



ESTEC
SCI-SAG/SAX



Alex Jeanes
Guillermo Buenadicha

Clock Sequence Generator and Bad Pixel Table ICD

XMM-SOC-ICD-0016-SSD

Issue 2

Revision 0

February - 2001

European Space Agency
Agence spatiale européenne

ESTEC

Postbus 299 - NL2200 AG Noordwijk - Keplerlaan - NL2201 AZ Noordwijk ZH -
Tel. (31) 71 5656565 - Fax (31) 71 5656040

XMM-SOC-ICD-
0016_issue2_0_review_05_02_01
.doc

DOCUMENT APPROVAL

Prepared by	Organisation	Signature	Date
A. Jeanes	SCI-SAX / Aurora		
G. Buenadicha	XMM-ISS/Vega		05 /02/01

Approved by	Organisation	Signature	Date

CHANGE RECORD SHEET

Date	Issue/ Rev	Page/Para affected	Description	Associated CR No	Approval Authority
09 Feb 99	1.1		Updates from RGS review		
05 Feb 01	2.0	Sections 4, 5 and 6	Inclusion of EPIC MOS icd's for CSG, BPT and OT, author G. Buenadicha		

CONTENTS

DOCUMENT APPROVAL.....	i
CHANGE RECORD SHEET.....	ii
CONTENTS.....	iii
1. INTRODUCTION.....	1
2. RGS CSG.....	2
2.1 Generation.....	2
2.2 Onboard.....	2
2.3 Uplink.....	3
2.4 Interface Details.....	3
2.4.1 Introduction.....	3
2.4.2 Source and Generation.....	3
2.4.3 Destination.....	3
2.4.4 Transfer Mechanism.....	3
2.4.5 Frequency.....	4
2.4.6 File Format.....	4
2.4.6.1 CSG Sequence Files.....	4
2.4.6.2 CSG Compilation File.....	4
2.4.6.3 CSG Transfer File.....	4
2.4.7 File names.....	4
2.4.7.1 CSG Sequence Files.....	4
2.4.7.2 CSG Compilation File.....	5
2.4.7.3 CSG Transfer File.....	5
3. RGS Bad Pixel Tables.....	6
3.1 Introduction.....	6
3.2 Relationship to Telecommands.....	6
3.3 Generation.....	7
3.4 Onboard.....	7
3.5 Uplink.....	8
3.6 Interface Details.....	8
3.6.1 Source and Generation.....	8
3.6.2 Destination.....	8
3.6.3 Transfer Mechanism.....	8
3.6.4 Frequency.....	8
3.6.5 File Format.....	8
3.6.5.1 Hot Pixel Tables.....	8

3.6.5.2	Hot Column Files.....	9
3.6.6	File Names.....	10
3.6.6.1	Hot Pixel Files.....	10
3.6.6.2	Hot Column Files.....	10
3.6.6.3	RGS Hot Pixel Transfer File.....	11
3.7	Example Hot Pixel Table Files	11
3.7.1	RGS Hot Pixel Table.....	11
3.7.2	RGS Hot Column Table	13
4.	EPIC MOS CSG.....	15
4.1	Generation.....	15
4.2	Onboard.....	15
4.3	Uplink.....	15
4.4	Interface Details	16
4.4.1	Introduction.....	16
4.4.2	Source and Generation.....	16
4.4.3	Destination.....	16
4.4.4	Transfer Mechanism.....	16
4.4.5	Frequency.....	17
4.4.6	File Formats and Names.....	17
4.5	SOC Processing.....	18
5.	EPIC MOS BPT.....	19
5.1	Generation.....	19
5.2	Onboard.....	19
5.3	Uplink.....	19
5.4	Interface details.....	19
5.4.1	Introduction.....	19
5.4.2	Source and generation.....	19
5.4.3	Destination.....	19
5.4.4	Transfer mechanism.....	20
5.4.5	Frequency.....	20
5.4.6	File formats and names	20
5.5	SOC processing.....	21
6.	EPIC MOS OT.....	23
6.1	Generation.....	23
6.2	Onboard.....	23
6.3	Uplink.....	23
6.4	Interface details.....	23
6.4.1	Introduction.....	23
6.4.2	Source and generation.....	23

6.4.3	Destination.....	24
6.4.4	Transfer mechanism.....	24
6.4.5	Frequency.....	24
6.4.6	File formats and names	24
6.5	SOC processing.....	24
LIST OF REFERENCES		26

1. INTRODUCTION

The purpose of this document is to define the transfer of Clock Sequence Generator (CSG) and Bad Pixel (as well as the Offset Tables in the case of EPIC MOS) files within the XMM Ground Segment, from the SAS or other parties to the ISS.

To enhance the understanding of the subject matter, the processing of the transferred items is also described.

This issue of the document addresses the needs for the RGS and EPIC MOS. Other instruments could be foreseen in future issues.

The CSG and Bad Pixels are in some respects related, as a failure in a large number of pixels may require a change to the CSG. However this is not always the case; therefore the CSG and the Bad Pixels are separate items for the purposes of update/generation/transfer/configuration.

The order of the document has been arranged to tackle one section of the interface in its entirety at a time. This makes for easier and cleaner reading.

2. RGS CSG

The CSGs are generated using the CSG Compiler. The CSG is controlled by the Instrument Controller (IC) on board. The CSGs reside in memory in the RGS Analogue Electronics (RAE).

Onboard it is possible to have approximately 50 CSGs loaded at once. However the CSG Compiler, in its current format, supports a maximum of 10. The details of how the CSG are generated is detailed below.

2.1 Generation

The underlying details of the CSG instruction set are detailed in [RD1 section 10.5]. As these are very detailed, a CSG Compiler was produced which generates CSG instructions from a higher level language. The CSG sequence files use this higher level language to allow users to generate the CSG in a simpler manner.

The RGS CSG Compiler is included in the ISS within the CSG Compiler unit. This requires CSG sequences as inputs and generates the CSG memory section. These sequences can share macros or definitions from other sequences, or defined separately in common include files.

The CSG compiler produces a CSG .csg file containing a section of CSG memory. The current .csg file contains a (2048,5) byte array followed by additional parameters for the RDE-sim. The ISS must modify this as follows:

- convert the byte array to (5,2048)
- strip the additional parameters
- produce the telecommand parts which contain only zero's.

The sequences are allocated Ids sequentially in the order that they are listed in the compile statement. This automatically determines the start addresses of each of the input sequences. Pointers to these addresses are then stored in addresses 1 through 10. Hence if you wish to use CSG-ID 5, your start address is also 5. Start address 5 then jumps to sequence number 5.

Therefore to generate a new CSG memory, all sequence files required for the CSG must be shipped to the ISS, together with a compile definition. It is important that the sequences are listed in the correct order to ensure that their CSG-Id is correct. Procedures must ensure that changed sequence files are not used by other versions of CSG which do not require update. These procedures will be documented in the IFOP.

2.2 Onboard

A memory, 40 bits wide, and 2048 deep is used to store the CSGs. The layout of the 40 bits is described in [RD-1 Section 10.5]. The CSG address range is 0 to 07FFh inclusive. As explained in

[RD-2 section 6.2.1], the IC processor reserves some fixed addresses for interrupt handling. This must be accounted for in the IC Main software image. The different sequences within the CSG are identified by a CSG-Id.

2.3 Uplink

The CSG memory is transmitted to the RGS using Type (6,1), load memory TCs. This is as a part of the IC main code. The clock sequence generator patterns reside in the RAE memory.

2.4 Interface Details

2.4.1 Introduction

The RGS CSG Sequence files are very similar to software code. A file can contain one of the following:

- Macro definition
- Pattern definition
- Integer constant definitions
- Time constant definitions

One file can make use of another file for its own purposes.

RD7 contains a suite of CSG sequence files which provide excellent examples of the files which will be transferred over this interface.

2.4.2 Source and Generation

The CSG sequence files are generated by the SOC on the SAS.

2.4.3 Destination

The CSG sequence files are sent to the Instrument Software Subsystem (ISS).

2.4.4 Transfer Mechanism

The suite of CSG sequence files will be transmitted in their entirety. These CSG sequence files must be stored independently of previous CSG sequence files to ensure new and old versions can be compiled as required.

The suite of CSG Sequence Files for a CSG program are sent as one CSG Transfer File. A CSG Compilation file is also included in the CSG Transfer File. The CSG Transfer file is transferred using FTP.

2.4.5 Frequency

Changes to the CSG programs will not be performed frequently. Annual changes can be expected, plus additional changes as a result of an anomaly.

2.4.6 File Format

2.4.6.1 CSG Sequence Files

CSG sequence files are ASCII format. The format of the files is defined in RD6.

2.4.6.2 CSG Compilation File

This file details how the CSG program should be compiled. It is an ASCII file and will contain a statement as follows:

```
csc input-file input-file ..... output-file
```

where:

input-file is one of the CSG sequence files with extension “.cs”
there can be any number of input-files (compiler currently handles a maximum of 10)
output-file is in the format RGS_n_CSG_vvv.csg

Note: The .incl files are not listed; they are automatically included in the appropriate places by the CSG Compiler.

2.4.6.3 CSG Transfer File

The CSG Transfer File is a TAR file containing all of the CSG Sequence files for a specific CSG image, plus one CSG Compilation File.

2.4.7 File names

2.4.7.1 CSG Sequence Files

The CSG sequence files shall be named in accordance to RD6, but constrained to the following definition:

```
RGS_n_CSG_vvv_description.ext
```

Where:

n is either “1” or “2”, and defines the RGS experiment
vvv is the version number, as an integer with leading zeroes
description is a variable number of the following characters [_a-zA-Z0-9]
ext is “cs” for macro definitions, and “incl” for include files

Example file names are :

RGS_1_CSG_001_1x1cd14.cs
RGS_1_CSG_001_1x1cdSlS2h02.cs
RGS_1_CSG_001_clocks.incl
RGS_2_CSG_999_timing.incl

2.4.7.2 CSG Compilation File

The CSG Compilation file shall have the following name:

RGS_n_CSG_vvv.comp where:

‘n’ is either ‘1’ or ‘2’ depicting the RGS instrument,
is an INTEGER with leading zero which represents the Version number of the CSG image
for the specified RGS instrument.

e.g. RGS_1_CSG_001.comp

2.4.7.3 CSG Transfer File

The CSG Transfer file shall have the following name:

RGS_n_CSG_vvv.TAR where:

‘n’ is either ‘1’ or ‘2’ depicting the RGS instrument,
is an INTEGER with leading zero which represents the Version number of the CSG image
for the specified RGS instrument.

e.g. RGS_1_CSG_001.TAR

3. RGS Bad Pixel Tables

3.1 Introduction

The bad pixel tables are processed on board by the Data Pre-Processor (DPP).

The bad pixel tables are organised into two parts. The first contains hot columns, or segments of hot columns. The second part contains individual bad pixels.

There are different versions of the hot pixel tables, per RGS ($n = 1$ or 2), as follows:

Code	File Name in RGS documentation	Comment
Hot pixel table	FMnHPT.bin.1x1	1x1 OCB all known hot pixels
Hot pixel table	FMnHPT.bin.3x3	3x3 OCB selected hot pixels only
Hot column table	FMnHCT.bin.1x1	1x1 OCB 5 hot columns, 3 columns with hot segments
Hot column table	FMnHCT.bin.3x3	3x3 OCB 1 hot column, 5 columns with hot segments

The correct pixel tables must be uploaded for the mode and OCB factor, whenever a change in operation is required, as the DPP only holds one full lookup table. This is potentially complex, if during one Exposure, the CCDs were to be operated with different OCB, however this is not supported in the flight software.

The DPP is controlled using a set of 16 parameters per CCD. These are uplinked using TC (5,1) with TID = 40, FID = 40/41. NOTE there is a dependency between the "Hot pixel table start C and D" parameters with the Hot Pixel Table in the DPP lookup table as explained below:

3.2 Relationship to Telecommands

The TC Packets G0538 (CCDs 1 to 4), and G0539 (CCDs 5 to 9) detail the processing of the Hot Pixel Tables as follows where 'n' is a CCD number:

Parameter reference	Name	Remarks
G1n41	CCD_n hot item processing	On or Off
G1n42	CCD_n hot pixel table address C side	The start address of the first hot pixel for node C
G1n43	CCD_n hot pixel table address D side	The start address of the first hot pixel for node D

G1n44	spare	spare
G1n45	spare	spare
G1n46	CCD_n hot column shift size	This defines the number of pixels in the segment. The value i defines the segment size using the formula, $\text{segment size} = 2^i$. Hence if the parameter value i is 3, the column segment size is 8, likewise if i is 4, the column segment size is 16.

The above table is applicable to RGS 1. For RGS 2, the 'G' for TCs and parameters should be replaced by 'L'.

3.3 Generation

The tables are originally generated by the RGS PI. The processing is similar to that of the Hotshot tool, which is described in RD4 starting on page 17. Two binary tables are produced, one for hot pixels, the second for hot columns or segments of columns.

During operations new versions of the tables will be generated by the XMM SOC using the SAS. The outputs will conform to this ICD.

3.4 Onboard

The DPP contains a lookup table.

Table contents	Start address in hex	End address in hex	Notes
Hot pixels	40h	4799h	2 16 bits words per pixel, y coord, then x-coord. 2 16 bit words with values FFFFh signal the end of the bad pixels for a CCD node. See RD-3 for details.
Hot columns, or segments of	4800h	6FFFh	Each CCD has 1024 16 bits words. CCD_1 is 4800h to 4BFFh inclusive, CCD_2 is 4C00h to 4FFFh inclusive, etc. The layout of column to address is defined in RD1 Table 10.4.1

3.5 Uplink

As explained in RD2, the DPP lookup table is transmitted to the RGS using Type (6,1), load memory TCs. The complete DPP memory is 64 Kwords (0 to FFFFh). The DPP memory loads are supported from the IC main software only.

The DPP also has parameters related to the hot pixel table which are loaded in two TC (5,1) TCs. These parameters are described above.

3.6 Interface Details

3.6.1 Source and Generation

The files are generated by the SOC on the SAS.

3.6.2 Destination

The files are sent to the Instrument Software Subsystem (ISS).

3.6.3 Transfer Mechanism

The files are transferred using FTP. All four files per instrument are always transferred. Each of these files containing the same version number. This is achieved via an RGS Hot Pixel Transfer File. This is simply a TAR file containing a set of four Hot Pixel Files.

3.6.4 Frequency

Changes to the Pixel files are expected to be performed on average 6 times per year. Typically all four files will be modified, as they are not independent.

3.6.5 File Format

3.6.5.1 Hot Pixel Tables

The RGS Hot Pixel Tables have a binary format. Each file contains one stream of data representing the image, without any formatting.

Files containing the Hot Pixel image are the size required to contain the Hot Pixels, which is typically less than the maximum of 47B0h. The hot pixels are listed per node. For a pixel the y value is written before the x value. Further they must be listed in the sequence in which they will be read out by the electronics (increasing Y and increasing X for Y). Two 16 bit words with value FFFFh, end the bad pixels for a node.

3.6.5.2 Hot Column Files

The Hot column files are always a complete set of 1024 16 bit words per CCD. Each file contains one stream of data representing the image, without any formatting.

It is possible to reject a complete column.

It is possible to reject segments of a column. The size of the segment is definable per CCD and must be compatible with that defined in the “hot column shift size” parameter G1n46 (see above).

If none of the columns or segments of a column are to be rejected, then the control word value is 0.

If a column is to be rejected, then the control word value is 65535 (FFFFh). A special situation occurs in HTR mode, where the most significant bit of the hot-column entry, determines whether or not the column is rejected.

If rejecting segment j of a column, then a 1 is assigned to the jth bit (order right to left) of the control word. Therefore 3 indicates rejection of the first two segments, 12 indicates the rejection of the 3rd and 4th segments. The number of segments in a column is dependent upon the segment size.

The columns are defined as if read out from the C node. The DPP x-counter always runs from 0 to xlen-1. The D node always starts at address 512; therefore there is a gap between the end of the C node and the start of the D node when OCB is not 1.

The RGS potentially requires 6 different memory maps for the Hot Columns. However the layout for 1x1 OCB C and D output is exactly the same as for the 1x1 C output; and 3x3 OCB for C only, is the same as for D only; therefore a maximum of four memory maps are required. The layout of these memory maps is defined in RD-1 Table 10.4.1. For the operations at the start of the mission, only 1x1 C and D, plus 3x3 C and D will be used.

For 1x1 OCB C and D output, we have the following:

Memory Address offset from start of CCD	CCD Column
0	C 0
511	C 511
512	D 0
1023	D 511

For 3x3 OCB C and D output we have the following:

Memory Address offset from start of CCD	CCD Column
0	C 0
171	C 171
172 to 511	Not Applicable
512	D 0
683	D 171
684 to 1023	Not Applicable

Explanation: address 0 in the memory map represents C column 0.

3.6.6 File Names

For the purpose of the ICD, a different file name convention is defined to that within the RGS instrument team. This provides more information on the contents.

3.6.6.1 Hot Pixel Files

The files containing the individual bad pixels are named as follows:

RGS_n_HPT_vvv.ixi

where:

n is either "1" or "2", and defines the RGS experiment
vvv is the version number, as an integer with leading zeroes
i is either "1" or "3" in both instances

Example file names are:

RGS_1_HPT_001.1x1
RGS_2_HPT_002.3x3

3.6.6.2 Hot Column Files

The files containing the column details of bad pixels are named as follows:

RGS_n_HCT_cd_vvv.ixi

where:

vvv is the version number, as an integer with leading zeroes
i is either “1” or “3” in both instances

Example file names are:

RGS_1_HCT_CD_001.1x1
RGS_2_HCT_CD_001.3x3

NOTE: The current baseline is to always use both the C and D node. However the filename specification allows for future changes.

3.6.6.3 RGS Hot Pixel Transfer File

The RGS Hot Pixel Transfer file shall have the following name:

RGS_n_HPT_vvv.TAR where:

‘n’ is either ‘1’ or ‘2’ depicting the RGS instrument,
is an INTEGER with leading zero which represents the Version number of the Hot Pixel Tables for the specified RGS instrument.

e.g. RGS_1_HPT_01.TAR

3.7 Example Hot Pixel Table Files

3.7.1 RGS Hot Pixel Table

Explanation of the delivered file: FM1HPT.bin.3x3 (e.g. RGS_1_HPT_001.3x3)

An octal dump using the Solaris command `od -x FM1HPT.bin.3x3` gives:

```
0000000 ffff ffff 0010 0025 0011 0025 0012 0025
0000020 0013 0025 ffff ffff ffff ffff 0000 0069
0000040 0001 0069 ffff ffff ffff ffff ffff ffff
0000060 ffff ffff ffff ffff ffff ffff 0017 0076
0000100 0040 0076 0041 0076 0042 0076 0052 0076
0000120 ffff ffff 0077 00a4 0078 00a3 0078 00a4
0000140 0079 00a4 ffff ffff ffff ffff ffff ffff
```

0000160 ffff ffff ffff ffff ffff ffff ffff ffff
0000200 ffff ffff
0000204

Notice the 18 markers ffff ffff signalling the end of the bad pixels for the CCD node. The hot pixel co-ordinates are in between, y followed by x. The above table can be annotated as follows (where the hot pixels are listed per node with X,Y co-ordinates):

1_C
1_D (37, 16) (37, 17) (37, 18) (37, 19)
2_C
2_D (105, 0) (105, 1)
3_C
3_D
4_C
4_D
5_C
5_D (118, 23) (118, 64) (118, 65) (118, 66) (118, 82)
6_C (164, 119) (163, 120) (164, 120) (164, 121)
6_D
7_C
7_D
8_C
8_D
9_C
9_D

Table size: 66

Start addresses: This data can be derived from the table and should be equal to the DPP task functional parameters hot-pixel start address C and D. These relate to parameter references G1n42 and G1n43, described above.

CCD	C	D
1	0	2
2	12	14
3	20	22

4	24	26
5	28	30
6	42	52
7	54	56
8	58	60
9	62	64

3.7.2 RGS Hot Column Table

Explanation of the delivered file: FM1HCT.bin.3x3 (e.g. RGS_1_HCT_CD_001.3x3)

An octal dump using the Solaris command `od -x FM1HCT.bin.3x3` gives:

```
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
0002100 0000 0000 0000 0000 0000 0000 0003 0000 0000
0002120 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0004500 0000 0000 0000 0000 0000 0000 007e 0000 0000
0004520 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0012260 0000 0000 0000 0000 0000 ffff 0000 0000 0000
0012300 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0022340 0000 0000 0000 0000 0000 0000 0000 00f8 0000
0022360 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0032220 0000 000f 0000 0000 0000 0007 0000 0000 0000
0032240 0000 0000 0000 0000 0000 0000 0000 0000 0000
*
0044000
```

This corresponds to:

Memory Location Octal	Memory Location Decimal	CCD number	Column	Value	Segments
2112	1098	1	549	00003	1 and 2
4512	2378	2	165	00126	2 to 7

12270	5304	3	604	65535	Whole column
		4			
22354	9452	5	630	00248	4 to 8
		6			
32222	13458	7	585	00015	1 to 4.
32230	13464		588	00007	1 to 3
		8			
		9			

Note to convert from Memory Location decimal to Column, determine the CCD by 2048 per CCD, then divide the remainder by 2 as there are 16 bits per column.

4. EPIC MOS CSG

The EPIC MOS Clock Sequences are a set of instructions for the EMAE clock sequencer that are kept in the memory. They define a programmed sequence for CCD readout for the possible modes of the instrument.

The sequences are delivered to the XMM SOC as binary files, output of a CSG Compiler program. It is the task of the Instrument Teams to provide the sequences every time that one could be needed.

The binary files are thereafter converted at the SOC into sequences suitable for uplink On Board.

4.1 Generation

The generation process for the sequences involves the use of a CSG Compiler. This is a specific program that is able to create the necessary binary file out of a high level language. This later is very useful to describe properly the sequence behaviour.

The CSG compiler (version 2.07 is installed in the ISS) takes as input a text file containing the program code for one sequence, and another text file containing the parameters. There are also some compilations directives as flags. This split makes easier to produce small modifications to one sequence, just changing the parameter file, or use it with different code files. Further knowledge on the compiler can be found in **RD-8**.

The output of the compilation process is a binary file, which is the deliverable for the ISS, together with a list file, which is a text/human readable format of the sequence, useful for later analysis.

4.2 Onboard

The sequences are uplinked On Board to the EMDH unit, which then in turn transfers them to the EMCR. The sequences are stored in the EMCR memory, which is able to keep up to 5 sequences (although only the last slot is used). They are thereafter applied to the relevant sequencer in the EMAE unit.

4.3 Uplink

The binary files, output of the CSG Compiler are processed by a specific tool at the SOC to convert them into DB. Sequences.

The sequences are uplinked On Board to a fixed memory position in the EMDH memory (starting at dec. 80432). Thereafter, a transfer is performed from there to the appropriate buffer in the EMCR memory, together with the information of the target sequencer to be addressed.

The target sequencers are described in the following table:

Target	CCDs
1	1
2	2-5
3	4-7
4	3-6
5	FW

The sequences are uplinked using series of TC EU056 (for MOS1). Then the TC E0074 is used to indicate the target slot in the EMCR memory for the sequence, and TC E0082 to show the target sequencer.

4.4 Interface Details

4.4.1 Introduction

This section describes the interface characteristics and also the file format and contents.

4.4.2 Source and Generation

The Instrument Team is in charge of the generation of the files. To do so, they will use the CSG Compiler.

4.4.3 Destination

The destination will be the XMM SOC, being the addressed persons the ISS engineer and the EPIC MOS engineer.

4.4.4 Transfer Mechanism

The transfer can be done via e-mail, with the release in the form of a compressed file (tar, zip, gzip...). The mail should address the persons named in 4.4.3. The transferred file should be complete (i.e. contain all the deliverables described in 4.4.6) and stored independently from previous deliveries.

4.4.5 Frequency

The Clock sequences do not have a predefined or periodic schedule for delivery. The need for them may arise from operational modifications, non conformances discovered or proposed enhancement of the instrument. Previous frequency shows more than one delivery per year.

4.4.6 File Formats and Names

The file will be a compressed file (tar, zip, both..) named at least with the release date in order to make it unique, plus other relevant information if desired.

The name will be therefore in the form of:

DD_MM_YY.ext

being * wildcards, and ext any applicable for compressed files.

The string of the delivery date (DD_MM_YY) conforms the release_date variable for the following file contents.

The file will contain the following structure and files:

release_date/ → will be the top directory (eg 20_09_00)

Contents:

release_date.txt → brief file explaining changes/add ons for the delivery

../code → Directory containing the source code for the compiler

../sources → Directory containing the output binary files

../list → Directory containing the output list files

../parameters → Directory containing the parameter files for the compiler

For all the files in these directories, the file extension is optional, but if present, it should be related to the type of file (e.g. lst for list files, asy

The files under /sources and /parameters can have any name selected by the developer, according to internal conventions (preferably adding some meaning), but every time that one of them is modified, the name should change with respect to previous deliveries.

The files under /code and /list should follow the name convention:

<source_file_used><parameter_file_used><release_rate>

4.5 SOC Processing

Once at the SOC, the delivery will be uncompressed, and the files to be used for Sequence generation are the binaries, i.e. those under the

../sources

directory. However, in some cases the list files will be used as well to give a human readable format of the file.

The processing options are two.

Use of the DB Sequence Generation Tool

In this case, the binary file is used to generate directly the tar file expected by the DB import facility, containing 4 txt files with the relevant information for the sequence.

See **RD-10** for further details.

Use of the ISS Image Handling utilities plus RAM loader

In this case, the binary file, together with the list file, are used to generate a memory image (since the Clock Sequences are uplinked to a fixed memory position in the EMDH) in the format expected by the XMCS RAM loader. After this, the RAM loader is used to generate the memory maintenance TC's needed for the sequence.

The use of one or the other will depend on operational considerations from the SOC.

5. EPIC MOS BPT

The EPIC MOS Bright pixel Tables are a set of values defining the positions in the instrument CCD's which are declared as not operational.

It is the task of the Instrument Teams (calibration) to provide the tables every time that one could be needed.

The files are thereafter converted at the SOC into sequences suitable for uplink On Board.

5.1 Generation

The files are generated by the Instrument Calibration teams out of the processing of the S/C data. They are text files.

5.2 Onboard

Description of the On-Board processing can be found in **RD-9**.

5.3 Uplink

In each MOS instrument, there are 8 Bright pixel Tables, one per HBR. Therefore, the uplink is done via specific TC's, forming all of them one sequence.

5.4 Interface details

5.4.1 Introduction

This section describes the interface characteristics and also the file format and contents.

5.4.2 Source and generation

The BPT's are generated by the calibration team of the EPIC MOS.

5.4.3 Destination

The destination will be the XMM SOC, being the addressed persons the ISS engineer and the EPIC MOS engineer.

5.4.4 Transfer mechanism

The transfer can be done via e-mail, with the release in the form of a compressed tar. The mail should address the persons named in 5.4.3. The transferred file should be complete (i.e. contain all the deliverables described in 5.4.6) and stored independently from previous deliveries.

5.4.5 Frequency

The BPT do not have a predefined or periodic schedule for delivery. The need for them will arise from the calibration strategy for the instrument. Previous frequency shows more than one delivery per year.

5.4.6 File formats and names

The delivery from the calibration team will be in the format of a tar compressed file without file extension for each MOS instrument.

The tar file will contain a set of 7 text files, each one with the data for each HBR (1 to 8) except for number 2, which corresponds to the CCD1 redundant node.

The CCD1 redundant node is not provided in the delivery, it will have to be recreated in the SOC processing.

The tar file will take the following naming convention:

mI_vNN_sc_upload

with :

I = 1 or 2 for MOS1 and MOS2

NN = version number (currently 04 , vesrion 03 was never used)

And the filenames per CCD in the tar file should be :

mI_vNN_sc_upload.hbrX,

i.e. identical to the tar filename with extension hbrX, where X indicates the HBR number (different from the CCD number)

X = 1, 3 to 8.

Each file will be a text file, with the following contents:

Num_pairs → decimal number indicating the number of X-Y coordinate pairs in the file.

<empty line> → must be present, needed for the SOC processing.

X <space> Y → X and Y coordinates for one pixel, decimal.

X <space> Y → X and Y coordinates for one pixel, decimal

....

X <space> Y → X and Y coordinates for last pixel, decimal.

The pairs MUST be ordered in increasing Y coordinate.

An example of one file would be:

11

```
501 260
187 277
245 385
245 386
500 391
500 394
442 437
442 440
442 443
442 453
384 458
```

The requisites are that the empty line in the second line must be present, and that the pairs must be separated at least with one blank space.

5.5 SOC processing

The SOC will take the tables as delivered by the calibration teams. The first step will be to recreate the missing HBR2 out of the HBR1, by complementing to 610 the X co-ordinate.

Then, the co-ordinate values will be converted from decimal to hexadecimal, and with the result as input, a sequence formed by 8 TC's will be built, including in each one the valid pairs and padding up to 50 pairs with 0's.

All this is implemented in the DB sequence generator tool, which will perform all the necessary steps and produce as output the compressed file expected by the Database for import, with the relevant naming convention.

Since the BPT are not uplinked via memory maintenance commands, the XMCS RAM loader cannot be used for the BPT.

6. EPIC MOS OT

The Offset Tables for the EPIC MOS instrument represent bias corrections for the CCD's in each observing mode. They are tables composed of column and row offsets which are used by the EDU unit to calculate the offset for each pixel.

6.1 Generation

The Offset Tables are generated by the instrument teams and the instrument calibration teams, based upon calibration exposures. They build a table of 610 Column and 602 row offsets per CCD and per mode. Sometimes, only the central CCD (i.e. number 1) will be calculated and provided.

6.2 Onboard

Description of the On-Board processing can be found in **RD-9**.

6.3 Uplink

The offset table is loaded as memory load to the EMDH tables are and then copied to the appropriate EDU via the EMCR.

There is a specific command which takes a start address and 101 parameters to load the EMDH tables. 12 of these commands are required to complete an offset table. Since there is a limitation in the number of parameters per sequence, the 12 TC's have to be split among two sequences.

The values are uplinked to fixed memory positions in the EMDH memory, starting at dec. 80432, and then the copy is commanded to the relevant EDU.

6.4 Interface details

6.4.1 Introduction

This section describes the interface characteristics and also the file format and contents.

6.4.2 Source and generation

The OT's are generated by the calibration team and/or the instrument teams of the EPIC MOS.

6.4.3 Destination

The destination will be the XMM SOC, being the addressed persons the ISS engineer and the EPIC MOS engineer.

6.4.4 Transfer mechanism

The transfer can be done via e-mail, with the release in the form of text files. The mail should address the persons named in 5.4.3. The transferred file should be stored independently from previous deliveries.

6.4.5 Frequency

The OT do not have a predefined or periodic schedule for delivery. The need for them will arise from the calibration strategy for the instrument. Previous frequency shows more than one delivery per year.

6.4.6 File formats and names

Each delivered file will follow the naming convention:

mI_ccdX_vNN_name

with :

I = 1 or 2 for MOS1 and MOS2

X = ccd number (1 to 7)

NN = version number (the next version will be 03)

name = string containing the operating mode (LW,SW and probably TI in the future) and the boresight (PN or RGS) for the central CCDs

The file will be a text file, with 1212 records, being the first 610 the column offset values, and the remaining 602 the row offsets. The values are decimal (0 .. 65536).

6.5 SOC processing

The processing options are two.

Use of the DB Sequence Generation Tool

In this case, the text file is first decimal to hexadecimal converted, and then used to generate directly the tar file expected by the DB import facility, containing 4 txt files with the relevant information for the sequence.

See **RD-10** for further details.

Use of the ISS Image Handling utilities plus RAM loader

In this case, the text file is used to generate a memory image (since the offset tables are uplinked to a fixed memory position in the EMDH) in the format expected by the XMCS RAM loader. After this, the RAM loader is used to generate the memory maintenance TC's needed for the sequence. *This second option is still to be implemented and under analysis.*

The use of one or the other will depend on operational considerations from the SOC.

LIST OF REFERENCES

- RD1: XMM – RGS User Manual, RGS-SRON-PM-98/010, issue 1.1, Jan 1999.
RD2: RGS Telecommand Structure, RGS-MSSL-SW-004, V8.0, 24/08/1998.
RD3: Hot Item Processing in the MSSL DPP S/W, V1.1, 22/01/1998.
RD4: CCD Image Analysis Software Requirements, RGS-ROL-GSE-016, V9.1, 11/97.
RD5: XMM SOC ISS ADD, XMM-SOC-ISS-ADD, Issue 1.2, 4/98.
RD6: CSG Compiler Reference Manual, RGS-SRON-GSE-SP-013, V1.0, 17/01/96.
RD7: RGS FM: onboard software, RGS-SRON-OPS-RP-98/002, V1, 04/11/98.
- RD8: Sequencer Programs for the EPIC-EMAE. EPIC-LUX-SP-114 Issue 1.2 May 1999.
RD9: EMCS User Manual, EPIC-EST-OP-001, Issue 2, Sept 1999
RD10: CLOCK, BPT and OT DB SEQUENCE GENERATORS, Draft document, Jan 2001