywords.* Valid values of low_high are:

- low :
- high :

NSUBINT (KECK1CCD) rw `int I11 %d` *[subintegrations]* (0.0 to 100.0) *Number of subintegrations. NSUBINT must divide evenly into TTIME. Defines when to shift CCD pixel rows up by ROWSHFT during integration. Shift rows every TTIME/NSUBINT.*

NUMAMPS (KECK1CCD) r `smallint I6 %d` *[amplifiers]* (1.0 to 99.0) *Number of CCD amps in use. Set via AMPLIST.*

OBSTYP (IRAF) rw `varchar(68) A %s` *Semantics Unknown*

OVRFLUSH (KECK1CCD) rw `int I11 %d` *[lines]* (0.0 to 2048.0) *Number of overscan rows to flush prior to reading overscan rows.*

PAUSE (KECK1CCD) w `char(4) A %d` (0.0 to 1.0) *Set to TRUE to pause an exposure.* Uses map KTLboolean to convert or format write_true:

write_true (SharedMemeMap) w `tinyint L1 %d` (0.0 to 1.0) *Generic map meme for KTL keywords that are write-only and can only be written True.* Valid values of write_true are:

- true :

PAUSEIP (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *TRUE while CCD exposure is paused.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

POSTLINE (KECK1CCD) rw `smallint I6 %d` *[lines]* (0.0 to 2048.0) *Number of overscan rows to read following overscan flush (OVRFLUSH). These rows are binned in the same fashion as the actual image data. The manner in which they are saved to the FITS file is controlled by ERASLINE.*

POSTPIX (KECK1CCD) rw `smallint I6 %d` *[pixel]* (0.0 to 2048.0) *Number of overscan pixels per amplifier. When set and read by KTL this is unbinned pixels. When written to a FITS file this is binned pixels. It should be set to a value which is divided evenly by the serial-direction binning value.*

PREFLUSH (KECK1CCD) rw `int I11 %d` *[lines]* (0.0 to 2048.0) *Number of prescan rows to flush prior to reading prescan rows.*

PRELINE (KECK1CCD) rw `smallint I6 %d` *[lines]* (0.0 to 2048.0) *Number of prescan rows to read and save following prescan flush (PREFLUSH). These are binned in the same fashion as the actual image data.*

PREPIX (KECK1CCD) rw `smallint I6 %d` *[pixel]* (0.0 to 2048.0) *Number of prescan pixels per amplifier. When set and read by KTL this is unbinned pixels. When written to FITS file this is binned pixels. Must be 0 mod the serial binning, must only be set if one knows the CCD gate geometry, else nonsense may result.*

PWINDOW (KECK1CCD) r `varchar(68) A %s` *Defined for Keck HIRES, but never fully implemented. The closest realizable approximation of the desired CCD readout window (DWINDOW) given the constraints of gate geometry and binning. Read only. To be reported in unbinned pixels.* Is really a:

window0 (KECK1CCD) `tuple tuple` *This bundle describes a single readout sub-raster of one CCD. It was originally defined with the intent of use with a mosaic of CCDs, but this has never been exercised. It uses zero-indexed, unbinned pixels.* window0 consists of elements:

WINchip (KECK1CCD) `smallint I5 %d` (0.0 to 10000.0) *To which chip in the mosaic does this sub-raster apply? Note that this keyword has never been used with a mosaic and that the semantics of this element are ill-defined.*

WINxlen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the serial-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINxstart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the serial-readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.*

WINylen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the parallel-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINystart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the parallel-readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.*

PWRBLOK (KECK1CCD) rw `real F12.8 %s` *[V]* (0.0 to 10.0) *Power to heater block in the dewar.* Uses map PWRBLOKM to convert or format PWRBLOKR:

PWRBLOKR (KECK1CCD) rw `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Power to heater block in the dewar expressed as a 'raw' value from an analog-to-digital converter.*

PWRBLOKM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar heater power block value (see PWRBLOKR) into a voltage (see PWRBLOK).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device en-*

coder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers. isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

PWRBLOKR (KECK1CCD) rw `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Power to heater block in the dewar expressed as a 'raw' value from an analog-to-digital converter.*

RESN2LV (KECK1CCD) r `real F12.8 %s` *[percent full]* (0.0 to 100.0) *Liquid nitrgen level in the reservoir source used to refill the dewar.* Uses map RESN2LVM to convert or format RESN2LVR:

RESN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Liquid nitrogen level of the reservoir source used to refill the dewar expressed as a 'raw' value from an analog-to-digital converter.*

RESN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar refill reservoir level (see RESN2LVR) into a percentage full (see RESN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

RESN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Liquid nitrogen level of the reservoir source used to refill the dewar expressed as a 'raw' value from an analog-to-digital converter.*

RESUME (KECK1CCD) w `char(4) A %d` (0.0 to 1.0) *Set to TRUE to resume an exposure.* Uses map KTLboolean to convert or format write_true:

write_true (SharedMemeMap) w `tinyint L1 %d` (0.0 to 1.0) *Generic map meme for KTL keywords that are write-only and can only be written True.* Valid values of write_true are:

- true :

ROWSHFT (KECK1CCD) rw `int I11 %d` *[rows]* (0.0 to 2048.0) *Number of CCD pixel rows to shift 'up' every TTIME/NSUBINT.*

SHUTCLOS (KECK1CCD) rw `char(10) A %d` (0.0 to 1.0) *TRUE if shutter is closed.* Uses map KTLboolean to convert or format closed_not:

closed_not (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (closed, not closed) keywords.* Valid values of closed_not are:

- closed :
- not closed :

SHUTOPEN (KECK1CCD) rw `char(8) A %d` (0.0 to 1.0) *TRUE if shutter is open.* Uses map KTLboolean to convert or format open_not:

open_not (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (open, not open) keywords.* Valid values of open_not are:

- open :
- not open :

SHUTSTAT (KECK1CCD) rw `int I11 %d` (0.0 to 4095.0) *A mask indicating one of four possible shutter states: ??? = open, ??? = closed, ??? = neither open or closed, ??? = both open and closed (this state is an error state).*

STOPEX (KECK1CCD) w `char(4) A %d` (0.0 to 1.0) *Set to TRUE to stop an exposure.* Uses map KTLboolean to convert or format write_true:

write_true (SharedMemeMap) w `tinyint L1 %d` (0.0 to 1.0) *Generic map meme for KTL keywords that are write-only and can only be written True.* Valid values of write_true are:

- true :

TAPEDEV (KECK1DAS) rw `varchar(68) A %s` *A character string defining the Unix filesystem name of the tape device to which newly acquired images may be written. This capability has never been implemented.*

TEMPDET (KECK1CCD) r `real F13.8 %s` *[degC]* (-220.0 to 25.0) *Detector temperature.* Uses map TEMPDETM to convert or format TEMPDETR:

TEMPDETR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Detector temperature expressed as a 'raw' value from an analog-to-digital converter.*

TEMPDETM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw detector temperature (see TEMPDETR) into degrees celsius (see TEMPDET).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

TEMPDETR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Detector temperature expressed as a 'raw' value from an analog-to-digital converter.*

TEMPSET (KECK1CCD) rw `real F13.8 %s` *[degC]* (-180.0 to 25.0) *Detector temperature set point.* Uses map TEMPSETM to convert or format TEMPSETR:

TEMPSETR (KECK1CCD) rw `int I11 %d` *[ADC value]* (0.0 to 4095.0) *The detector temperature set point expressed as a 'raw' value from an analog-to-digital converter.*

TEMPSETM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw detector temperature set point (see TEMPSETR) into degrees celsius (see TEMPSET).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

TEMPSETR (KECK1CCD) rw `int I11 %d` *[ADC value]* (0.0 to 4095.0) *The detector temperature set point expressed as a 'raw' value from an analog-to-digital converter.*

TODISK (KECK1DAS) rw `char(5) A %d` (0.0 to 1.0) *Should the next acquired image be written to disk? If false then the image will, by default, be written to OUTDIR with the name "backup.fits". If true, the file name is constructed as described by OUTFILE.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

TOTAPE (KECK1DAS) rw `char(5) A %d` (0.0 to 1.0) *Should newly acquired images be written to tape? This capability has* *never been implemented.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

TTIME (KECK1CCD) rw `int I11 %d` *[s]* (0.0 to 10000.0) *Target length of time for the exposure.*

UTB15VEN (KECK1CCD) rw `char(8) A %d` (0.0 to 1.0) *Enable +/-15 volt analog power in the CCD controller.* Uses map KTLboolean to convert or format enabled_dis:

enabled_dis (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for (enabled, disabled) keywords.* Valid values of enabled_dis are:

- enabled :
- disabled :

UTB30VEN (KECK1CCD) rw `char(8) A %d` (0.0 to 1.0) *Enable +30 volt analog power in the CCD controller.* Uses map KTLboolean to convert or format enabled_dis:

enabled_dis (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for (enabled, disabled) keywords.* Valid values of enabled_dis are:

- enabled :
- disabled :

UTBFANS (KECK1CCD) rw `char(3) A %d` (0.0 to 1.0) *CCD electronics enclosure fan state. The fans are normally controlled by a continuously running monitor on the CCD crate. The monitor will override external attempts to control the fans via this keyword.* Uses map KTLboolean to convert or format on_off:

on_off (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (on, off) keywords.* Valid values of on_off are:

- on :
- off :

UTBM15V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5) *-15 volt power supply voltage level. THIS VALUE SHOULD*

BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET! Uses map UTBM15VM to convert or format UTBM15VR:

UTBM15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *-15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBM15VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' -15 volt power supply voltage value from an analog-to-digital converter (see UTBM15VR) to a voltage (see UTBM15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBM15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *-15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBN2FIL (KECK1CCD) rw `char(3) A %d` (0.0 to 1.0) *Set to TRUE to initiate a dewar liquid nitrogen fill.* Uses map KTLboolean to convert or format on_off:

on_off (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (on, off) keywords.* Valid values of on_off are:

- on :
- off :

UTBN2HTR (KECK1CCD) rw `char(3) A %d` (0.0 to 1.0) *Obsolete. Originally, it was set to TRUE to turn on a heater in the liq-*

uid nitrogen reservoir to pressurize the reservoir, if the compressed air pressurization system failed. Uses map KTLboolean to convert or format on_off:

on_off (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (on, off) keywords.* Valid values of on_off are:

- on :
- off :

UTBP15V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5) *+15 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET!* Uses map UTBP15VM to convert or format UTBP15VR:

UTBP15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *+15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBP15VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +15 volt power supply voltage value from an analog-to-digital converter (see UTBP15VR) to a voltage (see UTBP15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBP15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *+15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBP30V (KECK1CCD) r `real` `E13.6` `%s` *[ADC value / 10]* (0.0 to 409.5) *+30 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET!* Uses map UTBP30VM to convert or format UTBP30VR:

UTBP30VR (KECK1CCD) r `int` `I11` `%d` *[ADC value]* (0.0 to 4095.0) *+30 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBP30VM (KECK1CCD) rw `varchar(68)` `A` `%s` *An array of linear unit conversion constants used to convert a 'raw' +30 volt power supply voltage value from an analog-to-digital converter (see UTBP30VR) to a voltage (see UTBP30V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple` `tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real` `E13.6` `%E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real` `E13.6` `%E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real` `E13.6` `%E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real` `E13.6` `%E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real` `E13.6` `%E` *The scale parameter in a mapping from device units to user units.*

UTBP30VR (KECK1CCD) r `int` `I11` `%d` *[ADC value]* (0.0 to 4095.0) *+30 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBP5V (KECK1CCD) r `real` `E13.6` `%s` *[ADC value / 10]* (0.0 to 409.5) *+5 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET!* Uses map UTBP5VM to convert or format UTBP5VR:

UTBP5VR (KECK1CCD) r `int` `I11` `%d` *[ADC value]* (0.0 to 4095.0) *+5 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBP5VM (KECK1CCD) rw `varchar(68)` `A` `%s` *An array of linear unit conversion constants used to convert a 'raw' +5* *volt power supply voltage value from an analog-to-digital converter (see UTBP5VR) to a voltage (see UTBP5V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBP5VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *+5 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.*

UTBTEMP (KECK1CCD) r `real F12.8 %s` *[degC]* (-250.0 to 50.0) *Temperature of the CCD controller utility board.* Uses map UTBTEMPM to convert or format UTBTEMPR:

UTBTEMPR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Temperature of the CCD controller utility board expressed as a 'raw' value from an analog-to-digital converter.*

UTBTEMPM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert the raw CCD controller utility board temperature from an analog-to-digital converter value (see UTBTEMPR) to a value in degrees celsius (see UTBTEMP).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBTEMPR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0) *Temperature of the CCD controller utility board expressed as a 'raw' value from an analog-to-digital converter.*

UTBTMPS (KECK1CCD) rw `real F12.8 %s` *[degC]* (-999.0 to 50.0) *Temperature set point for the CCD electronics enclosure (as measured by UTBTEMP).* Uses map UTBTMPSM to convert or format UTBTMPSR:

UTBTMPSR (KECK1CCD) rw `int I11 %d` *[DAC value]* (0.0 to 4095.0) *Temperature of the CCD electronics enclosure expressed as a 'raw' value from an analog-to-digital converter.*

UTBTMPSM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' CCD electronics enclosure temperature set point from an analog-to-digital converter (see UTBTMPSR) to a value in degrees celsius (see UTBTMPS).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBTMPSR (KECK1CCD) rw `int I11 %d` *[DAC value]* (0.0 to 4095.0)
Temperature of the CCD electronics enclosure expressed as a 'raw' value from an analog-to-digital converter.

VOFFSET0 (KECK1CCD) rw `int I11 %d` *[video offset]* (0.0 to 255.0)
Video offset for CCD controller analog board #0.

VOFFSET1 (KECK1CCD) rw `int I11 %d` *[video offset]* (0.0 to 255.0)
Video offset for CCD controller analog board #1.

VOFFSET2 (KECK1CCD) rw `int I11 %d` *[video offset]* (0.0 to 255.0)
Video offset for CCD controller analog board #2.

VOFFSET3 (KECK1CCD) rw `int I11 %d` *[video offset]* (0.0 to 255.0)
Video offset for CCD controller analog board #3.

WCRATE (KECK1CCD) rw `char(5) A %d` (0.0 to 1.0) *TRUE when writing image to CCD controller VME crate memory, i.e., when the CCD is reading out.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

WDISK (KECK1CCD) rw `char(3) A %d` (0.0 to 1.0) *TRUE when writing CCD image to disk from CCD controller VME crate memory.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

WINDOW (KECK1CCD) rw `varchar(68) A %s` *Defines window for readout. Values are: WINMODE, STARTROW, STARTCOL, NUMROWS, NUMCOLS. These are specified and reported as binned pixels.* Is really a:

winmod0b (KECK1CCD) `tuple tuple` *This bundle describes a single readout sub-raster of one CCD. It defines a mode of readout, plus origin and extent of a window. It uses zero-indexed, binned pixels. It was adapted from the earlier window0b meme.* winmod0b consists of elements:

WINmode (KECK1CCD) `smallint I5 %d` (0.0 to 10000.0)
Selects between two alternate modes for flushing the charge

from the rows outside the desired readout window. Slow mode is recommended for complete erasure of bright images when not using MPP mode. Used by LRIS. Valid values of WINmode are:

- 0 : slow mode
- 1 : fast mode

WINxlen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the serial-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINxstart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the serial-readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.*

WINylen (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The extent of the sub-raster along the parallel-readout direction of the CCD. The pixel binning, or lack thereof, is indicated by the parent meme.*

WINystart (KECK1CCD) `smallint I5 %d` *[pixel]* (0.0 to 10000.0) *The starting pixel of the sub-raster along the parallel-readout direction of the CCD. The first image pixel is denoted 0 (i.e, it is zero-indexed). The pixel binning, or lack thereof, is indicated by the parent meme.*

CTRL0CMD (DEIMOS2) w `varchar(68) A %s` *Carriage motor controller command. A command for the controller in Galil-1500 series controller command language.*

CTRL0MOD (DEIMOS2) rw `varchar(68) A tinyint` *Carriage motor controller control mode.* Uses map KTLenumerated to convert or format ctrl_mod:

ctrl_mod (SharedMemeMap) rw `tinyint I11 %d` (-3.0 to 2.0) *Generic map meme for motor controller mode keywords.* Valid values of ctrl_mod are:

- Ready : Read-only state. Motor controller tasks are ready for manual control box commands or commands from user interface processes.
- Warm start : Warm start. Stops programs and stages, clears requests, restarts bekins and uhaul.

- Cold start : Cold start. Executes a controller reset.
- Bekins failed : Read-only state. The bekins task on the controller failed.
- Uhaul failed : Read-only state. The uhaul task on the controller failed.
- Bekins & Uhaul failed : Read-only state. The bekins and uhaul tasks on the controller failed.

CTRL0RSP (DEIMOS2) r `varchar(68) A %s` *Carriage motor controller response to a motor controller command.*

CTRL0VER (DEIMOS2) r `varchar(68) A %s` *Carriage motor controller software & configuration data version information.*

CTRL1CMD (DEIMOS2) w `varchar(68) A %s` *Instrument motor controller #1 command. A command for the controller in Galil-1500 series controller command language.*

CTRL1MOD (DEIMOS2) rw `varchar(68) A tinyint` *Instrument motor controller #1 control mode.* Uses map KTLenumerated to convert or format ctrl_mod:

ctrl_mod (SharedMemeMap) rw `tinyint I11 %d` (-3.0 to 2.0) *Generic map meme for motor controller mode keywords.* Valid values of ctrl_mod are:

- Ready : Read-only state. Motor controller tasks are ready for manual control box commands or commands from user interface processes.
- Warm start : Warm start. Stops programs and stages, clears requests, restarts bekins and uhaul.
- Cold start : Cold start. Executes a controller reset.
- Bekins failed : Read-only state. The bekins task on the controller failed.
- Uhaul failed : Read-only state. The uhaul task on the controller failed.
- Bekins & Uhaul failed : Read-only state. The bekins and uhaul tasks on the controller failed.

CTRL1RSP (DEIMOS2) r `varchar(68) A %s` *Instrument motor controller #1 response to a motor controller command.*

CTRL1VER (DEIMOS2) r `varchar(68) A %s` *Instrument motor controller #1 software & configuration data version information.*

CTRL2CMD (DEIMOS2) w `varchar(68) A %s` *Instrument motor controller #2 command. A command for the controller in Galil-1500 series controller command language.*

CTRL2MOD (DEIMOS2) rw `varchar(68) A tinyint` *Instrument motor controller #2 control mode.* Uses map KTLenumerated to convert or format ctrl_mod:

ctrl_mod (SharedMemeMap) rw `tinyint I11 %d` (-3.0 to 2.0) *Generic map meme for motor controller mode keywords.* Valid values of ctrl_mod are:

- Ready : Read-only state. Motor controller tasks are ready for manual control box commands or commands from user interface processes.
- Warm start : Warm start. Stops programs and stages, clears requests, restarts bekins and uhaul.
- Cold start : Cold start. Executes a controller reset.
- Bekins failed : Read-only state. The bekins task on the controller failed.
- Uhaul failed : Read-only state. The uhaul task on the controller failed.
- Bekins & Uhaul failed : Read-only state. The bekins and uhaul tasks on the controller failed.

CTRL2RSP (DEIMOS2) r `varchar(68) A %s` *Instrument motor controller response to a motor controller command.*

CTRL2VER (DEIMOS2) r `varchar(68) A %s` *Instrument controller #2 software & configuration data version information.*

DISP0CFG (DEIMOS2) rw `varchar(68) A %s` *Path to the carriage controller dispatcher's root configuration file.*

DISP0LOG (DEIMOS2) rw `varchar(68) A %s` *Path to the log file for the carriage motor controller's dispatcher.*

DISP0MOD (DEIMOS2) rw `char(19) A tinyint` *Carriage controller dispatcher mode.* Uses map KTLenumerated to convert or format disp_mod:

disp_mod (SharedMemeMap) rw `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for dispatcher mode keywords.* Valid values of disp_mod are:

- Exit : Exit dispatcher (shutdown dispatcher process)
- Ready : Normal operating state. Devices can be calibrated and set to target states.

- Keyword : Keyword validation mode. No I/O to any device. No internal changes to dispatcher state via keywords, except to the dispatcher mode. Return SUCCESS status message, but don't simulate I/O requests.
- Simulate : Simulate I/O. No device I/O occurs. A success message is returned after device I/O request is simulated.
- Diagnostic : Run diagnostics routines selected via the DISPxTST keyword for the dispatcher.
- Reconfig : Reload the dispatcher configuration data from the configuration file.
- Configuration error : Read only state. Error in the configuration data in the configuration files.

DISP0MSG (DEIMOS2) rw `varchar(68) A %s` *Latest message regarding carriage dispatcher.*

DISP0TST (DEIMOS2) rw `int I11 %d` (0.0 to 10.0) *Carriage dispatcher test mode.*

DISP0VER (DEIMOS2) r `varchar(68) A %s` *Carriage dispatcher software version information.*

DISP0VRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Carriage controller dispatcher verbose log messages.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

DISP1CFG (DEIMOS2) rw `varchar(68) A %s` *Path to the instrument dispatcher' #1's root configuration file.*

DISP1LOG (DEIMOS2) rw `varchar(68) A %s` *Path to the log file for instrument motor controller #1's dispatcher.*

DISP1MOD (DEIMOS2) rw `char(19) A tinyint` *Instrument controller #1's controller dispatcher mode.* Uses map KTLenumerated to convert or format disp_mod:

disp_mod (SharedMemeMap) rw `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for dispatcher mode keywords.* Valid values of disp_mod are:

- Exit : Exit dispatcher (shutdown dispatcher process)

- Ready : Normal operating state. Devices can be calibrated and set to target states.
- Keyword : Keyword validation mode. No I/O to any device. No internal changes to dispatcher state via keywords, except to the dispatcher mode. Return SUCCESS status message, but don't simulate I/O requests.
- Simulate : Simulate I/O. No device I/O occurs. A success message is returned after device I/O request is simulated.
- Diagnostic : Run diagnostics routines selected via the DISPxTST keyword for the dispatcher.
- Reconfig : Reload the dispatcher configuration data from the configuration file.
- Configuration error : Read only state. Error in the configuration data in the configuration files.

DISP1MSG (DEIMOS2) rw `varchar(68) A %s` *Latest message regarding instrument controller #1's dispatcher.*

DISP1TST (DEIMOS2) rw `int I11 %d` (0.0 to 10.0) *Instrument controller #1's dispatcher test mode.*

DISP1VER (DEIMOS2) r `varchar(68) A %s` *Instrument controller #1's dispatcher software version information.*

DISP1VRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Instrument controller #1's dispatcher verbose log messages.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

DISP2CFG (DEIMOS2) rw `varchar(68) A %s` *Path to the root configuration file for instrument controller #2's dispatcher.*

DISP2LOG (DEIMOS2) rw `varchar(68) A %s` *Path to the log file for instrument controller #2's dispatcher.*

DISP2MOD (DEIMOS2) rw `char(19) A tinyint` (0.0 to 7.0) *Instrument controller #2's controller dispatcher mode.* Uses map KTLenumerated to convert or format disp_mod:

disp_mod (SharedMemeMap) rw `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for dispatcher mode keywords.* Valid values of disp_mod are:

- Exit : Exit dispatcher (shutdown dispatcher process)
- Ready : Normal operating state. Devices can be calibrated and set to target states.
- Keyword : Keyword validation mode. No I/O to any device. No internal changes to dispatcher state via keywords, except to the dispatcher mode. Return SUCCESS status message, but don't simulate I/O requests.
- Simulate : Simulate I/O. No device I/O occurs. A success message is returned after device I/O request is simulated.
- Diagnostic : Run diagnostics routines selected via the DISPxTST keyword for the dispatcher.
- Reconfig : Reload the dispatcher configuration data from the configuration file.
- Configuration error : Read only state. Error in the configuration data in the configuration files.

DISP2MSG (DEIMOS2) rw `varchar(68) A %s` *Latest message regarding instrument controller #2's dispatcher.*

DISP2TST (DEIMOS2) rw `int I11 %d` (0.0 to 10.0) *Instrument controller #2's test mode.*

DISP2VER (DEIMOS2) r `varchar(68) A %s` *Instrument controller #2's dispatcher software version information.*

DISP2VRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Instrument controller #1's dispacther verbose log messages.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

DOORCOLL (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Collimator access hatch position. If the hatch is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.

- open : Door or hatch is open.

DOORDWR1 (DEIMOS2) r `char(6)` `A` `%d` (0.0 to 1.0) *Dewar #1 access hatch position. If the hatch is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORDWR2 (DEIMOS2) r `char(6)` `A` `%d` (0.0 to 1.0) *Dewar #2 access hatch position. If the hatch is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORELEC (DEIMOS2) r `varchar(68)` `A` `%d` (0.0 to 1.0) *State of the electronics enclosure hatches on the instrument. Shows TRUE if a hatch is open.* Valid values of DOORELEC are:

- closed : All hatches are closed.
- open : A hatch is open.

DOORFILT (DEIMOS2) r `char(6)` `A` `%d` (0.0 to 1.0) *Filter access door position. If the door is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORGRAT (DEIMOS2) r `char(6)` `A` `%d` (0.0 to 1.0) *Grating access door position. If the door is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORMAN (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Man access hatch position. If the hatch is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORMASK (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Slit mask access door position. If the door is open the instrument rotator motor power is cut off (see ROTATPOW).* Uses map KTLboolean to convert or format closed_open:

closed_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a door status boolean state (CLOSED/OPEN) keywords.* Valid values of closed_open are:

- closed : Door or hatch is closed.
- open : Door or hatch is open.

DOORS (deimos2) r `char(1) L1 %d` (0.0 to 1.0) *State of the doors and hatches on the instrument. Shows TRUE if a door or hatch is open. The ROTATPOW keyword shows the same status, since an open door will kill power to the instrument rotator.* Valid values of DOORS are:

- closed : All doors and hatches are closed.
- open : A door or hatch is open.

DWFILCAL (pfcam+deimos) rw `char(5) A %d` (0.0 to 1.0) *Dewar filter calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

DWFILENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Dewar filter second encoder raw position.*

DWFILERR (pfcam+deimos) r `varchar(68) A tinyint` *Dewar filter command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

DWFILFIP (pfcam+deimos) r `char(7) A %d` (0.0 to 1.0) *Dewar filter fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

DWFILFOP (pfcam+deimos) rw `char(3) A %d` (0.0 to 1.0) *Dewar filter fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

DWFILLCK (pfcam+deimos) rw `char(10) A10 %s` (0.0 to 1.0) *Dewar filter lock software switch.*

DWFILLIM (pfcam+deimos) r `varchar(68) A tinyint` *Dewar filter limit status.* Uses map KTLenumerated to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

DWFILMOD (pfcam+deimos) rw `char(8) A tinyint` *Dewar filter control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.

- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

DWFILMSG (pfcam+deimos) r `varchar(68) A %s` *Dewar filter message.*

DWFILNAM (DEIMOS2) rw `varchar(68) A map` *Dewar filter ordinal position name.* Uses map TableLook to convert or format DWFILORD:

- 0 ↔ Home
- 1 ↔ Filter 1
- 2 ↔ Filter 2
- 3 ↔ Filter 3
- 4 ↔ Filter 4
- 7 ↔ Filter 7
- 5 ↔ FIlter 5
- 6 ↔ Filter 6

DWFILORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 8.0) *Dewar filter ordinal position.* Uses map TableLook to convert or format DWFILRAW:

- 0 ↔ 0
- 1600 ↔ 1
- 3200 ↔ 2
- 4800 ↔ 3
- 6400 ↔ 4
- 8000 ↔ 5
- 9600 ↔ 6
- 12200 ↔ 7

DWFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 13000.0) *Dewar filter raw position.*

DWFILORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 8.0) *Dewar filter ordinal position.* Uses map TableLook to convert or format DWFILRAW:

- 0 ↔ 0
- 1600 ↔ 1
- 3200 ↔ 2

- 4800 ↔ 3
- 6400 ↔ 4
- 8000 ↔ 5
- 9600 ↔ 6
- 12200 ↔ 7

DWFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 13000.0)
Dewar filter raw position.

DWFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 13000.0)
Dewar filter raw position.

DWFILSTA (pfcam+deimos) r `char(14) A tinyint` *Dewar filter status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

DWFILTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0)
Dewar filter raw position target.

DWFILTST (pfcam+deimos) rw `varchar(68) A tinyint` *Dewar filter wheel test selection.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.

- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

DWFILVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 360.0) *Dewar filter wheel position in degrees.* Uses map Linear to convert or format DWFILRAW:

- a0 : 0.00
- a1 : 1.00

DWFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 13000.0) *Dewar filter raw position.*

DWFILVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 30.0) *Dewar filter velocity.*

DWFILVRB (pfcam+deimos) rw `char(5) A8 %d` (0.0 to 1.0) *Dewar filter verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

DWFOCCAL (pfcam+deimos) rw `char(5) A %d` (0.0 to 1.0) *Dewar focus calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

DWFOCENC (deimos2) r `int I11 %d` *Dewar focus second encoder raw position.*

DWFOCERR (pfcam+deimos) r `varchar(68) A tinyint` *Dewar focus command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

DWFOCFIP (pfcam+deimos) r `char(7) A %d` (0.0 to 1.0) *Dewar focus fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

DWFOCFOP (pfcam+deimos) rw `char(3) A %d` (0.0 to 1.0) *Dewar focus fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

DWFOCLCK (pfcam+deimos) rw `char(10) A10 %s` (0.0 to 1.0) *Dewar focus lock software switch.*

DWFOCLIM (pfcam+deimos) r `varchar(68) A tinyint` *Dewar focus limit status.* Uses map KTLenumerated to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :

- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

DWFOCMOD (pfcam+deimos) rw `char(8) A tinyint` *Dewar focus control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

DWFOCMSG (pfcam+deimos) r `varchar(68) A %s` *Dewar focus message.*

DWFOCNAM (DEIMOS2) rw `varchar(68) A map` *Dewar focus ordinal position name.* Uses map TableLook to convert or format DWFOCORD:

- 0 ↔ Home
- 1 ↔ Nominal

DWFOCORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Dewar focus ordinal position.* Uses map TableLook to convert or format DWFOCRAW:

- 0 ↔ 0
- 5000 ↔ 1

DWFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar focus raw position.*

DWFOCORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Dewar focus ordinal position.* Uses map TableLook to convert or format DWFOCRAW:

- 0 ↔ 0
- 5000 ↔ 1

DWFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar focus raw position.*

DWFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar focus raw position.*

DWFOCSTA (pfcam+deimos) r `char(14) A tinyint` *Dewar focus status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

DWFOCTRG (DEIMOS2) r `int` `I11` `%d` *[encoder counts]* (0.0 to 10000.0) *Dewar focus raw position target.*

DWFOCTST (pfcam+deimos) rw `varchar(68)` `A` `tinyint` *Dewar focus test selection.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint` `I11` `%d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

DWFOCVAL (DEIMOS2) rw `real` `E13.6` `map` *[mm]* (0.0 to 10.0) *Dewar focus position in millimeters.* Uses map Linear to convert or format DWFOCRAW:

- a0 : 0.00
- a1 : 1.00

DWFOCRAW (DEIMOS2) rw `int` `I11` `%d` *[encoder counts]* (0.0 to 10000.0) *Dewar focus raw position.*

DWFOCVEL (DEIMOS2) rw `real` `E13.6` `%E` *[mm/s]* (0.0 to 0.25) *Dewar focus velocity.*

DWFOCVRB (pfcam+deimos) rw `char(5)` `A8` `%d` (0.0 to 1.0) *Dewar focus verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

DWXL8CAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Dewar translation stage calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

DWXL8ENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Dewar translation second encoder raw position.*

DWXL8ERR (DEIMOS2) r `varchar(68) A tinyint` *Dewar translation command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

DWXL8FIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Dewar translation fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

DWXL8FOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Dewar translation fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :

- on :

DWXL8LCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Dewar translation lock software switch.*

DWXL8LIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Dewar translation limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

DWXL8MOD (DEIMOS2) rw `char(8) A tinyint` *Dewar translation control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

DWXL8MSG (DEIMOS2) r `varchar(68) A %s` *Dewar translation message.*

DWXL8NAM (DEIMOS2) rw `varchar(68) A map` *Dewar translation ordinal position name.* Uses map TableLook to convert or format DWXL8ORD:

DWXL8ORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Dewar transaltion ordinal position.* Uses map TableLook to convert or format DWXL8RAW:

DWXL8RAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar translation raw position.*

DWXL8ORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Dewar transaltion ordinal position.* Uses map TableLook to convert or format DWXL8RAW:

DWXL8RAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar translation raw position.*

DWXL8RAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar translation raw position.*

DWXL8STA (DEIMOS2) r `char(14) A tinyint` *Dewar translation status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.

- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

DWXL8TRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar translation raw position target.*

DWXL8TST (DEIMOS2) rw `varchar(68) A tinyint` *Dewar translation test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

DWXL8VAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *Dewar translation position in millimeters.* Uses map Linear to convert or format DWXL8RAW:

DWXL8RAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Dewar translation raw position.*

DWXL8VEL (DEIMOS2) rw `real E13.6 %E` *[mm/s]* (0.0 to 0.25) *Dewar transaltion velocity.*

DWXL8VRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Dewar translation verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :

- true :

FIBARCFG (deimos2) rw `varchar(68) A %s` *Filter bar code reader configuration.*

FIBARCMD (deimos2) w `varchar(68) A %s` *Filter bar code reader command. A command for the bar code reader in its native command language. The response to the command is available via the FIBARRSP keyword.*

FIBARGET (deimos2) r `varchar(68) A %s` *Read filter bar code. Returns the last bar code read by the scanner. Use the FIBARSCN keyword to read a bar code.*

FIBARRSP (deimos2) r `varchar(68) A %s` *Filter bar code reader response to a bar code reader command sent via the FIBARCMD keyword.*

FIBARSCN (deimos2) w `char(3) A %d` (0.0 to 1.0) *Scan filter bar code reader. An ON value pulses the laser for the bar code reader. Use the FIBARGET keyword to get the last bar code scanned. An OFF value turns off the laser, if it is ON.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

G1TLTCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Grating #1 tilt calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

G1TLTENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Grating #1 tilt second encoder raw position.*

G1TLTERR (DEIMOS2) r `varchar(68) A tinyint` *Gratiing #1 tilt command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

G1TLTFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Grating #1 tilt fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

G1TLTFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Grating #1 tilt fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

G1TLTLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Grating #1 tilt lock software switch.*

G1TLTLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Grating #1 tilt limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :

- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

G1TLTMOD (DEIMOS2) rw `char(8) A tinyint` *Grating #1 tilt control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

G1TLTMSG (DEIMOS2) r `varchar(68) A %s` *Grating #1 tilt message.*

G1TLTNAM (DEIMOS2) rw `varchar(68) A map` *Grating #1 tilt ordinal position name.* Uses map TableLook to convert or format G1TLTORD:

- 0 ↔ Home
- 1 ↔ Nominal

G1TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #1 tilt ordinal position.* Uses map TableLook to convert or format G1TLTRAW:

- 0 ↔ 0
- 5000 ↔ 1

G1TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #1 tilt raw position.*

G1TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #1 tilt ordinal position.* Uses map TableLook to convert or format G1TLTRAW:

- 0 ↔ 0
- 5000 ↔ 1

G1TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #1 tilt raw position.*

G1TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #1 tilt raw position.*

G1TLTSTA (DEIMOS2) r `char(14) A tinyint` *Grating #1 tilt status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

G1TLTTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #1 tilt raw position target.*

G1TLTTST (DEIMOS2) rw `varchar(68) A tinyint` *Grating #1 tilt test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

G1TLTVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 43.0) *Grating #1 tilt position in degrees.* Uses map Linear to convert or format G1TLTRAW:

- a1 : 1.0
- a0 : 0.0

G1TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #1 tilt raw position.*

G1TLTVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 0.25) *Grating #1 tilt velocity.*

G1TLTVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Grating #1 tilt verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

G3TLTCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Grating #3 tilt calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

G3TLTENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Grating #3 tilt second encoder raw position.*

G3TLTERR (DEIMOS2) r `varchar(68) A tinyint` *Grating #3 tilt command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

G3TLTFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Grating #3 tilt fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

G3TLTFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Grating #3 tilt fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :

- on :

G3TLTLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Grating #3 tilt lock software switch.*

G3TLTLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Grating #3 tilt limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

G3TLTMOD (DEIMOS2) rw `char(8) A tinyint` *Grating #3 tilt control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

G3TLTMSG (DEIMOS2) r `varchar(68) A %s` *Grating #3 tilt message.*

G3TLTNAM (DEIMOS2) rw `varchar(68) A map` *Grating #3 tilt ordinal position name.* Uses map TableLook to convert or format G3TLTORD:

- 0 ↔ Home
- 1 ↔ Nominal

G3TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #3 tilt ordinal position.* Uses map TableLook to convert or format G3TLTRAW:

- 0 ↔ 0
- 5000 ↔ 1

G3TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #3 tilt raw position.*

G3TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #3 tilt ordinal position.* Uses map TableLook to convert or format G3TLTRAW:

- 0 ↔ 0
- 5000 ↔ 1

G3TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #3 tilt raw position.*

G3TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #3 tilt raw position.*

G3TLTSTA (DEIMOS2) r `char(14) A tinyint` *Grating #3 tilt status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

G3TLTTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #3 tilt raw position target.*

G3TLTTST (DEIMOS2) rw `varchar(68) A tinyint` *Grating #3 tilt test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

G3TLTVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 43.0) *Grating #3 tilt position in degrees.* Uses map Linear to convert or format G3TLTRAW:

- a0 : 0.0
- a1 : 1.0

G3TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #3 tilt raw position.*

G3TLTVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 0.25) *Grating #3 tilt velocity.*

G3TLTVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Grating #3 tilt verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

G4TLTCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Grating #4 tilt calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

G4TLTENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Grating #4 tilt second encoder raw position.*

G4TLTERR (DEIMOS2) r `varchar(68) A tinyint` *Gratiing #4 tilt command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

G4TLTFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Grating #4 tilt fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

G4TLTFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Grating #4 tilt fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

G4TLTLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Grating #4 tilt lock software switch.*

G4TLTLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Grating #4 tilt limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :

- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

G4TLTMOD (DEIMOS2) rw `char(8) A tinyint` *Grating #4 tilt control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

G4TLTMSG (DEIMOS2) r `varchar(68) A %s` *Grating #4 tilt message.*

G4TLTNAM (DEIMOS2) rw `varchar(68) A map` *Grating #4 tilt ordinal position name.* Uses map TableLook to convert or format G4TLTORD:

- 0 ↔ Home
- 1 ↔ Nominal

G4TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #4 tilt ordinal position.* Uses map TableLook to convert or format G4TLTRAW:

- 0 ↔ 0
- 5000 ↔ 1

G4TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #4 tilt raw position.*

G4TLTORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Grating #4 tilt ordinal position.* Uses map TableLook to convert or format G4TLTRAW:

- $0 \leftrightarrow 0$
- $5000 \leftrightarrow 1$

G4TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #4 tilt raw position.*

G4TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #4 tilt raw position.*

G4TLTSTA (DEIMOS2) r `char(14) A tinyint` *Grating #4 tilt status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

G4TLTTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #4 tilt raw position target.*

G4TLTTST (DEIMOS2) rw `varchar(68) A tinyint` *Grating #4 tilt test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.

- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

G4TLTVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 43.0) *Grating #4 tilt position in degrees.* Uses map Linear to convert or format G4TLTRAW:

- a0 : 0.0
- a1 : 1.0

G4TLTRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating #4 tilt raw position.*

G4TLTVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 0.25) *Grating #4 tilt velocity.*

G4TLTVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Grating #4 tilt verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

GRBARCFG (deimos2) rw `varchar(68) A %s` *Grating bar code reader configuration.*

GRBARCMD (deimos2) w `varchar(68) A %s` *Grating bar code reader command. A command for the bar code reader in its native command language. The response to the command is available via the GRBARRSP keyword.*

GRBARGET (deimos2) r `varchar(68) A %s` *Read grating bar code. Returns the last bar code read by the scanner. Use the FIBARSCN keyword to read a bar code.*

GRBARRSP (deimos2) r `varchar(68) A %s` *Grating bar code reader response to a bar code reader command sent via the GRBARCMD keyword.*

GRBARSCN (deimos2) w `char(3) A %d` (0.0 to 1.0) *Scan grating bar code reader. An ON value pulses the laser for the bar code reader. Use the GRBARGET keyword to get the last bar code* *scanned. An OFF value turns off the laser, if it is ON.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

GRCLAMP1 (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of grating slide clamp #1.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

GRCLAMP2 (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of grating slide clamp #2.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

GRCLAMP3 (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of grating slide clamp #3.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

GRCLAMP4 (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of grating slide clamp #4.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

GRCLAMP5 (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of grating slide clamp #5.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

GRCLAMPS (deimos2) rw `char(14) A %d` (0.0 to 1.0) *State of all grating slide clamps.* Uses map KTLboolean to convert or format clamp_pos:

clamp_pos (SharedMemeMap) rw `tinyint I11 %d` (0.0 to 5.0) *Generic map meme for lock or clamp state keywords.* Valid values of clamp_pos are:

- unlocked : Lock or clamp unlocked.
- locked : Lock or clamp locked.
- unlocking : Lock or clamp unlocking.
- locking : Lock or clamp locking.
- stage stuck : Neither locked or unlocked signal detected. Mechanical or electrical problem.
- signal failure : Both locked and unlocked signal detected. Electrical problem.

GRSELCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Grating selection calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

GRSELENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Grating selection second encoder raw position.*

GRSELERR (DEIMOS2) r `varchar(68) A tinyint` *Grating selection command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

GRSELFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Grating selection fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

GRSELFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Grating selection fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

GRSELLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Grating selection lock software switch.*

GRSELLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Grating selection limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :

- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

GRSELMOD (DEIMOS2) rw `char(8) A tinyint` *Grating selection control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

GRSELMSG (DEIMOS2) r `varchar(68) A %s` *Grating selection message.*

GRSELNAM (DEIMOS2) rw `varchar(68) A map` *Grating selection ordinal position name.* Uses map TableLook to convert or format GRSELORD:

- 0 ↔ Home
- 2 ↔ Mirror
- 1 ↔ Grating 1
- 3 ↔ Grating 3
- 4 ↔ Grating 4

GRSELORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 4.0) *Grating selection ordinal position.* Uses map TableLook to convert or format GRSELRAW:

- 0 ↔ 0
- 5000 ↔ 1
- 6000 ↔ 2
- 7000 ↔ 3
- 8000 ↔ 4

GRSELRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating selection raw position.*

GRSELORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 4.0) *Grating selection ordinal position.* Uses map TableLook to convert or format GRSELRAW:

- 0 ↔ 0
- 5000 ↔ 1
- 6000 ↔ 2
- 7000 ↔ 3
- 8000 ↔ 4

GRSELRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating selection raw position.*

GRSELRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating selection raw position.*

GRSELSTA (DEIMOS2) r `char(14) A tinyint` *Grating selection status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.

- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

GRSELTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating selection raw position target.*

GRSELTST (DEIMOS2) rw `int I11 %d` (0.0 to 10.0) *Grating selection test mode.*

GRSELVAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *Grating selection position in millimeters.* Uses map Linear to convert or format GRSELRAW:

- a0 : 0.00
- a1 : 1.00

GRSELRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Grating selection raw position.*

GRSELVEL (DEIMOS2) rw `real E13.6 %E` *[mm/s]* (0.0 to 0.25) *Grating selection velocity.*

GRSELVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Grating selection verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

HATCHPOS (deimos2) rw `char(14) A map` (0.0 to 5.0) *Instrument light path hatch position.* Uses map TableLook to convert or format cover_pos:

cover_pos (SharedMemeMap) rw `int I11 %d` (0.0 to 5.0) *Generic map meme for cover state keywords.* Valid values of cover_pos are:

- Closed : Cover closed.
- Open : Cover open.

- Closing : Cover closing.
- Opening : Cover opening.
- Stage Stuck : Neither closed or open signal detected. Mechanical or electrical problem.
- Signal Failure : Both closed and open signal detected. Electrical problem.

infoman (KTLinst) `bundle bundle` *The DEIMOS NOST1 (mostly infoman) keywords as needed by Drew to finish his proto UI.* infoman consists of elements:

FRAMENO (KECK1DAS) rw `int I11 %d` (0.0 to 9999.0) *Frame number of the next (or currently in-progress) exposure. This is appended after OUTFILE when constructing the name of the output disk file.*

IMTYPE (KECK1CCD) `varchar(68) A %s` *Indicates the manner in which a CCD with multiple amplifiers was read out.* Valid values of IMTYPE are:

- ONEAMP :
- SPIRAL :
- TWOAMPTOP :

OBJECT (NOST1) rw `varchar(68) A %s` *The value field shall contain a character string giving the name of the object observed.*

OBSERVER (NOST1) rw `varchar(68) A %s` *observer: The value field shall contain a character string identifying who acquired the data associated with the header.*

OUTDIR (KECK1DAS) rw `varchar(68) A %s` *A character string containing the default directory into which new CCD images are to be written. If this disk is full the images may be written elsewhere.*

OUTFILE (KECK1DAS) rw `varchar(68) A %s` *A character string defining the root name of the file(s) into which newly acquired data will be written if LTODISK is true. The full path is constructed from OUTDIR, OUTFILE, LFRAMENO, and DFORM.*

INSTRCAL (dispatch2) rw `char(5) A %d` (0.0 to 1.0) *Calibrate or reset calibration state of all devices that are part of the instrument.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

INSTRLCK (dispatch2) rw `char(10) A10 %s` (0.0 to 1.0) *Lock all device motion for the entire instrument. Stops all moving stages as quickly as possible. Prevents devices from changing state when set true. Terminates diagnostics.*

INSTRMOD (dispatch2) rw `char(8) A tinyint` *Instrument mode. (up, down, starting up, shutting down, error).* Uses map KTLenumerated to convert or format stage_mod:

- 0 ↔ shutdown
- 1 ↔ startup
- 3 ↔ starting up
- 2 ↔ shutting down
- 4 ↔ error

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

INSTRMSG (dispatch2) rw `varchar(68) A %s` *Message keyword for general purpose, instrument-wide information.*

IONDVPMP (deimos2) r `char(3) A %d` (0.0 to 1.0) *Ion pump status for the detector vessel ion pump.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

IONN2PMP (deimos2) r `char(3) A %d` (0.0 to 1.0) *Ion pump status for the liquid nitrogen can ion pump.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPAR1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *Argon comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPHE1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *Helium comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPHG1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *Mercury comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :

- on :

LAMPNE1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *Neon comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPQTZ1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *Quartz comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPS (lamps) rw `char(3) A %d` (0.0 to 1.0) *State of the calibration lamps. Shows ON if a lamp is on and can be used to turn all lamps off.* Uses map KTLboolean to convert or format off_on:

- 0 $\leftrightarrow$ off
- 1 $\leftrightarrow$ on

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPXT1 (lamps) rw `char(3) A %d` (0.0 to 1.0) *External (dome) comparison lamp #1 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

LAMPXT2 (lamps) rw `char(3) A %d` (0.0 to 1.0) *External (dome) comparison lamp #2 state.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU0CON (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *State of the auto/manual control switch on the manual control box for motor controller #0 (carriage motor controller).* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

MANU0FWD (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the forward button on the manual control box for motor controller #0 (carriage motor controller).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU0MOD (DEIMOS2) r `char(17) A tinyint` *State of the manual control mode switch on the manual control box for motor controller #0 (carriage motor controller).* Uses map KTLenumerated to convert or format man_mod:

man_mod (SharedMemeMap) r `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for manual control mode keywords.* Valid values of man_mod are:

- Low speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at slow jog speed.

- High speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at high jog speed.
- Discrete position : FWD and REV manual control buttons move selected stage one defined position forward or reverse.
- Home : FWD and REV manual control buttons home selected stage.
- Calibration lamps : FWD and REV manual control buttons cause selected lamp to turn on or off.
- Spare :
- Spare :
- Spare :

MANU0REV (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the reverse button on the manual control box for motor controller #0 (carriage motor controller).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU0SEL (DEIMOS2) r `varchar(68) A tinyint` (0.0 to 7.0) *State of the stage select switch on the manual control box for motor controller #0 (carriage motor controller).* Valid values of MANU0SEL are:

- rotator : Instrument rotator stage.
- spare : spare
- spare : spare
- spare : spare
- spare : spare
- spare : spare
- spare : spare
- spare : spare

MANU0STP (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the stop button on the manual control box for motor controller #0 (carriage motor controller).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU1CON (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *State of the auto/manual control switch on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

MANU1FWD (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the forward button on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU1HAT (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the hatch control button on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU1MOD (DEIMOS2) r `char(17) A tinyint` *State of the manual control mode switch on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLenumerated to convert or format man_mod:

man_mod (SharedMemeMap) r `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for manual control mode keywords.* Valid values of man_mod are:

- Low speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at slow jog speed.
- High speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at high jog speed.
- Discrete position : FWD and REV manual control buttons move selected stage one defined position forward or reverse.
- Home : FWD and REV manual control buttons home selected stage.
- Calibration lamps : FWD and REV manual control buttons cause selected lamp to turn on or off.
- Spare :
- Spare :
- Spare :

MANU1MSK (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the slit mask insertion button on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU1REV (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the reverse button on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU1SEL (DEIMOS2) r `varchar(68) A tinyint` (0.0 to 7.0) *State of the stage select switch on the manual control box for motor controller #1 (instrument motor controller #1).* Valid values of MANU1SEL are:

- TV filter/Quartz :
- TV focus/Neon :
- TV aperature/Argon :
- Spare/Mercury-Cadnium :
- Slit mask select/Helium :
- Spare/Spare :
- Spare/Spare :
- Spare/Spare :

MANU1STP (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the stop button on the manual control box for motor controller #1 (instrument motor controller #1).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU2CON (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *State of the auto/manual control switch on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

MANU2FWD (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the forward button on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU2GRC (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the grating clamp button on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU2MOD (DEIMOS2) r `char(17) A tinyint` *State of the manual control mode switch on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLenumerated to convert or format man_mod:

man_mod (SharedMemeMap) r `tinyint I11 %d` (0.0 to 7.0) *Generic map meme for manual control mode keywords.* Valid values of man_mod are:

- Low speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at slow jog speed.
- High speed jog : FWD and REV manual control buttons cause selected stage to jog forward or reverse at high jog speed.
- Discrete position : FWD and REV manual control buttons move selected stage one defined position forward or reverse.
- Home : FWD and REV manual control buttons home selected stage.
- Calibration lamps : FWD and REV manual control buttons cause selected lamp to turn on or off.
- Spare :
- Spare :
- Spare :

MANU2REV (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the reverse button on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

MANU2SEL (DEIMOS2) r `varchar(68) A tinyint` (0.0 to 7.0) *State of the stage select switch on the manual control box for motor controller #2 (instrument motor controller #2).* Valid values of MANU2SEL are:

- Grating tilt #1 :
- Grating tilt #2 :
- Grating tilt #4 :
- Grating slide :
- Dewar focus :
- Dewar translation :
- Filter :
- Spare :

MANU2STP (DEIMOS2) r `char(3) A %d` (0.0 to 1.0) *State of the stop button on the manual control box for motor controller #2 (instrument motor controller #2).* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

N2DWRFIL (DEIMOS2) rw `char(1) L1 map` (0.0 to 1.0) *Dewar nitrogen fill valve state.* Valid values of N2DWRFIL are:

- closed : Valve is closed.
- open : Valve is open. Dewar is filling.

N2DWRLVL (DEIMOS2) r `real E13.6 map` *[percent]* (0.0 to 100.0) *Dewar nitrogen level as a percentage of full.* Uses map Linear to convert or format N2DWRRAW:

N2DWRRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Dewar nitrogen level in raw A/D converter counts.*

N2DWRRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Dewar nitrogen level in raw A/D converter counts.*

N2FILMOD (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Dewar nitrogen fill control mode.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

N2RESLVL (DEIMOS2) r `real E13.6 map` *[percent]* (0.0 to 100.0) *50 liter reservoir nitrogen level as a percentage of full.* Uses map Linear to convert or format N2RESRAW:

N2RESRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *50 liter reservoir nitrogen level in raw A/D converter counts.*

N2RESRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *50 liter reservoir nitrogen level in raw A/D converter counts.*

OBSINPRG (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Observation in progress signal. Produces a visual or auditory signal on nasmyth platform.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

POWER (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Master power shutdown switch. Shuts off power to all DEIMOS electronics.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

POWH0MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *High voltage (+-15 volts) logic power supply voltage monitor for the carriage.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWH0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the carriage in raw A/D converter counts.*

POWH0VAL (DEIMOS2) r `real E13.6 map` *[V]* (-5.0 to 20.0) *External logic high power (+-15 volt) supply voltage for the carriage in volts.* Uses map Linear to convert or format POWH0RAW:

POWH0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the carriage in raw A/D converter counts.*

POWH2MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *High voltage (+-15 volts) logic power supply voltage monitor for the instrument.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWH2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the instrument in raw A/D converter counts.*

POWH2VAL (DEIMOS2) r `real E13.6 map` *[V]* (-5.0 to 20.0) *High voltage (+-15 volts) logic power supply voltage for the instrument in volts.* Uses map Linear to convert or format POWH2RAW:

POWH2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the instrument in raw A/D converter counts.*

POWL0MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Low voltage (+5 volts) logic power supply voltage monitor for the carriage.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWL0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Low voltage (+5 volts) logic power supply voltage for the carriage in raw A/D converter counts.*

POWL0VAL (DEIMOS2) r `real E13.6 map` *[V]* (-5.0 to 10.0) *Low voltage (+5 volts) logic power supply voltage for the carriage in volts.* Uses map Linear to convert or format POWL0RAW:

POWL0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Low voltage (+5 volts) logic power supply voltage for the carriage in raw A/D converter counts.*

POWL2MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Low voltage (+5 volts) logic power supply voltage monitor for the instrument.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWL2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Low voltage (+5 volts) logic power supply voltage for the instrument in raw A/D converter counts.*

POWL2VAL (DEIMOS2) r `real E13.6 map` *[V]* (-5.0 to 10.0) *Low voltage (+5 volts) logic power supply voltage for the instrument in volts.* Uses map Linear to convert or format POWL2RAW:

POWL2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Low voltage (+5 volts) logic power supply voltage for the instrument in raw A/D converter counts.*

POWM0MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Carriage motor power supply (POWM0) monitor.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWM0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage motor power supply (motor power supply #0) voltage in ADC counts.*

POWM0VAL (DEIMOS2) r `real E13.6 map` *[V]* (0.0 to 50.0) *Carriage motor power supply (motor power supply #0) voltage.* Uses map Linear to convert or format POWM0RAW:

POWM0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage motor power supply (motor power supply #0) voltage in ADC counts.*

POWM1MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Instrument motor power supply #1 (POWM1) monitor.* Uses map KTL-boolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWM1RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument motor power supply #1 voltage in ADC counts.*

POWM1VAL (DEIMOS2) r `real E13.6 map` *[V]* (0.0 to 50.0) *Instrument motor power supply #1 voltage.* Uses map Linear to convert or format POWM1RAW:

POWM1RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument motor power supply #1 voltage in ADC counts.*

POWM2MON (DEIMOS2) rw `char(6) A %d` (0.0 to 1.0) *Instrument motor power supply #2 (POWM2) monitor.* Uses map KTL-boolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

POWM2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument motor power supply #2 voltage in ADC counts.*

POWM2VAL (DEIMOS2) r `real E13.6 map` *[V]* (0.0 to 50.0) *Instrument motor power supply #2 voltage.* Uses map Linear to convert or format POWM2RAW:

POWM2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument motor power supply #2 voltage in ADC counts.*

ROTATCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Instrument rotator calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

ROTATENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Instrument rotator second encoder raw position.*

ROTATERR (DEIMOS2) r `varchar(68) A tinyint` *Instrument rotator command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

ROTATFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Instrument rotator fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

ROTATFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Instrument rotator fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

ROTATLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Instrument rotator lock software switch.*

ROTATLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Instrument rotator limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :

- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

ROTATMOD (DEIMOS2) rw `char(8) A %d` (0.0 to 1.0) *Instrument rotator control mode.* Uses map KTLboolean to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

ROTATMSG (DEIMOS2) r `varchar(68) A %s` *Instrument rotator message.*

ROTATNAM (DEIMOS2) rw `varchar(68) A map` *Instrument rotator ordinal position name.* Uses map TableLook to convert or format ROTATORD:

- 0 ↔ Home
- 1 ↔ Nominal

ROTATORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Instrument rotator ordinal position.* Uses map TableLook to convert or format ROTATRAW:

- 0 ↔ 0
- 5000 ↔ 1

ROTATRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Instrument rotator raw position.*

ROTATORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 2.0) *Instrument rotator ordinal position.* Uses map TableLook to convert or format ROTATRAW:

- 0 ↔ 0
- 5000 ↔ 1

ROTATRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Instrument rotator raw position.*

ROTATPOW (deimos2) r `char(1) L1 %s` (0.0 to 1.0) *Instrument rotator motor power. Tripped by an open door signal. See also DOORS, DOORDWR1, DOORDWR2, DOORMAN, DOORCOLL, DOORGRAT, DOORFILT, DOORMASK, DOORMEL1, DOORCEL1.*

ROTATRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Instrument rotator raw position.*

ROTATSTA (DEIMOS2) r `char(14) A tinyint` *Instrument rotator status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

ROTATTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Instrument rotator raw position target.*

ROTATTST (DEIMOS2) rw `varchar(68) A tinyint` *Instrument rotator test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

ROTATVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 10.0) *Instrument rotator position in degrees.* Uses map Linear to convert or format ROTATRAW:

ROTATRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Instrument rotator raw position.*

ROTATVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 0.25) *Instrument rotator velocity.*

ROTATVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Instrument rotator verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

ROTCLAMP (deimos2) r `char(8) A %d` (0.0 to 1.0) *State of the instrument rotation band clamp.* Uses map KTLboolean to convert or format unlock_lock:

unlock_lock (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (UNLOCKED/LOCKED) keywords.* Valid values of unlock_lock are:

- unlocked : Device unlocked or clamp released.
- locked : Device unlock or clamp set.

SLBARCFG (deimos2) rw `varchar(68) A %s` *Slit mask bar code reader configuration.*

SLBARCMD (deimos2) w `varchar(68) A %s` *Slit mask bar code reader command. A command for the bar code reader in its native command language. The response to the command is available via the SLBARRSP keyword.*

SLBARGET (deimos2) r `varchar(68) A %s` *Read slit mask bar code. Returns the last bar code read by the scanner. Use the FIBARSCN keyword to read a bar code.*

SLBARRSP (deimos2) r `varchar(68) A %s` *Slit mask bar code reader response to a bar code reader command sent via the SLBARCMD keyword.*

SLBARSCN (deimos2) w `char(3) A %d` (0.0 to 1.0) *Scan slit mask bar code reader. An ON value pulses the laser for the bar code reader. Use the SLBARGET keyword to get the last bar code scanned. An OFF value turns off the laser, if it is ON.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

SLCASCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Slit mask cassette calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

SLCASENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *Slit mask cassette second encoder raw position.*

SLCASERR (DEIMOS2) r `varchar(68) A tinyint` *Slit mask cassette command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

SLCASFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *Slit mask cassette fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

SLCASFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *Slit mask cassette fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

SLCASLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Slit mask cassette lock software switch.*

SLCASLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *Slit mask cassette limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :

- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

SLCASMOD (DEIMOS2) rw `char(8) A tinyint` *Slit mask cassette control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

SLCASMSG (DEIMOS2) r `varchar(68) A %s` *Slit mask cassette message.*

SLCASNAM (DEIMOS2) rw `varchar(68) A map` *Slit mask cassette ordinal position name.* Uses map TableLook to convert or format SLCASORD:

- 0 $\leftrightarrow$ Home

- 1 ↔ Mask_1
- 2 ↔ Mask_2
- 3 ↔ Mask_3
- 4 ↔ Mask_4
- 5 ↔ Mask_5
- 6 ↔ Mask_6
- 7 ↔ Mask_7
- 8 ↔ Mask_8
- 9 ↔ Mask_9
- 10 ↔ Mask_10
- 11 ↔ LongSlit_1
- 12 ↔ LongSlit_2
- 13 ↔ GridHole
- 14 ↔ Clear
- 15 ↔ Unload

SLCASORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 15.0) *Slit mask cassette ordinal position.* Uses map TableLook to convert or format SLCASRAW:

- 0 ↔ 0
- 5000 ↔ 1
- 5500 ↔ 2
- 6500 ↔ 3
- 7000 ↔ 4
- 7500 ↔ 5
- 8000 ↔ 6
- 8500 ↔ 7
- 9000 ↔ 8
- 9500 ↔ 9
- 10000 ↔ 10
- 10500 ↔ 11
- 11000 ↔ 12
- 11500 ↔ 13
- 12000 ↔ 14
- 12500 ↔ 15
- 13000 ↔ 16

SLCASRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 15000.0) *Slit mask cassette raw position.*

SLCASORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 15.0) *Slit mask cassette ordinal position.* Uses map TableLook to convert or format SLCASRAW:

- 0 ↔ 0
- 5000 ↔ 1
- 5500 ↔ 2
- 6500 ↔ 3
- 7000 ↔ 4
- 7500 ↔ 5
- 8000 ↔ 6
- 8500 ↔ 7
- 9000 ↔ 8
- 9500 ↔ 9
- 10000 ↔ 10
- 10500 ↔ 11
- 11000 ↔ 12
- 11500 ↔ 13
- 12000 ↔ 14
- 12500 ↔ 15
- 13000 ↔ 16

SLCASRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 15000.0) *Slit mask cassette raw position.*

SLCASRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 15000.0) *Slit mask cassette raw position.*

SLCASSTA (DEIMOS2) r `char(14) A tinyint` *Slit mask cassette status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.

- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

SLCASTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Slit mask cassette raw position target.*

SLCASTST (DEIMOS2) rw `varchar(68) A tinyint` *Slit mask cassette test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

SLCASVAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *Slit mask cassette position in millimeters.* Uses map Linear to convert or format SLCASRAW:

- a0 : 0.00
- a1 : 1.00

SLCASRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 15000.0) *Slit mask cassette raw position.*

SLCASVEL (DEIMOS2) rw `real E13.6 %E` *[mm/s]* (0.0 to 0.25) *Slit mask cassette velocity.*

SLCASVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *Slit mask cassette verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

SLMSKPOS (deimos2) rw `char(14) A %d` (0.0 to 1.0) *Slit mask insertion stage position.* Uses map KTLboolean to convert or format insert_pos:

insert_pos (SharedMemeMap) rw `tinyint I11 %d` (0.0 to 5.0) *Generic map meme for insertion state keywords.* Valid values of insert_pos are:

- out : OUT position of insertion system.
- in : IN position of insertion system.
- moving out : Moving to OUT position of insertion system.
- moving in : Moving to IN position of insertion system.
- stage stuck : Neither OUT or IN signal detected. Mechanical or electrical problem.
- signal failure : Both OUT and IN signal detected. Electrical problem.

SYSDIS (DEIMOS2) r `char(5) A8 %d` (0.0 to 1.0) *System disable switch status.* Uses map KTLboolean to convert or format false_true:

- 0 ↔ System enabled
- 1 ↔ System disabled

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

TMIRRCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *Tent mirror translation stage calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

TMIRRLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *Tent mirror translation lock software switch.*

TMIRRMSG (DEIMOS2) r `varchar(68) A %s` *Tent mirror translation message.*

TMIRRNAM (DEIMOS2) rw `varchar(68) A map` *Tent mirror translation ordinal position name.* Uses map TableLook to convert or format TMIRRORD:

- 1 ↔ Posn 1
- 2 ↔ Posn 2
- 3 ↔ Posn 3

TMIRRORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Tent mirror transaltion ordinal position.* Uses map TableLook to convert or format TMIRRRAW:

- 1000 ↔ 1
- 5000 ↔ 2
- 9000 ↔ 3

TMIRRRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Tent mirror translation raw position.*

TMIRRORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *Tent mirror transaltion ordinal position.* Uses map TableLook to convert or format TMIRRRAW:

- 1000 ↔ 1
- 5000 ↔ 2
- 9000 ↔ 3

TMIRRRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Tent mirror translation raw position.*

TMIRRRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Tent mirror translation raw position.*

TMIRRTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *Tent mirror translation raw position target.*

TMIRRVAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *Tent mirror translation position in millimeters.* Uses map Linear to convert or format TMIRRRAW:

- a0 : 0.00
- a1 : 1.00

TMIRRRAW (DEIMOS2) rw `int` `I11` `%d` *[encoder counts]* (0.0 to 10000.0) *Tent mirror translation raw position.*

TVAPECAL (DEIMOS2) rw `char(5)` `A` `%d` (0.0 to 1.0) *TV autoguider aperture calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.
- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

TVAPEENC (DEIMOS2) r `int` `I11` `%d` (0.0 to 10000.0) *TV autoguider aperture second encoder raw position.*

TVAPEERR (DEIMOS2) r `varchar(68)` `A` `tinyint` *TV autoguider aperture command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint` `I11` `%d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

TVAPEFIP (DEIMOS2) r `char(7)` `A` `%d` (0.0 to 1.0) *TV autoguider aperture fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

TVAPEFOP (DEIMOS2) rw `char(3)` `A` `%d` (0.0 to 1.0) *TV autoguider aperture fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint` `L1` `%s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

TVAPELCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *TV autoguider aperture lock software switch.*

TVAPELIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *TV autoguider aperture limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

TVAPEMOD (DEIMOS2) rw `char(8) A tinyint` *TV autoguider aperture control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

TVAPEMSG (DEIMOS2) r `varchar(68) A %s` *TV autoguider aperture message.*

TVAPENAM (DEIMOS2) rw `varchar(68) A map` *TV autoguider aperture ordinal position name.* Uses map TableLook to convert or format TVAPEORD:

- 0 ↔ Home
- 1 ↔ Narrow
- 2 ↔ Medium
- 3 ↔ Wide

TVAPEORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 8.0) *TV autoguider aperture ordinal position.* Uses map TableLook to convert or format TVAPERAW:

- 0 ↔ 0
- 1000 ↔ 1
- 5000 ↔ 2
- 9000 ↔ 3

TVAPERAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider aperture raw position.*

TVAPEORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 8.0) *TV autoguider aperture ordinal position.* Uses map TableLook to convert or format TVAPERAW:

- 0 ↔ 0
- 1000 ↔ 1

- 5000 ↔ 2
- 9000 ↔ 3

TVAPERAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0)
TV autoguider aperture raw position.

TVAPERAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0)
TV autoguider aperture raw position.

TVAPESTA (DEIMOS2) r `char(14) I11 tinyint` (0.0 to 6.0) *TV autoguider aperture status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

TVAPETRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0)
TV autoguider aperture raw position target.

TVAPETST (DEIMOS2) rw `varchar(68) A tinyint` *TV autoguider aperture test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0)
Generic map meme for stage test mode keywords. Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.

- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

TVAPEVAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *TV autoguider aperture position in millimeters.* Uses map Linear to convert or format TVAPERAW:

TVAPERAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider aperture raw position.*

TVAPEVEL (DEIMOS2) rw `real E13.6 %E` *[mm/s]* (0.0 to 0.25) *TV autoguider aperture velocity.*

TVAPEVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *TV autoguider aperture verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

TVFILENC (deimos2) r `int I11 %d` *TV autoguider filter second encoder raw position.*

TVFILERR (pfcam+deimos) r `varchar(68) A tinyint` *TV autoguider filter command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

TVFILFIP (pfcam+deimos) r `char(7) A %d` (0.0 to 1.0) *TV autoguider filter fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

TVFILFOP (pfcam+deimos) rw `char(3) A %d` (0.0 to 1.0) *TV autoguider filter fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

TVFILLCK (pfcam+deimos) rw `char(10) A10 %s` *TV autoguider filter lock software switch.*

TVFILLIM (pfcam+deimos) r `varchar(68) A tinyint` *TV autoguider filter limit status.* Uses map KTLenumerated to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :

- Check signals: FWD #2 & REV #1 limits active :

TVFILMOD (pfcam+deimos) rw `char(8) A tinyint` *TV autoguider filter control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.
- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

TVFILMSG (pfcam+deimos) r `varchar(68) A %s` *TV autoguider filter message.*

TVFILNAM (DEIMOS2) rw `varchar(68) A map` *TV autoguider filter ordinal position name.* Uses map TableLook to convert or format TVFILORD:

- 0 ↔ Home
- 1 ↔ B
- 2 ↔ V
- 3 ↔ R
- 4 ↔ I
- 5 ↔ Clear

TVFILORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 6.0) *TV autoguider filter ordinal position.* Uses map TableLook to convert or format TVFILRAW:

- 0 ↔ 0
- 1600 ↔ 1
- 3200 ↔ 2
- 4800 ↔ 3
- 6400 ↔ 4

- 8000 ↔ 5

TVFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 120000.0) *TV autoguider filter raw position.*

TVFILORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 6.0) *TV autoguider filter ordinal position.* Uses map TableLook to convert or format TVFILRAW:

- 0 ↔ 0
- 1600 ↔ 1
- 3200 ↔ 2
- 4800 ↔ 3
- 6400 ↔ 4
- 8000 ↔ 5

TVFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 120000.0) *TV autoguider filter raw position.*

TVFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 120000.0) *TV autoguider filter raw position.*

TVFILSTA (pfcam+deimos) r `char(14) A tinyint` *TV autoguider filter status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.
- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

TVFILTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider filter raw position target.*

TVFILTST (pfcam+deimos) rw `varchar(68) A tinyint` *TV autoguider filter test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

TVFILVAL (DEIMOS2) rw `real E13.6 map` *[deg]* (0.0 to 10.0) *TV autoguider filter position in degrees.* Uses map Linear to convert or format TVFILRAW:

TVFILRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 120000.0) *TV autoguider filter raw position.*

TVFILVEL (DEIMOS2) rw `real E13.6 %E` *[deg/s]* (0.0 to 0.25) *TV autoguider filter velocity.*

TVFILVRB (pfcam+deimos) rw `char(5) A8 %d` (0.0 to 1.0) *TV autoguider filter verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

TVFOCCAL (DEIMOS2) rw `char(5) A %d` (0.0 to 1.0) *TV autoguider focus calibration state.* Uses map KTLboolean to convert or format reset_homed:

reset_homed (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for stage calibration boolean state (RESET/HOMED) keywords.* Valid values of reset_homed are:

- reset : Stage not calibrated. Calibrate the stage during the next attempt to move the stage.

- homed : The stage is calibrated (read state). Calibrate the stage immediately (write state).

TVFOCENC (DEIMOS2) r `int I11 %d` (0.0 to 10000.0) *TV autoguider focus second encoder raw position.*

TVFOCERR (DEIMOS2) r `varchar(68) A tinyint` *TV autoguider focus command error number.* Uses map KTLenumerated to convert or format stage_err:

stage_err (SharedMemeMap) r `tinyint I11 %d` (0.0 to 1.0) *Generic map meme for stage error number keywords.* Valid values of stage_err are:

- No error : Last command succeeded.
- Error : Command failed (Generic error number).

TVFOCFIP (DEIMOS2) r `char(7) A %d` (0.0 to 1.0) *TV autoguider focus fiducial light sensor input.* Uses map KTLboolean to convert or format blocked_open:

blocked_open (SharedMemeMap) r `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for fiducial light sensor boolean state (BLOCKED/OPEN) keywords.* Valid values of blocked_open are:

- blocked : No light detected at fiducial sensor. Light source off or light path blocked.
- open : Light detected at fiducial sensor. Light source on and light path not blocked.

TVFOCFOP (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *TV autoguider focus fiducial light source control.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

TVFOCLCK (DEIMOS2) rw `char(10) A10 %s` (0.0 to 1.0) *TV autoguider focus lock software switch.*

TVFOCLIM (DEIMOS2) r `varchar(68) A %d` (0.0 to 63.0) *TV autoguider focus limit status.* Uses map KTLbitmask to convert or format stage_lim:

stage_lim (SharedMemeMap) r `tinyint I11 %d` (0.0 to 15.0) *Generic map meme for stage limit keywords. This map applies to stages* *with two inputs reporting the secondary limit status. For stages with a single input reporting both secondary limits states use Meme 4258(stage_lim_1sec).* Valid values of stage_lim are:

- Not in a limit :
- In REV primary limit :
- In FWD secondary limit :
- Check signals: FWD #2, FWD #1 & REV #1 limits active :
- Check signals: FWD #2 & REV #2 limits active :
- Check signals: FWD #2, REV #2 & REV #1 limits active :
- Check signals: FWD #2, REV #2 & FWD #1 limits active :
- Check signals: All limits active :
- In FWD primary limit :
- Check signals: FWD #1 & REV #1 limits active :
- Check signals: REV #2 limit active, but not REV #1 :
- In REV secondary limit :
- Check signals: REV #2 & FWD #1 limits active :
- Check signals: REV #2, REV #1 & FWD #1 limits active :
- Check signals: FWD #2 limit active, but not FWD #1 :
- Check signals: FWD #2 & REV #1 limits active :

TVFOCMOD (DEIMOS2) rw `char(8) A tinyint` *TV autoguider focus control mode.* Uses map KTLenumerated to convert or format stage_mod:

stage_mod (SharedMemeMap) rw `tinyint I11 %d` *[control mode]* (0.0 to 5.0) *Generic map meme for stage control mode keywords.* Valid values of stage_mod are:

- Halt : Halt stage. Prevent it from moving.
- Pos : Position mode. Use the default stage configuration to select motors on or off after moves.
- Jog : Jog mode. Prevent jogging unless stage is calibrated.
- Pos ON : Engineering position mode. Leave motors on after moves.
- Pos OFF : Engineering position mode. Shut motors off after moves.

- Jog ENGR : Engineering jog mode. Allow jogging when the stage isn't calibrated.

TVFOCMSG (DEIMOS2) r `varchar(68) A %s` *TV autoguider focus message.*

TVFOCNAM (DEIMOS2) rw `varchar(68) A map` *TV autoguider focus ordinal position name.* Uses map TableLook to convert or format TVFOCORD:

- 0 ↔ Home
- 1 ↔ Nominal

TVFOCORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *TV autoguider focus ordinal position.* Uses map TableLook to convert or format TVFOCRAW:

- 0 ↔ 0
- 5000 ↔ 1

TVFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider focus raw position.*

TVFOCORD (DEIMOS2) rw `tinyint I11 map` (0.0 to 5.0) *TV autoguider focus ordinal position.* Uses map TableLook to convert or format TVFOCRAW:

- 0 ↔ 0
- 5000 ↔ 1

TVFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider focus raw position.*

TVFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider focus raw position.*

TVFOCSTA (DEIMOS2) r `char(14) A tinyint` *TV autoguider focus status.* Uses map KTLenumerated to convert or format dc_stage_status:

dc_stage_status (SharedMemeMap) r `tinyint I11 %d` (0.0 to 6.0) *Generic map meme for stage status keywords.* Valid values of dc_stage_status are:

- Not Calibrated : The stage hasn't been homed yet.
- Calibrating : The stage is finding it's home reference position.
- Ready : The stage is stationary. It has been calibrated. It is ready to move.

- Moving : The stage is moving to a target position.
- Jogging : The stage is accelerating to, jogging at, or decellerating from a target velocity.
- Diagnostics : A selected diagnostics routine is running.

TVFOCTRG (DEIMOS2) r `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider focus raw position target.*

TVFOCTST (DEIMOS2) rw `varchar(68) A tinyint` *TV autoguider focus test mode.* Uses map KTLenumerated to convert or format stage_tst:

stage_tst (SharedMemeMap) rw `tinyint I11 %d` *[test mode]* (0.0 to 4.0) *Generic map meme for stage test mode keywords.* Valid values of stage_tst are:

- Off : Test mode off.
- Sim1 : Simulation mode #1. Reads return keyword id as value. Writes do nothing, but receive proper notification messages.
- Sim2 : Simulation mode #2. Reads return simulated keyword value. Writes set simulated keyword value immediately.
- Sim3 : Simulation mode #3. Reads return simulated keyword value. Writes set simulated keyword value slowly to simulate motion.
- Diag1 : Diagnostic mode #1. First diagnostic mode. Effects TBD.

TVFOCVAL (DEIMOS2) rw `real E13.6 map` *[mm]* (0.0 to 10.0) *TV autoguider focus position in millimeters.* Uses map Linear to convert or format TVFOCRAW:

TVFOCRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0) *TV autoguider focus raw position.*

TVFOCVEL (DEIMOS2) rw `real E13.6 %E` *[mm/s]* (0.0 to 0.25) *TV autoguider focus velocity.*

TVFOCVRB (DEIMOS2) rw `char(5) A8 %d` (0.0 to 1.0) *TV autoguider focus verbose message flag.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :

- true :

TVPOW (DEIMOS2) rw `char(3) A %d` (0.0 to 1.0) *TV autoguider AC power.* Uses map KTLboolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

UPSC1LOW (DEIMOS2) r `char(5) A8 %d` (0.0 to 1.0) *CCD electronics UPS battery power low.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

UPSC1MON (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *CCD electronics UPS monitor.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

UPSM1LOW (DEIMOS2) r `char(5) A8 %d` (0.0 to 1.0) *Motor electronics UPS battery power low.* Uses map KTLboolean to convert or format false_true:

false_true (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (FALSE/TRUE) keywords.* Valid values of false_true are:

- false :
- true :

UPSM1MON (DEIMOS2) r `char(6) A %d` (0.0 to 1.0) *Motor electronics UPS monitoring.* Uses map KTLboolean to convert or format manual_auto:

manual_auto (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

1.2.4 Map Definitions

DWRN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar liquid nitrogen level value (see DWRN2LVR) into a percentage full (see DWRN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

KTLbitmask (MapTypes) `int I11 %d` *[bitmask] This is the generic map meme for a keyword whose value is derived from a set of bits. In that sense it is an unsigned integer. The mmaps entries for such memes will be attached to the main meme or its isa, not to this map meme.*

KTLboolean (MapTypes) `tinyint I1 %d` (0.0 to 1.0) *A special case of the KTL Enumerated map, (q.v.). ktlBoolean has two legal int values, 0 and 1. When setting 't', 'true', or a nonzero int =¿1; and 'f' or 'false' =¿ 0. FIORD keywords report and accept string values such as 'open', 'closed', etc.*

KTLenumerated (MapTypes) `tinyint tinyint tinyint` *This map type is a KTL Enumerated map, meaning that KTL rather than the crate does the conversion from binary to ascii. Practically speaking, all this means is that the list of ascii res_vals ends up in fiord_table.h*

Linear (MapFuncs) `map map` *A map which indicates that you should perform a linear conversion from the min and max of the first meme to the min and max of the second meme (ISA target). Example: mumbleVAL ISA mumbleRAW, map is Linear.* Linear consists of elements:

- **a0** (MapParams) `float E20.12 %.12E` *Coefficient of the zero order term in a polynomial map. Typically this will be the constant term.*
- **a1** (MapParams) `float E20.12 %.12E` *Coefficient of the first order term in a polynomial map. Typically this will be the coefficient of the linear term.*

PWRBLOKM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar heater power block value (see PWRBLOKR) into a voltage (see PWRBLOK).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

RESN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar refill reservoir level (see RESN2LVR) into a percentage full (see RESN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

TableLook (MapTypes) `map map` *This map type indicates a simple table lookup using the Mmaps table. CodeGen will presume that this lookup is not being done within KTL/fiord code. Use of this map is probably equivalent to having a null map_mid.*

TEMPDETM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw detector temperature (see TEMPDETR) into degrees celsius (see TEMPDET).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

TEMPSETM (KECK1CCD) rw `varchar(68)` `A` `%s` *An array of linear unit conversion constants used to convert a raw detector temperature set point (see TEMPSETR) into degrees celsius (see TEMPSET).* Is really a:

isaMAP (TYPEDEFS) `tuple` `tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real` `E13.6` `%E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real` `E13.6` `%E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real` `E13.6` `%E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real` `E13.6` `%E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real` `E13.6` `%E` *The scale parameter in a mapping from device units to user units.*

UTBM15VM (KECK1CCD) rw `varchar(68)` `A` `%s` *An array of linear unit conversion constants used to convert a 'raw' -15 volt power supply voltage value from an analog-to-digital converter (see UTBM15VR) to a voltage (see UTBM15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple` `tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real` `E13.6` `%E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real` `E13.6` `%E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real` `E13.6` `%E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real` `E13.6` `%E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real` `E13.6` `%E` *The scale parameter in a mapping from device units to user units.*

UTBP15VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +15 volt power supply voltage value from an analog-to-digital converter (see UTBP15VR) to a voltage (see UTBP15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBP30VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +30 volt power supply voltage value from an analog-to-digital converter (see UTBP30VR) to a voltage (see UTBP30V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBP5VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +5 volt power supply voltage value from an analog-to-digital converter (see UTBP5VR) to a voltage (see UTBP5V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBTEMPM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert the raw CCD controller utility board temperature from an analog-to-digital converter value (see UTBTEMPR) to a value in degrees celsius (see UTBTEMP).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

UTBTMPSM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' CCD electronics enclosure temperature set point from an analog-to-digital converter (see UTBTMPSR) to a value in degrees celsius (see UTBTMPS).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

Chapter 2

DEIMOS Database Schema

2.1 Overview

This section provides detailed documentation of table design and field type and usage for all the schema (proposed and production) designed for DEIMOS.

2.2 Database Schema for Keywords (memes)

2.2.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.2.2 Regular Memes

MemeData (Memes) `schema schema` *The group of tables needed to describe memes: Memes, Mbundles, Mmaps.* MemeData consists of elements:

Memes (Memes) `table table` (EXTNAME = 'Memes') *Table of memes (meaning units); FITS keywords, syntactic subunits of FITS keywords, and database fields are all memes.* Memes consists of elements:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

name (DataBase) `char(16) A16 %s` *The name of a Meme (a FITS keyword, database column or table name, FITS HDU name, Map, or any other meme).*

alt_name (DataBase) `varchar(80) A80 %s` *Mostly for KTL, but could be useful elsewhere; this attribute allows us to establish a lengthier alias for these short and sometimes cryptic 8-char FITS keyword names. Alt use in device bundles: attrib list (colon-sep) used only by CodeGen*

context (DataBase) `char(16) A16 map` *Arbitrary attribute for grouping memes into related families, such as keywords relevant to an instrument, keywords relevant to a test phase, etc. It also disambiguates keywords with the same name but different meaning or format., e.g. PARANG.* Is an instance of mcontext:

mcontext (DataBase) `char(16) A16 %s` *Context code; the context code is attached to a meme to distinguish that meme from other memes which might happen to have the same name.*

See AZ in context HIRES vs AZ in context KECK1DCS, for example. The Mcontexts table defines meme contexts.

access (DataBase) `char(2) A2 %s` *Keyword access can be RW, R, or W (this is mostly useful in the context of KTL, but could be used for database access control as well).*

syty (DataBase) `varchar(40) A40 %s` *The 'sybase type' or "system type" is the host datatype, as opposed to the FORTRAN format. Host datatypes map well to sybase datatypes, such as real, float, int, smallint, etc.* Valid values of syty are:

- char : fixed length string
- datetime : 8 bytes, good to 3.33ms
- file : bundle of FITS HDU
- float : 64-bit float
- header : FITS header or subset thereof
- int : 32-bit integer
- map : a map type
- real : 32-bit float
- schema : a bundle of RDB tables
- smalldatetime : 4 bytes, good to 1 min
- smallint : 16-bit integer
- table : FITS/Sybase table bundle
- tinyint : 8-bit integer
- tuple : multi-component value
- varchar : variable length string

mfid (DataBase) `int I6 %d` *Primary Key (record ID) of the Mformats table*

units (DataBase) `char(40) A40 %s` *If the keyword value is expressed in known units, the units are stored here. IAU standard style is recommended.*

semantics (DataBase) `varchar(255) A255 %s` *The semantics (meaning and usage) of the entity are stored here. In the case of a meme, this is the meaning of the meme (or keyword); in the case of an enumerated value, it is the meaning of the value.*

isa_mid (DataBase) `int I4 map` *The MID of a known, canonical, re-usable meme which serves as a typedef for other memes. The purpose of the typedef may be to refer to the same meme* *by various names with different functions, or to convert/reformat a meme.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

map_mid (DataBase) `int I11 map` *In the Memes table, the map_mid points to a Map Type meme (context should be MapTypes). The Map Type meme may or may not have a Bundle of parameters. In any case, the Mmaps table will tell us about the map.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

ctrl_mid (DataBase) `char(12) A map` *Control MID: the MID of a counter keyword which controls the instantiation of an indexed keyword (FITS). This is actually a list of mids, as a keyword can be multiply indexed.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

rpt_mid (DataBase) `int I4 map` *Repeat MID: the MID of a counter keyword which controls the repetition of a group in a tuple keyword (FITS)* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

ifmt (DataBase) `char(24) A24 %s` *The printf format for the index part of a an indexed keyword name. Note that it is generally not possible to describe this using a Fortran format.*

minv (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the minimum permissible value for that keyword.*

maxv (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then maxv is the maximum permissible value for that keyword.*

nmin (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the nominal minimum value for that keyword.*

nmax (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the nominal maximum value for that keyword.*

defv (DataBase) `char(40) A40 %s` *If the keyword is missing from the header, but a defv value exists, the keyword may be inserted into the header with that value.*

nulv (DataBase) `char(40) A40 %s` *If the keyword exists in the header with a missing value, and a nulv values exists for that keyword, the missing value can be replaced with the nulv value. In a FITS table this string fills NULL values.*

tolv (DataBase) `float E20.12 %.12E` *Tolerance value: for numeric keywords representing such quantities as stage position, this means the 'close enough' value. An actual value that is within +- tolv of the target value will be accepted as a successful move.*

wildcard (DataBase) `char(20) A20 %s` *This field documents the value that a portion of a tuple meme should take on if its values is given as a wildcard instead of as an explicit value. The only use so far is for meme 1352. Alt use in Device Bundles: Device base ID.*

comment (DataBase) `varchar(80) A80 %s` *In a FITS keyword/value record this comment text should appear following the / that follows the value. If the maintainer wishes to attach a very brief comment, which is not part of the semantics, this field is provided for convenient annotation.*

uri (DataBase) `varchar(255) A255 %s` *If the keyword cannot be documented adequately via semantics and comments, a URI can be attached referring the reader to other online documentation, source code, etc.*

startd (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Beginning date at which the use of this meme in this context is valid. Should apply to bundles, not to normal memes, for the purpose of establishing dated FITS header and KTL service definitions.*

endd (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Ending date at which the use of this meme in this context is valid. See startd; this applies to bundles, not normal memes.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the* *table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

logtype (Database) `char(2) char(2) char(2)` *The 'logtype' attribute of a KTL keyword meme is a 2 char value presently E, S, or ES. E keywords are Event type keywords (strings), S are state type keywords (continuously varying values) and ES are strings with useful int equivs like ERR and CAL*

Mcontexts (Memes) `table table` (EXTNAME = 'Mcontexts') *Table of meme contexts and their meanings. Each context name is associated with an author, date, and meaning. 'Context' is a wholly arbitrary attribute for the developers' convenience. It enables us to disambiguate memes of identical name, for example.* Mcontexts consists of elements:

mcontext (DataBase) `char(16) A16 %s` *Context code; the context code is attached to a meme to distinguish that meme from other memes which might happen to have the same name. See AZ in context HIRES vs AZ in context KECK1DCS, for example. The Mcontexts table defines meme contexts.*

descrip (DataBase) `varchar(255) A255 %s` *Descrip is a generic text field in which one can add brief descriptive information to an database entity (record). The description is relevant to the table in which the field appears.*

author (DataBase) `varchar(40) A40 %s` *Various MemesWorld items can have authors. Software agents have authors. Meme contexts also (who invented the context?). Path contexts (drawings) also have authors. Although the product is different, the author field always indicates authorship.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

instrument (DataBase) `char(12) A12 %s` *Instrument name abbreviation for identifying contexts.*

subsystem (DataBase) `char(20) A20 %s` *Instrument subsystem (for identifying contexts)*

org (DataBase) `char(20) A20 %s` *Organization abbreviation (for identifying contexts). This might be UCOLick or Keck or CARA or NOAO . . . it indicates the general locus of authority for this context.*

doc (DataBase) `char(40) A40 %s` *Document (RFC, paper, etc) for identifying a meme context. This should probably be a URI. It's intended to point you to an official document (RFC, KSD, LOTR) describing the information in this context.*

authority (DataBase) `varchar(40) A40 %s` *String indicating authority over a context. This should be a person's name or email address. It answers the question, 'Who really knows what the memes in this context mean?'*

Mbundles (Memes) `table table` (EXTNAME = 'Mbundles') *Table of bundles or associations of memes; each master mid (tmid) is associated with several elements (emid).* Mbundles consists of elements:

mbid (DataBase) `int I11 %d` *Unique Record ID (primary key) of the Mbundles table.*

tmid (DataBase) `int I4 map` *The MID of a compound meme (table, header, or tuple) which is associated with multiple other memes in the Mbundles table.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

emid (DataBase) `int I4 map` *The MID of an element meme which is associated with multiple other memes under one compound meme (table, header, or tuple) in the Mbundles table.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

eordr (DataBase) `smallint I6 %d` *Establishes the fixed order of elements in a meme bundle. This enables us to define predictable, repeatable FITS headers and Sybase schema.*

einkey (DataBase) `char(2) A2 %s` *special characteristics of an element of a bundle of memes.* Valid values of einkey are:

- I : indexed field
- N : may be NULL
- O : optional?
- P : primary key component
- T : interpret as table

opt (DataBase) `char(1) A1 %s` *This field indicates whether a keyword is Optional in a bundle. All keywords not Optional* *are Required.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Mmaps (Memes) `table table` (EXTNAME = 'Mmaps') *The Mmaps table describes the mapping of meme values. Unit coversions, lookup tables, and lists of legal values are the customary types of map. See the context MapTypes for more information.* Mmaps consists of elements:

mmid (DataBase) `int I11 %d` *Primary key field (record ID) of table Mmaps. This field numbers each record for easy identification.*

res_mid (DataBase) `int I11 map` *Result MID of map table. The Result MID is the MID of the meme whose value resulted from the mapping, i.e. whose isa_mid points to a meme whose value is converted or mapped into its own. Just remember that this meme ISA something else.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

res_val (DataBase) `varchar(100) A %s` *Result Value of a meme map record. The result value is the value taken on by the res_mid meme as a result of the map transforming the value of the meme which this meme ISA.*

par_mid (DataBase) `int I11 map` *Parameter MID of a meme map record. This is NULL if the map is a simple lookup on a table of values of the ISA meme. If this is not null, then the map type must have been algorithmic rather than table driver and mmaps contains a list of parm values.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

par_val (DataBase) `varchar(100) A %s` *Parameter value of the par_mid meme. This is either a value of the 'deeper' meme which the mapped meme ISA (internal representation), or the value of the meme (coefficient) stored in par_mid.*

mcomment (DataBase) `varchar(255) A %s` *Comment field for explicating the mysteries of the Mmaps records.*

KTLgrp (Memes) `table table` (EXTNAME = 'KTLgrp') *Table defining KTL message groups or broadcast groups. Roughly speaking this means a bundle of memes which are applicable to a single hardware subsystem. See URI for more information.* Is really a:

Memes (Memes) `table table` (EXTNAME = 'Memes') *Table of memes (meaning units); FITS keywords, syntactic subunits of FITS keywords, and database fields are all memes.* Memes consists of elements:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

name (DataBase) `char(16) A16 %s` *The name of a Meme (a FITS keyword, database column or table name, FITS HDU name, Map, or any other meme).*

alt_name (DataBase) `varchar(80) A80 %s` *Mostly for KTL, but could be useful elsewhere; this attribute allows us to establish a lengthier alias for these short and sometimes cryptic 8-char FITS keyword names. Alt use in device bundles: attrib list (colon-sep) used only by CodeGen*

context (DataBase) `char(16) A16 map` *Arbitrary attribute for grouping memes into related families, such as keywords relevant to an instrument, keywords relevant to a test phase, etc. It also disambiguates keywords with the same name but different meaning or format., e.g. PARANG.* Is an instance of mcontext:

mcontext (DataBase) `char(16) A16 %s` *Context code; the context code is attached to a meme to distinguish that meme from other memes which might happen to have the same name. See AZ in context HIRES vs AZ in context KECK1DCS, for example. The Mcontexts table defines meme contexts.*

access (DataBase) `char(2) A2 %s` *Keyword access can be RW, R, or W (this is mostly useful in the context of KTL, but could be used for database access control as well).*

syty (DataBase) `varchar(40) A40 %s` *The 'sybase type' or "system type" is the host datatype, as opposed to the FORTRAN format. Host datatypes map well to sybase datatypes, such as real, float, int, smallint, etc.* Valid values of syty are:

- char : fixed length string
- datetime : 8 bytes, good to 3.33ms
- file : bundle of FITS HDU
- float : 64-bit float
- header : FITS header or subset thereof
- int : 32-bit integer
- map : a map type
- real : 32-bit float
- schema : a bundle of RDB tables
- smalldatetime : 4 bytes, good to 1 min
- smallint : 16-bit integer
- table : FITS/Sybase table bundle
- tinyint : 8-bit integer
- tuple : multi-component value
- varchar : variable length string

mfid (DataBase) `int I6 %d` *Primary Key (record ID) of the Mformats table*

units (DataBase) `char(40) A40 %s` *If the keyword value is expressed in known units, the units are stored here. IAU standard style is recommended.*

semantics (DataBase) `varchar(255) A255 %s` *The semantics (meaning and usage) of the entity are stored here. In the case of a meme, this is the meaning of the meme (or keyword); in the case of an enumerated value, it is the meaning of the value.*

isa_mid (DataBase) `int I4 map` *The MID of a known, canonical, re-usable meme which serves as a typedef for other memes. The purpose of the typedef may be to refer to the same meme by various names with different functions, or to convert/reformat a meme.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

map_mid (DataBase) `int I11 map` *In the Memes table, the map_mid points to a Map Type meme (context should be MapTypes). The Map Type meme may or may not have a Bundle of parameters. In any case, the Mmaps table will tell us about the* *map.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

ctrl_mid (DataBase) `char(12) A map` *Control MID: the MID of a counter keyword which controls the instantiation of an indexed keyword (FITS). This is actually a list of mids, as a keyword can be multiply indexed.* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

rpt_mid (DataBase) `int I4 map` *Repeat MID: the MID of a counter keyword which controls the repetition of a group in a tuple keyword (FITS)* Is an instance of mid:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

ifmt (DataBase) `char(24) A24 %s` *The printf format for the index part of a an indexed keyword name. Note that it is generally not possible to describe this using a Fortran format.*

minv (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the minimum permissible value for that keyword.*

maxv (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then maxv is the maximum permissible value for that keyword.*

nmin (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the nominal minimum value for that keyword.*

nmax (DataBase) `float E20.12 %.12E` *If a keyword value is numeric, then minv is the nominal maximum value for that keyword.*

defv (DataBase) `char(40) A40 %s` *If the keyword is missing from the header, but a defv value exists, the keyword may be inserted into the header with that value.*

nulv (DataBase) `char(40) A40 %s` *If the keyword exists in the header with a missing value, and a nulv values exists for that keyword, the missing value can be replaced with the nulv value. In a FITS table this string fills NULL values.*

tolv (DataBase) `float E20.12 %.12E` *Tolerance value: for numeric keywords representing such quantities as stage position, this means the 'close enough' value. An actual value that is within +- tolv of the target value will be accepted as a successful move.*

wildcard (DataBase) `char(20) A20 %s` *This field documents the value that a portion of a tuple meme should take on if its values is given as a wildcard instead of as an explicit value. The only use so far is for meme 1352. Alt use in Device Bundles: Device base ID.*

comment (DataBase) `varchar(80) A80 %s` *In a FITS keyword/value record this comment text should appear following the / that follows the value. If the maintainer wishes to attach a very brief comment, which is not part of the semantics, this field is provided for convenient annotation.*

uri (DataBase) `varchar(255) A255 %s` *If the keyword cannot be documented adequately via semantics and comments, a URI can be attached referring the reader to other online documentation, source code, etc.*

startd (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Beginning date at which the use of this meme in this context is valid. Should apply to bundles, not to normal memes, for the purpose of establishing dated FITS header and KTL service definitions.*

endd (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Ending date at which the use of this meme in this context is valid. See startd; this applies to bundles, not normal memes.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

logtype (Database) `char(2) char(2) char(2)` *The 'logtype' attribute of a KTL keyword meme is a 2 char value presently E, S, or ES. E keywords are Event type keywords (strings), S are state type keywords (continuously varying values) and ES are strings with useful int equivs like ERR and CAL*

KTLgrp consists of elements:

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

sidlist (DataBase) `varchar(80) A80 %s` *List of service IDs for a KTL broadcast group. This is a space separated list of obscure all-caps symbolic names relevant only when inside music/traffic code.*

style (DataBase) `char(12) A12 %s` *KTL style (a jargon word from inside CARA) indicates the type of communication used to acquire or set a keyword. Note this is an enum meme.* Valid values of style are:

- CA : Channel Access
- CCX : Some other protocol
- music : Music-style keywords
- RPC : RPC protocol

bcast (DataBase) `int I6 %d` *Broadcast group code: used to define groups of keywords for continuous monitoring*

Mpinouts (Memes) `table table` (EXTNAME = 'Mpinouts') *A table associating emids (memes in their incarnation as KTL bundle members, to be precise) with Lick drawing numbers, signal types, access types, r/w sense, etc. Intended for engineering support: instant reference from any KTL kw to the as-builts.* Mpinouts consists of elements:

mpnid (Database) `int I6 %d` *Pkey of table Mpinouts*

mbid (DataBase) `int I11 %d` *Unique Record ID (primary key) of the Mbundles table.*

dwgno (DataBase) `char(12) A12 %s` *Lick Observatory drawing number. This refers to the PCAD drawings documenting the control electronics.*

sense (DataBase) `char(3) A3 %s` *Is the signal read or write, from the controller's perspective?*

ctrltype (DataBase) `char(12) A12 %s` *Type of controller, as in Galil servo motor controller vs Piezo, Bar Code Reader, etc.*

sigtype (DataBase) `char(6) A6 %s` *Signal type, as in analog, digital, channel.*

sigid (DataBase) `char(4) A4 %s` *ID of the signal on the prints, as in 7, 2w, etc.*

acctype (DataBase) `char(6) A6 %s` *Type of access point for signal, e.g. cable, tblock, slot.*

accid (DataBase) `char(4) A4 %s` *Identifier of access point for signal, like J5 or P100.*

pinid (DataBase) `char(4) A4 %s` *ID of a pin on which a signal becomes accessible, as in 100, CC, etc.*

descrip (DataBase) `varchar(255) A255 %s` *Descrip is a generic text field in which one can add brief descriptive information to an database entity (record). The description is relevant to the table in which the field appears.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

DeviceID (Template) `int int int` *This is the internal Device ID number used in the Real Time code. Dean has established conventions for these values (they are not arbitrary, in other words) and for keyword message numbers derived from them.*

2.2.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

2.3 Database Schema for System Design

2.3.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.3.2 Regular Memes

MFlowData (MemeFlow) `schema schema` *The group of tables needed to describe meme flow: Agents, Memes, and the incidental format/media/timing information.* MFlowData consists of elements:

Agents (DataBase) `table table` (EXTNAME = 'Agents') *Table of software. hardware, and human agents.* Agents consists of elements:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

aname (DataBase) `varchar(20) A20 %s` *Human-readable name of agent*

atype (DataBase) `char(1) A1 %s` *single character agent type code*

author (DataBase) `varchar(40) A40 %s` *Various MemesWorld items can have authors. Software agents have authors. Meme contexts also (who invented the context?). Path contexts (drawings) also have authors. Although the product is different, the author field always indicates authorship.*

lang (DataBase) `char(10) A5 %s` *language in which agent was written if agent is a code entity*

docURI (DataBase) `varchar(255) A100 %s` *URI (currently a CVS reference) of documentation about an agent. At the moment this refers to a LaTeX file for inclusion into the CDR source, but this is an interim arrangement and will be improved later.*

rev (DataBase) `char(12) A10 %s` *Revision level of agent if agent is a code entity*

ref (DataBase) `char(12) A12 map` *Currently used for the Agent Context. The schema will be revised for consistent nomenclature in July 97.* Is an instance of acontext:

acontext (DataBase) `char(12) A12 %s` *Agent context, an agent attribute designed to group agents into functional sets. Some agents might be specific to one instrument, for example.*

rank (DataBase) `char(12) A12 %s` *This field may eventually be used for the forcible alignment of nodes in the digraph output by tracememe.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

owner (DataBase) `int I6 map` *The ID of a superagent which contains a bunch of paths or a bunch of other superagents* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

adesc (DataBase) `varchar(255) A %s` *A capsule description of hardware, software, or human agent. This field should really be called 'descrip' and will be renamed in July 97.*

srcURI (DataBase) `varchar(255) A100 %s` *A CVS reference to the source code for an agent. All DEIMOS source code is maintained under CVS so any agent written by our staff or CARA staff should have a CVS URI.*

rel (DataBase) `char(20) A20 %s` *The released revision of a software agent – as opposed to rev, which is the latest CVS revision checked in.*

Acontexts (MemeFlow) `table table` (EXTNAME = 'Acontexts') *Agents have context: an agent can be instrument specific, generic, etc.* Acontexts consists of elements:

acontext (DataBase) `char(12) A12 %s` *Agent context, an agent attribute designed to group agents into functional sets. Some agents might be specific to one instrument, for example.*

descrip (DataBase) `varchar(255) A255 %s` *Descrip is a generic text field in which one can add brief descriptive information to an database entity (record). The description is relevant to the table in which the field appears.*

AgentTypes (MemeFlow) `table table` (EXTNAME = 'AgentTypes') *Table of recognized types of agents* AgentTypes consists of elements:

atype (DataBase) `char(1) A1 %s` *single character agent type code*

atdesc (DataBase) `varchar(40) A40 %s` *agent type description*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Formats (MemeFlow) `table table` (EXTNAME = 'Formats') *Table of permissible formats for Mpaths* Formats consists of elements:

formid (DataBase) `char(3) A3 %s` *Format ID string; uniquely identifies an information format, such as ascii plaintext, html, latex source, etc.*

format (DataBase) `char(20) A20 %s` *Format description text. Meme bundles are expressed in some format during transmission.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Media (MemeFlow) `table table` (EXTNAME = 'Media') *Table of permissible media for Mpaths* Media consists of elements:

medid (DataBase) `char(3) A3 %s` *Medium ID, unique medium identifier for media such as filesystem file, raw disk IO, TCP, etc.*

medium (DataBase) `char(20) A20 %s` *Medium description*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Timing (MemeFlow) `table table` (EXTNAME = 'Timing') *Table of permissible timings for Mpaths.* Timing consists of elements:

timid (DataBase) `char(12) A2 %s` *Unique Timing ID, identifies a timing such as On Demand, Cron Job, etc.*

timing (DataBase) `char(20) A20 %s` *Text description of Timing*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Mpaths (MemeFlow) `table table` (EXTNAME = 'Mpaths') *Table of meme flow path segments: each segment represents the transit of a meme from one agent to another agent.* Mpaths consists of elements:

mpid (DataBase) `int I11 %d` *Unique ID of a Flow path segment*

ptype (DataBase) `char(1) A1 %s` *This one character code is a hack to force the Mpaths table to model both meme flow and development dependencies.*

mid (DataBase) `int I11 %d` *Unique ID of a meme. Many of the memes are FITS keywords, but others are FITS tables, meme bundles, etc.*

sendr (DataBase) `int I4 map` *ID of an agent which is the Sender in a Meme Path record.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

rcvr (DataBase) `int I4 map` *ID of an agent which is the Receiver in a Meme Path record.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

ctrlr (DataBase) `int I4 map` *ID of an agent which controls the Timing of a data transfer modelled in Meme Paths.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

medid (DataBase) `char(3) A3 %s` *Medium ID, unique medium identifier for media such as filesystem file, raw disk IO, TCP, etc.*

timid (DataBase) `char(12) A2 %s` *Unique Timing ID, identifies a timing such as On Demand, Cron Job, etc.*

formid (DataBase) `char(3) A3 %s` *Format ID string; uniquely identifies an information format, such as ascii plaintext, html, latex source, etc.*

elaps (DataBase) `int I11 %d` *[ms] The number of milliseconds required for a meme to traverse a path.*

pcontext (DataBase) `char(20) A20 %s` *The pcontext is a means of grouping meme flow paths into 'drawings', as for a partic-*

ular subsystem or functional area.

pcomment (DataBase) `varchar(255) A255 %s` *Commentary information about a meme flow path.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

owner (DataBase) `int I6 map` *The ID of a superagent which contains a bunch of paths or a bunch of other superagents* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

Pcontexts (MemeFlow) `table table` (EXTNAME = 'Pcontexts') *Paths have context, that is, they are grouped in drawings. If you use dot-sep context names like DEIMOS.DB.op.out you can make larger and smaller scale drawings of areas of the system.* Pcontexts consists of elements:

pcontext (DataBase) `char(20) A20 %s` *The pcontext is a means of grouping meme flow paths into 'drawings', as for a particular subsystem or functional area.*

title (DataBase) `char(40) A40 %s` *The user-defined title of a flow drawing (pcontext). Though the pcontext string is often meaningful, an English language title is also useful.*

author (DataBase) `varchar(40) A40 %s` *Various MemesWorld items can have authors. Software agents have authors. Meme contexts also (who invented the context?). Path contexts (drawings) also have authors. Although the product is different, the author field always indicates authorship.*

descrip (DataBase) `varchar(255) A255 %s` *Descrip is a generic text field in which one can add brief descriptive information to an database entity (record). The description is relevant to the table in which the field appears.*

2.3.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

2.4 Database Schema for Glass

2.4.1 Legend

MemeName (Context) RW `Sybase_Type` `F77_fmt` `C_fmt` *[Units]* *The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.4.2 Regular Memes

GlassData (DEIMOS.Glass) `schema` `schema` *This schema contains the physical glass component attributes for DEIMOS.* GlassData consists of elements:

SLAgratings (DEIMOS.glass) `table` `table` (EXTNAME = 'SLAgratings') *Sorry Steve! I didn't need to destroy them after all. Oops.* SLAgratings consists of elements:

filtid (DEIMOS.glass) `int` `I5` `%d` *[recid]* (0.0 to 999999.0) *Unique numeric ID of a record in the Filters table, hence unique ID of a filter*

gratid (DEIMOS.glass) `int` `I11` `%d` *Unique Id of a grating*

FilName (DEIMOS.glass) `varchar(20)` `A20` `%s` *Human-readable name of a filter*

GratingName (DEIMOS.glass) `varchar(20)` `A20` `%s` *Name of a grating.*

FilType (DEIMOS.glass) `char(2)` `A2` `%s` *Type of filter* Valid values of FilType are:

- LP : long pass (blue block)
- M : medium band
- N : narrow band
- SP : short pass (red block)
- W : wide band

GroovDens (DEIMOS.glass) `float` `F8.0` `%.0f` *[1/m]* *Density of grooves on a grating. A mirror has a groove density of zero.*

BlazeAng (DEIMOS.glass) `real` `F7.4` `%.4f` *[deg]* (0.0 to 90.0) *Blaze angle of a grating.*

RefCoat (DEIMOS.glass) `varchar(20) A20 %s` *Reflective coating on the surface of an optic. Possibly an index into a table of coating types and characteristics.*

TransCoat (DEIMOS.glass) `varchar(20) A20 %s` *Transmissive coating on the surface of an optic. Possibly an index into a table of coating types and characteristics.*

ScanURI (DEIMOS.glass) `varchar(255) A %s` *URI of a FITS file containing efficiency curve(s) for an optic. Transmission for filters, reflection for mirrors, efficiency for gratings, etc. Different orders and polarizations will all be in the same file, in different HDUs or table columns.*

Xdim (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *x dimension of an optic. For non-rectangular optics this is a characteristic value.*

Ydim (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *y dimension of an optic. For non-rectangular optics this is a characteristic value.*

Zdim (DEIMOS.glass) `real F8.6 %.6f` *[m]* (0.0 to 9.0) *z dimension (thickness) of an optic*

OffsetX (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *x offset of filter from optical axis*

OffsetY (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *y offset of filter from optical axis*

CoatingI (DEIMOS.glass) `varchar(20) A20 %s` *Coating on the incident side of a filter. Possibly an index into a table of coating types and characteristics*

CoatingE (DEIMOS.glass) `varchar(20) A20 %s` *Coating on the emergent face of a filter. Possibly an index into a table of coating types and characteristics.*

ScanURI (DEIMOS.glass) `varchar(255) A %s` *URI of a FITS file containing efficiency curve(s) for an optic. Transmission for filters, reflection for mirrors, efficiency for gratings, etc. Different orders and polarizations will all be in the same file, in different HDUs or table columns.*

CWave (DEIMOS.glass) `real F13.11 %.11f` *[m]* (0.0 to 1.0) *Characteristic wavelength for the filter. The middle of the band for pass and stop filters, and the location of the cutoff for blocking filters.*

XdimR (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *x dimension of the ruled portion of a grating.*

FWHM (DEIMOS.glass) `real F13.11 %.11f` *[m]* (0.0 to 1.0) *Full Width at Half Max for bandpass filters.*

YdimR (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *y dimension of the ruled portion of a grating.*

ZeroPoint (DEIMOS.glass) `real F7.3 %.3f` *A photometric zero point used in the conversion from DN through this filter to a standard magnitude system.*

Xdim (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *x dimension of an optic. For non-rectangular optics this is a characteristic value.*

ZeroURI (DEIMOS.glass) `varchar(255) A %s` *URI of a reference containing description of the calibrations used to calculate ZeroRef.*

Ydim (DEIMOS.glass) `real F9.6 %.6f` *[m]* (0.0 to 99.0) *y dimension of an optic. For non-rectangular optics this is a characteristic value.*

Zdim (DEIMOS.glass) `real F8.6 %.6f` *[m]* (0.0 to 9.0) *z dimension (thickness) of an optic*

FocalShift (DEIMOS.glass) `real F10.7 %.7f` *[m]* *A canonical value for the amount of camera refocus required when this filter is inserted.*

VendorId (DEIMOS.rdb) `int I11 map` *Unique ID of Vendor who supplied the item.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

VendorPart (DEIMOS.glass) `varchar(20) A20 %s` *Vendor part number for the item.*

RcvDate (DEIMOS.glass) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Date filter was received and stocked at observatory* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0)
The calendar year using the Gregorian calendar.

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

RcvDate (DEIMOS.glass) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
Date filter was received and stocked at observatory Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0)
The calendar year using the Gregorian calendar.

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

OwnerId (DEIMOS.glass) `int I11 map` *Unique ID of the observer who obtained the filter.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

VendorId (DEIMOS.rdb) `int I11 map` *Unique ID of Vendor who supplied the item.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

SLAmirrors (DEIMOS.glass) `table table` (EXTNAME = 'SLAmirrors')
Sorry Steve! I didn't need to destroy them after all. Oops.

Layers (DEIMOS.glass) `table table` (EXTNAME = 'Layers') *Table of Layers in filters* Layers consists of elements:

layid (DEIMOS.glass) `int I11 %d` (0.0 to 999999.0) *Unique numeric ID of a layer in a filter*

filtid (DEIMOS.glass) `int I5 %d` *[recid]* (0.0 to 999999.0) *Unique numeric ID of a record in the Filters table, hence unique ID of a filter*

LayNum (DEIMOS.glass) `smallint I5 %d` (0.0 to 100.0) *Number of layer in filter*

GlassType (DEIMOS.glass) `varchar(20) A %s` *Short code for a type of filter glass, possibly an index into a table of longer descriptions of filter glass types and characteristics*

Thickness (DEIMOS.glass) `real F8.6 %.6f` *[m]* (0.0 to 10.0)
spatial thickness of the filter (layer)

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

SLAmirrors (DEIMOS.glass) `table table` (EXTNAME = 'SLAmirrors')
Sorry Steve! I didn't need to destroy them after all. Oops. SLAmirrors consists of elements:

mirid (DEIMOS.glass) `int I11 %d` *Unique Id of a mirror*

MirName (DEIMOS.glass) `varchar(20) A20 %s` *Name of a mirror.*

RefCoat (DEIMOS.glass) `varchar(20) A20 %s` *Reflective coating on the surface of an optic. Possibly an index into a table of coating types and characteristics.*

TransCoat (DEIMOS.glass) `varchar(20) A20 %s` *Transmissive coating on the surface of an optic. Possibly an index into a table of coating types and characteristics.*

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

2.4.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

2.5 Database Schema for CCD Detectors

2.5.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.5.2 Regular Memes

ccdData (DEIMOS.ccd) `schema schema` *A bundle of tables which collectively express what we know about the CCDs used in DEIMOS (or other instruments).* ccdData consists of elements:

CCD (DEIMOS.ccd) `table table` (EXTNAME = 'CCD') *A table describing CCD detectors* CCD consists of elements:

ccdid (DataBase) `int I11 %d` *Unique id of a record in CCD table. Generated.*

name (DataBase) `varchar(30) A30 %s` *Name of a CCD (nomenclature to be cleaned up in July 97) The 'name' meme is almost a universal concept, but the char length needed to express it varies. This should perhaps be modelled with an ISA relationship (format conversion).*

VendorId (DEIMOS.rdb) `int I11 map` *Unique ID of Vendor who supplied the item.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

comments (DataBase) `varchar(255) A255 %s` *Generic comments field, like generic descrip field. These might really be the same thing. Alt use in Memes that are InfoTypes: output filename*

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Amplifier (DEIMOS.ccd) `table table` (EXTNAME = 'Amplifier') *A table describing amplifiers on CCDs.* Amplifier consists of elements:

- **ampid** (DataBase) `int I11 %d` *Unique ID of a record in Amplifiers. Generated.*
- **ccdid** (DataBase) `int I11 %d` *Unique id of a record in CCD table. Generated.*
- **gain** (DataBase) `float E20.12 %.12E` *A Gain setting for a CCD amplifier.*
- **noise** (DataBase) `float E20.12 %.12E` *A noise measurement for a CCD amplifier.*
- **isFunctional** (DataBase) `char(1) L1 %s` *A boolean flag indicating whether or not some hardware component is in working condition.*
- **posX1** (DataBase) `int I11 %d` *An X position in CCD pixels.*
- **posY1** (DataBase) `int I11 %d` *A Y position in CCD pixels.*

DefectType (DEIMOS.ccd) `table table` (EXTNAME = 'DefectType') *Table of standard defect types found on CCDs* DefectType consists of elements:

- **defectTypeid** (DataBase) `int I11 %d` *Primary key field of defectType table. Generated.*
- **defectType** (DataBase) `varchar(40) A40 %s` *Name of a specific type of CCD defect, as in 'Dead Pixel'.*

Defect (DataBase) `table table` (EXTNAME = 'Defect') *Table describing specific defects found on specific CCDs.* Defect consists of elements:

- **defectid** (DataBase) `int I11 %d` *Primary key field of Defects table. Generated.*
- **ampid** (DataBase) `int I11 %d` *Unique ID of a record in Amplifiers. Generated.*
- **ccdid** (DataBase) `int I11 %d` *Unique id of a record in CCD table. Generated.*
- **defectTypeid** (DataBase) `int I11 %d` *Primary key field of defectType table. Generated.*
- **pixX1Loc** (DataBase) `int I11 %d` *UNKNOWN*
- **pixY1Loc** (DataBase) `int I11 %d` *UNKNOWN*
- **pixX2Loc** (DataBase) `int I11 %d` *UNKNOWN*
- **pixY2Loc** (DataBase) `int I11 %d` *UNKNOWN*

Setup (DEIMOS.ccd) `table table` (EXTNAME = 'Setup') *Table describing a setup for a CCD test.* Setup consists of elements:

- **setupid** (DataBase) `int I11 %d` *The primary key field of the Setup table. Generated.*

ccdid (DataBase) `int I11 %d` *Unique id of a record in CCD table. Generated.*

dewarid (DataBase) `int I11 %d` *Primary key field of Dewar table. Generated.*

controllerid (DataBase) `int I11 %d` *Primary key field for Controller table. Generated.*

waveid (DataBase) `int I11 %d` *The primary key field for the Waveform table.*

temperature (DataBase) `float E20.12 %.12E` *The temperature used in a Setup for a CCD test.*

comments (DataBase) `varchar(255) A255 %s` *Generic comments field, like generic descrip field. These might really be the same thing. Alt use in Memes that are InfoTypes: output filename*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Waveform (DEIMOS.ccd) `table table` (EXTNAME = 'Waveform') *Table for describing waveforms used in Setups for testing CCDs.* Waveform consists of elements:

waveid (DataBase) `int I11 %d` *The primary key field for the Waveform table.*

filename (DataBase) `varchar(100) A100 %s` *Name of file in which a waveform description is found. This should really be a URI and possibly a CVS URI.*

comments (DataBase) `varchar(255) A255 %s` *Generic comments field, like generic descrip field. These might really be the same thing. Alt use in Memes that are InfoTypes: output filename*

ccdTest (DEIMOS.ccd) `table table` (EXTNAME = 'ccdTest') *A table describing the setup and results of a detector test.* ccdTest consists of elements:

testid (DataBase) `int I11 %d` *Primary key field of Test table. Generated.*

ccdid (DataBase) `int I11 %d` *Unique id of a record in CCD table. Generated.*

setupid (DataBase) `int I11 %d` *The primary key field of the Setup table. Generated.*

tdesc (DataBase) `varchar(100) A100 %s` *Description of a CCD test. This is obviously a very terse description, and a URI* *should point to more detailed material if needed.*

tresult (DataBase) `varchar(100) A100 %s` *The result of a CCD test.*

nid (DataBase) `int I11 %d` *Unique ID of a PostIt Note.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

Dewar (DEIMOS.ccd) `table table` (EXTNAME = 'Dewar') *Table describing a Dewar for use with CCDs.* Dewar consists of elements:

dewarid (DataBase) `int I11 %d` *Primary key field of Dewar table. Generated.*

dname (DataBase) `char(20) A20 %s` *Name of dewar. Like every other component of an observing system, dewars acquire nicknames like 'Blue Dewar 2'.*

nitrocap (DataBase) `float E20.12 %.12E` *UNKNOWN*

MFRDATE (DEIMOS.glass) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Date of manufacture of a filter* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

RcvDate (DEIMOS.glass) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
Date filter was received and stocked at observatory Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0)
The calendar year using the Gregorian calendar.

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

2.5.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

2.6 Database Schema for Mask Design

2.6.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.6.2 Regular Memes

DesignData (DEIMOS.mask) `schema schema` *A bundle of tables into which the mask acceptor inserts records when a new mask design file is received.* DesignData consists of elements:

Objects (DEIMOS.mask) `table table` (EXTNAME = 'Objects') *Defines the RDB table containing the lists of objects used in slitlets. (Compare with FITS table ObjectCat.)* Objects consists of elements:

ObjPKey (CATALOG) `int I6 %d` *A unique key value assigned to each object in the catalog table. In the context of a FITS HDU this has no permanent significance, but serves to permit cross indexing from other tables.*

DesId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask design.*

OBJECT (NOST1) rw `varchar(68) A %s` *The value field shall contain a character string giving the name of the object observed.*

RA_OBJ (HFWG_R3) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0) *Right Ascension of an object.*

DEC_OBJ (HFWG_R3) `float F11.7 %.7f` *[deg]* (-90.0 to 90.0) *Declination of an object.*

RADECSYS (WCS95) `char(8) A8 %-8s` *Specifies the frame of reference of the equatorial coordinates; default = 'FK4' for EQUINOX 1950.0 and 'FK5' for 2000.0. Other values are 'GAPPT' and 'FK4-NO-E'. (The map for this meme is shared by RADECPNT.)* Valid values of RADECSYS are:

- FK4 : 4th Fundamental Catalog
- FK4-NO-E : without E-terms

- FK5 : 5th Fundamental Catalog
- GAPPT : Geocentric Apparent

EQUINOX (NOST1) `float F13.6 %.6f` *[annum]* (1500.0 to 3000.0)
The value field shall contain a floating point number giving the equinox in years for the celestial coordinate system in which positions given in either the header or data are expressed.

MJD-OBS (WCS95) `float F17.9 %.9f` *[diem]* (0.0 to 100000.0)
The Modified Julian Date of the time of the observation measured in TAI. Note that this specification of the WCS draft precludes encoding of images obtained prior to the year 1955.

mag (CATALOG) `real F7.3 %.3f` (-99.0 to 99.0) *Magnitude of an object.*

pBand (DEIMOS.mask) `char(6) A6 %s` *photometric passband used to measure mag.*

RadVel (CATALOG) `float F14.3 %.3f` *[m/s]* (-299792458.0 to 299792458.0)
Radial Velocity of an object.

MajAxis (CATALOG) `real F11.3 %.3f` *[arcsec]* (0.0 to 648000.0)
Length of major axis of an extended object.

MaskDesign (DEIMOS.mask) `table table` (EXTNAME = 'MaskDesign')
Defines the RDB table containing the lists of mask designs. MaskDesign consists of elements:

DesId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask design.*

DesPId (DEIMOS.mask) `int I11 map` *Unique ID of a Person or Vendor who creates a slitmask design. This may differ from the subsequent blueprint creator who will own the slitmask.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

DesName (DEIMOS.mask) `varchar(68) A %s` *Observer's pet name for a slitmask design. This will typically have a date appended to it when the slitmask blueprint is created from the slitmask design.*

DesAgent (DEIMOS.mask) `int I11 map` *Unique ID of the agent used to create a slitmask design. This will be inferred from the FITS meme DesTool.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

DesDate (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
The FITS DATE of creation of the mask design HDU. It shall

be expressed using the new Y2K FITS date format. Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

CatAuth (DEIMOS.mask) `varchar(68) A %s` *FITS AUTHOR of a catalog.*

CatName (DEIMOS.mask) `varchar(68) A %s` *Observer's pet name for a catalog.*

CatDate (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Date of construction of FITS HDU containing a catalog.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0)
The calendar year using the Gregorian calendar.

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

RA_PNT (HFWG_R3) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0)
Right Ascension of the telescope pointing axis.

DEC_PNT (HFWG_R3) `float F11.7 %.7f` *[deg]* (-90.0 to 90.0)
Declination of the telescope pointing axis.

RADECPNT (SLA) `char(8) A8 map` *The RADECSYS in which the telescope pointing coordinates are expressed.* Is an instance of RADECSYS:

RADECSYS (WCS95) `char(8) A8 %-8s` *Specifies the frame of reference of the equatorial coordinates; default = 'FK4' for EQUINOX 1950.0 and 'FK5' for 2000.0. Other values are 'GAPPT' and 'FK4-NO-E'. (The map for this meme is shared by RADECPNT.)* Valid values of RADECSYS are:

- FK4 : 4th Fundamental Catalog
- FK4-NO-E : without E-terms
- FK5 : 5th Fundamental Catalog
- GAPPT : Geocentric Apparent

PA_PNT (DEIMOS.mask) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0)
The position angle of the DEIMOS mask on the sky.

EQUINPNT (SLA) `real F13.6 %.6f` *[annum]* (1855.0 to 2100.0)
The EQUINOX in which the telescope pointing axis coordinates are expressed.

MJD_PNT (SLA) `float F17.9 %.9f` *[diem]* (0.0 to 100000.0)
The MJD_WCS applying to the reference frame in which the telescope pointing axis coordinates are expressed.

LST_Des (DEIMOS.mask) `real F7.3 %.3f` *[deg]* *The anticipated Local Sidereal Time used during the design of the slitmask.*

DateDes (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
The anticipated UT date(/time) of the observation which was used during the design of the slitmask. It shall be expressed using the new Y2K FITS date format. Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0)
The calendar year using the Gregorian calendar.

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

DesiSlits (DEIMOS.mask) `table table` (EXTNAME = 'DesiSlits')
Defines the RDB and FITS tables containing the lists of slitlets in mask designs. DesiSlits consists of elements:

dSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask design. This will be converted into a dSlitId upon data ingest.*

DesId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask design.*

slitRA (DEIMOS.mask) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0)
Right Ascension of the center of the slitlet.

slitDec (DEIMOS.mask) `float F11.7 %.7f` *[deg]* (-90.0 to 90.0)
Declination of the center of the slitlet.

slitTyp (DEIMOS.mask) `char(1) A1 %s` *The type of the slitlet in a mask design (or FITS blueprint).* Valid values of slitTyp are:

- A : Alignment box (parallelogram)
- E : ellipse (circle)
- G : Guide (ghost)
- P : parallelogram
- S : Scratch (parallelogram)

slitLen (DEIMOS.mask) `real F11.3 %.3f` *[arcsec] Projected length of a slitlet on the sky.*

slitLPA (DEIMOS.mask) `real F8.3 %.3f` *[deg]* (-180.0 to 360.0)
Projected position angle of the length direction of a slitlet on the sky.

slitWid (DEIMOS.mask) `real F11.3 %.3f` *[arcsec] Projected width of a slitlet on the sky.*

slitWPA (DEIMOS.mask) `real F8.3 %.3f` *[deg]* (-180.0 to 360.0)
Projected position angle of the width direction of a slitlet on the sky.

SlitObjMap (DEIMOS.mask) `table table` (EXTNAME = 'SlitObjMap')
Defines the RDB and FITS tables containing the map of objects which fall into each slitlet. SlitObjMap consists of elements:

dSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask design. This will be converted into a dSlitId upon data ingest.*

ObjPKey (CATALOG) `int I6 %d` *A unique key value assigned to each object in the catalog table. In the context of a FITS HDU this has no permanent significance, but serves to permit cross indexing from other tables.*

MaskBlu (DEIMOS.mask) `table table` (EXTNAME = 'MaskBlu')
Defines the RDB table containing characteristics of a mask blueprint. MaskBlu consists of elements:

BluId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask blueprint.*

DesId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask design.*

BluPId (DEIMOS.mask) `int I11 map` *Unique ID of a Person or Vendor who creates a slitmask blueprint. This will designate the 'owner' of a milled slitmask.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

BluName (DEIMOS.mask) `varchar(68) A %s` *Observer's pet name for a slitmask blueprint. This will be visible in the GUI which permits selection of the active slitmask. Typically it will be a design name with an added date.*

BluAgent (DEIMOS.mask) `int I11 map` *Unique ID of the agent used to convert a slitmask design into a slitmask blueprint. This will be inferred from the FITS meme BluTool.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

BluDate (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *The FITS DATE of creation of the mask blueprint HDU. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

LST_Use (DEIMOS.mask) `real F7.3 %.3f` *[deg]* *The Local Sidereal Time of the intended use of the slitmask.*

DateUse (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *The UT date(/time)of the intended use of the slitmask. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0)
The calendar month of the year.

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0)
The calendar date of the month.

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0)
The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

TeleId (DEIMOS.mask) `int I11 %d` *Unique ID of a telescope.*

RefrId (DEIMOS.mask) `int I11 %d` *Unique ID of an algorithm for atmospheric refraction.*

AtmTempC (ATMOSPHERE) `real F5.1 %.1f` *[degC]* (-100.0 to 50.0)
The ambient atmospheric temperature at the telescope in Celsius.

AtmPres (ATMOSPHERE) `real F5.1 %.1f` *[mbar]* (0.0 to 1100.0)
The atmospheric pressure at the telescope. This should be the actual pressure, not the pressure reduced to sea level.

AtmHumid (ATMOSPHERE) `real F5.3 %.3f` (0.0 to 1.2000000000000002)
The ambient relative humidity at the telescope (as a fraction).

AtmTTLap (ATMOSPHERE) `real F7.5 %.5f` *[K/m]* *The adiabatic temperature lapse rate in the troposphere above the observatory.*

RefWave (REFRACTION) `real F12.10 %.10f` *[m]* (0.0 to 1000.0)
The atmospheric refraction model is applied for a specific observation wavelength. The value should be within, and p ossibly towards the blue end of, the desired spectral bandpass.

DistId (DEIMOS.mask) `int I11 %d` *Unique ID of an algorithm for image distortion at the DEIMOS slitmask.*

BluSlits (DEIMOS.mask) `table table` (EXTNAME = 'BluSlits')
Defines the RDB and FITS tables containing characteristics of slitlets in a mask blueprint. BluSlits consists of elements:

bSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask blueprint. This will be converted into a bSlitId upon data ingest.*

BluId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask blueprint.*

dSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask design. This will be converted into a dSlitId upon data ingest.*

slitX1 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint X coordinate of perimeter point 1.*

slitY1 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint Y coordinate of perimeter point 1.*

slitX2 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint X coordinate of perimeter point 2.*

slitY2 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint Y coordinate of perimeter point 2.*

slitX3 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint X coordinate of perimeter point 3.*

slitY3 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint Y coordinate of perimeter point 3.*

slitX4 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint X coordinate of perimeter point 4.*

slitY4 (DEIMOS.mask) `float F9.6 %.6f` *[m]* *The slitmask blueprint Y coordinate of perimeter point 4.*

ExtendObj (DEIMOS.mask) `table table` (EXTNAME = 'ExtendObj') *Defines the RDB table containing the lists of extended objects.* ExtendObj consists of elements:

ObjPKey (CATALOG) `int I6 %d` *A unique key value assigned to each object in the catalog table. In the context of a FITS HDU this has no permanent significance, but serves to permit cross indexing from other tables.*

MajAxPA (CATALOG) `real F7.3 %.3f` *[deg]* (-180.0 to 360.0) *Position angle of Major axis of an extended object in the RA/Dec system measured from North towards East.*

MinAxis (CATALOG) `real F11.3 %.3f` *[arcsec]* (0.0 to 648000.0) *Length of minor axis of an extended object.*

NearObj (DEIMOS.mask) `table table` (EXTNAME = 'NearObj') *Defines the RDB table containing the lists of nearby objects.* NearObj consists of elements:

ObjPKey (CATALOG) `int I6 %d` *A unique key value assigned to each object in the catalog table. In the context of a FITS HDU this has no permanent significance, but serves to permit cross indexing from other tables.*

PM_RA (CATALOG) `real F9.4 %.4f` *[arcsec/annum]* *Proper motion of an object in right ascension. The annum is Besselian if RADECSYS is FK4 and Julian if RADECSYS is FK5.*

PM_Dec (CATALOG) `real F9.4 %.4f` *[arcsec/annum]* *Proper motion of an object in declination. The annum is Besselian if RADECSYS is FK4 and Julian if RADECSYS is FK5.*

Parallax (CATALOG) `real F7.4 %.4f` *[arcsec]* (0.0 to 10.0) *(Trigonometric) Parallax of an object*

2.6.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

Identity (Map) `map map` *A map meme which indicates that its meme has all the characteristics (min, max, legal values, ...) of another meme. In actuality this is probably an abuse of our current understanding of memes schema.*

2.7 Database Schema for Mask Fabrication

2.7.1 Legend

MemeName (Context) RW `Sybase_Type` `F77_fmt` `C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.7.2 Regular Memes

MillData (DEIMOS.mask) `schema` `schema` *A bundle of tables requested by the mill control program when it prepares to fabricate a slitmask.* MillData consists of elements:

MaskBlu (DEIMOS.mask) `table` `table` (EXTNAME = 'MaskBlu') *Defines the RDB table containing characteristics of a mask blueprint.* MaskBlu consists of elements:

BluId (DEIMOS.mask) `int` `I11` `%d` *Unique ID of a slitmask blueprint.*

DesId (DEIMOS.mask) `int` `I11` `%d` *Unique ID of a slitmask design.*

BluPId (DEIMOS.mask) `int` `I11` `map` *Unique ID of a Person or Vendor who creates a slitmask blueprint. This will designate the 'owner' of a milled slitmask.* Is an instance of pid:

pid (DataBase) `int` `I11` `%d` *Unique ID of a Person or Vendor*

BluName (DEIMOS.mask) `varchar(68)` `A` `%s` *Observer's pet name for a slitmask blueprint. This will be visible in the GUI which permits selection of the active slitmask. Typically it will be a design name with an added date.*

BluAgent (DEIMOS.mask) `int` `I11` `map` *Unique ID of the agent used to convert a slitmask design into a slitmask blueprint. This will be inferred from the FITS meme BluTool.* Is an instance of agentid:

agentid (DataBase) `int` `I11` `%d` *Unique Agent ID number*

BluDate (DEIMOS.mask) `smalldatetime` `A19` `%04d-%02d-%02dT%02d:%02d:%02d` *The FITS DATE of creation of the mask blueprint HDU. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

LST_Use (DEIMOS.mask) `real F7.3 %.3f` *[deg] The Local Sidereal Time of the intended use of the slitmask.*

DateUse (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *The UT date(/time)of the intended use of the slitmask. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the* *new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

TeleId (DEIMOS.mask) `int I11 %d` *Unique ID of a telescope.*

RefrId (DEIMOS.mask) `int I11 %d` *Unique ID of an algorithm for atmospheric refraction.*

AtmTempC (ATMOSPHERE) `real F5.1 %.1f` *[degC]* (-100.0 to 50.0) *The ambient atmospheric temperature at the telescope in Celsius.*

AtmPres (ATMOSPHERE) `real F5.1 %.1f` *[mbar]* (0.0 to 1100.0) *The atmospheric pressure at the telescope. This should be the actual pressure, not the pressure reduced to sea level.*

AtmHumid (ATMOSPHERE) `real F5.3 %.3f` (0.0 to 1.2000000000000002) *The ambient relative humidity at the telescope (as a fraction).*

AtmTTLap (ATMOSPHERE) `real F7.5 %.5f` *[K/m]* *The adiabatic temperature lapse rate in the troposphere above the observatory.*

RefWave (REFRACTION) `real F12.10 %.10f` *[m]* (0.0 to 1000.0) *The atmospheric refraction model is applied for a specific observation wavelength. The value should be within, and p ossibly towards the blue end of, the desired spectral bandpass.*

DistId (DEIMOS.mask) `int I11 %d` *Unique ID of an algorithm for image distortion at the DEIMOS slitmask.*

BluSlits (DEIMOS.mask) `table table` (EXTNAME = 'BluSlits')
Defines the RDB and FITS tables containing characteristics of slitlets in a mask blueprint. BluSlits consists of elements:

bSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask blueprint. This will be converted into a bSlitId upon data ingest.*

BluId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask blueprint.*

dSlitPK (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask design. This will be converted into a dSlitId upon data ingest.*

slitX1 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint X coordinate of perimeter point 1.*

slitY1 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint Y coordinate of perimeter point 1.*

slitX2 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint X coordinate of perimeter point 2.*

slitY2 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint Y coordinate of perimeter point 2.*

slitX3 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint X coordinate of perimeter point 3.*

slitY3 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint Y coordinate of perimeter point 3.*

slitX4 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint X coordinate of perimeter point 4.*

slitY4 (DEIMOS.mask) `float F9.6 %.6f` *[m] The slitmask blueprint Y coordinate of perimeter point 4.*

MaskDesign (DEIMOS.mask) `table table` (EXTNAME = 'MaskDesign')
Defines the RDB table containing the lists of mask designs. MaskDesign consists of elements:

DesId (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask design.*

DesPId (DEIMOS.mask) `int I11 map` *Unique ID of a Person or Vendor who creates a slitmask design. This may differ from the subsequent blueprint creator who will own the slitmask.* Is an instance of pid:

pid (DataBase) `int I11 %d` *Unique ID of a Person or Vendor*

DesName (DEIMOS.mask) `varchar(68) A %s` *Observer's pet name for a slitmask design. This will typically have a date* *appended to it when the slitmask blueprint is created from the slitmask design.*

DesAgent (DEIMOS.mask) `int I11 map` *Unique ID of the agent used to create a slitmask design. This will be inferred from the FITS meme DesTool.* Is an instance of agentid:

agentid (DataBase) `int I11 %d` *Unique Agent ID number*

DesDate (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *The FITS DATE of creation of the mask design HDU. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

CatAuth (DEIMOS.mask) `varchar(68) A %s` *FITS AUTHOR of a catalog.*

CatName (DEIMOS.mask) `varchar(68) A %s` *Observer's pet name for a catalog.*

CatDate (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *Date of construction of FITS HDU containing a catalog.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

- **GregYMD** (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:
  - **GregYear** (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*
  - **CalenMonth** (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*
  - **CalenDate** (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*
- **HMSp** (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:
  - **SexHourP** (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*
  - **SexMinute** (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*
  - **SexSecond** (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

RA_PNT (HFWG_R3) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0) *Right Ascension of the telescope pointing axis.*

DEC_PNT (HFWG_R3) `float F11.7 %.7f` *[deg]* (-90.0 to 90.0) *Declination of the telescope pointing axis.*

RADECPNT (SLA) `char(8) A8 map` *The RADECSYS in which the telescope pointing coordinates are expressed.* Is an instance of RADECSYS:

RADECSYS (WCS95) `char(8) A8 %-8s` *Specifies the frame of reference of the equatorial coordinates; default = 'FK4' for EQUINOX 1950.0 and 'FK5' for 2000.0. Other values are 'GAPPT' and 'FK4-NO-E'. (The map for this meme is shared by RADECPNT.)* Valid values of RADECSYS are:

- FK4 : 4th Fundamental Catalog
- FK4-NO-E : without E-terms
- FK5 : 5th Fundamental Catalog
- GAPPT : Geocentric Apparent

PA_PNT (DEIMOS.mask) `float F12.7 %.7f` *[deg]* (-180.0 to 360.0) *The position angle of the DEIMOS mask on the sky.*

EQUINPNT (SLA) `real F13.6 %.6f` *[annum]* (1855.0 to 2100.0) *The EQUINOX in which the telescope pointing axis coordinates are expressed.*

MJD_PNT (SLA) `float F17.9 %.9f` *[diem]* (0.0 to 100000.0) *The MJD_WCS applying to the reference frame in which the telescope pointing axis coordinates are expressed.*

LST_Des (DEIMOS.mask) `real F7.3 %.3f` *[deg]* *The anticipated Local Sidereal Time used during the design of the slitmask.*

DateDes (DEIMOS.mask) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *The anticipated UT date(/time) of the observation which was used during the design of the slitmask. It shall be expressed using the new Y2K FITS date format.* Is really a:

ISOdatetime (MapTuples) `tuple tuple` *A structure containing all the elements of date and time of day for an ISO-like format – 'yyyy-mm-ddThh:mm:ss'.* ISOdatetime consists of elements:

GregYMD (MapTuples) `tuple tuple` *Bundle describing a Gregorian calendar date in an ISO-style ordering where the year is fully expressed. It can be represented using the new FITS date format – 'yyyy-mm-dd'.* GregYMD consists of elements:

GregYear (DEMO) `smallint I4 %04d` *[annum]* (1582.0 to 9999.0) *The calendar year using the Gregorian calendar.*

CalenMonth (DEMO) `tinyint I2.2 %02d` *[month]* (0.0 to 13.0) *The calendar month of the year.*

CalenDate (DEMO) `tinyint I2.2 %02d` *[diem]* (0.0 to 32.0) *The calendar date of the month.*

HMSp (MapTuples) `tuple tuple` *A positive sexagesimal hour, minute, second structure.* HMSp consists of elements:

SexHourP (TYPEDEFS) `smallint I2 %2d` *[hour]* (0.0 to 24.0) *The hour portion of a positive sexagesimal string – 'HH:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d` *date on which record was entered (or updated) in database*

2.7.3 Map Definitions

ForeignKey (MapTypes) `map map` *A map meme which indicates that its meme can be used as a foreign key into another table. The foreign occurrence of the meme is indicated by the isa_mid.*

Identity (Map) `map map` *A map meme which indicates that its meme has all the characteristics (min, max, legal values, ...) of another meme. In actuality this is probably an abuse of our current understanding of memes schema.*

2.8 Database Schema for Operation

2.8.1 Legend

MemeName (Context) RW `Sybase_Type F77_fmt C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.8.2 Regular Memes

OperData (Operation) `schema schema` *Collection of tables used only during actual instrument operation: logging and alarms.* OperData consists of elements:

Alarms (Operation) `table table` (EXTNAME = 'Alarms') *Table intended for the configuration of the watcher process.* Alarms consists of elements:

alid (DataBase) `int I11 %d` *Unique Id number of an Alarm record. Generated.*

severity (DataBase) `char(1) A1 %s` *A VMS-like severity level code (I, W, E, F) which determines the means of delivery, measures taken to get the user's attention, etc. when a watcher process delivers a message.*

condition (DataBase) `varchar(255) A255 %s` *Boolean condition expression to be evaluated by watcher processes for the delivery of alarms, warnings, etc.*

message (DataBase) `varchar(255) A255 %s` *Message text to be delivered by a watcher process if the associated condition is true.*

uri (DataBase) `varchar(255) A255 %s` *If the keyword cannot be documented adequately via semantics and comments, a URI can be attached referring the reader to other online documentation, source code, etc.*

handler (DataBase) `varchar(255) A255 %s` *Tcl code to be executed by a watcher process if the associated condition is true.*

uid (DataBase) `smallint I5 %d` *(Sybase) ID code of database Key Entry person. This tells us who entered the record in the table. The stamp field tells us when.*

stamp (DataBase) `smalldatetime A19 %04d-%02d-%02dT%02d:%02d:%02d`
date on which record was entered (or updated) in database

ParaLog (Operation) `table table` (EXTNAME = 'ParaLog') *Incomplete cardboard version of a parametric log table. A logger process would write one record to this table every N seconds perpetually. Each field is a KTL readable keyword. This mockup is made of a few keywords that looked interesting.* ParaLog consists of elements:

LogStamp (Operation) `datetime A23 %04d-%02d-%02dT%02d:%02d:%06.3f`
This is a date/time stamp from a logger process. It has to be good to a second at worst.

DWRN2LV (KECK1CCD) r `real F12.8 %s` *[percent full]* (0.0 to 100.0)
Liquid nitrogen level in dewar. Uses map DWRN2LVM to convert or format DWRN2LVR:

DWRN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
Dewar liquid nitrogen level expressed as a 'raw' value from an analog-to-digital converter.

PWRBLOK (KECK1CCD) rw `real F12.8 %s` *[V]* (0.0 to 10.0)
Power to heater block in the dewar. Uses map PWRBLOKM to convert or format PWRBLOKR:

PWRBLOKR (KECK1CCD) rw `int I11 %d` *[ADC value]* (0.0 to 4095.0)
Power to heater block in the dewar expressed as a 'raw' value from an analog-to-digital converter.

RESN2LV (KECK1CCD) r `real F12.8 %s` *[percent full]* (0.0 to 100.0)
Liquid nitrgen level in the reservoir source used to refill the dewar. Uses map RESN2LVM to convert or format RESN2LVR:

RESN2LVR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
Liquid nitrogen level of the reservoir source used to refill the dewar expressed as a 'raw' value from an analog-to-digital converter.

TEMPDET (KECK1CCD) r `real F13.8 %s` *[degC]* (-220.0 to 25.0)
Detector temperature. Uses map TEMPDETM to convert or format TEMPDETR:

TEMPDETR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
Detector temperature expressed as a 'raw' value from an analog-to-digital converter.

UTBM15V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5)
-15 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET! Uses map UTBM15VM to convert or format UTBM15VR:

UTBM15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
-15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.

UTBP15V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5)
+15 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET! Uses map UTBP15VM to convert or format UTBP15VR:

UTBP15VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
+15 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.

UTBP30V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5)
+30 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET! Uses map UTBP30VM to convert or format UTBP30VR:

UTBP30VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
+30 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.

UTBP5V (KECK1CCD) r `real E13.6 %s` *[ADC value / 10]* (0.0 to 409.5)
+5 volt power supply voltage level. THIS VALUE SHOULD BE A VOLTAGE, BUT IT HAS NOT BEEN IMPLIMENTED YET! Uses map UTBP5VM to convert or format UTBP5VR:

UTBP5VR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
+5 volt power supply voltage level expressed as a 'raw' value from an analog-to-digital converter.

UTBTEMP (KECK1CCD) r `real F12.8 %s` *[degC]* (-250.0 to 50.0)
Temperature of the CCD controller utility board. Uses map UTBTEMPM to convert or format UTBTEMPR:

UTBTEMPR (KECK1CCD) r `int I11 %d` *[ADC value]* (0.0 to 4095.0)
Temperature of the CCD controller utility board expressed as a 'raw' value from an analog-to-digital converter.

AIRMASS (KECK1DCS) r `float F12.8 %.8f` (1.0 to 100.0)
air mass (secant of elevation)

AZ (KECK1DCS) r `float F13.8 float` *[deg]* (-180.0 to 360.0)
Telescope azimuth Uses map AngleAz to convert or format DMS:

DMS (MapTuples) `tuple tuple` *A sexagesimal degree, arcminute, arcsecond structure.* DMS consists of elements:

SexDegree (TYPEDEFS) `smallint I4 %4d` *[deg]* (-180.0 to 360.0)
The degree portion of a sexagesimal string – '=DDD:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

DEC (KECK1DCS) r `char(11) A11 float` *[degree (sexagesimal)]*
Current declination measured in the coordinate frame specified by FRAME (FK4, FK5 or apparent) and at the epoch of mean equinox and equator specified by EQUINOX. Uses map AngleDec to convert or format DMS90:

DMS90 (MapTuples) `tuple tuple` *A sexagesimal degree, arcminute, arcsecond structure limited to +/-90 degrees; thus suitable for declinations.* DMS90 consists of elements:

SexDeg90 (TYPEDEFS) `smallint I3 %3d` *[deg]* (-90.0 to 90.0)
The degree portion of a sexagesimal string describing a declination – '=DD:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

EL (KECK1DCS) r `float F11.8 float` *[deg]* (0.0 to 90.0) *Telescope elevation angle* Uses map AngleDec to convert or format DMS90:

DMS90 (MapTuples) `tuple tuple` *A sexagesimal degree, arcminute, arcsecond structure limited to +/-90 degrees; thus suitable for declinations.* DMS90 consists of elements:

SexDeg90 (TYPEDEFS) `smallint I3 %3d` *[deg]* (-90.0 to 90.0)
The degree portion of a sexagesimal string describing a declination – '=DD:MM:SS.sss'.

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0)
The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0)
The second portion of a sexagesimal string – 'HH:MM:SS.sss'.

PARANG (KECK1DCS) r `float F13.8 float` *[deg]* (-180.0 to 360.0)
The angle between vertical and north as measured on the sky where the telescope is pointing. The use of 'PARANG' for this quantity is a misnomer. This should really be VERTANG. Uses map AngleAz to convert or format DMS:

DMS (MapTuples) `tuple tuple` *A sexagesimal degree, arcminute, arcsecond structure.* DMS consists of elements:

SexDegree (TYPEDEFS) `smallint I4 %4d` *[deg]* (-180.0 to 360.0) *The degree portion of a sexagesimal string – '=DDD:MM:SS.sss'.*

SexMinute (TYPEDEFS) `smallint I2.2 %02d` *[minute]* (0.0 to 59.0) *The minute portion of a sexagesimal string – 'HH:MM:SS.sss'.*

SexSecond (TYPEDEFS) `real F7.4 %07.4f` *[second]* (0.0 to 60.0) *The second portion of a sexagesimal string – 'HH:MM:SS.sss'.*

TUBETEMP (KECK1DCS) r `float F13.8 %.8f` *[degC]* (-100.0 to 50.0) *Current tube temperature (a weighted mean of various temperatures around the telescope structure).*

BARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Instrument (barrel) electronics enclosure temperature in raw A/D converter counts.*

BARGLRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Coolant (glycol) temperature in raw A/D converter counts for the instrument (barrel) electronics.*

CARENRAW (DEIMOS2) rw `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Carriage electronics enclosure temperature in raw A/D converter counts.*

DWFOCFOP (pfcam+deimos) rw `char(3) A %d` (0.0 to 1.0) *Dewar focus fiducial light source control.* Uses map KTL-boolean to convert or format off_on:

off_on (SharedMemeMap) rw `tinyint L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

CARXTRAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *External (dome) temperature in raw A/D converter counts for the carriage electronics cooling system.*

POWH0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the carriage in raw A/D converter counts.*

POWH2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *High voltage (+-15 volts) logic power supply voltage for the instrument in raw A/D converter counts.*

POWL0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0) *Low voltage (+5 volts) logic power supply voltage for the car-*

riage in raw A/D converter counts.

POWL2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0)
Low voltage (+5 volts) logic power supply voltage for the instrument in raw A/D converter counts.

POWM0RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0)
Carriage motor power supply (motor power supply #0) voltage in ADC counts.

POWM1RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0)
Instrument motor power supply #1 voltage in ADC counts.

POWM2RAW (DEIMOS2) r `int I11 %d` *[12 bit adc value]* (0.0 to 4095.0)
Instrument motor power supply #2 voltage in ADC counts.

ROTATRAW (DEIMOS2) rw `int I11 %d` *[encoder counts]* (0.0 to 10000.0)
Instrument rotator raw position.

2.8.3 Map Definitions

AngleAz (MapUnits) `float float float` *[rad]* (-1.0 to 2.0) *An angle in radians with azimuth-like limits (or angles with unclear limits). (minv, maxv, &c. are expressed in units of pi when unit is radians.) This map indicates that the external formatted meme has an internal 64-bit IEEE binary representation.*

AngleDec (MapUnits) `float float float` *[rad]* (-0.5 to 0.5) *An angle in radians with declination-like limits. (minv, maxv, &c. are expressed in units of pi when unit is radians.) This map indicates that the external formatted meme has an internal 64-bit IEEE binary representation.*

DWRN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar liquid nitrogen level value (see DWRN2LVR) into a percentage full (see DWRN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

KTLboolean (MapTypes) `tinyint I1 %d` (0.0 to 1.0) *A special case of the KTL Enumerated map, (q.v.). ktlBoolean has two legal int values, 0 and 1. When setting 't', 'true', or a nonzero int =¿1; and 'f' or 'false' =¿ 0. FIORD keywords report and accept string values such as 'open', 'closed', etc.*

PWRBLOKM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar heater power block value (see PWRBLOKR) into a voltage (see PWRBLOK).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

RESN2LVM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw dewar refill reservoir level (see RESN2LVR) into a percentage full (see RESN2LV).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

TEMPDETM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a raw detector temperature (see TEMPDETR) into degrees celsius (see TEMPDET).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

UTBM15VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' -15 volt power supply voltage value from an analog-to-digital converter (see UTBM15VR) to a voltage (see UTBM15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

UTBP15VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +15 volt power supply voltage value from an analog-to-digital converter (see UTBP15VR) to a voltage (see UTBP15V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

UTBP30VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +30 volt power supply voltage value from an analog-to-digital converter (see UTBP30VR) to a voltage (see UTBP30V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to* *human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

UTBP5VM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert a 'raw' +5 volt power supply voltage value from an analog-to-digital converter (see UTBP5VR) to a voltage (see UTBP5V). NOT IMPLIMENTED FOR VOLTAGE YET!* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

pre_bias (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*

scale (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*

post_bias (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*

minb (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*

maxb (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

UTBTEMPM (KECK1CCD) rw `varchar(68) A %s` *An array of linear unit conversion constants used to convert the raw CCD controller utility board temperature from an analog-to-digital converter value (see UTBTEMPR) to a value in degrees celsius (see UTBTEMP).* Is really a:

isaMAP (TYPEDEFS) `tuple tuple` *The Keck Instrument Control System uses maps to describe conversions from device encoder units to human-comprehensible units. These consist of arrays of 5 32-bit floating point numbers.* isaMAP consists of elements:

- **pre_bias** (MapParams) `real E13.6 %E` *The pre-bias parameter in a mapping from device units to user units.*
- **scale** (MapParams) `real E13.6 %E` *The scale parameter in a mapping from device units to user units.*
- **post_bias** (MapParams) `real E13.6 %E` *The post-bias parameter in a mapping from device units to user units.*
- **minb** (MapParams) `real E13.6 %E` *The lower_bound parameter in a mapping from device units to user units.*
- **maxb** (MapParams) `real E13.6 %E` *The upper_bound parameter in a mapping from device units to user units.*

2.9 Database Schema for Calibration/Pipeline

2.9.1 Legend

MemeName (Context) RW `Sybase_Type` `F77_fmt` `C_fmt` *[Units] The attributes displayed here if available are: the Meme name, context, access (read/write/both), datatype, FORTRAN format, C format, and the value's units and legitimate range (if any). A table may also have a FITS EXTNAME. This text is the Semantics.*

2.9.2 Regular Memes

CalibData (DEIMOS.Calib) `schema` `schema` *This bundle doesn't exist yet. It's a place holder for the table definitions for Calibration data archive.* CalibData consists of elements:

Lambdas (DataBase) `table` `table` (EXTNAME = 'Lambdas') *Table of standard lambdas for pipeline reduction and calibration.* Lambdas consists of elements:

- **source** (DataBase) `char(20)` `A20` `%s` *Source of a standard calibration lambda. Could be the name of a lamp, or NightSky, or....*
- **lambda** (DataBase) `float` `E20.12` `%.12E` *Lambda (wavelength) of known feature in known source.*
- **flux** (DataBase) `float` `E20.12` `%.12E` *Flux of known lambda.*
- **accuracy** (DataBase) `float` `E20.12` `%.12E` *[lambda] An errorbar for standard lambdas.*
- **correction** (DataBase) `float` `E20.12` `%.12E` *If an empirical correction of the standard (catalog) lambda is required, here is a place for it.*

Spectra (DataBase) `table` `table` (EXTNAME = 'Spectra') *Table of standard spectra for use in calibration and pipeline reduction.* Spectra consists of elements:

- **spectrid** (DataBase) `int` `I11` `%d` *Unique record ID of Spectra table.*
- **specURI** (DataBase) `varchar(255)` `A100` `%s` *URL of Spectrum (FITS file).*
- **gratid** (DEIMOS.glass) `int` `I11` `%d` *Unique Id of a grating*
- **gratilt** (DataBase) `float` `E20.12` `%.12E` *Grating Tilt, which must be remembered when interpreting standard spectra.*

- **source** (DataBase) `char(20) A20 %s` *Source of a standard calibration lambda. Could be the name of a lamp, or NightSky, or....*
- **ELAPTIME** (KECK1CCD) r `int I11 %d` *[s]* (0.0 to 10000.0) *Elapsed time of the exposure.*
- **slitwidth** (DataBase) `float E20.12 %.12E` *width of slit used to take standard spectrum.*
- **MaskId** (DEIMOS.mask) `int I11 %d` *Unique ID of a slitmask. Corresponds directly to the barcode label on the mask.*
- **bSlitPK** (DEIMOS.mask) `int I6 %d` *The primary key into the table of slitlets in a mask blueprint. This will be converted into a bSlitId upon data ingest.*

Coeff1 (DataBase) `table table` (EXTNAME = 'Coeff1') *Table of coefficients for pipeline reduction. This table can be used for interpolation. The Coeff2 table can be used for solution.* Coeff1 consists of elements:

- **cf1id** (DataBase) `int I11 %d` *Unique Record ID of table Coeff1.*
- **gratid** (DEIMOS.glass) `int I11 %d` *Unique Id of a grating*
- **gratilt** (DataBase) `float E20.12 %.12E` *Grating Tilt, which must be remembered when interpreting standard spectra.*
- **coeffname** (DataBase) `char(2) A2 %s` *Coefficient name. Refer to S. Faber's notes. This would be one of p, q, r, s, t..*
- **maskX** (DataBase) `float E20.12 %.12E` *X position on slitmask (seems like this s/b an ISA).*
- **maskY** (DataBase) `float E20.12 %.12E` *Y position on slitmask (seems like this s/b an ISA).*
- **value** (DataBase) `float E20.12 %.12E` *Value of the coefficient coeffname.*
- **deltaDay** (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after Phase II Day calibration.*
- **deltaNight** (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after Phase II Night calibration.*
- **deltaFinal** (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after final calibration.*

Coeff2 (DataBase) `table table` (EXTNAME = 'Coeff2') *Final (general) coefficient table – you solve for coeff1 by knowing values for a, b, c, tilt, etc. You have to read S. Faber's description of this process to understand what this table is doing.* Coeff2 consists of elements:

cf2id (DataBase) `int I11 %d` *Unique Record ID for Coeff2 table.*

gratid (DEIMOS.glass) `int I11 %d` *Unique Id of a grating*

coeffname (DataBase) `char(2) A2 %s` *Coefficient name. Refer to S. Faber's notes. This would be one of p, q, r, s, t..*

term (DataBase) `char(2) A2 %s` *One of tilt, X, Y. See S. Faber's notes on pipeline reduction, or this will make no sense at all.*

coef2name (DataBase) `char(2) A2 %s` *One of a, b, c, d, e.*

value (DataBase) `float E20.12 %.12E` *Value of the coefficient coeffname.*

deltaDay (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after Phase II Day calibration.*

deltaNight (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after Phase II Night calibration.*

deltaFinal (DataBase) `float E20.12 %.12E` *Correction applied to 'value' after final calibration.*