PROJEKT *PROJECT*

ASIC

TITEL *TITLE*

SCTMTC ASIC User's Manual

Utfärdat av *Issued by*
Martin Rönnbäck

Funktion *Function*
ASIC Object Manager

Godkänt av *Approved by*
Håkan Kärnhagen

Funktion *Function*
Project Manager

Datum *Date*
2006-03-14

Saab Ericsson Space AB

Postadress *Postal address*	Telefon *Telephone*	Telefax *Telefax*	Organisationsnummer *Registered number*	Momsreg.-nr *VAT No*
SE-405 15 Göteborg Sweden	+46 (0)31 735 00 00	+46 (0)31 735 40 00	556134-2204	SE556134220401

SUMMARY

The SCTMTC ASIC User's Manual defines how the SCTMTC ASIC is to be used.

This document will eventually be replaced by the SCTMTC ASIC data sheet produced by Atmel.

DOCUMENT CHANGE RECORD

Changes between issues are marked with an outside-bar.

Issue	Date	Paragraphs affected	Change information
3	2003-02-07		Updated before the Design Review
4	2003-02-26	See change bars	Updated after the Design Review
5	2003-05-22	5.4.3, 6.3.2, 6.10.3	Updated description of NewFar interrupt
		6.10, 6.10.1	Added power-on reset register
		7.1.2	Updated TAP register
		7.1.3	Updated description of reset
		7.10	Updated signal list
		7.17	Added pinout
		4.7, 4.8.1, 4.9.6, 5.1.2, 5.4.1, 5.4.2, 5.4.3, 5.4.5, 5.4.6, 5.4.9, 6.1.3, 6.2.2, 6.3.2, 6.3.3, 6.4.1, 6.4.2, 6.4.3, 6.5.2, 6.5.3, 6.5.5, 6.6.2, 6.9.2, 6.10, 6.10.1, 6.10.2, 6.10.3, 6.10.5, 6.10.6, 6.10.7, 6.11.3, 7.5.1, 7.6.1, 7.6.2, 7.10, 7.10.2, 7.10.3, 7.10.4, 7.10.7, 7.10.8, 7.10.9, 7.11, 7.12, 7.13, 7.14, 7.15, 7.16	Updated or added section
6	2003-09-09	Marked	Updates from review
7	2003-11-03	5.4.2.3	Flush usage description updated
		5.4.2.5, 6.2.2, 6.10.2.5, 7.2.1.5	EEPROM write cycle anomaly
		5.4.3.1.1.3	Dynamic mode anomaly
		6.2.1, 6.2.2, 6.5, 6.6, 6.7, 6.8, 6.9	Clarification regarding initialisation and functions available to the user
		6.2.3.7, 6.2.3.8, 6.10.2	Correctable error addresses anomaly
		6.8, 6.10.8	Updated available functions in SPW module
		6.10.7.3	CfE flag in TME TM Status Register anomaly
		7.1.2.1	Corrected hexadecimal value for TAP register
		7	Added timing paramters
		7.12	Updated operating conditions

Released

Issue	Date	Paragraphs affected	Change information
8	2004-02-13	1	Added clarifying statement
		4.10	Updated list of abbreviations
		5.1.2, 5.3, 5.4.3.1.1, 7.3.1.5, 7.10, 7.17	Updated description of selection mechanisms
		5.2, 5.3, 5.4.2.4, 6.2.2, 6.10.2.5, 7.2.1	Updated description of non-volatile memories
		5.4.5.1	Added clarification
		5.4.5.5, 7.5.3.1	Clarified ussage of the status interface
		5.4.7.3.2.1	Updated polarity of Synchronisation flag
		5.4.7.12, 6.10.7.5	Clarified clock frequency relations
		5.4.8.2	Added clarification regarding bandwidth from SPW to TME
		6.1.4.2	Clarified description of refresh
		6.1.4.3	Updated usage constraints
		6.2.2.5.1	Clarified scrubber end address configuration
		6.2.3.7	Clarified register behaviour
		6.2.4.1	Updated usage constraint
		6.2.5.1	Added example
		6.3.2.1	Clarified interrupt usage
		6.7.2.2.3	Added clarification
		6.8	Clarified SPW configuration
		6.10	Clarified usage of unused register bits
		6.10.1.4	Updated register access type and added note
		6.10.2.4	Added clarification
		6.10.2.5	Added clarification regarding timeout
		6.11.1	Updated configuration block memory
		7.18	Added JTAG bit order
9	2004-05-07	6.1.3.7	Added section about power-on reset status
		6.2.2.7	Clarified buffer flush usage
		7.5.3.1	Added clarification about interface disabling
10	2005-03-03	5.4.1.4.3, 5.4.7.2, 6.7.2.1, 6.10.7.2	Changed term for idle packets on the telemetry link to Idle Packets
		5.4.7.2, 6.7.2.1	Clarified relation between VC buffer size and maximum TM packets for that VC
		5.4.7.3.4	Clarified VC ID for idle frames
		6.2.3.8	Added section about EDAC errors during read-fill
		6.10	Depict 32 bits in all register descriptions
		6.10.2.3	Clarified that register contents of PIM_FFCEAR are unaffected by a write
		6.10.2.5	Clarified that error trap registers are cleared when PIM_ET is written
		6.10.2.5	Added description of PIM_EI register contents
		6.10.6.3	Clarified differences between CPDM_SRR and CPDM_SR
		6.10, 6.10.7.3	Removed TME TM Status Register
		6.11.3.1	Added reference to section where the contents of the look-up table entries are described
		7.2.2	Added note about WEN behaviour when write waitstates is set to 0
11	2006-03-14	5.4.5.2	Updated CSEL action table
		6.2.3.3, 6.10.2.5	Clarified that TME does not affect error trap
		6.6.3.3	Clarified CPDM abort delay
		6.10.2.5	Removed TME from error trap table
		6.10.5.4	Added note about CS_RSTAT Timeout bit
		7.7.1.3	Added TimeStrobe timing

Released

TABLE OF CONTENTS PAGE

1 INTRODUCTION

The Single Chip Telemetry and Telecommand (SCTMTC) ASIC is an integrated device providing on-board telemetry and telecommand services via standardised interfaces.

This document will eventually be replaced by the SCTMTC ASIC data sheet produced by Atmel.

2 SCOPE

This User's Manual defines how the SCTMTC ASIC is to be used.

2.1 How to read this manual

The document has been divided in four main chapters, describing the relevant data formats, the functionality, the software aspects and finally the hardware aspects. Each of these chapters has been partitioned into sub-chapters for each major module. Therefore, to fully appreciate the functionality of a module, e.g. the telecommand decoder, the corresponding sub-chapter in all the four main chapters should be read by the user. Note that some sub-chapters have been omitted since not carrying any specific data for the corresponding module.

2.2 Turbo code patent

Implementers should be aware that a wide class of turbo codes is covered by a patent by France Télécom and Télédiffusion de France under US Patent 5,446,747 and its counterparts in other countries. Potential user agencies should direct their requests for licenses to:

Mr. Christian Hamon
CCETT GIE/CVP
4 rue du Clos Courtel
BP59
35512 CESSON SEVIGNE Cedex
France
Tel: +33 2 99 12 48 05
Fax: +33 2 99 12 40 98

3 DOCUMENTS

The latest issue of a document is valid, if not specified.

3.1 Applicable Documents

[TC_STD]	Packet Telecommand Standard ESA PSS-04-107, issue 2, April 1992
[TC_SPEC]	Telecommand Decoder Specification ESA PSS-04-151, issue 1, September 1993
[TM_STD]	Packet Telemetry Standard ESA PSS-04-106, issue 1, January 1988
[TMCOD_STD]	Telemetry Channel Coding Standard ESA PSS-04-103, issue 1, September 1989
[MOD_STD]	Radio Frequency and Modulation Standard, ESA PSS-04-106, issue 1, December 1989

3.2 Reference Documents

[JTAG]	Standard Test Access Port and Boundary-Scan Architecture and Supplement IEEE-STD-1149.1 and IEEE-STD-1149.1b
[CCSDS_TC1]	Telecommand – Part 1 – Channel Service CCSDS 201.0-B-3, June 2000
[CCSDS_TC2]	Telecommand – Part 2 – Data Routing Service CCSDS 202.0-B-3, June 2001
[CCSDS_TC2.1]	Telecommand – Part 2.1 – Command Operation Procedures CCSDS 202.1-B-2, June 2001
[CCSDS_TC3]	Telecommand – Part 3 – Data Management Service CCSDS 203.0-B-2, June 2001
[CCSDS_TM]	Packet Telemetry CCSDS 102.0-B-5, November 2000
[CCSDS_TMCOD]	Telemetry Channel Coding CCSDS 101.0-B-6, October 2002
[CCSDS_TMSYNC]	TM Synchronization and Channel Coding, CCSDS 131.0-R-1, Red Book, July 2002
[CCSDS_TMLINK]	TM Space Data Link Protocol, CCSDS 132.0-R-1, Red Book, December 2001
[CCSDS_SOURCE]	Space Packet Protocol, CCSDS 133-0-R-1, Red Book, December 2001
[CCSDS_TCSYNC]	TC Synchronization and Channel Coding, CCSDS 231.0-R-1, Red Book, July 2002
[CCSDS_TCLINK]	TC Space Data Link Protocol, CCSDS 232.0-R-1, Red Book, December 2001
[ECSS_SPACE]	SpaceWire - Links, Nodes, Routers and Networks, ECSS-E-50-12, Draft 4

4 DEFINITIONS

This section and the following subsections define the typographic and naming conventions used throughout this document.

4.1 Requirement Numbering

Requirement numbering is inapplicable for this document.

4.2 Bit Numbering

The following conventions are used for bit numbering:

- The Most Significant Bit (MSB) of a vector has the leftmost position.
- The Least Significant Bit (LSB) of a vector has the rightmost position.
- Unless otherwise indicated, the MSB of a vector has the highest bit number and the LSB the lowest bit number.

4.3 Names

The following conventions are used for all names (for signals and registers some extra conventions are defined below):

- A name may never start with a digit; e.g. 1553 could instead be M1553.
- A dollar sign ($) in a name is used as a wildcard representing a number. (If the dollar sign ($) is used in a context it must then be defined somewhere else in the document)
- An asterisk (*) in a name is used as a wildcard representing one or more characters.

4.4 Radix

The following conventions is used for writing numbers:

- Binary numbers are indicated by the subscript "$_2$", e.g. 1_2, $1011\_1010\_1011\_1110_2$, 010010_2 etc.
- Decimal numbers are indicated by the subscript "$_{10}$", e.g. $67{,}8723_{10}$, 47860_{10}.
- Hexadecimal numbers are indicated by the subscript "$_{16}$", e.g. E_{16}, $BABE_{16}$.
- Unless the Radix is explicitly declared as above the number should be considered to be decimal number.

4.5 Signal Names

The following conventions are used for signal names:

- Signal names are written in Italics, e.g. *SignalName*.
- Active low signals have a capital N appended to their name, e.g. *SignalNameN*.
- Bus indices are indicated with brackets, e.g. *SignalName[12:3]*.
- Signals maybe grouped into subsignals, e.g. *SignalName.SubSignal*.
- Signals with two functions are named with the name and then the first functionality followed by the second function, e.g. *SignalNameFunction1Function2N*. The second function is the valid when the signal is deasserted (thus the suffix *N* in the name).

4.6 Externally Accessible Register Names

The following convention is used for externally accessible registers.

- Register names are underlined, e.g. RegisterName.
- Fields of a register are indicated by the name of the register and the field, separated by a period and underlined, e.g. RegisterName.Field.

4.7 Primitive Polynomials

In this document, PN sequence generators are defined by their primitive polynomial on the following format: $x^{k1} + x^{k2} + ... x^{kN}$ The terms are integers representing the XOR positions of a generator in which the bit-shift moves in the direction of increasing bit number.

Example:
The primitive polynomial $x^8 + x^2 + x^1 + 1$ defines the PN sequence generator seen in the figure below.

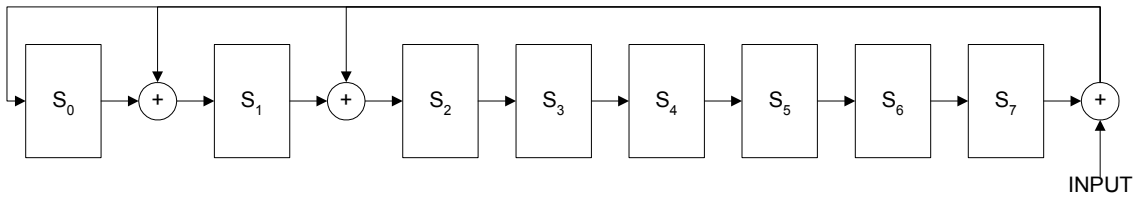


Figure 4-1 PN sequence generator defined by $x^8 + x^2 + x^1 + 1$

4.8 Terminology

4.8.1 General SCTMTC ASIC terminology

4.8.1.1 General

ASIC Module	An ASIC internal module
Issue an Interrupt	The corresponding bit has been set in the pending interrupt register in the ASIC/FPGA module
Reset Assertion	An internal reset activation as seen by ASIC/FPGA modules

4.8.1.2 Basic Data Types

Byte	8 bits of data
HalfWord	16 bits of data
Word	32 bits of data

4.8.1.3 Registers

Register Read	A read access to an externally accessible ASIC register
Register Write	A write access to an externally accessible ASIC register
Set	Indicates that the bit in the register is 1
Clear	Indicates that the field or bit in the register is 0
Reset	Indicates that the field or bit in the register is set to its default Value (indicated in the Register definition chapter)

4.8.1.4 Signals

Assert	To put a signal into its active state. A signal is *asserted* when in its active state
Deassert	To put a signal into its inactive state. A signal is *deasserted* when in its inactive state

4.8.1.5 Direct memory access

DMA Channel	A unit for accessing memory
DMA Error	Signal to DMA Channel for failing memory access
DMA Read	A transfer of data to a DMA channel from a DMA controller, e.g. read data from memory
DMA Write	A transfer of data from a DMA channel to a DMA controller, e.g. write data to memory.

4.8.1.6 CPDM Selector Module (CSEL) specific

Accept	A request is accepted if it is allowed to execute in the mode the CSEL is operating in.
Ground Request	The telecommand decoder (TC) tries to send a CPDU Telecommand Packet to the CPDM.

Ongoing Sequence	A sequence is ongoing when it has been handed over to the CPDM and until the CPDM frees the buffer.
PM Request	The Processor Module (PM) tries to send a CPDU Telecommand Packet to the CPDM.
Remote CPDM	The CPDM connected to the remote CSEL.
Remote CSEL	A redundant CPDM Selector in a system, with which the CSEL communicates.
Remote Status	The copy of the status of the remote CSEL, kept by the CSEL.
RM Request	The RM tries to send a CPDU Telecommand Packet to the CPDM.

4.8.1.7 Command Pulse Distribution Module (CPDM) specific

Clean	A packet is considered clean if it passes the clean check and, if it is of Lockout type, it does not contain any invalid pulses.
Dirty	If a packet is not clean, it is considered dirty.
Duration	Unit for pulse output timing defined by [TC_SPEC].
Free a buffer	Freeing a buffer allows the originator of the CPDU Telecommand Packet to reuse the memory area.
Free a buffer with error	Freeing a buffer with error informs the originator of the CPDU Telecommand Packet that an error occurred during the clean check, legal check or packet execution.
Illegal	A packet that is clean but not legal is considered illegal.
Legal	A packet is considered legal if it passes the legal check.
Lockout type	A packet of Lockout type does not have access to all pulses, if the packet contains any of the inaccessible pulses, it is considered dirty.
Ground type	A packet of Ground type has a status register separated from all other packets.
Ongoing	A pulse is considered ongoing as soon as the CPDM has asserted *CpdmArm*.
Abort	A packet can be aborted before all its pulses has been executed, this leads to any ongoing pulse being shortened to a predefined value if it's remaining execution time is longer than this time. All remaining pulses are discarded.

4.8.1.8 Packet Telemetry Encoder Module (TME) specific

Packet	A packet is a data structure as defined by [TM_STD] and [CCSDS_TM]. Also referred to as Telemetry Packet, Source Packet or similar. Indicated in the Transfer Frame by the Data Field Synchronisation flag equals zero.
Data Block	A data block is a general data structure. Indicated in the Transfer Frame by the Data Field Synchronisation flag equals one.

4.8.1.9 SpaceWire Module (SPW) specific

Active VRC/VTC $	Virtual receive or transmit channel, VRC or VTC$ is currently receiving or transmitting data.
Inactive VTC/VRC $	No data is currently being received or transmitted on virtual receive or transmit channel, VRC or VTC$.

4.9 Data structures

4.9.3 Packet Telecommand Decoder (PDEC3) specific

4.9.3.1 Command Link Transmission Unit Structure

The structure of the Command Link Transmission Unit (CLTU) processed by the PDEC3 is shown in Figure 4-2.

Command Link Transmission Unit											
Start Sequence	Telecommand Code Block Field										Tail Sequence
	TC Code Block 1			TC Code Block 2			etc.	TC Code Block N			
	Information	Error Control		Information	Error Control			Information	Error Control		
		7 Parity bits	1 Filler bit		7 Parity bits	1 Filler bit			7 Parity bits	1 Filler bit	
2 octets	7 octets	1 octet		7 octets	1 octet			7 octets	1 octet		8 octets
	8 octets			8 octets				8 octets			

Figure 4-2 Command Link Transmission Unit structure

A valid CLTU consists of:

- The Start Sequence with the nominal value $EB90_{16}$, and which marks the beginning of the CLTU.
- The Telecommand (TC) Code Block field with 2 to 147 TC Code Blocks, each consisting of seven Information octets and one Error Control octet. In the Error Control octet the Parity bits are a (63,56) modified Bose-Chaudhuri-Hocquenghem (BCH) code generated by the polynomial $G(x) = x^7 + x^6 + x^2 + x^0$, and the Filler bit shall nominally be zero.
- The Tail Sequence, which marks the end of the TC Code Block field. It is a TC Code Block with more than one bit-error, and with a Filler bit nominally being one.

4.9.3.2 Telecommand Transfer Frame Structure

The structure of the Telecommand Transfer Frame together with the Fill octets is shown in Figure 4-3. The Information octets (i.e., without the Error Control octets) from all TC Code Blocks in a CLTU are concatenated to form a Telecommand Transfer Frame. Any leftover Information octets after the Frame, as determined by the Frame Length field in the Frame Header, are called Fill octets.

Concatenated Information octets from the CLTU Code Blocks			
Telecommand Transfer Frame			Fill octets
Frame Header	Frame Data field containing a Telecommand Segment or FARM-1 Control Command	Frame Error Control field	
5 octets	1 to 1017 octets	2 octets	0 to 6 octets
N*7 octets ($2 \leq N \leq 147$)			

Figure 4-3 Telecommand Transfer Frame structure, also showing the Fill octets

The Telecommand Transfer Frame consists of:

- The Frame Header, further described in section 4.9.3.2.1.
- The Frame Data field, further described in section 4.9.3.2.2.
- The Frame Error Control field, which is a Cyclic Redundancy Code (CRC) used for detecting any remaining errors which may exist in the Transfer Frame. After initialising the encoder to all ones, it is generated over the entire Transfer Frame (except the Frame Error Control field) using the polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$.

4.9.3.2.1 Frame Header Structure

The structure of the Frame Header processed by the PDEC3 is shown in Figure 4-4.

Frame Header							
Version Number	Bypass flag	Control Command flag	Reserved field A	Spacecraft Id	Virtual Channel Id	Frame Length	Frame Sequence Number
2 bits 0..1	1 bit 2	1 bit 3	2 bits 4..5	10 bits 6..15	6 bits 0..5	10 bits 6..15	8 bits
2 octets					2 octets		1 octet

Figure 4-4 Frame Header structure

The Frame Header consists of:

- The static fields, being the Version Number field, the Reserved field A, the Spacecraft Identifier field and the Virtual Channel Identifier field, and which are simply compared to static values stored in the external non-volatile memory. The Version Number field and Reserved field A should be set to 00_2 to be compliant with [TC_STD].
- The Bypass flag and the Control Command flag, which are used to determine the Transfer Frame type. The possible combinations and their interpretation are shown in Table 4-1.
- The Frame Length field specifies the length of the TC Transfer Frame, with its value being (total number of octets in the Transfer Frame – 1).
- The Frame Sequence Number field, which is denoted N(S). It is used differently depending on the Transfer Frame type:
  - ♦ For AD Frames, this field is used by the FARM-1 for the sequence control.
  - ♦ For BC and BD Frames, the 8-bit field shall be zero.

Bypass flag	Control Command flag	Interpretation
0	0	AD Frame: sequence-controlled data
0	1	Illegal combination
1	0	BD Frame: expedited data
1	1	BC Frame: FARM-1 Control Command

Table 4-1 Interpretation of Bypass and Control Command flags

4.9.3.2.2 Frame Data Field Structures

The Frame Data field of an AD or a BD Frame contains a TC Segment, further described in section 4.9.3.3.

The Frame Data field of a BC Frame contains a Frame Acceptance and Reporting Mechanism (FARM-1) Control Command. Two such commands are supported:

- "Unlock", further described in section 4.9.3.3.1.
- "Set V(R) to V*(R)", further described in section 4.9.3.3.2.

4.9.3.3 Telecommand Segment Structure

The structure of the Telecommand Segment processed by the PDEC3 is shown in Figure 4-5.

<table>
<tr><td colspan="6">Telecommand Segment</td></tr>
<tr><td colspan="3">Segment Header</td><td rowspan="3">Segment Data field containing a CPDU packet, an Authentication Unit Control Command, or data to be provided to the spacecraft users</td><td colspan="2">Authentication Tail (optional)</td></tr>
<tr><td rowspan="2">Sequence flags</td><td colspan="2">MAP Identifier</td><td colspan="2">Logical Authentication Channel (LAC)</td><td>Signature</td></tr>
<tr><td>Control flag</td><td>MAP Address</td><td>LAC Id</td><td>LAC Count</td><td></td></tr>
<tr><td>2 bits
0..1</td><td>1 bit
2</td><td>5 bits
3..7</td><td>0 to 1016 octets, or
0 to 1007 octets</td><td>2 bits
0..1</td><td>30 bits
2..31</td><td>40 bits
32..71</td></tr>
<tr><td colspan="3">1 octet</td><td></td><td colspan="3">0 or 9 octets</td></tr>
<tr><td colspan="6">1 to 1017 octets</td></tr>
</table>

Figure 4-5 Telecommand Segment structure

The TC Segment consists of:

- The Segment Header, used to route the TC Segment to different users, as determined by the MAP Identifier.
- The Segment Data field, further described in section 4.9.3.3.3.
- The Authentication Tail, only included for TC Segments that shall be authenticated. It consists of a Logical Authentication Channel (LAC) with a LAC Identifier and a 30-bit LAC Count value, followed by a 40-bit Signature. The interpretation of the LAC Id field is shown in Table 4-2.

LAC Id (binary)	**Interpretation**
00	Principal LAC
01	Auxiliary LAC
10	Recovery LAC
11	Illegal combination

Table 4-2 Interpretation of LAC Id field

4.9.3.3.1 FARM-1 Control Command "Unlock" Structure

The structure of the FARM-1 Control Command "Unlock" executed by the PDEC3 is shown in Figure 4-6. This command is used to reset the sequence-controlled service.

"Unlock" command
Command Identifier = 0000_0000
1 octet

Figure 4-6 FARM-1 Control Command "Unlock" structure

4.9.3.3.2 FARM-1 Control Command "Set V(R) to V*(R)" Structure

The structure of the FARM-1 Control Command "Set V(R) to V*(R)" executed by the PDEC3 is shown in Figure 4-7. This command is used to preset the expected Frame Sequence Number V(R) to any desired value. The new value for V(R) is contained in the third octet of the command.

"Set V(R) to V*(R)" command	
Command Identifier = 1000_0010_0000_0000	V*(R): new value for V(R)
2 octets	1 octet

Figure 4-7 FARM-1 Control Command "Set V(R) to V*(R)" structure

Released

4.9.3.3.3 Segment Data Field Structures

The Segment Data Field contains either:

- A packet to be processed by the Command Pulse Distribution Unit (CPDU), further described in section 4.9.4.1.1.
- A Control Command for the Authentication Unit (AU), further described in section 4.9.3.3.4.
- Data to be provided to the spacecraft users (preceded by the Segment Header, but without Authentication Tail, if any).

4.9.3.3.4 Authentication Unit Control Command Structure

For an AU Control Command the Segment Header shall be 255, i.e. the MAP Identifier shall be 63 and the Sequence flags shall be 11_2 (unsegmented).

There are seven different AU Control Commands, defined in Table 4-3, grouped into three groups, of which the structures are shown in Figure 4-8, Figure 4-9 and Figure 4-10.

Group	Command Identifier	Command name
Group 1	0000_0000_2	Dummy segment
	0000_0101_2	Select fixed key
	0000_0110_2	Select programmable key
	0000_0111_2	Load fixed key into programmable key
Group 2	0000_1001_2	Set New LAC Count value
Group 3	0000_1010_2	Change programmable key block A
	0000_1011_2	Change programmable key block B

Table 4-3 AU Control Commands

Authentication Unit group 1 Control Command
Command Identifier
1 octet

Figure 4-8 Authentication Unit group 1 Control Command structure

Authentication Unit group 2 Control Command		
Command Identifier	LAC value to be set	
	LAC Id	LAC Count
8 bits 0..7	2 bits 0..1	30 bits 2..30
1 octet	4 octets	

Figure 4-9 Authentication Unit group 2 Control Command structure

Authentication Unit group 3 Control Command		
Command Identifier	Start Address	Key specific pattern
1 octet	1 octet	7 octets

Figure 4-10 Authentication Unit group 3 Control Command structure

Released

4.9.3.4 Command Link Control Word Structure

The structure of the Command Link Control Word (CLCW) generated by the PDEC3 is shown in Figure 4-11.

Command Link Control Word							
Control Word Type	CLCW Version Number	Status Field	COP In Effect	Virtual Channel Identifier		Reserved Field	
1 bit 0	2 bits 1..2	3 bits 3..5	2 bits 6..7	6 bits 8..13		2 bits 14..15	
2 octets							
No RF Availabl e	No Bit Lock	Lock Out	Wait	Retransm it	FARM B Count	Report Type	Report Value
1 bit 16	1 bit 17	1 bit 18	1 bit 19	1 bit 20	2 bits 21..22	1 bit 23	8 bits 24..31
2 octets							

Figure 4-11 Command Link Control Word structure

The CLCW consists of:

- The Control Word Type flag, which shall be zero to identify the word as containing CLCW.
- The CLCW Version Number field, which identifies the CLCW structure. This field shall be set to 00_2.
- The Status Field, reserved for future applications, which shall be set to 000_2.
- The COP In Effect field, which shall be set to 01_2 to indicate that COP-1 is in use.
- The Virtual Channel Identifier, which identifies the TC channel of the PDEC3.
- The Reserved field, which shall be set to 00_2.
- The No RF Available flag, which shall be one if none of the connected RF subsystems have detected a valid RF signal, and otherwise shall be zero.
- The No Bit Lock flag, which shall be one if none of the connected demodulation subsystems have achieved bit lock, and otherwise it shall be zero.
- Frame Acceptance and Reporting Mechanism (FARM-1) variables, being the Lockout flag, the Wait flag, the Retransmit flag, the FARM-B Count and the Report value.
- The Report Type bit, which shall be zero.

4.9.4 Command Pulse Distribution Module (CPDM) specific

4.9.4.1.1 Command Pulse Distribution Unit Packet Structure

The structure of the CPDU packets is shown in Figure 4-12. CPDU packets are not processed by PDEC3 but transferred to the Command Pulse Distribution Module.

CPDU Telecommand Packet													
Packet Header							Packet Data Field						
Packet Identification				Packet Seq-uence Control		Packet Length	N Command Instructions (s)						Packet Error Control
Version Number	Type	Data Field Header flag	Appli-cation Process Id	Sequence flags	Sequence Count		Command Instruction 1				..	Command Instruction N	
							Output No. LSB	Output No. MSB	Parity	Pulse Length			
3 bits 0…2	1 bit 3	1 bit 4	11 bits 5..15	2 bits 0..1	14 bits 2..15	16 bits 0..15	8 bits	4 bits	1 bit	3 bits		16 bits	16 bits 0..15
2 octets				2 octets		2 octets	2*N octets						2 octets

Figure 4-12 Command Pulse Distribution Unit packet structure

The CPDU packet consists of:

- The Packet Identification field, being the Version Number field, the Type field, the Data Field Header flag, and the Application Process Identifier field, which are simply compared to static values stored in the external non-volatile memory.
- The Sequence flags, which for a CPDU packet shall be 11_2 (standalone packet).
- The Sequence Count, which identifies the CPDU packet in a sequence of CPDU packets.
- The Packet Length field specifies the length of the CPDU packet, with its value being (total number of octets in the CPDU packet – 7).
- Between 1 and 504 CPDU command instruction when the TC Segment for the CPDU is not authenticated, and between 1 and 499 commands when it is authenticated. Each CPDU command instruction consists of:
  - Output Number (12 bits), specifying one of 4096 possible CPDU outputs.
  - Pulse Length (3 bits), which determines the length of the CPDU pulse. The pulse length is defined as $D \cdot 2^L$, where D is the CPDM duration and L is the 3-bit pulse length.
  - One optional parity bit. If command instruction parity is enabled in the CPDM, the total number of ones in the command instruction shall equal an odd number (odd parity).

The Packet Error Control field, which is a CRC used for detecting any remaining errors which may exist in the CPDU packet. After initialising the encoder to all ones, it is generated over the entire CPDU Packet (except the Packet Error Control field) using the polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$

Released

4.9.7 Packet Telemetry Encoder (TME) specific

4.9.7.1 Telemetry Transfer Frame

The Transfer Frame generated by the Telemetry Encoder is shown in Figure 4-13. It consists of the Primary header, the data field and the optional Secondary header and Transfer Frame trailer.

Telemetry Transfer Frame			
Transfer Frame Header		Transfer Frame Data Field	Transfer Frame Trailer
Primary Header	Secondary Header	ket | Packet | Packet | Pac	
6 octets	0/4 octets	variable	0/2/4/6 octets
223/446/892/1115/239/478/956/1195 octets			

Figure 4-13 Channel Access Data Unit structure

4.9.7.1.1 Telemetry Primary Header Structure

The structure of the primary header generated by the Telemetry Encoder is shown in Figure 4-14.

Transfer Frame Primary Header										
Frame Identification				Master Channel Frame Count	Virtual Channel Frame Count	Frame Data Field Status				
Vers-ion	S/C Id	VC Id	OPCF Flag			Sec. Header Flag	Sync Flag	Packet Order Flag	Segm. Length Id	First Header Pointer
2 bits 0..1	10 bits 2..11	3 bits 12..14	1 bit 15	8 bits	8 bits	1 bit 0	1 bit 1	1 bit 2	2 bits 3..4	11 bits 5..15
2 octets				1 octet	1 octet	2 octets				

Figure 4-14 Transfer Frame Primary Header structure

4.9.7.1.2 Telemetry Secondary Header Structure

The structure of the secondary header that optionally can be generated by the Telemetry Encoder is shown in figure below.

Transfer Frame Secondary Header		
Secondary Header Identifier		Secondary Header Data
Sec. Header Version No.	Sec. Header Length	
2 bits 0..1	6 bits 2..7	24 bits 0..23
4 octets		

Figure 4-15 Transfer Frame Secondary Header structure

4.9.7.1.3 Telemetry Transfer Frame Trailer Structure

The structure of the transfer frame trailer generated by the Telemetry Encoder is shown in figure below. The Operational Control Field (OPCF) and the Frame Error Control Word (FECW) can be independently enabled and disabled.

Transfer Frame trailer	
Operational Control Field	Frame Error Control Word
CLCW	CRC
32 bits 0..31	16 bits 0..15
0/4 octets	0/2 octets

Figure 4-16 Transfer Frame Trailer structure

4.9.7.2 Source/ Telemetry Packet Structure

The Source Packet defined in [TM_STD] and [CCSDS_TMCOD] recommendation is used in this document and is listed in Figure 4-17, although the Segmentation Flags are interpreted differently in the two referenced documents. The Telemetry Packet defined in ESA PSS [TM_STD] standard has also this same format, although the Packet Length and Data Field are interpreted differently. The differences have no effect on the TME.

Source Packet / Telemetry Packet									
Packet Header							Packet Data Field		
Packet Identification				Packet Sequence Control		Packet Length	Data Field Header (Optional)	Source Data	Packet Error Control (Optional)
Version Number	Type	Data Field Header Flag	Appli-cation Process Id	Segmen-tation Flags	Source Sequence Count				
0..2	3	4	5..15	16..17	18..31	32..47			
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits	16 bits	var.	var.	var.

Figure 4-17 Source/ Telemetry Packet structure

4.9.7.3 Reed-Solomon Codeblock

If Reed-Solomon encoding is enabled the codeblock with Attached Synchronisation Marker generated by the Telemetry Encoder is shown in Figure 4-18.

Attached Synchronisation Marker	Reed-Solomon Codeblock	
	Telemetry Transfer Frame	Reed-Solomon check symbols
4 octets	223/446/892/1115/ 239/478/956/1195 octets	32/64/128/160/ 16/32/64/80 octets

Figure 4-18 Reed-Solomon Codeblock with Attached Synchronisation Marker

Released

4.9.7.4 Turbo Codeblock

If turbo encoding is enabled the codeblock with Attached Synchronisation Marker generated by the Telemetry Encoder is shown in Figure 4-19.

Attached Synchronisation Marker	Turbo Codeblock	
32/r bits	k/r bits	4/r bits

Figure 4-19 Turbo Codeblock with Attached Synchronisation Marker

Note: All the block lengths are specified in bits rather than octets.

Note: The turbo encoder standard does **not** support the following frame lengths: 239/478/956/1195 octets.

4.9.8 SpaceWire (SPW) specific

4.9.8.1 SpaceWire Packet Structure

A SpaceWire packet consists of a destination address, a cargo and an end or packet (EOP or EEP) marker.

<destination address><cargo><end of packet>

where:

- The destination address consists of a list of zero or more bytes, called destination identifiers: <destination address> = <id 0><id 1> ... <id N-1>
- The cargo contains zero or more bytes

The end of packet is either an EOP, indicating a normal termination of a packet, or an EEP, indicating a packet in which an error has occurred.

4.9.9 Control Interface (CI) specific

All accesses to the Control Interface are word based. Before the actual access the Control Interface word-aligns the address and shortens the packet size to the nearest lower word if it is not word based.

Accesses to the Control Interface are big-endian, i.e. the most significant bit of the most significant byte is sent first in every packet. Note that *N* in the following figures should be divisible by 4.

4.9.9.1 Packet Wire write packets

To perform a write operation on the Packet Wire interface the following sequence of bytes (a packet) shall be entered (the leftmost byte entered first):

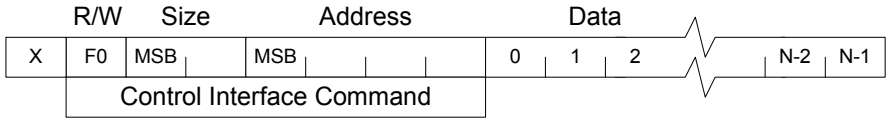


Figure 4-20 Packet Wire write packet

The packet content is the same as for the parallel interface but with an additional dummy byte at the beginning to fill up to two complete 32-bit words.

4.9.9.2 Packet Wire read packets

To perform a read operation on the Packet Wire interface the packet sent to the Control Interface is shorter, containing only the dummy byte, the Read/Write indicator, the size and the address:

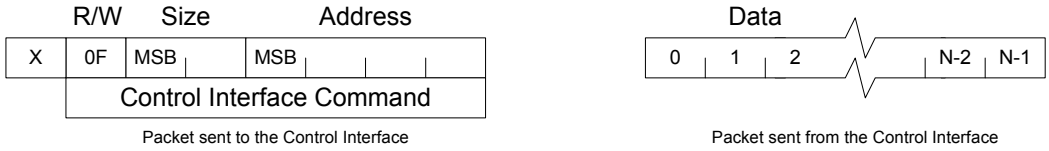


Figure 4-21 Packet Wire read packet and result

Note: A read access to an unmapped address will result in an internal bus error which in turn will not return any data over the Control Interface. Refer to sections 6.8 and 6.9 for further details.

4.9.9.3 SpaceWire write packets

To perform a write operation on the SpaceWire interface the following sequence of bytes (a packet) shall be entered (the leftmost byte entered first):

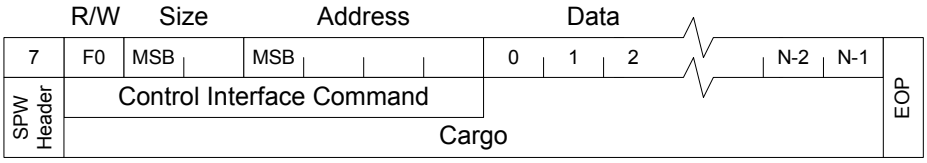


Figure 4-22 SpaceWire write packet

The first byte is used as routing information by the SpaceWire module.

4.9.9.4 SpaceWire read packets

To perform a read operation on the SpaceWire interface the packet sent to the Control Interface is shorter, containing only the Read/Write indicator, the size and the address:

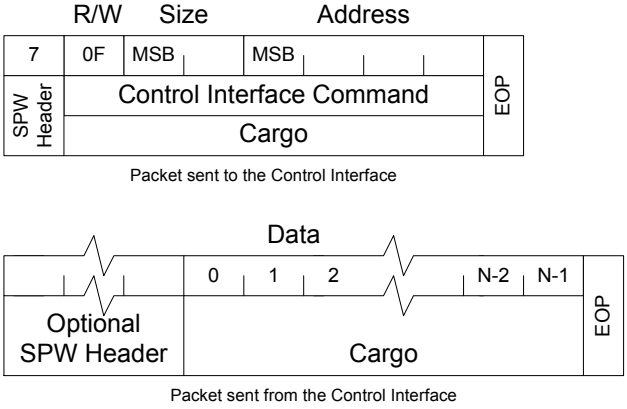


Figure 4-23 SpaceWire read packet and result

The first byte is used as routing information by the SpaceWire module.

Note: A read access to an unmapped address will result in an internal bus error which in turn will not return any data over the Control Interface. Refer to sections 6.8 and 6.9 for further details.

4.10 Abbreviations

ASIC	Application Specific Integrated Circuit
ASM	Attached Synchronisation Marker
AU	Authentication Unit
AUSR	Authentication Unit Status Report
BAT	Bandwidth Allocation Table
BPSK	Bi-Phase Shift Keying
CADU	Channel Access Data Unit
CAR	Clock and Reset
CCSDS	Consultative Committee for Space Data Systems
CI	Control Interface
CLCW	Command Link Control Word
CLTU	Command Link Transmission Unit
COP	Command Operation Procedure
CPDM	Command Pulse Distribution Module
CPDU	Command Pulse Distribution Unit
CRC	Cyclic Redundancy Code
CSEL	Command Pulse Distribution Selector
DMA	Direct Memory Access
DTC	Direct Telecommand
ECSS	European Cooperation for Space Standardisation
EDAC	Error Correction And Detection
EEP	Error End of Packet
EEPROM	Electrically Erasable Programmable Read Only Memory
EMC	Electro Magnetic Compliance
EOM	End Of Message
EOP	End Of Packet
ESA	European Space Agency
FAR	Frame Analysis Report
FARM	Frame Acceptance and Reporting Mechanism
FECW	Frame Error Control Word
FHP	First Header Pointer
FIFO	First In First Out memory
GND	Ground
Id	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IO	Input/Output
LAC	Logical Authentication Channel
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LSW	Least Significant Word
LUT	Lookup Table
MAP	Multiplexed Access Point
MSB	Most Significant Bit
MSW	Most Significant Word
MTBF	Mean Time Between Failure
NA	Not Applicable

NC	No Connect
NRZ	Non Return to Zero
NRZ-L	Non Return to Zero Level
NRZ-M	Non Return to Zero Mark
NW	Negative Window
OCD	Output Command Driver
OPCF	Operational Control Field
PCM	Pulse Code Modulation
PCM	Pulse Code Modulation
PDEC	Packet Telecommand Decoder
PIM	Internal Bus Master
PM	Processor Module
PPS	Pulse Per Second
PROM	Read Only Memory
PSK	Phase Shift Keying
PW	Positive Window
PWT	Packet Wire Transmitter
QFP	Quad Flat Pack
RM	Reconfiguration Module
R-S	Reed-Solomon
RTC	Real Time Clock
SCID	Spacecraft Identifier
SCTMTC	Single Chip Telemetry and Telecommand
SES	Saab Ericsson Space AB
SEU	Single Event Upset
SP-L	Split Level
SPW	SpaceWire
SRAM	Static Random Access Memory
SW	Software
TAP	Test Access Point
TB	Test Bench
TBC	To Be Confirmed
TBD	To Be Determined
TC	Telecommand
TM	Telemetry
TME	Packet Telemetry Encoder
TME	Telemetry Encoder
VC	Virtual Channel
VCA	Virtual Channel Assembler
VCM	Virtual Channel Multiplexer
VRC	Virtual Receive Channel
VTC	Virtual Transmit Channel

5 FUNCTIONAL OVERVIEW

The Single Chip Telemetry and Telecommand (SCTMTC) ASIC is mainly an integrated telecommand decoder and telemetry encoder device. It can operate in stand-alone mode, not requiring any programming or support from an on-board computer. It can also operate in tight integration with a computer, featuring fast communication links for both commanding and data transfer. The different capabilities of the SCTMTC ASIC can work together, or separately when only part of the functionality is required in a system, e.g. a telecommand decoder or telemetry encoder only.

5.1 System Overview

The Single Chip Telemetry and Telecommand (SCTMTC) ASIC has been designed with a generic system architecture in mind. This architecture is assumed to consist of certain elements, although this is not a requirement for successful usage of the device. The main elements in this architecture are:

- Processor Module (PM)
  Comprising a processor or similar unit that is in control of the SCTMTC ASIC. A PM communicates with the SCTMTC ASIC by means of serial control interfaces. Dedicated serial interfaces are used for the reception of telecommands and transmission of telemetry from the PM. A PM is, however, not needed to control SCTMTC, since SCTMTC is autonomous.
- Reconfiguration Module (RM)
  Comprising an autonomous unit, possibly a processor, which is allowed to send sequences of command instructions to the CPDU described hereafter.
- Telecommand Decoder (TC)
  Comprising the Packet Telecommand Decoder (PDEC3) which is integrated in the SCTMTC ASIC.
- Command Pulse Distribution Unit (CPDU)
  Comprising the Command Pulse Distribution (CPDM) and Selector (CSEL) modules integrated in the SCTMTC ASIC, as well as a set of external high and low power drivers implemented by the user.
- Telemetry Encoder (TM)
  Comprising the Packet Telemetry Encoder (TME) which is integrated in the SCTMTC ASIC.

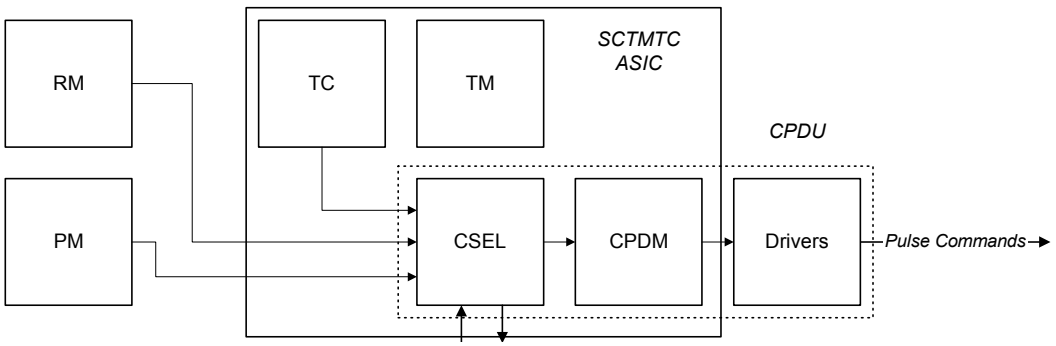


Figure 5-1 System block diagram

In this document, the PM acronym will be used to indicate any unit that can access the Internal Bus of the SCTMTC ASIC. This might be done via the SpaceWire interface or the Control Interface Module.

An RM is not part of the SCTMTC ASIC, but the acronym is used to indicate an external unit that can be connected to the CPDU via the External CPDU Interfaces. The CPDU handles such units in a special manner that will be explained when describing the CSEL module.

The acronym TC generally refers to the on-chip PDEC3 telecommand decoder. However, the CPDU has a specific TC request interface that is handled in a special manner that is explained in the description of the CSEL module. This interface can also be connected to the External CPDU Interface if required.

5.1.1 Examples of systems using the SCTMTC ASIC

5.1.1.1 Cross coupling

The SCTMTC ASIC is equipped with ready to use interfaces allowing cross coupling on several functional levels. The telecommand decoder features multiple inputs to receive telecommand bit streams from nominal and redundant receivers. The telemetry encoder has only a single output, but by external buffering it can be connected to multiple transmitters. The Command Link Control Word (CLCW) is output from each telecommand decoder on a redundant interface. Each telemetry encoder can fetch the CLCW from multiple telecommand decoders, making the CLCW communication fully redundant. An example of such a configuration can be seen in Figure 5-2.

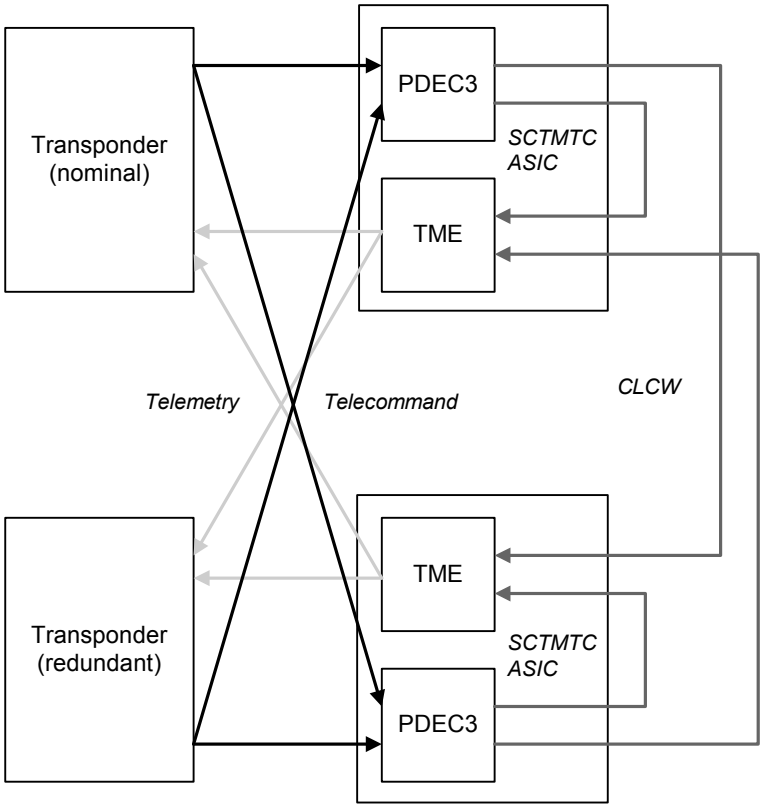


Figure 5-2 Cross-coupled TM/TC system

5.1.1.2 Stand alone TC system

The SCTMTC ASIC can be used for implementing only a telecommand decoder, not using the built in telemetry encoder, as can be seen in Figure 5-3. The telemetry encoder can be disabled by means of configuration registers.

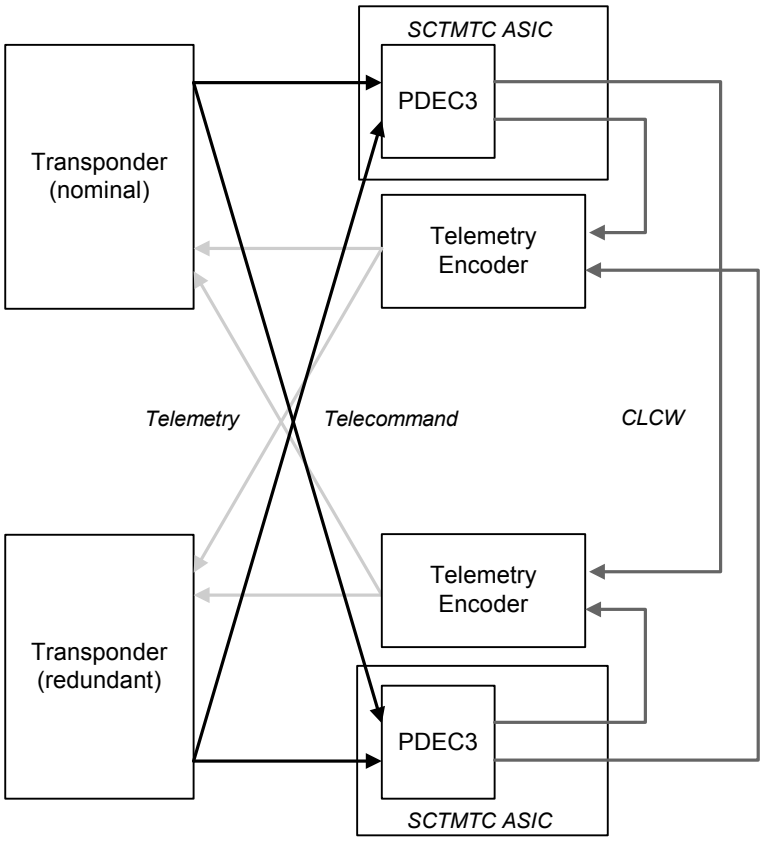


Figure 5-3 Stand-alone TC system

5.1.1.3 Stand alone TM system

The SCTMTC ASIC can be used for implementing only a telemetry encoder, not using the built in telecommand decoder, as can be seen in Figure 5-4. Since it is not possible to disable the telecommand decoder functionality by means of configuration registers in the SCTMTC ASIC, the function should be disabled by tying all pins of the input channels on decoder to defined deasserted values, preventing any commands from being interpreted. This will also ensure that the power dissipation is minimised.

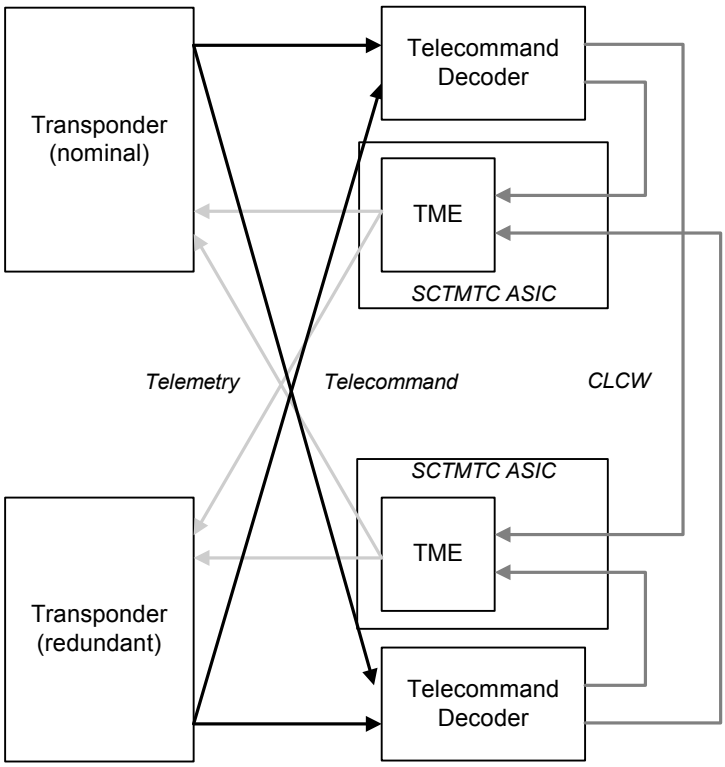


Figure 5-4 Stand-alone TM system

5.1.1.4 External Authentication or Decipher function

There is currently a need for external authentication units as well as external decipher functions in the telecommand decoder functionality. The SCTMTC ASIC supports such units and functions as will be described hereafter. The internal authentication unit is disabled and all telecommand segments are routed to a single MAP interface to the external authentication unit. When a telecommand segment has been authorised, the external authentication unit routes the telecommand segment to its final destination. The internal Command Pulse Distribution Module (CPDM) is accessible via the External CPDU Interface using a Packet Wire protocol.

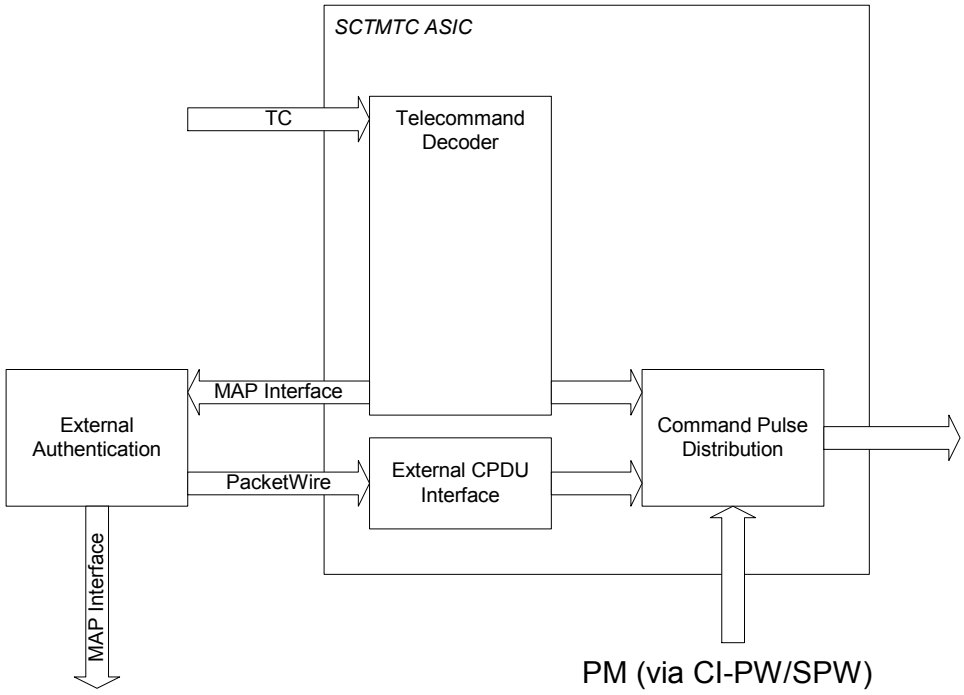


Figure 5-5 External AU accessed via MAP

Since the telecommand decoder has the ability to route all segments to the same destination, it is easy to encrypt the entire telecommand segment, including the segment header, and route all segments to the external authentication unit, where the segment is decrypted and authenticated. If the segment header is not encrypted, the internal routing mechanism can be utilised to route some segments to the external authentication and let some segments be routed to another destination without authentication (or with a different authentication unit).

It is also possible to use the internal authentication unit and external decryption. After encryption an authentication tail is added to the segment. The segment is authorised by the internal authentication unit and routed to the appropriate MAP interface where decryption is performed. After decryption the segment may be sent to the internal CPDM.

Released

5.1.2 Compliance and compatibility with standards

The SCTMTC ASIC complies with and implements the following packet telecommand standards: [TC_STD], [CCSDS_TC1], [CCSDS_TC2.0], [CCSDS_TC2.1] and [CCSDS_TC3], with the exceptions listed hereafter. The Segmentation Layer does not implement segment re-assembly for packets that span more than one segment. Instead, the segments are forwarded unaltered to the user for further processing. The transmission of segment data and packets is independent of their format. The packet format is only considered for CPDU operation. The decoder supports telecommand transfer frame lengths of up to 1024 octets.

The SCTMTC ASIC complies with and implements the packet telecommand decoder specification [TC_SPEC], with the exception listed hereafter. The transfer of the CLCW between the telecommand decoder and the telemetry encoder is based on 32 bits rather than on 16 bits, and the signal polarities have been modified. Optional priority mode has been added. Optional re-routing or re-addressing of segments has been added. Optional parity has been included for the Command Instructions in CPDU Telecommand Packets. Also, optional extension of the number of addressable pulse outputs has been added supporting up to 4096 outputs. Finally, optional protection or lockout of selectable pulse outputs has been added. Optional cross coupling support has been added to prevent parallel CPDU execution.

The SCTMTC ASIC complies with and implements the following telemetry coding standards: [TMCOD_STD] and [CCSDS_TMCOD], with the exceptions listed hereafter. Reed-Solomon interleave depths 3 and 8 are not implemented.

The SCTMTC ASIC complies with and implements the following packet telemetry standards: [TM_STD] and [CCSDS_TM], with the exceptions listed hereafter. Variable Telemetry Transfer Frame lengths are not supported. Explicitly, the lengths 669/717 and 1784/1912 corresponding to the Reed-Solomon interleave depths 3 and 8, respectively, are not implemented. Packetisation in not performed by the encoder, instead the user is expected to provide complete or segmented packets that will be transferred unaltered by the encoder. The transmission of packets is independent of the packet format. The packet format is only considered for idle packet insertion by the encoder, supporting both Source Packets and Telemetry Packets. Due to implementation restrictions, only packets of a minimum size of 7 octets are supported.

The SCTMTC ASIC is compatible with the incoming new CCSDS recommendations [CCSDS_TMSYNC], [CCSDS_TMLINK], [CCSDS_SOURCE], [CCSDS_TCSYNC] and [CCSDS_TCLINK], which are all based on existing recommendations.

The SCTMTC ASIC complies with the ECSS SpaceWire standard [ECSS_SPACE], with the exceptions listed hereafter. The 6-bit time-counters are implemented, but not required in the application. The reserved logical address for routers is neither implemented nor used in the application. The error handling will lead to nine words being deleted in the receiver buffer to make space for successful start-up. In case of link errors, such as disconnection, during a data transfer the received packet is aborted and all data to that destination are consumed by the SpaceWire module until an Exceptional End of Packet (EEP) or an End Of Message (EOM) is received.

5.2 Functions

The Single Chip Telemetry and Telecommand (SCTMTC) ASIC has the following functions:

- Test Access Port
  The ASIC includes an IEEE standard 1149.1 [JTAG] TAP block, which can be used for manufacturing test of printed circuit boards on which the ASIC is mounted.

- Clock and Reset block
  This block ensures that there is a clock of the correct frequency for each block and module in the ASIC. The block also generates the reset signal to each block and module in the ASIC. The block includes a reset register, which can be used to reset some of the modules in the ASIC.

- Configuration block
  The Configuration block will after power-up configure the ASIC according to the mission PROM, i.e. setup SpaceWire source clock, TME frame length etc. The Configuration block can also make a continuos refresh of all or part of the configuration registers inside SCTMTC.

- Memory Interface
  The memory interface supports SRAM and PROM and provides EDAC protection of the memory. The EDAC corrects single bit errors and detects double bit errors. The memory interface also includes a scrubber.

- Packet Telecommand Decoder [PDEC3]
  The telecommand decoder is fully compliant with ESA and CCSDS standards, specifications and recommendations. In addition, a Command Pulse Distribution Unit (CPDU) is composed of the CPDM, CSEL and ExtCpduIf described hereafter.

- External Command Pulse Distribution Unit Interface [ExtCpduIf]
  The ExtCpduIf provides a way for an external unit to generate command pulses using the internal CPDM. The input interface is based on the Packet Wire protocol including ready and abort signals.

- Command Pulse Distribution Selector [CSEL]
  The CSEL arbitrates access to the CPDM between three input request sources: the Reconfiguration Module (RM), the Telecommand Decoder (TC), and the Processor Module (PM). CSEL arbitrates between the input request sources, while CPDM executes the command sequence from the source that has been selected. The CSEL module includes a status interface, which ensures that the nominal and redundant CPDUs in a system are not interrupting each other.

- Command Pulse Distribution Module [CPDM]
  The CPDM receives telecommand segments on which it performs clean and legal checks. Each command instruction contained in the telecommand segment will result in an output pulse being generated for a specified time duration.

- Packet Telemetry Encoder [TME]
  The telemetry encoder and telemetry channel encoder are fully compliant with ESA and CCSDS standards and recommendations. The telemetry encoder physically implements eight Virtual Channels, which are named A to H.

- SpaceWire [SPW]
  The SpaceWire interface can be configured for transfer of data to one or several telemetry encoder Virtual Channels. The routing is done using the header of the SpaceWire packet. The SpaceWire interface can also be used for configuring the SCTMTC ASIC.

- Control Interface Module [CI]
  The Control Interface Module provides a connection between external serial interfaces to the internal bus of the SCTMTC ASIC to allow an external PM to control and configure the device. The CI is connected to the SpaceWire interface described above and to a serial Packet Wire interface dedicated for control.

5.3 Interfaces

The SCTMTC ASIC has the following interfaces:

- IEEE 1149.1 TAP interface

- Clock and Reset
  Separate clock inputs for system, telemetry and SpaceWire bit rates.

- Memory Interface
  a) Support for 8 or 16 bit wide data bus, with optional EDAC protection.
  b) Support for SRAM and PROM.

- Packet Telecommand Decoder
  a) Six separate serial telecommand input streams.
  b) Four RF available status inputs.
  c) Two external interfaces allowing readout of the CLCW. The internal Packet Telemetry Encoder is connected via one of these external interfaces if used.
  d) Five dedicated, i.e. internally decoded, and one general serial MAP interface providing the accepted Telecommand Segments. Each dedicated MAP interface has separate Terminal and Sender Ready pins. All MAP interfaces share the clock, data and abort signals.
  e) AU enable/disable selection.
  f) Enable/disable for priority selection of telecommand input streams.
  g) Configuration pins for defining the configuration of the TC input channels.

- External CPDU Interface
  Packet Wire based interface for receiving telecommand segments bound for the internal CPDU function.

- Command Pulse Distribution Unit
  a) Command pulse output interface
  b) Clock alive detection interface
  c) Remote CPDU status interface, to communicate with a redundant CPDU
  d) External setting of operating mode

- Packet Telemetry Encoder
  a) Eight Packet Wire interfaces for telemetry input.
  b) Four external interfaces for retrieval of the CLCW from Packet Telecommand Decoders. The interfaces have separate data input but share control signals.
  c) Telemetry Transfer Frame output interfaces: unencoded, encoded and modulated. All interfaces are serial bit stream with an associated clock. Except for the unencoded output, there are five different formats: PSK Squarewave, NRZ-L, NRZ-M, SP-L and I/Q. The five share the same interface. In addition, the unencoded output also has the *TmeUnEncSync* signal that is asserted while each and every Attached Synchronisation Marker is output.
  d) A strobe to allow the on-board time to be sampled synchronously with the generated Transfer Frames, named *TmeTimeStrb*.

- SpaceWire interface (nominal and redundant)
  For data transfers to the Telemetry Encoder (TME) and for communication with the Control Interface Module (CI) for control and configuration of the SCTMTC ASIC.

- Packet Wire interface (receive and transmit)
  For communication with the Control Interface Module (CI) for control and configuration of the SCTMTC ASIC.

5.4 Block diagrams

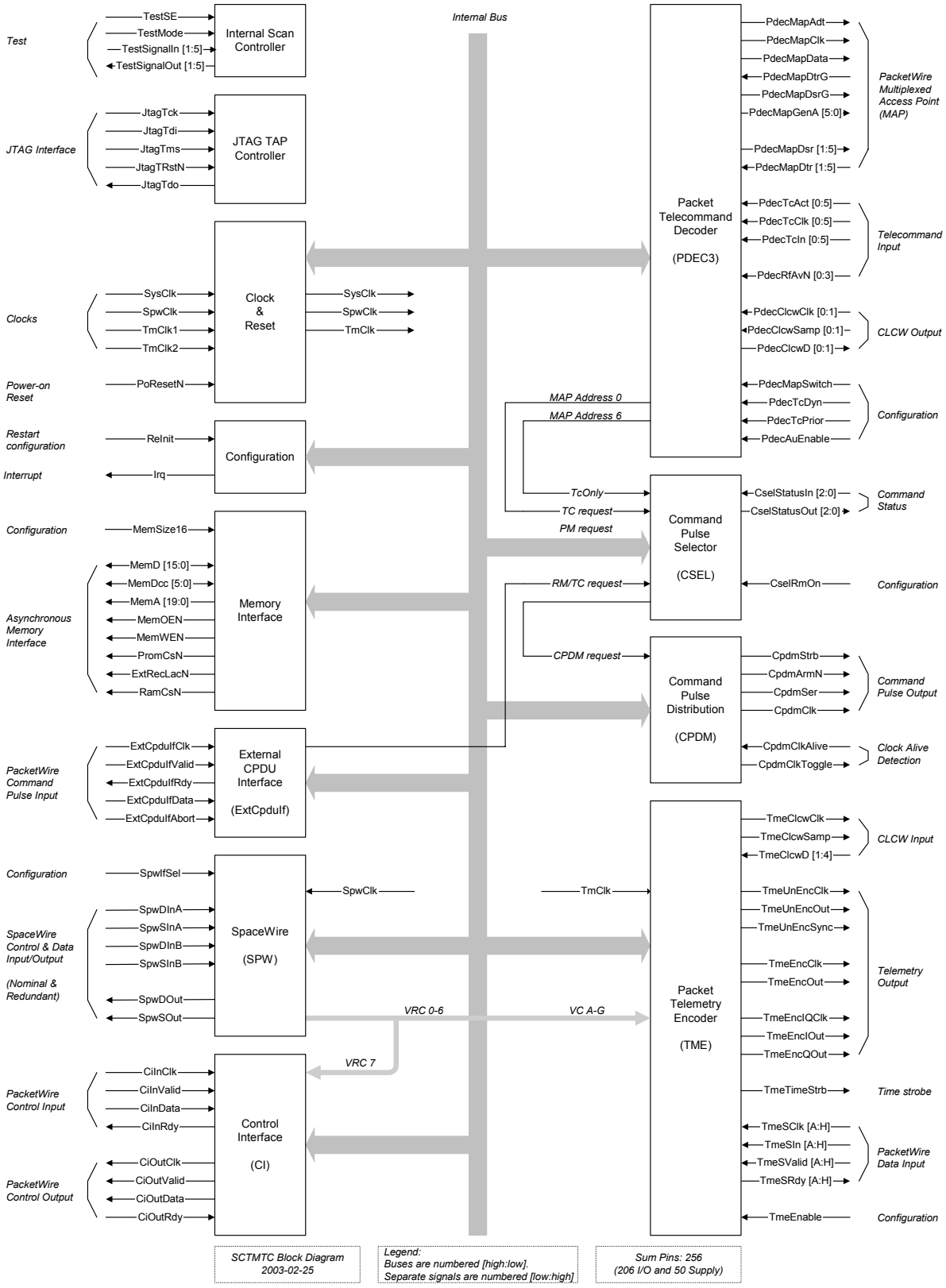


Figure 5-6 SCTMTC ASIC Block Diagram

5.4.1 General SCTMTC ASIC Functions

5.4.1.1 Internal Scan Controller block

The SCTMTC ASIC implements internal scan chains for testability. These are only to be used during manufacturing test.

5.4.1.2 Test Access Port (TAP) block

The design incorporates support for testing using a Test Access Port (TAP) defined according to the IEEE 1149.1 standard. The TAP allows access to the chip pins using a boundary-scan register. Benefits resulting from incorporating a Test Access Port are improved system level verification, fault injection and testing support. The TAP does not include the possibility to force the inputs on the die. It only includes the functionality to force and sample pins.

5.4.1.3 Clock and Reset (CAR) block

The intention of the CAR block inside the SCTMTC ASIC is to give the Configuration block the possibility to define a configuration of the ASIC during power up. It should neither be used nor accessed during normal operation.

The Clock and Reset block makes it possible to configure the internal clock for different modules. It also distributes the power-on reset signal, *PoResetN*, inside the chip. The block includes an enable/disable functionality, which makes it possible to enable/disable certain modules after power up. Note that all modules and blocks inside the SCTMTC ASIC cannot be reset or kept inactivated this way, e.g. it is not possible to deactivate the telecommand decoder by means of a register access.

All outputs of the SCTMTC are be driven inactive by the assertion of the *PoResetN* and all tri-state outputs will be in high impedance mode, i.e. tri-stated. The outputs will be kept in this state until the *PoResetN* is deasserted and until the module is enabled (for modules that are possible to disable). There are only a few modules that can be fully configured, see the table below.

Module	Possible clock source	Possible to disable	Possible to reset (excluding PoResetN)
Clock and Reset	*SysClk*	No	No
Memory Interface	*SysClk*	No	No
Configuration	*SysClk*	No	No
PDEC3	*SysClk*	No	No
CPDM and CSEL	*SysClk*	No	No
TME	*SysClk, TmClk1, TmClk2*	Yes	Yes
SpaceWire	*SysClk and SpwClk*	Yes	Yes
CI	*SysClk*	No	No
ExtCpduIf	*SysClk*	No	No

Table 5-1 Module clocks and disable

The clock configuration and module enabling is protected by an Arm-Enable functionality to ensure that no unwanted changes of the clock configuration is made. For details regarding the control see section 6.

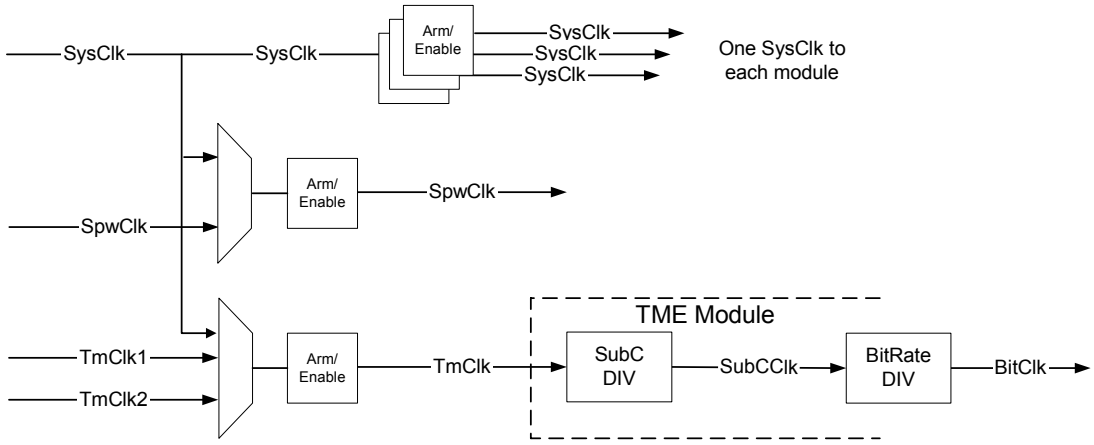


Figure 5-7 Clock distribution scheme

5.4.1.4 Configuration block

The Configuration block configures the SCTMTC ASIC after power-up according to the mission PROM, i.e. initialises Virtual Channel Identifiers, telemetry bit rates, modulation of SpaceWire source clock etc. The Configuration block can also perform a continuous refresh of all or part of the configuration registers inside the SCTMTC ASIC.

There are two levels of refresh implemented in the Configuration block, i.e. two different counters used for generating the periodical refresh. The intention of having two levels of refresh is to be able to differentiate between configuration registers inside the SCTMTC ASIC that are never to be updated by other values than the ones in the mission PROM, referred to as system level refresh, and configuration registers that might need other values than those fetched from the mission PROM during the mission, referred to as user level refresh. The user can disable the user level of refresh by disabling the corresponding timer and handle the refresh via the Control Interface (CI). Note that the user level refresh has higher priority than the system level refresh. It is therefore important to select the refresh period of the user level refresh high enough not to disturb the system level refresh. An ongoing refresh can, however, not be interrupted by any other event.

Since the block includes two levels of refresh counters it is possible to have a sort of fallback refresh scheme if the user is to take over the refresh from the user refresh level counter for a period. By having the system refresh level counter updating (enabling) the user level counter periodically then the user will have to disable the user level counter periodically in order to prevent the user level counter to initialise a refresh. This will provide the system with a watchdog monitoring feature for the user handled refresh of the SCTMTC configuration registers, if the user software is malfunctioning then the SCTMTC ASIC can be made to go back to a safe operating mode.

Note that the SCTMTC ASIC is designed using hardened flip-flops, which will allow missions to omit the refresh function in benign operating environments (cf. section 7.12).

The SCTMTC ASIC can be used in stand-alone operation, it can be configured to act as a telemetry encoder or telecommand decoder only, if required.

The selection between Packet Wire and SpaceWire interfaces for the Control Interface is done during configuration.

5.4.1.4.1 Reinitialisation

It is possible to force the Configuration block to perform the power-up configuration during operation by asserting *ReInit*. This provides a means of resetting the SCTMTC ASIC.

5.4.1.4.2 Interrupt Handling

The SCTMTC ASIC has several interrupt sources in each module described in the subsequent sections. Each module has an individual interrupt that is merged as a single interrupt output signal *Irq* from the SCTMTC ASIC. Each interrupt source can be masked and cleared in the corresponding module. Refer to the module descriptions for further details.

5.4.1.5 Module Interconnections

The SCTMTC ASIC is based on several individual modules each implementing a complex function, e.g. the packet telemetry encoder is implemented in a single module. Each module has been described as self-contained as possible, to simplify the understanding of its function. However, there is some interaction between the different functions that is not part of the description of each module, since the interaction is considered being part of the system functionality. Several such interconnection and interactions between the different function, i.e. modules, are described hereafter.

5.4.1.5.1 SpaceWire to Packet Telemetry Encoder

The SpaceWire module includes eight internal parallel interfaces of which the first seven (number 0 to 6) are internally connected to the Packet Telemetry Encoder (TME) internal parallel interfaces A to G, respectively. The SpaceWire packet header is used as routing information. This makes it possible to send SpaceWire packets that include CCSDS packets to the TME and also have them routed to different Virtual Channels on the telemetry downlink. Large CCSDS packets may be divided into smaller SpaceWire packets and thus several CCSDS packets can be transferred simultaneously using interleaved SpaceWire packets. It is also possible to abort a CCSDS packet transfer by using the SpaceWire mechanism of *Exceptional End of Packet* (EEP). If a Virtual Channel in the TME is not configured for reception from the SpaceWire module, any data sent to that channel will be discarded.

5.4.1.5.2 SpaceWire to Control Interface

One parallel interface of the SpaceWire module (number 7) is connected to the Control Interface Module (CI). This makes it possible to send SpaceWire packets which include a command that will be executed in the Control Interface Module in the same way as data are sent via the external Packet Wire interface on the Control Interface Module. It is possible to enable or disable this functionality via the Clock and Reset block during the configuration of the device.

5.4.1.5.3 Packet Telemetry Encoder connections

The Packet Telemetry Encoder (TME) supports eight Virtual Channels, for which the physical implementations are named A to H. Eight external serial Packet Wire interfaces can be connected to each Virtual Channel A to H. It is also possible to connect Virtual Channel A to G to the SpaceWire interface. For each Virtual Channel, either the Packet Wire or the SpaceWire interface can be used at a time.

Idle Packet or Idle Source Packet insertion is supported for Virtual Channels A to D. An abort function is implemented for Virtual Channels A to G, but can only be used in conjunction with the SpaceWire interface. The abort function is further described in section 5.4.7.2. Idle Transfer Frames are only generated on physical Virtual Channel H.

Note that the actual Virtual Channel Identifier that is transmitted in the Telemetry Transfer Frame is programmable for each physical Virtual Channel.

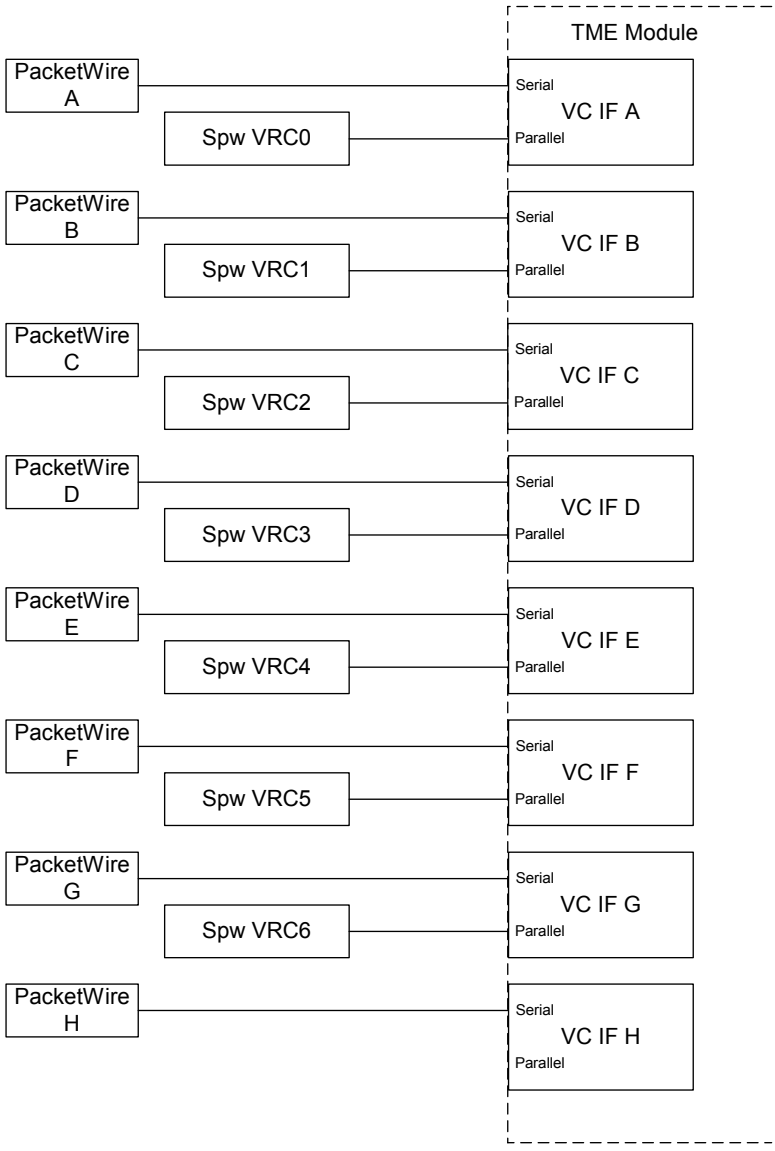


Figure 5-8 Packet Telemetry Encoder connections

5.4.1.5.4 Sources to CPDM Selector

It is possible to feed the pulse command request interface of the CPDM Selector Module (CSEL) from three different sources, PDEC3, Control Interface/SpaceWire (PM) and ExtCpduIf, as shown in the figure below.

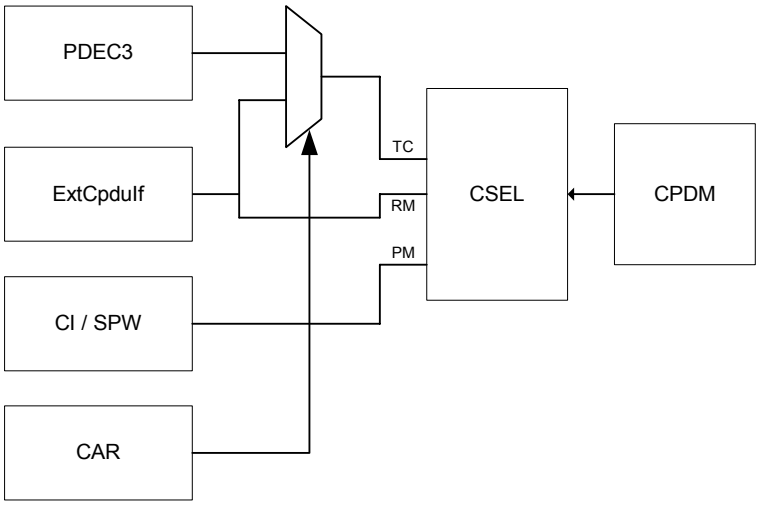


Figure 5-9 CSEL input multiplexing

The connectivity is set up during the configuration of the SCTMTC ASIC (set in the CAR block using the mission PROM). There are two scenarios that are possible:

- PDEC3 is connected to the CSEL TC input. ExtCpduIf is connected to the CSEL RM input, if enabled. CI/SPW is connected to the CSEL PM input. This is the nominal configuration.
- PDEC3 has no internal connection to the CSEL. ExtCpduIf is connected to the CSEL TC input. The CSEL RM input is unused. CI/SPW is connected to the CSEL PM input. This is an alternative configuration where there is no internal connection between the PDEC3 and the CSEL. The external interface is therefore connected to the CSEL TC input since the corresponding requests will never be discarded by the CSEL.

5.4.1.5.5 Packet Telecommand Decoder to CPDM Selector

The PDEC3 module is internally connected to the CSEL module using a dedicated internal MAP interface. This allows the telecommand decoder to set the CSEL in the TC Only mode that is described in detail in section 5.4.5.1. The intention of this is to be able to shut down a “babbling idiot” connected to the External CPDU Interface or the Control Interface from ground, thus ensuring that CPDU Telecommand Segments from ground are always executed.

5.4.2 Memory Interface

The Memory Interface interconnects the internal modules in the SCTMTC ASIC with an external memory space. Since the access time bandwidth of the external memory space is essential to the operation of many of the modules, it is described here in conjunction with the Internal Bus. The memory interface together with the internal bus form the backbone of the SCTMTC ASIC, since all control and data flow through them during configuration and normal operation of the device.

5.4.2.1 Internal Bus

The Internal Bus interconnects all modules in the SCTMTC ASIC. The internal bus is based on a combined priority and round-robin arbitration scheme where each module is given a slot. The modules are divided in two different bandwidth domains, one for the Telemetry Encoder (TME) module only and one for all other modules. This separation is necessary since the telemetry encoding has high constraints on bandwidth and latency. The telecommand decoding has high constrains on latency, especially when authentication is being used. To adapt the performance of the internal bus, the user must configure the bandwidth allocation given to the telemetry encoder for different operating scenarios as will be explained hereafter. The bandwidth allocation is set by means of the PIM TME Ratio Register during configuration (see section 6.10.2.2).

For a given system frequency, a maximum telecommand bit rate can be achieved by giving a maximum bandwidth ratio to the telemetry encoder as listed in Table 5-3.

		System frequency	
		16 MHz	**20 MHz**
Telecommand rate	**Authentication**	**Maximum Telemetry ratio setting**	
50 kbits/s	w AU	55	75
	w/o AU	90	90
100 kbits/s	w AU	15	25
	w/o AU	45	60
150 kbits/s	w AU	2	9
	w/o AU	20	32
200 kbits/s	w AU	-	-
	w/o AU	10	18
256 kbits/s	w AU	-	-
	w/o AU	10	10

Table 5-2 Telecommand bit rates and telemetry ratio setting

Note: Holds for PROM with up to 3 wait states, SRAM with up to 1 wait state read and 2 wait state write. The maximum Telecommand Frame length is 256 octets. The SRAM and PROM/EEPROM data width can be either 8 or 16 bits.

Note: An '-' marker in the table means that the telecommand decoder can be operated at the given frequency if no telemetry encoder is used. The telemetry ratio setting should be programmed to 1.

For a given system frequency, a maximum telemetry bit rate can be achieved by giving a maximum bandwidth ratio to the telemetry encoder as listed in Table 5-2.

			System frequency 16 MHz		20 MHz	
Telemetry	**Constraints**	**SRAM wait states**	**Memory width 8**	**16**	**8**	**16**
			Minimum Telemetry ratio setting			
5 Mbits/s	non-Turbo	1 read, 2 write	5	3	4	2
10 Mbits/s			9	4	8	4
15 Mbits/s			66	6	19	5
20 Mbits/s			-	15	-	7
25 Mbits/s	Reed-Solomon and Secondary Header, non-Turbo	0 read, 1 write	-	9	22	7
30 Mbits/s			-	19	-	8
35 Mbits/s			-	-	-	13
40 Mbits/s			-	-	-	31

Table 5-3 Telemetry bit rates and telemetry ratio setting

Note: Holds for PROM with up to 3 wait states. The SRAM and PROM data width is shown in the table. The SRAM wait states are shown in the table. An '-' marker in the table means that no telemetry can be generated in the given configuration.

Released

			System frequency 16 MHz		20 MHz	
Telemetry	**Constraints**	**SRAM wait states**	**Memory width 8**	**16**	**8**	**16**
			Minimum Telemetry ratio setting			
0,5 Mbits/s	Turbo	1 read, 2 write	11	4	8	3
1,0 Mbits/s			38	9	23	7
1,5 Mbits/s		0 read, 1 write	46	22	68	13
2,0 Mbits/s			-	85	57	27
2,5 Mbits/s			-	-	-	83
3,0 Mbits/s			-	-	-	61

Table 5-4 Telemetry bit rates with Turbo coding and telemetry ratio setting

Note: Holds for PROM with up to 3 wait states. An '-' marker in the table means that no telemetry can be generated in the given configuration.

By configuring the SCTMTC ASIC for a given telemetry ratio as shown in Table 5-2, Table 5-3 and Table 5-4, the resulting maximum telemetry and telecommand bit rates can be achieved.

For all other modules, such as the Control Interface (CI) via SpaceWire or Packet Wire, the memory bandwidth is guaranteed at a system frequency of 16 MHz with PROM with up to 3 wait states, SRAM with up to 1 wait state read and 2 wait state write, independent of memory data width.

5.4.2.2 Memory Map

The memory map of the SCTMTC ASIC is shown in Table 5-5 and Table 5-6. The default memory configuration for SCTMTC ASIC is shown in Table 5-7. The register address map is described in section 6.10. The internal bus includes functionality to support modules that require protection of certain registers. Only the configuration block can access the protected registers.

Area	**Type**	**Start address**	**End address**	**Write access**	**Chip select**	**Data width**
Configuration	Non-volatile	000_0000_{16}	000_8000_{16}-**$0FF\_8000_{16}$**	Enabled	*PromCsN*	8 or 16, by pin
External Recovery LAC Counter	Non-volatile	200_0000_{16}	200_0000_{16}-**$2FF\_8000_{16}$**	Restricted to PDEC3 only	*ExtRecLacN*	8 or 16 program mable
PDEC3 buffer	SRAM	300_0000_{16}	300_1000_{16}	Restricted to PDEC3 only	*RamCsN*	8 or 16 program mable
External CPDU Interface		300_1000_{16}	300_2000_{16}	Restricted to CPDU only		
Reserved		300_2000_{16}	300_3000_{16}	Restricted		
PDEC3 AU		300_3000_{16}	300_4000_{16}	Restricted to PDEC3 only		
User area 1		300_4000_{16}-**$3FF\_8000_{16}$**	300_0000_{16}-**$3FF\_8000_{16}$**	Configurable		
User area 2		300_4000_{16}-**$3FF\_8000_{16}$**	300_0000_{16}-**$3FF\_8000_{16}$**	Configurable		
TME / PM		**300_0000_{16}-** $3FF\_8000_{16}$	300_0000_{16}-**$3FF\_8000_{16}$**	Enabled		
Registers	-	700_0000_{16}	702_0000_{16}	Enabled	-	-

Table 5-5 SCTMTC ASIC memory map (default values in bold face)

Note: Only 20 external address pins are available for the Memory Interface, limiting the useful area range for the *PromCsN* from 000_0000_{16} to $00F\_FFFF_{16}$, for the *ExtRecLacN* from 200_0000_{16} to $20F\_FFFF_{16}$, and for the *RamCsN* from 300_0000_{16} to $30F\_FFFF_{16}$.

Note: User areas 1 and 2 are in practice disabled after reset, since the start and end addresses have the same value. It is not permitted to program overlapping memory areas for User area 1, User area 2 and the TME/PM area.

Note: The memory area corresponding to *PromCsN* must always exist and thus the end address can not be configured to zero. Bit 15 of the end address is therefore locked to logical one.

Note: The end address is the first address outside the relevant memory area.

Note: Memory allocated to the TME should not be accessed via the Control Interface.

Area	Start address	Size	Comment
PDEC3 AU key	000_0000_{16}	170_{16}	Allocated 512 bytes
PDEC3 configuration	000_0200_{16}	$C0_{16}$	
Configuration reset pointer	000_02C0_{16}	4	Pointing at 000_1000_{16}
Configuration data	000_0400_{16}	$C00_{16}$	Allocated 3072 bytes
Configuration sequence	000_1000_{16}	1000_{16}	Allocated 4096 bytes
Recovery LAC pointer	200_0000_{16}	4	First in *ExtRecLacN* area.

Table 5-6 Fixed locations in SCTMTC ASIC memory map

Memory type	Size	Chip select
Non-volatile	512k * 16 or 1024k * 8 (size selectable by pin)	*PromCsN*
Non-volatile	512k * 16	*ExtRecLacN*
SRAM	512k * 16	*RamCsN*

Table 5-7 Default memory configuration

The SCTMTC memory map includes areas to which write accesses are *restricted* to certain modules by the hardware. Each such area has a fixed size of 4096 bytes. The Packet Telecommand Decoder (PDEC3) module is allocated two such areas, one used as a general buffer and one used to hold the programmable key of its Authentication Unit. In addition, if the PDEC3 Authentication Unit (AU) uses an external recovery LAC counter, it will be allocated one such area which has a dedicated chip select signal *ExtRecLacN*, else the area will not be restricted.

The External CPDU Interface (ExtCpduIf) module is allocated one restricted area in which CPDU Telecommand Segments are temporarily stored while being received over the Packet Wire interface and until being fully processed by the Command Pulse Distribution Manager (CPDM). In addition, there are two programmable user areas for which the write protection can by configured and controlled via the internal bus.

Accesses to unmapped address space will result in access errors, as described in section 6. It is explained in sections 6.8 and 6.9 how this affects the Control Interface (CI) behaviour. The memory data width is programmable for the three chip selects listed in Table 5-5. The size of the *PromCsN* controlled area is set by means of an external pin. The size of the others is decided during configuration.

5.4.2.3 Memory interfacing capability

The memory interface allows the SCTMTC ASIC internal modules and blocks to access the memory space. For each external memory bank it is possible to configure:

- memory size, 8 or 16 bits
- read wait states, 0 to 7
- write wait states, 1 to 7 (0 wait state setting is not useful for write accesses)
- Error Detection and Correction (EDAC) protection, on or off
- write access protection, restricted write, enabled write or disabled write

5.4.2.4 Non-volatile Memories

The SCTMTC ASIC supports the usage of external non-volatile memories for the two memory banks selected by the *PromCsN* and *ExtRecLacN* chip select signals. The interface provided is a simple asynchronous memory interface. The usage of Programmable Ready Only Memory (PROM) devices (non-address latching type) is directly supported by means of a configurable number of wait states to be inserted for read accesses.

The SCTMTC ASIC offers no support for writing Electrically Erasable PROM (EEPROM) devices that have constraints regarding write cycle timing.

5.4.2.5 Error Detection and Correction, EDAC

The Memory Interface has Error Detection and Correction (EDAC) capability. It is capable of interfacing either 8 or 16 bit wide memories. Hamming code is used, with 6 check bits that are stored in parallel with the data. The EDAC corrects single bit errors and detects double bit errors. It is possible to enable/disable the EDAC, the reset value being disabled. The protection is provided for the SRAM memory only.

An interrupt is generated when an error is found in the EDAC protected memory area. Different interrupts are generated for correctable and uncorrectable errors.

It should be noted that even if EDAC protection is used, there is a possibility that data in the external memory might become corrupted due to multiple bits being changed by a single SEU. The rate of such upsets is dependent on memory technology and should always be assessed when SEU sensitive memories are being used with the SCTMTC ASIC.

The external SRAM memory, selected by *RamCsN*, does not require any initialisation after power-on before EDAC usage. Any initialisation associated with memory scrubbing, cf. section 5.4.2.6, is done after the EDAC has been enabled.

5.4.2.6 Memory Scrubbing

It is possible to enable a background low priority scrubbing function. The scrubber can read the entire external SRAM memory, but only one continuous address range can be programmed. The scrubbing rate is programmable. Each time a counter elapses, a scrubber read access is performed. On reception of a correctable error during a scrubber read access an atomic read modify write access with the corrected data is initialised at the same address. On reception of an uncorrectable error during a scrubber read access an interrupt is generated. The scrubbing is not stopped by this event.

The memory scrubber end address can not be set higher than the end address of the TME area of memory bank 3. If the user areas are to be placed at higher addresses than the TME area and scrubbing is needed for these the TME area must be set up to overlap both SGM areas. Since the scrubber can scrub a single continuous memory area only it is recommended to place the user areas directly after the programmable AU key of the PDEC3 in order to be able to scrub both the AU key and the user areas.

It is advisable to initialise the memory area the scrubber operates in since false error reports will be received otherwise.

5.4.3 Packet Telecommand Decoder Module (PDEC3)

The Packet Telecommand Decoder (PDEC3) module is compliant with the Packet Telecommand (TC) protocol and specification in [TC_STD] and [TC_SPEC]. The decoder is also compatible with the CCSDS recommendation stated in [CCSDS_TC*].

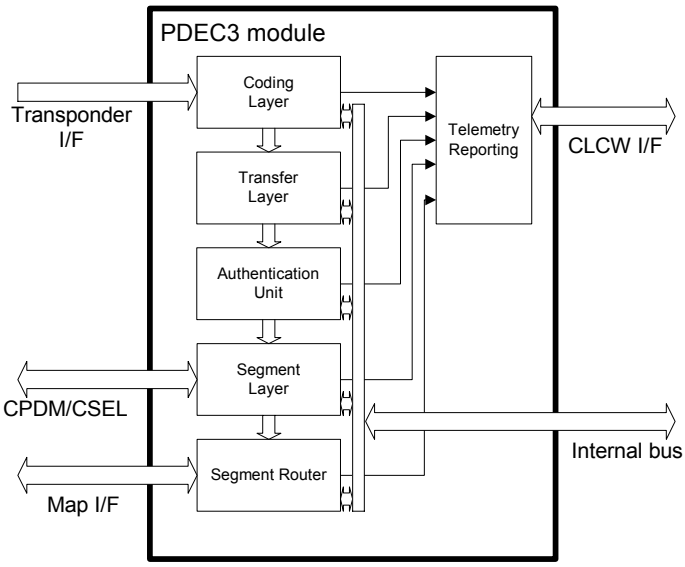


Figure 5-10 PDEC3 module

The decoder receives bit streams on multiple TC Channel inputs. The streams are assumed to have been generated in accordance with the Physical Layer. In the Coding Layer, the decoder searches all input streams simultaneously until a start sequence is detected. Only one of the TC Channel inputs is selected for further processing. The selected stream is bit-error corrected and the resulting Telecommand Transfer Frame is passed to the next layer. In the Transfer Layer, the decoder verifies that the received frame is clean and legal, and that the correct spacecraft is being addressed. The resulting Telecommand Segment is temporarily stored in an external memory and is passed to the next layer. In the Segmentation Layer, the decoder makes the telecommand segments available for readout on multiple external Multiplexed Access Point (MAP) interfaces, via dedicated internal interfaces or over the internal bus.

Specifically addressed telecommand segments are provided to the Command Pulse Distribution Unit (CPDU) where additional verification is made in accordance with the Packetisation Layer. From the resulting CPDU Telecommand Packets, command instructions are extracted and executed as pulse commands. The CPDU is thus not part of the PDEC3 module.

Telecommand Segments can optionally be protected by means of message authentication. The Authentication Layer is placed between the Transfer and Segmentation Layers. The authentication unit is a built-in function in the decoder. There is no direct support for an external authentication unit that can operate between the two layers. Instead, telecommand segments are output on the MAP interfaces with the Segment Trailer intact, allowing an external unit to process the information and to provide further distribution of authenticated telecommand segments. The externally authenticated telecommand segments could be routed to the CPDU, since it is implemented external to the decoder.

The Command Link Control Word (CLCW) is made available for readout by two telemetry encoders for transmission to the ground. In addition, the CLCW and other telecommand decoder reports, such as the Frame Analysis Report (FAR) and the Authentication Unit Status Report (AUSR), can be read via the internal bus.

The operation of the decoder is mainly configured by means of accesses to an external non-volatile memory, where mission critical parameters such as the spacecraft identifier are stored.

5.4.3.1 Coding Layer

The Coding Layer synchronises the incoming bit stream and provides an error correction capability for the Command Link Transmission Unit (CLTU). The Coding Layer receives a dirty bit stream together with control information on whether the physical channel is active or inactive for the multiple TC channels, each consisting of the *PdecTcIn**, *PdecTcClk** and *PdecTcAct** input signals.

The bit stream is assumed to be NRZ-L encoded, as the standards specify for the Physical Layer. There are no assumptions made regarding the periodicity or continuity of the clock signal *PdecTcClk** while a channel is inactive, i.e. *PdecTcAct** is deasserted.

Searching for the Start Sequence, the Coding Layer finds the beginning of a CLTU and decodes the subsequent TC Code Blocks. As long as no errors are detected, or errors are detected and corrected, the Coding Layer passes clean octets of data to the Transfer Layer. When a TC Code Block with an uncorrectable error is encountered, it is considered as the Tail Sequence, its contents are discarded and the Coding Layer returns to the Start Sequence search mode.

The Coding Layer provides status information for CLCW and FAR to be generated in the Telemetry Reporting.

It is possible to enable an optional de-randomiser according to [CCSDS_TC1].

5.4.3.1.1 Synchronisation and Selection of TC Channel

Synchronisation is performed by means of bit-by-bit search for a Start Sequence on the TC channel inputs. The detection of the Start Sequence is tolerant to a single bit error anywhere in the Start Sequence pattern. The Coding Layer searches both for the specified pattern as well as the inverted pattern. When an inverted Start Sequence pattern is detected, the subsequent bit-stream is inverted till the detection of the Tail Sequence. The Start Sequence search can operate in two different modes: Standard Selection Mode or Priority Selection Mode. Only the Standard Selection Mode is compliant with [TC_SPEC]. The modes are described hereafter.

5.4.3.1.1.1 Standard Selection Mode

In the Standard Selection Mode, the detection is accomplished by a simultaneous search on all active channels. The first TC channel where the Start Sequence is found is selected for the CLTU decoding. The selection mechanism is restarted on any of the following events:

- The TC channel's *PdecTcAct** is deasserted, or
- a Tail Sequence is detected, or
- a Code Block rejection is detected, or
- an abandoned CLTU is detected, or
- the clock timeout expires.

The Standard Selection search mechanism is based on a round-robin search between all input channels, where one channel is searched each system clock cycle. Inactive channels are not skipped in the search mechanism, but they can not become selected, i.e. if only one channel is active it is still only checked once each 6^{th} clock cycle. As a result of this the priority between the input channels is not even. If the same data is expected on two channels, e.g. if data is received through two transponders, it is recommended to select one of the following pairs: 0 – 3, 1 – 4, or 2 – 5, to achieve equal priority between the two inputs.

5.4.3.1.1.2 Priority Selection Mode

In the Priority Selection Mode, the TC channel 0 has highest priority, followed by TC channel 1, followed by the remaining four inputs which have equal priority.

In Priority Selection mode, TC channel 0 alone is selected for the Start Sequence search, when *PdecTcAct0* is asserted. All the other channels are immediately disabled, even if one was already selected and receiving data. TC channel 0 will remain selected, when it has been selected, until any of the following events occurs:

- *PdecTcAct0* is deasserted, or
- the clock timeout expires, or
- the Start Sequence timeout expires.

TC channel 1 alone is selected for the Start Sequence search, when TC channel 0 is inactive (including the case of a timeout) and *PdecTcAct1* is asserted. All the lower priority TC channels are immediately disabled, even if one was already selected and receiving data. TC channel 1 will remain selected, when it has been selected, until any of the following events occurs:

- *PdecTcAct0* is asserted and TC channel 0 is selected as described above, or
- *PdecTcAct1* is deasserted, or
- the clock timeout expires, or
- the Start Sequence timeout expires.

The selection between the remaining inputs is performed as in the standard selection mode, when both TC channels 0 and 1 are inactive (including the case of timeout).

5.4.3.1.1.3 TC Channel Timeouts

As a protection mechanism in case of input failure, a clock timeout is provided for all selection modes. The Clock Timeout expires when no falling edge on the *PdecTcClk** input of the selected TC channel in decode mode has been detected for a period of 33.6 ± 20 % million *SysClk* cycles. When the clock timeout has expired, the TC channel in question is ignored (i.e. considered inactive) until its *PdecTcAct** input is deasserted.

In Priority Selection Mode, a Start Sequence timeout is also provided for TC channels 0 and 1. The Start Sequence timeout expires when no start sequence has been detected for the selected channel in search mode for a period of 1320 ± 20 % million *SysClk* cycles. When the start sequence timeout has expired, the TC channel in question is ignored (i.e. considered inactive) until its *PdecTcAct** input is deasserted.

The occurrence of a time out can be observed in the PDEC3 **Monitor** register.

5.4.3.1.2 CLTU Decoding

CLTU decoding functions according to the state machine illustrated in Figure 5-11 and defined in Table 5-8. The events and the corresponding actions in the state machine are those defined in Table 5-9.

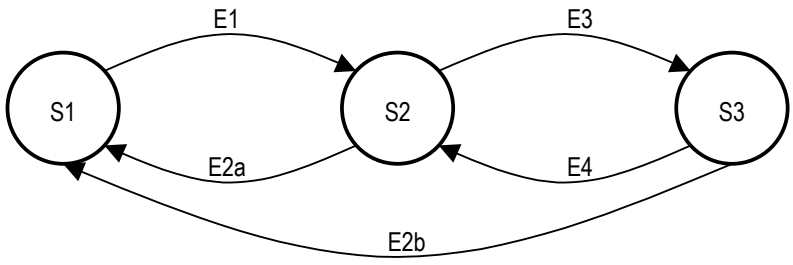


Figure 5-11 CLTU decoding state diagram

State number	State name	State definition
S1	Inactive	All TC channels are inactive.
S2	Search	The incoming bit-stream is searched for the Start Sequence.
S3	Decode	TC Code Blocks, which are either free of errors or contain correctable errors, are received and decoded.

Table 5-8 CLTU decoding states

Event number	Event name	Event definition
E1	Channel activation	At least one TC channel is active, as defined by: • At least one *PdecTcAct** signal for a TC channel for which a timeout has not expired is asserted. Action: • Go to the Search state S2.
E2a	Channel deactivation	The last active TC channel becomes inactive, as defined by: • Its *PdecTcAct** input is deasserted Action: • Go to the Inactive state S1.
E2b	Channel deactivation	The selected TC channel becomes inactive, as defined by: • Its *PdecTcAct** input is deasserted, or • The first TC Code Block after the Start Sequence is rejected (i.e. the CLTU is too short), or • The 148th Code Block after the Start Sequence is accepted (i.e. the CLTU is too long), or • The clock timeout expires, or in priority mode, the Start Sequence timeout expires, or in priority mode, the *PdecTcAct** input of a higher priority TC channel is asserted. Actions: • Discard any received Information octets since the Start Sequence was detected, and • Report the CLTU as abandoned, and • Go to the Inactive state S1 (in case at least one TC channel is still active, event E1 will then immediately occur).
E3	Start Sequence found	The Start Sequence is detected. Action: • Go to the Decode state S3.
E4	Code Block rejection	TC Code Block 2 to 148 is rejected as a Tail Sequence. Actions: • The contents of the rejected Code Block, which is considered to be the Tail Sequence, are discarded, and • The Information octets of all preceding Code Blocks of the CLTU are transferred to the Transfer Layer, and go to the Search state S2.

Table 5-9 CLTU decoding events and actions

Released

5.4.3.1.3 TC Code Block Decoding

The received TC Code Blocks are decoded using the (63,56) modified BCH code. Any single bit error in a received TC Code Block is corrected. A TC Code Block is rejected as a Tail Sequence if more than one bit error is detected.

Figure 5-12 schematically shows the decoder for the modified (63,56) BCH code.

- The Even-Odd Detector (EOD) and the 6-stage Shift Register (SR) are set to zero before receiving each TC Code Block.
- The input switch is set in position 1 (non-inverted data) for the 56 Information bits, and in position 2 (inverted data) for the seven Parity bits.
- Stopping all shift registers during the last bit period of the TC Code Block ignores the Filler bit.
- The contents of SR are transferred to the 6-stage Position Location Register (PLR) after the 63rd shift by momentarily closing the switches. The contents of the Buffer Register (BR) and PLR are shifted at the same time, to allow correction of a single-bit error at the output of BR, if any. The correction occurs when the contents of PLR are 00_0001_2 and the value of Hold is 1, as indicated in the figure.

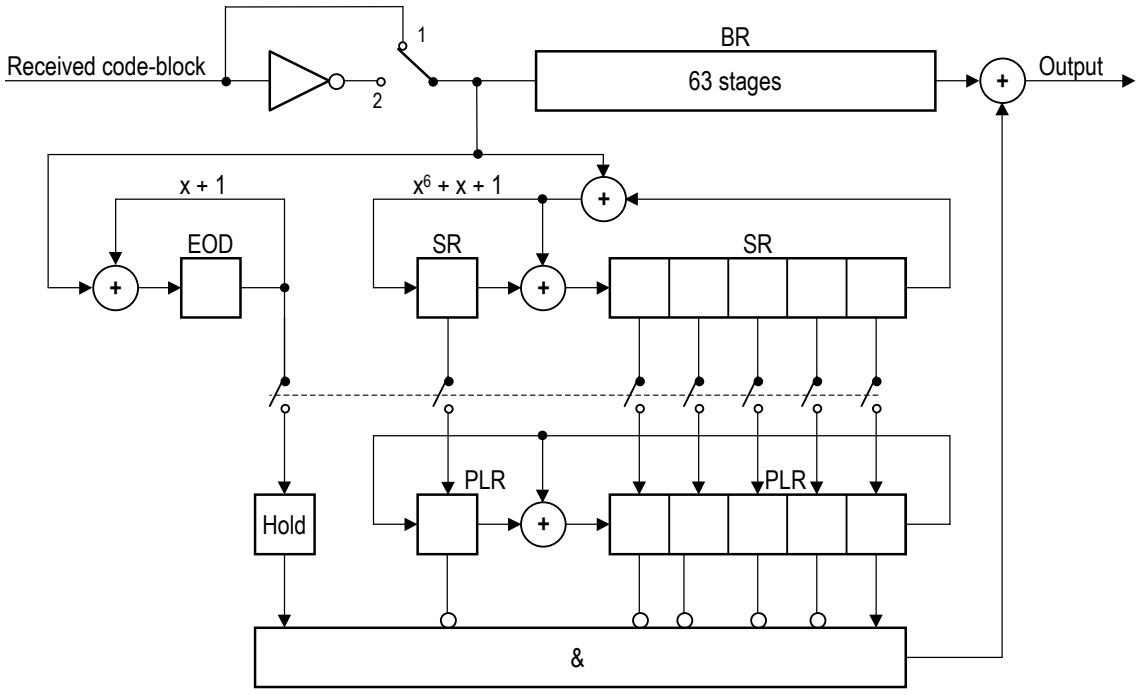


Figure 5-12 (63,56) modified Hamming decoder

The strategy used to decide whether to accept the TC Code Block, or to reject it as a Tail Sequence and discard its Information octets is shown in Table 5-10 TC Code Block acceptance strategy.

EOD value	SR value	Filler bit value	Decision
0	0	Ignored	No errors: the TC Code Block is accepted without correction
0	≠0	Ignored	Detection of even errors: the TC Code Block is rejected (event E4 in Table 5-9)
1	0	Ignored	Detection of non-correctable (i.e. more than one) odd errors: the TC Code Block is rejected (event E4 in Table 5-9)
1	≠0	0	Detection of odd errors in an incorrect Code Block: the Code Block is accepted with correction of one bit
1	≠0	1	Detection of odd errors in an incorrect Code Block, but the Filler bit is one: the Code Block is rejected (event E4 in Table 5-9)

Table 5-10 TC Code Block acceptance strategy

5.4.3.1.4 De-Randomiser

In order to maintain bit synchronisation with the received telecommand signal, the incoming signal must have a minimum bit transition density. If a sufficient bit transition density is not ensured for the channel by other methods (e.g., by use of certain modulation techniques or data that is phase-coherent with the subcarrier) then the randomiser defined in this section is required. Its use is optional otherwise. The presence or absence of randomisation is fixed for a physical channel and is managed (i.e., its presence or absence is not signaled but must be known a priori by the spacecraft and ground system). A random sequence is exclusively OR-ed with the input data to increase the frequency of bit transitions. On the receiving end, the same random sequence is exclusively OR-ed with the decoded data, restoring the original data form.

The random sequence is generated using the following polynomial:

$$h(x) = x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$$

At the receiving end, the derandomisation is applied to the successfully decoded TC data. The de-randomiser remains in the “all-ones” state until the CLTU Start Sequence has been detected. The pattern is exclusively OR-ed, bit by bit, to the successfully decoded data (after the Error Control Bits have been removed). The de-randomiser is reset to the “all-ones” state following a failure of the decoder to successfully decode a TC codeblock or other loss of TC data.

5.4.3.2 Transfer Layer

The Transfer Layer operates on the information octets received from the Coding Layer, which are considered as a candidate Transfer Frame, with the possible presence of Fill octets. The Transfer Layer checks the correctness of each Transfer Frame, which is split into a clean validation followed by a legal validation. Legal Transfer Frames are then provided to the Frame Acceptance and Reporting Mechanism (FARM-1), whose main purpose is to handle automatic request for retransmission for guaranteeing sequence correctness according to the Command Operation Procedure (COP-1). The Transfer Layer accepts frame lengths up to 1024 bytes.

There are two types of services, the sequence-controlled and the expedited service:

- The sequence-controlled service (AD Frames) is used for normal spacecraft communication. This service is based on an automatic request for retransmission (ARQ) procedure of the "Go-back N" type, and which requires that the Command Link Control Word (CLCW) is provided to the ground. This service guarantees that TC Segments will be accepted in the correct order, without loss or duplication. To configure the sequence control machine special control frames (BC Frames) are used. BC Frames are consumed within the Transfer Layer. The TC Segment of accepted AD Frames and BD Frames are transferred to the Authentication Unit.
- The expedited service (BD Frames) is used for exceptional spacecraft communication, for example when no CLCW is available on ground or during unexpected situations requiring unimpaired access to the spacecraft from the ground. This service guarantees that TC Segments will be accepted in the correct order, but any number of TC Segments may be lost.

The Transfer Layer also provides status information for generating the CLCW and the FAR to the Telemetry Reporting.

5.4.3.2.1 Frontend and Backend Buffers

The decoder has two data buffers in the external memory in which it stores the incoming data, and processes it up to and including the Segmentation Layer.

The frontend buffer is used for storing the incoming candidate Transfer Frame while being received, which is then processed by the Transfer Layer. If it is an AD Frame or BD Frame accepted by the FARM-1, its TC Segment is transferred to the backend buffer. The frontend buffer is then emptied to be ready to receive a new candidate Transfer Frame.

The backend buffer is used for storing the TC Segment for the Segmentation Layer. Furthermore, if the TC Segment is to be authenticated, the Authentication Unit (AU) will process it while in the backend buffer. The backend buffer is emptied by one of the following events. The backend buffer is emptied by the AU, if the authentication fails (and the TC Segment is discarded) or if it is an AU Control Command (which is consumed by the AU). The backend buffer is emptied, if the Segmentation Layer transfers the TC Segments remaining in the backend buffer to a different application. If a legal BD Frame arrives to the Transfer Layer and the backend buffer already contained data (whether partly read out or not), the data will be erased to give place for the new TC Segment.

5.4.3.2.2 Clean Transfer Frame Validation

Every candidate Transfer Frame received from the Coding Layer is verified to be clean by performing the following tasks, in the order presented.

The number of octets received is verified to be greater than seven, i.e. at least two Code Blocks must have been received. The number of Fill octets are verified to be between zero and six, i.e. the number of received Information octets shall be between (value of the Frame Length field + 1) and (value of the Frame Length field + 7). The CRC decoder is initialised to all ones, and the CRC is calculated over the entire Transfer Frame, including the Frame Error Control field, using the standard polynomial specified. The result of the CRC calculation is then verified to be zero. Any Fill octets are discarded.

If the Transfer Frame passes all of the above controls, it is transferred to the Legal Transfer Frame Validation. If any of the controls fail, i.e. the frame is not clean, the candidate Transfer Frame is discarded and reported as being dirty.

5.4.3.2.3 Legal Transfer Frame Validation

Every clean Transfer Frame received from the Clean Transfer Frame Validation is verified to be legal by performing the following tasks.

The static fields of the received Frame Header are verified to be identical to the value of the three Frame Header octets stored in external non-volatile memory, i.e. the Version Number, the Reserved field A, the Spacecraft Identifier, and the Virtual Channel Identifier.

The Transfer Frame type is verified to be an AD, a BC or a BD Frame. If the Transfer Frame is of type BC, the Frame Data field is verified to contain either an “Unlock” or a “Set V(R) to V*(R)” FARM-1 Control Command. If the Transfer Frame is of type BC or BD, the Frame Sequence Number field is verified to be zero.

If any of the controls fail, i.e. the frame is not legal, the Transfer Frame is discarded and reported as being illegal, together with information on which controls failed and which of the static fields did not match, if any. If the Transfer Frame passes all of the above controls it is transferred to the Frame Acceptance and Reporting mechanism.

5.4.3.2.4 Frame Acceptance and Reporting Mechanism

The Frame Acceptance and Reporting Mechanism (FARM-1) is a finite-state machine. The main FARM-1 states are summarised below:

- Open (S1), in which BD Frames and in-sequence AD Frames are accepted. BC Frames are accepted.
- Wait (S2), in which there is no buffer space available to receive a new frame. AD Frames are discarded and requested to be retransmitted. BD Frames are accepted, erasing any data that was already in the backend buffer (aborted data transfer).
- Lockout (S3), which is a safe state entered in case a protocol error is detected, and in which neither AD Frames nor the "Set V(R) to V*(R)" BC Frame are accepted. BD Frames are accepted. The "Unlock" BC Frame is accepted, which resets FARM-1 to the Open state.

The FARM-1 uses the following variables:

- The Lockout flag, which is set to one whenever FARM-1 is in the Lockout state.
- The Wait flag, which is set to one when FARM-1, is in the Wait state. It can also be set to one in the Lockout state.
- The Retransmit flag, which is set to one whenever FARM-1 knows that an AD Frame has been lost in transmission or has been discarded because there was no buffer space available. This flag can be set to one in all states, and it is always set to one in the Wait state.
- The FARM-B counter, which is a modulo-four count of BD and BC type Frames.
- The Receiver Frame Sequence Number V(R), which contains the value of the Frame Sequence Number N(S) expected in the next AD Frame.

The Lockout, Wait and Retransmit flags, the FARM-B counter and V(R) are provided in the CLCW.

5.4.3.2.4.1 FARM-1 Sliding Window

The purpose of the sliding window is to protect FARM-1 against uncontrolled transfers, by limiting the number of Transfer Frames that can be transmitted ahead of the last acknowledged Transfer Frame. The sliding window is defined by two parameters in the external non-volatile memory:

- The width of its positive part, PW.
- The width of its negative part, NW.

The positive window area starts at V(R) (the next expected AD Frame) and extends PW Transfer Frames in the positive direction. The negative window area starts at V(R) - 1 (the last accepted AD Frame) and extends NW frames in the negative direction. The total width of the window is W = PW + NW.

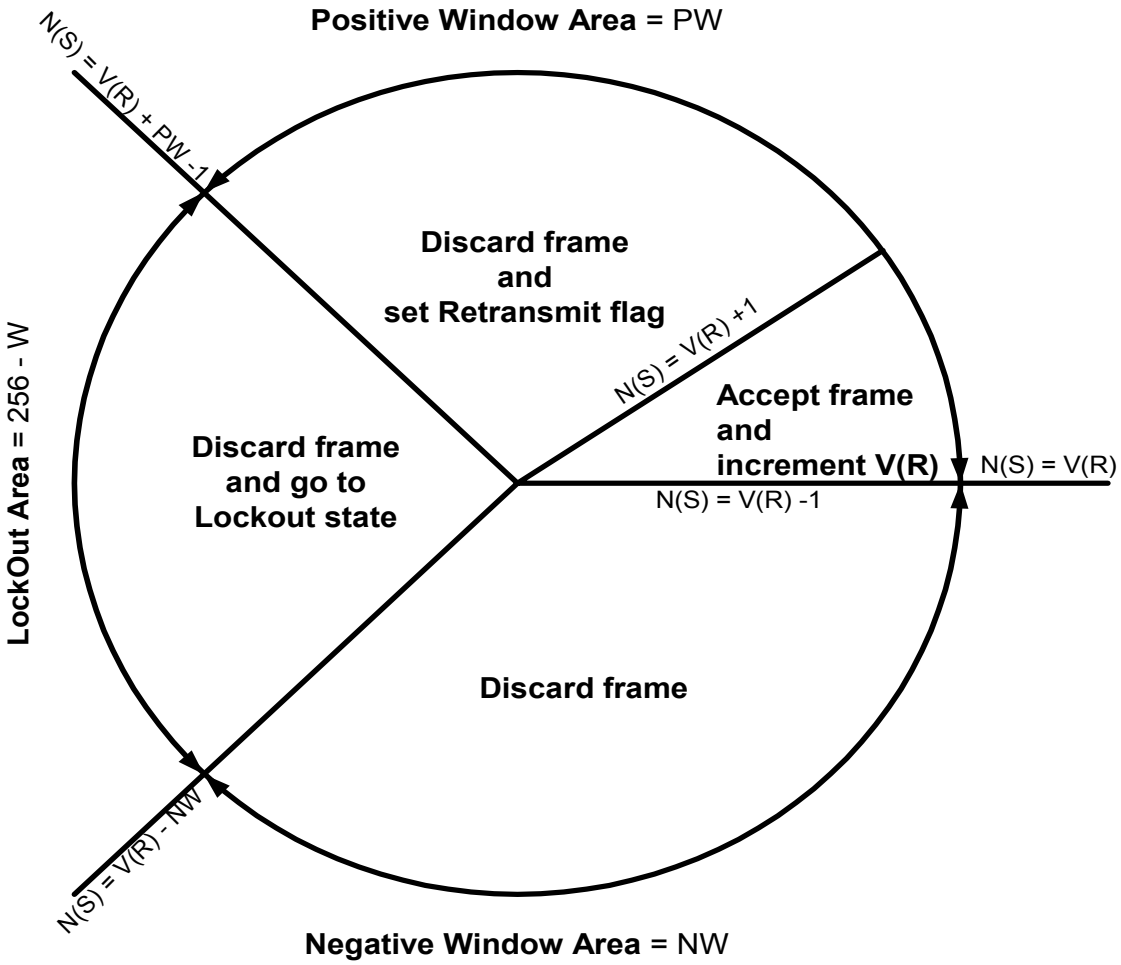


Figure 5-13 FARM sliding window concept

An AD Frame is inside the positive part of the sliding window when:

$N(S) \geq V(R)$, and

$N(S) \leq V(R) + PW - 1$

An AD Frame is inside the negative part of the sliding window when:

$N(S) < V(R)$, and

$N(S) \geq V(R) - NW$

An AD Frame is outside the sliding window (i.e. in the lockout area) when:

$N(S) > V(R) + PW - 1$, and

$N(S) < V(R) - NW$

All operations for the FARM-1 sliding window are performed modulo-256.

5.4.3.2.4.2 FARM-1 State Machine

The FARM-1 state machine remains in a state until an event occurs, in which case the operations specified for that event in the current state shall be executed, as shown in Table 5-11.

FARM-1 event				**Current FARM-1 state**		
Description			**No.**	**Open (S1)**	**Wait (S2)**	**Lockout (S3)**
Legal AD Frame arrives	N(S)=V(R)	Back-end buffer is empty	E1	Accept frame, Increment V(R), Retrans. flag:=0	(Not possible)	Discard frame
		Back-end buffer is not available	E2	Discard frame, Retrans. flag:=1, Wait flag:=1, go to state S2	Discard frame	Discard frame
	Inside positive part of sliding window with N(S)≠V(R)		E3	Discard frame, Retrans. flag:=1	Discard frame	Discard frame
	Inside negative part of sliding window		E4	Discard frame	Discard frame	Discard frame
	Outside sliding window		E5	Discard frame, Lockout flag:=1, go to state S3	Discard frame, Lockout flag:=1, go to state S3	Discard frame
Legal BD Frame arrives (implying buffer released)			E6 + E10	Accept frame, increment FARM-B count	Accept frame, increment FARM-B count, Wait flag:=0, go to state S1	Accept frame, Increment FARM-B count, Wait flag:=0
Legal BC Frame with "Unlock" FARM-1 Control Command arrives			E7	Increment FARM-B count, Retrans. flag:=0,	Increment FARM-B count, Retrans. flag:=0, Wait flag:=0, go to state S1	Increment FARM-B count, Retrans. flag:=0, Wait flag:=0, Lockout flag:=0, Go to state S1
Legal BC Frame with "Set V(R) to V*(R)" FARM-1 Control Command arrives			E8	Increment FARM-B count, Retrans. flag:=0, V(R):=V*(R)	Increment FARM-B count, Retrans. flag:=0, Wait flag:=0, V(R):=V*(R), go to state S1	Increment FARM-B count
No legal frame arrives			E9	No action	No action	No action
Back-end buffer is released			E10	No action	Wait flag:=0, Go to state S1	Wait flag:=0

Table 5-11 FARM-1 state table

When a BD Frame is accepted (event E6), any data still remaining in the backend buffer, whether partly read out or not, will first be erased. If the erased data had been made available at the MAP interface, an aborted data transfer will be signaled. Since event E6 implies that the backend buffer is released (event E10) by the erasure, these two events are grouped together in Table 5-11.

Taking into account all valid combinations of the Lockout, Wait and Retransmit flags, expanded FARM-1 states can be defined, as shown in Table 5-12. The expanded FARM-1 state diagram based on these states is shown in Table 5-11.

Lockout flag	Wait flag	Retransmit flag	Expanded FARM-1 state
0	0	0	Open (S1)
0	0	1	Open – Retransmit (S1R)
0	1	1	Wait (S2)
1	0	0	Lockout (S3)
1	0	1	Lockout – Retransmit (S3R)
1	1	1	Lockout – Wait (S3W)

Table 5-12 Expanded FARM-1 state definition

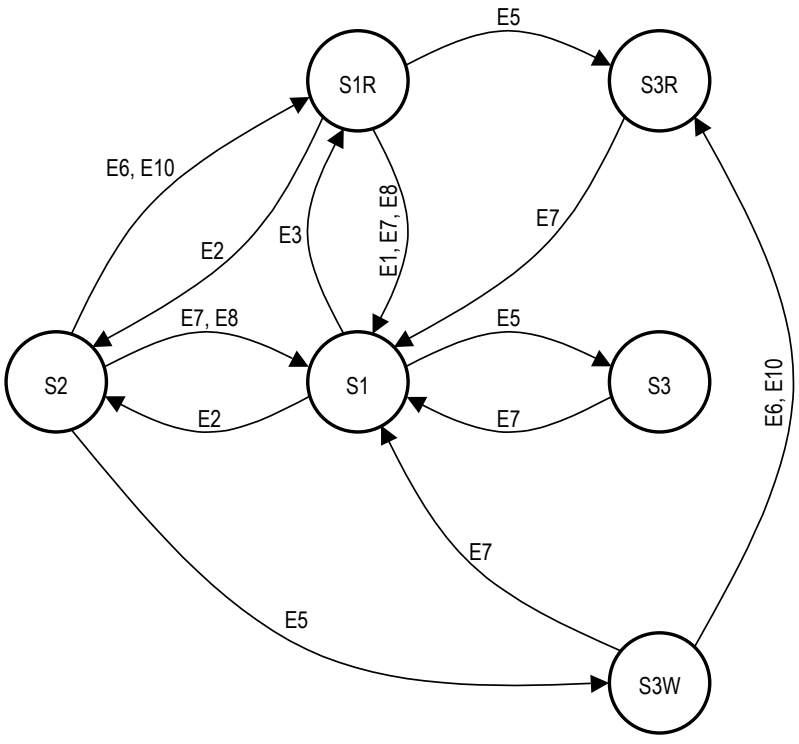


Figure 5-14 Expanded FARM-1 state diagram

5.4.3.2.5 Transfer Layer State After Reset

After power-on reset the Transfer Layer will be in a “cold start” state, as follows:

- FARM-1 is in the Lockout state (S3)
- Lockout flag = 1
- Retransmit flag = 0
- Wait flag = 0
- FARM-B counter = 0
- V(R) = 0
- The frontend and backend buffers are empty.

5.4.3.3 Authentication Unit

The supported authentication technique is a "plain-text-with-appended-signature" system. It consists of appending a digital signature at the end of the TC Segment, without encrypting the data. The Signature is a 5-octet value generated from a secret key, the TC Segment and a LAC Counter value. The Authentication Unit regenerates the Signature for the received TC Segment, and the command is only accepted if the two Signatures match. Three different LAC Counters are provided.

The Authentication Unit consists of the following sub-blocks:

- The Authentication Processor, which regenerates a Signature for each authenticated TC Segment and compares it to the Signature provided in the Authentication Tail.
- The Supervisor, which transfers correctly authenticated TC Segments either to the Segmentation Layer after deletion of the Authentication Tail, or to the Control Command Processor.
- The Control Command Processor, which executes the AU Control Commands.

The Authentication Keys are stored in the external memory, and they are accessed by both the Authentication Processor and the Supervisor.

The authentication sub-layer implemented in the Authentication Unit (AU) is optional. Authentication can be disabled and enabled through the *PdecAuEnable* input. The status of the enabling/disabling is only checked once for each new TC Segment that arrives, allowing the status to be changed at any time without affecting ongoing AU processing.

If the AU is not active, including for MAPs that are not to be authenticated, as described in section 5.4.3.3.2, the entire TC Segment received from the Transfer Layer is forwarded unchanged to the Segmentation Layer. When the AU is disabled this includes TC Segments on the MAP otherwise reserved for AU Control Commands. When the AU is active, successfully authenticated TC Segments are transferred to the Segmentation Layer with the Authentication Tail removed. TC Segments on the MAP reserved for AU Control Commands are consumed within the Authentication Unit.

5.4.3.3.1 Authentication Keys

Two Authentication Keys are supported:

- The Fixed Key, intended for start-up and emergency operations, which is stored in external non-volatile memory.
- The Programmable Key, intended for normal operation, which is stored in external memory.

The contents of the Programmable Key can be modified by some AU Control Commands, as described in section 5.4.3.3.5.

Both Authentication Keys have the structure shown in Figure 6-13, consisting of:

- 60 Hard Knapsack weights numbered W0 to W59, each 48 bits long with bit 0 being the MSB and bit 47 being the LSB.
- The 60-bit Hashing Function coefficient C.

5.4.3.3.2 Selection of Authenticated MAPs

When the Authentication Unit is enabled, all TC Segments with the 5-bit MAP Address in the TC Segment Header being lower than or equal to the Authenticated MAP Id Pointer stored in the external non-volatile memory are processed by the Authentication Unit. If correctly authenticated, the authenticated TC Segment is transferred to the Segmentation Layer with the Authentication Tail removed. If not correctly authenticated, the authenticated TC Segment is discarded. Note that the specification regarding the selection of MAPs to be authenticated is inherently inconsistent in [TC_SPEC].

When the Authentication Unit is enabled, all AU Control Commands (with the 6-bit MAP Identifier being 63) are authenticated by the Authentication Unit. If correctly authenticated, the AU Control Commands are transferred to the Supervisor where they are consumed. If not correctly authenticated, the AU Control Commands are discarded. TC Segments on all other MAPs are transferred directly to the Segmentation Layer without any processing.

Since the selection is performed using the MAP Address and not the entire MAP Identifier, authenticated MAPs are selected in pairs. For example, if the Authenticated MAP Id Pointer has the value three, the first four pairs of MAPs (i.e. MAPs 0 and 32, 1 and 33, 2 and 34, 3 and 35), as well as MAP 63 (AU Control Commands) will be processed by the Authentication Unit.

5.4.3.3.3 Authentication Processor

The Authentication Processor shown in Figure 5-15 consists of the Hashing Function, the Hard Knapsack, the Deletion Box and the Signature Comparator. The received TC Segment is separated into the message **m**, the received LAC value **l** and the Signature **s**. The functions are specified in the following sections. The Authentication Key used by the Authentication Processor is the one last selected, except for the AU Control Commands for selecting the Authentication Key.

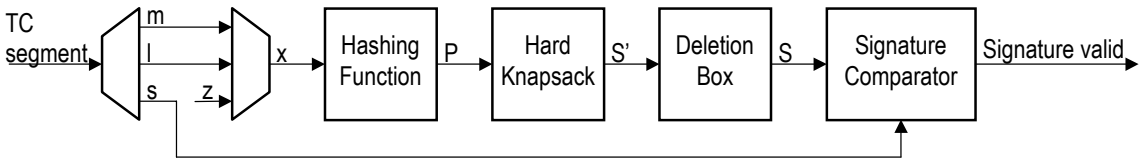


Figure 5-15 Authentication Processor

5.4.3.3.3.1 Hashing Function

The Hashing Function compresses the received TC Segment into a 60-bit pre-signature **P**. This also ensures that the pre-signature is secret, so that there is as much uncertainty about the input to the Hard Knapsack as possible. The Hashing Function consists of a 60-bit Linear Feedback Shift Register (LFSR), as shown in Figure 5-16. The 60-bit feedback coefficient C(0..59) is part of the Authentication Key.

The Hashing function operates in the following manner:

- The LFSR is initialised to the 60-bit value **P'** = $1000\ldots0000_2$ (bit **P'**(0) = 1, all other bits = 0) before the processing of each authenticated TC Segment begins. The LFSR is then fed bit-by-bit with the extended message **x**, which consists of:
- the received message **m,** i.e. the TC Segment without the Authentication Tail;
- the received LAC value **l**, i.e. the LAC Id and the LAC Count;
- the virtual fill **z**, consisting of 24 zeros, which are generated by the Authentication Unit.

The result **P** is the value in the LFSR after the last bit of the extended message **x** has been shifted in.

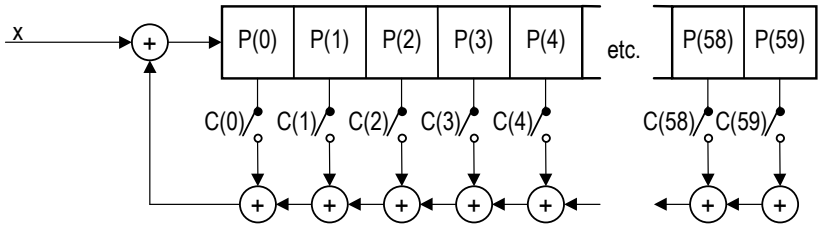


Figure 5-16 Hashing Function

5.4.3.3.3.2 Hard Knapsack

The purpose of the Hard Knapsack is to ensure that it is not possible to deduce the pre-signature **P** from the value of the Signature. The Hard Knapsack is based on the concept of the modular knapsack.

The Hard Knapsack function operates in the following manner. It consists of the 60 weights W0 to W59 being part of the Authentication Key, and it is defined by the following transformation:

$$S' = (\sum_{j=0}^{j=59} P_j W_j) \bmod Q$$

where:

- $\mathbf{Q} = 2^{48}$
- the bits $\mathbf{P}_j$ of the pre-signature **P** select the corresponding weights $\mathbf{W}_j$ of the knapsack to form the integral sum modulo **Q**.

The result is the 48-bit knapsack sum **S´**(0..47), with the MSB being **S´**(0).

5.4.3.3.3.3 Deletion Box

The Deletion Box deletes the eight least significant bits of the 48-bit knapsack sum **S´**, i.e. bits **S´**(40) through **S´**(47). The result is the 40-bit Signature **S**(0..39).

5.4.3.3.3.4 Signature Comparator

The Signature Comparator compares the received Signature **s** with the generated Signature **S**. The result is provided to the Supervisor for the authorisation of the TC Segment.

5.4.3.3.4 Authentication Supervisor

The Supervisor consists of the Logical Authentication Channel (LAC) Counters and the Final Authorisation Function, which are described in the following sections.

5.4.3.3.4.1 LAC Counters

A LAC Counter is used to associate every TC Segment with an authentication sequence number, to ensure that identical TC Segments will not produce the same signature (except at large intervals of time). The selected LAC Counter is incremented by one every time a TC Segment is successfully authenticated. Both the Signature and the associated LAC value are part of the Authentication Tail.

Three LAC Counters are provided:

- The 30-bit Principal LAC Counter.
- The 30-bit Auxiliary LAC Counter.
- The 8-bit Recovery LAC Counter, for which the 22 most significant bits of the 30 bit LAC value are permanently set to one.

If Recovery LAC Configuration LSB is set, an external recovery LAC shall be used. The external recovery LAC resides on address 0200_0000_{16}. The external recovery LAC is fetched after reset, when Recovery LAC Configuration has been checked, and updated as the internal LAC Counters. If the LSB is kept unset, an internal (registers) recovery LAC will be used instead.

5.4.3.3.4.2 Final Authorisation

A TC Segment processed by the Authentication Unit is authorised by performing the following checks, in the order given hereafter. The length of the received TC Segment including the Authentication Tail is verified to be at least ten octets. The Signature **s** of the received TC Segment is verified to be identical with the generated Signature **S**. The received LAC Id must be valid. The value of the received LAC Count field is verified to be identical to the value of the LAC Counter identified by the received LAC Id. If any of the conditions fails, i.e. the TC Segment is not authorised, the TC Segment shall be discarded and the reason reported to the Telemetry Reporting.

If all of the conditions are met, the TC Segment is authorised and accepted as follows. If the TC Segment is authorised, the LAC Counter indicated by the received LAC Id is incremented by one in a wrap-around manner. If the TC Segment is authorised and the MAP Identifier equals 63, the TC Segment is transferred to the Control Command Processor, which will consume it as an AU Control Command (it will never be transferred to the Segmentation Layer). If the TC Segment is authorised and the MAP Identifier do not equal 63, the Authentication Tail is removed and the Segment Header together with the Segment data field is forwarded **as a TC Segment** to the Segmentation Layer.

5.4.3.3.5 Control Command Processor

The Control Command Processor receives the AU Control Commands on MAP 63 from the Final Authorisation Function. An AU Control Command is considered executable if the following checks pass

- The Segment Header shall have the value 255.
- The Command Identifier shall be valid.
- The length of the AU Control Command shall be one octet (for a group 1 command), five octets (for a group 2 command) or nine octets (for a group 3 command).
- If it is the “Set New LAC count value” Control Command, the value of the “LAC Id to be set”-field shall be valid.
- If it is the “Change programmable key block B” Control Command, the value of the Start Address field shall not be higher than 111 (outside the Authentication Key).
- Any AU Control Command not conforming to the above shall be discarded and reported to the Telemetry Reporting as being non-executable.

Executable AU Control Commands are executed as described in the following subsections.

5.4.3.3.5.1 Dummy Segment

This command is intended for testing purposes, and shall have no effect when executed. However, as being correctly authenticated and executable, the telemetry reports are updated accordingly.

5.4.3.3.5.2 Select Fixed Key

This command is **always** authenticated using the Fixed Key, regardless of which Authentication Key was previously selected. If the authentication was successful, the Fixed Key will be selected, otherwise the key previously in use remains selected.

5.4.3.3.5.3 Select Programmable Key

This command is **always** authenticated using the Programmable Key, regardless of which Authentication Key was previously selected. If the authentication was successful, the Programmable Key will be selected, otherwise the key previously used remains selected.

5.4.3.3.5.4 Load Fixed Key into Programmable Key

This Control Command loads the Fixed Key into the Programmable Key memory.

5.4.3.3.5.5 Set New LAC Count Value

This command sets the value of either of the Principal, the Auxiliary or the Recovery LAC Counters, as identified by the LAC Id field of the LAC value to be set. The new LAC value will be set **after** the LAC Counter used by the Final Authorisation Function has been incremented, i.e. it will not be incremented until the next time the same LAC is used. Since the Recovery LAC Counter only has eight bits, the 22 MSBs of the LAC Count value to be set are ignored.

5.4.3.3.5.6 Change Programmable Key Block

There are two commands for modifying the value of any five-octet block of the Programmable Key:

- “Change programmable key block A” concerns a modification starting at any of the first 256 octets of the Programmable Key;
- “Change programmable key block B” concerns a modification starting at any of the last 112 octets of the Programmable Key.

Generation of the five octets to be written to the Programmable Key is be accomplished as follows:

- Once the TC Segment has been authorised by the Final Authorisation Function, the TC Segment except the Signature (i.e. [**m, l**] in Figure 5-15) is inverted and passed once more through the Authentication Processor. The 24 bits of virtual fill **z** are inserted without being inverted. The resulting 40-bit pseudo-signature is then loaded into the Programmable Key memory as follows:
- Bits 32 through 39 of the "pseudo-signature" are loaded into the octet indicated by the Start Address field in Bank A or Bank B, as applicable;
- Bits 24 through 31 of "pseudo-signature" are loaded into the octet at the next higher address (i.e. at Start Address + 1);
- And so on, until bits 0 through 7 are loaded into the octet indicated by the (Start Address field + 4) in Bank A or Bank B, as applicable.

If during the loading any of the addresses falls outside the Programmable Key, the loading will simply be ignored for these addresses.

5.4.3.3.6 Authentication Unit State After Reset

After power-on reset the Authentication Unit is in a "cold start" state, as follows:

- Key in use: Fixed key

Contents of the Principal, Auxiliary and Recovery LAC Counters: all ones. The contents of the Programmable Key are undefined after reset.

5.4.3.4 Segmentation Layer and Router

The telecommand decoder only implements demultiplexing and routing of Telecommand Segments. It does not implement segment re-assembly into Telecommand Packets.

The Segmentation Layer demultiplexes the Telecommand Segments, placed in the backend buffer by the Transfer Layer, to different destinations. The destinations can be one of many Multiplexed Access Point (MAP) serial interfaces or one of three internal interfaces accessed over the internal bus:

- Command Pulse Distribution Module
- CPDM Selector Module (CSEL)
- Processor Module

Only one interface can be active at any one time since there is only one backend buffer. If a new BD Frame is accepted before the existing Telecommand Segment has been fully read out, the transfer is aborted, which is signaled to the external user by the decoder.

5.4.3.4.1 Routing

The routing of Telecommand Segments is performed by means of a MAP Identification Lookup table, as shown in Figure 5-18. The lookup table is stored in external non-volatile memory. The MAP Identifier of the Telecommand Segment to be routed is used as an index to the lookup table. The information in the lookup table is then in turn used for indicating to which destination the Telecommand Segment is to be transmitted. The format of the lookup table entries is described in Figure 5-17 and features a parity bit and a valid bit.

The decoder will look up the correct MapAdr by indexing the lookup table with the MAP Identifier from the Telecommand Segment Header. For each routed segment the lookup value is stored in the MapAdr register, and the 6-bit value MapGenA is provided on the *PdecMapGenA* output. *PdecMapGenA* will not change until a new segment is to be output. The decoder verifies the validity of the lookup value as follows:

- The Par bit must be the odd parity of the MapAdr value, i.e. the number of ones in the entire MapAdr value is odd.
- The Val bit must be one.

If both controls pass, the following take place:

- If MapGenA is equal to 0, the Telecommand Segment is transferred to the CPDM
- If MapGenA is equal to 1-5, the Telecommand Segment is output on the corresponding dedicated MAP interface
- If MapGenA is equal to 6, the telecommand decoder asserts the internal *TcOnly* input on the CSEL module, see section 5.4.5.1.
- If MapGenA is equal to 7, the Telecommand Segment is transferred to the PM via the internal bus, i.e. the SCTMTC ASIC local memory. This will cause an interrupt, which will give the PM the possibility to retrieve the TC packet via the Control Interface.
- Otherwise, the Telecommand Segment is output on the general MAP interface (named MAP G).

If any of the above controls fail, the Telecommand Segment is discarded, emptying the backend buffer (the MapAdr register and the *PdecMapGenA* output are still updated).

If the *PdecMapSwitch* input is asserted, Telecommand Segments destined for MAP interface 1 (MapGenA = 1) are redirected to MAP interface 2 and Telecommand Segments destined for MAP interface 2 (MapGenA = 2) are redirected to MAP interface 1. This is done to be able to automatically route MAP 1 segments to the currently active processor in a redundant system.

MapAdr		
Par	Val	MapGenA
bit 7	bit 6	bit 5..0

Figure 5-17 MapAdr look-up value structure

Released

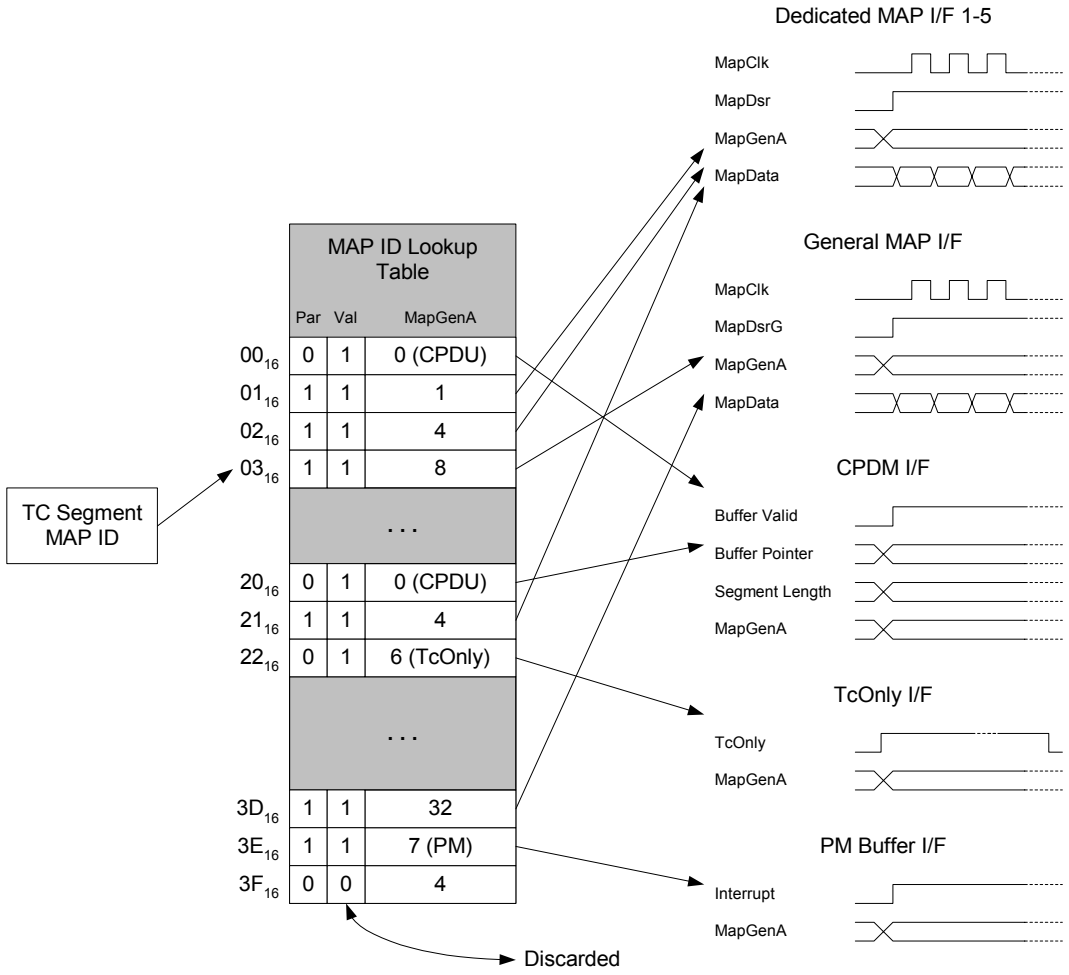


Figure 5-18 Example of MAP ID Lookup

5.4.3.4.2 MAP Interfaces

Each MAP interface serially transfers one Telecommand Segment at a time, with the transfer frequency being individually programmable for each MAP Identifier.

The MAP interface supports data-flow control on a per-octet basis through the *PdecMapDtr** inputs. The transfer frequency and the data-flow control is linked to the Wait state of the FARM-1, which is entered whenever a new legal AD Frame arrives before the previously accepted TC Segment has been read out completely through the MAP interface. The Data Terminal Ready input *PdecMapDtr** should be asserted by the receiving device when it is ready to receive TC Segment Data on the corresponding MAP interface. It can be permanently asserted, asserted once for each entire TC Segment, or asserted on a per-octet basis, depending on the application.

Released

The availability of multiple *PdecMapDtr** inputs, allows for simple design of systems with multiple users on the MAP interfaces, since each user can have its own pair of handshake signals. This also provides a simple way of avoiding having one user blocking all other users by incorrectly deasserting the common *PdecMapDtrG* input. Note also that if a new BD Frame is accepted before the existing Telecommand Segment has been fully read out, the transfer is aborted, which is signaled to the external user by the decoder, which further reduces the risk of a user blocking the MAP interface.

The clock output *PdecMapClk* is used to serially clock out the segment data. *PdecMapClk* starts toggling when both the active *PdecMapDsr** and *PdecMapDtr** signals are asserted. Once it has started toggling, it will continue for eight full clock periods, regardless of the value of the *PdecMapDtr** input. It can however be stopped if the data transfer is aborted. The *PdecMapClk* frequency is determined on a per-MAP basis by the MapClkFreq look-up table in the external non-volatile memory as $f_{MapClk} = f_{SysClk}/2^X$, where *X* is the 4-bit look-up value MapClkFreq[MapId] with a valid range from 1 to 13, and MapId is the value used for looking up MapAdr with a valid range from 0 to 63. To avoid the FARM-1 entering the Wait state, the MAP frequency should be at least ten times higher than the TC Channel input frequency. However, in case all TC Segments having equal length, a factor of two is sufficient.

The Abort Data Transfer output *PdecMapAdt* is used to indicate that the FARM-1 has aborted the transfer of a TC Segment after that it has been signaled as available by asserting the active *PdecMapDsr** output, but before it has been completely read out. When a Segment Data transfer is aborted, *PdecMapAdt* will be asserted first. Subsequently, the active *PdecMapDsr** is deasserted to indicate that no more Segment Data is available, and finally *PdecMapAdt* is deasserted.

Five MAP Interface Status registers allows information about the last transferred TC Segments to be read out through the internal bus. The MapCStat register contains information about the last TC Segment transferred on any of the six MAP Interfaces. Each of the other Map*Stat registers contain information about the last TC Segment transferred on the corresponding Map Interface. Consequently, the MapCStat register is identical to the Map*Stat register for the MAP Interface that was last used.
Each of the Map*Stat registers contain a count of the number of octets actually transferred on the MAP Interface (which can be different from the TC Segment length if the TC Segment was aborted), as well as a flag indicating whether the last transfer was aborted by the decoder. The contents of the Map*Stat registers are only updated after a transfer has been completed, as signaled by *PdecMapDsr** being deasserted. The read-out mechanism ensures that the contents are not changed while a register is being read out.

5.4.3.4.3 Transfer to the CPDM

The decoder has a dedicated internal interface to communicate with a Command Pulse Distribution Module (CPDM). If the CPDM is busy when a new TC Segment for the CPDM arrives, the TC Segment remains in the backend buffer until the CPDM has finished processing the current CPDU commands, after which it is transferred.

In case yet another TC Frame arrives while the TC Segment in the backend buffer is waiting for the CPDM, there are two cases:

- If the incoming TC frame is an AD Frame, it is discarded and the Wait state is entered since the backend buffer is not available. This ensures that CPDU commands transferred using the sequence-controlled service are always executed in sequence.
- If the incoming TC frame is a BD Frame it is accepted, and the TC Segment in the backend buffer is erased, without this being reported.

The incoming frame is accepted and the TC Segment in the backend buffer is erased without this being reported, if the backend buffer is waiting for the CPDM and if the incoming TC frame is a BD Frame. The transfer of the TC Segment to the CPDM takes less time than receiving a new frame, and therefore the transfer can never be aborted while ongoing.

5.4.3.4.4 Transfer to the PM

The decoder has a dedicated internal interface to communicate with a PM via the internal bus. To hand over a buffer over the internal bus, the decoder will issue the TCNew interrupt. BPTR will contain the memory address of the first byte of the TC Segment transferred to the PM buffer area. SLEN will contain the length of the TC Segment transferred to the PM buffer area. The backend buffer is considered free when BFREE is written. If the decoder reclaims the backend buffer when it has been handed over to the internal bus, before BFREE has been written, the decoder will issue the TCAbort interrupt.

5.4.3.4.5 Transfer to the CSEL

The decoder has a dedicated interface to communicate with a CPDM Selector Module (CSEL). This interface is only used for asserting the internal TcOnly input to the CSEL module, as described in section 5.4.5.1.

5.4.3.5 Telemetry Reporting

Telemetry Reporting comprises the Command Link Control Word (CLCW), the Frame Analysis Report (FAR), and the Authentication Unit Status Report (AUSR). Any of the reports can be read out through the Internal Control Bus, and the CLCW can be read through any of two CLCW interfaces. The CLCW interfaces are intended to be connected to two telemetry encoders. The reports can be read out at any time, and they are double-buffered to ensure that once a read-out has started, the contents of the report is not affected by the arrival of new report information.

5.4.3.5.1 Command Link Control Word

The structure of the CLCW generated by the decoder is defined in [TC_SPEC]. Two of the flags are generated combinatorially from the instantaneous values of input pins:

- The No RF Available flag shall be generated from the *PdecRfAvN** inputs; the value of the flag is the logical AND of *PdecRfAvN**.
- The No Bitlock flag shall be generated from the *PdecTcAct* inputs; the value of the flag is the logical NOR of *PdecTcAct**.

In addition to being read out through the internal bus, the CLCW can be read out serially through the two synchronous bit serial interfaces *PdecClcw**. The bits are numbered from 0 (MSB) to 31 (LSB). The interfaces are independent and can be active simultaneously. The value of the CLCW is sampled on the rising edge of the *PdecClcwSamp$* input, and the MSB is then provided at the corresponding *PdecClcwD$* output. For each subsequent rising edge of the corresponding *PdecClcwClk$* input a new CLCW bit is output on *PdecClcwD$*, until all 32 bits have been clocked out (which requires 31 rising edges). The value of *PdecClcwD$* is undefined after any 32nd rising edge of *PdecClcwClk$*.

5.4.3.5.2 Frame Analysis Report

The Frame Analysis Report (FAR) is intended for testing and checkout of the telecommand decoder, and it is generated from information provided from the Coding Layer, the Transfer Layer and the Authentication Unit. The complete FAR is generated whenever one of the events E2b or E4 has occurred in the Coding Layer, as soon as possible after the processing by the applicable layers has completed. It consists of 40 bits, located in the FAR and CBCNT registers. The decoder has a copy of the FAR and CBCNT registers, FARCP and CBCNTCP, which are independent of FAR and CBCNT and behave as these. Since the read out of these registers is destructive, a copy has been added to allow two independent users to access the information.

If a received frame is declared illegal, the FAR.IReason field is set to indicate the reason. In case multiple illegal reasons are reported, the one with the lowest unsigned value is chosen. If no received frame has been declared illegal, the FAR.IReason field is set to “No Illegal report”. In case multiple frame analysis results are reported for the FAR.FrameAna field, the one with the lowest unsigned value is chosen.
The FAR.Stat field shall be set to “New analysis data” and, provided that any earlier FAR has been read out, the NewFAR interrupt will be issued every time a new FAR is generated. The FAR.Stat field is set to “Old analysis report” after every time the FAR register has been read.

If a legal frame to be authenticated is received, the FAR.AuAna field shall be set to indicate the result of the AU processing. In case multiple error cases are reported, they are prioritised as follows:

- “Incorrect length of the TC Segment” is chosen before
- “TC Segment rejected because of error in the Signature”, which is chosen before
- “TC Segment rejected because of error in the LAC”, which is chosen before
- “Non-executable authorised AU Control Command”.

If no legal frame to be authenticated has been received, the FAR.AuAna field is set to "No authentication report". The FAR.LastMap field will only be updated with the value of the TC Segment MAP Identifier field when a new TC Segment has been accepted by the FARM-1. In all other cases this field retains its previous value. The FAR.Type field is set to "No Legal frame" whenever the FAR is generated without any Legal Frame having been received. The CBCNT contains the entire Code Block counter, while the FAR.CbCnt field contains the least significant part of CBCNT.

5.4.3.5.3 Authentication Unit Status Report

The Authentication Unit Status Report (AUSR) provides the status of the Authentication Unit. The AUSR is not generated by any particular event, but shall contain the instantaneous values at any time of the LAC counters and which Authentication Key is selected. It consists of 96 bits, split into three 32-bit registers AUSR1 to AUSR3. The decoder has copies of AUSR1 through AUSR3, AUSRCP1 through AURSCP3, which are independent of AUSR1 through AUSR3 and behave like these.

5.4.4 External CPDU Interface Module (ExtCpduIf)

The purpose of the External CPDU Interface Module (ExtCpduIf) is to provide a way for an external user to interface a Command Pulse Distribution Unit (CPDU). This is done by connecting it via a CPDM Selection Module (CSEL). The input interface is based on the Packet Wire protocol including ready and abort signals. See 5.4.1.5.2.

Data received by the ExtCpduIf are transferred to the CPDU using a buffer pointer and a length indicator, defining where the data can be fetched over the internal bus. The ExtCpduIf operates independently of the data contents, since it is left to the CPDU to verify that it contains a clean and legal CPDU telecommand segment.

The ready signal provides the means to halt a transmission until data can be safely received. Ready signalling is performed on octet basis. The abort signal provides the means for the sender to end a transmission prematurely and thus preventing the data from being sent to the CPDU. The ExtCpduIf is not capable of handling error messages from the CPDU, i.e. the user will not know whether a telecommand segment has been executed, discarded or aborted. The ExtCpduIf will not accept new data until the CPDU has handled the previously received telecommand segment. It is not possible to abort a command sequence after a complete telecommand segment has been received on the input interface and has been forwarded to the CPDU.

The ExtCpduIf will not forward any data that violate the Packet Wire protocol. This includes framing and overrun errors. The ExtCpduIf will not forward a telecommand segment to the CPDU if any of the data transfers over the internal bus has completed with an error. If a telecommand segment is larger than a programmed maximum allowable size, the ExtCpduIf will not forward the telecommand segment to the CPDU.

The ExtCpduIf does not perform any protocol check of the incoming data.

5.4.5 CPDM Selector Module (CSEL)

The CPDM Selector Module (CSEL) arbitrates access to a Command Pulse Distributor Module (CPDM) between three input request sources: the Reconfiguration Module (RM), the Telecommand Decoder (TC), and the Processor Module (PM). CSEL arbitrates between the input request sources, while CPDM executes the command sequence from the source that has been selected.

CSEL and CPDM are part of a Command Pulse Distribution Unit (CPDU). Most applications comprise two CPDUs, of which only one at a time should be executing a command sequence. The local CSEL therefore interfaces a remote CSEL with which it exchanges status information to co-ordinate the actions of the two CPDUs.

5.4.5.1 Operating Modes

CSEL can operate in three main modes: Reconfiguration Module On (RM ON), Reconfiguration Module Off (RM OFF) and Telecommand Decoder Only (TC Only). Transitions between modes are possible by issuing mode transition commands, which is done by changing the external *CselRmOn* input pin, or by means of an internal connection, the *CselTcOnly* signal, from the telecommand decoder PDEC3.

CSEL will transit to the RM ON mode when an assertion of the *CselRmOn* input pin is detected.

CSEL will transit to the RM OFF mode when a deassertion of the *CselRmOn* input signal is detected. CSEL will allow any ongoing sequence to finish before performing the transition.

CSEL will transit to the TC Only mode when an assertion of the internal *CselTcOnly* signal is detected. CSEL will allow any local ongoing sequence to finish before performing the transition. Note that the CSEL does not await the completion of sequences in the remote CSEL when transiting to TC Only mode. Transits to the TC Only mode by issuing this command are temporary; CSEL will return to its previous mode after a programmable time period. The timer value is accessible by means of a register.

It is possible to change the operating mode of CSEL at any time. The operating mode of CSEL is readable via a register. After reset, CSEL operating mode is:

- RM ON, if *CselRmOn* is asserted
- RM OFF, if *CselRmOn* is deasserted

	RM ON	**RM OFF**	**TC Only**
RM Request	Yes	No	No
PM Request	Yes	Yes	No
TC Request	Yes	Yes	Yes

Table 5-13 Requests accepted in different modes

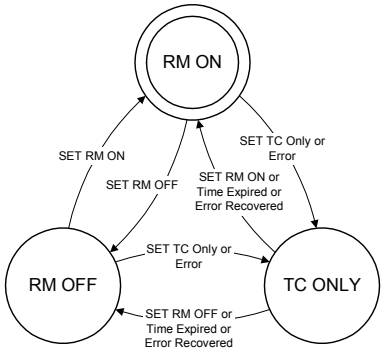


Figure 5-19 CSEL modes

5.4.5.2 Arbitration

CSEL decides what to do with an incoming request based on its own status and on the status of the remote CSEL.

An incoming RM request is accepted and considered for execution if CSEL operates in the RM ON mode. An incoming RM request will be discarded when CSEL operates in either of the RM OFF or TC Only modes. In RM ON mode, all types of requests are accepted, with RM requests being prioritised.

An incoming PM request is accepted and considered for execution if CSEL operates in the RM ON mode or the RM OFF mode. An incoming PM request will be discarded when CSEL operates in the TC Only mode.

An incoming TC request is always accepted and considered for execution. In TC Only mode, only TC requests are accepted and have unrestricted access to CPDM, since CSEL does not interfere with the signalling between the TC request interface and the CPDM interface. CSEL can also enter a fail silent state due to an error and will automatically transit to the TC Only mode.

Table 5-14 lists the actions taken on incoming requests depending on the current state of the two CSELs in the system.

	Both CPDU:s Idle			Ongoing Sequence					
	RM ON	**TC Only**	**RM OFF**	**RM Same**	**RM Other**	**TC Same**	**TC Other**	**PM Same**	**PM Other**
Incoming RM request	**E**	**D**	**D**	**D**	**D**	**A**	**A**	**A**	**A**
	Immediate execution of requested sequence	Request discarded	This request should not occur. (If it still does, the request is discarded)	-	The incoming RM request is discarded when the ongoing sequence finishes. This situation is a part of the normal toggling delay concept.	Allow ongoing pulse to terminate. Then abort ongoing TC sequence, and start executing the RM request.	An incoming RM request in one chain can abort an ongoing TC sequence in the other chain, respecting the pulse termination delay.	Allow ongoing pulse to terminate. Then abort ongoing PM sequence, and start executing the RM request.	An incoming RM request in one chain can abort an ongoing PM sequence in the other chain, respecting the pulse termination delay.
Incoming TC request	**E**	**E**	**E**	**W**	**W**	**W**	**W**	**W**	**W**
	Immediate execution of requested sequence	Immediate execution of requested sequence	Immediate execution of requested sequence	Ongoing RM sequence is allowed to complete. The incoming request is executed ASAP.	Wait until the other CPDU is finished. Then execute the incoming request ASAP.	Ongoing TC sequence is allowed to complete. The incoming request is executed ASAP unless it has been erased by the PDEC in the meantime.	Wait until the other CPDU is finished. Then execute the incoming request ASAP.	Ongoing PM sequence is allowed to complete. The incoming request is executed ASAP.	Wait until the other CPDU is finished. Then execute the incoming request ASAP.
Incoming PM request	**E**	**D**	**E**	**D**	**D**	**D**	**D**	**DA**	**D**
	Immediate execution of requested sequence	Request discarded	Immediate execution of requested sequence	The incoming request is discarded.	The incoming request is discarded.	The incoming request is discarded.	The incoming request is discarded.	The incoming request is discarded and the ongoing sequence is aborted.	The incoming request is discarded.

Table 5-14 Actions taken on incoming requests

Released

D Indicates a situation when the incoming request shall be discarded

A Indicates a situation where an incoming request shall abort an ongoing command sequence.

W Indicates a situation where the ongoing sequence shall be allowed to terminate before the incoming request is allowed to execute.

E Indicates a situation where the incoming request is allowed to execute immediately.

5.4.5.2.1 Reconfiguration Module Requests

RM requests have the highest priority and abort any ongoing sequence when accepted. They can only be accepted in the RM ON mode. CSEL will immediately discard any RM request received in any other operating mode. In systems with two CPDUs, the RM requests are arbitrated between the local and the remote CSEL to prevent the corresponding sequence from begin executed by both CPDUs. When a command sequence in a CPDU Telecommand Packet resulting from an RM request has started, it cannot be aborted by any other request.

When in RM ON mode, CSEL will handle RM requests as follows. If CSEL receives an RM request while the remote CSEL is already busy with an RM request, it will wait until the remote CSEL is not busy with its request anymore and then discard the incoming RM request. In addition, CSEL will abort any ongoing TC or PM sequence in CPDM if the remote CSEL becomes busy with an RM request, independently of the operating mode.

However, if the remote CSEL is not busy with an RM request, CSEL will signal to the remote CSEL that itself is now busy with an RM request. It will allow some time for the remote CSEL to signal back if it in turn has become busy with an RM request as well. If this is the case, CSEL will wait until the remote CSEL is not busy with its request anymore and then discard the incoming RM request.

If the RM request is not discarded as described above, CSEL will abort any ongoing PM or TC sequence in CPDM. If the remote CSEL is busy with an TC or PM request, CSEL will wait until the remote CSEL is not busy with its request anymore. However, if the remote CSEL begins to service an RM request during this wait, CSEL will wait until the remote CSEL is not busy with its request anymore and then discard the incoming RM request. Finally, if the RM request has not been discarded as described above, CSEL will forward the incoming RM request to CPDM.

When CPDM completes the execution of a sequence or when a request is discarded, CSEL will signal to the remote CSEL that it is not busy with the request anymore.

5.4.5.2.2 Telecommand Decoder Requests

TC requests have the second highest priority in the RM ON mode; otherwise they are always prioritised. A TC request is never discarded by CSEL however a ongoing TC sequence can be aborted by an RM request.

When in the RM ON or RM OFF mode, CSEL will handle TC requests as follows. Provided that CPDM is not busy executing a sequence, CSEL is not busy with an RM request, and the remote CSEL is not busy with a TC, RM or PM request, CSEL will signal to the remote CSEL that itself is now busy with a TC request. It will allow some time for the remote CSEL to signal back its status. If any of the preceding conditions change during this wait, CSEL will signal to the remote CSEL that it is not busy with the TC request anymore and wait for the conditions to be valid again before repeating this procedure. However, if the conditions are fulfilled, CSEL will forward the incoming TC request to CPDM. When CPDM completes the execution of the sequence, CSEL will signal to the remote CSEL that it is not busy with the request anymore.

When in the TC Only mode, CSEL will handle a TC request as follows. Provided that CPDM is not busy, CSEL will signal to the remote CSEL that itself is now busy with a TC request. Simultaneously, CSEL will forward the incoming TC request to CPDM. When CPDM completes the execution of the sequence, CSEL will signal to the remote CSEL that it is not busy with the request anymore. In other words, CSEL handles TC requests without checking the status of the remote CSEL. TC requests are directly sent to CPDM to prevent a deadlock due to an error in a CSEL or CPDM.

5.4.5.2.3 Processor Module Requests

PM requests always have the lowest priority. They can only be accepted in the RM ON or RM OFF mode. CSEL will immediately discard any PM request received in any other operating mode. (PM request are made via the Control Interface)

When in the RM ON or RM OFF mode, CSEL will handle a PM request as follows. A PM request will be immediately discarded if CSEL is busy with an RM, TC or PM request, or if the remote CSEL is busy with an RM, TC or PM request. If the PM request is not discarded as described above, CSEL will signal to the remote CSEL that it is now busy with a PM request. It will allow some time for the remote CSEL to signal back if it in turn has become busy with a RM, TC or PM request. If this is the case, or if CSEL itself has become busy with an RM or TC request, CSEL will signal to the remote CSEL that it is not busy with the request anymore and the incoming PM request will be discarded. However, if the PM request is not discarded as above, CSEL will forward the incoming PM request to CPDM. When CPDM completes the execution of the sequence, CSEL will signal to the remote CSEL that it is not busy with the request anymore.

5.4.5.3 Request Interfaces

The RM request interface features a buffer pointer and a buffer length indicator, defining from where CSEL and CPDM can fetch the command sequence. The source can validate this information with a dedicated internal request signal. Dedicated internal signals are used to inform when a request has been executed, discarded or completed with an error. CSEL will abort any RM sequence and transit to the TC Only mode and become fail silent if the internal request signal is deasserted before the sequence has been executed by CPDM or discarded by CSEL. From this follows that the RM source cannot send a new request before the previous request has finished.

The TC request interface features a buffer pointer and a buffer length indicator, defining from where CSEL and CPDM can fetch the command sequence. The source can validate this information with a dedicated internal request signal. A dedicated internal signal is used to inform when a request has been executed. CSEL will abort any TC sequence if the internal request signal is deasserted before the sequence has been executed by CPDM. From this follows that the TC source cannot send a new request before the previous request has finished.

The PM request interface features a buffer pointer and a buffer length register, defining from where CSEL and CPDM can fetch the command sequence. The source can submit a request and validate this information by writing to a request register. CSEL will abort any ongoing PM sequence if the request register is written to before the previous PM sequence has been either completed by CPDM or discarded by CSEL. The new request will also be discarded. Interrupts are used to inform when a request has been executed, discarded, completed with an error or aborted.

5.4.5.4 CPDM Request Interface

CSEL sends telecommand segments to the CPDM via the CPDM Request Interface. The interface provides the CPDM with a buffer pointer and a telecommand segment length indicator. This information is used by the CPDM to fetch the telecommand segment from an appointed address via the internal bus interface. This information is validated with a dedicated internal request output signal.

It is possible to indicate if a telecommand packet is to be restricted to contain only certain command instructions by means of a lockout function, which is done for all Processor Modules. It is possible to indicate if the telecommand packet is from a specific source, for which status reports are to be separated from those of other telecommand packets, which is done for all Telecommand Decoders.

A dedicated internal output signal is used for aborting the execution of a command sequence. Dedicated internal inputs are used to signal when a command sequence has been executed or been abandoned with an error.

If the request signal is released before a command sequence has been completed, this is considered a protocol error. If the buffer pointer or telecommand segment length indicator is changed during the command sequence execution, this is considered a buffer error.

5.4.5.5 Status Interface

CSEL keeps track of the status and actions of a remote CSEL to prevent several CPDM packets from being executed simultaneously in a system. CSEL reports its own status on a regular basis via the CSEL Status interface on which it also receives the status of the remote CSEL. Before granting a request to any of the input sources, CSEL checks the status of the remote CSEL to avoid any conflicts. For all such cases, the remote CSEL is given 16 *SysClk* periods to respond.

The status register for CSEL indicates whether the module is fail silent, busy with an RM request, busy with a TC request, or busy with a PM request. The status register for the remote CSEL indicates whether the module is busy with an RM request, busy with a TC or PM request, and whether there is an error on the status interface.

The main purpose of the status interaface is to synchronise two redundant CPDUs and prevent simultaneous commands from being output from the two CPDUs. To be able to detect stuck-at faults (especially stuck-at-0, which could lead to the CSEL interpreting the remote CSEL as being idle when it is in fact busy) on the link, it is protected by a parity bit. CSEL interprets a parity error in the received status as a physical link error, and enters TC Only mode, since it can not determine the state of the remote CSEL. In this case the redundant CSEL will still be fully operational.

To prevent a deadlock situation from occuring due to the CSEL Status interface, a programmable maximum timeout period is allowed for the remote CSEL during which it can signal that it is busy with an RM, TC or PM request. The timer is restarted whenever the remote CSEL status is idle. The timer value is accessible by means of a register. If the timer expires, an interrupt will be issued and CSEL will consider the remote CSEL to be idle. This will also be reflected in the corresponding status register, which will not change until the remote CSEL status becomes idle. The timeout status can be observed via a register.

5.4.5.6 Fail Silence

To ensure fail silence within the CPDM Selector Module, it consists of two identical decision chains for which the internal outputs are compared before being output. Each chain individually performs the task of selecting actions for incoming requests. In addition, each chain keeps its own status information. Both chains must generate the same result to allow a request to be forwarded to CPDM and status to be reported on the CSEL Status interface.

5.4.5.7 Error Detection and Recovery

CSEL will enter the TC Only mode and become silent on the CSEL Status interface if any outputs differ between the two decision chains in CSEL. An interrupt will be issued on the detection of an internal error. CSEL will deassert its CSEL Status interface signals, which is interpreted as being idle by the remote CSEL. All signals on the interface towards CPDM will be deasserted, which will result in protocol errors between the two modules, which in turn will cause CPDM to abort any ongoing sequence. CSEL will remain fail silent in the TC Only mode until a change on the *CselRmOn* input signal will cause a transit to the RM ON or RM OFF mode.

CSEL will enter the TC Only mode if an illegal code, i.e. wrong parity, is received from the remote CSEL via the CSEL Status interface. An interrupt will be issued on the detection of such an error. Any RM or PM requests will be discarded and any ongoing RM or PM sequences in CPDM will be aborted. CSEL will remain in the TC Only mode until the error has been removed and will then return to its previous operating mode. An interrupt will be issued when this occurs. It is still possible to force CSEL to leave the TC Only mode by changing the *CselRmOn* input signal value, but this might result in an immediate return to the TC Only mode if the illegal code persists.

When entering the TC Only mode due to the errors above, no automatically timed return condition exists.

5.4.6 Command Pulse Distribution Module (CPDM)

The Command Pulse Distribution Module (CPDM) receives telecommand on which it performs clean and legal checks segments via the CPDM request interface from CSEL. If a segment passes these checks, CPDM will interpret the data field of the contained telecommand packet as command instructions. Each command instruction in the data field will result in a specific pulse to be generated for a specified time duration on the Command Pulse Interface. A sequence of command instructions in a telecommand packet is referred to as a command sequence.

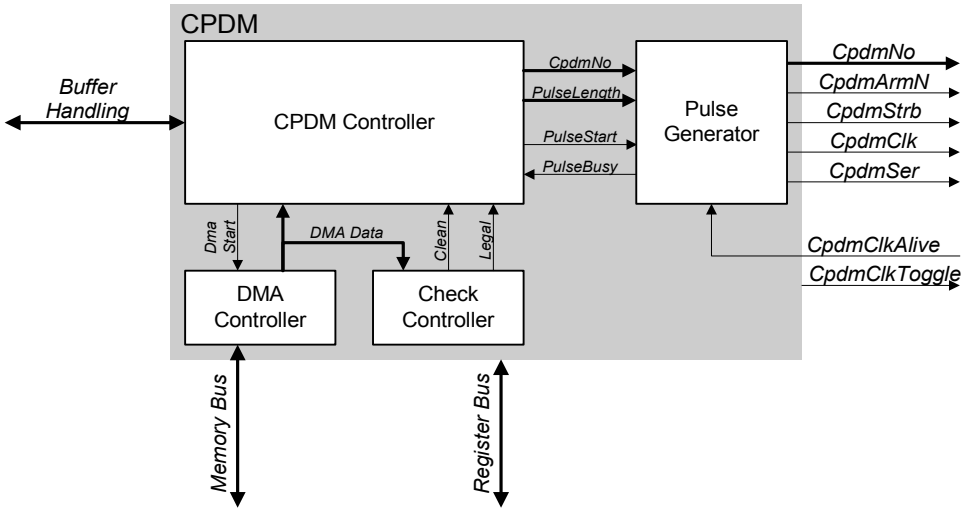


Figure 5-20 Command Pulse Distribution Module

The CPDM Selector Module (CSEL) arbitrates access to CPDM between various input request sources, while CPDM executes the command sequence from the source that has been selected. The CPDM can only be accessed via CSEL. CSEL and CPDM are part of a Command Pulse Distribution Unit (CPDU), together with external command pulse drivers. As this might be the only remaining spacecraft control mechanism in case of a failing computer it is vital that no single faults in the system block the capability to send a command through CPDU. This implies that CSEL, CPDM and the drivers must be free from single faults that can cause a pulse output to be permanently asserted. Consequently, no single point failure will cause the CPDM to refuse to abort a command sequence, to execute an illegal or dirty telecommand packet or to output an incorrect command pulse.

5.4.6.1 CPDU Telecommand Packet

The CPDU telecommand packet and its contents, command instructions, are specified in 4.9.4.1.1. The command instruction contains an eight bit field to specify the command pulse output to one of 256, a three bit field to specify the length of the pulse, and a reserved five bit field of which the contents are undefined and should be ignored by the CPDU. CPDM implements this protocol, but also extends it by utilising the five reserved bits for defining command instruction parity and extension of the command pulse specification to one of 4096 outputs. In addition, CPDM can disallow certain command pulses to be generated by means of a user lockout function, which is available for command sequences sent by the PM. The command instructions can be executed in compliance with [TC_SPEC] provided that the protocol extensions are not enabled in CPDM.

5.4.6.2 CPDM Request Interface

The CPDM receives telecommand segments via the CPDM Request Interface. The interface provides the CPDM with a buffer pointer and a telecommand segment length indicator. This information is used by the CPDM to fetch the telecommand segment from an appointed address via the internal bus interface. This information is validated with a dedicated internal request input signal.

It is possible to indicate if a telecommand packet is to be restricted to contain only certain command instructions by means of a lockout function. It is possible to indicate if the telecommand packet is from a specific source, normally ground, for which status reports are to be separated from those of other telecommand packets. A dedicated internal input signal is used for aborting the execution of a command sequence. Dedicated internal output are used to signal when a command sequence has been executed or been abandoned with an error.

If the request signal is released before a command sequence has been completed, this is considered a protocol error. If the buffer pointer or telecommand segment length indicator is changed during the command sequence execution, this is considered a buffer error. CPDM will abandon the command pulse generation when such errors occur.

5.4.6.3 Segment and Packet Validation

CPDM performs clean and legal checks on any received telecommand segment containing a CPDU telecommand packet. The clean check is performed as follows:

- The Sequence Flags in the Segment Header are verified to be 11_2.
- The number of octets in the received CPDU telecommand packet are verified to be consistent with the contents of the Packet Length field, i.e. the 6 most significant bits of the Packet Length field must be zero, and the value of the 10 least significant bits must be equal to (the telecommand segment length - 8).
- The number of octets in the CPDU telecommand packet is verified to be an even number of at least ten.
- The CRC decoder is initialised to all ones, and the CRC is calculated over the entire telecommand packet, including the Packet Error Control field, using the polynomial $x^{16} + x^{12} + x^5 + 1$. The resulting CRC is verified to be all zeroes.
- If the packet originates from the Control Interface (PM), the lockout check is performed on the telecommand packet as in 5.4.6.4
- If parity check is enabled, the command parity check is performed on all CPDU commands as in 5.4.6.5.

If any of the clean checks above fail, the telecommand packet is considered dirty and the telecommand segment is abandoned with an error.

Legal check is performed on the received telecommand packet as follows:

- The Packet Identification field of the received Packet Header is verified to be identical to configuration register contents.
- The Packet Header Sequence Flags field is verified to be 11_2.

If any of the legal checks above fail, the telecommand packet is considered illegal and the telecommand segment is abandoned with an error. Note that the Packet Identification field comprises the Version Number, Type, Data Field Header Flag, and Application Process Identifier fields.

The outcome of the clean and legal checks is reported in status registers. The Sequence Count from the Packet Header of the last telecommand packet that was declared clean and legal is reported status register.

5.4.6.4 User Lockout

It is possible to prevent CPDU telecommand packets sent by the PM to be executed if they contain specific command pulses. The lockout procedure can be used for preventing software from sending critical command pulses. This lockout procedure is applied per telecommand packet generated by the Control Interface and is enabled via the CPDM request interface. CPDM checks that a lockout enabled telecommand packet does not contain any inaccessible command pulse outputs as a part of the clean check. A command pulse output is considered inaccessible if its output number is less than a lockout threshold, checking all but the most significant bit of the output number. By not comparing the most significant bit results in two lockout areas, which can for example be used for separating low and high power command pulses. Note that all command pulse output numbers in a telecommand packet must comply with the lockout threshold value for a telecommand packet to be considered clean. The lockout condition is then re-checked for each command instruction to be generated.

The command output number that is compared with the lockout threshold comprises the standard 8-bit field (least significant) and the extended 4-bit field (most significant). It is thus possible to use the lockout function with both standard and extended command instruction interpretations. The lockout threshold value is accessible through a register. Setting the lockout threshold to all zero, which should be done when backward compatibility is required, can disable the lockout function.

5.4.6.5 CPDU Command Parity

CPDM provides optional parity check on the command pulse information in CPDU telecommand packets. If parity check is enabled, via a register, CPDM checks the parity on each command instruction both during clean check and once again right before the command pulse is executed. When enabled, parity check is applied to all telecommand packets. The parity over the entire command instruction is odd; i.e. the number of ones including the parity bit must be odd. The command parity is not part of the standard CPDU telecommand packet format and should be consequently be disabled when backward compatibility is required.

5.4.6.6 Command Pulse Timing Reference

The command pulse output timing is based on multiples of *D*, where *D* stands for *duration*. In order to generate correct output timing, CPDM needs to know the system clock frequency. This is achieved by programming in a register the number of system clock cycles that elapse during the time *D*. The *duration* defines the shortest possible strobe length and is typically selected such that 10 ms ≤ *D* ≤ 15 ms.

5.4.6.7 Command Pulse Interface

A legal CPDU telecommand packet contains one or more command instructions, which are executed one after the other, in the original sequence. If the parity check is enabled, CPDM will perform the parity check before executing a command instruction. If a parity error is detected in a command instruction, CPDM will stop executing the telecommand packet with an error, without generating the corresponding command pulse.

Before each command pulse, CPDM outputs all bit fields of the complete command instruction as received in the telecommand packet. This includes the command pulse output number, the pulse length, and the reserved fields containing parity and extension of the command pulse output number specification. The serial interface consists of a data signal and a clock signal. Data bits are valid on rising edges of the clock signal, while the falling edges are used to send the inverse of the data bit. This enables detection of stuck-at faults.

After all bits have been output on the serial interface, the data output will contain the most significant bit of the extended command pulse output number for the remaining time duration of the command pulse generation. This is done to allow external selection, for example between high and low power commands.

The command pulse is then generated by means of an arm signal and a strobe signal. The arm signal will be asserted before the strobe signal is asserted, and it will be deasserted after the strobe signal is deasserted. The strobe signal will be asserted for the time duration specified in the command instruction, i.e. $2^{(\text{pulse length})} \cdot D$. The detailed timing of the command pulse interface is described in 7.6.2.

5.4.6.8 Pulse Generation Timeout

The CPDM contains a timer to ensure that all pulses terminate. If the pulse timer expires, CPDM will immediately stop the ongoing pulse output. CPDM will asynchronously deassert all its outputs on the Command Pulse Interface. The telecommand packet execution will stop with an error and the remaining command instructions will not be executed. The time out period is defined as $160 \cdot D$.

5.4.6.9 Abort

CPDM command sequence execution can be aborted prematurely, e.g. to allow a higher priority command sequence to be executed earlier. CPDM is then required to shorten any ongoing pulses and abort the ongoing command sequence. If a command pulse is being output on the Command Pulse Interface when a command sequence abort occurs, CPDM will perform the following. If the remaining time of the pulse is greater than $5 \cdot D$, the remaining time duration of the pulse will be shortened to $5 \cdot D$. The pulse will be allowed to complete, with the reduced time duration, and no new pulses will be output. The execution of the telecommand packet will be stopped.

When an abort occurs while the command instruction is being output serially on the Command Pulse Interface, CPDM will allow the serial transfer to complete, but no command pulse strobe will be output. The execution of the telecommand packet will be stopped. When an abort occurs before the output of a command instruction, CPDM will stop executing the telecommand packet. Register flags are set when an abort occurred during the execution of the last telecommand packet.

5.4.6.10 Error Handling

A protocol error is an error in the signalling between CPDM and other modules, resulting in CPDM possibly losing the buffer containing the CPDU telecommand packet. A buffer error is an error in the signalling between CPDM and other modules, resulting in a change in the buffer address or telecommand segment length during a telecommand packet execution. A memory access error is when an access over the internal bus is terminated with an error. On any such error, CPDM will allow any ongoing pulse to complete, but will not begin outputting any new pulses. The execution of the telecommand packet will be stopped with an error. Register flags are set when an error occurred during the execution of the last telecommand packet.

5.4.6.11 Clock Active Detection

If the system clock stops while CPDM is outputting a pulse, the CPDM outputs must not remain asserted. To prevent this, CPDM should be connected to an external clock detection circuit, which asserts the *CpdmClkAlive* input signal while the system clock functions properly. The CPDM outputs the system clock divided by 32 on the *CpdmClkToggle* output signal for this purpose. CPDM deasserts asynchronously all its outputs on the Command Pulse Interface when the *CpdmClkAlive* input is deasserted. CPDM does not perform any internal reset when *CpdmClkAlive* is deasserted. Reset of the CPDM can only done by means of asserting the power on reset input. The CPDM will continue to operate undisturbed internally provided that the system clock is available, even if the *CpdmClkAlive* input is deasserted.

5.4.6.12 Interrupt Generation

CPDM makes a distinction between telecommand packets from a specific source, normally ground, for which status reports are separated from those of all other sources. CPDM has therefore two separate status report registers. An interrupt is generated when a status register is updated. A status register contains the outcome of the clean and legal check, the sequence count of the last accepted telecommand packet, an abort indication, an error indication and a register update indicator.

An interrupt is generated when CPDM finishes executing a command sequence in a telecommand packet, independently if this occurs with or without an error.

5.4.7 Packet Telemetry Encoder Module (TME)

The Telemetry Encoder Module (TME) generates telemetry data according to standard protocols defined in [TM_STD] and [CCSDS_TM]. The TME incorporates the user interface to the telemetry encoder, the control of the temporary buffering of user data, the generation of the telemetry Transfer Frame and the downlink bandwidth arbitration between Virtual Channels (VCs). The Telemetry Transfer Frames can be optionally encoded with Reed-Solomon and/or Convolutional encoding, or just Turbo encoding. The TME module can receive input data for up to eight VCs. The received data is stored in an external shared memory, and it is subsequently inserted into telemetry Transfer Frames.

5.4.7.1 Reset of encoders and modulators

The VC Input Interfaces, Frame Generator, Reed-Solomon Encoder, Turbo Encoder, Pseudo-randomiser, Convolutional Encoder, and modulators are reset held inactive while the external *TmeEnable* signal is deasserted. The TME configuration registers are not affected by the *TmeEnable* signal. This results in no data being accepted on Packet Wire or SpaceWire. If data is sent on SpaceWire the result is congestion.

5.4.7.2 VC Input Interfaces

For each Virtual Channel, there is one VC Input Interface (VC*) that receives receive data which is stored temporary in an external memory buffer. Each interface detects the beginning of packet and calculates the corresponding First Header Pointer.

Certain VC Input Interfaces have the ability to fill incomplete telemetry Transfer Frames with Idle Packets. The Idle Packet Version Number is programmable. The Idle Packet insertion will start after a pre-programmed number of polls from the Frame Generator as explained later. When the input interface is idle and the memory buffer is not full a poll counter starts. The counter is incremented on each poll from the Frame Generator; when it expires the Idle Packet insertion commences.

For each VC Input Interface, a variable amount of memory can be allocated for the memory buffer. The size of this memory buffer limits the size of telemetry packets possible to transmit on a particular VC. When transmitting asynchronous data, there are no limits.

Each VC input interface provides two types of interface; serial or parallel, of which only one can be active at a time. The serial interface is based on Packet Wire and comprises a packet delimiter, a bit clock, a data bit and a discrete ready-to-receive signal as discussed above. The parallel interface is directly connected to the SpaceWire module (SPW).

The following parameters can be programmed:

- Whether to use the serial or the parallel input.
- Whether the input data is packet or non-packet data.
- Whether the First Header Pointer should be calculated for non-packet data or not.
- Idle Packet version field value (000_2, 100_2), common for all VCs.
- Idle Packet insertion timeout (Immediate, 1/4/16/64/256/1024 polls or No Idle Packet insertion).
- Values of the Data Field Synchronisation flag and the Packet Order flag (0_2, 1_2).
- Value of the Segmentation Length Identifier field (2 bits).
- The number of frames that have been allocated for each Virtual Channel in the external shared memory (in steps of 8 frames).
- The ready-to-receive threshold (262, 518, 1030, length of Transfer Frame Data Field).

The SCTMTC ASIC provides 8 serial input interfaces, named A to H, and 7 internal connections, named A to G, to the SpaceWire Module (SPW). Idle Packet insertion is supported for the VC Input Interfaces named A to D.

5.4.7.2.1 Serial Interfaces

Each Virtual Channel (A-H) can receive data in serial form from the serial interface, *TmeSIn**, *TmeSClk** and *TmeSValid** (* = A...H). The serial interface also features a *TmeSRdy** output that indicated whether there is room for two octets of data to be received in the internal buffer. The *TmeSRdy** is deasserted when the SPW interface is selected.

5.4.7.2.2 Parallel Interfaces

Each Virtual Channel (A-G) can receive data in parallel form from the SpaceWire Module (SPW). Packets received from SPW can be aborted without violating the telemetry protocol of the Transfer Frame in which they were to be inserted.

Each VC is allocated a certain amount of memory, which is split into several transfer frame buffers. A frame buffer containing any packet will not be output as a Telemetry Transfer Frame until the whole packet has been received, even if the packet spills over into subsequent frame buffers. When a packet is aborted, no new data will be stored in the frame buffer until the beginning of the next new packet, and data already stored from the aborted packet will be overwritten.

There is however a limitation on how large packets can be inserted in a given configuration of the SCTMTC ASIC, mainly dependent on the size of the external memory buffer. If all frame buffers allocated to a VC contain part of a large packet that is being received over SpaceWire, a dead lock situation will occur. The reason for the dead lock is that none of the buffer frames can be released for transmission to ground since the packet is still being received and has not been completed. When all allocated frame buffers are full, no more data can be received and the dead lock occurs. The way to resolve this dead lock situation is to abort the ongoing packet insertion via SpaceWire by disconnecting the SpaceWire link, see section 5.4.8.5.1.

This is also valid for non-packet data since always internally delimited when transferred over the parallel interface from the SpaceWire link, thus non-packet data can be aborted. This does not affect the generation of the dynamic or static First Header Pointer.

5.4.7.2.3 First Header Pointer

The First Header Pointer is required for the packet chaining process during the packet de-multiplexing on the ground. The VC Input Interfaces generate Transfer Frame data in one of two possible states; active or idle, as defined in [TM_STD]. The data contained in an active Transfer Frame can be either packets or non-packet data delimited into data blocks. Packets are referred to as synchronous data in [TM_STD] whereas non-packet data are referred to as asynchronous data.

In the active state, packets may be of varying lengths. It is unlikely that an integer number of packets will be exactly contained within a Transfer Frame, some Packets will thus be split between multiple Transfer Frames. The VC Input Interfaces calculate the position of the first byte of the first packet to be placed in each Transfer Frame, in accordance with [TM_STD], and then stores the First Header Pointer value in the external memory buffer together with the packet data.

For the VC Input Interfaces, the above also applies to non-packet data, allowing data blocks to be identified in a data stream. This is an extension to [TM_STD], permitting the First Header Pointer also to be dynamically calculated for non-packet data in the same way as it is performed for packets.

Released

5.4.7.2.4 Idle Packet Insertion

The VC Input Interface (A-D) can optionally generate and insert Idle Packets (see Table 5-15), to fill up an incomplete Transfer Frame. This ensures that user packets do not remain inaccessible due to an incomplete Transfer Frame being resident in the external memory buffer. This function is only available when the VC Input Interface is operating with packets, referred to as synchronous data in [TM_STD].

The Idle Packet insertion process starts when all of the following conditions are met:

- The Virtual Channel operates with packets (Data Field Synchronisation Flag is 0_2)
- The source is not sending data
- The Virtual Channel has no Transfer Frame Data Field completed and available for transmission
- The Virtual Channel contains an incomplete Transfer Frame Data Field (containing non-Idle Packets or part of a non-Idle Packet) that is resident in the external memory buffer
- The poll threshold is not set to 111_2
- The appropriate number of polls have been counted as described in section 6.7.2.1.

The Idle Packet comprises a Packet Header with a length of six octets and a Data Field with a fixed length of eight octets, giving a total Idle Packet length of 14 octets. The Idle Packets have been kept short, in the case a user packet should arrive during the Idle Packet insertion process.

The VC Input Interface generates the Idle Packet according to:

Field Contents	
Packet Identification	
Version Number MSB (bit 0)	Value of PV in the TmEnCfg0 register
Version Number LSBs (bit 1 to 2)	00_2
Type (bit 3)	0_2
Data Field Header Flag (bit 4)	0_2
Application Process Identifier (bit 5 to 15)	$111\_1111\_1111_2$
Packet Sequence Control	
Segmentation Flags (bit 0 to 1)	11_2
Source Sequence Count (bit 2 to 15)	$00\_0000\_0000\_0000_2$
Packet Length	
Packet Length (bit 0 to 15)	$0000\_0000\_0000\_0111_2$
Packet Data Field	
Packet Data Field (bit 0 to 63)	Pseudo random data

Table 5-15 Idle Packets

The VC Input Interfaces generates pseudo-random data for the Idle Packet Data Fields. The generator is free running, and will not be initialised by any event except a module reset, or an inadvertent all-zero state. The pseudo-random data generator polynomial is $x^9 + x^4 + 1$.

The Idle Packet insertion process stops after the last Idle Packet has been stored in the external memory buffer. In the case that the last Idle Packet did not fit in the Data Field, the remainder of the packet will be written to the Data Field of the next Transfer Frame. The time that is required for the VC Input Interface to complete a Data Field with Idle Packets depends on how much space there is left to be filled.

Should user packet insertion begin on a VC Input Interface after the Idle Packet insertion process has started, the current Idle Packet will be completed in parallel with the storing of user packet data, after which the Idle Packet insertion process will be suspended. As soon as the user packet insertion ends, the Idle Packet insertion process resumes and continues until the Data Field has been completed.

The VC Input Interfaces guarantees that the writing of an Idle Packets in the current Data Field will be completed before the following Data Field has been filled with user packets. Each VC Input Interface is able to receive Telemetry Packets and store data in the external memory buffer simultaneously with the insertion of an Idle Packet.

When a Transfer Frame contains no user packets or when it contains only part of an Idle Packet, the Idle Packet insertion process will not start. The situation where a Data Field contains only part of an Idle Packet occurs when an Idle Packet spilled over from the previous Transfer Frame and it is the only data in the current Transfer Frame.

5.4.7.3 Frame Generator

The data and the First Header Pointers stored in the external memory buffer by the VC Input Interfaces are used by the frame generator for generating Telemetry Transfer Frames according to [TM_STD]. The Frame Generator also generates an Attached Synchronisation Marker and optionally leaves space for insertion of Reed-Solomon check symbols or for trellis termination for the Turbo encoder. The Transfer Frame and the Reed-Solomon codeword form a Reed-Solomon codeblock. Alternatively, the Transfer Frame is turbo encoded into a turbo codeblock. An overview of these structures can be seen in Figure 4-18 and Figure 4-19.

The Frame Generator performs the following functions:

- Generate the Attached Synchronisation Marker according to [TM_STD].
- Generate Packet Telemetry transfer frames for the selected VC as follows:
  - Generate the Primary Header, including reading the First Header Pointer from the external memory buffer.
  - Generate the Secondary Header, if enabled.
  - Read input data stored in the external memory buffer, and insert it into the Transfer Frame Data Field. The TME does not take errors on the internal bus into account during data fetches.
  - Insert the CLCW provided by the CLCW Interface block into the Operational Control Field (OPCF), if enabled.
  - Generate the Cyclic Redundancy Code (CRC) and insert it into the Frame Error Control Word (FECW) field, if enabled.
  - Leave a space for the Reed-Solomon check symbols, if the Reed-Solomon Encoder is enabled.
- Generate an Idle Transfer Frame from VC[H] when there is not sufficient data available to generate a normal Transfer Frame.
- Maintain and update the Master and VC Frame Counters.
- Generate the *TmeTimeStrb* output to allow the on-board time to be sampled synchronously with the generated transfer frames.

5.4.7.3.1 Attached Synchronisation Marker

The synchronisation marker delimits the boundaries of a Transfer Frame. If the frame is neither Reed-Solomon nor Turbo encoded, it is used by the ground equipment to acquire synchronisation with the frame boundaries. If the frame is Reed-Solomon or Turbo encoded, the marker serves to synchronise the codeblocks. All Reed-Solomon or Turbo codeblocks and Transfer Frames have an attached synchronisation marker. The attached synchronisation marker pattern for Transfer Frames and Reed-Solomon codeblocks is according to the following tables:

Mode	Attached Synchronisation Marker
Normal	$1ACF\_FC1D_{16}$
Alternative	$352E\_F853_{16}$

Table 5-16 Attached Synchronisation Marker for Transfer Frames and Reed-Solomon Codeblocks

Turbo codeblocks have a variable attached synchronisation marker that depends on the code rate as shown in Table 5-17.

Code Rate	Attached Synchronisation Marker
1/2	$0347\_76C7\_2728\_95B0_{16}$
1/3	$25D5\_C0CE\_8990\_F6C9\_461B\_F79C_{16}$
1/4	$0347\_76C7\_2728\_95B0\_FCB8\_8938\_D8D7\_6A4F_{16}$
1/6	$25D5\_C0CE\_8990\_F6C9\_461B\_F79C\_$ $DA2A\_3F31\_766F\_0936\_B9E4\_0863_{16}$

Table 5-17 Attached Synchronisation Marker for Turbo Codeblocks

5.4.7.3.2 Telemetry Transfer Frame

The generated Transfer Frame consists of two required and three optional parts, for which the structure can be seen in Figure 4-13. The required parts are the Primary Header and the Data Field and the optional blocks are the Secondary Header, the Operational Control Field (OPCF) and the Frame Error Control Word (FECW).

5.4.7.3.2.1 Transfer Frame Primary Header

Each Transfer Frame is required to have a Primary Header, for which the structure can be seen in Figure 4-14. The first two octets make up the Frame Identification, which consists of the following fields:

- Version number, always 00_2.
- Spacecraft Identification (SCID)
- Virtual Channel Identification, is the number of the Virtual Channel currently being output. The Virtual Channel coding is shown in Table 5-18.

Virtual Channel ID	Virtual Channel
000_2	VC0
001_2	VC1
010_2	VC2
011_2	VC3
100_2	VC4
101_2	VC5
110_2	VC6
111_2	VC7

Table 5-18 Virtual Channel Identification coding

- Operational Control Field Flag, indicates whether the Operational Control Field is present in the Trailer or not.

The next two octets contain the Master and Virtual Channel Transfer Frame counters:
- Master Channel Frame Count, provides an 8-bit sequential count of each Transfer Frame generated.
- Virtual Channel Frame Count, provides an 8-bit sequential count of the Transfer Frames generated by each of the separate Virtual Channels.

They are used by the ground equipment to verify that the Transfer Frames have been received and in the correct order.

The last two octets of the Primary Header make up the Transfer Frame Data Field Status. It provides information to enable packets to be extracted from the Data Field.
- Secondary Header Flag, indicates whether a secondary header is present or not.
- Synchronisation Flag, indicates whether the Frame Data Field comprises synchronously inserted standard packets (Synchronisation Flag = 0) or not.
- Packet Order Flag
- Segment Length Identifier
- First Header Pointer (FHP), indicates the position of the first packet that begins in the Transfer Frame Data Field.

5.4.7.3.2.2 Transfer Frame Secondary Header

The secondary header is optional. An overview of its structure is shown in Figure 4-15. It contains the following fields:
- Secondary Header Version Number, always 00_2.
- Secondary Header Length, always 00_0011_2.
- Secondary Header Data, provides 24 additional bits of the Virtual Channel Frame Count.

The Frame Generator assigns the eight LSBs of the Virtual Channel Frame Counter to the Virtual Channel Frame Count field in the Primary Header and assigns the 24 MSBs of the Virtual Channel Frame Counter to the Secondary Header Data Field as shown in Figure 5-21.

Virtual Channel Frame Counter							
MSB	32-bit Virtual Channel Frame Counter						LSB
0			23	24			31
MSB	Secondary Header Data		LSB	MSB	VC Frame Count		LSB
0			23	0			7

Figure 5-21 Virtual Channel Frame Counter structure

5.4.7.3.2.3 Transfer Frame Data Field

Data that has been written to the external memory by the VC Input Interfaces is inserted in the data field of the Transfer Frames. If not enough data has been received to make up a complete Transfer Frame the complete Data Field is filled with pseudo-random data as explained in 5.4.7.3.4.

5.4.7.3.2.4 Transfer Frame Trailer

The Transfer Frame Trailer contains none, either or both of the following fields. Its structure is shown in Figure 4-4.

OPCF
The Operational Control Field (OPCF) is optional. If used, the 32-bit Command Link Control Word (CLCW), provided by the CLCW interface, is inserted into the OPCF of each Transfer Frame generated. See chapter 6.7.2.3 for details on the CLCW interface.

FECW
The two byte Frame Error Control Word (FECW) are optional. It consists of a Cyclic Redundancy Code (CRC) checksum over the whole Transfer Frame, excluding the FECW and the Synchronisation Marker, using the generator polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$. The shift register is initialised to 'all ones' before each frame.

5.4.7.3.3 Coding Space

The Frame Generator can insert blank space between the end of the Telemetry Transfer Frame and the beginning of the subsequent Attached Synchronisation Marker to accommodate Reed-Solomon and Turbo coding. The Telemetry Channel Coding Standard, [TMCOD_STD], specifies a Reed-Solomon code block with 8 bits per symbol and 255 symbols per codeword, the first 223/239 symbols containing the information symbols and the last 32/16 symbols making up the check symbols. In order to comply with this standard, Transfer Frames and Reed-Solomon codewords therefore have a length of 223*I and 32*I or 239*I and 16*I symbols respectively, where I is the interleaving depth. The interleaving depths supported by this telemetry encoder are 1, 2, 4 and 5 resulting in the Transfer Frame lengths as specified by [CCSDS_TMCOD] and Reed-Solomon codewords of lengths 32/16, 64/32, 128/64 and 160/80 respectively.

When Reed-Solomon encoding is enabled, no telemetry data is output while the Reed-Solomon codes are being generated. This is shown in the Figure 5-22 and Figure 5-23 below. When the Turbo encoding is enabled a similar logical zero gap with the length of four bits is inserted between frames.

Transfer Frame	Sync. Marker	Transfer Frame	Sync. Marker	Transfer Frame
	4 bytes	223/446/892/1115 bytes 239/478/956/1195 bytes		

Figure 5-22 TmeUnEncOut output bit stream, no Reed-Solomon

Logical zero	Sync. Marker	Transfer Frame	Logical zero	Sync. Marker
	4 bytes	223/446/892/1115 bytes 239/478/956/1195 bytes	32/64/128/160 bytes 16/32/64/80 bytes	

Figure 5-23 TmeUnEncOut output bit stream, with Reed-Solomon

5.4.7.3.4 Idle Transfer Frames

After reset Transfer Frames are continuously output from the Frame Generator. Whenever there is not enough data in the external buffer memory to build a complete Transfer Frame or if a VC Input Interface with complete Transfer Frames is not included in the BAT, Idle Transfer Frames have to be generated. For example, this is the case immediately after reset when not enough data will have been received by the VC Input Interfaces.

An Idle Transfer Frame is treated as a normal frame but is distinguished by its First Header Pointer value, which is $111\_1111\_1110_2$. The Virtual Channel ID for an Idle Transfer Frame can be programmed, but is always equal to the Virtual Channel ID of VC H. The Frame Generator will generate the correct flags and counter values, and provide a fill pattern for the Data Field. The fill pattern is generated by a pseudo-randomiser with the generator polynomial $x^9 + x^4 + 1$. The randomiser is only active when the Data Field of an Idle Telemetry Frame is being output and it is systematically preset when VC Frame Counter value is zero. This ensures that the pseudo-random pattern generation is deterministic.

Released

5.4.7.3.5 Time Strobe

The Frame Generator provides the active high *TmeTimeStrb* for sampling the on-board time according to section 7.4 of [TM_STD]. *TmeTimeStrb* is asserted on the rising edge of the *TmeUnEncSync* signal when the Synchronisation Marker is started to be output. The *TmeTimeStrb* is asserted for 128 bit clock periods. *TmeTimeStrb* from the Frame Generator associated with Virtual Channel 0 is intended to be used for sampling the on-board spacecraft time. The periodicity is programmable.

5.4.7.4 CLCW Interface

The 32-bit Command Link Control Word (CLCW), as specified in [TC_STD], is retrieved from a packet telecommand decoder using a synchronous bit serial interface, as shown in Figure 7-18. The CLCW is retrieved once per transmitted frame. The CLCW interface provides a mechanism for multiplexing the CLCW from up to four telecommand decoders.

As a programmable option, it is possible to overwrite the CLCW information received from the telecommand decoders. Bit 16 and 17 of the CLCW, No RF Available and No Bit Lock respectively, are then directly derived from the *PdecRfAvN** and *PdecTcAct** inputs. This provides compatibility with older implementations of telemetry encoders.

5.4.7.5 VC Selection with Bandwidth Allocation Table (BAT)

The Frame Generator features two built-in algorithms for selecting which VC is to be output in a Transfer Frame: Bandwidth Allocation and Priority Selection. Examples of the different selection algorithms can be found in the sections below.

5.4.7.5.1 Bandwidth Allocation

In this mode the VC Selection uses adaptive frame ordering. The Bandwidth Allocation Table (BAT) contains 32 internal 3-bit registers of which the value (0 to 7) indicates the VC[A...H] to be considered.

While outputting a Transfer Frame the Frame Generator scans the BAT for the next VC to be output in a round-robin fashion. The algorithm selects the next entry in the BAT which points to a VC Input Interface that has received enough data for a Telemetry Transfer Frame to be created. If a VC does not have an entry in the BAT that points to it, no Transfer Frames from that VC will be output, even if no other Transfer Frames are available.

The round-robin scanning continues until a VC with sufficient data has been found, or all entries in the BAT have been examined. If none of the VC Input Interfaces pointed to by the BAT entries has enough data, the Frame Generator will start generating an Idle Transfer Frame.

The VC selection starts when approximately 60 - 80 % of the previous frame has been output to allow the algorithm to complete before the next frame. The exact figures are: 1024 bits after the first bit of the Attached Synchronisation Marker for a frame length of 223/239 bytes; 2048 bits for 446/478 bytes; 6144 bits for 892/956 bytes; and 8192 bits for 1115/1195 bytes. Since the selection occurs while the previous Transfer Frame is being output, the first Virtual Channel to be output after reset has been chosen to be VC[H].

For each Transfer Frame, all entries in the BAT are scanned once, starting and ending on the same entry. Consequently a VC may receive between 0 and 33 polls for each Transfer Frame being output, depending on the BAT programming.

5.4.7.5.2 Priority Selection

In this mode the BAT pointer is reset to restart at the first entry in the BAT (address 0) at the beginning of each transfer frame. Since this occurs after each selection (Transfer Frame), the result will effectively be a priority VC selection. Except for the repeated resetting the selection process works exactly as in the case of Bandwidth Allocation.

5.4.7.6 Reed Solomon Encoder

The Reed-Solomon Encoder generates codewords according to the ESA Telemetry Channel Coding Standard [TMCOD_STD].

The encoder generates Reed-Solomon codewords by receiving information symbols, which are transmitted unmodified, while calculating the corresponding check symbols, which in turn are transmitted after the information symbols. The check symbol calculation is disabled during reception and transmission of symbols not related to the encoding, such as for synchronisation markers. The check symbol calculation is independent of any previous codewords and it is possible to begin immediately at the reception of the first information symbol after power-up.

The Reed-Solomon polynomials used are:

$$g_{esa}(x) = \prod_{i=112}^{143}(x+\alpha^i) = \sum_{j=0}^{32} G_j x^j$$

for transfer frame lengths being a multiple of 223, and

$$g_{esa}(x) = \prod_{i=120}^{135}(x+\alpha^i) = \sum_{j=0}^{16} G_j x^j$$

for transfer frame lengths being a multiple of 239.
α is a root of the irreducible field polynomial:
$f_{esa}(x) = x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$

The input and output of the Reed-Solomon encoder is represented in dual basis, and the coefficients of the generator polynomial are represented in conventional basis. The relationship between the two bases is:

$$T_{\alpha\ell} = \begin{bmatrix} 0&0&1&1&0&1&1&1 \\ 0&1&0&1&1&1&1&1 \\ 1&0&0&0&0&1&1&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&1&1&1&1&1&1 \\ 0&0&1&0&1&0&1&1 \\ 0&1&1&1&1&0&0&1 \\ 0&1&1&1&1&0&1&1 \end{bmatrix} \quad T_{\alpha\ell}^{-1} = \begin{bmatrix} 1&1&1&0&1&1&0&1 \\ 0&1&0&1&1&1&1&1 \\ 0&0&0&1&0&1&1&1 \\ 0&1&0&1&1&0&1&0 \\ 1&0&0&0&1&0&0&0 \\ 0&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1 \\ 1&0&0&1&1&0&0&0 \end{bmatrix}$$

where $T_{\alpha\ell}$ represents a transformation from the conventional to the dual basis.

The Reed-Solomon Encoder operates in three principal modes:

- Receiving mode, receiving and transmitting information symbols while calculating check symbols.
- Transmitting mode, transmitting check symbols.
- Bypass mode, receiving and transmitting symbols while clearing the check symbol memory.

No explicit initialisation of the Reed-Solomon Encoder is required before correct operation. Codeword generation begins immediately at the reception of the first information symbol after power-up. Also, unmodified Attached Synchronisation Markers are transmitted after power-up.

The internal control logic is reset and resynchronised at the start and the end of each new Transfer Frame transmitted, confining any bit error due to a Single Event Upset to only a single codeword.

5.4.7.7 Pseudo-randomiser

In order to maintain bit synchronisation with the received telemetry signal, every ground data capture system requires that the incoming signal have a minimum bit transition density, see [MOD_STD]. For Reed-Solomon and Turbo codeblocks or Transfer Frames, which are not convolutionally encoded, transition density is not ensured and therefore requires a method to ensure sufficient transitions.

The method used is to exclusive-OR each bit of the Codeblock or Transfer Frame with a standard pseudo-random sequence. The first bit of Codeblock or Transfer Frame is exclusively-ORed with the first bit of the pseudo-random sequence, followed by the second bit of the Codeblock or Transfer Frame with the second bit of the pseudo-random sequence, and so on. The Attached Synchronisation Marker is not pseudo-randomised.

The pseudo-random sequence is generated using a LFSR (Linear Feedback Shift Register). A LFSR is a sequential shift register with combinatorial feedback logic around it that causes it to pseudo-randomly cycle through a sequence of binary values. The following generator polynomial is used:

$$h(x) = x^{8+} x^7 + x^5 + x^3 + 1$$

This sequence begins at the first bit of the Codeblock or Transfer Frame and repeats after 255 bits, continuing repeatedly until the end of the Codeblock or Transfer Frame. The sequence generator is re-initialised to an all-ones state during each synchronisation marker period.

The first 40 bits of the pseudo-random sequence from the generator are shown below; the left-most bit is the first bit of the sequence to be exclusive-ORed with the first bit of the Codeblock or Transfer Frame; the second bit of the sequence is exclusive-ORed with the second bit of the Codeblock or Transfer Frame, and so on.

$$1111\_1111\_0100\_1000\_0000\_1110\_1100\_0000\_1001\_1010_2...$$

The pseudo-random sequence generator is implemented as shown in the logic diagram in Figure 5-24.

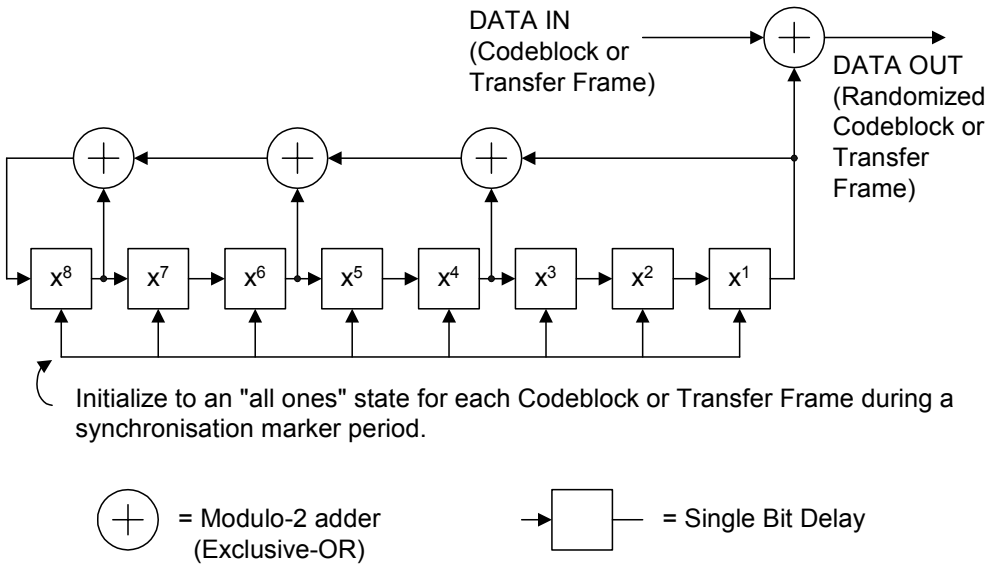


Figure 5-24 Pseudo-Randomiser Logic Diagram

5.4.7.8 Convolutional Encoder

The Convolutional Encoder implements two convolutional encoding schemes. The ESA PSS standard [TMCOD_STD] specifies a basic convolutional code without puncturing. This basic convolutional code is also specified in the CCSDS recommendation [CCSDS_TMCOD], which in addition specifies a punctured convolutional code.

The basic convolutional code has a code rate of 1/2, a constraint length of 7, and the connection vectors G1 = 111_1001 and G2 = 101_1011 with symbol inversion on output path, where G1 is associated with the first symbol output.

The punctured convolutional code has a code rate of 1/2 which is punctured to 2/3, 3/4, 5/6 or 7/8, a constraint length of 7, and the connection vectors G1 = 111_1001 and G2 = 101_1011 without any symbol inversion. The encoder also supports rate 1/2 unpunctured coding with aforementioned connection vectors and no symbol inversion.

Convolutional encoding can be used in conjunction with Reed Solomon encoding. Used this way, the convolutional code is the inner code, while the Reed-Solomon code is the outer code. The encoder is implemented according to the logic diagram shown in Figure 5-25.

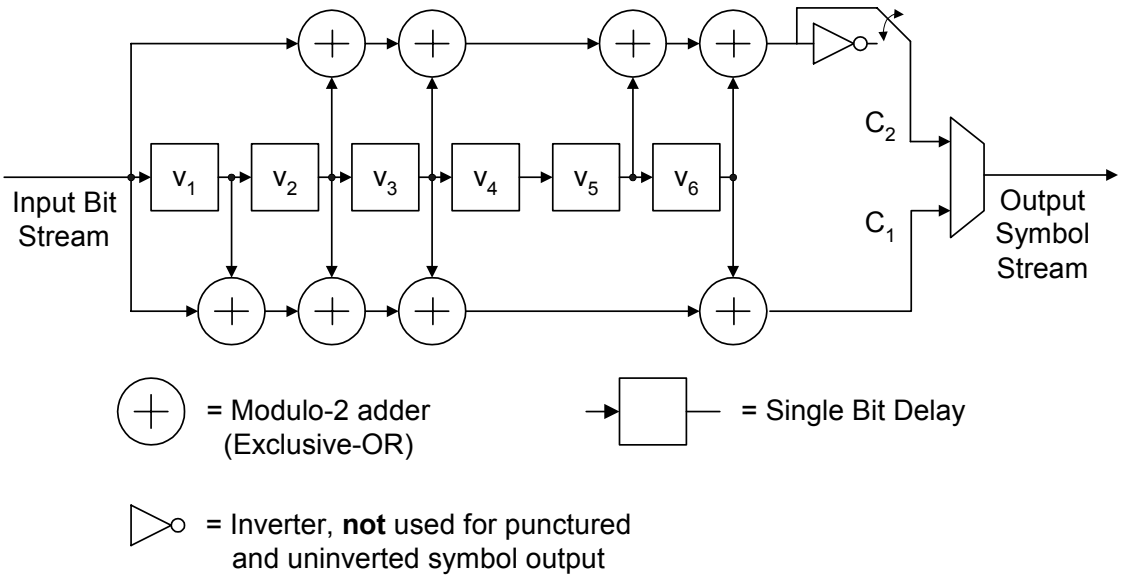


Figure 5-25 Convolutional Encoder implementation

Puncturing pattern *1 = transmitted symbol* *0 = non-transmitted symbol*	**Code rate**	**Output sequence** C_1(t*)*, C_2(t*) denote values at bit time* t
C_1: 1 0 C_2: 1 1	2/3	$C_1(1)$ $C_2(1)$ $C_2(2)$...
C_1: 1 0 1 C_2: 1 1 0	3/4	$C_1(1)$ $C_2(1)$ $C_2(2)$ $C_1(3)$...
C_1: 1 0 1 0 1 C_2: 1 1 0 1 0	5/6	$C_1(1)$ $C_2(1)$ $C_2(2)$ $C_1(3)$ $C_2(4)$ $C_1(5)$...
C_1: 1 0 0 0 1 0 1 C_2: 1 1 1 1 0 1 0	7/8	$C_1(1)$ $C_2(1)$ $C_2(2)$ $C_2(3)$ $C_2(4)$ $C_1(5)$ $C_2(6)$ $C_1(7)$...

Table 5-19 Punctured Convolutional Output Sequences

Released

5.4.7.9 Turbo Encoder

The Turbo Encoder encodes data according to [CCSDS_TMCOD]. A turbo encoder is a combination of two simple encoders. The input is a frame of *k* information bits. The two component encoders generate parity symbols from two simple recursive convolutional codes, each with a small number of states. The information bits are also sent uncoded.

Note: The Turbo Encoder does not support frame lengths of 239/478/956 and 1195 octets.

The Turbo Encoder uses the external buffer memory to temporarily store the information block during the encoding procedure. The external buffer memory is organised in two areas, the first is used for the code information block that is being received and the second is used for the information block that is being coded and transmitted. The interpretation of the two memory areas is shifted between frames to avoid data copying. The encoder both writes and reads data over the internal bus. The word format is always eight bits and 4096 words are required, 2048 for each of the two areas.

No output is generated by the encoder while the first information block is received, and the corresponding frame and attached synchronisation marker delimiters are not asserted during this time period. Each information block will be delayed by the time corresponding to the transmission of one information block. This will cause a corresponding static timing offset for the time strobe generated by the Frame Generator.

5.4.7.9.1 Block diagram

The encoder block diagram is shown in Figure 5-26. Each input frame of *k* information bits is held in a frame buffer, and the bits in the buffer are read out in two different orders for the two component encoders. The first component encoder (a) operates on the bits in unpermuted order (“in a”), while the second component encoder (b) receives the same bits permuted by the interleaver (“in b”). The read-out addressing for “in a” is a simple counter, while the addressing for “in b” is specified by the turbo code permutation described below.

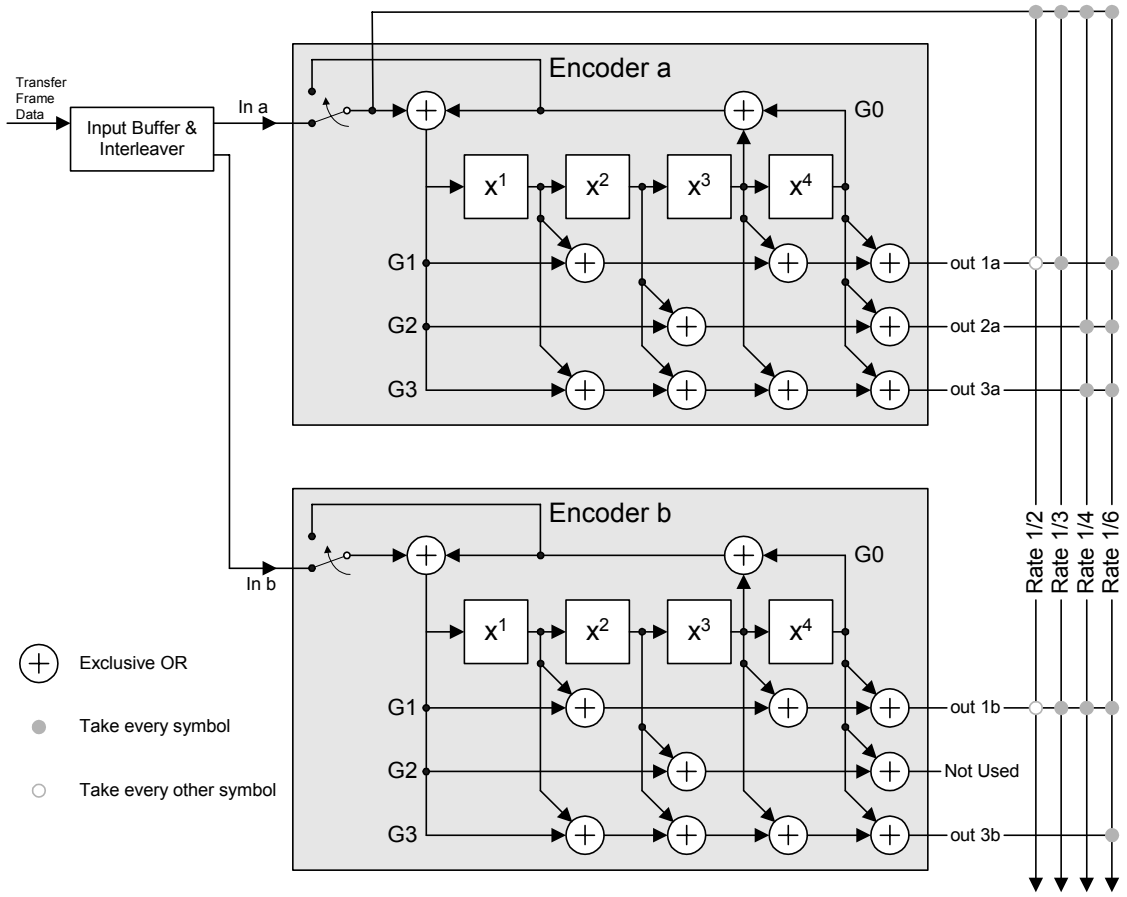


Figure 5-26 Turbo Encoder Block Diagram

5.4.7.9.2 Permutation

The interleaver for turbo codes is a fixed bit-by-bit permutation of the entire block of data. Unlike the symbol-by-symbol rectangular interleaver used with Reed-Solomon codes, the turbo code permutation scrambles individual bits and resembles a randomly selected permutation in its lack of apparent orderliness.

Released

The permutation for each specified block length k is given by a particular reordering of the integers 1, 2, . . ., k as generated by the following algorithm.

$$\pi(s) = 2(t + k_1 / 2 * c + 1) - m$$
$$c = (p_q j + 21m) \bmod k_2$$
$$q = t \bmod 8 + 1$$
$$t = (19i + 1) \bmod 4$$
$$j = \left\lfloor \frac{s-1}{2} \right\rfloor - ik_2$$
$$i = \left\lfloor \frac{s-1}{2k_2} \right\rfloor$$
$$m = (s-1) \bmod 2$$

where: k and k_2 are the transfer frame lengths in bits and octets respectively. They are related to the interleave depth as shown hereafter.

Interleave depth, I	**Transfer Frame length**		
	k, bit length	k_1, bits/octet	k_2, octet length
1	1784	8	223
2	3568	8	446
4	7136	8	892
5	8920	8	1115

Table 5-20 Parameters k, k_1 and k_2 for Specified Interleave Depths

Released

- p_q denotes one of the following prime integers: $p_1 = 31$; $p_2 = 37$; $p_3 = 43$; $p_4 = 47$
- $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

The interpretation of the permutation numbers is such that the sth bit read out on line "in b" in Figure 5-26 is the $\pi(s)$th bit of the input information block

5.4.7.9.3 Specification

Both component encoders in Figure 5-26 are initialised with 0s in all registers, and both are run for a total of k+4 bit times, producing an output Codeblock of (k+4)/r encoded symbols, where r is the nominal code rate. The codeblock lengths with respect to the code rates and the information block length are shown in Table 5-15.

Information block bit length *k*	**Codeblock bit length *n***			
	rate 1/2	rate 1/3	rate 1/4	rate 1/6
1784 (=223 * 1 octet)	3576	5364	7152	10728
3568 (=223 * 2 octets)	7144	10716	14288	21432
7136 (=223 * 4 octets)	14280	21420	28560	42840
8920 (=223 * 5 octets)	17848	26772	35696	53544

Table 5-21 Codeblock Lengths for Turbo Codes

For the first *k* bit times, the input switches are in the lower position (as indicated in the figure) to receive input data. For the final 4 bit times, these switches move to the upper position to receive feedback from the shift registers. This feedback cancels the same feedback sent (unswitched) to the leftmost adder and causes all four registers to become filled with zeros after the final 4 bit times. Filling the registers with zeros is called terminating the trellis. During trellis termination the encoder continues to output nonzero encoded symbols. In particular, the "systematic uncoded" output (line "out 0a" in the figure) includes an extra 4 bits from the feedback line in addition to the *k* information bits.

Backward and forward connection vectors for the Turbo Encoder:

- Backward connection vector for both component codes and all code rates:
  G0 = 10011
- Forward connection vector for both component codes and rates 1/2 and 1/3:
  G1 = 11011
  - Puncturing of every other symbol from each component code is necessary for rate 1/2.
  - No puncturing is done for rate 1/3
- Forward connection vectors for rate 1/4:
  G2 = 10101, G3 = 11111 (1^{st} component code); G1 = 11011 (2^{nd} component code). No puncturing is done for rate 1/4
- Forward connection vectors for rate 1/6:
  G1 = 11011, G2 = 10101, G3 = 11111 (1^{st} component code); G1 = 11011, G3 = 11111 (2^{nd} component code). No puncturing is done for rate 1/6

In Figure 5-26, the encoded symbols are multiplexed from top-to-bottom along the output line for the selected code rate to form the Turbo Codeblock. For the rate 1/3 code, the output sequence is (out 0a, out 1a, out 1b); for rate 1/4, the sequence is (out 0a, out 2a, out 3a, out 1b); for rate 1/6, the sequence is (out 0a, out 1a, out 2a, out 3a, out 1b, out 3b). These sequences are repeated for (*k*+4) bit times.

For the rate 1/2 code, the output sequence is (out 0a, out 1a, out 0a, out 1b), repeated (*k*+4)/2 times. Note that this pattern implies that out 1b is the first to be punctured, out 1a is the second, and so forth. The Turbo Codeblocks constructed from these output sequences are depicted in Figure 5-27 for the four nominal code rates.

The Attached Synchronisation Marker (ASM) differs for the four code rates. The number of bits are proportional to the code rate, with the marker length being 32/r-bit for r = 1/2, 1/3, 1/4 and 1/6, e.g. r = 1/6 gives a sequence of 192 bits. As for Reed-Solomon coding, the ASM is used for frame synchronisation. (See section 5.4.7.3.1)

The Turbo encoder output is non-return-to-zero level encoded. The encoder increases the output bit rate with a factor of two to six, depending on the selected coding rate. It is not allowed to simultaneously engage both Turbo and Reed-Solomon and/or Convolutional encoding. The Turbo encoding introduces a latency corresponding to one Transfer Frame transmission duration, since a complete Transfer Frame needs to be available and buffered before encoding can commence. The encoder is slaved to the telemetry Transfer Frame generation process.

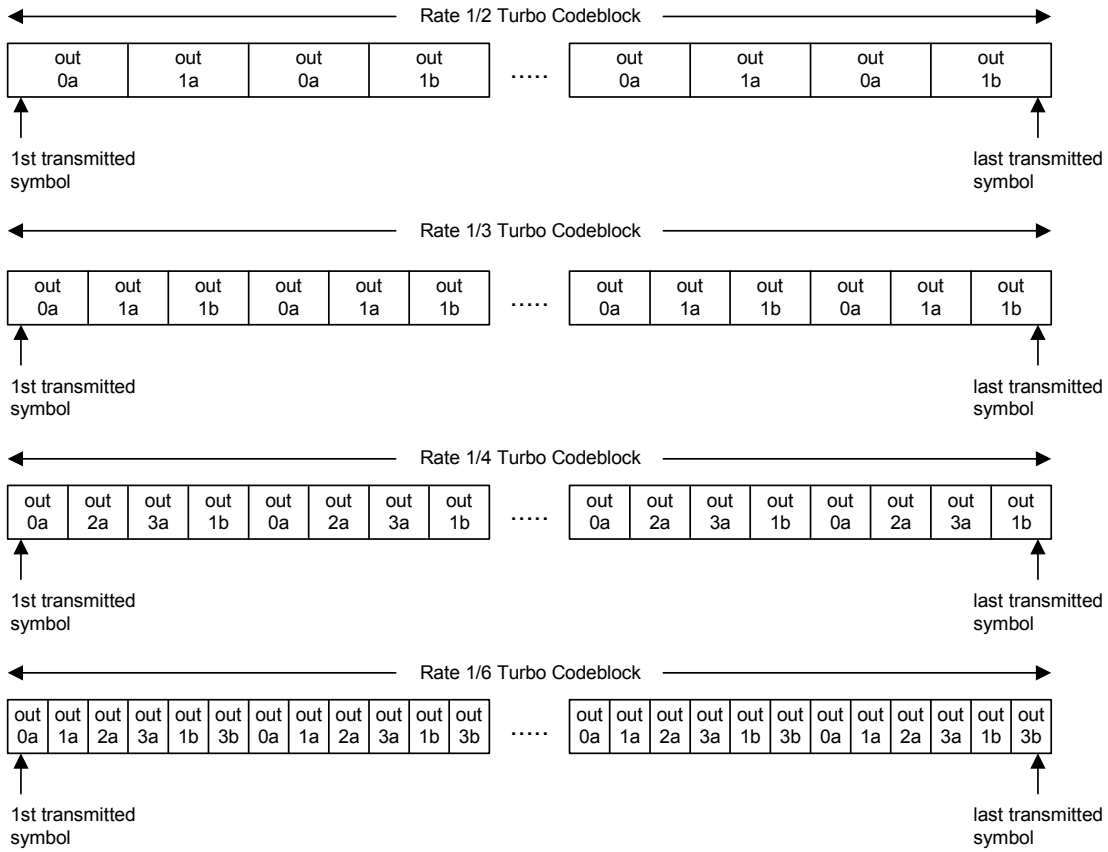


Figure 5-27 Turbo Codeblocks for Different Code Rates

Released

5.4.7.10 Test Pattern Generator

Instead of the normal Transfer Frame output, a test bit pattern can be output from the TME. The test pattern generator will, when enabled, continuously generate one of four selectable bit patterns.

5.4.7.11 Modulation

Data output from the TME is in different modulation formats: PSK Square, NRZ-L, NRZ-M, SP-L and I/Q. The TME has different outputs used for different modulations, however some of the modulations can be used in conjunction with each other. The order between the different modulators and the encoders can be seen in Figure 5-28. The output from the different modulators is always spike free.

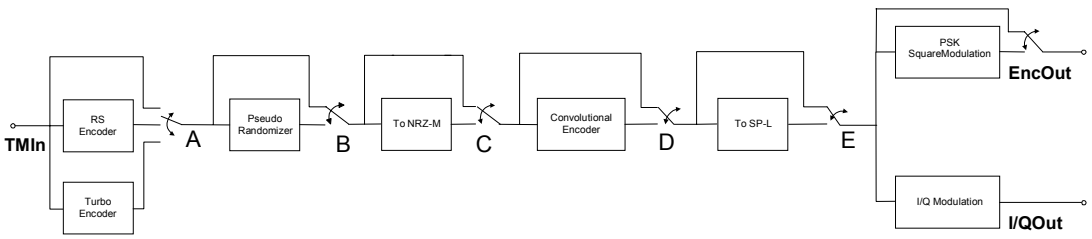


Figure 5-28 Order between encoders and modulation

The different modulations can be switched on and off individually, except for the I/Q modulation output which is always active.

5.4.7.11.1 NRZ, SP-L and I/Q modulation

The relationships between the different modulations are as depicted in Figure 5-29. In the picture all modulations are related to the NRZ-L format. Note that the I/Q modulation is designed so that even bits are output on the Q-channel and odd bits are output on the I-channel, i.e. the first bit of a frame is always output on the Q-channel.

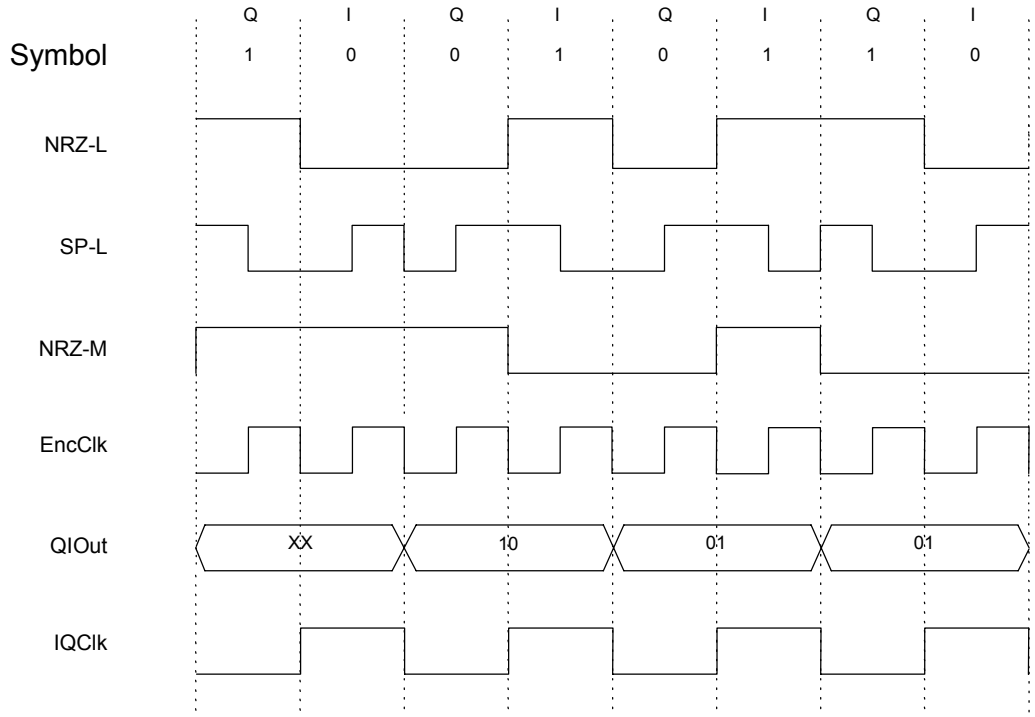


Figure 5-29 Modulation formats

5.4.7.11.2 PSK Square modulation

On the *TmeEncOut* data output, the bitstream can be optionally PSK Square modulated. It is possible to chose which phase of the SubCarrier (*SubCClk*), $0^o/180^o$, that shall correspond to a '1' or a '0'. The number of *SubCClk* periods used for each input bit can be programmed by changing the relationship between the *BitClk* and the *SubCClk*.

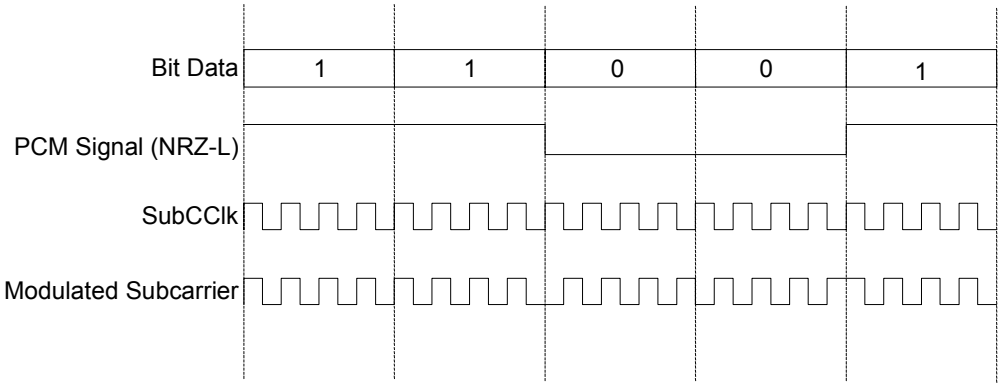


Figure 5-30 PSK Square modulation

5.4.7.12 Clock Divider

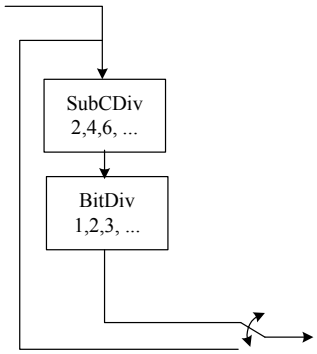


Figure 5-31 Clock Divider

The TME divides the incoming *TMClk** in steps to generate the *BitClk* and *SubCClk*. The clock divider can optionally be bypassed when no PSK modulation is used.

In order for the PSK Square modulation to function, the *SubCClk* rate shall be n * 4 * *BitClk* rate. The following frequencies shall be supported for each modulation. If SP-L is enabled , the output rates for all other modulations will be decreased by a factor 2.

The maximum Sub Carrier frequency and *BitClk* frequency are limited by the system clock frequency, depending on the modulation. The frequencies of the Sub Carrier and *BitClk* are calculated as follows:

$$f_{SubCClk} = f_{TMClk} / (SubCDivisor * 2)$$

$$f_{BitClk} = f_{SubCClk} / (BitRateDivisor + 1)$$

The user has to ensure that the frequencies comply with Table 5-22.

Modulation method	Max *BitClk* frequency	Max SubCarrier frequency	Max Output Data rate
Square PSK	*SysClk*/*8*	*SysClk*/2	*SysClk*/2
NRZ	*2*SysClk*	NA	*2*SysClk*
I/Q	2**SysClk*	NA	2**SysClk*
SP-L	*2*SysClk*	NA	*SysClk*

Table 5-22 Maximum *BitClk* and Sub Carrier frequency related to *SysClk*

5.4.8 SpaceWire Module (SPW)

The SpaceWire (SPW) module provides an interface for commanding the Control Interface (CI) module or to transmit CCSDS packets to the Packet Telemetry Encoder (TME). The module complies with the ECSS SpaceWire standard [ECSS_SPACE]. The module features two SpaceWire links that can be used one at a time to implement redundancy. The selection between the two links is done by means of an external selection signal.

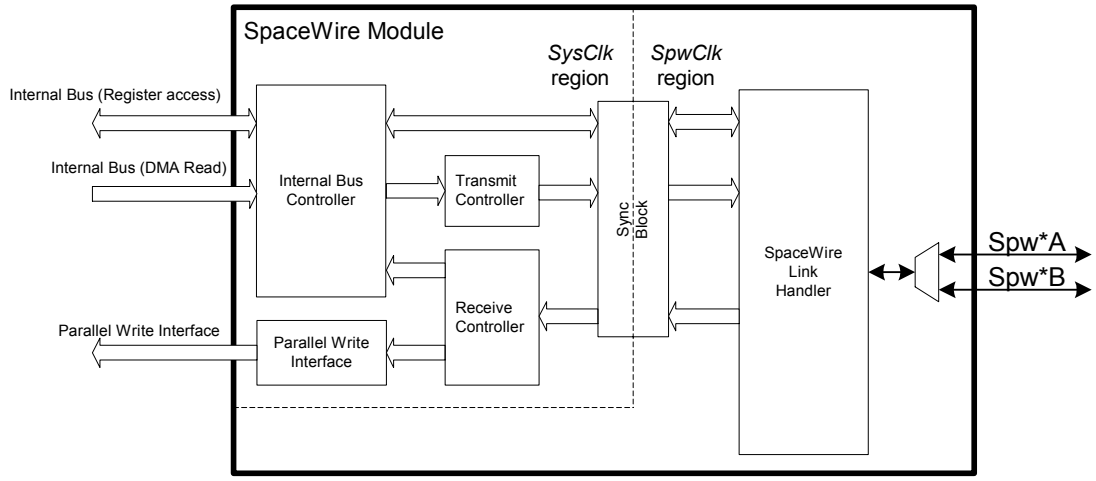


Figure 5-32 SpaceWire functional block diagram

Released

5.4.8.1 SpaceWire Link

When enabled the SpaceWire module starts connecting on the SpaceWire link. The start-up procedure of the SpaceWire link includes timeouts, which restarts the SpaceWire link if the other end does not respond within a certain time. This procedure makes the SpaceWire link self-synchronising. Two connected links must exchange at least one NULL and one FCT character to establish the communication path for data transmission.

Flow control is handled with the exchange of Flow Control Tokens (FCT). The reception of one FCT enables transmission of eight data characters. The sender keeps track of how many data characters the receiving end can receive, while the receiver keeps track of the number of FCTs sent. This prevents the sender from overflowing the receivers FIFOs.

The SpaceWire link is protected by a timeout on the data and strobe inputs. If neither of these has changed within 850 ns the link is considered disconnected and must be reconnected before any more data can be sent. To avoid disconnection the SPW continously transmits data on the link. If no data is avaiable for transmission a NULL token is sent.

5.4.8.1.1 Error Handling

If an error occurs on the SpaceWire link the SPW empties its transmit buffer and inserts an EEP end marker in its receive buffer. The SPW ensures that the receive buffer contains at least 8 empty slots, to enable the transmission of at least one FCT during the reconnection procedure, i.e. at least 9 slots are needed for the error handling. If the receive buffer can not provide 9 empty slots, the last received characters are deleted until an EEP and 8 empty slots can be inserted.

5.4.8.2 Data rates

In order to produce a data rate of 10 Mbit/s ± 10 %, which is required for a reliable start-up of the SpaceWire link, the SpaceWire protocol part of the module features a dedicated clock input *SpwClk* that is a multiple of 10 MHz. This also allows the interface to send data at higher transfer rates than the system clock frequency used by the part of the module that interfaces the internal bus, being clocked on the *SysClk* input. The *SpwClk* frequency defines the maximum reception rate on the SpaceWire link; the SpaceWire module can receive data or control characters at 4/3 times the *SpwClk* frequency.

The SpaceWire module provides two data rates: the default rate and the nominal rate. The two data rates are programmable as any integer fraction between 1 and 255 of the *SpwClk* frequency. The default rate should always be 10 Mbit/s ± 10 % and is used when establishing a connection bewteen the two SpaceWire modules on the SpaceWire link. It can also be used for transmitting all NULL tokens to reduce the transition density on the link when it is idle. The nominal rate is used for transmitting data, time codes and all control characters that does not use the default rate.

Due to the internal interface between the SPW module and the TME module, data can not be sent faster than one byte every second system clock cycle from the SPW to the TME, resulting in a maximum bit rate from the SpaceWire link to the TME of 4 times the system clock frequency.

5.4.8.3 Routing

In addition to implementing the SpaceWire protocol, the module includes a router service. The module can route incoming SPW packets to up to eight different destinations, or Virtual Receive Channels (VRC). The routing of the SPW packets is done by means of a destination address that is attached in front of the cargo in the SPW packet. When the SPW packet arrives at the SPW module interface, its destination address should only comprise one remaining destination identifier. This destination identifier is used by the SPW module for the final routing. The destination identifier is then removed and the cargo is received via the selected VRC. The routing of a SPW packet to a specific VRC ends after the reception of an end of packet marker, which can either be End Of Packet (EOP) or Error End of Packet (EEP) and is removed by the router. This approach is endorsed by the ECSS standard for SPW packet routing.

Seven of the VRCs are connected to the Virtual Channel (VC) inputs of the Packet Telemetry Encoder (TME) module. VRC[0:6] are connected to VC[A:G]. The eighth VRC, VRC[7], is connected to the Control Interface (CI) module.

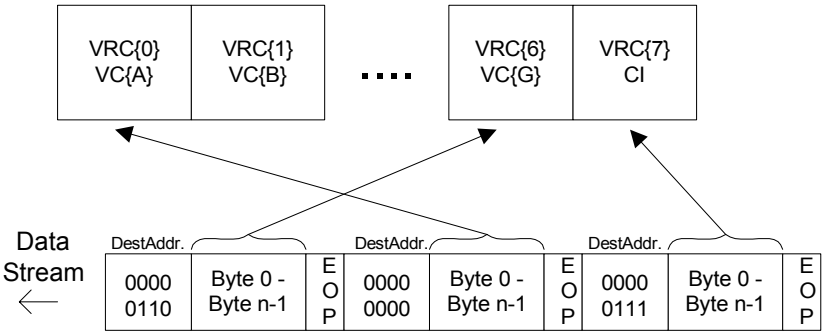


Figure 5-33 Receive Packet Routing

5.4.8.4 Application layer

The module also implements an application layer on top of the SpaceWire protocol and the routing described previously. Two application types are supported; CCSDS packet communication with the Packet Telemetry Encoder (TME), and command communication with the Control Interface (CI)

5.4.8.4.1 CCSDS Packets

The SPW module provides support for handling large CCSDS packets that can be segmented into smaller SPW packets. This can be used to reduce latency requirements over the SpaceWire link, allowing interleaving of SPW packets belonging to different CCSDS packets.

A CCSDS packet would be split in several SPW packets, for which the SPW SpaceWire packet would be followed by an additional empty SPW packet, including a destination identifier for routing purposes. This will lead to an internal custom End Of Message (EOM) token being generated that is communicated to the addressed VRC. In this way a telemetry VC input interface connected to a VRC can distinguish between different CCSDS packets. If splitting of CCSDS packets in several SPW packets is not enabled, each SPW packet will be considered a complete CCSDS packet.

The SPW module is also able to tell the telemetry VC input interface that an erroneous end of packet has occurred, i.e. an EEP marker was received, by aborting the complete CCSDS packet insertion. The CCSDS packet abort function of the telemetry VC input interface is described more in detail in 5.4.7.2.2. The effect will be that a corrupted CCSDS packet is retracted from the telemetry without violating the packet telemetry protocol.

5.4.8.4.2 Control Interface

The SPW module interfaces the Control Interface (CI) by means of a direct inter-module connection for sending commands and data. It also communicates with the internal bus by means of a direct memory access channel, that is controlled by the CI, which is used for generating a response to a command by transmitting data fetched from the internal bus address space.

The CI module controls the SPW module via a register interface that is accessible over the internal bus. This interface is also used by the Configuration block to configure the SPW module during initialisation after a power-up reset.

The format and protocol used for interfacing the CI module is described in 5.4.9. The router removes the destination identifier and the end of packet marker. Packets ending with an error are propagated to the CI module, which will handle any error cases.

5.4.8.5 Congestion and resolution

Since there is only one physical SPW link, which services multiple VRCs, there is always the possibility of congestion should one of the VRCs not be able to receive data. The reason for the congestion could be that the buffer memory of a telemetry VC has been filled or similar. This would in principle stop any communication over the SPW link. The actual problem lies in the fact that receiver FIFO in the SpaceWire Protocol block will not be emptied by the addressed VRC since the corresponding destination is not able to read out any more data. The receiver FIFO will thus be filled and prevent any new data to be received over the SpaceWire link. If the same link were also used for controlling the SCTMTC ASIC, it would not be possible for the user to reset the system by means of commanding. To overcome this problem, the SPW module provides an error handling mechanism that will allow the transmitting end of the SPW link to remove the congestion in a graceful way.

5.4.8.5.1 Resolution

The congestion problem is solved by means of a SpaceWire error handling mechanism. The principle is to utilise the disconnect error in the ECSS SpaceWire standard to reset the SpaceWire link as well as the VRC channels in the internal router and application layers. The objective is to prevent any incomplete or corrupted CCSDS packet to propagate to the TME.

Assuming that congestion occurs due a telemetry VC input interface not being able to receive the data that is first to be read out from the SPW receiver FIFO. The receiver FIFO will eventually fill up and no more data can be transmitted over the SpaceWire link and a dead lock is a fact. The sender can at this stage cause a disconnect error in the receiver by stopping the generation of transitions on the data and clock lines on the SpaceWire link. The disconnection will generate an internal error event in the SPW receiver module, which will force the insertion of an EEP in the receive FIFO. The next step is to abort the CCSDS packet that is currently being input on any of the telemetry VC input interfaces to allow this interface to continue reading data from the FIFO. Only one channel is aborted, since the currently active channel is assumed to be the erroneous one. Normally transfers on all other channels are unaffected.

To prevent any remaining data of an aborted CCSDS packet to be sent incorrectly to the telemetry VCs, the aborted VRC will ignore all SpaceWire packet data until it receives an EOM or EEP token. After the disconnection occurs this token already resides in the receive FIFO, which means that the sender does not have to provide it. The sender is responsible for ensuring that no more data of the aborted CCSDS packet is sent, which may include clearing any send FIFOs.

If the sender has two packets pending both may be corrupt. The first packet is aborted as described above. This packet's EOM does however reside in the receive FIFO, which ends this packet. The SPW interprets the next byte after the EOM as a routing header and starts transmitting on the designated VRC, which will be aborted by the EEP inserted in the FIFO.

5.4.8.5.2 Handling

The above approach to congestion resolution works only with point-to-point links. For router based networks CCSDS packet splitting is not supported together with congestion resolution and the congestion can thus only affect one channel.

In a point-to-point scenario, all data transmitted are generated by a host that should also detect a congestion problem by means of timeout. The host should then disconnect the link, stop sending any CCSDS packets, re-initialise the VRCs by sending an EOM on the aborted, including channels that were idle at the time of disconnection, and then re-send CCSDS packets as necessary.

5.4.8.6 Erroneous First Header Pointer in Telemetry Frames

When aborting a packet from SpaceWire on a TME VCA there is a chance that the first header pointer of the next frame becomes erroneous. No data, except the aborted packet, is lost and the frame is correct in all other aspects. The probability of the erroneous first header pointer is very low, and decreases with larger packets. This error can occur if a packet is aborted during the first 40-50 bytes and is not a problem during normal operation.

5.4.9 Control Interface Module (CI)

The Control Interface module provides a means of accessing a system through a Packet Wire interface or through a SpaceWire interface connected to the Control Interface via an internal parallel interface.

The selection between the two interfaces is done by means of a configuration register in the Clock and Reset block (CAR). The configuration bit is only checked when both interfaces are idle to avoid corrupted packets.

When receiving a packet through the selected interface the Control Interface checks the header to determine the type of access and the destination. The two least significant bits of the Address and Size fields are ignored since the Control Interface only operates on 32-bit words.

Accesses to the Control Interface are big-endian. The most significant bit of the most significant byte is sent first.

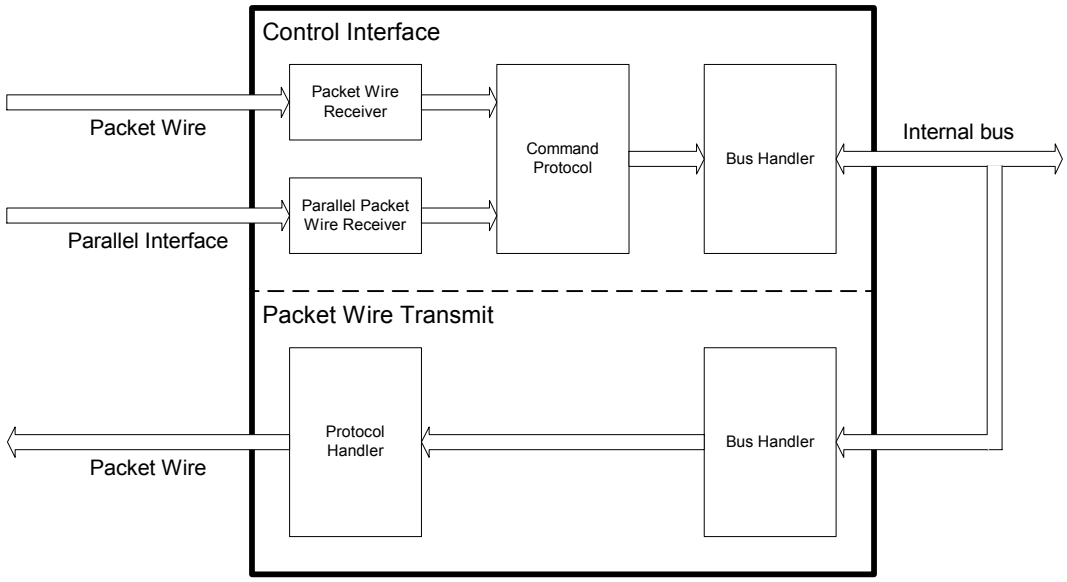


Figure 5-34 Control Interface module

5.4.9.1 Writing

When receiving a packet containing write data the Control Interface continues to receive the entire packet and write the data to memory. If data arrives faster than the Control Interface can write to memory, flow control is used to stop the input data stream. The Control Interface uses the Size field of the packet header to determine the amount of data to write to memory. If the packet contains more data than the Size field indicates the Control Interface continues to receive the remaining data but does not write any data to memory.

5.4.9.2 Reading

When a packet containing a read request is received the Control Interface determines the source of the packet. Data is read from memory via the internal bus and output on the corresponding interface, i.e. Packet Wire if the packet originated from the Packet Wire input interface and SpaceWire if the packet originated from the SpaceWire link.

5.4.9.3 Packet Wire interface

The Control Interface contains a single Packet Wire input and a single Packet Wire output. The interface is not redundant.

Packets sent to the Control Interface through the Packet Wire interface contain a dummy byte prior to the packet header. The dummy byte is ignored by the Control Interface and can be used for routing information at the sending end.

When receiving write data through the Packet Wire interface the Control Interface receives data until a complete 32-bit word has been received, which is written to memory via the internal bus. If the packet ends before a complete 32-bit word has been received the received data is discarded.

When data is read through the Packet Wire interface the Control Interface reads data through the internal bus on a 32-bit word-basis and output it on the Packet Wire output interface.

5.4.9.4 SpaceWire interface

The Control Interface is connected to a SpaceWire interface. The SpaceWire interface has a redundant input interface.

Packets sent to the Control Interface through the SpaceWire interface contain a routing header of one byte; this is needed since the SpaceWire link is used to transfer data to the other recipients as well as the Control Interface. The routing byte is removed before the packet is forwarded to the Control Interface.

When receiving write data through the SpaceWire interface the Control Interface receives data until a complete 32-bit word has been received, which is written to memory via the internal bus. If the packet ends before a complete 32-bit word has been received the received data is discarded.

When data is read through the SpaceWire interface the Control Interface reads data through the internal bus on a word-basis and output it on the SpaceWire interface.

5.4.9.5 Access errors

The SCTMTC ASIC implements a full address decoding to avoid incorrect accesses from affecting internal registers etc. If an unmapped address is accessed via the Control Interface, the error can be observed by means of status registers as described in sections 6.8 and 6.9 in detail. The overall effect is that for write accesses no immediate feed back is provided to the user. For read accesses, no data will be returned by the SCTMTC ASIC. When using the SpaceWire link for control a read access to an unmapped address results in a single EEP being transmitted, possibly preceded by one or more routing bytes.

5.4.9.6 Data reception on the unselected interface

The Control Interface allows data to arrive at the unselected interface by constantly signalling that it is ready for reception on the unselected interface. Data is consumed within the Control Interface without affecting the selected interface. This is vital for the SpaceWire module since it contains several virtual receive channels and is used for sending telemetry data to the TME as well as control packets; a packet erroneously sent to the inactive Control Interface can not be allowed to block telemetry. The Packet Wire interface is not as sensitive to this since it is a point-to-point link with no virtual channels; it is however treated as the parallel interface towards the SpaceWire module.

6 SOFTWARE INTERFACE

This section describes the SCTMTC ASIC software interface. The description is divided into subsections, which treat different parts of the SCTMTC ASIC. The first subsection describes general functions that do not sort under any of the ASIC functional modules and functions that are common to all modules (e.g. the interrupt handling). In the following subsections each functional module in the ASIC is described separately, and in the last subsection the ASIC registers are described.

Most module subsections are divided into:

- **Initialisation**
  Describes the initialisation sequence for the module (if applicable)
- **Operation/Usage**
  Describes how each module shall be used during normal operation
- **Error Handling**
  Describes how to handle internal errors in the module, e.g. Address error.
- **Usage Constraints**
  Describes actions that are not allowed and the resulting consequences.
- **Examples**
  Gives a few examples how to perform different tasks using the module.

6.1 General SCTMTC ASIC Functions

6.1.1 Internal Scan Controller block

NA

6.1.2 Test Access Port (TAP) block

NA

6.1.3 Clock and Reset (CAR) block

Even though it is possible to change the configuration of the SCTMTC ASIC using the registers inside the CAR block it cannot be recommended to do so during normal operation. The intention of the block is for the SCTMTC ASIC internal configuration block to be able to configure the SCTMTC ASIC during power up.

6.1.3.1 Operation/Usage

The changes of the configuration for a module; Clock source, Clock enabling, Module enabling and Module reset is made via registers. The configuration is protected by a Arm functionality, i.e. all changes in configuration has to be preceded by a Arm of the corresponding configuration change.

6.1.3.2 The ARM-configuration functionality

The configuration functionality of the modules inside SCTMTC made in the CAR block is such that all changes to the configuration made on one of the CAR * Register must be preceded by the setting of the corresponding bit in the CAR Arm * Register. After the write access to the CAR * Register the corresponding bit in the CAR Arm * Register will be cleared.

6.1.3.3 Disabling a module

For the disabling of a module from power-up see the Configuration Block. If one wants to disable a module after having it enabled. Then first reset the module using the:
CAR Arm Reset Register [CAR_ArmModReset] and the **CAR Reset Register [CAR_ModReset]**
then shut down the clock (if needed) by using the
CAR Arm Clock Enable Register [CAR_ArmClkEna] and the **CAR Clock Enable Register [CAR_ClkEna]**.

Note, a disabled module will have its outputs driven to their inactive state and all tri-state set in high impedance mode, i.e. tri-stated.

6.1.3.4 Changing clock source for SpaceWire and TME

For default clock source selection from power-up see the Configuration block. If one wants to change the clock source of either the TME or the SPW. Then

1. reset the module using the **CAR Arm Reset Register [CAR_ArmModReset]** and the **CAR Reset Register [CAR_ModReset]**
2. then shut down the clock by using the **CAR Arm Clock Enable Register [CAR_ArmClkEna]** and the **CAR Clock Enable Register [CAR_ClkEna]**
3. then change the clock configuration using the **CAR Arm Clock Source Register [CAR_ArmClkSrc]** and the **CAR Clock Source Register [CAR_ClkSrc]**
4. then enable the clock to the module again by using the **CAR Arm Clock Enable Register [CAR_ArmClkEna]** and the **CAR Clock Enable Register [CAR_ClkEna]**
5. then deassert the module reset by using the **CAR Arm Reset Register [CAR_ArmModReset]** and the **CAR Reset Register [CAR_ModReset].**

6.1.3.5 Selecting control interface

The SCTMTC ASIC provides two control interfaces: SpaceWire and Packet Wire, which share the common resource Control Interface and can not be active at the same time. Selection between SpaceWire and Packet Wire as source for the Control Interface is done through the CtrlIf Select Register. When CtrlIf Select Register.Sel is cleared Packet Wire is used and when it is set SpaceWire is used through Virtual Receive Channel 7.

6.1.3.6 Configure CSEL connections

Apart from the control interfaces the PDEC3 and External CPDU Interface can be connected to CSEL. The CSEL connections are configured by the Clock and Reset block. The TC input can be connected to either the PDEC3 or the External CPDU Interface and the RM input can be connected to the External CPDU Interface.

Connect PDEC3 to the CSEL TC input by clearing CAR Pdec3 Csel Connect Register.Pdec3Csel using CAR Arm Pdec3 Csel Connect Register and CAR Pdec3 Csel Connect Register. If the External CPDU Interface is disabled PDEC3 is permanently connected to CSEL. When the PDEC3 is connected to the TC input of the CSEL the External CPDU Interface is automatically connected to the RM input.

Connect External CPDU Interface to the CSEL TC input by setting CAR Pdec3 Csel Connect Register.Pdec3Csel using CAR Arm Pdec3 Csel Connect Register and CAR Pdec3 Csel Connect Register. The External CPDU Interface must be enabled.

6.1.3.7 Power-on reset

By using the CAR Power On Reset Register it is possible to find out if a power-on reset has occurred during operation. The register is reset to 0 during power-on reset. By writing 1 to it after initialisation it is possible to determine if the SCTMTC ASIC has lost power.

To be able to use the power-on reset register, do the following:
After power-on initialisation, write 1 to CAR Power On Reset Register

To check if there has been a loss of power read CAR Power On Reset Register. The register contains 0 if a power-on reset has occurred.

6.1.3.8 Error Handling

The intention of the Arm-Enable function as described above is to avoid unintentional change of the configuration. During normal operation this functionality shall not be used by any other function than the Configuration block.

6.1.3.9 Usage Constraints

As described in the configuration change section above there is a sequence of register writes that has to be fulfilled if the ASIC is to be functional after reconfiguration. E.g. if the clock configuration is changed without asserting the reset to the module first the module can not be expected to be operational after the configuration.

6.1.4 Configuration block

6.1.4.1 Initialisation

The block is self-initialised. The configuration block will also initialise all other modules and blocks at power on reset.

When not set by mission PROM the refresh counters should be initialised accordingly:

- Set the GenClk $ Assert Register to 0.
- Set the GenClk $ Deassert Register to 1.

6.1.4.2 Operation/Usage

It is possible to change the periodicity of the register refresh inside the Configuration block. The block includes two different refresh counters, one for system level and one for user level refresh. Where the control of each counter is made independently. Counter 1 is used for the system level refresh, and counter 2 is used for the user level refresh. Note that the user level refresh has higher priority than the system level refresh. This means that the duration of the user level refresh must be shorter than the time between two consecutive user refresh cycles.

It is good design practice to start the system level refresh as a part of the user level refresh to ensure that it is running. (The system level refresh, however, shall not start the user level refresh, since this is controlled by the user.)

6.1.4.2.1 Starting a refresh counter

In order to start one of the counters do the following step:

- Set the period for the refresh by writing to the GenClk $ Period Register.

Note that a period of two *SysClk* cycles equals to a value of 1 i.e. the value written to the GenClk $ Period Register shall be the desired period minus one.

- Set the Per$ field of the GenClk Interrupt Mask Register to enable a refresh with the set period.

- Set the DirectEn field of the GenClk$ Control Set Register.

Once this register has been written the DirectEn field of the GenClk$ Control Register will be set and the GenClk$ counter will start.

6.1.4.2.2 Shutting down a refresh counter

In order to stop the GenClk$ counter do the following step:

- Set the DirectEn field of the GenClk$ Control Clear Register.

Once this register has been written the DirectEn field of the GenClk$ Control Register will be cleared and the GenClk$ counter will stop.

6.1.4.2.3 Checking external interrupt source

To check which module has generated an external interrupt read Conf External Interrupt Register. This register is only a status register, which reflects the modules' interrupts. The bits in the register are cleared when the corresponding module's * Pending Interrupt Register or * Pending Interrupt Masked Register is read.

6.1.4.3 Usage constraints

The contents of the mission PROM are crucial to the Configuration Block. The PROM programming is described in section 6.11.1.

A system level refresh sequence should always be used and running.

Do not write to the GenClk Interrupt Mask Register. If the Per1 flag is cleared in GenClk Interrupt Mask Register the system level refresh is disabled.

6.1.4.4 Interrupt Handling

The interrupt registers give complete freedom to the software, by providing means to mask interrupts, clear interrupts, force interrupts and read interrupt status. The table below describes the basic interrupt structure that is implemented by each individual module.

Register	Acronym	Read	Write
Pending Interrupt Masked Status Register	PIMSR	Reads PIR **and** IMR	-
Pending Interrupt Masked Register	PIMR	Reads PIR **and** IMR PIR is cleared	-
Pending Interrupt Status Register	PISR	Reads PIR	-
Pending Interrupt Register	PIR	Reads PIR PIR is cleared	Writes PIR **or** write data
Interrupt Mask Register	IMR	Reads IMR	Writes IMR

Table 6-1 Interrupt Register Summary

When an interrupt occurs the corresponding bit in the Pending Interrupt Register is set. The normal sequence to initialise and handle a module interrupt is:

- Set up the software interrupt-handler to accept an interrupt from the module.
- Read the Pending Interrupt Register to clear any spurious interrupts.
- Initialise the Interrupt Mask Register, unmasking each bit that should generate the module interrupt.
- When an interrupt occurs, read the Pending Interrupt Register in the software interrupt-handler to determine the causes of the interrupt. This will also clear the Pending Interrupt Register.
- Handle the interrupt, taking into account all causes of the interrupt.

Masking interrupts: After reset, all interrupt bits are masked, since the Interrupt Mask Register is zero. To enable generation of a module interrupt for an interrupt bit, set the corresponding bit in the Interrupt Mask Register.

Clearing interrupts: All bits of the Pending Interrupt Register are cleared when it is read or when the Pending Interrupt Masked Register is read. Reading the Pending Interrupt Masked Register yields the contents of the Pending Interrupt Register masked with the contents of the Interrupt Mask Register.

Forcing interrupts: When the Pending Interrupt Register is written, the resulting value is the original contents of the register logically OR-ed with the write data. This means that writing the register can force (set) an interrupt bit, but never clear it.

Reading interrupt status: Reading the Pending Interrupt Status Register yields the same data as a read of the Pending Interrupt Register, but without clearing the contents.

Reading interrupt status of unmasked bits: Reading the Pending Interrupt Masked Status Register yields the contents of the Pending Interrupt Register masked with the contents of the Interrupt Mask Register, but without clearing the contents.

6.2 Memory Interface

6.2.1 Initialisation

The following steps should be performed by the configuration block at power-up to initialise the Memory Interface, i.e. all values below should be set in the initialisation PROM (cf. section 6.11.1):

- Configure each memory bank concerning:
  - Data width
  - Wait states
  - EDAC protection
  - Static write protection
  - Write restrictions
- Configure the memory area locations
- Configure the TME bandwidth ratio
- Configure the memory scrubber

6.2.2 Operation/Usage

The following registers of the Memory Interface are write protected and can only written during power-on configuration by the configuration block:

- PIM TME Ratio Register
- PIM MUnit $ Register
- PIM MUnit $ Address Low Register
- PIM MUnit $ Address High Register

6.2.2.1 Memory Banks

The Memory Interface provides three memory banks, which can be individually configured. Configuration is done using the PIM MUnit * Register. Certain aspects of some memory banks can not be configured. The following mapping is done between the PIM MUnit * Registers and the chip select signals:

MUnit	**Chip select**
MUnit 0 Register	*PromCsN*
MUnit 2 Register	*ExtRecLacN*
MUnit 3 Register	*RamCsN*

The configurations described within the following sub-sections can only be set up during configuration.

6.2.2.1.1 Configuring memory data width

The memory data width for memory banks 0, 2 and 3 can be configured to be 8 or 16 bits through the PIM MUnit $ Register.Size field. The memory data width of memory bank 0 (*PromCsN*) is configured via the *MemSize16* external input.

Configure memory bank *N* to use 8 bits data width by writing 0 to PIM MUnit *N* Register.Size.

Configure memory bank *N* to use 16 bits data width by writing 1 to PIM MUnit *N* Register.Size. Setting the Size field to 2 or 3 is equal to setting it to 1.

6.2.2.1.2 Configuring wait states

For each memory bank it is possible to configure the number of wait states for read and write accesses. The number of wait states is programmable between 0 and 7, even though 0 wait states for write accesses is not recommended.

Configure the number of wait states for read accesses from memory bank *N* through PIM.MUnit *N* Register.WsRd.

Configure the number of wait states for write accesses to memory bank *N* through PIM.MUnit *N* Register.WsWr.

6.2.2.1.3 Enabling and disabling EDAC protection

EDAC protection is provided individually for memory banks 2 (*ExtRecLacN*) and 3 (*RamCsN*) and disabled for memory bank 0 (*PromCsN*).

Enable EDAC protection for memory bank *N* by setting PIM MUnit *N* Register.EDAC.

Disable EDAC protection for memory bank *N* by clearing PIM MUnit *N* Register.EDAC.

6.2.2.1.4 Enabling and disabling write restriction

Write restriction is only programmable for memory bank 2 (*ExtRecLacN*), for all other banks it is disabled, even though memory bank 3 (*RamCsN*) contains certain areas that have restricted write enabled. When write restriction is enabled the Memory Interface only accepts write accesses from one function to that memory bank; for memory bank 2 (*ExtRecLacN*) that function is the PDEC3 authentication sub-layer, which uses the memory bank for the external recovery LAC.

Enable write restriction on memory bank 2 (*ExtRecLacN*), thereby restricting the memory bank to the external recovery LAC, by setting PIM MUnit 2 Register.WrRes.

Disable write restriction on memory bank 2 (*ExtRecLacN*) by clearing PIM MUnit 2 Register.WrRes.

Write restriction has higher priority than both write protection mechanisms. If write restriction is enabled for memory bank 2 write protection is always disabled regardless of the value of the corresponding configuration bits (cf. section 6.2.2.3).

6.2.2.2 Enabling External Recovery LAC

The external recovery LAC of the PDEC3 is located in memory bank 2 (*ExtRecLacN*). To use the external recovery LAC, configure PIM MUnit 2 Register as follows:

- Enable write restriction by setting the WrRes flag
- Disable static write protection by clearing the WrDis flag
- Configure memory size as byte if data bits 7-0 are connected to the external recovery LAC and as halfword if data bits 15-8 are connected to the external recovery LAC
- Configure wait states according to the selected memory type
- Ensure that the LSB of the Recovery LAC Configuration byte in the PDEC3 configuration PROM is set

6.2.2.3 Write Protection

The Memory Interface provides two levels of write protection: static write protection and dynamic write protection. The static write protection is configured at power-on reset while the dynamic write protection can be switched on and off during operation by software, e.g. for protecting User Areas 1 and 2. A memory bank is write protected if any of the write protection mechanisms are enabled. Note that write restriction has higher priority than the two write protection mechanisms. This is only applicable for memory bank 2 (*ExtRecLacN*), which can be restricted to the external recovery LAC counter of the PDEC3. It is not possible to enable write protection for a memory area with restricted write enabled.

6.2.2.3.1 Static write protection

Static write protection is provided for all memory banks except memory bank 3 (*RamCsN*), which is always writable. The static write protection is configured in the PIM MUnit $ Register during configuration only.

Enable static write protection for memory bank *N* by setting PIM MUnit *N* Register.WrDis.

Disable static write protection for memory bank *N* by clearing PIM MUnit *N* Register.WrDis.

6.2.2.3.2 Dynamic write protection

The Memory Interface provides a dynamic write protection mechanism on the following memory areas:

- Memory bank 2 (*ExtRecLacN*)
- Memory bank 3 - User Area 1 (*RamCsN*)
- Memory bank 3 - User Area 2 (*RamCsN*)

Enable dynamic write protection for a memory area by writing to PIM Write Disable Set Register with the corresponding bit set. Writes to this register sets all bits where the corresponding bit in the written data is set, while leaving all other bits unaffected.

Disable dynamic write protection for a memory area by writing to PIM Write Disable Clear Register with the corresponding bit set. Writes to this register clears all bits where the corresponding bit in the written data is set, while leaving all other bits unaffected.

The current status of the write protection mechanism can be read through PIM Write Disable Status Register. Write protection is enabled for all areas where the corresponding bit is set.

6.2.2.4 Memory Area Locations

The Memory Interface provides information about the location of the different memory areas through the PIM MUnit * Address Low Register and PIM MUnit * Address High Register. Some of the memory areas are fixed in location either entirely or only the start address. Information is available for the following areas:

- Memory bank 0 (*PromCsN*)
- Memory bank 2 (*ExtRecLacN*)
- Memory bank 3 – User Area 1 (*RamCsN*)
- Memory bank 3 – User Area 2 (*RamCsN*)
- Memory bank 3 –TME / PM (*RamCsN*)

The start addresses of the following memory areas are configurable:

- Memory bank 3 – User Area 1 (*RamCsN*)
- Memory bank 3 – User Area 2 (*RamCsN*)
- Memory bank 3 –TME / PM (*RamCsN*)

The end addresses of the following memory areas are configurable:

- Memory bank 0 (*PromCsN*)
- Memory bank 2 (*ExtRecLacN*)
- Memory bank 3 – User Area 1 (*RamCsN*)
- Memory bank 3 – User Area 2 (*RamCsN*)
- Memory bank 3 –TME / PM (*RamCsN*)

The memory area locations can only be set up during configuration.

Set the start address of a memory area by writing the first byte address of the memory area to the corresponding PIM MUnit * Address Low Register. Since the 15 LSBs of the registers are fixed to 0, the address must be a multiple of 32 kbyte. The 3 MSBs of the registers are fixed.

Set the end address of a memory area by writing the first byte address after the memory area to PIM MUnit * Address High Register, e.g. a value of $03FF8000_{16}$ means that the last accessible address of the memory area is $03FF7FFF_{16}$. Since the 15 LSBs of the registers are fixed to 0, the address must be a multiple of 32 kbyte. The 3 MSBs of the registers are fixed.

If the start and end addresses of the memory areas in memory bank 3, *RamCsN*, are set in such a way that the areas overlap, the Memory Interface selects the memory areas in the following order:

1. The restricted areas, i.e. PDEC buffer area, ExtCpduIf packet area, and PDEC programmable key
2. User area 1
3. User area 2
4. TME / PM area

6.2.2.5 Memory Scrubber

6.2.2.5.1 Configuring the memory scrubber

The following aspects of the memory scrubber must be configured in order for the memory scrubber to work properly:

- Start address
- End address
- Scrubbing rate

The scrubber starts at the start address and scrubs one word at a time until it reaches the end address and then cyclically continues from the start address. The time between scrubbing two consecutive words is determined by the scrubbing rate. The scrubbing rate defines the number of clock cycles between two consecutive accesses, i.e. if the scrubbing rate is *N* the scrubber makes an access each N^{th} cycle.

Set the start address of the memory scrubber by writing the word aligned-byte address to PIM Scrubber Start Address Register.

Set the end address of the memory scrubber by writing the address of the first byte after the area to be scrubbed (as a word-aligned byte address) to PIM Scrubber End Address Register; this byte will not be scrubbed.

Set the scrubbing rate by writing the desired number of clock cycles between two consecutive accesses to PIM Scrubber Config Register. If the memory scrubber registers are to be refreshed by either the system level refresh or the user level refresh it is important that the scrubbing rate is selected high enough to allow the entire area to be scrubbed between two subsequent refresh sequences. This is necessary since the scrubber is restarted each time the PIM Scrubber Config Register is written.

6.2.2.5.2 Enabling and disabling the memory scrubber

The memory scrubber enabling and disabling is controlled via the PIM Status Register.ScuEn flag.

Enable the memory scrubber by writing PIM Status Set Register with the ScuEn flag set.

Disable the memory scrubber by writing PIM Status Clear Register with the ScuEn flag set.

6.2.2.5.3 Checking memory scrubber status

The present address of the memory scrubber can be read through PIM Scrubber Address Register.

6.2.2.6 Configuring TME Bandwidth Ratio

It is possible to configure the TME bandwidth ratio of the Memory Interface, i.e. the total number of *SysClk* cycles dedicated to the TME memory accesses, which constitutes a TME period. Between two TME periods one other access is allowed (two accesses if a read-modify-write is performed). If no other DMA channel tries to access the memory at this point the new TME period starts immediately.

Set the number of TME cycles to N, $0 \leq N \leq 127$, by writing N to PIM TME Ratio Register.

For a high performance TM-only system the TME ratio is typically set to 90, while a more mixed system requires the TME ratio to be reduced to 10-30 to prevent latency problems in other functions.

It is only possible to set the TME bandwidth ratio during configuration.

6.2.2.7 Forcing Propagation of Memory Writes

The Memory Interface contains internal buffers, one for the TME accesses and one for all other accesses. It is possible to force the contents of these buffers to be written to memory.

Ensure that any previous buffer flush has finished prior to forcing a buffer to be emptied. This is done by waiting until PIM Status Register.FlushAct is cleared.

Force the TME buffers to be emptied by writing to PIM FlushT Register.

Force the other buffers to be emptied by writing to PIM FlushO Register.

6.2.3 Error Handling

6.2.3.1 Address Error During Register Access

If a non-existing register is accessed in any of the modules register areas, the Memory Interface does the following:

- Signals a DMA error to the requesting module
- Stores the address in PIM First Failing Address Register
- Stores the data of the access in PIM First Failing Data Register
- Provides information about the access in PIM Error Info Register
- Sets the PIM Error Trap Register.ETAPi flag
- Issues the ET interrupt

When the information has been stored these registers are locked and can not change until being unlocked. The registers are unlocked by writing PIM Error Trap Register.

6.2.3.2 Access to Unmapped Memory

If an address outside the memory map, i.e. outside the memory area locations, is accessed, the Memory Interface does the following:

- Signals a DMA error to the requesting module
- Stores the address in PIM First Failing Address Register
- Stores the data of the access in PIM First Failing Data Register
- Provides information about the access in PIM Error Info Register
  No information about memory area is stored since it is unavailable
- Sets the PIM Error Trap Register.ETABcc flag
- Issues the ET interrupt

When the information has been stored these registers are locked and can not change until being unlocked. The registers are unlocked by writing PIM Error Trap Register.

6.2.3.3 Uncorrectable EDAC Errors During Memory Read

If an uncorrectable error is encountered during a memory read, the Memory Interface does the following:

- Signals a DMA error to the requesting module
- Stores the address in PIM First Failing Address Register
- Stores the data of the access in PIM First Failing Data Register
- Provides information about the access in PIM Error Info Register
- Sets the PIM Error Trap Register.ETUde flag
- Issues the ET interrupt

When the information has been stored these registers are locked and can not change until being unlocked. The registers are unlocked by writing PIM Error Trap Register.

Error Trap is not affected by errors in accesses from TME. Uncorrectable errors from TME are ignored.

6.2.3.4 Uncorrectable EDAC Errors During Memory Scrubbing

If the memory scrubber detects an uncorrectable EDAC error, the Memory Interface does the following:

- Issues the ScuErr interrupt

6.2.3.5 Timeout During Register Access

The Memory Interface contains a timer for register accesses to avoid deadlock if does not receive any response to an access. If the timer expires, the Memory Interface does the following:

- Signals a DMA error to the requesting module
- Stores the address in PIM First Failing Address Register
- Stores the data of the access in PIM First Failing Data Register
- Provides information about the access in PIM Error Info Register
- Sets the PIM Error Trap Register.ETBto flag
- Issues the ET interrupt

When the information has been stored these registers are locked and can not change until being unlocked. The registers are unlocked by writing PIM Error Trap Register.

6.2.3.6 Access Violation

If a module tries to write a write restricted memory area, the Memory Interface does the following:

- Signals a DMA error to the requesting module
- Stores the address in PIM First Failing Address Register
- Stores the data of the access in PIM First Failing Data Register
- Provides information about the access in PIM Error Info Register
- Sets the PIM Error Trap Register.ETAcc flag
- Issues the ET interrupt

When the information has been stored these registers are locked and can not change until being unlocked. The registers are unlocked by writing PIM Error Trap Register.

6.2.3.7 Correctable EDAC Errors

If the memory scrubber detects a correctable EDAC error or a correctable EDAC error is encountered during a normal memory access, the Memory Interface does the following:

- Stores the address in PIM First Failing Correctable Error Address Register
- Corrects the EDAC error
- Issues the CErr interrupt

When the information has been stored the PIM First Failing Correctable Error Address Register becomes locked and can not change until it is unlocked. Unlock the PIM First Failing Correctable Error Address Register by writing it.

The address stored in the register is incorrect if the TME encounters the correctable error. If the TME is disabled the address can always be trusted.

6.2.3.8 EDAC Errors During Read-Fill

When an incomplete 32-bit word is written, a read-modify-write must be made when writing it to memory. When this happens, a read-fill is initiated, to complete the word with data from the external memory. If an EDAC error is encountered during the read-fill the following happens, depending on the nature of the error.

6.2.3.8.1 Correctable EDAC errors during read-fill

When a correctable EDAC error is detected during a read-fill, the Memory Interface does the following:

- Stores the address in PIM First Failing Correctable Error Address Register
- Corrects the EDAC error
- Issues the CErr interrupt
- Generates new, correct EDAC checkbits for the entire word
- Writes the entire word to memory (without any EDAC error)

When the information has been stored the PIM First Failing Correctable Error Address Register becomes locked and can not change until it is unlocked. Unlock the PIM First Failing Correctable Error Address Register by writing it.

6.2.3.8.2 Uncorrectable EDAC errors during read-fill

When an uncorrectable EDAC error is detected during a read-fill, the Memory Interface does the following:

- Generates new, correct EDAC checkbits for the entire word
- Writes the entire word to memory (without any EDAC error)

When doing this, the data read from memory could be corrupted, due to the uncorrectable error. The new data, however, is unaffected.

6.2.3.9 Software handling of errors

When an error occurs an interrupt is issued informing software the nature of the error. If the ET interrupt bit is set, the reason for the error is stored in PIM Error Trap Register. Determine the source of the error by reading PIM Error Info Register.EI Dma. Various information about the access is available in PIM Error Info Register, PIM First Failing Address Register, and PIM First Failing Data Register. When status has been read, write any value to PIM Error Trap Register to allow next erroneous access to be detected.

When receiving the CErr interrupt, a correctable EDAC error has been detected. When the information has been checked, write any value to PIM First Failing Correctable Address Register to allow next EDAC correction to be detected. Note that the address in the register is incorrect if the TME detected the error, which can result in addresses without EDAC protection being shown in this register.

Uncorrectable errors detected by the memory scrubber need no handling. The scrubber does not stop scrubbing memory due to uncorrectable errors.

6.2.4 Usage Constraints

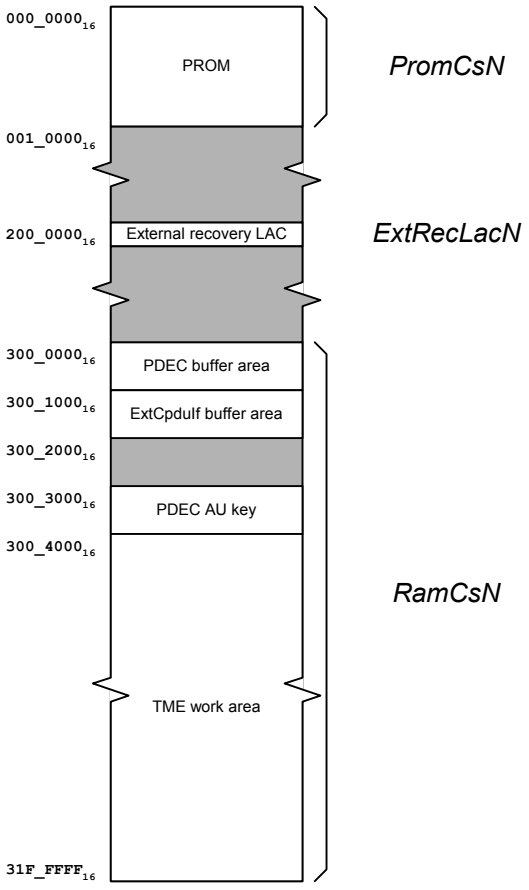
6.2.4.1 Functional

The following usage constraints apply to memory accesses:

- Do not write memory used by the TME through any of the control interfaces when the TME is running.
  Should this be done the contents of the memory can be inconsistent with the data used by the TME.

6.2.5 Examples

6.2.5.1 Configuring memory



To configure the external memory as in the figure above, do the following:

1. Memory bank 0 is configured as a write protected 64 kbyte byte sized PROM with 7 wait states Wr and 3 wait states Rd by writing the following registers:
   PIM MUnit 0 Register is set to $4EC_{16}$
   PIM MUnit 0 Address High Register is set to 10000_{16}
2. The external recovery LAC is configured as a write restricted (but not write protected for obvious reasons) halfword sized non-volatile memory with 7 wait states Wr and 3 wait state Rd by writing the following registers:
   PIM MUnit 2 Register is set to $8ED_{16}$
   PIM MUnit 2 Address High Register is set to 200_8000_{16}
3. Memory bank 3 is configured as a 2 Mbyte halfword sized EDAC protected RAM with 1 wait state Wr and 0 wait states Rd by writing the following registers:
   PIM MUnit 3 Register is set to 121_{16}
   PIM MUnit TME / PM Address Low Register is set to 300_0000_{16}
   PIM MUnit TME / PM Address High Register is set to 320_0000_{16}

Apart from this the scrubber has to be set up. Configure the scrubber to scrub the PDEC authentication key and the entire TME working area as follows:

1. Write the start address of the area, 300_3E90_{16} (which is the start of the authentication key), to PIM Scrubber Start Address Register.
2. Write the first address after the end of the area, 320_0000_{16}, to PIM Scrubber End Address Register.
3. Set the scrubber rate to 1 word scrubbed each 230^{th} us by writing $E6A_{16}$ to PIM Scrubber Configuration Register.
4. Start the scrubber by writing 1 to PIM Status Set Register.

Released

6.3 Packet Telecommand Decoder Module (PDEC3)

6.3.1 Initialisation

The PDEC3 is self-initialising. After a Reset Assertion, the PDEC3 fetches the first 8 octets from configuration PROM located at addresses 0000_0200_{16} through 000_0207_{16}. The PDEC3 is ready to execute when these 8 octets have been received. The configuration PROM contents are described further in Table 6-19.

If the memory buffer interface is to be used, the interrupt mask should be set to enable interrupts.

6.3.2 Operation/Usage

6.3.2.1 Checking Frame Analysis Report

Read the 32 bit Frame Analysis Report (FAR) by accessing the Frame Analysis Report register. Since the FAR only contains the 6 least significant bits of the code block counter, the full code block counter can be read from the Code Block Counter register when Transfer Frame lengths larger than 256 octets are used.

If the FAR contains a new report, the Stat flag of Frame Analysis Report is cleared. The Stat flag is set when Frame Analysis Report is read. The PDEC3 issues the NewFar interrupt whenever the Stat flag in Frame Analysis Report becomes cleared. This means that no new interrupt will be received until PDEC Frame Analysis Report has been read. Since the Stat flag is reset to zero, the register must be read once after power-on reset to enable interrupt reception.

Note: Code Block Counter is only updated when Frame Analysis Report is read.

The PDEC3 provides independent copies of the Frame Analysis Report and Code Block Counter that behave in the same way as these, except for the interrupt generation, which is associated solely with the Stat flag in PDEC Frame Analysis Report. The copies are named Frame Analysis Report Copy and Code Block Counter Copy. Since the read out of these registers is destructive, the copies have been added to allow two independent users, such as an essential telemetry function and software, to access the information.

All copy registers are placed in sequence to allow these to be read out through a single control interface access. Since the interrupt is associated with the original PDEC Frame Analysis Report an interrupt controlled readout mechanism can not use the copy registers.

6.3.2.2 Checking Command Link Control Word

Read the 32 bit Command Link Control Word (CLCW) through Command Link Control Word.

6.3.2.3 Checking Authentication Unit Status Report

The status of the Authentication Unit can be read by doing the following:

- Read AU Status Report Part 1
- Read the following registers in any order:
  - AU Status Report Part 2
  - AU Status Report Part 3

Note: AU Status Report Part 2 and AU Status Report Part 3 are only updated when AU Status Report Part 1 is read, otherwise they contain old data.

The PDEC3 provides independent copies of AU Status Report Part $ that behave in the same way as these. The copies are named AU Status Report Copy Part $. Since the read out of these registers is destructive, the copies have been added to allow two independent users, such as an essential telemetry function and software, to access the information.

All copy registers are placed in sequence to allow these to be read out through a single control interface access. Note that if the copy registers are used the interrupt functionality is disabled.

6.3.2.4 Checking the Status of the MAP Interface

6.3.2.4.1 Checking the MAP Link Address

The MAP Link Address can be read through MAP Link Address. The contents of this register are updated as soon as a new segment bound for any MAP interfaces has been received.

Note: MAP Link Address is updated even if the MapAdr and MapClkFreq values that are fetched from configuration PROM, see Table 6-19, are incorrect, cf. section 6.3.3.4.1.

6.3.2.4.2 Checking Status of General MAP Interface

The status of the general MAP Interface can be read through the General MAP Interface Status Register.

6.3.2.4.3 Checking Status of Dedicated MAP Interfaces

The status of the dedicated MAP Interfaces can be read through the following registers:

- MAP Interface 1 Status Register
- MAP Interface 2 Status Register
- MAP Interface 3 Status Register
- MAP Interface 4 Status Register
- MAP Interface 5 Status Register

6.3.2.4.4 Checking Status of Last MAP Used

The status of the MAP used for the last transfer can be read through Common Interface Status Register.

Note: The contents of this register is the same as the contents of the last updated of MAP Interface 1 Status Register through MAP Interface 5 Status Register and General MAP Interface Status Register.

6.3.2.5 External Recovery LAC

The PDEC3 supports the use of an external recovery LAC instead of the internal. If external recovery LAC is used, the value of the recovery LAC is fetched from external memory after reset. When a telecommand frame causes the recovery LAC to change, the PDEC3 updates the value of the external recovery LAC.

The external recovery LAC is enabled by setting the LSB of the Recovery LAC Configuration located in the configuration PROM. The external recovery LAC is disabled by clearing the LSB of the Recovery LAC Configuration, set in the configuration PROM.

When enabled the external recovery LAC is located at address 200_0000_{16}, which is located in memory bank 2. It is important that memory bank 2 is properly configured and that the memory is non-volatile. For further description of how to configure memory bank 2 for an external recovery LAC see section 6.2.2.2.

6.3.2.6 De-Randomiser

The PDEC3 provides a de-randomiser function, which can be enabled by means of a configuration octet in mission PROM. When enabled the de-randomiser function descrambles the individual bits in a frame according to the following polynomial:

$$h(x) = x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$$

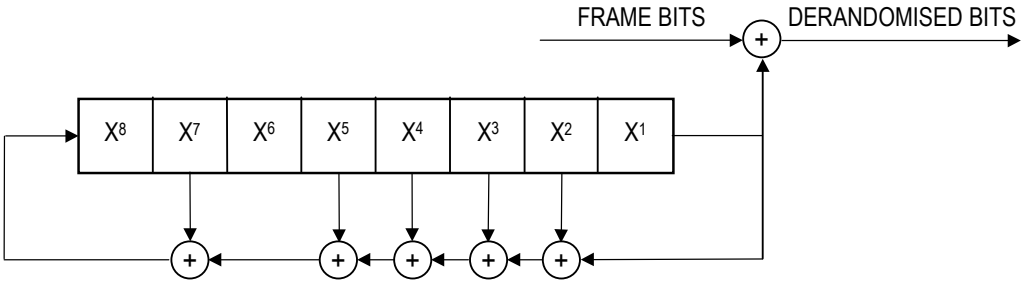


Figure 6-1 De-Randomiser

The de-randomiser function is enabled by setting the LSB of the De-Randomiser Configuration located in the configuration PROM.

The de-randomiser function is disabled by clearing the LSB of the De-Randomiser Configuration.

6.3.2.7 Receiving a PM Buffer Segment

The PDEC3 provides an internal MAP Interface for handing over buffers containing Telecommand Segments directly to the PM and software.

When sending a Telecommand Segment to the software, the PDEC3 does the following:

- Set Buffer Pointer to point to the first byte in the Telecommand Segment
- Set Segment Length to the total number of bytes in the Telecommand Segment
- Issue the TCNew interrupt

When the Telecommand Segment has been processed, signal this to the PDEC3 by writing Buffer Free.

6.3.2.8 Routing of Telecommand Segments

Segment routing is determined by using the MAP ID of the segment header to look up the destination in PROM. The lookup table (LUT) entries are interpreted as depicted in Figure 6-2. Another LUT is used to determine the transfer speed on an external MAP Interface. This LUT is also indexed with the MAP ID of the segment header, which means that there is a one-to-one mapping between the MapAdr LUT and the MapClkFreq LUT.

MapAdr		
Par	Val	MapGenA
bit 7	bit 6	bit 5..0

Figure 6-2 Value returned from the MAP ID lookup

For the LUT values to be valid, the following shall hold

- The valid flag (Val) should be set
- The total number of ones in the entire MapAdr value should be odd
- The MapClkFreq lookup value should be at least 1 and at most 13.

To invalidate a single MapAdr value, ensure that the Val flag is cleared. The validity of the LUT entries is checked and the result is interpreted as follows:

1. If MapGenA is equal to 0, the TC Segment is transferred to the CPDM
2. If MapGenA is equal to 1-5, the TC Segment is output on the corresponding dedicated MAP Interface
3. If MapGenA is equal to 6, the telecommand decoder asserts the internal *TcOnly* input on the CSEL module.
4. If MapGenA is equal to 7, the TC Segment is transferred to the PM buffer area interface
5. Otherwise, the TC Segment is output on the general MAP Interface (MAP G).

To generate the output clock for the designated MAP interface, the MapClkFreq look-up value is used. The MapClkFreq value determines the output clock frequency in relation to the system clock frequency as follows:

$$f_{MAP} = \frac{f_{SysClk}}{2^{MapClkFreq}}$$

Note: The PDEC3 provides a hardware feature to switch MAP interface 1 and 2 to allow routing to an active and a redundant system where the same interface is used independent of which system is currently active.

6.3.3 Error Handling

6.3.3.1 Coding Layer

6.3.3.1.1 Abandoned CLTU

A CLTU can be abandoned in the coding layer for several reasons. The PDEC3 reports this by setting Frame Analysis Report.FrameAna to "Abandoned CLTU".

The following events cause a CLTU to be abandoned:

- Its *PdecTcAct** input is deasserted
- The first TC Code Block after the Start Sequence is rejected (i.e. the CLTU is too short)
- The 148th Code Block after the Start Sequence is accepted (i.e. the CLTU is too long)
- The clock timeout expires
- In priority mode, the Start Sequence timeout expires
- In priority mode, the *PdecTcAct** input of a higher priority TC channel is asserted.

6.3.3.2 Transfer Layer

6.3.3.2.1 Discarding Frames

Frames can be discarded in several ways. When a frame is discarded, the PDEC3 releases the frame buffer and reports the reason in Frame Analysis Report.FrameAna and Frame Analysis Report.IReason.

The main reasons for discarding a frame is:

- Frame is dirty
- Frame is illegal
- Frame is discarded by the FARM-1

6.3.3.2.2 BD Frame Discard

BD frames should normally not be discarded after being declared clean and legal. If, however, the authentication sub-layer is busy performing authentication on a previous segment, this action can not be interrupted. The PDEC3 discards the BD frame and reports "AD frame discarded because of Wait" in Frame Analysis Report.FrameAna.

Note: The combination Frame Analysis Report.Type set to "BD Frame" and Frame Analysis Report.FrameAna set to "AD frame discarded because of Wait" is abnormal and indicates that frames are received in a higher pace than the PDEC3 can handle.

6.3.3.2.3 DMA Error and Timeout in Transfer Layer

If the PDEC3 receives a DMA error while writing to the frame buffer or if the timeout expires, the frame is discarded and, if it is declared clean and legal, "Illegal for one reason" is reported in Frame Analysis Report.FrameAna and Frame Analysis Report.IReason is set to "No Illegal Report".

Note: This combination of Frame Analysis Report.FrameAna and Frame Analysis Report.IReason should always be regarded as indicating DMA errors.

6.3.3.3 Authentication Sub-Layer

6.3.3.3.1 Authentication Error

The Telecommand Segment might be rejected during authentication for several reasons:

- Incorrect length
- Error in LAC
- Error in signature

If any of these errors occur, the PDEC3 rejects the segment and reports the reason in Frame Analysis Report.AuAna.

6.3.3.3.2 Non-Executable AU Control Command

If the authentication sub-layer receives and authorises a control command with illegal format, it is considered non-executable. The segment is discarded and "Non-executable authorised AU Control Command" is reported in Frame Analysis Report.AuAna.

6.3.3.3.3 DMA Error and Timeout During Authentication

If the PDEC3 receives a DMA error or if a DMA transfer times out while reading memory to authenticate a segment, the segment is rejected and "Error in Signature" is reported in Frame Analysis Report.AuAna.

6.3.3.3.4 DMA Error and Timeout During Control Command Execution

If the PDEC3 receives a DMA error or the timeout expires on a DMA transfer while reading an octet of the fixed key to copy it to the programmable key, the octet is simply skipped and thus not copied. While writing to the programmable key, the PDEC3 ignores DMA errors. The PDEC3 does not report DMA errors during control command execution.

Note: DMA errors during control command execution may leave the programmable authentication key in an unknown state.

This can only happen as a result of an SEU. The error is detected by sending authenticated frames using the programmable key; some of these will be discarded due to signature error. Since only a few bytes of the key will be erroneous it can still be possible to send authenticated frames.

Recover from the error by selecting the fixed key and then copying the fixed key to the programmable key again.

6.3.3.4 Segmentation Layer

6.3.3.4.1 Error in MapAdr or MapClkFreq

If either the parity of MapAdr is incorrect or Val is cleared, see Figure 6-3, the PDEC3 discards the segment. Map Link Address is updated but none of the other MAP Interface registers.

MapAdr		
Par	Val	MapGenA
bit 7	bit 6	bits 5..0

Figure 6-3 MAP address look-up value structure

If the value of the received MapClkFreq is less than 1 or greater than 13 then the PDEC3 discards the segment. Map Link Address is updated but none of the other MAP Interface registers.

6.3.3.4.2 DMA Error in Segmentation Layer

A DMA error in the segmentation layer leads to the segment being discarded and only parts, or possibly nothing, of the segment being sent on the designated MAP. Depending on when the DMA error occurs, the MAP Interface registers might be updated.

Note: A DMA error while reading MapAdr causes Map Link Address to remain unchanged.

6.3.3.4.3 Loss of Backend Buffer

If a BD frame arrives while a segment is being sent on a MAP Interface, the PDEC3 aborts the transfer on the selected MAP Interface by asserting *PdecMapAdt*. A segment residing in the backend buffer waiting for the CPDM to finish executing a previous packet from the PDEC3 is also discarded without any report (cf. section 5.4.3.2.1).

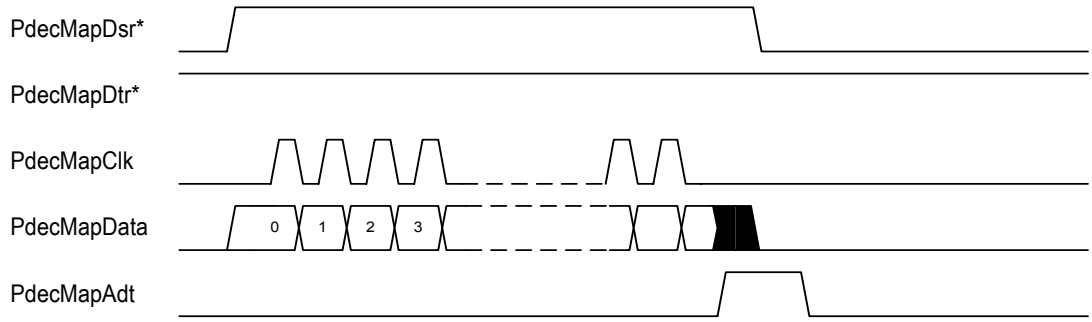


Figure 6-4 Aborted transfer of a TC Segment

6.3.3.5 Memory Buffer Interface Abort

If a BD frame arrives while a telecommand is being processed by the software, the PDEC3 steals the PM buffer area. This is signaled to the software by issuing the TCAbort interrupt.

Stop processing the Telecommand Segment when receiving a TCAbort interrupt, since the data has been altered.

Note: Do not write Buffer Free after receiving a TCAbort interrupt.

6.3.4 Usage Constraints

Not applicable

6.3.5 Examples

Not applicable

6.4 External CPDU Interface Module (ExtCpduIf)

6.4.1 Initialisation

Not applicable

6.4.2 Operation/Usage

6.4.2.1 Receiving data

The External CPDU Interface receives telecommand segments on a Packet Wire interface and stores them in memory starting at address 0300_2001_{16}. ExtCpduIf can receive segments containing at most 1023 bytes including the segment header.

Send a telecommand segment containing a CPDU packet to ExtCpduIf by doing the following:

- Create a complete telecommand segment (cf. section 4.9.3.3) containing a valid CPDU packet (cf. section 4.9.4.1.1)
- Send the segment to ExtCpduIf on the Packet Wire input starting with the MSB of the segment header

6.4.3 Error Handling

In case of error the handling is that the segment that is being received is not sent to the CPDU. Below are some examples of error that can happen.

6.4.3.1 Overflow Error

The External CPDU Interface limits the size of packets to 1023 bytes. If the transmitter tries to send more data than allowed the External CPDU Interface discards the entire packet without sending it to the CPDU. The maximum offset can be read through ExtCpduIf DMA Max Offset Register.

6.4.3.2 Framing Error

The length of the packet was not a multiple of 8 bits, i.e. somewhere in the received packet there is an error.

6.4.3.3 Overrun Error

This can only occur if the transmitter ignored the *ExtCpduIfRdy* signal and the FIFO has become full.

6.4.3.4 Abort

The segment has been aborted by the sender and shall therefore not be sent to the CSEL.

6.4.4 Usage Constraints

Not applicable

6.5 CPDM Selector Module (CSEL)

It is possible to set up CSEL during configuration only. Users have access to the following functions of CSEL:

- Enabling and disabling interrupts
- Sending PM sequences to the CPDM for execution
- Reading all status and configuration registers
- All error handling

None of the other functions described within the following sub-sections are available to users. The descriptions are to be viewed as instructions about how to select the configuration PROM values.

6.5.1 Initialisation

The following steps should be performed by the configuration block at power-up to initialise the CSEL module, i.e. all values below should be set in the initialisation PROM (cf. section 6.11.1):

- During configuration, setup CSEL 1 Second Register to hold the value $N - 1$, where N is the number of clock cycles elapsed in 1 second.
- Set Maximum TC Only time
- Set CSEL Status Timeout value
- Setup interrupt mask to enable interrupts (optional)

6.5.2 Operation/Usage

6.5.2.1 Operating Mode

The CSEL has three different operating modes, RM ON, RM OFF, and TC Only. In RM ON mode, all requests are accepted, with RM requests being prioritised. In RM OFF mode, all RM requests are discarded. In TC Only mode, only Ground accesses are accepted and the Ground has unrestricted access to the CPDM.

Transits between modes are controlled by hardware. The software can check the current mode by reading CSEL Status Register.

The CSEL transits to TC Only mode if it detects an error.

The TC Only mode is temporary. The CSEL automatically returns to its previous mode after $D \cdot N$ cycles, where $D - 1$ is the value of CSEL Max TC Only Time Register and $N - 1$ is the value of CSEL 1 Second Register. The timeout is restarted if a new "Set TC Only" command is given.

6.5.2.2 Set Maximum TC Only Time

To set the time the CSEL spends in TC Only mode before returning to its previous mode, during configuration, write $D - 1$ to CSEL Max TC Only Time Register, where D is the number of seconds the CSEL should wait before leaving TC Only mode after the Set TC Only command has been given.

Typically the TC Only timeout is set to about 10-20 seconds, which is long enough to allow one CLTU to be sent to disable both RMs.

This can only be done during configuration.

6.5.2.3 Set CSEL Status Timeout Value

To set the timeout for the CSEL Status reception, during configuration, write $T - 1$ to CSEL Remote Status Timeout Time Register, where T is the number of seconds within which the CSEL requires the incoming CSEL Status to change to idle.

This value should be higher than the time required for the CPDM to execute two maximum length CPDU Packets. Depending on the system this time can be anything between a few seconds to several minutes but in a typical system the timeout value is set to about 20 seconds, which is long enough for two packets containing ten pulses of 1 second each.

This can only be done during configuration.

6.5.2.4 Request CPDU Packet Execution

Request the execution of a CPDU Packet in the CPDM by doing the following:

- Build a Telecommand Segment containing a CPDU Packet in memory. For a description of the Telecommand Segment structure, refer to section 4.9.3.3 and section 4.9.4.1.1.
- Write the address to the first byte of the Telecommand Segment to the CSEL PM Buffer Pointer Register.
- Write the length of the Telecommand Segment to the CSEL PM Segment Length Register.
- Write $BEC0FFEE_{16}$ to the CSEL PM Request Register to start the request.

Note: Do not write to any of these registers during a PM request; this is considered illegal, see further section 6.5.4.1.

The CSEL handles the request and issues one of the interrupts listed in Table 6-2 when done.

Interrupt	Cause
CpdmDone	The packet was successfully executed by the CPDM
CpdmError	The CPDM signaled an error while executing the packet
Discard	The CSEL discarded the request without sending it to the CPDM
Abort	The packet was aborted by a higher priority request

Table 6-2 Interrupts concerning result of execution of PM requests

6.5.2.5 Checking Status

The CSEL status can be read through CSEL Status Register. The register contains the following information:

- BusyGnd, the CSEL has received and is processing a Ground request
- BusyPm, the CSEL has received and is processing a PM request
- BusyRm, the CSEL has received and is processing an RM request
- FailSilent, the CSEL has entered fail silent mode and only accepts Ground requests
- CpdmBusy, the CPDM is processing a request
- Mode, the current operating mode of the CSEL

6.5.2.6 Checking Status of Remote CPDM Selector

The status of the remote CSEL, which is received on CSEL Status, can be read through CSEL Remote Status Register. The register contains the following information:

- GndPm, the remote CSEL is processing a Ground or PM request
- BusyRm, the remote CSEL is processing an RM request
- Silent, the remote CSEL does not send any status at all
- Error, illegal status detected
- Timeout, the remote CSEL has not stopped processing a request in time

6.5.3 Error Handling

The CSEL can experience three kinds of errors: CSEL Status error, CSEL Status timeout, and internal errors.

6.5.3.1 CSEL Status Error

The CSEL Status is protected by a parity bit, allowing single error detection. If the parity of the incoming code is incorrect the CSEL does the following:

- Issues the RSError interrupt
- Clears CSEL Remote Status Register and sets the Error flag in CSEL Remote Status Register
- Changes mode to TC Only
- Aborts any PM or RM, i.e. ExtCpduIf if it is connected to the RM request interface, sequence currently executing in the CPDM
- Discards any pending PM or RM sequence as a result of becoming TC Only

If a PM request is being processed in the CSEL or CPDM when the error occurs the CSEL terminates the request and issues either the Discard or the Abort interrupt.

Note: CSEL Remote Status Register is not updated until the error has been recovered. The CSEL remains in TC Only mode until the error has been recovered.

When the error on CSEL Status has been corrected, i.e. a valid code is received, the CSEL does the following:

- Issues the NoErr interrupt
- Sets CSEL Remote Status Register according to CSEL Status
- Changes mode to the mode determined by the current state of the *CselRmOn* input

6.5.3.2 CSEL Status Timeout

If CSEL Status shows that the remote CPDM is busy and does not become idle before the timeout has expired the CSEL does the following:

- Issues the RSTime interrupt
- Clears CSEL Remote Status Register and sets the Timeout flag in CSEL Remote Status Register

When CSEL Status becomes idle or illegal, the CSEL updates CSEL Remote Status Register accordingly and clears the Timeout flag.

Note: The CSEL ignores the remote CSEL after timeout on CSEL Status, independent of the status of the remote CSEL, until CSEL Status changes.

A timeout on the status link indicates that the remote CSEL is erroneous in some way. It is highly recommended that the remote CSEL set in RM OFF or TC Only mode.

6.5.3.3 Internal Error

If the CSEL detects an internal error, it does the following:

- Issues the IErr interrupt
- Changes mode to TC Only
- Becomes silent on CSEL Status and sets the Silent flag in CSEL Status Register

6.5.4 Usage Constraints

6.5.4.1 Functional

The following usage constraints apply to PM requests:

- Do not write to either CSEL PM Buffer Pointer Register or CSEL PM Segment Length Register during a PM request, i.e. after CSEL PM Request Register has been written and before an interrupt has been received that the request has finished. Should this occur, the CPDM will abandon the command pulse generation.

- Do not write to the CSEL PM Request Register during a PM request. Should this be done, the CSEL immediately terminates the packet, possibly resulting in only parts of the CPDU packet being executed. The CSEL issues the Abort and Discard interrupts.

- Do not alter the contents of the Telecommand Segment during a PM request. Should this be done, illegal CPDU commands might be executed. There is however the lockout functionality that will reduce the risk for inadvertently issuing CPDU command that should not be accessible by the PM.

6.5.5 Examples

6.5.5.1 Setting up the CSEL to operate in 16 MHz

1. During configuration:
2. Write $15{,}999{,}999_{10}$ to the CSEL 1 Second Register.
3. Set TC Only mode timeout to 15 seconds by writing 14_{10} to the CSEL Max TC Only Time Register.
4. Set CSEL Status timeout to 20 minutes, to allow even the longest CPDU packet to be executed, by writing $1{,}199_{10}$ to the CSEL Remote Status Timeout Time Register.
5. Enable all interrupts.

6.5.5.2 Sending a PM request for execution in the CPDM

1. Build a Telecommand Segment containing a CPDU packet and place it in memory, cf. Figure 6-5.
2. Write the address of the Telecommand Segment, the buffer pointer, to the CSEL PM Buffer Pointer Register.
3. Write the length of the Telecommand Segment, N in Figure 6-5, to the CSEL PM Segment Length Register.
4. Check the CSEL operating mode. If the CSEL is TC Only, wait until it leaves this mode.
5. Send the request to the CSEL by writing $BEC0FFEE_{16}$ to the CSEL PM Request Register.
6. If either the CPDM or the remote CPDM is busy, the CSEL issues a Discard interrupt. If this happens, wait until the following flags in the CSEL Status Register are cleared:
   - BusyGnd
   - BusyRm
   - CpdmBusy

   and the following flags in the CSEL Remote Status Register are cleared:
   - GndPm
   - BusyRm.

   Restart from paragraph 4.

   Note: The buffer pointer and segment length are already set up and need not be changed.
7. Wait on any of the interrupts listed in Table 6-2 from the CSEL.

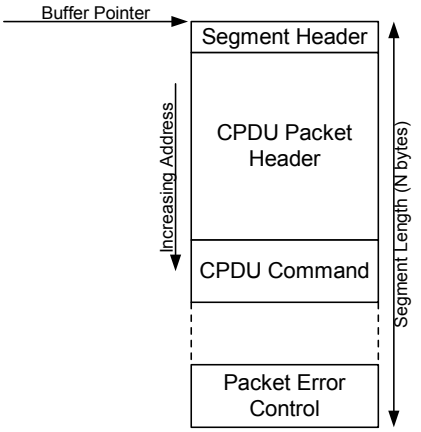


Figure 6-5 Telecommand Segment in memory

6.5.5.3 Setting up CSEL status timeout

In a system the maximum pulse strobe length is 8 *durations* and the *duration*, *D*, is set to 13 ms (cf. section 6.6.2.3). Packets sent to the External CPDU Interface contain no more than 10 commands and telecommands from ground contain a maximum of 15 commands. The system clock frequency is 8 MHz.

To eliminate the possibility of a remote status timeout during normal execution set up the remote status timeout as follows:

$t_{TC} = 8 \cdot 0.013 \text{ s} \cdot 15 = 1.56 \text{ s}$
$t_{RM} = 8 \cdot 0.013 \text{ s} \cdot 10 = 1.04 \text{ s}$

$t_{TO} > t_{TC} + t_{RM} = 2.6 \text{ s}$

The timeout must be set higher than 2.6 s.

1. During configuration:
2. Write $7{,}999{,}999_{10}$ to the CSEL 1 Second Register.
3. Set CSEL Status timeout to 3 seconds by writing 2 to the CSEL Remote Status Timeout Time Register.

6.5.5.4 Using TC Only mode for exclusive access

A system is visible from the ground station very seldom and only during 10 minutes at a time. During this time it is vital that the ground station can reconfigure the system without interference from the External CPDU Interface. The system clock frequency is 16 MHz and the ground station can send CPDU Packets of any length contained in a 256-byte telecommand transfer frame. The *duration*, *D*, is set to 13 ms (cf. section 6.6.2.3).

A 256-byte TC frame can contain a maximum of 120 CPDU command instructions. Each instruction can be $128 \cdot 0.013 \text{ s} = 1.664 \text{ s}$ long. The maximum execution time for a CPDU Packet is 199.68 s. Due to the long execution time it is advisable to set the CSEL status timeout high enough to allow at least two such packets to be executed without a timeout in the remote CSEL.

Configure the system as follows:
1. Write $15{,}999{,}999_{10}$ to the CSEL 1 Second Register.
2. Set TC Only mode timeout to 10 minutes to allow the ground station to take control over the CPDM during the communications window by writing 599_{10} to the CSEL Max TC Only Time Register.
3. Set CSEL Status timeout to 400 seconds by writing 399_{10} to the CSEL Remote Status Timeout Time Register.

When the 10-minute window commences, do the following:
1. Send a telecommand segment routed to MAP 6 to set TC Only mode. The TC Only command should be sent to both CSELs in a redundant system. Both CSELs are now TC Only and nothing can interfere with the ground commands.
2. Reconfigure the system by sending telecommand segments routed to the CPDM, MAP 0.

CSEL returns to its previous mode automatically when the 10-minute timeout expires.

6.6 Command Pulse Distribution Module (CPDM)

It is possible to set up CPDM during configuration only. Users have access to the following functions of CPDM:

- Enabling and disabling interrupts
- Reading all status and configuration registers
- All error handling

None of the other functions described within the following sub-sections are available to users. The descriptions are to be viewed as instructions about how to select the configuration PROM values.

6.6.1 Initialisation

The following steps should be performed by the configuration block at power-up to initialise the CPDM module, i.e. all values below should be set in the initialisation PROM (cf. section 6.11.1):

- Set the pulse duration
  See section 6.6.2.3.
- Set the serial output clock frequency
  See section 6.6.2.4.
- Configure the CPDM regarding:
  – Packet Identification
  – CPDU command parity
  See section 6.6.2.5.1and 6.6.2.5.2
- Set the lockout threshold
  See section 6.6.2.6
- Set up the interrupt mask to enable interrupts (optional)

6.6.2 Operation/Usage

6.6.2.1 Checking Status of Ground Telecommand Packets

The information about CPDU telecommand packets originating from the specific source, normally ground, is stored in a separate register. This is done to allow a telemetry function to report the status of the last telecommand packet from ground.

Check the status by reading CPDM Status Report Register.

If the Done flag is set, all fields of the register have been updated, otherwise the CPDM is still processing the telecommand packet.

6.6.2.2 Checking Status of Telecommand Packets

The CPDM logs the information about the processing of the last received CPDU telecommand packet originating from any other source than the specific source, normally ground, in a status register.

Check the status by reading CPDM Status Register.

If the Done flag is set, all fields of the register have been updated, otherwise the CPDM is still processing the telecommand packet.

6.6.2.3 Setting the Pulse Duration

Command pulse related timing in CPDM is based on multiples of *D*, where *D* stands for *duration*. In order to generate correct timing the CPDM needs to know the system clock frequency.

To set up the pulse duration, write $N - 1$ to CPDM Pulse Duration Register. N is the number of system clock cycles in the pulse duration. For correct behaviour, select $10\ \text{ms} \le D \le 15\ \text{ms}$.

$$N = \frac{D}{T_{SysClk}}$$

The pulse duration, *D*, defines the shortest possible strobe width. The width of the strobe output is determined by the pulse length field of the CPDU packet (cf. section 4.9.4.1.1) and is calculated as follows:

$$T_{\text{Strobe width}} = 2^{(\text{Pulse length})} \cdot D$$

The maximum strobe width is $128 \cdot D$.

Note: The pulse duration can only be set during configuration.

System clock frequency (MHz)	**CPDM Pulse Duration Register value**		
	***D* = 10 ms**	***D* = 13 ms**	***D* = 15 ms**
10	99999_{10}	129999_{10}	149999_{10}
11	109999_{10}	142999_{10}	164999_{10}
12	119999_{10}	155999_{10}	179999_{10}
13	129999_{10}	168999_{10}	194999_{10}
14	139999_{10}	181999_{10}	209999_{10}
15	149999_{10}	194999_{10}	224999_{10}
16	159999_{10}	207999_{10}	239999_{10}

Table 6-3 Pulse duration values at different frequencies

6.6.2.4 Setting up the Serial Output Clock Frequency

For every CPDU Command Instruction, the CPDM outputs the Pulse Number on a serial interface. The serial interface consists of a data signal and a clock signal, *CpdmClk*. The frequency of the serial output clock is defined as follows:

$$f_{CpdmClk} = \frac{f_{SysClk}}{2^{(X+2)}}$$

Where *X* is the value of CPDM Clock Ratio Register. To set up the serial output clock frequency, write *X* to CPDM Clock Ratio Register.

$$X = \log_2\left(\frac{f_{SysClk}}{f_{CpdmClk}}\right) - 2$$

For correct behaviour and avoid pulse timeout, select a *CpdmClk* period less than 1 *duration* (*D*). It is recommended to select $f_{CpdmClk} \geq 16$ kHz to avoid excessive shift-out time. The serial output clock frequency can only be set during configuration.

6.6.2.5 Configuring the CPDM

The CPDM is configured by writing the CPDM Configuration Register.

After reset the configuration is as follows:
- Packet Identification = 0
- CPDU command instruction parity is disabled

Note: The configuration can only be set during configuration.

6.6.2.5.1 Setting the Packet Identification

As a part of the legal check, performed on all CPDU telecommand packets, the CPDM compares the Packet Identification provided in the Packet Header with the value stored in Packet ID. If the values differ, the telecommand packet is considered illegal and is discarded.

Ensure that the Packet ID field contains the correct Packet Identification.

6.6.2.5.2 CPDU Command Instruction Parity

The CPDM provides a CPDU Command Instruction parity function. If the parity check is enabled, the CPDM assumes odd parity over the entire Command Instruction, i.e. the total number of ones is assumed to be odd. The parity check is performed as a part of the clean check when receiving the CPDU telecommand packet and once again right before the CPDU Command Instruction is executed.

Enable the CPDU Command Instruction parity check by setting ParEn.

Disable the CPDU Command Instruction parity check by clearing ParEn.

Note: The CPDU Command Instruction parity is not compliant with [TC_SPEC].

6.6.2.6 Setting the Lockout Threshold

A user can be prevented from sending packets containing certain CPDU pulses. The CPDM checks that a lockout enabled telecommand packet does not contain any inaccessible pulses as a part of the clean check. A pulse is considered inaccessible if (Output Number modulo 2048) < Lockout threshold. If the CPDU telecommand packet contains any inaccessible pulses, the telecommand packet is considered dirty and is discarded. Only the 11 least significant bits of the Output Number are checked against the lockout threshold value, resulting in two lockout areas as shown in Figure 6-6.

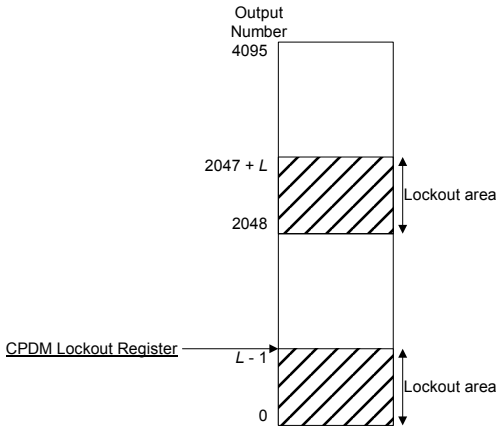


Figure 6-6 Lockout areas

Set the lockout threshold to *L* by writing *L* to CPDM Lockout Register.
This disables all pulses for which the following holds:
Output Number < *L* or 2048 ≤ Output Number < 2048 + *L*

Setting the lockout threshold to 0 disables the user lockout function.

Note: The lockout threshold can only be set during configuration.

6.6.2.7 Configuring the minimum time between two pulses

It is not possible to directly configure the time between two consecutive pulses on *CpdmStrb*. The minimum time between the deassertion of *CpdmStrb* and the next assertion is defined by the *duration*, *D*, and the serial output clock frequency as follows:
$T_{Low} = 3 \cdot D / 8 + 17 \cdot T_{CpdmClk}$

By decreasing the serial output clock frequency the time between two pulses can be increased with preserved value of *D*.

6.6.3 Error Handling

6.6.3.1 Dirty CPDU Telecommand Packet

If a CPDU telecommand packet is found dirty, the CPDM does the following:
- Sets the LstPkt field in either CPDM Status Register or CPDM Status Report Register to 11_2.
- Discards the packet and reports an error to the sender

The following is checked in the clean check:
- Sequence Flags in Segment Header
- Packet Length
- Packet Error Control (CRC)
- Lockout, if applicable
- CPDU Command Instruction parity, if enabled

6.6.3.2 Illegal CPDU Telecommand Packet

If a CPDU telecommand packet is found clean, but illegal, the CPDM does the following:
- Sets the LstPkt field in either CPDM Status Register or CPDM Status Report Register to 10_2.
- Discards the telecommand packet and reports an error to the sender

The following is checked in the legal check:
- Sequence Flags in Packet Header
- Packet Identification

6.6.3.3 Aborted CPDU Telecommand Packet

If the CPDU telecommand packet execution is aborted prematurely, the CPDM does the following:
- If a pulse is being output, as indicated by *CpdmArm* being asserted, the CPDM finishes sending the pulse. The length of the pulse is shortened so that it continues no longer than 5·D.
- Sets the Abort flag in either CPDM Status Register or CPDM Status Report Register.
- Stops processing the current telecommand packet and starts processing the new telecommand packet.

6.6.3.4 Errors During Telecommand Packet Execution

If an error is detected while the CPDM is processing a CPDU telecommand packet, the CPDM does the following:
- Finishes any ongoing pulse output but doesn't start any new
- Sets the Error flag in either CPDM Status Register or CPDM Status Report Register.
- Stops processing the telecommand packet and reports an error to the sender.

6.6.3.5 Pulse Timeout

The CPDM has a pulse timeout to ensure that all pulses terminate. The pulse timeout is $160 \cdot D$ and starts when the serial output starts.

If the pulse timeout expires, the CPDM does the following:

- Immediately stops the pulse output
- Sets the Error flag in either CPDM Status Register or CPDM Status Report Register.
- Stops processing the telecommand packet and reports an error to the sender.

6.6.4 Examples

6.6.4.1 Achieving a minimum time between consecutive pulses

This example shows how to achieve a minimum time of 5 ms between any two consecutive pulses. The clock frequency of the system is 16 MHz and D is required to be 13 ms. The serial output clock frequency has no lower limit.

The serial output clock frequency is calculated as follows:

$$\frac{3 \cdot D}{8} + 17 \cdot T_{SysClk} \cdot 2^{X+2} \geq 5 \cdot 10^{-3} \Rightarrow \frac{3 \cdot 13 \cdot 10^{-3}}{8} + 17 \cdot 62.5 \cdot 10^{-9} \cdot 2^{X+2} \geq 5 \cdot 10^{-3} \Rightarrow$$

$$X \geq \log_2 \left(\frac{5 \cdot 10^{-3} - 3 \cdot 13 \cdot 10^{-3} / 8}{17 \cdot 62.5 \cdot 10^{-9}} \right) - 2 \Rightarrow X \geq 4.88$$

Released

This results in a minimum time between two pulses (as defined by *CpdmStrb* assertion) of 5.011 ms.

1. Set the *duration* to 13 ms by writing $32C7F_{16}$ to CPDM Pulse Duration Register.
2. Set the serial output clock ratio to 5 by writing 5 to CPDM Clock Ratio Register.

6.7 Packet Telemetry Encoder Module (TME)

6.7.1 Initialisation

After power up, the TME will not produce any telemetry output until it has been provided a bit clock, its reset has been released, a valid access to the InitTM register has been performed and the *TmeEnable* signal is asserted. The TME will then start generating Transfer Frames, starting with an Idle Transfer Frame.

These frames will be Reed-Solomon and convolutionally encoded as determined by the TmEnCfg1 register status after reset. The TME will also try to retrieve CLCW data on the CLCW interface as determined by the TmEnCfg0 and TmEnCfg1 registers. In order to change the configuration of the TME, the register of the required field needs to be written. By holding the *TmeEnable* signal deasserted after reset, the TME will be disabled. In this state all registers can be accessed, to allow for configuration of a disabled TME.

The TME is initialised by the configuration block after power-up, i.e. all values needed should be set in the initialisation PROM (cf. section 6.11.1).

Note: Any register writes to any register listed in 6.10.7.2 should be followed up by a software reset to ensure proper function of the TME.

6.7.2 Operation/Usage

This section describes how to configure and control the TME.

6.7.2.1 VC Input Interfaces

The following parameters can be programmed for the VC Input Interface:

- Whether to use the Packet Wire or SpaceWire interface.
- Whether the input data is packet or non-packet data
- Whether the First Header Pointer should be calculated dynamically for non-packet data or have a static value of all ones.
- Idle Packet version field value (000_2, 100_2), common for all VCs.
- Idle Packet insertion timeout (Immediate, 1/4/16/64/256/1024 polls or No Idle Packet insertion), only available for VCA A-D.
- Value of the Data Field Synchronisation flag and the Packet Order flag (0_2, 1_2).
- Value of the Segmentation Length Identifier field (2 bits).

The size of each VC Frame Buffer in number of frames (in steps of 8 frames).

VC IF Selection

Packet Wire or SpaceWire interface is selected by setting VcCfg*.SelS for Packet Wire, and clearing it for SpaceWire.

Non-packet Data
The TME can be configured for non-packet data by setting the VcCfg*.DFS bit. In non-packet data mode a dynamic First Header Pointer can optionally be calculated in the same way as for Source/Telemetry Packets. This is described further below.

Dynamic First Header Pointer
The First Header Pointer reset value is dependent on the VcCfg*.DFHP field. If VcCfg*.DFHP = 0, and the InitTM register is written, the First Header Pointer is initiated to 'all ones' and 'all zeros' if VcCfg*.DFHP = 1. This allows the VC Input Interfaces to operate in full compliance with [TM_STD] for non-packet data.

If *the packet delimiter* was not de-asserted while all the data filling up a Transfer Frame was input, e.g. if a Telemetry Packet or data block is longer than the data field of the Transfer Frame or VcCfg*.DFHP = 0 and VcCfg*.Dfs = 1, the First Header Pointer is set to 'all ones'; First Header Pointer = $111\_1111\_1111_2$. A summary of the FHP values can be seen in Table 6-4.

VC input IF State	First Header Pointer Setting
Idle	$111\_1111\_1110_2$ (all ones minus one)
Active	$111\_1111\_1111_2$ (all ones) when no first packet header in frame data field or VcCfg*.Dfs = 1 and it's corresponding VcCfg*.DFHP = 0. Variable from $000\_0000\_0000_2$ (all zero) for the first octet to maximum count value for the last octet in the frame data field for TM Packets and for non-packet data blocks when VcCfg*.DFHP = 1.

Table 6-4 First Header Pointer Settings

Idle Packet Insertion
VC[A..D] support Idle Packet insertion when operating in packet mode. Idle Packet insertion will start after a pre-programmed number of polls from the Frame Generator determined by the VcCfg*.PProg register field. When the Frame Generator polls a VC Input Interface that has an incomplete Transfer Frame resident in the memory buffer, a Poll Counter within the VC Input Interface starts. If the Poll Counter reaches the value programmed, ranging from 0 to 1024, the Idle Packet Insertion process is activated and the Poll Counter is reset. The Poll Counter is also reset if a user packet start is detected.

The Idle Packet insertion process stops when the Transfer Frame Data Field is complete and the remainder, if any, of the last Idle Packet has been stored in the Transfer Frame Data Field for the next frame.

Transfer Frame Primary Header
The value of the Data Field Synchronisation flag and the Packet Order flag in the Telemetry Transfer Frame Primary Header are the same as programmed in the VcCfg*.DFS and VcCfg*.POF fields.

Memory Allocation

For each VC, a variable amount of memory can be allocated for the external memory buffer. The number of frame buffers for which memory is allocated to each VC is determined by the VcCfg*.VcBufSize register field. The value in this field is always a multiple of 8. The physical memory block used for a single frame buffer is according to Table 6-5.

Transfer Frame length	Block size
223 octets	256 – 32 = 224 octets
446 octets	512 – 64 = 448 octets
892 octets	1024 –128 = 896 octets
1115 octets	1024 + 128 = 1152 octets
239 octets	256 – 16 = 240 octets
478 octets	512 – 32 = 480 octets
956 octets	1024 – 64 = 960 octets
1195 octets	1024 + 256 = 1280 octets

Table 6-5 Single Frame Memory Usage

If VcCfg*.VcBufSize = 0 for a VC Input Interface, the corresponding interface is disabled, and will not respond to any data transmitted.

For a VC transmitting telemetry packets, enough memory must be allocated to hold an entire maximum size packet plus an additional frame, since the TME can not start transmitting the packet until it has been fully written to memory.

The number of frames to allocate for a single VC can be calculated as follows:

$$\left\lceil \frac{L_{PKT}}{L_F - L_H - L_{SH} - L_{OPCF} - L_{FECW}} \right\rceil + 1$$

L_{PKT}	Maximum packet length in octets including header and CRC
L_F	Frame length in octets for the VC
L_H	Frame header size in octets = 6
L_{SH}	Secondary header size in octets = 4 or 0 (depending on configuration)
L_{OPCF}	OPCF size in octets = 2 or 0 (depending on configuration)
L_{FECW}	FECW size in octets = 4 or 0 (depending on configuration)

Example: A VC with frame size 1115, using secondary header, OPCF, and FECW inteded for transmitting packets of maximum size (65542 octets) needs to be allocated a buffer area of at least 61 frames. Since memory is allocated as multiples of 8 frames, an area large enough to hold 64 frames must be allocated, which calculates to 73728 octets.

Telemetry Data Input Flow

The TME can receive data on 8 different interfaces, which can optionally be connected to serial Packet Wire interfaces or different channels on the SpaceWire link. When data is input on serial Packet Wire interfaces, no special care needs to be taken concerning full buffers in the TME. The TME will deassert the corresponding *TmeSRdy* signal to stop the Packet Wire transmitter from inputting more data. If the Packet Wire transmitter fails to stop transmitting data within 2 bytes from the point where *TmeSRdy* is deasserted, data will be lost.

For telemetry data input from the SpaceWire link, there are some issues that must be considered. If a telemetry memory buffer becomes full during input of a packet from SpaceWire, congestion will occur on SpaceWire since the ongoing packet can not be completed. This will block out all other SpaceWire channels until the full telemetry buffer has enough free memory to receive the rest of the ongoing packet.

If the SpaceWire link is used to feed more than one of the VCAs in the TME, or if SpaceWire is used as control interface together with data transmission to the TME, congestion is not desirable, and should be avoided.

The following procedure can be used when transmitting telemetry data over the SpaceWire link to avoid congestion:

1. Read the corresponding TmeVcBufStat* register to find out how much more data that can be received by the VCA. The register indicates how many kByte data that can be received (at the least) in 1024 byte steps.
2. If the buffer status indicates that 0 kByte data can fit in the buffer, there may still be room for a smaller packet. There is a bit, TmeVcCfg*.VCAR that indicates if a threshold level of free memory in the corresponding telemetry memory buffer is exceeded. The threshold levels can be set in the TmeVcCfg*.VT field to four different levels. If the packet to transmit is smaller than the currently programmed VT threshold level, read the VCAR bit to see if the packet can be received completely by the TME.
3. If the packet fits in the telemetry memory buffer according to the TmeVcBufStat* register or the VCAR bit, transmit the packet on SpaceWire. If not, wait and check the registers again after some time. The time to wait until the next try is dependent on the bitrate of the telemetry link, the bandwidth programmed for the VCAs in the BAT and on the amount of data transmitted on other VCAs. If priority selection is used instead of bandwidth allocation, all data in higher priority link buffers must be transmitted on the telemetry link before a lower priority buffer will deallocate memory in its buffer.

6.7.2.2 Frame Generator

The following parameters can be programmed:

- Use normal or alternative Attached Synchronisation Marker
- Transfer Frame length (223, 446, 892, 1115, 239, 478, 956 or 1195 octets).
- Which Virtual Channel Identifier that shall be used for each VC. Two VCs should not use the same identifier.
- Spacecraft Identifier (10 bits).
- Insertion of Secondary Header, OPCF (CLCW) and FECW fields, respectively (no, yes).
- Rate of *TmeTimeStrb* output (once per 1/2/4/8/16/32/64/128/256 frames from VC 0).

Attached Synchronisation Marker

For Telemetry Transfer Frames and R-S Codeblocks two different Attached Synchronisation Markers can be selected depending on the value of TmEnCfg0.ASM.

TmEnCfg0.ASM	Attached Synchronisation Marker
0	$1ACF\_FC1D_{16}$
1	$352E\_F853_{16}$

Table 6-6 Attached Synchronisation Marker for Frames and R-S Codeblocks

Transfer Frame

The generated Transfer Frame consists of required and optional parts, for which the structure can be seen in Figure 4-13. The TmEnCfg0.SHF, TmEnCfg0.Opcf and TmEnCfg0.Fecw register fields, respectively, determine which of the optional fields that are present. The TmEnCfg0.Len field specifies the length of the Transfer Frame as 223, 446, 892, 1115, 239, 478, 956, 1195 bytes.

Primary Header

Each Transfer Frame is required to have a Primary Header, which structure can be seen in Figure 4-14. It is possible to select which VC[A...H] that will be connected to each virtual channel [VCID0...VCID7] by programming the VcIdCfg$ registers according to Table 6-7:

Input Interface	Virtual Channel ID used
VC[A]	VcIdCfg0.VCAID
VC[B]	VcIdCfg0.VCBID
VC[C]	VcIdCfg0.VCCID
VC[D]	VcIdCfg0.VCDID
VC[E]	VcIdCfg1.VCEID
VC[F]	VcIdCfg1.VCFID
VC[G]	VcIdCfg1.VCGID
VC[H]	VcIdCfg1.VCHID

Table 6-7 Virtual Channel Identification Configuration

The Operational Control Field Flag has the same value as TmEnCfg0.Opcf .
Secondary Header Flag has the same value as TmEnCfg0.SHF.
Synchronisation Flag has the same value as VcCfg$.DFS.
Packet Order Flag has the same value as VcCfg$.POF.
Segment Length Identifier has the same value as VcCfg$LSeg.

Secondary Header
The secondary header is optional and is present only if the TmEnCfg0.SHF register field is set to 1.

Transfer Frame Trailer
The Transfer Frame Trailer contains none, either or both of the OPCF and CLCW fields. Its structure is shown in Figure 4-4. The Operational Control Field (OPCF) is present when the TmEnCfg0.Opcf register field is set to 1. The two byte Frame Error Control Word (FECW) is present when the TmEnCfg0.Fecw register field is set to 1.

6.7.2.2.1 Reed Solomon Space

The interleaving depths supported by the telemetry encoder are 1, 2, 4 and 5 resulting in the Transfer Frame lengths as specified by TmEnCfg0.Len and Reed-Solomon codewords of lengths 32/16, 64/32, 128/64 and 160/80 respectively.

Interleave depth, I	**Transfer Frame length** [Base 223 / Base 239]	**R-S check symbols** [Base 223 / Base 239]	**Telemetry Codeblock length**
1	223/239	32/16	255
2	446/478	64/32	510
4	892/956	128/64	1020
5	1115/1195	160/80	1275

Table 6-8 Block sizes

6.7.2.2.2 Idle Transfer Frames

Idle Transfer Frames are always generated from VC[H]. The Virtual Channel Identifier field of the Idle Transfer Frames is the same as programmed in the VcIdCfg1.VCHID register field.

6.7.2.2.3 Time Strobe

The Frame Generator provides the active high *TmeTimeStrb* for sampling the on-board time according to section 7.4 of [TM_STD]. *TmeTimeStrb* is asserted on the rising edge of the *TmeUnEncSync* signal when the Attached Synchronisation Marker is started to be output.

The sampling interval is determined by the value of the TmEnCfg0.Time register field, in accordance with [TM_STD]. The sampling interval of 256 frames (TmEnCfg0.Time = 1000) corresponds to the assertion of *TmeTimeStrb* for each Transfer Frame with the Virtual Channel Count LSB value of 0, while the sampling interval of 1 frame (TmEnCfg0.Time = 0000) corresponds to the Virtual Channel Count LSB values of 0, 1, 2, 3, 4 ... 253, 254 and 255. The *TmeTimeStrb* is only asserted for Virtual Channel Identifier 0. For further description about how a VC input is mapped to a Virtual Channel Identifier, see section 6.7.2.2. Time strobe is generated for idle frames only if VCH is mapped to Virtual Channel Identifier 0.

6.7.2.3 CLCW Interface

The CLCW interface provides a mechanism for multiplexing the CLCW from up to four packet telecommand decoders. Which data input is used is determined by the TmEnCfg1.SCW register field. *TmeClcwD0* is used when TmEnCfg1.SCW = 00_2, *TmeClcwD1* is used when TmEnCfg1.SCW = 01_2. TmEnCfg1.SCW can be set to 10_2 to automatically toggle between the *TmeClcwD0* and *TmeClcwD1* as CLCW source. A Transfer Frame with an even Master Channel Frame Count will have a CLCW retrieved from the *TmeClcwD0* and *TmeClcwD1* inputs for odd Transfer Frames. If TmEnCfg1.SCW = 11_2 , the CLCW will be retrieved from one of the four synchronous bit serial interfaces in a round-robin fashion, starting with *TmeClcwD0* for the first frame.

By setting TmEnCfg0.COW = 1_2, bit 16 and 17 of the CLCW reports retrieved from the telecommand decoders will be overwritten using the information on the *PdecRfAvN$* and *PdecTcAct$* input pins instead. The corresponding bits are fetched from the telecommand decoder CLCW reports if TmEnCfg0.COW = 0_2. This allows two different types of system architectures.

6.7.2.4 VC Selection with BAT

The Frame Generator features two built-in algorithms for selecting from which VC Input Interface to output a Transfer Frame:

- Bandwidth Allocation, TmEnCfg1.PS is cleared.
- Priority Selection, TmEnCfg1.PS is set.

6.7.2.5 Reed Solomon Encoder

The Reed-Solomon Encoder is enabled by setting the TmEnCfg1.RS register field to 1.

6.7.2.6 Pseudo-randomiser

The Pseudo-Randomiser is enabled by setting the TmEnCfg.PR register field to 1.

6.7.2.7 Convolutional Encoder

The Convolutional Encoder is enabled by setting the TmEnCfg1.CV register field to 1. The six different Convolute Rates 1/2, 1/2 no inversion, 2/3 punctured, 3/4 punctured, 5/6 punctured and 7/8 punctured are selectable via the TmEnCfg1.CvR register field.

6.7.2.8 Turbo Encoder

The Turbo Encoder is enabled by setting the TmEnCfg.TU register field to 1. The four different nominal code rates, r = 1/2, 1/3, 1/4, or 1/6, are selectable via the TmEnCfg1.TuR register field.

Note: Turbo Encoder does not support frame lengths of 239, 478, 956 and 1195 bytes.

6.7.2.9 Test Pattern Generator

When the register field TmEnCfg1.TST is set to one, the test pattern generator will continuously generate a bit pattern determined by the TmEnCfg1.TstPat field as shown in Table 6-9. The test pattern generator overrides all other outputs from the TME.

TmEnCfg1.TstPat	**Test bit pattern**
00	0
01	01
10	0011
11	0000_1111

Table 6-9 Test bit pattern

6.7.2.10 Parameters that can be changed on the fly

The following TME configuration parameters could be changed on the fly without a reset of the encoder:

- TmEnCfg0.Time, TmEnCfg0.PV and TmEnCfg0.COW
- TmEnCfg1.SCW, TmEnCfg1.PS, TmEnCfg1.TstPat
- VcCfg*.PProg, VcCfg*.DFS, VcCfg*.LSeg, and VcCfg*.VT

When changing any of these parameters the current frame might become corrupted.

6.7.2.11 Modulation

The different modulations can be enabled by setting the corresponding TfmCtrl.NRZ, TfmCtrl.SP, TfmCtrl.PSK and TfmCtrl.PSKS register field.

For PSK Square modulation the phase of the Modulated SubCarrier can be shifted in order for a Modulated '1' to start at a rising or falling edge of the Modulated SubCarrier. This is selectable via the TfmCtrl.PSKS.ZC register field.

Any changes made to the TfmCtrl register must be followed up by a write, with the value AA_{16}, to the TFMCommit register for the changes to take place. All changes will take place directly after the end of the ongoing Telemetry Transfer Frame output but the next Transfer Frame may be lost.

6.7.2.12 Clock Divider

The clock divider uses two divisor register values to divide the *TMClk* to generate *SubCClk* and the *BitClk*.

The frequencies are calculated as follows:

$$f_{SubCClk} = f_{TMClk} / (SubCDivisor * 2)$$

$$f_{BitClk} = f_{SubCClk} / (BitRateDivisor + 1)$$

If the SubCDivisor and the BitRateDivisor registers are both 0 the *BitClk* frequency will be the same as the *TMClk* frequency. The value 0 should not be used for the SubCDivisor in any other case.

When PSK modulation is enabled, the ratio between the BitClk frequency and the SubCClk frequency must be a multiple of 4. This ratio indicates the number of SubCarrier periods that will be used for each modulated symbol in the PSK output data stream.

All changes made to the SubCDivisor and the BitRateDivisor registers must be followed up by a write, with the value AA_{16}, to the TFMCommit register for the changes to take place. All changes will take place directly after the end of the ongoing Telemetry Transfer Frame output but the next Transfer Frame may be lost.

Note: All parameters in the TfmCtrl, SubCDivisor and BitRateDivisor registers can be changed on the fly and do not require a reset of the TME. The TFMCommit register however needs to be accessed for the changes to take place.

6.8 SpaceWire Module (SPW)

It is possible to set up SPW during configuration only. Users have access to the following functions of SPW:

- Transmitting a block of data
- Time-code reception
- Reading all status and configuration registers
- All error handling

None of the other functions described within the following sub-sections are available to users. The descriptions are to be viewed as instructions about how to select the configuration PROM values.

6.8.1 Initialisation

The following steps should be performed by the configuration block at power-up to initialise the SPW module, i.e. all values below should be set in the initialisation PROM (cf. section 6.11.1):

- Configure the SPW regarding
  – Routing (optional)
  See section 6.8.2.1.
- Set up the transmission rate, making sure that the nominal rate is 10±1 Mbps.
  See section 6.8.2.4.
- Set up the SpaceWire timing, thereby deasserting the SpaceWire link handler reset signal.
  See section 0.
- Enable the SpaceWire link and configure it regarding
  – Transfer rate for NULL tokens
  See section 6.8.2.2 and 6.8.2.3.
- Configure transmission
  See section 6.8.2.6
- Set up the parallel write interfaces regarding
  – End of Message
  See section 6.8.2.7.
  Note that End of Message is enabled if a parallel write interface is left unconfigured.
- Set up interrupt mask to enable interrupts. (optional)

Note: The SpaceWire link remains reset until the SpaceWire module timing has been set up, i.e. nothing can be transmitted and the link remains idle.

6.8.2 Operation/Usage

6.8.2.1 Configuring the SPW

The SPW is configured by writing the SPW Configuration Register. Note: When altering a field in the SPW Configuration Register care must be taken to preserve the values of the remaining fields.

After reset the configuration is as follows:

- Default routing enabled to VRC 0.

Configuring the SPW can only be done during configuration.

6.8.2.1.1 Setting a default routing address

The SPW can be configured to send all received packets to a single VRC. When default routing is enabled all bytes in the packet are treated as data bytes. Set the default routing address to *N* by doing the following:

- Clear the RoutRxEn flag.
- Set the DefRout field to *N*.

If $N \geq 8$ the SPW issues the AddrErr interrupt whenever a packet is received and discards all packet data.

6.8.2.1.2 Enable routing

When routing is enabled, the SPW treats the first byte of every SpaceWire packet as the destination address and forwards the packet to this VTC.

Enable routing by setting the RoutRxEn flag.

6.8.2.1.3 Selecting link interface

The SPW has two separate incoming SpaceWire links: a nominal and a redundant, which can be observed via the SelB flag. The interface is selected by the external *SpwIfSel* input. Interface A is selected when *SpwIfSel* is deasserted and interface B is selected when *SpwIfSel* is asserted.

6.8.2.2 Configuring the SpaceWire Link

The SpaceWire Link is configured by writing the SPW Link Handler Configuration Register.

After reset the SpaceWire link handler configuration is as follows:

- NULLs are transmitted at nominal rate.

In addition to this, the link is disabled and auto restart is disabled, cf. section 6.8.2.3.

Configuring the SpaceWire link handler can only be done during configuration.

6.8.2.2.1 Setting NULL transfer rate

The SPW provides two different transfer rates, default rate and nominal rate. Data is always transferred at nominal rate. To decrease the power consumption of the SPW when it is idle, NULL characters (which are transmitted when the SpaceWire link is idle) can be transmitted at a lower rate, the default rate.

The transfer rate of NULL characters is set to the nominal rate by clearing the TxDef4Null flag. The transfer rate of NULL characters is set to the default rate by setting the TxDef4Null flag.

6.8.2.3 Enabling and Disabling the SpaceWire Link

Enabling and disabling of the SpaceWire link is done by writing the SPW Link Handler Configuration Register.

The SpaceWire link can be enabled in two different ways: active start-up and automatic start-up. Active start-up enables the link directly while automatic start-up enables the link as soon as the SPW receives a NULL character on the SpaceWire link.

Automatic start-up is the preferred configuration for SCTMTC, since active start-up is typically used by a software controlled SpaceWire interface. Automatic start-up allows SCTMTC to wait until the link is started by the other unit. Note that both units on a SpaceWire link must not use automatic start-up, since this prevents the link from connecting.

Automatic restart allows the SpaceWire link to reconnect after a link disconnection. The function must be enabled for SCTMTC.

Enabling/disabling the SpaceWire link can only be done during configuration.

6.8.2.3.1 Active start-up

Active start-up of the SpaceWire link is selected by setting the LinkStart flag.

Note: Active start-up overrides automatic start-up if both are set.

6.8.2.3.2 Automatic start-up

Automatic start-up of the SpaceWire link is selected by setting the AutoLinkEn flag and clearing the LinkStart flag.

6.8.2.3.3 Automatic restart

The automatic restart function ensures that the values of AutoLinkEn and LinkStart flags are kept when reaching the Run state in order to allow the link to restart after being disabled. Automatic restart must be enabled for SCTMTC.

Enable automatic restart by setting the AutoRestartEn flag.

6.8.2.3.4 Link disable

Disabling the SpaceWire link forces the link FSM to immediately leave the Run state as soon as it is reached.

The link is disabled by setting the LinkDis flag.

Note: If the AutoRestartEn flag is set the link might start up directly after being disabled. If the LinkDis flag is not cleared before the link has reached the Run state again, the link is disabled once again etc.

6.8.2.4 Setting up Transmission Rate

To set the SpaceWire link transmission rate, write $2^{16} \cdot (N - 1) + (D - 1)$ to the SPW Clock Division Register.
N is the ratio between the *SpwClk* and the nominal transmission rate. $1 \leq N \leq 64$
D is the ration between the *SpwClk* and the default transmission rate. $1 \leq D \leq 64$

The nominal transmission rate is used to transmit data, time codes, and most control characters. The default rate is used to transmit NULL characters when the SPW Link Handler Configuration Register.TxDef4Null flag is set and during link start-up, it shall be set to 10 MHz ± 10 %.

To achieve a default transmission rate of 10 MHz ± 10 % the following table can be used:

***SpwClk* frequency (MHz)**	**Value for TxDefDiv**
0-8	n/a
9-11	0
12-17	n/a
18-22	1
23-26	n/a
27-33	2
34-35	n/a
36-44	3
45-54	4
55-64	5
65-70	6

Table 6-10 Values of TxDefDiv to achieve a 10 MHz transmission rate

The minimum allowed transmission rate is 2 MHz. If either tranmission rate is set lower then 2 MHz the SpaceWire link will time out.

The transmission rate can only be set during configuration.

6.8.2.5 Setting up SpaceWire Timings

To set up the time references for the SpaceWire link, write $2^{16} \cdot T_1 + T_2$ to the SPW Tick Number Register, where T_1 is the number of *SpwClk* cycles in 850 ns and T_2 is the number of *SpwClk* cycles in 12.8 µs.

To comply with SpaceWire timing constraints T_1 and T_2 must be selected as follows:
$727 \text{ ns} \leq T_1 \leq 1000 \text{ ns}$
$11.64 \text{ µs} \leq T_2 \leq 14.33 \text{ µs}$

The 850 ns time reference is used as a link time out. If the SpaceWire link inputs, *DIn* and *SIn*, are stable for 850 ns, the link is considered erroneous. This timeout sets the lower limit of the transmission rate on the SpaceWire link to 2 MHz.

The 12.8 µs time reference is used as a connection time out when setting up the Spacewire link after it has been disabled.
Note: The value 0 for any of the time references results in immediate time out.

The SpaceWire timings rate can only be set during configuration.

6.8.2.6 Configuring a DMA Read Channel

DMA read channel 0 is configured by writing the SPW Transmit Channel Configuration Register 0. Note: Only to be handled by the Control Interface (CI)

The DMA read channel can only be configured during configuration.

6.8.2.6.1 Packet splitting

When packet splitting is enabled, the SPW divides the data block into several blocks of length *N*. Each block is sent as a separate SpaceWire packet. To split a block into chunks of 2^N bytes each, set the SplitSize field to $N + 1$. $0 \leq N \leq 14$ To disable packet split, set the SplitSize field to 0.

Note: Packet splitting decreases the bandwidth of the SpaceWire link:

$$B_S = B_{Nom} \cdot \frac{(N_H + N_C) \cdot 10 + L_{EOP}}{(N_H \cdot \lceil \frac{N_C}{S} \rceil + N_C) \cdot 10 + L_{EOP}}$$

B_{Nom} = Bandwidth without split
B_S = Bandwidth with split
N_H = Number of header bytes
N_C = Number of cargo bytes
L_{EOP} = EOP end marker bits (= 4)
S = Number of cargo bytes in a split packet = $2^{\text{SplitSize} - 1}$

6.8.2.6.2 Adding packet header

VTC 0 can be configured to add a header of 1-8 bytes to every SpaceWire packet transferred on the VTC. Header bytes are also added to single EOPs or EEPs transmitted on VTC 0 since these are considered to be SpaceWire packets. When VTC 0 is set to send CCSDS packets the trailing EOP, denoting the end of a CCSDS packet is also preceded by the specified number of header bytes.

The header is primarily intended for sending routing bytes to allow SpaceWire packets from SCTMTC to be routed through a SpaceWire network.

To add a packet header to SpaceWire packets, do the following:

- Set the PktHeadEn flag
- Set the PktHeadLen field to $N - 1$, where N is the number of header bytes. $1 \leq N \leq 8$
- Write the first four header bytes to SPW Packet Header High Word Register 0.
- Write the last four header bytes to SPW Packet Header Low Word Register 0.

The packet header bytes should be placed as shown in Figure 6-7. The header bytes are transmitted in order of increasing index, i.e. header byte 0 is transmitted first, followed by header byte 1 etc. If less than 8 header bytes are used the ones with the lowest numbers are used.

SPW Packet Header High Word Register 0				SPW Packet Header Low Word Register 0			
MSB			LSB	MSB			LSB
Header Byte 0	Header Byte 1	Header Byte 2	Header Byte 3	Header Byte 4	Header Byte 5	Header Byte 6	Header Byte 7

Figure 6-7 Placement of packet header bytes in configuration registers

Destination address			**Cargo**			EOP
Header byte 0	Header byte 1	Header byte 2	Data byte 0	Data byte 1	...	

Figure 6-8 SpaceWire packet with three header bytes

To prevent the SPW from adding a packet header, clear the PktHeadEn flag.

6.8.2.6.3 Setting channel to send CCSDS packets

VTC 0 can be configured to treat a block of data as one or several CCSDS packets. The SPW sends each CCSDS packet in the data block as separate SpaceWire packets. When CCSDS packet sending is enabled, each CCSDS packet is always followed by an EOM, independent of the packet splitting configuration.

Set VTC 0 to treat send data as CCSDS packets by setting the CCSDSEn flag.

Disable CCSDS handling by clearing the CCSDSEn flag.

If packet splitting is enabled, the SPW splits the data block between CCSDS packets and at every 2^N byte. A SPW packet without cargo is sent after each CCSDS packet to indicate an end of message.

6.8.2.7 Configuring a Parallel Write Interface

A parallel write interface needs to be configured regarding how it treats EOP and EEP markers. Each parallel write interface can be configured to regard either the end of a single SpaceWire packet or a SpaceWire packet without cargo as an end of message.

Configure parallel interface $ to treat single end of packet markers as end of message by setting the SPW Parallel Write Configuration Register $.PPEOMDis.

Configure parallel interface $ to only treat double end of packet markers as end of message by clearing the SPW Parallel Write Configuration Register $.PPEOMDis.

Treating double EOP markers as an end of message is only useful when the sender uses packet splitting.

The parallel write interfaces can only be configured during configuration.

6.8.2.8 Receiving Time Code

The SPW issues the RxTimeCode interrupt when a new time code is received. When receiving this interrupt, get the received time code by reading the SPW Receive Time Code Register.

6.8.2.9 Checking Status

The SPW status contains information about which channels are active on the SpaceWire link. Check the SPW status by reading the SPW Status Register.

6.8.2.10 Checking SpaceWire Link Status

Check the SpaceWire link status by reading the SPW Link Status Register. The status register contains the following:

- SpaceWire link state
- Receive buffer full flag
- Transmit buffer full flag

6.8.2.11 Resolving Congestion

If for some reason one of the recipients does not respond when the SPW tries to send data, the flow control on the SpaceWire link will halt all transfers. The SPW provides a mechanism to use link errors, e.g. disconnect, for aborting transfers on the parallel interfaces, thus resolving the congestion. When using this method, all VRCs will enter a data consumption mode, where no incoming data is forwarded to the destination until an EOP or EOM, depending on the configuration of the VRCs, has been received in a packet routed to that particular VRC.

To resolve a congestion situation, do the following:

- Force a disconnection error from the transmitter on the SpaceWire link
- Wait until the SpaceWire link has returned to Run state
- Stop tranmitting the packets that were likely to be partly contained in the receive buffer
- If any part of a packet resides in the sender's transmit buffer, ensure that this data is either not sent or ended with an EEP
- Retransmit any lost packets
- Continue transmitting data

6.8.3 Error Handling

6.8.3.1 SpaceWire Link Errors

If an error occurs on the SpaceWire link, the SPW does the following:

- Leave the Run state
- Issue one of the following interrupts:
  - CrErr Credit error detected (receive buffer overflow)
  - ESCErr Invalid character sequence on the link
  - ParErr Parity error
  - DisErr Disconnection error (inputs have not changed for 850 ns)
- Issues a Tx0 interrupt, if VTC 0 is active
- Empties the receive buffer if it contains data
- Aborts transfer on the active VRC

If a packet is being received at the time of the link error the transfer on that particular VRC is aborted and the VRC consumes all data from the receive buffer until it receives either an EOM or EOP, depending on how the VRC is configured, or an EEP. Since the EEP is inserted automatically the sender does not have to cope with this. Note: If the link is configured to restart automatically, this is done immediately.

6.8.3.2 Address Error

If the SPW receives a packet and default routing is enabled and the default routing address is higher than 7 or default routing is disabled and the first byte received is higher than 7, the SPW does the following:

- Issue the AddrErr interrupt
- Discard the entire packet

6.8.3.3 EEP Marker Reception

An EEP marker indicates that the packet ends erroneously. The cause could be a link error or an error at the transmitter.

If the SPW receives an EEP marker at the end of a packet destined for VRC $, the SPW does the following:

- Abort the transfer on the parallel write interface

6.8.3.4 CCSDS Packet Length Error

If VTC is configured to treat data blocks as CCSDS packets, the SPW considers a CCSDS packet to be as long as defined by the length field in the header. If the SPW reaches the end of the block and the current CCSDS packet is supposed to contain more data, the SPW does the following:

- Issues the Tx interrupt
- End the channel transfer and insert an EEP marker

6.8.3.5 DMA Read Error

If the SPW receives a DMA error while VTC is reading data from memory, the SPW does the following:

- Issues the Tx interrupt
- End the channel transfer and insert an EEP marker

Note: The erroneous byte is not transmitted on the SpaceWire link.

6.8.4 Usage Constraints

6.8.4.1 Functional

The following apply to systems with routers:

- Do not configure the SPW modules in the system to use default routing and add no header to SpaceWire packets if CCSDS packets are used.

  Routers may remove single EOP and EEP markers. Since a CCSDS packet is terminated with a packet without cargo, this can be removed if it does not contain a destination address.

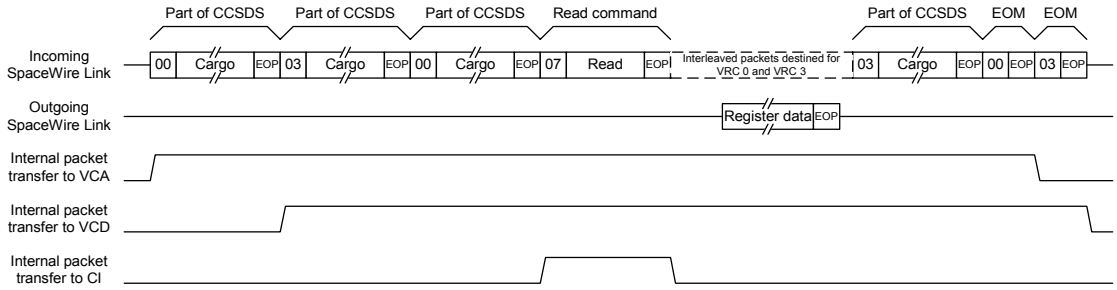
6.8.5 Examples

6.8.5.1 Receiving Interleaved Packets

Two large CCSDS packets containing 7,000 bytes each destined for VCA and VCD of the TME are to be received. SpaceWire is used as control interface and a control interface packets must be allowed to be sent rather quickly, therefore packet splitting is used for the two larger packets since it is a simple system without routers. VRC 7, which is connected to the Control Interface does not expect CCSDS packets. No header is added to the data sent from SCTMTC.

1. Configure VRC 0 and VRC 3 to treat EOMs as end of CCSDS packet by clearing SPW Parallel Write Configuration Register *$*.PPEOMDis.
2. Configure VRC 7 to treat singel EOPs as end of packet by setting SPW Parallel Write Interface 7.PPEOMDis.
3. Disable default routing by setting SPW Configuration Register.RoutRxEn to allow packets to be sent to several VRCs.
4. Select SpaceWire as control bus by setting CAR CtrlIf Select Register.Sel.
5. On the sending end, which is not the SCTMTC ASIC:
   Enable packet split for the transmit channels that are to send the large packets to the TME.
   Ensure that packet splitting is not enabled for control interface packets.
   Enable double EOPs as end of message for the TME packets.
   Enable single EOP as end of message for the control interface packets.

6. Start sending the large TME packets on the sending end. The SPW will start receiving the packets and route them to the correct destination. The sender can easily switch between sending the two packets at any time by terminating the current data stream with a singel EOP. The CCSDS packets are not considered to be fully transmitted until a packet without cargo is routed to the same VRC.
7. At some moment a Control Interface packet must be sent due to a high priority interrupt requiring a readout of SCTMTC status. Start sending the packet, which will be interleaved with the SpaceWire packets constituting the CCSDS packets.
8. SPW will receive the Control Interface packet and route it to Control Interface, where it is decoded and recognised as a read command.
9. Control Interface sets up a read transfer in the SPW to read out the registers.
10. While the reception of the large packets are still in progress the transfer of the read data commences. Allowing for the sending end to receive the register values prior to the end of its own transfer.



6.9 Control Interface Module (CI)

6.9.1 Initialisation

The following steps should be performed by the configuration block at power-up to initialise the Control Interface, i.e. all values below should be set in the initialisation PROM (cf. section 6.11.1):

- Set the period of the transmission clock through PWT Clock Configuration Register
- Enable interrupts through CI Interrupt Mask Register and PWT Interrupt Mask Register

If the SpaceWire link is used for sending packets to the Control Interface it need to be set up for the Control Interface to work properly.

If it is used for control, initialise the SpaceWire module as follows (performed by the configuration block, cf. section 0):

- Enable routing to remove the routing header of SpaceWire packets
- Set up transmission rates and SpaceWire timings
- Enable the link; ensure that the AutoRestartEn and either of LinkStart or AutoLinkEn are set

AutoRestartEn should be set to allow the link to reconnect after a disconnection error. If the AutoRestartEn flag is not set it is not possible to force a disconnection error to resolve congestion on the SpaceWire link.

It is strongly recommended to disable End of Message detection for parallel write interface number 7 in the SpaceWire module. If End of Message detection is enabled, the Control Interface packets will not end until a packet without cargo, i.e. with only the routing header directing the packet to the Control Interface, arrives through the SpaceWire link. If parallel write interface 7 is left unconfigured End of Message detection is enabled, which requires the sender to always end a Control Interface packet with double EOPs, where the second EOP shall have only a routing byte. This configuration prevents splitting of Control Interface packets.

Note: It is not possible to communicate with the Control Interface using SpaceWire if the SpaceWire module is not properly configured.

6.9.2 Operation/Usage

6.9.2.1 Interface Selection

The Control Interface selects between the two interfaces by means of the Sel bit in CAR CtrlIf Select Register (cf. section 6.1.3.5). The unselected interface is kept ready and receives all data sent to it to prevent deadlock.

Interface is selected during configuration and can not be changed.

6.9.2.2 Write Access

A packet containing a write access consists of a header, which defines the destination address and the data size, and data. The Control Interface writes the amount of data specified by the size field in the packet header to memory; if data remains in the packet, it is discarded. If a packet contains less data than the packet header specifies, the Control Interface stops writing at the end of packet.

The Control Interface only accepts word-aligned writes containing complete 32-bit words, this is done by ignoring the two LSBs of both the address field and the size field of the packet header. If the address is not word-aligned the write takes place on the first word-aligned address lower than the actual address. If the number of bytes is not divisible by four the bytes in the uncompleted 32-bit word are discarded.

Data is written to memory in sequence starting with the first byte on the address specified by the Address field.

The Control Interface does not provide any automatic mechanism for continuing a write where the previous stopped.

6.9.2.2.1 Write access through Packet Wire interface

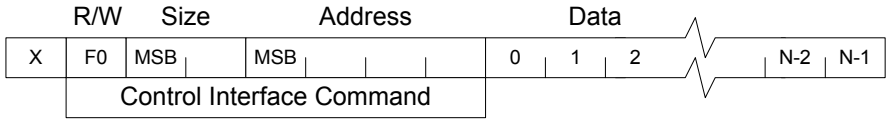


Figure 6-9 Packet Wire write packet

To perform a write access through the Packet Wire interface, set up a packet as follows:

- The first byte is ignored by the Control Interface and can be used as routing information by the application
- The RW field shall be set to $F0_{16}$ to indicate a write access
- The Size field shall be set to the number of bytes in the packet data field, *N*
  Most significant bit is sent first
- The Address field shall point to the word-aligned destination address in memory where the first data byte is stored
  Most significant bit is sent first
- The data field contains the write data, the number of bytes shall be divisible by four and equal the number specified in the Size field
  The first data byte is written to byte address 0, the second to byte address 1, etc.

The fields of a write packet on the Packet Wire interface are defined in Table 6-11.

Field	Description
Dummy byte	Discarded
RW	Read/write flag, set to $F0_{16}$ for write accesses
Size	Number of data bytes in the packet, only complete words are accepted
Address	Start address, word-aligned
Data	Data bytes

Table 6-11 Write access

6.9.2.2.2 Write access through SpaceWire interface

Packets sent to the Control Interface through the SpaceWire interface are transferred to the Control Interface through an internal parallel interface. The SpaceWire packets should have the structure shown in Figure 6-10. The first byte is used as routing information by the SpaceWire module and is removed before the packet is sent to the Control Interface.

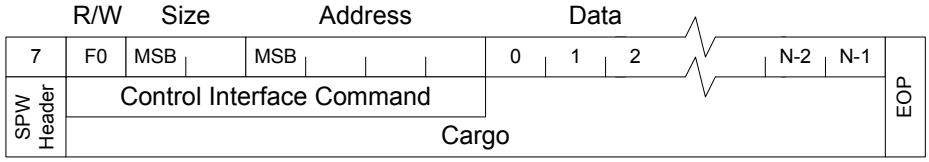


Figure 6-10 SpaceWire write packet

To perform a write access through the SpaceWire interface, set up a packet as follows:

- The first byte shall be set to 7 to route the packet to the Control Interface
- The RW field shall be set to $F0_{16}$ to indicate a write access
- The Size field shall be set to the number of bytes in the packet data field, *N*
  Most significant bit is sent first
- The Address field shall point to the word-aligned destination address in memory where the first data byte is stored
  Most significant bit is sent first
- The data field contains the write data, the number of bytes shall be divisible by four and equal the number specified in the Size field
  The first data byte is written to byte address 0, the second to byte address 1, etc.

The fields of a write packet on the SpaceWire interface are defined in Table 6-12.

Field	Description
First byte	Routing information to the SpaceWire module
RW	Read/write flag, set to $F0_{16}$ for write accesses
Size	Number of data bytes in the packet, only complete words are accepted
Address	Start address, word-aligned
Data	Data bytes

Table 6-12 Write access

6.9.2.3 Read Access

A packet containing a read access consists of a header only. The header defines the address to the requested block and the number of bytes to read. The Control Interface does not perform the read itself; it selects between two senders, depending on the origin of the packet.

Data is transmitted through the interface that is compatible with the interface used for sending the read access. The read data is transferred in sequence starting with the byte with the lowest byte address.

6.9.2.3.1 Read access through Packet Wire interface

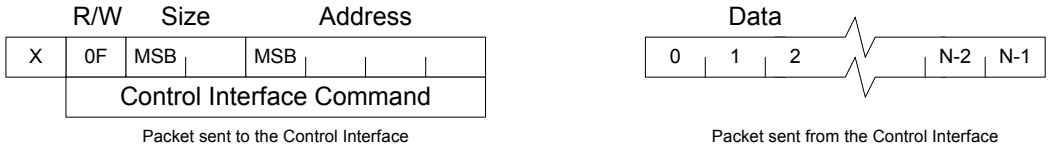


Figure 6-11 Packet Wire read packet and result

To perform a read access through the Packet Wire interface, set up a packet as follows:

- The first byte is ignored by the Control Interface and can be used as routing information by the application
- The RW field shall be set to $0F_{16}$ to indicate a read access
- The Size field shall be set to the number of bytes that are to be read
  Most significant bit is sent first
- The Address field shall point to the word-aligned destination address in memory where read starts
  Most significant bit is sent first

The fields of a read packet on the Packet Wire interface are defined in Table 6-13.

Field	Description
Dummy byte	Discarded
RW	Read/write flag, set to $0F_{16}$ for read accesses
Size	Number of data bytes to read, only complete words are read
Address	Start address, two LSBs are ignored

Table 6-13 Read access

Data is read from memory and transmitted on a Packet Wire link. The data bytes are transferred in sequence starting with the lowest byte address.

6.9.2.3.2 Read access through SpaceWire interface

Packets sent to the Control Interface through the SpaceWire interface are transferred to the Control Interface through an internal parallel interface. The SpaceWire packets should have the structure shown in Figure 6-12. The first byte is used as routing information by the SpaceWire module and is removed before the packet is sent to the Control Interface.

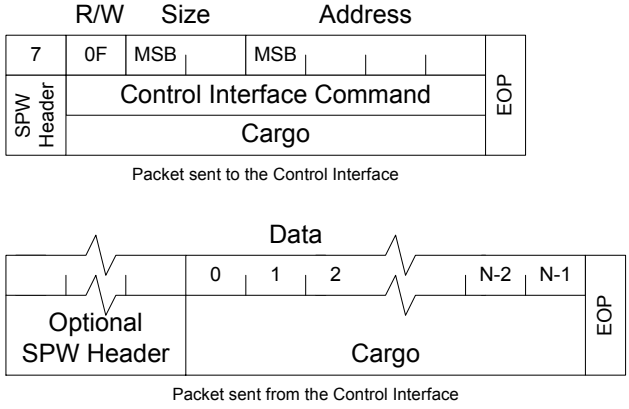


Figure 6-12 SpaceWire read packet and result

To perform a read access through the SpaceWire interface, set up a packet as follows:

- The first byte shall be set to 7 to route the packet to the Control Interface
- The RW field shall be set to $0F_{16}$ to indicate a read access
- The Size field shall be set to the number of bytes that are to be read
  Most significant bit is sent first
- The Address field shall point to the word-aligned destination address in memory where read starts
  Most significant bit is sent first

The fields of a read packet on the parallel interface are defined in Table 6-14.

Field	Description
First byte	Routing information to the SpaceWire module
RW	Read/write flag, set to $0F_{16}$ for read accesses
Size	Number of data bytes to read, only complete words are read
Address	Start address, two LSBs are ignored

Table 6-14 Read access

Data is read from memory and transmitted on a SpaceWire link. The data bytes are transferred in sequence starting with the lowest byte address.

6.9.2.4 Packet Wire Transmit Clock Configuration

The transmission clock period is programmable and is usually configured at start-up. The period of the transmission clock, *CiOutClk*, is:

$T_{CiOutClk} = 2 \cdot (1 + Period) \cdot T_{BusClk}$

where:

Period = Value of PWT Clock Configuration Register.Period.
Period has a value between 0 and 255.

$T_{CiOutClk}$ = Period of *CiOutClk*.

T_{SysClk} = Period of the system clock.

A table with the relations between Period and the frequency of *CiOutClk* at different frequencies of the system clock is found below. The frequencies of *CiOutClk* in the table are not exact.

f_{SysClk} = 20 MHz		f_{SysClk} = 16 MHz		f_{SysClk} = 10 MHz	
$f_{CiOutClk}$	**Period**	**$f_{CiOutClk}$**	**Period**	**$f_{CiOutClk}$**	**Period**
39 kHz	255	31.25 kHz	255	19.5 kHz	255
50 kHz	199	50 kHz	159	50 kHz	99
100 kHz	99	100 kHz	79	100 kHz	49
200 kHz	49	200 kHz	39	200 kHz	24
500 kHz	19	500 kHz	15	500 kHz	9
1 MHz	9	1 MHz	7	1 MHz	4
2 MHz	4	2 MHz	3	1.25 MHz	3
3.33 MHz	2	2.67 MHz	2	1.67 MHz	2
5 MHz	1	4 MHz	1	2.5 MHz	1
10 MHz	0	8 MHz	0	5 MHz	0

Table 6-15 Period as a function of *CiOutClk* frequency and *SysClk* frequency

6.9.2.5 Manual Packet Wire transfer

To manually perform a transfer through the Packet Wire output interface, do the following:

- If necessary, write a data block to memory
- Set up the base address of the block by writing PWT DMA Base Address Register
- Set up the size of the block by writing PWT DMA Block Size Register
  This resets PWT DMA Offset Register and starts the transfer.

Data is read through the internal bus and sent on the Packet Wire link until the entire block has been transferred. During the transfer the PWT Status Register.DMA Block flag is set.

6.9.2.6 Splitting SpaceWire write packets

It is possible but not recommended to configure the SpaceWire module to allow Control Interface packets to be split into several SpaceWire packets. Since read packets are very short this feature is only interesting when using the Control Interface to write large memory areas with a single packet. Packet splitting for the Control Interface is enabled at the receiving end by enabling EOM detection for parallel write interface number 7 in the SpaceWire module. When EOM detection is enabled the SpaceWire module will not end the packet transmission towards the Control Interface until it receives an empty SpaceWire packet routed to the Control Interface (Virtual Receive Channel 7).

6.9.3 Error Handling

6.9.3.1 Illegal Access Type

If a packet containing an illegal access is received, the Control Interface does the following:

- Receives the entire packet and discards all data
- Issues the Illegal interrupt

6.9.3.2 Memory Access Error

6.9.3.2.1 Error during memory write

If an error occurs during memory access, the Control Interface does the following:

- Receives the remaining bytes of the packet and discards them
- Issues the BusError interrupt

During write accesses memory access error might occur if the address is mapped in read-only memory, if the memory area is not mapped at all, etc.

6.9.3.2.2 Error during memory read through SpaceWire

If an error occurs during a memory read through the SpaceWire interface, the SpaceWire module does the following:

- Stops reading memory
- Terminates the packet with an EEP control character
- Sets the transmit channel status DmaRdErr flag and issues the Tx0 interrupt

Errors can occur during a read access if the memory area is not mapped, if an EDAC error occurs, etc.

6.9.3.2.3 Error during memory read through Packet Wire

If an error occurs during a memory read through the Packet Wire interface, the Packet Wire Transmitter does the following:

- Stops reading memory
- Truncates the packet
- Issues the DMA Error interrupt

6.9.3.3 Erroneous Length

A packet is of erroneous length if it does not contain a complete header or if the number of complete 32-bit words in the packet does not match the expected number of complete 32-bit words.

6.9.3.3.1 Incomplete packet header

If the packet is over before the entire packet header has been received, the Control Interface does the following:

- Discards the packet
- Issues the Incomplete interrupt

6.9.3.3.2 Too short packets

If a packet contains less data than specified by the Size field in the packet header (adjusted to complete 32-bit words by ignoring the two LSBs), the Control Interface does the following:

- Writes all received complete words
- Issues the Incomplete interrupt

The Control Interface does not detect that a packet is too short until all data have been written to memory.

6.9.3.3.3 Too long packets

If a packet contains more data than specified by the Size field in the packet header (adjusted to complete 32-bit words by ignoring the two LSBs), the Control Interface does the following:

- Writes complete words to memory until the expected number of words have been written
- Receives the remaining bytes of the packet and discards them
- Issues the Overflow interrupt

Example: If the Size field is set to 43_{16} bytes the Control Interface expects 40_{16} bytes, i.e. 10_{16} 32-bit words. If 41_{16} bytes are received the Control Interface still regards the packet as being too long and issues the Overflow interrupt.

6.9.4 Usage Constraints

6.9.4.1 Functional

- Do not try to access an address that is not word-aligned.

  Should this be done, the Control Interface word-aligns the address, which results in the first lower word-aligned address being accessed instead of the intended.

- Do not try to read or write data sizes that are not divisible by four (i.e. contains only 32-bit words.)

  Should this be done, the Control Interface ignores the remaining bytes in the incomplete word, resulting in the access being shorter than intended. For write accesses the Overflow flag is set.

- Do not send a read packet before the previous read data has been received via the SpaceWire or Packet Wire interface.

  Should this be done, the first packet might be corrupted and the second packet lost.

6.9.5 Examples

6.9.5.1 Register Read

The registers at byte address 7000220_{16} and 7000224_{16} are read through the SpaceWire link. The first register contains 12345678_{16} and the second register contains $AABBCCDD_{16}$.

The following packet is set up and sent through the SpaceWire interface:

	R/W	Size		Address			
07	0F	00	08	07	00	02	20

The registers are read and the following result is transmitted on the SpaceWire link:

Data							
12	34	56	78	AA	BB	CC	DD

6.9.5.2 Register Write

The two registers on byte address 7000220_{16} and 7000224_{16} are written in a single access over the Packet Wire interface with the values 12345678_{16} and $AABBCCDD_{16}$ respectively.

The following packet is set up and sent through the Packet Wire interface:

	R/W	Size		Address				Data							
00	F0	00	08	07	00	02	20	12	34	56	78	AA	BB	CC	DD

6.10 Register Definition Summary

The registers in the tables hereafter are grouped by modules and blocks. The detailed definition of each register is provided in the subsequent sections. Note that those definitions are made from a user perspective, i.e. how they can be accessed from the Control Interface (CI). The read or write capability of a register is indicated after the register name in the subsequent sections with an R or W, respectively. Some registers are listed as only readable, although they are possible to write during the configuration sequence as described in 6.11.1.

Unused register bits are denoted '-'; these should be ignored on register reads and set to zero on register writes.

All registers inside the SCTMTC ASIC are 32 bits wide.

Clock and Reset block

Byte address	Register name	Acronym
0700_0824_{16}	CAR Arm Clock Source Register	[CAR_ArmClkSrc]
0700_0820_{16}	CAR Clock Source Register	[CAR_ClkSrc]
0700_0880_{16}	CAR TM Clock Source Register	[CAR_TmClkSrc]
0700_082C_{16}	CAR Arm Clock Enable Register	[CAR_ArmClkEna]
0700_0828_{16}	CAR Clock Enable Register	[CAR_ClkEna]
0700_0834_{16}	CAR Arm Reset Register	[CAR_ArmModReset]
0700_0830_{16}	CAR Reset Register	[CAR_ModReset]
0700_083C_{16}	CAR Arm Pdec3 Csel Connect Register	[CAR_ArmCselCon]
0700_0838_{16}	CAR Pdec3 Csel Connect Register	[CAR_CselCon]
0700_0854_{16}	CAR CtrlIf Select Register	[CAR_CtrlIfSel]
0700_0890_{16}	CAR Power On Reset Register	[CAR_PwrOnRst]

Configuration Block

Byte address	Register name	Acronym
0700_3044_{16}	Conf External Interrupt Register	[CNF_ExtIrqR]
0700_4000_{16}	GenClk Pending Interrupt Masked Status Register	[GEN_PIMSR]
0700_4004_{16}	GenClk Pending Interrupt Masked Register	[GEN_PIMR]
0700_4008_{16}	GenClk Pending Interrupt Status Register	[GEN_PISR]
0700_400C_{16}	GenClk Pending Interrupt Register	[GEN_PIR]
0700_4010_{16}	GenClk Interrupt Mask Register	[GEN_IMR]
0700_4060_{16}	GenClk SW Start Register	[GEN_START]
0700_4080_{16}	GenClk 1 Control Register	[GEN_1CTRL]
0700_4084_{16}	GenClk 1 Control Set Register	[GEN_1CTRLSET]
0700_4088_{16}	GenClk 1 Control Clear Register	[GEN_1CTRLCLR]
0700_408C_{16}	GenClk 1 Period Register	[GEN_1PERIOD]
0700_4090_{16}	GenClk 1 Assert Register	[GEN_1ASSERT]
0700_4094_{16}	GenClk 1 Deassert Register	[GEN_1DEASSERT]
0700_4098_{16}	GenClk 1 Counter Register	[GEN_1COUNTER]
0700_40A0_{16}	GenClk 2 Control Register	[GEN_2CTRL]
0700_40A4_{16}	GenClk 2 Control Set Register	[GEN_2CTRLSET]
0700_40A8_{16}	GenClk 2 Control Clear Register	[GEN_2CTRLCLR]
0700_40AC_{16}	GenClk 2 Period Register	[GEN_2PERIOD]
0700_40B0_{16}	GenClk 2 Assert Register	[GEN_2ASSERT]
0700_40B4_{16}	GenClk 2 Deassert Register	[GEN_2DEASSERT]

Byte address	Register name	Acronym
0700_40B8_{16}	GenClk 2 Counter Register	[GEN_2COUNTER]
0700_40C0_{16}	GenClk 3 Control Register	[GEN_3CTRL]
0700_40C4_{16}	GenClk 3 Control Set Register	[GEN_3CTRLSET]
0700_40C8_{16}	GenClk 3 Control Clear Register	[GEN_3CTRLCLR]
0700_40CC_{16}	GenClk 3 Period Register	[GEN_3PERIOD]
0700_40D0_{16}	GenClk 3 Assert Register	[GEN_3ASSERT]
0700_40D4_{16}	GenClk 3 Deassert Register	[GEN_3DEASSERT]
0700_40D8_{16}	GenClk 3 Counter Register	[GEN_3COUNTER]

Memory Interface

Byte address	Register name	Acronym
0700_0000_{16}	PIM Pending Interrupt Masked Status Register	[PIM_PIMSR]
0700_0004_{16}	PIM Pending Interrupt Masked Register	[PIM_PIMR]
0700_0008_{16}	PIM Pending Interrupt Status Register	[PIM_PISR]
0700_000C_{16}	PIM Pending Interrupt Register	[PIM_PIR]
0700_0010_{16}	PIM Interrupt Mask Register	[PIM_IRQMR]
0700_0020_{16}	PIM Scrubber Configuration Register	[PIM_ScuCNFR]
0700_0024_{16}	PIM Scrubber Start Address Register	[PIM_ScuSAR]
0700_0028_{16}	PIM Scrubber End Address Register	[PIM_ScuEAR]
0700_002C_{16}	PIM TME Ratio Register	[PIM_TmeRat]
0700_0040_{16}	PIM Status Register	[PIM_SR]
0700_0044_{16}	PIM Scrubber Address Register	[PIM_ScuAR]
0700_0048_{16}	PIM First Failing Correctable Error Address Register	[PIM_FFCEAR]
0700_004C_{16}	PIM Write Disable Status Register	[PIM_WrDis]
0700_0060_{16}	PIM Status Set Register	[PIM_SRSet]
0700_0064_{16}	PIM Status Clear Register	[PIM_SRClr]
0700_0068_{16}	PIM Write Disable Set Register	[PIM_WrDisSet]
0700_006C_{16}	PIM Write Disable Clear Register	[PIM_WrDisClr]
0700_0070_{16}	PIM FlushO Register	[PIM_FLUSHO]
0700_0074_{16}	PIM FlushT Register	[PIM_FLUSHT]
0700_0080_{16}	PIM Error Trap Register	[PIM_ET]
0700_0084_{16}	PIM Error Info Register	[PIM_EI]
0700_0088_{16}	PIM First Failing Address Register	[PIM_FFAR]
0700_008C_{16}	PIM First Failing Data Register	[PIM_FFDR]
0700_0090_{16}	PIM EDAC Check Error Inject Register	[PIM_ECEI]
0700_00C0_{16}	PIM MUnit 0 Register (*PromCsN*)	[PIM_MUnit0]
0700_00C8_{16}	PIM MUnit 2 Register (*ExtRecLacN*)	[PIM_MUnit2]
0700_00CC_{16}	PIM MUnit 3 Register (*RamCsN*)	[PIM_MUnit3]
0700_011C_{16}	PIM MUnit User Area 1 Address Low Register	[PIM_MUSgm1L]
0700_0120_{16}	PIM MUnit User Area 2 Address Low Register	[PIM_MUSgm2L]
0700_0124_{16}	PIM MUnit TME / PM Address Low Register	[PIM_MURamL]
0700_0140_{16}	PIM MUnit 0 Address High Register (*PromCsN*)	[PIM_MU0H]
0700_0148_{16}	PIM MUnit 2 Address High Register (*ExtRecLacN*)	[PIM_MU2H]
0700_015C_{16}	PIM MUnit User Area 1 Address High Register	[PIM_MUSgm1H]
0700_0160_{16}	PIM MUnit User Area 2 Address High Register	[PIM_MUSgm2H]
0700_0164_{16}	PIM MUnit TME / PM Address High Register	[PIM_MURamH]

Packet Telecommand Decoder (PDEC3)

Byte address	Register name	Acronym
0700_2800_{16}	PDEC3 Pending Interrupt Masked Status	[PDEC_PIMSR]
0700_2804_{16}	PDEC3 Pending Interrupt Masked	[PDEC_PIMR]
0700_2808_{16}	PDEC3 Pending Interrupt Status	[PDEC_PISR]
0700_280C_{16}	PDEC3 Pending Interrupt	[PDEC_PIR]

Released

Byte address	Register name	Acronym
0700_2810_{16}	PDEC3 Interrupt Mask	[PDEC_IMR]
0700_2840_{16}	PDEC3 Frame Analysis Report	[PDEC_FAR]
0700_2844_{16}	PDEC3 Code Block Counter	[PDEC_CBCNT]
0700_2848_{16}	PDEC3 Command Link Control Word	[PDEC_CLCW]
0700_2850_{16}	PDEC3 Map Link Address	[PDEC_MAPADR]
0700_2854_{16}	PDEC3 General Map Interface Status	[PDEC_MAPGSTAT]
0700_2858_{16}	PDEC3 Map Interface 1 Status	[PDEC_MAP1STAT]
0700_285C_{16}	PDEC3 Map Interface 2 Status	[PDEC_MAP2STAT]
0700_2860_{16}	PDEC3 Map Interface 3 Status	[PDEC_MAP3STAT]
0700_2864_{16}	PDEC3 Map Interface 4 Status	[PDEC_MAP4STAT]
0700_2868_{16}	PDEC3 Map Interface 5 Status	[PDEC_MAP5STAT]
0700_286C_{16}	PDEC3 Common Interface Status	[PDEC_MAPCSTAT]
0700_2880_{16}	PDEC3 Authentication Status Report 1	[PDEC_AUSR1]
0700_2884_{16}	PDEC3 Authentication Status Report 2	[PDEC_AUSR2]
0700_2888_{16}	PDEC3 Authentication Status Report 3	[PDEC_AUSR3]
0700_2890_{16}	PDEC3 Buffer Free	[PDEC_BFREE]
0700_2894_{16}	PDEC3 Buffer Pointer	[PDEC_BPTR]
0700_2898_{16}	PDEC3 Segment Length	[PDEC_SLEN]
0700_289C_{16}	PDEC3 Monitor	[PDEC_MON]
0700_28A0_{16}	PDEC3 Frame Analysis Report Copy	[PDEC_FARCP]
0700_28A4_{16}	PDEC3 Code Block Counter Copy	[PDEC_CBCNTCP]
0700_28A8_{16}	PDEC3 Authentication Status Report Copy 1	[PDEC_AUSRCP1]
0700_28AC_{16}	PDEC3 Authentication Status Report Copy 2	[PDEC_AUSRCP2]
0700_28B0_{16}	PDEC3 Authentication Status Report Copy 3	[PDEC_AUSRCP3]

Released

External CPDU Interface (ExtCpduIf)

Byte address	Register name	Acronym
0700_1440_{16}	ExtCpduIf Status Register	[XRM_STAT]
0700_6000_{16}	ExtCpduIf DMA BaseAddress Register	[XRM_DBAR]
0700_6004_{16}	ExtCpduIf DMA Offset Register	[XRM_DOR]
0700_6008_{16}	ExtCpduIf DMA Data Register	[XRM_DDR]
0700_6014_{16}	ExtCpduIf DMA Max Offset Register	[XRM_DMOR]

CPDM Selector (CSEL)

Byte address	Register name	Acronym
0700_1000_{16}	CSEL Pending Interrupt Masked Status Register	[CS_PIMSR]
0700_1004_{16}	CSEL Pending Interrupt Masked Register	[CS_PIMR]
0700_1008_{16}	CSEL Pending Interrupt Status Register	[CS_PISR]
0700_100C_{16}	CSEL Pending Interrupt Register	[CS_PIR]
0700_1010_{16}	CSEL Interrupt Mask Register	[CS_IMR]
0700_1020_{16}	CSEL 1 Second Register	[CS_1SEC]
0700_1024_{16}	CSEL Max TC Only Time Register	[CS_MAXTC]
0700_1028_{16}	CSEL Remote Status Timeout Time Register	[CS_RSTT]
0700_1040_{16}	CSEL Status Register	[CS_STAT]
0700_1044_{16}	CSEL Remote Status Register	[CS_RSTAT]
0700_1080_{16}	CSEL PM Request Register	[CS_PMREQ]
0700_1084_{16}	CSEL PM Buffer Pointer Register	[CS_PMBUFF]
0700_1088_{16}	CSEL PM Segment Length Register	[CS_PMLEN]

Command Pulse Distribution Module (CPDM)

Byte address	Register name	Acronym
0700_0C00_{16}	CPDM Pending Interrupt Masked Status Register	[CPDM_PIMSR]

Byte address	Register name	Acronym
0700_0C04_{16}	CPDM Pending Interrupt Masked Register	[CPDM_PIMR]
0700_0C08_{16}	CPDM Pending Interrupt Status Register	[CPDM_PISR]
0700_0C0C_{16}	CPDM Pending Interrupt Register	[CPDM_PIR]
0700_0C10_{16}	CPDM Interrupt Mask Register	[CPDM_IMR]
0700_0C20_{16}	CPDM Pulse Duration Register	[CPDM_PD]
0700_0C24_{16}	CPDM Clock Ratio Register	[CPDM_CR]
0700_0C28_{16}	CPDM Configuration Register	[CPDM_CNF]
0700_0C2C_{16}	CPDM Lockout Register	[CPDM_LCK]
0700_0C40_{16}	CPDM Status Report Register	[CPDM_SRR]
0700_0C44_{16}	CPDM Status Register	[CPDM_SR]

Packet Telemetry Encoder (TME)

Byte address	Register name	Acronym
0700_3C00_{16}	TME TM Encoder Pending Interrupt Masked Status Register	[TME_PIMSR]
0700_3C04_{16}	TME TM Encoder Pending Interrupt Masked Register	[TME_PIMR]
0700_3C08_{16}	TME TM Encoder Pending Interrupt Status Register	[TME_PISR]
0700_3C0C_{16}	TME TM Encoder Pending Interrupt Register	[TME_PIR]
0700_3C10_{16}	TME TM Encoder Interrupt Masked Register	[TME_IMR]
0700_3C20_{16}	TME TM Encoding Configuration Register 0	[TME_TmEnCfg0]
0700_3C24_{16}	TME TM Encoding Configuration Register 1	[TME_TmEnCfg1]
0700_3C28_{16}	TME TM Identification Configuration Register	[TME_TmIdCfg]
0700_3C60_{16}	TME Initialise TM Register	[TME_InitTm]
0700_3C80_{16}	TME Bandwidth Allocation Table Register 0	[TME_BAT0]
0700_3C84_{16}	TME Bandwidth Allocation Table Register 1	[TME_BAT1]
0700_3C88_{16}	TME Bandwidth Allocation Table Register 2	[TME_BAT2]
0700_3C8C_{16}	TME Bandwidth Allocation Table Register 3	[TME_BAT3]
0700_3CA0_{16}	TME VC Input Configuration Registers A	[TME_VcCfgA]
0700_3CA4_{16}	TME VC Input Configuration Registers B	[TME_VcCfgB]
0700_3CA8_{16}	TME VC Input Configuration Registers C	[TME_VcCfgC]
0700_3CAC_{16}	TME VC Input Configuration Registers D	[TME_VcCfgD]
0700_3CB0_{16}	TME VC Input Configuration Registers E	[TME_VcCfgE]
0700_3CB4_{16}	TME VC Input Configuration Registers F	[TME_VcCfgF]
0700_3CB8_{16}	TME VC Input Configuration Registers G	[TME_VcCfgG]
0700_3CBC_{16}	TME VC Input Configuration Registers H	[TME_VcCfgH]
0700_3CE0_{16}	TME VCID Configuration Register 0	[TME_VcIdCfg0]
0700_3CE4_{16}	TME VCID Configuration Register 1	[TME_VcIdCfg1]
0700_3CF0_{16}	TME VC Buffer Status Register 0	[TME_VCBufStat0]
0700_3CF4_{16}	TME VC Buffer Status Register 1	[TME_VCBufStat1]
0700_3D30_{16}	TME TFM Control Register	[TME_TfmCtrl]
0700_3D40_{16}	TME TFM SubC Divisor Register	[TME_TfmSubCDiv]
0700_3D44_{16}	TME TFM Bit Rate Divisor Register	[TME_TfmBRDiv]
0700_3D48_{16}	TME TFM Commit Register	[TME_TfmCommit]
0700_C800_{16}	TME DMA Base Address Register	[TME_TmeDBAR]

SpaceWire (SPW)

Byte address	Register name	Acronym
0700_3400_{16}	SPW Pending Interrupt Masked Status Register	[SPW_PIMSR]
0700_3404_{16}	SPW Pending Interrupt Masked Register	[SPW_PIMR]
0700_3408_{16}	SPW Pending Interrupt Status Register	[SPW_PISR]
0700_340C_{16}	SPW Pending Interrupt Register	[SPW_PIR]
0700_3410_{16}	SPW Interrupt Mask Register	[SPW_IMR]
0700_3420_{16}	SPW Configuration Register	[SPW_CR]
0700_3424_{16}	SPW Link Handler Configuration Register	[SPW_SCR]

Byte address	Register name	Acronym
0700_3428_{16}	SPW Tick Number Register	[SPW_TNR]
0700_342C_{16}	SPW Clock Division Register	[SPW_CDR]
0700_3440_{16}	SPW Status Register	[SPW_SR]
0700_3444_{16}	SPW Link Status Register	[SPW_SSR]
0700_3448_{16}	SPW Receive Time Code Register	[SPW_RTCR]
0700_3460_{16}	SPW Initialisation Register	[SPW_IR]
0700_3584_{16}	SPW Packet Header High Word Register 0	[SPW_PHHWR0]
0700_3508_{16}	SPW Packet Header Low Word Register 0	[SPW_PHLWR0]
0700_3580_{16}	SPW Parallel Write Configuration Register 0	[SPW_PWCR0]
0700_3590_{16}	SPW Parallel Write Configuration Register 1	[SPW_PWCR1]
0700_35A0_{16}	SPW Parallel Write Configuration Register 2	[SPW_PWCR2]
0700_35B0_{16}	SPW Parallel Write Configuration Register 3	[SPW_PWCR3]
0700_35C0_{16}	SPW Parallel Write Configuration Register 4	[SPW_PWCR4]
0700_35D0_{16}	SPW Parallel Write Configuration Register 5	[SPW_PWCR5]
0700_35E0_{16}	SPW Parallel Write Configuration Register 6	[SPW_PWCR6]
0700_35F0_{16}	SPW Parallel Write Configuration Register 7	[SPW_PWCR7]
0700_3504_{16}	SPW Transmit Channel Configuration Register 0	[SPW_TCCR0]
0700_B408_{16}	SPW Read DMA Data Register	[SPW_RDDR]
0700_B480_{16}	SPW Read DMA Base Address Register 0	[SPW_RDBAR0]
0700_B484_{16}	SPW Read DMA Block Size Register 0	[SPW_RDBSR0]
0700_B488_{16}	SPW Read DMA Offset Register 0	[SPW_RDOR0]

Control Interface (CI)

Byte Address	Register Name	Acronym
0700_4800_{16}	CI Pending Interrupt Masked Status Register	[CI_PIMSR]
0700_4804_{16}	CI Pending Interrupt Masked Register	[CI_PIMR]
0700_4808_{16}	CI Pending Interrupt Status Register	[CI_PISR]
0700_480C_{16}	CI Pending Interrupt Register	[CI_PIR]
0700_4810_{16}	CI Interrupt Mask Register	[CI_IMR]
0700_4C00_{16}	PWT Pending Interrupt Masked Status Register	[PWT_PIMSR]
0700_4C04_{16}	PWT Pending Interrupt Masked Register	[PWT_PIMR]
0700_4C08_{16}	PWT Pending Interrupt Status Register	[PWT_PISR]
0700_4C0C_{16}	PWT Pending Interrupt Register	[PWT_PIR]
0700_4C10_{16}	PWT Interrupt Mask Register	[PWT_IMR]
0700_4C24_{16}	PWT Clock Configuration Register	[PWT_CLK]
0700_4C40_{16}	PWT Status Register	[PWT_STAT]
0700_C000_{16}	PWT DMA Base Address Register	[PWT_DBAR]
0700_C004_{16}	PWT DMA Offset Register	[PWT_DOR]
0700_C008_{16}	PWT DMA Data Register	[PWT_DDR]
0700_C018_{16}	PWT DMA Block Size Register	[PWT_DBSR]

6.10.1 General SCTMTC ASIC registers

6.10.1.1 Internal Scan Controller registers

Not applicable

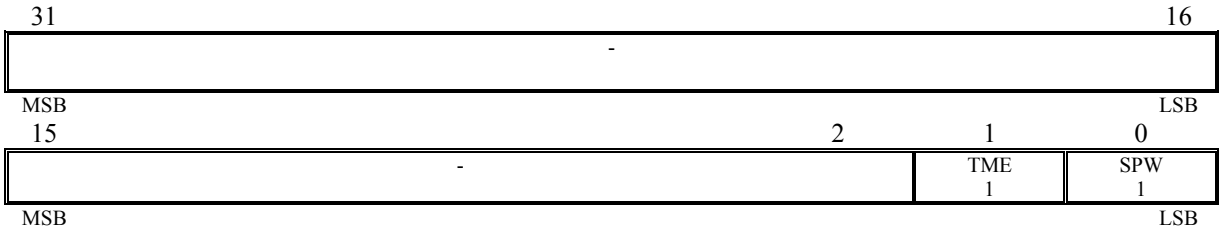
6.10.1.2 Test Access Port (TAP) registers

Not applicable

6.10.1.3 Clock and Reset (CAR) registers

All registers in the Clock and Reset (CAR) block are protected by means of Triple Modular Redundancy (TMR), not requiring any external refresh during a mission after being programmed following a power-on condition.

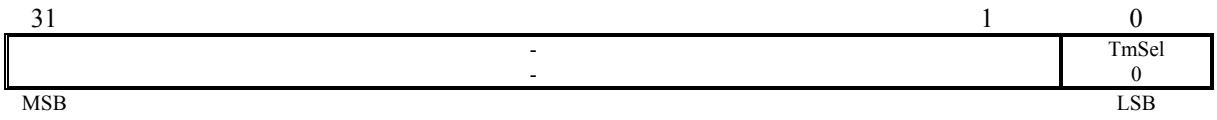
CAR Arm Clock Source Register [CAR_ArmClkSrc] **R**
CAR Clock Source Register [CAR_ClkSrc] **R**



Bit/field	Value	Description
SPW	0	Use *SysClk* as source for SpaceWire clock.
	1	Use *SpwClk* as source for SpaceWire clock.
TME	0	Use *SysClk* as source for Telemetry clock.
	1	Use one of *TmClk** as source for Telemetry clock.

When writing to the CAR Clock Source Register only the fields where the corresponding bit in the CAR Arm Clock Source Register is set are affected.
The CAR Arm Clock Source Register is cleared when writing to the CAR Clock Source Register. Writing to the register can only be done during configuration.

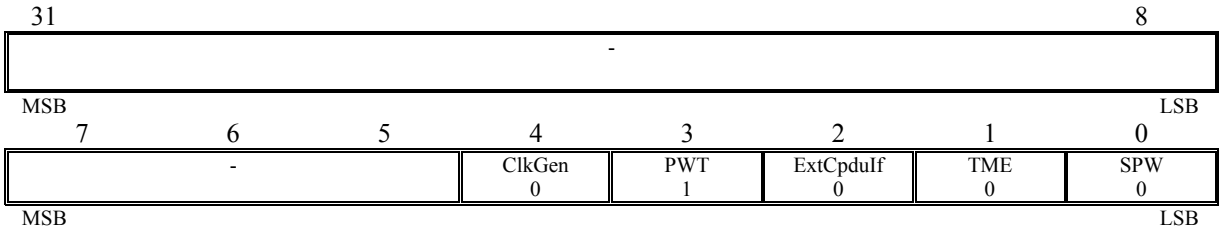
CAR TM Clock Source Register [CAR_TmClkSrc] **RW**



Bit/field	Value	Description
TmSel	0	Use *TmClk1* as source for Telemetry clock.
	1	Use *TmClk2* as source for Telemetry clock

This register only matters when CAR Clock Source Register.Tm is set to 1.

CAR Arm Clock Enable Register [CAR_ArmClkEna] R
CAR Clock Enable Register [CAR_ClkEna] R

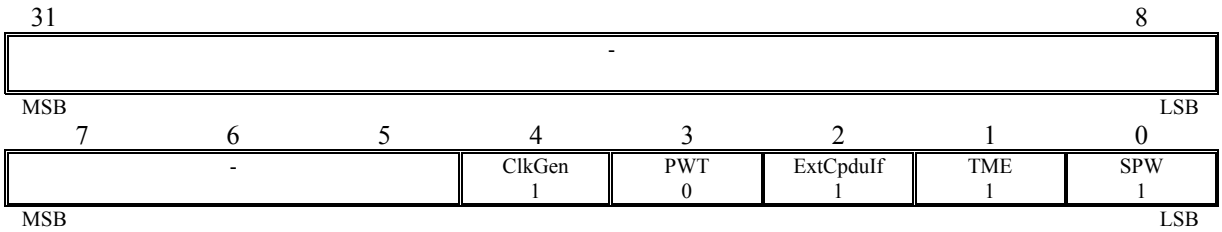


Bit/field		Description
ClkGen		Enable refresh counters
Pwt		Enable Pwt module
ExtCpduIf		Enable External CPDU interface module.
TME		Enable Telemetry module.
SPW		Enable Spacewire module.

When writing to the CAR Clock Enable Register only the fields where the corresponding bit in the CAR Arm Clock Enable Register is set are affected. The CAR Arm Clock Enable Register is cleared when writing to the CAR Clock Enable Register. Writing to the register can only be done during configuration.

Released

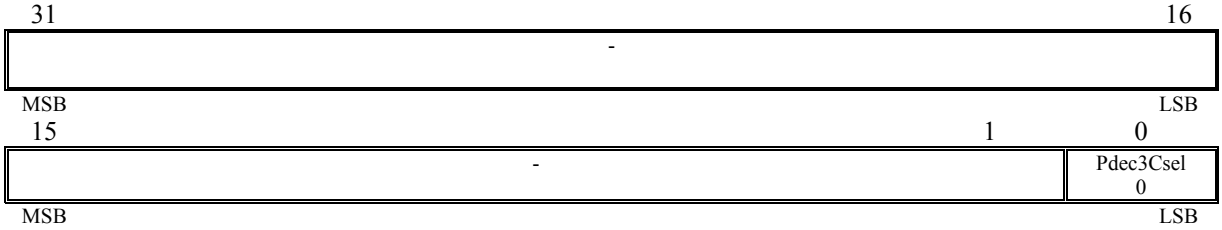
CAR Arm Reset Register [CAR_ArmModReset] R
CAR Reset Register [CAR_ModReset] R



Bit/field		Description
ClkGen		Reset refresh counters
Pwt		Reset Pwt module
ExtCpduIf		Reset External CPDU interface module.
TME		Reset Telemetry module.
SPW		Reset Spacewire module.

When writing to the CAR Reset Register only the fields where the corresponding bit in the CAR Arm Reset Register is set are affected. The CAR Arm Reset Register is cleared when writing to the CAR Reset Register. Writing to the register can only be done during configuration.

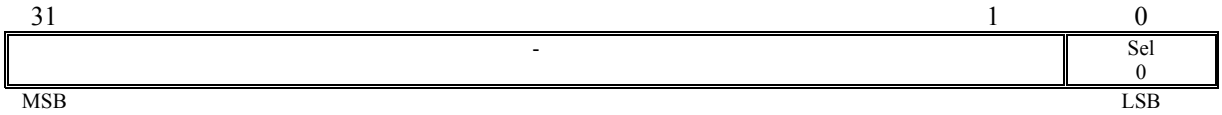
CAR Arm Pdec3 Csel Connect Register [CAR_ArmCselCon] R
CAR Pdec3 Csel Connect Register [CAR_CselCon] R



Bit/field	Value	Description
Pdec3Csel	0	Pdec3 is internally connected to the TC input of CSEL.
	1	ExtCpduIf is internally connected to the TC input of CSEL

When writing to the CAR Pdec3 Csel connect Register only the fields where the corresponding bit in the CAR Arm Pdec3 Csel connect Register is set are affected. The CAR Arm Pdec3 Csel connect Register is cleared when writing to the CAR Pdec3 Csel connect Register. Writing to the register can only be done during configuration.

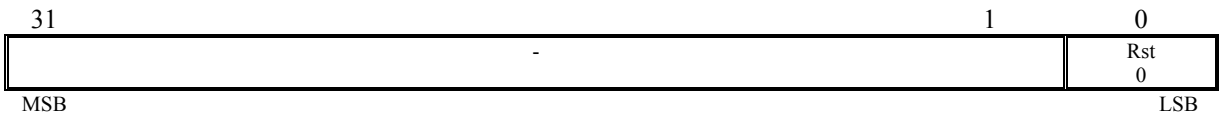
CAR CtrlIf Select Register [CAR_CtrlIfSel] R



Bit/field	Value	Description
Sel	0	Select Packet Wire interface.
	1	Select Spacewire interface.

Writing to the register can only be done during configuration.

CAR Power On Reset Register [CAR_PwrOnRst] RW



Bit/field	Value	Description
Rst	0	A power on reset has been seen.
	1	

Note that this is just a simple read/write register that is reset to 0. To be able to detect a power on reset, write 1 to it.

6.10.1.4 Configuration block registers

Conf External Interrupt Register [CNF_ExtIrqR] **R**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Conf 3	-													Conf 2 0	Conf 1 0

MSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-		PDEC3 0	-	SPW 0	CI 0	PWT 0	TME 0	-		MemIf 0	CPDM 0	CSEL 0	-		

LSB

Field	Value	Description
CSEL	1	Interrupt from CSEL
CPDM	1	Interrupt from CPDM
MemIf	1	Interrupt from Memory Interface
TME	1	Interrupt from TME
PWT	1	Interrupt from PWT, as a result of a CI-PWT read
CI	1	Interrupt from Control Interface
SPW	1	Interrupt from SpaceWire module
PDEC3	1	Interrupt from PDEC3
Conf 1	1	Interrupt from Configuration block
Conf 2	1	Interrupt from Configuration block
Conf 3	1	Interrupt from Configuration block

Released

GenClk Pending Interrupt Masked Status Register [GEN_PIMSR] **R**
GenClk Pending Interrupt Masked Register [GEN_PIMR] **R**
GenClk Pending Interrupt Status Register [GEN_PISR] **R**
GenClk Pending Interrupt Register [GEN_PIR] **RW**
GenClk Interrupt Mask Register [GEN_IMR] **RW**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-															

MSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Deass 3 0	Ass 3 0	Per 3 0	Sta rt3 0	Deass 2 0	Ass 2 0	Per 2 0	Start 2 0	Deass 1 0	Ass 1 0	Per 1 0	Start1 0

LSB

Field	Description
Start$	GenClk$ has changed state to running
Per$	GenClk$ has reached its period end
Ass$	GenClk$ has been changed from deasserted to asserted
Deass$	GenClk$ has been changed from asserted to deasserted

Note that it is possible to write to GenClk Interrupt Mask Register and enable/disable interrupts. This should normally only be done during configuration.

GenClk SW Start Register [GEN_START] **N/A**

31															0
-															
MSB															LSB

Field	Value	Description
NA		Writing to this register will start the GenClk$ which have their SWEn bit set and are not running.

Writing to the register can only be done during configuration.

GenClk $ Control Register [GEN_$CTRL] **R**
GenClk $ Control Set Register [GEN_$CTRLSET] **W**
GenClk $ Control Clear Register [GEN_$CTRLCLR] **W**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MSB															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	Polarity 1	Direct En 0	Res-erved 0	SW En 0
															LSB

Writing to the GEN_1CTRLSET and GEN_1CTRLCLR registers can only be done during configuration.

GenClk $ Control Register [GEN_$CTRL]

Field	Value	Description
SWEn	0	SW triggered start of GenClk$ is disabled
	1	SW triggered start of GenClk$ is enabled
DirectEn	0	GenClk$ is stopped
	1	GenClk$ is running
Pol	0	GenClk$ is active low
	1	GenClk$ is active high

Note: If the Pol field change value while the clock is disabled an edge will be generated on the GenClk$ signal. If the DirectEn field is set at the same time as the Pol field changed value no edge is generated.

GenClk $ Control Set Register [GEN_$CTRLSET]

Field	Value	Description
SWEn	1	Set SWEn triggered start of GenClk$ to enabled
DirectEn	1	Set GenClk$ to enabled
Pol	1	Set GenClk$ to active high

GenClk $ Control Clear Register [GEN_$CTRLCLR]

Field	Value	Description
SWEn	1	Clear SWEn triggered start of GenClk$ to disabled
DirectEn	1	Clear GenClk$ to disabled
Pol	1	Clear GenClk$ to active low

GenClk $ Period Register [GEN_$PERIOD] RW

31	0
Period 0	
MSB	LSB

Field	Value	Description
Period		Clock period-1 of generated clock

Writing to the GEN_1PERIOD register can only be done during configuration.

GenClk $ Assert Register [GEN_$ASSERT] RW

31	0
Assert 0	
MSB	LSB

Field	Value	Description
Assert		Number of cycles from period start where the generated clock is asserted.

Writing to the GEN_1ASSERT register can only be done during configuration.

GenClk $ Deassert Register [GEN_$DEASSERT] RW

31	0
Deassert 0	
MSB	LSB

Field	Value	Description
Deassert		Number of cycles from period start where the generated clock is deasserted.

Writing to the GEN_1DEASSERT register can only be done during configuration.

GenClk $ Counter Register [GEN_$COUNTER] RW

31	0
Counter 0	
MSB	LSB

Field	Value	Description
Counter		Current value of period counter.

Note: This register is only accessible for test purposes or to utilise the clock as a watchdog i.e. as long as the counter never reaches the assert value no edge is generated. Writing to the GEN_1COUNTER register can only be done during configuration.

6.10.1.4.1 Interrupt Handling registers

Please refer to the interrupt registers of the individual modules.

6.10.2 Memory Interface registers

6.10.2.1 Interrupt registers

PIM Pending Interrupt Masked Status Register [PIM_PIMSR] **R**
PIM Pending Interrupt Masked Register [PIM_PIMR] **R**
PIM Pending Interrupt Status Register [PIM_PISR] **R**
PIM Pending Interrupt Register [PIM_PIR] **RW**
PIM Interrupt Mask Register [PIM_IMR] **RW**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

MSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	CErr 0	Scu Err 0	ET 0

LSB

Field	Description
ET	Error trapped in First failing registers
ScuErr	Scrubber uncorrectable error during read
CErr	Correctable error trapped in First Failing Correctable Error Address Register

6.10.2.2 Configuration registers

PIM Scrubber Configuration Register [PIM_ScuCNFR] **RW**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

MSB

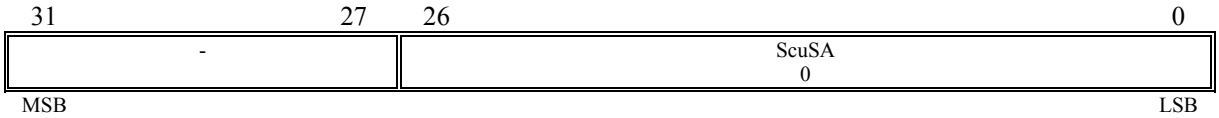
15	0
ScuRa $FFFF_{16}$	

LSB

Field	Value	Description
ScuRa	$00FF_{16}$-$FFFF_{16}$	Scrubber access rate, i.e number of clock cycles between each access. Bits 0 to 7 are locked to ones.

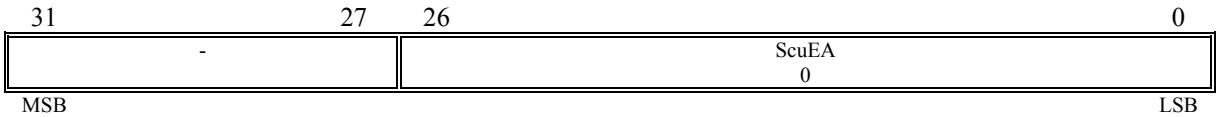
Released

PIM Scrubber Start Address Register [PIM_ScuSAR] RW



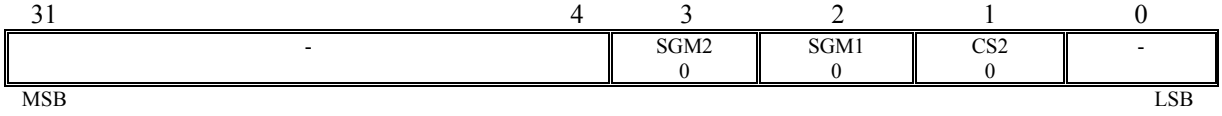
Field	Value	Description
ScuSA	Any	Scrubber Start Address, Reload value for the cyclic function of the scrubber. Observe that the scrubber works on word addresses and thus the lowest 2 bits of the address are ignored.

PIM Scrubber End Address Register [PIM_ScuEAR] RW



Field	Value	Description
ScuEA	Any	Scrubber End Address, End address for the cyclic function of the scrubber. Observe that the scrubber works on word addresses and thus the lowest 2 bits of the address are ignored.

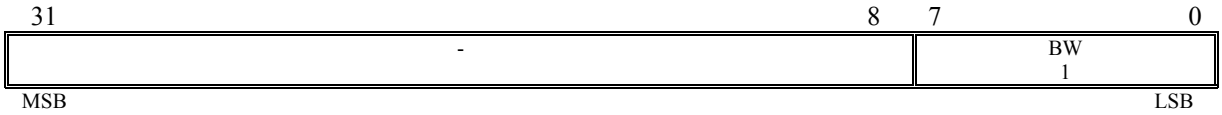
PIM Write Disable Status Register [PIM_WrDis] R
PIM Write Disable Set Register [PIM_WrDisSet] W
PIM Write Disable Clear Register [PIM_WrDisClr] W



Bit/field	Value	Description
CS2		Disable Write in *ExtRecLacN* area. If write restriction is enabled for the *ExtRecLacN* area setting this bit has no effect.
SGM1		Disable Write in User Area 1, part of *RamCsN* area.
SGM2		Disable Write in User Area 2, part of *RamCsN* area.

Writing to the Set register will set any bit where the written data is one.
Writing to the Clear register will clear any bit where the written data is one.

PIM TME Ratio Register [PIM_TmeRat] **R**

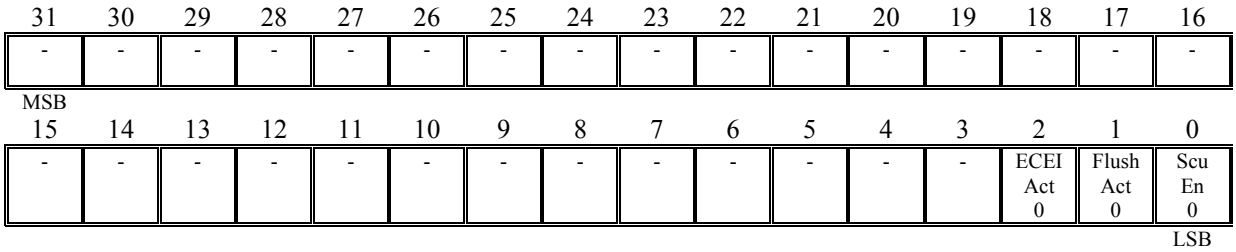


Bit/field	Value	Description
BW	Any	TME memory bandwidth ratio in clock cycles

Note: The **PIM_TmeRat** register can only be written to during configuration.

6.10.2.3 Status registers

PIM Status Register [PIM_SR] **R**
PIM Status Set Register [PIM_SRSet] **W**
PIM Status Clear Register [PIM_SRClr] **W**



PIM Status Register

Field	Value	Description
ScuEn	0	Scrubber function is disabled
	1	Scrubber function is enabled
FlushAct	0	Flush is no longer active
	1	Flush has been activated an is still running
ECEIAct	0	EDAC Check Error Inject Inactive
	1	EDAC Check Error Inject Active

PIM Status Set Register

Field	Value	Description
ScuEn	1	Enable Scrubber
FlushAct	1	No effect
ECEIAct	1	Activate EDAC Check Error Injection

PIM Status Clear Register

Field	Value	Description
ScuEn	1	Disable Scrubber
FlushAct	1	No effect
ECEIAct	1	Deactivate EDAC Check Error Injection. Note that the deactivation happens automatically when one error injection has been made.

PIM Scrubber Address Register [PIM_ScuAR] **R**

31 27	26 0
-	ScuA

MSB LSB

Field	Value	Description
ScuA	Any	Scrubber present Address Position.

PIM First Failing Correctable Error Address Register [PIM_FFCEAR] **RW**

31 27	26 0
-	CEA

MSB LSB

Field	Value	Description
CEA	Any	The address of the first failing correctable data error accessed by scrubber or any module.

The first access with correctable data error will lock the register from further traps. A write access to the PIM First Failing Correctable Error Address Register will release the register to normal operation, but does not affect the register content.

Note that the address in this register is incorrect if the TME encountered the error.

Released

6.10.2.4 Command registers

See also Status registers, Command is used to Set or Clear Status bits.

PIM FlushO Register [PIM_FLUSHO] **W**
PIM FlushT Register [PIM_FLUSHT] **W**

31 0
-

MSB LSB

Field	Value	Description
n/a	Any	Access flushes data from all internal buffers to target memory

PIM FlushT flushes data from the TME memory access buffers and PIM FlushO flushes data from other modules' memory access buffers. When a buffer is flushed all data stored in it is invalidated. Normally these registers are only used for testing the functionality of the memory interface.
Note: The two should never be used simultaneously. E.g. after a write to PIM FlushO, you should wait for the PIM Status Register.Flush Act bit to be 0 before doing a PIM FlushT. A flush command when any flush is active will be ignored.

6.10.2.5 Module definable registers

PIM Error Trap Register [PIM_ET] **RW**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

MSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	ET Acc 0	ET Bto 0	ET Ude 0	ETA Bcc 0	ETA Pi 0

LSB

Field	Description
ETAPi	AErr during PiRegister access bus access
ETABcc	Address Error, no MUnit selected, cycle aborted
ETUde	EDAC Uncorrectable Error
ETBto	Bus timeout after 256 *SysClk* cycles, no internal ready has been generated on a register access, cycle aborted.
ETAcc	Access violation, unauthorised access

PIM Error Trap Register is used to latch the cause for the erroneous Buscycle. The first access with any of the above defined signals active in end of the Buscycle will lock the register from further error traps. Also the PIM Error Info Register, PIM First Failing Address Register and PIM First Failing Data Register registers will be locked. A write access to the PIM Error Trap Register will release and reset all of the above mentioned registers to normal operation.

Error Trap is not affected by errors in accesses from TME.

PIM Error Info Register [PIM_EI] **R**

31	15
-	

MSB

14	8	7	4	3	2	1	0
EI Dma		EI MUnit 0		EISize 0		EI Wr 0	EI Rd 0

LSB

Bit	Value	Description
EIRd	1	Read access
EIWr	1	Write access
EISize	Size	DMA channel access DataSize
EIMUnit	MUnit	MUnit Memory mapped unit (not valid if ETABcc active in PIM_Error Trap Register) according to Table 6-16
EIDma	DmaUnit	DMA channel accessing when error trapped according to Table 6-17

MUnit	Memory area
0	*PromCsN*
1	N/A
2	*ExtRecLacCsN*
3	PDEC3 buffer
4	ExtCpduIf area
5	N/A
6	PDEC3 programmable AU key
7	User area 1
8	User area 2
9	TME / PM
10-14	N/A
15	SCTMTC registers

Table 6-16 MUnit numbering

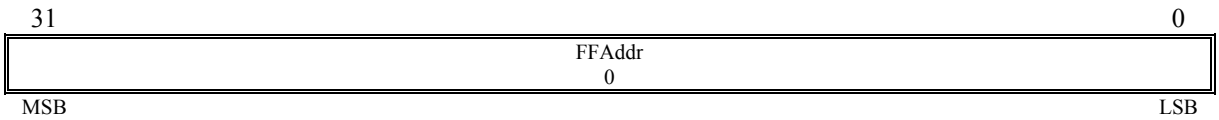
DmaUnit	Module	DMA channel
0-22	N/A	N/A
23	CPDM	Rd
24	ExtCpduIf	Wr
25-28	N/A	N/A
29	PDEC3	SIB Rd
30	PDEC3	SIB Wr
31	PDEC3	AUB Rd/Wr
32	PDEC3	SLRB Rd
33-42	N/A	N/A
43	Configuration Block	Fetch
44	Configuration Block	Execute
45	SPW	Rd
46	N/A	N/A
47	CtrlIf	Wr
48	PWT	Rd
49-127	N/A	N/A

Table 6-17 DMA channel numbering

PIM Error Info Register is used to latch the information about the erroneous Buscycle at the same time as the PIM Error Trap Register latches and locks further latching. A write access to the PIM Error Trap Register will re-enable latching of the PIM Error Info Register.

This register is reset when PIM Error Trap Register is written.

PIM First Failing Address Register [PIM_FFAR] R

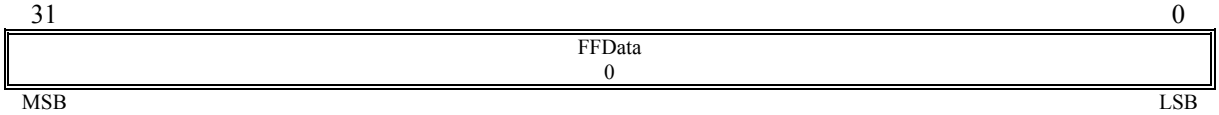


Field	Value	Description
FFAddr	Any	Address latched during the erroneous Buscycle.

PIM First Failing Address Register is used to latch the address of the erroneous Buscycle at the same time as the PIM Error Trap Register latches and locks further latching. A write access to the PIM Error Trap Register will re-enable latching of the PIM First Failing Address Register.

This register is reset when PIM Error Trap Register is written.

PIM First Failing Data Register [PIM_FFDR] R



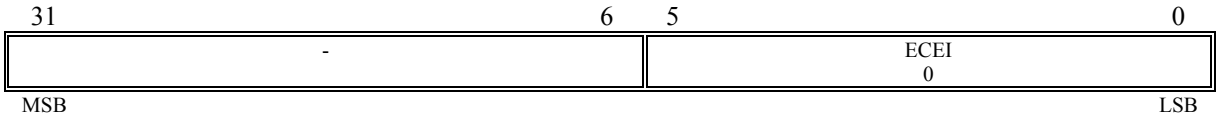
Field	Value	Description
FFData	Any	Data latched during the erroneous Buscycle.

PIM First Failing Data Register is used to latch the data of the erroneous Buscycle at the same time as the PIM Error Trap Register latches and locks further latching. A write access to the PIM Error Trap Register will re-enable latching of the PIM First Failing Data Register.

This register is reset when PIM Error Trap Register is written.

Released

PIM EDAC Check Error Inject Register [PIM_ECEI] RW



Field	Value	Description
ECEI	Any	Checkbit data pattern

PIM EDAC Check Error Inject Register is used to replace the checkbits during write access. This function is only active if armed by a write with '1' to Activate EDAC Check Error Injection bit in PIM Status SetRegister. It is deactivated automatically after the first write memory cycle after it has been armed, i.e. only active during one write memory cycle.

PIM EDAC Check Error Inject Register is only to be used during off-line test, since it is impossible to control which memory write access the error is injected in during normal operation.

PIM MUnit $ Register [PIM_MUnit$] **R**

31 12	11	10	9	8	7 5	4 2	1 0
-	WrRes	WrDis 0	Reserved	EDAC 0	WsWr 7	WsRd 7	Size

MSB LSB

Bit/field	Value	Description
Size	0	Memory width is Byte (8 bit)
	1	Memory width is HalfWord (16 bit)
	2-3	Memory width is HalfWord
WsRd	0-7	Wait States for read cycle
WsWr	0-7	Wait States for write cycle
EDAC	1	Enable EDAC protection
WrDis	1	Disable write
WrRes	1	Enable restricted write

Writing to the register can only be done during configuration.

Write accesses to a memory area are controlled by three different functions: write restriction, static write protection, and dynamic write protection. Write restriction has the highest priority of these three. If write restriction is enabled for a memory area that area can not be write protected; it can always be written by the dedicated DMA channel. The two write protection mechanisms have equal priotity; if any of the write protections are enabled the memory area is read-only.

Released

The PIM MUnit $ Registers have the following reset values for the WrRes and Size fields.

	MUnit 0	**MUnit 2**	**MUnit 3**
WrRes	0	1	0
Size	= *MemSize16*	1	1

Some fields in the PIM MUnit $ Registers are not writable, as specified by the table below. Fields not mentioned are always writable.

	MUnit 0	**MUnit 2**	**MUnit 3**
WrRes	Read Only		Read Only
WrDis			Read Only
EDAC	Read Only		
Size	Read Only		

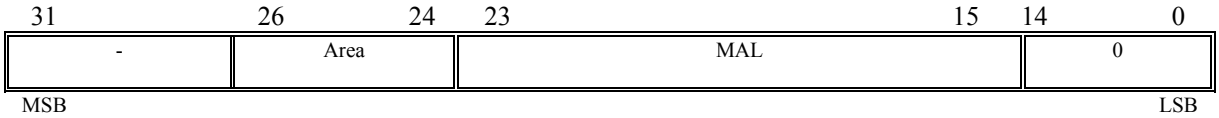
MUnit 0 corresponds to *PromCsN*

MUnit 2 corresponds to *ExtLacCsN*

MUnit 3 corresponds to *RamCsN*

Note: PIM MUnit 0 Register.WsWr and PIM MUnit 0 Register.WsRd are TMR to prevent SEUs from disrupting accesses to the refresh sequences.

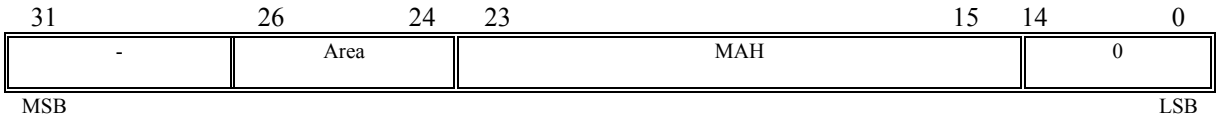
PIM MUnit $ Address Low Register [PIM_MU$L] R



Field	Value	Description
Area		Static field defining the most significant bits.
MAL	Address	Programmable part of Start address of the MUnit

The MUnit $ Address Low Register defines the MUnit chip select start address. This register is one of an array of registers each defining the start address of each MUnit handled by the Memory Interface. Writing to the register can only be done during configuration.

PIM MUnit $ Address High Register [PIM_MU$H] R



Field	Value	Description
Area		Static field defining the most significant bits.
MAH	Address	Programmable part of End address of the MUnit.

Released

The PIM MUnit $ Address High Register defines the MUnit chip select end address. This register is one of an array of registers each defining the end address of each MUnit handled by the Memory Interfac. Note that the end address is the first address after the available area. E.g. an end address of $03FF\_8000_{16}$ means that the last accessible address is $03FF\_7FFF_{16}$. Writing to the register can only be done during configuration.

6.10.3 Packet Telecommand Decoder Module (PDEC3) registers

6.10.3.1 Interrupt Registers

PDEC Pending Interrupt Masked Status Register [PDEC_PIMSR] **R**
PDEC Pending Interrupt Masked Register [PDEC_PIMR] **R**
PDEC Pending Interrupt Status Register [PDEC_PISR] **R**
PDEC Pending Interrupt Register [PDEC_PIR] **RW**
PDEC Interrupt Mask Register [PDEC_IMR] **RW**

31 … 3	2	1	0
-	New FAR 0	TC Abort 0	TC New 0

MSB LSB

Bit/field	Description
TCNew	The backend buffer is handed over to the PM
TCAbort	The PM is no longer owner of the backend buffer
NewFAR	New Frame Analysis Report available , i.e. the Stat flag has been cleared

6.10.3.2 Report registers

PDEC Frame Analysis Report [PDEC_FAR] **R**

31	30 … 29 … 28	27 … 26 … 25	24 … 23 … 22 … 21 … 20 … 19	18 … 17 … 16
Stat 0	FrameAna 000_2	IReason 000_2	CbCnt 00_0000_2	ErrCnt 000_2

MSB

15	14 … 13 … 12	11 … 10 … 9 … 8 … 7 … 6 … 5	4	3 … 2 … 1	0
Type 01_2	Channel 111_2	LastMap 11_1111_2	0	AuAna 000_2	0

LSB

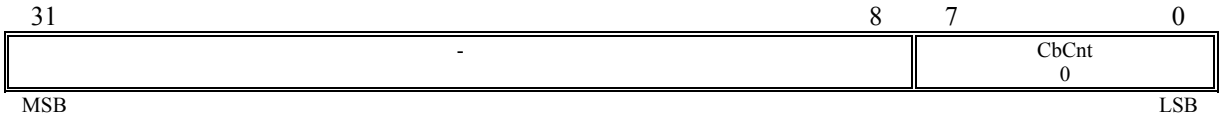
Released

Bit/field	Value	Description
AuAna		Authentication process analysis:
	000	No authentication report
	001	Authorised TC Segment with data
	010	Authorised and executable AU Control Command
	011	Authorised "Dummy Segment"
	100	TC Segment rejected because of error in the Signature
	101	TC Segment rejected because of error in the LAC
	110	Non-executable authorised AU Control Command
	111	Incorrect length of the TC Segment, i.e. length less than 10 octets
LastMap	Any	Number of last MAP Identifier
Channel	Any	Selected TC channel input
Type	00	AD Frame
	01	No Legal Frame
	10	BD Frame
	11	BC Frame
ErrCnt	Any	Number of single-error TC Code Block corrections, saturates at 111
CbCnt	Any	Number of accepted TC Code Blocks modulo 64
IReason		Reason for frame declared Illegal (in case of multiple reasons, the reason of lowest value will be presented):
	000	No Illegal report
	001	Error in Version Number and Reserved A and B fields
	010	Illegal combination (AC) of Bypass and Control Command flags
	011	Spacecraft Identifier did not match
	100	VC Identifier bits 0 (MSB) to 4 did not match
	101	VC Identifier bit 5 (LSB) did not match
	110	N(S) of BC or BD Frame not set to all zeroes
	111	Incorrect BC Control Command format
FrameAna		Frame analysis (in case of multiple possibilities, the report of lowest value will be presented):
	000	Abandoned CLTU
	001	Frame declared Dirty
	010	Frame declared Illegal for one reason
	011	Frame declared Illegal for multiple reasons
	100	AD Frame discarded because of Lockout
	101	AD Frame discarded because of Wait
	110	AD Frame discarded because of N(S) or V(R)
	111	Frame accepted by FARM-1
Stat	0	New analysis data
	1	Old analysis data

Note: Reading of PDEC_FAR will cause PDEC_CBCNT to be updated, to ensure that they are consistent.

The Stat flag is set when the register is read and cleared when a new report arrives.

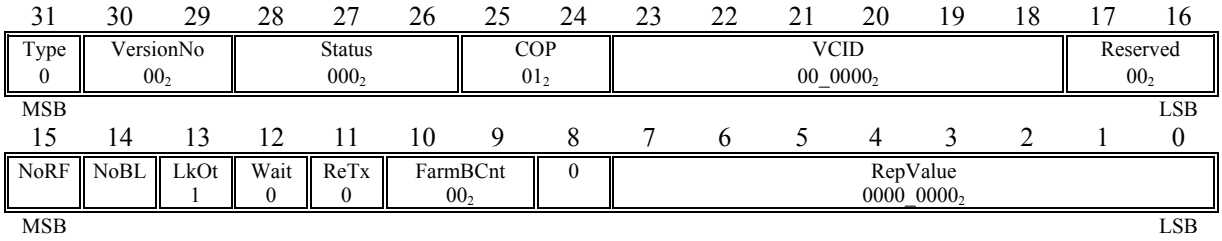
PDEC Code Block Counter [PDEC_CBCNT] R



Bit/field	Value	Description
CbCnt	Any	Entire code block counter

Note: The contents of PDEC_CBCNT are only updated when PDEC_FAR is read, to ensure consistency.

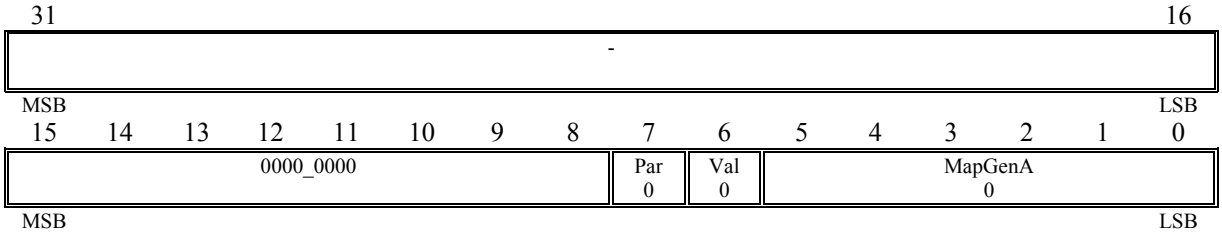
PDEC Command Link Control Word [PDEC_CLCW] R



Bit/field	Value	Description
RepValue	Any	Expected Frame Sequence Number in the next AD Frame, V(R)
FarmBCnt	Any	FARM B count; 2-bit wrap-around count of accepted BC and BD Frames
ReTx	Any	Value of FARM-1 Retransmit flag
Wait	Any	Value of FARM-1 Wait flag
LkOt	Any	Value of FARM-1 Lockout flag
NoBL		No bit lock flag, combinatorially generated from *PdecTcAct*:
	0	At least one of the *PdecTcAct** inputs are asserted
	1	No *PdecTcAct** input is asserted
NoRF		No RF available flag, combinatorially generated from *PdecRfAvN*:
	0	At least one of the *PdecRfAvN** inputs are asserted
	1	No *PdecRfAvN** input is asserted
Reserved	00	Reserved for future applications
VCID	Any	Virtual Channel Identifier
COP	01	COP-1 is in use
Status	000	Reserved for future applications
VersionNo	00	Identifies the structure of the CLCW
Type	0	Identifies the word as containing CLCW

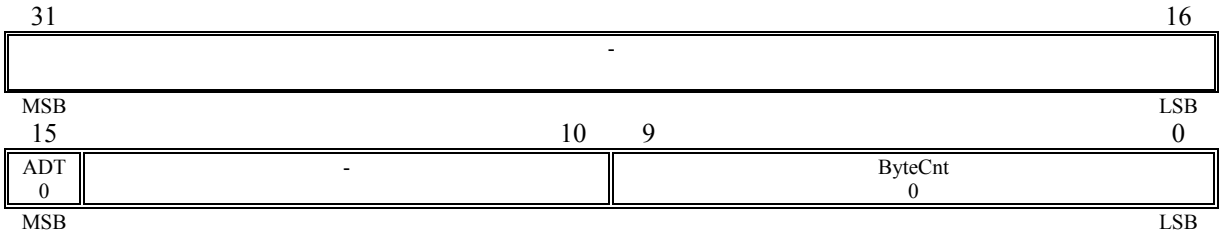
6.10.3.3 Map Status registers

PDEC MAP Link Address [PDEC_MAPADR] **R**



Bit/field	Value	Description
MapGenA	Any	Which MAP Interface to route the TC Segment to, obtained as a look-up value in the PROM/EEPROM selected by the MAP Identifier of the Segment
Val	0	MapAdr value is invalid and the TC Segment shall be discarded
	1	MapAdr value is valid to use for routing the TC Segment
Par	Any	Expected parity for bits 0 to 6 of MapAdr, as defined in section 6.3.2.8

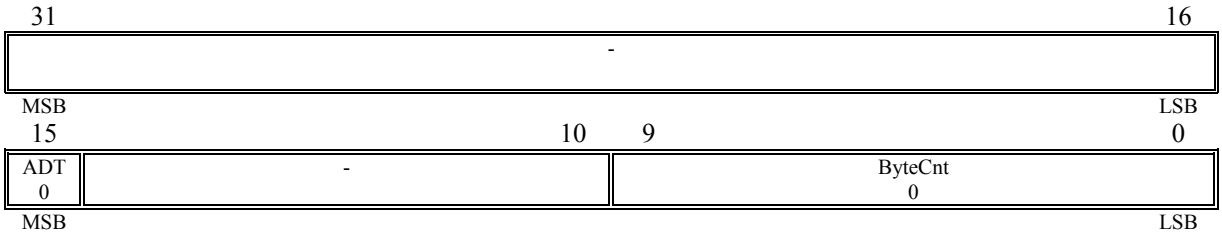
PDEC General MAP Interface Status Register [PDEC_MAPGSTAT] **R**



Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through the general MAP Interface
ADT	0	The last Segment transferred through the general MAP Interface was not aborted
	1	The last Segment transferred through the general MAP Interface was aborted by the PDEC3 before it was completely transferred.

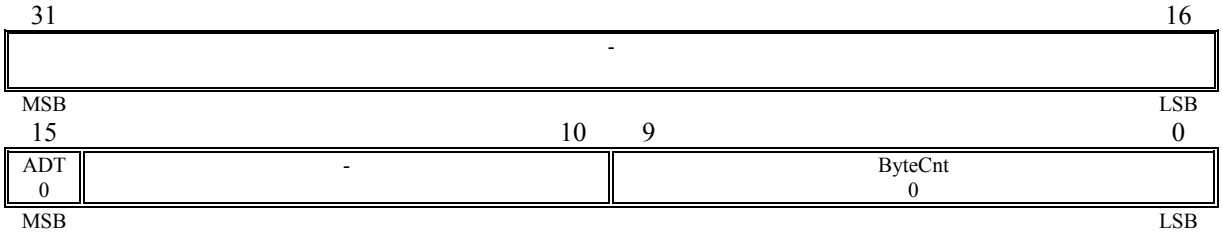
Released

PDEC MAP Interface 1 Status Register [PDEC_MAP1STAT] R



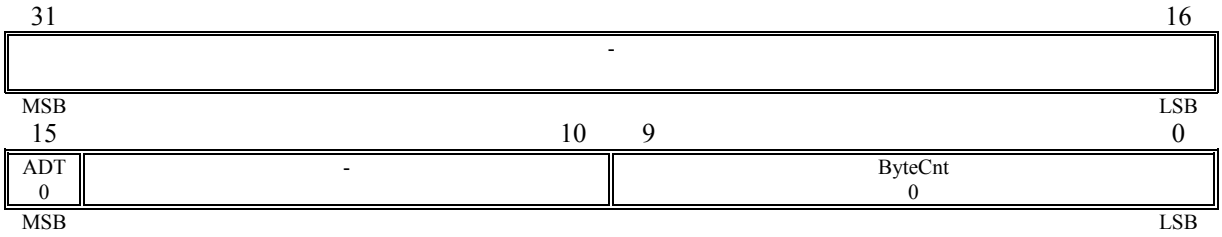
Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through MAP Interface 1
ADT	0	The last Segment transferred through MAP Interface 1 was not aborted
	1	The last Segment transferred through MAP Interface 1 was aborted by the PDEC3 before it was completely transferred.

PDEC MAP Interface 2 Status Register [PDEC_MAP2STAT] R



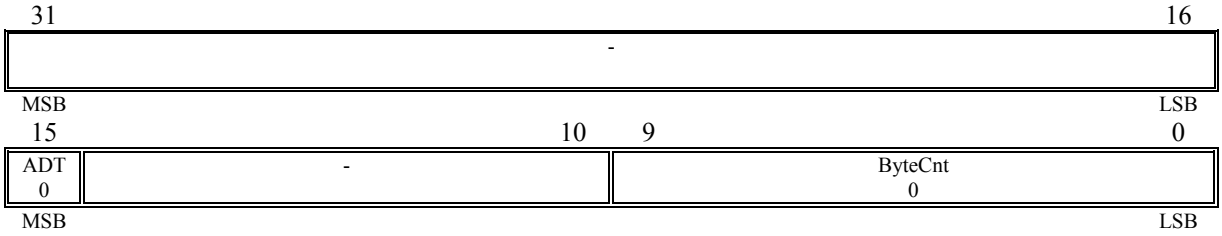
Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through MAP Interface 2
ADT	0	The last Segment transferred through MAP Interface 2 was not aborted
	1	The last Segment transferred through MAP Interface 2 was aborted by the PDEC3 before it was completely transferred.

PDEC MAP Interface 3 Status Register [PDEC_MAP3STAT] R



Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through MAP Interface 3
ADT	0	The last Segment transferred through MAP Interface 3 was not aborted
	1	The last Segment transferred through MAP Interface 3 was aborted by the PDEC3 before it was completely transferred.

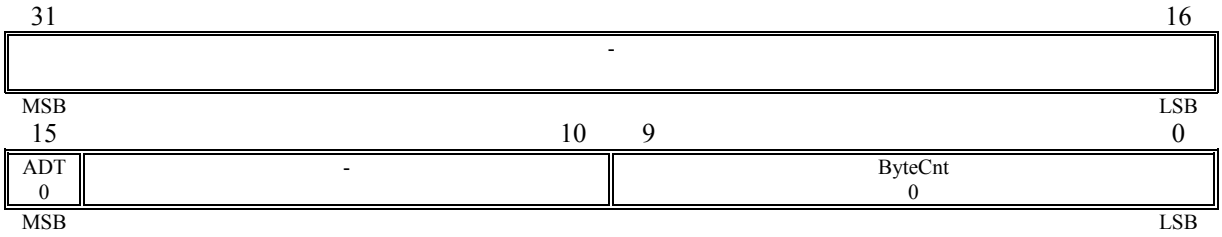
PDEC MAP Interface 4 Status Register [PDEC_MAP4STAT] R



Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through MAP Interface 4
ADT	0	The last Segment transferred through MAP Interface 4 was not aborted
	1	The last Segment transferred through MAP Interface 4 was aborted by the PDEC3 before it was completely transferred.

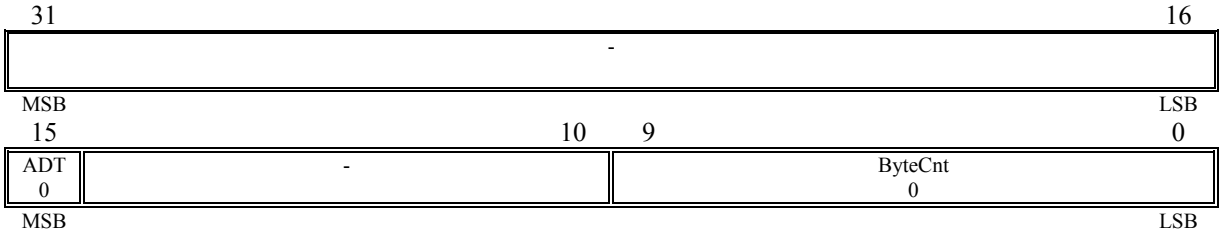
Released

PDEC MAP Interface 5 Status Register [PDEC_MAP5STAT] **R**



Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through MAP Interface 5
ADT	0	The last Segment transferred through MAP Interface 5 was not aborted
	1	The last Segment transferred through MAP Interface 5 was aborted by the PDEC3 before it was completely transferred.

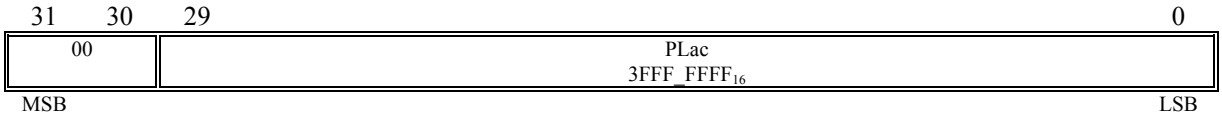
PDEC Common Interface Status Register [PDEC_MAPCSTAT] **R**



Bit/field	Value	Description
ByteCnt	Any	Byte count for last Segment transferred through any of the MAP Interfaces
ADT	0	The last Segment was not aborted
	1	The last Segment was aborted by the PDEC3 before it was completely transferred.

6.10.3.4 Authentication Status registers

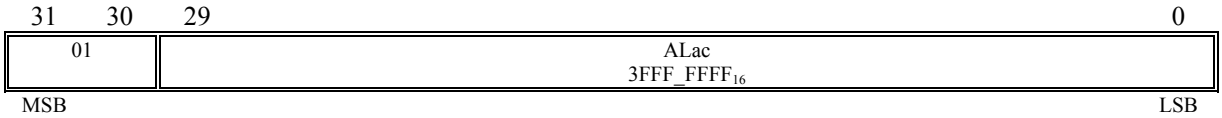
PDEC AU Status Report Part 1 [PDEC_AUSR1] **R**



Field	Value	Description
PLac	Any	Principal LAC Counter

Note: Reading of PDEC_AUSR1 will cause the other AUSR registers to be updated, to ensure that the parts are consistent.

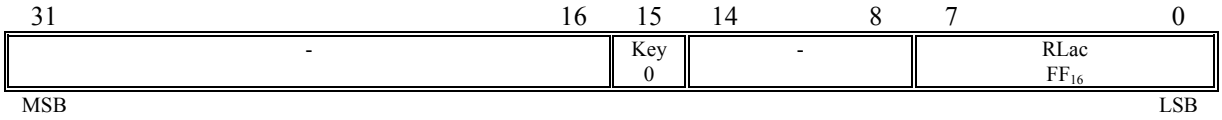
PDEC AU Status Report Part 2 [PDEC_AUSR2] **R**



Field	Value	Description
ALac	Any	Auxiliary LAC Counter

Note: The contents of PDEC_AUSR2 are only updated when PDEC_AUSR1 is read, to ensure that all three parts are consistent.

PDEC AU Status Report Part 3 [PDEC_AUSR3] **R**



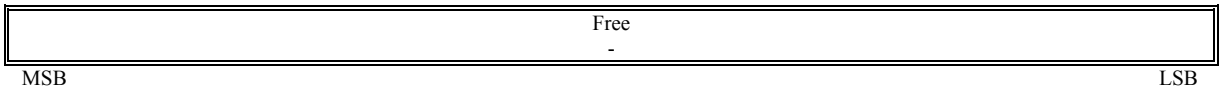
Bit/field	Value	Description
RLac	Any	Recovery LAC Counter value
Key	0	Fixed key in use by AU
	1	Programmable key in use by AU

Note: The contents of PDEC_AUSR3 are only updated when PDEC_AUSR1 is read, to ensure that all three parts are consistent.

Released

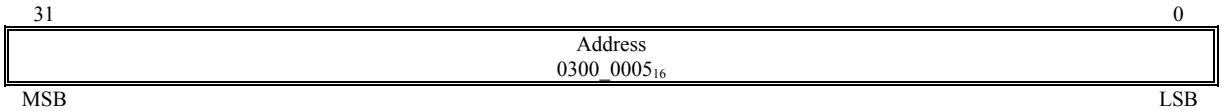
6.10.3.5 PM Buffer Area Handling registers

PDEC Buffer Free [PDEC_BFREE] **W**



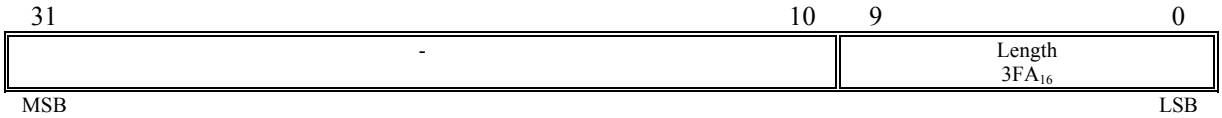
Bit/field	Value	Description
Free	Any	Backend buffer is handed back to the PDEC3

PDEC Buffer Pointer [PDEC_BPTR] **R**



Bit/field	Value	Description
Address	Any	Address to first byte in the TC Segment in the backend buffer

PDEC Segment Length [PDEC_SLEN] **R**

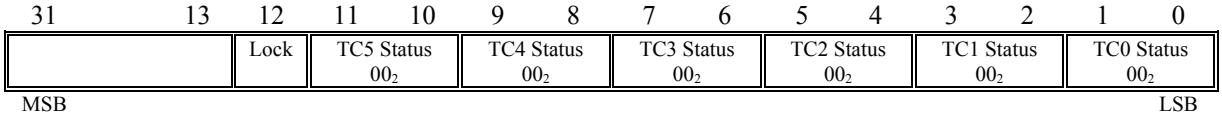


Bit/field	Value	Description
Length	Any	Number of bytes in TC Segment

Note: The register is calculated based on the frame length, which is reset to 0, minus the frame header size and frame CRC size. The reset value of PDEC_SLEN is therefore $3FA_{16} = -6_{10}$.

6.10.3.6 Monitor registers

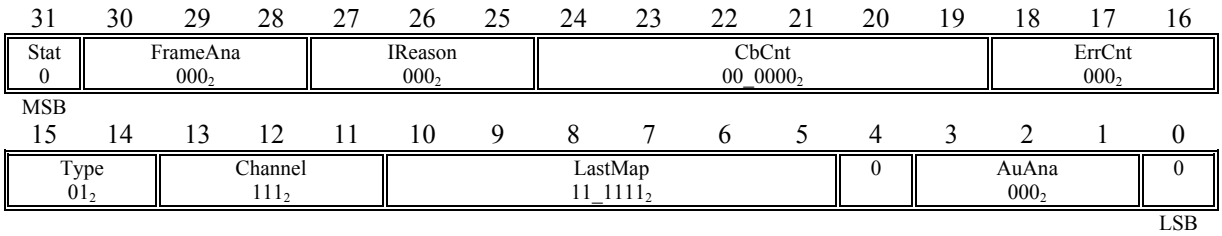
PDEC Monitor [PDEC_MON] **R**



Bit/field	Value	Description
Tc$ Status		Status of TC Channel $
	00	TC Channel is inactive
	01	TC Channel is active
	10	TC Channel has timed out (either clock or start sequence time out)
Lock	0	No Start Sequence has been found
	1	Start Sequence found, CLTU is being processed

6.10.3.7 Copy registers

PDEC Frame Analysis Report Copy [PDEC_FARCP] **R**

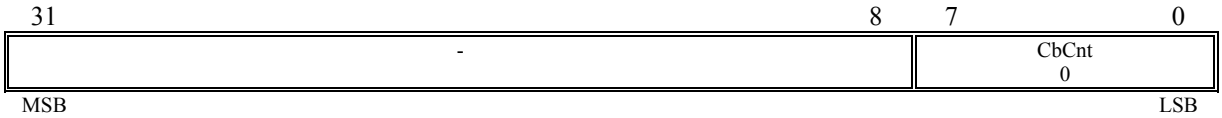


Refer to PDEC Frame Analysis Report in section 6.10.3.2 for a description of the different fields.

Note: Reading of PDEC_FARCP will cause PDEC_CBCNTCP to be updated, to ensure that they are consistent.

The Stat flag is set when the register is read and cleared when a new report arrives.

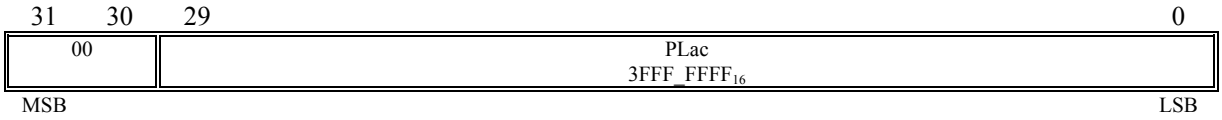
PDEC Code Block Counter Copy [PDEC_CBCNTCP] R



Bit/field	Value	Description
CbCnt	Any	Entire code block counter

Note: The contents of PDEC_CBCNTCP are only updated when PDEC_FARCP is read, to ensure consistency.

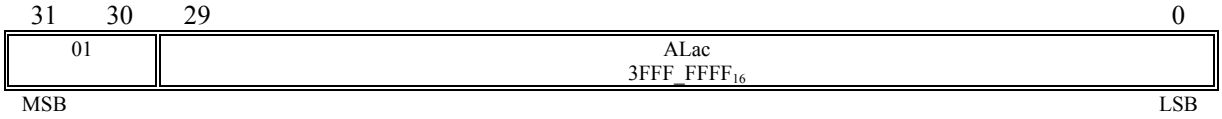
PDEC AU Status Report Copy Part 1 [PDEC_AUSRCP1] R



Field	Value	Description
PLac	Any	Principal LAC Counter

Note: Reading of PDEC_AUSRCP1 will cause the other AUSRCP registers to be updated, to ensure that the parts are consistent.

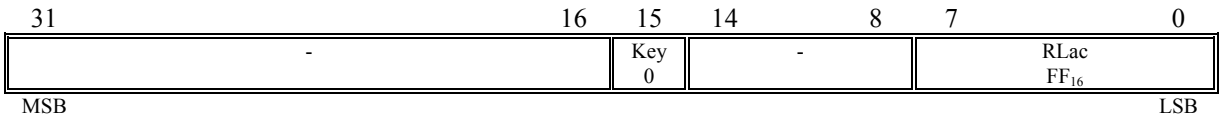
PDEC AU Status Report Copy Part 2 [PDEC_AUSRCP2] R



Field	Value	Description
ALac	Any	Auxiliary LAC Counter

Note: The contents of PDEC_AUSRCP2 are only updated when PDEC_AUSRCP1 is read, to ensure that all three parts are consistent.

PDEC AU Status Report Copy Part 3 [PDEC_AUSRCP3] R

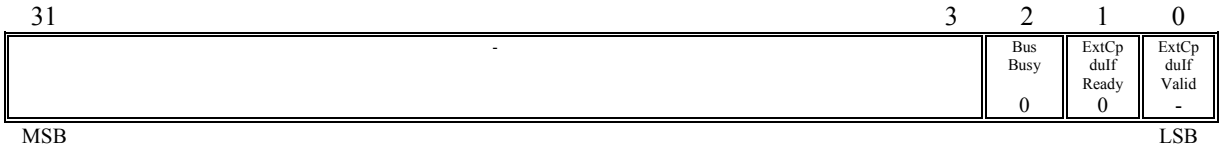


Bit/field	Value	Description
RLac	Any	Recovery LAC Counter value
Key	0	Fixed key in use by AU
	1	Programmable key in use by AU

Note: The contents of PDEC_AUSRCP3 are only updated when PDEC_AUSRCP1 is read, to ensure that all three parts are consistent.

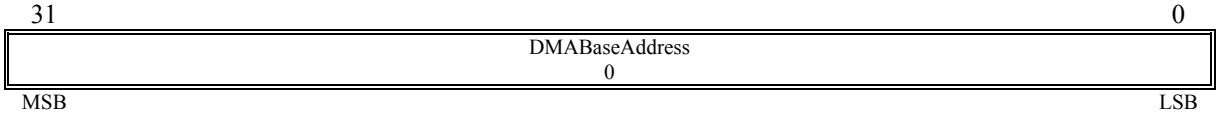
6.10.4 External CPDU Interface Module (ExtCpduIf) registers

ExtCpduIf Status Register [XRM_STAT] **R**



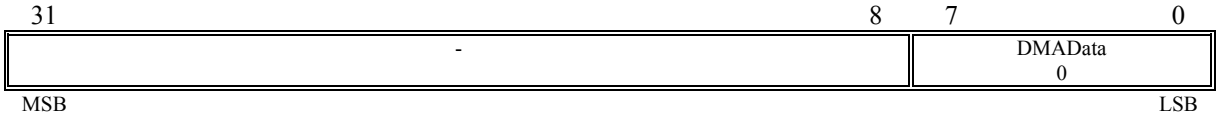
Field	Description
ExtCpduIfValid	The current value of the *ExtCpduIfValid* input.
ExtCpduIfReady	The current value of the active *ExtCpduIfRdy* output.
BusBusy	The module is busy with DMA accesses.

ExtCpduIf DMA Base Address Register [XCI_DBAR] **R**



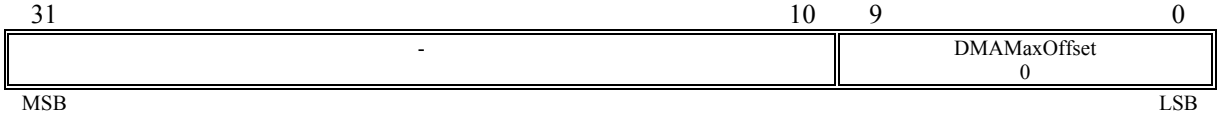
Field	Description
DMA Base Address	Byte address of the first DMA access in the DMA block transfer.

ExtCpduIf DMA Data Register [XCI_DDR] **R**



Field	Description
DMAData	Data to be written to memory.

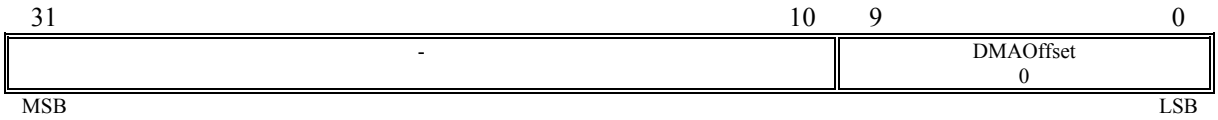
ExtCpduIf DMA Max Offset Register [XCI_DMOR] **R**



Field	Description
DMA Max Offset	Upper limit of the ExtCpduIf DMA Offset Register. This prevents the ExtCpduIf from writing outside its designated memory area.

Released

ExtCpduIf DMA Offset Register [XCI_DOR] R



Field	Description
DMA Offset	Byte address pointer relative to DMA Base Address. The DMA access address is DMA Base Address + DMA Offset. The value is incremented on each DMA write.

6.10.5 CPDM Selector Module (CSEL) registers

6.10.5.1 Interrupt registers

CSEL Pending Interrupt Masked Status Register [CS_PIMSR] **R**
CSEL Pending Interrupt Masked Register [CS_PIMR] **R**
CSEL Pending Interrupt Status Register [CS_PISR] **R**
CSEL Pending Interrupt Register [CS_PIR] **RW**
CSEL Interrupt Mask Register [CS_IMR] **RW**

31 … 8	7	6	5	4	3	2	1	0
-	IErr	RS Error	RS Time	Abort	Disc-ard	Cpdm Done	Cpdm Error	NoErr
	0	0	0	0	0	0	0	0

MSB LSB

Field	Description
NoErr	Previous CSEL Status error has been corrected
CpdmError	The CPDM failed to execute the PM request
CpdmDone	The CPDM finished with the PM request
Discard	The PM request was discarded by the CSEL
Abort	The PM request was not completely executed
RSTime	Remote CSEL has not finished in time
RSError	Illegal value on *CselStatusIn*
IErr	The CSEL detected an internal error

Released

6.10.5.2 Timing Reference registers

CSEL 1 Second Register [CS_1SEC] **N/A**

31 … 26	25 … 0
-	Count $20\,000\,000_{10}$

MSB LSB

Field	Description
Count	Timing reference value. For correct timing, the field shall be set to $N-1$, where N is the number of system clock cycles in 1 s.

Writing to the register can only be done during configuration.

6.10.5.3 Configuration registers

CSEL Max TC Only Time Register [CS_MAXTC] **N/A**

31 … 16	15 … 0
-	Count 255_{10}

MSB LSB

Field	Description
Count	Maximum number of seconds the CSEL will spend in the TC Only mode after a Set TC Only command – 1, i.e. a value of 0 gives a 1 second delay and the maximum delay is 65536 s.

Writing to the register can only be done during configuration.

CSEL Remote Status Timeout Time Register [CS_RSTT] **N/A**

31 … 11	10 … 0
-	Count 2047_{10}

MSB LSB

Field	Description
Count	Number of seconds the remote CSEL is allowed to stay in any busy state – 1, i.e. a value of 0 gives a 1 second delay and the maximum delay is 2048 s.

Writing to the register can only be done during configuration.

6.10.5.4 Status registers

CSEL Status Register [CS_STAT] **R**

31 … 7	6	5	4	3	2	1	0
-	Mode		Cpdm Busy	Fail Silent	Busy Rm	Busy Pm	Busy Gnd
	0	0	0	0	0	0	0

MSB LSB

Field	Value	Description
BusyGnd	1	The CSEL has accepted a Ground request and is busy completing it
BusyPm	1	The CSEL has accepted a PM request and is busy completing it
BusyRm	1	The CSEL has accepted a RM request and is busy completing it
FailSilent	1	CSEL has entered fail silent mode
CpdmBusy	1	The CSEL has sent a request that is still being processed to the CPDM
Mode	00	RM ON mode
	01	RM OFF mode
	10	TC Only mode

CSEL Remote Status Register [CS_RSTAT] **R**

31 ... 5	4	3	2	1	0
-		Time-out 0	Error 0	Busy Rm 0	Gnd Pm 0

MSB LSB

Field	Value	Description
GndPm	1	Remote CSEL has received or is executing a Ground or PM sequence
BusyRm	1	Remote CSEL has received or is executing an RM sequence
Error	1	Remote CSEL sends illegal status
Timeout	1	CSEL Status interface has timed out

Note: If Timeout is high, all the others will be low.

6.10.5.5 Command registers

CSEL PM Request Register [CS_PMREQ] **W**

31 ... 0
Request

MSB LSB

Field	Value	Description
Request	$BEC0FFEE_{16}$	Send a segment specified by [CS_PMBUFF] and [CS_PMSLEN]

CSEL PM Buffer Pointer Register [CS_PMBUFF] **RW**

31 ... 0
Address 0

MSB LSB

Field	Value	Description
Address	Any	Pointer to buffer in a PM request

CSEL PM Segment Length Register [CS_PMLEN] **RW**

31 ... 10	9 ... 0
-	Length 0

MSB LSB

Field	Value	Description
Length	Any	Length of segment in a PM request

6.10.6 Command Pulse Distribution Module (CPDM) registers

6.10.6.1 Interrupt Registers

CPDM Pending Interrupt Masked Status Register [CPDM_PIMSR] **R**
CPDM Pending Interrupt Masked Register [CPDM_PIMR] **R**
CPDM Pending Interrupt Status Register [CPDM_PISR] **R**
CPDM Pending Interrupt Register [CPDM_PIR] **RW**
CPDM Interrupt Mask Register [CPDM_IMR] **RW**

31	2	1	0
-	Done 0	New Gnd 0	New Status 0

MSB LSB

Field	Description
New Status	The CPDM Status Register has been updated
New Gnd	The CPDM Status Report Register has been updated
Done	The CPDM has finished executing a telecommand packet

6.10.6.2 Configuration Registers

CPDM Pulse Duration Register [CPDM_PD] **R**

31	20	19	0
-		Count $32C7F_{16}$	

MSB LSB

Field	Description
Count	Timing reference value for pulse output. For correct timing, the field shall be set to $N-1$, where N is the number of system clock cycles in the *duration*, D. For correct behaviour, select 10 ms ≤ D ≤ 15 ms.

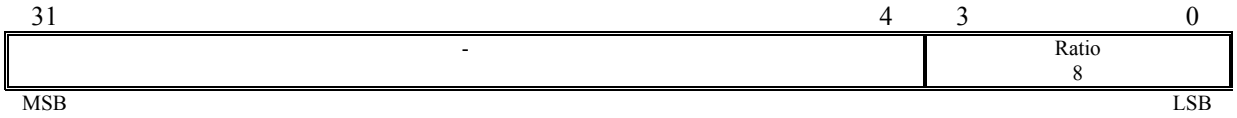
The reset value is chosen to set D to 13 ms at 16 MHz system frequency.

To allow the timings specified in section 7.6.2, the *duration*, D, should be at least 8. The register value should not be set to less than 7 to allow the. This has no real impact on the system clock since this requirement can be met as long as the clock frequency exceeds 800 Hz.

Writing to the register can only be done during configuration.

Released

CPDM Clock Ratio Register [CPDM_CR] R

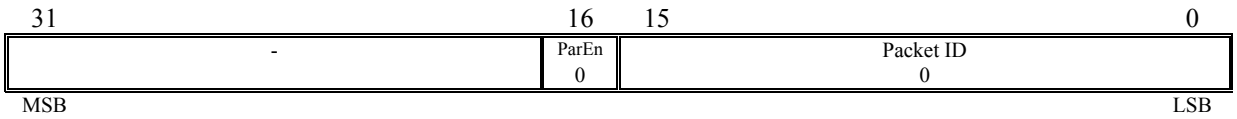


Field	Description
Ratio	Ratio between *SysClk* and *CpdmClk*, i.e. $f_{CpdmClk} = f_{SysClk} / 2^{(Ratio + 2)}$

The reset value results in a 15.625 kHz *CpdmClk* at 16 MHz system frequency.

Writing to the register can only be done during configuration.

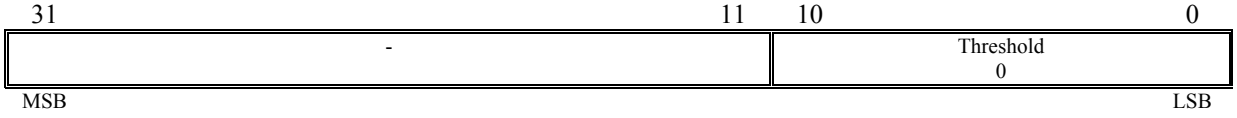
CPDM Configuration Register [CPDM_CNF] R



Field	Value	Description
Packet ID	Any	Is compared with the Packet Identification field in the CPDU Packet Header.
ParEn	0	Disable parity check on CPDU command instructions
	1	Enable parity check on CPDU command instructions

Writing to the register can only be done during configuration.

CPDM Lockout Register [CPDM_LCK] R



Field	Description
Threshold	Lowest allowed Output Number for a packet of Lockout type.

Writing to the register can only be done during configuration.

6.10.6.3 Status registers

CPDM Status Report Register [CPDM_SRR] **R**

31	18	17	16	15 14	13 0
-	Done 0	Error 0	Abort 0	LstPkt 00	SeqCnt $3FFF_{16}$

MSB LSB

Field	Value	Description
SeqCnt	Any	Source sequence count of last legal Ground CPDU packet
LstPkt	00	Value only present after reset
	01	Last Ground CPDU packet accepted as Clean and Legal
	10	Last Ground CPDU packet Clean but Illegal, and thus erased
	11	Last Ground CPDU packet Dirty, and thus erased
Abort	0	Last Ground CPDU packet was not aborted
	1	Last Ground CPDU packet was aborted
Error	0	No error was detected during the execution of the last Ground CPDU packet[1]
	1	Last Ground CPDU packet was not fully executed due to an error
Done	0	SeqCnt and LstPkt have been updated but not Abort and Error
	1	All fields in the register have been updated

1) This flag is not set for dirty or illegal packets

This register is used for reporting execution status of CPDU packets from the PDEC3.

Released

CPDM Status Register [CPDM_SR] **R**

31	18	17	16	15 14	13 0
-	Done 0	Error 0	Abort 0	LstPkt 00	SeqCnt $3FFF_{16}$

MSB LSB

Field	Value	Description
SeqCnt	Any	Source sequence count of last legal RM or PM CPDU packet
LstPkt	00	Value only present after reset
	01	Last RM or PM CPDU packet accepted as Clean and Legal
	10	Last RM or PM CPDU packet Clean but Illegal, and thus erased
	11	Last RM or PM CPDU packet Dirty, and thus erased
Abort	0	Last RM or PM CPDU packet was not aborted
	1	Last RM or PM CPDU packet was aborted
Error	0	No error was detected during the execution of the last RM or PM CPDU packet[1]
	1	Last RM or PM CPDU packet was not fully executed due to an error
Done	0	SeqCnt and LstPkt have been updated but not Abort and Error
	1	All fields in the register have been updated

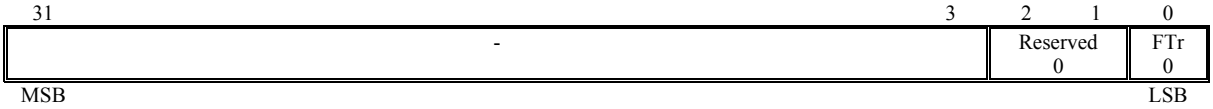
1) This flag is not set for dirty or illegal packets

This register is used for reporting execution status of CPDU packets from PM or ExtCpduIf.

6.10.7 Packet Telemetry Encoder Module (TME) registers

6.10.7.1 Interrupt registers

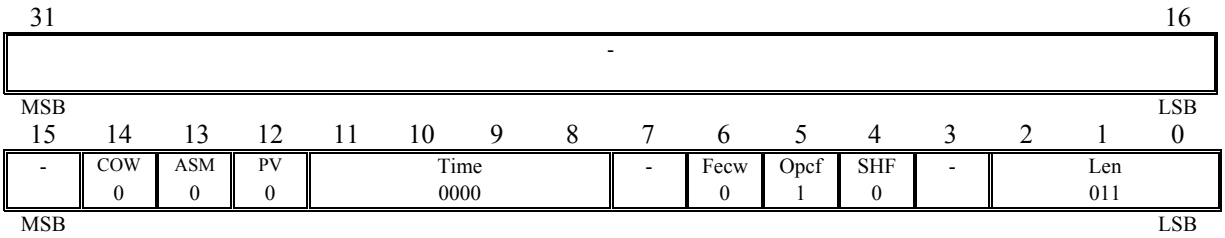
TME Pending Interrupt Masked Status Register [TME_PIMSR] **R**
TME Pending Interrupt Masked Register [TME_PIMR] **R**
TME Pending Interrupt Status Register [TME_PISR] **R**
TME Pending Interrupt Register [TME_PIR] **RW**
TME Interrupt Mask Register [TME_IMR] **RW**



Bit/field	Description
FTr	Frame transmitted

6.10.7.2 Configuration registers

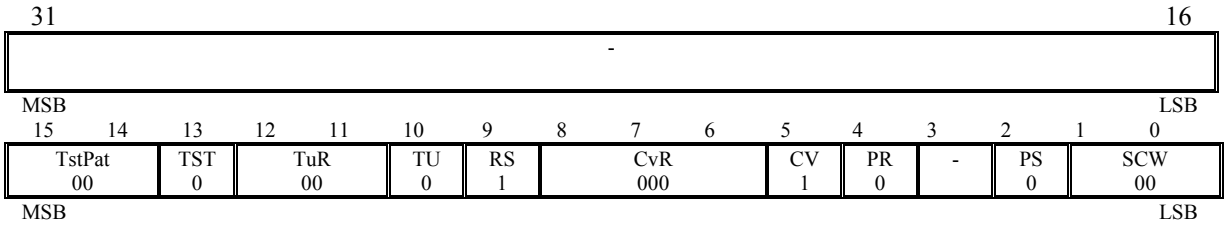
TME TM Encoding Configuration Register 0 [TME_TmEnCfg0] **RW**



Bit/field	Value	Description
Len	000	Transfer frame length 223 octets
	001	Transfer frame length 446 octets
	010	Transfer frame length 892 octets
	011	Transfer frame length 1115 octets
	100	Transfer frame length 239 octets
	101	Transfer frame length 478 octets
	110	Transfer frame length 956 octets
	111	Transfer frame length 1195 octets
SHF	0	No Secondary Header
	1	Secondary Header generated
Opcf	0	No Operational Control Field
	1	Operational Control Field present in the Transfer Frame Trailer
Fecw	0	No Frame Error Control Word
	1	Frame Error Control Word present in the Transfer Frame Trailer
Time	0000	TIME Strobe asserted for VC0 Count MOD 1; 0, 1, 2, … 253, 254, 255
	0001	TIME Strobe asserted for VC0 Count MOD 2; 0, 2, 4, 6, … 250, 252, 254
	0010	TIME Strobe asserted for VC0 Count MOD 4; 0, 4, 8, 16, … 246, 248, 252
	0011	TIME Strobe asserted for VC0 Count MOD 8; 0, 8, 16, 32, … 232, 240, 248
	0100	TIME Strobe asserted for VC0 Count MOD 16; 0, 16, 32, 48, … 208, 224, 240
	0101	TIME Strobe asserted for VC0 Count MOD 32; 0, 32, 64, 96, 128, 160, 192, 224
	0110	TIME Strobe asserted for VC0 Count MOD 64; 0, 64, 128, 192
	0111	TIME Strobe asserted for VC0 Count MOD 128; 0, 128
	1000	TIME Strobe asserted for VC0 Count MOD 256; 0
	Other	Invalid setting
PV	0	Idle Telemetry Packet Version Field equals 000
	1	Idle Telemetry Packet Version Field equals 100
ASM	0	Synchmark $1ACF\_FC1D_{16}$ used
	1	Synchmark $352E\_F853_{16}$ used
COW	0	CLCW bit 16 and 17 from TC decoder are not overwritten

Bit/field	Value	Description
	1	CLCW bit 16 and 17 are overwritten with information from *PdecRfAvN$* and *PdecTcAct$* inputs

TME TM Encoding Configuration Register 1 [TME_TmEnCfg1] RW



Bit/field	Value	Description
SCW	00	Select CLCW from the *ClcwD0* input
	01	Select CLCW from the *ClcwD1* input
	10	Select CLCW alternatively from the *ClcwD0* and *ClcwD1* inputs
	11	Select CLCW from *ClcwD0*, *ClcwD1*, *ClcwD2* and *ClcwD3* in a round robin fashion
PS	0	Bandwidth Allocation VCA Selection
	1	Priority VCA Selection
PR	0	Pseudo-Randomise function disabled
	1	Pseudo-Randomise function enabled
CV	0	No Convolutional Encoding
	1	Output Data stream Convolutional Encoded
CvR	00-	Convolute rate 1/2 (with symbol inversion)
	01-	Convolute rate 1/2, no symbol inversion (not [CCSDS] requirement)
	100	Convolute rate 2/3, punctured
	101	Convolute rate 3/4, punctured
	110	Convolute rate 5/6, punctured
	111	Convolute rate 7/8, punctured
RS	0	No Reed-Solomon Encoding
	1	Output Data stream Reed-Solomon Encoded
TU	0	No Turbo Encoding
	1	Output Data stream Turbo Encoded
TuR	00	Turbo nominal code rate = 1/2
	01	Turbo nominal code rate = 1/3
	10	Turbo nominal code rate = 1/4
	11	Turbo nominal code rate = 1/6
TST	0	Test pattern generation disabled
	1	Test pattern generation enabled
TstPat	Any	Select test pattern according to Table 6-9

TME TM Identification Configuration Register [TME_TmIdCfg] RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-															

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						ScId 00_0000_0000									

MSB LSB

Bit/field	Value	Description
ScId	Any	Spacecraft Identification

TME Bandwidth Allocation Table Register 0 [TME_BAT0] RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	r7 111_2			-	r6 110_2			-	r5 101_2			-	r4 100_2		

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	r3 011_2			-	r2 010_2			-	r1 001_2			-	r0 000_2		

MSB LSB

TME Bandwidth Allocation Table Register 1 [TME_BAT1] RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	r15 111_2			-	r14 110_2			-	r13 101_2			-	r12 100_2		

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	r11 011_2			-	r10 010_2			-	r9 001_2			-	r8 000_2		

MSB LSB

TME Bandwidth Allocation Table Register 2 [TME_BAT2] RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	r23 111_2			-	r22 110_2			-	r21 101_2			-	r20 100_2		

MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	r19 011_2			-	r18 010_2			-	r17 001_2			-	r16 000_2		

MSB LSB

TME Bandwidth Allocation Table Register 3 [TME_BAT3] RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-	r31 111_2			-	r30 110_2			-	r29 101_2			-	r28 100_2		

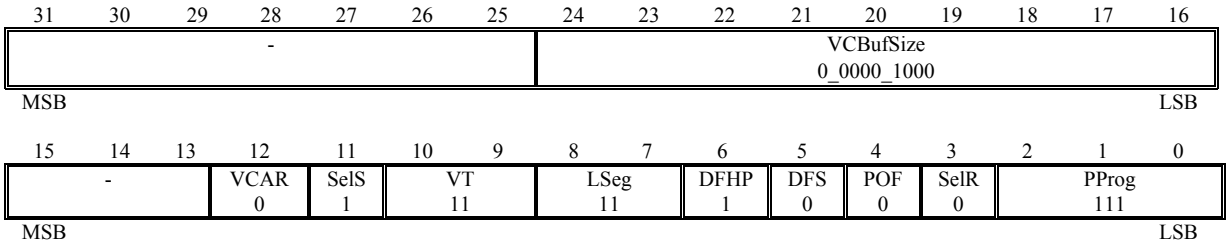
MSB LSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	r27 011_2			-	r26 010_2			-	r25 001_2			-	r24 000_2		

MSB LSB

Bit/field	Value	Description
r$ ($ = 0-31)	000	BAT field set to VC[A]
	001	BAT field set to VC[B]
	010	BAT field set to VC[C]
	011	BAT field set to VC[D]
	100	BAT field set to VC[E]
	101	BAT field set to VC[F]
	110	BAT field set to VC[G]
	111	BAT field set to VC[H]

TME VC Input Configuration Register * [TME_VcCfg*] RW

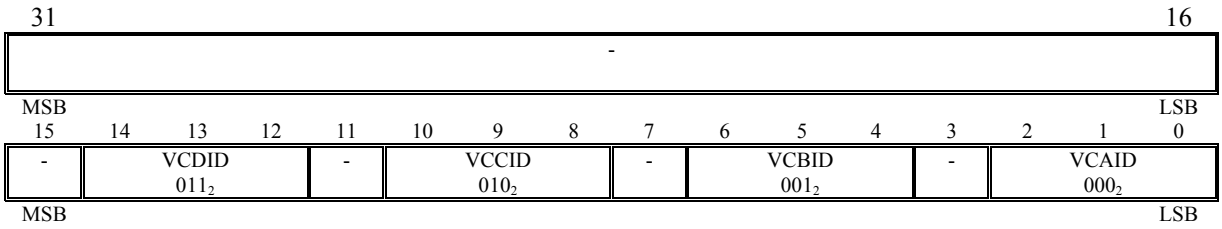


The VC Input Configuration Register index number, * = [A, ..., H]

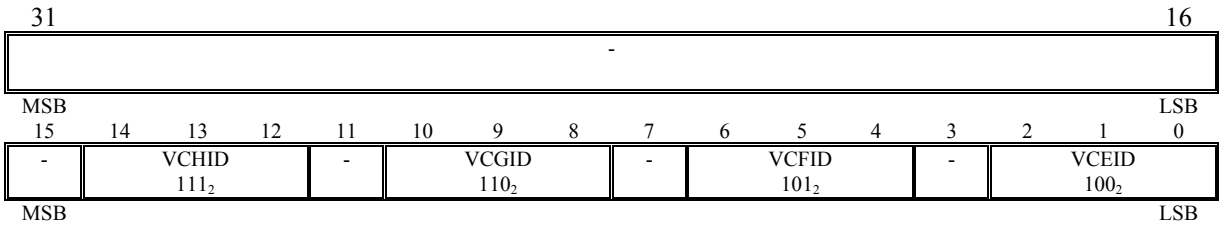
Bit/field	Value	Description
PProg (Valid for VCA[A...D] only. Not write-able for VCA[E...H])	000	Immediate Idle Packet Insertion
	001	1 VC poll
	010	4 VC poll
	011	16 VC poll
	100	64 VC poll
	101	256 VC poll
	110	1024 VC poll
	111	Idle Packet Insertion will not be started
SelR	0	Select nominal input signals (*TmeS**)
	1	Not allowed value
POF	Any	Value to be used for Packet Order flag in the transfer frame
DFS		Value to be used for Data Field Synchronisation flag in the transfer frame
	0	Telemetry Packets
	1	Non-packet data
DFHP		Switch on and off the Dynamic FHP insertion. This field has no effect when DFS = 0 (Telemetry Packets), which always use Dynamic FHP.
	0	Dynamic FHP disabled for Non-packet data
	1	Dynamic FHP enabled for Non-packet data
LSeg	Any	Value to be used for Segmentation Length field in the transfer frame
VT		VC input threshold:
	00	- 262 (256+6) bytes
	01	- 518 (512+6) bytes
	10	- 1030 (1024+6) bytes
	11	- length of transfer frame data field
SelS	0	Select parallel input interface via SpaceWire
	1	Select serial input interface via Packet Wire
VCAR	0	No space left in memory according to VT (read-only)
	1	Space left in memory according to VT (read-only)

Bit/field	Value	Description
VCBufSize	Any	Value reflects the number of frames for which memory space shall be allocated in external buffer memory for the corresponding VCA. The lowest three bits of this field are not writeable, which means that frames are allocated eight at a time. If this field is zero, the VC Input Interface will not respond to any data input.

TME VCID Configuration Register 0 [TME_VcIdCfg0] RW



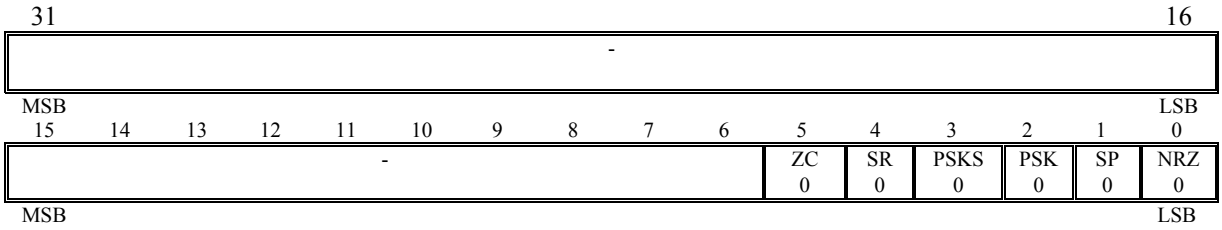
TME VCID Configuration Register 1 [TME_VcIdCfg1] RW



Bit/field	Value	Description
VC*ID (* = A-H)	000	VCID for VC Input Interface * set to 0
	001	VCID for VC Input Interface * set to 1
	010	VCID for VC Input Interface * set to 2
	011	VCID for VC Input Interface * set to 3
	100	VCID for VC Input Interface * set to 4
	101	VCID for VC Input Interface * set to 5
	110	VCID for VC Input Interface * set to 6
	111	VCID for VC Input Interface * set to 7

Note: There is nothing to prevent a user from connecting more than one VC Input Interface to the same VCID. This will however make it hard to determine from which VC the telemetry data was received.

TME TFM Control Register [TME_TfmCtrl] **RW**

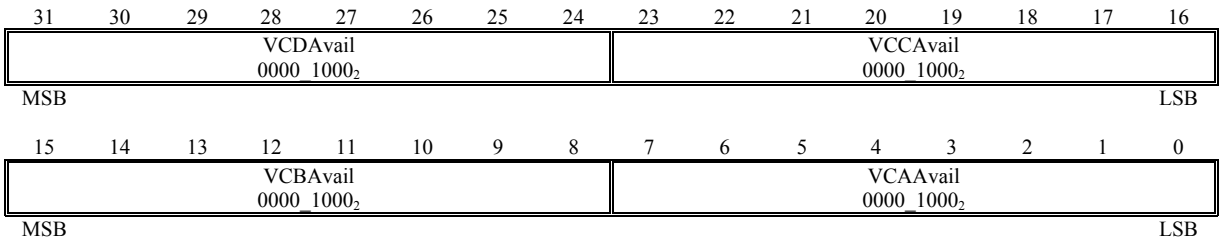


Bit/field	Value	Description
NRZ	0	NRZ-L format selected (Normal digital 1's and 0's)
	1	NRZ-M modulation enabled (Each 1 changes the level of the signal)
SP	0	SP-L modulation disabled
	1	SP-L modulation enabled (Each 1 gives falling edge on signal and each 0 gives rising edge of signal. Needs double output frequency.)
PSK	0	PSK Square modulation disabled
	1	PSK Square modulation enabled
PSKS	0	Must be set to 0.
SR	0	Must be set to 0.
ZC	0	PSK Square output modulation symbol '1' starts with rising edge and symbol '0' with falling.
	1	PSK Square output modulation symbol '0' starts with rising edge and symbol '1' with falling.

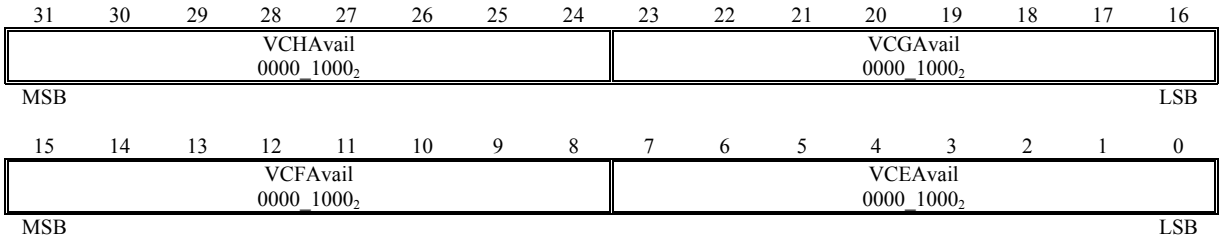
Released

6.10.7.3 Status registers

TME VC Buffer Status Register 0 [TME_VcBufStat0] **R**



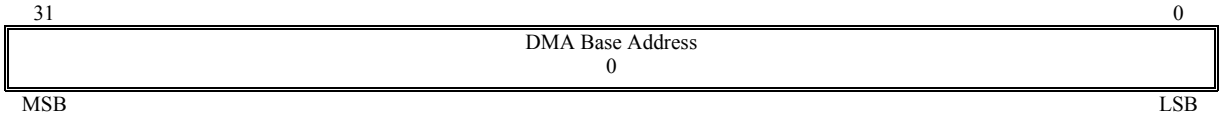
TME VC Buffer Status Register 1 [TME_VcBufStat1] **R**



Bit/field	Value	Description
VC*Avail	0 to 254	Indicates the number of kb of data that can be received (i.e. stored in the memory buffer) by the corresponding VC Input Interface at a certain time.
	255	Indicates that at least 255 kb of data can be received by the corresponding VC Input Interface.

6.10.7.4 DMA registers

TME DMA Base Address Register [TME_TmeDBAR] **RW**



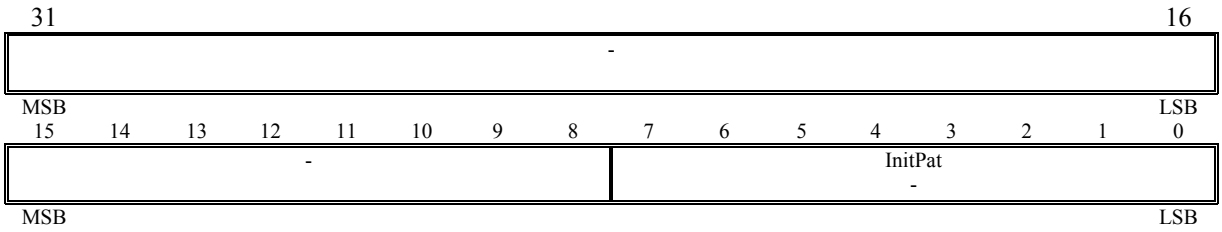
Field	Description
DMA Base Address	The byte base address for a DMA access.

Note: The TmeDBAR cannot be set below address $0300\_4000_{16,}$ since the memory below this address in the *RamCsN* area is used by other modules for dedicated purposes.

Released

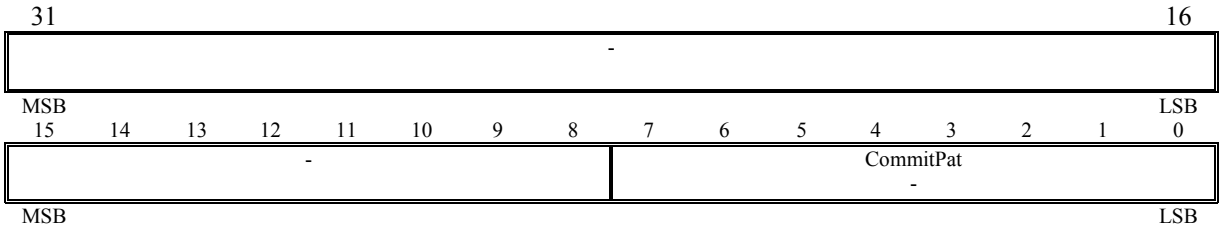
6.10.7.5 Command registers

TME Init TM Register [TME_InitTm] **W**



Bit/field	Value	Description
InitPat	AA_{16}	The TM Encoder operation is reset when AA_{16} is written to this register, including abandoning any data stored in the external memory buffer. The programmed parameters are however not affected.
	Other	No effect

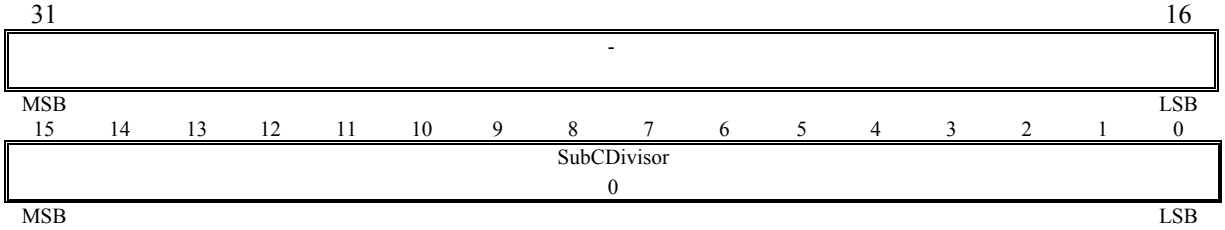
TME TFM Commit Register [TME_TFMCommit] **W**



Bit/field	Value	Description
Commit Pat	AA_{16}	Value contained in TfmCtrl, TfmSubCDiv and TfmBRDiv are committed, which means that the register values will affect the TME.
	Other	No effect

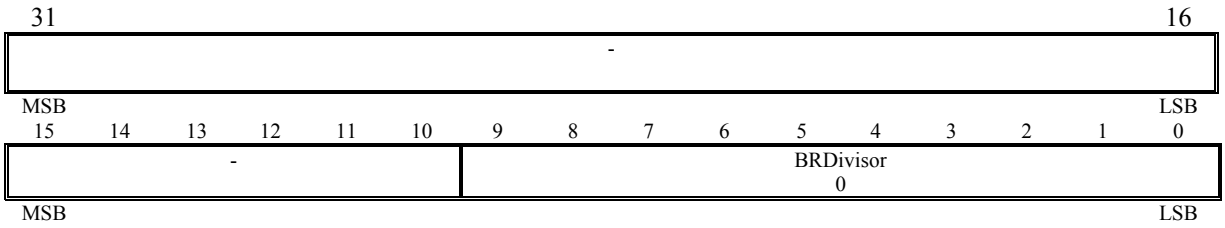
6.10.7.6 Divisor registers

TME TFM SubC Divisor Register [TME_TfmSubCDiv] RW



Bit/field	Description
SubC Divisor	The divisor used to set the frequency of the *SubCClock*. $f_{SubCClk} = f_{TmClk}$ / (SubCDivisor * 2)

TME TFM Bit Rate Divisor Register [TME_TfmBRDiv] RW



Bit/field	Description
BR Divisor	The divisor used to set the frequency of the *BitClock*. $f_{BitClk} = f_{SubCClk}$ / (BRDivisor + 1)

6.10.8 SpaceWire Module (SPW) registers

6.10.8.1 Interrupt registers

SPW Pending Interrupt Masked Status Register [SPW_PIMSR] **R**
SPW Pending Interrupt Masked Register [SPW_PIMR] **R**
SPW Pending Interrupt Status Register [SPW_PISR] **R**
SPW Pending Interrupt Register [SPW_PIR] **R**
SPW Interrupt Mask Register [SPW_IMR] **R**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-															Tx0 0

MSB

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	Addr Err 0	Res-erved	Rx Time Code 0	Cr Err 0	ESC Err 0	Par Err 0	Dis Err 0

LSB

Field	Description
DisErr	Link interface disconnection error detected
ParErr	Link interface parity error detected
ESCErr	Link interface ESC error detected
CrErr	Link interface credit error detected
RxTimeCode	Time-code received
AddrErr	Received destination address out-of-range
Tx	VTC interrupt

Writing to the register can only be done during configuration.

6.10.8.2 Configuration registers

SPW Configuration Register [SPW_CR] **R**

31	6	5	2	1	0
-		DefRout 0		Rout Rx En 0	SelB 0

LSB

Field	Value	Description
SelB	0	Nominal interface A is used (Read only)
	1	Redundant interface B is used (Read only)
RoutRxEn	0	No destination addresses, packets sent to VRC DefRout.
	1	Destination addresses extracted and removed, packets sent to their corresponding destination address
DefRout	0 to 7	Default routing address. This destination address is used if the RoutRxEn field is cleared.

Writing to the register can only be done during configuration.

SPW Link Handler Configuration Register [SPW_SCR] R

31 … 7	6	5 4	3	2	1	0
-	Auto RestartEn 0	Reserved	Tx Def4 Null 0	Auto Link En 0	Link Dis 0	Link Start 0

LSB

Field	Value	Description
LinkStart	0	SpaceWire link cannot proceed to Started state
	1	SpaceWire link can proceed to Started state
LinkDis	0	SpaceWire link enabled
	1	SpaceWire link disabled
AutoLinkEn	0	No auto start
	1	Autostart enabled, enables the link at NULL detection
TxDef4Null	0	Transmission of NULLs at nominal data rate
	1	Transmission of NULLs at default data rate
AutoRestartEn	0	LinkStart and AutoLinkEn bits are cleared when reaching the Run state.
	1	LinkStart and AutoLinkEn unaffected when reaching the Run state.

Writing to the register can only be done during configuration.

SPW Tick Number Register [SPW_TNR] R

31 … 25	24 … 16
-	Ticks850ns $1FF_{16}$

MSB

15 … 13	12 … 0
-	Ticks12u8 $1FFF_{16}$

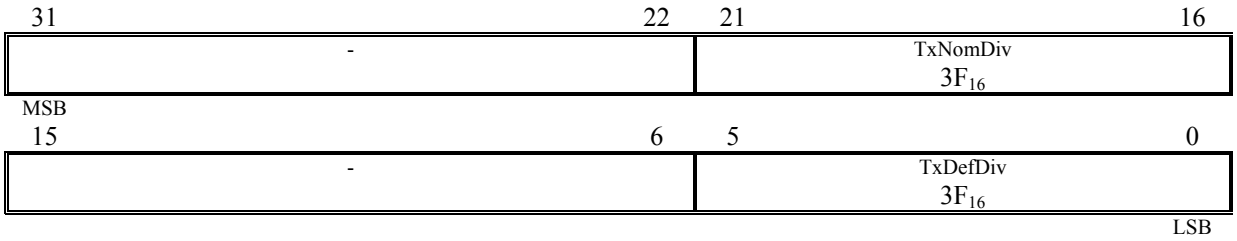
LSB

Field	Description
Ticks850ns	Number of *SpwClk* periods for the time-out of 850 ns (e.g. *SpwClk* is 20 MHz, Ticks850ns needs to be 17). A value of zero indicates immediate time-out.
Ticks12u8	Number of *SpwClk* periods for the time-out of 12.8 us (e.g. *SpwClk* is 20 MHz, Ticks12u8 needs to be 256). A value of zero indicates immediate time-out.

Writing to the register can only be done during configuration.

Released

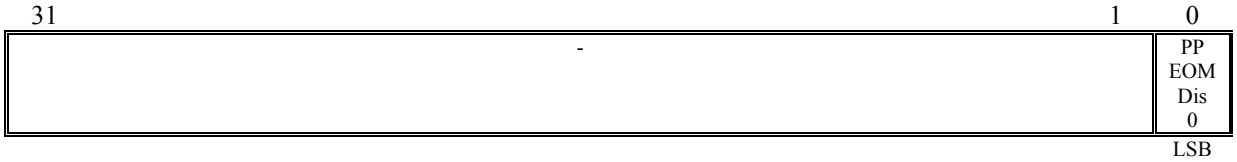
SPW Clock Division Register [SPW_CDR] R



Field	Description
TxNomDiv	The nominal transmitter data rate is *SpwClk* / (TxNomDiv + 1) (e.g. *SpwClk* 60 MHz, transmitter data rate should be 30 Mhz => TxNomDiv needs to be 1)
TxDefDiv	The default transmitter data rate of 10 MHz is *SpwClk* / (TxDefDiv + 1) (e.g. *SpwClk* is 30 Mhz => TxDefDiv needs to be 2)

Writing to the register can only be done during configuration.

SPW Parallel Write Configuration Register $ [SPW_PWCR$] R



Field	Value	Description
PPEOMDis	0	EOM detection enabled, EOM indicated by two EOPs
	1	EOM detection disabled, EOM indicated by a single EOP or EEP

Writing to the register can only be done during configuration.

SPW Transmit Channel Configuration Register [SPW_TCCR] R

31 … 9	8	7 … 4	3 … 1	0
-	CCSDSEn 0	SplitSize 0	PktHeadLen 0	PktHeadEn 0

LSB

Field	Value	Description
PktHeadEn	0	Destination address insertion disabled
	1	Destination address insertion enabled
PktHeadLen	Any	Number of destination address header bytes inserted = PktHeadLen + 1
SplitSize	0000_2	No packet splitting
	1-15	Packet split length = $2^{(SplitSize - 1)}$
CCSDSEn	0	Transmission of SpaceWire packets
	1	Transmission of CCSDS packets

Writing to the register can only be done during configuration.

SPW Packet Header High Word Register [SPW_PHHWR] R

31 … 24	23 … 16
PktHead0 0	PktHead1 0

MSB

15 … 8	7 … 0
PktHead2 0	PktHead3 0

LSB

Field	Description
PktHead$	Destination address value

Writing to the register can only be done during configuration.

SPW Packet Header Low Word Register [SPW_PHLWR] R

31 … 24	23 … 16
PktHead4 0	PktHead5 0

MSB

15 … 8	7 … 0
PktHead6 0	PktHead7 0

LSB

Field	Description
PktHead$	Destination address value

Writing to the register can only be done during configuration.

6.10.8.3 Status registers

SPW Status Register [SPW_SR] **R**

31 … 9	8 … 5	4 … 2	1	0
-	ActiveVRC 0	ActiveVTC 0	VRC Act 0	VTC Act 0

LSB

Field	Value	Description
VTCAct	0	All VTCs are inactive
	1	A VTC is active
VRCAct	0	All VRCs are inactive
	1	A VTC is active
ActiveVTC	0	The last selected VTC
ActiveVRC	0 to 7	The last selected VRC

SPW Link Status Register [SPW_SSR] **R**

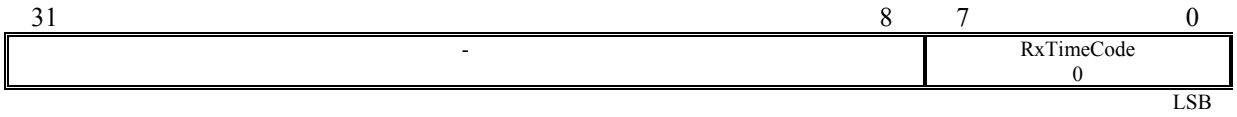
31 … 5	4	3	2 … 0
-	Tx Buf 1	Rx Buf 0	LinkState 0

MSB

Field	Value	Description
LinkState	000_2	SpaceWire link state machine in ErrorReset state
	001_2	SpaceWire link state machine in ErrorWait state
	010_2	SpaceWire link state machine in Ready state
	011_2	SpaceWire link state machine in Started state
	100_2	SpaceWire link state machine in Connecting state
	101_2	SpaceWire link state machine in Run state
RxBuf	0	The receive buffer is empty
	1	The receive buffer is not empty
TxBuf	0	The transmit buffer is empty
	1	The transmit buffer is not empty The reset value is one because the internal transmit buffer empty signal of the link is deasserted during reset. If no transmission has been started this value will be cleared after writing to the SPW Tick Number Register, i.e. when releasing the link reset.

Released

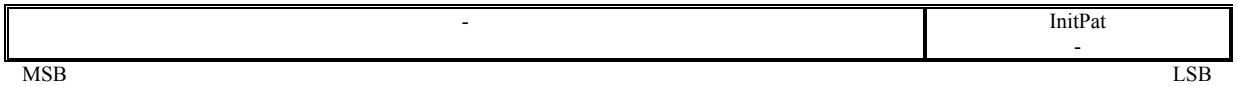
SPW Receive Time Code Register [SPW_RTCR] **R**



Field	Description
RxTimeCode	The last received time-code

6.10.8.4 Command registers

SPW Initialisation Register [SPW_IR] **N/A**

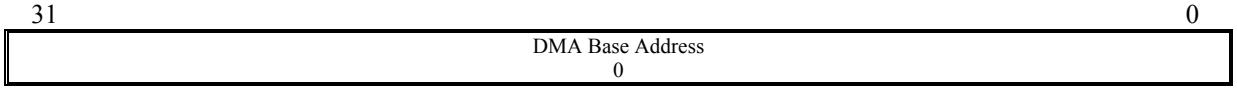


Field	Value	Description
InitPat	AA_{16}	Resets and initialises the SpaceWire module without affecting register values.

Writing to the register can only be done during configuration.

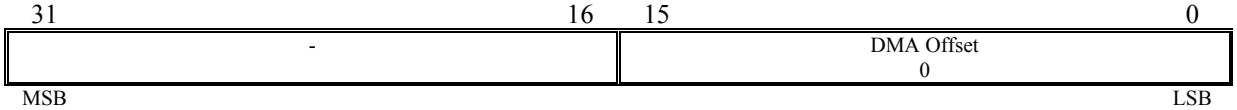
6.10.8.5 SpaceWire Read DMA registers

SPW Read DMA Base Address Register_0 [SPW_RDBAR0] **RW**



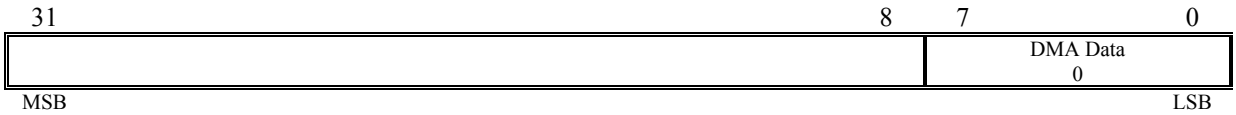
Field	Description
DMA Base Address	Byte base address for a DMA access.

SPW Read DMA Offset Register 0 [SPW_RDOR0] **RW**



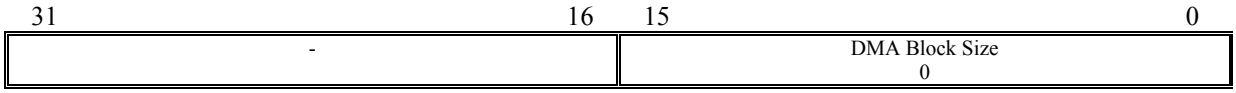
Field	Description
DMA Offset	Byte address pointer relative to DMA Base Address. The DMA access address is DMA Base Address + DMA Offset.

SPW Read DMA Data Register [SPW_RDDR] RW



Field	Description
DMA Data	DMA data field.

SPW Read DMA Block Size Register 0 [SPW_RDBSR0] RW



Field	Description
DMA Block Size	Total number of bytes to be transferred during a DMA block transfer. Writing to this register is used as an automatic start of a DMA block transfer.

Released

6.10.9 Control Interface Module (CI) registers

6.10.9.1 Interrupt registers

CI Pending Interrupt Masked Status Register [CI_PIMSR] **R**
CI Pending Interrupt Masked Register [CI_PIMR] **R**
CI Pending Interrupt Status Register [CI_PISR] **R**
CI Pending Interrupt Register [CI_PIR] **RW**
CI Interrupt Mask Register [CI_IMR] **RW**

31	3	2	1	0
-	Incomplete 0	Overflow 0	Bus Error 0	Illegal 0

MSB LSB

Field	Description
Illegal	A packet containing an illegal access type has been received
BusError	An error occurred during memory access
Overflow	A packet containing more data than specified in the packet header was received
Incomplete	An end of packet was received before all expected data of the packet had been received

PWT Pending Interrupt Masked Status Register [PWT_PIMSR] **R**
PWT Pending Interrupt Masked Register [PWT_PIMR] **R**
PWT Pending Interrupt Status Register [PWT_PISR] **R**
PWT Pending Interrupt Register [PWT_PIR] **RW**
PWT Interrupt Mask Register [PWT_IMR] **RW**

31	3	2	1	0
-		DMA Error 0	DMA Comp-lete 0	Free 0

MSB LSB

Field	Description
DMA Error	Error during DMA block transfer
DMA Complete	DMA block transfer complete
Free	The PWT DMA Data Register is free to receive another byte

Released

6.10.9.2 Configuration Registers

PWT Clock Configuration Register [PWT_CLK] **RW**

31 … 9	8 … 1	0
-	Period 0	-

MSB LSB

Field	Value	Description
Period	0-255	The period of the transmission clock, *CiOutClk*, is: $T_{CiOutClk} = 2 \cdot (1 + Period) \cdot T_{BusClk}$ *Period* is the value of Period $T_{CiOutClk}$ is the period if the transmission clock T_{SysClk} is the period of the system clock

6.10.9.3 Status Registers

PWT Status Register [PWT_STAT] **R**

31 … 4	3	2	1	0
-	Dirty 0	Ready -	Valid 0	DMA Block 0

MSB LSB

Field	Description
Dirty	PWT_DDR is NOT free to receive a new byte when this bit is set
Ready	The current value of *CiOutRdy*
Valid	The current value of *CiOutValid*
DMA Block	There is an ongoing DMA block transmission when this bit is set

6.10.9.4 DMA registers

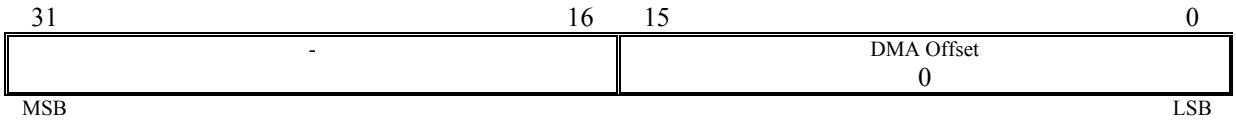
PWT DMA Base Address Register [PWT_DBAR] **RW**

31 … 0
DMA Base Address 0

MSB LSB

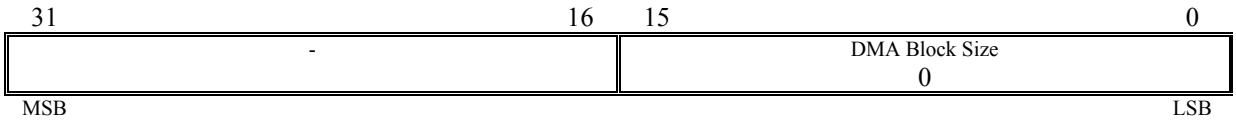
Field	Description
DMA Base Address	Byte base address for a DMA access.

PWT DMA Offset Register [PWT_DOR] RW



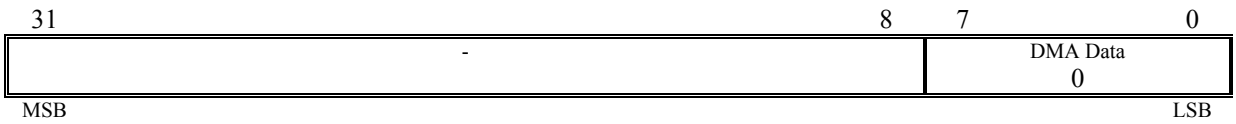
Field	Description
DMA Offset	Byte address pointer relative to DMA Base Address. The DMA access address is DMA Base Address + DMA Offset.

PWT DMA Block Size Register [PWT_DBSR] RW



Field	Description
DMA Block Size	Total number of bytes to be transferred during a DMA block transfer. Writing this register clears the PWT DMA Offset Register and starts automatically a DMA block transfer.

PWT DMA Data Register [PWT_DDR] RW



Field	Description
DMA Data	DMA data field.

Released

6.11 Memory Usage and Mapping

6.11.1 Configuration Block memory usage

The configuration of the SCTMTC ASIC is done automatically at power-on reset. The sequence is shown in Table 6-18. The sequence is executed from the location where the configuration pointer is pointing to, until the end of sequence. The two refresh level sequences, user and system, are included in the power-on sequence as indicated in the table. The configuration sequence has an initialisation part which is executed once before the user level refresh sequences, and a completion part that is executed once after the system level refresh sequences.

The initialisation part configures all registers that are protected by Triple Modular Redundancy (TMR) and are critical to the enabling, clocking and resetting of the different modules. It also sets up the configuration sequence. The completion part commits all the changes to e.g. telemetry encoder etc., starts the SpaceWire links, and starts the counters for user and system level refresh.

The information in Table 6-18 should be interpreted as follows:

- The *Address* column indicates the byte address in the PROM that contains the data listed in the *Data* column.
- The *Address* column should be incremented by the user, each byte in the *Data* column corresponds to one *Address* increment.
- The Data column contains fixed and user programmable register contents, or just fixed data.
- The fixed register content shall not be modified by the user.
- The user programmable register content must be defined by the user. The corresponding bit definition can be found in the preceding sections.
- Some register content is defined as 16 bits, others as 32 bits. It is not allowed to provide a different data amount for a register than specified in the *Data* column.
- Fixed data in the *Data* column shall not be modified by the user.
- For both the *Address* and the *Data* column, only data marked as ‘X’ may be modified by the user.
- All values in both the *Address* and the *Data* column are presented in the hexadecimal format.

The periodicity of the user level refresh sequence is programmed with the GenClk1 registers.

The periodicity of the user level refresh sequence is programmed with the GenClk2 registers.

Address	Data	Comment
Configuration pointer setup		
000_02C0_{16} – 000_02C3_{16}	0000_1000_{16}	Configuration pointer
Power-on Configuration - Initialisation		
000_1000_{16} – 000_1033_{16}	1070_0080_{16}	CAR registers
	2034_{16}	CAR Arm Reset Register
	$001F_{16}$	Fixed register data [15:0]
	2030_{16}	CAR Reset Register
	$001F_{16}$	Fixed register data [15:0]
	2024_{16}	CAR Arm Clock Source Register
	$000F_{16}$	Fixed register data [15:0]
	2020_{16}	CAR Clock Source Register
	$XXXX_{16}$	User configurable register data [15:0]
	2080_{16}	CAR TM Clock Source Register
	$XXXX_{16}$	User configurable register data [15:0]
	$202C_{16}$	CAR Arm Clock Enable Register
	$001F_{16}$	Fixed register data [15:0]
	2028_{16}	CAR Clock Enable Register
	$XXXX_{16}$	User configurable register data [15:0]
	$203C_{16}$	CAR Arm Pdec3 Csel Connect Register
	0001_{16}	Fixed register data [15:0]
	2038_{16}	CAR Pdec3 Csel Connect Register
	$XXXX_{16}$	User configurable register data [15:0]
	2054_{16}	CAR CtrlIf Select Register
	$XXXX_{16}$	User configurable register data [15:0]
	2034_{16}	CAR Arm Reset Register
	$001F_{16}$	Fixed register data [15:0]
	2030_{16}	CAR Reset Register
	$XXXX_{16}$	User configurable register data [15:0]
000_1034_{16} – 000_104B_{16}	1070_0300_{16}	Configuration registers
	3020_{16}	Fixed data
	0000_{16}	Fixed data
	$2F38_{16}$	Fixed data
	3024_{16}	Fixed data
	0003_{16}	Fixed data
	0000_{16}	Fixed data
	2010_{16}	Fixed data
	0007_{16}	Fixed data
	$200C_{16}$	Fixed data
	0007_{16}	Fixed data

Released

000_104C_{16} – 000_104D_{16}	0000_{16}	End of power-on configuration initialisation sequence

Power-on Configuration and System Level Refresh		
000_1060_{16} – 000_1075_{16}	1070_0100_{16}	CSEL registers
	3020_{16}	CSEL 1 Second Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	2024_{16}	CSEL Max TC Only Time Register
	$XXXX_{16}$	User configurable register data [15:0]
	2028_{16}	CSEL Remote Status Timeout Time Register
	$XXXX_{16}$	User configurable register data [15:0]
	2010_{16}	CSEL Interrupt Mask Register
	$00XX_{16}$	User configurable register data [15:0]
000_1076_{16} – 000_1091_{16}	1070_00C0_{16}	CPDM registers
	3020_{16}	CPDM Pulse Duration Register
	$00XX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	2024_{16}	CPDM Clock Ratio Register
	$000X_{16}$	User configurable register data [15:0]
	3028_{16}	CPDM Configuration Register
	$000X_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$202C_{16}$	CPDM Lockout Register
	$XXXX_{16}$	User configurable register data [15:0]
	2010_{16}	CPDM Interrupt Mask Register
	$000X_{16}$	User configurable register data [15:0]
000_1092_{16} – 000_1099_{16}	1070_03C0_{16}	TME registers
	2028_{16}	TM Identification Configuration Register
	$XXXX_{16}$	User configurable register data [15:0]
000_109A_{16} – 000_10A1_{16}	1070_0C80_{16}	TME registers
	3000_{16}	TME DMA Base Address Register
	$XXXX\_XXXX_{16}$	User configurable register data [31:0]
000_10A2_{16} – 000_10EF_{16}	1070_0340_{16}	SPW registers
	2020_{16}	SPW Configuration Register
	0002_{16}	Fixed data
	2024_{16}[1]	SPW Link Handler Configuration Register
	$XXXX_{16}$[1]	User configurable register data [15:0]
	3028_{16}[1]	SPW Tick Number Register
	$XXXX_{16}$[1]	User configurable register data [31:16]
	$XXXX_{16}$[1]	User configurable register data [15:0]
	$302C_{16}$[1]	SPW Clock Division Register
	$XXXX_{16}$[1]	User configurable register data [31:16]

Released

	$XXXX_{16}$[1]	User configurable register data [15:0]
	$21F0_{16}$	SPW Parallel Write Configuration Register 7
	$XXXX_{16}$	User configurable register data [15:0]
	2104_{16}	SPW Transmit Channel Configuration Register 0
	$XXXX_{16}$	User configurable register data [15:0]
	3184_{16}	SPW Packet Header High Word Register 0
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3108_{16}	SPW Packet Header Low Word Register 0
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	2180_{16}	SPW Parallel Write Configuration Register 0
	$XXXX_{16}$	User configurable register data [15:0]
	2190_{16}	SPW Parallel Write Configuration Register 1
	$XXXX_{16}$	User configurable register data [15:0]
	$21A0_{16}$	SPW Parallel Write Configuration Register 2
	$XXXX_{16}$	User configurable register data [15:0]
	$21B0_{16}$	SPW Parallel Write Configuration Register 3
	$XXXX_{16}$	User configurable register data [15:0]
	$21C0_{16}$	SPW Parallel Write Configuration Register 4
	$XXXX_{16}$	User configurable register data [15:0]
	$21D0_{16}$	SPW Parallel Write Configuration Register 5
	$XXXX_{16}$	User configurable register data [15:0]
	$21E0_{16}$	SPW Parallel Write Configuration Register 6
	$XXXX_{16}$	User configurable register data [15:0]
	3010_{16}	SPW Interrupt Mask Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
000_10F0_{16} – 000_10FF_{16}	1070_0480_{16}	CI registers
	2424_{16}	PWT Clock Configuration Register
	$XXXX_{16}$	User configurable register data [15:0]
	2410_{16}	PWT Interrupt Mask Register
	$XXXX_{16}$	User configurable register data [15:0]
	2010_{16}	CI Interrupt Mask Register
	$XXXX_{16}$	User configurable register data [15:0]
000_1100_{16} – 000_1119_{16}	1070_0400_{16}	Configuration registers
	2010_{16}	GenClk Interrupt Mask Register

	0022_{16}	Fixed data
	$308C_{16}$	GenClk 1 Period Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3090_{16}	GenClk 1 Assert Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3094_{16}	GenClk 1 Deassert Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
000_111A_{16} – 000_1171_{16}	1070_0000_{16}	Memory interface registers
	$202C_{16}$	PIM TME Ratio Register
	$XXXX_{16}$	User configurable register data [15:0]
	$20C0_{16}$	PIM MUnit 0 Register (*PromCsN*)
	$XXXX_{16}$	User configurable register data [15:0]
	$20C8_{16}$	PIM MUnit 2 Register (*ExtRecLacN*)
	$XXXX_{16}$	User configurable register data [15:0]
	$20CC_{16}$	PIM MUnit 3 Register (*RamCsN*)
	$XXXX_{16}$	User configurable register data [15:0]
	$311C_{16}$	PIM MUnit User Area 1 Address Low Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3120_{16}	PIM MUnit User Area 2 Address Low Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3124_{16}	PIM MUnit TME / PM Address Low Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3140_{16}	PIM MUnit 0 Address High Register (*PromCsN*)
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3148_{16}	PIM MUnit 2 Address High Register (*ExtRecLacN*)
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$315C_{16}$	PIM MUnit User Area 1 Address High Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3160_{16}	PIM MUnit User Area 2 Address High Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3164_{16}	PIM MUnit TME / PM Address High Register
	$XXXX_{16}$	User configurable register data [31:16]

Released

	$XXXX_{16}$	User configurable register data [15:0]
	2020_{16}	PIM Scrub Configuration Register
	$XXXX_{16}$	User configurable register data [15:0]
	3024_{16}	PIM Scrub Start Address Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3028_{16}	PIM Scrub End Address Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	2060_{16}	PIM Status Set Register
	$XXXX_{16}$	User configurable register data [15:0]
000_1172_{16} – 000_1173_{16}	0000_{16}	End of system level refresh sequence

[1] Replace all these halfwords with 9000_{16} if SPW is not enabled.

Released

Power-on Configuration and User Level Refresh		
000_1180_{16} – 000_1187_{16}	1070_0280_{16}	PDEC3 registers
	2010_{16}	PDEC3 Interrupt Mask Register
	$XXXX_{16}$	User configurable register data [15:0]
000_1188_{16} – 000_11F3_{16}	1070_03C0_{16}	TME registers
	2020_{16}	TME TM Encoding Configuration Register 0
	$XXXX_{16}$	User configurable register data [15:0]
	2024_{16}	TME TM Encoding Configuration Register 1
	$XXXX_{16}$	User configurable register data [15:0]
	3080_{16}	TME Bandwidth Allocation Table Register 0
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3084_{16}	TME Bandwidth Allocation Table Register 1
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	3088_{16}	TME Bandwidth Allocation Table Register 2
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$308C_{16}$	TME Bandwidth Allocation Table Register 3
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30A0_{16}$	TME VC Input Configuration Registers A
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30A4_{16}$	TME VC Input Configuration Registers B
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30A8_{16}$	TME VC Input Configuration Registers C
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30AC_{16}$	TME VC Input Configuration Registers D
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30B0_{16}$	TME VC Input Configuration Registers E
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30B4_{16}$	TME VC Input Configuration Registers F

Released

	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30B8_{16}$	TME VC Input Configuration Registers G
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30BC_{16}$	TME VC Input Configuration Registers H
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$20E0_{16}$	TME VCID Configuration Register 0
	$XXXX_{16}$	User configurable register data [15:0]
	$20E4_{16}$	TME VCID Configuration Register 1
	$XXXX_{16}$	User configurable register data [15:0]
	2130_{16}	TME TFM Control Register
	$XXXX_{16}$	User configurable register data [15:0]
	2140_{16}	TME TFM SubC Divisor Register
	$XXXX_{16}$	User configurable register data [15:0]
	2144_{16}	TME TFM Bit Rate Divisor Register
	$XXXX_{16}$	User configurable register data [15:0]
	2010_{16}	TME Interrupt Mask Register
	$000X_{16}$	User configurable register data [15:0]
000_11F4_{16} – 000_1203_{16}	1070_0000_{16}	Memory interface registers
	2068_{16}	PIM Write Disable Set Register
	$000X_{16}$	User configurable register data [15:0]
	$206C_{16}$	PIM Write Disable Clear Register
	$000X_{16}$	User configurable register data [15:0]
	2010_{16}	PIM Interrupt Mask Register
	$XXXX_{16}$	User configurable register data [15:0]
000_1204_{16} – 000_122A_{16}	1070_0400_{16}	Configuration registers
	2010_{16}	GenClk Interrupt Mask Register
	0022_{16}	Fixed data
	$30AC_{16}$	GenClk 2 Period Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30B0_{16}$	GenClk 2 Assert Register
	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	$30B4_{16}$	GenClk 2 Deassert Register

Released

	$XXXX_{16}$	User configurable register data [31:16]
	$XXXX_{16}$	User configurable register data [15:0]
	2084_{16}	GenClk 1 Control Set Register
	0001_{16}	Fixed data
	$20A4_{16}$	GenClk 2 Control Set Register
	0001_{16}	Fixed data
	2060_{16}	GenClk SW Start Register
	0000_{16}	Fixed register data [15:0]
000_122B_{16} – 000_122C_{16}	0000_{16}	End of user level refresh sequence

Configuration Sequence - Completion		
000_1250_{16} – 000_125B_{16}	1070_03C0_{16}	TME registers
	2148_{16}	TME TFM Commit Register
	$00AA_{16}$	Fixed register data [15:0]
	2060_{16}	TME Initialise TM Register
	$00AA_{16}$	Fixed register data [15:0]
000_125C_{16} – 000_125D_{16}	0000_{16}	End of power-on configuration completion sequence

Released

Configuration Data		
000_0E00_{16} – 000_0E01_{16}	0000_{16}	Fixed data
000_0400_{16} – 000_0403_{16}	0000_1250_{16}	Fixed data
000_0404_{16} – 000_0407_{16}	0000_1180_{16}	Fixed data
000_0408_{16} – 000_040B_{16}	0000_1060_{16}	Fixed data
000_040C_{16} – 000_0483_{16}	0000_0E00_{16} – 0000_0E00_{16}	Fixed data
000_0484_{16} – 000_0487_{16}	0000_1060_{16}	System level refresh pointer
000_0488_{16} – 000_0493_{16}	0000_0E00_{16} – 0000_0E00_{16}	Fixed data
000_0494_{16} – 000_0497_{16}	0000_1180_{16}	User level refresh pointer
000_0498_{16} – 000_0CFF_{16}	0000_0E00_{16} – 0000_0E00_{16}	Fixed data.

Table 6-18 Configuration PROM memory contents

Released

6.11.3 Packet Telecommand Decoder Module (PDEC3) memory usage

6.11.3.1 PROM Usage

Two areas are mapped as described in Table 6-19 and Figure 6-13. The first area is used for initialisation data, while the second area is used for the fixed authentication key.

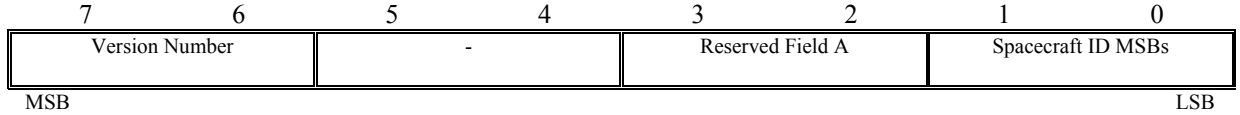
Byte address	Contents
000_0000_{16} – 000_016F_{16}	Authentication Key, starting with W0(40..47)
000_0200_{16}	Frame Header first octet, starting with Version Number field
000_0201_{16}	Frame Header second octet, Spacecraft Id LSBs
000_0202_{16}	Frame Header third octet, starting with Virtual Channel Id
000_0203_{16}	PW, Positive Window
000_0204_{16}	NW, Negative Window
000_0205_{16}	Authenticated MAP Id Pointer (only 5 LSBs used)
000_0206_{16}	De-randomiser Configuration
000_0207_{16}	Recovery LAC Configuration
000_0208_{16} - 000_023F_{16}	Unused
000_0240_{16} - 000_027F_{16}	MapClkFreq look-up table, starting with MapFreq[0]
000_0280_{16} - 000_02BF_{16}	MapAdr look-up table, starting with MapAdr[0]

Table 6-19 External PROM memory map

The contents of the fixed authentication key are depicted in Figure 6-13.

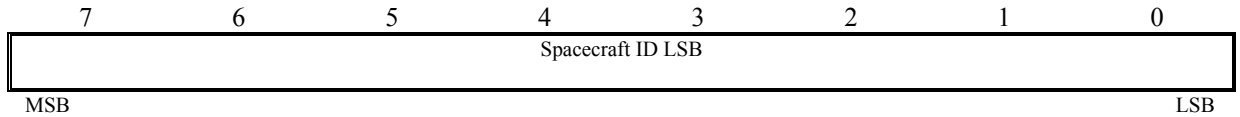
The contents of each of the configuration bytes are described below.

Frame Header first octet



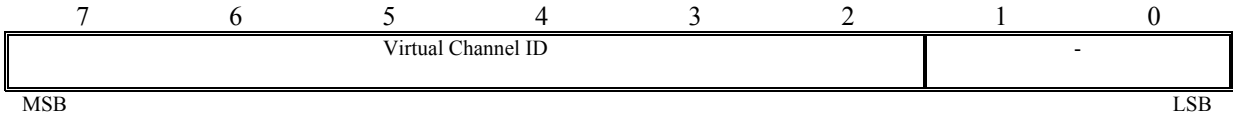
Note: To be compliant with [TC_STD] the Version Number and Reserved Field A shall be set to 0.

Frame Header second octet

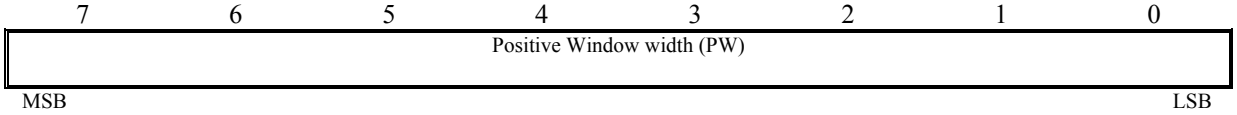


Released

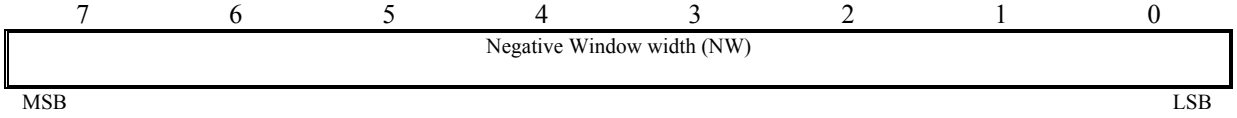
Frame Header third octet



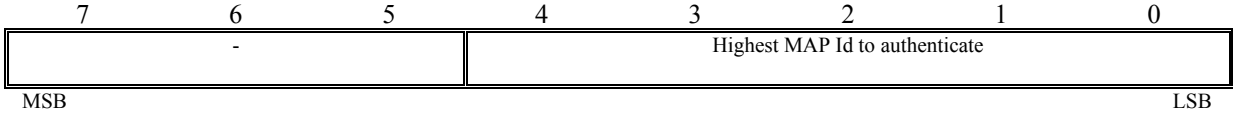
Positive Window



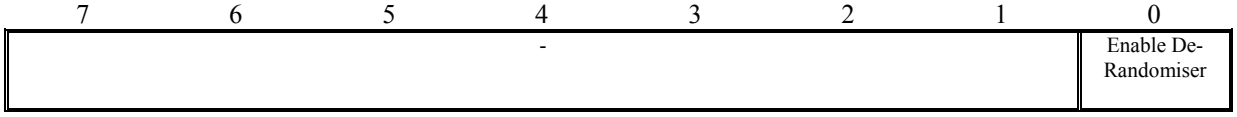
Negative Window



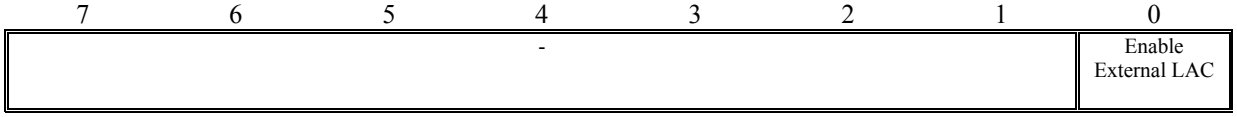
Authentication MAP Id Pointer



De-randomiser Configuration

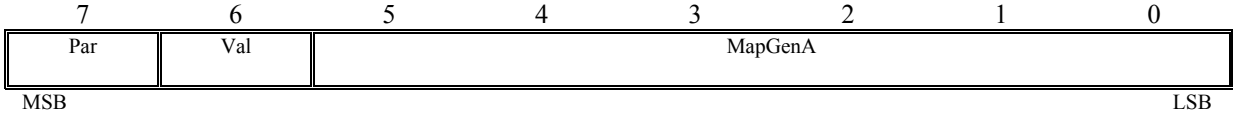


Recovery LAC Configuration



The entries in the MapAdr lookup table shall be as follows, Par being odd parity:

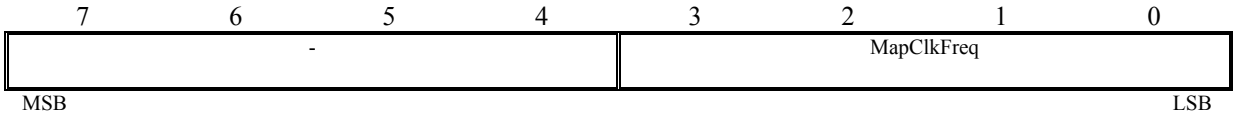
MapAdr lookup values



For a more detailed explanation of the MapAdr lookup values, refer to section 6.3.2.8.

The contents of the MapClkFreq lookup table shall be as follows:

MapClkFreq lookup values



Note: The values of MapClkFreq should be within the range 1-13 to be considered valid. Other values result in the segment being discarded.

For a more detailed explanation of the MapFreq lookup values, refer to section 6.3.2.8.

6.11.3.2 SRAM Usage

Two SRAM areas are mapped as described in Table 6-20 and Table 6-21. The first SRAM area is used for buffers to store incoming frames, while the second area is used for the programmable authentication key.

Byte address	Contents
300_0000_{16} - 300_03FF_{16}	Buffer 0 storage area
300_0400_{16} - 300_07FF_{16}	Buffer 1 storage area
300_0800_{16} - 300_0BFF_{16}	Buffer 2 storage area

Table 6-20 External SRAM memory map 1

Byte address	Contents
300_3E90_{16} - 300_3FFF_{16}	Authentication Key, starting with W0(40..47)

Table 6-21 External SRAM memory map 2

The contents of the programmable authentication key are depicted in Figure 6-13.

6.11.3.3 Authentication Keys

Both Authentication Keys have the structure shown in Figure 6-13, consisting of:

- 60 Hard Knapsack weights numbered W0 to W59, each 48 bits long with bit 0 being the MSB and bit 47 being the LSB.
- The 60-bit Hashing Function coefficient C.

Bank	Command address	PROM memory address	Authentication Key			
A	0	000_0000_{16}	40	W0(40..47)		47
	1	000_0001_{16}	32	W0(32..39)		39
	2	000_0002_{16}	24	W0(24..31)		31
	3	000_0003_{16}	16	W0(16..23)		23
	4	000_0004_{16}	8	W0(8..15)		15
	5	000_0005_{16}	0	W0(0..7)		7
	6	000_0006_{16}	40	W1(40..47)		47
	7	000_0007_{16}	32	W1(32..39)		39
			etc.			
	255	000_00FF_{16}	16	W42(16..23)		23
B	0	000_0100_{16}	8	W42(8..15)		15
			etc.			
	103	000_0167_{16}	0	W59(0..7)		7
	104	000_0168_{16}		(four unused bits)	59 C(59..56)	56
	105	000_0169_{16}	55	C(55 to 48)		48
	106	000_016A_{16}	47	C(47 to 40)		40
	107	000_016B_{16}	39	C(39 to 32)		32
	108	000_016C_{16}	31	C(31 to 24)		24
	109	000_016D_{16}	23	C(23 to 16)		16
	110	000_016E_{16}	15	C(15 to 8)		8
	111	000_016F_{16}	7	C(7 to 0)		0

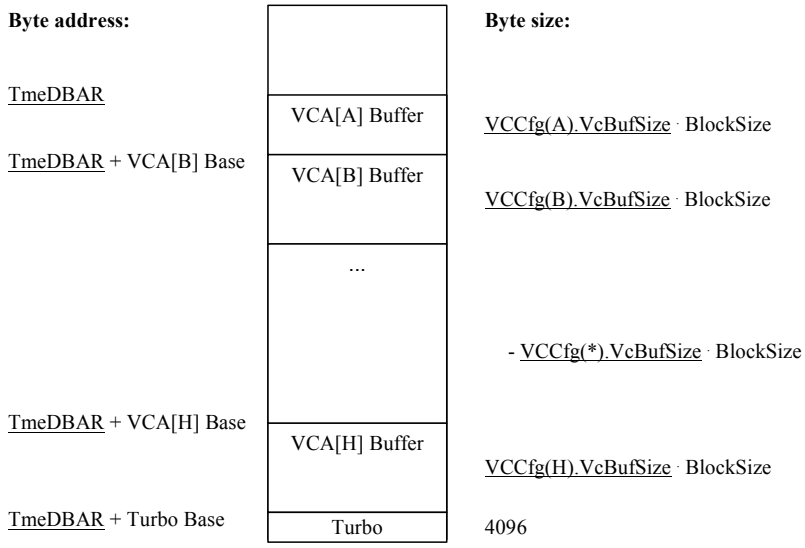
Figure 6-13 Authentication Key addressing and configuration PROM memory map

6.11.3.4 External Recovery LAC

The external recovery LAC is one byte and should be located in non-volatile writeable memory at address 200_0000_{16}.

6.11.7 Packet Telemetry Encoder (TME) memory usage

The memory map used by the TME is described in the picture below. BlockSize is defined in Table 6-8. TmeDBAR is defined in the TME DMA Base Address Register.



$$\text{VCA[*] Base} = \left(\sum_{n=A}^{*} \text{VCCfg(n).VcBufSize} \cdot \text{BlockSize} \right)$$

$$\text{Turbo Base} = \left(\sum_{n=A}^{H} \text{VCCfg(n).VcBufSize} \cdot \text{BlockSize} \right)$$

Figure 6-14 TME memory map

7 HARDWARE INTERFACE

7.1 General SCTMTC ASIC Functions

7.1.1 Internal Scan Controller block

The SCTMTC ASIC has several test inputs and outputs. All test inputs should be connected to logical zero during normal operation.

7.1.2 Test Access Port (TAP) block

See section 7.18 for the order of the boundary scan chain.

7.1.2.1 Test Access Port (TAP) register

The TAP interface includes an ID register, which can be read out using the JTAG interface.

Test Access Port Register **R**

31 28	27 12	11 1	0
Version 0001	Part Number $TE3_{36} = 94CB_{16}$	Manufacturer ID 58_{16}	- 1
MSB			LSB

Field	Description
	Constant one
Manufacturer ID	An 11 bit code which defines the manufacturer of the chip. This is the compressed manufacturer JEDEC code as defined by the standard.
Part Number	A 16-bit code that defines the part number of the chip, supplied by the manufacturer, as defined by the standard.
Version	The four bit chip version number

7.1.2.2 Test Access Port (TAP) interface

The TAP signals have the following internal pull:

TAP signal	Pull	Note
JtagTck	Down	
JtagTdi	Down	
JtagTms	Down	
JtagTrstN	Down	The pull down is none compliant with standard but better for failsafe and SEU reasons, since it will keep TAP controller logic disabled at all time except for board test.
JtagTdo	-	

Table 7-1 TAP signals' pull

Note that *JtagTrstN* shall be tied to ground during mission.

7.1.3 Clock and Reset (CAR) block

The Clock and Reset block distributes the *PoResetN* signal through out the chip. The block also allows changes in configuration of some of the modules inside the chip see chapter 5 for details, e.g. change in clock source to the SpaceWire and the TME modules.

7.1.3.1 Disabling a module

A disabling of a module after Power-On reset using the **CAR Arm Reset Register [ArmModReset]** have effect within 5 *SysClk* periods (t_{RESET4}) after the completion of a write access to register.

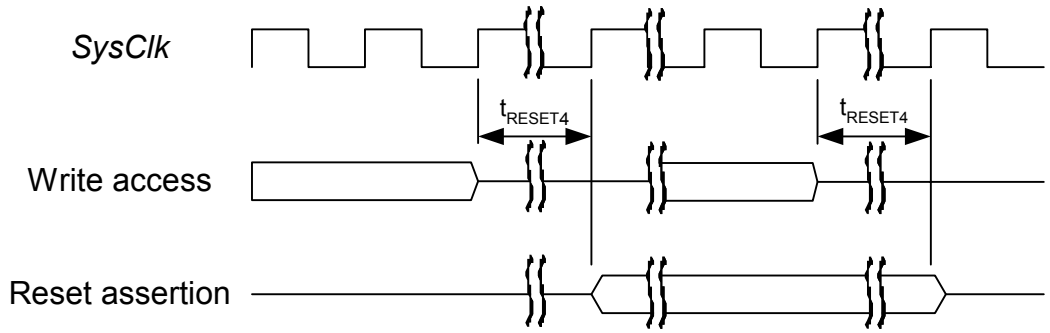


Figure 7-1 Software triggered reset

Note, a disabled module will have its outputs driven to their inactive state and all tristate set in high impedance mode, i.e. tri-stated.

7.1.3.2 Power-On Reset

Except for the normal Power-On behaviour as defined below the ASIC also includes a asynchronous inactivation of all outputs and tri-stating of all bi-directionals. E.g. when the *PoResetN* signal is asserted then the chip select signals of the memory interface will be deasserted and the data bus tri-stated asynchronously, i.e. with or without a toggling *SysClk*. The pins out from the module will be kept in this state as long as the module is not enabled.

A *PoResetN* assertion will have effect immediately after the *PoResetN* signal has been asserted. The Reset assertion ends 6 *SysClk* periods (t_{RESET3}) after the first positive edge of *SysClk* after the *PoResetN* is deasserted.

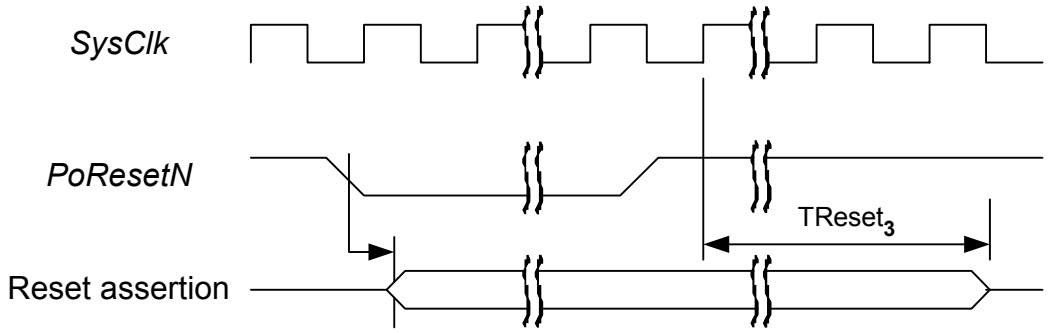


Figure 7-2 Hardware triggered reset

The outputs will be defined no later than 1 *SysClk* periods (t_{RESET6}) after the Reset assertion start. See figure below.

The chip will be reset and ready for operation no later than 2 ms (t_{RESET7}) after the Reset assertion end. This includes the configuration of the chip performed by the configuration block.

The outputs of the chip will be as defined below during and after the power-on reset even if the driving clock is unavailable:

Output	Value during and after reset
JtagTdo	0
MemD[15:0]	Tristate
MemDcc[5:0]	Tristate
MemWEN	1
CiOutValid	0
CselStatusOut	0
ExtCpduIfRdy	0
*PdecMapDsr**	0
Irq	0
*Tme**	0
CiInRdy	0

Table 7-2 Outputs defined during power-on reset independent of driving clock

Note: the state of the outputs can change quickly after *PoResetN* has been deasserted.

7.1.4 Configuration block

The configuration block has one output: *Irq*, which is the external interrupt signal from SCTMTC.

7.1.4.1 Configuration from PROM

The configuration block accesses the non-volatile memory bank selected by *PromCsN* during configuration and refresh of the chip. In order for this to work properly it is important that the PROM located at memory bank 0 is set up correctly (cf. section 6.11.1).

7.1.4.2 Interrupt Handling

External interrupts from the SCTMTC ASIC are signaled on *Irq*. *Irq* is asserted as long as there is an interrupt request from any module mapped to the external interrupt through Conf External Interrupt Enable Register. As soon as the last interupt bit in the externally mapped modules' pending interrupt register has been cleared *Irq* is deasserted.

1.1.1 Clock Timing

Designation	Description	Min	Max	Unit
t_{SysClk}	*SysClk* period	45		ns
$t_{SysClk2}$	*SysClk* pulse width high or low	16		ns
t_{SpwClk}	*SpwClk* period	13		ns
$t_{SpwClk2}$	*SpwClk* pulse width high or low	3		ns
t_{TmClk}	*TmClk** period	21.5		ns
t_{TmClk2}	*TmClk** pulse width high or low	8		ns
$t_{JTagTck}$	*JTagTck* period	66		ns
$t_{JTagTck2}$	*JTagTck* pulse width high or low	33		ns

Table 7-3 Clock Timing parameters for SCTMTC

Released

7.2 Memory Interface

7.2.1 Functional Description

7.2.1.1 Memory banks

The Memory Interface can be connected to up to three memory banks, identified by the three chip-select signals *PromCsN*, *ExtRecLacN*, and *RamCsN*. The memory banks can be configured individually through the PIM MUnit $ Register.

Memory bank 0, selected by *PromCsN*, is used for holding vital configuration parameters for the SCTMTC ASIC and must contain non-volatile memory. In order to allow configuration of the chip the data width of memory bank 0 must be known; this is configured by the *MemSize16* input signal. If a byte size memory is used for memory bank 0, *MemSize16* shall be tied low and if a halfword size memory is used *MemSize16* shall be tied high.

If an external recovery LAC counter is used together with the PDEC3 authentication unit, the counter is located in memory bank 2, selected by *ExtRecLacN*, which shall be of non-volatile type. This memory bank should only be used for storing the external recovery LAC counter.

Memory bank 3, selected by *RamCsN*, must contain SRAM, since it hosts all locked RAM buffers of the SCTMTC ASIC.

7.2.1.2 Address and data

MemD[15:0] is used for data. When connecting the Memory Interface to a halfword memory all 16 bits shall be connected. When connecting to a byte size memory, only bits 7-0 shall be connected

The memory address is output on *MemA[19:0]* during a memory access. *MemA[19:0]* always contain the byte-address to the data being accessed. This means that for halfword memories *MemA[0]* should not be connected.

7.2.1.3 EDAC check bits

When EDAC is enabled for a memory bank, *MemDcc[5:0]* is used for EDAC check bits.

7.2.1.4 Read/write accesses

The Memory Interface provides a write strobe, *MemWEN*, which is asserted during write accesses. Data is stable when *MemWEN* is deasserted at the end of the write cycle.

During read accesses the output enable signal, *MemOEN*, is asserted together with the appropriate chip-select signal.

7.2.2 Timing

The timing diagram for some different accesses can be seen in the figures below. Each of the burst accesses are separated so no bus contention between different burst cycles can occur.

Figure 7-3 shows a zero waitstate read cycle and a 1 waitstate burst with two accesses.

Designation	Description	Reference Edge	Min	Max	Unit
t_{MEMRC}	Memory Read cycle length		1+WSread[1]		t_{SysClk}
t_{MEMWC}	Memory Write cycle length		1+WSwrite[1]		t_{SysClk}
$t_{SysClk2Low}$	*SysClk* pulse width low				
t_{MEM1}	Read: *MemD* stable after *MemCSN* low.			t_{MEMRC} - 22.2	ns
t_{MEM2}	Read: *MemD* stable after *MemOEN* low.			t_{MEMRC} - 20.9	ns
t_{MEM3}	Read: *MemD* stable after *MemA* stable.			t_{MEMRC} - 24.5	ns
t_{MEM4}	Read: *MemD* hold after *MemA*/*MemCSN*/*MemOEN* change.		0		ns
t_{MEM5}	Write: *MemD* stable before	*MemWEN* ↑	t_{MEMWC} - $t_{SysClk2Low}$ - 12.5		ns
t_{MEM6}	Write: *MemD* stable after	*MemWEN* ↑	$t_{SysClk2Low}$ - 11.8		ns
t_{MEM7}	Write: *MemA* stable before	*MemWEN* ↑	t_{MEMWC} - $t_{SysClk2Low}$ - 5.5		ns
t_{MEM8}	Write: *MemA* stable after	*MemWEN* ↑	$t_{SysClk2Low}$ - 12.1		ns
t_{MEM9}	Write: *MemCSN* stable before	*MemWEN* ↑	t_{MEMWC} - $t_{SysClk2Low}$ - 3.2		ns
t_{MEM10}	Write: *MemCSN* stable after	*MemWEN* ↑	$t_{SysClk2Low}$ - 10.3		ns

Table 7-4 Memory timing parameters

[1] WSread and WSwrite are defined in PIM MUnit $ Register.

Note that if WSwrite is set to 0, no write strobe is generated during the write cycle.

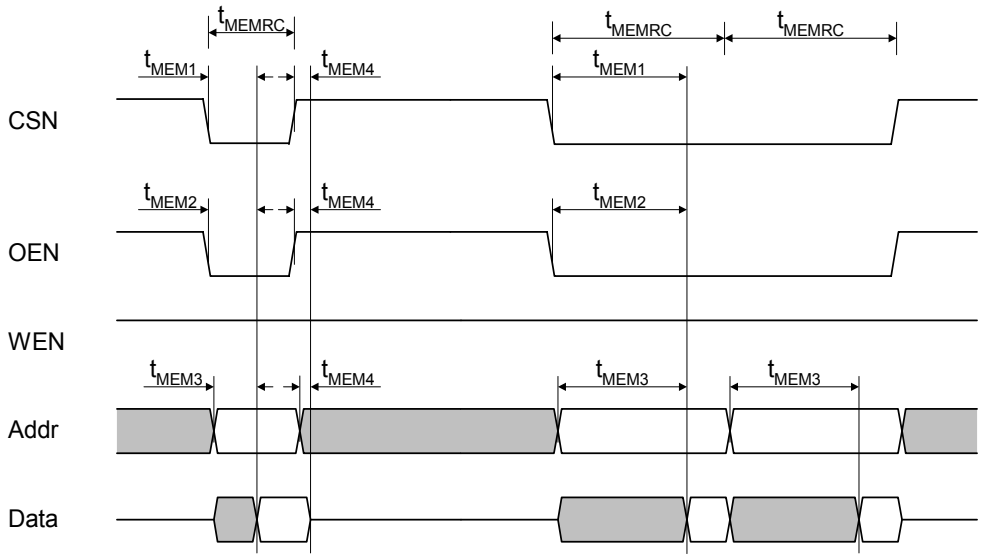


Figure 7-3 Memory read cycle

Figure 7-4 shows a two waitstate write cycle and a 1 waitstate burst with two accesses.

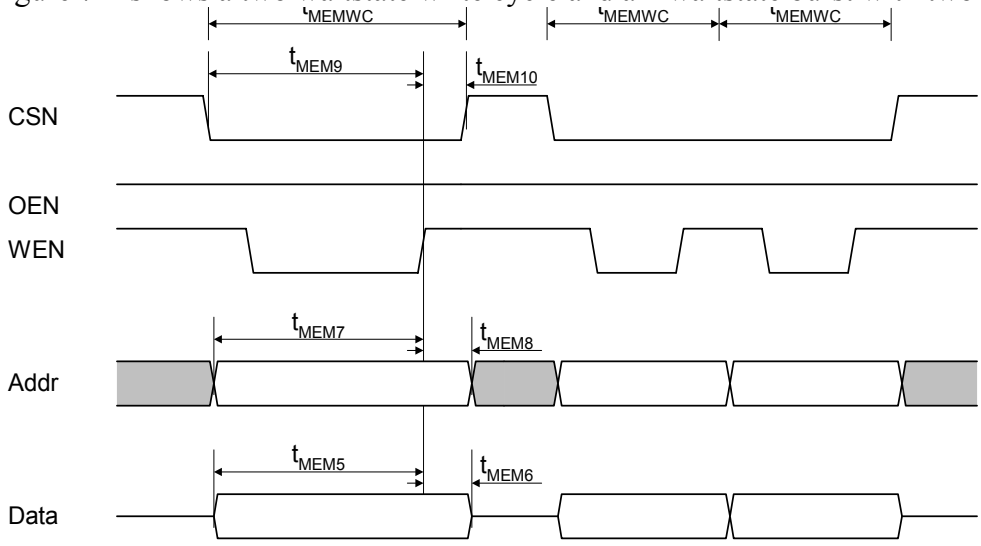


Figure 7-4 Memory write cycle

7.2.3 Application note

The following applies to usage of external recovery LAC only:

The external recovery LAC is only one byte wide, which means that a byte-size memory in memory bank 2 is sufficient. Since the Memory Interface ensures that the same data is written to all four bytes in the word containing the external recovery LAC counter, it is possible to connect the byte-size memory in memory bank 2 to either *MemD[15:8]* or *MemD[7:0]*.

When connecting the memory to *MemD[15:8]* ensure that *MemA[0]* is unconnected, since it has no use in a halfword memory. For a byte-size memory all address bits must be connected. If the memory can handle several consecutive write accesses to the same address *MemA* can be left unconnected and the address of the memory strapped low.

When the memory is connected to *MemD[15:8]* the size of memory bank 2 shall be configured to halfword and if it is connected to *MemD[7:0]* the size shall be configured to byte. If the connection of *MemD* is not compliant with the configuration the external recovery LAC will not work.

The Memory Interface writes the same data several times when updating the external recovery LAC. For a memory configured as byte size a total of four accesses are perfomed each time the external recovery LAC is updated, while only two accesses are needed to update a memory configured as halfword.

It is recommended to use a byte-size memory connected to *MemD[15:8]* for storing the external recovery LAC since this increases performance when accessing the external recovery LAC.

7.3 Packet Telecommand Decoder Module (PDEC3)

7.3.1 Functional Description

7.3.1.1 TC Channels

A TC channel is active when its *PdecTcAct$* signal is asserted. *PdecTcClk$* can start toggle when *PdecTcAct$* has been asserted for a time T3, see Table 7-5. The PDEC3 expects data on *PdecTcIn$* to be valid on falling edges of *PdecTcClk$*.

7.3.1.2 MAP Interfaces

The PDEC3 asserts *PdecMapDsr** when data is available for that MAP Interface. When *PdecMapDtr** is asserted, data is sent on *PdecMapData*. *PdecMapData* is valid on rising edges of *PdecMapClk*. Data is sent on an octet-by-octet basis. After each octet, *PdecMapDtr** is checked. A transfer can be aborted by the PDEC3 at any time by the assertion of *PdecMapAdt*.

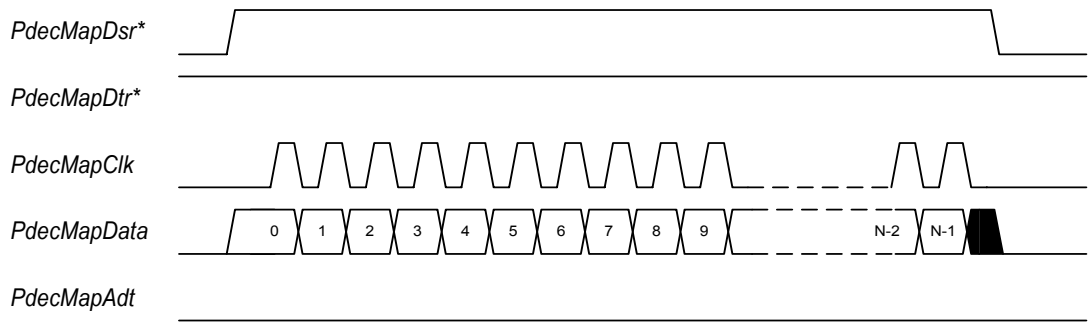


Figure 7-5 Transfer of an N-bit TC Segment without flow control

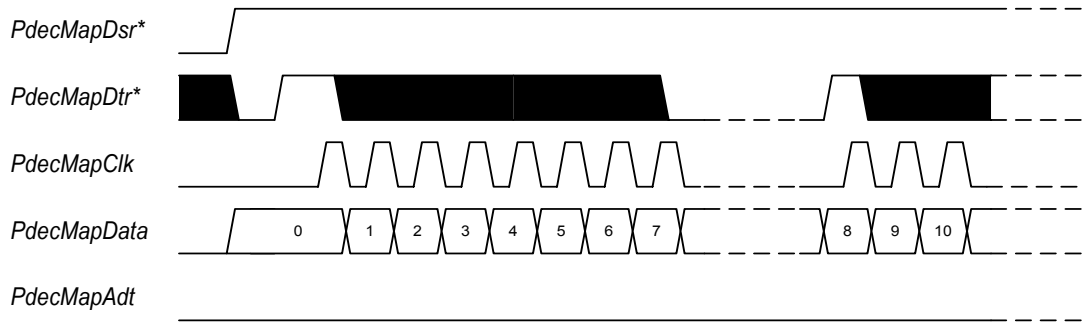


Figure 7-6 Transfer of a TC Segment with flow control

PdecMapSwitch is used to re-route TC Segments between MAP Interface 1 and MAP Interface 2. When *PdecMapSwitch* is deasserted, data is routed normally. When *PdecMapSwitch* is asserted, TC Segments destined for MAP Interface 1 (determined by MAP ID lookup) are sent on MAP Interface 2 and vice versa.

7.3.1.3 CLCW Interface

The synchronous bit serial interfaces are used for accessing CLCW status. The interfaces are activated by asserting *PdecClcwSamp$*. *PdecClcwClk$* is used to control the data stream. The PDEC3 changes *PdecClcwD$* on rising edges of *PdecClcwClk$* and ensures data to be stable on rising edges of *PdecClcwClk$*. Data is sent starting with MSB. The transfer is always restarted when *PdecClcwSamp$* is asserted, which allows a receiver to sample only parts of the data.

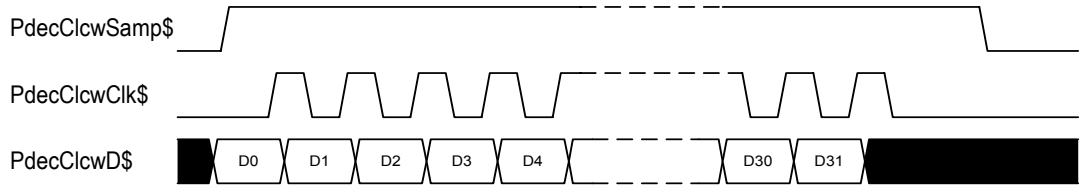


Figure 7-7 Serial interface for CLCW

7.3.1.4 AU Enabling and Disabling

The AU is enabled if *PdecAuEnable* is asserted when the AU starts to process a Telecommand Segment.

7.3.1.5 Start Sequence Searching Mode

PdecTcPrior is used to control the mode for start sequence search in the coding layer.

If both *PdecTcPrior* is deasserted, the PDEC3 is in standard selection mode, i.e. it searches all the incoming TC channels for a start sequence.

When *PdecTcPrior* is asserted, the PDEC3 is in priority search mode, in which TC channel 0 has the highest priority followed by TC channel 1. All other channels have equal priority.

Note: Priority search mode is not specified in [TC_SPEC].

7.3.1.6 RF Available

The *PdecRfAvN$* inputs are synchronised and used to generate the NoRF flag of the CLCW.

7.3.2 Timing

The output delay times, setup constraints and hold constraints for the PDEC3 timing are listed in Table 7-5, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

Designation	Description	Reference Edge	Min	Max	Unit
t_{PDEC1}	*PdecTcClk** period		8		t_{SysClk}
t_{PDEC2}	*PdecTcClk** pulse width high or low		1.5		t_{SysClk}
t_{PDEC3}	*PdecTcAct** high to first *PdecTcClk** falling	*PdecTcClk** ↓	1.5		t_{SysClk}
t_{PDEC4}	Last falling *PdecTcClk** to *PdecTcAct** low	*PdecTcClk** ↓	1.5		t_{SysClk}
t_{PDEC5}	*PdecTcIn** setup before *PdecTcClk** falling	*PdecTcClk** ↓	0		t_{SysClk}
t_{PDEC6}	*PdecTcIn** hold after *PdecTcClk** falling	*PdecTcClk** ↓	2.5		t_{SysClk}
t_{PDEC7}	*PdecClcwClk** period		3.5		t_{SysClk}
t_{PDEC8}	*PdecClcwClk** duty cycle		1.5		t_{SysClk}
t_{PDEC9}	*PdecClcwSamp** high to first rising *PdecClcwClk**	*PdecClcwClk** ↑	2.5		t_{SysClk}
t_{PDEC10}	*PdecClcwSamp** high to *PdecClcwD** valid	*PdecClcwSamp** ↑	2	4	t_{SysClk}
t_{PDEC11}	*PdecClcwSamp** width low		1.5		t_{SysClk}
t_{PDEC12}	*PdecClcwClk** rising to *PdecClcwD** valid	*PdecClcwClk** ↑	2	4	t_{SysClk}
t_{PDEC18}	*PdecMapClk* period		2	8192	t_{SysClk}
t_{PDEC19}	*PdecMapClk* duty cycle		Typ. 50		%
t_{PDEC20}	*PdecMapGenA* valid to *PdecMapDsr** high	*PdecMapDsr** ↑	6		t_{SysClk}
t_{PDEC21}	Last *PdecMapClk* falling to *PdecMapDsr** low	*PdecMapClk* ↓	0.5		t_{PDEC18}
t_{PDEC22}	*PdecMapDsr** width low		256 t_{SysClk} 1 t_{PDEC18} [1]		
t_{PDEC23}	*PdecMapDtr** high to *PdecMapClk* rising	*PdecMapClk* ↑	4		t_{SysClk}
t_{PDEC24}	*PdecMapDtr** setup to each 8th *PdecMapClk* falling	*PdecMapClk* ↓	3*t_{SysClk}+ 28ns		ns
t_{PDEC25}	*PdecMapDtr** hold after each 8th *PdecMapClk* falling	*PdecMapClk* ↓	0		ns
t_{PDEC26}	*PdecMapDsr** high to *PdecMapData* valid	*PdecMapDsr** ↑	-29.1	35.8	ns
t_{PDEC27}	*PdecMapClk* falling to *PdecMapData* valid	*PdecMapClk* ↓	-19.9	37.0	ns
t_{PDEC28}	Last *PdecMapClk* falling to *PdecMapAdt* high	*PdecMapClk* ↓	Typ. 1		t_{PDEC18}
t_{PDEC29}	*PdecMapAdt* high to *PdecMapDsr** low	*PdecMapDsr** ↓	Typ. 1		t_{PDEC18}
t_{PDEC30}	*PdecMapAdt* width high		Typ. 2		t_{PDEC18}
t_{PDEC31}	*PdecMapDsr** high to *PdecMapClk* rising	*PdecMapClk* ↑	1		t_{PDEC18}

1) The minimum time is the maximum of the two values where the *PdecMapClk* period is the period of the new telecommand segment.

Table 7-5 PDEC3 timing

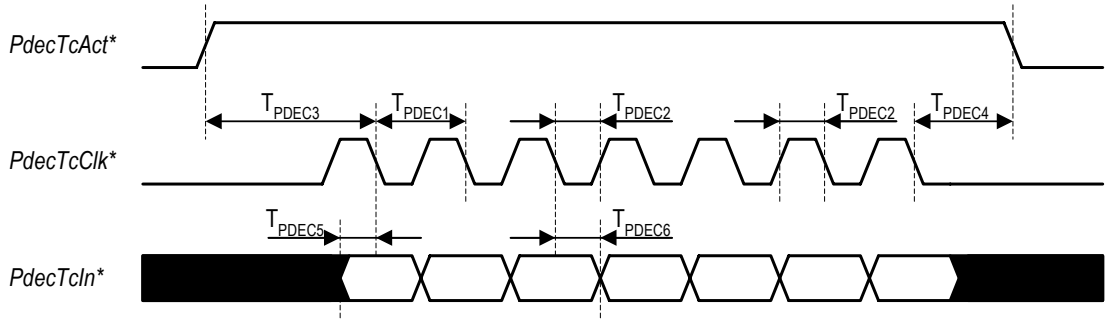


Figure 7-8 Timing parameters, Transponder Interface

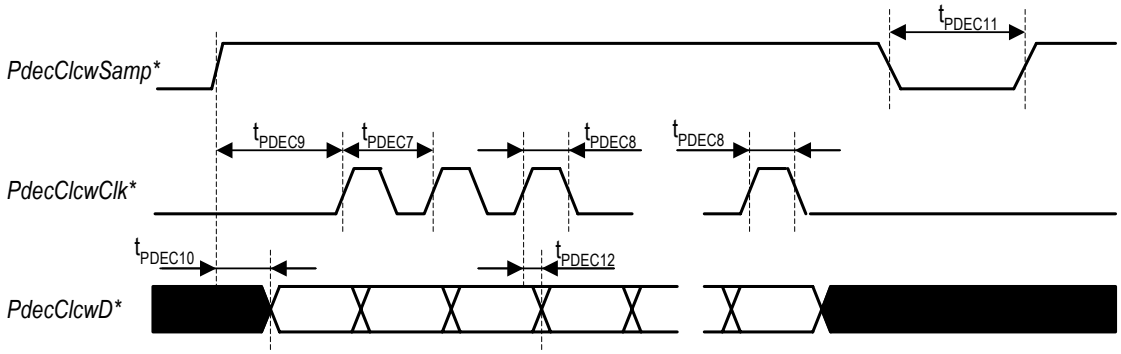


Figure 7-9 Timing parameters, CLCW Interface

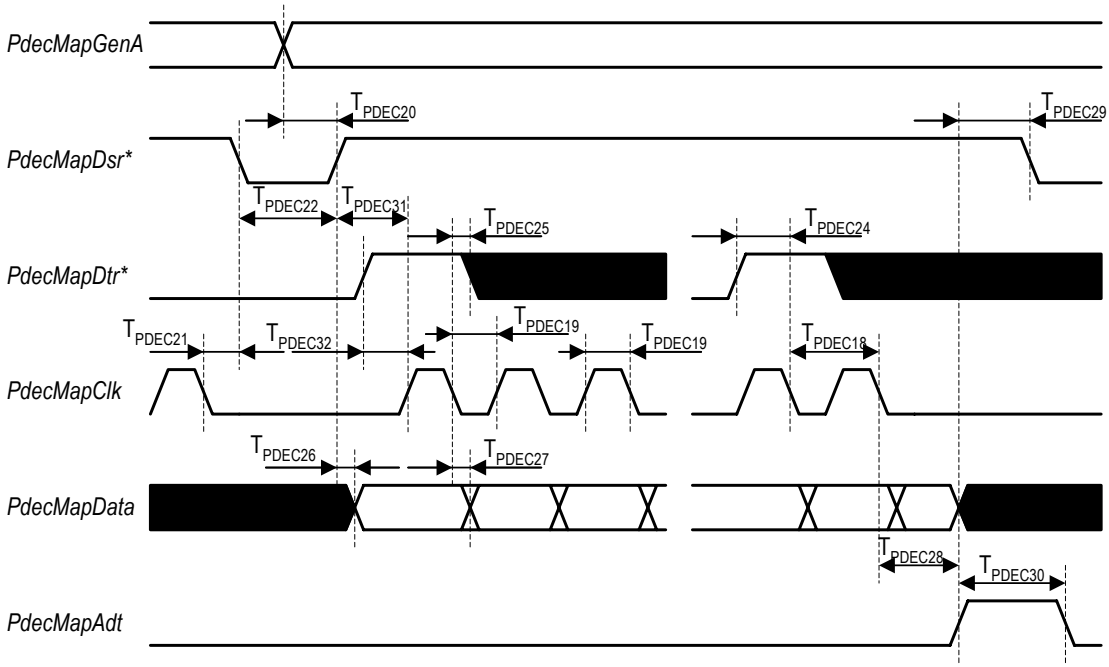


Figure 7-10 Timing parameters, MAP Interface

7.3.3 Application Note

Not applicable

7.3.4 Reset

After a reset the values of the PDEC3 Module output signals are as follows:

Signal	Value after reset
PdecMapAdt	0
PdecMapClk	0
PdecMapData	0
PdecMapDsr[5:1]	0
PdecMapDsrG	0
PdecMapGenA[5:0]	0
PdecClcwD[1:0]	0

Table 7-6 Reset Values for PDCE3

7.4 External CPDU Interface Module (ExtCpduIf)

This section defines the ASIC external hardware interface of the ExtCpduIf.

7.4.1 Functional Description

The ExtCpduIf consist of the following signals.

- ***ExtCpduIfAbort*** – indicates that the sender aborted the packet transmission. *ExtCpduIfAbort* is sampled on the falling edge of *ExtCpduIfValid*.
- ***ExtCpduIfClk*** – is used as the data clock for the transmission.
- ***ExtCpduIfData –*** serial data received on the Packet Wire channel. *ExtCpduIfData* is sampled by the ExtCpduIf on the rising edge of *ExtCpduIfClk*. Typically the transmitter changes *ExtCpduIfData* on the falling edge of *ExtCpduIfClk.*
- ***ExtCpduIfRdy*** – is used for flow control. *ExtCpduIfRdy* is asserted by the ExtCpduIf as long as it is able to receive more data, i.e. as long as the reception buffer is not full or if the packet is finished and the *ExtCpduIf* module has been freed by the CSEL. If the reception buffer of some reason becomes full *ExtCpduIfReady* is deasserted until the buffer is not full. To avoid loss of data the ExtCpduIf can still receive one more byte when *ExtCpduIfReady* is deasserted. If the packet is finished but the module has not been freed by the CSEL the *ExtCpduIfRdy* is deasserted to make sure the data is not overwritten before it is handled.
- ***ExtCpduIfValid*** – used to detect packet boundaries. *ExtCpduIfValid* is asserted by the sender at start of packet and is deasserted by the sender at end of packet.

7.4.2 Timing

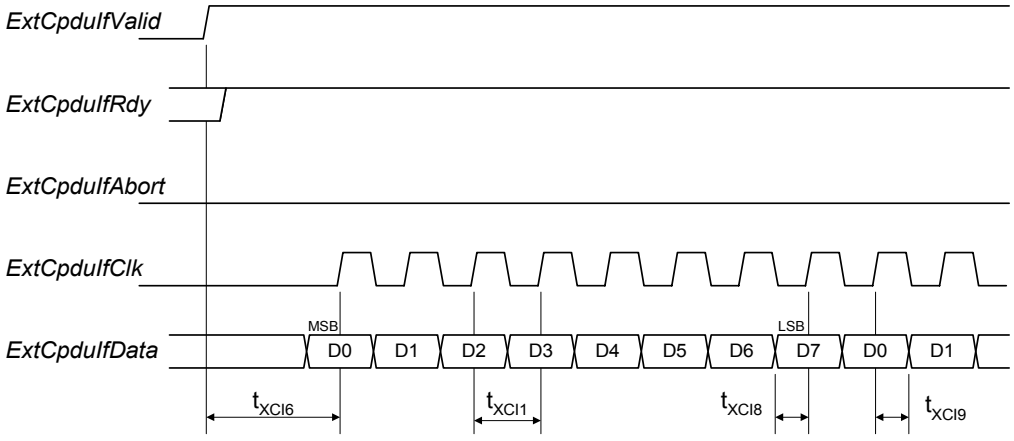


Figure 7-11 Start of packet

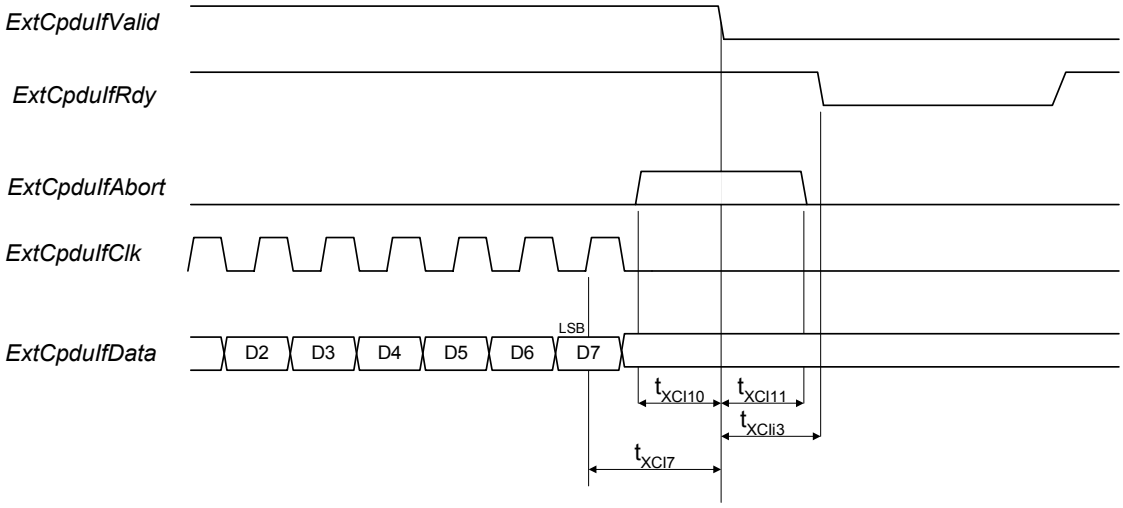


Figure 7-12 End of packet

The figures show the normal relationship between signals.

The output delay times, setup constraints and hold constraints for the ExtCpduIf timing are listed in Table 7-7 ExtCpduIf timing, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

Released

Designation	Descriptions	Reference edge	Min	Max	Unit
t_{XCI1}	*ExtCpduIfClk* period		1/2		t_{SysClk}
t_{XCI2}	*ExtCpduIfClk* pulse width high or low		3		ns
t_{XCI3}	*ExtCpduIfValid* low to *ExtCpduIfRdy* low	*ExtCpduIfValid* ↓	2	3	t_{SysClk}
t_{XCI6}	*ExtCpduIfValid* input setup time	*ExtCpduIfClk* ↑	3.5		ns
t_{XCI7}	*ExtCpduIfValid* input hold time	*ExtCpduIfClk* ↑	0.5		ns
t_{XCI8}	*ExtCpduIfData* input setup time	*ExtCpduIfClk* ↑	1.4		ns
t_{XCI9}	*ExtCpduIfData* input hold time	*ExtCpduIfClk* ↑	0.8		ns
t_{XCI10}	*ExtCpduIfAbort* input setup time	*ExtCpduIfValid* ↓	0.0		ns
t_{XCI11}	*ExtCpduIfAbort* input hold time	*ExtCpduIfValid* ↓	2.0		ns

Table 7-7 ExtCpduIf timing

7.4.3 Reset

After a reset the values of the *ExtCpduIf* module output signals are as follows:

Signal	Value after reset
ExtCpduIfRdy	0

Table 7-8 Reset Values for External CPDU Interface Module outputs

7.5 CPDM Selector Module (CSEL)

7.5.1 Functional Description

7.5.1.1 Operating Mode Control

The Operating Modes of the CSEL is described in section 6.5.2.1.

If the internal signal *CselTcOnly* (done by the telecommand decoder when receiving on MAP ID 6) is asserted for at least one clock cycle while the CSEL is operating in RM ON or RM OFF mode, the CSEL transits to TC Only mode and starts the TC Only timeout timer. When the timer expires the CSEL checks *CselRmOn* to determine its previous mode. If *CselTcOnly* is asserted while the CSEL is operating in TC Only mode, the TC Only timeout timer is restarted.

The CSEL transits to RM ON mode whenever *CselRmOn* is asserted and to RM OFF mode when it is deasserted. If the CSEL has become fail silent a change in *CselRmOn* enables the CSEL again as the operating mode changes to either RM ON or RM OFF.

CselTcOnly takes precedence over *CselRmOn* if a change is detected in both signals during the same clock cycle.

7.5.1.2 CSEL Status Interface

The CSEL outputs its current status on the CSEL Status interface, *CselStatusOut[2:0]*. The status is coded with a hamming distance of 2. Bit 0 is the BUSY RM flag, bit 1 is the BUSY GND/PM flag, and bit 2 is a parity bit. The parity over the entire code is even, i.e. the number of ones equals an even number. The legal codes are listed in Table 7-9; all other values are illegal.

***CselStatusOut[2:0]* value**	**Interpretation**
000_2	CSEL is idle or fail silent
110_2	BUSY GND/PM CSEL is processing a Ground or PM request
101_2	BUSY RM CSEL is processing an RM request
011_2	BUSY RM and BUSY GND/PM CSEL is processing both an RM request and a Ground or PM request

Table 7-9 CSEL Status interface values

The CSEL receives status from a remote CSEL on *CselStatusIn[2:0]*. The status is filtered since there is no guarantee that the three bits arrive simultaneously. A legal code must be stable for 5 consecutive clock cycles and an illegal code must be stable for 16 clock cycles before decode. *CselStatusIn* shall have pull-down to ensure that the idle code is received if the remote CSEL is powered down.

7.5.2 Timing

The output delay times, setup constraints and hold constraints for the CSEL timing are listed in Table 7-10, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

Designation	Parameter	Reference Edge	Min	Max	Unit
t_{CSEL1}	*CselStatusOut* output delay	*SysClk* ↑	11.8	45.3	ns

Table 7-10 CSEL timing

CselRmOn and *CselStatusIn* have no setup/hold requirement, i.e. they are treated as asynchronous inputs.

For a CSEL pair to work properly, the delay on *CselStatus** between the sender and receiver must not exceed 6 *SysClk* cycles.

7.5.3 Application Note

The CSEL inputs/outputs are standard CMOS with pull-downs on the inputs of the CSEL Status interface to generate all zeros if the remote CSEL is powered down; this is interpreted as the remote CSEL being silent.

7.5.3.1 Connecting the CSEL status interface

The purpose of the CSEL status interface is to synchronise two redundant SCTMTCs in order to prevent CPDU commands from being output simultaneously from both SCTMTC ASICs.

The CSEL status interface is only needed if there are several users requesting the CPDU function simultaneously and the use of the CPDU may result in simultaneous conflicting commands.

The following main use cases are assumed:

1. Ground telecommands only:

   In this case the SCTMTC CSEL only processes ground commands, like the earlier generations of TC Decoder devices. For this case the CSEL Status Interface should not be used and it is up to the ground operator to ensure that the two CPDUs do not execute conflicting commands (which can be the case if one CPDU is requested to execute a long sequence of commands, directly followed by a new telecommand addressed to the redundant CPDU).

2. Ground telecommands and commands via the Control Interface:

   In this use case the CSEL interface need to be connected if there is no other way of ensuring that simultaneous commands may not be issued. Such a mechanism can be to ensure that the ground TCs and the Control Interface never uses conflicting commands in the CPDU, e.g. the ground uses TCs from 0 to 31 and the Control Interface from 32 and up, and commands from the two groups do not affect the same function.

   The CSEL interface is not needed if conflicting commands is not a problem

3. Ground telecommands, commands via the Control Interface and commands from an autonomous unit (e.g. an external Reconfiguration Module) via the External CPDU Interface:

   In this case the CSEL status interface is most likely needed to allow synchronisation between the two CPDUs ensuring that only one of them is executing at a time.

The CSEL status interfaces may be interconnected directly. However, in this case the parity generation mechanism in side the transmitting interface is considered as part of the receiver fault containment region. If this is not acceptable, since an error in the parity generation function of the transmitter will cause the receiver to enter TC Only mode, the connection presented in Figure 7-13 can be implemented by external circuitry to obtain an external parity generator. Note that the CSEL link no longer can detect stuck-at errors on the chip outputs and that additional circuitry is needed to detect such an error. The action to do in case of a parity error on the CSEL outputs is mission dependant but one action could be to simply send all zeroes (which are interpreted as idle state) to the redundant SCTMTC chip, as shown in Figure 7-14. If this solution is implemented the external circuitry should be placed on the receiving board to allow detection of stuck-at faults between the boards.

In case *CselStatus[1:0]* are both stuck at low level the CSEL status will be interpreted as idle by the receiver. In case any or both of the two lines are stuck at high the receiver has a timeout that, after a programmable time, ignores the input link until it returns to idle state.

The parity generation logic can be placed either on the sending board or on the receiving board. By placing the parity generation logic on the sending board it is possible to detect stuck-at-0 faults in the board connections; on the other hand it is possible for an error on one board to propagate to the other SCTMTC, making it enter TC Only mode.

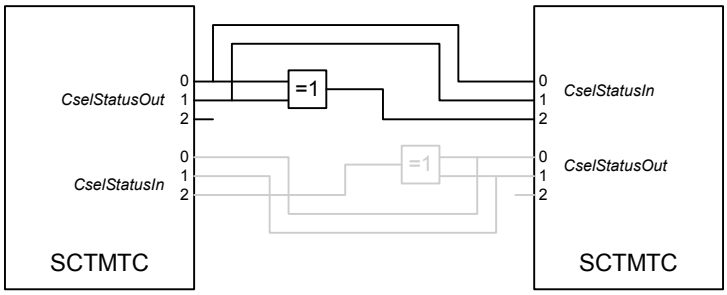


Figure 7-13 External parity generation for the CSEL status interface

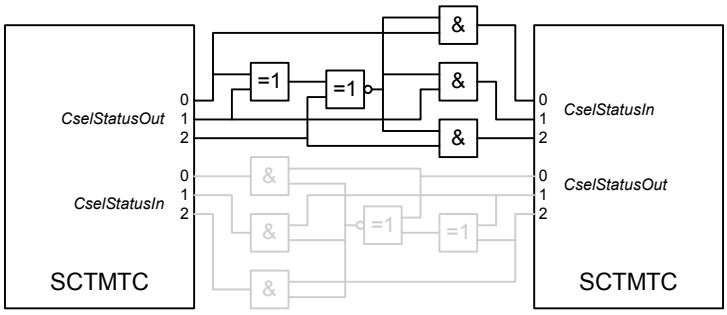


Figure 7-14 Disabling of CSEL status interface upon parity error detection

If an external autonomous unit is connected to the External CPDU Interfaces of two redundant SCTMTC it is possible to implement a dynamic CSEL status interface connection to connect the two CSELs when the autonomous unit is enabled as shown in Figure 7-15.

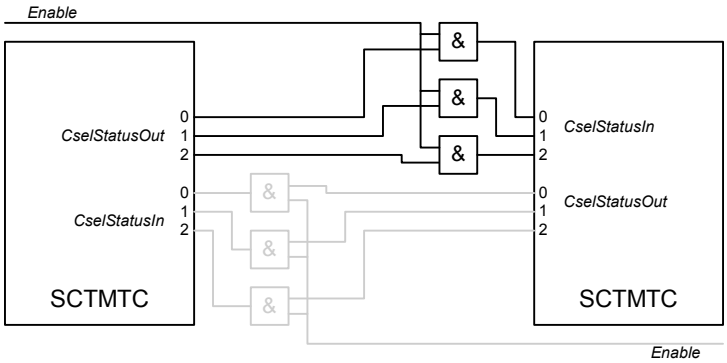


Figure 7-15 Dynamic CSEL status interface connection

When disabling the autonomous unit the complete CSEL interface will signal idle (or fail silent), thus making the redundant SCTMTC consider itself to be the only operational device. The schemes in the Figure 7-13 and Figure 7-14 can also be combined to allow disabling of the interface while keeping the external parity generation. Ensure that the circuit generating the Enable signal is designed to take all possible failure modes into account (cf. Figure 7-16).

If the two mechanisms are combined the parity generation logic should be placed after the disabling logic, as shown in Figure 7-16, to reduce the risk of an incorrect parity being generated.

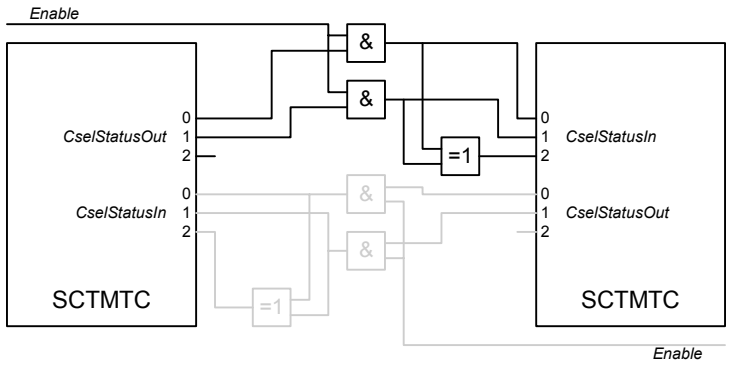


Figure 7-16 External parity generation and dynamic interface connection combined

7.5.4 Reset

After a reset the values of the CSEL Module output signals are as follows:

Signal	Value after reset
CselStatusOut[2:0]	000_2

Table 7-11 Reset Values for CSEL

7.6 Command Pulse Distribution Module (CPDM)

7.6.1 Functional Description

7.6.1.1 Command Pulse Output

The pulse output starts with the serial output of the Output Number and Pulse Length. The serial output is controlled by *CpdmClk* and data is sent on *CpdmSer*. Data is valid on rising edges of *CpdmClk* and the inverse of the previous bit is valid on falling edges of *CpdmClk*. When the serial output has finished, the CPDM outputs the pulse by asserting *CpdmArmN* followed by *CpdmStrb*. *CpdmStrb* is kept asserted for $D \cdot 2^L$, where L is the unsigned value of the Pulse Length. *CpdmArmN* is kept asserted some time after *CpdmStrb* has been deasserted.

7.6.1.2 Clock Alive Detection

The CPDM outputs the system clock on *CpdmClkToggle* for an external clock detection mechanism. When the clock is active, *CpdmClkAlive* should be asserted. The CPDM deasserts combinatorially all its pulse outputs when *CpdmClkAlive* is deasserted.

7.6.2 Timing

The output delay times, setup constraints and hold constraints for the CPDM timing are shown hereafter, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

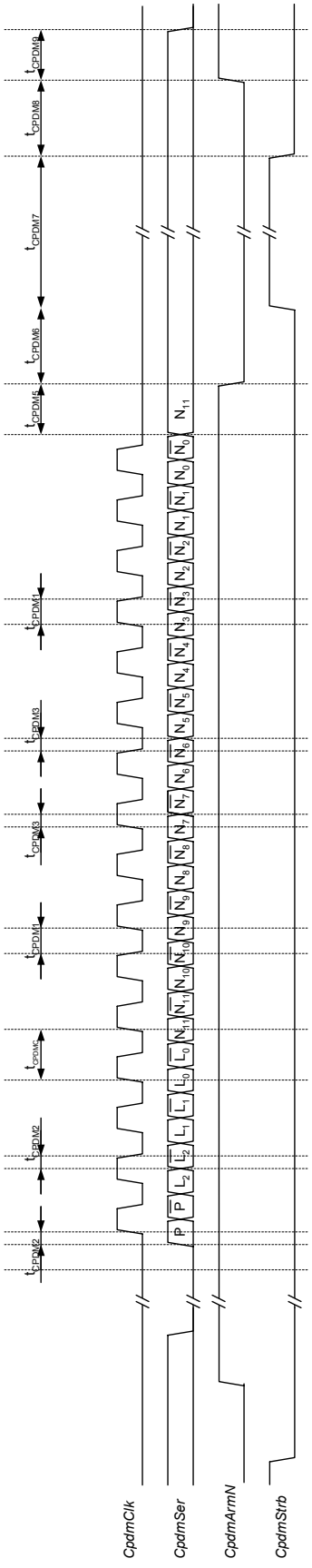
Released

Designation	Parameter	Reference edge	Min	Max	Unit
t_{CPDMC}	*CpdmClk* period		4	131072	ns
t_{CPDM1}	*CpdmClk* duty cycle		Typ. 50		%
t_{CPDM2}	*CpdmSer* setup before *CpdmClk* rising, falling	*CpdmClk* ↕	t_{CPDMC}/ 4-27.7		ns
t_{CPDM3}	*CpdmSer* hold after *CpdmClk* rising, falling	*CpdmClk* ↕	t_{CPDMC}/ 4-28.6		ns
t_{CPDM5}	Last *CpdmSer* output to *CpdmArmN* falling	*CpdmArmN* ↓	t_{CPDMC}- 28.3		ns
t_{CPDM6}	*CpdmArmN* falling to *CpdmStrb* rising	*CpdmArmN* ↓	D/8		ns
t_{CPDM7}	*CpdmStrb* width high		$D \cdot 2^L$		ns
t_{CPDM8}	*CpdmStrb* low to *CpdmArmN* rising	*CpdmArmN* ↑	D/8		ns
t_{CPDM9}	*CpdmSer* hold after *CpdmArmN* rising	*CpdmArmN* ↑	t_{CPDMC}		ns

Table 7-12 CPDM timing

D is the *duration*, i.e. the shortest possible pulse length.
L is the unsigned value of the three LSBs of the Pulse Length of the Command Instruction in the CPDU packet.
The serial output clock period is calculated as follows:
$2^{(X+2)} \cdot t_{SysClk}$
where X is defined by CPDM Clock Ratio Register.Ratio

P	Parity bit
$\overline{P}$	Parity bit inverted
L_2-L_0	Pulse Length bits
$\overline{L_2}$-$\overline{L_0}$	Pulse Length bits inverted
N_{11}-N_0	Output Number bits
$\overline{N_{11}}$-$\overline{N_0}$	Output Number bits inverted



7.6.3 Reset

After a reset the values of the CPDM Module output signals are as follows:

Signal	Value after reset
CpdmArmN	1
CpdmStrb	0
CpdmClk	0
CpdmSer	0

Table 7-13 Reset Values for CPDM

7.7 Packet Telemetry Encoder Module (TME)

There are two clocks in the TME; one for the internal bus, register and the VC Input Interfaces and one for controlling the transfer frame bit stream. The clocks are *SysClk* and *BitClk* respectively.

7.7.1 Functions

7.7.1.1 Serial Input Interface

Serial data is clocked in on the falling *TmeSClk** (* = A-H) edge, when *TmeSValid** = 1. The serial data is assembled in a shift register and when eight bits have been input, the resulting byte is internally forwarded and stored in the memory buffer as multiples of eight bits.

The input signal *TmeSValid** is used to delimit Telemetry Packets (data blocks in asynchronous mode). It is asserted while a Telemetry Packet (data block) is being input. In addition, *TmeSValid** defines the byte boundaries in the input data stream, the first byte explicitly and the following bytes each subsequent eight *TmeSClk** cycles. The first bit received after *TmeSValid** has been asserted is bit D0. This bit is the MSB. *TmeSValid** is asserted when data is input.

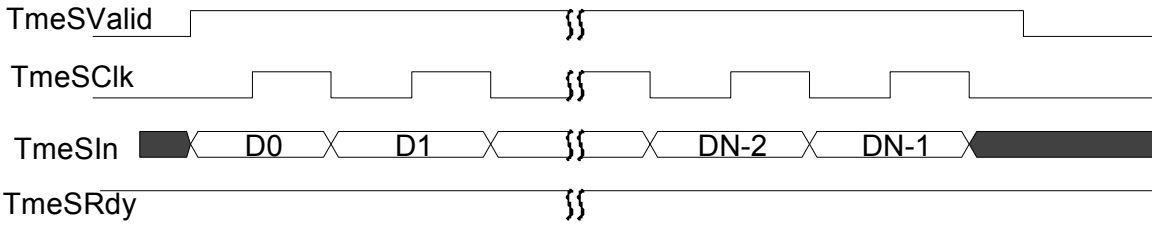


Figure 7-17 Serial Input Interface

7.7.1.2 CLCW Interface

The CLCW interface generates the *TmeClcwClk* and *TmeClcwSamp* output signals for controlling the CLCW interface, and inputs the *TmeClcwD0, TmeClcwD1, TmeClcwD2* or the *TmeClcwD3* signal from the packet telecommand decoder. The retrieval starts at the same time the frame starts to be output. All signals related to the CLCW interface are proportional to the *BitClk* frequency. The transfer rate equals *BitClk* divided by 50, e.g. if *BitClk* is 1 MHz the *TmeClcwClk* frequency will be 1 MHz / 40 = 25 kHz.

The interface has been designed to retrieve one CLCW per Transfer Frame generated, regardless of the frame length. The interface allows direct connection to the corresponding PDEC3 interface without additional components.

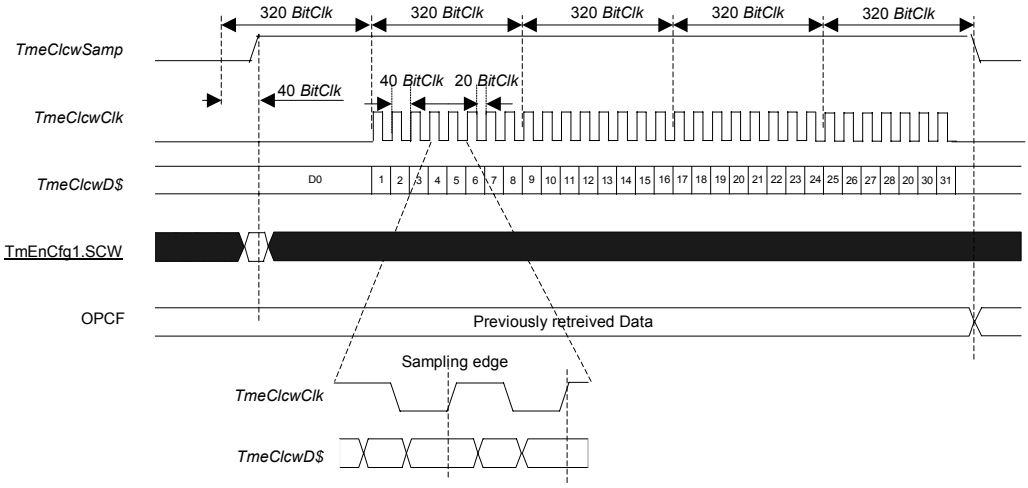


Figure 7-18 Synchronous bit serial interface for the CLCW

7.7.1.3 TimeStrobe

See section 5.4.7.12 for definitions of clocks used here.

The *TmeTimeStrb* output is asserted at the same time as the first synch marker bit is output in *TmeUnEncOut*.

The delay to *TmeEncOut* is three bits, which means three *BitClk* normally, or six *BitClk* when using SP-L coding.

When using Square PSK, an additional delay of one *SubCClk* is added to *TmeEncOut*.

TmeEncI/QOut is delayed two *BitClk* from *TmeEncOut*.

TmePskSineOut is delayed one *SubCSineClk* from *TmeEncOut*.

7.7.2 Timing

The output delay times, setup constraints and hold constraints for the TME timing are listed hereafter, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

Released

Designation	Parameter	Reference edge	Min	Max	Unit
t_{TME1}	*TmeSRdy** output delay	*SysClk* ↑	10.8	45.6	ns
t_{TME2}	*Tme*EncClk* output delay direct	*TMClk* ↑	6.5	23	ns
t_{TME3}	*Tme*EncClk* output delay clocked	*TMClk* ↑	9.7	30.5	ns
t_{TME4}	*Tme*EncOut* output delay	*TMClk* ↑	10.2	31.1	ns
t_{TME5}	*TmeUnEncSync*, *TmeEncIQClk*, *TmeEnc(I/Q)Out* output delay	*TMClk* ↑	9.9	36.9	ns
t_{TME6}	*TmeEncPskSine** output delay	*TMClk* ↑	6.9	30.6	ns
t_{TME7}	*TmeClcw**, *TmeTimeStrb* output delay	*TMClk* ↑	9.1	42.2	ns
t_{TME8}	*TmeClcwD$* input setup time	*TmeClcwClk* ↑	0		ns
t_{TME9}	*TmeClcwD$* input hold time	*TmeClcwClk* ↑	6.7		ns
t_{TME10}	*TmeSIn$* input setup time	*TmeSClk*$* ↑	6.9		ns
t_{TME11}	*TmeSIn$* input hold time	*TmeSClk*$* ↑	0.5		ns
t_{TME12}	*TmeSClk*$* period		$t_{SysClk}/2$		
t_{TME13}	*TmeSClk*$* pulse width high or low		3		ns
t_{TME14}	*TmeSValid$* setup	*TmeSClk*$* ↑	7		ns
t_{TME15}	*TmeSValid$* hold	*TmeSClk*$* ↑	0.2		ns
t_{TME16}	*Extra output delay when TmClk is driven from SysClk*		1.9	3.4	ns

Table 7-14 TME timing

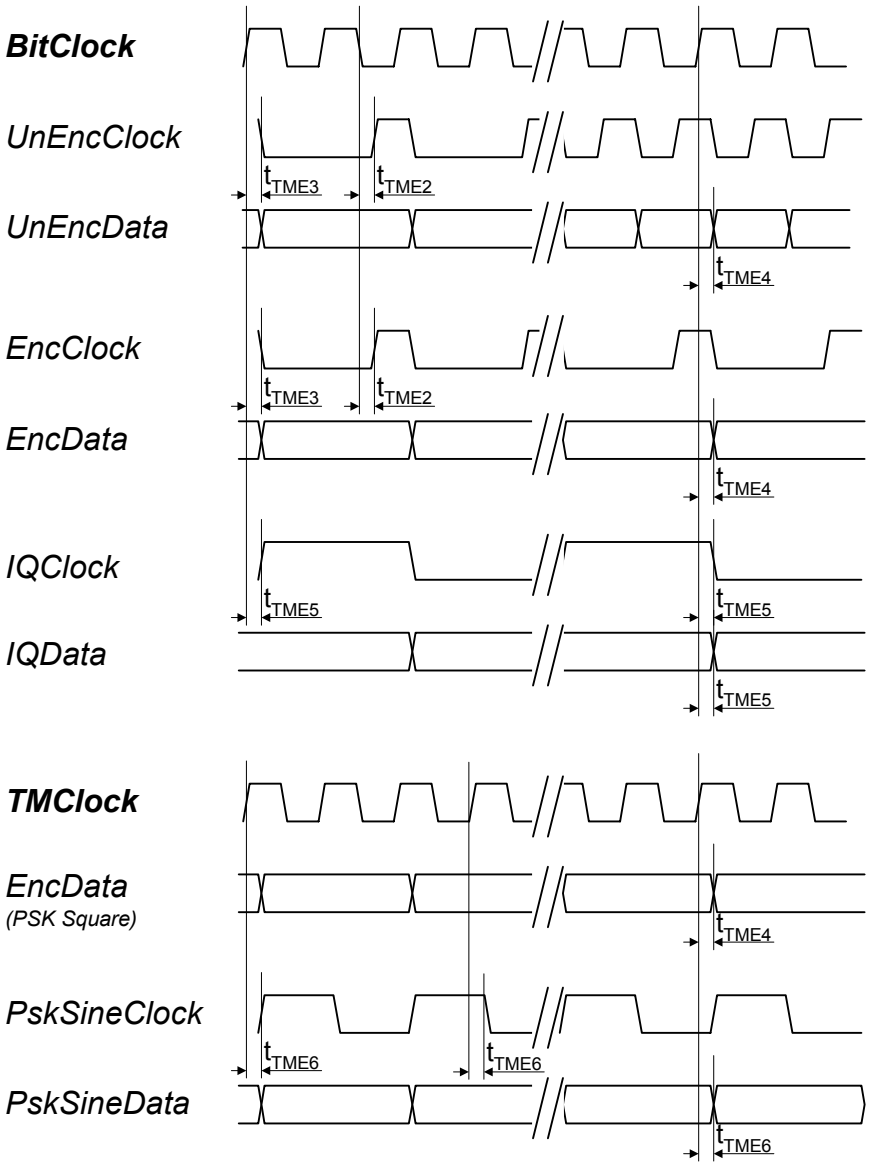


Figure 7-19 TME timing designation

7.7.3 Reset

There are two different types of resets in the TME: master reset and initialisation. A master reset affects the whole module, i.e. all configuration; status, states and memory elements are reset. Initialisation affects all internal logic but leaves the configuration registers intact. After a reset the values of the TME output signals will be as follows:

Signal	Value after reset	Signal	Value after reset
TmeSRdy[*]	0	*TmeClcwSamp*	0
TmeClcwClk	0	*TmeUnEncClk*	0
TmeTimeStrb	0	*TmeUnEncSync*	0
TmeUnEncOut	0	*TmeEncOut*	0
TmeEncClk	0	*TmeEncIOut*	0
TmeEncQOut	0	*TmeEncIQClk*	0

7.8 SpaceWire Module (SPW)

7.8.1 Functional Description

The *SpwClk* clock is used to clock the SpaceWire parts of the SPW.

The SPW has two sets of incoming SpaceWire link signals, named *Spw*InA* and *Spw*InB*. *SpwDIn** is used to transfer data. Data is sampled when either of *SpwDIn** or *SpwSIn** changes. These signals should not change at the same time. The selection between the sets is made using the *SpwIfSel* input, with *Spw*InA* being selected when *SpwIfSel* is deasserted, and *Spw*InB* being selected when *SpwIfSel* is asserted.

SpwDOut and *SpwSOut* transmit data from SPW. Only one of these changes at a time.

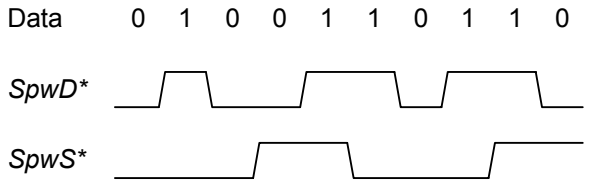


Figure 7-20 SpaceWire encoding

7.8.2 Timing

The output delay times, setup constraints and hold constraints for the SPW timing are listed in Table 7-15, for military conditions (T_A -55 ºC to +125 ºC, V_{DD} ±10 %). The load for all output signals is 50 pF.

Designation	Description	Reference edge	Min	Max	Unit
t_{SPW1}	*SpwDOut* output delay	*SpwClk* ↑		28.9	ns
t_{SPW2}	*SpwSOut* output delay	*SpwClk* ↑		29.1	ns
t_{SPW3}	*SpwEnA* output delay	*SysClk* ↑		40.4	ns
t_{SPW4}	*SpwEnB* output delay	*SysClk* ↑		36.2	ns
t_{SPW5}	Skew between *SpwDOut* and *SpwSOut*[1]			1	ns
t_{SPW6}	*SpwDIn* / *SpwSIn* edge to next *SpwDIn* / *SpwSIn* edge		2.5		ns

Table 7-15 SPW timing

[1] Output skew excluding pad delay. Typically the difference between falling and rising edge delay in the pad is the dominating factor in the total output skew.

7.8.2.1 Input Skew

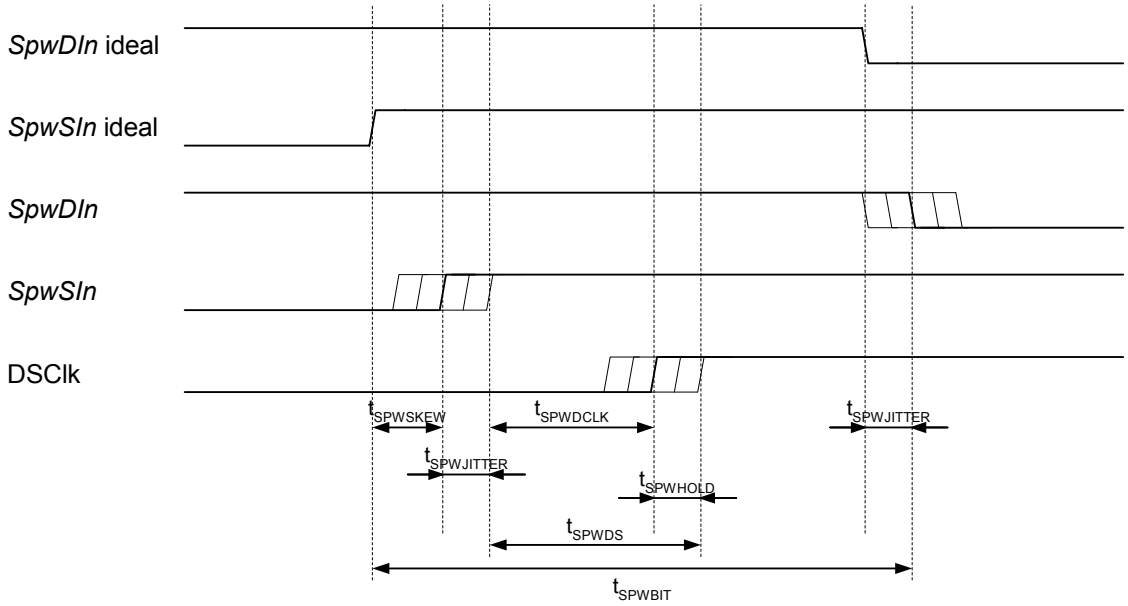


Figure 7-21 Skew and jitter

$t_{SPWSKEW}$ Skew between data and strobe signals.

$t_{SPWJITTER}$ The jitter on the data or strobe signal. $t_{SPWJITTER}$ data = $t_{SPWJITTER}$ strobe since they follow identical signal paths.

$t_{SPWDCLK}$ The delay in the receiver from the edge of the data or strobe signal, through the XOR operation which produces the clock signal, to the clocking in of the data in the input flip-flop. This may be regarded as the set-up time for the data input flip-flop from the edge of the data or strobe signal.

$t_{SPWHOLD}$ The hold time required for the data signal after the clocking of the data into the input flip-flop.

t_{SPWBIT} The unit interval or bit period. $t_{SPWBIT} = 1/f_{SPWBIT}$ where f_{SPWBIT} is the maximum bit rate on the SpaceWire link

For the SPW to function correctly the following must hold:
$t_{SPWBIT} > t_{SPWSKEW} + 2 \cdot t_{SPWJITTER} + t_{SPWDCLK} + t_{SPWHOLD}$

Released

7.8.3 Reset

After a reset the values of the SPW module output signals are as follows:

Signal	Value after reset
SpwDOut	0
SpwSOut	0
SpwEnA	NOT *SpwSel*
SpwEnB	*SpwSel*

Table 7-16 Reset Values for SPW

7.9 Control Interface Module (CI)

7.9.1 Functional Description

The Control Interface has two external Packet Wire interface, one for receiving commands and one for sending read data. The Packet Wire interfaces function as follows:
*Ci*Valid* indicates the start of a packet, this signal is asserted before the *Ci*Clk* starts to toggle. Data is sent on *Ci*Data*, which is valid on rising edges on *Ci*Clk*. The end of packet is signaled by the deassertion of *Ci*Valid*.

The *Ci*Rdy* signal is used for flow control; when *Ci*Rdy* is deasserted, the Packet Wire interface is temporarily stalled and *Ci*Clk* stops toggle.

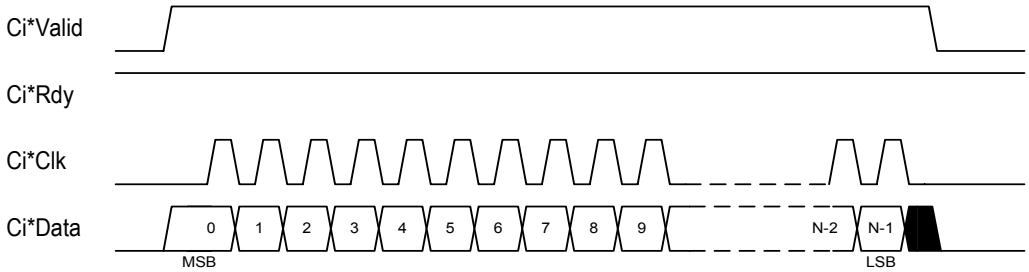


Figure 7-22 Transfer on Packet Wire interface without flow control

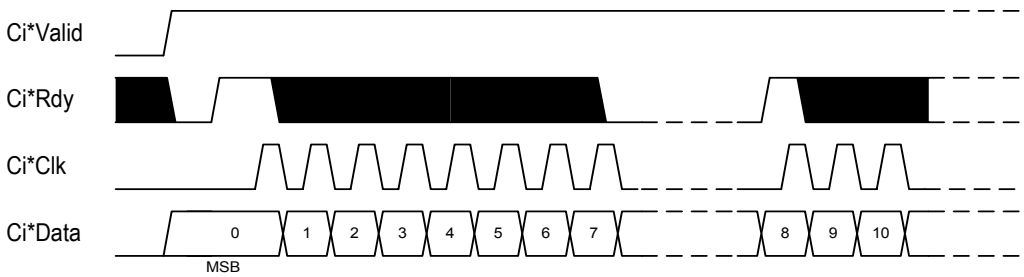


Figure 7-23 Transfer on Packet Wire interface with flow control

7.9.2 Timing

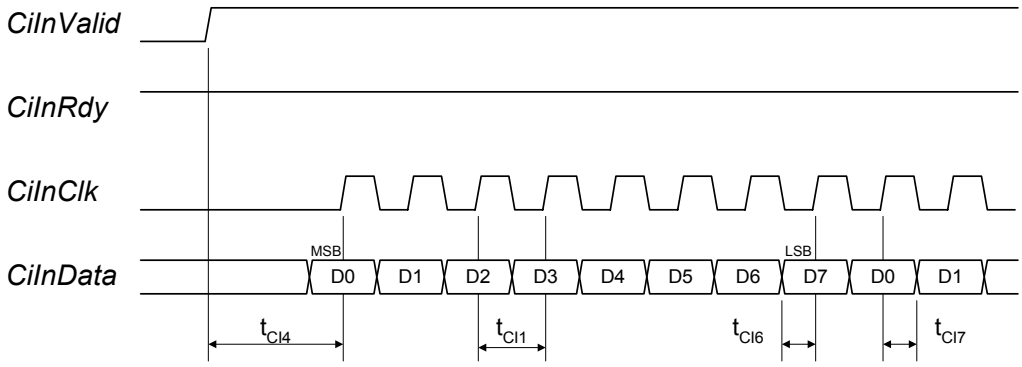


Figure 7-24 Start of packet

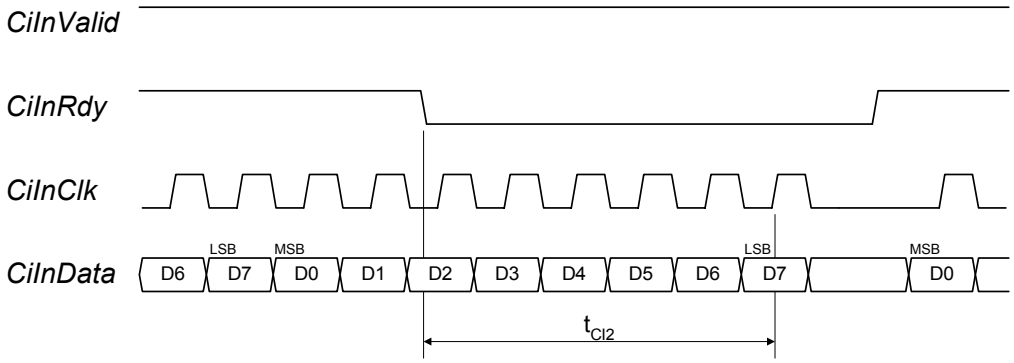


Figure 7-25 Transmission stalled by deasserting *CiInRdy*

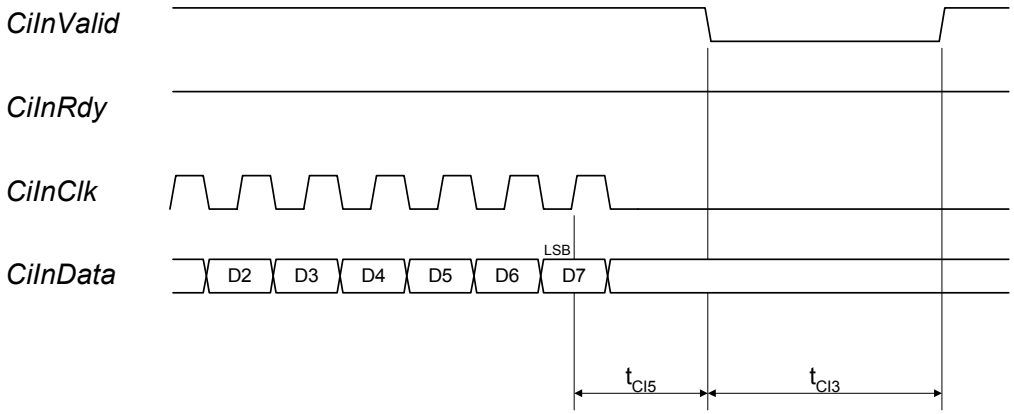


Figure 7-26 End of packet

Designation	Parameter	Reference edge	Min	Max	Unit
t_{CI1}	*CiInClk* period		0.5		t_{SysClk}
t_{CI2}	The time the sender have to stall the transaction when *CiInRdy* is deasserted. *CiInRdy* low to last bit in byte			16	t_{CI1}
t_{CI3}	*CiInValid* low to *CiInValid* high		6		t_{SysClk}
t_{CI4}	*CiInValid* input setup time	*CiInClk* ↑	4.1		ns
t_{CI5}	*CiInValid* input hold time	*CiInClk* ↑	0		ns
t_{CI6}	*CiInData* input setup time	*CiInClk* ↑	1.9		ns
t_{CI7}	*CiInData* input hold time	*CiInClk* ↑	0.2		ns

Table 7-17 Control Interface timing

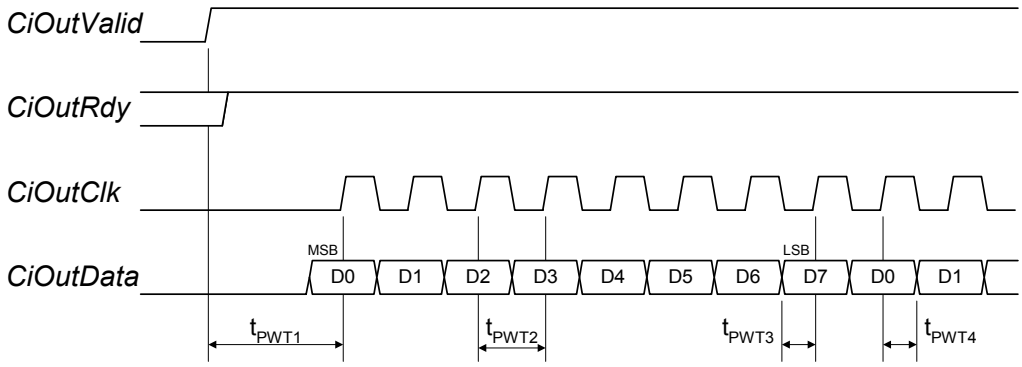


Figure 7-27 Start of packet

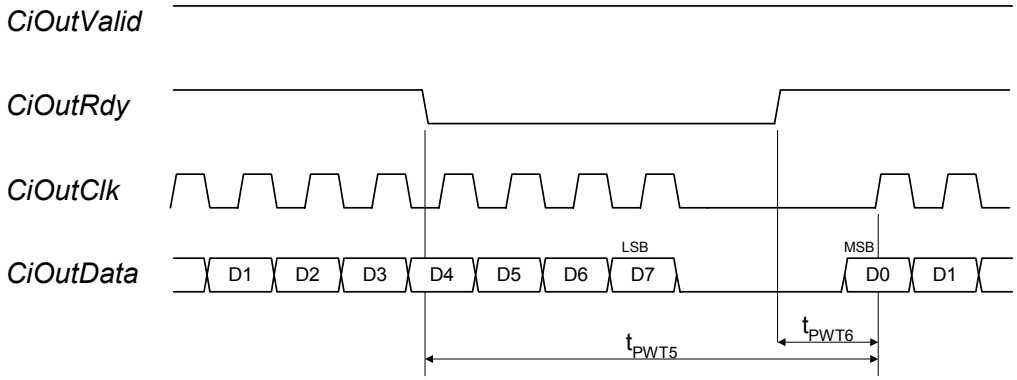


Figure 7-28 Transmission stalled by receiver (*CiOutRdy* deasserted)

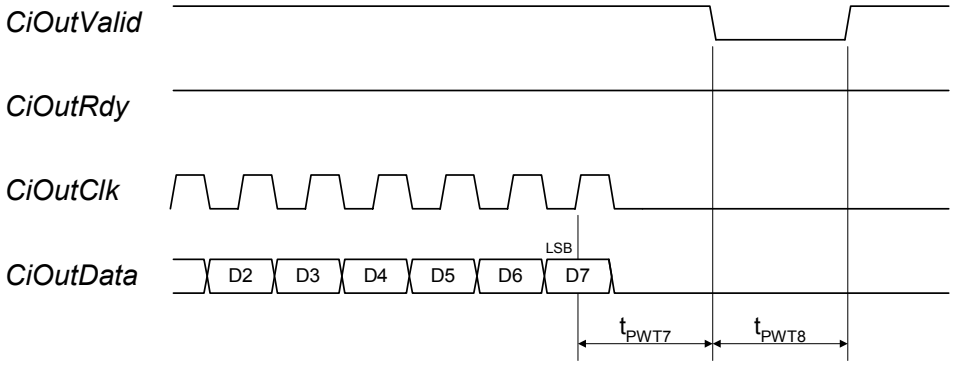


Figure 7-29 End of packet

Designation	Parameter	Reference edge	Min	Max	Unit
t_{PWT1}	*CiOutValid* high to *CiOutClk* high		1.5		t_{PWT2}
t_{PWT2}	*CiOutClk* period		2		t_{SysClk}
t_{PWT3}	*CiOutData* setup to *CiOutClk* high				t_{PWT2}
t_{PWT4}	*CiOutData* hold after *CiOutClk* high				t_{PWT2}
t_{PWT5}	*CiOutRdy* low before first bit in next byte		0.5 + 2		t_{PWT2} t_{SysClk}
t_{PWT6}	*CiOutRdy* high to *CiOutClk* high		0.5 + 2	1.5 + 3	t_{PWT2} t_{SysClk}
t_{PWT7}	*CiOutClk* high to *CiOutValid* low		1.5		t_{PWT2}
t_{PWT8}	*CiOutValid* low to *CiOutValid* high		1		t_{PWT2}
t_{PWT11}	*CiOutClk* output delay	*SysClk* ↑	8	35.4	ns
t_{PWT12}	*CiOutData* output delay	*SysClk* ↑	11	37.6	ns
t_{PWT13}	*CiOutValid* output delay	*SysClk* ↑	10	37.4	ns

Table 7-18 Packet Wire Output timing

Released

7.9.3 Reset

After a reset the values of the Control Interface module output signals are as follows:

Signal	Value after reset
CiInRdy	0
CiOutClk	0
CiOutData	0
CiOutValid	0

Table 7-19 Reset Values for Control Interface

7.10 Signal Definition Summary

The SCTMTC ASIC inputs and outputs are as defined in the tables below. For each signal, the type is indicated (CMOS, Schmitt). For each output signal, the drive strength is indicated (LD/MD/HD = Low/Mid/High Drive). PD stands for Pull-Down, and PDL stands for Pull-Down Low resistance. Clock in the rightmost column is the name of the reference clock if the signal is synchronous.

7.10.1 General SCTMTC ASIC signals

Signal	Type	Description	Clock
TestSE	I, CMOS, PD	Reserved for manufacturing test. Shall be tied to GND.	strap
TestMode	I, CMOS, PD	Reserved for manufacturing test. Shall be tied to GND.	strap
TestSignalIn[1:6]	I, CMOS	Reserved for manufacturing test. Shall be tied to GND.	strap
TestSignalOut[1:5]	O, CMOS, LD	Reserved for manufacturing test.	open

Table 7-20 Internal Scan Control interface signals

Signal	Type	Description	Clock
JtagTck	I, CMOS, PD	JTAG TAP Test Clock	-
JtagTdi	I, CMOS, PD	JTAG TAP Test Data In	*JtagTck*
JtagTms	I, CMOS, PD	JTAG TAP Test Mode Select	*JtagTck*
JtagTRstN	I, CMOS, PD	JTAG TAP Test Reset	*JtagTck*
JtagTdo	O, CMOS, LD	JTAG TAP Test Data Out	*JtagTck*

Table 7-21 Test Access Port interface signals

Signal	Type	Description	Clock
SysClk	I, CMOS	The main ASIC source clock. Can be configured to be the source clock for the *SpwClk* and *TmClk* as well.	-
SpwClk	I, CMOS	May be connected inside the ASIC to the internal *SpwClk*	-
TmClk1	I, CMOS	May be connected inside the ASIC to the *TmClk*	-
TmClk2	I, CMOS	May be connected inside the ASIC to the *TmClk*	-
PoResetN	I, Schmitt	Power On Reset. (Full reset)	-

Table 7-22 Clocks and Reset interface signals

Signal	Type	Description	Clock
Irq	O, CMOS, LD	Interrupt	*SysClk*
ReInit	I, Schmitt, PD	Re-initialise. Shall be tied to GND.	strap

Table 7-23 Configuration interface signals

7.10.2 Memory Interface

Signal	Type	Description	Clock
MemSize16	I, CMOS	Configuration for *PromCsN* area width	strap
MemD[15:0]	IO, CMOS, PD	Memory Data (big-endian)	*SysClk*
MemDcc[5:0]	IO, CMOS, PD	Memory Check Bits	*SysClk*
MemA[19:0]	O, CMOS, HD	Memory Address (byte address)	*SysClk*
MemOEN	O, CMOS, HD	Memory Output Enable	*SysClk*
MemWEN	O, CMOS, HD	Memory Write Enable	*SysClk*
PromCsN	O, CMOS, HD	Configuration PROM chip select	*SysClk*
ExtRecLacN	O, CMOS, HD	External Recovery LAC select (when external LAC is used)	*SysClk*
RamCsN	O, CMOS, HD	SRAM chip select	*SysClk*

Table 7-24 Memory Interface signals

7.10.3 Packet Telecommand Decoder Module (PDEC3) signals

Signal	Type	Description	Clock
PdecMapAdt	O, CMOS, LD	MAP Packet Abort.	*SysClk*
PdecMapClk	O, CMOS, MD	MAP Serial Clock.	*SysClk*
PdecMapData	O, CMOS, LD	MAP Serial Data.	*SysClk*
PdecMapDtrG	I, CMOS, PD	MAP data terminal ready. The asynchronous *PdecMapDtrG* input indicates that the destination for the general MAP interface is ready to receive the TC Segment, which can be used for flow control. When the input is high, the TC Segment will be transferred.	-
PdecMapDsrG	O, CMOS, LD	MAP data set ready. The *PdecMapDsrG* output indicates that a TC Segment is available for transfer on the general MAP interface. When the output is high the TC Segment is ready.	*SysClk*
PdecMapGenA[5:0]	O, CMOS, LD	6 bits MapGen address. These bits are generated from the real MAP address using the MAP allocation table.	*SysClk*
PdecMapDsr[1:5]	O, CMOS, LD	MAP data set ready. Each *PdecMapDsr** output indicates that a TC Segment is available for transfer on the corresponding MAP interface. When the output is high the TC Segment is ready. Only one *PdecMapDsr** output is active at any one time.	*SysClk*

Signal	Type	Description	Clock
PdecMapDtr[1:5]	I, CMOS, PD	MAP data terminal ready. Each asynchronous *PdecMapDtr** input indicates that the corresponding destination is ready to receive the TC Segment, which can be used for flow control. When the input is high, the TC Segment will be transferred.	-
PdecTcAct[5:0]	I, Schmitt, PD	TC channel active inputs, which serves as enable signals for the corresponding *PdecTcIn** and *PdecTcClk** inputs. A TC channel is active when its *PdecTcAct** input is high. These inputs can be asynchronous.	-
PdecTcClk[5:0]	I, Schmitt, PD	TC symbol clock inputs, recognised when the corresponding *PdecTcAct** input is asserted. These inputs can be asynchronous w.r.t. the ASIC clock.	-
PdecTcIn[5:0]	I, Schmitt, PD	TC symbol data stream inputs. The data shall be valid on the falling edge of the corresponding PdecTcClk* input.	-
PdecRfAvN[0:3]	I, Schmitt, PD	RF available indication for the CLCW. When the input is low, the "No RF Available" flag in the CLCW will be set to logic zero.	-
PdecClcwClk[0:1]	I, Schmitt, PD	CLCW clock. This asynchronous input clocks out the CLCW status using a protocol based on synchronous bit serial acquisition.	-
PdecClcwSamp[0:1]	I, Schmitt, PD	CLCW sample. This input sample the CLCW status before it is clocked out using a protocol based on synchronous bit serial acquisition. The CLCW is transferred when *PdecClcwSamp** is high. Each input must be synchronous to the corresponding *PdecClcwClk** input.	-
PdecClcwD[0:1]	O, CMOS, LD	CLCW data. This output provide the CLCW data serially after the rising edge of the corresponding *PdecClcwSamp** input or after the rising edge of the corresponding *PdecClcwClk** input.	*SysClk*
PdecMapSwitch	I, CMOS, PD	Swaps the signals between MAP interface 1 and MAP interface 2.	-

Signal	Type	Description	Clock
PdecTcPrior	I, CMOS, PD	TC channel priority mode configuration. When this static input is high, the active TC channels are selected with priority.	strap
PdecAuEnable	I, Schmitt, PD	AU enable input.	-

Table 7-25 PDEC3 interface signals

7.10.4 External CPDU Interfaces (ExtCpduIf) signals

Signal	Type	Description	Clock
ExtCpduIfClk	I, Schmitt, PD	Packet Wire serial bit clock	-
ExtCpduIfValid	I, Schmitt, PD	Packet valid	*ExtCpduIfClk*
ExtCpduIfRdy	O, CMOS, LD	Receiver ready, i.e. the interface is ready to receive a TC Segment.	*SysClk*
ExtCpduIfData	I, Schmitt, PD	Serial data	*ExtCpduIfClk*
ExtCpduIfAbort	I, Schmitt, PD	Abort packet	*ExtCpduIfClk*

Table 7-26 ExtCpduIf interface signals

7.10.5 CPDM Selector Module (CSEL) signals

Signal	Type	Description	Clock
CselStatusIn[2:0]	I, Schmitt, PD	Remote CSEL status	-
CselStatusOut[2:0]	O, CMOS, LD	CSEL status exchange	*SysClk*
CselRmOn	I, Schmitt, PD	Enables RM input access to the CPDM via the CSEL. When deasserted, it disables the RM input, i.e. if the ExtCpduIf is internally connected to the RM input it will not be able to transmit any pulse commands.	-

Table 7-27 CSEL interface signals

7.10.6 Command Pulse Distribution Module (CPDM) signals

Signal	Type	Description	Clock
CpdmStrb	O, CMOS, LD	This active high output provides the CPDU output pulse timing.	*SysClk*
CpdmArmN	O, CMOS, LD	This active low output provides an arming signal which can be used to power-on the CPDU output drivers	*SysClk*
CpdmSer	O, CMOS, MD	The pulse number is distributed serially.	*SysClk*
CpdmClk	O, CMOS, MD	Bit clock for the CpdmSer output	*SysClk*
CpdmClkAlive	I, Schmitt	Input indicating that the *CpdmClkToggle* is still toggling	-

Signal	Type	Description	Clock
CpdmClkToggle	O, CMOS, MD	The CPDM internal clock available as an external pin.	*SysClk*

Table 7-28 CPDM interface signals

7.10.7 Packet Telemetry Encoder Module (TME) signals

Signal	Type	Description	Clock
TmeClcwClk	O, CMOS, LD	CLCW clock. This output clocks out the CLCW status using a synchronous bit serial protocol.	*TmClk*
TmeClcwSamp	O, CMOS, LD	CLCW sample. This input defines the transfer of the CLCW using a synchronous bit serial protocol; the CLCW is transferred when *TmeClcwSamp* is set high.	*TmClk*
TmeClcwD[0:3]	I, CMOS, PD	Serial CLCW data. These inputs shall provide the data serially after the rising edge of the *TmeClcwSamp* input and after the rising edges of the *TmeClcwClk* input.	-
TmeUnEncClk	O, CMOS, HD	Clock output for the unencoded serial transfer frame, used to clock out *UnEncOut* and *UnEncSync*.	*TmClk*
TmeUnEncOut	O, CMOS, HD	Unencoded serial TM transfer frame output. This output is driven on the rising edge of *UnEncClk*.	*TmClk*
TmeUnEncSync	O, CMOS, MD	Synchronisation marker output indicator. This output is driven on the rising edge of *UnEncClk*, and it is set high while the synchronisation marker is output on *UnEncOut*.	*TmClk*
TmeEncClk	O, CMOS, HD	Clock output for the encoded serial transfer frames, used to clock out *TmeEncOut*.	*TmClk*
TmeEncOut	O, CMOS, HD	Encoded serial TM transfer frame output. This output is driven on the rising edge of *TmeEncClk*.	*TmClk*
TmeEncIQClk	O, CMOS, MD	Clock output for I/Q modulated output, used to clock out data on *TmeEncIOut* and *TmeEncQOut.*	*TmClk*
TmeEncIOut	O, CMOS, MD	I/Q modulated data output. This output is driven on the rising edge of *TmeEncIQClk*.	*TmClk*
TmeEncQOut	O, CMOS, MD	I/Q modulated data output. This output is driven on the rising edge of *TmeEncIQClk*.	*TmClk*

Signal	Type	Description	Clock
TmeTimeStrb	O, CMOS, LD	Time Sampling strobe output. Asserted to indicate that the on-board time should be sampled.	*TmClk*
TmeEnable	I, Schmitt, PD	Enable signal for the whole TME. When asserted the TME is enabled. When deasserted then all outputs from the TME are deasserted (inactive).	-
TmeSClk[A:H]	I, CMOS, PD	Packet Wire Serial Interface Clock. This input clocks serial data into a Virtual Channel serial interface.	-
TmeSIn[A:H]	I, CMOS, PD	Packet Wire Serial Data input. This input is sampled on the rising edge of TmeSClk* when TmeSValid is asserted.	-
TmeSRdy[A:H]	O, CMOS, LD	Packet Wire Virtual channel Ready. This signal is asserted to indicate that two octets can be received.	*SysClk*
TmeSValid[A:H]	I, CMOS, PD	Packet Wire Serial Interface Data Valid. This signal informs the corresponding VC interface when a telemetry packet begins/ends, and it also defines the byte boundaries in the input data stream. Input shall be asserted while data is being input.	-

Released

Table 7-29 TME interface signals

7.10.8 SpaceWire Module (SPW) signals

Signal	Type	Description	Clock
SpwDInA	I, CMOS	SpaceWire Serial Data Input from link A	-
SpwSInA	I, CMOS	SpaceWire Serial Strobe Input from link A	-
SpwDInB	I, CMOS	SpaceWire Serial Data Input from link B	-
SpwSInB	I, CMOS	SpaceWire Serial Strobe Input from link B	-
SpwDOut	O, CMOS, MD	SpaceWire Serial Data Output	*SpwClk*
SpwSOut	O, CMOS, MD	SpaceWire Serial Strobe Output	*SpwClk*
SpwIfSel	I, Schmitt, PD	Used for switching between the two SpaceWire interfaces, Low =A and High = B.	-

Table 7-30 SpaceWire interface signals

7.10.9 Control Interface Module (CI) signals

Signal	Type	Description	Clock
CiInClk	I, CMOS, PD	Input Packet Wire Serial Clock	-
CiInData	I, CMOS, PD	Input Packet Wire Serial Data	*CiInClk*

Signal	Type	Description	Clock
CiInRdy	O, CMOS, MD	Input Packet Wire Receiver Ready	*SysClk*
CiInValid	I, CMOS, PD	Input Packet Wire Packet Valid	*CiInClk*
CiOutClk	O, CMOS, MD	Output Packet Wire Serial Clock	*SysClk*
CiOutData	O, CMOS, MD	Output Packet Wire Serial Data	*SysClk*
CiOutRdy	I, CMOS, PD	Output Packet Wire Receiver Ready	-
CiOutValid	O, CMOS, MD	Output Packet Wire Packet Valid	*SysClk*

Table 7-31 CI interface signals

7.11 Absolute Maximum Ratings

The absolute maximum ratings for the SCTMTC ASIC are as defined in the table below.

Type	Value
Supply voltage, V_{DD}	4.0 V
Input voltage range	-0.5 to 6.0 V
Input Current per power pin	±60 mA
Input Current per signal pin	±10 mA
Continuous Output Current, one pin	±30 mA TBC
Soldering Lead Temp. 1.6 mm from case for max 10 s	$300^{O}C$
Storage Temperature	$-65^{O}C$ to $150^{O}C$
Maximum package power dissipation	2 W TBC

Table 7-32 Absolute maximum ratings

7.12 Operating Conditions

The device operates nominally under the operating conditions defined in the table below.

Parameter	Min	Typ	Max	Unit
Supply voltage	3.0	3.3	3.6	V
Operating temperature range	-55		125	^{O}C
SysClk frequency		16	22.2	MHz
SysClk duty cycle	30		70	%
SpwClk frequency		10	75	MHz
SpwClk duty cycle	5		95	%
*TmClk** frequency			*2* SysClk*	MHz
*TmClk** duty cycle	25		75	%
*TmeSClk** frequency		10	2**SysClk*	MHz
*TmeSClk** duty cycle	40		60	%
Radiation total dose	100			krad(Si)
SEU cross-section			$4 \cdot 10^{-6}$	cm^2/bit
SEU LET	110			MeV/mg/cm^2
SEU error rate			10^{-14}	errors/bit/day

Table 7-33 Operating Conditions

Duty cycle values are for typical clock frequencies. For exact clock pulse width info, see Table 7-3.

Released

7.13 Static Electrical Characteristics

The static electrical characteristics are defined using the supply voltage, operating temperature range, radiation dose and nominal clock frequency specified in section 7.12.

Para-meter		Condition	Min	Typ	Max	Unit
V_{IH}	High level input voltage	CMOS	2.0			V
		Schmitt	2.0			V
V_{IL}	Low level input voltage	CMOS			0.8	V
		Schmitt			0.8	V
V_{HYS}	Hysteresis	Schmitt		0.61		V
V_{OH}	High level output voltage	Low drive (LD): I_{OH}= -2.4 mA (TBC)	2.4			V
		Medium drive (MD): I_{OH}= -3.6 mA (TBC)	2.4			V
		High drive (HD): I_{OH}= -7.2 mA (TBC)	2.4			V
V_{OL}	Low level output voltage	Low drive (LD): I_{OL}= +2.4 mA (TBC)			0.4	V
		Medium drive (MD): I_{OL}= +3.6 mA (TBC)			0.4	V
		High drive (HD): I_{OL}= +7.2 mA (TBC)			0.4	V
I_{IL}	Low level input current	V_{in}=GND, V_{DD}=V_{DD}(max)	-1		1	µA
I_{IH}	High level input current	V_{in}=V_{DD}=V_{DD}(max)	-1		1	µA
I_{IHP}	High level input current, pull-down input	V_{in}=V_{DD}=V_{DD}(max)	10	30	70	µA
I_{OZ}	Output leakage current	Outputs disabled, GND < V_{out} < V_{DD}	-1		1	µA
I_{OZHP}	Output leakage current, pull-down output	Outputs disabled, V_{out} = V_{DD}	10		70	µA
C_{IN}	Input pin capacitance			2.4		pF
C_{IO}	I/O pin capacitance			6.6		pF
I_{DDSB}	Standby supply current	CLK = V_{DD}			TBD	µA
R_{PD}	Pull-down resistance		54	114	270	kΩ
R_{PDL}	Pull-down resistance low		27	57	135	kΩ

Table 7-34 Static electrical characteristics

Released

7.14 Dynamic Electrical Characteristics

All dynamic electrical characteristics are defined using the supply voltage, operating temperature range, radiation dose and nominal frequency of the clocks, specified in section 7.12.

Para-meter		Condition	Min	Typ	Max	Unit
I_{DD1}	Operating current	No load, V_{DD}=5.5V			TBD	mA
I_{DD2}	Operating current	C_L=50 pF, V_{DD}=5.5V			TBD	mA
t_r, t_f	Input rise and fall times	10%↔90%			TBD	ns
t_t	Output transition time	10%↔90%, C_L=50pF			TBD	ns
Δ_{tpdlh}	Capacitive propagation delay	LD output, C_L>50pF, Low to High transition	TBD		TBD	ns/pF
Δ_{tpdhl}	Capacitive propagation delay	LD output, C_L>50pF, High to Low transition	TBD		TBD	ns/pF
Δ_{tpdlh}	Capacitive propagation delay	HD output, C_L>50pF, Low to High transition	TBD		TBD	ns/pF
Δ_{tpdhl}	Capacitive propagation delay	HD output, C_L>50pF, High to Low transition	TBD		TBD	ns/pF

Table 7-35 Dynamic electrical characteristics

7.15 Packaging

Released

The package is a 256-pin MQFP with <TBD> mil pin spacing.

7.16 Thermal Characteristics

The thermal resistance R_{THJA} of the package is TBD ^{o}C/W.

7.17 Pinout

Signal	IO	Pin	Signal	IO	Pin
CiInClk	I	203	*MemA[6]*	O	49
CiInData	I	204	*MemA[7]*	O	47
CiInRdy	O	206	*MemA[8]*	O	46
CiInValid	I	207	*MemA[9]*	O	45
CiOutClk	O	209	*MemCs0N*	O	25
CiOutData	O	210	*MemCs2N*	O	24
CiOutRdy	I	208	*MemCs3N*	O	21
CiOutValid	O	212	*MemD[0]*	IO	20
CpdmArmN	O	228	*MemD[1]*	IO	18
CpdmClk	O	225	*MemD[10]*	IO	4
CpdmClkAlive	I	223	*MemD[11]*	IO	3
CpdmClkToggle	O	224	*MemD[12]*	IO	2
CpdmSer	O	227	*MemD[13]*	IO	256
CpdmStrb	O	229	*MemD[14]*	IO	255
CselRmOn	I	230	*MemD[15]*	IO	254
CselStatusIn[0]	I	233	*MemD[2]*	IO	17
CselStatusIn[1]	I	232	*MemD[3]*	IO	14
CselStatusIn[2]	I	231	*MemD[4]*	IO	13
CselStatusOut[0]	O	238	*MemD[5]*	IO	12
CselStatusOut[1]	O	237	*MemD[6]*	IO	11
CselStatusOut[2]	O	236	*MemD[7]*	IO	10
ExtCpduIfAbort	I	165	*MemD[8]*	IO	9
ExtCpduIfClk	I	171	*MemD[9]*	IO	6
ExtCpduIfData	I	169	*MemDcc[0]*	IO	253
ExtCpduIfRdy	O	168	*MemDcc[1]*	IO	252
ExtCpduIfValid	I	167	*MemDcc[2]*	IO	251
JtagTck	I	69	*MemDcc[3]*	IO	248
JtagTdi	I	66	*MemDcc[4]*	IO	246
JtagTdo	O	65	*MemDcc[5]*	IO	245
JtagTms	I	67	*MemOEN*	O	27
JtagTRstN	I	68	*MemSize16*	I	61
MemA[0]	O	59	*MemWEN*	O	28
MemA[1]	O	56	*PdecAuEnable*	I	134
MemA[10]	O	43	*PdecClcwClk[0]*	I	153
MemA[11]	O	41	*PdecClcwClk[1]*	I	131
MemA[12]	O	40	*PdecClcwD[0]*	O	150
MemA[13]	O	39	*PdecClcwD[1]*	O	129
MemA[14]	O	37	*PdecClcwSamp[0]*	I	151
MemA[15]	O	36	*PdecClcwSamp[1]*	I	130
MemA[16]	O	34	*PdecMapAdt*	O	97
MemA[17]	O	33	*PdecMapClk*	O	93
MemA[18]	O	31	*PdecMapData*	O	92
MemA[19]	O	30	*PdecMapDsr[1]*	O	89
MemA[2]	O	54	*PdecMapDsr[2]*	O	87
MemA[3]	O	53	*PdecMapDsr[3]*	O	85
MemA[4]	O	52	*PdecMapDsr[4]*	O	82
MemA[5]	O	50	*PdecMapDsr[5]*	O	80

Signal	IO	Pin	Signal	IO	Pin
PdecMapDsrG	O	91	*SysClk*	I	105
PdecMapDtr[1]	I	88	*TestMode*	I	63
PdecMapDtr[2]	I	86	*TestSE*	I	64
PdecMapDtr[3]	I	84	*TestSignalIn[1]*	I	62
PdecMapDtr[4]	I	81	*TestSignalIn[2]*	I	101
PdecMapDtr[5]	I	79	*TestSignalIn[3]*	I	102
PdecMapDtrG	I	90	*TestSignalIn[4]*	I	141
PdecMapGenA[0]	O	77	*TestSignalIn[5]*	I	244
PdecMapGenA[1]	O	76	*TestSignalIn[6]*	I	16
PdecMapGenA[2]	O	75	*TestSignalOut[1]*	O	213
PdecMapGenA[3]	O	74	*TestSignalOut[2]*	O	239
PdecMapGenA[4]	O	73	*TestSignalOut[3]*	O	240
PdecMapGenA[5]	O	70	*TestSignalOut[4]*	O	242
PdecMapSwitch	I	132	*TestSignalOut[5]*	O	243
PdecRfAvN[0]	I	100	*TmClk1*	I	103
PdecRfAvN[1]	I	98	*TmClk2*	I	104
PdecRfAvN[2]	I	95	*TmeClcwClk*	O	195
PdecRfAvN[3]	I	94	*TmeClcwD[0]*	I	201
PdecTcAct[0]	I	128	*TmeClcwD[1]*	I	198
PdecTcAct[1]	I	125	*TmeClcwD[2]*	I	197
PdecTcAct[2]	I	119	*TmeClcwD[3]*	I	196
PdecTcAct[3]	I	115	*TmeClcwSamp*	O	194
PdecTcAct[4]	I	112	*TmeEnable*	I	178
PdecTcAct[5]	I	109	*TmeEncClk*	O	189
PdecTcClk[0]	I	127	*TmeEncIOut*	O	190
PdecTcClk[1]	I	123	*TmeEncIQClk*	O	192
PdecTcClk[2]	I	117	*TmeEncOut*	O	188
PdecTcClk[3]	I	114	*TmeEncQOut*	O	191
PdecTcClk[4]	I	111	*TmeSClk[A]*	I	177
PdecTcClk[5]	I	108	*TmeSClk[B]*	I	172
PdecTcIn[0]	I	126	*TmeSClk[C]*	I	164
PdecTcIn[1]	I	120	*TmeSClk[D]*	I	159
PdecTcIn[2]	I	116	*TmeSClk[E]*	I	154
PdecTcIn[3]	I	113	*TmeSClk[F]*	I	148
PdecTcIn[4]	I	110	*TmeSClk[G]*	I	143
PdecTcIn[5]	I	107	*TmeSClk[H]*	I	138
PdecTcPrior	I	23	*TmeSIn[A]*	I	176
PoResetN	I	106	*TmeSIn[B]*	I	170
ReInit	I	202	*TmeSIn[C]*	I	163
Irq	O	235	*TmeSIn[D]*	I	158
SpwClk	I	219	*TmeSIn[E]*	I	152
SpwDInA	I	218	*TmeSIn[F]*	I	146
SpwDInB	I	215	*TmeSIn[G]*	I	142
SpwDOut	O	222	*TmeSIn[H]*	I	137
SpwIfSel	I	216	*TmeSRdy[A]*	O	174
SpwSInA	I	217	*TmeSRdy[B]*	O	173
SpwSInB	I	214	*TmeSRdy[C]*	O	161
SpwSOut	O	221	*TmeSRdy[D]*	O	156

Signal	IO	Pin
TmeSRdy[E]	O	155
TmeSRdy[F]	O	147
TmeSRdy[G]	O	144
TmeSRdy[H]	O	139
TmeSValid[A]	I	175
TmeSValid[B]	I	166
TmeSValid[C]	I	162
TmeSValid[D]	I	157
TmeSValid[E]	I	149
TmeSValid[F]	I	145
TmeSValid[G]	I	140
TmeSValid[H]	I	133
TmeTimeStrb	O	180
TmeUnEncClk	O	182
TmeUnEncOut	O	184
TmeUnEncSync	O	181
Vdd[1]		60
Vdd[10]		234
Vdd[11]		220
Vdd[12]		205
Vdd[13]		187
Vdd[14]		179
Vdd[15]		124
Vdd[16]		99
Vdd[17]		83
Vdd[18]		8
Vdd[19]		58
Vdd[2]		51
Vdd[20]		72
Vdd[21]		122
Vdd[22]		136
Vdd[23]		186
Vdd[24]		200
Vdd[25]		250
Vdd[3]		44
Vdd[4]		38
Vdd[5]		32
Vdd[6]		26
Vdd[7]		19
Vdd[8]		5
Vdd[9]		247
Vss[1]		55
Vss[10]		226
Vss[11]		211
Vss[12]		193
Vss[13]		183
Vss[14]		160
Vss[15]		118
Vss[16]		96
Vss[17]		78
Vss[18]		7
Vss[19]		57
Vss[2]		48
Vss[20]		71
Vss[21]		121
Vss[22]		135
Vss[23]		185
Vss[24]		199
Vss[25]		249
Vss[3]		42
Vss[4]		35
Vss[5]		29
Vss[6]		22
Vss[7]		15
Vss[8]		1
Vss[9]		241

7.18 JTAG Pin Order

The table below defines the order of bits in the Boundary Scan chain, counting from the *JTagTdi* input to the *JTagTdo* output. Thus the last signal in the table will be the first to appear on *JTagTdo* when shifting out values.
IO pins are controlled by the bits denoted “Enable”.

Bit	Signal
1	Enable: *MemD[0]*
2	Enable: *MemD[1]*
3	Enable: *MemD[2]*
4	Enable: *MemD[3]*
5	Enable: *MemD[4]*
6	Enable: *MemD[5]*
7	Enable: *MemD[6]*
8	Enable: *MemD[7]*
9	Enable: *MemD[8]*
10	Enable: *MemD[9]*
11	Enable: *MemD[10]*
12	Enable: *MemD[11]*
13	Enable: *MemD[12]*
14	Enable: *MemD[13]*
15	Enable: *MemD[14]*
16	Enable: *MemD[15]*
17	Enable: *MemDcc[0]*
18	Enable: *MemDcc[1]*
19	Enable: *MemDcc[2]*
20	Enable: *MemDcc[3]*
21	Enable: *MemDcc[4]*
22	Enable: *MemDcc[5]*
23	Unused
24	Unused
25	Unused
26	Unused
27	Unused
28	Unused
29	Unused
30	Unused
31	Unused
32	Unused
33	Unused
34	Unused
35	Unused
36	Unused
37	Unused
38	Unused
39	Unused
40	Unused
41	Unused
42	Unused
43	Unused
44	Unused
45	Unused
46	Unused
47	Unused
48	Unused

Bit	Signal
49	Unused
50	Unused
51	Unused
52	Unused
53	Unused
54	Unused
55	Unused
56	*TestSignalIn[1]*
57	Unused
58	*MemSize16*
59	Unused
60	*MemA[0]*
61	*MemA[1]*
62	Unused
63	Unused
64	*MemA[2]*
65	*MemA[3]*
66	Unused
67	*MemA[4]*
68	*MemA[5]*
69	Unused
70	*MemA[6]*
71	Unused
72	Unused
73	*MemA[7]*
74	*MemA[8]*
75	*MemA[9]*
76	Unused
77	*MemA[10]*
78	Unused
79	*MemA[11]*
80	*MemA[12]*
81	*MemA[13]*
82	Unused
83	Unused
84	*MemA[14]*
85	Unused
86	*MemA[15]*
87	Unused
88	*MemA[16]*
89	Unused
90	*MemA[17]*
91	Unused
92	*MemA[18]*
93	*MemA[19]*
94	Unused
95	*MemWEN*
96	*MemOEN*

Bit	Signal
97	Unused
98	*MemCs0N*
99	*MemCs2N*
100	Unused
101	*PdecTcPrior*
102	*MemCs3N*
103	Unused
104	Unused
105	*MemD[0]*
106	*MemD[1]*
107	Unused
108	*MemD[2]*
109	Unused
110	*TestSignalIn[6]*
111	Unused
112	*MemD[3]*
113	*MemD[4]*
114	Unused
115	*MemD[5]*
116	*MemD[6]*
117	Unused
118	*MemD[7]*
119	*MemD[8]*
120	Unused
121	*MemD[9]*
122	Unused
123	*MemD[10]*
124	*MemD[11]*
125	*MemD[12]*
126	*MemD[13]*
127	*MemD[14]*
128	Unused
129	*MemD[15]*
130	Unused
131	*MemDcc[0]*
132	*MemDcc[1]*
133	Unused
134	*MemDcc[2]*
135	*MemDcc[3]*
136	Unused
137	Unused
138	*MemDcc[4]*
139	*MemDcc[5]*
140	*TestSignalIn[5]*
141	Unused
142	*TestSignalOut[5]*
143	*TestSignalOut[4]*
144	Unused

Bit	Signal
145	Unused
146	Unused
147	*TestSignalOut[3]*
148	*TestSignalOut[2]*
149	*CselStatusOut[0]*
150	*CselStatusOut[1]*
151	Unused
152	*CselStatusOut[2]*
153	*Irq*
154	Unused
155	*CselStatusIn[0]*
156	*CselStatusIn[1]*
157	*CselStatusIn[2]*
158	Unused
159	*CselRmOn*
160	Unused
161	*CpdmStrb*
162	Unused
163	*CpdmArmN*
164	*CpdmSer*
165	Unused
166	Unused
167	*CpdmClk*
168	Unused
169	*CpdmClkToggle*
170	*CpdmClkAlive*
171	*SpwDOut*
172	*SpwSOut*
173	Unused
174	*SpwClk*
175	Unused
176	*SpwDInA*
177	*SpwSInA*
178	*SpwIfSel*
179	Unused
180	*SpwDInB*
181	*SpwSInB*
182	*TestSignalOut[1]*
183	Unused
184	Unused
185	*CiOutValid*
186	*CiOutData*
187	Unused
188	*CiOutClk*
189	Unused
190	*CiOutRdy*
191	Unused
192	*CiInValid*

Bit	Signal
193	*CiInRdy*
194	Unused
195	*CiInData*
196	*CiInClk*
197	Unused
198	*ReInit*
199	*TmeClcwD[0]*
200	Unused
201	*TmeClcwD[1]*
202	*TmeClcwD[2]*
203	Unused
204	*TmeClcwD[3]*
205	*TmeClcwClk*
206	*TmeClcwSamp*
207	*TmeEncIQClk*
208	*TmeEncIOut*
209	Unused
210	*TmeEncQOut*
211	Unused
212	*TmeEncClk*
213	*TmeEncOut*
214	Unused
215	*TmeUnEncOut*
216	Unused
217	Unused
218	*TmeUnEncClk*
219	*TmeUnEncSync*
220	Unused
221	*TmeTimeStrb*
222	*TmeEnable*
223	Unused
224	*TmeSClk[A]*
225	Unused
226	Unused
227	*TmeSIn[A]*
228	*TmeSValid[A]*
229	*TmeSRdy[A]*
230	*TmeSRdy[B]*
231	Unused
232	*TmeSClk[B]*
233	*ExtCpduIfClk*
234	Unused
235	*TmeSIn[B]*
236	*ExtCpduIfData*
237	*ExtCpduIfRdy*
238	*ExtCpduIfValid*
239	Unused
240	*TmeSValid[B]*

Bit	Signal
241	Unused
242	*ExtCpduIfAbort*
243	Unused
244	*TmeSClk[C]*
245	*TmeSIn[C]*
246	Unused
247	*TmeSValid[C]*
248	Unused
249	*TmeSRdy[C]*
250	Unused
251	*TmeSClk[D]*
252	*TmeSIn[D]*
253	*TmeSValid[D]*
254	Unused
255	*TmeSRdy[D]*
256	*TmeSRdy[E]*
257	Unused
258	*TmeSClk[E]*
259	*PdecClcwClk[0]*
260	*TmeSIn[E]*
261	Unused
262	*PdecClcwSamp[0]*
263	*PdecClcwD[0]*
264	*TmeSValid[E]*
265	Unused
266	Unused
267	*TmeSClk[F]*
268	*TmeSRdy[F]*
269	*TmeSIn[F]*
270	Unused
271	*TmeSValid[F]*
272	Unused
273	*TmeSRdy[G]*
274	Unused
275	*TmeSClk[G]*
276	*TmeSIn[G]*
277	*TestSignalIn[4]*
278	Unused
279	*TmeSValid[G]*
280	*TmeSRdy[H]*
281	Unused
282	*TmeSClk[H]*
283	*TmeSIn[H]*
284	Unused
285	*PdecAuEnable*
286	*TmeSValid[H]*
287	Unused
288	*PdecMapSwitch*

Bit	Signal
289	*PdecClcwClk[1]*
290	*PdecClcwSamp[1]*
291	*PdecClcwD[1]*
292	*PdecTcAct[0]*
293	*PdecTcClk[0]*
294	Unused
295	*PdecTcIn[0]*
296	Unused
297	*PdecTcAct[1]*
298	Unused
299	*PdecTcClk[1]*
300	*PdecTcIn[1]*
301	Unused
302	*PdecTcAct[2]*
303	Unused
304	*PdecTcClk[2]*
305	Unused
306	*PdecTcIn[2]*
307	*PdecTcAct[3]*
308	*PdecTcClk[3]*
309	Unused
310	*PdecTcIn[3]*
311	Unused
312	Unused
313	*PdecTcAct[4]*
314	*PdecTcClk[4]*
315	*PdecTcIn[4]*
316	*PdecTcAct[5]*
317	Unused
318	*PdecTcClk[5]*
319	*PdecTcIn[5]*
320	Unused
321	*PoResetN*
322	*SysClk*
323	*TmClk2*
324	*TmClk1*
325	Unused
326	*TestSignalIn[3]*
327	Unused
328	*TestSignalIn[2]*
329	Unused
330	*PdecRfAvN[0]*
331	Unused
332	*PdecRfAvN[1]*
333	Unused
334	*PdecMapAdt*
335	Unused
336	*PdecRfAvN[2]*

Bit	Signal
337	*PdecRfAvN[3]*
338	*PdecMapClk*
339	Unused
340	*PdecMapData*
341	*PdecMapDsrG*
342	Unused
343	*PdecMapDtrG*
344	*PdecMapDsr[1]*
345	*PdecMapDtr[1]*
346	Unused
347	*PdecMapDsr[2]*
348	*PdecMapDtr[2]*
349	*PdecMapDsr[3]*
350	Unused
351	Unused
352	*PdecMapDtr[3]*
353	*PdecMapDsr[4]*
354	Unused
355	*PdecMapDtr[4]*
356	Unused
357	*PdecMapDsr[5]*
358	Unused
359	*PdecMapDtr[5]*
360	*PdecMapGenA[0]*
361	Unused
362	*PdecMapGenA[1]*
363	*PdecMapGenA[2]*
364	Unused
365	*PdecMapGenA[3]*
366	*PdecMapGenA[4]*
367	Unused
368	*PdecMapGenA[5]*
369	Unused