SMCS332SpW

User Manual

Prepared by:	U. Liebstückel – ASE232 *System Engineer*		**Date:** 11.09.06
Checked by:	Dr. S. Fischer – ASE211 *Project Manager*		**Date:** 11.09.06
Checked by:	P. Rastetter - ASE4232 *System Engineer*		**Date:** 11/9/06
Released by:	Dr. S. Fischer – ASE211 *Project Manager*		**Date:** 11.09.06

Document Revision History

Revision	Date	Responsible	Modifications / Reasons for Change
1.0	June-05	Uwe Liebstückel	First release
1.1	Nov-05	Uwe Liebstückel	Section 11.3 Special Behaviour added Section 4.2.2.8 Checksum generation figure added
1.2	Nov-05	Uwe Liebstückel	Section 11.3 Special Behaviour deleted Section 12 Handling Empty Packets added
1.3	Jul-06	Uwe Liebstückel	Section 12 Handling Empty Packets changed Section 4.2.2.3 Updated
1.4	Sep-06	Stephan Fischer	Section 12 editorial modifications (ESA comments)

Table of Contents

1 Scope and Objectives

This document describes in detail the new SMCS332SpW. The SMCS332SpW provides an interface between three SpaceWire links according to the SpaceWire Standard ECSS-E-50-12A and a data processing node like a CPU.

1.1 List of applicable documents

Number		Document
AD1	ECSS-E-50-12A	SpaceWire -Links, nodes, routers and network 24 January 2003
AD2	SMCS332SpW_RS-01	SMCS332SpW Requirements Specification

1.2 Reference Documents

The documents below have been used as support for establishing this document and contain background information relating to the subjects addressed.

Number		Document
RD1	Issue 2, 21.04.1999	SMCS332 User Manual

1.3   List of Abbreviations

AD   Applicable Document

API   Application Programming Interface

ASIC   Application Specific Integrated Circuit

CODEC   COder / DECoder

COMI   Communication Memory Interface

DPU   Data Processing Unit

DSP   Digital Signal Processor

EOP   End Of Packet

EEP   Error End Of Packet

ESC   Escape

FCT   Flow Control Token

FIFO   First In First Out

FPGA   Field Programmable Gate Array

GPIO   General Purpose Input Output

JTAG   Joint Testing Action Group

HOCI   Host Control Interface

LVDS   Low Voltage Differential Signalling

LSB   Least Significant Bit

MSB   Most Significant Bit

PCB   Printed Circuit Board

PE   Processing Element

PPU   Protocol Processing Unit

PRCI   Protocol Command Interface

SIC   Simple Interprocessor Communication

SRAM   Static Random Access Memory

SSRAM   Synchronous Static Random Access Memory

SMCS   Scalable Multichannel Communication Subsystem

TBC   to be confirmed

TBD   to be defined

2   Introduction

The SMCS332SpW provides an interface between 3 SpaceWire links according to the SpaceWire Standard ECSS-E-50-12A specification and a data processing node consisting of a CPU and a communication data memory. The SMCS332SpW provides HW supported execution of the major parts of the simple interprocessor communication protocol, particularly:

·   transfer of data between two nodes of a multi-processor system with minimal host CPU intervention,

·   execution of simple commands to provide basic features for system control functions,

·   provision of fault tolerant features.

However, with disabling of features such as the protocol handling or with reduction of the transmit rate also low power usage is supported.
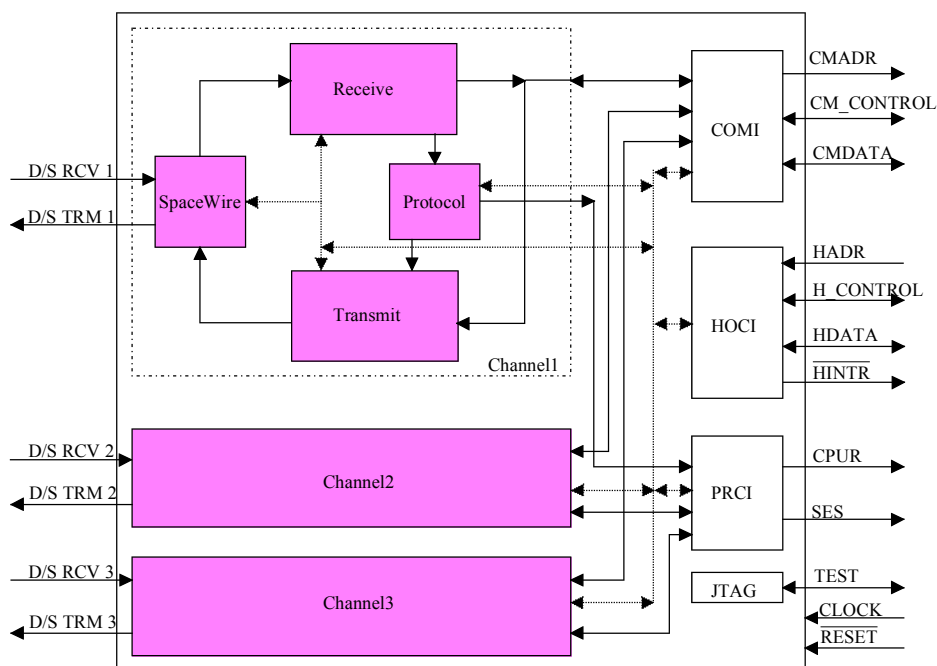
Figure 1: SMCS332SpW Block Diagram

Target applications are heterogeneous multi-processor systems supported by scalable interfaces including the little/big endian byte swapping. The SMCS332SpW connects modules with different processors (e.g. ADSP21020, SHARC, ERC32, TMS320C40). Any kind of network topology could be realized through the high speed point-to-point SpaceWire-links (see chapter Applications in sectioon 2.8).

2.1   Interfaces

The SMCS332SpW consists of the following blocks (See block diagram of the SMCS332SpW in Figure 1):

•   3 bidirectional SpaceWire [AD1] channels, all comprising the DS-link SpaceWire cell, receive and transmit sections (each including FIFOs) and a protocol processing unit (PPU). Each channel allows full duplex communication up to 200 Mbit/s in each direction. With protocol command execution a higher level of communication is supported. Link disconnect detection and parity check at character level are performed. A checksum generation for a check at packet level can be enabled.

The transmit rate is selectable between 1.25 and 200 Mbit/s. The start up transmit rate is 10 Mbit/s. For special applications the data transmit rate can be programmed to values even below 10 Mbit/s; the lowest possible SpaceWire transmit rate is 1.25 Mbit/s (the next values are 2.5 and 5 Mbit/s).

•   Communication Memory Interface (COMI) performs autonomous accesses to the communication memory of the module to store data received via the links or to read data to be transmitted via the links. The COMI consists of individual memory address generators for the receive and transmit direction of every SpaceWire link channel. The access to the memory is controlled via an arbitration unit providing a fair arbitration scheme. Two SMCS332SpW can share one DPRAM without external arbitration logic.

The data bus width is scalable (8/16/32 bit) to allow flexible integration with any CPU type.

Operation in little or big endian mode is configurable through internal registers.

The COMI address bus is 16 bit wide allowing direct access of up to 64K words (32 bit) of the DPRAM. Two chip select signals are provided to allow splitting of the 64k address space in two memory banks.

•   Host Control Interface (HOCI) gives read and write access to the SMCS332SpW configuration registers and to the SpaceWire channels for the controlling CPU. Viewed from the CPU, the interface behaves like a peripheral that generates acknowledges to synchronize the data transfers and which is located somewhere in the CPU's address space.

Packets can be transmitted or received directly via the HOCI. In this case the Communication Memory (DPRAM) is not strictly needed. However, in this case the packet size should be limited to avoid frequent CPU interaction.

The data bus width is scalable (8/16/32 bit) to allow flexible integration with any CPU type. The byte alignment can be configured for little or big endian mode through an external pin.

Additionally the HOCI contains the interrupt signalling capability of the SMCS by providing an interrupt output, the interrupt status register and interrupt mask register to the local CPU.

A special pin is provided to select between control of the SMCS332SpW by HOCI or by link. If control by link is enabled, the host data bus functions as a 32-bit general purpose interface (GPIO).

•   Protocol Command Interface (PRCI) that collects the decoded commands from all PPUs and forwards them to external circuitry via 5 special pins.

•   JTAG Test Interface that represents the boundary scan testing provisions specified by IEEE Standard 1149.1 of the Joint Testing Action Group (JTAG). The SMCS' test access port and on-chip circuitry is fully compliant with the IEEE 1149.1 specification. The test access port enables boundary scan testing of circuitry connected to the SMCS I/O pins.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 11 of 131

2.2   Operation Modes

According to the different protocol formats expected for the operation of the SMCS332SpW, two major operation modes are implemented into the SMCS332SpW. The operation modes are chosen individually for each link channel by setting the respective configuration registers via the HOCI or via the link.

- Transparent Mode (default after reset): This mode allows complete transparent data transfer between two nodes without performing any interpretation of the data bytes and without generating any acknowledges. It is completely up to the host CPU to interpret the received data and to generate acknowledges if required.

    The SMCS332SpW accepts EOP and EEP control tokens as packet delimiters and generates autonomously EOP or no EOP (as configured) marker after each end of a transmission packet.

    This mode also includes as a special sub mode:

    - Wormhole routing: This mode allows hardware routing of packets by the SMCS332SpW.

- Simple Interprocessor Communication (SIC) Protocol Mode: This mode executes the simple interprocessor communication protocol as described in chapter 13. The following capabilities of the protocol are implemented into the SMCS332SpW:

    - interpretation of the first 4 data characters as the header bytes of the protocol

    - autonomous execution of the simple control commands as described in the protocol specification

    - autonomous acknowledgement of received packets if configured

    In transmit direction no interpretation of the data is performed. This means that for transmit packets, the four header bytes must be generated by the host CPU and must be available as the first data read from the communication memory. EOP control characters are automatically inserted by the SMCS332SpW if one configured transfer from the communication memory has finished.

2.3   SMCS332SpW Control by SpaceWire link

A feature of the SMCS332SpW is the possibility to control the SMCS332SpW not only via HOCI but via one of the three links. This allows to use the SMCS332SpW in systems without a local controller (µController, FPGA etc.). Since the HOCI is no longer used in this operation mode, it is instead available as a set of general purpose I/O (GPIO) lines. The detailed description of this operation mode is given in section 5.4.

2.4   Wormhole Routing

The SMCS332SpW introduces a wormhole routing function similar to the routing implemented in the SpaceWire Router. Each of the three links and the SMCS332SpW itself can be assigned an eight bit address. When routing is enabled in the SMCS332SpW, the first byte of a packet will be interpreted as the address destination byte, analyzed and removed from the packet (header deletion). If this address matches one of the two other link addresses or the SMCS332SpW address assigned previously, the packet will be automatically forwarded to this link or the FIFO of the SMCS332SpW. If the header byte does not match a link address, the packet will be written to the internal FIFO as well and an error interrupt (maskable) will be raised.

2.5 PPU Functional Description

Since the Protocol Processing Unit (PPU) determines a major part of the SMCS332SpW functionality, the principal blocks of the PPU and their function are described here. This functionality is provided for every SpaceWire channel of the SMCS332SpW.

• Protocol Execution Unit: This unit serves as the main controller of the PPU block. It receives the character from the SpaceWire cell and interprets (in protocol mode) the four header data characters received after an EOP control character. If the address field matches the link channel address and the command field contains a valid command then forwarding of data into the receive FIFO is enabled. If the command field contains a "simple control command" then the execution request is forwarded to the command execution unit.

The protocol execution unit also enables forwarding of header data characters to the acknowledge generator and provides an error signal in case of address mismatch, wrong commands or disabled safety critical "simple control commands".

The protocol execution unit is disabled in "transparent" or "wormhole routing" operation mode.

• Receive, Transmit, Acknowledge: the transmit and receive FIFOs decouple the SpaceWire link related operations from the SMCS332SpW related operations in all modes and such allows to keep the speed of the different units even when the source or sink of data is temporarily blocked.

In the protocol mode a further FIFO (acknowledge FIFO) is used to decouple sending of acknowledges from receiving new data when the transmit path is currently occupied by a running packet transmission.

• Command Execution Unit: This unit performs activating resp. deactivating of the CPU reset and the specific external signals and provides the capability to reset one or all links inside the SMCS332SpW, all actions requested by the decoded commands from the protocol execution unit.

The unit contains a register controlling the enable/disable state of safety critical commands which is set into the 'enable' state upon command request and which is reset after a safety critical command has been executed.

The CPU reset and the specific external signals are forwarded to the Protocol Command Interface (PRCI).

2.6 Fault Tolerance

The SpaceWire standard specifies low level checks as link disconnect, credit value, sequence and parity at token level. The SMCS332SpW provides, through the Protocol Processing Unit, features to reset a link or all links inside the SMCS332SpW, to reset the local CPU or to send special signals to the CPU commanded via the links.

Additionally it is possible to enable a checksum coder/decoder to have fault detection capabilities at packet level.

2.7 Software Support

The SMCS332SpW is supported by VSPWorks from Wind River, a commercially available distributed real-time kernel. It is a multi-tasking as well as a multi-processor Operating System. The main goals are to enable programming at a higher level to configure and to perform communication and to administer the tasks on a board with multiple processes running in parallel.

The VSPWorks kernel supports multiple processors and application specific chips, e.g. the SHARC, ADSP21020, TMS320C40, SPARC ERC32 etc. Thus it is possible to run a heterogeneous multiprocessor system with a single Operating System without consideration of the hardware platform.

2.8   Application

The SMCS332SpW can be used for single board systems where standardised high speed interfaces are needed. Even "non-intelligent" modules such as A/D-converter or sensor interfaces can be assembled with the SMCS332SpW because of the "control by link" feature. The complete control of the SMCS332SpW can be done via link from a central controller-node.

The SMCS332SpW is a very high speed, scalable link-interface chip with fault tolerance features. The initial exploitation is for use in multi-processor systems where the standardization or the high speed of the links is an important issue and where reliability is a requirement. Further application examples are heterogeneous systems (as shown in Figure 2) or modules without any communication features as special image compression chips, certain signal processors (ADSP21020, MC56000), application specific programmable logic or mass memory.

Heterogeneous Systems. These systems are often used for applications where different kind of tasks are to be processed. Figure 2 below shows an application with multiple processing modules using the SMCS332SpW for interprocessor communication.
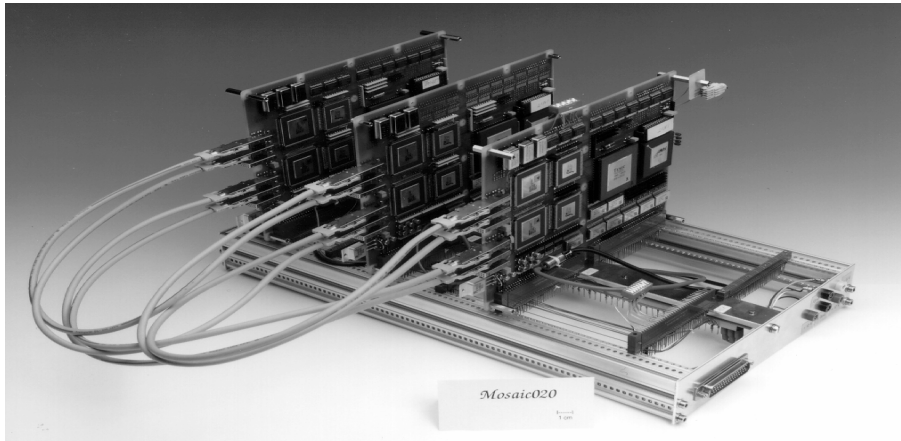
Figure 2: Multiprocessor System with SMCS332SpW link connections

2.9   SMCS332SpW connected with the old SMCS332

The new ECSS-E-50-12A SpaceWire standard, used by the SMCS332SpW and the IEEE-1355 standard used by the old SMCS332 are compatible, however the old SMCS332 has a slightly different startup behavior, the SMCS332 must be started first on a link connecting with the SMCS332SpW.

3   The SpaceWire link

The SpaceWire standard [AD1] defines a full duplex bit serial point to point link with a raw transmit rate of up to 400 Mbit/sec. The link consists of 2 signals in each direction, one for strobe and one for data. By coding the strobe that it only changes level when the data does not, clock recovery and data synchronization can be achieved by XOR-ing of data and strobe signals without having the need to run the strobe at very high frequencies. The Character level defines the data and control characters used to manage the flow of data across a link. The exchange level of the protocol is used to implement flow control which avoids overflow of the front end buffers. Also error detection is provided by implemented parity checks during transmission and by timeout supervision in case of inter-connect failures.

The SpaceWire standard aims only to define a transport medium between two nodes and covers the protocol layers only up to the packet level. This has two consequences:

- packets with address headers allow to use this link standard in networks using routers,

- since the standard does not define the data payload within the packets, an efficient transaction layer definition is missing.

To compensate for these deficiencies of the SpaceWire specification, the SMCS332SpW implementations (the SMCS332SpW and the SMCS116SpW) introduce an (optional) transaction layer extension to the SpaceWire protocol standard. This high-level protocol extension supports applications in fault tolerant systems, heterogeneous architectures, feature power saving modes and remote configuration of the communication controller and autonomous command execution. With this flexible and powerful protocol, the SpaceWire link has many advantages over commonly used interface solutions such as RS-485 etc.

3.1   Data/Strobe SpaceWire signals

The SpaceWire links use a protocol with two wires in each direction, one for data and one to carry a strobe signal and are also referred to as data/strobe (DS-Links). Each DS pair carries characters and an encoded clock. The characters can be data or control characters. The figure below shows the format of data and control characters on the data and strobe wires. Data characters are 10 bits long and consist of a parity bit, a control flag which is set to 0 to indicate a data character and 8 bits of data. Control characters are 4 bits long and consist of a parity bit, a control flag which is set to 1 to indicate a control character, and 2 bits to indicate the type of control character. One of the four possible control characters is the escape code ESC. The ESC code is used to form control codes. Two control codes are specified, the NULL code and the time code. The NULL code is transmitted whenever a link is not sending data or control characters. The time code is used to distribute system time over a SpaceWire network, which comprises a ESC control character followed by a single data character.

The DS-Link protocol ensures that only one of the two wires of the data strobe pair has an edge in each bit time. The levels on the data wire give the data bits transmitted. The strobe signal changes whenever the data signal does not. These two signals encode a clock together with the data bits, permitting asynchronous detection of the data at the receiving end. The data and control characters are of different lengths, for this reason the parity bit in any characters covers the parity of the data or control bits in the previous characters, and the control flag in the same character, as shown in the above figure. This allows single bit errors in the character type flag to be detected. Odd parity checking is used. Thus the parity bit is set/unset to ensure that the bits covered, inclusive of the parity bit (see below figure), always contain an odd number of 1's. The coding of the characters is shown in the table below To ensure the immediate detection of parity errors and to enable link disconnection to be detected NULL code are sent in the absence of other tokens.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 15 of 131
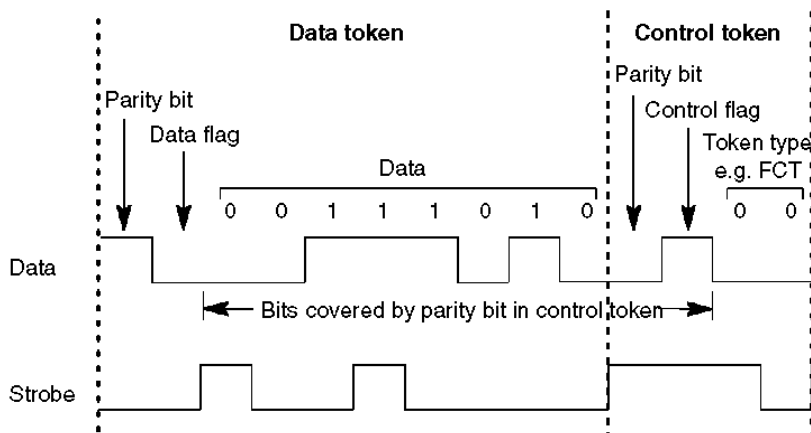
Character Type	Abbreviation	Coding
Data character		P0DDDDDDD
control characters:		
Flow control	FCT	P100
Normal End of Packet	EOP	P101
Error End of Packet	EEP	P110
Escape	ESC	P111
control codes:		
Null	NULL	ESC + FCT P1110100
Time code		ESC + DATA P11110DDDDDDDD

P = Parity bit

D = Data bit

3.2   Character level flow control

Character level flow control is performed in each SpaceWire module, and the additional flow control characters used are not visible to the higher-level packet protocol. The character level flow control mechanism prevents a sender from overrunning the input buffer of a receiving link. Each receiving link input contains a buffer for at least 8 normal-characters (16 normal-characters of buffering is in fact provided). Normal-characters are data character and EOP/EEP. Whenever the link input has sufficient buffering available to consume a further 8 normal-characters a FCT is transmitted on the associated link output, and this FCT gives the sender permission to transmit of further 8 normal-characters. Once the sender has transmitted 8 normal-characters it waits until it receives another FCT before transmitting any more tokens. The provision of more than 8 normal-characters of buffering on each link input ensures that in practice the next FCT is received before the previous block of 8 normal-characters has been fully transmitted, so the character level flow control does not restrict the maximum bandwidth of the link.

For further information see [AD1]

3.3 Link speeds

The SpaceWire links can support a range of communication speeds, which are programmed by writing to registers. At reset all links are configured to run at the base speed of 10 Mbits/sec. Only the transmission speed of a link is programmed as reception is asynchronous. This means that links running at different speeds can be connected, provided that each device is capable of receiving at the speed of the connected transmitter. The transmission speeds of all of the links on a given device are related to the maximum transmission rate programmed for the complete device. This maximum transmission rate is programmed through the transmit bitrate base register **TRS_CTRL** described in section 4.2.1.2. This max. transmission bitrate is divided down to obtain the transmission frequency for each link. The division factor can be programmed separately for each link via the Channel Link SpaceWire Mode Register **CH_x_DSM_MODR** register described in section 4.2.2.1. This arrangement allows each link to be run at different transmission speeds, as shown in the table below (shown for minimum and maximum link speeds):

Max. Transmit Speed	Link Speed Divider	Link Speed [Mbit/s]
80 (TRS_CTRL = 0x08)	1	80
	2	40
	4	20
	8	10
	16	5
	32	2,5
	64	1,25 (min. Link speed)
200 (TRS_CTRL = 0x14)	1	200 (max. Link speed)
	2	100
	4	50
	8	25
	16	12,5
	32	6,25
	64	3,125

3.4 Errors on links

Link inputs can detect parity and disconnection conditions as errors. A single bit odd parity system will detect single bit errors at the character level. The protocol to transmit NULL's in the absence of other characters enables disconnection of a link to be detected. A disconnection error indicates that:

• the link has been physically disconnected;

• an error has occurred on the link or at the other end of the link, which may have then stopped transmitting.

The status bits in the **CH_x_DSM_STAR** registers flags that a parity, credit, sequence (ESC) and/or disconnect error has occurred on the link. The errors can be detected independently. When a SpaceWire channel detects a parity, credit or sequence error it halts its output. This is detected as a disconnect error at the other end of the link, causing this to halt its output also. Detection of an error causes the link to be stopped. Thus, the disconnect behavior ensures that both ends are stopped. Each end can then be restarted.

The following procedure can be used to restart a link following an error (routing is assumed disabled):

1)   Disconnect is detected (cause: error)

2)   SMCS332SpW enters "exchange of silence"

> if the corresponding interrupt mask bit is enabled, the SMCS332SpW HINTR* signal will be asserted.

3)   The transmit section is reset by the SMCS332SpW after an error has occurred. Any currently on-going transmit transfer via COMI is stopped.

> The user can decide to read the receive-FIFO (if not empty) and then to reset the receive section or to reset the receive section immediately without reading the receive-FIFO.

> Resetting the receive section is done by setting and clearing bit1 in register CHx_CNTRL2.

> If the protocol mode is used the protocol unit should be reset by setting and clearing bit2 in register CHx_CNTRL2.

4)   If the auto start bit (bit2, CHx_DSM_CMDR) is set, the link will start automatically after the "protocol of silence", otherwise the user has to start the link by setting bit1 in CHx_DSM_CMDR.

> The configuration registers are not affected by the disconnect error.

5)   New transmit and receive transfers can be started.

3.5 SpaceWire state on start up

After power-on all LinkData and LinkStrobe signals are low, without clocks. Following power-on reset an initialization sequence sets the speed of the link clock. The DS-Links are initially inactive. They are configured and started by configuration writes. Their status can be determined by configuration reads.

Each link must be explicitly started by writing to the Start Transmit Node bit in its CHx_DSM_CMDR register. When a SpaceWire link is started up it transmits NULL's. Data may not be transferred over the link until the receiving link has sent a FCT, which it will do as soon as it has been started. In remote mode (control by link) however, the SMCS332SpW will send characters as soon as it receives a NULL on one of the three links, the control link.

The receiving link receives and correctly decodes the characters. However, only when the receiving link has been explicitly started by writing across the (internal) configuration bus can it send characters back. NULL's are then sent until data is required.

A start up sequence between SpaceWire devices cannot be defined in general. The start up depends heavily on the system (user) requirements and consequently on the procedure defined at system level to initialize the system or to recover from an error.

No master-slave arrangement is necessary. The SpaceWire links are "hot plug" able., which means SpaceWire links can be connected together at any time.

3.6 Link connections

SpaceWire links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 200 millimeters. For longer distances a matched 100 ohm transmission line should be used together with differential link transceivers (LVDS). The inputs and outputs have been designed to have minimum skew at the 1.5 V TTL threshold. Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

For more information see [AD1].

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 18 of 131

4   Register Set

This chapter describes the SMCS332SpW registers which can be read or written by the HOCI or via the link (in case the "control by link" is enabled) to control SMCS332SpW operations. All SMCS332SpW control operations are performed by writes or reads of the respective registers. Most of the control operations are obvious from the content of the registers.

General Conventions:

-           bit 0 (D0) = least significant bit,

-           bit 7 (D7) = most significant bit (or bit 15 or bit 31)

-           Dx:0 means data bit x until bit 0.

4.1   Register address map

The addresses of the SMCS332SpW registers are directly mapped with pins HADR7 - 0. The tables below shows the addresses of all the SMCS332SpW registers depending on the HOCI port width.

4.1.1   SMCS332SpW status and control registers

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	16	8				
00	00	00	SICR	SMCS332SpW Interface Control Register	00	r / w
01	01	01	TRS_CTRL	Transmit-Speed-Base Register	0A	r / w
02	02	02	RT_CTRL	Routing Enable / Status Register	00	r / w
03	03	03		reserved	00	
04	04	04	ISR	Interrupt Status Register	04010040	ro
		05				
	06	06				
		07				
08	08	08	IMR	Interrupt Mask Register	00000000	r / w
		09				
	0A	0A				
		0B				
0C	0C	0C	COMI_CS0R	COMI Chip Select 0 upper address boundary Register	FF	r / w
0D	0D	0D		reserved	00	
0E	0E	0E	COMI_ACR	COMI Arbitration Control Register	08	r / w
0F	0F	0F	PRCIR	PRCI Register	00	r / w

4.1.2   SMCS332SpW channel 1 status and control registers

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	**16**	**8**				
10	10	10	CH1_DSM_MODR	channel 1 DSM mode Register	00	r / w
11	11	11	CH1_DSM_CMDR	channel 1 DSM command Register	00	r / w
12	12	12	CH1_DSM_STAR	channel 1 DSM status Register	00	r / w
13	13	13	CH1_DSM_TSTR	channel 1 DSM test Register	00	r / w
14	14	14	CH1_ADDR	channel 1 address Register	00	r / w
15	15	15	CH1_RT_ADDR	channel 1 Route Address Register	00	r / w
16	16	16	CH1_PR_STAR	channel 1 Protocol Status Register	04	r / w
17	17	17		reserved	00	ro
18	18	18	CH1_CNTRL1	channel 1 control Register 1	00	r / w
19	19	19	CH1_CNTRL2	channel 1 control Register 2	00	r / w
1A	1A	1A	CH1_HTID	channel 1 Header Transaction ID byte	00	ro
1B	1B	1B	CH1_HCNTRL	channel 1 Header control byte	00	ro
1C	1C	1C	CH1_ESR1	channel 1 detailed error source register 1	00	r / w
1D	1D	1D	CH1_ESR2	channel 1 detailed error source register 2	00	r / w
1E	1E	1E		reserved	00	ro
1F	1F	1F	CH1_COMICFG	channel 1 COMI configuration register	00	r / w
20	20	20 21	CH1_TX_SAR	channel 1 transmit Start Address Register	0000	r / w
22	22	22 23	CH1_TX_EAR	channel 1 transmit End Address Register	0000	r / w
24	24	24 25	CH1_TX_CAR	channel 1 transmit Current Address Register	0000	ro
26	26	26	CH1_TX_FIFO	channel 1 transmit FIFO	--	wo
27	27	27	CH1_TX_EOPB	channel 1 transmit EOP Bit Register	--	wo
28	28	28 29	CH1_RX_SAR	channel 1 receive Start Address Register	0000	r / w
2A	2A	2A 2B	CH1_RX_EAR	channel 1 receive End Address Register	0000	r / w
2C	2C	2C 2D	CH1_RX_CAR	channel 1 receive Current Address Register	0000	ro
2E	2E	2E	CH1_RX_FIFO	channel 1 receive FIFO	xxxxxxxx	ro
2F	2F	2F	CH1_STAR	channel 1 Status Register	01	ro

4.1.3 SMCS332SpW channel 2 status and control registers

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	16	8				
30	30	30	CH2_DSM_MODR	channel 2 DSM mode Register	00	r / w
31	31	31	CH2_DSM_CMDR	channel 2 DSM command Register	00	r / w
32	32	32	CH2_DSM_STAR	channel 2 DSM status Register	00	r / w
33	33	33	CH2_DSM_TSTR	channel 2 DSM test Register	00	r / w
34	34	34	CH2_ADDR	channel 2 address Register	00	r / w
35	35	35	CH2_RT_ADDR	channel 2 Route Address Register	00	r / w
36	36	36	CH2_PR_STAR	channel 2 Protocol Status Register	04	r / w
37	37	37		reserved	00	ro
38	38	38	CH2_CNTRL1	channel 2 control Register 1	00	r / w
39	39	39	CH2_CNTRL2	channel 2 control Register 2	00	r / w
3A	3A	3A	CH2_HTID	channel 2 Header Transaction ID byte	00	ro
3B	3B	3B	CH2_HCNTRL	channel 2 Header control byte	00	ro
3C	3C	3C	CH2_ESR1	channel 2 detailed error source register 1	00	r / w
3D	3D	3D	CH2_ESR2	channel 2 detailed error source register 2	00	r / w
3E	3E	3E		reserved	00	ro
3F	3F	3F	CH2_COMICFG	channel 2 COMI configuration register	00	r / w
40	40	40 41	CH2_TX_SAR	channel 2 transmit Start Address Register	00	r / w
42	42	42 43	CH2_TX_EAR	channel 2 transmit End Address Register	00	r / w
44	44	44 45	CH2_TX_CAR	channel 2 transmit Current Address Register	00	ro
46	46	46	CH2_TX_FIFO	channel 2 transmit FIFO	00	wo
47	47	47	CH2_TX_EOPB	channel 2 transmit EOP Bit Register	00	wo
48	48	48 49	CH2_RX_SAR	channel 2 receive Start Address Register	00	r / w
4A	4A	4A 4B	CH2_RX_EAR	channel 2 receive End Address Register	00	r / w
4C	4C	4C 4D	CH2_RX_CAR	channel 2 receive Current Address Register	00	ro
4E	4E	4E	CH2_RX_FIFO	channel 2 receive FIFO	xxxxxxxx	ro
4F	4F	4F	CH2_STAR	channel 2 Status Register	01	ro

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 21 of 131

4.1.4   SMCS332SpW channel 3 status and control registers

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	16	8				
50	50	50	CH3_DSM_MODR	channel 3 DSM mode Register	00	r / w
51	51	51	CH3_DSM_CMDR	channel 3 DSM command Register	00	r / w
52	52	52	CH3_DSM_STAR	channel 3 DSM status Register	00	r / w
53	53	53	CH3_DSM_TSTR	channel 3 DSM test Register	00	r / w
54	54	54	CH3_ADDR	channel 3 address Register	00	r / w
55	55	55	CH3_RT_ADDR	channel 3 Route address Register	00	r / w
56	56	56	CH3__PR_STAR	channel 3 Protocol Status Register	04	r / w
57	57	57		reserved	00	ro
58	58	58	CH3_CNTRL1	channel 3 control Register 1	00	r / w
59	59	59	CH3_CNTRL2	channel 3 control Register 2	00	r / w
5A	5A	5A	CH3_HTID	channel 3 Header Transaction ID byte	00	ro
5B	5B	5B	CH3_HCNTRL	channel 3 Header control byte	00	ro
5C	5C	5C	CH3_ESR1	channel 3 detailed error source register 1	00	r / w
5D	5D	5D	CH3_ESR2	channel 3 detailed error source register 2	00	r / w
5E	5E	5E		reserved	00	ro
5F	5F	5F	CH3_COMICFG	channel 3 COMI configuration register	00	r / w
60	60	60 61	CH3_TX_SAR	channel 3 transmit Start Address Register	00	r / w
62	62	62 63	CH3_TX_EAR	channel 3 transmit End Address Register	00	r / w
64	64	64 65	CH3_TX_CAR	channel 3 transmit Current Address Register	00	ro
66	66	66	CH3_TX_FIFO	channel 3 transmit FIFO	00	wo
67	67	67	CH3_TX_EOPB	channel 3 transmit EOP Bit Register	00	wo
68	68	68 69	CH3_RX_SAR	channel 3 receive Start Address Register	00	r / w
6A	6A	6A 6B	CH3_RX_EAR	channel 3 receive End Address Register	00	r / w
6C	6C	6C 6D	CH3_RX_CAR	channel 3 receive Current Address Register	00	ro
6E	6E	6E	CH3_RX_FIFO	channel 3 receive FIFO	xxxxxxxx	ro
6F	6F	6F	CH3_STAR	channel 3 Status Register	01	ro

4.1.5 SMCS332SpW GPIO control registers

These registers are only enabled when the SMCS332SpW is configured for "control by link" using the 'BOOTLINK' pin (see sections 2.3, 5.4 and 9.2).

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	16	8				
		70	GPIO_DIR0	GPIO direction register 0	00	r / w
		71	GPIO_DIR1	GPIO direction register 1	00	r / w
		72	GPIO_DIR2	GPIO direction register 2	00	r / w
		73	GPIO_DIR3	GPIO direction register 3	00	r / w
		74	GPIO_DATA0	GPIO data register 0	00	r / w
		75	GPIO_DATA1	GPIO data register 1	00	r / w
		76	GPIO_DATA2	GPIO data register 2	00	r / w
		77	GPIO_DATA3	GPIO data register 3	00	r / w

4.1.6 Time code control registers

These registers controls the transmit and receive of the time code.

Port Width / Address (hex)			Register	Function	Reset Value (hex)	Access
32	16	8				
78	78	78	TIME_CNTRL	time code control register	00	r / w
79	79	79	TIME_CODE	time code value register	00	r / w

4.2   Register Description

4.2.1   General SMCS Registers

4.2.1.1   SMCS332SpW Interface Control Register (SICR)

- address: 0x00
- data width: 3 bit, D2:0
- access mode: read / write
- reset value: 0x00

Bit	Description
1:0	00 = Host Control Interface Data Port operates as 8 bit port (RESET) 01 = HOCI data port operates as 16 bit port 1X = HOCI data port operates as 32 bit port refer also to little/big endian mode of HOCI data port, which is hardware controlled by input level at the HOSTBIGE pin. Signal level of HOSTBIGE pin: 0: little endian 1: big endian
2	0 = COMI operates in little endian mode (RESET) 1 = COMI operates in big endian mode
7:3	always '0' / reserved

4.2.1.2   Transmit bitrate base Register (TRS_CTRL)

- address: 0x01
- data width: 5 bit, D4:0
- access mode: read / write
- reset value: 0x0A

Bit	Description
4:0	Multiplier value (0x0A after reset), (LSB setting will be ignored) Value(hex): max. bitrate: 0x08 80 Mbit/s 0x0A 100 Mbit/s 0x0C 120 Mbit/s (not possible if VCC= 3.3 Volt) 0x0E 140 Mbit/s (not possible if VCC= 3.3 Volt) 0x10 160 Mbit/s (not possible if VCC= 3.3 Volt) 0x12 180 Mbit/s (not possible if VCC= 3.3 Volt) 0x14 200 Mbit/s (not possible if VCC= 3.3 Volt) 0x00- 0x07 not possible.
7:5	always '0' / reserved

The max. transmit bitrate (in Mbit/s) of all 3 channels is the result of the multiplication between the input frequency at the CLK10 pin and the multiplier value (MUL). Select the MUL value only to one of the above values. A write of a new value in the transmit bitrate base register should only be taken when all links are running with 10 MBit/s. After the write of the new value, the PLL needs 10 us to run on the new frequency. After this time, the links can run with the new transmit bit rate.

Example: (CLK10 Frequency = 10 MHz)

max. transmit bitrate = CLK_10 * MUL
160 MBit/s = 10 * 0x10

4.2.1.3   Route Control Status Register (RT_CTRL)

- address: 0x02
- data width: 8 bit, D7:0
- access mode: D6:0: read only
                               D7: read / write
- reset value: 0x00

Bit	Description
1:0	Status channel 1 00: no connection 01: connection to local fifo 10: connection to channel 2 11: connection to channel 3
3:2	Status channel 2 00: no connection 01: connection to channel 1 10: connection to local fifo 11: connection to channel 3
5:4	Status channel 3 00: no connection 01: connection to channel 1 10: connection to channel 2 11: connection to local fifo
6	reserved
7	SMCS332SpW Wormhole Route Enable Bit: 0: Wormhole Routing disabled (Reset) 1: Wormhole Routing enabled

4.2.1.4   Interrupt Status Register (ISR)

- address: 0x04 – 0x07
- data width: 32 bit, D31:0
- access mode: read only
- reset value: 0x04010040

The following interrupts are stored in byte 0 of the ISR:
- address:
              8 bit mode: 0x04
              16 bit mode: 0x04
              32 bit mode: 0x04

ISR Byte 0

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 25 of 131

Bit	Description
0	channel 1: SpaceWire parity, disconnect, ESC or credit error. For more information refer to register CH1_DSM_STAR: channel SpaceWire status register
1	channel 1: error For more information over this error flag refer to register CH1_ESR1 and CH1_EXR2
2	channel 1: data from the communication memory are read. That means: channel1 transmit COMI address generator reach the value of the end address register CH1_TX_EAR
3	channel1: the received data from channel 1 is transmitted in the communication memory That means: channel1 receive COMI address generator reach the value of the end address register CH1_RX_EAR
4	channel 1: the EOP/EEP character was received. For more information refer to register CH1_STAR bit 4 and 5
5	channel 1: reset channel 1 command received (only if protocol mode enabled)
6	channel 1: transmit fifo is empty (set after reset)
7	channel 1: transmit fifo is full

The following interrupts are stored in byte 1 of the ISR:
- address:

8 bit mode:	0x05
16 bit mode:	0x04
32 bit mode:	0x04

ISR Byte 1

Bit	Description
0	channel 1: receive fifo is not empty
1	channel 1: receive fifo is full
2	channel 2: SpaceWire parity, disconnect, ESC or credit error. For more information refer to register CH2_DSM_STAR: channel 2 SpaceWire status register
3	channel 2: error For more information over this error flag refer to register CH2_ESR1 and CH2_EXR2
4	channel 2: data from the communication memory are read. That means: channel2 transmit COMI address generator reach the value of the end address register CH2_TX_EAR
5	channel2: the received data from channel 2 is transmitted in the communication memory That means: channel2 receive COMI address generator reach the value of the end address register CH2_RX_EAR
6	channel 2: the EOP/EEP character was received. For more information refer to register CH2_STAR bit 4 and 5
7	channel 2: reset channel 2 command received (only if protocol mode enabled)

The following interrupts are stored in byte 2 of the ISR:

- address:

8 bit mode:	0x06
16 bit mode:	0x06
32 bit mode:	0x04

ISR Byte 2

Bit	Description
0	channel 2: transmit fifo is empty (set after reset)
1	channel 2: transmit fifo is full
2	channel 2: receive fifo is not empty
3	channel 2: receive fifo is full
4	channel 3: SpaceWire parity, disconnect, ESC or credit error. For more information refer to register CH3_DSM_STAR: channel 3 SpaceWire status register
5	channel 3: error For more information over this error flag refer to register CH3_ESR1 and CH3_EXR2
6	channel 3: data from the communication memory are read. That means: channel3 transmit COMI address generator reach the value of the end address register CH3_TX_EAR
7	channel3: the received data from channel 3 is transmitted in the communication memory That means: channel3 receive COMI address generator reach the value of the end address register CH3_RX_EAR

The following interrupts are stored in byte 2 of the ISR:

- address:

8 bit mode:	0x07
16 bit mode:	0x06
32 bit mode:	0x04

ISR Byte 3

Bit	Description
0	channel 3: the EOP/EEP character was received. For more information refer to register CH3_STAR bit 4 and 5
1	channel 3: reset channel 3 command received (only if protocol mode enabled)
2	channel 3: transmit fifo is empty (set after reset)
3	channel 3: transmit fifo is full
4	channel 3: receive fifo is not empty
5	channel 3: receive fifo is full
6	TICK IN received interrupt
7	always '0' / reserved

To read the ISR when the HOCI is in 8/16 mode the following steps are required:

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 27 of 131

1.   read always all bytes

2.   read first byte 0, last byte 3

3.   when Byte 3 is read, the signal HINTR* will be deactivated for a minimum of two CLK cycles and all bits in the ISR will be cleared.

Note that the ISR will be latched when reading Byte 0. Only the interrupts flagged at that time will be reset when reading Byte 3. This makes sure that no interrupts will be lost that happen to be raised in the time between reading Byte 0 and Byte 3.

4.2.1.5   Interrupt Mask Register (IMR)

- address: 0x08 - 0x0B
- data width: 32 bit, D31:0
- access mode: read / write
- reset value: 0x00000000

All interrupts are masked after reset. A '1' written to the IMR enables the corresponding interrupt source in register ISR to activate the interrupt signal HINTR*.

4.2.1.6   COMI Chip Select0 Boundary Register (COMI_CS0R)

- address: 0x0C
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0xFF

The communication memory address space is divided into two banks. Each bank has a separate memory select pin CMCS0* and CMCS1*.

The upper COMI address signals CMADR15 - 8 are compared with the value of COMI_CS0R. Bank 0 starts at address 0x0000 until [COMI_CS0R * 256 + 255].

IF 0x0000 <= CMADR <= (COMI_CS0R * 0x100) + 0xFF THEN
        CMCS0* is active ELSE
        CMCS1* is active.

Example: COMI_CS0R = 0x3F.

Address 0x0000 -0x3FFF => CMCS0* is active.

Address 0x4000 -0xFFFF => CMCS1* is active

4.2.1.7   COMI Arbitration Control Register (COMI_ACR)

- address: 0x0E
- data width: 4 bit, D3:0
- access mode: read / write
- reset value: 0x08

Signal COCI is an input pin that requests access from the other SMCS332SpW to the communication memory bus.

When COCI is asserted, the SMCS332SpW COMI interface completes the current access, places the interface in a high-impedance state and then deasserts the output signal COCO to indicate to the requesting SMCS332SpW that it is no longer driving the COMI bus.

After [COMI_ACR - 1] CLK cycles and when the SMCS332SpW COMI needs the bus for further access to the communication memory, the SMCS332SpW asserts the COCO pin (connected with the COCI pin of the other SMCS332SpW) .

COMI_ACR = wait time between two accesses.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 28 of 131

COMI_ACR values from
>   0x00 = disable communication memory interface
>   0x01 = is not allowed
> to 0x0F = max. CLK cycles between two accesses.

See also the description for pins CAM, COCI, COCO and chapter 5.3 COMI arbitration.

4.2.1.8   PRCI Register (PRCIR)

>   - address: 0x0F
>   - data width: 5 bit, D4:0
>   - access mode: read / write
>   - reset value: 0x00

Gives the status of the protocol command interface signals CPUR* and SES0 - 3*.

Bit	Description
0	Status of signal CPUR* 0 = Signal CPUR* is inactive high 1 = Signal CPUR* is active low
1	status of signal SES0* 0 = Signal SES0* is inactive high 1 = Signal SES0* is active low
2	status of signal SES1* 0 = Signal SES1* is inactive high 1 = Signal SES1* is active low
3	status of signal SES2* 0 = Signal SES2* is inactive high 1 = Signal SES2* is active low
4	status of signal SES3* 0 = Signal SES3* is inactive high 1 = Signal SES3* is active low
7:5	always '0' / reserved

See also the Simple Interprocessor Communication Protocol Specification chapter 13

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 29 of 131

4.2.2 Channel 1 Registers

4.2.2.1 Channel 1 SpaceWire Mode Register (CH1_DSM_MODR)

- address: 0x10
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0x00

Bit	Description
2:0	Transmit bitrate selection (only if bit 3 set) 000 : transmit bitrate = max. transmit bitrate 1/1 001 : transmit bitrate = max. transmit bitrate 1/2 010 : transmit bitrate = max. transmit bitrate 1/4 011 : transmit bitrate = max. transmit bitrate 1/8 100 : transmit bitrate = max. transmit bitrate 1/16 101 : transmit bitrate = max. transmit bitrate 1/32 110 : transmit bitrate = max. transmit bitrate 1/64 111 : reserved see also bit 3 and register TRS_CTRL (Address 0x01)
3	base frequency selection 0 = transmit bitrate = 10 Mbit/s = frequency of signal CLK10 (after reset). 1 = for transmit bitrate see bits 2:0
4	always '0' / reserved
5	always '0'/reserved
6	always '0'/reserved
7	test mode if set, enables access to test register CH1_DSM_TSTR

Note: When reprogramming the link speed, it is recommended to wait at least 800 ns between two subsequent changes of link speed. It is also recommended to program only single steps; i.e. when switching from 1/1 to 1/32 link speed, it should be done in steps from 1/1 to 1/2, 1/4, 1/8, 1/16 and then 1/32, each with a 800 ns pause between two steps.

This register will keep its contents also when TRS_CTRL is modified.

4.2.2.2 Channel 1 SpaceWire Command Register (CH1_DSM_CMDR)

- address: 0x11
- data width: 4 bit, D3:0
- access mode: read / write
- reset value: 0x00

Bit	Description
0	Stop SpaceWire link if set, the SpaceWire cell goes in the "Error Reset" state. See [AD1] This bit is autoreset by itself.
1	Start SpaceWire link Starts the transmission of NULL's.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 30 of 131

Bit	Description
	This bit is autoreset by itself.
	This bit have to be set after an error (disconnect) to restart the SpaceWire link or set also Bit 2 to automatically restart the link after an error.
2	If set: automatically restart of the link after an error (disconnect, parity, ESC or credit error).
7:3	always '0' / reserved

4.2.2.3 Channel 1 Link SpaceWire Status Register (CH1_DSM_STAR)

- address: 0x12
- data width: 8 bit, D6:0
- access mode: read only
- reset value: 0x00

The status register is readable and bits D1, D2, D4, D5 and D6 are reset by reading the register.

Bit	Description
0	transmit link is running
1	disconnect error
2	parity error
3	is set when a NULL code is received
4	is set when a FCT control character is received
5	ESC character sequence error (the link received a single ESC control character, not a NULL code or time code)
6	credit error (the link received to much normal characters or FCT control characters)
7	always '0'/reserved

Differences between the SMCS332 and the SMCS332SpW for the bits D0, D3 and D4:

When D0, D3 and D4 are set, the SpaceWire link is/was in the "Run" state. See [AD1].

On an Space Wire link error (i.e. disconnect, parity,…) these bits will be cleared.

However, if the running link is stopped the bits D0, D3 and D4 are not reset.

Therefore, on a new start of the link these bits do not reflect the current situation of the link.

For this reason it is necessary to read the register before the start of the link.

Then, only bit 4 shall be checked whether the link is in the "Run" state or not..

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 31 of 131

4.2.2.4 Channel 1 SpaceWire Test Register (CH1_DSM_TSTR)

- address: 0x13
- data width: 4 bit, D3:0
- access mode: read / write
- reset value: 0x0

The test register is writeable only when bit 7 of register CH1_DSM_MODR is set. Once this bit has been set it remains so even when test mode is exited.

Bit	Description
0	enables internal feedback of transmit link to receive link (data will also be transmitted externally)
1	disable disconnect detection
2	link input mute
3	insert wrong parity If set, inverts the transmitted parity. The wrong parity bit is used to invert the parity generation bit and so invoke a parity error at the other end of the link. The other end of the link detects an error and stops the transmission (= disconnect).
4	link output mute
5	send EEP character instead of an EOP character
7:6	always '0'/reserved

The input mute bit holds the D and S inputs on the reset value '0'
The output mute bit holds the D and S outputs on the reset value '0'.

4.2.2.5 Channel 1 Address Register (CH1_ADDR)

- address: 0x14
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0x00

In protocol mode with enabled check destination bit:
The SpaceWire cell compares the first byte (destination address byte) of the received data packet with the value of register CH1_ADDR. If it does not match, an error condition occurs (see CH1_ESR1).

In wormhole routing mode:
The SpaceWire cell compares the first byte (destination address byte) of the received data packet with the value of register CH1_ADDR or with the value of register CHx_RT_ADDR from the other two SpaceWire channels. If it does not match, an error condition occurs (see CH1_ESR1). For more information see chapter 5.5

Bit	Description
7:0	Address of channel 1

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 32 of 131

4.2.2.6   Channel 1 Route Address Register (CH1_RT_ADDR)

- address: 0x15
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0x00

In wormhole routing mode:

The content of the channel 1 route address register builds the routing address of channel 1 and will be compared with the first byte of an incoming packet in the two other SpaceWire cells. The result of the comparison is used to determine the destination of the packet. It can be delivered to the internal channel FIFO or to one of the two other SpaceWire channels inside the SMCS332SpW. For more information see chapter 5.5

Bit	Description
7:0	register content will be compared with first byte of an incoming packet (if routing is enabled)

4.2.2.7   Channel 1 Protocol Status Register (CH1_PR_STAR)

- address: 0x16
- data width: 4 bit, D3:0
- access mode: read only
- reset value: 0x00

Shows status of internal protocol unit signals.

Bit	Description
0	only when protocol mode enabled: when set, execution of critical commands is enabled
1	only when protocol mode enabled: when set, acknowledge packet is received
2	only when bit 3 of register CH1_CNTRL1 is enabled protocol unit acknowledge FIFO is empty (set after reset)
3	only when bit 3 of register CH1_CNTRL1 is enabled protocol unit acknowledge FIFO is full
7:4	always '0' / reserved

4.2.2.8   Channel 1 Control Register 1 (CH1_CNTRL1)

- address: 0x18
- data width: 6 bit, D5:0
- access mode: read / write
- reset value: 0x00

Bit	Description
1:0	data transfer mode of channel 1 00: transparent mode (reset), this mode allows complete transparent data transfer between two nodes. 01: reserved 10: simple interprocessor communication protocol mode (known as protocol mode), this mode executes the simple interprocessor communication protocol as described in the protocol specification. 11: reserved
2	only when protocol mode enabled: check destination byte enabled compare first byte (header: destination address byte) of the received data packet with register CH1_ADDR
3	only when protocol mode enabled: send acknowledge packet enabled when set, the protocol unit sends acknowledge packets see also bit 2 and 3 of register CH1_PR_STAR
4	checksum enabled when set, the channel sends two checksum bytes at the end of the transmitted packet and checks the last two bytes. See Note.
5	header field control bit when set, the SMCS332SpW use the first byte of a packet as number of bytes which are transmitted as header bytes. If checksum enabled, this bytes are excluded from the checksum. The range for the header field is from minimum 2bytes (number byte + header byte) to maximum16 bytes. The size of the header field is 4, 8, 12 and 16 bytes. This means, that the data field starts at the next modulo 4 bytes. The rest of a 4-byte block which is not covered by the number of header bytes will not be transmitted. See for additional information chapter 5.6.
7:6	always '0' / reserved

Note: The Checksum generation and check module adds all bytes in the packet:

$$checksum[16:0] = checksum[16:0] + data[7:0] + checksum[16]$$

The checksum is generated in the SMCS332SpW as shown in the following figure.
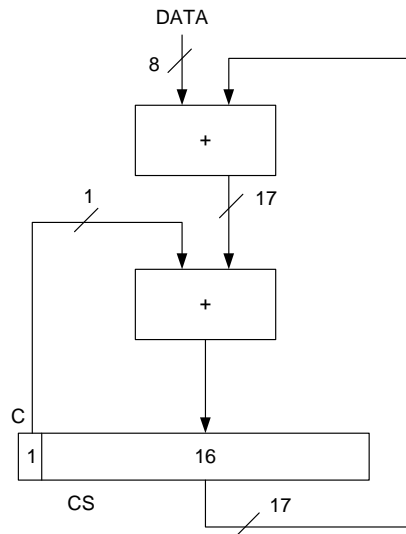
Figure Checksum generation

Example: the packet consists of 259 * 0xFF:

1. checksum	= 0x00000 + 0xFF + 0	= 0x000FF
2. checksum	= 0x000FF + 0xFF + 0	= 0x001FE
…		
256. checksum	= 0x0FE01 + 0xFF + 0	= 0x0FF00
257. checksum	= 0x0FF00 + 0xFF + 0	= 0x0FFFF
258. checksum	= 0x0FFFF + 0xFF + 0	= 0x100FE
259. checksum	= 0x100FE + 0xFF + 1	= 0x101FE

1. checksum byte send over SpaceWire = checksum[7:0]

2. checksum byte send over SpaceWire = checksum[15:8]

See also register CHx_ESR1 (0x1C) bit D4 (checksum error).

4.2.2.9   Channel 1 Control Register 2 (CH1_CNTRL2)

- address: 0x19
- data width: 3 bit, D2:0
- access mode: read / write
- reset value: 0x00

Bit	Description
0	reset/delete channel1 transmit and acknowledge section
1	reset/delete channel1 receive section
2	reset protocol unit
7:3	always '0' / reserved

Note: The above bits must be specifically set/cleared as required. There is no automatic reset.
See also the procedure following disconnect described in chapter 3.4.

4.2.2.10   Channel 1 Header Transaction ID (CH1_HTID)

     - address: 0x1A
     - data width: 8 bit, D7:0
     - access mode: read only
     - reset value: 0x00

When protocol mode enabled. If an error occurs, this register shows the third byte (transaction ID) of the received acknowledge packet.

Bit	Description
7:0	transaction ID of the received packet

4.2.2.11   Channel 1 Header Control Byte (CH1_HCNTRL)

     - address: 0x1B
     - data width: 8 bit, D7:0
     - access mode: read only
     - reset value: 0x00

When protocol mode is enabled. If an error occurs, this register shows the second byte (header control byte) of the received acknowledge packet.

Bit	Description
7:0	header control byte of the received packet

4.2.2.12   Channel 1 Error Source Register 1 (CH1_ESR1)

     - address: 0x1C
     - data width: 6 bit, D5:0
     - access mode: read / write
     - reset value: 0x00

The CH1_ESR1 and CH1_ESR2 registers give detailed information about the source of the channel 1 error interrupt (bit 1/byte 0 of the interrupt status register ISR).

The channel 1 error source register 1 is readable and the bits are reset only by writing '1', thus ensuring status never gets missed by software.

Bit	Description
0	in protocol mode: received acknowledge packet has a wrong control byte
1	in protocol mode: received acknowledge packet has a wrong destination byte (not equal with register CH1_ADDR)
2	in protocol mode: received acknowledge packet has a wrong transaction ID
3	in protocol mode:

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 36 of 131

Bit	Description
	received acknowledge packet has a wrong checksum (only, when generate checksum enabled, bit 4 of register CH1_CNTRL1).
4	in transparent mode: checksum error (if checksum enabled)
5	in transparent mode: wrong route - destination byte (if routing enabled) (destination byte not equal with CH1_ADDR or CH2_RT_ADDR or CH3_RT_ADDR)
7:6	always '0' / reserved

4.2.2.13   Channel 1 Error Source Register 2 (CH1_ESR2)

- address: 0x1D
- data width: 7 bit, D6:0
- access mode: read / write
- reset value: 0x00

Only in protocol mode:

The CH1_ESR2 register give detailed information about the source of the channel 1error interrupt (bit 1/byte 0 of the interrupt status register ISR).

The channel 1 error source register 2 is readable and the bits are reset only by writing '1', thus ensuring status never gets missed by software.

Bit	Description
0	Simple Interprocessor Command (SIC) sequence error. That means: the SMCS332SpW received an unexpected acknowledge packet.
1	Received SIC packet with a wrong checksum (only, when generate checksum enabled, bit 4 of register CH1_CNTRL1).
2	received SIC packet has a wrong control byte
3	received SIC packet has a wrong destination byte (not equal with register CH1_ADDR)
4	the received SIC command/control byte was not enabled (see also bit 0 in register CH1_PR_STAR)
5	received data packet has a wrong destination byte (not equal with register CH1_ADDR)
6	received data packet has a wrong checksum (generate checksum enabled, bit4 of register CH1_CNTRL1)
7	always '0' / reserved

4.2.2.14   Channel 1 COMI Configuration Register (CH1_COMICFG)

- address: 0x1F
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0x00

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 37 of 131

Bit	Description
1:0	transmit data port width 00: the COMI data port operates in transmit direction with 8 bit (after reset), each access of the COMI data port reads 8 bit from the communication memory 01: the COMI data port works with 16 bit 1X: the COMI data port works with 32 bit for little/big endian access of the COMI data port see bit 2 of register SICR
2	0: send EOP control character at the end of the packet (when bit 3 is not set), which was read from the communication memory 1: send also EOP control character at the end of the packet (when bit 3 is not set). see [RD1] Refer also to registers CH1_TXSAR and CH1_TXEAR.
3	0: send EOP control character at the end of the packet. 1 send NO EOP control character at the end of the packet.
5:4	receive data port width 00: the COMI data port operates in receive direction with 8 bit (after reset), each access writes 8 bit to the memory, bit 31 - 8 set to '0'. (see also bit 7) 01: the COMI data port works with 16 bit, (bit 31 - 16 set to '0') 1X: the COMI data port works with 32 bit for little/big endian access of the COMI data port see bit 2 of register SICR
6	no stop because of a received EOP/EEP 0: if channel 1 receive the EOP/EEP, the COMI receive address generator stops, no further data will be written in the communication memory. Only one packet will be received. 1: if channel 1 receive the EOP/EEP, the COMI receive address generator works until CH1_RX_EAR = current COMI address. More than one packet will be received.
7	expand sign bit (only in 8/16 bit mode of the receive channel 1, bit 5:4 = 0X) Little endian: 0: in 8 bit mode, set data bits 31 - 8 of the COMI to '0'. in 16 bit mode, set data bits 31 - 16 to '0' 1: in 8 bit mode: if MSB of the received data = '1' set data bits 31 - 8 of the COMI to '1' if MSB of the received data = '0' set data bits 31 - 8 of the COMI to '0' in 16 bit mode: if MSB of the received data = '1' set data bits 31 - 16 of the COMI to '1' if MSB of the received data = '0' set data bits 31 - 16 of the COMI to '0' Big endian: 0: in 8 bit mode, set data bits 24 - 0 of the COMI to '0'. in 16 bit mode, set data bits 15 - 0 to '0' 1: in 8 bit mode: if MSB of the received data = '1' set data bits 24 - 0 of the COMI to '1' if MSB of the received data = '0' set data bits 24 - 0 of the COMI to '0' in 16 bit mode: if MSB of the received data = '1' set data bits 15 - 0 of the COMI to '1' if MSB of the received data = '0' set data bits 15 - 0 of the COMI to '0'

4.2.2.15 Channel 1 Transmit Start Address Register (CH1_TX_SAR)

- address: 0x20 - 0x21
- data width: 16 bit, D15:0
- access mode: read / write

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 38 of 131

SMCS332SpW
User Manual

- reset value: 0x0000

The value of CH1_TX_SAR shows the start address, the value of CH1_TX_EAR shows the end address of a packet in the communication memory, which should be transmitted over channel 1.

Byte 0 at address 0x20:

Bit	Description
7:0	lower byte of the start address

Byte 1 at address 0x21: (only in 8 bit mode of the HOCI data port)

Bit	Description
7:0	upper byte of the start address

4.2.2.16 Channel 1 Transmit End Address Register (CH1_TX_EAR)

- address: 0x22 - 0x23
- data width: 16 bit, D15:0
- access mode: read / write
- reset value: 0x0000

As soon as the upper byte is written, the address generator starts with the address in CH1_TX_SAR.

Byte 0 at address 0x22:

Bit	Description
7:0	lower byte of the end address

Byte 1 at address 0x23: (only in 8 bit mode of the HOCI data port):

Bit	Description
7:0	upper byte of the end address

4.2.2.17 Channel 1 Transmit Current Address Register (CH1_TX_CAR)

- address: 0x24 - 0x25
- data width: 16 bit, D15:0
- access mode: read only
- reset value: 0x0000

The CH1_TX_CAR register shows the current address of the address generator.

Byte 0 at address 0x24:

Bit	Description
7:0	lower byte of the current address

Byte 1 at address 0x25: (only in 8 bit mode of the HOCI data port):

Bit	Description
7:0	upper byte of the current address

4.2.2.18   Channel 1 Transmit FIFO (CH1_TX_FIFO)

- address: 0x26
- data width: 8/16/32 bit, D31:0
- access mode: write only
- reset value: n/a

Beside the communication memory the host processor can send data over the transmit FIFO interface.
The FIFO data width is equal to the HOCI data port width.
The FIFO has a size of 32 bytes. The host processor can control the data transmission with bits 6 and 7 of the interrupt status register (ISR) or with the status bits 0 and 1 of the channel 1 status register (CH1_STAR).
At the end of the packet the host processor has to send an EOP character with register CH1_TX_EOPB.

4.2.2.19   Channel 1 Transmit EOP Bit Register (CH1_TX_EOPB)

- address: 0x27
- data width: 2 bit, D1:0
- access mode: write only
- reset value: n/a

For data transfer over the transmit FIFO, the host processor has to send the EOP character at the end of the packet.
If the host processor sets bit 0 or 1 or both, the SMCS332SpW sends an EOP character at the end of the packet.

Bit	Description
0	send an EOP character
1	send an EOP character
7:2	always '0' / reserved

4.2.2.20   Channel 1 Receive Start Address Register (CH1_RX_SAR)

- address: 0x28 - 0x29
- data width: 16 bit, D15:0
- access mode: read / write
- reset value: 0x0000

The value of CH1_RX_SAR shows the start address of an area in the communication memory, where the received data shall be written.

Byte 0 at address 0x28:

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 40 of 131

Bit	Description
7:0	lower byte of the start address

Byte 1 at address 0x29: (only in 8 bit mode of the HOCI data port):

Bit	Description
7:0	upper byte of the start address

4.2.2.21   Channel 1 Receive End Address Register (CH1_RX_EAR)

- address: 0x2A - 0x2B
- data width: 16 bit, D15:0
- access mode: read/ write
- reset value: 0x0000

As soon as the upper byte is written, the address generator starts with the address in CH1_RX_SAR.
The register CH1_RX_SAR (start address) and CH1_RX_EAR (last address) defines an area in the communication memory.
The receive address generator stops when:

	bit 6 = 0 in register CH1_COMICFG:	EOP/EEP was recevied.
or	bit 6 = 1 in register CH1_COMICFG:	current address = CH1_RX_EAR **and** the last data was written.

Byte 0 at address 0x2A:

Bit	Description
7:0	lower byte of the end address

Byte 1 at address 0x2B: (only in 8 bit mode of the HOCI data port):

Bit	Description
7:0	upper byte of the end address

4.2.2.22   Channel 1 Receive Current Address Register (CH1_RX_CAR)

- address: 0x2C - 0x2D
- data width: 16 bit, D15:0
- access mode: read only
- reset value: 0x0000

The CH1_RX_CAR register shows the current address of the receive address generator. The current address is the next "free" address.

Byte 0 at address 0x2C:

Bit	Description
7:0	lower byte of the current address

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 41 of 131

Byte 1 at address 0x2D: (only in 8 bit mode of the HOCI data port):

Bit	Description
7:0	upper byte of the current address

4.2.2.23 Channel 1 Receive FIFO (CH1_RX_FIFO)

    - address: 0x2E
    - data width: 8/16/32 bit, D31:0
    - access mode: read only
    - reset value: 0xXXXXXXXX

Beside the communication memory interface the host processor can read data from the receive FIFO interface. The FIFO data width is equal to the HOCI data port width. The FIFO has a size of 32 bytes. The host processor can control the data transfer with byte 1/bits 0 and 1 of the interrupt status register (ISR) or with the status bits 2 and 3 of the channel 1 status register (CH1_STAR). Bit 4 or 5 of the channel 1 status register (CH1_STAR) signals, whether a EOP or EEP character was received.

4.2.2.24 Channel 1 Status Register (CH1_STAR)

    - address: 0x2F
    - data width: 6 bit, D5:0
    - access mode: read only
    - reset value: 0x01

Bit	Description
0	transmit FIFO is empty; not changed/reset by read
1	transmit FIFO is full; not changed/reset by read
2	receive FIFO is not empty; not changed/reset by read
3	receive FIFO is full; not changed/reset by read
4	EOP received - reset after read
5	EEP received - reset after read
7:6	always '0' / reserved

4.2.3 Channel 2 Registers

see 4.2.2 Channel 1 Register

4.2.4 Channel 3 Registers

see 4.2.2 Channel 1 Register

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 42 of 131

4.2.5 Time Code Registers

4.2.5.1 Time Code Control Register (TIME_CNTRL)

- address: 0x78
- data width: 8 bit, D7:0
- access mode: D4:0: read / write
                    D7:5: read only
- reset value: 0x00

Bit	Description
1:0	Interrupt control bits: 00 = No interrupt signal to the interrupt controller. 01 = enable the internal interrupt signal generation to the interrupt controller only for a correct received TIME CODE character received from the SpaceWire links. 10 = enable the internal interrupt signal generation to the interrupt controller for all received TIME CODE characters. 11 = enable the internal interrupt signal generation to the interrupt controller for a false TIME CODE character received from the SpaceWire links. see also interrupt bit in ISR Byte 3, bit 6.
2	Time code value register control bit: 0 = overwrite of the time code value register with a received time code is enabled. 1 = No overwrite of the time code value register with a received time code.
3	TIME_CODE_SYNC signal control bit0: 0 = The TIME_CODE_SYNC signal is disabled. 1 = A falling edge of the TIME_CODE_SYNC input signal sends the time code register value over the SpaceWire links.
4	TIME_CODE_SYNC signal control bit1: 0 = No increment of the time code value register 1 = A falling edge of the TIME_CODE_SYNC input signal increments the time code register value by 1.
5	Status: time code character received on SpaceWire link 1. Only read, reset after read.
6	Status: time code character received on SpaceWire link 2. Only read, reset after read.
7	Status: time code character received on SpaceWire link 3. Only read, reset after read.

4.2.5.2 Time Code Value Register (TIME_CODE)

- address: 0x79
- data width: 8 bit, D7:0
- access mode: read / write
- reset value: 0x00

Bit	Description
7:0	Time code value register:

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 43 of 131

Bit	Description
	After a write access to this register, the new value will be send as a time code character over the active SpaceWire links.

5 SMCS332SpW Modes

This section describes the organization of data passing through the HOCI and COMI ports on the SMCS332SpW, the procedure used to arbitrate between two SMCS332SpW on the COMI port, control by link, the SMCS332SpW routing capability., the time code interface and the version control register.

5.1 HOCI Data Transfer

Big/Little endian selection of the HOCI is done using a special pin (HOSTBIGE) of the SMCS332SpW. By connecting this pin to either Vcc or GND the HOCI is configured to be in little or big endian mode as follows:

When Signal HOSTBIGE = '0' (GND), the HOCI data port is in little endian mode.
When Signal HOSTBIGE = '1' (Vcc), the HOCI data port is in big endian mode.

Little endian mode selected:

- **8 bit data port (default after reset)**
  - register byte 0 is connected with pin HDATA7 - HDATA0

- **16 bit data port**
  - register byte 0 is connected with pin HDATA7 - HDATA0
  - register byte 1 is connected with pin HDATA15 - HDATA8

- **32 bit data port**
  - register byte 0 is connected with pin HDATA7 - HDATA0
  - register byte 1 is connected with pin HDATA15 - HDATA8
  - register byte 2 is connected with pin HDATA23 - HDATA16
  - register byte 3 is connected with pin HDATA31 - HDATA24

Big endian mode selected:

- **8 bit data port (default after reset)**
  - register byte 0 is connected with pin HDATA31 - HDATA24

- **16 bit data port**
  - register byte 0 is connected with pin HDATA31 - HDATA24
  - register byte 1 is connected with pin HDATA23 - HDATA16

- **32 bit data port**
  - register byte 0 is connected with pin HDATA31 - HDATA24
  - register byte 1 is connected with pin HDATA23 - HDATA16
  - register byte 2 is connected with pin HDATA15 - HDATA8
  - register byte 3 is connected with pin HDATA7 - HDATA0

The registers of the SMCS332SpW are 1, 2 or 4 Bytes wide. That means, if the HOCI data port is in 8 bit mode, 4 read or write accesses are necessary to access a 4 Byte register (e. g. the interrupt mask register). In 16/32 bit mode the data bits 31 - 8 are '0' if an 8 bit register is read.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 45 of 131

5.2   COMI Data Transfer

Big/Little endian selection of the COMI is done using bit 2 of SICR.

When SICR (0x00) Bit 2 = '0', the COMI data port is in little endian mode.
When SICR (0x00) Bit 2 = '1', the COMI data port is in big endian mode.

Little endian mode selected:

-          **8 bit data port (default after reset)**
           - byte 0 is transferred via CMDATA7 - CMDATA0

-          **16 bit data port**
           - byte 0 is transferred via CMDATA7 - CMDATA0
           - byte 1 is transferred via CMDATA15 - CMDATA8

-          **32 bit data port**
           - byte 0 is transferred via CMDATA7 - CMDATA0
           - byte 1 is transferred via CMDATA15 - CMDATA8
           - byte 2 is transferred via CMDATA23 - CMDATA16
           - byte 3 is transferred via CMDATA31 - CMDATA24

Big endian mode selected:

-          **8 bit data port (default after reset)**
           - byte 0 is transferred via CMDATA31 - CMDATA24

  -        **16 bit data port**
           - byte 0 is transferred via CMDATA31 - CMDATA24
           - byte 1 is transferred via CMDATA23 - CMDATA16

-          **32 bit data port**
           - byte 0 is transferred via CMDATA31 - CMDATA24
           - byte 1 is transferred via CMDATA23 - CMDATA16
           - byte 2 is transferred via CMDATA15 - CMDATA8
           - byte 3 is transferred via CMDATA7 - CMDATA0

5.3   COMI Arbitration

The operation of two SMCS332SpW in master / slave mode is shown below:
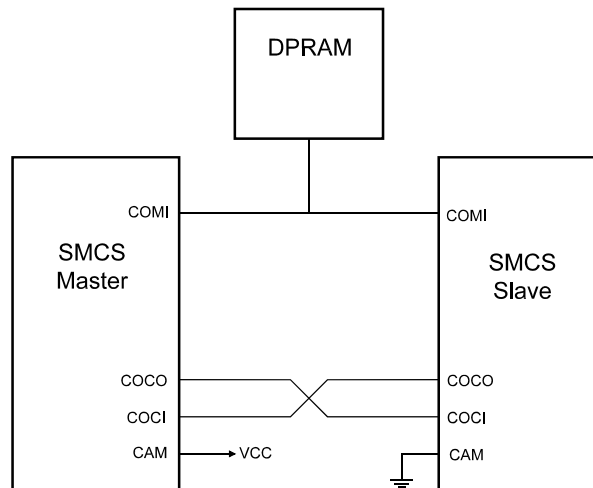
Two SMCS332SpW can work on a common memory without any external arbitration logic. However, if more than two SMCS332SpWs share a common communication memory an external arbiter is required.

The following describes the arbitration between two SMCS332SpWs. The rising edge of the RESET* signal loads the level of the CAM-signal into the SMCS332SpW. The level of the CAM-signal determines which SMCS332SpW owns the COMI BUS (is master and the other is slave).

The arbitration between the two SMCS332SpW follows a fair toggle and parking scheme. This means if SMCS332SpW -A got the communication memory bus after arbitration (or after reset) it "parks" on it until SMCS332SpW -B sends a request over his COCO output (active high) to the COCI input of SMCS332SpW -A. SMCS332SpW -B however sends only a request signal to SMCS332SpW -A if it needs the COMI bus for memory transfers.

After completion of the current memory transfer from SMCS332SpW -A, the SMCS332SpW -A COM interface becomes tristate and the SMCS332SpW -A COCO signal goes inactive (low). After SMCS332SpW -B receives the low level on its COCI, it occupies the COMI bus and enables its COM interface (drives the signals).

Now, SMCS332SpW -B "parks" on the bus until there is a request from SMCS332SpW -A.

If a bus owner releases the bus after a request from the other SMCS332SpW, it has to wait for N-1 Cycles (N = value from Register COMI_ACR), before it can request the bus again.

With the according values in the COMI_ACR register the bandwidth of the communication memory data bus can be split between the two SMCS332SpWs.

Example:

COMI_ACR SMCS332SpW -A = 4
COMI_ACR SMCS332SpW -B = 8
both SMCS332SpW read data from the communication memory
datawidth = 32 bit
CLK = 25 MHz => cycletime = 40ns

overall bandwidth:
 (4-1) + 1 arbitrationcycle + (8-1) + 1 arbitrationcycle = 12 cycles
 readcycles: (4-1) + (8-1) = 10 cycles

bandwidth 25 MHz * 10 / 12 * 4 Byte/cycle => 83.3 Mbyte/s
bandwidth SMCS332SpW -A:
        (8-1) = 7 readcycles
        25 MHz * 7/12 * 4 Bytes/cycle = 58.33 Mbyte/s

bandwidth SMCS332SpW -B:

(4-1) = 3 readcycles
25 MHz * 3/12 * 4 Bytes/cycle = 25 Mbyte/s

Note:
COMI_ACR value 0x01 is not allowed.
COMI_ACR value 0x00 means: disable communication memory interface.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 48 of 131

5.4   Control by Link

The SMCS332SpW offers the possibility to be used in a so-called "remote mode" which means that for controlling the SMCS332SpW (read/write registers) no local controller (µController, CPU, FPGA etc.) is required. Instead, the SMCS332SpW is being configured and controlled using one of the three links as a dedicated control link. Since the HOCI is no longer used to access the SMCS332SpW registers locally, it is instead available as a 32-bit bidirectional general purpose I/O (GPIO) port. The direction (in/out) of each bit of this port can be programmed individually.

5.4.1   Selecting remote mode

The external signal "BOOTLINK" (pin 32) must be tied to "1". This will set the SMCS332SpW to remote mode. It is not possible to switch the SMCS332SpW into remote mode by software.

5.4.2   Determination of the control link

After the reset signal is released the SMCS332SpW will wait for NULL characters on the three links. It will not start a link until it has received NULL characters on this link. Only then the SMCS332SpW will send NULL characters itself. After the links are active, the SMCS332SpW scans these links for data (characters). The link on which the SMCS332SpW receives data first is then identified and determined as "control link". This means that each of the 3 SMCS332SpW links can function as control link. Which of the links is used as control link is only determined by the first appearance of data on a link directly after it is started. That link then operates as "control link" until the SMCS332SpW is reset by the reset signal RESET

5.4.3   Protocol and Commands

The protocol of the SMCS332SpW in remote mode provides two commands: **Read** and **Write**. A read command requires at least one byte and a write command requires at least two bytes. Each command packet must be terminated by an EOP character. Data received on the control link is interpreted according to this simple protocol. Bit 7 (the MSB) of the first byte received on the control link determines whether the command is a **Read** (bit 7 = 1) or a **Write** (bit 7 = 0) command. All internal registers of the SMCS332SpW can be written

Read Command:

7 6 0		
1 register address	ignored	EOP
1. byte	2. - n. byte	

Read Response Packet:

7 6 0		
0 register address	register value	EOP
1. byte	2. byte	

Write Command:

7 6 0			
0 register address	write data	ignored	EOP
1. byte	2. byte	3. - n. byte	
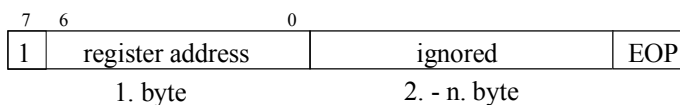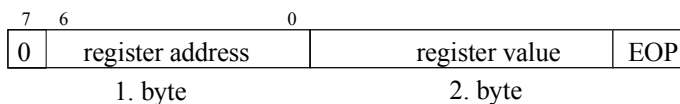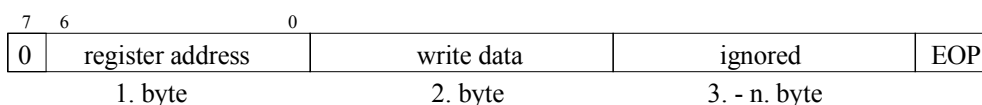
5.4.4   Host Data Bus / GPIO Port

If the SMCS332SpW is in remote mode, the HOCI cannot be used to control the SMCS332SpW locally. Instead, the 32-bit wide data bus of the HOCI can be used as GPIO port. Four registers are provided to control the direction of each line of the 32-bit wide GPIO port. This means that each line of the GPIO port can be configured as input or as output. After a reset, the GPIO port is configured that all lines are input lines. In addition, four extra registers are provided to write data to the GPIO port outputs and to read data from the GPIO port inputs.

The GPIO port outputs are latched so that the value written to a specific bit will not change unless programmed accordingly. As an example, a single-shot pulse on one of the GPIO output lines (0-1-0) needs to be programmed remotely by writing a '0', a '1' and another '0' to the appropriate GPIO register. The pulse width of such a pulse depends on the selected link speed and the operating frequency of the SMCS332SpW as well as the remote SpaceWire controller.

5.4.5   Restrictions

The SpaceWire link, which controls the SMCS332SpW, must be a direct link (point to point) connection.

Using the control link, all internal registers of the SMCS332SpW can be addressed. In addition, the GPIO port can be accessed via the control link. The only exception is that the receive part of the COMI associated with the control link cannot be used. This limitation is imposed since every byte received on the control link is interpreted as a command and therefore no data can be written into the communication memory via the control link. However, the transmit part of the COMI associated with the control link can be used to transfer data from the SMCS332SpW under remote control to another SpaceWire controller.

If data needs to be transferred via the COMI to an external device of the remotely controlled SMCS332SpW (e.g. memory), an additional link must be used for that purpose. The registers for that link can then be set via the control link.

The block diagram below shows a typical constellation of a remotely controlled SMCS332SpW controlled via link 2 from another SpaceWire device (in this case an SMCS332SpW). Note that if data is to be written to the RAM of the remote-controlled SMCS332SpW, an extra link (in the figure below this is link 1) needs to be used.
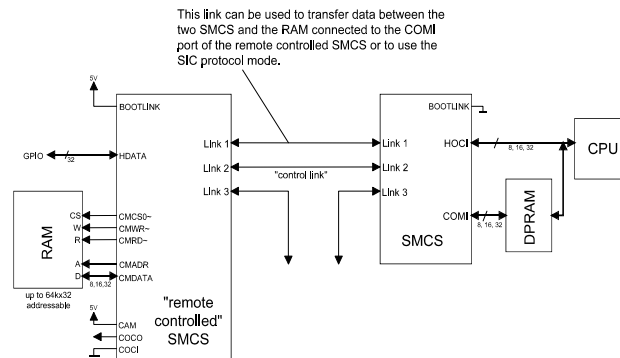
5.5   Wormhole Routing

5.5.1   Overview

To enable packets to be routed, each packet has a header at the front which contains routing information. The SMCS332SpW uses the header of each incoming packet to determine the link to be used to output the packet. Anything after the header is treated as the packet body until the packet terminator is received. This enables the SMCS332SpW to transmit packets of arbitrary length.

In most packet switching networks complete packets are stored internally, decoded and then routed to the destination node. This causes relatively long delays due to the high latency at each node. To overcome this limitation, the SMCS332SpW uses wormhole routing, in which the routing decision is taken as soon as the routing information, which is contained in the packet header, has been input. Therefore the packet header can be received, and the routing decision taken, before the whole packet has been transmitted by the source. A packet may be passing through several nodes at one time, thereby pipelining the transmission of the packet. The term wormhole routing comes from the analogy of a worm crawling through soil, creating a

hole that closes again behind its tail. Wormhole routing is invisible as far as the senders and receivers of packets are concerned, its only effect is to minimize the latency in message transmission.

The SMCS332SpW interprets the signals on its inputs as sequences of packets. It takes the first byte of data as the header of the packet, which determines what it will do with the whole packet. The length and contents of the remainder of the packet are arbitrary. The end of the packet is indicated by one of two distinguished termination characters, called EOP and EEP.

route address	packet body (0 or more byte)	EOP
1. byte	2. - n. byte	

The routing decision is taken as soon as the header of the packet has been input. If the output link is free, the packet (without header) is sent directly from input to output without being stored. If the output link is not free then the packet is buffered. To exploit the full bandwidth of the internal pathways on the SMCS332SpW, there is buffering on each path through the device. The buffering is fully hand shaken, FIFO buffering with minimal latency.

Note that if a packet is transmitted from a link running at a higher speed than the link on which it is received, there will be a loss of efficiency because the higher speed link will have to wait for data from the slower link. In most cases all the links in a network should be run at the same speed.

5.5.2 Wormhole routing on SMCS332SpW

The SMCS332SpW supports wormhole routing if in ROUTE_CTRL register D7 is set (bit7 = 1) and when operating in transparent mode. Wormhole routing could only be enabled for the whole SMCS332SpW not for dedicated channels only. The figure below shows the routing possibilities inside the SMCS332SpW. An incoming packet can be routed to the channelx receive fifo or to one of the two other links. The routing connection is terminated by the EOP or EEP of the routed packet.
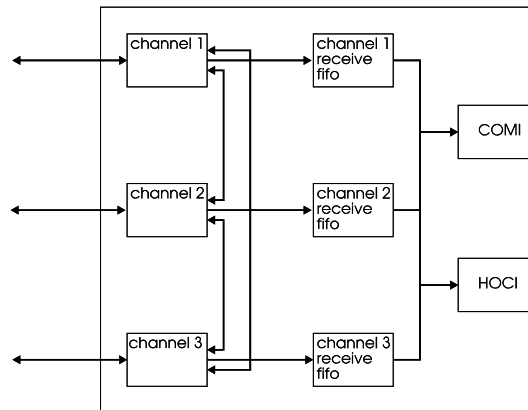
5.5.3 Routing Implementation on SMCS332SpW

When routing is enabled, the first byte of a received packet will be interpreted as the address destination byte (or routing header). The address byte will be removed from the packet (header deletion). The remaining packet is unchanged.

The address byte is compared with the contents of the CHx_ADDR and the contents of the register CHx_RT_ADDR of the two other channels. If the address byte matches with one of the three register contents, the packet will automatically forwarded to this link or the internal receive FIFO.

If the address byte does not match with the contents of one of the three registers the packet is forwarded to the internal receive FIFO and an error interrupt (if enabled) will be raised.

The figure below shows the address comparator. In the figure a packet comes in on channel 2. The leading byte (address byte) is compared with the contents of register CH2_ADDR, CH1_RT_ADDR, CH3_RT_ADDR and removed from the packet. If the address byte matches with the contents of one of the register the packet is routed along this channel.

If the address byte does not match at all the packet is routed to channel receive fifo and an interrupt is raised.
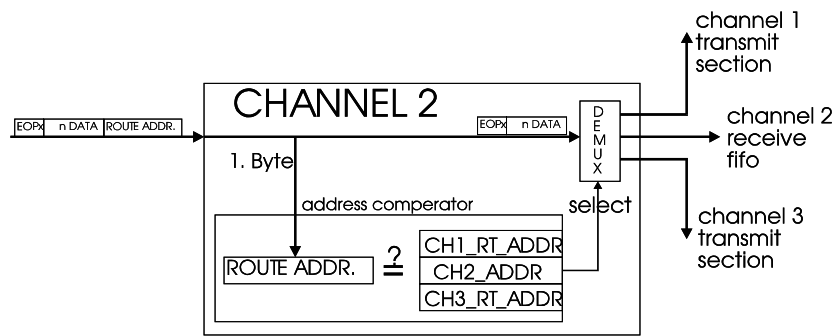
5.5.4 SMCS332332SpW Routing Examples

When routing, it is recommended that each of the following registers contains a different value: CH1_RT_ADDR, CH2_RT_ADDR, CH3_RT_ADDR and CHx_ADDR.

Initialization of SMCS332SpW for routing:

register: value:

- CH1_ADDR   = 0xF1
- CH2_ADDR   = 0xF2
- CH3_ADDR   = 0xF3
- CH1_RT_ADDR   = 0x21
- CH2_RT_ADDR   = 0x22
- CH3_RT_ADDR   = 0x23

 - RT_CTRL   = 0x80 (routing enabled)

Assume that data is incoming on channel 1.
The first byte will be compared with CH1_ADDR, CH2_RT_ADRR and CH3_RT_ADDR and depending on its value, one of the following actions will be taken.

Case 1) the first byte (address byte) of the received packet is 0xF1:
  ⇨ the packet (excluding address byte F1) will be routed to the internal receive FIFO of channel1.
Case 2) the first byte (address byte) of the received packet is 0x22:
  ⇨ the packet will be routed to the transmit section of channel2.
Case 3) the first byte (address byte) of the received packet is 0x23:
  ⇨ the packet will be routed to the transmit section of channel3.
Case 4) the first byte (address byte) of the received packet is neither 0xF1, 0x22 nor 0x23:
  ⇨ the packet will be routed to the internal receive FIFO of channel1 and error bit5 in CHx_ESR1 will be set.
   In parallel, an interrupt will be raised (if respective bit is set in IMR).

5.6 Header bytes generation

There are two modes to insert header bytes at the beginning of a packet.

A new mode controlled by the header field control bit and the well known SMCS332 mode with routing and checksum generation.

5.6.1 Header field control bit

For more flexibility for packets routed via a SpaceWire Router, the SMCS332SpW has a header field control bit, D5 in the register CHx_CNTRL1.

Description: If bit 5 in the register CHx_CNTRL1 is set, the SMCS332SpW will use the first byte of an incoming data to be transmitted (from COMI or HOCI) as number of bytes which are excluded from checksum (if checksum is enabled). The allowed range is from 1 to 15. The number byte and the associated header byte(s) defined as header field. The header field size is minimum 2 and maximum 16 bytes. It is always accessed in blocks of 4 bytes (the first 4, 8, 12 or 16 incoming bytes).

This means that data has to start at the next modulo 4 bytes. The rest of a 4-byte-block which is not covered by the number of header bytes will not be transmitted.

Example1:

CHx_CNTRL1 = 0x20

The header field consist of a first byte which contains the number of header bytes (value = 4) and 4 header bytes, give in total 5 bytes. The first 8 bytes comprise the header field. However the number byte and the last 3 bytes (of the 8 byte in total) will be NOT transmitted.

byte 0:   = 4   number of following header bytes (NOT transmitted!)
byte 1:   first header byte
byte 2:   second header byte
byte 3:   third header byte
byte 4:   forth header byte
byte 5:   ignored (NOT transmitted!)
byte 6:   ignored (NOT transmitted!)
byte 7:   ignored (NOT transmitted!)
byte 8:   first data byte

                        .

                        .

byte n: last data byte

transmitted packet via SpaceWire:
byte 1 - byte 4   (4 header bytes)
byte 8 - byte n   (data bytes)
EOP

Example2: No data bytes, short packets

CHx_CNTRL1 = 0x20

The header field consist of a first byte which contains the number of header bytes (value = 2) and 2 header bytes, give in total 3 bytes. The first 4 bytes comprise the header field. However the number byte and the last byte (of the 4 byte in total) will be NOT transmitted.

byte 0:   = 2   number of following header bytes (NOT transmitted!)
byte 1:   first header byte
byte 2:   second header byte
byte 3:   ignored (NOT transmitted!)
                     NO data byte

transmitted packet via SpaceWire:
byte 1
byte 2 (like data bytes)
EOP

Example3: with checksum generation

CHx_CNTRL1 = 0x30

The header field consist of a first byte which contains the number of header bytes excluded from checksum generation (value =3) and 3 header bytes, give in total 4 bytes.

byte 0: = 3   number of following header bytes excluded from checksum (NOT transmitted!)
byte 1:   header byte
byte 2:   header byte

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 53 of 131

byte 3:   header byte
byte 4:   first data byte (included in checksum)

.

.

byte n: last data byte (included in checksum)

transmitted packet via SpaceWire:
byte 1   (header byte)
byte 2   (header byte)
byte 3   (header byte)
byte 4 - byte n   (data bytes)
byte c1 - byte c2 (2 checksum bytes)
EOP

5.6.2   Routing and Checksum Generation

Assuming wormhole routing and checksum generation/check is enabled.

The first header byte will be removed from the packet on each SpaceWire Router, where the packet is routed through, if header deletion is enabled. If the header byte(s) is included in the checksum, the two checksums will never be equal. Therefore it is mandatory to exclude the header byte(s) from checksum if routing over SpaceWire Router is combined with checksum generation. The number of header bytes deleted during routing can range from one byte to several bytes. This depends on the number of routers a packet is routed through.

In this case the SMCS332SpW, generates two checksum bytes from the data bytes and appends these bytes at the end of the data bytes. The SMCS332SpW at the other end of the virtual link generates again a checksum from the received bytes(without the last 2 bytes) and compares these with the received checksum (last 2 bytes).

If routing is combined with checksum generation, set bit 4 in the register CHx_CNTRL1. The SMCS332SpW will use the first byte of the incoming data, to be transmitted (from HOCI or COMI), as number of bytes which should be excluded from checksum. The allowed range is from 1 up to 15. The number byte and the associated address byte(s) defined as header field. The header field size is minimum 2 and maximum 16 bytes. It is always accessed in blocks of 4 bytes (the first 4, 8, 12 or 16 incoming bytes). This means that data which should be included in the checksum has to start at the next modulo 4 bytes. The rest of a 4-byte-block which is not covered by the number of address bytes is not transmitted.

Example :

Bit 4 is set in register CHx_CNTRL1 **AND** Bit 7 in register RT_CTRL is set.

The first byte contains the number of header bytes excluded from checksum generation (value =1). The second byte is a header byte. In total 2 bytes. The first 4 bytes built the header field, but the last 2 bytes will be NOT transmitted.

byte 0: = 1   number of following header bytes excluded from checksum (NOT transmitted!)
byte 1:   header byte
byte 2:   ignored (NOT transmitted!)
byte 3:   ignored (NOT transmitted!)
byte 4:   first data byte (included in checksum)

.

.

byte n: last data byte (included in checksum)

transmitted packet via SpaceWire:
byte 1   (header byte)
byte 4 - byte n   (data bytes)
byte c1 - byte c2 (2 checksum bytes)
EOP

5.7 Time Code Interface

Both time code interface registers TIME_CNTRL and TIME_CODE controls the receiving and transmitting of time code characters. A time code on the SpaceWire links consists of an ESC control character and a data character: the time code value. The time code consists of the 6 bit time field (bit0-bit5) and two bits control flags.

5.7.1 SMCS33SpW transmit time code

After writing of a new value to the TIME_CODE register, the time code interface will send this new value to the 3 SpaceWire interfaces. The SpaceWire interface will send the time code only when it is in the RUN state.

A second way to send the time code is the TIME_CODE_SYNC signal. If bit 3 of the TIME_CNTRL register is set, a falling edge of the TIME_CODE_SYNC input signal sends the content of the TIME_CODE register over the active SpaceWire links. If bit 4 of the TIME_CNTRL register is set, the value of the TIME_CODE register will be incremented by 1 after the transmitting of the TIME_CODE value.

TIME_CNTRL Bit3	TIME_CNTRL Bit4	Description
0	0	No time code will be send after the falling edge of the TIME_CODE_SYNC signal
0	1	No time code will be send after the falling edge of the TIME_CODE_SYNC signal and the value of the TIME_CODE will not be incremented
1	0	A falling edge of the TIME_CODE_SYNC input signal sends the time code register value over the active SpaceWire links.
1	1	A falling edge of the TIME_CODE_SYNC input signal sends the time code register value over the active SpaceWire links and the TIME_CODE register bits 5-0 will be incremented by 1.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 55 of 131

5.7.2 SMCS332SpW receive time code

Status bits 5, 6 or 7 in register TIME_CNTRL will be set, when the SMCS332SpW receives a time code character over link1, 2 or 3. If bit 2 of the TIME_CNTRL register is set, the received time code will be written in the TIME_CODE register.

Interrupt generation: See also interrupt bit in ISR Byte 3, bit 6.

TIME_CNTRL Bit1	TIME_CNTRL Bit0	Description
0	0	No interrupt signal to the interrupt controller.
0	1	Generates an interrupt signal to the interrupt controller for a valid (TIME_CODE register (bit5-0) + 1) received TIME CODE character received from the SpaceWire links.
1	0	Generates an interrupt signal to the interrupt controller for all received TIME CODE characters.
1	1	Generates an interrupt signal to the interrupt controller for a invalid TIME CODE characters received from the SpaceWire links.

5.8 SMCS332SpW Version Control

The Software needs often the capability to check which version of the SMCS332 is used, the new SMCS332SpW or the older SMCS332. The solution for this problem is, the SW can use the new TIME_CNTRL register.

In the old version of the SMCS332 there is no register on this address, therefore when the software writes the value 0x01 to this address, they will read the value 0x00 from the old SMCS332 and 0x01 from the new SMCS332SpW.

6 Programming and Operation Modes

This section contains the descriptions how to operate the SMCS332SpW from the programmers point of view, together with some code examples.

6.1 SMCS332SpW Initialization

After power-up or any other reset the configuration registers of the SMCS332SpW are set to their default values (see chapter register description). For proper operation, each application has to adapt them according to their specific needs.

The following list shows registers that are important for the SMCS332SpW initialization. It represents a configuration guideline for the configuration subsystem. Each register description is accompanied by an exemplary value.

6.1.1 SMCS332SpW Interface Control Register

This register determines the HOCI data bus width and the COMI data bus endian mode. **This register should be the first to be configured, to ensure proper SMCS332SpW operation.** Setting the HOCI port to 32 bit all internal registers are accessible with one read or write operation (in 8 bit mode each byte of a 16 or 32 bit register must be accessed separately).

6.1.2 Transmit Bitrate Register (TRS_CTRL)

This allows configuration of the maximum transmit bit rates between 80 and 200MBit/s (see also Channel specific mode register). The reset value of the register (0x0A) results in a transmit bitrate of 100 MBit/s.

The Transmit bitrate register can be configured between 80 and 200 MBit/s in 20 MBit steps as follows:

Register Value	Max.Transmit Bitrate (MBit/s)
0x08	80
0x0A	100 (default)
0x0C	120
0x0E	140
0x10	160
0x12	180
0x14	200

6.1.3 SMCS332SpW Interrupt Status Register (ISR)

This register contains all interrupt status information. It should be read out initially after reset to clear all illegal latched interrupts.

6.1.4 SMCS332SpW Interrupt Mask Register

All interrupts are masked after reset. The interrupt mask bits are located according to their pendants in the ISR. A '1' written to the IMR enables the corresponding interrupt in ISR to be activated towards the CPU via the signal HINTR*

6.1.5 Channel specific configuration registers

The channel specific registers are configurable independently. This allows flexible communications with different nodes at the same time. The following registers are the most important to be set. (x stands for Channel no. 1, 2 or 3)

6.1.5.1 SpaceWire Mode Register (CHx_DSM_MODR)

The basic function of this register is to select the transmit speed.

Bit 3 enables the divider (bit 2 to 0) for transmit bit rate selection.

value	effect
0x00	10 MBit/s
0x0E	lowest transmission bitrate
0x08	highest transmission bitrate

6.1.5.2 COMI Configuration Register (CHx_COMICFG)

This register must be set up if data transfer via COMI is desired. It determines the COMI data bus width for receive and transmit separately, transmit EOP character or not at the end of a packet and the structure of data storage in the Communication Memory (refer to chapter "Data Transfer via COMI").

6.1.5.3 Control Register 1 (CHx_CNTRL1)

No configuration has to be made if transmission in transparent mode is desired. For operation with the SMCS332SpW in protocol mode the respective bits are to set, see chapter 13.

6.1.5.4 SpaceWire Command Register (CHx_DSM_CMDR)

The link can be started by writing a '1' to bit 1 of this register. The corresponding link begins to send NULLs, the first flow control characters (FCT) will be exchanged after a NULL has been received. Each link can be started or stopped independently but it should be considered that stopping a link will result in a link disconnect error at the receiving end.

6.2 Data transfer via COMI

The transmission via the Communication Memory Interface (COMI) is a very efficient way to transfer data packets over the SpaceWire-links. The CPU only fills the Communication Memory (DPRAM) with the relevant data words and sets the registers in the SMCS332SpW that point on these words. Further transfer activities are applied by the SMCS332SpW without any CPU intervention. For each link channel, 6 registers are provided. 4 registers to assign an area in the Communication Memory and 2 registers which carry the current address value.

Register	Function
CHx_TX_SAR	Channel x Transmission Start Address Register
CHx_TX_EAR	Channel x Transmission End Address Register
CHx_TX_CAR	Channel x Transmission Current Address Register
CHx_RX_SAR	Channel x Receive Start Address Register
CHx_RX_EAR	Channel x Receive End Address Register
CHx_RX_CAR	Channel x Receive Current Address Register

TX and RX registers are programmable independently thus there is no interference between TX and RX activities. With writing on the End Address Register the transmit/receive activity of the respective channel is initiated. With reading the Current Address Register the status of data transfer of this activity can be observed.

A successful packet transmission/reception or the occurrences of an error is indicated through an interrupt by the SMCS332SpW. The cause of an interrupt can be read from the Interrupt Status Register (ISR) via the Host Control Interface (HOCI).

The following events cause an interrupt and are important for the transmission via COMI in transparent mode:

- SpaceWire link error (disconnect, parity, credit and ESC error)
- packet transmission completed (TX Current Address Reg = TX End Address Reg.)
- reception segment in Communication Memory full
- packet reception completed (EOP or EEP received) (RX Current Address Reg. = RX End Address Reg.)
- check sum error (if enabled)

With setting a bit in the Interrupt Mask Register the respective event is enabled.

The interrupt capability can be switched off by masking all interrupt sources through setting the complete interrupt mask register to "0" when the system is not able to service it. In this case SMCS332SpW ISR can be polled to wait for the event.

The structure of storage of data words can be configured for each channel independently. The following list shows the switches that are possible:

- storage of one versus more packets per segment. If more than one packet is allowed to be stored in one segment, the user must take care about the separation of the different packets.
- data word width to 8 versus 16 versus 32
- sign bit expand versus no expand in 8/16 bit mode

Furthermore the SMCS332SpW supports both little and big endian mode on its COMI (as well as for the HOCI) to be compliant to the most processors.

Program Sequence

Before starting with transmission the channel must be configured for proper operation. Besides general SMCS332SpW configuration (see chapter SMCS332SpW initialization) at least the following channel specific registers must be set:

Register	Function
CHx_DSM_MODR	Channel Mode Register
CHx_DSM_CMDR	Channel Command Register
CHx_COMICFG	Channel COMI Configuration Register

The Channel Mode Register is basically used to select the transmit bitrate of the link. Then the link can be started by setting up the Channel Command Register. No data is transmitted yet, but the link sends NULL's and exchanges Flow Control Tokens with the opposite node (if started as well).

The Channel COMI Configuration Register determines the word width of the COMI data bus basically. Furthermore it configures whether one or more packets fit into one Communication Memory segment. In the first case the SMCS stops the RX-address generation after receiving an EOP/EEP and for the reception of another packet the receive channel must be set up again. In the second case the packets are appended and the SMCS332SpW only signals that the end of the segment is reached (RX generator finished). The CPU/User must have the knowledge to separate the packets itself.

Example #1:

Link no.1 shall transmit two packets of 16 words ą 32 bit with the maximum bit rate. The same link shall be configured to receive one packet of up to 64 words ą 16 bit. Link no.2 shall collect 32 bit word packets in the COMI segment without signaling reception of every packet. The transfers run via the COMI in little endian mode (default).

The Communication Memory is segmented in three parts.

The excerpt of a Pseudo C-code looks like this:

```
/* init */
ch1_rx_int_occured = 0;
ch1_tx_int_occured = 0;
ch2_rx_int_occured = 0;


/* usage: smcs_reg_write(SMCS base address, register address, value) */
/* configure COMI */
/* Channel 1:
 *  32 bit words transmit port followed by EOP,
 *  16 bit receive port, stop RX generator after EOP reception,
 *  no sign expand
 */
smcs_reg_write(SMCS, CH1_COMICFG, 0x13);


/* Channel 2: 32 bit word receive port, no RX generator stop after EOP reception,   * no
sign expand
 */
smcs_reg_write(SMCS, CH2_COMICFG, 0x70);


/* set interrupt mask (only access on SMCS_IMR0 with 32 bit HOCI) */
/* disable <CH1 TX generator finished>, <CH1 RX EOP/EEP received>,
```

```
 * <CH2 RX generator finished>
 */
smcs_reg_write(SMCS, SMCS_IMR0, 0x2014);


/* select transmit rate to max. Transmit bitrate*/
smcs_reg_write(SMCS, CH1_DSM_MODR, 0x08);
smcs_reg_write(SMCS, CH2_DSM_MODR, 0x08);
/* start the links */
smcs_reg_write(SMCS, CH1_DSM_CMDR, 0x02);
smcs_reg_write(SMCS, CH2_DSM_CMDR, 0x02);


/* start transfers */
smcs_reg_write(SMCS, CH1_TX_SAR, 0x00);
smcs_reg_write(SMCS, CH1_TX_EAR, 0x0F);
smcs_reg_write(SMCS, CH1_RX_SAR, 0x100);
smcs_reg_write(SMCS, CH1_RX_EAR, 0x13F);
smcs_reg_write(SMCS, CH2_RX_SAR, 0x200);
smcs_reg_write(SMCS, CH2_RX_EAR, 0x2FF);


while (1)
{
   static int tx1_count = 0;

   wait_for_int();                          /* wait for interrupts */

   if (ch1_tx_int_occured && (tx1_count == 0))
   {
      ch1_tx_int_occured = 0;
      tx1_count++;

      /* start transfer of second packet over Link #1 */
      smcs_reg_write(SMCS, CH1_TX_SAR, 0x10);
      smcs_reg_write(SMCS, CH1_TX_EAR, 0x1F);
   }


   if (ch1_tx_int_occured && ch1_rx_int_occured && ch2_rx_int_occured)
      break;
}
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 61 of 131

6.3 Data Transfer via HOCI

The Host Control Interface is mainly designed for the access to internal SMCS332SpW registers. But for small packets data transfer can be performed via this interface for the purpose of saving board space and power through the absence of the Communication Memory. Also applications with sample oriented processing can take advantage of this feature.

The CPU user makes usage of its direct access on the SMCS332SpW internal FIFOs to read/write the single data words to be transferred. With checking the FIFO flags the CPU must assure to handle the process of communication properly. This action results in a programming overhead of course, but saves board space and power.

Program Sequence

The transfer is controlled by 4 registers:

Register	Function
CHx_TX_FIFO	channel x Transmit FIFO
CHx_TX_EOPB	channel x Transmit EOP Bit Register
CHx_RX_FIFO	channel x Receive FIFO
CHx_STAR	channel x Status Register

For transmitting data the CPU writes the words directly to the channel transmit FIFO address. The end of a transmit packet is indicated by writing on the EOP-Bit- register. The EOP termination is appended to the last word as a control character on the link. To recognize the end of an incoming (receiving) packet the CPU reads the channel status or the interrupt status register. (Note: Reading the interrupt status register will clear its contents. Thus relevant information concerning other link channels must be saved for further evaluation if so required.)

Before accessing the FIFOs, the CPU must evaluate the status of the FIFO flags. For each channel, FIFO status flags are provided:

- transmit FIFO empty
- transmit FIFO full
- receive FIFO not empty
- receive FIFO full

With the first write access on the transmit FIFO the 'transmit FIFO empty'-flag becomes inactive. The 'transmit FIFO full' flag must be examined carefully to assure that no data words will be overwritten.

The first two data words once written in the transmit FIFO are not immediately sent towards the listening node but remain in the queue. Either with adding the next word to the packet or writing the EOP bit on the Transmit EOP Bit Register the SMCS332SpW begins transmission of one word or the whole packet respectively.

Example #2:

The following two C-code routines show a handling of transferring small packets. (m: SMCS number, n: channel number)

```c
#define TXfull      0x02;
#define RXnotEmpty  0x04;
#define RXrecvEOP   0x60;


int smcs_fifo_read(int SMCSm,        /* SMCS to use */
                   int Chn_STAR,
                   int Chn_RX_FIFO,
                   int amount,       /* number of words to read */
                   int *rxdpram)     /* pointer on receive buffer */
{
   int num;
   for (num = 0; num < amount; num++)
   {
      while (!(smcs_reg_read(SMCSm, CHn_STAR) & RxnotEmpty))
        ;                             /* wait for data */

      *rxdpram++ = smcs_reg_read(SMCSm, CHn_RX_FIFO);
      if (smcs_reg_read(SMCSm, CHn_STAR) & RXrecvEOP)
        break;
   }
   return num;                        /* return number of words read */
}
void smcs_fifo_write(int SMCSm,      /* SMCS to use */
                   int Chn_STAR,
                   int Chn_TX_FIFO,
int Chn_TX_EOPB,
                   int amount,       /* number of words to transmit */
                   int *txdpram)     /* pointer on transmit buffer */
{
   int num;
   for (num = 0; num < amount; num++)
   {
      while (smcs_reg_read(SMCSm, CHn_STAR) & TXfull)
        ;                             /* wait for space in fifo */
      smcs_reg_write(SMCSm, CHn_TX_FIFO, *txdpram++);
   }
                                      /* send EOP */
   smcs_reg_write(SMCSy, CHx_TX_EOPB, EOP);
}
```

6.3.1   Special behaviour in case of SpaceWire link error

Data written to the HOCI transmit FIFO are not transmitted immediately as already described in the previous section. This

causes a special behaviour if a link error occurs. The remainder of the last written packet remains in the internal pipe which is not cleared in case of a SpaceWire link error. After the link reconnects, this remainder will be transmitted immediately after a new data character or an EOP is written to the HOCI FIFO.

To prevent a transmission of the remaining bytes in front of a new packet after link reconnects, following sequence is proposed:

1. After the link disconnects, reset the transmit section of the channel by setting of bit D0 in register CHx_CNTRL2

2. If the packet isn't finished with an EOP then close the packet by writing to the EOP register 0x27 (transmit EOP)

3. Clear the bit D0 in register CHx_CNTRL2

4. restart the link

7 Signal Description

The Figure below shows the SMCS332SpW embedded in a typical module environment:
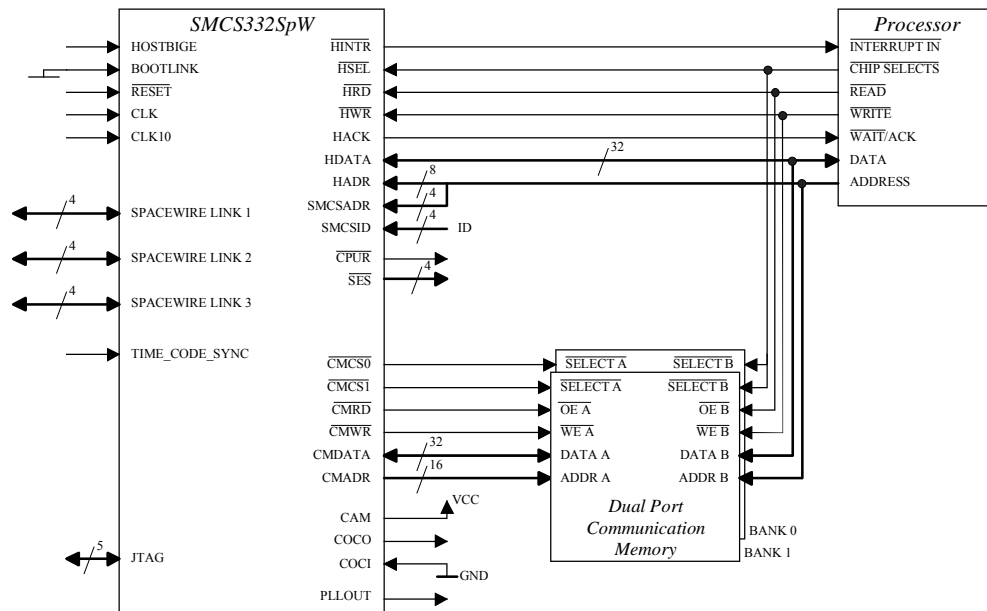
This section describes the pins of the SMCS. Groups of pins represent busses where the highest number is the MSB.

O = Output; I = Input; Z = High Impedance

O/Z = if using a configuration with two SMCS332SpWs these signals can directly be connected together (WIROR)

(*) = active low signal

Signal Name	Type	Function	max. output current [mA]	load [pF]
HSEL*	I	Select host interface		
HRD*	I	host interface read strobe		
HWR*	I	host interface write strobe		
HADR(7:0)	I	SMCS register address lines. This address lines will be used to access (address) the SMCS registers.		
HDATA(31:0)	I/O/Z	SMCS data	3	50
HACK	O/Z	host acknowledge. SMCS deasserts this output to add wait states to a SMCS access. After SMCS is ready this output will be asserted.	3	50
HINTR*	O	host interrupt request line	3	50
SMCSADR(3:0)	I	SMCS Address. The binary value of this lines will be compared with the value of the SMCS ID lines.		
SMCSID(3:0)	I	SMCS ID lines: offers possibility to use sixteen SMCS within one HSEL*		
HOSTBIGE	I	0: host I/F Little Endian		

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 65 of 131

Signal Name	Type	Function	max. output current [mA]	load [pF]
		1: host I/F Big Endian		
BOOTLINK	I	0: control by host 1: control by link		
CMCS(1:0)*	O/Z	Communication memory select lines. These pins are asserted as chip selects for the corresponding banks of the communication memory.	6	25
CMRD*	O/Z	Communication memory read strobe. This pin is asserted when the SMCS reads data from memory.	6	25
CMWR*	O/Z	Communication memory write strobe. This pin is asserted when the SMCS writes to data memory.	6	25
CMADR(15:0)	O/Z	Communication memory address. The SMCS outputs an address on these pins.	6	25
CMDATA(31:0)	IOZ	Communication memory data. The SMCS inputs and outputs data from and to com. memory on these pins.	3	25
COCI	I	Communication interface 'occupied' input signal	3	50
COCO	O	Communication interface 'occupied' output signal		
CAM	I	Communication interface arbitration master input signal (see section 4.3) 1: master 0: slave		
CPUR*	O	CPU Reset Signal (can be used as user defined flag)	3	50
SES(3:0)*	O	Specific External Signals (can be used as user defined flags)	3	50
LDI1	I	Link Data Input channel 1		
LSI1	I	Link Strobe Input channel 1		
LDO1	O	Link Data Output channel 1	12	25
LSO1	O	Link Strobe Output channel 1	12	25
LDI2	I	Link Data Input channel 2		
LSI2	I	Link Strobe Input channel 2		
LDO2	O	Link Data Output channel 2	12	25
LSO2	O	Link Strobe Output channel 2	12	25
LDI3	I	Link Data Input channel 3		
LSI3	I	Link Strobe Input channel 3		
LDO3	O	Link Data Output channel 3	12	25
LSO3	O	Link Strobe Output channel 3	12	25
TRST*	I	Test Reset. Resets the test state machine.		

Signal Name	Type	Function	max. output current [mA]	load [pF]
TCK	I	Test Clock. Provides an asynchronous clock for JTAG boundary scan.		
TMS	I	Test Mode Select. Used to control the test state machine.		
TDI	I	Test Data Input. Provides serial data for the boundary scan logic.		
TDO	O	Test Data Output. Serial scan output of the boundary scan path.	3	50
RESET*	I	SMCS Reset. Sets the SMCS to a known state. This input must be asserted (low) at power-up. The minimum width of RESET low is 5 cycles of CLK10 in parallel with CLK running.		
CLK	I	External clock input to SMCS (max. 25 MHz). Must be derived from RAM access time.		
CLK10	I	External clock input to SMCS DS-links (application specific, nominal 10 MHz). Used to generate to transmission speed and link disconnect timeout.		
TIME_CODE_SYNC	I	A falling edge on this signal sends (if enabled) the internal SpaceWire time code value over the links.		
PLLOUT	O	Output of internal PLL. Used to connect a network of external RC devices. No PLL clock output!		
VCC		Power Supply		
GND		Ground		

8 Electrical Specifications

8.1 Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.5 to +5.5 -0.5 to +3.6	V
I/O Voltage		-0.5 to VCC + 0.5	V
Operating Temperature Range (Ambient)	T_A	-55 to +125	°C
Junction Temperature	T_J	$T_J < T_A + 20$	°C
Storage Temperature Range	T_{stg}	-65 to +150	°C
Thermal Resistance: junction to ambient, still air	RTh_{JA}	38	°C/W

Stresses above those listed may cause permanent damage to the device.

8.2 DC Electrical Characteristics

SMCS332SpW can work with $V_{CC} = + 5\ V \pm 0.5V$ and $V_{CC} = + 3.3\ V \pm 0.3V$

Parameter	Symbol	Min.	Max.	Unit	Conditions
Operating Voltage	V_{CC}	4.5 3.0	5.5 3.6	V	
Input HIGH Voltage	V_{IH}	2.0		V	
Input LOW Voltage	V_{IL}		0.8	V	
Output HIGH Voltage	V_{OH}	2.4		V	max. output current [1]
Output LOW Voltage	V_{OL}		0.4	V	max. output current [1]
Output Short circuit current	I_{OS}		48 36	mA mA	$V_{OUT} = VCC$ $V_{OUT} = GND$

[1] see also the signal description in chapter 7.

Although specified for TTL outputs, all SMCS332SpW outputs are CMOS compatible and will drive to VCC and GND assuming no dc loads.

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 68 of 131

8.3   Power consumption

Max. power consumption figures at $V_{CC} = + 5.5V/ -55°C$, CLK = 25MHz are:

Operation Mode	Power consumption [mA]
not clocked	8
SMCS in RESET	75
SMCS in IDLE	115
Maximum	310

Max. power consumption figures at $V_{CC} = + 3.6V/ -55°C$, CLK = 15 MHz are:

Operation Mode	Power consumption [mA]
not clocked	4
SMCS in RESET	24
SMCS in IDLE	38
Maximum	100

8.4 PLL Filter

The pin PLLOUT should be connected as shown below:
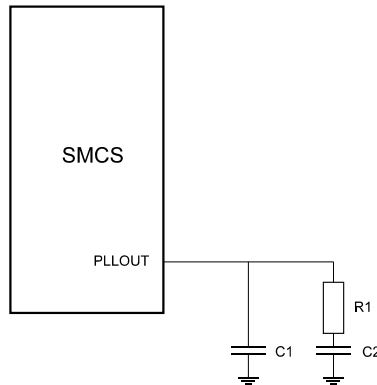
Values for V_{CC} = + 5 V ± 0.5V

    R1 = 1,8kΩ ± 5%, ¼W

    C1 = 33pF, ± 5%, 200V

    C2 = 820pF, ± 5%, 200V

Values for V_{CC} = + 3.3 V ± 0.3V

    R1 = 2,0kΩ ± 5%, ¼W

    C1 = 33pF, ± 5%, 200V

    C2 = 760pF, ± 5%, 200V

8.5 Power and Ground Guidelines

To achieve its fast cycle time, the SMCS332SpW is designed with high speed drivers on output pins. Large peak currents may pass through a circuit board's ground and power lines, especially when many output drivers are simultaneously charging or discharging their load capacitances. These transient currents can cause disturbances on the power and ground lines. To minimize these effects, the SMCS332SpW provides separate supply pins for its internal logic and for its external drivers.

All GND pins should have a low impedance path to ground. A ground plane is required in SMCS332SpW systems to reduce this impedance, minimizing noise.

The VCC pins should be bypassed to the ground plane using approximately 10 high-frequency capacitors (0.1 µF ceramic). Keep each capacitor's lead and trace length to the pins as short as possible. This low inductive path provides the SMCS332SpW with the peak currents required when its output drivers switch. The capacitors' ground leads should also be short and connect directly to the ground plane. This provides a low impedance return path for the load capacitance of the SMCS332SpW output drivers.

The following pins must have a capacitor: 20, 78, 129, and 155. The remaining capacitors should be distributed equally around the SMCS332SpW.

9 Timing Parameters

9.1 Clock Signals
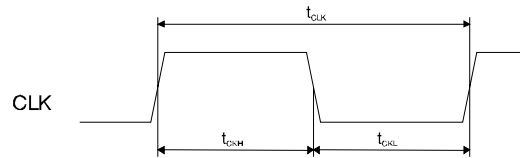
$V_{CC} = 5 \text{ V} \pm 0.5\text{V}$

Description	Symbol	Min.	Max.	Unit
CLK period [1]	t_{CLK}	40		ns
CLK width high	t_{CLKH}	16		ns
CLK width low	t_{CLKL}	16		ns

[1] Max. 25 MHz

$V_{CC} = 3.3 \text{ V} \pm 0.3\text{V}$

Description	Symbol	Min.	Max.	Unit
CLK period [1]	t_{CLK}	66,6		ns
CLK width high	t_{CLKH}	26,6		ns
CLK width low	t_{CLKL}	26,6		ns

[1] Max. 15 MHz
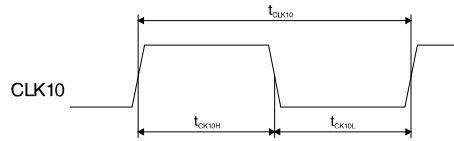
$V_{CC} = 5\ V \pm 0.5V$

Description	Symbol	Min.	Max.	Unit
CLK10 period [1]	t_{CLK10}	100	100	ns
CLK10 width high	t_{CLK10H}	40		ns
CLK10 width low	t_{CLK10L}	40		ns

[1] Typically 10 MHz

$V_{CC} = 3.3\ V \pm 0.3V$

Description	Symbol	Min.	Max.	Unit
CLK10 period [1]	t_{CLK10}	100	100	ns
CLK10 width high	t_{CLK10H}	40		ns
CLK10 width low	t_{CLK10L}	40		ns

[1] Typically 10 MHz

9.2 Reset
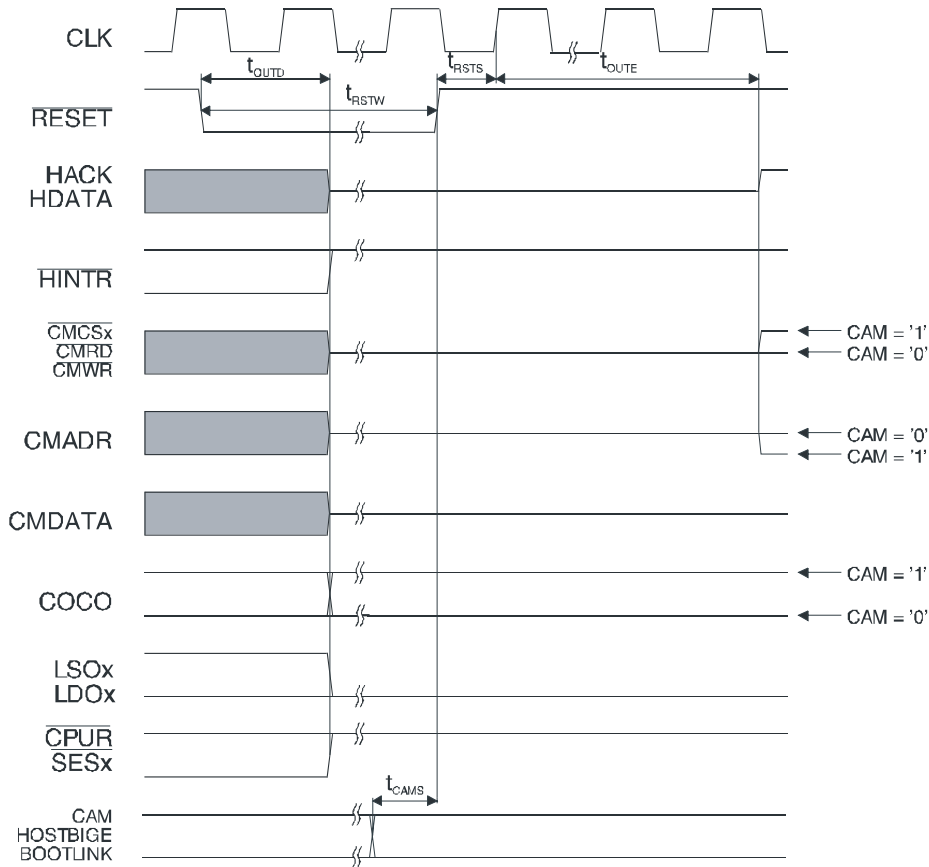
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
RESET* low pulse width	t_{RSTW}	$2 * t_{CLK}$		ns
Output disable after RESET* low	t_{OUTD}		42	ns
Output enable after CLK high	t_{OUTE}		$3 * t_{CLK} + 26$	ns
CAM, HOSTBIGE, BOOTLINK setup before RESET* high	t_{CAMS}	1		

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
RESET* low pulse width	t_{RSTW}	$2 * t_{CLK}$		ns
Output disable after RESET* low	t_{OUTD}		46	ns
Output enable after CLK high	t_{OUTE}		$3 * t_{CLK} + 28$	ns
CAM, HOSTBIGE, BOOTLINK setup before RESET* high	t_{CAMS}	1		

9.3   Host Read
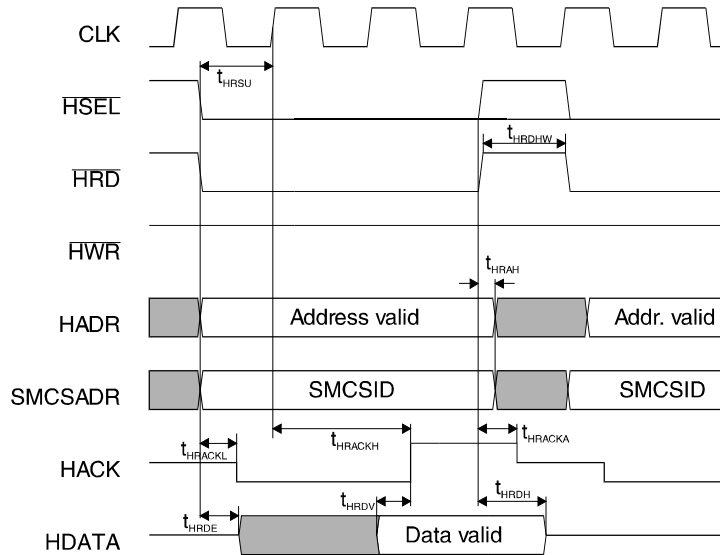
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
HSEL*, HRD* and SMCSADR and HADR setup before CLK high	t_{HRSU}	5		ns
HADR, SMCSADR hold after HSEL*, HRD* high	t_{HRAH}	0		ns
HRD* pulse width high	t_{HRDW}	5		ns
HACK low after HRD*, HSEL* active and SMCSADR valid [1]	t_{HRACKL}		14	ns
HACK high after CLK high	t_{HRACKH}	$1 * t_{CLK} + 5$	$3 * t_{CLK} + 21$	ns
HACK disable after HRD*, HSEL* inactive or SMCSADR invalid [2]	t_{HRACKA}		16	ns
HDATA valid before HACK high	t_{HRDV}	0		ns
HDATA hold after HRD*, HSEL* inactive or SMCSADR invalid [2]	t_{HRDH}	5	19	ns
HDATA enable after HRD*, HSEL* active and SMCSADR valid [1]	t_{HRDE}	4		ns

Notes:

[1] Signal HACK active when HRD* low and HSEL* low and SMCSADR = SMCSID

[2] Signal HACK disable when HRD* high or HSEL* high or SMCSADR ≠ SMCSID

EADS ASTRIUM

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 74 of 131

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
HSEL*, HRD* and SMCSADR and HADR setup before CLK high	t_{HRSU}	7		ns
HADR, SMCSADR hold after HSEL*, HRD* high	t_{HRAH}	0		ns
HRD* pulse width high	t_{HRDW}	7		ns
HACK low after HRD*, HSEL* active and SMCSADR valid [1]	t_{HRACKL}		23	ns
HACK high after CLK high	t_{HRACKH}	$1 * t_{CLK} +$	$3 * t_{CLK} + 35$	ns
HACK disable after HRD*, HSEL* inactive or SMCSADR invalid [2]	t_{HRACKA}		25	ns
HDATA valid before HACK high	t_{HRDV}	0		ns
HDATA hold after HRD*, HSEL* inactive or SMCSADR invalid [2]	t_{HRDH}	5	28	ns
HDATA enable after HRD*, HSEL* active and SMCSADR valid [1]	t_{HRDE}	4		ns

Notes:

[1] Signal HACK active when HRD* low and HSEL* low and SMCSADR = SMCSID

[2] Signal HACK disable when HRD* high or HSEL* high or SMCSADR ≠ SMCSID

EADS ASTRIUM

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 75 of 131

9.4 Host Write
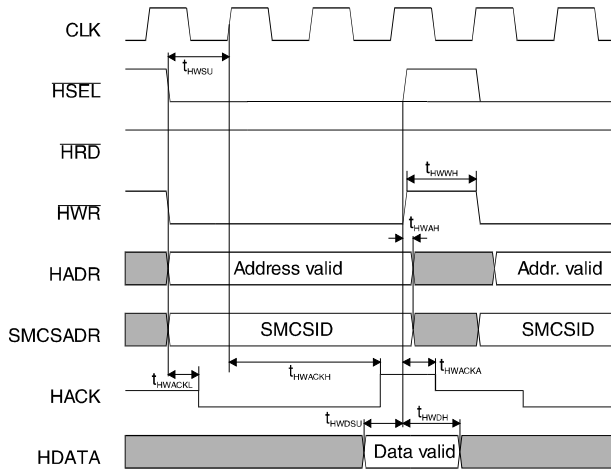
V_{CC} = 5 V ± 0.5V

Description	Symb.	Min.	Max.	Unit
HSEL*, HWR* and SMCSADR and HADR setup before CLK high	t_{HWSU}	5		ns
HADR, SMCSADR hold after HSEL* or HWR* high	t_{HWAH}	0		ns
HWR* pulse width high [1]	t_{HWWH}	$1 * t_{CLK} + 5$		ns
HACK low after HWR*, HSEL* active and SMCSADR valid [2]	t_{HWACKL}		14	ns
HACK high after HSEL* and HWR* and SMCSADR = SMCSID	t_{HWACKH}	$1 * t_{CLK} + 5$	$2.5 * t_{CLK} + 24$	ns
HACK disable after HWR* or HSEL* inactive or SMCSADR invalid [3]	t_{HWACKA}		15	ns
HDATA setup before HSEL* or HWR* high or SMCSADR ≠ SMCSID	t_{HWDSU}	5		ns
HDATA hold after HWR* or HSEL* inactive or SMCSADR invalid	t_{HWDH}	0		ns

Notes:

[1] To achieve the above timing (t_{HWWH}) without hardware, it may be necessary to avoid two subsequent write accesses to the HOCI port (e.g. by a NOP or any other instruction between two HOCI writes), depending on the type and clock frequency of the CPU.

[2] Signal HACK active when HRD* low and HSEL* low and SMCSADR = SMCSID

[3] Signal HACK disable when HRD* high or HSEL* high or SMCSADR ≠ SMCSID

V_{CC} = 3.3 V ± 0.3V

Description	Symb.	Min.	Max.	Unit
HSEL*, HWR* and SMCSADR and HADR setup before CLK high	t_{HWSU}	7		ns
HADR, SMCSADR hold after HSEL* or HWR* high	t_{HWAH}	0		ns
HWR* pulse width high [1]	t_{HWWH}	$1 * t_{CLK} + 6$		ns
HACK low after HWR*, HSEL* active and SMCSADR valid [2]	t_{HWACKL}		23	ns
HACK high after HSEL* and HWR* and SMCSADR = SMCSID	t_{HWACKH}	$1 * t_{CLK} + 5$	$2.5 * t_{CLK} + 41$	ns
HACK disable after HWR* or HSEL* inactive or SMCSADR invalid [3]	t_{HWACKA}		24	ns
HDATA setup before HSEL* or HWR* high or SMCSADR $\neq$ SMCSID	t_{HWDSU}	7		ns
HDATA hold after HWR* or HSEL* inactive or SMCSADR invalid	t_{HWDH}	0		ns

Notes:

[1] To achieve the above timing (t_{HWWH}) without hardware, it may be necessary to avoid two subsequent write write accesses to the HOCI port (e.g. by a NOP or any other instruction between two HOCI writes), depending on the type and clock frequency of the CPU.

[2] Signal HACK active when HRD* low and HSEL* low and SMCSADR = SMCSID

[3] Signal HACK disable when HRD* high or HSEL* high or SMCSADR $\neq$ SMCSID

9.5 COMI Read
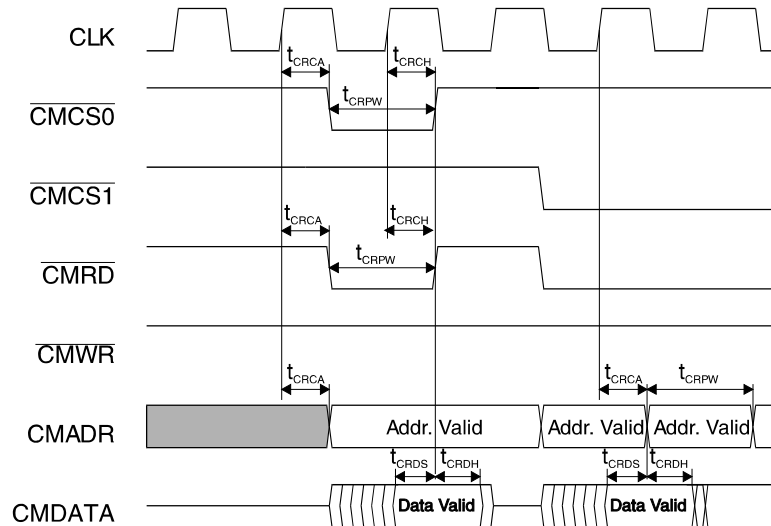
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
CMCS0*, CMCS1* and CMRD* low and CMADR valid after CLK high	t_{CRCA}		19	ns
CMCS0*, CMCS1* or CMRD* high after CLK high	t_{CRCH}		19	ns
CMCS0*, CMCS1*, CMRD*, CMADR pulse width	t_{CRPW}	t_{CLK} - 1		ns
CMDATA setup before CMCS0* or CMCS1* or CMRD* high or new address on CMADR	t_{CRDS}	4		ns
CMDATA hold after CMCS0* or CMCS1* or CMRD* high or new address on CMADR	t_{CRDH}	0		ns

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
CMCS0*, CMCS1* and CMRD* low and CMADR valid after CLK high	t_{CRCA}		24	ns
CMCS0*, CMCS1* or CMRD* high after CLK high	t_{CRCH}		25	ns
CMCS0*, CMCS1*, CMRD*, CMADR pulse width	t_{CRPW}	t_{CLK} - 1		ns
CMDATA setup before CMCS0* or CMCS1* or CMRD* high or new address on CMADR	t_{CRDS}	6		ns
CMDATA hold after CMCS0* or CMCS1* or CMRD* high or new address on CMADR	t_{CRDH}	0		ns

9.6 COMI Write
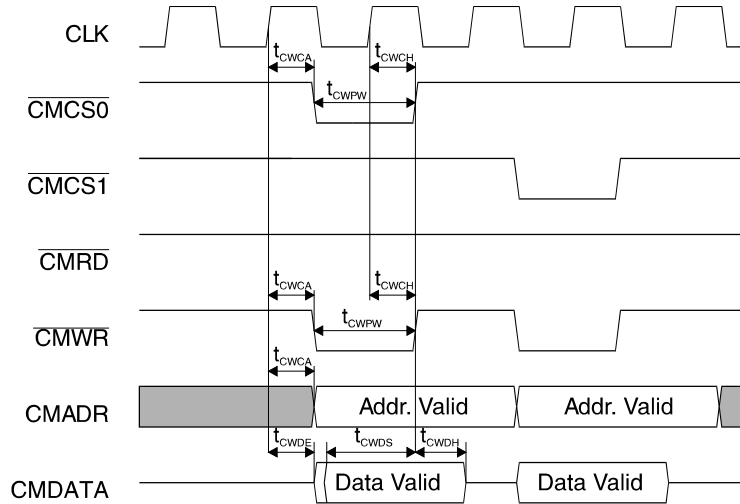
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
CMCS0*, CMCS1* and CMWR* low and CMADR valid after CLK high	t_{CWCA}		19	ns
CMCS0*, CMCS1* or CMWR* high after CLK high	t_{CWCH}		19	ns
CMCS0*, CMCS1*, CMWR* pulse width	t_{CWPW}	t_{CLK} - 1		ns
CMDATA valid after CLK high	t_{CWDE}		19	ns
CMDATA valid before CMCS0* or CMCS1* or CMWR* high	t_{CWDS}	27		ns
CMDATA hold after CMCS0* or CMCS1* or CMWR* high	t_{CWDH}		$t_{CLK}/2 + 18$	ns

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
CMCS0*, CMCS1* and CMWR* low and CMADR valid after CLK high	t_{CWCA}		24	ns
CMCS0*, CMCS1* or CMWR* high after CLK high	t_{CWCH}		25	ns
CMCS0*, CMCS1*, CMWR* pulse width	t_{CWPW}	t_{CLK} - 1		ns
CMDATA valid after CLK high	t_{CWDE}		34	ns
CMDATA valid before CMCS0* or CMCS1* or CMWR* high	t_{CWDS}	54		ns
CMDATA hold after CMCS0* or CMCS1* or CMWR* high	t_{CWDH}		$t_{CLK}/2 + 25$	ns

9.7 COMI Arbitration
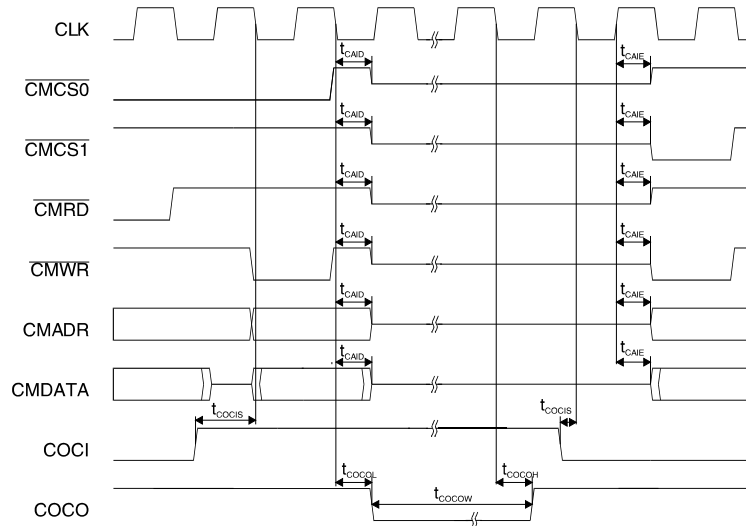
$V_{CC} = 5 \ V \pm 0.5V$

Description	Symbol	Min.	Max.	Unit
COM Interface disable after CLK low	t_{CAID}		16	ns
COM Interface enable after CLK high	t_{CAIE}		15	ns
COCI setup before CLK low	t_{COCIS}	3		ns
COCO low after CLK low	t_{COCOL}		4	ns
COCO high after CLK low	t_{COCOH}		7	ns
COCO pulse width [3]	t_{COCOW}		$N - 1 \ t_{CLK}$	ns

[3] N = content of COMI_ACR

$V_{CC} = 3.3 \ V \pm 0.3V$

Description	Symbol	Min.	Max.	Unit
COM Interface disable after CLK low	t_{CAID}		27	ns
COM Interface enable after CLK high	t_{CAIE}		24	ns
COCI setup before CLK low	t_{COCIS}	5		ns
COCO low after CLK low	t_{COCOL}		6	ns
COCO high after CLK low	t_{COCOH}		11	ns
COCO pulse width [3]	t_{COCOW}		$N - 1 \ t_{CLK}$	ns

[3] N = content of COMI_ACR

9.8 CPUR, SES, Interrupt
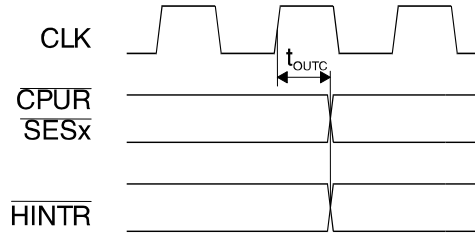
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
CPUR*, SESx*, HINTR* delay after CLK high	t_{OUTC}		20	ns

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
CPUR*, SESx*, HINTR* delay after CLK high	t_{OUTC}			ns

9.9 Links
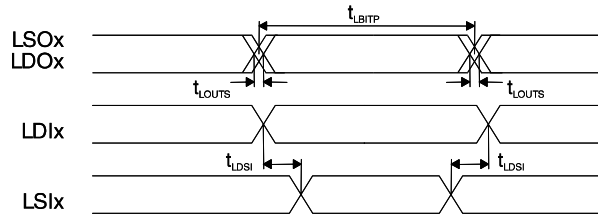
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
Bit Period	t_{LBITP}	4,5		ns
LDOx, LSOx output skew	t_{LOUTS}		0.5	ns
Data/Strobe edge separation	t_{LDSI}	2.5		ns

Note: t_{LDSI} is the minimum separation time between consecutive edges on the data and strobe inputs that the SMCS332SpW can discriminate correctly (for all speeds from 1.25 Mbps - 200 Mbps).

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
Bit Period	t_{LBITP}	9,0		ns
LDOx, LSOx output skew	t_{LOUTS}		1.0	ns
Data/Strobe edge separation	t_{LDSI}	5.5		ns

Note: t_{LDSI} is the minimum separation time between consecutive edges on the data and strobe inputs that the SMCS332SpW can discriminate correctly (for all speeds from 1.25 Mbps - 100 Mbps).

9.10 Test Port (JTAG)
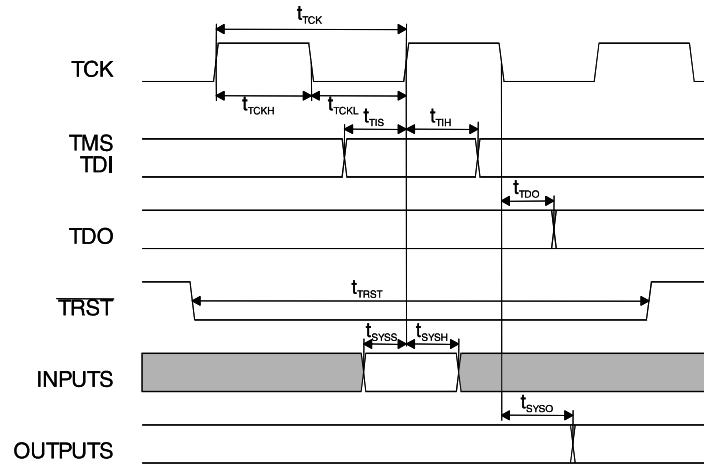
V_{CC} = 5 V ± 0.5V

Description	Symbol	Min.	Max.	Unit
TCK period	t_{TCK}	100		ns
TCK width high	t_{TCKH}	40		ns
TCK width low	t_{TCKL}	40		ns
TMS, TDI setup before TCK high	t_{TIS}	8		ns
TMS, TDI hold after TCK high	t_{TIH}	8		ns
TDO delay after TCK low	t_{TDO}		17	ns
TRST* pulse width	t_{TRST}	2 * t_{TCK}		ns
SMCS Inputs setup before TCK high	t_{SYSS}	8		ns
SMCS Inputs hold after TCK high	t_{SYSM}	8		ns
SMCS Outputs delay after TCK low	t_{SYSO}		27	ns

V_{CC} = 3.3 V ± 0.3V

Description	Symbol	Min.	Max.	Unit
TCK period	t_{TCK}	100		ns
TCK width high	t_{TCKH}	40		ns
TCK width low	t_{TCKL}	40		ns
TMS, TDI setup before TCK high	t_{TIS}	8		ns
TMS, TDI hold after TCK high	t_{TIH}	8		ns
TDO delay after TCK low	t_{TDO}		20	ns

Description	Symbol	Min.	Max.	Unit
TRST* pulse width	t_{TRST}	$2 * t_{TCK}$		ns
SMCS Inputs setup before TCK high	t_{SYSS}	10		ns
SMCS Inputs hold after TCK high	t_{SYSM}	10		ns
SMCS Outputs delay after TCK low	t_{SYSO}		31	ns

Note: The BSDL file is printed in the Annex of this document.

10 Mechanical Data

10.1 Package Dimensions
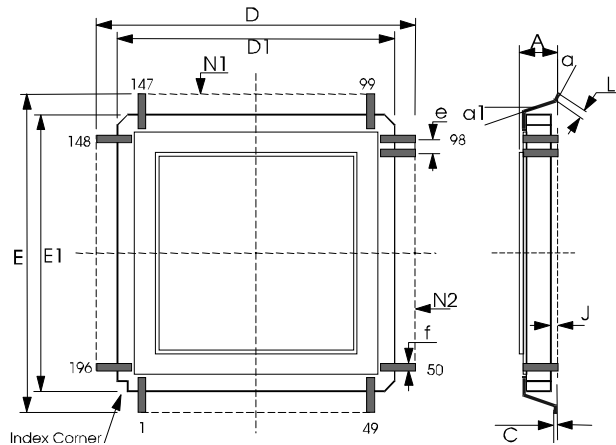
196-Pin   Ceramic Quad Flat Pack (MQFP)

SYMBOL	MILLIMETERS		INCHES	
	MIN	**MAX**	**MIN**	**MAX**
A	2.36	2.97	0.093	0.117
C	0.2 typ		0.008 typ	
D, E	38.74	39.75	1.525	1.565
D1, E1	34.03	34.54	1.340	1.360
e	0.635 typ		0.025 typ	
f	0.28 ref		0.011 ref	
J	0.15	0.30	0.006	0.012
L	0.61	1.01	0.024	0.040
a	$4° \pm 4°$			
a1	$5° \pm 5°$			
N1, N2	49 pins			

10.2 Pin Assignment

The table below lists the pins of the SMCS332SpW

Pin Number	Name	Pin Number	Name	Pin Number	Name
1	PLLOUT	67	HDATA18	133	CMDATA8
2	GND	68	HDATA19	134	VCC
3	VCC	69	HDATA20	135	GND
4	CLK	70	HDATA21	136	CMDATA9
5	RESET*	71	HDATA22	137	CMDATA10
6	CLK10	72	HDATA23	138	CMDATA11
7	HOSTBIGE	73	VCC	139	CMDATA12
8	TCK	74	GND	140	CMDATA13
9	TMS	75	HDATA24	141	CMDATA14
10	TDI	76	HDATA25	142	VCC
11	TRST*	77	HDATA26	143	GND
12	TDO	78	VCC	144	CMDATA15
13	VCC	79	GND	145	CMDATA16
14	GND	80	HDATA27	146	CMDATA17
15	HSEL*	81	HDATA28	147	CMDATA18
16	HRD*	82	HDATA29	148	CMDATA19
17	HWR*	83	VCC	149	CMDATA20
18	HACK	84	GND	150	VCC
19	HINTR*	85	HDATA30	151	GND
20	VCC	86	HDATA31	152	CMDATA21
21	GND	87	CPUR*	153	CMDATA22
22	HADR0	88	SES0*	154	CMDATA23
23	HADR1	89	SES1*	155	VCC
24	HADR2	90	SES2*	156	GND
25	HADR3	91	SES3*	157	CMDATA24
26	HADR4	92	CAM	158	CMDATA25
27	HADR5	93	COCI	159	CMDATA26
28	HADR6	94	COCO	160	VCC
29	HADR7	95	CMCS0*	161	GND
30	VCC	96	CMCS1*	162	CMDATA27
31	GND	97	VCC	163	CMDATA28
32	BOOTLINK	98	GND	164	CMDATA29

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 86 of 131

Pin Number	Name	Pin Number	Name	Pin Number	Name
33	SMCSADR0	99	CMRD*	165	CMDATA30
34	SMCSADR1	100	CMWR*	166	CMDATA31
35	SMCSADR2	101	CMADR0	167	GND
36	SMCSADR3	102	CMADR1	168	GND
37	SMCSID0	103	CMADR2	169	VCC_3VOLT
38	SMCSID1	104	CMADR3	170	GND
39	SMCSID2	105	CMADR4	171	GND
40	SMCSID3	106	VCC	172	VCC
41	VCC	107	GND	173	GND
42	GND	108	CMADR5	174	GND
43	HDATA0	109	CMADR6	175	NC
44	HDATA1	110	CMADR7	176	LDI1
45	HDATA2	111	CMADR8	177	LSI1
46	HDATA3	112	CMADR9	178	LDO1
47	HDATA4	113	CMADR10	179	LSO1
48	HDATA5	114	CMADR11	180	LDI2
49	HDATA6	115	VCC	181	LSI2
50	VCC	116	GND	182	NC
51	GND	117	CMADR12	183	VCC
52	HDATA7	118	CMADR13	184	VCC
53	HDATA8	119	CMADR14	185	VCC
54	HDATA9	120	CMADR15	186	LDO2
55	HDATA10	121	CMDATA0	187	LSO2
56	HDATA11	122	CMDATA1	188	LDI3
57	VCC	123	CMDATA2	189	LSI3
58	GND	124	VCC	190	LDO3
59	HDATA12	125	GND	191	LSO3
60	HDATA13	126	CMDATA3	192	TIME_CODE_SYNC
61	HDATA14	127	CMDATA4	193	GND
62	HDATA15	128	CMDATA5	194	GND
63	HDATA16	129	VCC	195	VCC
64	HDATA17	130	GND	196	GND
65	VCC	131	CMDATA6		
66	GND	132	CMDATA7		

11 Additional Informations

11.1 Frequently Asked Questions

Q: Is the SMCS332SpW compatible with the old SMCS332.?
A: Yes, the link standard IEEE 1355-1995 and ECSS-E-50-12A are compatible, however the old SMCS332 has a slightly different startup behavior, the SMCS332 must be started first on a link connecting with the SMCS332SpW.

Q: The nominal frequency for CLK10 is 10 MHz. Can it be different? What is the effect on the transmission rate?
A: It can be different, the transmission rate changes in direct proportion to the change of the CLK10 frequency. However, it is recommended to keep CLK10 at 10 MHz to be conforming to ECSS.

Q: What is the best way for adding a transfer rate of 5Mbps and still be able to select 10/12.5/25/50/100 Mbps ?
A: With these settings, transfers at rates down to 6.25 and 3.125 Mbps can be achieved. If the max speed to 80Mbps then the transmission speed can be selected to 5, 10, 20, 40 and 80 Mbps.

Q: What is the minimum transmission rate on an SpaceWire link?
A: SpaceWire defines the max. time between two bits with 850ns, which corresponds to 1.18 Mbit/s.

Q: Are the outputs of the SMCS332SpW TTL compatible?
A: Yes; the SMCS332SpW features CMOS outputs and TTL inputs.

Q: When the SMCS332SpW is connected to a dual port memory, can it monitor the "busy" wire of the DPRAM?
A: No. External logic is required for that. Due to the mechanism applied in the SMCS, it is not necessary to monitor the
busy wire when data is flowing via COMI.

Q: How does the COMI (COmmunication Memory Interface) works in big endian mode?
A: The COMI works in big endian mode in the same manner as it is described for the HOCI (HOst Control Interface).

Q: When in control-by-link mode, can the link used as control link be used to send data to the RAM connected to the SMCS332SpW?
A: No, another link must be used for this purpose. See section 4.4.5.

Q: In control-by-link mode, can I access the link control registers also from the links not working as control link?
A: Yes, all SMCS332SpW registers can be accessed via the control link. See section 4.4.

Q: When are the ISR bits reset?
A: This depends on the width of the host interface. After reset or in control-by-link mode, it is in 8-bit mode. To reset the ISR bits, do the following: 8-bit mode: Read address 0x04 to 0x07; 16-bit mode: Read address 0x04 and 0x06; 32-bit mode: Read address 0x04.

Q: What is the use of SMCS332SpW signals CPUR* and SES(3:0) ?
A: These signals could be used as user defined flags. Examples of possible uses are contained in the

SMCS332SpW protocol specification (see Appendix B).

Q: When using the SMCS332SpW internal link loopback test mechanism, do the link signals propagate outside the SMCS332SpW?

A: Yes, but you can avoid this by using the CHx_DSM_TSTR register bit 4: link output mute.

Q: On which datalines are bytes, received on SMCS332SpW serial link, transferred into the DPRAM through the COMI?

A: Suppose receiving 4 bytes (Byte0 to Byte3) on serial link, Byte0 being the first one.
      Note that the order of the data within one byte is the same in both little and big endian mode .
      The most right line carries the least significant and the most left line carries the most significant bit.

Little endian:

COMI data port is set to 32 Bit width:

```
datalines:  31-24       23-16       15-8        7-0

            Byte3       Byte2       Byte1       Byte0
```

COMI data port is set to 16 Bit width:

```
datalines:  31-24       23-16       15-8        7-0

            00          00          Byte1       Byte0 if CHx_COMICFG bit 7 = 0

            00          00          Byte3       Byte2

            FF          FF          Byte1       Byte0 if CHx_COMICFG bit 7 = 1
                                                      and Byte1 Bit 7='1'

            00          00          Byte3       Byte2 if CHx_COMICFG bit 7 = 1
                                                      and Byte3 Bit 7='0'
```

COMI data port is set to 8 Bit width:

```
datalines:  31-24       23-16       15-8        7-0

            00          00          00          Byte0 if CHx_COMICFG bit 7 = 0

            00          00          00          Byte1

            00          00          00          Byte2

            00          00          00          Byte3

            FF          FF          FF          Byte0 if CHx_COMICFG bit 7 = 1
                                                      and Byte0 Bit 7='1'

            00          00          00          Byte1 if CHx_COMICFG bit 7 = 1
                                                      and Byte1 Bit 7='0'
```


BIG endian:

COMI data port is set to 32 Bit width:

```
datalines:  31-24       23-16       15-8        7-0

            Byte0       Byte1       Byte2       Byte3
```

COMI data port is set to 16 Bit width:

```
datalines:  31-24       23-16       15-8        7-0

            Byte0       Byte1       00          00    if CHx_COMICFG bit 7 = 0

            Byte2       Byte3       00          00

            Byte0       Byte1       FF          FF    if CHx_COMICFG bit 7 = 1
                                                      and Byte1 Bit 7='1'

            Byte2       Byte3       00          00    if CHx_COMICFG bit 7 = 1
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 89 of 131

```
                                                and Byte3 Bit 7='0'
```

COMI data port is set to 8 Bit width:

```
datalines:  31-24       23-16       15-8        7-0
            Byte0       00          00          00    if CHx_COMICFG bit 7 = 0
            Byte1       00          00          00
            Byte2       00          00          00
            Byte3       00          00          00
            Byte0       00          00          00    if  CHx_COMICFG  bit  7  =  1
                                                      and Byte0 Bit 7='0'
            Byte1       FF          FF          FF    if  CHx_COMICFG  bit  7  =  1
                                                      and Byte1 Bit 7='1'
```

11.2 BSDL File for the SMCS332SpW

Below is the BSDL file required for using the JTAG port of the SMCS332SpW.

```
-- BSDL for SMCS332SpW
-- Uses HP's BSDL format and compiles correctly using HP's
-- parser or compiler of JTAG Technologies


-- Author:
-- date:



entity SMCS is

    generic (PHYSICAL_PIN_MAP : string := "UNDEFINED");

    port (
        BOOTLINK : in bit;
        BYPPLL : in bit;
        CAM : in bit;
        CLK : in bit;
        CLK10 : in bit;
        COCI : in bit;
        HADR : in bit_vector(0 to 7);
        HOSTBIGE : in bit;
        HRD : in bit;
        HSEL : in bit;
        HWR : in bit;
        LDI1 : in bit;
        LDI2 : in bit;
        LDI3 : in bit;
        LSI1 : in bit;
        LSI2 : in bit;
        LSI3 : in bit;
        RESET : in bit;
        SMCSADR0 : in bit;
        SMCSADR1 : in bit;
        SMCSADR2 : in bit;
        SMCSADR3 : in bit;
        SMCSID0 : in bit;
        SMCSID1 : in bit;
        SMCSID2 : in bit;
        SMCSID3 : in bit;
        TIME_C_SY : in bit;
        VCC_3VOLT : in bit;
        TCK : in bit;
        TDI : in bit;
```

```
        TDO : out bit;
        TMS : in bit;
        TRST : in bit;
        CMADR : out bit_vector(0 to 15);
        CMCS0 : out bit;
        CMCS1 : out bit;
        CMRD : out bit;
        CMWR : out bit;
        COCO : out bit;
        CPUR : out bit;
        HACK : out bit;
        HINTR : out bit;
        LDO1 : out bit;
        LDO2 : out bit;
        LDO3 : out bit;
        LSO1 : out bit;
        LSO2 : out bit;
        LSO3 : out bit;
        SES0 : out bit;
        SES1 : out bit;
        SES2 : out bit;
        SES3 : out bit;
        TEST : out bit;
        CMDATA : inout bit_vector(0 to 31);
        HDATA : inout bit_vector(0 to 31);
        GND : linkage bit_vector(0 to 28);
        VDD : linkage bit_vector(0 to 25);
        NC : linkage bit_vector(0 to 1);


use STD_1149_1_1990.all;


attribute PIN_MAP of SMCS : entity is PHYSICAL_PIN_MAP;


constant PGA_PACKAGE : PIN_MAP_STRING :=  -- QFP_Package ?
    "BOOTLINK:32," &
    "BYPPLL:168," &
    "CAM:92," &
    "CLK:4," &
    "CLK10:6," &
    "COCI:93," &
    "HADR:(22,23,24,25,26,27,28,29)," &
    "HOSTBIGE:7," &
    "HRD:16," &
    "HSEL:15," &
    "HWR:17," &
    "LDI1:176," &
    "LDI2:180," &
    "LDI3:188," &
    "LSI1:177," &
```

```
"LSI2:181," &
"LSI3:189," &
"RESET:5," &
"SMCSADR0:33," &
"SMCSADR1:34," &
"SMCSADR2:35," &
"SMCSADR3:36," &
"SMCSID0:37," &
"SMCSID1:38," &
"SMCSID2:39," &
"SMCSID3:40," &
"TIME_CODE_SYNC:192," &
"VCC_3VOLT:169," &
"TCK:8," &
"TDI:10," &
"TDO:12," &
"TMS:9," &
"TRST:11," &
"CMADR:(101,102,103,104,105,108,109,110,111,112,113,114,117,118,119,120)," &
"CMCS0:95," &
"CMCS1:96," &
"CMRD:99," &
"CMWR:100," &
"COCO:94," &
"CPUR:87," &
"HACK:18," &
"HINTR:19," &
"LDO1:178," &
"LDO2:186," &
"LDO3:190," &
"LSO1:179," &
"LSO2:187," &
"LSO3:191," &
"SES0:88," &
"SES1:89," &
"SES2:90," &
"SES3:91," &
"TEST:182," &
"CMDATA:(121,122,123,126,127,128,131,132,133,136,137,138,139,140,141," &
        "144,145,146,147,148,149,152,153,154,157,158,159,162,163,164," &
        "165,166)," &
"HDATA:(43,44,45,46,47,48,49,52,53,54,55,56,59,60,61,62,63,64,67,68,69," &
        "70,71,72,75,76,77,80,81,82,85,86)," &
"GND:(2,14,21,31,42,51,58,66,74,79,84,98,107,116,125,130,135,143,151,156," &
      "161,167,170,171,173,174,193,194,196)," &
"VDD:(3,13,20,30,41,50,57,65,73,78,83,97,106,115,124,129,134,142,150," &
      "155,160,172,183,184,185,195)," &
"NC:(1,175)";
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 93 of 131

```
  attribute TAP_SCAN_IN    of TDI : signal is true;
  attribute TAP_SCAN_MODE  of TMS : signal is true;
  attribute TAP_SCAN_OUT   of TDO : signal is true;
  attribute TAP_SCAN_RESET of TRST : signal is true;
-- unconfirmed TCK Fmax
  attribute TAP_SCAN_CLOCK of TCK : signal is (10.0e6, BOTH);


  attribute INSTRUCTION_LENGTH of SMCS : entity is 3;


  attribute INSTRUCTION_OPCODE of SMCS : entity is
     "BYPASS        (111,110,101,100)," &
     "EXTEST        (000)," &
     "SAMPLE        (001)," &
     "IDCODE        (010)," &
     "HIGHZ      (011)";


  attribute INSTRUCTION_CAPTURE of SMCS : entity is
        "101";


  attribute INSTRUCTION_DISABLE of SMCS : entity is
        "HIGHZ";


  attribute IDCODE_REGISTER of SMCS : entity is
     "0010" &           -- Version
     "0100010001010011" &   -- Part number
     "00001011000" &        -- ID of manufacturer
     "1";                   -- required by IEEE Std 1149.1-1990


  attribute REGISTER_ACCESS of SMCS : entity is
--       "BSREG (EXTEST, SAMPLE)," &
     "BOUNDARY (EXTEST, SAMPLE),"&
--       "IDREG (IDCODE)," &
     "BYPASS (BYPASS, HIGHZ)";
--       "BPREG (BYPASS, HIGHZ)";


  attribute BOUNDARY_CELLS of SMCS : entity is
      "BC_1";
  -- BC_1: output, control; BC_1: input;


  attribute BOUNDARY_LENGTH of SMCS : entity is 233;


  attribute BOUNDARY_REGISTER of SMCS : entity is

-- num   cell   port       func   safe  [ccell  disval  rslt]
  " 232 (BC_1, HOSTBIGE,   input,   X)," &
  " 231 (BC_1, CLK10,      input,   X)," &
  " 230 (BC_1, RESET,      input,   X)," &
  " 229 (BC_1, CLK,        input,   X)," &
  " 228 (BC_1, BYPPLL,     input,   X)," &
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 94 of 131

```
"  227 (BC_1, TIME_C_SY,  input, X)," &
"  226 (BC_1, LSO3,       output2, X)," &  -- output2 for internal tristate?
"  225 (BC_1, LDO3,       output2, X)," &
"  224 (BC_1, LSI3,       input,  X)," &
"  223 (BC_1, LDI3,       input,  X)," &
"  222 (BC_1, LSO2,       output2, X)," &
"  221 (BC_1, LDO2,       output2, X)," &
"  220 (BC_1, TEST,       output2, X)," &
"  219 (BC_1, LSI2,       input,  X)," &
"  218 (BC_1, LDI2,       input,  X)," &
"  217 (BC_1, LSO1,       output2, X)," &
"  216 (BC_1, LDO1,       output2, X)," &
"  215 (BC_1, LSI1,       input,  X)," &
"  214 (BC_1, LDI1,       input,  X)," &
"  213 (BC_1, VCC_3VOLT,  input, X)," &
"  212 (BC_1, *         , control, 0)," &  -- COMI Data Output Enable
"  211 (BC_1, CMDATA(31), output3, X, 212, 0,     Z)," &
"  210 (BC_1, CMDATA(31), input,  X)," &
"  209 (BC_1, CMDATA(30), output3, X, 212, 0,     Z)," &
"  208 (BC_1, CMDATA(30), input,  X)," &
"  207 (BC_1, CMDATA(29), output3, X, 212, 0,     Z)," &
"  206 (BC_1, CMDATA(29), input,  X)," &
"  205 (BC_1, CMDATA(28), output3, X, 212, 0,     Z)," &
"  204 (BC_1, CMDATA(28), input,  X)," &
"  203 (BC_1, CMDATA(27), output3, X, 212, 0,     Z)," &
"  202 (BC_1, CMDATA(27), input,  X)," &
"  201 (BC_1, CMDATA(26), output3, X, 212, 0,     Z)," &
"  200 (BC_1, CMDATA(26), input,  X)," &
"  199 (BC_1, CMDATA(25), output3, X, 212, 0,     Z)," &
"  198 (BC_1, CMDATA(25), input,  X)," &
"  197 (BC_1, CMDATA(24), output3, X, 212, 0,     Z)," &
"  196 (BC_1, CMDATA(24), input,  X)," &
"  195 (BC_1, CMDATA(23), output3, X, 212, 0,     Z)," &
"  194 (BC_1, CMDATA(23), input,  X)," &
"  193 (BC_1, CMDATA(22), output3, X, 212, 0,     Z)," &
"  192 (BC_1, CMDATA(22), input,  X)," &
"  191 (BC_1, CMDATA(21), output3, X, 212, 0,     Z)," &
"  190 (BC_1, CMDATA(21), input,  X)," &
"  189 (BC_1, CMDATA(20), output3, X, 212, 0,     Z)," &
"  188 (BC_1, CMDATA(20), input,  X)," &
"  187 (BC_1, CMDATA(19), output3, X, 212, 0,     Z)," &
"  186 (BC_1, CMDATA(19), input,  X)," &
"  185 (BC_1, CMDATA(18), output3, X, 212, 0,     Z)," &
"  184 (BC_1, CMDATA(18), input,  X)," &
"  183 (BC_1, CMDATA(17), output3, X, 212, 0,     Z)," &
"  182 (BC_1, CMDATA(17), input,  X)," &
"  181 (BC_1, CMDATA(16), output3, X, 212, 0,     Z)," &
"  180 (BC_1, CMDATA(16), input,  X)," &
"  179 (BC_1, CMDATA(15), output3, X, 212, 0,     Z)," &
```

```
"  178 (BC_1, CMDATA(15), input,   X)," &
"  177 (BC_1, CMDATA(14), output3, X, 212, 0,     Z)," &
"  176 (BC_1, CMDATA(14), input,   X)," &
"  175 (BC_1, CMDATA(13), output3, X, 212, 0,     Z)," &
"  174 (BC_1, CMDATA(13), input,   X)," &
"  173 (BC_1, CMDATA(12), output3, X, 212, 0,     Z)," &
"  172 (BC_1, CMDATA(12), input,   X)," &
"  171 (BC_1, CMDATA(11), output3, X, 212, 0,     Z)," &
"  170 (BC_1, CMDATA(11), input,   X)," &
"  169 (BC_1, CMDATA(10), output3, X, 212, 0,     Z)," &
"  168 (BC_1, CMDATA(10), input,   X)," &
"  167 (BC_1, CMDATA(9), output3, X, 212,  0,     Z)," &
"  166 (BC_1, CMDATA(9), input,   X)," &
"  165 (BC_1, CMDATA(8), output3, X, 212,  0,     Z)," &
"  164 (BC_1, CMDATA(8), input,   X)," &
"  163 (BC_1, CMDATA(7), output3, X, 212,  0,     Z)," &
"  162 (BC_1, CMDATA(7), input,   X)," &
"  161 (BC_1, CMDATA(6), output3, X, 212,  0,     Z)," &
"  160 (BC_1, CMDATA(6), input,   X)," &
"  159 (BC_1, CMDATA(5), output3, X, 212,  0,     Z)," &
"  158 (BC_1, CMDATA(5), input,   X)," &
"  157 (BC_1, CMDATA(4), output3, X, 212,  0,     Z)," &
"  156 (BC_1, CMDATA(4), input,   X)," &
"  155 (BC_1, CMDATA(3), output3, X, 212,  0,     Z)," &
"  154 (BC_1, CMDATA(3), input,   X)," &
"  153 (BC_1, CMDATA(2), output3, X, 212,  0,     Z)," &
"  152 (BC_1, CMDATA(2), input,   X)," &
"  151 (BC_1, CMDATA(1), output3, X, 212,  0,     Z)," &
"  150 (BC_1, CMDATA(1), input,   X)," &
"  149 (BC_1, CMDATA(0), output3, X, 212,  0,     Z)," &
"  148 (BC_1, CMDATA(0), input,   X)," &
"  147 (BC_1, *        , control, 0)," & -- COMI Adr Output Enable
"  146 (BC_1, CMADR(15), output3, X, 147,  0,     Z)," &
"  145 (BC_1, CMADR(14), output3, X, 147,  0,     Z)," &
"  144 (BC_1, CMADR(13), output3, X, 147,  0,     Z)," &
"  143 (BC_1, CMADR(12), output3, X, 147,  0,     Z)," &
"  142 (BC_1, CMADR(11), output3, X, 147,  0,     Z)," &
"  141 (BC_1, CMADR(10), output3, X, 147,  0,     Z)," &
"  140 (BC_1, CMADR(9),  output3, X, 147,  0,     Z)," &
"  139 (BC_1, CMADR(8),  output3, X, 147,  0,     Z)," &
"  138 (BC_1, CMADR(7),  output3, X, 147,  0,     Z)," &
"  137 (BC_1, CMADR(6),  output3, X, 147,  0,     Z)," &
"  136 (BC_1, CMADR(5),  output3, X, 147,  0,     Z)," &
"  135 (BC_1, CMADR(4),  output3, X, 147,  0,     Z)," &
"  134 (BC_1, CMADR(3),  output3, X, 147,  0,     Z)," &
"  133 (BC_1, CMADR(2),  output3, X, 147,  0,     Z)," &
"  132 (BC_1, CMADR(1),  output3, X, 147,  0,     Z)," &
"  131 (BC_1, CMADR(0),  output3, X, 147,  0,     Z)," &
"  130 (BC_1, CMWR,      output2, X)," &
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 96 of 131

```
"  129 (BC_1, CMRD,       output2, X)," &
"  128 (BC_1, CMCS1,      output2, X)," &
"  127 (BC_1, CMCS0,      output2, X)," &
"  126 (BC_1, COCO,     output2, X)," &  -- internal tristate?
"  125 (BC_1, COCI,       input,   X)," &
"  124 (BC_1, CAM,      input,   X)," &
"  123 (BC_1, SES3,     output2, X)," &
"  122 (BC_1, SES2,     output2, X)," &
"  121 (BC_1, SES1,     output2, X)," &
"  120 (BC_1, SES0,     output2, X)," &
"  119 (BC_1, CPUR,     output2, X)," &
"  118 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  117 (BC_1, HDATA(31), output3, X,       118,    0,      Z)," &
"  116 (BC_1, HDATA(31), input,   X)," &
"  115 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  114 (BC_1, HDATA(30), output3, X,       115,    0,      Z)," &
"  113 (BC_1, HDATA(30), input,   X)," &
"  112 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  111 (BC_1, HDATA(29), output3, X,       112,    0,      Z)," &
"  110 (BC_1, HDATA(29), input,   X)," &
"  109 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  108 (BC_1, HDATA(28), output3, X,       109,    0,      Z)," &
"  107 (BC_1, HDATA(28), input,   X)," &
"  106 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  105 (BC_1, HDATA(27), output3, X,       106,    0,      Z)," &
"  104 (BC_1, HDATA(27), input,   X)," &
"  103 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  102 (BC_1, HDATA(26), output3, X,       103,    0,      Z)," &
"  101 (BC_1, HDATA(26), input,   X)," &
"  100 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  99 (BC_1, HDATA(25), output3, X,100,    0,      Z)," &
"  98 (BC_1, HDATA(25), input,   X)," &
"  97 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  96 (BC_1, HDATA(24), output3, X, 97,    0,      Z)," &
"  95 (BC_1, HDATA(24), input,   X)," &
"  94 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  93 (BC_1, HDATA(23), output3, X, 94,    0,      Z)," &
"  92 (BC_1, HDATA(23), input,   X)," &
"  91 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  90 (BC_1, HDATA(22), output3, X, 91,    0,      Z)," &
"  89 (BC_1, HDATA(22), input,   X)," &
"  88 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  87 (BC_1, HDATA(21), output3, X, 88,    0,      Z)," &
"  86 (BC_1, HDATA(21), input,   X)," &
"  85 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  84 (BC_1, HDATA(20), output3, X, 85,    0,      Z)," &
"  83 (BC_1, HDATA(20), input,   X)," &
"  82 (BC_1, *    ,     control, 0)," & -- HOCI Data Output Enable
"  81 (BC_1, HDATA(19), output3, X, 82,    0,      Z)," &
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 97 of 131

```
"  80 (BC_1, HDATA(19), input,   X)," &
"  79 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  78 (BC_1, HDATA(18), output3, X, 79,    0,     Z)," &
"  77 (BC_1, HDATA(18), input,   X)," &
"  76 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  75 (BC_1, HDATA(17), output3, X, 76,    0,     Z)," &
"  74 (BC_1, HDATA(17), input,   X)," &
"  73 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  72 (BC_1, HDATA(16), output3, X, 73,    0,     Z)," &
"  71 (BC_1, HDATA(16), input,   X)," &
"  70 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  69 (BC_1, HDATA(15), output3, X, 70,    0,     Z)," &
"  68 (BC_1, HDATA(15), input,   X)," &
"  67 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  66 (BC_1, HDATA(14), output3, X, 67,    0,     Z)," &
"  65 (BC_1, HDATA(14), input,   X)," &
"  64 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  63 (BC_1, HDATA(13), output3, X, 64,    0,     Z)," &
"  62 (BC_1, HDATA(13), input,   X)," &
"  61 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  60 (BC_1, HDATA(12), output3, X, 61,    0,     Z)," &
"  59 (BC_1, HDATA(12), input,   X)," &
"  58 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  57 (BC_1, HDATA(11), output3, X, 58,    0,     Z)," &
"  56 (BC_1, HDATA(11), input,   X)," &
"  55 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  54 (BC_1, HDATA(10), output3, X, 55,    0,     Z)," &
"  53 (BC_1, HDATA(10), input,   X)," &
"  52 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  51 (BC_1, HDATA(9),  output3, X, 52,    0,     Z)," &
"  50 (BC_1, HDATA(9),  input,   X)," &
"  49 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  48 (BC_1, HDATA(8),  output3, X, 49,    0,     Z)," &
"  47 (BC_1, HDATA(8),  input,   X)," &
"  46 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  45 (BC_1, HDATA(7),  output3, X, 46,    0,     Z)," &
"  44 (BC_1, HDATA(7),  input,   X)," &
"  43 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  42 (BC_1, HDATA(6),  output3, X, 43,    0,     Z)," &
"  41 (BC_1, HDATA(6),  input,   X)," &
"  40 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  39 (BC_1, HDATA(5),  output3, X, 40,    0,     Z)," &
"  38 (BC_1, HDATA(5),  input,   X)," &
"  37 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  36 (BC_1, HDATA(4),  output3, X, 37,    0,     Z)," &
"  35 (BC_1, HDATA(4),  input,   X)," &
"  34 (BC_1, *     ,    control, 0)," & -- HOCI Data Output Enable
"  33 (BC_1, HDATA(3),  output3, X, 34,    0,     Z)," &
"  32 (BC_1, HDATA(3),  input,   X)," &
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 98 of 131

```
"  31 (BC_1, *     ,      control, 0)," & -- HOCI Data Output Enable
"  30 (BC_1, HDATA(2),  output3, X,31,    0,     Z)," &
"  29 (BC_1, HDATA(2),  input,   X)," &
"  28 (BC_1, *     ,      control, 0)," & -- HOCI Data Output Enable
"  27 (BC_1, HDATA(1),  output3, X,28,    0,     Z)," &
"  26 (BC_1, HDATA(1),  input,   X)," &
"  25 (BC_1, *     ,      control, 0)," & -- HOCI Data Output Enable
"  24 (BC_1, HDATA(0),  output3, X,25,    0,     Z)," &
"  23 (BC_1, HDATA(0),  input,   X)," &
"  22 (BC_1, SMCSID3, input,    X)," &
"  21 (BC_1, SMCSID2, input,    X)," &
"  20 (BC_1, SMCSID1, input,    X)," &
"  19 (BC_1, SMCSID0, input,    X)," &
"  18 (BC_1, SMCSADR3,  input,    X)," &
"  17 (BC_1, SMCSADR2,  input,    X)," &
"  16 (BC_1, SMCSADR1,  input,    X)," &
"  15 (BC_1, SMCSADR0,  input,    X)," &
"  14 (BC_1, BOOTLINK,  input,    X)," &
"  13 (BC_1, HADR(7),   input,    X)," &
"  12 (BC_1, HADR(6),   input,    X)," &
"  11 (BC_1, HADR(5),   input,    X)," &
"  10 (BC_1, HADR(4),   input,    X)," &
"  9 (BC_1, HADR(3),   input,    X)," &
"  8 (BC_1, HADR(2),   input,    X)," &
"  7 (BC_1, HADR(1),   input,    X)," &
"  6 (BC_1, HADR(0),   input,    X)," &
"  5 (BC_1, HINTR,     output2, X)," &
"  4 (BC_1, *   ,      control, 0)," & -- HACK Enable
"  3 (BC_1, HACK,      output3, X, 4,     0,     Z)," &
"  2 (BC_1, HWR,       input,    X)," &
"  1 (BC_1, HRD,       input,    X)," &
"  0 (BC_1, HSEL,      input,    X)";


end SMCS;* JTAG Technologies B.V.

* smcs.dsh created by BSDL90 version 2.2 date 07/05/1998 time 08:52:23


EBST



COMPONENT "SMCS"



STANDARD         : IEEE_1149.1_1990



PIN_LIST
**********************************
* PIN    PIN         PIN    PIN     *
* NR     NAME        TYPE   FAMILY  *
```

```
*********************************

  "32"  "BOOTLINK"  IN    -

  "168" "BYPPLL"    IN    -

  "92"  "CAM"       IN    -

  "4"   "CLK"       IN    -

  "6"   "CLK10"     IN    -

  "93"  "COCI"      IN    -

  "22"  "HADR0"     IN    -

  "23"  "HADR1"     IN    -

  "24"  "HADR2"     IN    -

  "25"  "HADR3"     IN    -

  "26"  "HADR4"     IN    -

  "27"  "HADR5"     IN    -

  "28"  "HADR6"     IN    -

  "29"  "HADR7"     IN    -

  "7"   "HOSTBIGE"  IN    -

  "16"  "HRD"       IN    -

  "15"  "HSEL"      IN    -

  "17"  "HWR"       IN    -

  "176" "LDI1"      IN    -

  "180" "LDI2"      IN    -

  "188" "LDI3"      IN    -

  "177" "LSI1"      IN    -

  "181" "LSI2"      IN    -

  "189" "LSI3"      IN    -

  "5"   "RESET"     IN    -

  "33"  "SMCSADR0"  IN    -

  "34"  "SMCSADR1"  IN    -

  "35"  "SMCSADR2"  IN    -

  "36"  "SMCSADR3"  IN    -

  "37"  "SMCSID0"   IN    -

  "38"  "SMCSID1"   IN    -

  "39"  "SMCSID2"   IN    -

  "40"  "SMCSID3"   IN    -

  "169" "VCC_3VOLT" IN    -

  "192" "TIME_C_S"  IN    -

  "8"   "TCK"       TCK   -

  "10"  "TDI"       TDI   -

  "12"  "TDO"       TDO   -

  "9"   "TMS"       TMS   -

  "11"  "TRST"      TRST  -

  "101" "CMADR0"    OUTZ  -

  "102" "CMADR1"    OUTZ  -

  "103" "CMADR2"    OUTZ  -

  "104" "CMADR3"    OUTZ  -

  "105" "CMADR4"    OUTZ  -

  "108" "CMADR5"    OUTZ  -

  "109" "CMADR6"    OUTZ  -

  "110" "CMADR7"    OUTZ  -
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 100 of 131

```
"111"  "CMADR8"    OUTZ  -
"112"  "CMADR9"    OUTZ  -
"113"  "CMADR10"   OUTZ  -
"114"  "CMADR11"   OUTZ  -
"117"  "CMADR12"   OUTZ  -
"118"  "CMADR13"   OUTZ  -
"119"  "CMADR14"   OUTZ  -
"120"  "CMADR15"   OUTZ  -
"95"   "CMCS0"     OUT   -
"96"   "CMCS1"     OUT   -
"99"   "CMRD"      OUT   -
"100"  "CMWR"      OUT   -
"94"   "COCO"      OUT   -
"87"   "CPUR"      OUT   -
"18"   "HACK"      OUTZ  -
"19"   "HINTR"     OUT   -
"178"  "LDO1"      OUT   -
"186"  "LDO2"      OUT   -
"190"  "LDO3"      OUT   -
"179"  "LSO1"      OUT   -
"187"  "LSO2"      OUT   -
"191"  "LSO3"      OUT   -
"182"  "TEST"      OUT   -
"88"   "SES0"      OUT   -
"89"   "SES1"      OUT   -
"90"   "SES2"      OUT   -
"91"   "SES3"      OUT   -
"121"  "CMDATA0"   IO    -
"122"  "CMDATA1"   IO    -
"123"  "CMDATA2"   IO    -
"126"  "CMDATA3"   IO    -
"127"  "CMDATA4"   IO    -
"128"  "CMDATA5"   IO    -
"131"  "CMDATA6"   IO    -
"132"  "CMDATA7"   IO    -
"133"  "CMDATA8"   IO    -
"136"  "CMDATA9"   IO    -
"137"  "CMDATA10"  IO    -
"138"  "CMDATA11"  IO    -
"139"  "CMDATA12"  IO    -
"140"  "CMDATA13"  IO    -
"141"  "CMDATA14"  IO    -
"144"  "CMDATA15"  IO    -
"145"  "CMDATA16"  IO    -
"146"  "CMDATA17"  IO    -
"147"  "CMDATA18"  IO    -
"148"  "CMDATA19"  IO    -
"149"  "CMDATA20"  IO    -
"152"  "CMDATA21"  IO    -
```

```
"153"  "CMDATA22"  IO    -
"154"  "CMDATA23"  IO    -
"157"  "CMDATA24"  IO    -
"158"  "CMDATA25"  IO    -
"159"  "CMDATA26"  IO    -
"162"  "CMDATA27"  IO    -
"163"  "CMDATA28"  IO    -
"164"  "CMDATA29"  IO    -
"165"  "CMDATA30"  IO    -
"166"  "CMDATA31"  IO    -
"43"   "HDATA0"    IO    -
"44"   "HDATA1"    IO    -
"45"   "HDATA2"    IO    -
"46"   "HDATA3"    IO    -
"47"   "HDATA4"    IO    -
"48"   "HDATA5"    IO    -
"49"   "HDATA6"    IO    -
"52"   "HDATA7"    IO    -
"53"   "HDATA8"    IO    -
"54"   "HDATA9"    IO    -
"55"   "HDATA10"   IO    -
"56"   "HDATA11"   IO    -
"59"   "HDATA12"   IO    -
"60"   "HDATA13"   IO    -
"61"   "HDATA14"   IO    -
"62"   "HDATA15"   IO    -
"63"   "HDATA16"   IO    -
"64"   "HDATA17"   IO    -
"67"   "HDATA18"   IO    -
"68"   "HDATA19"   IO    -
"69"   "HDATA20"   IO    -
"70"   "HDATA21"   IO    -
"71"   "HDATA22"   IO    -
"72"   "HDATA23"   IO    -
"75"   "HDATA24"   IO    -
"76"   "HDATA25"   IO    -
"77"   "HDATA26"   IO    -
"80"   "HDATA27"   IO    -
"81"   "HDATA28"   IO    -
"82"   "HDATA29"   IO    -
"85"   "HDATA30"   IO    -
"86"   "HDATA31"   IO    -
"2"    "GND0"      GND   -
"14"   "GND1"      GND   -
"21"   "GND2"      GND   -
"31"   "GND3"      GND   -
"42"   "GND4"      GND   -
"51"   "GND5"      GND   -
"58"   "GND6"      GND   -
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 102 of 131

```
"66"   "GND7"    GND    -
"74"   "GND8"    GND    -
"79"   "GND9"    GND    -
"84"   "GND10"   GND    -
"98"   "GND11"   GND    -
"107"  "GND12"   GND    -
"116"  "GND13"   GND    -
"125"  "GND14"   GND    -
"130"  "GND15"   GND    -
"135"  "GND16"   GND    -
"143"  "GND17"   GND    -
"151"  "GND18"   GND    -
"156"  "GND19"   GND    -
"161"  "GND20"   GND    -
"167"  "GND21"   GND    -
"170"  "GND22"   GND    -
"171"  "GND23"   GND    -
"173"  "GND24"   GND    -
"174"  "GND25"   GND    -
"193"  "GND26"   GND    -
"194"  "GND27"   GND    -
"196"  "GND28"   GND    -
"3"    "VDD0"    PWR    -
"13"   "VDD1"    PWR    -
"20"   "VDD2"    PWR    -
"30"   "VDD3"    PWR    -
"41"   "VDD4"    PWR    -
"50"   "VDD5"    PWR    -
"57"   "VDD6"    PWR    -
"65"   "VDD7"    PWR    -
"73"   "VDD8"    PWR    -
"78"   "VDD9"    PWR    -
"83"   "VDD10"   PWR    -
"97"   "VDD11"   PWR    -
"106"  "VDD12"   PWR    -
"115"  "VDD13"   PWR    -
"124"  "VDD14"   PWR    -
"129"  "VDD15"   PWR    -
"134"  "VDD16"   PWR    -
"142"  "VDD17"   PWR    -
"150"  "VDD18"   PWR    -
"155"  "VDD19"   PWR    -
"160"  "VDD20"   PWR    -
"172"  "VDD21"   PWR    -
"183"  "VDD22"   PWR    -
"184"  "VDD23"   PWR    -
"185"  "VDD24"   PWR    -
"195"  "VDD25"   PWR    -
"1"    "NC0"     PWR    -
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 103 of 131

```
  "175" "NC1"        PWR    -
  "182" "NC2"        PWR    -
**********************************
END PIN_LIST



INSTRUCTION_REGISTER instr


PURPOSE            : "select register and BST mode"
LENGTH             : 3
CAPTURE            : "101"


INSTRUCTION_LIST
******************************************************
* INSTR          INSTR    SELECTED BST      ACTIVE   *
* NAME           VALUE    REGISTER MODE     REGISTER *
******************************************************
  BYPASS         "111"    BYPASS   SAMPLE   BYPASS
  BYPASS         "110"    BYPASS   SAMPLE   BYPASS
  BYPASS         "101"    BYPASS   SAMPLE   BYPASS
  BYPASS         "100"    BYPASS   SAMPLE   BYPASS
  EXTEST         "000"    BOUNSCAN EXTERNAL BOUNSCAN
  SAMPLE_PRELOAD "001"    BOUNSCAN SAMPLE   BOUNSCAN
  IDCODE         "010"    IDENT    SAMPLE   IDENT
  HIGHZ          "011"    BYPASS   -        BYPASS
******************************************************
END INSTRUCTION_LIST


END INSTRUCTION_REGISTER



CELL BC_1_INPUT


DATA_INPUTS        : PI1
DATA_OUTPUTS       : PO1


MODE_LIST
************************************************
* MODE     SELECT  CAPTURE  PO1              *
************************************************
  EXTERNAL -       PI1      UPD
  SAMPLE   -       PI1      PI1
  INTERNAL -       PI1      UPD
************************************************
END MODE_LIST


END CELL
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 104 of 131

```
BYPASS_REGISTER BYPASS


PURPOSE             : "short cut"
LENGTH              : 1
CAPTURE             : "0"


END BYPASS_REGISTER



BOUNDARY_SCAN_REGISTER BOUNSCAN


PURPOSE             : "scan line along boundary"
LENGTH              : 233


CHAIN_LIST
*******************************************************************************
* CHAIN CELL           PIN        DATA    DATA    SELECT CONTROL DISABLE *
* POS   NAME           NAME       INPUT   OUTPUT  POS    POS     VALUE   *
*******************************************************************************
  232   BC_1_INPUT     "HOSTBIGE"  "7"     -      -      -       -
  231   BC_1_INPUT     "CLK10"     "6"     -      -      -       -
  230   BC_1_INPUT     "RESET"     "5"     -      -      -       -
  229   BC_1_INPUT     "CLK"       "4"     -      -      -       -
  228   BC_1_INPUT     "BYPPLL"    "168"   -      -      -       -
  227   BC_1_INPUT     "TIME_C_S"  -      "192"   -      -       -
  226   BC_1_INPUT     "LSO3"      -      "191"   -      -       -
  225   BC_1_INPUT     "LDO3"      -      "190"   -      -       -
  224   BC_1_INPUT     "LSI3"     "189"    -      -      -       -
  223   BC_1_INPUT     "LDI3"     "188"    -      -      -       -
  222   BC_1_INPUT     "LSO2"      -      "187"   -      -       -
  221   BC_1_INPUT     "LDO2"      -      "186"   -      -       -
  220   BC_1_INPUT     "TEST"      -      "182"   -      -       -
  219   BC_1_INPUT     "LSI2"     "181"    -      -      -       -
  218   BC_1_INPUT     "LDI2"     "180"    -      -      -       -
  217   BC_1_INPUT     "LSO1"      -      "179"   -      -       -
  216   BC_1_INPUT     "LDO1"      -      "178"   -      -       -
  215   BC_1_INPUT     "LSI1"     "177"    -      -      -       -
  214   BC_1_INPUT     "LDI1"     "176"    -      -      -       -
  213   BC_1_INPUT     "VCC_3VOLT" "169"   -      -      -       -
  212   BC_1_INPUT     "-"         -       -      -      -       -
  211   BC_1_INPUT     "CMDATA31"  -      "166"   -      212     0
  210   BC_1_INPUT     "CMDATA31" "166"    -      -      -       -
  209   BC_1_INPUT     "CMDATA30"  -      "165"   -      212     0
  208   BC_1_INPUT     "CMDATA30" "165"    -      -      -       -
  207   BC_1_INPUT     "CMDATA29"  -      "164"   -      212     0
  206   BC_1_INPUT     "CMDATA29" "164"    -      -      -       -
  205   BC_1_INPUT     "CMDATA28"  -      "163"   -      212     0
  204   BC_1_INPUT     "CMDATA28" "163"    -      -      -       -
  203   BC_1_INPUT     "CMDATA27"  -      "162"   -      212     0
```

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 105 of 131

202	BC_1_INPUT	"CMDATA27"	"162"	-	-	-	-
201	BC_1_INPUT	"CMDATA26"	-	"159"	-	212	0
200	BC_1_INPUT	"CMDATA26"	"159"	-	-	-	-
199	BC_1_INPUT	"CMDATA25"	-	"158"	-	212	0
198	BC_1_INPUT	"CMDATA25"	"158"	-	-	-	-
197	BC_1_INPUT	"CMDATA24"	-	"157"	-	212	0
196	BC_1_INPUT	"CMDATA24"	"157"	-	-	-	-
195	BC_1_INPUT	"CMDATA23"	-	"154"	-	212	0
194	BC_1_INPUT	"CMDATA23"	"154"	-	-	-	-
193	BC_1_INPUT	"CMDATA22"	-	"153"	-	212	0
192	BC_1_INPUT	"CMDATA22"	"153"	-	-	-	-
191	BC_1_INPUT	"CMDATA21"	-	"152"	-	212	0
190	BC_1_INPUT	"CMDATA21"	"152"	-	-	-	-
189	BC_1_INPUT	"CMDATA20"	-	"149"	-	212	0
188	BC_1_INPUT	"CMDATA20"	"149"	-	-	-	-
187	BC_1_INPUT	"CMDATA19"	-	"148"	-	212	0
186	BC_1_INPUT	"CMDATA19"	"148"	-	-	-	-
185	BC_1_INPUT	"CMDATA18"	-	"147"	-	212	0
184	BC_1_INPUT	"CMDATA18"	"147"	-	-	-	-
183	BC_1_INPUT	"CMDATA17"	-	"146"	-	212	0
182	BC_1_INPUT	"CMDATA17"	"146"	-	-	-	-
181	BC_1_INPUT	"CMDATA16"	-	"145"	-	212	0
180	BC_1_INPUT	"CMDATA16"	"145"	-	-	-	-
179	BC_1_INPUT	"CMDATA15"	-	"144"	-	212	0
178	BC_1_INPUT	"CMDATA15"	"144"	-	-	-	-
177	BC_1_INPUT	"CMDATA14"	-	"141"	-	212	0
176	BC_1_INPUT	"CMDATA14"	"141"	-	-	-	-
175	BC_1_INPUT	"CMDATA13"	-	"140"	-	212	0
174	BC_1_INPUT	"CMDATA13"	"140"	-	-	-	-
173	BC_1_INPUT	"CMDATA12"	-	"139"	-	212	0
172	BC_1_INPUT	"CMDATA12"	"139"	-	-	-	-
171	BC_1_INPUT	"CMDATA11"	-	"138"	-	212	0
170	BC_1_INPUT	"CMDATA11"	"138"	-	-	-	-
169	BC_1_INPUT	"CMDATA10"	-	"137"	-	212	0
168	BC_1_INPUT	"CMDATA10"	"137"	-	-	-	-
167	BC_1_INPUT	"CMDATA9"	-	"136"	-	212	0
166	BC_1_INPUT	"CMDATA9"	"136"	-	-	-	-
165	BC_1_INPUT	"CMDATA8"	-	"133"	-	212	0
164	BC_1_INPUT	"CMDATA8"	"133"	-	-	-	-
163	BC_1_INPUT	"CMDATA7"	-	"132"	-	212	0
162	BC_1_INPUT	"CMDATA7"	"132"	-	-	-	-
161	BC_1_INPUT	"CMDATA6"	-	"131"	-	212	0
160	BC_1_INPUT	"CMDATA6"	"131"	-	-	-	-
159	BC_1_INPUT	"CMDATA5"	-	"128"	-	212	0
158	BC_1_INPUT	"CMDATA5"	"128"	-	-	-	-
157	BC_1_INPUT	"CMDATA4"	-	"127"	-	212	0
156	BC_1_INPUT	"CMDATA4"	"127"	-	-	-	-
155	BC_1_INPUT	"CMDATA3"	-	"126"	-	212	0
154	BC_1_INPUT	"CMDATA3"	"126"	-	-	-	-

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 106 of 131

153	BC_1_INPUT	"CMDATA2"	-	"123"	-	212	0
152	BC_1_INPUT	"CMDATA2"	"123"	-	-	-	-
151	BC_1_INPUT	"CMDATA1"	-	"122"	-	212	0
150	BC_1_INPUT	"CMDATA1"	"122"	-	-	-	-
149	BC_1_INPUT	"CMDATA0"	-	"121"	-	212	0
148	BC_1_INPUT	"CMDATA0"	"121"	-	-	-	-
147	BC_1_INPUT	"-"	-	-	-	-	-
146	BC_1_INPUT	"CMADR15"	-	"120"	-	147	0
145	BC_1_INPUT	"CMADR14"	-	"119"	-	147	0
144	BC_1_INPUT	"CMADR13"	-	"118"	-	147	0
143	BC_1_INPUT	"CMADR12"	-	"117"	-	147	0
142	BC_1_INPUT	"CMADR11"	-	"114"	-	147	0
141	BC_1_INPUT	"CMADR10"	-	"113"	-	147	0
140	BC_1_INPUT	"CMADR9"	-	"112"	-	147	0
139	BC_1_INPUT	"CMADR8"	-	"111"	-	147	0
138	BC_1_INPUT	"CMADR7"	-	"110"	-	147	0
137	BC_1_INPUT	"CMADR6"	-	"109"	-	147	0
136	BC_1_INPUT	"CMADR5"	-	"108"	-	147	0
135	BC_1_INPUT	"CMADR4"	-	"105"	-	147	0
134	BC_1_INPUT	"CMADR3"	-	"104"	-	147	0
133	BC_1_INPUT	"CMADR2"	-	"103"	-	147	0
132	BC_1_INPUT	"CMADR1"	-	"102"	-	147	0
131	BC_1_INPUT	"CMADR0"	-	"101"	-	147	0
130	BC_1_INPUT	"CMWR"	-	"100"	-	-	-
129	BC_1_INPUT	"CMRD"	-	"99"	-	-	-
128	BC_1_INPUT	"CMCS1"	-	"96"	-	-	-
127	BC_1_INPUT	"CMCS0"	-	"95"	-	-	-
126	BC_1_INPUT	"COCO"	-	"94"	-	-	-
125	BC_1_INPUT	"COCI"	"93"	-	-	-	-
124	BC_1_INPUT	"CAM"	"92"	-	-	-	-
123	BC_1_INPUT	"SES3"	-	"91"	-	-	-
122	BC_1_INPUT	"SES2"	-	"90"	-	-	-
121	BC_1_INPUT	"SES1"	-	"89"	-	-	-
120	BC_1_INPUT	"SES0"	-	"88"	-	-	-
119	BC_1_INPUT	"CPUR"	-	"87"	-	-	-
118	BC_1_INPUT	"-"	-	-	-	-	-
117	BC_1_INPUT	"HDATA31"	-	"86"	-	118	0
116	BC_1_INPUT	"HDATA31"	"86"	-	-	-	-
115	BC_1_INPUT	"-"	-	-	-	-	-
114	BC_1_INPUT	"HDATA30"	-	"85"	-	115	0
113	BC_1_INPUT	"HDATA30"	"85"	-	-	-	-
112	BC_1_INPUT	"-"	-	-	-	-	-
111	BC_1_INPUT	"HDATA29"	-	"82"	-	112	0
110	BC_1_INPUT	"HDATA29"	"82"	-	-	-	-
109	BC_1_INPUT	"-"	-	-	-	-	-
108	BC_1_INPUT	"HDATA28"	-	"81"	-	109	0
107	BC_1_INPUT	"HDATA28"	"81"	-	-	-	-
106	BC_1_INPUT	"-"	-	-	-	-	-
105	BC_1_INPUT	"HDATA27"	-	"80"	-	106	0

104	BC_1_INPUT	"HDATA27"	"80"	-	-	-	-
103	BC_1_INPUT	"-"	-	-	-	-	-
102	BC_1_INPUT	"HDATA26"	-	"77"	-	103	0
101	BC_1_INPUT	"HDATA26"	"77"	-	-	-	-
100	BC_1_INPUT	"-"	-	-	-	-	-
99	BC_1_INPUT	"HDATA25"	-	"76"	-	100	0
98	BC_1_INPUT	"HDATA25"	"76"	-	-	-	-
97	BC_1_INPUT	"-"	-	-	-	-	-
96	BC_1_INPUT	"HDATA24"	-	"75"	-	97	0
95	BC_1_INPUT	"HDATA24"	"75"	-	-	-	-
94	BC_1_INPUT	"-"	-	-	-	-	-
93	BC_1_INPUT	"HDATA23"	-	"72"	-	94	0
92	BC_1_INPUT	"HDATA23"	"72"	-	-	-	-
91	BC_1_INPUT	"-"	-	-	-	-	-
90	BC_1_INPUT	"HDATA22"	-	"71"	-	91	0
89	BC_1_INPUT	"HDATA22"	"71"	-	-	-	-
88	BC_1_INPUT	"-"	-	-	-	-	-
87	BC_1_INPUT	"HDATA21"	-	"70"	-	88	0
86	BC_1_INPUT	"HDATA21"	"70"	-	-	-	-
85	BC_1_INPUT	"-"	-	-	-	-	-
84	BC_1_INPUT	"HDATA20"	-	"69"	-	85	0
83	BC_1_INPUT	"HDATA20"	"69"	-	-	-	-
82	BC_1_INPUT	"-"	-	-	-	-	-
81	BC_1_INPUT	"HDATA19"	-	"68"	-	82	0
80	BC_1_INPUT	"HDATA19"	"68"	-	-	-	-
79	BC_1_INPUT	"-"	-	-	-	-	-
78	BC_1_INPUT	"HDATA18"	-	"67"	-	79	0
77	BC_1_INPUT	"HDATA18"	"67"	-	-	-	-
76	BC_1_INPUT	"-"	-	-	-	-	-
75	BC_1_INPUT	"HDATA17"	-	"64"	-	76	0
74	BC_1_INPUT	"HDATA17"	"64"	-	-	-	-
73	BC_1_INPUT	"-"	-	-	-	-	-
72	BC_1_INPUT	"HDATA16"	-	"63"	-	73	0
71	BC_1_INPUT	"HDATA16"	"63"	-	-	-	-
70	BC_1_INPUT	"-"	-	-	-	-	-
69	BC_1_INPUT	"HDATA15"	-	"62"	-	70	0
68	BC_1_INPUT	"HDATA15"	"62"	-	-	-	-
67	BC_1_INPUT	"-"	-	-	-	-	-
66	BC_1_INPUT	"HDATA14"	-	"61"	-	67	0
65	BC_1_INPUT	"HDATA14"	"61"	-	-	-	-
64	BC_1_INPUT	"-"	-	-	-	-	-
63	BC_1_INPUT	"HDATA13"	-	"60"	-	64	0
62	BC_1_INPUT	"HDATA13"	"60"	-	-	-	-
61	BC_1_INPUT	"-"	-	-	-	-	-
60	BC_1_INPUT	"HDATA12"	-	"59"	-	61	0
59	BC_1_INPUT	"HDATA12"	"59"	-	-	-	-
58	BC_1_INPUT	"-"	-	-	-	-	-
57	BC_1_INPUT	"HDATA11"	-	"56"	-	58	0
56	BC_1_INPUT	"HDATA11"	"56"	-	-	-	-

55	BC_1_INPUT	"-"	-	-	-	-	-
54	BC_1_INPUT	"HDATA10"	-	"55"	-	55	0
53	BC_1_INPUT	"HDATA10"	"55"	-	-	-	-
52	BC_1_INPUT	"-"	-	-	-	-	-
51	BC_1_INPUT	"HDATA9"	-	"54"	-	52	0
50	BC_1_INPUT	"HDATA9"	"54"	-	-	-	-
49	BC_1_INPUT	"-"	-	-	-	-	-
48	BC_1_INPUT	"HDATA8"	-	"53"	-	49	0
47	BC_1_INPUT	"HDATA8"	"53"	-	-	-	-
46	BC_1_INPUT	"-"	-	-	-	-	-
45	BC_1_INPUT	"HDATA7"	-	"52"	-	46	0
44	BC_1_INPUT	"HDATA7"	"52"	-	-	-	-
43	BC_1_INPUT	"-"	-	-	-	-	-
42	BC_1_INPUT	"HDATA6"	-	"49"	-	43	0
41	BC_1_INPUT	"HDATA6"	"49"	-	-	-	-
40	BC_1_INPUT	"-"	-	-	-	-	-
39	BC_1_INPUT	"HDATA5"	-	"48"	-	40	0
38	BC_1_INPUT	"HDATA5"	"48"	-	-	-	-
37	BC_1_INPUT	"-"	-	-	-	-	-
36	BC_1_INPUT	"HDATA4"	-	"47"	-	37	0
35	BC_1_INPUT	"HDATA4"	"47"	-	-	-	-
34	BC_1_INPUT	"-"	-	-	-	-	-
33	BC_1_INPUT	"HDATA3"	-	"46"	-	34	0
32	BC_1_INPUT	"HDATA3"	"46"	-	-	-	-
31	BC_1_INPUT	"-"	-	-	-	-	-
30	BC_1_INPUT	"HDATA2"	-	"45"	-	31	0
29	BC_1_INPUT	"HDATA2"	"45"	-	-	-	-
28	BC_1_INPUT	"-"	-	-	-	-	-
27	BC_1_INPUT	"HDATA1"	-	"44"	-	28	0
26	BC_1_INPUT	"HDATA1"	"44"	-	-	-	-
25	BC_1_INPUT	"-"	-	-	-	-	-
24	BC_1_INPUT	"HDATA0"	-	"43"	-	25	0
23	BC_1_INPUT	"HDATA0"	"43"	-	-	-	-
22	BC_1_INPUT	"SMCSID3"	"40"	-	-	-	-
21	BC_1_INPUT	"SMCSID2"	"39"	-	-	-	-
20	BC_1_INPUT	"SMCSID1"	"38"	-	-	-	-
19	BC_1_INPUT	"SMCSID0"	"37"	-	-	-	-
18	BC_1_INPUT	"SMCSADR3"	"36"	-	-	-	-
17	BC_1_INPUT	"SMCSADR2"	"35"	-	-	-	-
16	BC_1_INPUT	"SMCSADR1"	"34"	-	-	-	-
15	BC_1_INPUT	"SMCSADR0"	"33"	-	-	-	-
14	BC_1_INPUT	"BOOTLINK"	"32"	-	-	-	-
13	BC_1_INPUT	"HADR7"	"29"	-	-	-	-
12	BC_1_INPUT	"HADR6"	"28"	-	-	-	-
11	BC_1_INPUT	"HADR5"	"27"	-	-	-	-
10	BC_1_INPUT	"HADR4"	"26"	-	-	-	-
9	BC_1_INPUT	"HADR3"	"25"	-	-	-	-
8	BC_1_INPUT	"HADR2"	"24"	-	-	-	-
7	BC_1_INPUT	"HADR1"	"23"	-	-	-	-

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 109 of 131

```
    6     BC_1_INPUT        "HADR0"     "22"     -          -        -        -

    5     BC_1_INPUT        "HINTR"     -        "19"       -        -        -

    4     BC_1_INPUT        "-"         -        -          -        -        -

    3     BC_1_INPUT        "HACK"      -        "18"       -        4        0

    2     BC_1_INPUT        "HWR"       "17"     -          -        -        -

    1     BC_1_INPUT        "HRD"       "16"     -          -        -        -

    0     BC_1_INPUT        "HSEL"      "15"     -          -        -        -
*****************************************************************************
END CHAIN_LIST


END BOUNDARY_SCAN_REGISTER



IDENTIFICATION_REGISTER IDENT


PURPOSE             : "device identification"

LENGTH              : 32

CAPTURE             : "00010100010001010011000010110001"

MANUFACTURER_IDCODE : "00001011000"

PARTNUMBER_IDCODE   : "0100010001010011"

VERSION_IDCODE      : "0010"


END IDENTIFICATION_REGISTER



GLOBAL_FAMILY "SMCS"


TCK_STOP_AT_ONE     : YES

TEST_FREQUENCY      : 0 10.0E6 10.0E6

TEMPERATURE_RANGE   : - - -


END GLOBAL_FAMILY

END COMPONENT

END EBST
```

12 Handling Empty Packets

12.1 Description

Due to an anomaly in the SpaceWire CODEC (Coder / Decoder) version 1.4, which is used in the SMCS332SpW, the SMCS332SpW behavior is affected when more than one empty packet is received in a row. In this case the credit count is not updated correctly. Please note, that empty packets are normally not expected. The SMCS332SpW never transmits empty packets; however the problem may occur when the SMCS332SpW receives empty packets.

For example if <data><EOP><EOP><EOP> is received then the receive credit counter is incremented only for one EOP. However, the transmit credit counter at the other end of the link would have been incremented once for each EOP. This gives rise to a discrepancy between the two FCT counters.

The handling of empty packets is specifically mentioned in the SpaceWire standard (sub-clause 8.9.3). EOPs and EEPs should always be counted in the credit counter (sub-clause 8.3 i). EOPs and EEPs are NChars. This means that the SpaceWire CODEC version 1.4 and therefore the SMCS332SpW are NOT fully compliant with the SpaceWire standard.

Detailed Description:

The SMCS332Spw has space for 16 characters in the receive buffer. After reading 8 characters (data or EOP/EEP) from the receive buffer, a FCT will be sent to the transmitting link node. Empty packets, except for the first empty packet after a link initialisation, are deleted before writing to the receive buffer, therefore these packets (EOP's) do not increment the receive credit counter. This causes a discrepancy between the receiver and the transmitter credit counter. A discrepancy of 8 (eight empty packets), causes a slow-down of the link. If the discrepancy is greater than 8, the link runs into a deadlock situation, because the SMCS332Spw will not send a FCT and the credit counter of the transmitter is expired.

Scenario 1

Credit Tx		Discr	Credit SMCS332	Rx Buffer	Remark
16	← 2 FCT	0	16		Link Initialisation
8	8 EOP →	**7**	15	1 EOP	7 empty packets*
7	1 EOP →	**8^1**	15		1 empty packet
0	7 Data Bytes →	**8^1**	8	7 Byte	
8	← 1 FCT	**8^1**	16		
7	1 EOP →	**8^1**	15	1 EOP	
6	1 EOP →	**9^2**	15		1 empty packet
0	6 Data Bytes →	**9^2**	9	6 Byte	
DEADLOCK , Discrepancy = 9					

EADS ASTRIUM

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 111 of 131

*after Link-Initialisation/Disconnect the RCVEOP (received EOP) flag is cleared, therefore an EOP/EEP transmitted directly after link connects is counted properly and can be read from Rx-Buffer

[1]A discrepancy between the two credit-counters of 8 causes a reduced transmission rate

[2]A discrepancy between the two credit-counters of more than 8 causes a deadlock situation

Scenario 2

Credit Tx		Discr	Credit SMCS332	Rx Buffer	Remark
16	← 2 FCT	0	16		Link Initialisation
9	7 Data Bytes →	0	9	7 Byte	
8	1 EOP →	0	8	1 EOP	
16	← 1 FCT	0	16		
15	1 EOP →	1	16		Empty packet
8	7 Data Bytes →	1	9	7 Byte	
7	1 EOP →	1	8	1 EOP	
15	← 1 FCT	1	16		
14	1 EOP →	2	16		Empty packet
7	7 Data Bytes →	2	9	7 Byte	
6	1 EOP →	2	8	1 EOP	
14	← 1 FCT	2	16		
13	1 EOP →	3	16		Empty packet
6	7 Data Bytes →	3	9	7 Byte	
5	1 EOP →	3	8	1 EOP	
13	← 1 FCT	2	16		
12	1 EOP →	4	16		Empty packet
5	7 Data Bytes →	4	9	7 Byte	

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 112 of 131

4	1 EOP →	4	8	1 EOP	
12	← 1 FCT	4	16		
11	1 EOP →	5	16		Empty packet
4	7 Data Bytes →	5	9	7 Byte	
3	1 EOP →	5	8	1 EOP	
11	← 1 FCT	5	16		
10	1 EOP →	6	16		Empty packet
3	7 Data Bytes →	6	9	7 Byte	
2	1 EOP →	6	8	1 EOP	
10	← 1 FCT	6	16		
9	1 EOP →	7	16		Empty packet
2	7 Data Bytes →	7	9	7 Byte	
1	1 EOP →	7	8	1 EOP	
9	← 1 FCT	7	16		
8	1 EOP →	8	16		Empty packet
1	7 Data Bytes →	8	9	7 Byte	
0	1 EOP →	8	8	1 EOP	
8	← 1 FCT	8	16		
7	1 EOP →	9	16		Empty packet
0	7 Data Bytes →	9	9	7 Byte	
DEADLOCK					

12.2 Workaround

A stop at either end of the link causes a disconnect and hence clears this problem. Note that there is NO need to reset the entire device

Please do make sure that this situation be treated properly if consecutive empty packets are expected.

13 Simple Interprocessor Communication Protocol Specification

13.1 Application Scenario

The purpose of this protocol specification is to provide a framework which allows the

exchange of data and simple system control commands between single nodes of a multiprocessor system. Nodes in this context are understood as computing nodes and controlling nodes (a physical node may be a mix of both). The exchange of data and commands is structured into packets.

A target multiprocessor system for this protocol is understood as a "small to medium size" system:

- the multiprocessor system is based on a message passing architecture with only local memory

- the protocol supports 256 different link addresses, this means the protocol is applicable to multiprocessor systems consisting of e.g. 256 nodes each with one link (not very likely), 64 nodes each with four links, 32 nodes each with 8 links, etc ...

- the interconnect network between the nodes is a 'flat' network, no hierarchical addressing is required and supported. Especially 'header deletion' is not supported (since it is not required in such relatively small networks)

- multicast and broadcast messages are not supported, only 'simple' destination fields in ECSS-E-50-12A terminology are provided

- the three previous addressing characteristics allow to reduce the size of the destination field of the packet header to one byte

- the target system is physically small enough to fit into one box: the interconnect capabilities covers connections between elements on a PC-board and between boards in a box (via a backplane or equivalent interconnect system). Interconnect between boxes is not covered. This has one major implication: it is assumed that the electrical environment within boxes is "clean" and no error is introduced on the links (e.g. by spikes, noise or HF interference).

Experience with bus systems has shown that indeed data transfer can be regarded as reliable and parity checking is sufficient to detect possible (very rare) transmission failures. In this case error recovery must be handled by some kind of control program.

It is assumed that a SW kernel is running on the nodes of the system, providing resources for the execution for the major part of the transaction level protocol functions. From the transaction level protocol functions, the specification scope of this document covers the frame of the data transfers between two nodes and the coding and execution of a range of required control commands. It is explicitly intended to specify only a minimum of transaction level functions and formats in order to keep flexibility and broaden the application range.

In general, the basic design approach can be described as trading off functionality for conceptual and implementation simplicity and speed.

13.2   Assumptions about the Environment

The protocol specified here is intended to work on top of the SpaceWire link protocol. This covers e.g. link startup, low level error handling, low level flow control, timeouts, routing provisions and the like.

This protocol assumes a link interface HW implementation based on a front-end which handles all SpaceWire related issues and a backend providing resources to interpret this protocol and perform the required actions e. g. in case of commands or acknowledges. The link interface is controlled (at least for configuration purposes) by the CPU of a node.

It is further assumed that within the protocol specific HW circuitry buffer storage (FIFOs) exist in addition to the receive and transmit FIFOs provided by the SpaceWire link frontend circuitry. These additional FIFOs are large enough to decouple (at least to a certain extent) the transmission speed on the SpaceWire links from the transfer speed between the node's communication memory and the link interface. If the receive FIFO is full, reception of further tokens is stalled utilizing the SpaceWire link low level flow control mechanism.

If more than one link interface HW is integrated on one chip the different link channels can share the interfaces for the protocol's control commands, for the data transferred between the node's communication memory and the links and the control interface provided to enable control by the CPU.

The transmission of data and command is structured into packets. Although this protocol does not limit the maximum packet length, it is recommended that the packet length is kept short to decrease latency and to avoid blocking of links due to long packets.

Since the protocol is intended to be used in HW implementations housed in shielded boxes with a "clean" electrical environment inside, no additional error protection/checking is performed on top of the parity checking already provided by the SpaceWire links. However, it is expected that the SW kernel running on the nodes of the system provides a certain set of control functions.

The protocol supports routing and assumes Inmos C104 like routing networks with one byte header and without header deletion. The protocol provides a structure to enable routing but does not define how the routing is actually performed and how certain network events (e.g. transmission errors or change of transmission speed) are propagated via the router.

13.3   Service Specification

Based on this protocol, data characters and simple system control commands are transported between nodes of a multiprocessor system. The protocol does interpret control commands if they are coded as specified for 'simple control commands' in para. 5. Data transferred via the link is not interpreted by the protocol but exchanged transparently with the CPU of a node. This implies that every acknowledgement required by the content of the data must be performed by SW (a higher level communication protocol) running on the CPU.

Since this protocol does not interpret the data transferred between nodes, specific functions and commands in addition to the ones specified in the description below can be freely embedded within the data.

The protocol is based on a packet communication structure, it allows a full duplex transfer of data and commands.

13.3.1   Transport of data between two nodes

The transaction level protocol described here provides transport of data structured into packets between the endpoints of a communication link. It does not interpret the data. It is up to other (here not specified) parts of a higher transaction level protocol (e.g. running on the CPU of the node) to control the transfer data from the link interface to buffers in the working memory of a node and to interpret the data.

The SpaceWire links which are used for the implementation of the low levels of the protocol transports data in pieces of 'data characters' consisting of 8 bits. It is also the smallest entity of data which is covered by the data transport service of this protocol. It is up to an implementation into which higher datastructures the 8 bit entity is organized.

Data transfer between two nodes is basically acknowledged by this protocol on a packet by packet basis. This acknowledge covers the correct reception of data up to the EOP 'end of packet marker' and the storage of the data into the link interface's receive FIFO. The acknowledge provided here does not cover the correct transfer of data from the link interface HW to the

node CPU's working memory. If required, it is up to higher levels of a communication protocol to provide this acknowledge. Refer to para. 4.1 for details.

13.3.2 Execution of control commands

The protocol supports two kinds of commands:

- Simple control commands which may be executed directly by a link interface HW (without intervention of the node's CPU). The coding and function of the simple commands are specified in this document.

- Complex control commands coded into the data transported between two nodes.

This protocol defines only the basic functionality of some complex commands, the coding of these commands and execution details are up to a specific implementation. It is likely that the complex control commands are executed by the SW kernel running on the node. This document gives only a few constraints for complex command implementation. Implementations of this protocol can define their own complex commands.

13.3.2.1 Simple Control Commands

Enable command execution

For security reasons, the execution of a set of safety critical simple commands must be enabled before a command can be executed. A safety critical simple command is executed only after an 'enable command execution' command has been received.

Activate CPU Reset

This command activates an external static signal CPUR which brings the CPU of the node into its reset state.

This command must be enabled.

Deactivate CPU Reset

This command deactivates an external signal CPUR which is set by the 'Reset CPU' command and such enables restart of the CPU's command execution.

This command must be enabled.

Activate External Signal 0 ... 3

This command activates one of four specific external signals SES0 ... SES3. The interpretation of these signals is application dependent.

This command must be enabled.

Deactivate External Signal 0 ... 3

This command deactivates one of four specific external signals SES0 ... SES3.

This command must be enabled.

Reset Link Interface HW

This command performs a hard reset (interrupting all currently running transmissions) of the link channel receiving this command.

This command must be enabled:

Reset Link Interface Unit HW

This command performs a hard reset (interrupting all currently running transmissions) of all link interfaces integrated within one link interface unit.

This command must be enabled.

Read Link Interface Status Register

This commands reads the status register of the link interface and sends the content to the requester within the response packet of this command.

This command can be executed without being previously enabled.

It depends on the implementation whether the CPU reset and the activate/deactivate signal commands are supported or not. It is only possible to support all of these commands or none of them. Subsets are not allowed.

13.3.2.2 Complex Control Commands

Currently the following complex commands are foreseen:

- enable complex command
- read status of complex command enable switch
- set link speed to minimum
- set link speed to maximum
- shut down link operation
- restart link operation

Enable Complex Command / Read Status of Complex Command Enable Switch

To enhance security, all safety relevant complex commands must be enabled by a specific command. This command activates a SW switch which allows the execution of one complex command. If the switch is set or not can be read back by the respective read status command.

Set Link Speed to Maximum / Set Link Speed to Minimum

These two commands are required since it is assumed that the link interfaces accept in their receiver part transmission speeds between the minimum and maximum range specified in para. 2 but that their transmission speed do not automatically follow the speed of the data received from the other node. This has following effect in case of speed increase : although one node is transmitting at full speed, the link bandwidth is near its minimum value since the FCTs and acknowledgements from the other node are sent with minimum speed. The node intending to change its transmission speed has to inform the other node so that their transmission speed can be adapted e.g. by CPU commands forwarded to the link I/F HW.

The actual link speed is indicated in the status register.

This command must be enabled.

Note that the described simple procedure works only for direct connected systems but that systems incorporating a router require more complicated procedures, e.g by involving the system controller.

Shut Down Link Operation / Restart Link Operation

This command is required since the link end which intents to stop link operation must inform the other end of the link. Otherwise the link shutdown might be interpreted as a link failure.

If a link has been shut down previously, it can be requested to start up again by a 'restart link operation' complex command. This command has to be issued via an already operating link to the CPU of a node which then initiates the actual restart operation for the link requested to start up.

Details of the link shutdown operation are given in para. 4.

Note that the simple procedures work only for direct connected systems but that systems incorporating a router require more complicated procedures, e.g by involving the system controller.

All complex control commands should be addressed. This means that a complex control command received via one link interface can control every other link interface on the same node.

13.4 Procedure Rules

13.4.1 Acknowledgements

Data Transmission and Simple Commands

With the exception of the reset command, the reception of each packet is acknowledged by an acknowledge packet. The acknowledge packet informs the requester node on the successful reception of the packet but does not contain information about a completed execution of a command. The acknowledgement covers

-        in case of data transfer commands (including encapsulated complex commands) the correct reception of the packet including the associated EOP marker and the forwarding of the data to the receive FIFO in the link interface. It does not cover any operation related to memory access of the node CPU's working (communication) memory.

-        in case of simple commands the correct reception of the packet including the associated EOP marker, the successful decoding of the command and the forwarding of the command to the command execution unit. The acknowledgement covers additionally the cases whether or not the command execution unit has been enabled previously for safety critical commands or not.

In case of data transmission or transmission of simple commands, the requesting node has to wait for the last acknowledge of a previous data transmission or the acknowledge of a previous simple command execution request or the high level acknowledge of a previous complex command before a new simple command execution request packet can be send. In other words: split transactions are NOT allowed.

Within this protocol only two forms of acknowledgement are required:

-        reception successful

-        reception with errors or problems with the execution of the requested command

If the second form of the acknowledgement is received, it is up to the requester node to read the status register of the responder node in order get more information about the malfunction.

The acknowledge packet structure is defined in para. 5. The acknowledge packet is generated immediately after the reception of the 'end of packet' marker of the received packet and forwarded to the transmission segment of the link channel. If data and acknowledge packets compete on access to the transmission segment, acknowledge packets have precedence but can occupy the transmission segment only after the previous transmission has been finished. Since a currently running transmission cannot be interrupted, a significant delay between end of the received packet and transmission of the acknowledgement can occur.

Complex Commands

Issuing complex commands results in two acknowledges:

-        the low level acknowledge of the successful reception of the data packet containing the complex command and

-        a high level acknowledge (contained in a response data packet) with information on e.g. the execution of the command.

Note that this high level acknowledge also releases a low level acknowledge indicating that the high level acknowledge data packet has been received.

The content and coverage of the high level acknowledge can be defined by a specific implementation but regarding sending complex commands, the same policy holds as for simple commands: a complex command can only be issued if the last acknowledge of a previous data transmission or the acknowledge of a previous simple command or the high level acknowledge of a previous complex command has been received. No split transactions are allowed.

Control Word Content not Specified

If the content of a control word in a requester packet is not specified (according to the control word specification in para. 5) then the rest of the packet up to the next EOP is ignored. After reception of the EOP of the requester packet, an acknowledge is sent back indicating an error. The requesting node can interpret the mirrored content of the control word to recognize the unspecified control word.

13.4.2   Access to Command Signal Output Ports

Basically, every link channel has its own set of command signal output ports. The "reset CPU" and "activate/deactivate specific external signal" commands of each link channel are forwarded to their command signal ports without any prioritization or other coordination.

In order to save pins when multiple link interfaces are integrated on one link interface unit chip, the SES0 ... SES3 signals of the different link channels can be combined in such a way that a logical 'or' function is realized:

if one of the link interfaces activates one of the specific external signals then the respective signal at the output pin will be activated.

e.g. if one link interface activates SES0 then the respective output pin is activated independent from the state of the other SES0 signals from the other link interfaces.

In this case it is a task of the system implementation to control via the 'enable command execution' command the access to the specific external signal output pins.

It is up to the implementation to chose a physical signal level for the 'active' and 'passive' states of the CPUR and SES0 ... SES3 signals.

13.4.3   Safety Critical Commands

Simple Control Commands

The execution of safety critical simple commands consists of a sequence of two steps:

-        enabling the execution of simple commands

-        execution of the simple command

For each of these step a separate simple command request packet from the requester node is required. The status register of the link channel includes one bit indicating if safety critical simple command execution is enabled or not.

After a critical simple command has been executed, the simple command execution state is immediately changed from 'enable' to 'disable'. This assures that every critical simple command execution must be preceded by a 'enable' command.

Complex Control Commands

The same two step procedure as for simple control commands is applied to complex control commands with the difference that the status of enabling a complex command is not stored in the link status register. Since it is assumed that the complex commands are executed by SW running on the CPU of a node, the status of complex commands is kept in a SW register and can be read by the 'read status of complex command enable switch' complex command.

13.4.4 Reset Commands

Reset Link Interface HW

Provided that safety critical command execution has been enabled, the execution of this command is performed immediately after correct reception of the command and the EOP marker. Due to the immediate execution, no acknowledge packet is sent. The command has the following results:

Link Frontend

After receiving a reset link channel command, the frontend performs the following actions analog to the SpaceWire DS-SE link procedures.

A auto-restart procedure is required so that the node requesting a 'reset link channel' command can assume that the other node is again operating after a certain amount of time after the 'reset link channel' command.

This requires that the link frontend which has executed the 'reset' command either automatically starts to transmit FCTs and NULL characters after having received tokens from the other node or that the node CPU observes reception of tokens after link interface reset and then enables the link interface to restart with transmission.

Channel Status Register

The status register is cleared (reset) but one bit is set indicating that the link channel has started up from an externally commanded reset.

Node CPU Interface

The interface configurations are not modified and the interface MUST be left operational but a signal is generated indicating to the CPU that an externally commanded reset has occurred. (This may be implemented as a specific signal or as an interrupt/read status register sequence)

Memory Interface / Transmission Path

Any currently running transmission at the link to be reset is interrupted immediately by the reset command. Since the state of transmission cannot be restored, the transmission path is completely reset including clearing of any transmission related configuration register of the memory interface and clearing of intermediate storage containing data already read for transmission from communication memory but not yet forwarded to the link frontend. The transmission path must be completely reconfigured by the CPU after this kind of reset.

Memory Interface / Reception Path

The reception path is left unchanged, especially all intermediate storage containing data received previously but not yet forwarded to the working memory of the node. The reception path can be left unchanged since it can be assumed that the last data packet previous to this reset command was received correctly, otherwise this reset command would have not been

received and decoded correctly. Data received previous to the reset command is still forwarded to the node's working memory.

Utilizing the interrupt signal indicating to the node CPU that an externally commanded link reset has occurred, the CPU can then decide on invalidating data already received via this channel.

Reset Link Interface Unit HW

Provided that safety critical command execution has been enabled, the execution of this command is performed immediately after correct reception of the command and the EOP marker. Due to the immediate execution, no acknowledge packet is sent. The command has the same results on ALL link channels integrated on one chip as the "reset link interface HW" command described above with the exception of the reception path of the memory interface.

The reception path of all link channel interfaces are reset in the same way as it is the case for the transmission path:

- clearing of all configuration registers

- clearing of all intermediated storage, marking FIFOs as empty

13.4.5 Shutdown Link Channel Operation / Restart Link Channel Operation

During link shutdown, basically the same actions are performed as for the 'reset link interface unit HW' command. It is up to the implementation to assure that no currently running traffic is disturbed by a link shutdown.

The 'restart link channel operation' complex command initiates a link frontend restart just like a restart from a simple 'reset link interface' command.

13.4.6 Read of Link Interface Status Register

If the link channel status register is read via the link, the content of the register must be kept constant during the read operation. Change of the link channel status register during a read operation is not allowed.

Some entries in the status register are transient in the sense that they are reset after a read. This holds especially for some error information which is regarded as invalid when it has been read. Details are given in para. 5.

13.5 Encoding (Format) of Transactions

13.5.1 Header and EOP Coding

DEST   Destination address of the packet

8 bit field

CNTRL 8 bit control word for commands and acknowledges

according to para. 5.2

TID   Transaction identifier of the packet

Unique number generated by the requester to identify acknowledges

8 bit field

SRC   Source address of the packet

8 bit field

EOP   End of Packet marker

13.5.2 Control Word Coding Specification

Bit	Content
7 (MSB)	Request/Acknowledge Packet 0 = Acknowledge 1 = Request
6	Acknowledge OK or Error 0 = Transaction Error Acknowledge 1 = OK Acknowledge This bit is not interpreted if the packet is a request packet.
5,4	Request Type Packet Specification (see table below)
3	Command Specification (see table below)
2,1,0	Detailed Command Specification (see table below)
Bits 5-0 are simply mirrored into the respective bits of an acknowledge control word	

Request Type and Command Specification Table

Req. Spec		Command Specification				Remarks
5	4	3	2	1	0	
0	0	x	x	x	x	reserved for future use, Bit 3-0 are not interpreted
0	1	0	x	x	x	Complex Command Request (coded within the datafield), Bit 2-0 are not interpreted
0	1	1	x	x	x	Complex Command Acknowledge (details coded in the datafield), Bit 2-0 are not interpreted
1	0	x	x	x	x	Transfer Data Packet, Bit 3-0 are not interpreted
1	1	0	0	0	0	Simple Command, Read Link Status Register
1	1	0	0	0	1	Command not specified
1	1	0	0	1	0	Enable Command Execution
1	1	0	0	1	1	Command not specified
1	1	0	1	0	1	Activate CPU Reset
1	1	0	1	0	0	Deactivate CPU Reset
1	1	0	1	1	0	Reset Link Interface HW
1	1	0	1	1	1	Reset All Link Interface HW
1	1	1	1	0	0	Activate Specific External Signal 0
1	1	1	1	0	1	Activate Specific External Signal 1
1	1	1	1	1	0	Activate Specific External Signal 2
1	1	1	1	1	1	Activate Specific External Signal 3
1	1	1	0	0	0	Deactivate Specific External Signal 0
1	1	1	0	0	1	Deactivate Specific External Signal 1
1	1	1	0	1	0	Deactivate Specific External Signal 2
1	1	1	0	1	1	Deactivate Specific External Signal 3

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 124 of 131

13.5.3   Link Interface Status Register Encoding

General Layout of the Link Interface Status Register:

Bit 3 - Bit 0 (LSB)   Link Interface Operation Mode

Bit 7 - Bit 4   Link Interface Operation Status

Bit 15 - Bit 8   Link Interface Error Status

Bit 23 - Bit 16   Link Interface Command Execution Status

Bit 31 - Bit 24   Reserved for Future Implementation

Bit	Remark	Transient
2-0 (LSB)	Link Speed 0xx = minimum speed 100 = maximum speed	
3	currently not used	
4	Restart from Commanded Reset 0 = no restart 1 = restart	X
7-5	currently not used	
8	Link Interface Received Packet with Wrong Address 0 = no wrong addressed packet received 1 = wrong addressed packet received	X
9	Received Command Field not Specified 0 = Command fields specified 1 = not specified	X
10	checksum error in Request Packet 0 = no checksum error occurred 1 = checksum error occurred	X
15-11	currently not used	
16	Execution of Critical Simple Control Commands Enable 0 = not enabled 1 = enabled	
17	CPU Reset Signal Status 0 = not active 1 = active	
19-18	currently not used	
23-20	Specific External Signal 0 ... 3 Status 0 = not active 1 = active	
31-24	currently not used	

13.5.4   Data Transfer Type Transactions

Requester Packet

DEST	CNTRL	TID	SRC	DATA	EOP

CNTRL =   Transfer Data, Complex Command Request or Complex Command Acknowledge

DATA   =   n * 8 bit data

Actions:

Transfer data to host working memory

DEST' = SRC

SRC' = DEST

CNTRL' = Transaction OK

or in case of errors

CNTRL' = Transaction Error

Response/Acknowledge Packet

DEST'	CNTRL'	TID	SRC'	EOP

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 126 of 131

13.5.5   Read Link Interface Status Register Transaction

Requester Packet

DEST	CNTRL	TID	SRC	EOP

CNTRL = Read Link Interface Status Register

Actions:

Reads the Link Interface Status Register

DEST' = SRC
SRC' = DEST
CNTRL' = Transaction OK
4_BYTES_DATA = Content of Link Channel Status Register
or in case of errors
CNTRL' = Transaction Error

Response/Acknowledge Packet

Normal Acknowledge

DEST'	CNTRL'	TID	SRC'	4_BYTES_DATA	EOP

Error Acknowledge

DEST'	CNTRL'	TID	SRC'	EOP

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 127 of 131

13.5.6   Enable Command Execution Transaction

Requester Packet

DEST	CNTRL	TID	SRC	EOP

CNTRL = Enable Safety Critical Command

Actions:

Enable the execution of a following safety critical command

DEST' = SRC

SRC' = DEST

CNTRL' = Transaction OK

or in case of errors

CNTRL' = Transaction Error

Response/Acknowledge Packet

DEST'	CNTRL'	TID	SRC'	EOP

13.5.7   Critical Simple Command Execution Transaction

Requester Packet

DEST	CNTRL	TID	SRC	EOP

CNTRL = Simple Command Request (refer to para 5.2)

Actions:

Execute the requested simple command

DEST' = SRC
SRC' = DEST
CNTRL' = Transaction OK
or in case of errors
CNTRL' = Transaction Error

Response/Acknowledge Packet

DEST'	CNTRL'	TID	SRC'	EOP

13.6 Glossary

Communication Memory   Part of a node's working memory reserved for communication.

Computing Node   A node which performs only computing tasks.

Controlling Node   A node which is allowed to perform system control tasks.
                                          A controlling node may also perform computing tasks.

Level 0   Signal Level

Level 1   Character Level

Level 2   Exchange Level

Level 3   Packet Level

Level 4   Transaction Level

Link Interface Unit   Integrated Unit (single chip / MCM) consisting of multiple link interfaces.

Link Interface   Circuitry to connect a node to one physical DS-SE link channel, consists generally
                                          of a SpaceWire link frontend, a protocol controller and interfaces to the node CPU.

Node   Entity of a multiprocessor system, consists of a CPU, memory and (multiple) link
                                          interfaces. The term is used in the communication related view of a
                                          multiprocessor system.

Requester   The link interface which sends a request packet on the link.

Responder   The link interface which sends a response packet on the link. The response packet
                                          may be a simple acknowledge or a data packet.

Working Memory   Data storage part of a node's memory.

14 Differences between the SMCS332SpW and the old SMCS332

14.1 Summary of changed/modified/added registers or register bits

Address	Register	Description SMCS332	Description SMCS332SpW
0x04 0x05 0x06 0x07	Interrupt status register ISR	ISR Byte0 D0: channel 1: IEEE1355 parity or disconnect error ISR Byte1 D2: channel 2: IEEE1355 parity or disconnect error ISR Byte2 D4: channel 3: IEEE1355 parity or disconnect error ISR Byte3 D6: always '0' / reserved	ISR Byte0 D0: channel 1: SpaceWire parity, disconnect, ESC or credit error ISR Byte1 D2: channel 2: SpaceWire parity, disconnect, ESC or credit error ISR Byte2 D4: channel 3: SpaceWire parity, disconnect, ESC or credit error ISR Byte3 D6: TICK IN received interrupt
0x08 0x09 0x0A 0x0B	Interrupt mask register	IMR Byte3 D6: always '0' / reserved	IMR Byte3 D6: mask bit for TICK IN received interrupt
0x10 0x30 0x50	CHx_DSM_MODR	D5 power save mode bit	D5 always '0' / reserved
0x12 0x32 0x52	CHx_DSM_STAR	D5 always '0' / reserved D6 always '0' / reserved	D5 ESC error D6 FCT error Attention: Only D4 shall be checked whether the link is in the "Run" state or not.
0x13 0x33 0x53	CHx_DSM_TSTR	D4 always '0' / reserved D5 always '0' / reserved	D4 link output mute D5 send EEP instead of EOP
0x18 0x38 0x58	CHx_CNTRL1	D5 always '0' / reserved	D5 header field control bit
0x1F 0x3F 0x5F	CHx_COMICFG	D2 **'0'** = send **EOP1** token at the end of the packet **'1'** = send **EOP2** token at the end of the packet D3 always '0' / reserved	D2 **'0'** = send **EOP** token at the end of the packet **if D3 = '0'** **'1'** = send **EOP** token at the end of the packet **if D3 = '0'** D3 **'0'** = send **EOP** token at

SMCS332SpW
User Manual

EADS Astrium GmbH, ASE2
Doc No: SMCS_ASTD_UM_100
Issue: 1.4
Updated: 9-Sep-2006
Page: 131 of 131

Address	Register	Description SMCS332		Description SMCS332SpW	
				the end of the packet **'1'** = send **NO EOP** token at the end of the packet	
0x27 0x47 0x67	CHx_TX_EOPB	D0	send an EOP1 token	D0	send an EOP token
		D1	send an EOP2 token	D1	send an EOP token
0x2F 0x4F 0x6F	CHx_STAR	D4	EOP1 received	D4	**EOP** received
		D5	EOP2 received	D5	**EEP** received
0x78	TIME_CNTRL	reserved		time code control register	
0x79	TIME_CODE	reserved		time code value register	

14.2 Pin Modifications

Pin number	Description SMCS332	Description SMCS332SpW
1	VCC	PLLOUT
3	GND	VCC
167	NC	GND
168	NC	GND
169	NC	VCC_3VOLT
170	NC	GND
171	NC	GND
175	LEN1	NC
182	LEN2	NC
192	LEN3	TIME_CODE_SYNC
196	PLLOUT	GND