MULLARD SPACE SCIENCE LABORATORY
UNIVERSITY COLLEGE LONDON
Authors: H.E.Huckle, C.Ho, S.Horner, J. Klarkowski, A. Welty

XMM-OM ICU-DPU Protocol Definitions

Document: XMM-OM/MSSL/SP/0014.07

Distribution:

XMM-OM Project Office	R Hunt	Orig
ESA PX	H Eggel	✓
CSL	P Rochus	
	S Roose	
Los Alamos National Observatory	C.Ho	✓
UCSB	T.Sasseen	✓
Royal Greenwich Observatory	R Bingham	
University College London	J Fordham	
Mullard Space Science Laboratory	R Card	
	M Carter	
	R Chaudery	✓
	A Dibbens	
	H Huckle	✓
	H Kawakami	
	T Kennedy	✓
	D Self	
	P Smith	✓
	P Thomas	
	M Whillock	✓
	K Mason	
	A Smith	

Author: Date: 6 Mar ‘98

OM Project Office Date:

Distributed: Date:

CHANGE RECORD

Issue	Date	Comments
1	9 Sep ‘93	Feedback from EOB1 Integration
2	21 Apr ‘94	Preparation for EOB2 Integration
3	24 Mar ‘95	Conversion to Word Format, Updates and Revisions
3onc	22 May ‘96-Jan ‘97	Major Revisions for DPU EQM model, issued for comment
4	26-Feb-97	Issued - format changed to letter for US printing
5	7 Aug 97	‘Snapshot’ of current status for EM Delivery
6	22-Sep-97	Update for EM User Manual Release
6onc		Corrected DA_BEGOF_EXP, DA_ENDOF_EXP Correct DP_FAQ Modified IC_SET_FRAME_TIME Removed IC_GEN_COMMAND_ID Added IC_POWER_DOWN_DPU Corrected list of Prom supported commands Tidied layout in places Corrected IC_SYNCH_CLK Removed IC_SEL_SCI_LNK_TST, IC_ENBL_RO(A)M_TST, DR_RA(O)_TST Added fuller description of Engineering Data. Corrected IC_DUMP_PROG_RAM length Added IC_SET_TRK and IC_BPE_BINNING Corrected various omissions Removed TBC’s.
7	6 Mar ‘98	Formal Release Tracking Flag added to status

TABLE OF CONTENTS

1. PURPOSE OF DOCUMENT

This document describes the data/command records to be passed between the XMM-OM DPU and ICU. The records are given in the form of pseudo 'C'-language data structures.

2. STANDARD DEFINITIONS

2.1 Typedefs For Basic Types (Machine Dependent)

```
typedef   unsigned short int  INT16;    /* 16 bit integer */
typedef   signed int          DPUINT;   /* 24 bit DPU words padded to 32 bits */
```

2.2 Command Mnemonics

Each mnemonic is headed by a two letter prefix, e.g. `IC`. The first letter indicates the origin of the data, `I` =I CU, `D` = DPU, the second letter indicates the nature of the data or the mechanism of data transmission, i.e. `C = COMMAND, D = DATA, A = ALERT, P = PRIORITY, R = RAM.`

The following standard abbreviations are used in constructing the mnemonics.

BEGOF	BEGINNING_OF
BFAST	BLUE_FAST_MODE_AREA
CID	COMMAND_ID
CLK	CLOCK
CMD	COMMAND
CMPRS	COMPRESSED DATA
CONF	CONFIGURE or CONFIGURATION
DIAGS	DIAGNOSTICS
DET	DETECTOR
DLVR	DELIVER
DPU	DIGITAL_PROCESSING_UNIT
DPUOS	DIGITAL_PROCESSING_UNIT_OPERATING_SYSTEM
DQ	DATA_QUEUE
DSP	DIGITAL_SIGNAL_PROCESSOR
ENBL	ENABLE
ENG	ENGINEERING
EOS	END_OF_SWAP_UNIT
EOT	END_OF_TASK
EVENTS	EVENTS_PROCESSING
EXP	EXPOSURE
EXTRCT	EXTRACT
E_ALERT	ERROR_ALERT
FAQ	FIELD_ACQUISITION
FILT	FILTER
FLD	FIELD (absolute pointing)
FST	FAST
GADE	GLOBAL_ADDRESS_ERROR
GEN	GENERATE
GS	GUIDE_STARS
HBEAT	HEART_BEAT
ICU	INSTRUMENT_CONTROL_UNIT
IMG	IMAGE
INIT	INITIALIZE
LNK	LINK
LRM	LOCAL_RAM
MEM	MEMORY (as in windows)
NORM	NORMALIZATION
N	AND
PROG	PROGRAM
RCVR	RECEIVER
REF	REFERENCE
REQ	REQUEST
SCI	SERIAL_COMMUNICATION_INTERFACE
SCI_WDW	SCIENCE_WINDOW
SEL	SELECT
S_ALERT	STATUS_ALERT
TRH	TRACKING_HISTORY
TRK	TRACKING
TST	TEST
WDW	WINDOW

The mnemonic values are listed below in numerical order.

2.2.1 DPU priority data type DP - 0XA2yy

```
#define    DP_WDW              0XA201
#define    DP_FAQ              0XA202
#define    DP_BFAST            0XA203

Place holders - reserved

#define    DP_EE_DUMP          0XA206
```

2.2.2 DPU regular data type DD - 0XA3yy

```
#define    DD_TRH              0XA301
#define    DD_REF              0XA302
#define    DD_IMG              0XA303
#define    DD_FST              0XA304
#define    DD_LOG              0XA305
#define    DD_ENG              0XA306

Place Holders - reserved

#define    DD_RAW              0XA307
```

2.2.3 ICU to DPU command IC - 0XA4yy

2.2.3.1 Basic DPUOS manipulation and IO commands

```
#define    IC_RESET_DSP        0XA401
#define    IC_LOAD_DPUOS       0XA402
#define    IC_ENBL_LOAD_CODE   0XA403
#define    IC_SEL_RAM_BANK     0XA404
#define    IC_LOAD_CODE        0XA405
#define    IC_SEL_KLINGON      0XA406
#define    IC_ENBL_DSP         0XA407
#define    IC_ENBL_EVENTS      0XA408
#define    IC_DIRECT_DATA      0XA409
#define    IC_POWER_DOWN_DPU   0XA40A
#define    IC_SEND_SCI_CMD     0XA40B
#define    IC_REQ_DATA         0XA40D
#define    IC_SYNCH_CLK        0XA40E
#define    IC_FLUSH_QUEUE      0XA40F
```

2.2.3.2 Memory management

```
#define    IC_DUMP_LOCAL_RAM    0XA410
#define    IC_LOAD_LOCAL_RAM    0XA411
#define    IC_DUMP_PROG_RAM     0XA414
#define    IC_DUMP_RAM          0XA415
#define    IC_DUMP_RAM_N_ZERO   0XA416
#define    IC_LOAD_GLOBAL_RAM   0XA417
```

2.2.3.3 Exposure configuration commands

```
#define    IC_SET_TRK           0xA420
#define    IC_BPE_BINNING       0xA421
#define    IC_REPORT_TRK        0XA422
#define    IC_REPORT_DIAGS      0XA423
#define    IC_SET_FRAME_TIME    0XA424
#define    IC_SET_EXP_TIME      0XA425
#define    IC_SET_EXP_ID        0XA426
#define    IC_CONF_GS_SEL       0XA427
#define    IC_LOAD_REF_STARS    0XA428
#define    IC_LOAD_FILT_CONF    0XA429
#define    IC_LOAD_MEM_WDW      0XA42A
#define    IC_LOAD_SCI_WDW      0XA42B
#define    IC_ENBL_VERBOSE      0XA42C
#define    IC_LOAD_DET_WDW      0xA42D
```

2.2.3.4 Major task invocation commands

```
#define    IC_INIT_DPU          0XA430
#define    IC_INIT_EXP          0XA431
#define    IC_ACQUIRE_FLD       0XA432
#define    IC_CHOOSE_GS         0XA433
#define    IC_TRACK_GS          0XA434
#define    IC_ABORT_DPU         0XA435
#define    IC_ENBL_ENG          0XA436
#define    IC_FLUSH_CMPRS       0XA437
```

2.2.4 DPU Data Alert data type DA - 0XA5yy

2.2.4.1 Synchronous Alert

```
#define    DA_HBEAT             0XA500
```

2.2.4.2 Asynchronous health status alert

```
Must send alerts

#define    DA_DPU_BOOT_READY    0XA510
#define    DA_DPUOS_READY       0XA511
#define    DA_ECHO_ICU_CMD      0XA512
#define    DA_BEGOF_EXP         0XA513
#define    DA_ENDOF_EXP         0XA514
#define    DA_DATA_ALERT        0XA515
#define    DA_DATA_END          0XA516
#define    DA_DONE_FLD_DATA     0XA517
#define    DA_COMPLETE_EXP      0XA518
```

2.2.4.3 Major milestone alerts

```
#define    DA_EOT_INIT_DPU      0XA520
#define    DA_EOT_ACQUIRE_FLD   0XA521
#define    DA_EOT_CHOOSE_GS     0XA522
#define    DA_EOT_INIT_EXP      0XA523
#define    DA_EOT_ENG           0XA524
#define    DA_SWITCH_FRAME      0XA525
#define    DA_TRK               0XA526
#define    DA_SENT_CMPRSSQ      0XA527
#define    DA_EOT_FLUSH_CMPRS   0XA528
```

2.2.4.4 Minor milestone alerts, may be turned off

```
#define    DA_EOS_INIT_DPU      0XA530
#define    DA_EOS_CALBGD        0XA531
#define    DA_EOS_SCANBS        0XA532
#define    DA_EOS_RSTR_STARS    0XA533
#define    DA_EOS_MAKEAGS_ID    0XA534
#define    DA_EOS_CALDRFT       0XA535
#define    DA_EOS_SCANGS        0XA536
#define    DA_EOS_DOWINDOW      0XA537
#define    DA_EOS_GETREF        0XA538
#define    DA_EOS_LOCATEGS      0XA539
#define    DA_EOS_ABORT         0XA53A
#define    DA_EOS_DLVRDATA      0XA53B
#define    DA_EOS_ENG           0XA53C
#define    DA_EOS_COMPRESS      0XA53D
#define    DA_DONE_SAA          0XA53E
```

2.2.4.5 Asynchronous exception alerts

```
#define    DA_CLK_SYNCH_ERROR   0XA580
#define    DA_PROG_RAM_ERROR    0XA581
#define    DA_BOTTOM_WHITEDSP   0XA582
#define    DA_CID_TOO_LARGE     0XA583
#define    DA_BAD_CMD           0XA584
#define    DA_DPU_ABORTING      0XA585
#define    DA_BAD_BLOCK         0XA586
#define    DA_ZERO_LENGTH_DQ    0XA587
```

2.2.4.6 Exception alerts from coloured DSPs

```
#define    DA_SCI_RCVR_ERROR    0XA590
#define    DA_SCI_HBEAT_ERROR   0XA591
#define    DA_RED_E_ALERT       0xA592
#define    DA_RED_S_ALERT       0xA593
#define    DA_BLUE1_E_ALERT     0xA594
#define    DA_BLUE1_S_ALERT     0xA595
#define    DA_BLUE2_E_ALERT     0xA596
#define    DA_BLUE2_S_ALERT     0xA597
```

2.2.4.7 Global memory access exception alerts

```
#define    DA_GADE_RPI          0XA5A0
#define    DA_GADE_RPIA_L       0XA5A1
#define    DA_GADE_RPIA_H       0XA5A2
#define    DA_GADE_WPI          0XA5A3
#define    DA_GADE_WPIA_L       0XA5A4
#define    DA_GADE_WPIA_H       0XA5A5
#define    DA_GADE_RBI          0XA5A6
#define    DA_GADE_RBIA_L       0XA5A7
#define    DA_GADE_RBIA_H       0XA5A8
#define    DA_GADE_WBI          0XA5A9
#define    DA_GADE_WBIA_L       0XA5AA
#define    DA_GADE_WBIA_H       0XA5AB
#define    DA_GADE_RSI          0XA5AC
#define    DA_GADE_RSIA_L       0XA5AD
#define    DA_GADE_RSIA_H       0XA5AE
#define    DA_GADE_WSI          0XA5AF
#define    DA_GADE_WSIA_L       0XA5B0
#define    DA_GADE_WSIA_H       0XA5B1
```

2.2.5 DPU RAM DUMP data type DR/0XA6yy

```
#define    DR_RAM_DUMP          0XA600
#define    DR_RAM_DUMP_N_ZERO   0XA603
#define    DR_PROG_DUMP         0XA604
#define    DR_LRM               0XA605
```

2.3 Command Recognition

The DPU has 2 states of operation - Boot Idle and DPUOS Idle. The DPU will recognise different commands in these two states as follows:

Command	Boot Idle	DPUOS Idle
IC_DUMP_PROG_RAM	Yes	-
IC_ENBL_LOAD_CODE	Yes	-
IC_LOAD_CODE	Yes	-
IC_LOAD_DPUOS	Yes	-
IC_POWER_DOWN_DPU	Yes	-
IC_SEL_RAM_BANK	Yes	-
IC_DUMP_LOCAL_RAM	Yes	Yes
IC_DUMP_RAM	Yes	Yes
IC_FLUSH_QUEUE	Yes	Yes
IC_LOAD_LOCAL_RAM	Yes	Yes
IC_REQ_DATA	Yes	Yes
IC_RESET_DSP	Yes	Yes
All Others	-	Yes

2.4 Mnemonic Changes

Between the EOB and EQM models, a revision of the mnemonics took place in order to make them more self consistent. The following are 'conversion tables' from the old to the new mnemonics.

OLD ALERT MNEMONIC	**NEW ALERT MNEMONIC**
DA_BAD_COMMAND	DA_BAD_CMD
DA_BEGOF_EXPOSURE	DA_BEGOF_EXP
DA_CLOCK_SYNCH_ERROR	DA_CLK_SYNCH_ERROR
DA_DONE_ABORT	DA_EOS_ABORT
DA_DONE_CALBGD	DA_EOS_CALBGD
DA_DONE_CALDRFT	DA_EOS_CALDRFT
DA_DONE_COMPRESSION	DA_EOS_COMPRESS
DA_DONE_DELIVERDATA	DA_EOS_DLVRDATA
DA_DONE_DOWINDOW	DA_EOS_DOWINDOW
DA_DONE_ENGINEERING	DA_EOS_ENG
DA_DONE_GETREF	DA_EOS_GETREF
DA_DONE_INITIALIZE	DA_EOS_INIT_DPU
DA_DONE_LOCATEGS	DA_EOS_LOCATEGS
DA_DONE_MAKEGS_ID	DA_EOS_MAKEAGS_ID
DA_DONE_RESTORE_STARS	DA_EOS_RSTR_STARS
DA_DONE_SCANBS	DA_EOS_SCANBS
DA_DONE_SCANGS	DA_EOS_SCANGS
DA_ECHO_ICU_COMMAND	DA_ECHO_ICU_CMD
DA_ENDOF_CHOOSE_GS	DA_EOT_CHOOSE_GS
DA_ENDOF_ENGINEERING	DA_EOT_ENG
DA_ENDOF_EXPOSURE	DA_ENDOF_EXP
DA_ENDOF_FIELD_ACQ	DA_EOT_ACQUIRE_FLD
DA_ENDOF_FIELD_DATA	DA_DONE_FLD_DATA
DA_ENDOF_INIT_EXP	DA_EOT_INIT_EXP
DA_ENDOF_INIT	DA_EOT_INIT_DPU
DA_HEARTBEAT	DA_HBEAT
DA_RETURN_FROM_WHITE_DSP	DA_BOTTOM_WHITEDSP
DA_SCI_HEART_BEAT_ERROR	DA_SCI_HBEAT_ERROR
DA_SCI_RECEIVER_ERROR	DA_SCI_RCVR_ERROR

OLD COMMAND MNEMONIC	NEW COMMAND MNEMONIC
IC_ACQUIRE_FIELD	IC_ACQUIRE_FLD
IC_CHOOSE_GUIDE_STARS	IC_CHOOSE_GS
IC_ENBL_BRIGHT_STAR_MODE	IC_ENBL_BRT_S_MODE
IC_ENBL_L_CODE	IC_ENBL_LOAD_CODE
IC_ENBL_THRESHOLD_MODE	IC_ENBL_THLD_MODE
IC_GEN_BAD_CID	IC_GEN_BAD_CMD_ID
IC_INITIALIZE	IC_INIT_DPU
IC_LOAD_MEM_WIND	IC_LOAD_MEM_WDW
IC_LOAD_SCI_WIND	IC_LOAD_SCI_WDW
IC_REPORT_TRACKING	IC_REPORT_TRK
IC_SELECT_RAM_BANK	IC_SEL_RAM_BANK
IC_SEND_SCI_COMMAND	IC_SEND_SCI_CMD
IC_SET_EXPOSURE_ID	IC_SET_EXP_ID
IC_SET_EXPOSURE_TIME	IC_SET_EXP_TIME
IC_TRACK_GUIDE_STARS	IC_TRACK_GS

3. GENERAL POINTS

3.1 Bit Transmission Order

In all cases, the most significant bit is transmitted first. The first line item of a structure is transmitted first.

3.2 Block Structure

For both ICU to DPU and DPU to ICU records:-

1. the first word of the record defines the block type

2. the contents of second word of the record define the word count i.e. the number of SUBSEQUENT 16 bit words in the block e.g. a single 32-bit word in a data record adds 2 to the word count.

3. (optional) the associated data, the length, in words, having been defined above by the word count.

4. REGULAR DPU TO ICU DATA BLOCKS (DD_xxx)

4.1 Overview Of Data Structures

4.1.1 DPU Internal Format

Regular science data is stored in the DPU in the form of several large ***DPU production data sets*** which may be many kilobytes in length. Each production data set consists of two parts, a **Standard Data Set Header** prefixing a **Data Body**.

The **Standard Data Set Header** has the following structure

```
DPUINT  exposure;        /* Exposure ID */
DPUINT  data_type;       /* e.g. DD_IMG */
DPUINT  data_set_id;     /* The data set ID, i.e. the science window ID */
DPUINT  data_set_param;  /* for DD_ENG = engineering mode sub-type */
                         /* for DD_TRH = frame_count, else 0 */
DPUINT  in_length;       /* Number of DPUINT's of uncompressed data */
DPUINT  out_length;      /* Number of 16-bit words of compressed data */
```

The **Data Body** may, or may not, be compressed. Compression will be (in the future releases) commandable. It is proposed that a `out_length` of `0` indicates that the data body is not compressed and a negative value indicates it was compressed then decompressed.

If it is compressed, the data body will be a concatenation of VBTWL compressed variable length blocks, each with its own self-describing 128 bit long VBTWL header. This header is currently defined as follows:

Item	Size (bits)	Description
blockid	16	Block ID within that particular window.
n_bits	16	Number of compressed bits, including the header.
n_tier	3	Number of tiers, up to 3, currently 3
im	3	Predictor scheme, 3 possible, may be reduced.
blocksize	10	Size of block (e.g. number of pixels (max 1023, typically 256).
it1	6	Number of bits for tier 1.
it2	5	Number of bits for tier 2.
it3	5	Number of bits for tier 3.
nlx2	8	Number of data in tier 2.
nlx3	8	Number of data in tier 3.
sync_word	16	VBTWL sync word = 0xbeef.
predictor	32	Value of predictor.

Notes:

1. `nlx1 = blocksize - nlx2 - nlx3.`
2. `n_bits = nheader (=128) + blocksize*it1 + nlx2*it2 + nlx3*it3.`
3. It is therefore a 16 byte header. For a typical blocksize of 256 pixels and a compression ratio of 3, we have a compressed data set of 2048 bits with a 128 bit header overhead.

4.1.2 Format of Data Transmitted to ICU

At the end of an exposure, each production data set is supplied to the ICU in the form of a series of ***blocks*** of data. These blocks are of type DD_xxx. They are obtained by chopping the above **DPU production data set** into **data sections** and prefixing each one with an **'Extended DPU Header'.** Each block is subject to the constraint that the **Extended DPU Header** plus **Data Section** does not exceed 1029 16-bit words overall in length. **Note** that it is possible for the 'chopping' action to cut through the middle of a DPUINT item.

The **Extended DPU Header** is defined as follows:

```
INT16   type;           /* e.g. DD_IMG */
INT16   n;              /* word count of number of INT16's to follows */
INT16   sync;           /* sync word = 0xf352 */
DPUINT  address;        /* address in Global RAM data originated */
```

NOTE. The original *uncompressed or decompressed* DPU **Data Body**, NOT the DPU data blocks, is described in the remainder of this section.

4.2 DD_ENG (A306)

Engineering Mode data format is a function of its sub-type. The sub-type (which is given in the associated Standard Data Set Header) is defined as follows:-

SUB-TYPE	DESCRIPTION	
0	Raw data	with both blue DSPs
1		with only blue DSP 1
2		with only blue DSP 2
3	Channel Boundaries	
4	Full frame, high resolution	
5	Centroiding confirmation data	
6	Intensifier characteristics	
7	M,N Images	

The following sections define these different formats.

4.2.1 Raw Data (Sub-Types 0,1 and 2)

This data consists of the raw 24 bit event data (i.e. not an accumulated image) from the detector as received by one or both of the blue DSPs. They are padded to 32 bits. It is **not** compressed.

```
typedef struct
  {
  INT16 raw_event[n_events] ; /* 24 bit events padded to 32 bits   */
  }
```

4.2.2 Channel Boundary Data (Sub-Type 3)

This data consists of the channel boundaries scaled by a thousand. It is **not** compressed. The boundaries are originally in the range -1.0 to +1.0 and thus the data supplied to the ICU is in the range -1000 to 1000. They are derived from the data given in sub-type 7.

```
typedef struct
  {
  INT16 x_channel_bounds[9]; /* Channel Bounds * 1000 converted to integers */
  INT16 y_channel_bounds[9]; /* Channel Bounds * 1000 converted to integers */
  }
```

4.2.3 Full Frame, High Resolution (Sub-Type 4)

This data consists of the **compressed** 2048*2048 pixels of a full frame, high resolution image. The data is in column (y axis) order.

```
typedef struct
  {
  DPUINT pixcnts[2048,2048]; /* Pixel Counts for a 2048 * 2048 image */
  }
```

4.2.4 Centroiding Confirmation Data (Sub-Type 5)

This data consists of the **compressed** centroiding confirmation data. The full 2048 * 2048 field is divided into 8 * 8 sub-images, each of 256 * 256 centroided (by 8) pixels. These sub-images are then modulo 8 binned to produce a set of 8 * 8 pixels pseudo images. The images are column (y axis) ordered, as are the pixels within them.

```
typedef struct
  {
  DPUINT pixcnts;   /* Pixel counts for 8 * 8 modulo 8 folded images */
  }
```

4.2.5 Intensifier Characteristics (Sub-Type 6)

This data consists of the **compressed** Event Height histogram generated as a 1*256 pseudo image when the MIC detector is in Engineering modes 6 or 7 (**note** these should not be confused with the **DPU** engineering modes specified by IC_ ENBL_ENG).

```
typedef struct
  {
  DPUINT counts[256];     /* Event Height Histogram */
  }
```

4.2.6 M,N Images (Sub-Type 7)

This data consists of the **compressed** 2* 256 * 256 m,n pseudo-images generated when the MIC detector is in Engineering modes 4 or 5 (**note** these should not be confused with the **DPU** engineering modes specified by IC_ ENBL_ENG). The channel boundaries (see sub-type 3) are derived from these data.

```
typedef struct
  {
  DPUINT x_m_n_image[256,256];  /* M,N image - x axis */
  DPUINT y_m_n_image[256,256];  /* M,N image - y axis */
  }
```

4.3 DD_FST (A304)

Fast Mode Data.

```
typedef struct
        {
        DPUINT    fast_events[n_fast_events];
        } FST;
```

4.4 DD_IMG (A303)

Image Mode Data..

```
typedef struct
        {
        DPUINT         pixcts[npixels];   /* counts in individual pixels */
        } IMG;
```

Each record contains image data in COLUMN order.

4.5 DD_RAW (A307)

Not Yet Implemented.

4.6 DD_REF (A302)

Reference Frame Data..

If 'Verbose' mode is disabled:

```
typedef struct
    {
    DPUINT          exposure;           /* index to OM exposure */
    DPUINT          roll_ctr_x;         /* roll centre x co-ordinate */
    DPUINT          roll_ctr_y;         /* roll centre y co-ordinate */
    DPUINT          stars_found;        /* number of good guide stars found */
    DPUINT          stars_rejected;     /* number of bad guide stars found */
    } REF;
```

If 'Verbose' mode is enabled, the following structure is required:

```
typedef struct
    {
    STAR            star;
    int             peak_coords;        /* place holder, not implemented */
    int             peak_counts;        /* place holder, not implemented */
    int             star_quality;
    } STAR_DATA;


typedef struct
    {
    DPUINT          exposure;           /* index to OM exposure */
    DPUINT          roll_ctr_x;         /* roll centre x co-ord */
    DPUINT          roll_ctr_y;         /* roll centre y co-ord */
    DPUINT          stars_found;        /* no good guide stars found */
    DPUINT          stars_rejected;     /* no bad guide stars found */
    STARS_DATA      ggs[nggs];          /* data on good guide stars */
    STARS_DATA      bgs[nbgs];          /* data on bad guide stars */
    } REF;
```

NOTE see the description of DD_TRH for a definition of the structure `STAR`

4.7 DD_TRH (A301)

Tracking History (note - this is enabled/disabled by IC_REPORT_TRK)..

```
typedef struct
  {
  DPUINT  x;                      /* x co-ordinate (centroided pixels) */
  DPUINT  y;                      /* y co-ordinate (centroided pixels) */
  } POS;

typedef struct
  {
  POS     ctr;                    /* centroid co-ordinates */
  DPUINT  cts;                    /* counts integrated over PSF */
  } STAR;

typedef struct

  DPUINT  nggs[nframes];          /* number of good guide stars found
  DPUINT  dx[nframes];            /* calc x drift (1/1000 centroided pixel) */
  DPUINT  dy[nframes];            /* calc y drift (1/1000 centroided pixel) */
  DPUINT  roll[nframes];          /* calc roll drift (1,000,000*sin(theta)) */
  DPUINT  quality[nframes];       /* quad sum of x and y residuals */
  STAR    gs[nframes,stars_found]; /* guide star array */
  } TRH;
```

where:

1. the structure STAR is defined as above
2. stars_found is the (constant) number of tracking guide stars set-up at the beginning of the exposure.

5. DPU PRIORITY DATA (DP_xxx)

5.1 Overview

Priority data blocks contain science data that is sent out as soon as it is available, unlike regular blocks (DD_xxx) which are sent out at the end of an exposure.

5.2 DP_BFAST (A203)

Priority Fast Data.

```
typedef struct
  {
  INT16   type;                     /* type = DP_BFAST */
  INT16   n;   /* word count */
  INT16   sync;                     /* sync word  = 0xF352*/
  DPUINT  exposure_id;              /* exposure ID */
  DPUINT  bfast_id;                 /* either 1 or 2, mapping to blue DSP 1 or 2 */
  DPUINT  sample_time;              /* units of BPE frames */
  DPUINT  n_fast_pixels;            /* number of pixels per fast time slice */
  DPUINT  fast_destination;         /* memory address of fast mode data stream */
  DPUINT  fast_end;                 /* termination address of fast mode data */
  DPUINT  n_fast_events;            /* total number of events in fast mode data */
  DPUINT  fast_address[(n-15)/2];   /* Array size = min(area_size,503)*/
  }   BFAST;
```

5.3 DP_FAQ (A202)

Field Acquisition Data.

The record structure is as follows

```
typedef struct
      {
 INT16    type;           /* type = DP_FAQ */
 INT16    n;              /* word count */
 INT16    sync;           /* sync word = 0xF352*/
 DPUINT   exposure_id;    /* exposure ID */
 DPUINT   iterations;     /* number of iterations to solution */
 DPUINT   xcommand;       /* commanded x co-ordinate (centroided pixels) */
 DPUINT   ycommand;       /* commanded y co-ordinate (centroided pixels) */
 DPUINT   dx;             /* calc x pointing error (1/1000 centroided pixel)*/
 DPUINT   dy;             /* calc y pointing error (1/1000 centroided pixel) */
 DPUINT   sintheta;       /* sin of calculated absolute roll error (1/1000) */
 DPUINT   ngsup;          /* number of uplinked guide stars */
 DPUINT   ngsfound;       /* number of guide stars found */
 POS      gs[16];         /* data on individual guide stars */
 POS      ref_stars[16];  /* positions of reference stars */
      } FAQ;
```

where the structure `POS` defined as above

5.4 DP_WDW (A201)

Window Parameter Data.

The following data structures are required by this data record

```
typedef struct
     {
     DPUINT        x0;        /* bottom left x co-ordinate of window */
     DPUINT        y0;        /* bottom left y co-ordinate of window */
     DPUINT        xsize;     /* x dimension of window (centroided pixels) */
     DPUINT        ysize;     /* y dimension of window (centroided pixels) */
     DPUINT        mmwid;     /* id of corresponding memory wndw */
     DPUINT        kompx;     /* MIC data decoding parameter */
     DPUINT        bb;        /* MIC data decoding parameter */
     DPUINT        aa;        /* MIC data decoding parameter */
     DPUINT        kompy;     /* MIC data decoding parameter */
     } DET_WIND;


typedef struct
     {
     DPUINT        x0;        /* bottom left x co-ordinate of window */
     DPUINT        y0;        /* bottom left y co-ordinate of window */
     DPUINT        xsize;     /* x dimension of window (centroided pixels) */
     DPUINT        ysize;     /* y dimension of window (centroided pixels) */
     DPUINT        mode;      /* operation mode (e.g., Fast) */
     DPUINT        p1;        /* mode-dependent parameter - see notes */
     DPUINT        p2;        /* mode-dependent parameter - see notes */
     DPUINT        p3;        /* mode-dependent parameter - see notes */
     DPUINT        p4;        /* mode-dependent parameter - see notes */
     DPUINT        base;      /* data storage address in DPU RAM */
     DPUINT        mmwid;     /* id of corresponding memory wndw */
     } SCI_WIND;


typedef struct
     {
     DPUINT        x0;        /* bottom left x co-ordinate of window */
     DPUINT        y0;        /* bottom left y co-ordinate of window */
     DPUINT        xsize;     /* x dimension of window (centroided pixels) */
     DPUINT        ysize;     /* y dimension of window (centroided pixels) */
     DPUINT        base;      /* data retrieval address in DPU RAM */
     } MEM_WIND;
```

The record structure is as follows

```
typedef struct
   {
   INT16        type;         /* type = DP_WDW */
   INT16        n;            /* word count */
   INT16        sync;         /* sync word = 0xF352*/
   DPUINT       exposure;     /* exposure ID */
   DPUINT       ndet;         /* number of detector windows */
   DPUINT       nsci;         /* number of science windows */
   DPUINT       nmem;         /* number of memory windows */
   DET_WINDOW   det[16];      /* detector window array */
   SCI_WINDOW   sci[16];      /* science window array */
   MEM_WINDOW   memory[16];   /* memory window array */
   } WDW;
```

where `DET_WIND, SCI_WIND`, and `MEM_WIND` are each structures, defined as above

Notes:

Mode has the following values

Mode	Description
0	Image Mode Window
1	Fast Mode Window
15	Tracking Window

If mode = 0 (Image Mode):

a) p1 is the pixel binning factor performed by the DPU in x (pixels are binned by 2**p1, thus p1 means no binning, p1 = 1 means bin 2 pixels together in x, etc.)

b) p2 is the pixel binning factor in y (same comment as for p1)

c) p3 is not used

d) p4 is the starting address in DPU global memory to which the accumulating image will be written.

If mode = 1 (Fast Mode):

A) p1 = sampling time. If p1 = 0, the user requested fast mode window cannot be allocated.

B) P2 = 1 or 2 and is the id of the fast_mode_area associated with this window.

If the mode is anything else, then p1, p2, and p3 are not used and p4 serves the same purpose as that given above, namely that it is the starting address in DPU global memory to which the accumulating data are written.

6. DPU RAM DUMPS (DR_xxx)

6.1 Overview Of Ram Dump Data Structures

Ram Dump data is stored in the DPU in the form of several large ***DPU Data Sets*** which may be many kilobytes in length. Each data set is supplied to the ICU in the form of a series of ***blocks*** of data. These blocks are of type DR_xxx. They are obtained by chopping a **DPU data set** into **Data Sections** and prefixing each one with an **'Extended DPU Header'** Each block is subject to the constraint that the **Extended DPU Header** plus **Data Section** does not exceed 1029 16 bit words overall in length. **Note** that it is possible for the 'chopping' action to cut through the middle of a DPUINT item.

The **Extended DPU Header** is defined as follows:

```
INT16   type;            /* e.g. DR_RAM_DUMP */
INT16   n;               /* word count of number of INT16's to follows */
INT16   sync;            /* sync word = 0xf352 */
DPUINT  address;         /* address in Global RAM data originated */
```

The **Data Body**, NOT the DPU data blocks, is described below.

6.2 DR_LRM (A605)

Response to IC_DUMP_LOCAL_RAM. Dump of `length` 24 bit words from local ram.

```
typedef struct
     {
     DPUINT  DSP;                /* DSP */
     DPUINT  local_address;      /* different from the address in the header */
     DPUINT  length;             /* Number 24 bit words */
     DPUINT  value[length];      /* Local RAM Dump (padded to 32 bits)*/
     } LRM;
```

6.3 DR_PROG_DUMP (A604)

```
typedef struct
    {
    DPUINT words[in_length/2];   /* program ram dump (n = word count in header)
*/
    } PROG_RAM;
```

6.4 DR_RAM_DUMP (A600)

Ram Dump (response to IC_DUMP_RAM command).

```
typedef struct
          {
          DPUINT     words[in_length/2]; /* ram dump (n = word count in header) */
          } RAM_DUMP;
```

6.5 DR_RAM_DUMP_N_ZERO (A603)

Ram Dump and zero (response to IC_DUMP_RAM_N_ZERO command).

```
typedef struct
         {
         DPUINT     words[in_length/2]; /* ram dump (n = word count in header) */
         } RAM_DUMP_N_ZERO;
```

7. DPU TO ICU ALERTS (DA_XXX)

7.1 Overview

All alerts (unless stated otherwise) are of the form

```
{
INT16       Type;           /* E.g. DA_BAD_BLOCK */
INT16       n;              /* No of 16 bit words to follow >=4*/
INT16       message[n-4];   /* The alert message - may be of zero length */
DPUINT      exposure;       /* Exposure Number */
DPUINT      timestamp;      /* Time stamp */
}
```

The rest of this section describes the message section only (except where stated).

7.2 DA_BAD_BLOCK (A586)

Indicates that a bad block was received. i.e. the data length in the block did not agree with the actual block length.

```
{
INT16     blocktype;          /* invalid block type */
}
```

7.3 DA_BAD_CMD (A584)

The command received is not in the table of valid DPU commands.

```
{
INT16       cmd;              /* The bad command type */
}
```

7.4 DA_BEGOF_EXP (A513)

Signals the beginning of exposure - i.e. the blue DSP event processing has been turned on.

```
{
DPUINT    frame_time;                /* Frame time 1/1000 secs */
DPUINT    frames_per_exposure;       /* No of frames for this exposure */
}
```

7.5 DA_BLUE1_E_ALERT (A594)

Indicates a blue DSP 1 error.

```
{
INT16     error;              /* DSP specific error word */
}
```

7.6 DA_BLUE1_S_ALERT (A595)

Indicates a blue DSP 1 status change.

```
{
INT16     status;             /* DSP specific status word */
}
```

7.7 DA_BLUE2_E_ALERT (A596)

Indicates a blue DSP 2 error.

```
{
INT16    error;             /* DSP specific error word */
}
```

7.8 DA_BLUE2_S_ALERT (A597)

Indicates a blue DSP{ 2 status change.

```
{
INT16    status;            /* DSP specific status word */
}
```

7.9 DA_BOTTOM_WHITEDSP (A582)

A ‘bug check’ - indicated that `CWHITE.C` module has ‘fallen through’ to an ‘impossible’ location.

```
{
INT16    status;            /* status */
}
```

7.10 DA_CID_TOO_LARGE (A583)

The command type received (the Command Identifier or CID) was not in the range A400(hex) to A4FF(hex).

```
{
INT16    cmd;               /* The command type  received */
}
```

7.11 DA_CLK_SYNCH_ERROR (A580)

A DPU clock synch error has been detected. This is always sent as a response to an IC_SYNCH_CLK command. It is context dependent (to be determined by the ICU) whether is actually an error.

```
{
DPUINT   command_time;      /* new time DPU Clock is set to */
DPUINT   old_dpu_time;      /* old incorrect DPU time */
}
```

7.12 DA_COMPLETE_EXP (A518)

This alert signals the completion of the tracking and shift and add of the last frame of an exposure. It also indicates the full submission of science data to the compression queue (to be compressed in the next exposure).

```
{
null;
}
```

7.13 DA_DATA_ALERT (A515)

Indicates that data of specified type is now available.

```
{
INT16    datatype;          /* type of data e.g. DD_IMG */
}
```

7.14 DA_DATA_END (A516)

Indicates that end-of-data has been reached.

```
{
null;
}
```

7.15 DA_DONE_FLD_DATA (A517)

Indicates field acquisition data has been obtained and that its OK to change filters.

```
{
null;
}
```

7.16 DA_DONE_SAA (A53E)

The Shift and add algorithm (SAA) on the red DSP signals to the white DSP when it completes. The White DSP then issues this alert.

```
{
null;
}
```

7.17 DA_DPU_ABORTING (A585)

Indicates that the DPU is entering swap unit SU_DPU_ABORTING.

```
{
INT16     abort_code;
}
```

7.18 DA_DPU_BOOT_READY (A510)

Issued on power up or after an IC_RESET_DSP command. - colloquially known as ‘Fred from the Dead’.

```
{
INT16     boot_status;       /* prior status after reboot */
}
```

7.19 DA_DPUOS_READY (A511)

Indicates that the DPU is up and running (colloquially known as 'Jim from the Rim')

```
{
INT16     dpu_status;        /* DPU Status */
}
```

7.20 DA_ECHO_ICU_CMD(A512)

Echoes successful reception of ICU command.

```
{
INT16        command_id  /* command type echoed */
}
```

7.21 DA_ENDOF_EXP (A514)

Indicates end of exposure initiated by IC_TRACK_GS. At this time the blue DSP event processing has been turned off

```
{
DPUINT    frame_time;              /* Frame time 1/1000 secs */
DPUINT-   frames_per_exposure;     /* No of frames for this exposure */
}
```

7.22 DA_EOS_ABORT (A53A)

Indicates that that swap unit SU_DPU_ABORT has finished. this would normally be run after an IC_ABORT_DPU command was received.

```
{
null;
}
```

7.23 DA_EOS_CALBGD (A531)

Indicates the end of the swap unit SU_CALBGD.

```
{
null;
}
```

7.24 DA_EOS_CALDRFT (A535)

Indicates the finish of swap unit SU_CALDFT.

```
{
null;
}
```

7.25 DA_EOS_COMPRESS (A53D)

Indicates end of compression swap unit.

```
{
null;
}
```

7.26 DA_EOS_DLVRDATA (A53B)

Indicates the end of swap unit SU_DELIVERDATA.

```
{
null;
}
```

7.27 DA_EOS_DOWINDOW (A537)

Indicates the finish of swap unit SU_DOWINDOW set-up.

```
{
null;
}
```

7.28 DA_EOS_ENG (A53C)

Indicates the end of swap unit SU_ENGINEERING.

```
{
null;
}
```

7.29 DA_EOS_GETREF (A538)

Indicates finish of the swap unit SU_GETREF.

```
{
null;
}
```

7.30 DA_EOS_INIT_DPU (A530)

Indicates the finish of swap unit SU_INITIALIZE. This is normally initiated by the DPU_INIT command.

```
{
null;
}
```

7.31 DA_EOS_LOCATEGS (A539)

Indicates end of swap unit SU_LOCATEGS.

```
{
null;
}
```

7.32 DA_EOS_MAKEAGS_ID (A534)

Indicates end of swap unit SU_MAKEAGS_ID.

```
{
null;
}
```

7.33 DA_EOS_RSTR_STARS (A533)

Indicates end of swap unit SU_RESTORE_STARS

```
{
null;
}
```

7.34 DA_EOS_SCANBS (A532)

Indicates end of swap unit SU_SCANBS.

```
{
null;
}
```

7.35 DA_EOS_SCANGS (A536)

Indicates end of swap unit SU_SCANGS.

```
{
null;
}
```

7.36 DA_EOT_ACQUIRE_FLD (A521)

Indicates the end of the ACQUIRE_FIELD task initiated by IC_ACQUIRE_FLD.

```
{
null;
}
```

7.37 DA_EOT_CHOOSE_GS (A522)

Indicates the end of the CHOOSE_GUIDE_STARS task initiated by IC_CHOOSE_GS.

```
{
null;
}
```

7.38 DA_EOT_ENG (A524)

Indicates the end of ENGINEERING task initiated by IC_ENBL_ENG.

```
{
null;
}
```

7.39 DA_EOT_FLUSH_CMPRS (A528)

Following reception of the IC_FLUSH_CMPRS command to force compression of data in the compression queue, this alert indicates that all data are compressed and submitted to the output data queue.

```
{
DPUINT    cmprss_exp;        /* ID of the exposure whose data sets have */
                             /*    been completely compressed */
}
```

7.40 DA_EOT_INIT_DPU (A520)

Indicates the end of the DPU Initialisation task INITIALIZE. This will have been initiated by the IC_INIT_DPU command. It is commonly known as “Dave from the Grave”

```
{
null;
}
```

7.41 DA_EOT_INIT_EXP (A523)

Indicates the end of the INIT_EXP task initiated by IC_INIT_EXP.

```
{
null;
}
```

7.42 DA_GADE_RBI (A5A6)

Indicates a white global memory access violation reading a big word.

```
{
DPUINT         address;  /* The address that is out of bound */
DPUINT         min;      /* Acceptable lower bound of memory address */
DPUINT         max;      /* Acceptable upper bound of memory address */
}
```

7.43 DA_GADE_RBIA_H (A5A8)

Indicates a white global memory access out-of-bounds violation reading a BIG array.

```
{
DPUINT   address;  /* The address that is out of bound */
}
```

7.44 DA_GADE_RBIA_L (A5A7)

Indicates a white global memory access wrong-value violation reading a BIG array.

```
{
INT16    error_mode;    /* 0: xsize error, 1: ysize error, 2: yskip error */
INT16    error_number;  /* values that is in error */
}
```

7.45 DA_GADE_RPI (A5A0)

Indicates a white global memory access violation reading a PROC word.

```
{
DPUINT    address;          /* The address that is out of bound */
DPUINT    min;              /* Acceptable lower bound of memory address */
DPUINT    max;              /* Acceptable upper bound of memory address */
}
```

7.46 DA_GADE_RPIA_H (A5A2)

Indicates a white global memory access out-of-bound violation reading a PROC array.

```
{
DPUINT    address;          /* The address that is out of bound */
} GADE_RPIA_H;
```

7.47 DA_GADE_RPIA_L (A5A1)

Indicates a white global memory access wrong-value violation reading a PROC array.

```
{
INT16    error_mode;     /* 0: xsize error, 1: ysize error, 2: yskip error */
INT16    error_number;   /* values that is in error */
}
```

7.48 DA_GADE_RSI (A5AC)

Indicates a white global memory access violation reading a SMALL word.

```
{
DPUINT          address;        /* The address that is out of bound */
DPUINT          min;            /* Acceptable lower bound of memory address */
DPUINT          max;            /* Acceptable upper bound of memory address */
}
```

7.49 DA_GADE_RSIA_H (A5AE)

Indicates a white global memory access out-of-bounds violation reading a SMALL array.

```
{
DPUINT         address;      /* The address that is out of bound */
}
```

7.50 DA_GADE_RSIA_L (A5AD)

Indicates a white global memory access wrong-value violation reading a SMALL array.

```
{
INT16          error_mode;   /* 0: xsize error, 1: ysize error, 2: yskip error
*/
INT16          error_number; /* values that is in error */
}
```

7.51 DA_GADE_WBI (A5A9)

Indicates a white global memory access violation writing a BIG word.

```
{
DPUINT    address;  /* The address that is out of bound */
DPUINT    min;      /* Acceptable lower bound of memory address */
DPUINT    max;      /* Acceptable upper bound of memory address */
}
```

7.52 DA_GADE_WBIA_H (A5AB)

Indicates a white global memory access out-of-bounds violation writing a BIG array.

```
{
DPUINT          address;        /* The address that is out of bound */
}
```

7.53 DA_GADE_WBIA_L (A5AA)

Indicates a white global memory access wrong-value violation writing a BIG array.

```
{
INT16     error_mode;    /* 0: xsize error, 1: ysize error, 2: yskip error */
INT16     error_number;  /* values that is in error */
}
```

7.54 DA_GADE_WPI (A5A3)

Indicates a white global memory access violation writing a PROC word.

```
{
DPUINT      address;          /* The address that is out of bound */
DPUINT      min;              /* Acceptable lower bound of memory address */
DPUINT      max;              /* Acceptable upper bound of memory address */
}
```

7.55 DA_GADE_WPIA_H (A5A5)

Indicates a white global memory access out-of-bounds violation writing a PROC array.

```
{
DPUINT     address;/* The address that is out of bound */
}
```

7.56 DA_GADE_WPIA_L (A5A4)

Indicates a white global memory access wrong-value violation writing a PROC array.

```
{
INT16     error_mode;    /* 0: xsize error, 1: ysize error, 2: yskip error */
INT16     error_number;  /* values that is in error */
}
```

7.57 DA_GADE_WSI (A5AF)

Indicates a white global memory access violation writing a SMALL word.

```
{
DPUINT         address;       /* The address that is out of bound */
DPUINT         min;           /* Acceptable lower bound of memory address */
DPUINT         max;           /* Acceptable upper bound of memory address */
}
```

7.58 DA_GADE_WSIA_H (A5B1)

Indicates a white global memory access out-of-bounds violation writing a SMALL array.

```
{
DPUINT         address;         /* The address that is out of bound */
}
```

7.59 DA_GADE_WSIA_L (A5B0)

Indicates a white global memory access wrong-value violation writing a SMALL array.

```
 {
INT16     error_mode;   /* 0: xsize error, 1: ysize error, 2: yskip error */
INT16     error_number; /* values that is in error */
 }
```

7.60 DA_HBEAT(A500)

DPU 'Heartbeat'.

```
{
INT16    status;            /* contents- see below */
}
```

The above status word will contain the following:

Item	No of Bits	Size (bits)	Comments
DPU State	15	1	MSB, 0 = DPU_BOOT, 1 = DPUOS
Blue1 DSP Status	14	1	
Blue2 DSP Status	13	1	
Red DSP Status	12	1	1 = on, 0 = off
bluedsp2 data capture	11	1	
bluedsp2 data capture	10	1	
Tracking	9	1	0 = On, 1 = Off
Current Swap Unit ID	4-8	5	
Current DPU Task ID	0-3	4	LSB

The DPU task ID's are as follows:

MNEMONIC	Number	Comment
IDLE	0	index to idle
ABORT	1	index to abort
INITIALIZE	2	index to reinitialise
INIT_EXP	3	index to INITIALIZE exposure task
ACQUIRE_FIELD	4	index to acquire field task
CHOOSE_GUIDE_STARS	5	index to choose guide star task
TRACK_GUIDE_STARS	6	index to track guide star task
COMPRESS_DATA	7	index to compress data task
FINISH_FRAME	8	index to finish frame task
ENGINEERING	9	index to engineering task
FLUSH_COMPRESS	10	index to flush compression task
RTRD	11	index to rtrd task

See IC_REPORT_DIAGS (A423) for a summary of the swap unit ID's

7.61 DA_PROG_RAM_ERROR (A581)

Reports any errors generated when checking the integrity of the program stored in the keep-alive RAM, using Hamming codes.

```
{
DPUINT     corrected_opcode;
INT16      syndrome;
INT16      offset_of_error;
INT16      error_opcode;
}
```

7.62 DA_RED_E_ALERT (A592)

Indicates a red DSP error.

```
{
INT16     error;             /* DSP specific error word */
}
```

7.63 DA_RED_S_ALERT (A593)

Indicates a red DSP status change.

```
{
INT16     status;            /* DSP specific status word */
}
```

7.64 DA_SCI_HBEAT_ERROR (A591)

Indicates failure to receive heartbeat from a DSP via the SCI. The SCI is the Serial Communications Interface. This is a single wire serial interface for commanding from the white DSP to the others.

```
{
INT16     diagnostic         /* which DSP has h/beat problems */
}
```

Note: `diagnostic` takes the values `0/1/2/3 = blue1/red/blue2/red` which differs from the usual DSP values used in the rest of this section.

7.65 DA_SCI_RCVR_ERROR (A590)

Indicates a SCI receiver error.

```
{
null;
}
```

7.66 DA_SENT_CMPRSSQ (A527)

Indicates that compressed data of the specified type has been submitted to the DPU data queue. It thus informs the ICU of the data about to be transmitted.

```
{
DPUINT    exposure;           /* Exposure ID of the compressed data set */
INT16     data_type;          /* The DD_xxx data type compressed */
INT16     data_set_id;        /* The data set ID, i.e. science window ID */
DPUINT    data_in_address;    /* Starting Address of uncompressed data */
DPUINT    data_in_length;     /* Number of DPUINT's of uncompressed data */
DPUINT    data_out_addr0;     /* Starting Address of compressed data */
DPUINT    data_out_length;    /* Number of 16 bit words of compressed data */
}
```

7.67 DA_SWITCH_FRAME (A525)

Indicates switching ping/ping tracking

```
{
INT16     frame_count;
}
```

7.68 DA_TRK (A526)

Tracking Reports - turned on by IC_REPORT_TRK.

```
{
INT16     frame_id;    /* frame ID */
DPUINT    xdrift;      /* tracking drift in x (1/1000 centroided pixel) */
DPUINT    ydrift;      /* tracking drift in y (1/1000 centroided pixel) */
DPUINT    roll;        /* tracking drift in roll (1,000,000 * sin(theta) */
}
```

7.69 DA_ZERO_LENGTH_DQ (A587)

Indicates an error condition of requesting a non-positive length for data queue.

```
{
INT16       datatype;       /* Data type */
}
```

8. ICU TO DPU COMMANDS (IC_xxx)

8.1 Overview

All commands (unless stated otherwise) are of the form

```
{
INT16       Type;          /* E.g. IC_ABORT_DPU */
INT16       n;             /* No of 16 bit words to follow */
INT16       params[n];     /* The command parameters - may be of zero length
*/
}
```

The rest of this section describes the command parameters only (except where stated).

Note

In this section, 'DSP' takes the following values.

0 = blue1
1 = blue2
2 = red
3 = white
4 = all
5 = blue1 and blue2

8.2 IC_ABORT_DPU (A435)

Command DPU to abort - it gracefully leaves an exposure by initiating the abort task.

```
{
null;
}
```

8.3 IC_ACQUIRE_FLD (A432)

Command DPU to execute ACQUIRE_FIELD task.

```
{
null;
}
```

8.4 IC_BPE_BINNING (A421)

Informs DPU of status of (‘on-chip’) binning of the detector.

```
{
INT16     bpe_binning;   /* 0=no binning, 1=binning */
}
```

8.5 IC_CHOOSE_GS (A433)

Command DPU to execute CHOOSE_GUIDE_STARS task.

```
{
null;
}
```

8.6 IC_CONF_GS_SEL (A427)

Set number of guide stars and their selection criteria.

```
{
INT16     number;             /* number of guide stars */
DPUINT    criteria_mask;      /* criteria mask */
}
```

8.7 IC_DIRECT_DATA (A409)

Direct detector data to either blue1, blue2 or both blue DSPs.

```
{
INT16     DSP;                /* DSP */
}
```

8.8 IC_DUMP_PROG_RAM (A414)

To be implemented. Requests a dump of Program RAM - response in DR_PROG_DUMP.

```
{
DPUINT    address;           /* Start Address */
INT16     length;            /* No 24 bit words to dump */
}
```

8.9 IC_DUMP_LOCAL_RAM (A410)

Requests a dump of local RAM - response in DR_LRM.

```
{
INT16     DSP;                 /* DSP */
INT16     local_address;       /* local address - < 32K = 0x8000 */
INT16     length;              /* length (must be non-zero) */
}
```

8.10 IC_DUMP_RAM (A415)

Dump n 24 bit words from global ram (an acceptance / confidence test mode of operation).

```
{
DPUINT    address;             /* start address */
INT16     nwords;              /* number 24 bit words */
}
```

8.11 IC_DUMP_RAM_N_ZERO (A416)

Dump n 24 bit words from global ram (an acceptance / confidence test mode of operation) and then zero the ram.

```
{
DPUINT    address;             /* start address */
INT16     nwords;              /* number 24 bit words */
}
```

8.12 IC_ENBL_DSP (A407)

Enable/disable DSP "DSP", (doesn't include white and all).

```
{
INT16     DSP;                 /* DSP */
INT16     enable               /* E/D = Enable/Disable = 1/0 */
}
```

8.13 IC_ENBL_ENG (A436)

Enable/disable DPU engineering mode data.

```
{
INT16     det;                 /* detector (always = 1) */
INT16     mode;                /* mode */
}
```

'mode' selects one of the following options:

Mode	Description	
0	Raw data	with both blue DSPs
1	Raw data	with only blue DSP 1
2	Raw data	with only blue DSP 2
3	Centroiding data - get channel boundaries and M,N values	
4	Full frame, high resolution	
5	Centroiding confirmation data	
6	Intensifier characteristics - event height information	

8.14 IC_ENBL_EVENTS (A408)

Enable/disable data generation from DSP "DSP" (doesn't include white and all).

```
{
INT16     DSP;                /* DSP */
INT16     enable;             /* E/D = Enable/Disable = 1/0 */
}
```

8.15 IC_ENBL_LOAD_CODE (A403)

Lock/unlock loading of code.

```
{
INT16  password;   /* password = 0xf352 = enabled, otherwise = disabled */
}
```

8.16 IC_ENBL_VERBOSE (A42C)

Enables long form of data.

```
{
INT16     data_type;          /* data type */
INT16     enable;             /* E/D = Enable/Disable = 1/0 */
}
```

'data_type' is

Data Type	Description	Comment
0	tracking	not implemented (but see IC_REPORT_TRK).
1	field acquisition	not implemented.
2	window parameters	not implemented.
3	reference frame.	modifies format of DD_REF record

8.17 IC_FLUSH_CMPRS (A437)

Forces compression of all entries in the compression queue. The alert DA_EOT_FLUSH_CMPRS is issued when all data are compressed and sent to the data queue.

```
{
null;
}
```

8.18 IC_FLUSH_QUEUE (A40F)

Request for the white processor to dump all data currently queued for output.

```
{
null;
}
```

8.19 IC_INIT_EXP (A431)

Invokes the INIT_EXP task. This sets up the DPU to acquire detector data in 1k * 1k format (i.e. detector binned by 2).

```
{
null;
}
```

8.20 IC_INIT_DPU (A430)

Zeroes memory, readies swap units.

```
{
null;
}
```

8.21 IC_LOAD_CODE (A405)

Load code into RAM.

```
{
INT16     address;    /* address offset = kkkk (hex)- see note below */
INT16     di[n-2];    /* 24 bit words ‘spanned’ across 16 bit words */
INT16     checksum;   /* checksum */
}
```

Note: The start address of the code load is 0xEjkkkk, where j is set by IC_SEL_RAM_BANK.

8.22 IC_LOAD_DET_WDW (A42D)

Loads Detector Windows directly

```
typedef struct
{
INT16      n_dtw;       /* Total Number of Windows */
INT16      i_dtw;       /* Specifies which of n_dtw this is */
DPUINT     x0;          /* Bottom left corner x co-ordinate */
DPUINT     y0;          /* Bottom left corner y co-ordinate */
DPUINT     xsize;       /* Width of window */
DPUINT     ysize;       /* Height of Window */
DPUINT     immw;        /* Window ID */
DPUINT     kompx;       /* These parameters control where in memory..   */
DPUINT     bb;          /*.. events for window immw are binned          */
DPUINT     aa;          /*                                              */
DPUINT     kompy;       /*                                              */
}
```

8.23 IC_LOAD_DPUOS (A402)

Instructs DPU to load the DPUOS. This is used when the DPU is in Boot Ready state.

```
{
null;
}
```

8.24 IC_LOAD_FILT_CONF (A429)

Inform DPU of type of filter being used (TBC).

```
{
INT16    filt;              /* filter (0 -> 11)*/
}
```

8.25 IC_LOAD_GLOBAL_RAM (A417)

Load Global Ram

```
{
DPUINT   address;           /* address */
DPUINT   data;              /* data */
}
```

8.26 IC_LOAD_MEM_WDW (A42A)

Uplink memory window configuration data;

X0,Y0 = bottom left window co-ordinates, Xsize,Ysize are window dimensions.

The following data structure is required

```
typedef struct
  {
  INT16      x0;        /* bottom left x co-ordinate of window */
  INT16      y0;        /* bottom left y co-ordinate of window */
  INT16      xsize;     /* x dimension of window (centroided pixels) */
  INT16      ysize;     /* y dimension of window (centroided pixels) */
  INT16      base;      /* data retrieval address in DPU RAM */
  } MEM_WIND16;

{
INT16                   nmwind;         /* no of memory winds (max 5) */
MEM_WIND16              mem_wind[nmwind];
}
```

NOTE: The uplinked entry `base` is ignored and re-assigned by the DPU.

8.27 IC_LOAD_LOCAL_RAM (A411)

Load White Local Ram

```
{
INT16     address;            /* address */
DPUINT    data;               /* data */
}
```

8.28 IC_LOAD_REF_STARS (A428)

Load list of field acquisition reference stars.

The following data structure is required by this data record

```
typedef struct
          {
          INT16               x;       /* x co-ordinate of reference star */
          INT16               y;       /* y co-ordinate of reference star */
          } REF_POS;
```

The parameters are as follows

```
{
INT16     nstars;             /* number of reference stars <= 32 */
REF_POS   star[nstars];       /* reference stars */
}
```

where `REF_POS` is defined as above

8.29 IC_LOAD_SCI_WDW (A42B)

Up-link science window configuration data;

X0,Y0 = bottom left window co-ordinates, Xsize,Ysize are window dimensions.

The following data structure is required

```
typedef struct
 {
      INT16      x0;            /* bottom left x co-ordinate of window */
      INT16      y0;            /* bottom left y co-ordinate of window */
      INT16      xsize;         /* x dimension of window (centroided pixels) */
      INT16      ysize;         /* y dimension of window (centroided pixels) */
      INT16      mode;          /* operation mode (e.g., Fast) */
      INT16      p1;            /* mode-dependent parameter - see notes */
      INT16      p2;            /* mode-dependent parameter - see notes */
      INT16      p3;            /* mode-dependent parameter - see notes */
      INT16      p4;            /* mode-dependent parameter - see notes */
      INT16      base;          /* data storage address in DPU RAM */
      INT16      mmwid;         /* id of corresponding memory wndw */
 }    SCI_WIND16;
```

The parameters as follows:-

```
{
INT16            nsciwind;     /* no of science windows to follow (max 5)*/
SCI_WIND16       sciwind[nsciwind];
}
```

Notes:

1. `base` is "don't care" in value
2. `Mode` has the following meanings:

Mode	**Description**
0	Image
1	Fast
15	Tracking Window - N/A

3. `P1..P4` are mode parameters - see DP_WDW for details.
4. The uplinked entries `base` and `mmw_id` are ignored and re-assigned by the DPU

8.30 IC_POWER_DOWN_DPU (A40A)

Powers down the DPU.

```
{
INT16     password;           /* password = !(0xF352) = 0x0CAD */
                              /* (Comprehensive Automatic Disabler !) */
}
```

8.31 IC_REPORT_DIAGS (A423)

Activate/deactivate log- keeping for swap unit "module".

```
{
INT16    module;              /* module number (see below) */
INT16    enable;              /* E/D = Enable/Disable = 1/0 */
}
```

Module (swap unit) Numbers have the following meanings:

NUMBER	SWAP UNIT	Comment
SWAP_INITIALIZE	0	index to swap unit code
SWAP_CALBGD	1	index to swap unit code
SWAP_SCANBS	2	index to swap unit code
SWAP_RESTORE_STARS	3	index to swap unit code
SWAP_MAKEAGS_ID	4	index to swap unit code
SWAP_CALDRFT	5	index to swap unit code
SWAP_SCANGS	6	index to swap unit code
SWAP_DOWINDOW	7	index to swap unit code
SWAP_GETREF	8	index to swap unit code
SWAP_LOCATEGS	9	index to swap unit code
SWAP_COMPRESS	10	index to swap unit code
SWAP_DPUABORT	11	index if major error
SWAP_DELIVERDATA	12	index for data delivery
SWAP_INIT_EXP	13	index for exposure init
SWAP_ENGINEERING	14	index for engineering mode
SWAP_SUBMITCMPRSSQ	15	index for submit compression mode
SWAP_RTRD	16	index for rtrd mode

8.32 IC_REPORT_TRK (A422)

Report tracking history - causes DA_TRK alerts to be sent.

```
{
INT16     enable;             /* E/D = Enable/Disable = 1/0 */
}
```

8.33 IC_REQ_DATA (A40D)

Request ’number’ blocks of data

```
{
INT16     number;             /* number of blocks requested */
}
```

8.34 IC_RESET_DSP (A401)

Hardware Reset of the selected DSP (not all, includes both).

```
{
null;
}
```

8.35 IC_SEL_KLINGON (A406)

Select which DSP will be the klingon (blue1, blue2, red or white).

```
{
INT16     DSP;                /* DSP */
}
```

8.36 IC_SEL_RAM_BANK (A404)

Specifies the bank of program RAM to be loaded. It is used in conjunction with IC_ENBL_LOAD_CODE and IC_LOAD_CODE.

```
{
INT16     eeprom;             /* Keep Alive RAM */
}
```

'eeprom' can take the following values:

eeprom	ADDRESS
0	E00000
1	E10000
2	E20000
3	E30000
4	E40000
5	E50000
6	E60000
7	E70000

8.37 IC_SEND_SCI_CMD (A40B)

Sends the command ‘message’ from the white DSP to the target DSP over the SCI interface.

```
{
INT16     DSP                /* 0/1/2/5 = blue1/blue2/red/blue_both */
INT16     sci_command[n];    /* command */
}
```

Applicability BR = both BO = Blue only RO = Red Only	Mnemonic	Code (hex)	Description
BR	REQST_HEART_BEAT	A0	tells a BLUE DSP to send heart beat message to the white DSP
BO	SEND_COUNTS_TO_PING	A1	tells a BLUE DSP to use the dtw constants that send cts to "ping" frame
BO	SEND_COUNTS_TO_PONG	A2	tells a BLUE DSP to use the dtw constants that send cts to "pong" frame
BR	ENABLE_KLINGON	A3	used to command a DSP into Klingon state, hold onto the global bus until an interrupt is received indicating start of the next millisecond
BR	DISABLE_KLINGON	A4	used the opposite of the one above
BO	ENABLE_EVENTS	A5	used in command to DSP to activate event processing
BO	DISABLE_EVENTS	A6	used in command to DSP to deactivate event processing
BO	ENABLE_ENGINEERING	A7	tells the specified DSP to enter the engineering mode
BR	DUMP_LOCAL_RAM	A8	message to BLUE DSPs to ask them to dump 32K local RAM to BIG_MIN area
BO	ENABLE_RAM_TEST	A9	message to BLUE DSPs to start testing global RAM accesses
BO	DISABLE_RAM_TEST	AA	used the opposite of the one above
BO	GET_DTW_PARAMETERS	AB	message to BLUE DSPs to ask them to fetch detector window parameters
BO	GET_BFAST_PARAMETERS	AC	message to BLUE DSPs to ask them to fetch bfast parameters
BO	DSBL_FAST_DATA_OUTPUT	AE	message to BLUE DSPs to ask them to stop collecting fast data
RO	DO_RED_TASK_IDLE	AB	tell Red DSP to idle
RO	DO_RED_TASK_ABORT	AC	tell Red DSP to abort
RO	DO_RED_TASK_ACCUMULATE	AD	tell Red DSP to run shift-and-add
RO	DO_RED_TASK_INITIALIZE	AE	tell Red DSP to INITIALIZE
RO	DO_RED_TASK_RTRD	AF	tell Red DSP to rtrd

8.38 IC_SET_EXP_ID (A426)

Specify exposure ID

```
{
DPUINT    exposure_id;        /* Exposure ID */
}
```

8.39 IC_SET_EXP_TIME (A425)

Define exposure duration as some integral number of tracking frames.

```
{
INT16     n_frames;           /* no frames */
}
```

8.40 IC_SET_FRAME_TIME (A424)

Set tracking frame duration (where 1 DPU cycle = 0.001 seconds).

```
{
DPUINT    n_dpu_cycles;       /* no DPU cycles */
}
```

8.41 IC_SET_TRK (A420)

Enable /disable tracking

```
{
INT16      tracking_on;   /* 0=on, 1=off, NOTE THIS!!! */
}
```

8.42 IC_SYNCH_CLK (A40E)

This command informs the DPU of the time at the next ZeroT signal from the ICU. (A DA_CLK_SYNCH_ERROR is always sent as a response to this command. Whether it is to be treated as an error is context dependent.)

```
{
INT16     time;              /* In units of 1 sec */
}
```

8.43 IC_TRACK_GS (A434)

Command DPU to execute TRACK_GUIDE_STARS task

```
{
null;
}
```