Modified Commanding Scheme

CDS-MSSL-TN-0131
Version 1.3

Distribution: MKC, CJ, RAH, DJP, JP, CDP (RAL)
KFJ, RAG, KN, ERB, AJM, PE (MSSL)
RAH please distribute further if necessary

Khalid al Janabi, E R Breeveld. MSSL, 19 April 1993.

Changes:

Version	Date	Comment
1.3	18/5/93	Health Monitor Destination added
1.3	3/6/93	Error Process Destination added

Contents

1. Purpose of this document

This document describes the modified commanding scheme.

It then details the scheme, with examples. These examples are very simple, using just a few commands.

2. Requirements

- The system must be able to cope with the highest ESA defined command rate.

The following come from the CDS Software Requirements Document (document number not assigned).

C1 The CDHS must be able to respond to commands from the ground and from the spacecraft.

C2 These commands will include:
- i) low level commands applying directly to subsystems or processors.
- ii) medium level commands putting the systems into pre-defined instrument modes, e.g., SNOOZE, STANDBY, etc. ... with some transitions forbidded for safety reasons.
- iii) high level single commands with a small number of arguments, which run stored command lists, e.g., for observing sequences.

C3 The CDHS must give real-time response to commands and accept time-tagged commands and stored commands.

C4 We must be able to sensibly interrupt, reset, or query the state of the CDHS whilst "higher level" commands are running.

C5 We must have the ability to store a series of commands which may be called by a single command, and to change and add series to the store.

The system must also be compatible with specific comments on commanding in other sections of the Requiremments Document.

3. Design

- All command blocks will have the same structure.

- All internal (on-board) command blocks will have the same structure as ground command blocks, with the exception of the checksum and MLA.

4. Command structure

Most CDS commands must be packed into an On-Board Data Handling (OBDH) command block before transmission to the spacecraft. Exceptions to this rule are for hardware or OBDH commands that are not sent from the EGSE.

A CDS OBDH command block is defined by a collection of 16 bit words, starting with the ESA defined MLA (memory load A) word followed by at least two 16 bits data words. The two data words fill the MLB (memory load B).

EID definition		CDS Terminology		Size
OBDH command block	MLA	MLA		1 word (16 bits)
	MLB	Command block	Block Header	1 word
			Block Body (arguments)	1 to 29 words
	Checksum	Checksum		1 word

The Block Body must have at least one 16 bit word (argument) to make up the full command block. Similar command blocks (with the same block header) can be com-bined into a single block. The largest block size available is 29 from the ground, or currently about 100 on-board.

Note that the checksum and MLA are stripped from the OBDH command block after checking on-board. Internal CDHS command blocks thus have the same structure without the checksum or MLA.

The word *command* (as opposed to *command block)* is ambiguous, as it could either mean a whole command block, or an individual argument in the block body. It should be avoided if possible. Unfortunately, the EID definition of a command is an MLA word that is not followed by any MLB words.

In the following sections, the command blocks will be illustrated as individual bits within each word, as follows:

Bit 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Bit15
MLA															
Block Header															
Argument 1															
...										Block Body					
Argument 'n'															
Checksum															

5. Block Header

Each block has a header word, which states the on-board *destination* (4 bits), *function* (5 bits), and *number of arguments to follow* (7 bits). There is no end of block argument, other than the checksum.

The *destination* is the module with the CDHS on-board software that the block is intended for. The *function* specifies what type of action is to be taken, and the *arguments* specify the exact action.

Thus the structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination				Function					Number of arguments ('n')						
Argument 1															
...															
Argument 'n'.															
Checksum															

This structure allows the commanding of up to 16 destinations, with 32 functions at each destination. Note that many more command blocks can be defined by each argument (up to 65536 different combinations per argument).

6. Command block implementation

All the currently defined destination are listed below. Note that these may be added to as the software develops.

Below is a list of the currently defined destinations:

Module	Destination number
CDHS power switching	1
IMIF	2
Watchdog	3
Deferred command store	4
Macro	5
Science	6
Primary memory dump	7
Secondary memory dump	8
(unused)	9
Engineering acquisition	A (hex)
CDHS engineering	B (hex)
CDHS housekeeping	C (hex)
Health Monitor	D (hex)
Error Process	E (hex)
(unused)	F (hex)

The following sections give further details about each destination. Note that the information about functions and arguments will change as each module is coded.

6.1. CDHS power switching command blocks

Block destination	1
Function	1
No. of arguments to follow	variable, depending on the number of arguments to be sent

This header is then followed by power switching arguments (as defined by the green book).

For examples, see the sections on single and multilple argument command blocks on pages 9 and 9.

6.2. IMIF command blocks

Block destination	2	
Function	4	VDS subsystem i.d.
	6	GIS subsystem i.d
	8	EPS subsystem i.d.
	A	MCU subsystem i.d.
	B	enable/disable VDS HK.
	C	enable/disable VDS heartbeat.
	D	enable/disable both (VDS HK and heartbeat)
No. of arguments to follow	variable, depending on the number of arguments to be sent	

This header is then followed by the subsystem arguments, as defined by the subsystems.

Additionally, Enable/Disable conditions are established by the argument hex 0001 for enable and hex 0002 for disable.

For examples, see the sections on single and multilple argument command blocks on pages 9 and 9.

6.3. Reset watchdog

Block destination	3
Function	1
No. of arguments to follow	always 3

The parameters to follow are F001, F002 and F003. This to ensure that no spurious watchdog reset occurs. For an example see the multilple argument command block section on page 9.

6.4. Deferred command store

Block destination	4	
Function	1	Disable/Enable deferred command store
	2	fill deferred command store
	3	set pointer
	4	query deferred command store
No. of arguments to follow	variable, depending on the number of arguments to be sent	

See the Deferred Command section on page 10 for more information of filling the deferred command store.

6.5. Macro process

The macro commanding process is subject to further discussion. The list of these command blocks is preliminary.

Block destination	5	
Function	0	Mode (including pause, restart and abort raster).
	1	fill Mode tables
	2	fill line list no. x
	3	fill series list no. x
	4	fill raster list no. x
	5	query raster list no. x
	6	query line list no. x
	7	query parameter list
	8	query series list no. x
	9	run raster no. x
	A	run series no. x
	B	run parameter list
	C	load line list no. x (load to science process).
	D	delay (wait for x ms)
	E	fill parameter list
No. of arguments to follow	variable, depending on the number of arguments to be sent	

Note that up to 32 functions can be defined, thus there is still plenty spare.

See Series Fill and Line Fill sections (pages 11 and 12) for examples of these fill command blocks.

6.6. Science

The science process is subject to further discussion. The list of these command blocks is preliminary.

The science process allows loading from the macro-process (parameters and line lists)

Block destination	6	
Function	0	reserved for inter experiment mode (master or standby)
	1	write GIS raster header
	3	write VDS raster header
	5	execute
No. of arguments to follow	variable, depending on the number of arguments to be sent	

6.7. Memory dump (special telemetry / special case)

Block destination	8	
Function	0	
No. of arguments to follow	variable, depending on the number of arguments to be sent	

It is anticipated that the arguments should contain two words for the starting address.

6.8. Engineering acquisition command blocks

Block destination	A	
Function	1	set science to engineering ratio
No. of arguments to follow	1	

The single argument is the interval for Engineering data to be transmitted in seconds.

For an example, see the section on single argument command blocks on page 9.

6.9. CDHS engineering command blocks

Block destination	B	
Function	1	engineering data reset
No. of arguments to follow	one	

The single argument is a physical location in the engineering telemetry data array to be reset to 0.

For an example, see the section on single argument command blocks on page 9.

6.10. CDS HK command blocks

Block destination	C	
Function	1	configure location in HK table
No. of arguments to follow	variable, depending on the number of arguments to be sent	

For examples, see the sections on single and multilple argument command blocks on pages 9 and 9.

7. Single argument command blocks

Single argument command blocks can be constructed by taking the destination and function to form the header, and following this by the single argument (the command body).

For instance,

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination =2				Function =8					Number of arguments ('n') =1						
Argument 1 =0xABAB															
Checksum															

The final command block would then look like:

(MLA)	! MLA as per EID
0x2401	! Destination = 2, Function = 8 (5 bits), no = 1 (7 bits)
0xABAB	! Argument 1
(checksum)	

8. Multiple argument command blocks

Multiple argument command blocks can be constructed by taking the destination and function to form the block header, and following this by the arguments. *Only command blocks with the same destination and function can be blocked together.*

For instance:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination =2				Function =8					Number of arguments ('n') =2						
Argument 1 =0xABAB															
Argument 2 =0xCBCB															
Checksum															

The final command would then look like:

(MLA)	! MLA as per EID
0x2402	! Destination = 2, Function = 8 (5 bits), no = 2 (7 bits)
0xABAB	! Argument 1
0xCBCB	! Argument 2
(checksum)	

9. Deferred Command blocks

There are two sorts of deferred command blocks, control and fill. Deferred Command Store *control* command blocks follow the single and multiple argument command structure outlined above.

Deferred command *fill* command blocks are collections of command blocks, each preceded by a 32 bit time. They are then blocked with a header and a location within the store.

Note that each deferred time should be followed by at least 2 arguments (header for destination and at least one argument). The deferred time is a 32 bit representation of the Local On Board Time (LOBT) in seconds (see EID-A section 3.3.9.2). For instance:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination =4				Function =2					Number of arguments ('n') =9						
Argument 1 =25															
Time 1 argument 1 =0x0034															
Time 1 argument 2 =0x1682															
Defer. dest. =2				Defer. Function =8					Defer. no. of arguments ('n') =1						
Defer. argument 1 =0xABAB															
Time 2 argument 1 =0x0034															
Time 2 argument 2 =0x17A6															
Defer. dest. =2				Defer. Function =4					Defer. no. of arguments ('n') =1						
Defer. argument 2 =0x8000															
Checksum															

The final command would then look like:

(MLA)	! MLA as per EID
0x4109	! Destination = 4, Function = 2 (5 bits), no = 9 (7 bits)
0x0019	! Argument 1, location in the deferred command store
0x0034	! Time 1 (most sig. word)
0x1682	! Time 1 (least sig. word)
0x2401	! Destination = 2, Function = 8 (5 bits), no = 1 (7 bits)
0xABAB	! Command CBEGHV4N
0x0034	! Time 2 (most sig. word)
0x17A6	! Time 2 (least sig. word)
0x2201	! Destination = 2, Function = 4 (5 bits), no = 1 (7 bits)
0x8000	! Command CBVNOOP
(checksum)	

10. Series Command blocks

Series command store control command blocks follow the single and multiple argument command structure outlined above.

Series fill command blocks are collections of single or multiple commands. They are blocked with a header and a series number and location within the series.

For instance:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination =5				Function =3					Number of arguments ('n') =6						
Argument 1 (msb) =7								Argument 1 (lsb) =3							
Series. dest. =2				Series. Function =8					Series. no. of arguments ('n') =2						
Series command 1 argument 1 =0xABAB															
Series command 1 argument 2 =0xBCBC															
Series. dest. =2				Series. Function =4					Series. no. of arguments ('n') =1						
Series command 3 argument 1 =0x8000															
Checksum															

Note that individual deferred commands do not have to have the same destination and function.

The final command would then look like:

```
(MLA)          ! MLA as per EID
0x5185         ! Destination = 5, Function = 3 (5 bits), no = 5 (7 bits)

0x0703         ! Series number 7, location in the series 3

0x2401         ! Destination = 2, Function = 8 (5 bits), no = 1 (7 bits)
0xABAB         ! Series Command 1
0xBCBC         ! Series Command 2

0x2201         ! Destination = 2, Function = 4 (5 bits), no = 1 (7 bits)
0x8000         ! Series Command 3

(checksum)
```

11. Line Store Command blocks

Line store control command blocks follow follow the single and multiple argument command structure outlined above.

Line store fill command blocks are collections of numbers. They are then blocked with a header and a line list number. (It may be necessary to add additional information or arguments depending on the design of the Macro module.)

For example, the line list store fill instructions:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MLA															
Destination =5				Function =2					Number of arguments ('n') =9						
Argument 1 (msb) =9								Argument 1 (lsb) =1A							
Line list argument 1 =33															
Line list argument 2 =45															
Line list argument 3 =1															
Line list argument 4 =128															
Line list argument 1 =68															
Line list argument 2 =95															
Line list argument 3 =1															
Line list argument 4 =128															
Checksum															

The final command would then look like:

```
(MLA)           ! MLA as per EID
0x5109          ! Destination = 5, Function = 2 (5 bits), no = 9 (7 bits)
0x091A          ! List number = 9, location in the list = 1A (hex)
0x0021          ! Pixel 1 x1
0x002d          ! Pixel 1 x2
0x0001          ! Pixel 1 y1
0x0080          ! Pixel 1 y2
0x0044          ! Pixel 2 x1
0x005f          ! Pixel 2 x2
0x0001          ! Pixel 2 y1
0x0080          ! Pixel 2 y2
(checksum)
```

12. Other lists

There is much room for expansion here, without changing the on-board router or EGSE code.

This scheme will easily accommodate Error actions, Raster Parameter Lists, Health Monitor values, Out-of-Limits actions etc.; all using standard block command fill instructions. Only additional entries in the EGSE database would be needed.