APPENDIX

A. Telecommand Verification

A.1 Verification of Single Telecommands

Every Telecommand (except for TCs entered in ROBUST1 or ROBUST2) can be verified in HK Telemetry by looking at the parameters:

- CRCNT (Command received counter)
- CVCNT (Command verified counter)
- LASTC (Last Command)

CRCNT and CVCNT must be incremented by one, LASTC must have the value specified in the LASTC-column of the following table:

T/C mnemonic	T/C Parameter	affected HK Parameter	HK Par value	LASTC	TC notes	HK notes
ZEENOOPD	---	---	---	0100		
ZEELSOPM	xx	---	---	02xx	1	
ZEELDEPM	xx	---	---	03xx	1	
ZEEJDEPS	xx			04xx	2	a
ZEELDBYS	xx			05xx	2	b
ZEELDWDS	yy			06yy	2,3	b
ZEELBYCM	xx	---	---	07xx	1	
ZEERBT2S	---				20	
ZEEFGSFS	11 01 12 02			0911 0901 0912 0902	18 19 18 19	
ZEEBKGDS	xx	---	---	0Axx	--	
ZEEIMSKS	xx	---	---	0Bxx	--	
ZEERGIFE	---	PGACF	1	0C00	4	
ZEERGINE	01 02	PGACF PGACF	1 1	0D01 0D02	4	
ZEEGSAFE	---	HVON1/HVON2	0/0	0E00	--	
ZEEGUN1M	xx	---	---	12xx	1	c
ZEEGUN2M	xx	---	---	13xx	1	c
ZEEHVONN	00 01 02 03	HVON1/HVON2 HVON1/HVON2 HVON1/HVON2 HVON1/HVON2	0/0 1/0 0/1 1/1	1400 1401 1402 1403	--	
ZEETOGLS	xx	---	---	15xx	--	
ZEEPASSE	AF any other	PW221/PW241 PW222/PW242 PW221/PW241 PW222/PW242	1/0 1/0 0/1 0/1	16AF	7 17	
ZEEMCP1S	xy	HVRF1 HVCR1 HVER1 PRGN1	analog analog analog analog	17xy	8	d d d e

T/C mnemonic	T/C Parameter	affected HK Parameter	HK Par value	LASTC	TC notes	HK notes
ZEEMCP2S	xy	HVRF2 HVCR2 HVER2 PRGN2	analog analog analog analog	18xy	8	d d d e
ZEEGDCOS	xx		---	19xx	2	
ZEELDWIS	xx	---	---	1Axx	1	
ZEEBDHKS	xx	GDCHS VASEC		1Bxx	9	f g
ZEESNHKS	xx	STPS1 STPS2		1Cxx	--	h h
ZEEGDMPS	xx	---	---	1Dxx	--	
ZEESUMOS	xx	SUBMO	xx	1Exx	10	
ZEESIMOS	xx	MODID WRSCM	xx 0	1Fxx	2	i i
ZEECOPYS	xx	---	---	20xx	11	
ZEECSUMS	xx	CHKSU		21xx	12	j
ZEECOPLS	--	---	---	2200	13	
ZEECP2RS	xx	---	---	23xx	14	
ZEEPSW1E	AF any other	PW221/PW241 PW221/PW241	1/0 0/1	40AF	7 17	
ZEEPSW2E	AF any other	PW222/PW242 PW222/PW242	1/0 0/1	41AF	7 17	
ZEERPGAF	---	FPGAi FPGA1 FPGA2	0 0 0	4200		
ZEEMST1S	xy	HVRF1 HVCR1 HVER1 PRGN1	analog analog analog analog	43xy		d d d e
ZEEMST2S	xy	HVRF2 HVCR2 HVER2 PRGN2	analog analog analog analog	44xy		d d d e
ZEEOPTSS	xx	O_SR1/O_SR2	analog	45xx	15	
ZEEGUNSS	xx	G_AN1/G_AN2	analog	46xx	15	
ZEESENSS	xx	BASA1/BASA2 ACPO1/ACPO2 COPO1/COPO2	00 18 FF	47xx	15	
ZEECAT1S	xx	G_BC1	analog	48xx		
ZEECAT2S	xx	G_BC2	analog	49xx		
ZEEPCKMS	xy	PACMO	y AND 0x07	4Axy	16	
ZEESTBQS	00 01 02 04	M8STA M8STA M8STA M8STA	00/-- 01/STBY1 02/STBY2 04/QSCNT	4B00 4B01 4B02 4B04		
ZEEXCRSS	--	---	--	4D00		
ZEERAMWS	xx	---	--	4Exx		
ZEETMCSS	xx	---	--	4Fxx		
ZEEMMDPS	00	---	--	5000		
ZEEINITS	xx	---	--	51xx		
ZEESOBBS	xx	---	--	52xx		
ZEESTBYS	--	---	--	ABCD		
ZEEWWMDS	--	MODID	5	ABCD		
ZEERSSRS	--	---	--	5500		
ZEESBXOS	--	---	--	5600		
ZEEPG2BS	--	FPGA2	1	ABCD		
ZEEPGALS	--	PGACF	3	ABCD		
ZEEDMPSS	--	---	--	----	21	
ZEEDMPHS	--	---	--	----	21	

Telecommand notes:

1. must be followed by command ZEEDATAM xxxx
2. must be preceded by command sequence
   ZEELDEPM xx
   ZEEDATAM xxxx
3. must be followed by yy commands ZEEDATAM xxxx
4. must be preceded by command sequence
   ZEELDEPM 01
   ZEEDATAM 0000
   ZEEGDCOS xx,
   where : xx = 03, if command ZEERGIFE
   xx = 00, 01 or 02, if command ZEERGINE
5. can be preceded by command sequence
   ZEELDEPM xx
   ZEEDATAM xxxx
6. can be preceded by command ZEEGFRSS or ZEEGFFSS
7. Disable plug must be unplugged, otherwise no effect
8. If password not sent, no effect on parameters HVRFx, HVCRx, HVPSx
9. if RAM DATA are to be monitored, use the following sequence:
   ZEELSOPM xx
   ZEEDATAM xxxx
   ZEEBDHKS yy,
   where : yy = C0, to get 16 consecutive RAM locations
   yy = C1, to get one RAM location 16 times
10. possible values for xx: 01, 02, 03, 05, 06, 07
11. must be preceded by command sequence
    ZEELSOPM xx
    ZEEDATAM xxxx
    ZEELDEPM xx
    ZEEDATAM xxxx
12. must be preceded by command sequence
    ZEELSOPM xx
    ZEEDATAM xxxx
    ZEELBYCM xx
    ZEEDATAM xxxx
13. must be preceded by command sequence
    ZEELSOPM xx
    ZEEDATAM xxxx
    ZEELDEPM xx
    ZEEDATAM xxxx
    ZEELBYCM xx
    ZEEDATAM xxxx
14. possible values for xx: 01=pgai, 02=pga1

15. possible values for xx: 01 = GDU1 only
    02 = GDU2 only
    03 = both GDUs
16. if y is not 0 and if SIMO is not a 4ms routine, no action
17. if commands ZEEPASSE, ZEEPSW1E, ZEEPSW2E are sent with any other parameter than AF, the 'command verified counter' HK Parameter CVCNT will not be incremented.
18. no change in HSK tm; SCI tm word 20 becomes $000x, where x indicates that an FGM test is on. x=1 means FGM test;
19. no change in HSK tm; SCI tm word 20 becomes $a76f if both tests have been turned off, otherwise see item 18 above.
20. Forced jump to ROBUST2 invalidates HK data
21. The ROBUST2 TC RBT2S causes SCI and HK tm to become all $A5A5

HK Parameter notes:

a) can affect Parameter MODID

b) can affect almost all HK Parameters, depending on T/C Parameters of preceding commands

c) only for FMs, not for EM

d) the T/C Parameter xx is arbitrary. The following command ZEEDATAM xxxx specifies the GUN current level and affects the HK Parameter G_BC1 (in case of ZEEGUN1M) and G_BC2 (in case of ZEEGUN2M)

e) low nibble y of T/C parameter affects this value

f) high nibble x of T/C parameter affects this value

g) GDCHS = lower six bits of T/C parameter xx

h) VASEC = upper two bits of T/C parameter xx

i) determined by bits 5 and 4 of T/C parameter xx in the following way:
   if Bit 5 = 0
   then STPS1 = Bit 4 (STPS2 not affected)
   if Bit 5 = 1
   then STPS2 = Bit 4 (STPS1 not affected)

j) if jump address specified in preceeding T/Cs is wrong, MODID stays unchanged and WRSCM is set to 1

k) value is the byte checksum of the memory area as specified by the command sequence which has to precede T/C ZEECSUMS (see Telecommand note '12')

A.2 Verification of Frequently Used Telecommand Blocks

Description	T/C sequence	affected HK Parameter	value
Turn on EEPROMs	ZEELDEPM 01 ZEEDATAM 4001 ZEELDBYS 01	EEPON	0
Turn off EEPROMs	ZEELDEPM 01 ZEEDATAM 4001 ZEELDBYS 00	EEPON	1
Turn off start circuit and scratch RAM	ZEELDEPM A0 ZEEDATAM 0000 ZEELDBYS 80	CONPC	decrease

A.3 Verification of GDU System Commands

This can be done by comparing the system commands (CMD17, CMD21 and CMD22) with the HK Parameters GD117, GD121, GD122 for GDU1 and GD217, GD221, GD222 for GDU2, respectively. To do this, the system commands have to be put into HK Telemetry

a) CMD 17

Send the following sequence of Telecommands to put CMD 17 of both GDUs into Telemetry:

```
ZEELSOPM 80
ZEEDATAM 088A
ZEEBDHKS C0
```

Verify, that LASTC = 1BC0 and RAMAD = 80088A
HK Parameters RAMAC_01 and RAMAC_03 then show CMD17 of GDU1 and GDU2, respectively, i.e. RAMAC_01 must have the same contents as GD117 and RAMAC_03 must be equal to GD217.

b) CMD 21

Send the following sequence of Telecommands to put CMD 21 of both GDUs into Telemetry:

```
ZEELSOPM 80
ZEEDATAM 08AA
ZEEBDHKS C0
```

Verify, that LASTC = 1BC0 and RAMAD = 8008AA
HK Parameters RAMAC_01 and RAMAC_03 then show CMD21 of GDU1 and GDU2, respectively, i.e. RAMAC_01 must have the same contents as GD121 and RAMAC_03 must be equal to GD221.

c) CMD 22

Send the following sequence of Telecommands to put CMD 22 of both GDUs into Telemetry:

```
ZEELSOPM 80
ZEEDATAM 08B2
ZEEBDHKS C0
```

Verify, that LASTC = 1BC0 and RAMAD = 8008B2
HK Parameters RAMAC_01 and RAMAC_03 then show CMD22 of GDU1 and GDU2, respectively.
Bits 6 - 14 of RAMAC_01 and GD122 must be equal. The same applies to RAMAC_03 and GD222.

B. Using the Controller

This section describes general operator/EDI system interactions.

1. turn on EEPROM
2. turn on start circuit/turn on scratch RAM
3. a) changing the memory dump address using LDWD
   b) changing the memory dump address using LDBY
4. write protect/unprotect EEPROM
5. upload software
6. absolute jump to manual modes
7. select telemetry submode (1,5,7)
8. select science mode
9. set housekeeping telemetry variable section data
10. display verion control numbers in housekeeping tm
11. select sensor housekeeping options
12. select science telemetry data
13. set interrupt mask
14. set SAFE GDU values
15. set background task
16. compute checksums
17. start dump GDU frame to RAM
18. non-contiguous 16 words in HK variable section
19. BM3 debugging telemetry mode

1. Turn on EEPROM

When the system comes up, the EEPROMs may not be powered up. To turn them on, write $01 to location 14001.

```
ZEELDEPM   01        high byte of power control address
ZEEDATAM   4001      low bytes of power control address
ZEELDBYS   01
```

2. Turn off start circuit/turn on scratch RAM

The EDI controller has a separate startup linear power supply which is used to kick start and power the front end of the DC/DC converter. Once the DC/DC converter is running, a dedicated 5V_AUX supply on the secondary side can be used in place of the startup supply. When the 5V_AUX is operating, the start circuit can be set to run mode, which will reduce the current consumption of the linear supply.

The start/run circuit is controlled by bit 7 at address $A00000 (1=run circuit, 0=start circuit).
Power to the scratch RAM is controlled by bit 0 at address $A00000 (1=RAM power on, 0=RAM power off).

turn off start circuit/turn on scratch RAM

```
ZEELDEPM   A0        high byte of power control address
ZEEDATAM   0000      low bytes of power control address
ZEELDBYS   81        bit 7=1 run circuit (start circuit off)
                     bit 0=1 scratch RAM on
```

turn off start circuit, turn off scratch RAM

```
ZEELDEPM   A0        high byte of power control address
ZEEDATAM   0000      low bytes of power control address
ZEELDBYS   80        bit 7=1 run circuit (start circuit off)
                     bit 0=0 scratch RAM off
```

3.

a) Changing the memory dump address using LDWD

RAM location $D470 holds the address of the first word of memory to be dumped in BM1 or BM3. It is initialized to $0000 0000. To change the memory dumped in telemetry, put a different address at $D470. For instance, to display the current value the destination pointer variable, set $D470 to $D310.

```
ZEELDEPM   00        high byte of memory dump pointer address
ZEEDATAM   D470      low bytes of memory dump pointer address
ZEELDWDS   02
ZEEDATAM   D310      low word of address of destination pointer (DP
                     address is automatically incremented one word after
                     command)
ZEEDATAM   0000      high word of address of destination pointer
```

b) Changing the memory dump address using LBDY

This example has the same effect as the commands shown above.

```
ZEELDEPM   00        high byte of memory dump pointer address
ZEEDATAM   D470      low bytes of memory dump pointer address
ZEELDBYS   10        byte 0 of destination pointer address (DP
                     address is automatically incremented one byte
                     after command)
ZEELDBYS   D3        byte 1
ZEELDBYS   00        byte 2
ZEELDBYS   00        byte 3
```

4. Write protect/write enable the EEPROMs

The EEPROM write enable state is **toggled** by sending low power command 2, ZEDIPL2. This can not be done by the EDI onboard software. However, the software can **turn off** the write enable by writing anything to any even address in the PGA address ranges.

write-disable EEPROMs by writing to $1C000

```
ZEELDEPM   01        high byte of write protect address
ZEEDATAM   C000      low bytes of write protect address
ZEELDBYS   00
```

5. Upload software

There are a series of steps required to upload new software described elsewhere in this manual.

6. Absolute jump to manual modes

To jump to a mode explicitly (not using the SIMOS TC), e.g., to Mode 8:

```
ZEELDEPM   00        high byte of EEPROM entry point address
ZEEDATAM   6006      low bytes of EEPROM entry point address
ZEEJDEPS   66        checksum of jump address
```

To jump to Mode 0:

```
ZEELDEPM   00        high byte of EEPROM entry point address
ZEEDATAM   02F0      low bytes of EEPROM entry point address
ZEEJDEPS   F2        checksum of jump address
```

To jump to power-on Mode 8 location:

```
ZEELDEPM   00        high byte of EEPROM entry point address
ZEEDATAM   6000      low bytes of EEPROM entry point address
ZEEJDEPS   60        checksum of jump address
```

7. Select telemetry submode

The system initializes itself in telemetry nominal mode 1. To change telemetry modes use the submode telecommand:

```
ZEESUMOS  05        change to telemetry submode 05 valid modes are 1,5,7.
```

Before selecting submode 07 (bm3) the operator should turn on Scratch RAM and put something in it using the GDMP telecommand.

8. Select science mode

The science mode defines the operation of the EDI experiment. To change science modes, enter the start address of the mode in destination pointer, and then jump to that code by sending the change science mode command. The science mode selection must be preceded by the the appropriate Init Table selection.

select science mode 1 (GEOS)

```
ZEELDEPM   02        high byte of science mode entry point address
ZEEDATAM   455D      low bytes of science mode entry point address
ZEESIMOS   01        select science mode 1
```

9. Set housekeeping telemetry variable section data

Housekeeping telemetry contains a 16 word variable section. The 16 words contain GDU data by default. The operator can select different data with the housekeeping telemetry telecommand.

select a particular GDU and GDU group/channel

```
ZEEBDHKS   8C        GDU2, group/channel 1/4 (each field is 3 bits)
ZEEBDHKS   4F        GDU1, group/channel 1/7
```

select EVT data--word 2 (GDU1, word 30)

```
ZEEBDHKS   C4        EVT data, word 2
```

select variable RAM data--1 location 16 times

```
ZEELSOPM   00        high byte of source address
ZEEDATAM   D4E8      low word of source address
ZEEBDHKS   C1        RAM data, 16 times
```

select variable RAM data--16 locations

```
ZEELSOPM   00        high byte of source address
ZEEDATAM   7FFA      low word of source address
ZEEBDHKS   C0        RAM data, 16 locations
```

select default GDU channels

```
ZEEBDHKS   00        default GDU channels
```

NOTE : changes can be made for only one option in any one execution of the ZEEBDHKS command.

10. Display version numbers in housekeeping telemetry

To display version numbers in telemetry, set the source pointer address to the appropriate version control number address and send telecommand ZEEBDHKS.

Firmware/software version numbers are stored minus 6 from the end of each PROM or EEPROM.

```
PROM            0000 1FFA
EEPROM MODE8    0000 7FFA
EEPROM HIMOD    0002 7FFA
```

PGAi , PGA1 version numbers are stored in the odd bytes at the beginning of the PGAi, 1 EEPROMs.

```
PGA1            0001 C000
PGAi            0007 C000
```

To show the PROM version number:

```
ZEELSOPM   00        high byte of memory dump pointer address
ZEEDATAM   1FFA      low bytes of memory dump pointer address
ZEEBDHKS   C0
```

When these commands are processed (may take up to 5 seconds), telemetry will show the version number in the first two words of the variable section.

11. Select sensor HK

Sensor status is reported in housekeeping telemetry. By default the words contain a different channel each TMR, and it takes 8 TMRs to step through all eight channels. The operator may select an unchanging channel with the select sensor housekeeping telecommand.

```
ZEESNHKS   0C        select sensor 1, MUX 4
ZEESNHKS   29        select sensor 2, MUX 1
```

NOTE : changes can be made for only one sensor in any one execution of the ZEESNHKS command.

12. Select science telemetry data for PACMO=0 formats

For telemetry submode .1 : for channel 0, select GDU1, word 28 with compression

```
ZEELDEPM   80
ZEEDATAM   0078
ZEETMCSS   05
```

for channel 1, select GDU1, word 23 with accumulation

```
ZEELDEPM   80
ZEEDATAM   0064
ZEETMCSS   0A
```

For telemetry submode .5 : for channel 2, select GDU2, word 11

```
ZEELDEPM   80
ZEEDATAM   0036
ZEETMCSS   20
```

Notes : ch2 is the high byte of the BM1 ch2,3 data word. Option byte (LB0) of TMCSS: 7654 3210

7 = channel 4	3 = channel 1
6 = channel 3	2 = channel 0
5 = channel 2	1 = compress (channel 0,2, or 3)
4 = not used	0 = accumulate (channel 1)

13. Set interrupt mask

The EDI is an interrupt driven system. A byte in RAM memory controls which interrupts are active. When the byte is $00, no vectorred interrupts are recognized. The byte is normally set to $BF.

```
ZEEIMSKS  00        disable all interrupts
ZEEIMSKS  BF        enable all interrupts
```

The interrupt to bit map is:

```
bit 0 --  GDU                  bit 4 --  N/A
bit 1 -- sun pulse (~1.5 s)    bit 5 --  fgm (16 ms)
bit 2 --  tmr (4.0 s)          bit 6 --  a/d converter
bit 3 --  rti (10 ms)          bit 7 --  ati (nom. 62.5 ms)
```

14. Set GDU commands to safe values

To set GDU commands to safe voltage values, use the SAFE command.

```
ZEEGSAFE            no arguments
```

15. Set background tasks

The EDI software is designed to run with time left over between interrupts. This time is available to do various background activities.

```
ZEELDEPM   00        high byte of RAM test entry point
ZEEDATAM   0473      low word of RAM test task entry point
ZEEBKGDS   77        set DP address as background task if checksum
                     matches address checksum
```

16. Compute checksum

Checksums can be used to verify correct versions of code or tables or to check that memory has not been corrupted. To compute a checksum specify the start address, the number of bytes to sum, and the expected checksum.

to compute a checksum on ROM0

```
ZEELSOPM   00        high byte of start address
ZEEDATAM   0000      low word of start address
ZEELBYCM   1000      byte count
ZEECSUMS   4A        expected checksum
```

17. Start dump of GDU frame to RAM

This command is used before selecting burst mode 3. This command causes a selected frame of GDU commands and data to be written to RAM. Before issuing this command, scratch RAM must be turned on.

```
ZEEGDMPS   03        dump frame 3 from both GDUs to scratch RAM
ZEEGDMPS   1x        dump frame x from GDU 1
ZEEGDMPS   2x        dump frame x from GDU 2
```

18. Non-contiguous 16 words in HSK variable section

In science modes only, the HSK variable section may be used to display a user-selected, non-contiguous 16 words. To use this option, load an icf file like WORDHSK.ICF to define the address of the selected 16 words. Send telecommands:

```
ZEELDEPM   00
ZEEDATAM   D6EF
ZEELDBYS   01
ZEEBDHKS   C1
```

19. BM3 debugging telemetry mode

This mode writes user-selected 16 words to Scratch-RAM every BCI.

Create and execute an ICF file to define the addresses of the 16 words. Send Telecommand ZEEGDMPS 40. Scratch RAM will be filled in 64 seconds (for a 2ms BCI). The data can be dumped to telemetry using

ZEESUMOS 07.

C. GDU Commands & Data

C.1 Command & Data Frames

The base address for GDU DATA is X'800000'. There are 8 or 15 data frames of GDU DATA. In the table below in the DATA Addr column, fr means frame. The start address is the frame number (0-15) times X'80' plus X'800000'; thus frame 5 start address is X'800280' and for frame C is X'800600'. The first data word in each frame is the frame start address + 8.

The base address for GDU CMDs is X'800800'. Each CMD has an associated INCRement. In practice, increments are non-zero only for CMDs 1,2 and 16. The increment word address is the CMD Addr minus 2; thus the INCR Addr for GDU1 CMD1 is X'800808' and the INCR Addr for GDU2 CMD2 is X'800814'.

CMD1 Addr	CMD2 Addr	CMD #	CMD abbrev.	CMD Name	Bits	DATA1 Addr	DATA2 Addr	DATA Name	Bits
02	06	0	OPT_AU	Upper Injector	12	fr+08	fr+0A	Correlator Accu 1	14
0A	0E	1	GUN_XD	Deflection	12	fr+0C	fr+0E	Correlator Accu 2	14
12	16	2	GUN_YD	Deflection	12	fr+10	fr+12	Correlator Accu 3	14
1A	1E	3	GUN_BC	Beam Current	12	fr+14	fr+16	Correlator Accu 4	14
22	26	4	GUN_AN	Anode	12	fr+18	fr+1A	Correlator Accu 5	14
2A	2E	5	GUN_FC	Focus	12	fr+1C	fr+1E	Correlator Accu 6	14
32	36	6	GUN_BW	Beam Width	12	fr+20	fr+22	Correlator Accu 7	14
3A	3E	7	OPT_OA	Outer Analyzer	12	fr+24	fr+26	Correlator Accu 8	14
42	46	8	SEN_SR	Sensor Reference	12	fr+28	fr+2A	Correlator Accu 9	14
4A	4E	9	OPT_IA	Inner Analyzer	12	fr+2C	fr+2E	Correlator Accu 10	14
52	56	10	OPT_ET	Extractor	12	fr+30	fr+32	Correlator Accu 11	14
5A	5E	11	OPT_SP	Suppressor	12	fr+34	fr+36	Correlator Accu 12	14
62	66	12	OPT_RC	Retainer Cone	12	fr+38	fr+3A	Correlator Accu 13	14
6A	6E	13	OPT_EL	Lower Deflector	12	fr+3C	fr+3E	Correlator Accu 14	14
72	76	14	OPT_AL	Lower Injector	12	fr+40	fr+42	Correlator Accu 15	14
7A	7E	15	OPT_EU	Upper Deflector	12	fr+44	fr+46	Reset frame counter	14
82	86	16		Frame Counter	12	fr+48	fr+4A	Correl Max, Addr	16
8A	8E	17		Correlator Control 1	15	fr+4C	fr+4E	Correl Status	16
92	96	18		Split beam Delay	15	fr+50	fr+52	Delta Gyrotime	16
9A	9E	19		Correlator Control 2	15	fr+54	fr+56	Spare (TS)	16
A2	A6	20		Logic Cell Array	8	fr+58	fr+5A	Logic Cell Arr Readout	8
AA	AE	21		GDU Sys Command 1	15	fr+5C	fr+5E	Status Sys Command 1	16
B2	B6	22		GDU Sys Command 2	15	fr+60	fr+62	Status Sys Command 2	16
BA	BE	23		Sensor Command 1	15	fr+64	fr+66	FR_C (upper) FRHK	16
C2	C6	24		Sensor Command 2	12	fr+68	fr+6A	Sensor Status	12
CA	CE	25		Sensor Command 3	12	fr+6C	r+6E	Sensor Status	12
D2	D6	26		Sensor Command 4	12	fr+70	fr+72	Sensor Status	12
DA	DE	27		Spare (TSIM)	12	fr+74	fr+76	Sensor Status	12
E2	E6	28		Split beam Offsets 1	15	fr+78	fr+7A	GDU HK Voltages	14
EA	EE	29		Split beam Offsets 2	15	fr+7C	fr+7E	Accumulator 1	12
F2	F6	30		Sync X'8001' or X'89AF' (enable HV)	16	fr+80	fr+82	Accumulator 2	12
FA	FE	31		Sync X'0000'	16	fr+84	fr+86	Accumulator DIF	16

C.2 RAM Addresses for GDU Increments & Commands

(offset from $80 0800)

	GDU1		GDU2			GDU1		GDU2	
#	INCR	CMD	INCR	CMD	#	INCR	CMD	INCR	CMD
0	0	2	4	6	16	80	82	84	86
1	8	A	C	E	17	88	8A	8C	8E
2	10	12	14	16	18	90	92	94	96
3	18	1A	1C	1E	19	98	9A	9C	9E
4	20	22	24	26	20	A0	A2	A4	A6
5	28	2A	2C	2E	21	A8	AA	AC	AE
6	30	32	34	36	22	B0	B2	B4	B6
7	38	3A	3C	3E	23	B8	BA	BC	BE
8	40	42	44	46	24	C0	C2	C4	C6
9	48	4A	4C	4E	25	C8	CA	CC	CE
10	50	52	54	56	26	D0	D2	D4	D6
11	58	5A	5C	5E	27	D8	DA	DC	DE
12	60	62	64	66	28	E0	E2	E4	E6
13	68	6A	6C	6E	29	E8	EA	EC	EE
14	70	72	74	76	30	F0	F2	F4	F6
15	78	7A	7C	7E	31	F8	FA	FC	FE

C.3 RAM Addresses for GDU Data Words

Offset from $80 0000;
For Frame n: add $80*n to Data Word Address.
Equation for address: (8 + frame * $80 + # * 4 (+2 if GDU2)) AND $7FF)

For Frame = 0:

#	GDU1	GDU2	#	GDU1	GDU2
0	8	A	16	48	4A
1	C	E	17	4C	4E
2	10	12	18	50	52
3	14	16	19	54	56
4	18	1A	20	58	5A
5	1C	1E	21	5C	5E
6	20	22	22	60	62
7	24	26	23	64	66
8	28	2A	24	68	6A
9	2C	2E	25	6C	6E
10	30	32	26	70	72
11	34	36	27	74	76
12	38	3A	28	78	7A
13	3C	3E	29	7C	7E
14	40	42	30	80	82
15	44	46	31	84	86

C.4 GDU Command Description

Bit 15 shall be high for all words except word 31. This will be done by hardware in the CONTROLLER).

Digital Commands to be converted to Analog Reference Voltages:CMD 0 -- OPT_AU -- Optics Upper Injector Voltages

These voltages depend linearly on selected beam energy and nonlinearly on required beam deflection.

Scaling : +1V/dig; max value : 3500

CMD 1 & 2 -- GUN_XD & GUN_YD -- Gun Deflection Voltages

Depend linearly on selected beam energy and non-linearly on required beam deflection. The CONTROLLER must multiply these by the current beam energy.

The tabulated references represent (Ua+Ud), where Ud is the deflection voltage, either Vx or Vy, which can be positive or negative but always |Ud|<Ua. Ua is the anode voltage (see below).

For zero deflection, the number must equal GUN_AN's value and corresponds to Ua.

Range : From 0 up to 'GUN_AN' digits
Scaling : 1V/digit

A positive X-deflection voltage will move the beam direction to the positive X axis (S/C spin axis) which is aligned with the +X axis of the instrument. A positive Y-deflection will move the beam in spin direction which shall be aligned with the Y direction of the instrument. (For a definition of the instrument axes, see EID/B.)

CMD 3 -- GUN_BC -- Beam Current

The beam current may be varied between zero and appr. 4uA. Zero current is obtained by setting all bits to 0.

Scaling : Table TBD; max value: 4095

CMD 4 -- GUN_AN -- Anode Voltage

The Gun anode voltage is 1.70 times the Cathode voltage CA. Hardware will convert GUN AN to the negative reference voltage CA. Beam Energy is GUN_AN/1.7.
Max value corresponds an anode voltage of 1.7kV and a cathode voltage of -1kV.

Scaling : +0.5V/dig; max value: 3400.

CMD 5 -- GUN_FC -- Focus Voltage

This voltage defines where the crossover in the beam (Focus) is situated in the deflection system. The reference will be a linear function of the energy and a nonlinear function of the deflection from the axis of the gun.

Scaling : 0.3125V/dig; max value: 1000

CMD 6 -- GUN_BW -- Beam Aperture (Wehnelt)

It defines primarily the aperture of the electron bundle entering from the cathode into the optical system of the gun and finally the aperture of the outer beam. The aperture may vary also with the deflection from the axis of the gun.

Scaling : -0.0124V/dig; max value: 4095

Offsets, maximum and minimum for Optics Commands:
Fixed offsets for OPTICS command scaling

f_opt_OA	$fc18	1keV: -1000 for Optics CMDs 7,9,12	OA,IA,RC
f_opt_SR	$f830	1keV: -2000 for Optics CMDs 8,10,11	SR,ET,SP
f_opth_OA	$fc18/2	500eV: -1000/2 for Optics CMDs 7,9,12	OA,IA,RC
f_opth_SR	$f830/2	500eV: -2000/2 for Optics CMDs 8,10,11	SR,ET,SP

OPTICS max and min

mx_opt_OA	$0fff	4095 = max for Optics CMDs 7,9,10,11,12	OA
mx_opt_SR	$0bb8	3000 = max for Optics CMD 8	SR
mx_opt_EL	$0c17	3095 = max for Optics CMDs 00,13,14,15	AU,EU,EL,AL

OPTICS min

mn_opt_	0	0 = min for all except CMD 8	SR
mn_opt_SR	$03e8	1000 = min for CMD 8	

CMD 7 -- OPT_OA -- Outer Analyzer Voltage

Scaling : 1V/dig; offset: -1kV; max value 4095

CMD 8 -- SEN_SR -- Sensor Voltage Reference

Scaling : 1V/dig; offset: -2kV; max value 4095

CMD 9 -- OPT_IA -- Inner Analyzer Voltage

Scaling : 1V/dig; offset: -1kV; max value 4095

CMD 10 -- OPT_ET -- Extractor Voltage

Scaling : 1V/dig; offset: -2kV; max value 4095

CMD 11 -- OPT_SP -- Suppressor Voltage

Scaling : 1V/dig; offset: -2kV; max value 4095

CMD 12 -- OPT_RC -- Retainer Cone Reference

Depends nonlinearly on polar angle.

Scaling : 1V/dig; offset: -1kV; max value 4095

CMD 13 & 15 -- OPT_EL & OPT_EU -- Optics Lower & Upper Deflector Voltages

These voltages depend linearly on selected beam energy and nonlinearly on required beam deflection.

Scaling : +1V/dig; max value: 3000

CMD 14 -- OPT_AL -- Optics Lower Injector Voltages

Same as for OPT_EL and OPT_EU

Scaling : +1V/dig; max value: 3500

In case the Voltage Frequency Converters are on (FON high), the reference EU defines the frequency in the background channel and AL the frequency in the signal channel of the Test Pulse Generator, respectively.

Digital Commands:

CMD 16 -- Frame Counter

This 12-bit-counter is automatically incremented by the CONTROLLER hardware if the increment of the channel is set to 1. Occasionally, it must be verified by software that the counter status sent to both GDUs is identical.

CMD 17 -- Correlator Command 1 [-eee nnnd ggnp mmsr]

BIT 0 RSS -- correlator reset

Toggling this bit causes reset of all correlator counters and of the reset-frame-counter DATA 15 immediately after next readout in the next frame

BIT 1 SETC -- set code start

Toggling this bit starts the following operations, when the 2 LSBs of CMD 16 match CMD 18 bits FR_ST[1:0]:

- reset gun-code, correl.-code and DATA 18
- load CMD 19 bits 7-0 into down counter
- start gun-code and down-counter at the same time
- start correl.-code when down-counter reaches 0

BIT 3-2 CR[1:0] -- code-clock frequency

The final code clock frequency is

$$f(code-clock) = \frac{2^{23}}{m \times n}$$

CR1	CR0	m
0	0	16
0	1	8
1	0	4
1	1	2

The setting of these bits becomes valid, when the 2 LSBs of CMD 16 match CMD 19 bits FR_ST[1:0]:

BIT 4 EN_PGA1_CD

1 enables link for code of correlator #7 (auto-track correlator) to clock enable of inputs of split-beam accumulators in PGA1
0 clock-enable inputs of split-beam accumulators in PGA1 enabled continuously

BIT 5 NOC -- No code

0 codes are active
1 gun-beam continuously 'on' all correlator counters continuously enabled

BIT 6 SEL_GC -- select gun code

0 gun-code type A (GDU1)
1 gun-code type B (GDU2)

BIT 7 SEL_CC -- select correlator code

0 correlator-code type A (GDU2)
1 correlator-code type B (GDU1)

Notes : Setting becomes active only if the code-start command procedure runs after change of this bit

Bits 7-6 are set in the controller: GDU1 detector1/gun2 b'10' X'80'
GDU2 detector2/gun1 b'01' X'40'

BIT 8 DRI_SE -- select drift sensitivity

0 The sensitivity setting, sent in CMD 17 bits 14-12 of the same GDU frame is stored inside the PGA and is used as tracking sensitivity
1 The sensitivity setting, sent in CMD17 bits 14-12 of the same GDU frame is stored inside the PGA and is used as drift speed

BIT 11-9 DT[2:0] -- Sets frequency of clock CLK_DT

Determines code-shift time-increment delta_t

DT 210	n	CLK_DT [Hz]	[~MHz]	[sec]	DeltaT [~ns]
000	1	2^23	8	1/(2^24)	60
001	2	2^22	4	1/(2^23)	120 (119)
010	4	2^21	2	1/(2^22)	240 (238)
011	8	2^20	1	1/(2^21)	500 (476)
100	8	2^20	1	1/(2^21)	500 (476)
101	16	2^19	0.5	1/(2^20)	1000 (954)
110	32	2^18	0.25	1/(2^19)	2000 (1907)
111	64	2^17	0.125	1/(2^18)	4000 (3815)

BIT 14-12SE[2:0] – Time tracking loop-sensivity

Number s of sensor-pulses nessessary to change correlator-code delay by one increment of

$$\frac{1}{\left(2 \times f\left(CLK\_DT\right)\right)}$$

SE			
2	**1**	**0**	**n**
0	0	0	no code-delay variation
0	0	1	3
0	1	0	5
0	1	1	9
1	0	0	17
1	0	1	33
1	1	0	65
1	1	1	129

CMD 18 -- Correlator Command

BIT 13-0 GY[13:0]

Initial correlator-code delay for correlator #7 against gun-code in units of $\frac{1}{f(CLK\_DT)}$;

This replaces CMD19 BIT 7-0 for correlator B versions later than January 2000.
Note: CMD18 is also decoded in PGA1 and used for the delay of the split beam accumulators. The split beam accumulator delay will not work correctly since a different clock is used for delay count-down. Fixing this requires a change to PGA1.

CMD 19 -- Correlator Command [-ffd drrr c--- ----]

BIT 6-0 not used
BIT 7 short/long code selection
0 = short code (15 chips PNC)
1 = long code (127 chips PNC)

BIT 10-8 FR_CLR[2:0]

Automatic correlator reset after readout every f GDU-frames. Reset and readout of latest data before reset occur in frame r

FR_CLR			**F**	**R**
0	0	0	no automatic reset	
0	0	1	1	
0	1	0	2	1,3,5,7,...
0	1	1	4	3,7,11,15,...
1	0	0	8	7,15,..
1	0	1	16	15,...

BIT 12-11 DRI

DRI1	**DRI0**	
0	0	code shift controlled by time tracking loop
		continuous (not code dependent) variation of correlator code delay, if the maximum counts are not found in correlators 6,7,8 delay increasing, if most counts in corr. #9...#14 delay decreasing, if most counts in corr. #5...#0 Additional condition for switching from tracking to drift-mode is the content of the correlator holding the most counts Enable the drift, if MAX >= n
0	1	1
1	0	8
1	1	64

BIT 14-13 FR_ST[1:0]

Frame number at which code starts. If the 2 LSBs of CMD 16 match these bits, CMD 17 SETC starts working

CMD 20 -- Configuration Data of FPGA2

Only lower 8 bits used, one Byte of configuration data. Must be prepared by a reset command (CMD 21(8)). Consecutive Bytes must be sent in consecutive Frames without interruption until the whole program is transmitted. There is a bit in HK (bit15,DATA22) indicating whether or not FPGA2 is configured.

CMD 21 -- GDE System Commands 1

BIT 14 -- SYOF

Disables the synchronisation of the low voltage converter in order to check the frequency when free running (1=disable).

BIT 13 -- HVON

Enables High Voltages of OPTICS and GUN. If low, the cascades are off, all HV amplifiesr are unpowered, and reference voltages are forced to zero. If high (but without a valid password), the voltages and the beam current are limited to a safe levels for tests in air.

BIT 12 -- <<spare>>

BIT 11 -- INOF

Switches off initial supply of low voltage converters in order to save power (1=off).

BIT 10 -- R_C

Reset accumulators 1 and 2 and the differential frame counter FR_C after next readout (toggle)

BIT 9 -- RHK

Reset HK counter and the differential frame counter FR_HK and switch to the selected multiplexer channel, all after the next readout (toggle)

BIT 8 -- PG2 -- Start/hold configuration of FPGA2

If low, FPGA2 is cleared and waits for new configuration. To configure FPGA2, the first configuration byte has to be in CMD 20 of the current frame when PG2 goes high. See also bits 3,2 of CMD22

BIT 7 -- RDC -- Start readout of selected configuration

Must also be high during configuration of FPGA2.

BIT 6 -- RP2 -- Select FPGA for readout

If low, FPGA1 is selected for readout.
If high, FPGA2 is selected for readout or configuration.

BIT 5-3 -- HG4, HG2, HG1 -- Select Group of HK channels

If 7, the power converter frequency will be transferred.

BIT 2-0 -- HK4, HK2, HK1 -- Select HK channel in HK channel group

CMD 22 -- GDE System Commands 2

BIT 14 -- CINV –

Toggling this bit changes the state of the bistable modulator in case the correlator in FPGA2 is not available. This way one can always force the beam chopper on. If high, the beam modulation is inverted. Therefore CINV must be low during normal operation.

BIT 13 -- TON -- Test pulser on

If high, test pulses are routed to SENSOR.

BIT 12 -- TPN -- Test pulse stream

If high, the pulse stream of the test generators are pseudo noise strings of 255 chips otherwise series of equidistant pulses.

BIT 11-10 -- SOU2, SOU1 -- Select source of input EVTS

SOU2	SOU1	
0	0	No Input
0	1	EVT1 from SENSOR. EVT2 connected to accumulator 1
1	0	EVT2 from SENSOR. EVT1 connected to accumulator 1
1	1	Test Pulses

BIT 9 -- Spare

BIT 8-7 -- SP2, SP1 -- Beam splitting

SP2	SP1	Beam Splitting
0	0	Switch every frame
0	1	Switch every second frame
1	0	Switch every fourth frame
1	1	Switch every eights frame

BIT 6 -- FON -- Test generator power on

Switches power for the voltage frequency converters of the Test Generator on.

BIT 5 – ARES

If high, the accumulators read out into DATA29 and DATA30 will be automatically reset every eight frames (after the readout in frame 7).
This feature was added to PGA1 on August 26, 1997.

BIT 4 <<spare>>

BIT 3-2 -- PG2_2, PG2_1

Added on December 16, 1997
These bits have been added to prevent clearing of FPGA2 while going through the radiation belts. Clearing of FPGA2 now requires a distinct setting of three bits:

bit 8 of CMD21 and
bits 3,2 of CMD22.

In order to clear the configuration of FPGA2, these bits have to be set to 1 (bit3) and 0 (bit2),

respectively, in addition to setting bit 8 of CMD21 to 0. After successful configuration of FPGA2, these bits should be set to 0 (bit 3) and 1 (bit2), respectively.

BIT 1-0 <<spare>>

CMD 23-26 -- Sensor Commands

May be transmitted to the GDU in each frame, but are transmitted to the SENSOR only when requested by the CONTROLLER, by toggling bit 12 in CMD 23. Only 12LSB are used.

CMD 23 -- Sensor Commands 1

Bit			
15	---		
14	---		
13	---		
12	Toggle bit		
11	Sensor Cmd MSB (Must be 0)		
10	MX2		Note 1
9	MX1		
8	MX0	Instruction	
7	Spare	Register	
6	EXE IMMED		Note 2
5	SWAP ACCUM/CORR		Note 3
4	TST PLSR ENA		Note 4
3	AP7	Accumulator	
2	AP6	Pointer	
1	AP5	Field	
0	AP4		

CMD 24 -- Sensor Commands 2

Bit			
15	---		
14	---		
13	---		
12	---		
11	AP3	Accumulator	Note 5
10	AP2	Pointer	
9	AP1	Field	
8	AP0		
7	REVRS LOOK (R0)		Note 6
6	BA6 (S3) Segment		
5	BA5 (S2)		
4	BA4 (S1)	Base	Note 7
3	BA3 (S0)	Address	
2	BA2 (E2) Element		
1	BA1 (E1)		
0	BA0 (E0)		

CMD 25 -- Sensor Commands 3

Bit			
15	---		
14	---		
13	---		
12	---		
11	CP7		
10	CP6		
9	CP5	Correlator	
8	CP4	Pointer	Note 8
7	CP3	Field	
6	CP2		
5	CP1		
4	CP0		
3	PWST EN		Note 9
2	DT2	Pmp & Dead	
1	DT1	Time Control	Note 10
0	DT0		

CMD 26 -- Sensor Commands 4 ("Critical")

Bit			
15	---		
14	---		
13	---		
12	---		
11	Spare		
10	A2	Preamp	
9	A1	Gain	Note 11
8	A0		
7	Spare		
6	Spare		
5	Spare		
4	Spare		
3	Spare		
2	HV2	MCP	Note 12
1	HV1	HV	
0	HV0		

CMD 27 -- <<Spare>>

Used by Tracking Simulator

Bits 6-0	cts/frame	cts/sec
0	0	0
1	1	4096
2	8	32,768
3	32	131K
4	128	524K
5	256	1M

CMD 28 -- Beam Split Offsets 1

BIT 14 -- <<spare>>
BIT 13-11 -- Detector Elevation Offset for Beam Direction 1
BIT 10-8 -- Detector Elevation Offset for Beam Direction 2

One digit causes about half a degree Elevation change. (The Sensor Azimuth field can be made sufficiently wide to accept all 4 beam directions)

BIT 7-4 -- Gun Offset Ya
BIT 3-0 -- Gun Offset Xa

The offset bits will be added by the GDU to XD and YD at such a bit location that the direction of the beam is changed by about 0.3 degree/step, eg. bits 3-0 of CMD 28 are added to bits 4-1 of GUN_XD. Maximal offset is about 5 degrees, but depends on X and Y (see also CMD 22(8-7)).
The main direction of the electron beam is given by the voltages X and Y in CMDs 1 and 2. Relative to that the beam may be directed in four directions in sequence:

X0 = X+Xa	Y0 = Y+Ya-Yb
X1 = X+Xa-Xb	Y1 = Y+Ya
X2 = X-Xa	Y2 = Y-Ya+Yb
X3 = X-Xa+Xb	Y3 = Y-Ya.

Xa, Xb, Ya and Yb are the offsets (all >0) given by CMDs 28 and 29. To turn the beam-split off, all offsets must be set to 0. The switching speed is set by the CMD SP described above.

CMD 29 -- Beam Split Offsets 2

BIT 14 -- <<spare>>
BIT 13-11 -- Detector Elevation Offset for Beam Direction 3
BIT 10-8 -- Detector Elevation Offset for Beam Direction 4
BIT 7-4 -- Gun Offset Yb
BIT 3-0 -- Gun Offset Xb

CMD 30 -- Frame Synchronization/Password

First and last bit of this word must be high for synchronisation purposes. The other bits will be used as a password. As long as the password is not valid all High voltages and the cathode power are limited to safe values and the MCP HV is zero.
The password, including the leading and trailing 1's, is 89AFH. When the instrument is in air, send always 8001H.

Used also by Tracking Simulator

CMD 31 – Synchronization --- All bits of this word must be low.

C.4.1 SENSOR COMMAND NOTES

(1) ANALOG STATUS MUX CHANNEL (3 bits, MX0-MX2, MX0=LSB)
Determines which of the digitized analog monitors are selected, according to the following table:

Channel 0 = +2.5V Ref	::	Channel 4 = Preamp Gain
Channel 1 = HVPS Curr	::	Channel 5 = -5V LVPS
Channel 2 = HVPS Error	::	Channel 6 = +5V LVPS
Channel 3 = HVPS Ref	::	Channel 7 = Temp Mon

(2) EXECUTE IMMEDIATE
This bit must be set to 1 for the command to be executed upon receipt by the Sensor.

(3) SWAP ACCUMULTATOR/CORRELATOR
When set to 1, the outputs of the Sensor Correlator and Accumulator Channels are swapped with each other.

(4) TEST PULSER ENABLE
When set to 1, the Sensor test pulser is enabled, and the Sensor will accept test pulse inputs from the GDE.

(5) ACCUMULATOR POINTER FIELD (AP0-AP7)
Selects which of the 8 anodes in the current Sensor "Anode Field" are routed to the Accumulator data channel. The "LSB" end of the bit-mapped Pointer Field corresponds to the anode specified by the Anode Base Address. The 7 remaining bits map to the 7 anodes adjacent to the Base Address anode, moving in a counter clockwise direction when viewing the anode array from the top (from the MCP side). Any combination of the 8 anodes in the Anode Field may be selected (including none or all). Anodes may be simultaneously routed to both the Accumulator and Correlator data channels.

(6) REVERSE LOOK
Setting this bit shifts the active anode field by 180 degrees. It is equivalent to shifting the Anode Base Address by 64.

(7) ANODE BASE ADDRESS (BA0 - BA6; E0-E2 & S0-S3)
Specifies the edge of the 8-anode field that is available for use by the data channels. The anode field is composed of the 8 contiguous anodes beginning with the base address and moving counter-clockwise when the anode array is viewed from the MCP side. When Base Address = 0, the 8-anode field is bordered by the instrument -Xu axis, and is located immediately counterclockwise of the -Xu axis. Higher Base Address positions are offset by the appropriate number of anodes in a counter clockwise direction from this Base Address = 0 location.

(8) CORRELATOR POINTER FIELD (CP0 - CP7)
Selects which of the 8 anodes in the anode field is routed to the Correlator channel, in the same fashion as the Accumulator Channel described above.

(9) POWER STROBING ENABLE (PWST EN)
When power strobing is enabled (PWST EN =1), then only those preamps that are selected by either the Accumulator Pointer Field or the Correlator Pointer Field are powered. When PWST = 0, all preamps are continuously powered, independent of the pointer fields.

(10) PREAMP DEADTIME CONTROL (DT0 - DT2)

Sets the deadtime that is applied to disable counting during the Sensor commanding interval. This provides the ability to mask out noise counts that may be associated with Sensor commanding.

Code	Inhibit time	Inhibit Description (both Events channels)
000	8 uS	Settling time only
001	25 uS	Multiplexor loading + settling time
010	66 uS	Sensor command cycle + settling time
011	8 uS	Exe gate external (Engineering Test only) + settling time
100	32 uS	Settling time only
101	49 uS	Multiplexor loading + settling time
110	90 uS	Sensor command cycle + settling time
111	32 uS	Exe gate external (Engineering Test only) + settling time

The inhibit description defines the components of the inhibit duration. Note that there are two options for settling time, 8uS and 32uS in combination with other inhibiting options during the sensor command cycle.

(11) PREAMP GAIN

Sets the gain of the preamps. The 8 gain levels will be spaced quasi-logarithmically, to cover a dynamic range of TBD (about 75). Note that high gain (Preamp Gain = 7) corresponds to low threshold.

(12) MCP HIGH VOLTAGE (HV0 - HV2)

Sets the MCP bias voltage. The levels are currently set to the following nominal values.

Step 0 = 0V	:	Step 4 = 2125 V
Step 1 = 1250 V	:	Step 5 = 2200 V
Step 2 = 1975 V	:	Step 6 = 2280 V
Step 3 = 2050 V	:	Step 7 = 2360 V

These are the total voltages supplied by the HV stack, including: the acceleration voltage between the MCP output and the anode array, the MCP stack, and the matching voltage between the MCP input and the HV stack (equal to the acceleration voltage).

C.5 GDU Data Description

DATA 0	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 0
DATA 1	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 1
DATA 2	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 2
DATA 3	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 3
DATA 4	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 4
DATA 5	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 5
DATA 6	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 6
DATA 7	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 7
DATA 8	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 8
DATA 9	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 9 Option B only
DATA 10	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 10 "
DATA 11	Bits 15-14	not used
	Bits 13-0	Correlator Accumulator 11 "
DATA 12	Bits 15-12	Address of Correlator Accumulator with minimal content.
	Bits 11-0	Content of this accumulator
DATA 13	Bits 15-14	not used
	Bits 13-0	varies depending on 15/127 chip correlator:
		Short –A: 7SMAX/5 Short-B: SMAX
		Long-A: 63SMAX/61 Long-B: SMAX
DATA 14	Bits 15-14	not used
	Bits 13-0	varies depending on 15/127 chip correlator:
		Short –A: SMAX+(ALL/5) Short-B: SMAX-(ALL-5)
		Long-A: SMAX+(3ALL/127) Long-B: SMAX-(3ALL/127)
DATA 15	Bit 15 to 14	not used
	Bits 13-0	ALL
DATA 16	Bits 15-12	Address of the Correlator Accumulator with maximal content.
	Bits 11-0	Content of this Accumulator.
DATA 17	Mirrors Correlator Commands in word 17	
DATA 18	Correlator Data	
	Bits 15	MAX_EQ 1 max. counts found in correlator #7; time tracking loop locked
		0 time tracking loop unlocked; BITs 14-0 invalid
	Bits 14-0	D_GY[14:0]
		Delta gyration time from code-start on in units of 1/(2*f(CLK_DT)) Note that D_GY is a signed number
DATA 19	Spare, used by Tracking Simulator	
DATA 20	Bits 15-8	Logic Cell Array Readout Byte, if readout was requested.

	Bits 7 to 0	not used	
DATA 21	Mirrors GDU Systems Commands 1 (CMD 21)		
DATA 22	Digital HK/System Commands 2 (CMD 22)		
	Bit 15	DONE2	High if FPGA2 is configured
	Bit 14	CINV	|
	Bit 13	TON	|
	Bit 12	TPN	|
	Bit 11	SOU2	|
	Bit 10	SOU1	| See CMD 22
	Bit 9	PUH	|
	Bit 8	SP2	|
	Bit 7	SP1	|
	Bit 6	FON	|
	Bit 5	ARES	
	Bit 4	SPAR1	Pin K16
	Bit 3	S2	| Current phase in
	Bit 2	S1	| beam switch cycle.
	Bit 1	HVFU	High if password is correct
	Bit 0	SRQ	Sensor Cmd Request status
DATA 23	Frame Counters		
	upper byte FR_C	: since last accumulator reset	
	lower byte FRHK	: since last HK counter reset	
DATA 24-27	Sensor Status		
	12 LSB only; valid data only if CMD 23(12) was toggled		
	Sensor Status data reordered (LPARL diagram 3 Aug 1991)		
DATA 24	Bit		
	15-12	---	
	11	Sensor Status MSB	(Always 0)
	10	Spare	(Always 0)
	9	OVRFL	Note A
	8	DISARM	Note B
	7	PWST	Note C
	6	DT2	
	5	DT1	Note D
	4	DT0	
	3	DA7	
	2	DA6	
	1	DA5	
	0	DA4	
DATA 25	Bit		
	15-12	---	
	11	DA3	Note E
	10	DA2	
	9	DA1	
	8	DA0	
	7	AC7	
	6	AC6	
	5	AC5	
	4	AC4	Note F
	3	AC3	
	2	AC2	

	1	AC1	
	0	AC0	
DATA 26	Bit		
	15-12	---	
	11	Always 0	
	10	MX2	
	9	MX1	
	8	MX0	Note G
	7	Spare	(always 0)
	6	EXE IMMED	
	5	SWAP ACC/COR	
	4	TST PLSR ENA	
	3	AP7	
	2	AP6	
	1	AP5	
	0	AP4	
DATA 27	Bit		
	15-12	---	
	11	AP3	Note H
	10	AP2	
	9	AP1	
	8	AP0	
	7	REVRS LOOK (R0)	
	6	BA6 (S3)	
	5	BA5 (S2) Segment	
	4	BA4 (S1)	
	3	BA3 (S0)	Note I
	2	BA2 (E2)	
	1	BA1 (E1) Element	
	0	BA0 (E0)	
DATA 28	GDU HK		
	Bits 15-12	not used	
	Bit 12	Sign of GDU HK Voltage. 1 for positive voltages.	
	Bits 11-0	Voltage of selected HK channel; The peak voltage is 2.5V. The scaling is 100 counts/V/1.953ms (8 frames). Thus, the resolution is 1/250/1.953ms. For an integration period of 80 frames, the resolution approaches 4E-4 (f = 51.2kHz/V). For a definition of the HK parameters, see Section 8.3.4 If group 7 is selected the current power converter frequency is selected. The nominal 65536Hz results in 128 counts/1.953ms.	
DATA 29	Bits 15-12	not used	
	Bits 11-0	Signal Accumulator 1	
DATA 30	Bits 15-12	not used	
	Bits 11-0	Signal Accumulator 2	
DATA 31	Bits 15-14	current phase in beam switch cycle	
	Bits 13-12	not used	
	Bits 11-0	ACCU DIF	

Counts of Correlator EVT channel, delayed to match the arrival time of electrons of a given beam switch step.

C.5.1 Sensor Status Notes

A) OVERFLOW
When =0, this indicates an overflow in the A/D conversion for the digitized analog status data (DA0-DA7).
B) DISARM
When = 1, the Sensor HV is disarmed.
C) POWER STROBE (PWST)
PWST=1 indicates that power strobing is enabled. See PWST command for explanation.
D) DEADTIME (DT0 - DT2)
See Deadtime command for description.
E) ANALOG STATUS (DA0-DA7)
Digitized analog monitor data for the channel specified by MX0 - MX2.
F) ACCUM + CORREL POINTER FIELD
These 8 bits are the logical OR of the Accumulator and Correlator pointer fields, rotated left by a number of bit postions (0 - 7) that is given by the value of the "element code" (3 LSB's of the base address). The ORed and rotated value is used to confirm proper wrapping of the Segment/Element pattern within the Sensor logic.

Examples:

Accum Pointer	Correl Pointer	Base Addrss	Acc + Corr Field
10010001	00111100	00111000	10111101
10010001	00111100	00111011	11101101

G) INSTRUCTION REGISTER
Contents of the Sensor Instruction Register.
H) ACCUMULATOR POINTER FIELD
See description of Accumulator Pointer in the Commands section.
I) BASE ADDRESS
See description of the Base Address in the Commands section.

Depending on DATA 26(10-8), MX2-0:

Channel	Analog Status
0	2.5V
1	HVCR (Current)
2	HVER (Error)
3	HVRF (Ref)
4	PRGN (Gain)
5	N_5V
6	P_5V
7	STMP (Temp.)

Sensor Status data are valid only in the frame where the sensor receives a command. i.e. when bit 12 in CMD 23 has been toggled.

The Analog Status (bits DA7-DA0 in DATA 24/25) can be any one of 8 quantities, as identified by the 3-bit multiplexer status (MX2-MX0 in DATA 26).

To see a certain Analog Status channel, say number 3, one sends a GDU CMD 23, with the the MX bits set to 3, and toggles bit 12 of CNMD23. Channel 3 then is set up; the next time one sends a commnd to the Sensor, channel 3 is digitized; the third time a command is sent, channel 3 is finally read out, and the MX bits in DATA

26 will say 3. In other words, the Analog Status channels read out concurrent with a command sent, will always reflect the requested channel two commands back, but are properly identified by the channel # that is read out!!

If one just wants to see the same Analog Status channel continuously, channel 3 for example, one first has to send the channel 3 request command twice before getting channel 3 data. From the third time onwards, channel 3 will be valid.

If one wants to periodically step through all 8 Analog Status channels, say from 0 to 7, one starts sending the entire sequence 0,1,2,3,4,5,6,7,0,1,2,... Data for the first "0" are returned in the frame that the request for the first "2" is issued, etc.

The digital status (except the MX bits which indicate the current Analog channel) refers to the previous data request, i.e., to the previous time the Sensor was commanded.

D. Software-Patch Procedures

D.1 Procedure for Creating Software Patch (IPCH format)

D.1.1 Overview

When EEPROMs need to be patched, patch files will be created offline by the EDI team. This procedure is therefore of interest only to the EDI team. The files resulting from this procedure (IPCH files) will be delivered for upload (see D.2 „Procedure for Uploading Software Patch"), along with a procedure for uploading the patch which includes expected checksums, number of commands and other verification information. The patch files will be delivered in the IPCH file format specified in ESOC's CRID document (COMMAND REQUEST INTERFACE DOCUMENT CL-ESC-ID-0003). A summary is given in chapter D.3.

NOTE: On power-on, the checksums of the first four EEPROMs are verified by a poweron routine. The checksums of all remaining EEPROMs are verified by a routine on the $4000 EEPROM. The calculated checksums are compared to numbers stored in bytes starting at address $4008. Code EEPROMs always have a $4A checksum, but PGA EEPROM checksums vary with configuration file. Therefore an upload to a PGA EEPROM requires also an upload to the $4000 EEPROM to update the matching PGA checksum byte.

An upload consists of IPCH patch data files and also IPCH patch command files. The patch command files are required for copying data from EEPROM to RAM, from RAM back to EEPROM, for initiation of checksum computations in RAM and EEPROM and other tasks. See procedure D.2 for more details. There are two kinds of upload, called *full upload* and *delta upload*. In a full upload patch data are created for all 16K bytes (32K for PGA2 EEPROMs) in one complete EEPROM Intel Hex file. In a delta upload, only those bytes which have changed with the new version of the EEPROM Intel Hex file are written to the IPCH patch file. Most uploads will be *delta uploads*.

The program ECS (EDI Commanding System) is used to create IPCH patch files, an upload history log and a list of the action items required to perform the actual upload. This procedure (D.1) describes how to run ECS. In this description, the term 'session' means the preparation of a set of patch files for all EEPROMS to be updated in the same uplink with the spacecraft. The ECS program maintains a copy of the current state of the EDI controller EEPROMs in directory

drv:\ECS\CLUSESSIONnnnn

where *nnnn* is a four digit serial number which identifies the session, and *drv:* is a generic notation for the drive on which ECS is installed. When a session is run for a new set of uploads, a new directory CLUSESSION*mmmm* (*mmmmm*=*nnnnn*+1) is created. This directory reflects the state of the controller after a successful upload with the files created in session *mmmm*.

The contents of each directory CLUSESSION*nnnn* are:

File	Description
SC*i*_UploadProc*nnnn*.TEX	Procedure for uploading the patch (in LaTex format)
EE_*zz*_*i*.HEX	Mapped (see D1.2.2.2) Intel Hex Files for code / table EEPROMs identified by *zz* (see D.3); *i*=1,2,3,4 identifying the spacecraft
ODD_*zz*_*i*.HEX	Mapped Intel Hex Files for odd-byte EEPROMS (*zz* = 1c or 7c)
IPCH_*yymmdd*JSMC_00000000_*ssss*.CR	IPCH files, as output by the program ECS (numbered serially via '*ssss*'); *yymmdd* is the date (year/month/day) of the IPCH file creation.
IPCH*ssss*.TXT	Copies of the IPCH files in DOS filename convention (8.3) for testing the upload on the CLU EDI Lab Controller prior to applying them to the Flight Controller(s) on the S/C.

The serial number '*ssss*' of the patch files is not for a single session but for the lifetime of the Cluster-II mission. This is to ensure that the files are unique in case there would be IPCH files from two separate sessions on the same day. A log file containing information about all sessions is also maintained.

All EEPROMs are identified by 4 byte version numbers. The format of the version numbers is:

EEPROM	Format	Location (zero relative)	Notes
Code and table EEPROMs	zz tm dd vy	3ffa-3ffe	all hex
PGA1 EEPROM (1c)	1v dd mm yy	odd bytes 1,3,5,7	all decimal
PGAi EEPROM (7c)	4v mm dd yy	odd bytes 1,3,5,7	all decimal
PGA2 EEPROMs (begin) PGA2 EEPROMs (end)	zz im dd vy zz im dd vy	0000-0003 3ffa-3ffe	all hex all hex
TS (test) EEPROMs (44,48)	zz 1m dd by	0000-0003	(all hex, ground use only)

where : *zz* = address byte (for EEPROMs less than $1x x000, zz is xx;
(for EEPROMs greater than $1x x000, zz is xx+1
t = 0 for code EEPROMs, GDU ID modulo-16 for gun table EEPROMs
i = id byte (1=emulator, 2=flight, 3=Tracking Simulator, 4=test)
dd = day
mm
or
m = month
y = year (year – 1996)
yy = year
v = version

D.1.2 Creating IPCH files

.1 Inputs

- **New Intel Hex files**
  The Intel Hex files to be used as sources for the creation of IPCH files are either code/table files created by the INTROL assembler/linker, or PGA configuration files created by the XilinX software. These files should be copied to *drv:\ECS\NewIntelHexFiles* Before copying the files, any old files from previous sessions in this directory should be zipped and archived.

- **Current state of the EEPROMs**
  In case of a delta upload, additional inputs are the mapped Intel Hex files which reflect the current state of the EEPROMs on the spacecraft. These files are stored in *drv:\ECS\CLUSESSIONnnnn*, where *nnnn* is the highest session number existing so far.

- **Auxiliary ICF files**
  A number of auxiliary IPCH files are needed for the upload to an EEPROM, for tasks like copying data from EEPROM to RAM and vice versa, initiating computation of checksums, etc. These IPCH files will be created from corresponding ICF files, which have to be in directory *drv:\ECS\ICF*. The ICF files which need to be in this directory are listed in D.3
  The master directory for these ICF files is DKA400:[CLUSTER.MODULES] on the VAX station MPECL4.

- **Telecommand mnemonics**
  A database of EDI telecommand mnemonics is stored in *drv:\ECS\DB\MNEMONICS\MNEMONICS.DB* It is used for the conversion of the auxiliary ICF files into IPCH files. If new telecommands are added to EDI and if these new telecommands are to be used in any of the auxiliary ICF files (or later versions of these) then the file MNEMONICS.DB will have to be updated (using the database editor which shipped with Borland Delphi). The database can be viewed from within ECS by selecting 'Database', then 'Mnemonics' from the main menu of ECS.

.2 Outputs

All outputs, except for the history log file HISTORY.DB, go to directory *drv:\ECS\CLUSESSIONmmmm* where *mmmm* is the new session number, derived from the contents of the log file HISTORY.DB. Unless entries in the log file have been deleted manually, *mmmm* will be *nnnn*+1, where *nnnn* is the number of the previous session.

- **IPCH files**
  There will be two sets of IPCH files, the official set and the set for testing the upload on the EDI Lab System (S/C simulator + Lab Controller). See the D.1 (Overview, Files) for a description of the file names.

- **Mapped Intel Hex files**
  The input Intel Hex files for an EEPROM created by the INTROL assembler/linker do not necessarily contain data for each address of the EEPROM. *Mapped* Intel Hex files are Intel Hex files which *do* contain data for each address of an EEPROM. These are created from the input Intel Hex files by filling the bytes at the unused addresses with $FF and by inserting the fudge word (in case of code/table EEPROMs). The fudge word makes the overall EEPROM checksum $4A and makes both the even-byte and odd-byte checksums $A5. See the table in D.1 (Overview, Files) for a description of file names.

- **Upload Procedure**
  A file 'SC*i*_UploadProc*mmmm*.tex' is created. This file must be processed by LaTeX programs on a VMS or UNIX system to create a descriptive text file used to guide the operator sequentially through the steps required to do the actual upload. Here is an example of commands to process the file with LaTeX:

  latex SC*i*_UploadProc*mmmm*.tex
  dvips SC*i*_UploadProc*mmmm*.dvi -o SC*i*_UploadProc*mmmm*.ps

  The .ps file can then be printed using whatever the appropriate print command (e.g. lpr on UNIX).

- History log file
  The log file HISTORY.DB, residing in directory *drv:\ECS\DB\HISTORY*, contains information about all sessions. This file is updated with each new session. It can be viewed selecting 'Database' and then 'History' from the main menu of ECS.

- Error messages
  The ECS program will generate error messages during operation only if the input Intel files contain errors. If such is the case, it is possible that one try to load an image to a wrong destination or that the errors in the Intel files were introduced during file transfer. Retransmit the files to eliminate the errors.

.3 Starting ECS

Creating IPCH files using ECS is done on an IBM compatible PC system running MS-Windows 95/98. An ECS Icon should be present on the Desktop or reachable over the Start button. Double-click on this icon to start ECS. If the Icon is not present for some reason, open the Windows Explorer, select the directory *drv:\ECS* and double-click on ECS.EXE

.4 Creating IPCH files

Once the Main menu of ECS is displayed on the screen, use the mouse to select 'Software Uploads' and ‘’Start’. A subwindow will open. The title of the window is ‘CLUSTER SOFTWARE UPLOADS'.

a) In the field ‘Operator“ type in your name or select it in the predefined list of user names by single click.
b) Option 1 : Press the ‘Load’ button under ‘Parameter’. Then select an prepared UPL-file with the complete upload information.. After this option proceed directly with step h)
   Option 2 : n the field ‘Spacecraft’ select one or more destination spacecrafts for upload. The fields ‘EEPROM address’, ‘Files’ and ‘Upload’ are enabled now for input.
c) Select an EEPROM in the field ‘EEPROM address’ for the first selected SC.
d) Enter the complete path and the name of the new Intel Hex file to be uploaded in the edit field. There are two options: single click in the edit field, then type in the file name of the Intel Hex File (including path)
   double click in the edit field, then select the Intel Hex files in the box popping up.
   If there is no upload at this address for a selected spacecraft leave the field empty. There is no action during file creation.
e) If a full upload is desired, check the box in the field ‘Upload’.
f) If there are 2 or more EEPROMS to be updated then repeat steps c) to e).
g) If there is more than one SC is selected repeat step c) to f) for the other SCs.
   After step g) it is useful to store the complete upload information just entered. Press the ‘Save’ button under ‘Parameter’ and enter a filename. Later this file can be used with the ‘Load’ option.
h) Before starting the creation of the IPCH files, check the selections made in steps a) to f). For each EEPROM (not only for the EEPROMs to be updated!) click on the respective button in the EPROM field, then check Edit and Upload fields. Make sure that for EEPROMs that are not intended to be updated, the Spacecraft box is not checked and that the edit field is empty.
   If the upload information is saved, it may be easier to view the UPL-file for this check
i) (Optionally) Click the "Preview" button. This will perform tests on the input files. The steps will be shown in the message window. No files will be created.
j) Changes can still be done for any EEPROM by repeating the steps c) to e).
k) Start the IPCH file creation by clicking on the button "Start IPCH-File Creation". The single steps performed will be monitored in the message window. The steps are:
   - basic initialization
   - reading current state
   - reading new files
   - reading database
   - generating EEPROM fudge word
   - reading PGA fudge word
   - calculating EDI checksum
   - verification of true differences
   - termination (in case of "Preview")
   - creating IPCH files
   - creating new session directory
   - update of history database
   - creating Upload Procedure
l) Exit the application by a single click on "Close Window".

.5 Rejecting Sessions

Only the last session can be deleted. This is done by removing the information from the database, and by deleting the *CLUSESSIONnnnn* directory.
To remove the session, select 'Database', then 'History' from the main menu of ECS.
Click the ‘Delete last Session’ button. Confirm the upcoming message box with ‘YES’. ECS now will remove the complete session from HISTORY.DB and delete the complete session directory.

.6 Testing Created Sessions

Each new upload has to be tested on the EDI simulator. The description of this procedure is in appendix D.4.

D.2 Procedure for Uploading Software Patch (IPCH format)

Title Page

EXPERIMENT : EDI
PROCEDURE : EDI_EEPROM_LOAD
PURPOSE : To load new tables, software or PGA configurations
APPLICATION : Commissioning and Routine Operations
CONSTRAINTS :

- Spacecraft in ground station contact for duration of load
- maximum number of patch telecommands
  for full 16 kB upload: 128 blocks * (64+3 TCs) + a maximum of 113 TCs for control and verification
  for full 32 kB upload: 256 blocks * (64+3) TCs + a maximum of 113 TCs for control and verification
- EDI must be in NM telemetry
- EDI will be in "quiescent" or "standby" mode and therefore will not collect science data during uploads.
- power requirements are lower than baseline
- verification of the modification will usually not be immediately after the upload because it may be dependent on ambient plasma/magnetic field conditions

FUNCTION :

- the section of EEPROM to be updated is copied to RAM
- the upload file(s) are applied to the RAM copy of EEPROM
- EEPROMs are write-enabled and a EEPROM-specific, special copy file is executed to copy the patch back to EEPROM
- proper transfer is checked through checksum tests and diagnostic files

In order to perform an upload, the operator needs to have:

- a printout of the postscript file SC*i*_UploadProc*mmmm*.ps
- a number of IPCH files named after the conventions of the ESOC CRID document (see D.3).
- a command file for write-enabling the EDI EEPROMs

P r o c e d u r e

Below is a generic procedure for the upload of an EDI software patch. The ECS program (see D.1) creates an actual procedure (SC*i*_UploadProc*nnnn*) containing the instructions for the operator. The instructions include the files to fetch, items to verify and the values of CVCNT to be expected after each step. The capability to reset the command counters has been implemented so that absolute values of command counters can be specified. Checking the fudge words has now been automated so that the operator only needs to check a single HK-word (FGCNT).

Step No.	**Description**	**HK parameter check** (in addition to CRCNT, CVCNT, LASTC)
1	Verify some EDI HK parameters; record telecommand (TC) counters	PGACF=2 or 3 EEPON=0 MODID=8 M8STA=02 or 04
2	For a delta upload copy EEPROM to RAM	
3	Load patch file and calculate RAM checksum	CHKSU
4	Write enable EEPROMs (ZEDILP2A or B)	EEWRP=0
5	Copy from RAM back to EEPROM	MODID=0 EEWRP=1
6	EEPROM checksum test	CHKSU
7	Jump to mode 8	MODID=8
---	**repeat steps 2-7 for each EEPROM to be updated**	
8	Compare fudge words for EEPROMs	MODID=8 FGCNT=0

If EDI branches to ROBUST2 during the software upload the operator should follow the steps desrcibed in chapter 9.2.1, section A for recovery. The table below lists the action to be taken by the operator in case of a HK parameter check failure. Unless otherwise noted, the procedure should be aborted if the performed action/remedy is not successful.

Step No.	HK parameter check failure	Action/Remedy
1	PGACF	abort procedure
	EEPON	send sequences #350 (EEPON) and #71 (MODE8)
	MODID	if MODID=0, send sequence #71 (MODE8) in case of any other MODID, abort procedure
	M8STA	abort procedure
2	LASTC/CRCNT/CVCNT	repeat step; abort if still wrong
3	LASTC/CRCNT/CVCNT	send sequence #300 (NOOPS) to clear internal command queue, then repeat step; abort if still wrong
	CHKSU	repeat step; abort if still wrong

4	EEWRP	repeat step; abort if still wrong
5	LASTC/CRCNT/CVCNT	if in Robust2, then abort, otherwise send sequence #300 (NOOPS) to clear internal command queue, then repeat step; abort if still wrong
	EEWRP	repeat step; abort if still wrong
	MODID	restart at previous step (write enable EEPROMs)
6	LASTC/CRCNT/CVCNT	repeat step; abort if still wrong
	CHKSU	restart at step 4 (Write enable EEPROMs)
7	LASTC/CRCNT/CVCNT/MODID	repeat step
8	LASTC/CRCNT/CVCNT	send sequence #300 (NOOPS) to clear internal command queue, then repeat step
	MODID/FGCNT	Branch to procedure "EDI Software Upload Contingency Procedure"

D.3 IPCH File Format

Reference : 1) Cluster CRID, Issue 2.2, 3 July, 1995
2) EDI upload definitions due to ESOC,17.05.96

1.0 File Naming Convention
In all alphabetic fields only uppercase characters are used. Files are named in the following way:

CL_*iii_yymmdd_vvvv*.IPCH

where : *iii* instrument acronym ('EDI')
yymmdd is the file generation time (note: EDI suggests the time of the IPCH files)
vvvv is the version number starting at 1 and incrementing for each file. *vvvv* uniquely identifies a file.

2.0 Data Definition
Files consist of a standard file header, two file specific headers and a variable number of patch records containing a patch header and the patch data. Comment lines are identified by an exclamation mark(!) in column 1 (leftmost column). A comment line can be placed anywhere in the file except within the patch data.

Field	Bytes	Position	Description	contents
			Standard File Header	
1	4	1 - 4	filetype identifier	'IPCH' (fixed)
2	4	6 - 9	file version number	number with leading zeros
3	20	11 - 30	upload file generation time (UTC) (1)	'yyyy-mm-ddThh:mm:ssZ'
			File Specific Header (first record)	
1	8	1 - 8	command definition name	characters padded with trailing blanks (2)
2	7	10 - 16	instrument name	'EDI' padded with leading blanks
3	4	18 - 21	spacecraft validity mask	IDs of applicable spacecraft padded with leading blanks e.g.'134'
4	3	23 - 25	number of patch blocks	number padded with leading blanks (in decimal)
			File Specific Header (second record)	
1	68	1 - 68	command definition description library reference (3)	characters padded with trailing blanks
			Patch Block Header	
1	6	1 - 6	start address in hex	hex number padded with leading blanks
2	6	8 - 13	number of ensuing 16 bit hex data words (starting at next record)	hex number padded with leading blanks, max number = 50 (hex 32)
3	4	15 - 18	16 bit check-sum (4)	hex number padded with leading blanks
			Patch Data (5)	

```
-The patch words are 16 bit data words ( hex representation, case insensitive),
seperated by one space (at most 16 words per line).
Example see below.

ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff
ffff ffff ffff ffff ffff

--------------------------------------------------------------------------------
```

Notes :

(1) Upload file generation time is the generation time of the IPCH files

(2) Command definition name contains information about the EEPROM to be uploaded, the order of sending the IPCH files to upload one EEPROM, the source Intel Hex file and auxiliary files. This name will be archived in the command history file and will be the only reference of an upload done on the spacecraft. The command definition name is also in the command definition description library reference available until the upload.

Description	**Corresponding auxiliary ICF file**
for each EEPROM to be uploaded	
copy EEPROM to RAM (for delta uploads only)	C2Rzz.ICF
load patch data RAM checksum computation	Patch (no ICF) RCK16K.ICF / RCK32K.ICF
copy RAM to EEPROM	C2Ezz.ICF
EEPROM checksum computation	CKzz.ICF or CKzz_2.ICF
jump to mode 8	FJM8.ICF
after all uploads	
display fudge words in HK variable section	FUDGE.ICF

n is an incrementing serial number, starting at 0.
The date of the reference intel file is determined by *ddd* (day of year in decimal notation.)
i is the S/C number (i=1,2,3,4 for FM5, FM6, FM7, FM8, respectively.)
x and *zz* are alphanumeric and hex numbers defining the EEPROM to be uploaded as follows:

EEPROM--*x* --	*zz*	EEPROM--*x* --	*zz*	EEPROM--*x* --	*zz*
04000 a	04	0c4000 m	c4	184000 y	85
08000 b	08	0c8000 n	c8	188000 z	89
24000 c	24	0e4000 o	e4	1a4000 0	a5
28000 d	28	0e8000 p	e8	1a8000 1	a9
44000 e	44	104000 q	05	1c4000 2	c5
48000 f	48	108000 r	09	1c8000 3	c9
64000 g	64	124000 s	25	1e4000 4	e5
68000 h	68	128000 t	29	1e8000 5	e9
84000 i	84	144000 u	45	01c000 6	1c
88000 j	88	148000 v	49	07c000 7	7c
a4000 k	a4	164000 w	65	03c000 8	3c
a8000 l	a8	168000 x	69	05c000 9	5c

(3) The version number of the Intel Hex file containing the updates is given here as it is stored in the Intel Hex file.
(4) The checksum value is the sum modulo 65536 of all 16 bit patch data words.
(5) The maximum length of a patch block is 67 16-bit words (3 TC's to specify destination address and number of data words to follow + a maximum of 64 data words). Larger patches need to be split into several patch blocks (a patch block consists of a Patch Block Header and Patch Data). One IPCH file can contain more than one patch block.
(6) EEPROM pairs a4:a8, c4:c8, e4:e8 are 32K PGA2 EEPROMs and all of the $1xxxxxx EEPROMs are 32K EEPROM gun table pairs. Each pair is identified by the start address of the low 16k block. It is not permitted on a full upload to load just 16k of this pair.

D.4 Procedure for testing a new upload

D.4.1 Overview

Sections D.4.1 to D.4.6 describe procedures that were used for testing uploads to EEPROMs on the Equator-S satellite. There are no analogous procedures for Cluster-II EEPROM upload testing. Procedures for Cluster-II RAM upload preparation and testing are described in Sections D.4.7 and D.4.8.

The IPCH format files created in each ECS session must be tested on the Cluster EDI bench Controller before applying them to the EDI instrument on the Cluster-II S/C. To do this, all IPCH files are read by a program called SP2EDI.EXE and converted into the ICF file format known to the S/C simulator. Then the upload is done to the EDI lab test system following the procedure given in the file SC*i*_UploadProc*nnnn*. After the upload, the contents of all EEPROMs are copied to scratch RAM and then dumped into a file in telemetry mode BM3. The dump file is cut into mapped Intel Hex files, one for each EEPROM.
These mapped Intel Hex files are compared to mapped versions of the ECS input Intel Hex files. The success of an upload is verified if no differences occur. In what follows the following terms are used:

drv	identifies hard disk partition drive letter
codepath	identifies a path to a file or set of files containing executable code
pgapath	identifies a path to a file or set of files containing PGA files
gunpath	identifies a path to a file or set of files containing gun tables
aa	address identifier of an EEPROM file
even#,odd#	Gun number
j,k	Gun number – 16

It is assumed that the person who performs this test knows how to start and operate the EDI Cluster S/C Simulator and the EDI Controller, upload program EDIUPXY.EXE and DOS batch files in general.

D.4.2 Initial state

The initial contents of the EEPROMs in the EDI bench Controller must match the state of the EEPROMs which are in use onboard the S/C. That state is reflected in the files of the ECS session previous to the one which is being tested. These files are stored under the names given in column 4 in the table in D.4.4.6 in subdirectories *drv*:ECS\PREVIOUS\ or *drv*:ECS\PREVIOUS*i* where common files are stored in PREVIOUS and S/C specific files are stored in subdirectories (*i*=5,6,7,8). These directories hold files for all EEPROMs, but only some of them will have to be reloaded to recreate the onboard configuration. Files should be loaded using EDIUPXY.EXE and the resulting EDIUPXY.LOG file should be saved as part of the upload documentation.

D.4.3 Input Files, Batch Files

Inputs to the ECS process are in *drv*:ECS\NewIntelFiles. Only files for EEPROMs updated in this ECS session are found in this directory.

Inputs to the upload verification process are:
files in *drv*:ECS\NewIntelFiles for EEPROMs affected by the upload
files for all other EEPROMs in *drv*:ECS\PREVIOUS, and *drv*:ECS\PREVIOUS*i.*
DMP file from the S/C simulator

All of these files are listed in the table in D.4.5. Note that PGA2 and Gun Table files are 32K big, but the program which splits SCS Dump files into pieces splits them into 16K pieces. This difference is handled by the compare programs in step 7 of D.4.4.

Batch files have been created to simplify the process of verifying a new upload.

- Copy Intel Hex files for the executable code and odd-byte PGAs to the IPCH subdirectory
  CPYCHEX *drv codepath pgapath*
- Copy Intel Hex files for Gun tables to the IPCH directory
  CPYGHEX *drv gunpath odd# even# j k*
- Rename Intel Hex files for executable code and odd-byte PGAs to EP*aa*.0 format
  CRENAME
- Rename Intel Hex files for Gun tables to GzEP*aa*.0 format
  GRENAME *odd# even# j k*
- Rename OUT files from S/C simulator DMP file for code / PGAs to RMEE*aa*.0 format
  RCRNAME
- Rename OUT files from S/C simulator DMP file for Gun Tables to RMEE*aa*.0 format
  RGRNAME
- Delete files for EEPROMs not in use; Delete files before verification process starts
  DELNOTIU; PURGE
- Compare upload session source files to DMP of controller EEPROMs after upload
  CMPALLG for Gun table files
  CMPALLC for all other files

If EEPROMs which are not currently in use become used, the batch files which operate on files of that type will have to be changed. Unused EEPROMs are: 08000, 48000, 64000, 68000, 3c000, 5c000.

D.4.4 Testing the Uploads

0. Create initial state of EEPROMs on Cluster EDI bench Controller (see D.4.2)
1. Create IPCH files (see D.1)
2. Set up files on the Cluster S/C Simulator PC

- change directory to C:\IPCH
- enter the command 'PURGE.BAT'. This will delete all files matching the wildcard specifications IPCH*.TXT, IPCH*.ICF, *.AFF, *.0, *.OUT, *.DMP. If this is the first or only verification for this session (if there are no S/C specific changes), enter the command 'DEL EEMAPS.LOG' to delete the comparison log file.
- copy the files IPCH*ssss*.txt to the directory C:\IPCH
- copy EEPROM input files (see D.4.3). Rename files for easier identification using CRENAME and GRENAME.
- use an editor to create a batch file IPCH2ICF.BAT in C:\IPCH (or edit the existing ICH2ICF.BAT). The batch file has to have one command line for each IPCH file as shown in the line below:
  sp2edi ipch*ssss*.txt <i.rsp

where the real numbers occurring in the IPCH file names have to be inserted for *ssss*. This will create ICF files IPCH*ssss*.ICF, corresponding to the IPCH files IPCH*ssss*.TXT

- Execute the batch file

3. Start Cluster S/C Simulator and EDI Controller

- Start the Cluster S/C Simulator
- Switch on the Current meter box
- Select the EDI Science display (<S>). Turn on power for EDI controller wait a few seconds, then send TC NOOP in order to wake up telemetry
- configure PGAs,

4. Upload the patches

Follow the instructions in SC*i*_UploadProc*nnnn* to upload the patches. Instead of the real IPCH files, send the ICF files created in step 2.

Note: to toggle the EEPROM write use A-F2 from S/C simulator or use the switch on the interface box.

5. Dump contents of all EEPROMs

Send the ICF file GEEDUMP.ICF. This command file will cause all EEPROMs which are in use to be written to scratch RAM with the GUN table EEPROMs first. In order to dump the contents of scratch RAM to a file,

- start the SCS with batch file MODE07B.BAT
- open a dump file <F6> and name the dump file EEDUMP.DMP
- send TC SUMOS 07, verify SUBMO=07 in HK
- go to EDI Science display screen
- wait until dump address (words 2&3 of science TM) is larger than B88100 (for dumps which are intended to verify gun tables only, wait for an address larger than B40100)
- wait one more TM update, then press <F6> and <F5> to close the dump file (and the log file)
- send TC SUMOS 01
- quit the simulator

6. Cut dump file into mapped Intel-Hex files

In directory C:\IPCH and execute the following command:

EEBM3 /i eedump.dmp /flags 1 /o eea.out

The program will take a while to complete. It creates the 16K files EE*x*.OUT, where *x* = a,b,c,d,e,f,g ... x,y,z, 0,1, 2,3,4,5,6,7. These files are mapped Intel Hex files which reflect the status of the EEPROMs after the upload. Some OUT files do not have Intel file counterparts. Gun tables and PGA2 files are stored in a single 32K Intel file, thus two 16K OUT files are compared against a single 32K Intel file.

Use RGRNAME and RCRNAME to rename the OUT files to include the EEPROM address in the name. (Optional: Delete files for EEPROMs which are not in use. Use DELNOTIU.BAT.)

DMP Hex file	Renamed DMP Hex file	EEPROM start address	Original Intel file Name	Renamed Input Intel Hex file
EEa.OUT	RMEE05.OUT	$104000	GDUMMy5.0	G*j*EP05.0
EEb.OUT	RMEE09.OUT	$108000		
EEc.OUT	RMEE25.OUT	$124000	GDUMMx5.0	G*j*EP25.0
EEd.OUT	RMEE29.OUT	$128000		
EEe.OUT	RMEE45.OUT	$144000	GDUNNy5.0	G*k*EP45.0
EEf.OUT	RMEE49.OUT	$148000		
EEg.OUT	RMEE65.OUT	$164000	GDUNNx5.0	G*k*EP65.0
EEh.OUT	RMEE69.OUT	$168000		

EEi.OUT	RMEE85.OUT	$184000	GDUMMy.0	G*j*EP85.0
EEj.OUT	RMEE89.OUT	$188000		
EEk.OUT	RMEEa5.OUT	$1a4000	GDUMMx.0	G*j*EPa5.0
EEl.OUT	RMEEa9.OUT	$1a8000		
EEm.OUT	RMEEc5.OUT	$1c4000	GDUNNy.0	G*k*EPc5.0
EEn.OUT	RMEEc9.OUT	$1c8000		
EEo.OUT	RMEEe5.OUT	$1e4000	GDUNNx.0	G*k*EPe5.0
EEp.OUT	RMEEe9.OUT	$1e8000		
EEq.OUT	RMEE04.OUT	$04000	MODE8.0	EP04.0
EEr.OUT	RMEE08.OUT	$08000	---	EP08.0
EEs.OUT	RMEE24.OUT	$24000	HIMOD.0	EP24.0
EEt.OUT	RMEE28.OUT	$28000	HIMDX.0	EP28.0
EEu.OUT	RMEE44.OUT	$44000	ITS07a8a.0	EP44.0
EEv.OUT	RMEE48.OUT	$48000	---	EP48.0
EEw.OUT	RMEE64.OUT	$64000	---	EP64.0
EEx.OUT	RMEE68.OUT	$68000	---	EP68.0
EEy.OUT	RMEE84.OUT	$84000	TESTS.0	EP84.0
EEz.OUT	RMEE88.OUT	$88000	OPTICS.0	EP88.0
EE0.OUT	RMEEa4.OUT	$a4000	Cxxxxxxx.0	EPa4.0
EE1.OUT	RMEEa8.OUT	$a8000		
EE2.OUT	RMEEc4.OUT	$c4000	Cyyyyyyy.0	EPc4.0
EE3.OUT	RMEEc8.OUT	$c8000		
EE4.OUT	RMEEe4.OUT	$e4000	Czzzzzzz.0	EPe4.0
EE5.OUT	RMEEe8.OUT	$e8000		
EE6.OUT	RMEE1c.OUT	$1c000	Ixxxxxxx.0	EP1c.0
EE7.OUT	RMEE7c.OUT	$7c000	FG4MS.MCS	EP7c.0

7. Compare EEPROM contents (Output Intel Hex files) with Input

- Execute the batch file CMPALLC.BAT. This file runs programs to compare non-Gun Table Intel Hex files (EP*aa**.0, *xx*=04,24,28, …) to the mapped Output Intel Hex files (RMEE*aa, aa*=04,24,28, …). Each compare includes two steps. First a program (EEMAP32.EXE) takes an Input Intel Hex file (EP*aa*.0) and maps it (→EP*aa*.AFF), then another program (EEPGA32.EXE for odd-byte PGA EEPROMs and EECMP32.EXE for all other EEPROMs) compares the mapped Input Intel hex files (EE*aa*.AFF) with the mapped Output Intel Hex files (RMEE*aa*.OUT).
- Execute the batch file CMPALLG.BAT ***j k***. This file runs map and compare programs for the Gun Table EEPROM files.
- Both sets of programs write to a log file (EEMAPS.LOG). The LOG file lists the input file in the format of the right most column in the table above, the file checksum (or checksums in the case of 32K files) and the result of the comparison. If the comment is "no differences to patch" for all files, the upload test is successful.
- If the Gun Tables are among the EEPROMs that have changed in this upload session, repeat the verification process for the gun tables of the remaining S/C.

D.4.5 Troubleshooting

If, for some reason, the Controller goes to the failure mode ROBUST2 ($A5A5 in science and hk telemetry) during the upload, the code which was in the EEPROMs prior to the upload should be restored.
a) Turn off power, wait a few seconds
b) Turn on power, wait a few seconds, then send TC NOOP
c) Turn on EEPROMs (<S-F4>)
d) If the controller appears in Mode0 (instead of Mode8) as might happen if EEPROM 4000 is not in synch with the PGA EEPROMs, it is best to do a jump to the Mode8 Power-on address (LDEPM 00, DATAM 6000, JDEPS 60) before proceeding.
e) Reload the Controller EEPROMs (Use "LOADINI i" from directory \IPCH or Use ALLZ.BAT from the current EDI source subdirectory, probably ACODE in *drv:* at MPE or *u:\CLU2\ACODE* at UNH)

D.4.6 Caveats

This procedure is intended to verify that the output produced by the ECS program (i.e. the IPCH files) is correct. As such it is not only used during the mission life time of the Cluster-II S/C for the creation of real patches to EDI, but also whenever changes to the ECS program are made. In the latter case it is not required that the input intel hex files contain code which is newer than the code in the EEPROMs of the Cluster EDI Lab Controller. One should, however, be warned that sometimes pieces of code residing in different EEPROMs have to match. Therefore it is generally wise to use Intel Hex files which belong together.

D.4.7 Creating IPCH files for RAM uploads

In September 2003, the EEPROM write-enable switch failed on SC2. As a consequence of this failure, a modified version of the EDI code was created which executes in RAM rather than EEPROM. The code is loaded to RAM via several large IPCH files. Changes to the RAM code, called "delta uploads", and auxiliary programs to setup and verify the RAM code are also loaded with IPCH files. IPCH files are created with a DOS program *ih2ipch2.exe*. *ih2ipch2.exe* can be thought of as a poor man's ECS. The program produces IPCH files required for loading code to the satellite, but it does not archive files or produce procedure descriptions for ESOC. These tasks must be done by the EDI team. Text file *ipch.in* contains parameters that control *ih2ipch2.exe* operation. At least seven parameters must be given for any execution of *ih2ipch2.exe.* (listed below) *ih2ipch2.exe* also produces a log file ipch.log.

Input parameters:

- name of executable file to be converted to IPCH
- for delta uploads, name of previous version of executable file or for full uploads "dummy"
- IPCH file number
- destination address (EDI memory address where file is to be loaded)
- adjustment between executable file start address and destination address
- type of run: 2=compare files and create delta upload
  1=full load of .0 file; include a "jump to" at the end of the IPCH file
  0=full load of .0 file; do not include a "jump to" at the end of the IPCH file
- number of bytes in the input file (roughly)

Examples of ipch.in execution parameters:

Full upload to RAM 6c000:	6c000.0	dummy	20	6c000	c000	0	8000
Delta upload to RAM 6c000:	6c000.0	6c000.028	30	6c000	c000	2	8000
Auxiliary file to zero RAM:	ramzero.0	dummy	11	10000	0000	1	200

*a line that begins with an * is a comment line

D.4.8 Loading and Testing RAM code

- **Options for loading RAM code to the bench system**

There are two ways to proceed at almost every step, choose a. or b. In general, option a. is easier and quicker for bench testing. However, option b. provides a better match for what actually happens in space and therefore should be used as a final step before creating the IPCH files to send to ESOC.

a. Load .0 file directly using programs *ediram.exe* or *edhiram.exe*. These programs accept the filename on the execution line and prompt for load address, offset and number of bytes.

Ex: ediram filename.0 Responses: 10000 0 200
loads up to 200 (hex) bytes of filename.0 to address 10000 with no offset

Ex: edhiram 6c000.0 Responses: 6c000 c000 8000
loads up to 8000 (hex) bytes of 6c000.0 to address 6c000 an offset of c000

b. Load code using IPCH files via program *sp2edi.exe.*

Ex: sp2edi edi_0001.ipch Responses: send (s) and quit (q)

- **Existing IPCH files**

IPCH files for auxiliary programs exist on the bench test system and also at ESOC. These files are:

edi_0001.ipch	Set all of upper RAM to $ff. This program must be loaded before any RAM testing.
edi_0002.ipch	Checksum RAM beginning at $4C000
edi_0003.ipch	Checksum RAM beginning at $6C000

The bench system and ESOC also have copies of the IPCH files containing EDI RAM code. As of September 2004, these files are:

edi_0020.ipch and edi_0021.ipch	Code for RAM part beginning at $4C000
edi_002x.ipch	where x=2,3,6,8. Code for RAM part beginning at $6C000

- **Loading RAM code on bench test system**

✓	Turn on EDI test setup and optionally Tracking Simulator. Configure as required.		
✓	Clear upper ram.	Option a: ediram ramzero.0, from simulator: JDEP 05	Option b: sp2edi edi_0001.ipch
✓	Load $4c000 RAM.	Option a: edhiram 4c000.0	Option b: sp2edi edi_0020.ipch sp2edi edi_0021.ipch
✓	Cksum $4c000.	Option a: From simulator: execute cksum_4c.icf, verify $4a	Option b: sp2edi edi_0002.ipch Start simulator, verify $4a
✓	Load $6c000 RAM.	Option a: edhiram 6c000.028	Option b: sp2edi edi_0022.ipch sp2edi edi_0023.ipch sp2edi edi_0026.ipch sp2edi edi_0028.ipch
✓	Cksum $6c000.	Option a: From simulator: execute cksum_6c.icf, verify $4a	Option b: sp2edi edi_0003.ipch Start simulator, verify $4a
✓	Load modified code.	Option a: edhiram 6c000.0	Option b: modify ipch.in,

execute ip2ipch2,
sp2edi edi_0030.ipch

- ✓ Test the modified code
- ✓ Correct checksum

When the EDI team is satisfied that the modified code is working properly, create files for ESOC. The first step in this process is to correct the chksum to $4a by changing Byte 4 in the load file. To do this, first determine the chksum of the load file and the value of byte four by looking in ipch.log. The most recent information is at the end of the file. The lines with the chksum looks like this:

6c000.0 cksums 58 a0 f8 ← Checksum for 6c000.0 is $f8.
First 6 bytes: 30bd:0000 30bd:c000 6c 0a 06 18 c0 4a ← This line says byte four is $c0

Compute a new byte four: $f8 - $c0 = $38 ← Part chksum - byte four = correction
$4a - $38 = $12 ← Desired chksum - correction = $12 new byte four

Change version.s32 to include the new byte 4, remake the load file. Load/test again (use option b.)

- ✓ Create the documentation file to send to ESOC with the IPCH files.
  The easiest way to do this is to modify a previous file. (See U:\CLU2\SC_B\DOC\ram1005.doc on Tracking Simulator PC and other files in this subdirectory). The ESOC file should actually be .pdf so convert it using something like CutePDF on Windows or OpenOffice on Linux.

D.5 Uploads of FGM and STAFF Parameters

STAFF data and addresses are no longer of interest since the failure of the analog-to-digital converters. The value of FGM (and STAFF) numbers historically can be determined from the Cluster EDI In-Orbit Operations Documentation, Software Upload Log webpages. Look in the column headed "FGM data."

Addresses for FGM and STAFF data:

FGM/STAFF item		**EEPROM constants**				**RAM copy**			
	FGM range	Matrix	FGM offsets	STAFF gains	STAFF offsets*	Matrix	FGM offsets	STAFF gains	STAFF offsets*
Pointers						mtrxadd=$df40	foffadd=$df44		
FGM matrix/	**2**	$87720	$877b0	$877b6		$de94	$d51e	$d524	
FGM offsets	**3**	$87744	$877c2	$877c8		$deb8	$d530	$d536	
	4	$87768	$877d4	$877da		$dedc	$d542	$d548	
STAFF gains	**5**	$8778c	$877e6	$877ec		$df00	$d554	$d55a	
	6 (%)		$877f8	$877fe			$d566	$d56c	
	7 (#)		$8780a	$878a0			$d578	$d57e	
STAFF offsets					$8781c				
Working copy								$d288	$d3fa

* STAFF offsets are not range dependent.
(%) FGM not calibrated in range 6
(#) FGM range 7 does not exist in space. No EEPROM space reserved for range7 matrix.

In Mode8 at POWERON (or a jump to address $6000), all of the rotation matrix, FGM offsets and STAFF gains are copied from EEPROM to RAM (addresses $deXX and $d5XX). Address pointers for rotation matrix and FGM offsets are set to the Range3 numbers in RAM. STAFF gains and offsets are set to Range3 numbers ($d288 and $d3fa). When a range change occurs, the pointer addresses are changed to point to the numbers for the new range and the STAFF gains are re-copied. FGM/STAFF information in RAM can be modified by telecommand. The new numbers will take effect the next time FGM changes to the changed range.

In Himodes initializations, all of the STAFF gain data are again copied from EEPROM to RAM ($d51e, etc). Address pointers are initialized to one set of data based on the current range and the STAFF gains are set to the current range. If himod operation without STAFF has been selected, all the STAFF gains are reset to zero. Himodes operation can be either 4ms or 2ms. The STAFF gain numbers are stored at the correct magnitude for 4ms BCIs. In 2ms BCIs the STAFF gain numbers are divided by 2 before they are used.

Some care should be taken changing FGM variables by telecommand. It cannot be done in Mode0. It should be done in Mode8 only. Even in Mode8, changing the FGM offsets or the rotation matrix means potentially changing numbers while they are in use. In order to minimize the possibility of conflict during change, it is a good idea to set the interrupt mask so as to turn off the FGM interrupt while making changes (TC IMSKS 9F) and to set the RAM FGM range value to 7 after changing values in RAM and before re-enabling the FGM ISR (TC IMSKS BF). Setting the range to 7 will cause the first FGM ISR to reset the address pointers to the correct, current range data.

D.5.1 Values Stored at Launch

Rotation matrix:

hex values			in other words		
$4000	0	0	1*16384	0	0
0	$3f96	$73e	0	cos6.5*16384	sin6.5*16384
0	-($73e)	$3f96	0	-sin6.5*16384	cos6.5*16384

FGM offsets and STAFF gains and offsets :

FGM offsets for all axes in all ranges: $800

STAFF offsets for all coordinates is $200

STAFF gains are different for each range but the same for all coordinates.

range2:	$0444	decimal	1092	~~	1088
range3:	$0111	decimal	273	~~	272
range4:	$0044	decimal	68	~~	64
range5:	$0011	decimal	17	~~	16
range7:	$0001				

Numbers which more closely reflect reality can be found at the EDI web site:

mpebj.mpe-garching.mpg.de

select Software Upload Log and then FGM data.

E. Comprehensive Performance Test (CPT)

Title Page

EXPERIMENT : EDI
PROCEDURE : EDI_CPT (Comprehensive Performance Test)
PURPOSE : Comprehensive Performance Test of Instrument Hardware
APPLICATION : First Step in Commissioning
CONSTRAINTS :

- Ground contact with spacecraft
- Duration about. 120 minutes
- Spacecraft must be in NM telemetry

Procedure

NOTES :

- For each Control File, operator to check that HK parameters CVCNT and LASTC are correct
- For most of the steps the EDI team will perform additional checks.
- The referenced ICF files are stored on VAX station MPECL4 in directory [CLUSTER.MODULES] on drive DKA400:

Step No.	Description	ICF file	LASTC	CVCNT	Notes
1	Confirm that S/C is in NM telemetry and that Gun temperatures are within acceptable limits (S/C powered thermistors)	---	---	---	
2	EDI power on, send TC ZEENOOPD within 30 seconds	---	$0100	$01	1
3	check RAM 02C000 – 033FFF	FRAM02C.ICF	$0ADD	$11	2
4	configure PGAi	FEDION.ICF	$0080	$2B	1
5	check GDU CMD/DATA RAM	FGDRAM3.ICF	$0ADD	$4B	2
6	set variable section to default	FGDUHK.ICF	$1B00	$4C	---
7	configure PGA1	FPGA1.ICF	$0D01	$50	1, 5
8	configure PGA2	FPGA2A.ICF	$0000	$69	2, 5
9	GDE test pulser operation	FGDPLS.ICF	$01FE	$7B	3
10	set safe values for GDU CMDs	FGSAFE.ICF	$0E00	$7C	---
11	Sensor test pulser operation	FSNPLS.ICF	$2A40	$0E	3
12	set safe values for GDU CMDs	FGSAFE.ICF	$0E00	$0F	---
13	set sensor preamp gain to 4	FGAIN4.ICF	$1840	$19	---
14	check preamp noise floor	FACCST.ICF	$0648	$5B	3
15	set sensor HK back to stepping	FSNHK.ICF	$1C38	$5D	---
16	exercise sensor power strobing	FPWSTR.ICF	$0002	$8F	---
17	turn on HV stacks	FHVON3.ICF	$1403	$90	1
18	set up moderate high voltages	FHVAIR.ICF	$0064	$EA	1
19	test cathode integrity	FCATH.ICF	$02D0	$EE	---
20	set safe values for GDU CMDs	FGSAFE.ICF	$0E00	$EF	---

21	exercise low power command 2 on main interface: ZEDILP2A	---	---	---	check EEWRP=0
22	exercise low power command 2 on main interface: ZEDILP2A	---	---	---	check EEWRP=1
23	exercise low power command 1 on main interface: ZEDILP1A, switch S/C to redundant interfaces, wait >45 seconds until EDI TM reappears.	---	$0000	$00	loss of Telemetry after ZEDILP1A
24	test scratch RAM	FPSRAM3.ICF	$0ADD	$13	4
25	test remaining processor RAM	FPURAM3.ICF	$0ADD	$3D	4
26	set variable section to default	FGDUHK.ICF	$1B00	$3E	---
27	send telecommand ZEEPGALS	---	$ABCD $0000	$3F $40	1, 5, 6
28	test interrupt counters	FIRCTR.ICF	$1BC0	$46	4
29	exercise low power command 2 on redundant interface: ZEDILP2B	---	---	---	check EEWRP=0
30	exercise low power command 2 on redundant interface: ZEDILP2B	---			check EEWRP=1
31	dump version numbers	VERSION.ICF	$1BC0	$98	---
32	jump to mode 0	JM0.ICF	$04F2	$9B	---
33	turn off EEPROM power	EEOFF.ICF	$0500	$9E	---
34	EDI power off	---	---	---	---

Notes:

1 abort procedure in case of a failure
2 repeat step in case of a failure; if the error persists, abort procedure
3 abort procedure in case of excessive count rates
4 repeat step once in case of a failure, then proceed
5 temporary loss of telemetry possible (1-2 TM cycles)
6 TC ZEEPGALS takes about 30 seconds to execute. During command execution LASTC will be $ABCD and CVCNT will be incremented by one. At the end, LASTC will change to $0000 and CVCNT will again be incremented (while CRCNT will not!)