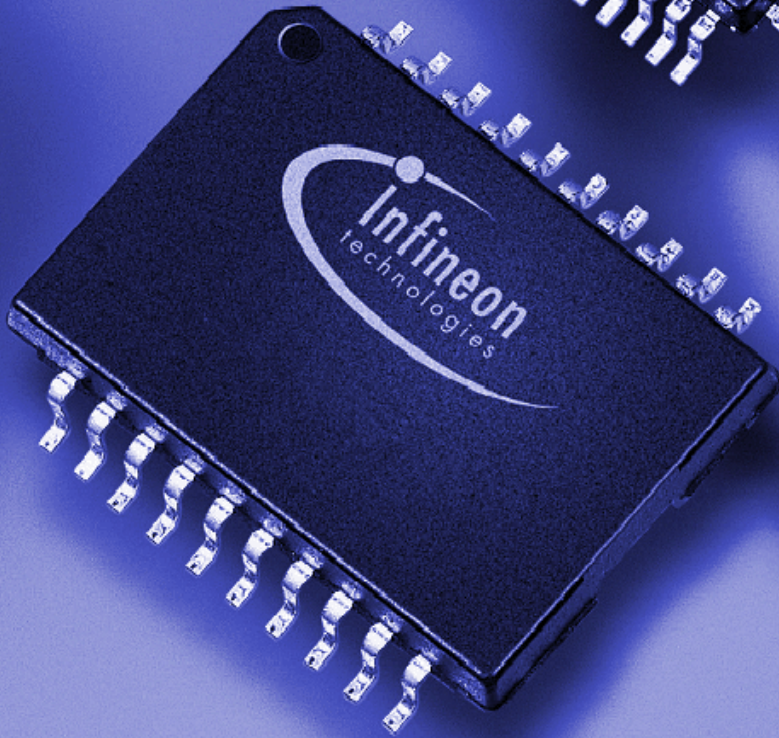


MUNICH256

Multichannel Network Interface  
Controller for HDLC/PPP

PEB 20256 E Version 2.1



Datacom



Never stop thinking.

**Edition 04.2001**

**Published by Infineon Technologies AG,  
St.-Martin-Strasse 53,  
D-81541 München, Germany**

**© Infineon Technologies AG 4/9/01.  
All Rights Reserved.**

**Attention please!**

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

**Information**

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or our Infineon Technologies Representatives worldwide (see address list).

**Warnings**

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

MUNICH256

Multichannel Network Interface  
Controller for HDLC/PPP

PEB 20256 E Version 2.1

Datacom



Never stop thinking.



For questions on technology, delivery and prices please contact the Infineon Technologies Offices in Germany or the Infineon Technologies Companies and Representatives worldwide: see our webpage at <http://www.infineon.com>

## Preface

The Multichannel Network Interface Controller for HDLC/PPP is a Multichannel Protocol Controller for a wide area of telecommunication and data communication applications.

### Organization of this Document

This Data Sheet is divided into ten chapters and is organized as follows:

- **Chapter 1 MUNICH256 Overview**  
Gives a general description of the product and its family, lists the key features, and presents some typical applications.
- **Chapter 2 Pin Description**  
Lists pin locations with associated signals, categorizes signals according to function, and describes signals.
- **Chapter 3 General Overview**  
This chapter provides short descriptions of all the internal functional blocks.
- **Chapter 4 Functional Description**  
Gives a detailed description of all functions.
- **Chapter 5 Interface Description**  
This chapter provides functional diagrams of all interfaces.
- **Chapter 6 Channel Programming / Reprogramming Concept**  
This chapter provides a detailed description of the channel programming concept.
- **Chapter 7 Reset and Initialization procedure**  
Gives examples of the initialization procedure and operation.
- **Chapter 8 Register Description**  
Gives a detailed description of all on-chip registers.
- **Chapter 9 Electrical Characteristics**

Gives a detailed description of all electrical DC and AC characteristics, and provides timing diagrams for all interfaces.

- **Chapter 10 Package Outline.**

Shows the mechanical values of the device package.





<b>Table of Contents</b>		<b>Page</b>
<b>1</b>	<b>MUNICH256 Overview</b> .....	20
1.1	General Features .....	21
1.2	Logic Symbol .....	22
1.3	General System Integration MUNICH256 .....	23
<b>2</b>	<b>Pin Description</b> .....	25
2.1	Pin Diagram 16-Port Mode MUNICH256 .....	25
2.2	Pin Diagram 28-Port Mode MUNICH256 .....	26
2.3	Pin Definition and functions .....	27
2.4	PCI Bus Interface .....	28
2.5	SPI Interface .....	34
2.6	Local Microprocessor Interface .....	35
2.7	Serial Interface 16-port mode .....	39
2.8	Serial Interface 28-port mode .....	41
2.9	Test Interface .....	43
2.10	Power Supply and No-connect Pins .....	44
<b>3</b>	<b>General Overview</b> .....	46
3.1	Functional Overview .....	46
3.2	Block Diagram .....	47
3.3	Internal Interface .....	47
3.4	Block Description .....	48
<b>4</b>	<b>Functional Description</b> .....	51
4.1	Port Handler .....	51
4.1.1	Selectable port configuration .....	51
4.1.2	External Timing Mode .....	52
4.1.3	Local Port Loop .....	53
4.1.4	Remote Payload Loop .....	54
4.1.5	Remote Channel Loopback .....	55
4.1.6	Test Breakout .....	56
4.2	Time slot Handler .....	57
4.2.1	Channelized Modes .....	57
4.2.2	Unchannelized Mode .....	58
4.3	Data Management Unit .....	59
4.3.1	Descriptor Concept .....	59
4.3.2	Receive Descriptor .....	60
4.3.3	Data Management Unit Receive .....	64
4.3.4	Transmit Descriptor .....	66
4.3.5	Data Management Unit Transmit .....	69
4.3.6	Byte Swapping .....	71
4.3.7	Transmission Bit/Byte Ordering .....	72
4.4	Buffer Management .....	72
4.4.1	Internal Receive Buffer .....	72

<b>Table of Contents</b>		<b>Page</b>
4.4.2	Internal Transmit Buffer .....	74
4.5	Protocol Description .....	76
4.5.1	HDLC Mode .....	76
4.5.2	Bit Synchronous PPP with HDLC Framing Structure .....	77
4.5.3	Octet Synchronous PPP .....	77
4.5.4	Transparent Mode .....	78
4.6	Mailbox .....	78
4.7	Interrupt Controller .....	80
4.7.1	Layer Two interrupts .....	80
4.7.1.1	General Interrupt Vector Structure .....	82
4.7.1.2	System Interrupts .....	84
4.7.1.3	Port Interrupts .....	85
4.7.1.4	Channel Interrupts .....	87
4.7.1.5	Command Interrupts .....	92
4.7.2	Mailbox Interrupts to the Local Bus .....	94
<b>5</b>	<b>Interface Description .....</b>	<b>96</b>
5.1	PCI Interface .....	96
5.1.1	PCI Read Transaction .....	96
5.1.2	PCI Write Transaction .....	97
5.2	SPI Interface (ROM Load Unit) .....	98
5.2.1	Accesses to a SPI EEPROM .....	99
5.2.2	SPI Read Sequence .....	99
5.2.3	SPI Write Sequence .....	100
5.3	Local Microprocessor Interface .....	101
5.3.1	Intel Mode .....	102
5.3.1.1	Slave Mode .....	102
5.3.1.2	Master Mode .....	102
5.3.2	Motorola Mode .....	105
5.3.2.1	Slave Mode .....	105
5.3.2.2	Master Mode .....	105
5.4	Serial Line Interface .....	107
5.4.1	Interface Timing in 16-port mode .....	108
5.4.2	Interface Timing in 28-port mode .....	111
5.5	JTAG Interface .....	112
<b>6</b>	<b>Channel Programming / Reprogramming Concept .....</b>	<b>115</b>
6.1	Channel Commands .....	116
6.2	Transmit Channel Commands .....	116
6.3	Receive Channel Commands .....	118
<b>7</b>	<b>Reset and Initialization procedure .....</b>	<b>121</b>
7.1	Chip Initialization .....	121
7.2	Mode Initialization .....	122

<b>Table of Contents</b>		<b>Page</b>
<b>8</b>	<b>Register Description</b> .....	123
8.1	Register Overview .....	123
8.1.1	PCI Configuration Register Set (Direct Access) .....	123
8.1.2	PCI Slave Register Set (Direct Access) .....	125
8.1.3	PCI and Local Bus Register Set (Direct Access) .....	127
8.2	Detailed Register Description .....	129
8.2.1	PCI Configuration Register .....	129
8.2.2	PCI Slave Register .....	144
8.2.3	Serial Interface Timing .....	161
8.2.3.1	Clock Input Timing .....	161
8.2.3.2	Transmit Cycle Timing .....	161
8.2.3.3	Transmit Synchronization Timing .....	161
8.2.3.4	Receive Cycle Timing .....	161
8.2.3.5	Receive Synchronization Timing .....	161
8.2.4	Interface Timing in 16-port mode .....	161
8.2.5	Interface Timing in 28-port mode .....	161
8.2.6	Mailbox Interrupts to the Local Bus .....	161
8.2.7	Selectable port configuration .....	161
8.2.8	External Timing Mode .....	161
8.2.9	Local Port Loop .....	161
8.2.10	Remote Payload Loop .....	161
8.2.11	Remote Channel Loopback .....	161
8.2.12	Test Breakout .....	161
8.3	Serial Interface 16-port mode .....	161
8.4	Serial Interface 28-port mode .....	161
8.5	Pin Diagram 16-Port Mode MUNICH256 .....	161
8.6	Pin Diagram 28-Port Mode MUNICH256 .....	161
8.7	General System Integration MUNICH256 .....	161
8.8	General Features .....	161
8.8.1	PCI and Local Bus Slave Register Set .....	194
<b>9</b>	<b>Electrical Characteristics</b> .....	203
9.1	Important Electrical Requirements .....	203
9.2	Absolute Maximum Ratings .....	203
9.3	DC Characteristics .....	203
9.4	AC Characteristics .....	205
9.4.1	PCI Bus Interface Timing .....	206
9.4.2	SPI Interface Timing .....	208
9.4.3	Local Microprocessor Interface Timing .....	209
9.4.3.1	Intel Bus Interface Timing (Slave Mode) .....	209
9.4.3.2	Intel Bus Interface Timing (Master Mode) .....	211
9.4.3.3	Motorola Bus Interface Timing (Slave Mode) .....	214
9.4.3.4	Motorola Bus Interface Timing (Master Mode) .....	216

9.4.4	Serial Interface Timing .....	220
9.4.4.1	Clock Input Timing .....	220
9.4.4.2	Transmit Cycle Timing .....	221
9.4.4.3	Transmit Synchronization Timing .....	222
9.4.4.4	Receive Cycle Timing .....	223
9.4.4.5	Receive Synchronization Timing .....	224
9.4.5	JTAG Interface Timing .....	225
9.4.6	Reset Timing .....	226
<b>10</b>	<b>Package Outline .....</b>	<b>227</b>
<b>11</b>	<b>List of Abbreviations .....</b>	<b>229</b>

<b>List of Figures</b>	<b>Page</b>
Figure 1-1	MUNICH256 16-port Mode Logic Symbol . . . . . 22
Figure 1-2	MUNICH256 28-port Mode Logic Symbol . . . . . 23
Figure 1-3	System Integration of the MUNICH256 . . . . . 24
Figure 2-1	MUNICH256 Pin Configuration 16-Port Mode . . . . . 25
Figure 2-2	MUNICH256 Pin Configuration 28-Port Mode . . . . . 26
Figure 3-1	MUNICH256 Block Diagram . . . . . 47
Figure 4-1	Port Configuration . . . . . 52
Figure 4-2	Local Port Loop . . . . . 53
Figure 4-3	Remote Payload Loop . . . . . 54
Figure 4-4	Remote Channel Loop . . . . . 55
Figure 4-5	Test Breakout . . . . . 56
Figure 4-6	Time slot Assignment in Channelized Modes . . . . . 58
Figure 4-7	Descriptor Structure . . . . . 60
Figure 4-8	Receive Buffer Thresholds . . . . . 74
Figure 4-9	Transmit Buffer Thresholds . . . . . 75
Figure 4-10	HDLC Frame Format . . . . . 76
Figure 4-11	Bit Synchronous PPP with HDLC Framing Structure . . . . . 77
Figure 4-12	Mailbox Structure . . . . . 79
Figure 4-13	Layer Two Interrupts (Channel, command, port and system interrupts) . . . . . 81
Figure 4-14	Interrupt Queue Structure in System Memory . . . . . 82
Figure 4-15	Mailbox Interrupt Notification . . . . . 94
Figure 5-1	PCI Read Transaction . . . . . 97
Figure 5-2	PCI Write Transaction . . . . . 98
Figure 5-3	SPI Read Sequence . . . . . 100
Figure 5-4	SPI Write Sequence . . . . . 100
Figure 5-5	Intel Bus Mode . . . . . 103
Figure 5-6	Intel Bus Arbitration . . . . . 103
Figure 5-7	Motorola Bus Mode . . . . . 106
Figure 5-8	Motorola Bus Arbitration . . . . . 106
Figure 5-9	Supported Frame Structures . . . . . 108
Figure 5-10	T1 Mode Frame Timing . . . . . 109
Figure 5-11	E1, 4.096 MHz and 8.192 MHz Interface Timing in 16-port mode . . . . . 110
Figure 5-12	Unchannelized Mode Interface Timing . . . . . 111
Figure 5-13	T1-mode Interface Timing in 28-port Mode . . . . . 111
Figure 5-14	E1-mode Interface Timing in 28-port Mode . . . . . 112
Figure 5-15	Block Diagram of Test Access Port and Boundary Scan Unit . . . . . 113
Figure 8-1	Clock Input Timing . . . . . 161
Figure 8-2	Transmit Cycle Timing . . . . . 161
Figure 8-3	Transmit Synchronization Timing . . . . . 161
Figure 8-4	Receive Cycle Timing . . . . . 161
Figure 8-5	Receive Synchronization Timing . . . . . 161
Figure 8-6	Supported Frame Structures . . . . . 161

<b>List of Figures</b>	<b>Page</b>
Figure 8-7	T1 Mode Frame Timing . . . . . 161
Figure 8-8	E1, 4.096 MHz and 8.192 MHz Interface Timing in 16-port mode . . . 161
Figure 8-9	Unchannelized Mode Interface Timing . . . . . 161
Figure 8-10	T1-mode Interface Timing in 28-port Mode . . . . . 161
Figure 8-11	E1-mode Interface Timing in 28-port Mode . . . . . 161
Figure 8-12	Mailbox Interrupt Notification . . . . . 161
Figure 8-13	Port Configuration . . . . . 161
Figure 8-14	Local Port Loop . . . . . 161
Figure 8-15	Remote Payload Loop . . . . . 161
Figure 8-16	Remote Channel Loop . . . . . 161
Figure 8-17	Test Breakout . . . . . 161
Figure 8-18	MUNICH256 Block Diagram . . . . . 161
Figure 8-19	MUNICH256 Pin Configuration 16-Port Mode . . . . . 161
Figure 8-20	MUNICH256 Pin Configuration 28-Port Mode . . . . . 161
Figure 8-21	System Integration of the MUNICH256 . . . . . 161
Figure 8-22	MUNICH256 16-port Mode Logic Symbol . . . . . 161
Figure 8-23	MUNICH256 28-port Mode Logic Symbol . . . . . 161
Figure 9-1	Input/Output Waveform for AC Tests . . . . . 205
Figure 9-2	PCI Clock Cycle Timing . . . . . 206
Figure 9-3	PCI Input Timing Measurement Conditions . . . . . 206
Figure 9-4	PCI Output Timing Measurement Conditions . . . . . 207
Figure 9-5	SPI Interface Timing . . . . . 208
Figure 9-6	Intel Read Cycle Timing (Slave Mode) . . . . . 209
Figure 9-7	Intel Write Cycle Timing (Slave Mode). . . . . 209
Figure 9-8	Intel Read Cycle Timing (Master Mode, LRDY controlled) . . . . . 211
Figure 9-9	Intel Write Cycle Timing (Master Mode, LRDY controlled). . . . . 211
Figure 9-10	Intel Read Cycle Timing (Master Mode, Wait state controlled) . . . . . 212
Figure 9-11	Intel Write Cycle Timing (Master Mode, Wait state controlled) . . . . . 212
Figure 9-12	Intel Bus Arbitration Timing . . . . . 213
Figure 9-13	Motorola Read Cycle Timing (Slave Mode) . . . . . 214
Figure 9-14	Motorola Write Cycle Timing (Slave Mode) . . . . . 214
Figure 9-15	Motorola Read Cycle Timing (Master Mode, LDTACK controlled). . . 216
Figure 9-16	Motorola Write Cycle Timing (Master Mode, LDTACK controlled). . . 216
Figure 9-17	Motorola Read Cycle Timing (Master Mode, Wait state controlled). . 217
Figure 9-18	Motorola Write Cycle Timing (Master Mode, Wait state controlled). . 217
Figure 9-19	Motorola Bus Arbitration Timing. . . . . 218
Figure 9-20	Clock Input Timing . . . . . 220
Figure 9-21	Transmit Cycle Timing . . . . . 221
Figure 9-22	Transmit Synchronization Timing. . . . . 222
Figure 9-23	Receive Cycle Timing . . . . . 223
Figure 9-24	Receive Synchronization Timing . . . . . 224
Figure 9-25	JTAG Interface Timing . . . . . 225

Figure 9-26 Reset Timing ..... 226





<b>List of Tables</b>	<b>Page</b>
Table 1-1	Interface Configuration . . . . . 20
Table 4-1	Interface configuration . . . . . 51
Table 4-2	Receive Descriptor Structure . . . . . 61
Table 4-3	Transmit Descriptor Structure . . . . . 67
Table 4-4	Example for little/big Endian with BNO = 3 . . . . . 72
Table 4-5	Example for little big Endian with BNO = 7 . . . . . 72
Table 4-6	Interrupt Vector Structure . . . . . 82
Table 5-1	Correspondence between PCI memory space and chip select . . . . . 101
Table 5-2	C/BE to LA/LBHE mapping in Intel bus mode (8 bit port mode) . . . . . 104
Table 5-3	C/BE to LA/LBHE mapping in Intel bus mode (16 bit port mode) . . . . . 104
Table 5-4	C/BE to LA/LSIZE0 mapping in Motorola bus mode (8 bit port mode) . . . . . 107
Table 5-5	C/BE to LA/LSIZE0 mapping in Motorola bus mode (16 bit port mode) . . . . . 107
Table 6-1	Channel Specification Registers and Channel Commands . . . . . 115
Table 8-1	PCI Configuration Register Set . . . . . 123
Table 8-2	PCI Slave Register Set . . . . . 125
Table 8-3	PCI and Local Bus Slave Register Set . . . . . 127
Table 8-4	Threshold Codings . . . . . 155
Table 8-5	Clock Input Timing . . . . . 161
Table 8-6	Transmit Cycle Timing . . . . . 161
Table 8-7	Transmit Synchronization Timing . . . . . 161
Table 8-8	Receive Cycle Timing . . . . . 161
Table 8-9	Receive Synchronization Timing . . . . . 161
Table 8-10	Bit Shift . . . . . 161
Table 8-11	Interface configuration . . . . . 161
Table 8-12	Bit Shift . . . . . 172
Table 9-1	Absolute Maximum Ratings . . . . . 203
Table 9-2	DC Characteristics . . . . . 203
Table 9-3	DC Characteristics (Non-PCI Interface Pins) . . . . . 204
Table 9-4	DC Characteristics (PCI Interface Pins) . . . . . 205
Table 9-5	PCI Clock Characteristics . . . . . 206
Table 9-6	PCI Interface Signal Characteristics . . . . . 207
Table 9-7	SPI Interface Timing . . . . . 208
Table 9-8	Intel Bus Interface Timing . . . . . 210
Table 9-9	Intel Bus Interface Timing (Master Mode) . . . . . 213
Table 9-10	Motorola Bus Interface Timing . . . . . 215
Table 9-11	Motorola Bus Interface Timing (Master Mode) . . . . . 218
Table 9-12	Clock Input Timing . . . . . 220
Table 9-13	Transmit Cycle Timing . . . . . 221
Table 9-14	Transmit Synchronization Timing . . . . . 222
Table 9-15	Receive Cycle Timing . . . . . 223
Table 9-16	Receive Synchronization Timing . . . . . 224

Table 9-17	JTAG Interface Timing . . . . .	225
Table 9-18	Reset Timing . . . . .	226



## 1 MUNICH256 Overview

The MUNICH256 is a highly integrated protocol controller that implements HDLC, PPP and transparent (TMA) protocol processing for 256 channels. An on-chip data management unit is optimized to transfer data packets via a PCI interface by minimizing the bus load.

The serial interface of the device can be configured in a 16-port mode and additionally in a 28-port mode. The 16-port mode provides a clock pin, a data pin and a frame synchronization pin for each port and direction. The 28-port mode provides a clock pin and a data pin per port and direction. In this mode frame boundaries are indicated by clock gaps. **Table 1-1** below shows the pin configuration and the supported frame structures in the 16-port mode and the 28-port mode.

**Table 1-1 Interface Configuration**

	Port mode	
	16-port mode	28-port mode
<b>Supported Interfaces</b>		
1.544 MBit/s channelized	x	x
2.048 MBit/s channelized	x	x
4.096 MBit/s channelized	x	
8.192 MBit/s channelized	x	
Unchannelized	x	x
<b>Supported Pins</b>		
Receive Data	x	x
Receive Clock	x	x
Receive Synchronization Pulse	x	
Transmit Clock	x	x
Transmit Data	x	x
Transmit Synchronization Pulse	x	
<b>Frame Indication</b>		
Gapped Clock		x
Synchronization Pulse	x	

## 1.1 General Features

- Configurable port interface which operates in 16-port mode or 28-port mode.
- In 16-port mode protocol processing on up to 16 T1, E1, channelized 4 MBit/s, channelized 8 MBit/s or unchannelized links for frame relay, router or DSLAM applications with a maximum aggregate data rate of up to 90 Mbit/s per direction at 66 MHz PCI frequency
- In 28-port mode protocol processing on up to 28 T1, E1 or unchannelized links. T1, E1 frame boundaries are indicated by clock gaps
- Support of 256 bidirectional channels, which can be assigned arbitrarily to a maximum of 16 links, for HDLC, PPP or transparent mode (TMA) processing
- Concatenation of any, not necessarily consecutive, time slots to logical channels on each physical link. Supports DS0, fractional T1/E1 or T1/E1 channels
- Additional support of unchannelized modes, with data rates of up to 45 Mbit/s on port zero and 8.192 Mbit/s on all other ports
- Provides 32kB data buffer in transmit direction and 12kB data buffer in receive direction
- Independently selectable pay load loops for each port
- Provides a test function which allows to switch one out of 16 (28) ports to a test port
- System interface is a PCI 32 bit, 66 MHz Rev. 2.1 compliant bus interface, which supports configuration of subsystem ID / subsystem vendor ID via a serial EEPROM interface
- Integrates a local microprocessor master and slave interface (demultiplexed 16 bit address and data bus in Intel mode or Motorola mode) which allows access to the local bus via the PCI bus or which can communicate with a PCI host processor through an on-chip mailbox
- JTAG boundary scan according to IEEE1149.1 (5 pins)
- 0.25  $\mu\text{m}$ , 2.5V core technology
- I/Os are 3.3V tolerant and have 3.3V driving capability
- Package P-BGA 388 (35mm x 35mm, pitch 1.27mm)
- Full scan path and BIST of on-chip RAMs for production test
- Performance: 90 Mbit/s data throughput per direction at 66 MHz
- Estimated power consumption: 3W at 66 MHz
- Also available as device with extended temperature range -40..+85°C

## 1.2 Logic Symbol

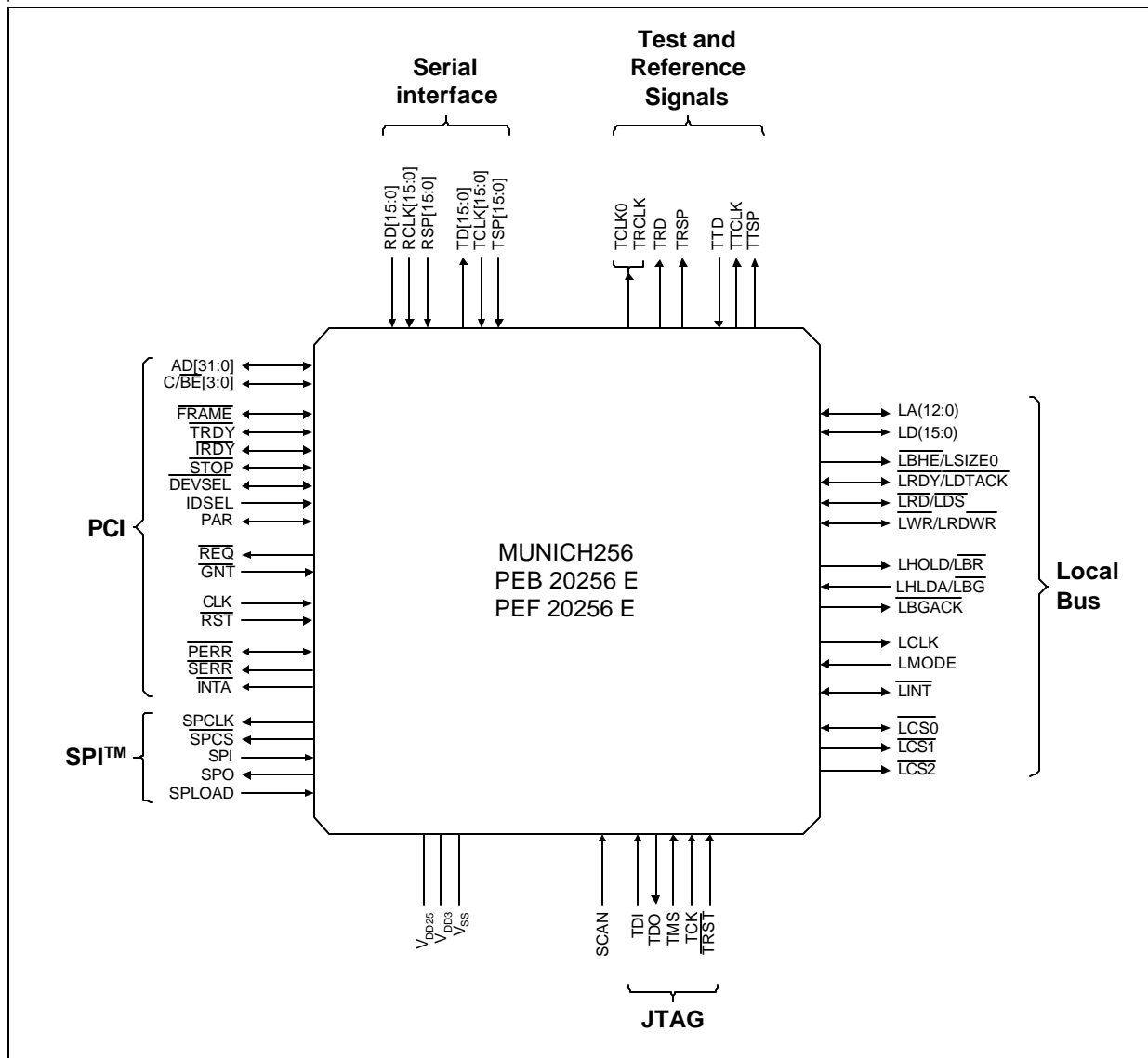


Figure 1-1 MUNICH256 16-port Mode Logic Symbol

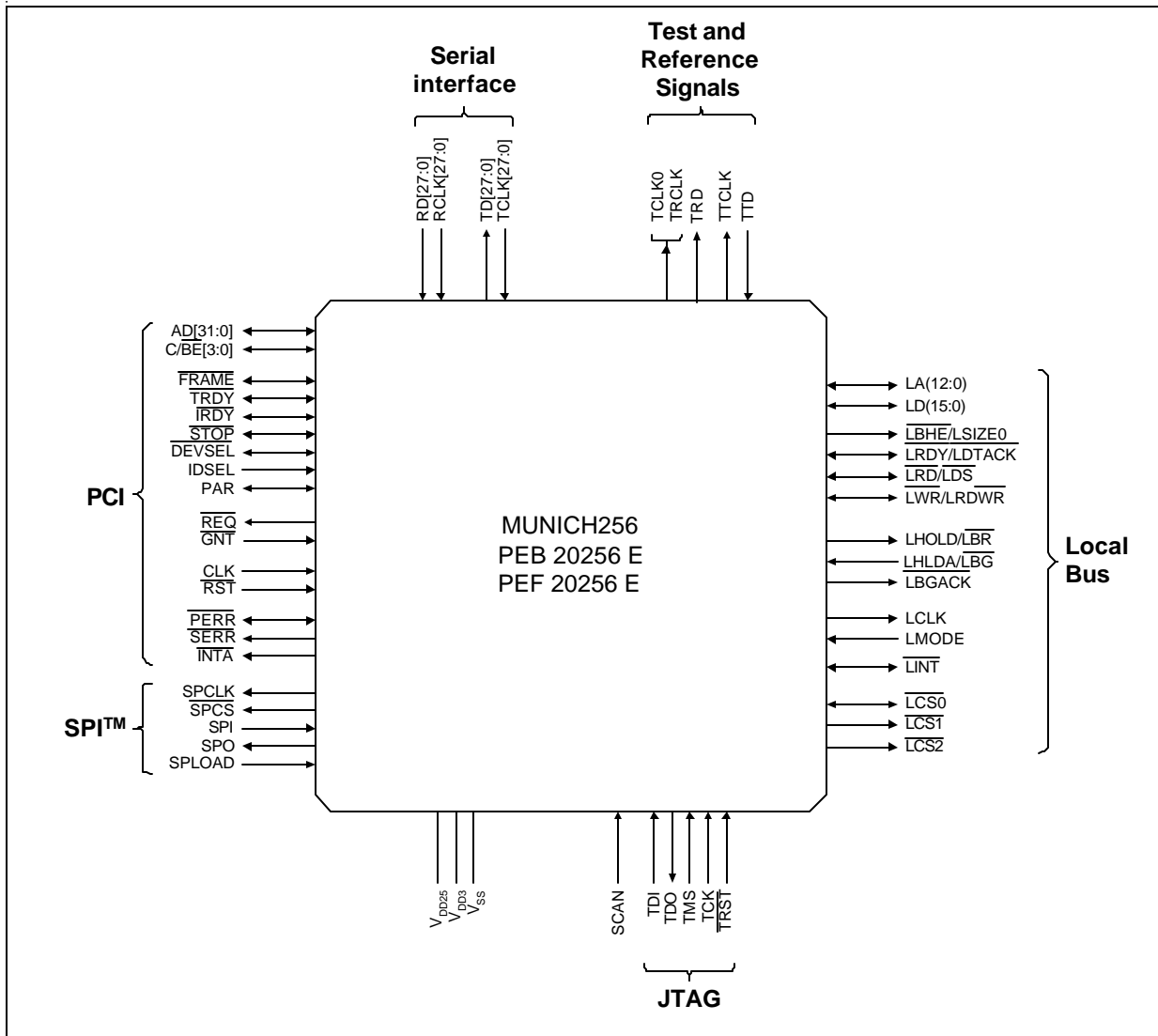


Figure 1-2 MUNICH256 28-port Mode Logic Symbol

### 1.3 General System Integration MUNICH256

The MUNICH256 provides the HDLC/PPP or transparent (TMA) protocol handling for channelized or unchannelized applications with up to 16 links. Protocol data is transferred to the packet RAM via the PCI bus and handled (e.g. for layer3 protocol handling) by a central CPU. An integrated mailbox allows to exchange information between a local CPU and the line card processor.

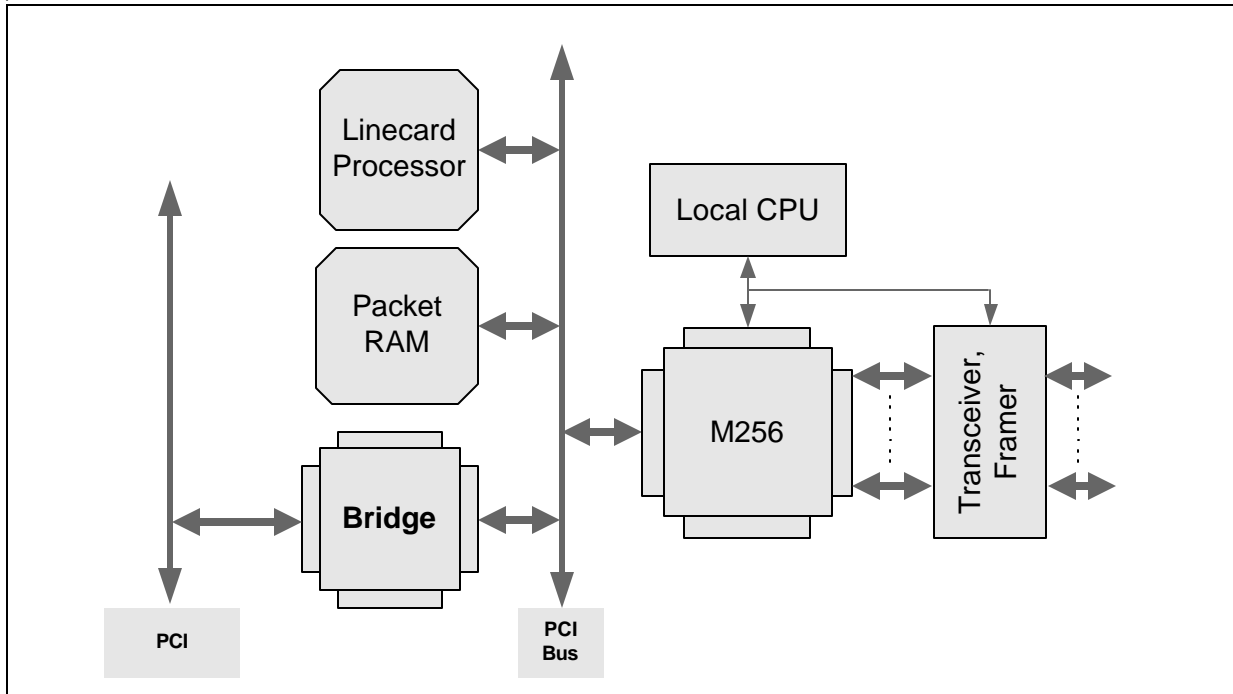


Figure 1-3 System Integration of the MUNICH256



## 2 Pin Description

### 2.1 Pin Diagram 16-Port Mode MUNICH256

(Top view)

	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																																				
AF	VSS	LD(8)	NC22	LD(5)	VDD25	LD(11)	LD(13)	LA(1)	VSS	LA(4)	LA(8)	VDD25	LA(10)	LA(11)	VDD25	AD(0)	AD(5)	VSS	AD(6)	AD(9)	AD(12)	VDD25	PAR	$\overline{\text{STOP}}$	NC24	VSS																																				
AE	LD(2)	VDD25	LD(4)	NC23	NC20	LD(8)	VSS	LD(12)	LA(0)	VDD25	LA(5)	VSS	LA(9)	LA(12)	VSS	AD(2)	VDD25	$\frac{C}{BE(0)}$	AD(10)	VSS	AD(14)	$\overline{\text{SERR}}$	$\overline{\text{DE}}/SE/L$	NC25	VDD25	NC28																																				
AD	VSS	LD(1)	VSS	NC17	NC18	VDD3	LD(7)	LD(9)	VDD3	LD(14)	LD(2)	LA(6)	VDD3	LBHE/LSIZE0	AD(1)	AD(4)	AD(8)	VDD3	AD(13)	$\frac{C}{BE(1)}$	VDD3	$\overline{\text{TRDY}}$	NC27	VSS	NC29	AD(17)																																				
AC	$\overline{\text{LINT}}$	$\overline{\text{LCS2}}$	$\overline{\text{LRD}}/LDS$	VDD3	NC16	NC19	NC21	LD(6)	LD(10)	VDD3	LD(15)	LA(3)	LA(7)	$\overline{\text{NTA}}$	AD(3)	AD(7)	VDD3	AD(11)	AD(15)	$\overline{\text{FERR}}$	$\overline{\text{IRDY}}$	NC26	VDD3	NC31	AD(16)	AD(21)																																				
AB	VDD25	UHLDA/LBG	$\overline{\text{LWR}}/LRD/TWR$	LD(0)	<table border="1" style="margin: auto;"> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> </table>																		VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	NC30	$\overline{\text{FRAM}}/E$	AD(20)	VDD25
VSS	VSS	VSS	VSS	VSS																			VSS																																							
VSS	VSS	VSS	VSS	VSS																			VSS																																							
VSS	VSS	VSS	VSS	VSS																			VSS																																							
VSS	VSS	VSS	VSS	VSS																			VSS																																							
VSS	VSS	VSS	VSS	VSS																			VSS																																							
VSS	VSS	VSS	VSS	VSS																			VSS																																							
AA	RD(1)	U-HOLD/LBER	VDD3	$\overline{\text{LRDY}}$																			$\frac{C}{BE(2)}$	VDD3	AD(23)	DSEL																																				
Y	VDD25	VSS	LCLK	$\overline{\text{LCS0}}$																			AD(18)	AD(19)	VSS	VDD25																																				
W	RD(2)	RCLK(1)	U-MODE	$\overline{\text{LCS1}}$																			AD(22)	VDD3	AD(25)	AD(26)																																				
V	VSS	TCLK(14)	VDD3	$\overline{\text{LBGAC}}/K$																			AD24	$\frac{C}{BE(3)}$	AD(27)	VSS																																				
U	TCLK(13)	VDD25	RD(3)	VDD3																			VDD3	AD(28)	VDD25	AD(29)																																				
T	TCLK(9)	TCLK(11)	RCLK(3)	RCLK(2)																			AD(30)	AD(31)	$\overline{\text{REQ}}$	$\overline{\text{GNT}}$																																				
R	VDD25	VSS	TCLK(12)	TCLK(15)																			CLK	$\overline{\text{RST}}$	VSS	VDD25																																				
P	TCLK(7)	TCLK(8)	TCLK(10)	VDD3																			SPL0A/D	VDD3	SPI	SPO																																				
N	TTCLK	TCLK(6)	VDD3	TCLK(5)																			SPCLK	$\overline{\text{SPCS}}$	RSP(15)	TSP(15)																																				
M	VDD25	VSS	TRD	TD(14)	RSP(14)	TSP(14)	VSS	VDD25																																																						
L	TCLK(4)	TD(13)	VDD3	TD(11)	RSP(12)	TSP(12)	TSP(13)	RSP(13)																																																						
K	TD(15)	VDD25	RCLK(4)	VDD3	RES14	RES15	VDD25	RES16																																																						
J	VSS	RD(4)	VDD3	RD(6)	RES11	VDD3	RES13	VSS																																																						
H	TD(12)	RD(5)	RCLK(6)	VDD3	RSP(11)	RES9	RES10	RES12																																																						
G	VDD25	VSS	RCLK(7)	RCLK(9)	RSP(9)	RSP(10)	VSS	VDD25																																																						
F	RCLK(5)	RD(7)	VDD3	TMS	TSP(8)	VDD3	TSP(10)	TSP(11)																																																						
E	VDD25	RD(8)	SCAN	NC12	NC0	RES7	TSP(9)	VDD25																																																						
D	RD(8)	VSS	TDO	VDD3	NC15	NC9	TD(10)	TD(6)	RD(10)	VDD3	RCLK(13)	TD(0)	TCLK(1)	RCLK(14)	RES20	TSP(0)	VDD3	RSP(2)	RSP(4)	TSP(6)	RSP(7)	RES3	VDD3	NC7	RES8	RSP(8)																																				
C	RCLK(9)	TOK	VSS	NC14	NC10	VDD3	TD(7)	TD(4)	VDD3	RD(12)	TD(2)	$\overline{\text{TCLKO}}/TRCLK$	TCLK(2)	VDD3	TTD	RCLK(6)	TSP(1)	TSP(2)	TSP(4)	RSP(5)	VDD3	TTSP	RES6	VSS	NC3	NC1																																				
B	$\overline{\text{TRST}}$	VDD25	NC13	NC8	TD(9)	TD(5)	VSS	RCLK(10)	RCLK(11)	VDD25	TD(1)	VSS	TCLK(3)	RD(15)	VSS	RD(0)	VDD25	RSP(1)	RSP(3)	VSS	TSP(7)	TRSP	RES5	NC6	VDD25	NC2																																				
A	VSS	NC11	TDI	TD(8)	VDD25	TD(3)	RD(11)	RCLK(12)	VSS	RD(13)	TCLK(0)	VDD25	RD(14)	RCLK(15)	VDD25	RES21	RSP(0)	VSS	TSP(3)	TSP(5)	RSP(6)	VDD25	RES4	NC4	NC5	VSS																																				

Figure 2-1 MUNICH256 Pin Configuration 16-Port Mode

## 2.2 Pin Diagram 28-Port Mode MUNICH256

(Top view)

	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1																												
AF	VSS	LD(3)	NC22	LD(5)	VDD25	LD(11)	LD(13)	LA(1)	VSS	LA(4)	LA(8)	VDD25	LA(10)	LA(11)	VDD25	AD(0)	AD(5)	VSS	AD(6)	AD(9)	AD(12)	VDD25	PAR	STOP	NC24	VSS																												
AE	LD(2)	VDD25	LD(4)	NC23	NC20	LD(8)	VSS	LD(12)	LA(0)	VDD25	LA(5)	VSS	LA(9)	LA(12)	VSS	AD(2)	VDD25	C/BE(0)	AD(10)	VSS	AD(14)	SERR	DEVSEL	NC25	VDD25	NC28																												
AD	VSS	LD(1)	VSS	NC17	NC18	VDD3	LD(7)	LD(9)	VDD3	LD(14)	LD(2)	LA(6)	VDD3	LBHE/LS2E0	AD(1)	AD(4)	AD(8)	VDD3	AD(13)	C/BE(1)	VDD3	TRDY	NC27	VSS	NC29	AD(17)																												
AC	UINT	UCSS	TRDY/TDS	VDD3	NC16	NC19	NC21	LD(6)	LD(10)	VDD3	LD(15)	LA(3)	LA(7)	INTA	AD(9)	AD(7)	VDD3	AD(11)	AD(15)	PERR	TRDY	NC26	VDD3	NC31	AD(16)	AD(21)																												
AB	VDD25	LHLDV/LBS	LWR/LRD/WR	LD(0)																			NC30	FRAME	AD(20)	VDD25																												
AA	RD(1)	LHOLD/LBR	VDD3	LRDY																			C/BE(2)	VDD3	AD(23)	DSEL																												
Y	VDD25	VSS	LCLK	LCSD																			AD(18)	AD(19)	VSS	VDD25																												
W	RD(2)	RCLK(1)	LMODE	LCSD																			AD(22)	VDD3	AD(25)	AD(26)																												
V	VSS	TD(26)	VDD3	LBGRACK																			AD(4)	C/BE(3)	AD(27)	VSS																												
U	TD(25)	VDD25	RD(3)	VDD3																			VDD3	AD(28)	VDD25	AD(29)																												
T	TD(21)	TD(23)	RCLK(3)	RCLK(2)	<table border="1"> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> <tr><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td><td>VSS</td></tr> </table>						VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS											AD(30)	AD(31)	REG	GNT
VSS	VSS	VSS	VSS	VSS	VSS																																																	
VSS	VSS	VSS	VSS	VSS	VSS																																																	
VSS	VSS	VSS	VSS	VSS	VSS																																																	
VSS	VSS	VSS	VSS	VSS	VSS																																																	
VSS	VSS	VSS	VSS	VSS	VSS																																																	
R	VDD25	VSS	TD(24)	TD(27)																			CLK	RST	VSS	VDD25																												
P	TD(19)	TD(20)	TD(22)	VDD3																			SPLLOAD	VDD3	SPI	SPO																												
N	TTCLK	TD(18)	VDD3	TD(17)																			SPCLK	SPCS	RCLK(27)	RD(27)																												
M	VDD25	VSS	TRD	TD(14)																			RCLK(26)	RD(26)	VSS	VDD25																												
L	TD(16)	TD(13)	VDD3	TD(11)																			RCLK(24)	RD(24)	RD(25)	RCLK(25)																												
K	TD(15)	VDD25	RCLK(4)	VDD3																			TCLK(25)	TCLK(26)	VDD25	TCLK(27)																												
J	VSS	RD(4)	VDD3	RD(6)																			TCLK(22)	VDD3	TCLK(24)	VSS																												
H	TD(12)	RD(5)	RCLK(6)	VDD3																			RCLK(23)	TCLK(20)	TCLK(21)	TCLK(23)																												
G	VDD25	VSS	RCLK(7)	RCLK(8)																			RCLK(21)	RCLK(22)	VSS	VDD25																												
F	RCLK(5)	RD(7)	VDD3	TMS																			RD(20)	VDD3	RD(22)	RD(23)																												
E	VDD25	RD(9)	SCAN	NC12																			NC0	TCLK(18)	RD(21)	VDD25																												
D	RD(8)	VSS	TDO	VDD3	NC15	NC9	TD(10)	TD(6)	RD(10)	VDD3	RCLK(13)	TD(0)	TCLK(1)	RCLK(14)	RES20	TCLK(4)	VDD3	TCLK(9)	RCLK(16)	RD(18)	RCLK(19)	TCLK(14)	VDD3	NC7	TCLK(19)	RCLK(20)																												
C	RCLK(8)	TCK	VSS	NC14	NC10	VDD3	TD(7)	TD(4)	VDD3	RD(12)	TD(2)	TCLKO/TRCLK	TCLK(2)	VDD3	TTD	RCLK(8)	TCLK(6)	TCLK(8)	RD(16)	RCLK(17)	VDD3	TCLK(13)	TCLK(17)	VSS	NC3	NC1																												
B	TRST	VDD25	NC13	NC8	TD(8)	TD(5)	VSS	RCLK(10)	RCLK(11)	VDD25	TD(1)	VSS	TCLK(3)	RD(15)	VSS	RD(0)	VDD25	TCLK(7)	TCLK(11)	VSS	RD(19)	TCLK(12)	TCLK(16)	NC6	VDD25	NC2																												
A	VSS	NC11	TDI	TD(8)	VDD25	TD(3)	RD(11)	RCLK(12)	VSS	RD(13)	TCLK(0)	VDD25	RD(14)	RCLK(15)	VDD25	RES21	TCLK(5)	VSS	TCLK(10)	RD(17)	RCLK(18)	VDD25	TCLK(15)	NC4	NC5	VSS																												

Figure 2-2 MUNICH256 Pin Configuration 28-Port Mode

## 2.3 Pin Definition and functions

### Signal Type Definitions:

The following signal type definitions are partly taken from the PCI Specification Rev. 2. 1:

- I** *Input* is a standard input- only signal.
- O** *Totem Pole Output* is a standard active driver.
- t/s, I/O** *Tri-State or I/O* is a bidirectional, tri-state input/output pin.
- s/t/s** *Sustained Tri-State* is an active low tri-state signal owned and driven by one and only agent at a time. The agent that drives an s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.
- o/d** *Open Drain* allows multiple devices to share a line as a wire-OR. A pull-up is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

### Signal Name Conventions:

- NCn** *No-connect Pin n*
- Such pins are not bonded with the silicon. Although any potential at these pins will not impact the device it is recommended to leave them unconnected. No-connect pins might be used for additional functionality in later versions of the device. Leaving them unconnected will guarantee hardware compatibility to later device versions.
- Reserved** Reserved pins are for vendor specific use only and should be connected as recommended to guarantee normal operation.

*Note: The signal type definition specifies the functional usage of a pin. This does not reflect necessarily the implementation of a pin, e.g. a pin defined of signal type 'Input' may be implemented with a bidirectional pad.*

## 2.4 PCI Bus Interface

Pin No.	Symbol	Input (I) Output (O)	Function
T3, T4, U1, U3, V2, W1, W2, V4, AA2, W4, AC1, AB2, Y3, Y4, AD1, AC2, AC8, AE6, AD8, AF6, AC9, AE8, AF7, AD10, AC11, AF8, AF10, AD11, AC12, AE11, AD12, AF11	AD(31:0)	t/s	<p><b>Address/Data Bus</b></p> <p>A bus transaction consists of an address phase followed by one or more data phases.</p> <p>When the MUNICH256 is the bus master, AD(31:0) are outputs in the address phase of a transaction. During the data phases, AD(31:0) remain outputs for write transactions, and become inputs for read transactions.</p> <p>When the MUNICH256 is bus slave, AD(31:0) are inputs in the address phase of a transaction. During the data phases, AD(31:0) remain inputs for write transactions, and become outputs for read transactions.</p> <p>AD(31:0) are tri-state when the MUNICH256 is not involved in the current transaction.</p> <p>AD(31:0) are updated and sampled on the rising edge of CLK.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
V3, AA4, AD7, AE9	$\overline{C/BE}(3:0)$	t/s	<p><b>Command/Byte Enable</b></p> <p>During the address phase of a transaction, <math>\overline{C/BE}(3:0)</math> define the bus command. During the data phase, <math>\overline{C/BE}(3:0)</math> are used as byte enable lines. The byte enable lines are valid for the entire data phase and determine which byte lanes carry meaningful data. <math>\overline{C/BE}(0)</math> applies to byte 0 (LSB) and <math>\overline{C/BE}(3)</math> applies to byte 3 (MSB).</p> <p>When the MUNICH256 is bus master, <math>\overline{C/BE}(3:0)</math> are outputs.</p> <p>When the MUNICH256 is bus slave, <math>\overline{C/BE}(3:0)</math> are inputs.</p> <p><math>\overline{C/BE}(3:0)</math> are tri-stated when the MUNICH256 is not involved in the current transaction.</p> <p><math>\overline{C/BE}(3:0)</math> are updated and sampled on the rising edge of CLK.</p>
AF4	PAR	t/s	<p><b>Parity</b></p> <p><math>\overline{PAR}</math> is even parity across AD(31:0) and <math>\overline{C/BE}(3:0)</math>. <math>\overline{PAR}</math> is stable and valid one clock after the address phase. <math>\overline{PAR}</math> has the same timing as AD(31:0) but delayed by one clock.</p> <p>When the MUNICH256 is Master, <math>\overline{PAR}</math> is output during address phase and write data phases and input during read data phase. When the MUNICH256 is Slave, <math>\overline{PAR}</math> is output during read data phase and input during write data phase.</p> <p><math>\overline{PAR}</math> is tri-stated when the MUNICH256 is not involved in the current transaction.</p> <p>Parity errors detected by the device are indicated on <math>\overline{PERR}</math> output.</p> <p><math>\overline{PAR}</math> is updated and sampled on the rising edge of CLK.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AB3	$\overline{\text{FRAME}}$	s/t/s	<p><b>Frame</b>  <math>\overline{\text{FRAME}}</math> indicates the beginning and end of an access. <math>\overline{\text{FRAME}}</math> is asserted to indicate a bus transaction is beginning. While <math>\overline{\text{FRAME}}</math> is asserted, data transfers continue. When <math>\overline{\text{FRAME}}</math> is deasserted, the transaction is in the final phase. When the MUNICH256 is bus master, <math>\overline{\text{FRAME}}</math> is an output. When the MUNICH256 is bus slave, <math>\overline{\text{FRAME}}</math> is an input. <math>\overline{\text{FRAME}}</math> is tri-stated when the MUNICH256 is not involved in the current transaction. <math>\overline{\text{FRAME}}</math> is updated and sampled on the rising edge of CLK.</p>
AC6	$\overline{\text{IRDY}}$	s/t/s	<p><b>Initiator Ready</b>  <math>\overline{\text{IRDY}}</math> indicates the bus master's ability to complete the current data phase of the transaction. It is used in conjunction with <math>\overline{\text{TRDY}}</math>. A data phase is completed on any clock where both <math>\overline{\text{IRDY}}</math> and <math>\overline{\text{TRDY}}</math> are sampled asserted. During a write, <math>\overline{\text{IRDY}}</math> indicates that valid data is present on AD(31:0). During a read, it indicates the master is prepared to accept data. Wait cycles are inserted until both <math>\overline{\text{IRDY}}</math> and <math>\overline{\text{TRDY}}</math> are asserted together. When the MUNICH256 is bus master, <math>\overline{\text{IRDY}}</math> is an output. When the MUNICH256 is bus slave, <math>\overline{\text{IRDY}}</math> is an input. <math>\overline{\text{IRDY}}</math> is tri-stated, when the MUNICH256 is not involved in the current transaction. <math>\overline{\text{IRDY}}</math> is updated and sampled on the rising edge of CLK.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AD5	$\overline{\text{TRDY}}$	s/t/s	<p><b>Target Ready</b>  <math>\overline{\text{TRDY}}</math> indicates a slave's ability to complete the current data phase of the transaction. During a read, <math>\overline{\text{TRDY}}</math> indicates that valid data is present on AD(31:0). During a write, it indicates the target is prepared to accept data.            When the MUNICH256 is Master, <math>\overline{\text{TRDY}}</math> is an <u>input</u>. When the MUNICH256 is Slave, <math>\overline{\text{TRDY}}</math> is an output. <math>\overline{\text{TRDY}}</math> is tri-stated, when the MUNICH256 is not involved in the current transaction.  <math>\overline{\text{TRDY}}</math> is updated and sampled on the rising edge of CLK.</p>
AF3	$\overline{\text{STOP}}$	s/t/s	<p><b>Stop</b>  <math>\overline{\text{STOP}}</math> is used by a slave to request the current master to stop the current bus transaction.            When the MUNICH256 is bus master, <math>\overline{\text{STOP}}</math> is an <u>input</u>. When the MUNICH256 is bus slave, <math>\overline{\text{STOP}}</math> is an output. <math>\overline{\text{STOP}}</math> is tri-stated, when the MUNICH256 is not involved in the current transaction.  <math>\overline{\text{STOP}}</math> is updated and sampled on the rising edge of CLK.</p>
AA1	IDSEL	I	<p><b>Initialization Device Select</b>            When the MUNICH256 is slave in a transaction, where IDSEL is active in the address phase and C/BE(3:0) indicates an configuration read or write, the MUNICH256 assumes a read or write to a configuration register. In response, the MUNICH256 asserts DEVSEL during the subsequent CLK cycle.            IDSEL is sampled on the rising edge of CLK.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AE4	$\overline{\text{DEVSEL}}$	s/t/s	<p><b>Device Select</b></p> <p>When activated by a slave, it indicates to the current bus master that the slave has decoded its address as the target of the current <u>transaction</u>. If no bus slave activates <math>\overline{\text{DEVSEL}}</math> within six bus CLK cycles, the master should abort the transaction.</p> <p>When the <u>MUNICH256</u> is bus master, <math>\overline{\text{DEVSEL}}</math> is input. If <math>\overline{\text{DEVSEL}}</math> is not activated within six clock cycles after an address is output on AD(31:0), the <u>MUNICH256</u> aborts the transaction.</p> <p>When the <u>MUNICH256</u> is bus slave, <math>\overline{\text{DEVSEL}}</math> is output. <math>\overline{\text{DEVSEL}}</math> is tri-stated, when the <u>MUNICH256</u> is not involved in the current transaction.</p>
AC7	$\overline{\text{PERR}}$	s/t/s	<p><b>Parity Error</b></p> <p>When activated, indicates a parity error over the AD(31:0) and C/<math>\overline{\text{BE}}</math>(3:0) signals (compared to the PAR input). It has a delay of two CLK cycles with respect to AD and C/<math>\overline{\text{BE}}</math>(3:0) (i.e., it is valid for the cycle immediately following the corresponding PAR cycle).</p> <p><math>\overline{\text{PERR}}</math> is asserted relative to the rising edge of CLK.</p>
AE5	$\overline{\text{SERR}}$	o/d	<p><b>System Error</b></p> <p>The <u>MUNICH256</u> asserts this signal to indicate an address parity error and report a <u>fatal system error</u>.</p> <p><math>\overline{\text{SERR}}</math> is an open drain output activated on the rising edge of CLK.</p>
T2	$\overline{\text{REQ}}$	t/s	<p><b>Request</b></p> <p>Used by the <u>MUNICH256</u> to request control of the PCI bus. It is tri-state during <u>reset</u>.</p> <p><math>\overline{\text{REQ}}</math> is activated on the rising edge of CLK.</p>



Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
T1	$\overline{\text{GNT}}$	I	<p><b>Grant</b></p> <p>This signal is asserted by the arbiter to grant control of the PCI to the MUNICH256 in response to a bus request via <math>\overline{\text{REQ}}</math>. After <math>\overline{\text{GNT}}</math> is asserted, the MUNICH256 will begin a bus transaction only after the current bus Master has deasserted the <math>\overline{\text{FRAME}}</math> signal.</p> <p><math>\overline{\text{GNT}}</math> is sampled on the rising edge of CLK.</p>
R4	CLK	I	<p><b>Clock</b></p> <p>Provides timing for all PCI transactions. Most PCI signals are sampled or output relative to the rising edge of CLK. The PCI clock is used as internal system clock. The maximum CLK frequency is 66 MHz.</p>
R3	$\overline{\text{RST}}$	I	<p><b>Reset</b></p> <p>An active <math>\overline{\text{RST}}</math> signal brings all PCI registers, sequencers and signals into a consistent state. All PCI output signals are driven to high impedance.</p>
AC13	$\overline{\text{INTA}}$	o/d	<p><b>Interrupt Request</b></p> <p>When an interrupt status is active and unmasked, the MUNICH256 activates this open-drain output.</p>

## 2.5 SPI Interface

Pin No.	Symbol	Input (I) Output (O)	Function
P2	SPI	I	<b>SPI Serial Input</b> SPI is a data input pin, where data coming from an external EEPROM is shifted in. SPI is sampled on the rising edge of SPCLK. A pull-up resistor is recommended if the SPI interface is not used.
P1	SPO	O	<b>SPI Serial Output</b> SPO is a push/pull serial data output pin. Opcodes, byte addresses and data is updated on the falling edge of SPCLK. It is tri-state during reset.
N4	SPCLK	O	<b>SPI Clock Signal</b> SPCLK controls the serial bus timing of the SPI bus. SPCLK is derived from the PCI bus clock with a frequency of 1/78 of the PCI bus clock. It is tri-state during reset.
N3	$\overline{\text{SPCS}}$	O	<b>SPI Chip Select</b> SPCS is used to select an external EEPROM. It is tri-state during reset.
P4	SPLOAD	I	<b>Enable SPI Load Functionality</b> Connecting SPLOAD to $V_{DD3}$ enables the SPI bus after reset. In this case parts of the PCI configuration space can be configured via an external EEPROM.

## 2.6 Local Microprocessor Interface

Pin No.	Symbol	Input (I) Output (O)	Function
W24	LMODE	I	<b>Local Bus Mode</b> By connecting this pin to either $V_{SS}$ or $V_{DD3}$ the bus interface can be adapted to either Intel or Motorola environment. LMODE = $V_{SS}$ selects Intel bus mode. LMODE = $V_{DD3}$ selects Motorola bus mode.
Y24	LCLK	O	<b>Local Clock</b> Reference output clock derived from the PCI clock.
AE13, AF13, AF14, AE14, AF16, AC14, AD15, AE16, AF17, AC15, AD16, AF19, AE18	LA(12:0)	I/O	<b>Address bus</b> These input address lines select one of the internal registers for read or write access. <i>Note: Only LA(7:0) are evaluated during read/write accesses to the MUNICH256.</i> In local bus master mode the address lines are output. If local bus master functionality is disabled these pins are input only.
AC16, AD17, AF20, AE19, AF21, AC18, AD19, AE21, AD20, AC19, AF23, AE24, AF25, AE26, AD25, AB23	LD(15:0)	I/O	<b>Data Bus</b> Bidirectional tri-state data lines.
Y23	$\overline{\text{LCS0}}$	I	<b>Chip Select</b> This active low signal selects the MUNICH256 as bus slave for read/write operations.

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AC24	$\overline{\text{LRD}}$	I/O	<b>Read (Intel Bus Mode)</b> This active low signal selects a read transaction.
	or $\overline{\text{LDS}}$	I/O	<b>Data strobe (Motorola Bus Mode)</b> This active low signal indicates that valid data has to be placed on the data bus (read cycle) or that valid data has been placed on the data bus (write cycle).
AB24	$\overline{\text{LWR}}$	I/O	<b>Write Enable (Intel Bus Mode)</b> This active low signal selects a write cycle.
	or $\overline{\text{LRDWR}}$	I/O	<b>Read Write Signal (Motorola Bus Mode)</b> This input signal distinguishes write from read operations.
AA23	$\overline{\text{LRDY}}$	I/O	<b>Ready (Intel bus mode)</b> This signal indicates that the current bus cycle is complete. The MUNICH256 asserts $\overline{\text{LRDY}}$ during a read cycle if valid output data has been placed on the data bus. In write direction $\overline{\text{LRDY}}$ will be asserted when input data has been latched.
	or $\overline{\text{DTACK}}$	I/O	In local bus master mode MUNICH256 evaluates $\overline{\text{LRDY}}$ to finish a transaction. <b>Data Transfer Acknowledge (Motorola bus mode)</b> This active low input indicates that a data transfer may be performed. During a read cycle data becomes valid at the falling edge of $\overline{\text{DTACK}}$ . The data is latched internally and the bus cycle is terminated. During a write cycle the falling edge of $\overline{\text{DTACK}}$ marks the latching of data and the bus cycle is terminated.

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AC26	$\overline{\text{LINT}}$	I/od	<p><b>Interrupt Request</b> This line indicates general interrupt requests of the mailbox. The interrupt sources can be masked via registers. In local bus master mode the MUNICH256 can monitor external interrupts indicated via <math>\overline{\text{LINT}}</math>.</p>
AC25, W23	$\overline{\text{LCS2}}$ , $\overline{\text{LCS1}}$	O	<p><b>Chip Select 2, 1</b> These signals select external peripherals when MUNICH256 is the local bus master. As long as the local bus master functionality is disabled these outputs are set to tri-state.</p>
AD13	$\overline{\text{LBHE}}$  or  LSIZE0	O   O	<p><b>Byte High Enable (Intel Bus Mode)</b> In local bus master mode this signal indicates a data transfer on the upper byte of the data bus LD(15:8). This signal has no function in slave mode. When local bus master functionality is disabled this output is tri-state.</p> <p><b>Byte Access (Motorola Bus Mode)</b> In local bus master mode this signal indicates byte transfers. This signal has no function when the MUNICH256 is local bus slave. When local bus master functionality is disabled this output is tri-state.</p>
AA25	LHOLD  or  $\overline{\text{LBR}}$	O   O	<p><b>Bus Request (Intel Bus Mode)</b> This pin indicates a requests to become local bus master. When local bus master functionality is disabled this output is tri-state.</p> <p><b>Bus Request (Motorola Bus Mode)</b> <math>\overline{\text{LBR}}</math> indicates a request to become local bus master. When local bus master functionality is disabled this output is set to tri-state.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AB25	LHLDA or $\overline{\text{LBG}}$	I  I	<b>Hold (Intel Bus Mode)</b> LHLDA indicates that the external processor has released control of the local bus. <b>Bus Grant (Motorola Bus Mode)</b> $\overline{\text{LBG}}$ indicates that the MUNICH256 may access the local bus.
V23	$\overline{\text{LBGACK}}$	O	<b>Bus Grant Acknowledge (Motorola Bus Mode)</b> $\overline{\text{LBGACK}}$ is driven low when the MUNICH256 has become bus master. When local bus master functionality is disabled this output is tri-state.

## 2.7 Serial Interface 16-port mode

Pin No.	Symbol	Input (I) Output (O)	Function
M24	TRD	O	<p><b>Test Receive Data</b> In serial test mode the incoming data stream of one selected port is directly feeded to this output. When test breakout functionality is disabled this output is tri-state.</p>
C15	TCLKO or TRCLK	O  O	<p><b>Transmit Clock Out</b> This signal provides a clock reference for transmit data of high speed port zero.</p> <p><b>Test Receive Clock</b> In serial test mode the receive clock of one selected port is directly feeded to this output.</p>
B5	TRSP	O	<p><b>Test Receive Synchronization Pulse</b> In serial test mode the receive synchronization of one selected port is directly feeded to this output. When test breakout functionality is disabled this output is tri-state.</p>
N26	TTCLK	O	<p><b>Test Transmit Clock</b> In serial test mode the clock provided via TTCLK replaces the transmit clock output of the selected transmit line. When test breakout functionality is disabled this output is tri-state.</p>
C12	TTD	I	<p><b>Test Transmit Data</b> In serial test mode the data stream provided via TTD replaces the transmit data stream of the selected transmit line.</p>
C5	TTSP	O	<p><b>Test Transmit Synchronization Pulse</b> In serial test mode the transmit synchronization pulse of one selected port is directly feeded to this output. When test breakout functionality is disabled this output is tri-state.</p>

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function												
R23, V25, U26, R24, T25, P24, T26, P25, P26, N25, N23, L26, B14, C14, D14, A16	TCLK(15:0)	I	<p><b>Transmit Clock</b></p> <p>This signal provides the data clock for TD.</p> <table> <tr> <td>T1/DS1</td> <td>24-channel</td> <td>1.544 MHz</td> </tr> <tr> <td>CEPT</td> <td>32-channel</td> <td>2.048 MHz</td> </tr> <tr> <td></td> <td>64-channel</td> <td>4.096 MHz</td> </tr> <tr> <td></td> <td>128-channel</td> <td>8.192 MHz</td> </tr> </table> <p>Unchannelized: Up to 45 MHz on port zero or 8.192 MHz on every other port.</p>	T1/DS1	24-channel	1.544 MHz	CEPT	32-channel	2.048 MHz		64-channel	4.096 MHz		128-channel	8.192 MHz
T1/DS1	24-channel	1.544 MHz													
CEPT	32-channel	2.048 MHz													
	64-channel	4.096 MHz													
	128-channel	8.192 MHz													
K26, M23, L25, H26, L23, D20, B22, A23, C20, D19, B21, C19, A21, C16, B16, D15	TD(15:0)	O	<p><b>Transmit Data</b></p> <p>Serial data sent by this output port is push-pull for active bits in the PCM frame and tri-state for inactive bits. Transmit data can be updated on the rising or falling edge of the transmit clock.</p>												
N1, M3, L2, L3, F1, F2, E2, F4, B6, D7, A7, C8, A8, C9, C10, D11	TSP(15:0)	I	<p><b>Transmit Synchronization Pulse</b></p> <p>This signal provides the reference for the transmit frame synchronization. A low to high transition marks the frame boundary in a PCM frame (for details refer to <a href="#">Chapter 5.4.1</a>). The transmit synchronization pulse is sampled on the rising or falling edge of the transmit clock.</p>												
A13, D13, D16, A19, B18, B19, C26, G23, G24, H24, F26, K24, T24, T23, W25, C11	RCLK(15:0)	I	<p><b>Receive Clock</b></p> <p>This signal provides the data clock for RD.</p> <table> <tr> <td>T1/DS1</td> <td>24-channel</td> <td>1.544 MHz</td> </tr> <tr> <td>CEPT</td> <td>32-channel</td> <td>2.048 MHz</td> </tr> <tr> <td></td> <td>64-channel</td> <td>4.096 MHz</td> </tr> <tr> <td></td> <td>128-channel</td> <td>8.192 MHz</td> </tr> </table> <p>Unchannelized: Up to 45 MHz on port zero or 8.192 MHz on every other port.</p>	T1/DS1	24-channel	1.544 MHz	CEPT	32-channel	2.048 MHz		64-channel	4.096 MHz		128-channel	8.192 MHz
T1/DS1	24-channel	1.544 MHz													
CEPT	32-channel	2.048 MHz													
	64-channel	4.096 MHz													
	128-channel	8.192 MHz													
B13, A14, A17, C17, A20, D18, E25, D26, F25, J23, H25, J25, U24, W26, AA26, B11	RD(15:0)	I	<p><b>Receive Data</b></p> <p>Serial data is received at this PCM input port. Receive data can be sampled on the rising or falling edge of the receive clock.</p>												



Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
N2, M4, L1, L4, H4, G3, G4, D1, D6, A6, C7, D8, B8, D9, B9, A10	RSP(15:0)	I	<b>Receive Synchronization Pulse</b> This signal provides the reference for the receive PCM frame synchronization. A low to high transition marks the frame boundary in a PCM frame (for details refer to <a href="#">Chapter 5.4.1</a> ). The receive synchronization pulse can be sampled on the rising or falling edge of the receive clock.
D5, A4, B4, C4, E3, D2, H3, H2, J4, H1, J2, K4, K3, K1, D12, A11	RES3..16 RES20..21		<b>Reserved Pins 3..16, 20..21</b>  <b>These pins are reserved in 16-pin mode.</b>  <b>A pull-up resistor to <math>V_{DD3}</math> is recommended.</b>

## 2.8 Serial Interface 28-port mode

Pin No.	Symbol	Input (I) Output (O)	Function
C15	TCLKO  or TRCLK	O  O	<b>Transmit Clock Out</b> This signal provides a clock reference for transmit data of high speed port zero. <b>Test Receive Clock</b> In serial test mode the receive clock of one selected port is directly feeded to this output.
M24	TRD	O	<b>Test Receive Data</b> In serial test mode the incoming data stream of one selected port is directly feeded to this output.
N26	TTCLK	O	<b>Test Transmit Clock</b> In serial test mode the incoming receive clock of one selected port is directly feeded to this output.

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
C12	TTD	I	<b>Test Transmit Data</b> In serial test mode the data stream provided via TTD replaces the transmit data stream of the selected transmit line.
K1, K3, K4, J2, H1, J4, H2, H3, D2, E3, C4, B4, A4, D5, C5, B5, B8, A8, D9, C9, B9, C10, A10, D11, B14, C14, D14, A16	TCLK(27:0)	I	<b>Transmit Clock</b> This signal provides the data clock for TD. In framed modes (T1/E1) a clock gap indicates the frame boundary. T1/DS1 24-channel 1.544 MHz CEPT 32-channel 2.048 MHz Unchannelized: Up to 45 MHz on port zero or up to 8.192 MHz on every other port.
R23, V25, U26, R24, T25, P24, T26, P25, P26, N25, N23, L26, K26, M23, L25, H26, L23, D20, B22, A23, C20, D19, B21, C19, A21, C16, B16, D15	TD(27:0)	O	<b>Transmit Data</b> Serial data sent by this output port is push-pull for active bits in the PCM frame and tri-state for inactive bits. Output is tri-state until port is enabled for transmission. Transmit data is updated on the rising or falling edge of the selected transmit clock.
N2, M4, L1, L4, H4, G3, G4, D1, D6, A6, C7, D8, A13, D13, D16, A19, B18, B19, C26, G23, G24, H24, F26, K24, T24, T23, W25, C11	RCLK(27:0)	I	<b>Receive Clock</b> This signal provides the data clock for RD. In framed modes (T1/E1) a clock gap indicates the frame boundary. T1/DS1 24-channel 1.544 MHz CEPT 32-channel 2.048 MHz Unchannelized: Up to 45 MHz on port zero or 8.192 MHz on every other port.

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
N1, M3, L2, L3, F1, F2, E2, F4, B6, D7, A7, C8, B13, A14, A17, C17, A20, D18, E25, D26, F25, J23, H25, J25, U24, W26, AA26, B11	RD(27:0)	I	<b>Receive Data</b> Serial data is received at this PCM input port. Receive data can be sampled on the rising or falling edge of the receive clock.
D12, A11	RES20..21		<b>Reserved Pins 20, 21</b>  These pins are reserved in 28-port mode.  A pull-up resistor to $V_{DD3}$ is recommended.

## 2.9 Test Interface

Pin No.	Symbol	Input (I) Output (O)	Function
C25	TCK	I	<b>JTAG Test Clock</b> This pin is connected with an internal pull-up resistor.
F23	TMS	I	<b>JTAG Test Mode Select</b> This pin is connected with an internal pull-up resistor.
A24	TDI	I	<b>JTAG Test Data Input</b> This pin is connected with an internal pull-up resistor.
D24	TDO	O	<b>JTAG Test Data Output</b>

Pin No.	Symbol	Input (I) Output (O)	Function
B26	$\overline{\text{TRST}}$	I	<b>JTAG Test Reset</b> This pin is connected with an internal pull-down resistor.
E24	SCAN	I	<b>Full Scan Path Test</b> When connected to $V_{DD3}$ the MUNICH256 works in a vendor specific test mode. It is recommended to connect this pin to $V_{SS}$ .

## 2.10 Power Supply and No-connect Pins

Pin No.	Symbol	Input (I) Output (O)	Function
AF1, AE7, AF9, AE12, AE15, AF18, AE20, AF26, AD3, AD24, AD26, Y2, Y25, V1, V26, R2, T12, T11, R12, R11, T14, T13, R14, R13, T16, T15, R16, R15, R25, P12, P11, N12, N11, P14, P13, N14, N13, P16, P15, N16, N15, M2, M12, M11, L12, L11, M14, M13, L14, L13, M16, M15, L16, L15, M25, J1, J26, G2, G25, C3, C24, D25, A1, B7, A9, B12, B15, A18, B20, A26	$V_{SS}$	I	<b>Ground 0V</b> All pins must have the same level.
AE2, AF5, AE10, AF12, AF15, AE17, AF22, AE25, AB1, AB26, Y1, Y26, U2, U25, R1, R26, M1, M26, K2, K25, G1, G26, E1, E26, B2, A5, B10, A12, A15, B17, A22, B25	$V_{DD25}$	I	<b>Supply Voltage 2.5V <math>\pm</math> 0.25V</b> All pins must have the same level.

Pin Description

Pin No.	Symbol	Input (I) Output (O)	Function
AC4, AD6, AD9, AC10, AD14, AD18, AC17, AD21, AC23, AA3, AA24, W3, U4, V24, U23, P3, P23, N24, L24, J3, K23, J24, H23, F3, F24, D4, C6, D10, C13, D17, C18, C21, D23	$V_{DD3}$	I	<b>Supply Voltage</b> 3.3V $\pm$ 0.3V All pins must have the same level.
E4, C1, B1, C2, A3, A2, B3, D3, B23, D21, C22, A25, E23, B24, C23, D22, AC22, AD23, AD22, AC21, AE22, AC20, AF24, AE23, AF2, AE3, AC5, AD4, AE1, AD2, AB4, AC3	NC0..31		<b>No-connect Pins 0..31</b>  <b>It is recommended not to connect these pins.</b>

## 3 General Overview

### 3.1 Functional Overview

#### MUNICH256

The MUNICH256 is a highly integrated WAN protocol controller that performs HDLC, PPP and transparent (TMA) protocol processing on 256 full duplex serial channels and a configurable port mode with 16 or 28 links.

Dependent on the port mode a link can be operated in T1/E1, in channelized 4.096 MHz/ 8.192 MHz mode (16-port mode only) or in unchannelized mode. The internal framing function is switched off in this mode.

In 16-port mode the system interface consists of one receive clock input, one receive synchronization pulse input and one receive data input for each receive line. In transmit direction each link consists of one transmit clock input, one transmit synchronization input and one transmit data output. Synchronization pulses are not supported in unchannelized mode.

In 28-port mode the system interface consists of a receive clock input and a receive data input. In transmit direction a transmit clock input and a transmit data output is provided. Frame boundaries are indicated by clock gaps.

The device provides a maximum aggregate data rate of 90 Mbit/s per direction, assuming a PCI frequency of 66 MHz (45 Mbit/s at 33 MHz). The following clock rates are supported where the sum of all clock rates does not exceed the above throughput limitation:

In 16-port mode:

- T1 mode with 1.544 MHz on any port.
- E1 mode with 2.048 MHz on any port.
- Channelized mode with 4.096 MHz on any port.
- Channelized mode with 8.192 MHz on any port.
- Unchannelized mode with up to 45 MHz on port zero.
- Unchannelized mode with up to 8.192 MHz on all other ports.

In 28-port mode:

- T1 mode (1.544 MHz) with gapped clock on any port.
- E1 mode (2.048 MHz) with gapped clock on any port.
- Unchannelized mode with up to 45 MHz on port zero.
- Unchannelized mode with up to 8.192 MHz on all other ports.

A variety of loop modes is provided to support remote as well as inloop testing of the device.

Two bus interfaces, a PCI Rev. 2.1 compliant bus interface and a 16 bit Intel/Motorola style bus interface, connect the device to system environment. Device configuration and

channel operation is provided through the PCI bus interface. The local bus interface provides access to the internal mailbox. The MUNICH256 supports PCI PnP capability by loading the subsystem ID and the subsystem vendor ID via a SPI interface into the PCI configuration space.

### 3.2 Block Diagram

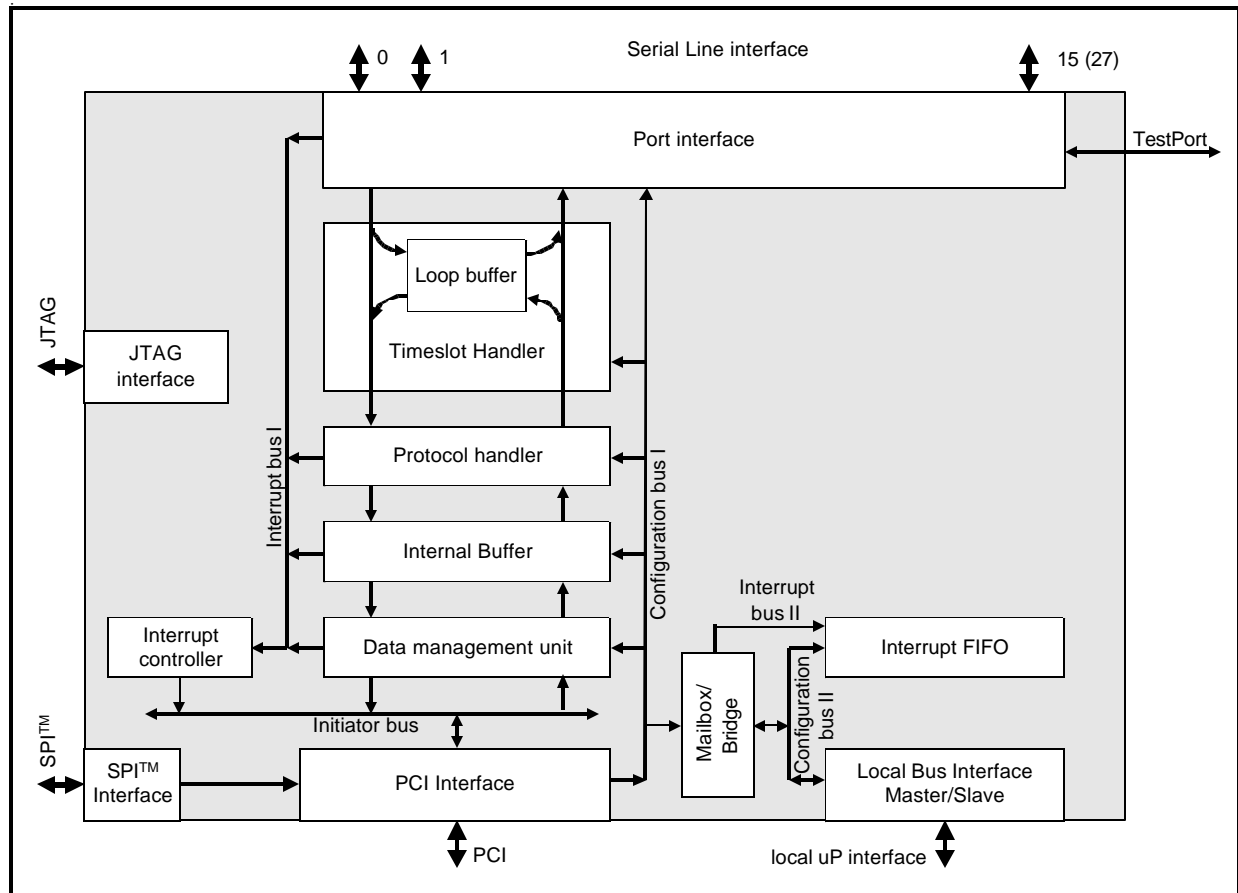


Figure 3-1 MUNICH256 Block Diagram

### 3.3 Internal Interface

The device consists of several macro functions as shown in **Figure 3-1**. The internal modules are connected by busses/signals according to Infineons on-chip bus.

The main busses are:

- The initiator bus, on which the DMA requests of the data management units and the interrupt controller are arbitrated and funneled into the PCI interface.
- The configuration busses, which serve as the standard programming interface to access the chip internal registers and functions either via PCI bus or via the local bus interface.

- The interrupt busses, which collect all interrupt information and forward them to the corresponding interrupt handler.

The chip's core functions are all operated with the PCI clock. Transfers between clocking regions (serial clocks and system clock) are implemented only in the serial interface.

### 3.4 Block Description

The following section gives a brief overview to the function of each block. For a detailed description of each function refer to [“Functional Description” on Page 51](#).

#### Serial port interface

The Serial Port Interface consists of the subfunctions receive and transmit. This block provides the function of serial/parallel and parallel/serial conversion for up to 16 (or 28 when configured for 28 port mode) incoming and outgoing serial data streams. Serial data is then transferred between the internal clocking system, which is derived from the PCI clock, and the various line clocks. This provides a unique clocking scheme on the internal interfaces. The aggregate bandwidth of all enabled ports can be up to 90 MBit/s in each direction with a PCI clock frequency of 66 MHz.

#### Time slot assigner

The time slot assigner exchanges data with the serial interface on a 8 bit parallel bus, thus funneling all data of up to 28 interfaces. The time slot assigner provides freely programmable mapping of any time slot or any combination of time slots to 256 logical channels. A programmable mask can be provided to allow subchanneling of the available time slots which allows channel data rates starting at 8kbit/s.

At the protocol machine interface the time slot assigner and the protocol machine exchanges channel oriented data (8 bit) together with the time slots masks.

#### Protocol handler

Two protocol machines, one for receive direction and one for transmit direction, provide protocol handling for up to 256 logical channels and a maximum serial aggregate data rate of up to 90 Mbit/s per direction. The protocol machines implement four modes, which can be programmed independently for each logical channel: HDLC, bit-synchronous PPP, octet-synchronous PPP and Transparent Mode A, including frame synchronous TMA.

#### Internal buffer

The internal buffers provides channelwise buffering of raw (unformatted/deformatted) data for 256 logical channels. Channel specific thresholds can be programmed



independently in transmit and receive direction. In order to avoid transmit underrun conditions each transmit channel has two control parameters for smoothing the filling/emptying process (transmit forward threshold, transmit refill threshold). In receive direction each channel has a receive burst threshold. To avoid unnecessary waste of bus bandwidth, e.g. in case of transmission errors, the receive buffer provides the capability to discard frames which are smaller than a programmable threshold.

### **Data management units**

The data management units provide direct data transfer between the system memory and the internal buffers. Each channel has an associated linked list of descriptors, which is located in system memory and handled by the data management units. This linked list is the interface between the system processor and the MUNICH256 for exchange of data packets. The descriptors and the data packets can be stored arbitrarily in 32 bit address space of system memory, thus allowing full scatter/gather assembly of packets. In order to optimize PCI bus utilization, each descriptor is read in one burst and held on-chip afterwards.

### **Interrupt controller**

Two interrupt controllers manage internal interrupts. Interrupts from the mailbox are passed in the form of interrupt vectors to an internal interrupt FIFO which can be read from the local bus. All system, port and channel related interrupt information is passed to the main interrupt controller which is connected to the PCI system. A programmable DMA with nine channels stores these interrupts in the form of interrupt vectors in different interrupt queues in system memory.

### **PCI interface**

The PCI interface unit combines all DMA requests from the internal data management unit and the interrupt controller and translates them into PCI Rev. 2.1 compliant bus accesses. The PCI interface optionally includes the function of loading the subsystem vendor ID and the subsystem ID from an external SPI compliant EEPROM.

### **Mailbox, internal bridge and global registers**

The mailbox is used to exchange data between the PCI attached microprocessor and the local bus microprocessor and provides a doorbell function between the two interfaces.

Controlled by an arbiter an internal bridge connects the configuration bus I and the configuration bus II. It is NOT possible to access the configuration bus I and therefore the 'HDLC' registers or the PCI bridge from the local bus.

### **Local bus interface**

The local bus interface provides access between the local microprocessor and the on-chip configuration bus II, in order to access the mailbox. The local bus interface provides a switchable Intel-style or Motorola-style processor interface.

### **JTAG**

Boundary Scan logic according to IEEE 1149.1.

## 4 Functional Description

### 4.1 Port Handler

The port handler is the interface between the serial ports and the chip internal protocol functions. It converts incoming serial data into parallel data for further internal processing and in the outgoing direction it converts parallel data into a serial bit stream.

#### 4.1.1 Selectable port configuration

The serial interface of the device can be configured in a 16-port mode and additionally in a 28-port mode. The 16-port mode provides a clock pin, a data pin and a frame synchronization pin for each port and direction. The 28-port mode provides a clock pin and a data pin per port and direction. In this mode frame boundaries are indicated by clock gaps. **Table 4-1** and **Figure 4-1** respectively show the pin configuration and the supported frame structures in the 16-port mode and the 28-port mode.

**Table 4-1 Interface configuration**

	Port mode	
	16-port mode	28-port mode
<b>Supported Interfaces</b>		
1.544 MBit/s channelized	x	x
2.048 MBit/s channelized	x	x
4.096 MBit/s channelized	x	
8.192 MBit/s channelized	x	
Unchannelized	x	x
<b>Supported Pins</b>		
Receive Data	x	x
Receive Clock	x	x
Receive Synchronization Pulse	x	
Transmit Clock	x	x
Transmit Data	x	x
Transmit Synchronization Pulse	x	
<b>Frame Indication</b>		

Functional Description

Gapped Clock		x
Synchronization Pulse	x	

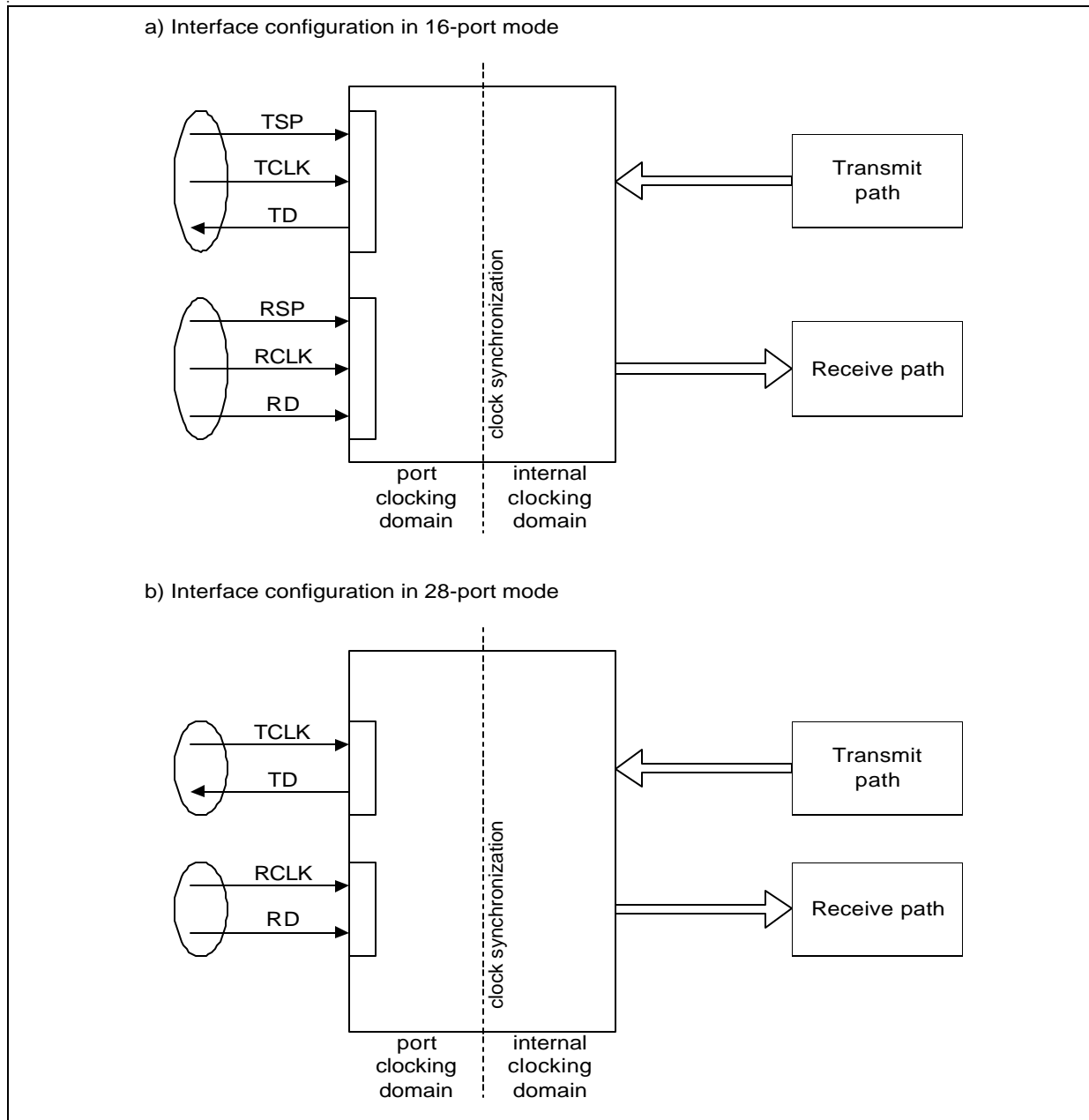


Figure 4-1 Port Configuration

### 4.1.2 External Timing Mode

Each transmit port is clocked using the external timing reference TCLK(x). Since all ports have their individual transmit clock each port can be operated independent of each other.

The same functionality as given for the transmit direction applies in receive direction.

### 4.1.3 Local Port Loop

A local port loop can be closed in the port interface. It mirrors the outgoing bit stream of one port to the receive part of the same port. This allows to prepare data in system memory, which is processed by the MUNICH256 in transmit direction, mirrored to the receiver and stored in system memory again. In order to ensure that the local port loop works even without an incoming receive clock, the receiver of the selected port is operated with the transmit clock.

When closing the local port loop, the corresponding transmit clock  $TCLK(x)$  and transmit frame synchronization pulse  $TSP(x)$  are used to operate the transmitter and the receiver. Receive data  $RD(x)$ , the receive clock  $RCLK(x)$  and the receive synchronization pulse  $RSP(x)$  are ignored in that case.

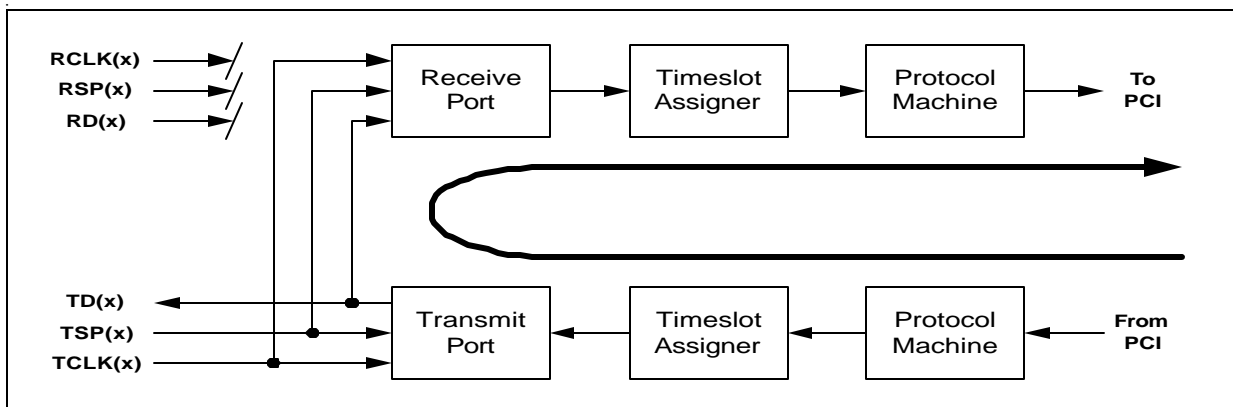


Figure 4-2 Local Port Loop

#### 4.1.4 Remote Payload Loop

The MUNICH256 supports a remote payload loop for each of the 16 (28) lines), where the incoming serial data stream of a selected port is mirrored to the outgoing serial data stream of the same port. The F-bit (T1 mode) is not looped.

In receive direction the payload is stored in an internal loop buffer and in transmit direction this payload data is taken from the loop buffer and inserted in the outgoing bit stream. This internal loop buffer compensates for receive clock and transmit clock jitter. Nevertheless, the average clock rate of the receive port and the transmit port must be the same. After first activation the payload of one frame is written to the loop FIFO. Then the time slot assigner starts reading data bytes out of the loop buffer and inserts them into the transmit data path. Due to a receive/transmit clock jitter the read pointer may move towards the write pointer. In case the distance between write pointer and read pointer is equal plus/minus 1 byte a slip of the read pointer will occur.

For ports configured in 8.192 MBit/s mode the receive clock and the transmit clock must be synchronous.

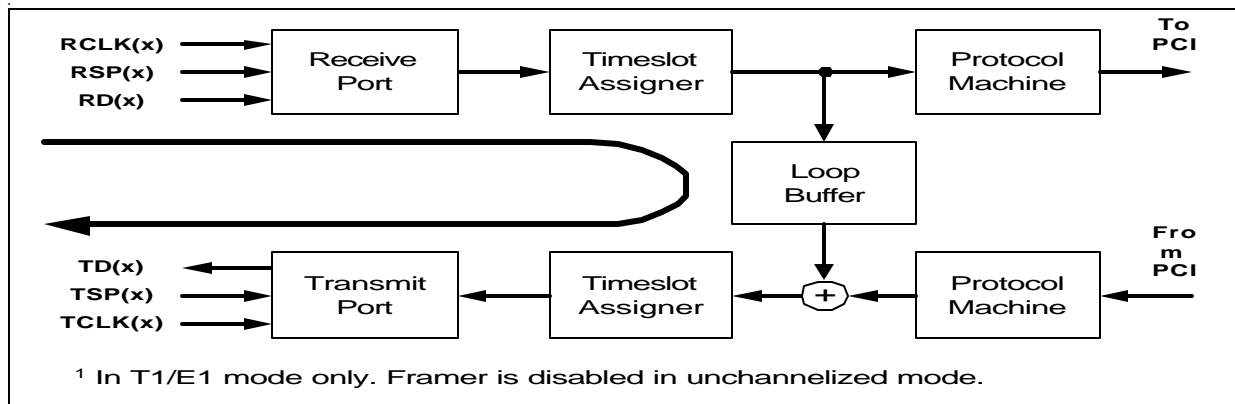


Figure 4-3 Remote Payload Loop

### 4.1.5 Remote Channel Loopback

A remote channel loop can be switched for one logical channel at a time. Incoming serial data located in the receive payload of one port is mirrored to the corresponding transmit channel (same channel number). An internal jitter attenuator compensates jitter between receive clock and transmit clock. Nevertheless, the average clock rate of the receive port and the transmit port must be the same. After first activation of the loop 32 receive bytes are written to the loop FIFO. Then the time slot assigner starts reading data bytes out of the loop buffer and inserts them into the transmit data path. Hence, the initial distance between the FIFO read pointer and the FIFO write pointer is 32 bytes. Due to a receive/transmit clock jitter the read pointer may move towards the write pointer. In case the distance between write pointer and read pointer is equal plus/minus 1 byte a slip of the read pointer will occur.

In channelized and unchannelized mode the transmit and receive masks of all time slots belonging to the looped channel must be the same. The aggregate bit rate of the transmit section and the aggregate bit rate of the receive section has to be identical.

In receive direction incoming data of the selected channel is stored in the loop buffer. In transmit direction data is clocked out of the loop buffer and transmitted in the time slots of the selected channel. Received data is processed normally.

The remote channel loop can also be used to loop the complete payload except the 'F'-bit (T1). Therefore one logical channel must be setup which includes all payload time slots.

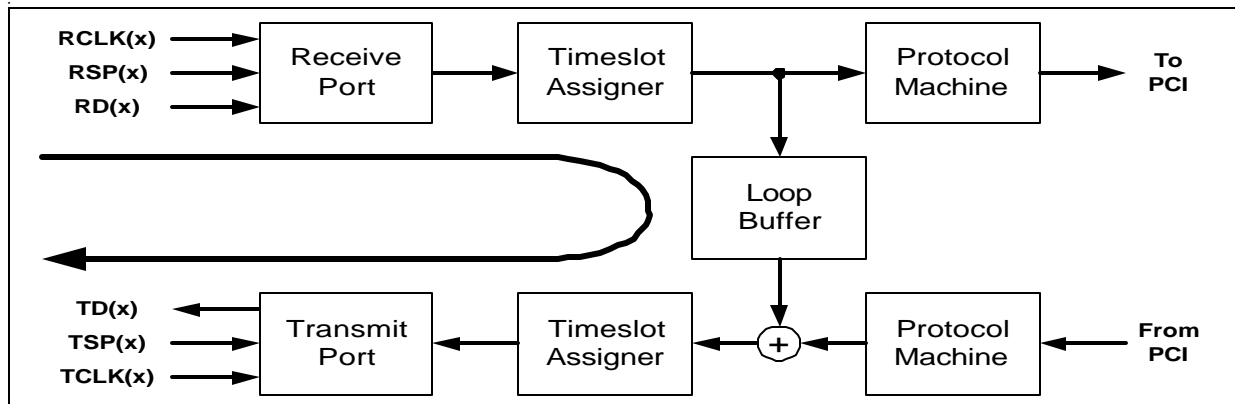


Figure 4-4 Remote Channel Loop

### 4.1.6 Test Breakout

The test breakout function provides the capability to multiplex one of the incoming 16 (28) receive links to the outgoing test receive port, that is the incoming receive clock signal RCLK(x) is mapped to the test receive clock output TRCLK, the receive synchronization pulse RSP(x) is mapped to TRSP (16-port mode only) and the incoming receive data signal RD(x) is mapped to the test data output TRD. In the opposite direction one of the 16 (28) transmit data output signals TD(x) can be replaced with the test transmit data signal TTD. Furthermore the corresponding transmit synchronization pulse input TSP(x) (16-port mode only) and the transmit clock input signal TCLK(x) can be monitored on the test port outputs TTSP and TTCLK. There is no processing function in the data path. Each output signal is a buffered version of the corresponding input signal, e.g. output signal TD(x) is a buffered version of the incoming signal TTD.

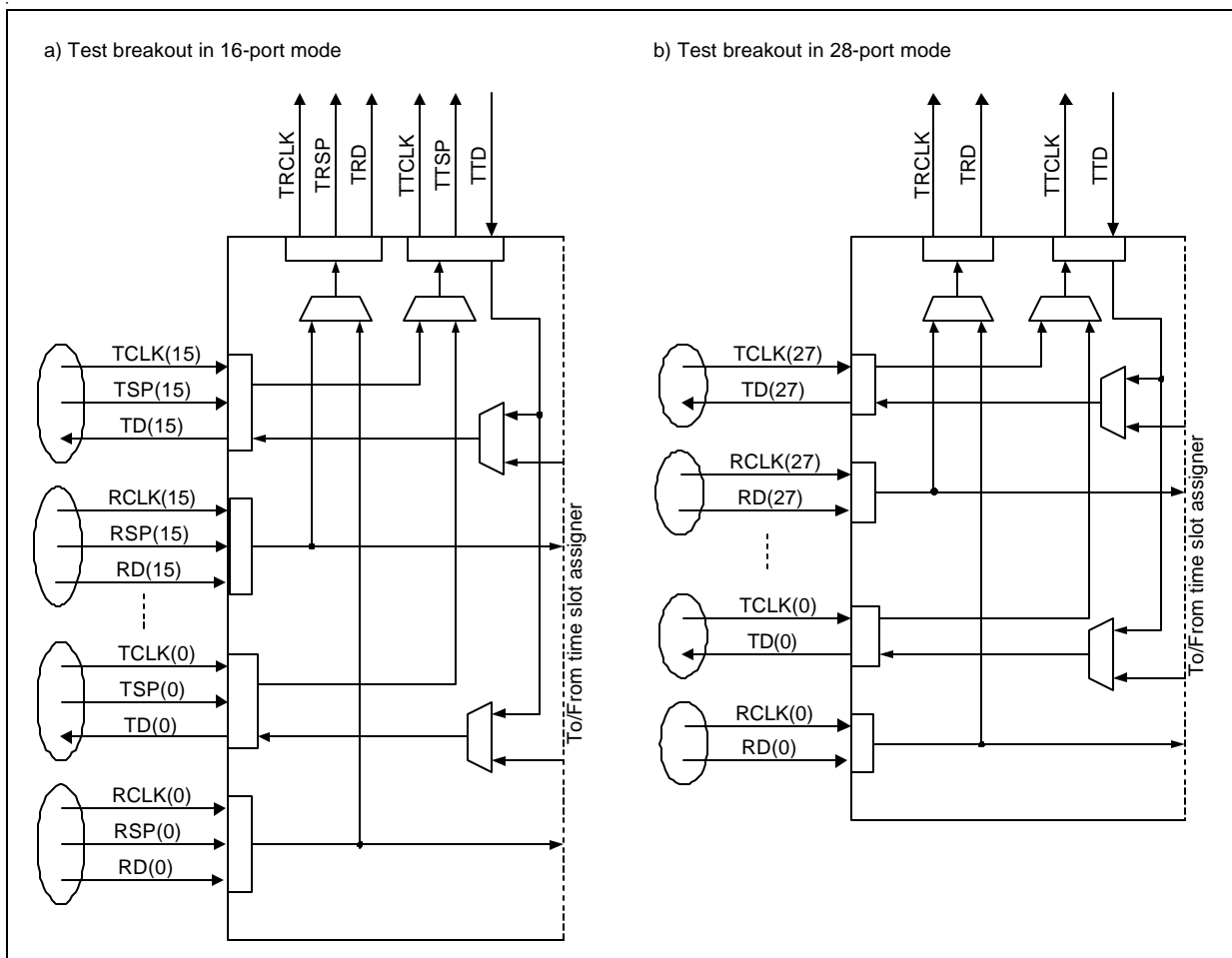


Figure 4-5 Test Breakout



## 4.2 Time slot Handler

### 4.2.1 Channelized Modes

The time slot handler assigns any combination of time slots of ports configured in T1 or E1, 4.096 MHz or 8.192 MHz mode to logical channels. The assigned time slots are connected internally and the bit stream of one logical channel is mapped continuously over the selected time slots. Since the receiver and the transmitter operate independently of each other, the assignment of time slots to logical channels can be done separately in receive and transmit direction. Any time slot can be assigned to any channel and any sequence of time slots can be assigned to one channel.

In normal operation each time slot consists of eight bits and all bits are used for data transmission. An available mask function provides the capability to mask selected bits, which in turn are disabled for data transmission. This provides the possibility to operate time slots with less than 64 kBit/s throughput. So, instead of mapping the bit stream of one logical channel over all bits of the assigned time slots, the bit stream is mapped continuously over all unmasked bits of the time slots belonging to that channel.

Masked bits are tri-stated in transmit direction. In receive direction masked data bits are discarded.

**Figure4-6** shows a simple assignment process. In this case one port is configured in E1 mode and time slots two and three are assigned to logical channel 5. The bit mask of time slot two is set to  $FE_H$ , which disables bit zero of that time slot, and the bit mask of the third time slot is set to  $FD_H$ , which disables bit one.

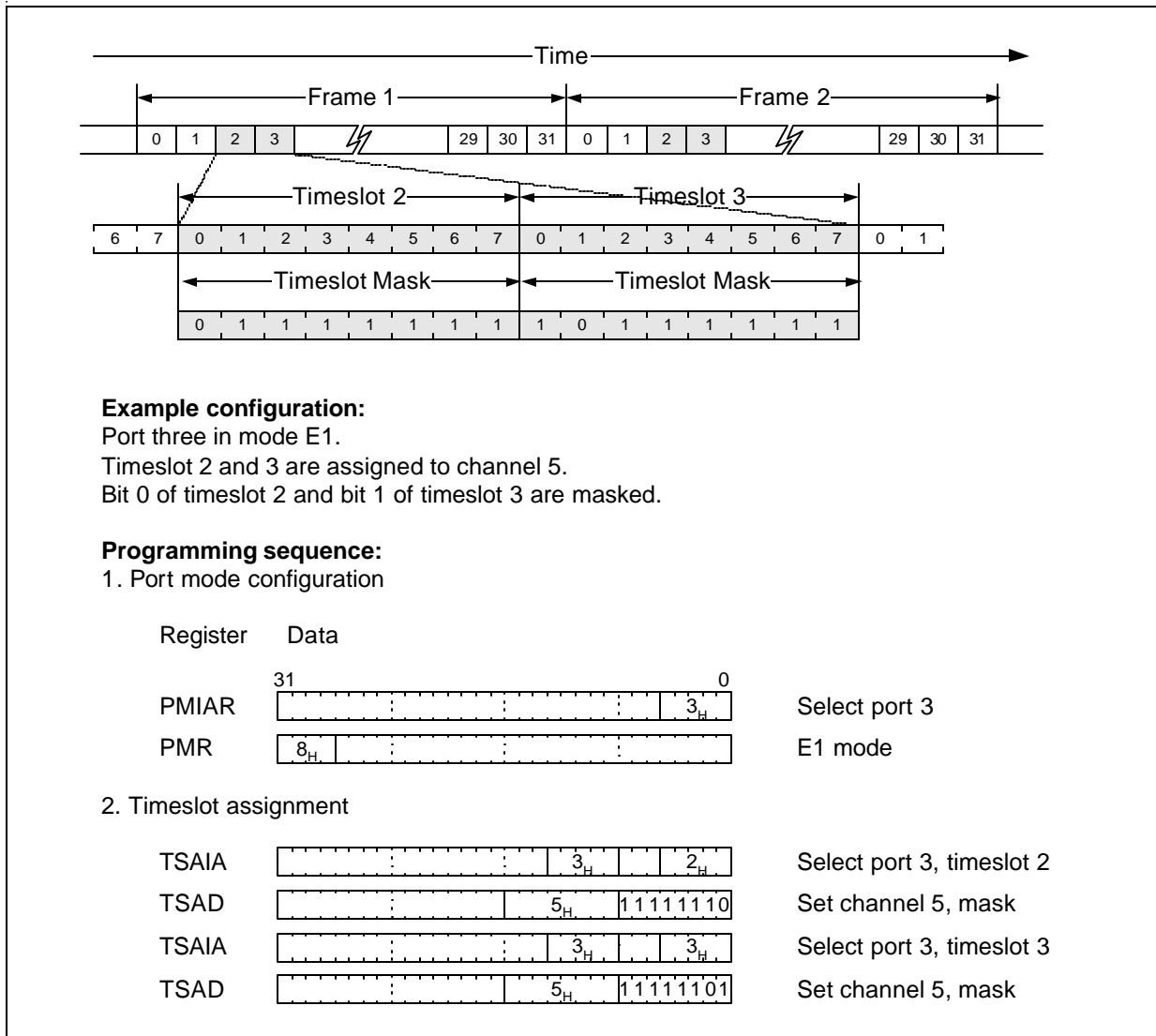


Figure 4-6 Time slot Assignment in Channelized Modes

### 4.2.2 Unchannelized Mode

In unchannelized mode the complete incoming and outgoing serial bit stream belongs to one logical channel. Although unchannelized mode does not have a time slot scheme, this mode is handled internally the same way as a port configured in T1 mode. Therefore the time slot assignment must be done as in T1 mode, with exception that 'time slots' zero to 23 must be assigned to one channel and that no time slot may be set to inhibit. The function of bit masks, which is available in T1, E1, 4.096 MHz or 8.192 MHz mode, is not available in unchannelized mode. Each bit of the time slots 0-23 must be enabled, setting the masks to FF<sub>H</sub>. Furthermore unchannelized mode does not provide the function of synchronizing bits to an external frame synchronization pulse.

### 4.3 Data Management Unit

Each packet or part of a packet is referenced by a descriptor. The descriptors form a link list, thus connecting all packets together. Packet data as well as descriptors are located in system memory. Both the MUNICH256 and the system CPU operate on these data structures.

Each logical channel has its dedicated linked list of descriptors, one for receive direction and one for transmit direction. This type of data structure allows channel specific memory organization which can be specified by the system processor. It provides an optimized way to transfer data packets between the system processor and the MUNICH256.

The MUNICH256 has a flexible DMA controller to transfer data either from the internal receive buffer to the shared memory (receive direction) or from the shared memory to the internal transmit buffer (transmit direction). Each DMA works on one linked list. Each linked list located in system memory is associated with one of the 256 transmit channels or one of 256 receive channels.

The address generator of the DMA controller supports full link list handling. Descriptors are stored independently from the data buffers, thus allowing full scatter/gather assembly and disassembly of data packets.

#### 4.3.1 Descriptor Concept

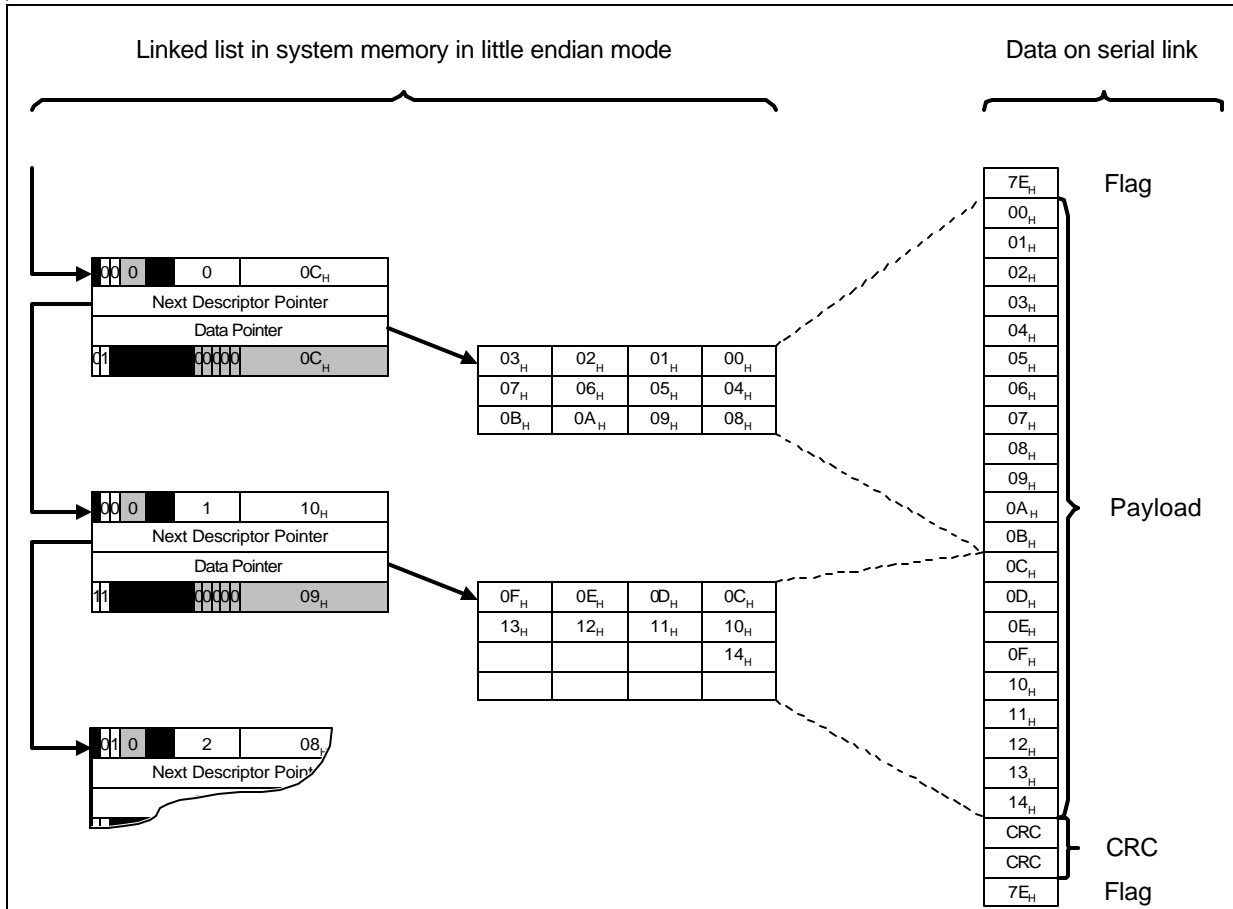
A descriptor is used to build a linked list, where each member of the linked list points to a data section. A descriptor consists of four DWORDS<sup>1)</sup>. The first three DWORDS, containing link and packet information, are provided by the system CPU and the last DWORD contains status information, which is written when the MUNICH256 has finished operation on a descriptor.

The data section itself can be of any size up to the maximum size of 65535 bytes per descriptor and is defined in the first DWORD of a descriptor. Each logical data packet can be split into one or multiple parts, where each part is referenced by one descriptor, and all parts are referenced by a linked list of descriptors. The descriptor containing the last part of a data packet is marked with a frame end bit. The descriptor following the marked descriptor therefore contains the beginning of the next data packet (**Figure 4-7**). The last descriptor in a linked list is marked with a hold indication.

For ease of programming the transmit descriptor and the receive descriptor are structured the same way, thus allowing to link a receive descriptor directly into the linked list of the transmit queues with minimum descriptor processing.

---

1)



**Figure 4-7 Descriptor Structure**

Although the data management unit works 32-bit oriented, it is possible to begin a transmit data section at an uneven address. The two least significant bits of the transmit data pointer determine the beginning of the data section and the number of bytes in the first DWORD of the data section, respectively. In receive direction the address of the data sections must be DWORD aligned.

### 4.3.2 Receive Descriptor

Each receive descriptor is initialized by the host CPU and stored in system memory as part of a linked list. The MUNICH256 reads a descriptor, when requested to do so from the host by a receive command or after branching from one receive descriptor to the next receive descriptor. Each receive descriptor contains four DWORDs, where the first three DWORDs contain link and packet information and the last DWORD contains status information. Once the descriptor is processed the status information will be written back to system memory by the MUNICH256 (Receive status update). When the MUNICH256

Functional Description

branches to a new descriptor it reads the link and packet information entirely and stores it in its on-chip channel database.

**Table 4-2 Receive Descriptor Structure**

DWORD ADDR.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
00 <sub>H</sub>	0	HOLD	RHI	OFFSET(2:0)			0	0	0	0	DescriptorID(5:0)					
04 <sub>H</sub>	NextReceiveDescriptorPointer(31:2)															
08 <sub>H</sub>	ReceiveDataPointer(31:2)															
0C <sub>H</sub>	FE	C	0	0	0	0	0	0	0	0	0	MFL	RFOD	CRC	ILEN	RAB

DWORD ADDR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00 <sub>H</sub>	NO(15:0)															
04 <sub>H</sub>	NextReceiveDescriptorPointer(31:2)														0	0
08 <sub>H</sub>	ReceiveDataPointer(31:2)														0	0
0C <sub>H</sub>	BNO(15:0)															

**HOLD**

Hold indication

HOLD indicates that a descriptor is the last element of a linked list containing valid information.

0 Next descriptor is available in the shared memory. After checking the HOLD bit the data management unit branches to the next receive descriptor.

1 This descriptor is the last one that is available for a channel. This means that the data section where this descriptor points to is the last data section which is available for data storage. After processing of descriptor has finished, the data management unit repolls the descriptor one time to check if HOLD has already been cleared. If HOLD is still set the corresponding receive channel is deactivated as long as the system CPU does not request a new activation via a 'Receive Hold Reset' command or forces the MUNICH256 to branch to a new linked list via a 'Receive Abort/Branch' command.

*Note: When repolling a descriptor the MUNICH256 checks the HOLD bit and the bit field NextReceiveDescriptorPointer. All other information are NOT updated in the internal channel database.*

---

**Functional Description**

RHI	<p>Receive Host Initiated Interrupt</p> <p>This bit indicates that the MUNICH256 shall generate a 'Receive Host Initiated' interrupt vector after it has finished processing the descriptor.</p> <p>0 Data management unit does not generate an interrupt vector after it has processed the receive descriptor.</p> <p>1 Data management unit generates an interrupt vector, as soon as all data bytes are transferred into the current data section and the status information is updated.</p>
OFFSET	<p>Offset of unused data section.</p> <p>This bit field allows to reserve memory space in increments of DWORDs for an additional header. If the marked descriptor is the first one of a new packet the data management unit will write data at the address <code>ReceiveDataPointer+4xOFFSET</code>.</p> <p><i>Note: Offset x 4 must be smaller than NO.</i></p> <p><i>Note: This option is not available in transparent mode.</i></p>
DescriptorID	<p>This bit field is read by the data management unit and written back in the corresponding interrupt status of a channel interrupt vector which is generated by the data management unit. This value provides a link between the descriptor and the corresponding interrupt vector.</p>
NO	<p>Byte Number</p> <p>This bit field defines the size of the receive data section allocated by the host. The maximum buffer length is 65535 bytes and it has to be a multiple of 4 bytes. Data bytes are stored in the receive data section according to the selected mode (little endian or big endian).</p> <p><i>Note: Please note that the device handles the status (CRC, flag and frame status) of frame based protocols (HDLC, PPP) internally in the same way as payload data. Therefore byte number should include four bytes more than the maximum length of incoming frames. Nevertheless, the frame status will be deleted from the end of the data stream and be attached as a status word to the receive descriptor. The frame status will not be written to the data section.</i></p>

---

**Functional Description**

**NextReceiveDescriptorPointer**

This pointer contains the start address of the next valid receive descriptor. After completion of the current receive descriptor the data management unit branches to the next receive descriptor to continue data reception.

System CPU can force the MUNICH256 to branch to the beginning of a new linked list via the command 'Receive Abort/Branch'. In this case the receive descriptor address provided via register CSPEC\_FRDA is used as the next receive descriptor pointer to be branched to.

**ReceiveDataPointer**

This pointer contains the start address of the receive data section. The start address must be DWORD aligned.

**FE**

**Frame End**

It indicates that the current receive data section (addressed by ReceiveDataPointer) contains the end of a frame. This bit is set by the data management unit after transferring the last data of a frame from the internal receive buffer into the receive data section which is located in the shared memory. Moreover the bit field BNO and the status bits are updated, the complete (C) bit is set and a 'Frame End' interrupt vector is generated.

**C**

**Complete**

This bit indicates that

- filling the data section has completed (with or without errors),
- processing of this descriptor was aborted by a 'Receive Abort/Branch' command,
- or the end of frame (PPP, HDLC) was stored in the receive data section.

The complete bit releases the descriptor.

**BNO**

**Byte Number of Received Data**

The data management unit writes the number of data bytes stored in the current data section into bit field BNO.

---

**Functional Description**

When the MUNICH256 completes a data section, which included the end of a frame (C bit and FE bit are set), or when the MUNICH256 branches to a new linked list due to a 'Receive Abort/Branch' command the status information bits RAB, ILEN, CRC, RFOD and MFL are updated as part of the receive status update. In the abort scenario, the C bit will always be set. Bit FE will be set only, if the particular channel operates in HDLC or PPP mode.

RAB	Receive Abort
	This bit is set when
	<ul style="list-style-type: none"><li>•the incoming serial data stream contained an abort sequence, or</li><li>•an incoming frame was aborted by the command 'Receive Abort/Branch', or</li><li>•when a channel is switched off while a frame is being received.</li></ul>
ILEN	Illegal length
	This bit is set, when the length of the incoming data packet was not a multiple of eight bits.
CRC	CRC Error
	This bit is set, when the checksum of an incoming data packet was different to the internally calculated checksum.
RFOD	Receive Frame Overflow
	This bit is set, when a receive buffer overflow occurred during data reception.
MFL	Maximum Frame Length
	This bit is set, when the length of the incoming data packet exceeded the value programmed in CONF1.MFL.

### 4.3.3 Data Management Unit Receive

The *data management unit receive* transfers data for each of the 256 logical receive channels from the internal receive buffer to the data sections of the corresponding channel. To fulfill the task it has to be initialized for operation, which is described in **“Channel Programming / Reprogramming Concept” on Page115**. Relevant part of the channel information for the data management unit is the address pointer to the first receive descriptor, the channel interrupt queue and the channel interrupt mask.

The first receive descriptor of a channel is fetched from system memory and stored in the chip internal channel database the first time the receive buffer requests a data transfer for the channel. The descriptor contains a pointer to the data section, the size of the provided data section and a pointer to the next receive descriptor.

The data transfer is requested as soon as a programmed receive buffer threshold is reached. This threshold is programmed during channel setup on a per channel basis. Task of the data management unit is to calculate the maximum number of bytes that can



---

**Functional Description**

be stored in the receive data section and to compare this with the length of the requested data transfer.

In case that the requested transfer length from the receive buffer fits into the provided data section the data management unit transfers the data block to system memory in one single burst. If the requested transfer length exceeds the available space of the data section the transfer is divided into two or more parts. Data packets are written to the data section until the given data section is filled or the end of a packet is reached.

If the data section in the shared memory is completely filled with data, the data management unit updates the status word of the receive descriptor by setting the complete (C) bit and the number of bytes (BNO), which are stored in the data section. In this case the number of bytes written to the data section equals the size of the data section.

If the data packet, which is written to system memory, contains the remaining part of a completely received packet, the data management unit updates the status word of the receive descriptor by setting the complete bit together with the frame end (FE) bit. The BNO field is updated on the actual value of bytes written to the data section. If enabled, the data management unit generates a 'Frame End' channel interrupt vector.

With the next receive buffer request the data management unit branches to the next receive descriptor, which was referenced in the next descriptor field of the current processed descriptor. To keep track of the linked list the data management unit provides the possibility to issue a 'Receive Host Initiated' interrupt vector, which is generated after the status word was updated. To enable this interrupt vector the bit RHI must be set in a descriptor.

**Descriptor hold operation**

Processing of the descriptor list is controlled by the HOLD bit, which is located in the first DWORD of each receive descriptor. The HOLD bit indicates that the marked descriptor is the last descriptor containing a valid data buffer. The data management unit will not branch to a next descriptor until the hold condition is removed or a 'Receive Abort' command forces the MUNICH256 to branch to the beginning of a new linked list. Since the HOLD bit marks the last descriptor in a linked list, it may prevent that further received data packets can be written to system memory.

When a given data section is filled, and does not contain the end of a frame (frame based protocols) and the requested transfer length could not be satisfied, the data management unit polls the HOLD bit of the current receive descriptor once more. If the HOLD bit is removed, it branches to the next descriptor. When the HOLD bit is still '1', an internal poll bit is set and the data management unit does not branch to the next descriptor. Additionally a 'Hold Caused Receive Abort' interrupt vector is generated. The status of the descriptor in the shared memory is aborted (RAB bit set) and the complete bit and the frame end bit are set in the receive descriptor. The rest of the frame will be discarded. As long as the HOLD bit remains set further data of the same channel is

---

**Functional Description**

discarded and for each discarded frame a 'Silent Discard' interrupt vector with the bits HRAB and RAB set is generated.

If the current data section was filled and does contain the end of frame a 'Frame End' interrupt vector is generated and the descriptor is updated on the FE bit and the C bit. Therefore the status of this receive descriptor is error free. With the next request of the receive buffer, the data management unit repolls the HOLD bit of the current receive descriptor. If the hold bit is removed, it branches to the next descriptor. If the HOLD bit is still '1', an internal poll bit is set. As long as the HOLD bit remains set, further data of the same channel is discarded and for each discarded frame a 'Silent Discard' interrupt vector with bits HRAB and RAB set is generated.

When the receive buffer request matches exactly the remaining size of the data section and the data block does not contain the end of a packet, it is stored completely in the data section. The descriptor is updated immediately (C bit set). With the next receive buffer request, the data management unit repolls the HOLD bit of the current receive descriptor. If the HOLD bit is removed, it branches to the next descriptor. If the HOLD Bit is still '1', an internal poll bit is set. Additionally a 'Hold Caused Receive Abort' interrupt vector is generated and the rest of the frame is discarded. As long as the HOLD bit remains set further data of the same channel is discarded and for each discarded frame a 'Silent Discard' interrupt vector is generated.

The system CPU can remove the hold condition, when the next receive descriptor is available in shared memory. Therefore the CPU has to execute a 'Receive Hold Reset' command, which will reactivate the channel. When the receive buffer requests a new data transfer, the data management unit will repoll the last receive descriptor. If the HOLD bit was removed, the data management unit branches to the next receive descriptor pointed to by bit field NextReceiveDescriptor.

*Note: In protocol modes HDLC and PPP data from receive buffer is discarded until the end of a received frame is reached. As soon as the beginning of a new frame is received, the data management unit starts to fill the data section.*

*Note: In transparent mode data transferred from receive buffer is written immediately to the data section of the next receive descriptor.*

If the CPU issues a 'Receive Hold Reset' command and does not remove the HOLD bit (erroneous programming), no action will take place.

#### **4.3.4 Transmit Descriptor**

The transmit descriptor in shared memory is initialized by the host CPU and is read afterwards by the MUNICH256. The address pointer to the first transmit descriptor is stored in the on-chip channel database, when requested to do so by the host CPU via the 'Transmit Init' command. The first three DWORDs of a transmit descriptor are read when the transmit buffer requests a data transfer for this channel and then they are stored in the on-chip memory. Also they are read when branching from one transmit

**Functional Description**

descriptor to the next transmit descriptor. Therefore all information in the next descriptor must be valid when the data management unit branches to a descriptor. The last DWORD of a transmit descriptor optionally is written by the MUNICH256 when processing of a descriptor has finished.

**Table 4-3 Transmit Descriptor Structure**

DWORD ADDR.	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
00 <sub>H</sub>	FE	HOLD	THI	CEN	0	0	0	0	0	0	DescriptorID(5:0)					
04 <sub>H</sub>	NextTransmitDescriptorPointer(31:2)															
08 <sub>H</sub>	TransmitDataPointer(31:0)															
0C <sub>H</sub>	0	C	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DWORD ADDR.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00 <sub>H</sub>	NO(15:0)															
04 <sub>H</sub>	NextTransmitDescriptorPointer(31:2)														0	0
08 <sub>H</sub>	TransmitDataPointer(31:0)															
0C <sub>H</sub>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**FE** Frame end  
It indicates that the current transmit data section (addressed by transmit data pointer) contains the end of a frame. After the last byte is read from system memory this bit is passed to the transmit buffer and to the protocol machine. The bit FE informs the transmit buffer to move a stored frame to the protocol machine even if the programmed transmit forward threshold is not reached (see **“Internal Transmit Buffer” on Page74**). The protocol machine is informed to append the checksum (HDLC, PPP) and then to send the interframe time-fill. Providing a transmit descriptor with FE = '0' and HOLD = '1' is an error.

**HOLD** Hold indication  
It indicates that this descriptor is the last valid element of a linked list.

0 Next descriptor is available in the shared memory. The data management unit branches to the next descriptor as soon as processing of the current descriptor has finished.

1 The current descriptor is the last descriptor containing valid data in the data section. As soon as the data management unit has transferred the data contained in the data section to the internal buffer, it tries one more time to read the descriptor. In case that

**Functional Description**

the hold indication is still set, it stores further requests of the receive buffer in its channel database. The channel can be reactivated by issuing a 'Transmit Hold Reset' command or by providing a new linked list via the 'Transmit Abort/Branch' command, in which case not served requests are processed.

*Note: When repolling a descriptor the MUNICH256 checks the HOLD bit and the bit field NextTransmitDescriptorPointer. All other information are NOT updated in the internal channel database.*

- NO** Byte Number
- The byte number defines the number of bytes stored in the data section to be transmitted. Thus the maximum length of data buffer is 65535 bytes. In order to provide dummy transmit descriptors NO = 0 is allowed in conjunction with the FE bit set. In this case (NO = 0) a 'Transmit Host Initiated' interrupt vector and/or the C-bit will be generated/set when the data management unit recognizes this condition. It is an error to set NO = 0 without FE bit set.
- THI** Transmit Host Initiated Interrupt
- This bit indicates that the MUNICH256 shall generate a 'Transmit Host Initiated' interrupt vector after it has finished operating on the descriptor.
- 0 Data management unit does not generate an interrupt vector after it has processed the transmit descriptor.
- 1 Data management unit generates an interrupt vector, as soon as all data bytes are transferred to the internal transmit buffer and the status information is updated.
- DescriptorID** This bit field is read by the data management unit and written back in the corresponding interrupt status of a channel interrupt vector which is generated by data management unit. This value provides a link between the descriptor and the corresponding interrupt vector.
- NextTransmitDescriptorPointer**
- This pointer contains the start address of the next transmit descriptor. It has to be DWORD aligned. After sending the indicated number of data bytes, the data management unit branches to the next transmit descriptor. The transmit descriptor is read entirely at the beginning of transmission and stored in on-chip memory. Therefore all informations in the descriptor must be valid.
- System CPU can force the MUNICH256 to branch to the beginning of a new linked list via the command 'Transmit Abort/Branch'. In this case the transmit descriptor address provided via register CSPEC\_FTDA is used as the next transmit descriptor pointer to be branched to.

### TransmitDataPointer

This 32-bit pointer contains the start address of the transmit data section. Although the data management unit works DWORD oriented, it is possible to begin transmit data section at byte addresses.

### CEN Complete Enable

This bit is set by the CPU if the complete bit mechanism is desired:

0 The data management unit will NOT update the transmit descriptor with the C bit. In this mode the use of the THI interrupt is recommended.

1 The data management unit will set the C bit.

### C Complete

This bit is set by the data management unit, when the bit CEN of a descriptor is set and when it

- completed reading a data section normally, or
- it was aborted by a 'Transmit Off' command or by a 'Transmit Abort/Branch' command.

The complete bit releases the descriptor.

## 4.3.5 Data Management Unit Transmit

The *data management unit transmit* provides the interface between system memory on one side and the internal transmit buffer on the other side. The data management unit handles requests of the transmit buffer, controls the address and burst length calculation, initiates data transfers from system memory to the transmit buffer and handles the linked lists on a per channel basis.

For initialization the CPU programs the first transmit descriptor address, the interrupt mask, the interrupt queue and starts the channel with the 'Transmit Init' command. For detailed description of channel commands refer to ["Channel Commands" on Page 116](#). The data management unit then fetches the given information and stores them in its on-chip channel database.

The first transmit descriptor is fetched from system memory and stored in the chip internal channel database the first time the transmit buffer requests data for a channel. It contains a pointer to the data buffer, the length of the data section as well as a pointer to the next transmit descriptor. After the first descriptor is stored internally a 'Transmit Command Complete' interrupt vector is generated.

Data transfers are requested as long as the number of empty locations is below a programmable refill threshold. The number of empty locations is reported from the transmit buffer to the data management unit. Task of the data management unit is to calculate the number of bytes that can be loaded from the data section based on the NO

---

**Functional Description**

field of the transmit descriptor and to compare this with the number of bytes requested by the transmit buffer.

Depending on the bit field NO in the transmit descriptor several read accesses must be performed by the data management unit. It stops serving the request as soon as the requested amount of data was transferred to the transmit buffer, when a Frame End bit (FE) in the processed transmit descriptor is set or when the channel was aborted using a 'Transmit Abort' command. Serving the request can also be suspended, when the programmed transmit burst length (CONF3.TPBL) is reached. All these events may result in open transmit buffer locations, but the data management unit stores this information as open requests in the channel database and processes these requests continuously.

The data management unit alternately serves requests issued by the transmit buffer or open requests stored in its internal channel database. If there are open requests for a channel, data transmission will be initiated. The procedure is the same as described above. It stops, if the requested amount of data is served or when the FE bit field is set.

If a transmit descriptor has its FE bit set and all data of the data section is moved to the transmit buffer, the data management unit serves requests of further channels or looks for open requests in its database. Therefore open requests from other channels are served faster and possible underruns can be avoided. The next transmit descriptor will be retrieved with the next data transfer of the channel.

When the data management unit completed reading a data section associated with a transmit descriptor, it updates the complete (C) bit in the status word of the transmit descriptor if the complete enable (CEN) bit is set. Additionally a 'Transmit Host Initiated' interrupt vector is generated if the THI bit is set in the transmit descriptor. Afterwards the data management unit the MUNICH256 branches to the next transmit descriptor.

**Descriptor hold operation**

The data transfer is controlled by the HOLD bit, which is located in the first DWORD of a transmit descriptor. The HOLD bit indicates that the marked descriptor is the last descriptor in a linked list. The data management unit will not branch to the next descriptor until the hold condition is removed or a 'Transmit Abort' command forces the MUNICH256 to branch to a new linked list.

If the HOLD bit and the frame end bit are set together in a descriptor, the data management unit transfers all data of the belonging data section to the transmit buffer and optionally sets the C-bit in the current transmit descriptor. When a new data transfer is requested (either from the transmit buffer or an open request) the data management unit repolls the descriptor. If the HOLD bit is removed, it will branch to the next transmit descriptor. If the HOLD bit is still set, that channel is suspended for further operation. Following requests from the transmit buffer will not be served, but the number of requested data is stored in the open request registers.

---

**Functional Description**

If the HOLD bit is detected in a descriptor and the frame end bit is not set, the data management unit will transfer all data of the belonging data section to the transmit buffer. Afterwards it generates a 'Hold Caused Transmit Abort' interrupt vector in order to inform the host CPU about the erroneous descriptor structure. In PPP and HDLC mode the abort status is propagated to the transmit buffer and the protocol machine, so that a abort sequence is sent on the serial side. In TMA mode the data management unit generates a 'Hold Caused Transmit Abort' interrupt vector every time it recognizes the HOLD bit. Then it reads the transmit descriptor once more. If the HOLD bit is removed it branches to the next transmit descriptor and proceeds with normal operation. Otherwise, when the HOLD bit is still set, the channel is suspended for further operation and an internal poll bit is set. Following requests from the transmit buffer will not be served, but the number of requested data is stored in the open request register.

The host CPU can remove the hold condition, when the next transmit descriptor is available in system memory. Therefore the CPU has to execute a 'Transmit Hold Reset' command, which will reactive the channel. When the transmit buffer requests a new data transfer or when open request are stored in the on-chip database the data management unit repolls the transmit descriptor and checks the HOLD bit again. If the HOLD bit is removed it branches to next transmit descriptor.

If the CPU issues a 'Transmit Hold Reset' command and does not remove the HOLD bit (erroneous programming), no action will take place. Nevertheless, the CPU always has to issue a 'Transmit Hold Reset' command when it removes the HOLD bit in a descriptor, no matter the data management unit has already seen the HOLD bit or not.

### **4.3.6 Byte Swapping**

The MUNICH256 operates per default as a little endian device. To support integration into big endian environments, the data management unit provides an internal byte swapping mechanism, which can be enabled via bit CONF1.LBE.

The big endian swapping applies only to the data section pointed to by the receive and transmit descriptors in the shared memory.

*Note: Byte swapping only effects the organization of packet data in system memory. All internal registers, as well as the descriptors, address pointers or interrupt vectors are handled with little endian byte ordering.*

**Table 4-4 Example for little/big Endian with BNO = 3**

BNO	Little Endian				Big Endian			
3	-	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	-

**Table 4-5 Example for little big Endian with BNO = 7**

BNO	Little Endian				Big Endian			
7	Byte3	Byte 2	Byte 1	Byte 0	Byte 0	Byte 1	Byte 2	Byte3
	-	Byte 6	Byte 5	Byte 4	Byte 4	Byte 5	Byte 6	-

### 4.3.7 Transmission Bit/Byte Ordering

Data is transmitted beginning with byte zero in increasing order. Vice versa data received is stored starting with byte zero. The position of byte zero depends on the selected endian mode.

Each byte itself consists of eight bits starting with bit zero (LSB) up to bit seven (MSB). Data on the serial line is transmitted starting with the LSB. The first bit received is stored in bit zero.

## 4.4 Buffer Management

### 4.4.1 Internal Receive Buffer

The internal receive buffer provides buffering of frame data and status between the protocol handler and the receive data management units. Internal buffers are essential to avoid data loss due to the PCI bus latency, especially in the presence of multiple devices on the same PCI bus, and to enable a minimized bus utilization through burst accesses.

The incoming data from the protocol handler is stored in a receive central buffer shared by all the 256 channels. The buffer is written by the protocol handler every time a complete DWORD is ready or the last byte of a frame has been received. Each channel has an individual programmable threshold code, which determines after how many DWORDs a data transfer into the shared memory is generated. The threshold therefore defines the maximum burst length for a particular channel in receive direction. A data transfer is also requested as soon as a frame end has been reached. Programming the burst length to be greater than 1 DWORD avoids too frequent accesses to the PCI bus, thereby optimizing use of this resource.

For real time channels with lowest possible latency (example: constant bit rate) a value of one DWORD can be selected for the burst length.

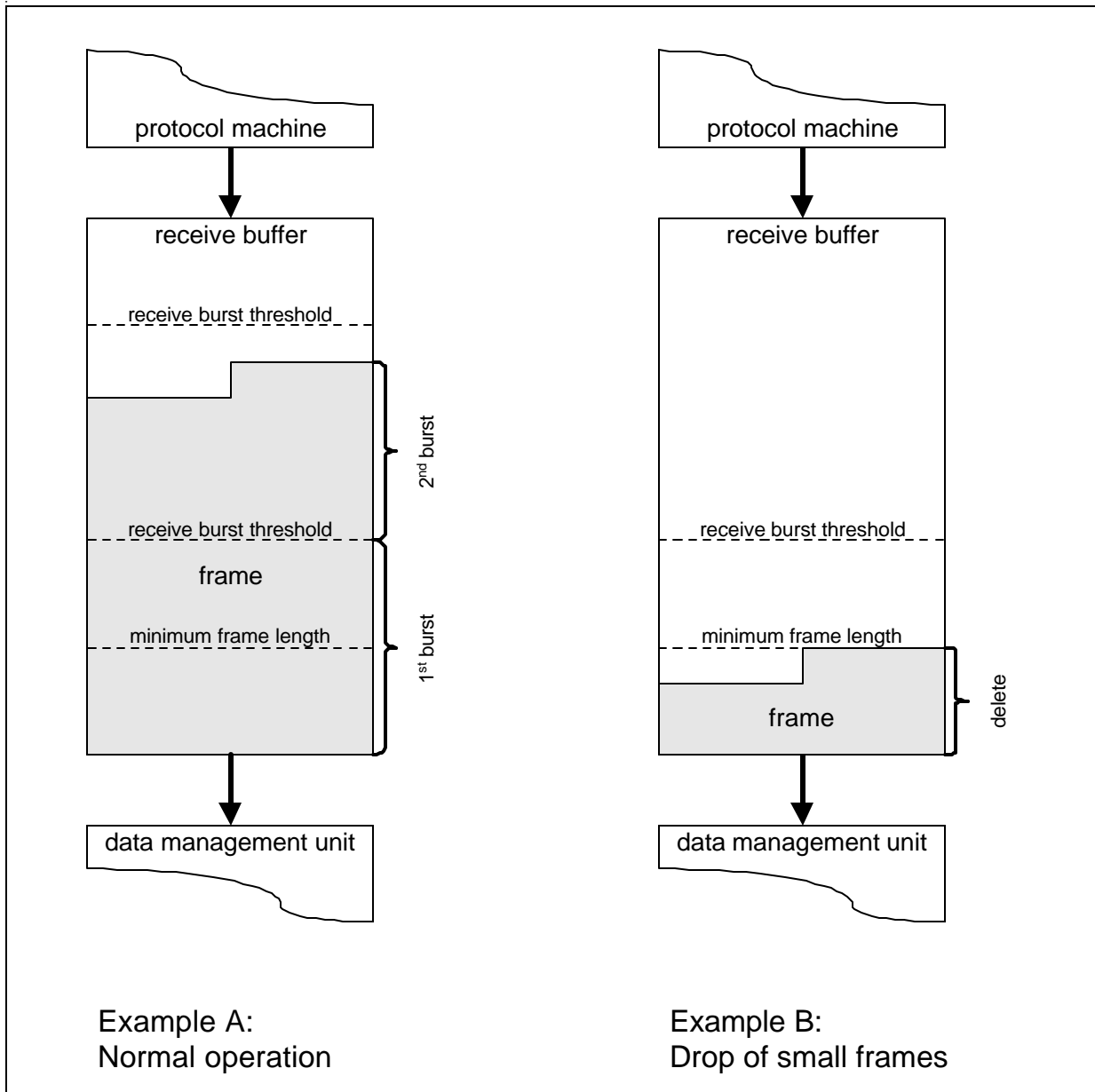


---

## **Functional Description**

The total size of the internal receive buffer is 12 kByte. If all the 256 channels are active, the average burst threshold should be programmed with 8 DWORDs, so that 4 DWORDs are available on the average to compensate for PCI latency and avoid data loss. However if less than 256 channels are active or if only 64 KBit/s channels are used, the burst threshold may be programmed to a higher value. In other words, the sum of all channel thresholds shall not exceed the maximum receive buffer locations.

In order to prevent an overload condition from one particular channel (e.g. receiving only small or invalid frames), the receive buffer provides the capability to delete frames which are smaller or equal than a programmable threshold. All frames that have been dropped will be counted and an interrupt vector will be generated as soon as a programmable threshold has been reached. The actual value of the counter can be read in the small frame dropped counter register.



**Figure 4-8 Receive Buffer Thresholds**

For performance monitoring the receive buffer provides the capability to monitor the receive buffer utilization and to generate interrupts when certain fill thresholds have been reached.

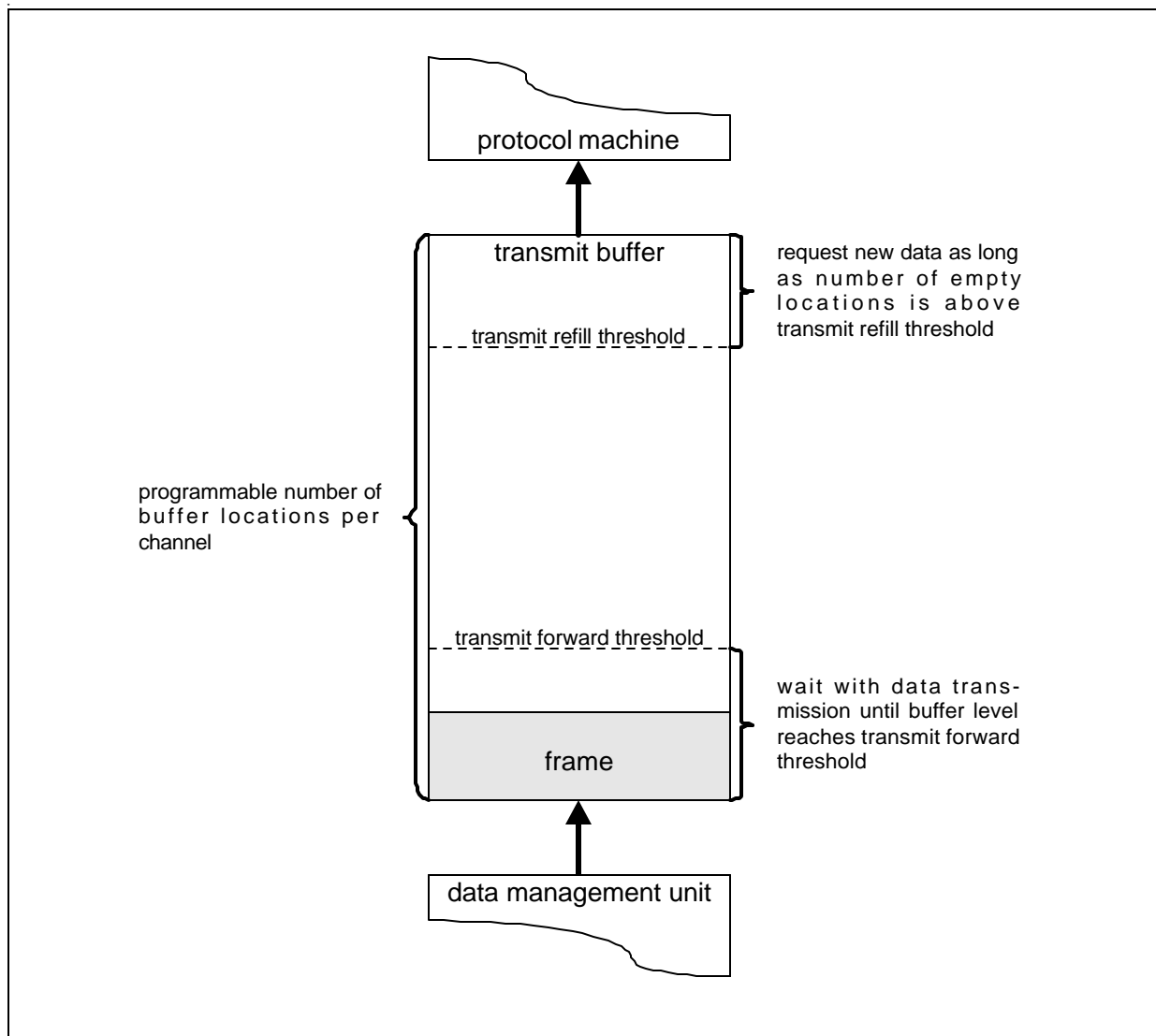
#### 4.4.2 Internal Transmit Buffer

The internal transmit buffer with a total size of 32 kByte stores protocol data before it is processed by the protocol machine. The transmit buffer is essential to ensure that enough data is available during transmission, since PCI latency and usage of multiple

**Functional Description**

channels limit access to system memory for a particular channel. A programmable transmit buffer size and two programmable threshold are configurable by the host CPU for each channel.

*Note: The sum of both thresholds must be smaller than the transmit buffer size of a particular channel.*



**Figure 4-9 Transmit Buffer Thresholds**

The threshold values have the following effect:

- Data belonging to one channel stored in the internal transmit buffer will only be transferred to the protocol machine when the transmit forward threshold is reached or if a complete frame is stored inside the transmit buffer. This mechanism avoids data underrun conditions.

**Functional Description**

- As long as the amount of data stored in the transmit buffer is below the transmit refill threshold the data management unit will keep filling the buffer by initiating PCI burst transfers.

*Note: Since there is a delay between the time the transmit buffer requests data from the data management unit and the time the data management unit serves the request, the actual number of empty locations may be higher than the transmit refill threshold. To determine the maximum PCI burst length an additional parameter is available which limits these requests up to a maximum of 64 DWORDs.*

## 4.5 Protocol Description

The protocol machines provide protocol handling for up to 256 channels. The protocol machines implement 4 modes, which can be programmed independently for each channel: HDLC, bit-synchronous PPP, octet-synchronous PPP and transparent mode A.

The configuration of each logical channel is programmed via the PCI bus and will be stored inside the protocol machines. Furthermore the current state for the protocol processing (CRC check, 1 bit count,...) is also stored inside the protocol machines.

Each protocol machine (receive, transmit) handles a maximum of 256 channels and a maximum aggregate bit rate of up to 90 Mbit/s.

### 4.5.1 HDLC Mode

<b>Flag</b> 0111 1110	<b>Address</b> 8 bits	<b>Control</b> 8 bits	<b>Information</b> ≤0 Bits	<b>CRC</b> 16/32 bits	<b>Flag</b> 0111 1110
--------------------------	--------------------------	--------------------------	-------------------------------	--------------------------	--------------------------

**Figure 4-10 HDLC Frame Format**

The frame begin and frame end synchronization is performed with the flag character 7E<sub>H</sub>. Shared opening and closing flag is supported in receive direction and can be programmed in the channel configuration register for transmit direction. Shared '0' bit between two flags is only supported in receive direction. Interframe time-fill can be programmed to either flag 7E<sub>H</sub> or FF<sub>H</sub> indicating idle.

In receive operation, prior to Frame check sum (FCS) computation, any '0' bit that directly follows five contiguous '1' bits is discarded. When closing flag is recognized, a CRC check, octet boundary check, MFL (maximum frame length) check, a short frame check and an additional small frame check are performed. Short frames have less than 4 octets if CRC16 is used or less than 6 octets if CRC32 is used. An aborted frame is recognized if 7 or more '1's are received.

In transmit operation after the CRC computation a '0' bit is inserted after every sequence of five contiguous '1' bits. When frame end is indicated in the belonging transmit descriptor the calculated CRC is transmitted and a flag is generated. If an underrun

occurs in the internal transmit buffer (because of PCI latency e.g.) an abort sequence with 7 '1's is transmitted and an underrun interrupt is generated. The abort sequence is also generated if the host CPU resets or aborts a channel during the transmission of a frame.

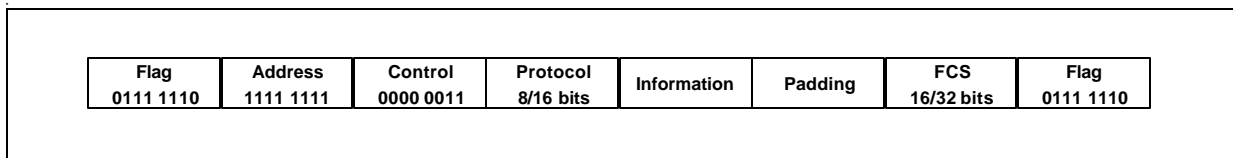
An invert option is provided to invert all the data output or data input between serial line and protocol machines or vice versa.

The following CRC modes are supported:

- 16 bit CRC  $1+x^5+x^{12}+x^{16}$
- 32 bit CRC  $1+x+x^2+x^4+x^5+x^7+x^8+x^{10}+x^{11}+x^{12}+x^{16}+x^{22}+x^{23}+x^{26}+x^{32}$

Optionally CRC transfer and check can be disabled.

### 4.5.2 Bit Synchronous PPP with HDLC Framing Structure



**Figure 4-11 Bit Synchronous PPP with HDLC Framing Structure**

Same as HDLC. The handling of the abort sequence differs from that in HDLC mode. If  $7E_H$  is programmed as interframe time fill character, the abort sequence consists of 7 "1"s. If  $FF_H$  is programmed as interframe time fill character, the abort sequence consists of 15 "1"s.

The same programmable parameters as in HDLC mode apply to bit synchronous PPP.

### 4.5.3 Octet Synchronous PPP

This mode uses a frame structure similar to the bit synchronous PPP mode. The frame begin and end synchronization is performed with the flag character ( $7E_H$ ). Use of a shared opening and closing flag is supported if programmed in the channel configuration register. Use of a shared '0' bit between two flags is not supported. A 16 or 32 bit CRC is computed over all service data read from the transmit buffer and appended to the end of the frame.

The octet synchronous PPP mode uses octet stuffing instead of '0' bit stuffing in order to replace control characters used by intervening hardware equipment. This allows transparent transmission and also recognition and removal of spurious characters inserted by such equipment.

A 32 bit per channel asynchronous control character map (ACCM) specifies characters in the range  $00_H-1F_H$  to be stuffed/destuffed in service data and FCS field. In addition, the DEL control character ( $7F_H$ ) and any of 4 ACCM extension characters stored in a programmable 32 bit register can be selected for character stuffing/destuffing. When a

---

**Functional Description**

character specified to be mapped is found in service data or the FCS field, it is replaced by a 2 octet sequence consisting of 7D<sub>H</sub> (Control Escape) followed by the character EXORed with 20<sub>H</sub> (e.g. 13<sub>H</sub> is mapped to 7D<sub>H</sub> 33<sub>H</sub>). In addition to the per channel specification of characters to be mapped, the control escape sequence 7D<sub>H</sub> and 7E<sub>H</sub> in the service data stream are always mapped. Opening and closing flags are not affected. The abort sequence consists of the control escape character followed by a flag character 7E<sub>H</sub> (not stuffed).

Between two frames, the interframe time fill character is always 7E<sub>H</sub>.

If in the transmit direction a data underrun occurs during transmission of a frame and the frame has not finished, an abort sequence is automatically sent (escape character followed by a flag) and an underrun interrupt vector will be generated. If the transmit buffer indicates an empty condition for a channel between two frames (idle or interframe fill), the protocol machine will continue to send interframe time fill characters. Also an abort sequence will be generated if a channel is reset or an abort command is issued during transmission of a frame.

The following CRC modes are supported:

- 16 bit CRC  $1+x^5+x^{12}+x^{16}$
- 32 bit CRC  $1+x+x^2+x^4+x^5+x^7+x^8+x^{10}+x^{11}+x^{12}+x^{16}+x^{22}+x^{23}+x^{26}+x^{32}$

CRC computation/check or removing can be disabled.

#### 4.5.4 Transparent Mode

When programmed in transparent mode, the protocol machine performs fully transparent data transmission/reception without HDLC framing, i.e. without

- Flag insertion/removing
- CRC generation/CRC check
- Bit stuffing/destuffing (0 bit insertion/removal).

An option 'Transparent Mode Pack' is provided to support subchanneling. If subchanneling is used (logical channels of less than 64 kbit/s), masked bits in the protocol data are set high and each bit in shared memory maps directly to enabled (not masked) bits on the serial line. Otherwise they contain protocol data, that is each byte in shared memory maps directly to a time slot.

A programmable transparent flag can be programmed which will be inserted between payload data or is removed during reception of a payload data.

An invert option is provided to invert the outgoing or incoming data stream.

#### 4.6 Mailbox

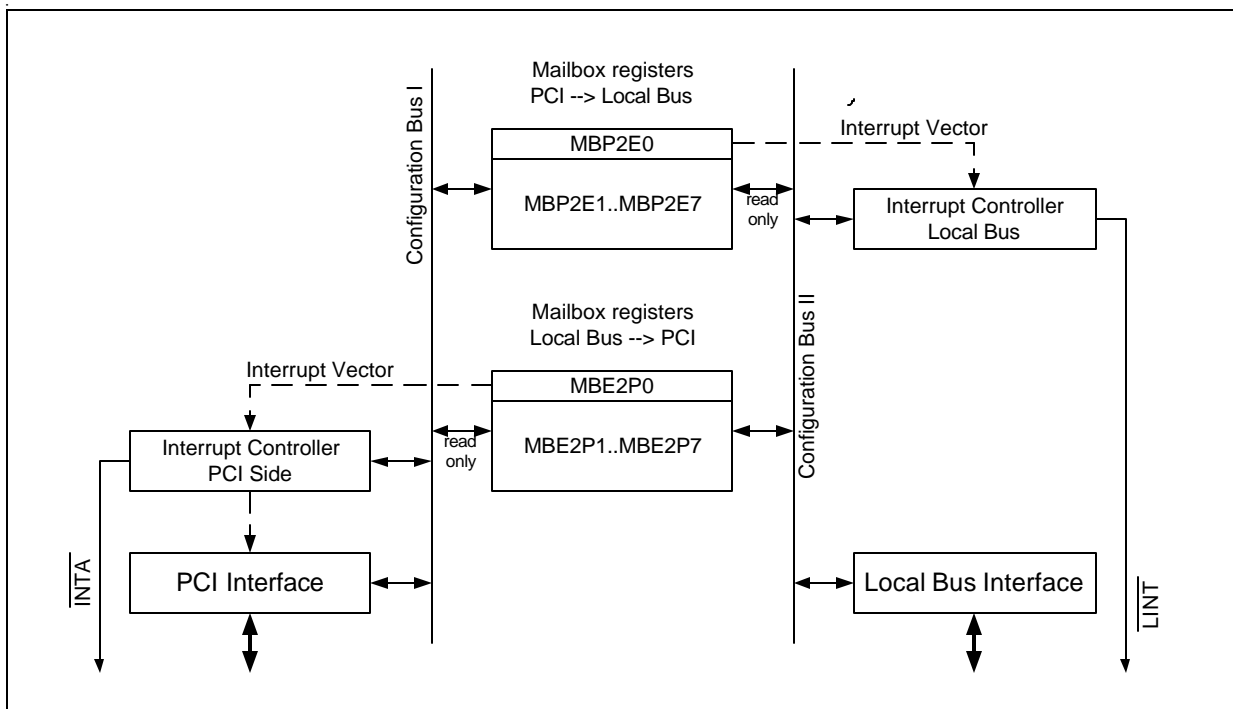
The MUNICH256 contains a mailbox to allow communication between two intelligent peripherals connected to the PCI bus and the local microprocessor bus. The mailbox is organized in two pages of eight registers. The first page is used to store information from

**Functional Description**

the PCI side and to read the information from the local microprocessor side. The second page is used for the opposite direction, from the local microprocessor side to the PCI side. Each page consists of one status register and seven data registers.

The mailbox provides a ‘doorbell’ capability. In this case an interrupt vector can be generated to inform the addressed intelligent peripheral that new information has been stored in the mailbox. This interrupt vector will be generated on write accesses to the status register of the selected page.

As an **example**, consider when the PCI host system wants to transfer data to an intelligent peripheral. First it loads data into the mailbox data registers MBP2E1 through MBP2E7, and then writes a status information to the mailbox status register MBP2E0. This last action causes an interrupt vector to be written to the interrupt FIFO which is connected to the local bus. The presence of an interrupt vector results in assertion of pin  $\overline{\text{LINT}}$ . The intelligent peripheral recognizes the interrupt pin asserted and reads the interrupt vector out of the interrupt FIFO (which results in deassertion of pin  $\overline{\text{LINT}}$ ), and then reads data from the mailbox data registers.



**Figure 4-12 Mailbox Structure**

**Alternately**, consider when an intelligent peripheral connected to the local bus wants to transfer data to the PCI host system. First it loads data into the mailbox data registers MBE2P1 through MBE2P7 and then it writes status information to the mailbox status register MBE2P0. This causes a system interrupt vector to be written to the PCI host system, indicating that valid data is contained in the mailbox data registers.

This interrupt vector will be written to the interrupt queue specified in CONF1.SYSQ and together with this the pin  $\overline{INTA}$  will be asserted. The processor sees the interrupt pin asserted, reads the register GISTA in order to determine the interrupt queue, and then writes a '1' to the interrupt status acknowledge register GIACK to clear the interrupt. Next, it reads the interrupt vector which contains a copy of the mailbox status register and then reads the mailbox data registers.

## **4.7 Interrupt Controller**

All layer two interrupts (channel, port, system and command interrupts) are handled via an internal interrupt controller which forwards those interrupts to external interrupt queues. This interrupt controller is connected to the PCI interrupt pin  $\overline{INTA}$ .

Mailbox interrupts are handled via an internal interrupt FIFO which is connected to the local bus interrupt pin  $\overline{LINT}$  (normal operation). Additionally the interrupts stored in the internal interrupt FIFO can be notified via the PCI interrupt pin  $\overline{INTA}$ .

The MUNICH256 also provides the capability to bridge the local bus interrupt  $\overline{LINT}$  to the PCI bus.

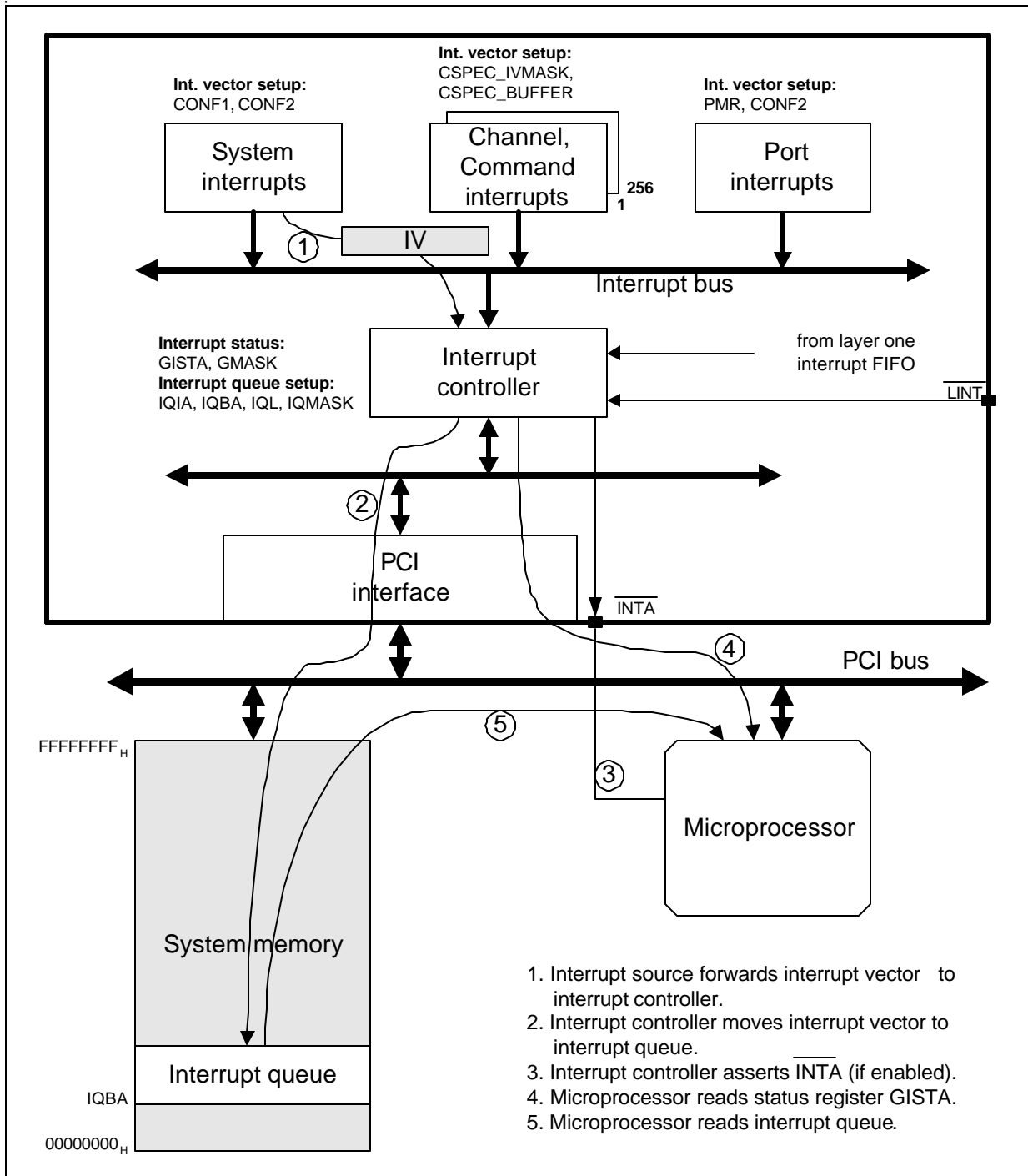
### **4.7.1 Layer Two interrupts**

All channel interrupts, port interrupts and system interrupts are written in form of interrupt vectors to interrupt queues.

Each interrupt vector has an interrupt source. An interrupt source is either a channel, the port handler or certain device functions (system interrupts). After reset no interrupt vector is generated since port and system interrupts are masked and channels are in their idle state.

Each interrupt source forwards its interrupt vector to the interrupt controller, together with the information in which interrupt queue the vector should be forwarded. The interrupt controller moves the interrupt vector to the selected interrupt queue. Channel interrupts can optionally be forwarded to a dedicated high priority interrupt queue (interrupt queue seven). A programmable interrupt queue high priority mask determines channel interrupts, which shall be forwarded into the high priority interrupt queue instead of queueing them in the selected interrupt queue. This function is available for each interrupt queue and allows to queue important interrupt conditions in the high priority queue.





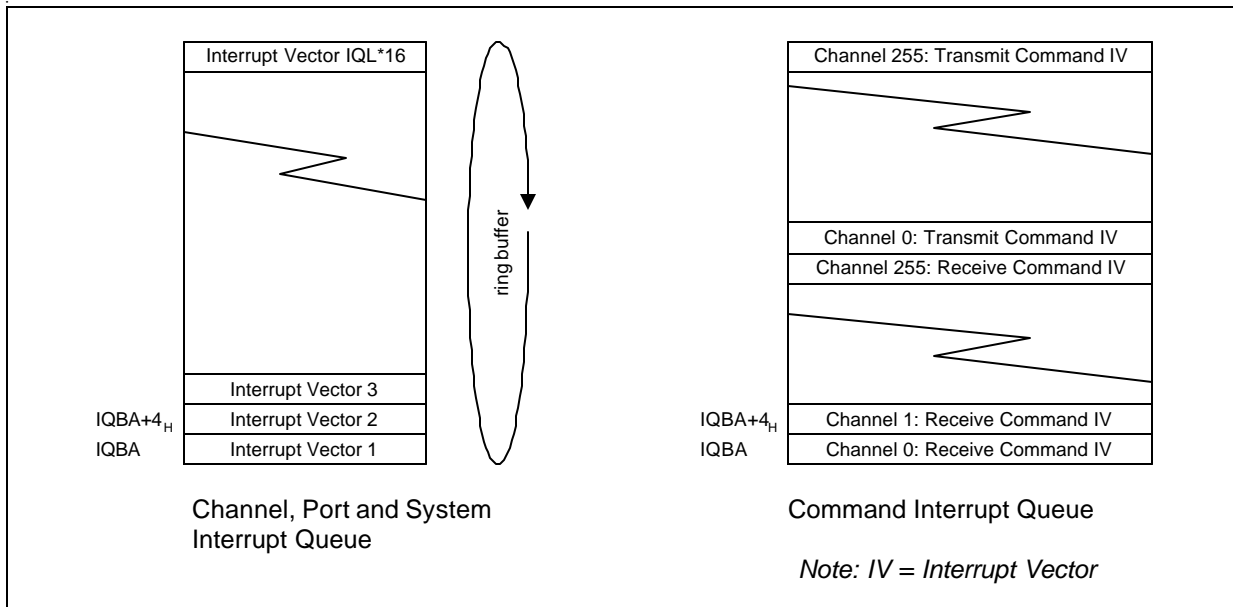
**Figure 4-13 Layer Two Interrupts (Channel, command, port and system interrupts)**

As soon as the interrupt controller has written an interrupt vector to one of the nine interrupt queues the PCI interrupt pin INTA is asserted. The global interrupt status register indicates in which interrupt queue the interrupt vector can be found. Each of the

**Functional Description**

nine interrupt queues can be masked. In this case the interrupt pin  $\overline{INTA}$  is not asserted, but the interrupt vector is still written into the assigned interrupt queue.

An interrupt queues is a reserved memory locations in system memory. The MUNICH256 supports up to eight interrupt queues which are organized in form of ring buffers with a programmable start address and a programmable size per interrupt queue. Additionally there is one fixed sized command interrupt queue where command interrupts are stored. The size of this queue is two times 256 DWORDS (**Figure 4-14**).



**Figure 4-14 Interrupt Queue Structure in System Memory**

**4.7.1.1 General Interrupt Vector Structure**

Each interrupt vector is 32 bit wide and contains several subfields, which indicate the interrupt group and depend on the interrupt group the interrupt information. Bit 31 of the interrupt vector is generally set to '1' by the MUNICH256 and allows the system CPU to clear the bit in order to mark processed interrupts.

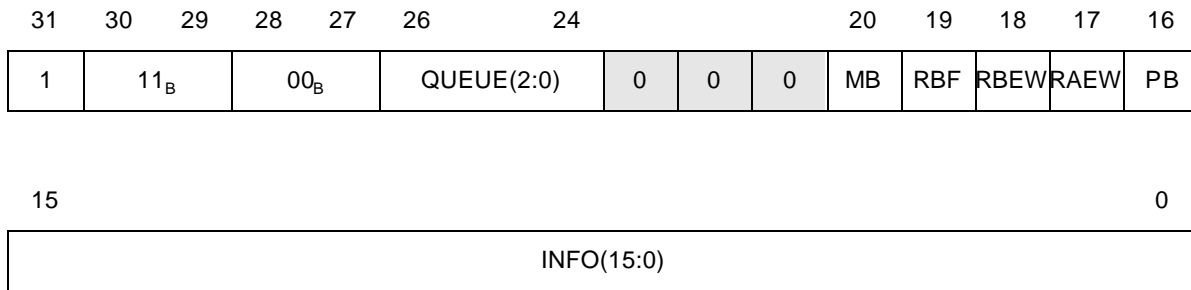
**Table 4-6 Interrupt Vector Structure**

	31	30	29	28	27	26	24	23		16
	1	TYPE(1:0)	STYPE(1:0)	QUEUE(2:0)			INT(23:0)			
	15	INT(23:0)								0

**Functional Description**

TYPE	<p>Interrupt type</p> <p>The interrupt vectors are divided into four basic groups, where TYPE determines the interrupt group. A further classification of interrupts is done with the subtype indication.</p> <p>00<sub>B</sub> Command interrupts</p> <p>01<sub>B</sub> Channel interrupts</p> <p>10<sub>B</sub> Port interrupts</p> <p>11<sub>B</sub> System interrupts</p>
STYPE	<p>Interrupt subtype</p> <p>A specific interrupt type is divided into several subtypes. In general STYPE(1) indicates the data path (transmit, receive) generating the interrupt.</p>
QUEUE	<p>Interrupt queue</p> <p>The interrupt vectors are written into 9 external interrupt queues located in the shared memory. Corresponding to these 9 queues are 9 interrupt queue start addresses and 8 interrupt queue length registers, since the interrupt queue 8 has a fixed length of 2 x 256).</p>
INT	<p>Interrupt Information</p> <p>INT itself contains the interrupt information. The meaning of INT is dependent on TYPE and STYPE indication.</p>

### 4.7.1.2 System Interrupts

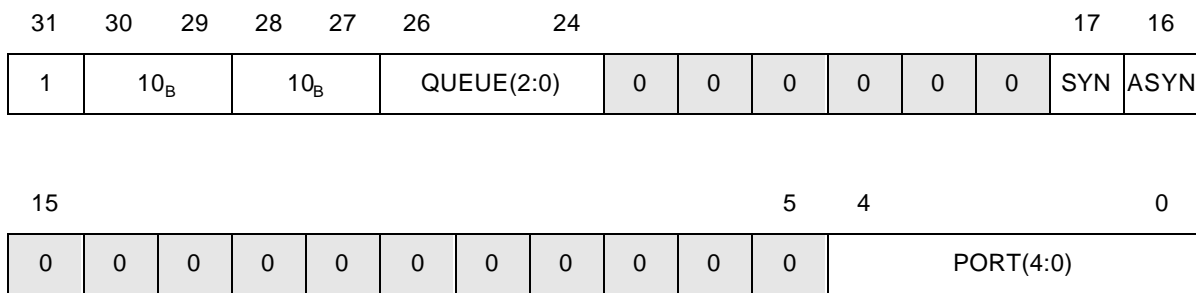


- MB**            Mailbox
- The 'Mailbox' interrupt vector is generated, in case that the local microprocessor has written data to the mailbox status register MBE2P0. The bit field INFO contains a copy of MBE2P0.
- RBAF**            Receive Buffer Access Failed
- The 'Receive Buffer Access Failed' interrupt vector is generated, when the protocol machine discarded packets due to permanent inaccessibility of the receive buffer. This interrupt is issued as soon as the programmable threshold stored in register RBAFT is reached. The actual value of discarded packets is stored in register RBAFC.
- RBEW**            Receive Buffer Queue Early Warning
- The 'Receive Buffer Queue Early Warning' interrupt vector is generated, when the receive buffer data threshold has been exceeded (RBTH.RBTH). This interrupt can be masked via bit CONF1.RBIM.
- RAEW**            Receive Buffer Action Queue Early Warning
- The 'Receive Buffer Action Queue Early Warning' interrupt vector is generated, when the receive data action queue threshold (RBTH.RBAQTH) has been exceeded. The receive buffer action queue stores all requests of the receive buffer to forward data packets to system memory. This interrupt vector can be masked via bit CONF1.RBIM.
- PB**                PCI Access Error
- The 'PCI Access Error' interrupt vector is generated, when system software tries to read/write internal registers with accesses that do not enable all byte lanes, e.g. the access is not a full 32 bit access. The bit field INFO contains the register address which was tried to access.
- INFO**            Contains additional interrupt information data according to the bit, which is set: See specific interrupt for details.

### 4.7.1.3 Port Interrupts

Port interrupt vectors indicate the synchronous or asynchronous state of a port. Immediately after enabling both, the port and the port interrupts, port interrupts are generated indicating the synchronous or asynchronous state of a port. After this initial interrupt vector generation, further interrupts are written only when the state of a port changes from synchronous state to asynchronous state or vice versa. Port interrupts are enabled by resetting the corresponding mask bit in register PMR.

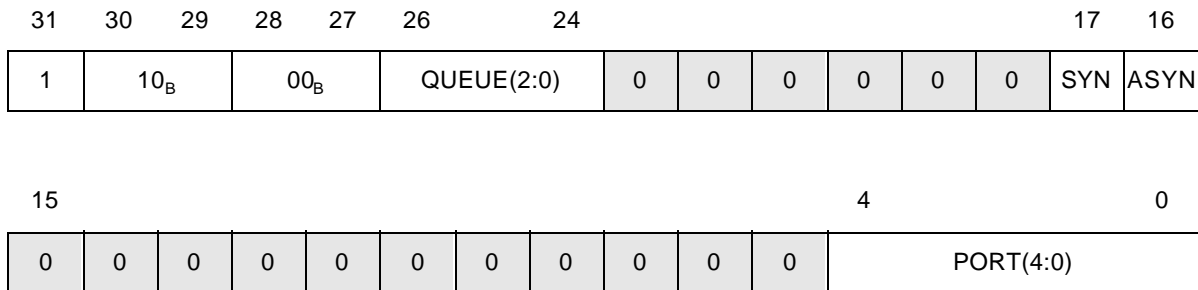
#### Transmit interrupts



- PORT**      Port Number  
This bit field identifies the port for which the information in the interrupt vector is valid.
- SYN**      Synchronization achieved  
Port has changed from asynchronous state to synchronous state. This interrupt is available for ports configured in T1 or E1, 4.096 MHz mode or 8.192MHz mode. In unchannelized mode there is no synchronous state.  
A transmit port changes to the synchronous state if the number of bits between two synchronization pulses is equal to a multiple of the number of frame bits of the selected mode. The first synchronization pulse after a port is enabled causes the port to change to the synchronous state.
- ASYN**      Asynchronous State  
The transmitter generates an 'Asynchronous State' interrupt vector if a port has changed from synchronous to asynchronous state. This interrupt is available for ports configured in T1, E1, 4.096MHz mode or 8.192 MHz mode. In unchannelized mode there is no asynchronous state. In general a port is in asynchronous state when a port is disabled.  
A port changes to the asynchronous state if the number of bits between two synchronization pulses is not equal to a multiple of the number of frame bits of the selected mode.

Functional Description

Receive Interrupts



- PORT**            Port Number  
This bit field identifies the port for which the information in the interrupt vector is valid.
- SYN**             Synchronization achieved  
Port has changed from asynchronous state to synchronous state. This interrupt is available for ports configured in T1, E1, 4.096MHz mode or 8.192 MHz mode. In unchannelized mode there is no synchronous state. A receive port changes to the synchronous state if the number of bits between two synchronization pulses is equal to a multiple of the number of frame bits of the selected mode. The first synchronization pulse after a port is enabled causes the port to change to the synchronous state.
- ASYN**            Asynchronous state  
Port has changed from synchronous to asynchronous state. This interrupt is available for ports configured in T1 or E1, 4.096 MHz mode or 8.192MHz mode. In unchannelized mode there is no asynchronous state. In general a port is in asynchronous state when a port is disabled. A port changes to the asynchronous state if the number of bits between two synchronization pulses is not equal to a multiple of the number of frame bits of the selected mode.

#### 4.7.1.4 Channel Interrupts

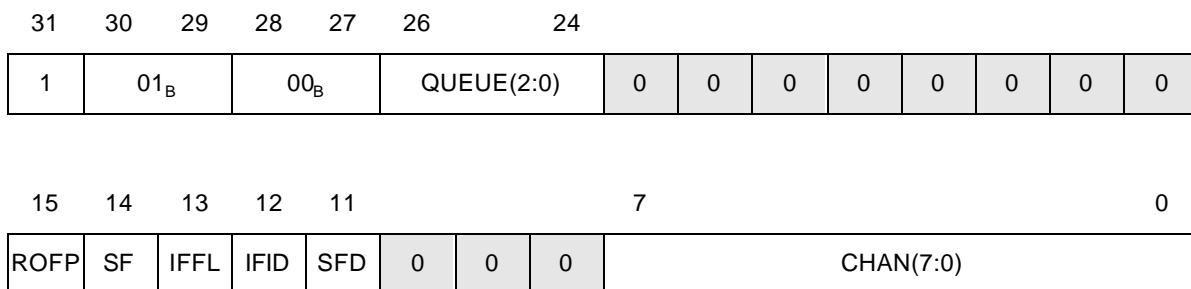
Channel interrupt are divided into two subtypes:

- Receive Interrupt I and Transmit Interrupt I
- Receive Interrupt II and Transmit Interrupt II

Subtype I contains interrupts which indicate the general status of a channel. These interrupts are not linked to a descriptor.

Subtype II contains interrupts which indicate a channel or packet status that is linked to a descriptor. Each interrupt vector contains a descriptor ID which can be used for tracking purposes.

##### Receive Interrupt I

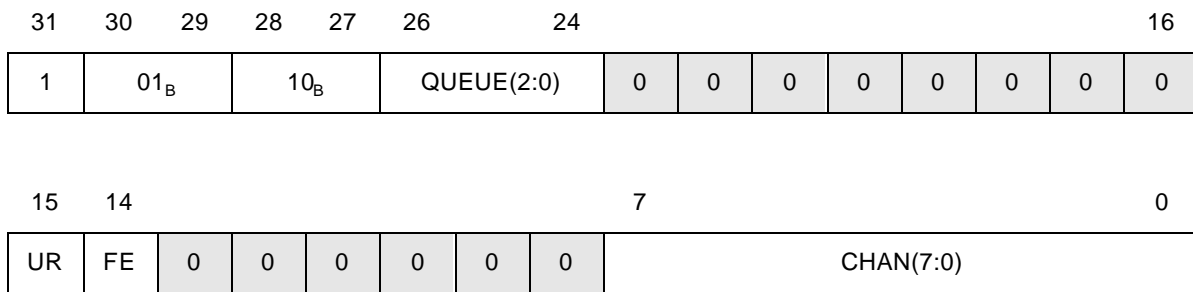


- ROFP**            Receive Buffer Overflow  
The 'Receive Buffer Overflow' interrupt vector is generated, when one or more whole frames or short frames or changes of interframe time-fill (HLDC, PPP) or data in general (TMA) has been discarded due to the inaccessibility of the internal receive buffer.
- SF**                Short Frame Detected  
The 'Short Frame Detected' interrupt vector is generated, when the receiver detected a frame which length matches the condition defined in CONF1.SFL.
- IFFL**            Interframe Time-fill Flag  
The 'Interframe Time-fill Flag' interrupt vector is generated, when the receiver detected a interframe time-fill change from FF<sub>H</sub> to 7E<sub>H</sub>.
- IFID**            Interframe Time-fill Idle  
The 'Interframe Time-fill Idle' interrupt vector is generated, when the receiver detected a interframe time-fill change from 7E<sub>H</sub> to FF<sub>H</sub>.

**Functional Description**

- SFD**            Small Frames Dropped  
The 'Small Frames Dropped' interrupt vector is generated, when the receiver discarded N small frames. The length of small frames is defined in CONF3.MINFL and the threshold value N is defined in register SFDT.
- CHAN**           Channel Number  
This bit field identifies the channel for which the information in the interrupt vector is valid.

**Transmit Interrupt I**

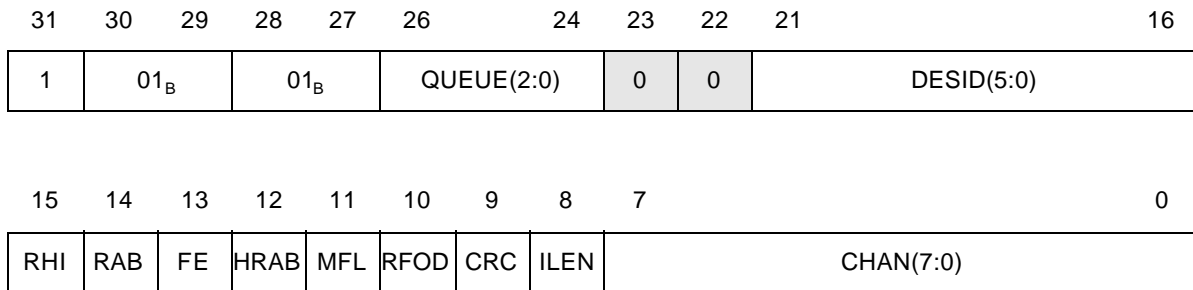


- UR**            Underrun  
The 'Underrun' interrupt vector is generated, when the transmit buffer was not able to provide data to the protocol machine transmit. If this happens during transmission of a HDLC or PPP packet, the transmitter will end the already started data packet with an abort sequence.
- FE**            Frame End  
The 'Frame End' interrupt vector is generated, when one complete data packet has been transmitted via serial side.
- CHAN**           Channel Number  
This bit field identifies the channel for which the information in the interrupt vector is valid.



Functional Description

Receive Interrupt II

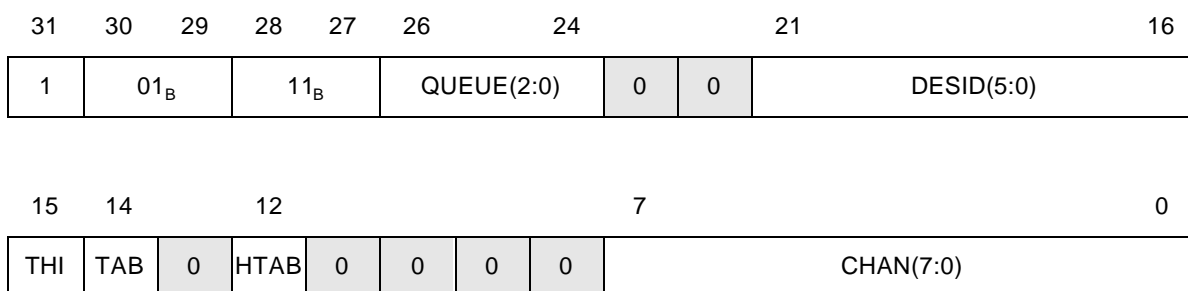


- CHAN** Channel Number  
This bit field identifies the channel for which the information in the interrupt vector is valid.
- RHI** (Receive) Host Initiated Interrupt  
The '(Receive) Host Initiated' interrupt vector will be issued, if bit RHI is set in a receive descriptor and processing of this descriptor has finished. After receiving this interrupt vector, system software can release the descriptor, e.g. put the descriptor into a free pool.
- RAB** Receive Abort  
The 'Receive Abort' interrupt vector is generated, when an incoming data packet is aborted (more than 6 '1' in case of HDLC or more than 15 '1' in case of PPP) or if the receiver got a receive abort command from the system CPU.
- FE** Frame End  
The 'Frame End' interrupt Vector is generated, when one complete frame has been received completely and has been stored in system memory.
- HRAB** Hold Caused Receive Abort  
The 'Hold Caused Receive Abort' interrupt vector is generated, when the receiver discarded the first data packet after it has found a HOLD bit in a receive descriptor.
- RAB, HRAB** Silent Discard  
The 'Silent Discard' interrupt vector (bit RAB and HRAB set together) occurs, if two or more frames have been discarded by the receiver due to continuous inaccessibility of receive descriptor. This occurs, if receive descriptor has HOLD bit set and receiver gets further data packets. The interrupt vector will be generated for each packet discarded.

**Functional Description**

- MFL**            **Maximum Frame Length Exceeded**  
 The 'Maximum Frame Length Exceeded' interrupt vector is generated, when the length of a received data packet exceeded the frame length defined in CONF1.MFL.
- RFOD**           **Receive Frame Overflow DMA**  
 The 'Receive Frame Overflow DMA' interrupt indicates that protocol handler was unable to transfer data to the receive buffer. As soon as receive buffer can store data again, this interrupt is generated.
- CRC**            **CRC Error**  
 The 'CRC Error' interrupt vector is generated, when the internally calculated CRC and the CRC of a received packet did not match.
- ILEN**           **Invalid Length**  
 The 'Invalid Length' interrupt vector is generated, when the bit length of received frame was not divisible by 8.

**Transmit Interrupt II**



- DESID**           **Descriptor ID**  
 This bit field is a copy of the descriptor ID of the transmit descriptor which is currently in use. It can be used for tracking purposes.
- THI**            **(Transmit) Host Initiated Interrupt**  
 The '(Transmit) Host Initiated' interrupt vector is generated, if bit THI is set in a transmit descriptor and processing of this descriptor has finished. After receiving this interrupt vector, system software can release the descriptor, e.g. put the descriptor into a free pool.
- TAB**            **Transmit Abort**  
 The 'Transmit Abort' interrupt vector is generated, either when the 'Transmit Abort/Branch' command was given and therefore one frame could not be transmitted completely or when NO and FE were set to 0 in a transmit descriptor and previous frame was incompletely specified.

---

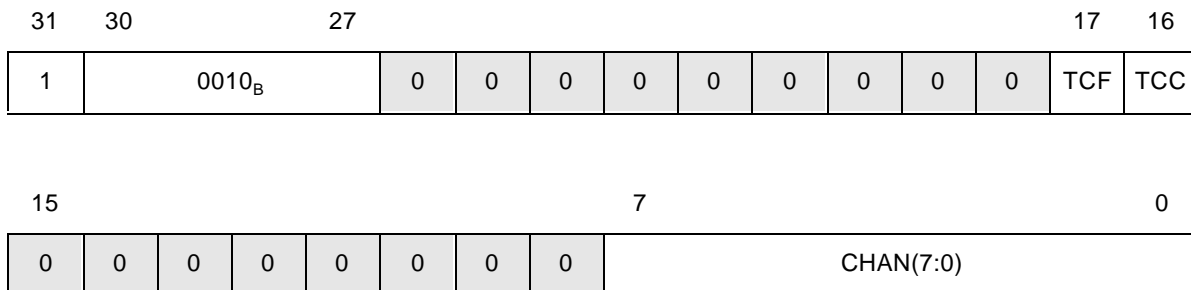
**Functional Description**

HTAB	<b>Hold Caused Transmit Abort</b> The 'Hold Caused Transmit Abort' interrupt vector is generated, when data management unit retrieved a transmit descriptor where HOLD was set and FE equals 0. The interrupt will be generated after the data section was transferred completely. After transmission of frame based protocols (HDLC, PPP) protocol machine appends abort sequence due to incomplete packet.
CHAN	<b>Channel Number</b> This bit field identifies the channel for which the information in the interrupt vector is valid.

### 4.7.1.5 Command Interrupts

Command interrupts are written to the command interrupt queue (interrupt queue eight).

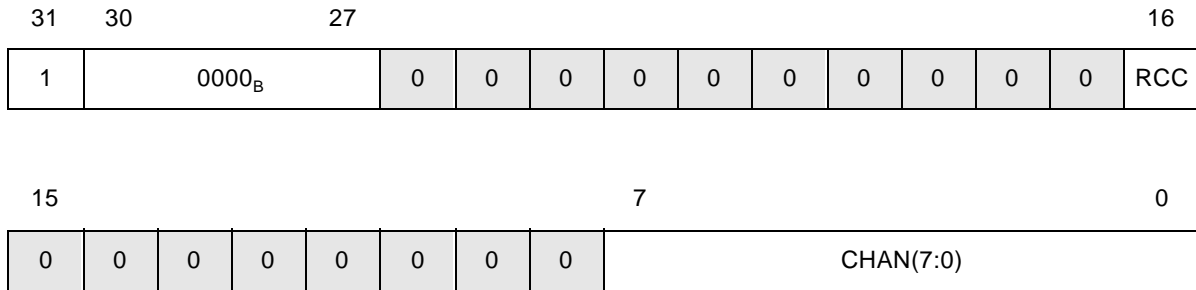
#### Transmit Interrupts



- TCF**      Transmit Command Failed
- The 'Transmit Command Failed' interrupt vector is issued, if the command 'Transmit Init' given via register *CSPEC\_CMD.XCMD* could not be finished. This happens, when
- system software tried to allocate more buffer locations for a channel than were available.
  - system software specified thresholds (transmit forward threshold, transmit refill threshold), which were greater than the specified transmit buffer size.
- Note: The sum of both thresholds must be smaller than the transmit buffer size of a particular channel. Erroneous programming does NOT result in the 'Transmit Command Failed' interrupt vector.*
- TCC**      Transmit Command Complete
- The 'Transmit Command Complete' interrupt vector is issued after successful completion of commands 'Transmit Init' and 'Transmit Off', which can be issued via register *CSPEC\_CMD.XCMD*.
- CHAN**      Channel Number
- This bit field contains the channel number of the affected channel.

Functional Description

Receive Interrupts



- RCC            Receive Command Complete  
The 'Receive Command Complete' interrupt vector is issued after successful completion of commands 'Receive Init' and 'Receive Off', which can be issued via register *CSPEC\_CMD.RCMD*.
- CHAN          Channel Number  
This bit field contains the channel number of the affected channel.

### 4.7.2 Mailbox Interrupts to the Local Bus

Mailbox interrupts are stored in an internal interrupt FIFO which is located inside the MUNICH256 and can be read from either the local microprocessor or (for test purposes) via the chip internal bridge from the host processor located on the PCI bus.

The interrupt FIFO triggers the  $\overline{\text{LINT}}$  pin which indicates that there is at least one interrupt vector available. Then the interrupt FIFO can be read from either PCI side or local bus side. The interrupt vector contains a last indication when there is no further interrupt vector stored in the internal interrupt FIFO.

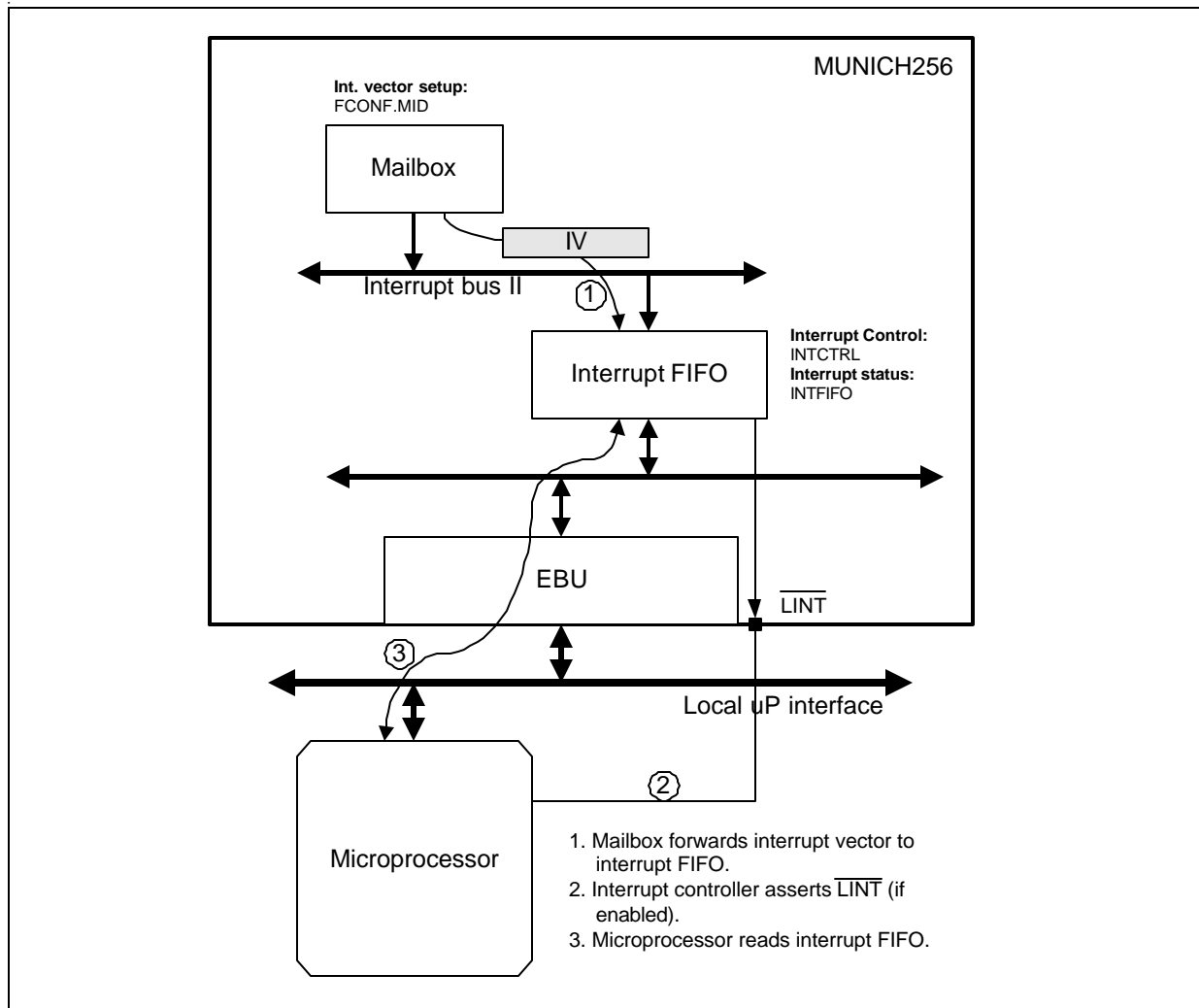
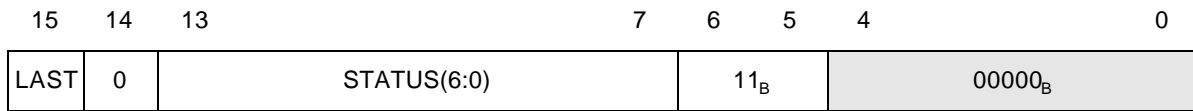


Figure 4-15 Mailbox Interrupt Notification

Functional Description



The 'Mailbox' interrupt vector is generated, in case that the host CPU on PCI side has written data to the mailbox status register MBP2E0.

- LAST**            Last indication
- LAST indicates that at least one more valid interrupt vector is stored in the internal interrupt FIFO. This bit is generated at read access time.
- 0            There is at least one more interrupt in the internal interrupt FIFO.
- 1            This interrupt is the last interrupt that is stored in the internal interrupt FIFO.
- STATUS**        Mailbox Information
- The bit field STATUS contains a copy of MBE2P0.MB(6:0).

## 5 Interface Description

### 5.1 PCI Interface

A 32-bit and 66 MHz capable PCI bus controller provides the interface between the MUNICH256 and the host system. PCI Interface pins are measured as compliant to the 3.3V signalling environment according to the PCI specification Rev. 2.1.

The PCI bus controller operates as initiator or target. Commands are supported as follows:

- Master memory read single DWORD/burst of up to 64 DWORDs with zero wait cycles.
- Master memory write single DWORD/burst of up to 64 DWORDs with zero wait cycles.
- Slave memory read single DWORD.
- Slave memory write single DWORD.

Fast back-to-back transfers are provided for slave accesses only. All read/write accesses to the MUNICH256 must be 32-bit wide, that is all bytes must be enabled. Non 32-bit accesses result in system interrupt.

Refer also to the PCI specification Rev. 2.1 for detailed information about PCI bus protocol.

#### 5.1.1 PCI Read Transaction

The transaction starts with an address phase which occurs during the first cycle when  $\overline{\text{FRAME}}$  is activated (clock 1 in **Figure 5-1**). During this phase the bus master (initiator) outputs a valid address on AD(31:0) and a valid bus command on  $\overline{\text{C/BE}}$  (3:0). The first clock of the first data phase is clock 3. During the data phase  $\overline{\text{C/BE}}$  indicate which byte lanes on AD(31: 0) are involved in the current data phase.

The first data phase on a read transaction requires a turnaround cycle. In **Figure 5-1** the address is valid on clock 2 and then the master stops driving AD. The target drives the AD lines following the turnaround when  $\overline{\text{DEVSEL}}$  is asserted. ( $\overline{\text{TRDY}}$  cannot be driven until  $\overline{\text{DEVSEL}}$  is asserted.) The earliest the target can provide valid data is clock 4. Once enabled, the AD output buffers of the target stay enabled through the end of the transaction.

A data phase may consist of a data transfer and wait cycles. A data phase completes when data is transferred, which occurs when both IRDY and TRDY are asserted. When either is deasserted a wait cycle is inserted. In the example below, data is successfully transferred on clocks 4, 6 and 8, and wait cycles are inserted on clocks 3, 5 and 7. The first data phase completes in the minimum time for a read transaction. The second data phase is extended on clock 5 because  $\overline{\text{TRDY}}$  is deasserted. The last data phase is extended because  $\overline{\text{IRDY}}$  is deasserted on clock 7. The Master knows at clock 7 that the next data phase is the last. However, the master is not ready to complete the last



## Interface Description

transfer, so  $\overline{\text{IRDY}}$  is deasserted on clock 7, and  $\overline{\text{FRAME}}$  stays asserted. Only when  $\overline{\text{IRDY}}$  is asserted can  $\overline{\text{FRAME}}$  be deasserted, which occurs on clock 8.

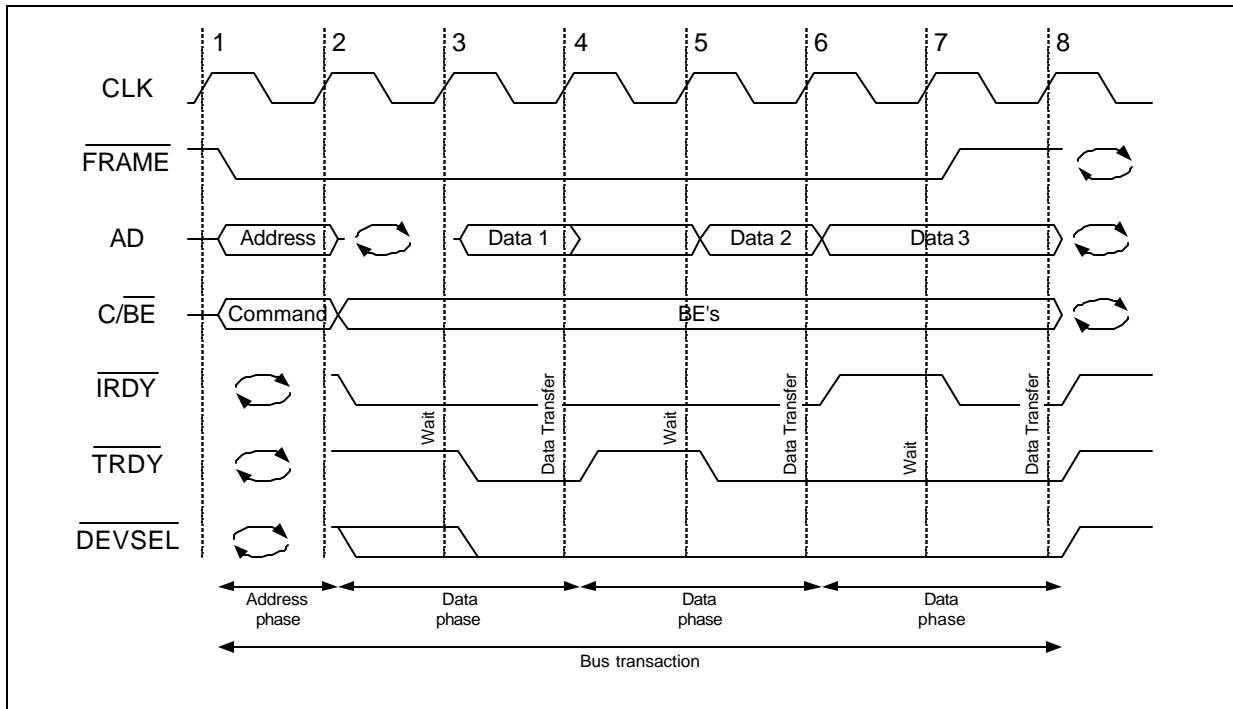


Figure 5-1 PCI Read Transaction

### 5.1.2 PCI Write Transaction

The transaction starts when  $\overline{\text{FRAME}}$  is activated (clock 1 in **Figure 5-2**). A write transaction is similar to a read transaction except no turnaround cycle is required following the address phase. In the example, the first and second data phases complete with zero wait cycles. The third data phase has three wait cycles inserted by the target. Both initiator and target insert a wait cycle on clock 5. In the case where the initiator inserts a wait cycle (clock 5), the data is held on the bus, but the byte enables are withdrawn. The last data phase is characterized by  $\overline{\text{IRDY}}$  being asserted while the  $\overline{\text{FRAME}}$  signal is deasserted. This data phase is completed when  $\overline{\text{TRDY}}$  goes active (clock 8).

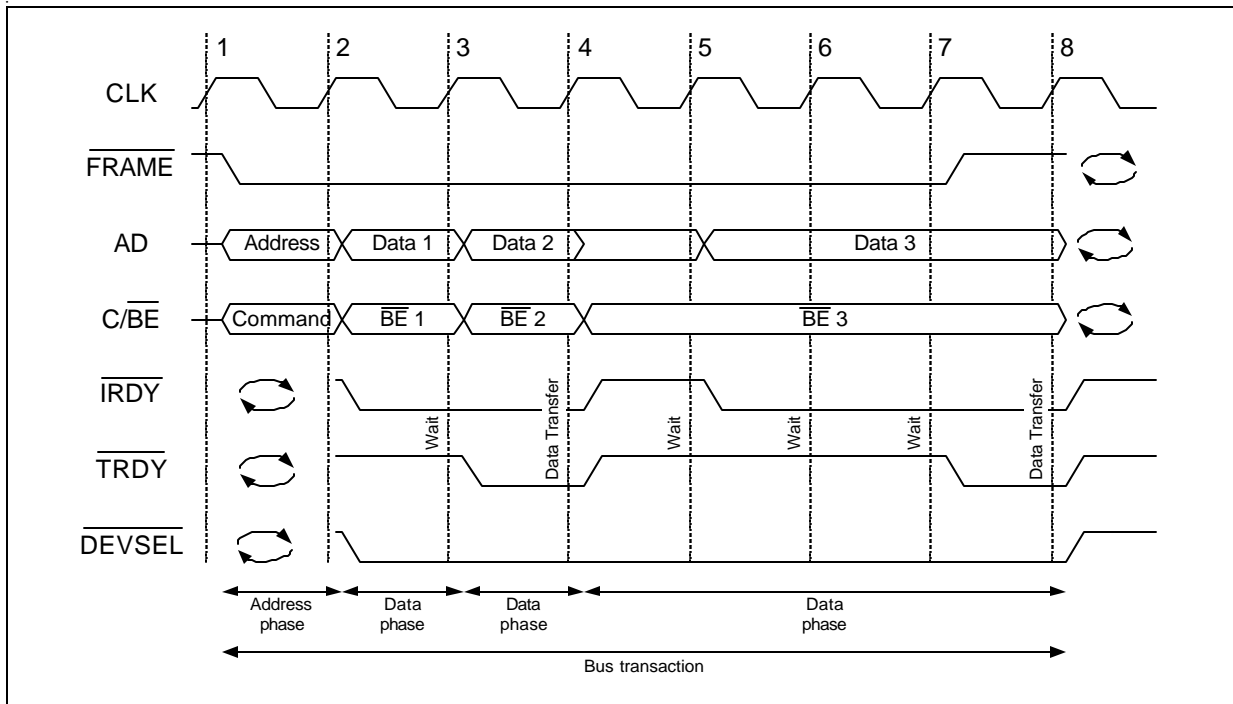


Figure 5-2 PCI Write Transaction

## 5.2 SPI Interface (ROM Load Unit)

Additional pins, which are not covered from the PCI specification, but are closely related, are the SPI pins. Via the SPI pins the vendor ID and the vendor subsystem ID can be loaded into the corresponding PCI configuration registers during start-up of the device.

The SPI Interface supports EEPROMs with an eight bit address space.

After a system reset, the MUNICH256 starts reading the first byte out of the connected EEPROM at address  $00_H$ . If this byte is equal  $AA_H$ , the device continues reading out the memory contents. Everytime four bytes are read out of the EEPROM (starting with byte address  $01_H$ ), the EEPROM interface writes the read information to the PCI configuration space. The first four bytes will be written to the PCI configuration space address  $00_H$ , the next four bytes to the PCI configuration space address  $04_H$  and so on. So the contents of the EEPROM, starting with EEPROM byte address  $01_H$ , will be mapped over the PCI configuration space after a system reset. During this configuration phase, all accesses to the PCI interface will be answered with 'retry' by the PCI interface.

If the first byte in the EEPROM is not equal  $AA_H$ , the EEPROM interface stops loading the PCI configuration space immediately, and the PCI interface can be accessed. The PCI configuration space in this case contains the default values.

The configuration mechanism through the serial interface can be disabled by pin SPLOAD. If this pin is connected to '0', the configuration mechanism is disabled. The

bridge can be accessed through the PCI Interface directly after a system reset. In this case the PCI configuration space contains the default values.

### 5.2.1 Accesses to a SPI EEPROM

The EEPROM contents can also be controlled (read and write) by the software. For this, a special EEPROM control register is implemented as part of the PCI configuration space. To start a read/write transaction to an connected EEPROM, you have to set the command, the byte address (for read-/write data commands), the data to be written and the start indication by writing to the EEPROM control register SPI in the PCI configuration space. If the interface detects SPI.START asserted (= '1'), it interprets the command and starts the read-/write transaction to the connected EEPROM. After the transaction has finished, the EEPROM control module deasserts the start bit. If the command was a read command (Read Status Register, Read Data from Memory Array), the byte that was read out of the EEPROM is available in the data register. For transactions started with the EEPROM Control register, the interface does not check if an EEPROM is connected to the SPI bus, because the EEPROM is full passive. A full functional description of the SPI commands and their usage as well as a description of the EEPROMs status register can be found in the description of the EEPROM that will be selected by a board vendor.

#### Byte Address

For read and write transaction to the connected EEPROM, the byte address must be written in this register before the transaction is started.

#### Data

For the write status register transaction and the write data to memory array transactions, the data that has to be written to the EEPROM must be written to this register before the transaction is started. After a read status register transaction or a read data from memory array transaction has finished (Bit SPI.START is deasserted), the byte received from the EEPROM is available in this register.

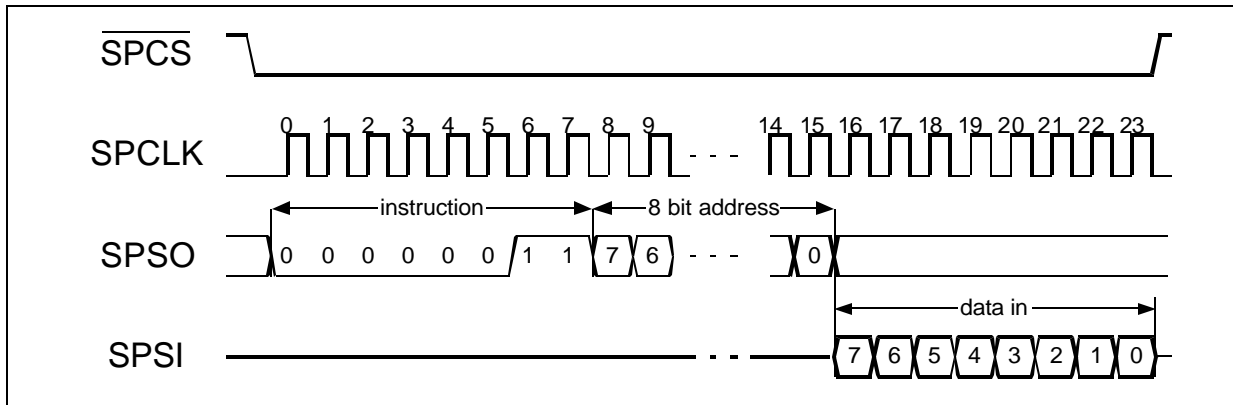
#### Start

To start the EEPROM transaction defined via register SPI the bit SPI.START must be set to '1' by a write transaction through the PCI interface. After the transaction is finished, the EEPROM start bit is deasserted by the EEPROM interface controller. This signal has to be polled by system software.

### 5.2.2 SPI Read Sequence

The MUNICH256 selects an external EEPROM by pulling  $\overline{\text{SPCS}}$  low. The eight bit read sequence is transmitted followed by the eight bit address. After the read instruction and

address is sent, the data stored in the memory at the selected address is shifted in on the SPSI pin. The read operation is terminated by setting SPCS high (see **Figure 5-3**).

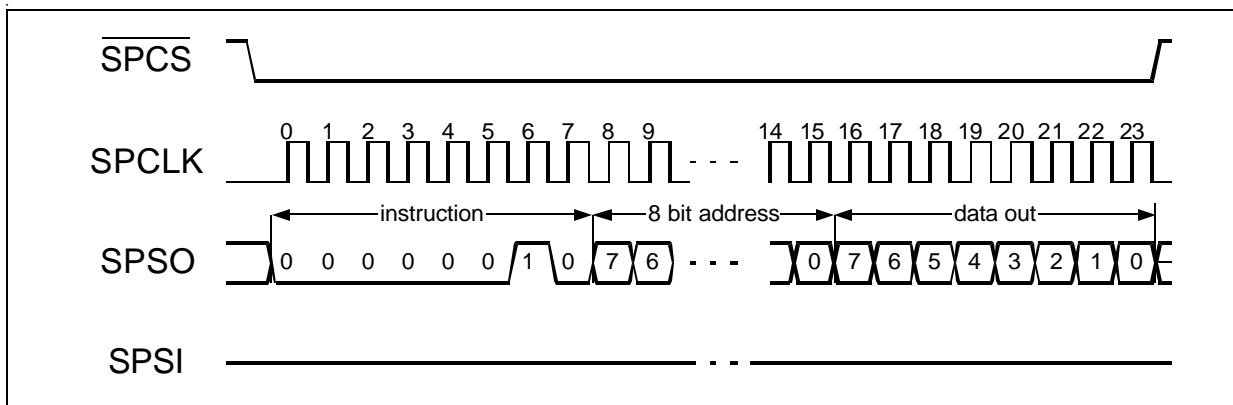


**Figure 5-3 SPI Read Sequence**

### 5.2.3 SPI Write Sequence

Prior to any attempt to write data to an external EEPROM, the write enable latch must be set by issuing the WREN instruction. This is done by setting  $\overline{\text{SPCS}}$  low and then clocking out the WREN instruction. After all eight bits of the instruction are transmitted, the SPCS will be brought high to set the write enable latch.

Once the write enable latch is set, the user may proceed by issuing a write instruction, followed by the eight bit address and then the data to be written. In order that data will actually be written to the EEPROM, the SPCS is set high after the least significant bit (D0) of the data byte has been clocked in. Refer to **Figure 5-4** for detailed illustrations on the byte write sequence. While the write is in progress, the register bit SPI.START may be read to check the status of the transaction. When a write cycle is completed, the register bit SPI.START is reset.



**Figure 5-4 SPI Write Sequence**

### 5.3 Local Microprocessor Interface

The Local Microprocessor Interface is a demultiplexed switchable Intel or Motorola style interface with master and slave functionality. The MUNICH256 provides a local clock output LCLK, which is a feed through of the PCI system clock as clock reference for the local microprocessor interface. The local bus master capability allows to access peripherals located on the local bus via the PCI interface. Bit FCONF.LME enables the bus master capability.

The base address register two is disabled per default and can be enabled during start-up of the internal PCI interface. This is done by setting bit MEM.BAR2 in the PCI configuration space.

The MUNICH256 supports a maximum of three 8 kByte pages of memory on the local address bus. The correspondence between the accessed PCI memory space (mapped via base address register 2) and the asserted chip selects is shown in table 5-1. The mapping of the PCI byte enables to the local bus address is dependent on the selected bus mode and is explained in detail in the corresponding section.

**Table 5-1 Correspondence between PCI memory space and chip select**

Page	AD(14:0)	LCS2	LCS1
0	0000 <sub>H</sub> - 1FFF <sub>H</sub>	1	0
1	2000 <sub>H</sub> - 3FFF <sub>H</sub>	0	1
2	4000 <sub>H</sub> - 5FFF <sub>H</sub>	0	0
3	6000 <sub>H</sub> - 7FFF <sub>H</sub>	Not valid	

### 5.3.1 Intel Mode

#### 5.3.1.1 Slave Mode

In Intel slave mode the bus interface supports 16-bit transactions in demultiplexed bus operation. It uses the local bus port pins LA(12:1) for the 16 bit address and the local bus port pins LD(15:0) for 16 bit data. A read/write access is initiated by placing an address on the address bus and asserting  $\overline{\text{LCS0}}$  (**Figure 5-5**). The external processor then activates the respective command signal ( $\overline{\text{LRD}}$ ,  $\overline{\text{LWR}}$ ). Data is driven onto the data bus either by the MUNICH256 (for read cycles) or by the external processor (for write cycles). After a period of time, which is determined by the access time to the internal registers valid data is placed on the bus, which is indicated by asserting the active low signal  $\overline{\text{LRDY}}$ .

*Note:  $\overline{\text{LCS0}}$  need not be deasserted between two subsequent cycles to the same device.*

#### Read cycles

Input data can be latched and the command signal can be deactivated now. This causes the MUNICH256 to remove its data from the data bus which is then tri-stated again.  $\overline{\text{LRDY}}$  is driven high and will be tri-stated as soon as  $\overline{\text{LCS0}}$  is deasserted.

#### Write cycles

The command signal can be deactivated now. If a subsequent bus cycle is required, the external processor can place the respective address on the address bus.

#### 5.3.1.2 Master Mode

A read/write access from the PCI bus to the 16 bit demultiplexed local bus is initiated by accessing the PCI memory space base which is controlled by the base address register 2. Each valid read or write access to this base address triggers the local bus master interface which in turn starts arbitration for the local bus by asserting LHOLD (see (1) in **Figure 5-6**). As soon as the MUNICH256 gets access to the local bus (LHLDA asserted) it starts the local bus latency timer and begins a read/write transaction as the bus master. The signal LHOLD remains asserted while a transaction is in progress or as long as the local bus latency timer is not expired. A read/write transaction begins when the MUNICH256 places a valid address on the address bus, sets the  $\overline{\text{LBHE}}$  signal which indicates a 8- or 16-bit bus access and asserts the chip select signals  $\overline{\text{LCS1}}$  and/or  $\overline{\text{LCS2}}$ . Then the MUNICH256 activates the respective command signals ( $\overline{\text{LRD}}$ ,  $\overline{\text{LWR}}$ ). Data is driven onto the data bus either by the MUNICH256 (for write cycles) or by the accessed device (for read cycles).

A transaction is finished on the local bus when the external device asserts  $\overline{\text{LRDY}}$  (ready controlled bus cycles) or when the internal wait state timer expires.

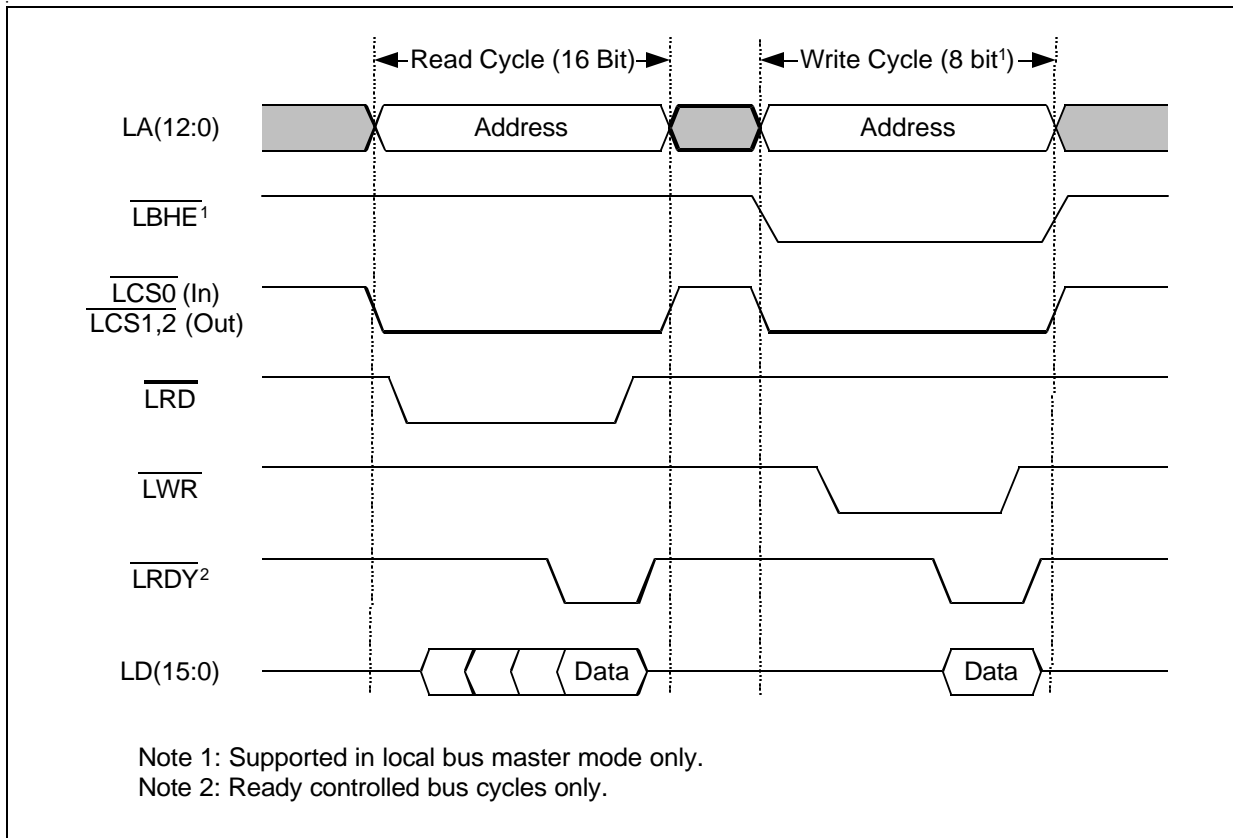


Figure 5-5 Intel Bus Mode

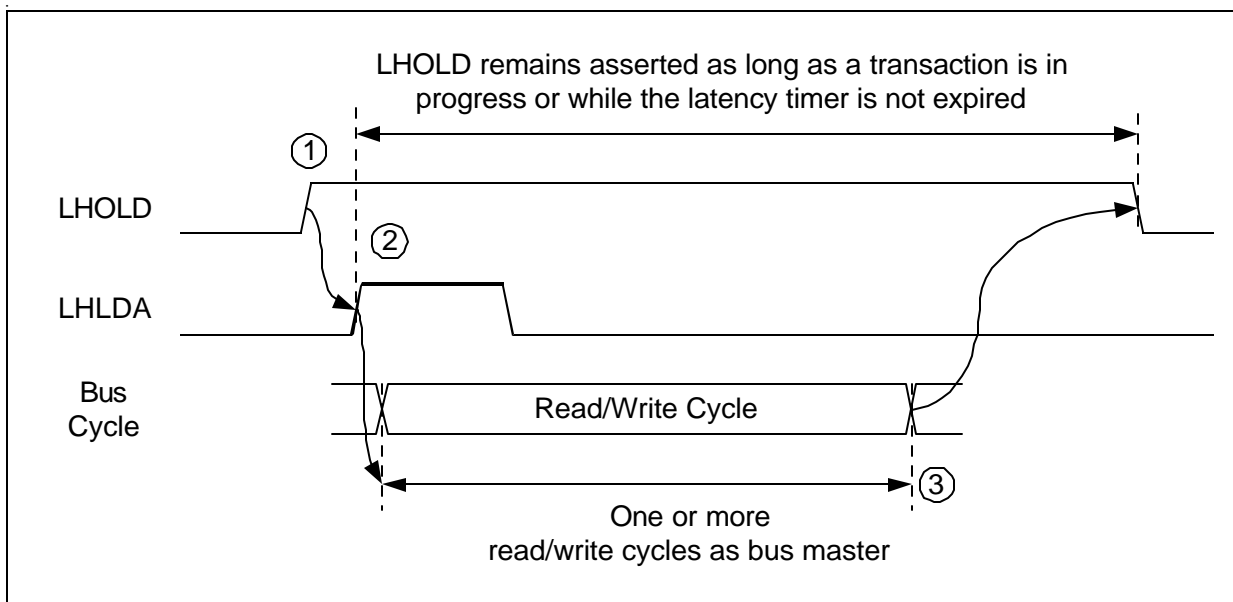


Figure 5-6 Intel Bus Arbitration

Valid C/ $\overline{\text{BE}}$  combinations and the correspondence between local address,  $\overline{\text{LBHE}}$  and the mapping of PCI data to the local data bus are shown in table 5-2 and table 5-3. All

Interface Description

accesses not shown in the table result in generation of a 'PCI Access Error' interrupt vector.

**Table 5-2**  $\overline{C/BE}$  to  $\overline{LA/LBHE}$  mapping in Intel bus mode (8 bit port mode)

$\overline{C/BE}(3:0)$	$\overline{LA}(1:0)$	$\overline{LBHE}$	$\overline{LD}(15:8)$	$\overline{LD}(7:0)$
1110 <sub>B</sub>	00 <sub>B</sub>	1	-	AD(7:0)
1101 <sub>B</sub>	01 <sub>B</sub>	1	-	AD(15:8)
1011 <sub>B</sub>	10 <sub>B</sub>	1	-	AD(23:16)
0111 <sub>B</sub>	11 <sub>B</sub>	1	-	AD(31:24)

**Table 5-3**  $\overline{C/BE}$  to  $\overline{LA/LBHE}$  mapping in Intel bus mode (16 bit port mode)

$\overline{C/BE}(3:0)$	$\overline{LA}(1:0)$	$\overline{LBHE}$	$\overline{LD}(15:8)$	$\overline{LD}(7:0)$
1110 <sub>B</sub>	00 <sub>B</sub>	1	-	AD(7:0)
1101 <sub>B</sub>	01 <sub>B</sub>	0	AD(15:8)	-
1011 <sub>B</sub>	10 <sub>B</sub>	1	-	AD(23:16)
0111 <sub>B</sub>	11 <sub>B</sub>	0	AD(31:24)	-
1100 <sub>B</sub>	00 <sub>B</sub>	0	AD(15:8)	AD(7:0)
0011 <sub>B</sub>	10 <sub>B</sub>	0	AD(31:24)	AD(23:16)



## 5.3.2 Motorola Mode

### 5.3.2.1 Slave Mode

The demultiplexed bus modes use the local bus port pins LA(12:1) for the 16-bit address and the local bus port pins LD(15:0) for 16-bit data. A read/write access is initiated by placing an address on the address bus and asserting  $\overline{\text{LCS0}}$  together with the command signal  $\overline{\text{LWRRD}}$  (see **“Motorola Bus Mode” on Page 106**). The data cycle begins when the signal  $\overline{\text{LDS}}$  is asserted. Data is driven onto the data bus either by the MUNICH256 (for read cycles) or by the external processor (for write cycles). After a period of time, which is determined by the access time to the internal registers, valid data is placed on the bus, which is indicated by asserting the active low signal  $\overline{\text{LDTACK}}$ .

*Note:  $\overline{\text{LCS0}}$  need not be deasserted between two subsequent cycles to the same device.*

#### Read cycles

Input data can be latched and the data strobe signal can be deactivated now. This causes the MUNICH256 to remove its data from the data bus which is then tri-stated again.  $\overline{\text{LDTACK}}$  is driven high and will be tri-stated as soon as  $\overline{\text{LCS0}}$  is deasserted.

#### Write cycles

The data strobe signal can be deactivated now. If a subsequent bus cycle is required, the external processor can place the respective address on the address bus.

### 5.3.2.2 Master Mode

As in Intel mode a read/write access from the PCI bus to the 16-bit demultiplexed local bus is initiated by accessing the PCI memory space base mapped by the base address register 2. Each valid read or write access to this base address triggers the local bus master interface which in turn starts arbitration for the local bus using the interface signals  $\overline{\text{LBR}}$  and  $\overline{\text{LBG}}$  and  $\overline{\text{LBGACK}}$ . As soon as the MUNICH256 gets access to the local bus it places a valid address on the address bus, sets the  $\overline{\text{LSIZE0}}$  signal which indicates a 8- or 16-bit bus access and asserts the corresponding chip select signal. The signal  $\overline{\text{LWRRD}}$  indicates a read or write operation. The data cycle begins when the signal  $\overline{\text{LDS}}$  is asserted. Data is driven onto the data bus either by the MUNICH256 or by the external component.

A transaction is finished on the local bus when the external device asserts the active low signal  $\overline{\text{LDTACK}}$  or when the internal wait state timer expires.

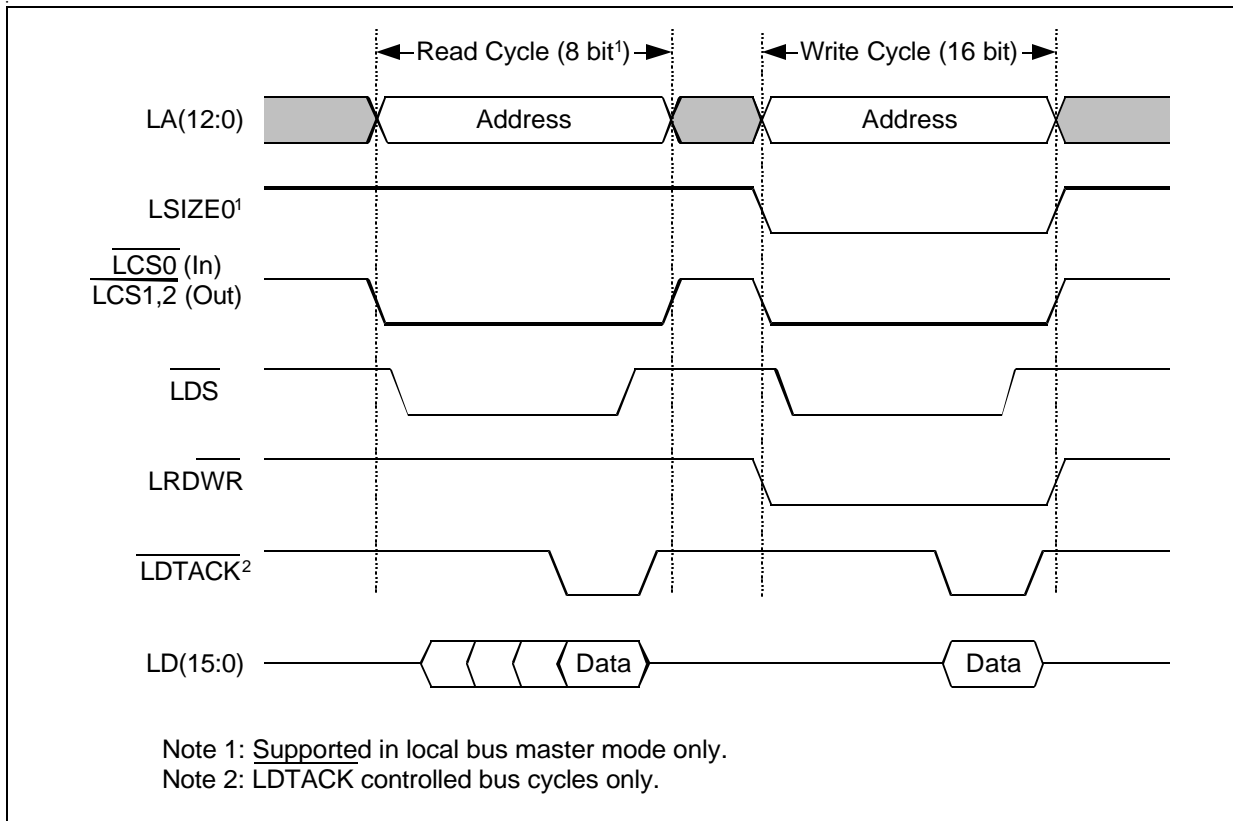


Figure 5-7 Motorola Bus Mode

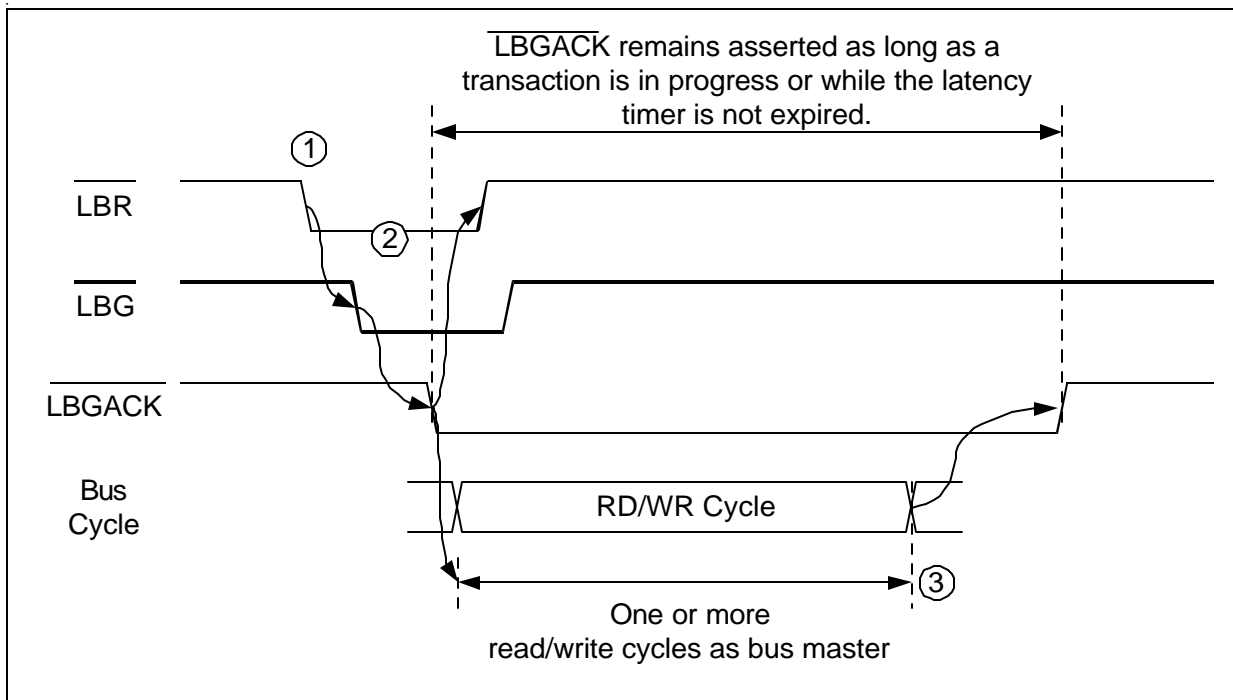


Figure 5-8 Motorola Bus Arbitration

Interface Description

The address and byte enable signals on the PCI bus are mapped to the local bus according to table 5-4 and table 5-5. It can be seen that the MUNICH256 supports different valid  $C/\overline{BE}$  combinations which result in either a 8- or 16-bit access to the local bus interface. All accesses not shown in the table result in generation of a 'PCI Access Error' interrupt vector. Byte swapping for 16 bit data transfers can be disabled.

**Table 5-4**  $\overline{C/\overline{BE}}$  to LA/LSIZE0 mapping in Motorola bus mode (8 bit port mode)

$C/\overline{BE}(3:0)$	LA(1:0)	LSIZE0	LD(15:8)	LD(7:0)
1110 <sub>B</sub>	00 <sub>B</sub>	1	AD(7:0)	-
1101 <sub>B</sub>	01 <sub>B</sub>	1	AD(15:8)	-
1011 <sub>B</sub>	10 <sub>B</sub>	1	AD(23:16)	-
0111 <sub>B</sub>	11 <sub>B</sub>	1	AD(31:24)	-

**Table 5-5**  $\overline{C/\overline{BE}}$  to LA/LSIZE0 mapping in Motorola bus mode (16 bit port mode)

$C/\overline{BE}(3:0)$	LA(1:0)	LSIZE0	LD(15:8)	LD(7:0)
1110 <sub>B</sub>	00 <sub>B</sub>	1	AD(7:0)	
1101 <sub>B</sub>	01 <sub>B</sub>	1	-	AD(15:8)
1011 <sub>B</sub>	10 <sub>B</sub>	1	AD(23:16)	-
0111 <sub>B</sub>	11 <sub>B</sub>	1	-	AD(31:24)
1100 <sub>B</sub>	00 <sub>B</sub>	0	AD(7:0)	AD(15:8)
0011 <sub>B</sub>	10 <sub>B</sub>	0	AD(23:16)	AD(31:24)

## 5.4 Serial Line Interface

The serial interface of the interface can be configured in a 16-port mode and additionally in a 28-port mode. Dependent on the port configuration (16-port mode or 28-port mode) the MUNICH256 supports T1, E1, channelized 4.096 MHz, channelized 8.192 MHz or unchannelized frame structures (**Figure 5-9**).

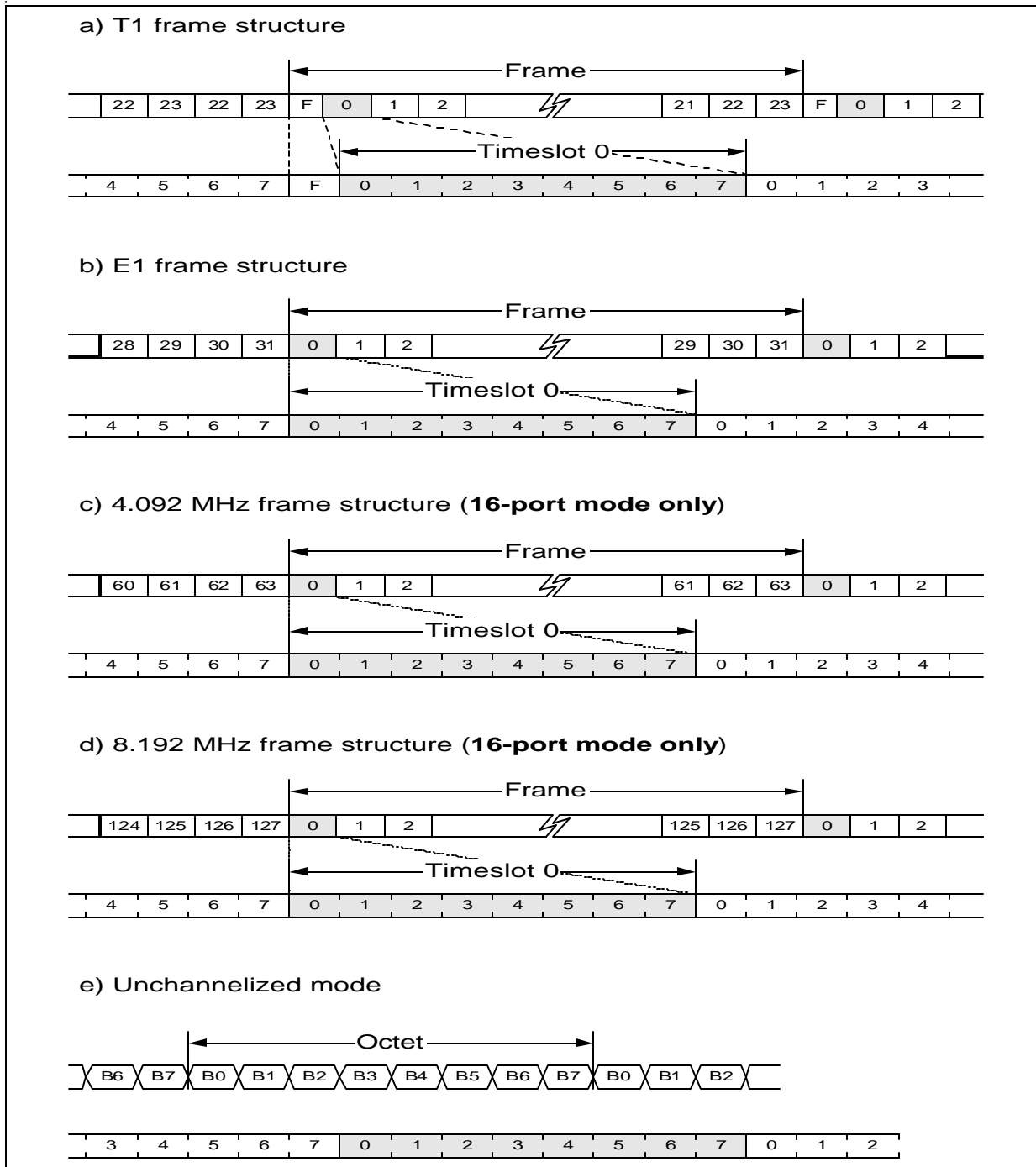


Figure 5-9 Supported Frame Structures

### 5.4.1 Interface Timing in 16-port mode

In 16-port mode each receive port has a receive data input RD(x), a receive synchronization input RSP(x) and the corresponding receive clock input RCLK(x). In transmit direction each port consists of the transmit data output TD(x), the transmit

Interface Description

synchronization input TSP(x) and the transmit clock input TCLK(x). In channelized mode (T1, E1, 4.096 MHz and 8.192 MHz) the high level after a low to high transition of the frame synchronization pulse marks the last bit of a frame (default value in transmit direction) respectively the first bit of a frame (default value in receive direction). The active edge of the frame synchronization pulse can be shifted in a range of -4 to +3.

RD(x), RSP(x) and TSP(x) can be sampled on the rising or falling edge of the receive clock respectively the transmit clock. Outgoing data is updated on the rising or falling edge of TCLK(x).

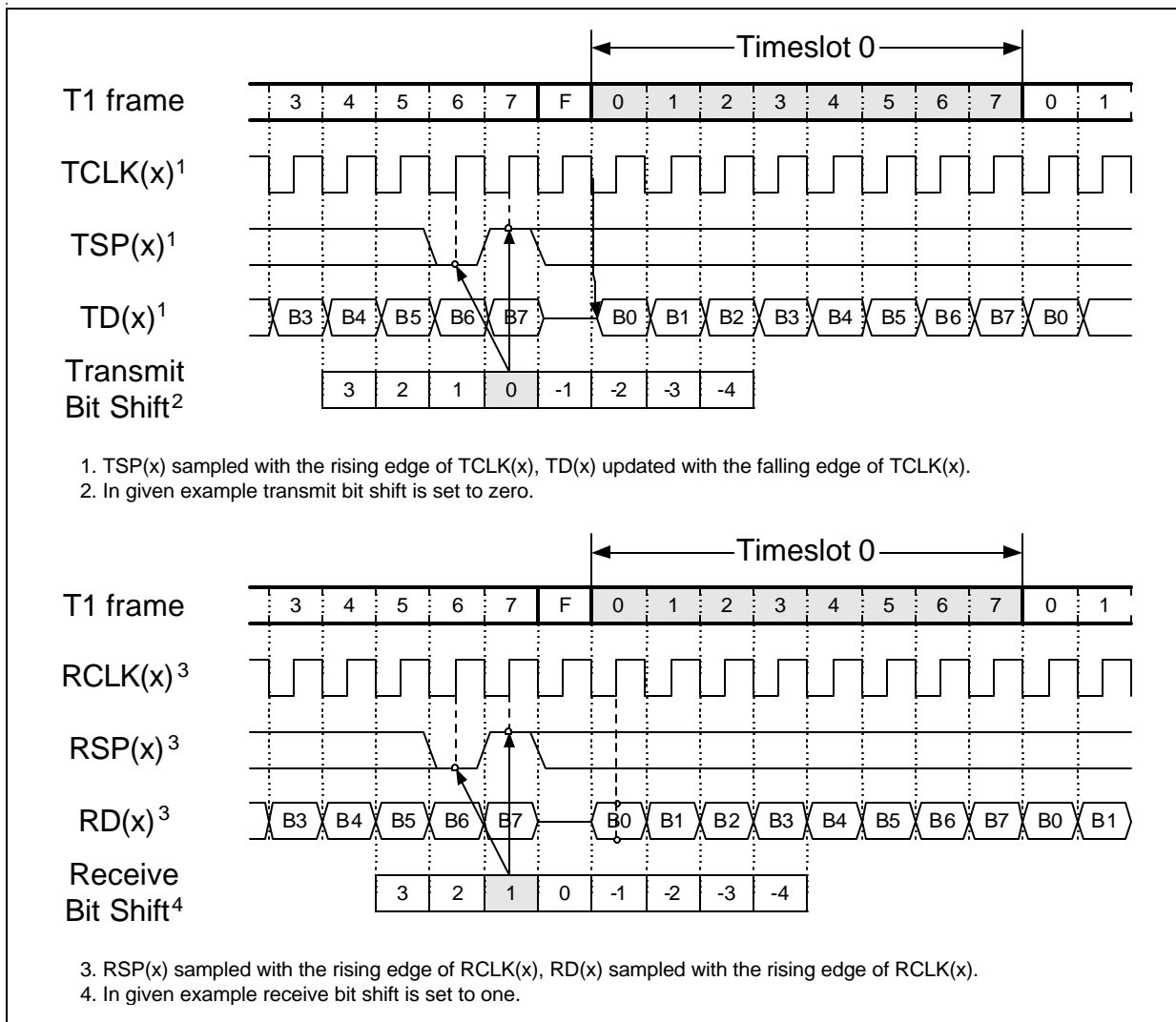


Figure 5-10 T1 Mode Frame Timing

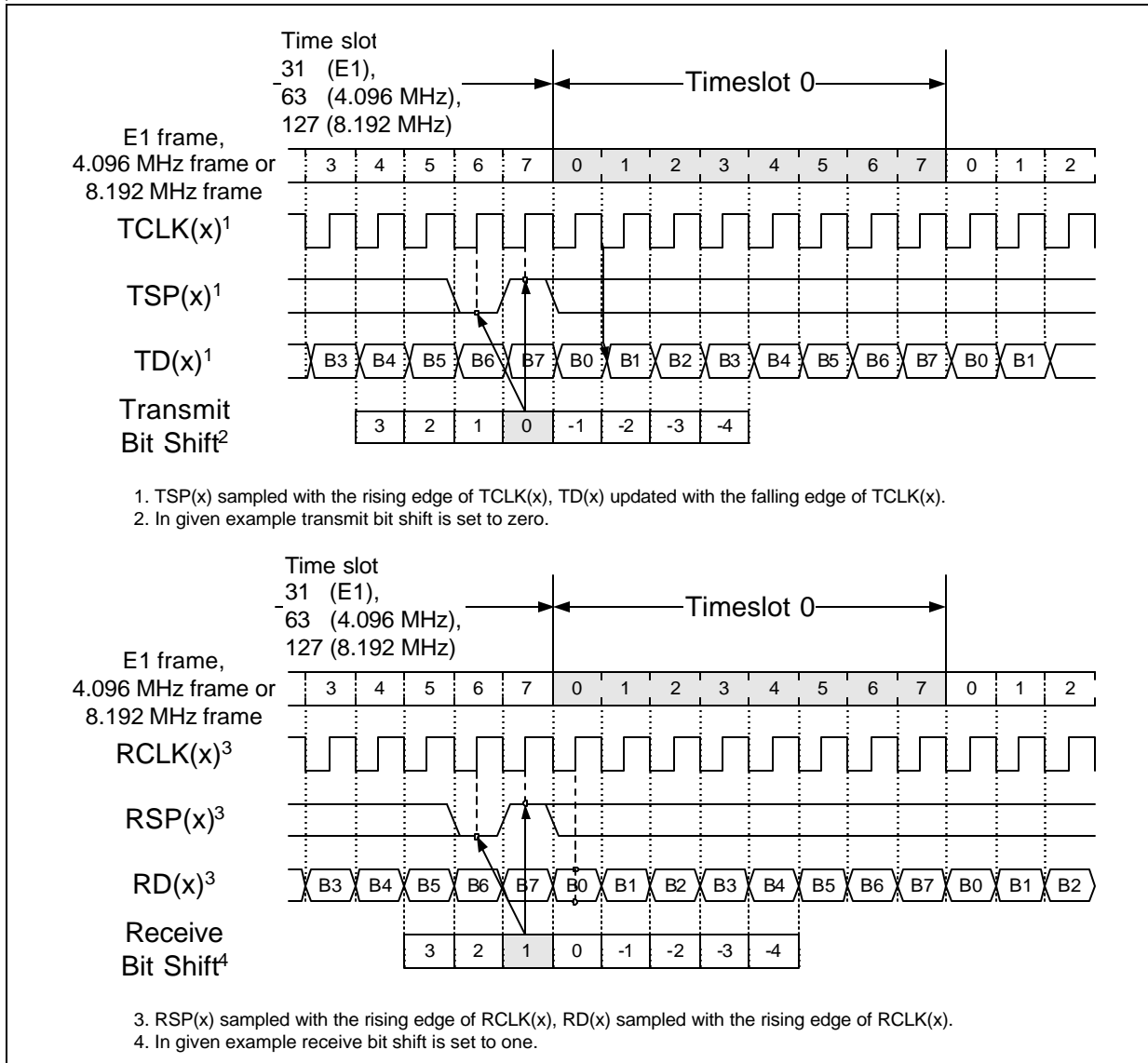


Figure 5-11 E1, 4.096 MHz and 8.192 MHz Interface Timing in 16-port mode

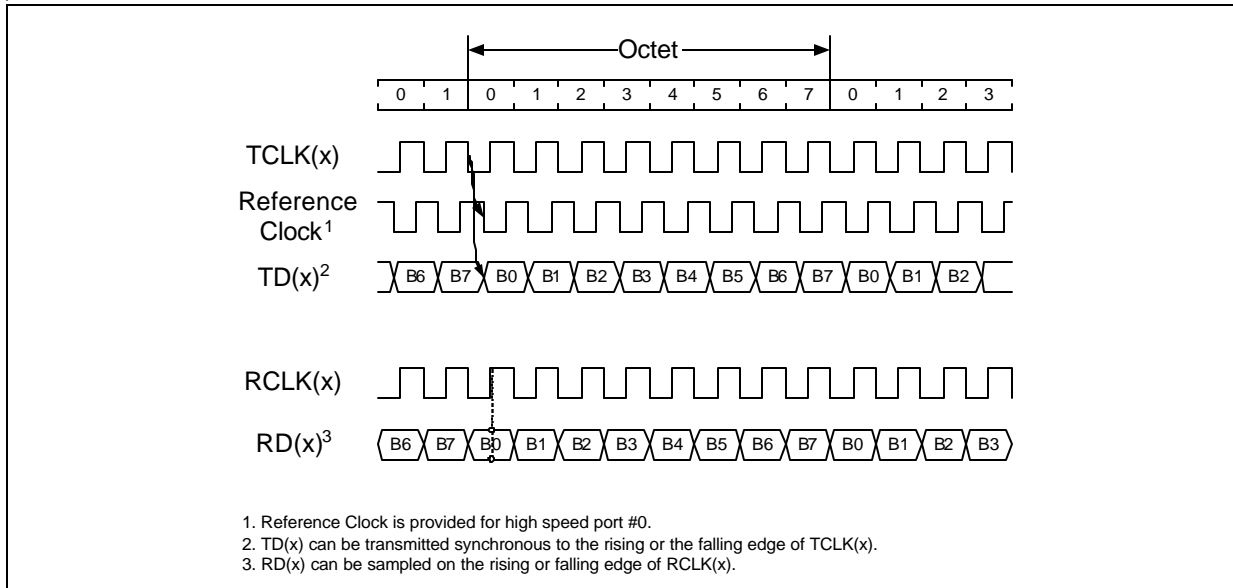


Figure 5-12 Unchannelized Mode Interface Timing

### 5.4.2 Interface Timing in 28-port mode

The MUNICH256 in 28-port mode supports T1, E1 and unchannelized frame structures on the serial side. Each receive port has a receive data input RD(x) and the corresponding receive clock input RCLK(x). In transmit direction each port consists of the transmit data output TD(x) and the transmit clock input TCLK(x). In T1 and E1 mode a clock gap marks the beginning of a frame. The timing of the unchannelized mode is identical to the 16-port mode.

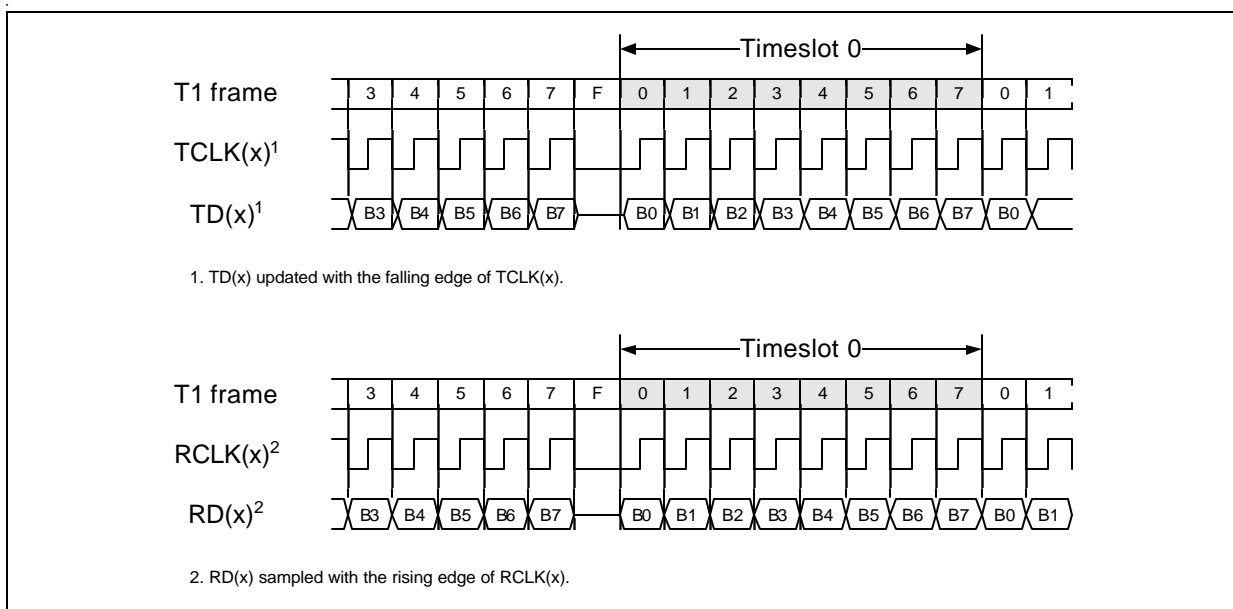


Figure 5-13 T1-mode Interface Timing in 28-port Mode

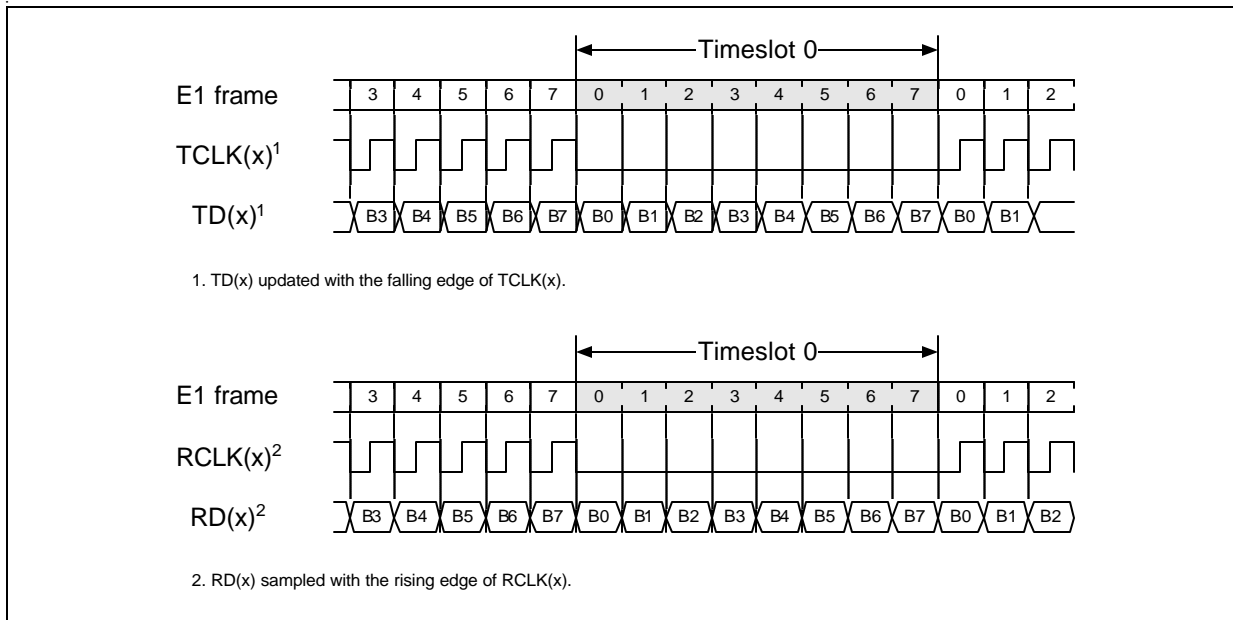
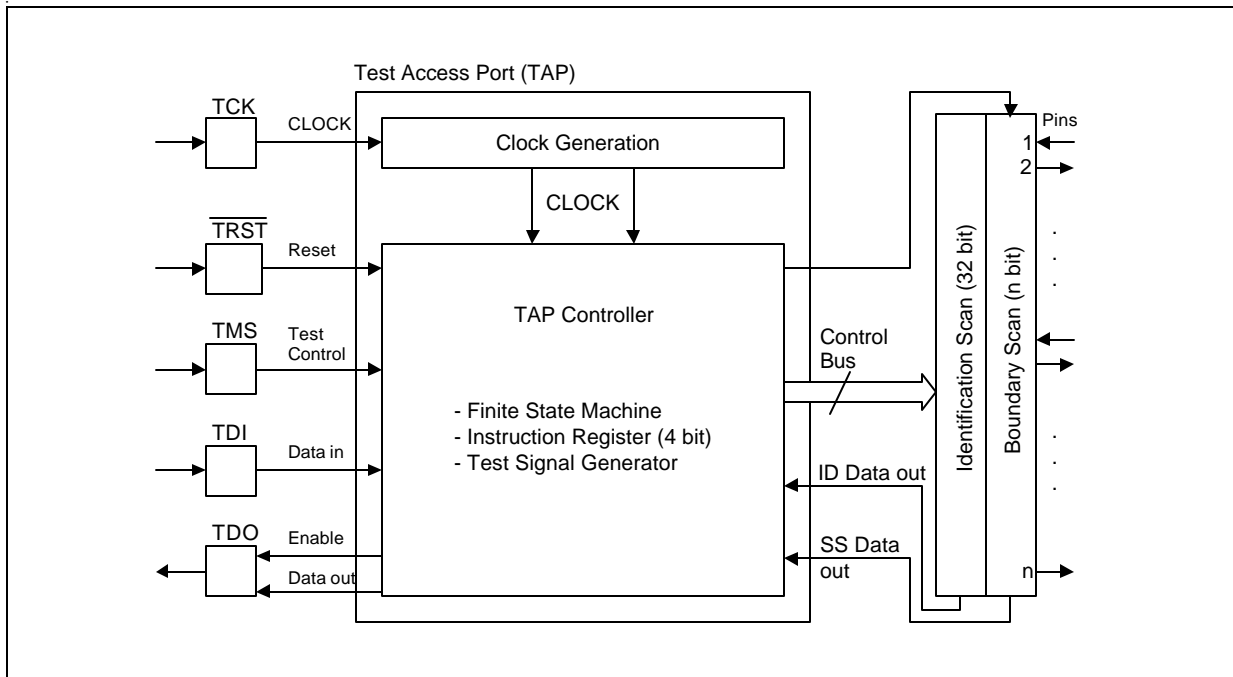


Figure 5-14 E1-mode Interface Timing in 28-port Mode

## 5.5 JTAG Interface

A test access port (TAP) is implemented in the MUNICH256. The essential part of the TAP is a finite state machine (16 states) controlling the different operational modes of the boundary scan. Both, TAP controller and boundary scan, meet the requirements given by the JTAG standard: IEEE 1149.1. **Figure 5-15** gives an overview about the TAP controller.





**Figure 5-15 Block Diagram of Test Access Port and Boundary Scan Unit**

If no boundary scan operation is planned  $\overline{\text{TRST}}$  has to be connected with  $V_{SS}$ . TMS and TDI do not need to be connected since pull-up transistors ensure high input levels in this case. Nevertheless it would be a good practice to put the unused inputs to defined levels. In this case, if the JTAG is not used:  
TMS = TCK = '1' is recommended.

Test handling (boundary scan operation) is performed via the pins TCK (Test Clock), TMS (Test Mode Select), TDI (Test Data Input) and TDO (Test Data Output) when the TAP controller is not in its reset state, i. e.  $\overline{\text{TRST}}$  is connected to  $V_{DD3}$  or it remains unconnected due to its internal pull up. Test data at TDI are loaded with a clock signal connected to TCK. '1' or '0' on TMS causes a transition from one controller state to another; constant '1' on TMS leads to normal operation of the chip.

An input pin (I) uses one boundary scan cell (data in), an output pin (O) uses two cells (data out, enable) and an I/O-pin (I/O) uses three cells (data in, data out, enable). Note that most functional output and input pins of the MUNICH256 are tested as I/O pins in boundary scan, hence using three cells. The boundary scan unit of the MUNICH256 contains a total of  $n = 484$  scan cells. The desired test mode is selected by serially loading a 4-bit instruction code into the instruction register via TDI (LSB first).

**EXTEST** is used to examine the interconnection of the devices on the board. In this test mode at first all input pins capture the current level on the corresponding external interconnection line, whereas all output pins are held at constant values ('0' or '1'). Then the contents of the boundary scan is shifted to TDO. At the same time the next scan vector is loaded from TDI. Subsequently all output pins are updated according to the new

---

**Interface Description**

boundary scan contents and all input pins again capture the current external level afterwards, and so on.

**INTEST** supports internal testing of the chip, i. e. the output pins capture the current level on the corresponding internal line whereas all input pins are held on constant values ('0' or '1'). The resulting boundary scan vector is shifted to TDO. The next test vector is serially loaded via TDI. Then all input pins are updated for the following test cycle.

**SAMPLE/PRELOAD** is a test mode which provides a snapshot of pin levels during normal operation.

**IDCODE**: A 32-bit identification register is serially read out via TDO. It contains the version number (4 bits), the device code (16 bits) and the manufacturer code (11 bits). The LSB is fixed to '1'.

The ID code field is set to

Version : 2<sub>H</sub>

Part Number : 005B<sub>H</sub>

Manufacturer : 083<sub>H</sub> (including LSB, which is fixed to '1')

*Note: Since in test logic reset state the code '0011' is automatically loaded into the instruction register, the ID code can easily be read out in shift DR state.*

**BYPASS**: A bit entering TDI is shifted to TDO after one TCK clock cycle.

**CLAMP** allows the state of signals driven from component pins to be determined from the boundary-scan register while the bypass register is selected as the serial path between TDI and TDO. Signals driven from the MUNICH256 will not change while the CLAMP instruction is selected.

**HIGHZ** places all of the system outputs in an inactive drive state.

## 6 Channel Programming / Reprogramming Concept

For channel programming the MUNICH256 provides a on-chip channel specification data structure. All information necessary to setup a channel has to be provided using this data structure. As soon as all channel information has been written to the channel specification registers the information can be released using simple channel commands, which have to be written to register CSPEC\_CMD. The relevant channel information will then be copied to the chip internal channel database. The channel specification registers, which need to be programmed before a command can be executed, are shown in **Table 6-1**.

Before initializing a channel the time slot assignment process for the affected channel must be completed. Vice versa after shutting down a channel the time slots associated with the affected channel should be set to inhibit. Otherwise if a time slot is reprogrammed afterwards, strange behavior can be expected on the serial side.

For each channel a simple sequence of channel commands must be ensured. After reset each channel is in its 'off' state. Therefore, the first command to start a channel is 'Transmit Init' or 'Receive Init'. This brings the channel into the operational state. In this state all commands except 'Transmit Init', 'Receive Init' or 'Transmit Idle' can be given. To bring a channel back into the idle state a 'Transmit Off' or 'Receive Off' command has to be programmed. For certain channel commands system software has to wait before new commands can be given for the same channel. This is due to internal buffer allocation functions which require some processing time. Notification of system software is done in form of command interrupt vectors, which signal that a command has successful or even unsuccessful completed.

**Table 6-1 Channel Specification Registers and Channel Commands**

Register	Transmit Commands							Receive Commands				
	Transmit Init	Transmit Off	Transmit Abort/Branch	Transmit Hold Reset	Transmit Idle	Transmit Debug	Transmit Update FNUM	Receive Init	Receive Off	Receive Abort/Branch	Receive Hold Reset	Receive Debug
CSPEC_MODE_REC												
CSPEC_REC_ACCM												
CSPEC_MODE_XMIT												

Channel Programming / Reprogramming Concept

Register	Transmit Commands							Receive Commands				
	Transmit Init	Transmit Off	Transmit Abort/Branch	Transmit Hold Reset	Transmit Idle	Transmit Debug	Transmit Update FNUM	Receive Init	Receive Off	Receive Abort/Branch	Receive Hold Reset	Receive Debug
CSPEC_XMIT_ACCM												
CSPEC_BUFFER												
CSPEC_FRDA												
CSPEC_FTDA												
CSPEC_IMASK												

## 6.1 Channel Commands

The following section describes all receive and transmit channel commands and the programming sequence in details.

## 6.2 Transmit Channel Commands

### Transmit Init

Before a 'Transmit Init' command is given, the MUNICH256 will not transmit data for a channel. After the 'Transmit Init' command the channel database of the affected channel is initialized according to the parameters in the channel specification registers.

After initialization the transmit buffer prepares the buffer locations for the selected channel and the data management unit starts processing the linked list and fills the prepared buffer locations. In order to prevent a transmit underrun condition, the transmit buffer is filled up to the transmit forward threshold before data is sent to the serial side. The protocol machine formats data according to the given channel parameters and the data is placed in the time slots assigned to the selected channel. When no or not sufficient data is available, the device sends the idle code according the selected protocol mode.

If the command was successful, a 'Transmit Command Complete' interrupt vector is generated after the first transmit descriptor is read pointed to by register CSPEC\_FTDA. In case that there is insufficient transmit buffer space, the command cannot be

---

## Channel Programming / Reprogramming Concept

completed internally and the device responds with a 'Transmit Command Failed' interrupt vector. Furthermore the MUNICH256 will not start processing the linked list for this particular channel.

New commands for the same channel may be given after the user received the 'Transmit Command Complete' interrupt vector. Prior to new initialization of the same channel it must be turned off using the 'Transmit Off' command.

### Transmit Off

After 'Transmit Off' the transmit channel is disabled immediately and the time slots assigned to the selected channel are tri-stated. The transmit buffer releases all buffer locations assigned to the channel. The data management unit updates the last processed descriptor with the complete bit if enabled and generates a 'Transmit Host Initiated' interrupt vector if the THI bit in the last descriptor was set. All channel related informations are cleared from the internal channel database.

A 'Transmit Command Complete' interrupt vector is generated when the channel command is finished. After that time processing of the linked list is completely stopped. New commands for the same channel may be given after the user received the 'Transmit Command Complete' interrupt vector.

### Transmit Abort/Branch

The 'Transmit Abort/Branch' command is performed on the serial side and in the data management unit. The data management unit stops immediately processing the current descriptor and branches to a new descriptor pointed to by CSPEC\_FTDA. Data which is already stored in the transmit buffer is sent on the serial side. The protocol machine will append an abort sequence if data in transmit buffer was not complete due to 'Transmit Abort/Branch' command. System software is informed about the aborted frame by a 'Transmit Abort' channel interrupt vector. If no data is stored in the transmit buffer this command does not affect the serial side and no 'Transmit Abort' interrupt vector is generated. Data transmission is continued with a new frame when the data management unit branched to the new descriptor list.

A 'Transmit Command Complete' interrupt vector is generated after the management unit released the old descriptor list. New commands for the same channel may be given after the user received the 'Transmit Command Complete' interrupt vector.

### Transmit Hold Reset

The 'Transmit Hold Reset' command must be given after system software has set the HOLD bit of a descriptor from '1' to '0'. In case that the MUNICH256 is in hold condition it reads the descriptor which had its HOLD bit set and tests the HOLD bit of the descriptor. If the HOLD bit is set to '0' the data management unit branches to the next descriptor and continues data transmission. Otherwise the particular channel remains in hold condition.

---

## Channel Programming / Reprogramming Concept

The MUNICH256 will NOT generate a 'Transmit Command Complete' interrupt vector after this command is programmed.

### Transmit Update FNUM

The 'Transmit Update FNUM' command changes the parameter CSPEC\_MODE\_XMIT.FNUM in the internal channel database, which allows to change dynamically the number of idle flags that are inserted between two frames.

The MUNICH256 will NOT generate a 'Transmit Command Complete' interrupt vector after this command is programmed.

### Transmit Idle

The 'Transmit Idle' command starts the MUNICH256 to send the value CSPEC\_MODE\_XMIT.TFLAG in the time slots of the selected channel. This command can only be given if a channel is turned off.

The MUNICH256 will NOT generate a 'Transmit Command Complete' interrupt vector after this command is programmed.

### Transmit Debug

The 'Transmit Debug' command allows to read back the current settings of the internal channel database. After the 'Transmit Debug' command has been programmed system software can read back the current values of the channel specification registers. Register CSPEC\_FTDA contains the value of the next transmit descriptor.

The MUNICH256 will NOT generate a 'Transmit Command Complete' interrupt vector after this command is programmed.

*Note: The setting of the internal channel database is not copied into the channel specification registers and therefore the values read can not be used to program another channel. After system software has used the 'Transmit Debug' command it must reprogram the channel specification registers to setup a new channel.*

## 6.3 Receive Channel Commands

### Receive Init

Before a 'Receive Init' command is given, the MUNICH256 will not process data for a channel. After the 'Receive Init' command the channel database of the affected channel is initialized according to the parameters programmed in channel specification registers.

After initialization data received in those time slots assigned to the selected channel is processed and stored in the internal receive buffer. The data management unit starts storing this data in the linked list which starts at CSPEC\_FRDA. The protocol machine deformats and checks data according to the given channel parameters.

---

## Channel Programming / Reprogramming Concept

A 'Receive Command Complete' interrupt vector is generated after the channel information is copied into the internal channel database.

New commands for the same channel may be given after the MUNICH256 issued the 'Receive Command Complete' interrupt vector. Prior to new initialization of the same channel it must be turned off using the 'Receive Off' command.

### Receive Off

The 'Receive Off' command disables the receive channel immediately. Further incoming data is discarded until the next 'Receive Init' command is given. Data already stored in the receive buffer is written to system memory. If a frame is destroyed by the 'Receive Off' command a 'Receive Abort' channel interrupt vector is generated.

A 'Receive Command Complete' interrupt vector is generated after remaining data in the receive buffer is written to system memory. After that time processing of the linked list is stopped and the channel information is cleared from the internal channel database.

New commands for the same channel may be given after the MUNICH256 issued the 'Receive Command Complete' interrupt vector.

### Receive Abort/Branch

The 'Receive Abort/Branch' command is performed in the data management unit. The data management unit stops immediately processing the current descriptor and branches to a new descriptor pointed to by CSPEC\_FRDA. In case that the 'Receive Abort/Branch' command is issued while a packet is written to system memory a 'Receive Abort' interrupt vector is generated and the rest of the frame already stored in receive buffer is discarded. Data reception is continued with a new frame when the data management unit branched to the new descriptor list.

A 'Receive Command Complete' interrupt vector is generated after the channel information is copied into the internal channel database. New commands for the same channel may be given after the MUNICH256 issued the 'Receive Command Complete' interrupt vector.

### Receive Hold Reset

The 'Receive Hold Reset' command must be given after system software has set the HOLD bit of a receive descriptor from '1' to '0'. In case that the MUNICH256 is in hold condition it reads the descriptor which had its HOLD bit set and tests the HOLD bit of the descriptor. If the HOLD bit is set to '0' the data management unit branches to the next descriptor and continues data reception. Otherwise the particular channel remains in hold condition.

The MUNICH256 will NOT generate a 'Receive Command Complete' interrupt vector after this command is programmed.

---

## Channel Programming / Reprogramming Concept

### Receive Debug

The 'Receive Debug' command allows to read back the current settings of the internal channel database. After the 'Receive Debug' command has been programmed system software can read back the current values of the channel specification registers. Register CSPEC\_FRDA contains the value of the next receive descriptor.

The MUNICH256 will NOT generate a 'Receive Command Complete' interrupt vector after this command is programmed.

*Note: The setting of the internal channel database is not copied into the channel specification registers and therefore the values read can not be used to program another channel. After system software has used the 'Receive Debug' command it must reprogram the channel specification registers to setup a new channel.*



## 7 Reset and Initialization procedure

Since the term “initialization” can have different meanings, the following definition applies:

### Chip Initialization

Generating defined values in all on-chip registers, RAMs (if required), flip-flops etc.

### Mode Initialization

Software procedure, that prepares the device to its required operation, i.e. mainly writing on-chip registers to prepare the device for operation in the respective system environment.

### Operational programming

Software procedures that setup, maintain and shut down operational modes, i.e. initialize logical channel or maintain framing operations on selected ports.

## 7.1 Chip Initialization

### Hardware reset

The hardware reset  $\overline{RST}$  has to be applied to the device. Chip input  $\overline{TRST}$  must be activated prior to or while asserting  $\overline{RST}$  and should be held asserted as long as the boundary scan operation is not required. System clock must start running during reset. During reset:

- All I/Os and all outputs are tri-state.
- All registers, state machines, flip-flops etc. are set asynchronously to their reset values and all internal modules are set to their initial state.
- All interrupts are masked.
- The register bit CONF1.STOP is set to ‘1’.

After hardware reset ( $\overline{RST}$  deasserted) system clock CLK is assumed to be running. Serial clocks must be low/high or running. The PCI and the local bus interface pins go into their idle state. All serial line outputs are tri-state.

The PCI interface becomes active and depending on input pin SPLOAD starts to read subsystem ID/subsystem vendor ID and Memory commands out of external EEPROM via the SPI interface. The serial clock is derived from the PCI clock. As long as this procedure is active, the PCI interface answers all accesses with retry. After the PCI interface has finished its self initialization it can be configured with PCI configuration cycles.

In parallel to PCI self initialization the internal modules start their RAM initialization. As long as the RAM initialization is running the internal modules indicate this condition with

---

## Reset and Initialization procedure

their initialization in progress signal. The register bit CONF1.IIP is the result of all signals. As soon as all internal modules have finished their RAM initialization the register bit CONF1.IIP is deasserted. Software must poll the register bit CONF1.IIP until this bit has been deasserted. Read access to registers other than CONF1 is prohibited and may result in unexpected behavior of the design. Write accesses are not allowed.

Chip initialization is finished when CONF1.IIP is '0'.

### Software Reset

Alternately the MUNICH256 provides the capability to issue a software reset via register bit CONF1.SRST. During software reset all interfaces except PCI interface are forced into their idle state. After software reset is set the MUNICH256 starts its self initialization and IIP will be asserted. Chip initialization is finished when CONF1.IIP is deasserted. Afterwards the software reset bit must be set to '0' to allow further operation.

## 7.2 Mode Initialization

After chip initialization is finished the system software has to setup the device for the required function.

The system software has to poll bit CONF1.IIP (FCONF.IIP). As soon as CONF1.IIP is deasserted, the system software has to clear bit CONF1.STOP and has to set the general operating modes in register CONF1.

The port mode has to be programmed. It is assumed, that port clocks are active according to the selected port mode. The ports shall be disabled, thus no incoming data is forwarded to the time slot assigner and the outputs are still tri-state.

### Transmit direction

The ports have to be enabled via register TEN. The transmit port synchronizes to the external synchronization pulse. After a port has been enabled payload data is provided from the time slot assigner. Since the time slot assignment is in reset state, that is all time slots are set to inhibit, data bits are tri-state.

### Receive direction

The ports have to be enabled via register REN. The receiver synchronizes to the external synchronization pulse. As soon as frame synchronization has been achieved, incoming payload data is passed to the time slot assigner. Since the time slot assignment is in the reset state, that is all time slots are set to inhibit, data bits are discarded.

## 8 Register Description

The register description of the MUNICH256 is divided into two parts, an overview of all internal registers and in the second part a detailed description of all internal registers.

### 8.1 Register Overview

The first part of the register overview describes the PCI configuration space registers. The second part describes the register set which can be accessed from PCI side only. These registers are used to setup the main operation modes and to run the channel engines of the device. The last part describes the register set of the mailbox and the local interrupt FIFO. These registers may be accessed through the local microprocessor interface or via PCI.

*Note: Register locations not contained in the following register tables are “reserved”. In general all write accesses to reserved registers are discarded and read access to reserved registers result in 00000000<sub>H</sub>. Nevertheless, to allow future extensions, system software shall access documented registers only, since writes to reserved registers may result in unexpected behavior. The read value of reserved registers shall be handled as don't care.*

*Unused and reserved bits are marked with a gray box. The same rules as given for register accesses apply to reserved bits, except that system software shall write the documented default value in reserved bit locations.*

#### 8.1.1 PCI Configuration Register Set (Direct Access)

Table 8-1 PCI Configuration Register Set

Register	Access	Address	Reset value	Comment	Page
<b>Standard configuration space register</b>					
DID/VID	R	00 <sub>H</sub>	2106110A <sub>H</sub>	Device ID/Vendor ID	129
STA/CMD	R/W	04 <sub>H</sub>	02A00000 <sub>H</sub>	Status/Command	130
CC/RID	R	08 <sub>H</sub>	02800001 <sub>H</sub>	Class Code/Revision ID	132
BIST/ HEAD/ LATIM/ CLSIZ	R/W	0C <sub>H</sub>	00000000 <sub>H</sub>	Built-in Self Test/ Header Type/ Latency Timer/ Cache Line Size	133
BAR1	R/W	10 <sub>H</sub>	00000000 <sub>H</sub>	Base Address 1	134
BAR2	R/W	14 <sub>H</sub>	00000000 <sub>H</sub>	Base Address 2	135
BARX	R	14 <sub>H</sub> -24 <sub>H</sub>	00000000 <sub>H</sub>	Base Address Not Used	

Register Description

Register	Access	Address	Reset value	Comment	Page
CISP	R	28 <sub>H</sub>	00000000 <sub>H</sub>	Cardbus CIS Pointer	
SSID/ SSVID	R	2C <sub>H</sub>	00000000 <sub>H</sub>	Subsystem ID/ Subsystem Vendor ID	136
ERBAD	R	30 <sub>H</sub>	00000000 <sub>H</sub>	Expansion ROM Base Adr.	
Reserved	R	34 <sub>H</sub>	00000000 <sub>H</sub>	Reserved	
Reserved	R	38 <sub>H</sub>	00000000 <sub>H</sub>	Reserved	
MAXLAT/ MINGNT/ INTPIN/ INTLIN	R/W	3C <sub>H</sub>	06020100 <sub>H</sub>	Maximum Latency/ Minimum Grant/ Interrupt Pin/ Interrupt Line	137
<b>User defined configuration space register</b>					
SPI	R/W	40 <sub>H</sub>	0000001F <sub>H</sub>	SPI Access Register	138
REQ	R/W	44 <sub>H</sub>	00000000 <sub>H</sub>	REQ/GNT Config Register	140
MEM	R/W	48 <sub>H</sub>	000007E6 <sub>H</sub>	PCI Memory Command	141
DEBUG	R	4C <sub>H</sub>	00000000 <sub>H</sub>	PCI Debug Support	143

### 8.1.2 PCI Slave Register Set (Direct Access)

This section shows all registers which are located on the first configuration bus. These registers are used to setup the basic operating modes of the device and to setup the port, time slots and channels. System software has access to these registers via the PCI bus.

**Table 8-2 PCI Slave Register Set**

Register	Access	Address	Reset value	Comment	Page
<b>General Control</b>					
CONF1	R/W	040 <sub>H</sub>	820000F1 <sub>H</sub>	Configuration Register 1	161
CONF2	R/W	044 <sub>H</sub>	00000000 <sub>H</sub>	Configuration Register 2	164
CONF3	R/W	048 <sub>H</sub>	00090000 <sub>H</sub>	Configuration Register 3	166
RBAFT	W	04C <sub>H</sub>	00000000 <sub>H</sub>	Receive Buffer Access Failed Interrupt Threshold	167
SFDT	W	050 <sub>H</sub>	00000000 <sub>H</sub>	Small Frame Dropped Interrupt Threshold Register	168
<b>Interrupt control PCI bus side</b>					
IQIA	R/W	0E0 <sub>H</sub>	00000000 <sub>H</sub>	Interrupt Queue Initialization	186
IQBA	R/W	0E4 <sub>H</sub>	00000000 <sub>H</sub>	Interrupt Queue Base Addr.	188
IQBL	R/W	0E8 <sub>H</sub>	00000000 <sub>H</sub>	Interrupt Queue Length	189
IQMASK	R/W	0EC <sub>H</sub>	00000000 <sub>H</sub>	Interrupt Queue Mask	190
GISTA/GIACK	R/W	0F0 <sub>H</sub>	00000000 <sub>H</sub>	Global Interrupt Status/ Global Interrupt Acknowledge	191
GMASK	R/W	0F4 <sub>H</sub>	FFFFFFFF <sub>H</sub>	Interrupt Mask	193
<b>Channel specification registers (* = CSPEC)</b>					
*_CMD	W	000 <sub>H</sub>	00000000 <sub>H</sub>	Command	144
*_MODE_REC	R/W	004 <sub>H</sub>	00000000 <sub>H</sub>	Mode Receive	146
*_REC_ACCM	R/W	008 <sub>H</sub>	00000000 <sub>H</sub>	Receiver ACCM Map	149
*_MODE_XMIT	R/W	014 <sub>H</sub>	00000000 <sub>H</sub>	Mode Transmit	150
*_XMIT_ACCM	R/W	018 <sub>H</sub>	00000000 <sub>H</sub>	Transmit ACCM Map	153
*_BUFFER	R/W	020 <sub>H</sub>	00200000 <sub>H</sub>	Buffer Configuration	154
*_FRDA	R/W	024 <sub>H</sub>	00000000 <sub>H</sub>	First Receive Descriptor Addr.	157

Register Description

Register	Access	Address	Reset value	Comment	Page
*_FTDA	R/W	028 <sub>H</sub>	00000000 <sub>H</sub>	First Transmit Descriptor Address	158
*_IMASK	R/W	02C <sub>H</sub>	00000000 <sub>H</sub>	Interrupt Vector Mask	159
<b>Port and time slot control registers</b>					
PMIAR	R/W	060 <sub>H</sub>	00000000 <sub>H</sub>	Port Mode Indirect Access	169
PMR	R/W	064 <sub>H</sub>	0104C000 <sub>H</sub>	Port Mode	170
REN	R/W	068 <sub>H</sub>	00000000 <sub>H</sub>	Receive Enable	173
TEN	R/W	06C <sub>H</sub>	00000000 <sub>H</sub>	Transmit Enable	174
TSAIA	R/W	070 <sub>H</sub>	00000000 <sub>H</sub>	Time slot Assignment Indirect Access	175
TSAD	R/W	074 <sub>H</sub>	02000000 <sub>H</sub>	Time slot Assignment Data	177
<b>PPP character map/ demap registers</b>					
REC_ACCMX	R/W	080 <sub>H</sub>	00000000 <sub>H</sub>	Receive Extended ACCM Map	179
XMIT_ACCMX	R/W	090 <sub>H</sub>	00000000	Transmit Extended ACCM Map	183
<b>Receive buffer control</b>					
RBMON	R	0B0 <sub>H</sub>	02000BFF <sub>H</sub>	Receive Buffer Monitor	184
RBTH	R/W	0B4 <sub>H</sub>	02000001 <sub>H</sub>	Receive Buffer Threshold Report	185
<b>Maintenance</b>					
RBAFC	R	084 <sub>H</sub>	00000000 <sub>H</sub>	Receive Buffer Access Failed Counter	180
SFDIA	R/W	088 <sub>H</sub>	00000000 <sub>H</sub>	Small Frame Dropped Indirect Access	181
SFDC	R	08C <sub>H</sub>	00000000 <sub>H</sub>	Small Frame Dropped Counter	182

### 8.1.3 PCI and Local Bus Register Set (Direct Access)

This section describes the registers which are located on the configuration bus II (see also These registers can be accessed either from PCI bus via the internal bus bridge or from the local bus side.

*Note: Since the local bus is 16-bit wide and the PCI bus is 32-bit wide, the upper 16 bit of data coming from/to PCI are discarded.*

*Note: Please note that read accesses to local bus registers via PCI bus and therefore the internal bus bridge may result in latencies which exceed the 16 clock rule of PCI specification. Exceeding the 16 clock rule results in target initiated retry on PCI bus. In this case the read cycle needs to be repeated.*

**Table 8-3 PCI and Local Bus Slave Register Set**

Register	Access	Address (PCI)	Address (Local Bus)	Reset value	Comment	Page
FCONF	R/W	100 <sub>H</sub>	00 <sub>H</sub>	8080 <sub>H</sub>	Configuration Register	194
MTIMER	R/W	104 <sub>H</sub>	00 <sub>H</sub>	0001 <sub>H</sub>	Master Local Bus Timer	196
<b>Interrupt control for local bus side</b>						
INTCTRL	R/W	108 <sub>H</sub>	04 <sub>H</sub>	0001 <sub>H</sub>	Interrupt Control	197
INTFIFO	R	10C <sub>H</sub>	06 <sub>H</sub>	FFFF <sub>H</sub>	Interrupt FIFO	198
<b>Mailbox registers</b>						
MBE2P0	R/W	140 <sub>H</sub>	20 <sub>H</sub>	0000 <sub>H</sub>	Mailbox Local Bus to PCI Command	199
MBE2P1	R/W	144 <sub>H</sub>	22 <sub>H</sub>	0000 <sub>H</sub>	Mailbox Local Bus to PCI Data Registers 1 through 7	200
MBE2P2		148 <sub>H</sub>	24 <sub>H</sub>			
MBE2P3		14C <sub>H</sub>	26 <sub>H</sub>			
MBE2P4		150 <sub>H</sub>	28 <sub>H</sub>			
MBE2P5		154 <sub>H</sub>	2A <sub>H</sub>			
MBE2P6		158 <sub>H</sub>	2C <sub>H</sub>			
MBE2P7		15C <sub>H</sub>	2E <sub>H</sub>			
MBP2E0	R/W	160 <sub>H</sub>	30 <sub>H</sub>	0000 <sub>H</sub>	Mailbox PCI to Local Bus Command	201

Register Description

Register	Access	Address (PCI)	Address (Local Bus)	Reset value	Comment	Page
MBP2E1	R/W	164 <sub>H</sub>	32 <sub>H</sub>	0000 <sub>H</sub>	Mailbox PCI to Local Bus Data Registers 1 through 7	202
MBP2E2		168 <sub>H</sub>	34 <sub>H</sub>			
MBP2E3		16C <sub>H</sub>	36 <sub>H</sub>			
MBP2E4		170 <sub>H</sub>	38 <sub>H</sub>			
MBP2E5		174 <sub>H</sub>	3A <sub>H</sub>			
MBP2E6		178 <sub>H</sub>	3C <sub>H</sub>			
MBP2E7		17C <sub>H</sub>	3E <sub>H</sub>			



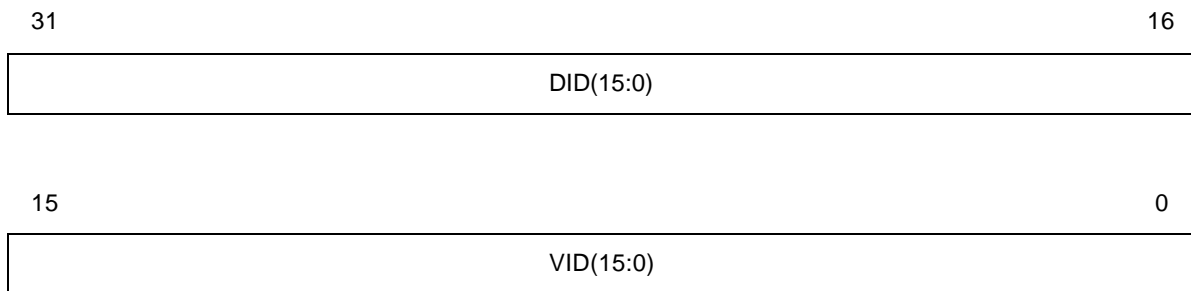
## 8.2 Detailed Register Description

### 8.2.1 PCI Configuration Register

#### DID/VID

#### Device ID/Vendor ID

Access : read  
Address : 00<sub>H</sub>  
Reset Value : 2106110A<sub>H</sub>



**DID** Device ID  
The device ID identifies the particular device. It is hardwired to value 2106<sub>H</sub>.

**VID** Vendor ID  
The vendor ID identifies the manufacturer of the device. It is hardwired to value 110A<sub>H</sub>.

**STAT/CMD**  
**Status/Command Register**

Access : read/write  
Address : 04<sub>H</sub>  
Reset Value : 02A00000<sub>H</sub>

31	30	29	28	27	26	25	24	23	22	21						16	
DPE	SSE	RMA	RTA	0	01 <sub>B</sub>		DPED	1	0	1	0	0	0	0	0		
							8	6							2	1	0
0	0	0	0	0	0	0	SE	0	PER	0	0	0	0	BM	MS	0	

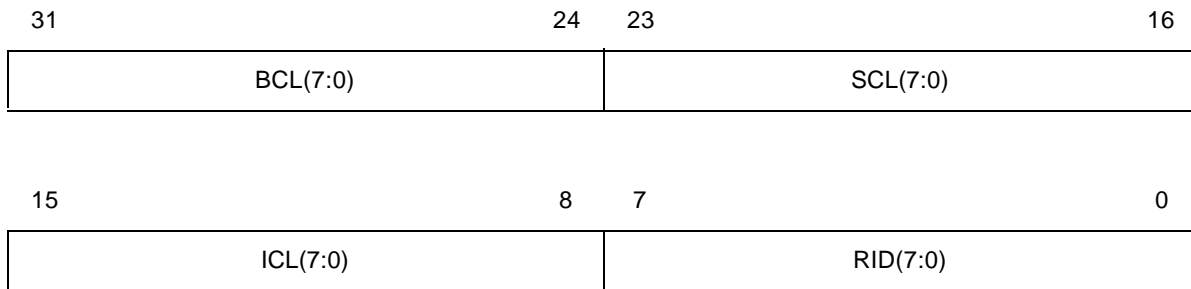
- DPE**            Detected Parity Error  
This bit will be asserted whenever the MUNICH256 detects a parity error.  
0        No parity error detected.  
1        Parity error detected. This bit will be cleared by writing a '1' to this bit position.
- SSE**            Signaled System Error  
This bit will be asserted whenever the MUNICH256 asserted  $\overline{\text{SERR}}$ . For system error conditions see bit SE.  
0        No system error signaled.  
1        System error has been signaled. This bit will be cleared by writing a '1' to this bit position.
- RMA**            Received Master Abort  
This bit will set whenever a transaction in which the MUNICH256 acted as bus master was terminated with master abort.  
0        No master abort detected.  
1        Transaction terminated with master abort. This bit will be cleared by writing a '1' to this bit.

Register Description

RTA	<p>Received Target Abort</p> <p>This bit will be set whenever a transaction in which the MUNICH256 acted as bus master was terminated with target abort.</p> <p>0 No target abort detected.</p> <p>1 Transaction terminated with target abort. This bit will be cleared by writing a '1' to this bit.</p>
DPED	<p>Data Parity Error Detected</p> <p>0 No data parity error detected.</p> <p>1 The following three conditions are met:</p> <ul style="list-style-type: none"> <li>•The bus agent asserted <math>\overline{\text{PERR}}</math> itself or observed <math>\overline{\text{PERR}}</math> asserted.</li> <li>•The bus agent acted as bus master for the operation in which the error occurred.</li> <li>•The Parity Error Response Bit is set</li> </ul>
SE	<p><math>\overline{\text{SERR}}</math> Enable</p> <p>This bit enables assertion of <math>\overline{\text{SERR}}</math> in case of severe system errors.</p> <p>0 Assertion of <math>\overline{\text{SERR}}</math> disabled.</p> <p>1 Enables report of</p> <ul style="list-style-type: none"> <li>•Address parity errors</li> <li>•Master abort</li> <li>•Target abort</li> </ul>
PER	<p>Parity Error Response</p> <p>This bit enables reporting of parity errors via pin <math>\overline{\text{PERR}}</math>.</p> <p>0 Assertion of <math>\overline{\text{PERR}}</math> disabled.</p> <p>1 Enables the assertion of <math>\overline{\text{PERR}}</math>. See also Data Parity Error Detected.</p>
BM	<p>Bus Master</p> <p>This bit controls a device ability to act as a master on PCI bus.</p> <p>0 Disables the device from generating PCI accesses.</p> <p>1 Allows the device to act as bus master.</p>
MS	<p>Memory Space</p> <p>This bit controls the device response to memory space accesses.</p> <p>0 Response to memory space accesses disabled.</p> <p>1 Allows a device to respond to memory space accesses.</p>

**CC/RID**  
**Class Code/Revision ID**

Access : read  
Address : 08<sub>H</sub>  
Reset Value : 02800001<sub>H</sub>



The class code, consisting of base class, subsystem class and interface class, is used to identify the generic function of the device and, in some cases, a specific register-level programming interface.

- BCL**            Base Class

The base class is hardwired to 02<sub>H</sub>, which identifies this device as a network controller.
- SCL**            Sub Class

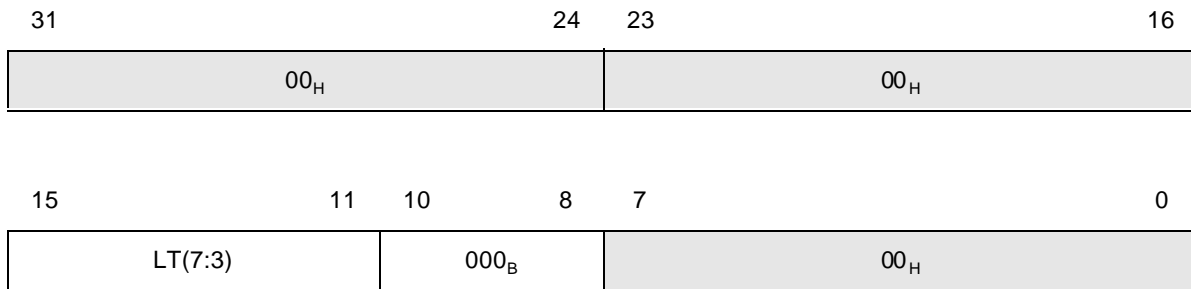
The sub class is hardwired to 80<sub>H</sub>, which together with the base class identifies this device as 'Other network controller'.
- ICL**            Interface Class

The interface class is hardwired to 00<sub>H</sub>.
- RID**            Revision ID

The revision ID identifies the current version of the device. It is hardwired to 01<sub>H</sub>.

**BIST/Header Type/Latency Timer/Cache Line Size**

Access : read/write  
 Address : 0C<sub>H</sub>  
 Reset Value : 00000000<sub>H</sub>

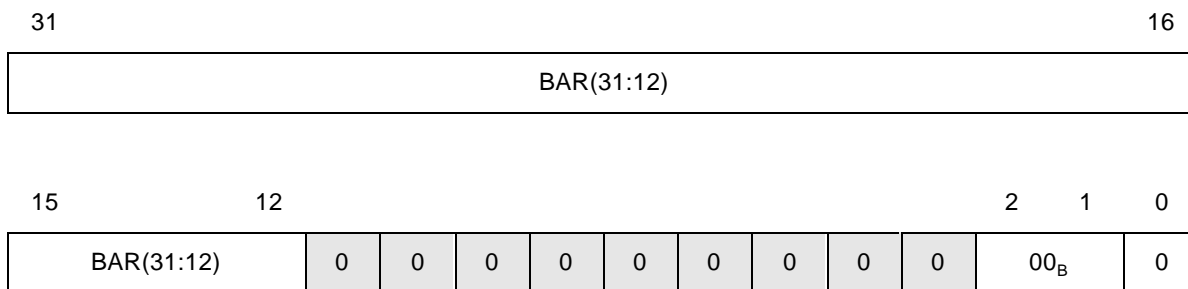


**LT**                      Latency Timer

The value of this register times eight specifies, in units of PCI clocks, the value of the latency timer for this PCI bus master.

**BAR1**  
**Base Address 1**

Access : read/write  
Address : 10<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



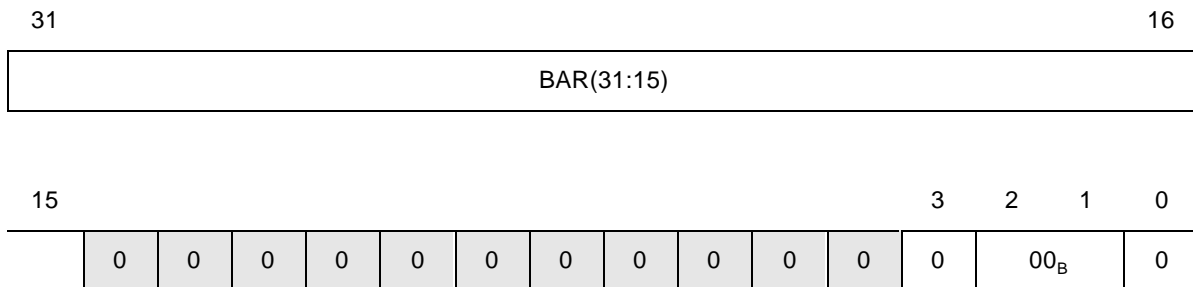
The first base address of the MUNICH256 is marked as non-prefetchable and can be relocated anywhere in 32 bit address space of PCI memory. The MUNICH256 supports memory accesses only.

**BAR**                      **Base Address**

The base address will be used for determining the address space of the MUNICH256 and to do the mapping of the address space. Since the device allocates a total of 4 kByte address space BAR(31:12) are implemented as read/writable.

**BAR2**  
**Base Address 2**

Access : read/write  
Address : 14<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



The second base address of the MUNICH256 is marked as non-prefetchable and can be relocated anywhere in 32 bit address space of PCI memory. The MUNICH256 supports memory accesses only. All accesses to memory regions defined by BAR2 will be mapped to the local bus.

**BAR**            **Base Address**

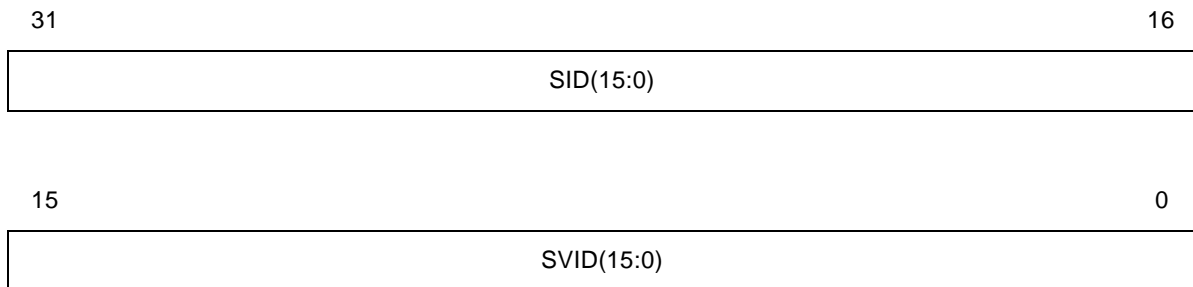
The base address will be used for determining the address space of the memory regions located on the local bus of the MUNICH256 and to set the mapping of the address space. The MUNICH256 can access a total of 24 kByte address space on the local bus as a bus master.

In those applications where the master functionality of MUNICH256 is not needed the second base address register BAR2 may be disabled using bit MEM.BAR2 in the PCI user configuration space.

**SID/SVID**

**Subsystem ID/Subsystem vendor ID**

Access : read  
Address : 2C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**SID**            Subsystem ID  
The subsystem ID uniquely identifies the add-in board or subsystem where the system resides. The value of SID may be reconfigured after the reset phase of the system via the SPI interface.

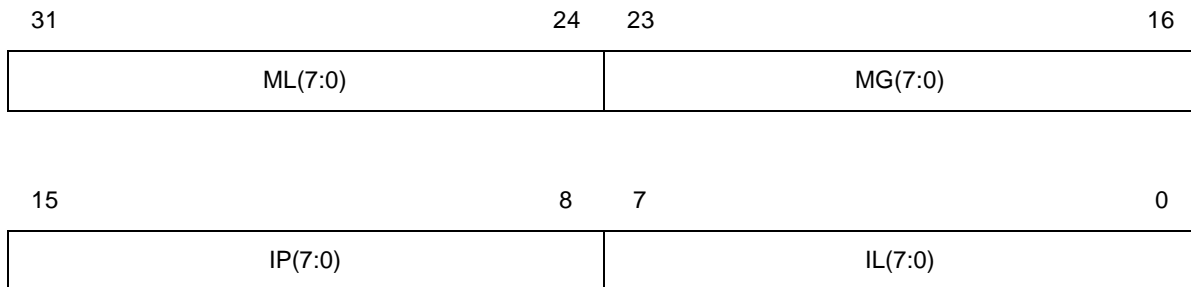
**SVID**           Subsystem Vendor ID  
The subsystem vendor ID identifies the vendor of an add-in board or subsystem. The value may be reconfigured after the reset phase of the system via the SPI interface.



**ML/MG/IP/IL**

**Maximum Latency/Minimum Grant/Interrupt Pin/Interrupt Line**

Access : read/write  
Address : 3C<sub>H</sub>  
Reset Value : 06020100<sub>H</sub>



- ML**            Maximum Latency

This value specifies how often the device needs to access the PCI bus in multiples of 1/4 us. The value is hardwired to 06<sub>H</sub>.
- MG**            Minimum Grant

This value specifies how long of a burst period the device needs, assuming a clock rate of 33 MHz in multiples of 1/4 us. The value is hardwired to 02<sub>H</sub>.
- IP**            Interrupt Pin

The interrupt pin register tells which interrupt pin the device uses. Refer to section 6.2.4 and to section 2.2.6 of the PCI specification Rev. 2.1. The value is hardwired to 01<sub>H</sub>.
- IL**            Interrupt Line

The interrupt line register is used to communicate interrupt line routing information.



---

**Register Description**

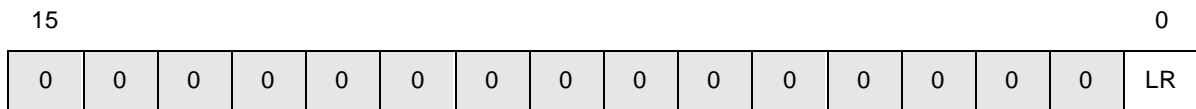
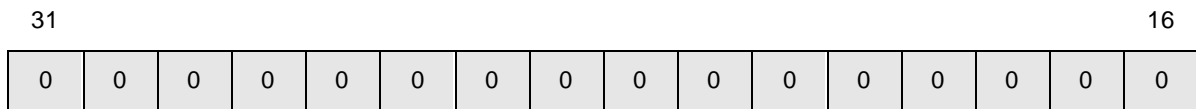
SD

SPI Data

For the write status register transactions and the write data to memory array transactions, the data, that has to be written to the EEPROM, must be written to this register before the transaction is started. After a read status register transaction or read data from memory array transaction has finished (start bit is deasserted), the byte received from the EEPROM is available in this register.

**LR**  
**Long Request Register**

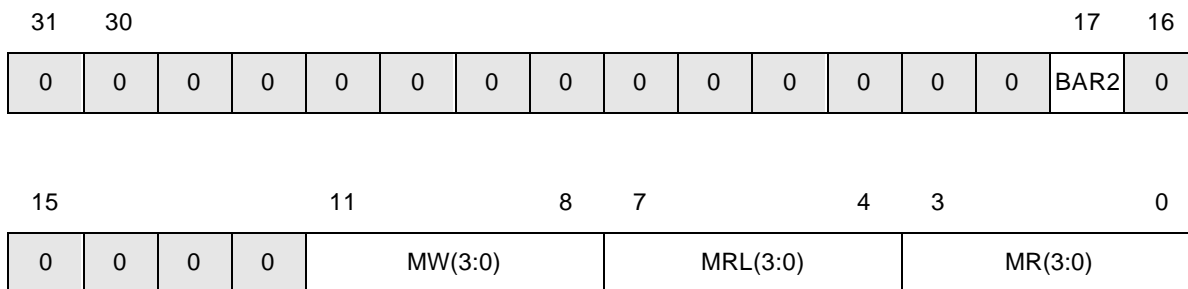
Access : read/write  
Address : 44<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



- LR                      Long Request
- 0            The PCI interface deasserts the  $\overline{REQ}$  signal in parallel with the assertion of the FRAME signal.
  - 1            The  $\overline{REQ}$  signal will be deasserted in parallel with the deassertion of FRAME.

**MEM**  
**PCI Memory Command Register**

Access : read/write  
Address : 48<sub>H</sub>  
Reset Value : 000007E6<sub>H</sub>



- BAR2** Enable Base Address Register 2  
Setting this bit enables Base Address Register 2. Per default base address register two is disabled. If an EEPROM is connected to the SPI interface the value of this bit can be loaded via the EEPROM. Additionally this bit can set using standard PCI configuration write commands.  
0 Base Address Register 2 is disabled.  
1 Base Address Register 2 is enabled.
- MW** Memory Write Command  
The value of this register contains the write command to be used during initiator transfers and is set to memory write after reset. The value of this register is configurable during setup of the bridge either by loading the value from EEPROM or by writing from PCI side.
- MRL** Memory Read Command (Long transfers)  
The value of this register defines command to be used for read transfers which are equal or more than two DWORDs and is set to memory read line after reset. The value of this register is configurable during run time of the bridge either by loading the value from EEPROM or by writing from PCI side.
- MR** Memory Read Command  
The value of this register defines command to be used for read transfers of single DWORDs. The value of this register is configurable during run

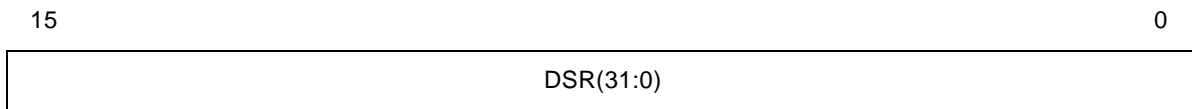
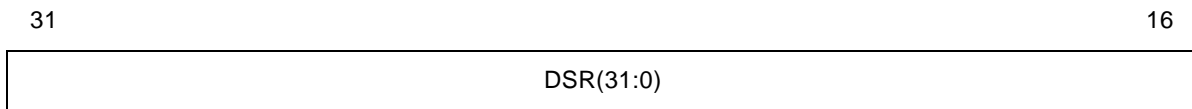
---

**Register Description**

time of the bridge either by loading the value from EEPROM or by reading or writing from PCI side.

**DEBUG**  
**PCI Debug Support Register**

Access : read  
Address : 4C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**DSR** Debug Support register

The value of this register contains the address of the next initiator transfer during normal operation. In case of disconnect, retry, master abort and target abort the register contains the address of the failed transaction.





---

**Register Description**

CMDX	Command Transmit For detailed description of transmit commands and programming sequences refer to <a href="#">Chapter 6.2</a> . 01 <sub>H</sub> Transmit Init 02 <sub>H</sub> Transmit Off 04 <sub>H</sub> Transmit Abort/Branch 08 <sub>H</sub> Transmit Hold Reset 10 <sub>H</sub> Transmit Debug 20 <sub>H</sub> Transmit Idle 40 <sub>H</sub> Transmit Update
CMDR	Command Receive For detailed description of receive commands and programming sequences refer to <a href="#">Chapter 6.3</a> . 01 <sub>H</sub> Receive Init 02 <sub>H</sub> Receive Off 04 <sub>H</sub> Receive Abort/Branch 08 <sub>H</sub> Receive Hold Reset 10 <sub>H</sub> Receive Debug
CHAN	Channel select 0..255 Selects the channel to be programmed or debugged.

*Note: Transmit init for a channel must be programmed only after reset or after a transmit off command, i.e. two transmit init commands for the same channel are not allowed.*

**CSPEC\_MODE\_REC**  
**Channel Specification Mode Receive Register**

Access : read/write  
Address : 004<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

31			28	27			24	23							16
0	0	0	DEL	ACCMX(3:0)				RFLAG(7:0)							
15	14	13	12	11	10	9	8							1	0
0	SFDE	TFF	INV	TMP	CRCX	CRC 32	CRC DIS	0	0	0	0	0	0	PMD(1:0)	

- DEL** DEL (Delete) Demap  
This bit enables demapping of the control character DEL (7F<sub>H</sub>). This bit is valid in PPP modes only.  
0 Disable demapping of control character DEL.  
1 Enable demapping of control character DEL.
- ACCMX** Extended ACCM  
In addition to the *Channel Specification Receive ACCM Map* the user can select four global user definable characters for character demapping in PPP modes. Setting one or more of the bits ACCM(3) through ACCM(0) enables the corresponding character which can be found in register REC\_ACCMX.  
0 Disable the selected character in REC\_ACCMX for character demapping.  
1 Enable the corresponding character in register REC\_ACCMX for character demapping.
- RFLAG** Receive Flag  
Used in transparent mode only. The RFLAG constitutes the flag that is filtered from the received bit stream if enabled via bit TFF.

## Register Description

SFDE	<p>Short/Small Frame Drop Enable</p> <p>This bit enables either the drop of short frames or the drop of small frames. This bit is valid in HLDC and PPP modes only.</p> <p>0 Short Frame Drop. Frames smaller than four bytes payload data (CRC32) or smaller than two bytes payload data (CRC16) are dropped. This function is not available if bit CRCX is enabled.</p> <p>1 Small Frame Drop. Frames (Payload and CRC) which are smaller or equal to CONF3.MINFL are dropped.</p>
TFF	<p>TMA Flag</p> <p>This bit enabled flag extraction in TMA mode and is available if non of the bits belonging to this channel is masked.</p> <p>0 No flag extraction</p> <p>1 Enable flag extraction. The flag specified in RFLAG will be extracted from the received data stream.</p>
INV	<p>Bit Inversion</p> <p>When bit inversion is enabled incoming channel data is inverted before processed by the protocol machine. E.g. incoming octet 81<sub>H</sub> will be recognized as idle flag in HDLC mode.</p> <p>0 No Bit Inversion</p> <p>1 Bit Inversion</p>
TMP	<p>Transparent Mode Packing</p> <p>This bit enables the transparent mode packing and is valid in TMA mode only. This feature is applicable if at least one bit in any time slot is masked.</p> <p>0 Incoming masked bits are substituted with '1'. The non-used (masked) data bits are substituted by '1's.</p> <p>1 If subchanneling is used in transparent mode (i.e. less than 8 bits of a time slot are used), the non-used (masked) data bits are discarded.</p>
CRCX	<p>CRC Transfer</p> <p>This bit enables the capability to store the CRC checksum of incoming data packets in system memory together with the payload data.</p> <p>0 The CRC checksum from the incoming data packet will be removed from the packet and not transferred to the shared memory.</p> <p>1 The CRC checksum together with the payload data is transferred to the shared memory.</p>

---

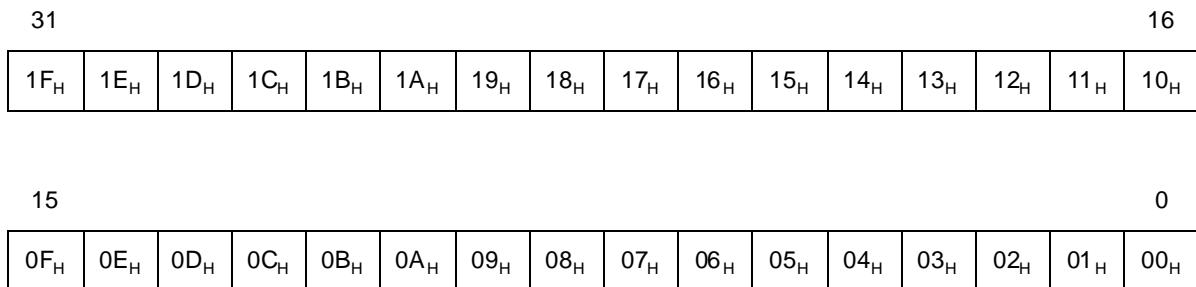
**Register Description**

CRC32	<p>CRC32 Select</p> <p>This bit selects the generator polynomial in the receiver. The checksum of incoming data packets will be compared against CRC16 or CRC32. CRC Select is valid in HDLC and PPP modes only.</p> <p>0      Select CRC16 checksum.</p> <p>1      Select CRC32 checksum.</p>
CRCDIS	<p>CRC Check Disable</p> <p>This bit disables CRC Check in HDLC and PPP protocol modes.</p> <p>0      CRC check is enabled.</p> <p>1      CRC check is disabled.</p>
PMD	<p>Protocol Machine Mode</p> <p>These bit fields select the protocol machine mode in receive direction.</p> <p>00<sub>B</sub>    Select HDLC operation.</p> <p>01<sub>B</sub>    Select Bit synchronous PPP.</p> <p>10<sub>B</sub>    Select Byte synchronous PPP.</p> <p>11<sub>B</sub>    Select Transparent Mode.</p>

**CSPEC\_REC\_ACCM**

**Channel Specification Receive ACCM Map Register**

Access : read/write  
Address : 008<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



Any of the given characters can be selected for character demapping. If a bit is set the corresponding character is expected to be mapped by the control ESC character and is removed if received. These bits are valid in octet synchronous PPP modes only.

*Note: If this register needs to be reprogrammed, it must be done **before** accessing the register CSPEC\_MODE\_REC.*



Register Description

INV	<p>Bit Inversion</p> <p>If bit inversion is enabled outgoing channel data is inverted after processed by the protocol machine. E.g. a outgoing idle flag is transmitted as octet 81<sub>H</sub> in HDLC mode.</p> <p>0      Disable bit inversion.</p> <p>1      Enable bit inversion.</p>
TMP	<p>Transparent Mode Pack</p> <p>This bit enables the transparent mode packing and is valid in TMA mode only. This feature is applicable if at least one bit in any time slot is masked.</p> <p>0      If subchanneling is used outgoing masked bits of data octet are discarded and substituted with '1'.</p> <p>1      If subchanneling is used outgoing masked bits are sent as '1'. The remaining bits of data are sent in the next time slot.</p>
CRC32	<p>CRC 32 Select</p> <p>This bit selects the generator polynomial in the transmitter. The checksum of outgoing data packets will be generated according to CRC16 or CRC32. CRC32 Select is valid in HDLC and PPP modes only.</p> <p>0      Select CRC16 generation.</p> <p>1      Select CRC32 generation.</p>
CRCDIS	<p>CRC Disable</p> <p>This bit enables generation and transmission of a CRC checksum. CRC disable is valid in HDLC and PPP modes only.</p> <p>0      CRC generation and transmission is disabled.</p> <p>1      CRC generation and transmission is enabled.</p>
ACCMX	<p>Enable extended ACCM character</p> <p>The selected bits in bit field ACCMX denote the enabled characters in XMIT_ACCMX.</p> <p>In addition to the <i>Channel Specification Transmit ACCM Map</i> the user can select four global user definable characters for character mapping in PPP modes. Setting one or more of the bits ACCM(3) through ACCM(0) enables the corresponding character which can be found in register XMIT_ACCMX.</p> <p>0      Disable the selected character in XMIT_ACCMX for character mapping.</p> <p>1      Enable the corresponding character in register XMIT_ACCMX for character mapping.</p>

---

**Register Description**

DEL	<p>DEL (Delete) Map Flag</p> <p>This bit enables mapping of the control character DEL (7F<sub>H</sub>). This bit is valid in PPP modes only.</p> <p>0      Disable mapping of DEL.</p> <p>1      Enable mapping of DEL.</p>
PMD	<p>Protocol Machine Mode</p> <p>This bit field selects the protocol machine mode in transmit direction.</p> <p>00<sub>B</sub>    Select HDLC operation.</p> <p>01<sub>B</sub>    Select Bit synchronous PPP.</p> <p>10<sub>B</sub>    Select Byte synchronous PPP.</p> <p>11<sub>B</sub>    Select Transparent Mode.</p>



**CSPEC\_XMIT\_ACCM**

**Channel Specification Transmit ACCM Map Register**

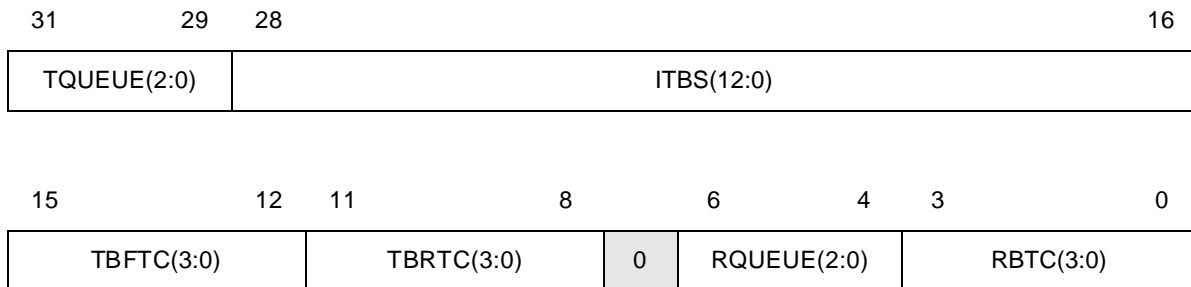
Access : read/write  
 Address : 018<sub>H</sub>  
 Reset Value : 00000000<sub>H</sub>

31															16
1F <sub>H</sub>	1E <sub>H</sub>	1D <sub>H</sub>	1C <sub>H</sub>	1B <sub>H</sub>	1A <sub>H</sub>	19 <sub>H</sub>	18 <sub>H</sub>	17 <sub>H</sub>	16 <sub>H</sub>	15 <sub>H</sub>	14 <sub>H</sub>	13 <sub>H</sub>	12 <sub>H</sub>	11 <sub>H</sub>	10 <sub>H</sub>
															0
0F <sub>H</sub>	0E <sub>H</sub>	0D <sub>H</sub>	0C <sub>H</sub>	0B <sub>H</sub>	0A <sub>H</sub>	09 <sub>H</sub>	08 <sub>H</sub>	07 <sub>H</sub>	06 <sub>H</sub>	05 <sub>H</sub>	04 <sub>H</sub>	03 <sub>H</sub>	02 <sub>H</sub>	01 <sub>H</sub>	00 <sub>H</sub>

Any of the given characters can be selected for character mapping. If a bit is set the corresponding character will be mapped by the control ESC character. These bits are valid in octet synchronous PPP modes only.

**CSPEC\_BUFFER**  
**Channel Specification Buffer Configuration Register**

Access : read/write  
Address : 020<sub>H</sub>  
Reset Value : 00200000<sub>H</sub>



- TQUEUE** Transmit Interrupt Vector Queue  
This bit field determines the interrupt queue where channel interrupts transmit will be stored.
- ITBS** Individual transmit buffer size  
*Note: Please note that the internal architecture is 32 bit wide. Therefore each buffer location corresponds to four data octets.*  
The transmit buffer size configures the number of internal transmit buffer locations for a particular channel. Buffer locations will be allocated on command transmit init and released after command transmit off.  
*Note: The sum of transmit forward threshold and transmit refill threshold must be smaller than the internal buffer size.*
- TBRTC** Transmit Buffer Refill Threshold Code  
*Note: Please note that the internal architecture is 32 bit wide. Therefore each buffer location corresponds to four data octets.*  
TBRTC is a coding for the transmit refill threshold. Please refer to **Table 8-4** for correspondence between code and threshold.  
The internal transmit buffer has a programmable number of buffer locations per channel. When the number of free locations reaches the transmit buffer refill threshold the internal transmit buffer requests new data from the data management unit.

Register Description

**TBFTC** Transmit Buffer Forward Threshold Code  
*Note: Please note that the internal architecture is 32 bit wide. Therefore each buffer location corresponds to four data octets.*

TBFTC is a coding for the transmit buffer forward threshold. Please refer to **Table 8-4** for correspondence between code and threshold.

The transmit buffer forward threshold code determines the number of buffer locations which must be filled until the protocol machine starts transmission. Nevertheless the transmit buffer forwards data packets to the protocol machine as soon as a whole packet or the end of a packet is stored in the transmit buffer.

**RQUEUE** Receive Interrupt Queue.  
This bit field determines the interrupt queue number where channel interrupts receive will be stored.

**RBTC** Receive Buffer Threshold Code  
*Note: Please note that the internal architecture is 32 bit wide. Therefore each buffer location corresponds to four data octets.*

RBTC is a coding for the receive buffer threshold. Please refer to **Table 8-4** for correspondence between code and threshold.

The receive buffer threshold determines the maximum packet size in DWORDs which will be stored in the internal receive buffer for a specific channel. When the packet size reaches the receive buffer threshold or a packet has been completely received, the packet will be forwarded to system memory.

**Table 8-4 Threshold Codings**

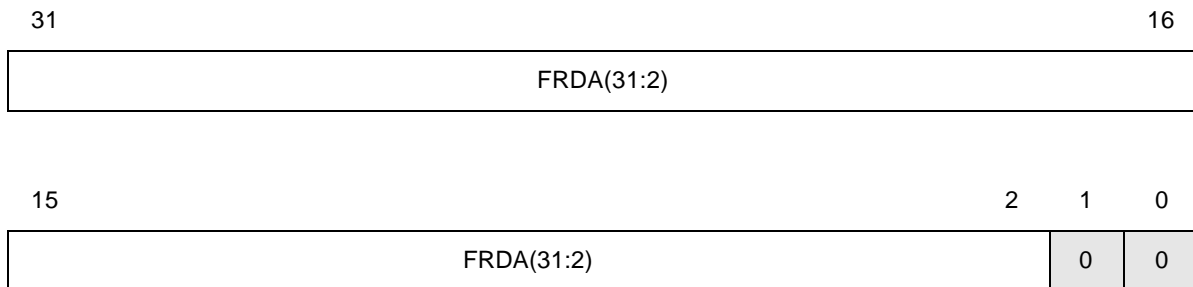
Coding	Threshold in DWORDs	RBTC	TBRTC	TBFTC	TPBL
0000 <sub>B</sub>	1	x	x	x	x
0001 <sub>B</sub>	4	x	x	x	x
0010 <sub>B</sub>	8	x	x	x	x
0011 <sub>B</sub>	12	x	x	x	x
0100 <sub>B</sub>	16	x	x	x	x
0101 <sub>B</sub>	24	x	x	x	x
0110 <sub>B</sub>	32	x	x	x	x
0111 <sub>B</sub>	40	x	x	x	x
1000 <sub>B</sub>	48	x	x	x	x

Register Description

Coding	Threshold in DWORDS	RBTC	TBRTC	TBFTC	TPBL
1001 <sub>B</sub>	64	x	x	x	x
1010 <sub>B</sub>	96	Not Valid		x	Not Valid
1011 <sub>B</sub>	128			x	
1100 <sub>B</sub>	192			x	
1101 <sub>B</sub>	256			x	
1110 <sub>B</sub>	384			x	
1111 <sub>B</sub>	512			x	

**CSPEC\_FRDA**  
**Channel Specification FRDA Register**

Access : read/write  
Address : 024<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



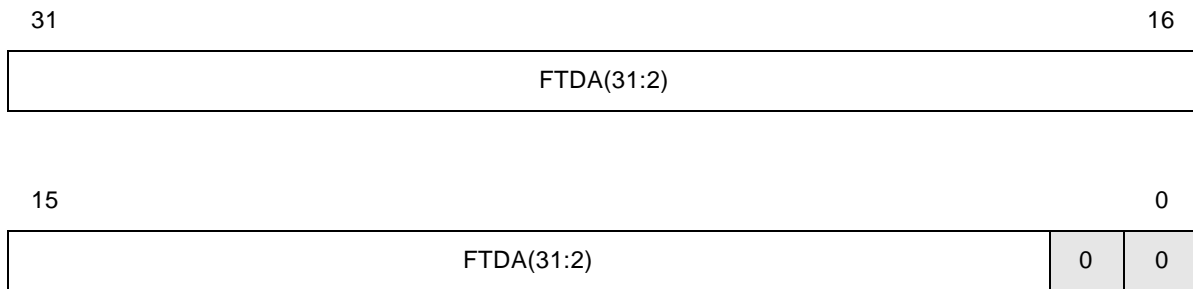
**FRDA**      First Receive Descriptor Address

This 30-bit pointer contains the start address of the first receive descriptor. The receive descriptor is read entirely after the first request of the receive buffer and stored in the on-chip channel database. Therefore all information in the descriptor pointed to by FRDA must be valid when the data management unit branches to this descriptor.

The user can specify a new First Receive Descriptor Address using receive abort/branch command. In this case the First Receive Descriptor Address (FRDA) is used as a pointer to a new linked list. See details on commands in section **“Channel Commands” on Page 116**.

**CSPEC\_FTDA**  
**Channel Specification FTDA Register**

Access : read/write  
Address : 028<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**FTDA**      **First Transmit Descriptor Address**

This 30-bit pointer contains the start address of the first transmit descriptor. The transmit descriptor is read entirely after the first request of the transmit buffer and stored in the on-chip channel database. Therefore all information in the descriptor pointed to by FTDA must be valid when the data management unit branches to this descriptor.

The user can specify a new First Transmit Descriptor Address using the 'Transmit Abort/Branch' command. In this case the first transmit descriptor address (FTDA) is used as a pointer to a new linked list. See details on commands in [Chapter6.2](#).

**CSPEC\_IMASK**  
**Channel Specification Interrupt Vector Mask Register**

Access : read/write  
Address : 02C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

31	30		28					23	22						16
0	TAB	0	HTAB	0	0	0	0	UR	TFE	0	0	0	0	0	TCC
15	14	13	12	11	10	9	8	7	6	5		3	2		0
0	RAB	RFE	HRAB	MFL	RFOD	CRC	ILEN	RFOP	SF	IFTC	0	SFD	SD	0	RCC

For each channel or command related interrupt vector an interrupt vector generation mask is provided. Generation of an interrupt vector itself does not necessarily result in assertion of the interrupt pin. For description of interrupt concept and interrupt vectors see [Chapter 4.7.1](#).

The following definition applies:

- 1 The device will not generate the corresponding interrupt vector, i.e. the interrupt vector is masked.
- 0 An interrupt condition results in generation of the corresponding interrupt vector.

**Channel Interrupt Vector Transmit**

- TAB Mask 'Transmit Abort'
- HTAB Mask 'Hold Caused Transmit Abort'
- UR Mask 'Transmit Underrun'
- TFE Mask 'Transmit Frame End'

**Command Interrupt Vector Transmit**

- TTC Mask 'Transmit Command Complete'

**Command Interrupt Vector Receive**

RAB	Mask 'Receive Abort'
RFE	Mask 'Receive Frame End'
HRAB	Mask 'Hold Caused Receive Abort'
MFL	Mask 'Maximum Frame Length Exceeded'
RFOD	Mask 'Receive Frame Overflow DMU'
CRC	Mask 'CRC Error'
ILEN	Mask 'Invalid Length'
RFOP	Mask 'Receive Frame Overflow'
SF	Mask 'Short Frame Detected'
IFTC	Mask 'Interframe Time-fill Flag' and 'Interframe Time-fill Idle'
SFD	Mask 'Short Frame Dropped'
SD	Mask 'Silent Discard'
RCC	Mask 'Receive Command Complete'



## CONF1 Configuration Register 1

Access : read/write  
Address : 040<sub>H</sub>  
Reset Value : 820000F1<sub>H</sub>

31						25	24	23			21	20					16
IIP	0	0	0	0	0	STOP	SRST	28/16	0	MFLE	MFL(12:0)						
						8	7	6	5	4	3	2	1	0			
MFL(12:0)						MBIM	PBIM	RBIM	RFIM	SFL	RBM	LBE	1				

- IIP** Initialization in Progress (Read Only)
- After reset (hardware reset or software reset) the internal RAM's are self initialized by the MUNICH256. During this time (approx. 250  $\mu$ s) no other accesses to the device than reading register CONF1 or FCONF are allowed. This bit must be polled until it has been deasserted by the MUNICH256.
- 0 Self initialization has finished.
  - 1 Self initialization in progress.
- STOP** Stop
- After reset the MUNICH256 can be switched to 'Fast Initialization' mode. During stop mode internal RAM's will not be accesses by internal state machines. This mode is for test purposes only and allows writing or reading the internal RAM's.
- 0 Device is in normal operation. This bit must be set to zero after chip initialization. See also **"Mode Initialization" on Page 122**.
  - 1 Device is in 'Fast Initialization Mode'. This function is used for test purposes only.
- SRST** Software Reset
- This bit issues a software reset to the MUNICH256. During software reset all interfaces except PCI interface are forced into their idle state. After software reset is set the MUNICH256 starts its self initialization and

Register Description

	IIP will be asserted. When IIP is deasserted system software can reset SRST to '0' to start normal operation again.
	0 Normal operation
	1 Start software reset.
28/16	Select 28/16-port mode
	This bit switches between the 28-port mode and the 16-port mode.
	0 Switch to 16-port mode.
	1 Switch to 28-port mode.
MFLE	Maximum Frame Length Check Enable
	0 Disable maximum frame length check.
	1 Enable maximum frame length check.
MFL	Maximum Frame Length
	MFL defines the maximum length of incoming data packets. Packets exceeding the specified length are reported in the status field of the receive descriptor and if selected in an additional channel interrupt.
MBIM	Mailbox Interrupt Vector Mask
	This bit enables or disables mailbox system interrupt vectors generated by the mailbox.
	0 Enable interrupt vector.
	1 Disable interrupt vector.
PBIM	PCI Bridge Interrupt Vector Mask
	This bit enables or disables the 'PCI Access Error' interrupt vector generated by the PCI bridge.
	0 Enable interrupt vector.
	1 Disable interrupt vector.
RBIM	Receive Buffer Interrupt Vector Mask
	This bit enables or disables system interrupt vectors 'Receive Buffer Queue Early Warning' and 'Receive Buffer Action Queue Early Warning' which are generated by the receive buffer. RBIM is valid only if bit RBM is set.
	0 Enable interrupt vector.
	1 Disable interrupt vector.
RFIM	Receive Buffer Failed Interrupt Vector Mask
	This bit enables or disables the 'Receive Buffer Access Failed' interrupt vector.
	0 Enable interrupt vector.

---

**Register Description**

	1	Disable interrupt vector.
SFL		Short Frame Length
		This bit is a global parameter which defines the length of short frames for all channels.
	0	Short frame is defined as a frame containing less than 4 bytes (CRC16) or less than 6 bytes (CRC32).
	1	Short frame is defined as a frame containing less than 2 bytes (CRC16) or less than 4 bytes (CRC32).
RBM		Receive Buffer Monitor
		This bit is provided to switch between two monitoring functions of the receive buffer. Receive buffer monitor functions are available in register RBTH and RBMON.
	0	The minimum free pool count is captured in register RBTH.
	1	An interrupt is generated, if the free pool counter falls below the value programmed in register RBTH.
LBE		Little/Big Endian Byte Swap
		This bit enables the little or big endian mode, which affects the data structures pointed to by data pointer of receive or transmit descriptor in system memory. Registers, interrupt vectors or descriptors are not affected by little/big endian byte swap.
	0	Switch data section to little endian mode.
	1	Switch data section to big endian mode.

**CONF2**  
**Configuration Register 2**

Access : read/write  
Address : 044<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

31	30	28	27	26	24	23	22	21	20	16
0	SYSQ(2:0)	0	PORTQ(2:0)	0	TBE	TCOD	00000 <sub>B</sub>			
15	13	12	8	7						
RCL	0	0	LPID(4:0)	LCID(7:0)						

- SYSQ**      System Interrupt Queue  
SYSQ sets up the interrupt queue where system interrupt vectors will be written to. One system interrupt queue can be selected for system interrupts.
- PORTQ(2:0)**    Port Interrupt Vector Queue  
PORTQ sets up the interrupt queue where port interrupt vectors will be written to. One interrupt queue can be selected for port interrupts.
- TBE**      Test Breakout Enable  
This bit enables the test breakout function. The incoming signals of the port selected via LPID are switched to the test ports and the incoming signals on the test port replace the output signals of the selected port. Setting TBE enables the selected port (tri-state no longer active) and has priority over functions selected in register PMR and priority over bit TCOD. The port may be disabled using register REN and TEN to disable internal processing while test function is active.  
0      Disable test function.  
1      Enable test function.
- TCOD**      Transmit Clock Out Disable

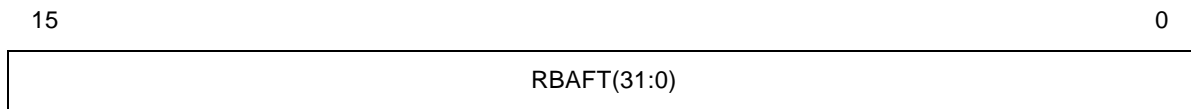
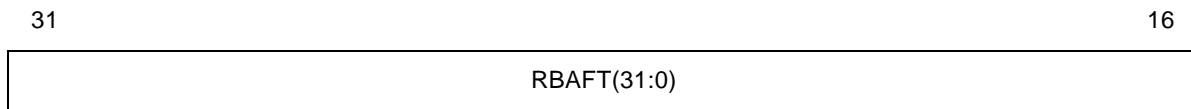
**Register Description**

	0	The incoming transmit clock of port zero is visible on pin TCLKO. This function is available when port zero is operated in unchannelized mode.
	1	Pin TCLKO is set to tri-state.
RCL		Remote Channel Loop
		The remote channel loop switches incoming data of one channel to the outgoing bit stream of the same channel. The bit rate of the receiver and the transmitter must be the same. The channel to be looped can be selected using bit field LCID. One channel at a time can be looped.
	0	Disable remote channel loop.
	1	Enable remote channel loop.
LPID		Port Identifier
		This bit field selects the port which shall be switched to the test port. See also bit CONF1.TBE.
LCID		Loop Channel Identifier
		This bit field selects the channel which shall be looped through the internal loop buffer.



**RBAFT**  
**Receive Buffer Access Failed Interrupt Threshold Register**

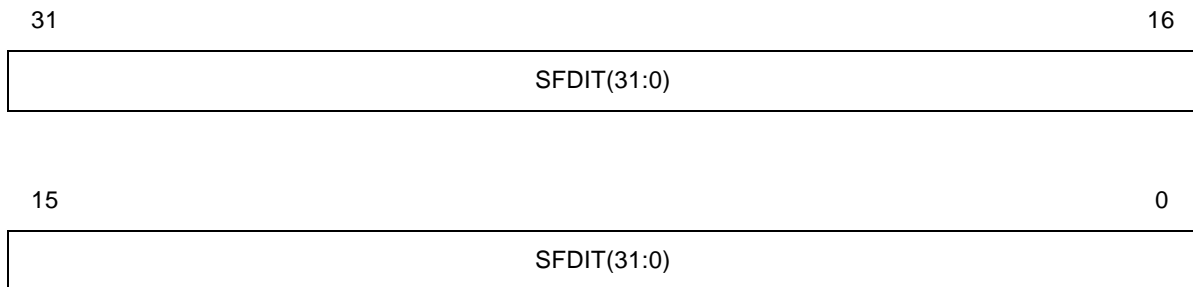
Access : read/write  
Address : 04C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**RBAFT**      Receive Buffer Access Failed Interrupt Threshold  
This register sets the threshold for the 'Receive Buffer Access Failed' interrupt vector.

**SFDT**  
**Small Frame Dropped Interrupt Threshold Register**

Access : read/write  
Address : 050<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**SFDIT**      Small Frame Dropped Interrupt Vector Threshold

The programmed threshold defines the threshold for the 'Small Frame Dropped' interrupt vector. As soon as the internal number of dropped, small frames reaches the programmed value a channel interrupt vector with bit SFD set will be generated. The actual value of dropped frames can be read using register SFDC. The value is applied to all 256 channels.





**PMR**  
**Port Mode Register**

Access : read/write  
Address : 064<sub>H</sub>  
Reset Value : 0104C000<sub>H</sub>

31				28				24				22				18				16											
PCM(3:0)				0	0	0	TBS(2:0)			0	0	0	RBS(2:0)																		
															15	14	13	12	11	10	9	8	7	6	5						0
RIM	TIM	RXF	TXR	RSF	TSF			0	0	RPL	LPL	0	0	0	0	0															

*Note: Effected port is selected via register PMIAR. All settings in this register affect the selected port only.*

- PCM**            Select Port Mode  
This bit field selects the port mode.  
0000<sub>B</sub> T1 mode (1.544 MHz)  
1000<sub>B</sub> E1 mode (2.048 MHz)  
1001<sub>B</sub> 4.096 MHz mode (16-port mode only)  
1010<sub>B</sub> 8.192 MHz mode (16-port mode only)  
1111<sub>B</sub> Unchannelized mode
- TBS**            Transmit Bit Shift  
This bit field defines the position of the transmit bits relative to the external transmit synchronization pulse. See **“Interface Timing in 16-port mode” on Page108** for interface characteristics and **Table 8-12** for correspondence between programmed value and bit shift.
- RBS**            Receive Bit Shift  
This bit field defines the position of the receive bits relative to the external receive synchronization pulse. See **“Interface Timing in 16-port mode” on Page108** for interface characteristics and **Table 8-12** for correspondence between programmed value and bit shift.

---

**Register Description**

RIM	<p>Receive Synchronization Error Interrupt Vector Mask</p> <p>This bit disables generation of the port interrupt vector receive. See <a href="#">“Port Interrupts” on Page85</a> for description of interrupt vectors.</p> <p>0      Enable</p> <p>1      Disable</p>
TIM	<p>Transmit Synchronization Error Interrupt Vector Mask</p> <p>This bit disables generation of the port interrupt vector transmit. See <a href="#">“Port Interrupts” on Page85</a> for description of interrupt vectors.</p> <p>0      Enable</p> <p>1      Disable</p>
RXF	<p>Receive Data Falling</p> <p>This bit selects the sample mode of incoming receive data. Receive data is sampled on the rising or falling edge of the incoming receive clock.</p> <p>0      Sample receive data on rising edge of receive clock.</p> <p>1      Sample receive data on falling edge of receive clock.</p>
TXR	<p>Transmit Data Rising</p> <p>This bit selects the synchronization mode for transmit data. Transmit data is updated on the rising or falling edge of the selected transmit clock.</p> <p>0      Transmit data is updated on the falling edge of the corresponding transmit clock.</p> <p>1      Transmit data is updated on the rising edge of the corresponding transmit clock.</p>
RSF	<p>Receive Synchronization Pulse Falling</p> <p>This bit selects the sample mode for incoming receive synchronization pulse. The receive synchronization pulse can be sampled on the rising or falling edge of the incoming receive clock.</p> <p>0      Sample receive synchronization pulse on the rising edge of the receive clock.</p> <p>1      Sample receive synchronization pulse on the falling edge of the receive clock.</p>

Register Description

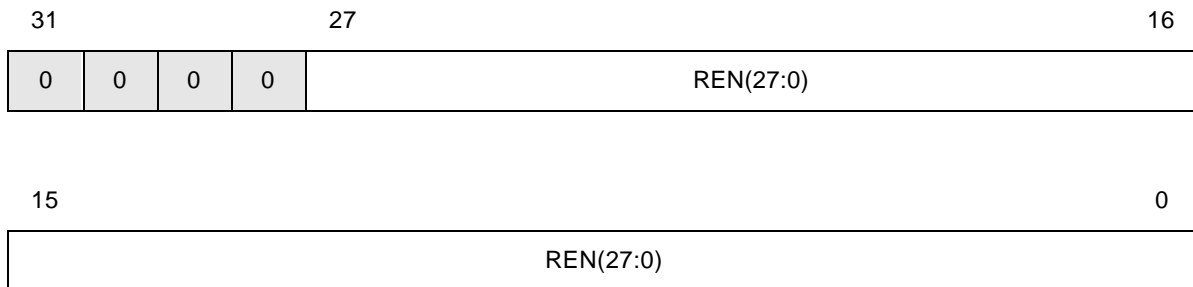
TSF	<p>Transmit Synchronization Pulse Falling</p> <p>This bit selects the sample mode for incoming transmit synchronization pulse. The transmit synchronization pulse can be sampled on the rising or falling edge of the selected transmit clock.</p> <p>0      Sample transmit synchronization pulse on the rising edge of the selected transmit clock.</p> <p>1      Sample transmit synchronization pulse on the falling edge of the selected transmit clock.</p>
RPL	<p>Remote Payload Loop</p> <p>This bit enables the remote payload loop of the selected port.</p> <p>0      Disable remote payload loop.</p> <p>1      Enable remote payload loop.</p>
LPL	<p>Local Port Loop</p> <p>This bit enables the local port loop on the selected port. When local loops are closed, the corresponding transmit clock and the synchronization pulse is switched to the receive port.</p> <p><i>Note: The transmit bit shift (PMR.TBS) and the receive bit shift (PMR.RBS) of the selected port must be identical.</i></p> <p>0      Disable local port loop.</p> <p>1      Enable local port loop.</p>

**Table 8-12 Bit Shift**

Programmed value	Bit shift
000 <sub>B</sub>	-4
001 <sub>B</sub>	-3
010 <sub>B</sub>	-2
011 <sub>B</sub>	-1
100 <sub>B</sub>	0
101 <sub>B</sub>	1
110 <sub>B</sub>	2
111 <sub>B</sub>	3

**REN**  
**Receive Enable Register**

Access : read/write  
Address : 068<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



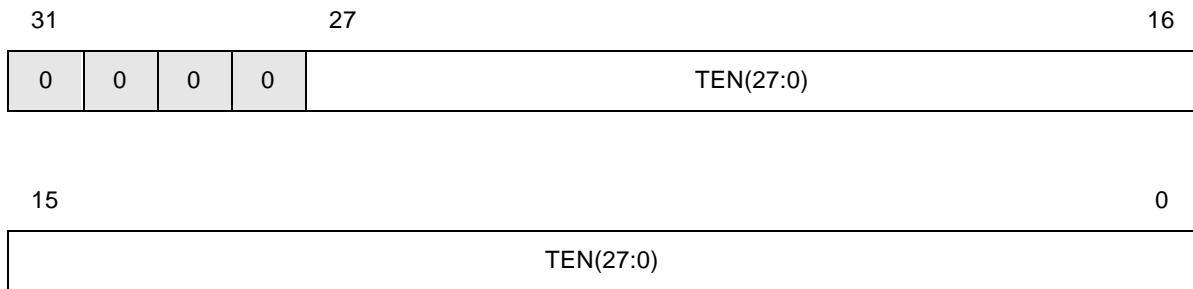
**REN**      Receive Enable

Setting a bit in this bit field enables the receive function of the selected port. After reset all ports are disabled and thus all incoming receive data is discarded. While a port is disabled communication between port handler, time slot assigner and synchronization function is disabled. A port should be enabled if it is correctly configured using registers PMIAR and PMR.

- 0      Disable receive port.
- 1      Enable receive port.

**TEN**  
**Transmit Enable Register**

Access : read/write  
Address : 06C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**TEN**      **Transmit Enable**

This bit field enables the transmit function of the selected port. After reset all transmit ports are disabled and thus all TD lines are set to tri-state. While a port is reset the communication between port handler, time slot assigner and synchronization function is disabled. After the port mode has been selected using register PMIAR and PMR a transmit port can be enabled.

- 0      Disable transmit port.
- 1      Enable transmit port. Bits which are masked in the time slot mask register are tri-stated.



---

**Register Description**

**TSNUM**      Time Slot Number

This bit field selects the time slots, which can be accessed via register TSAIA.

Valid time slot numbers are:

- 0..23    T1, Unchannelized
- 0..31    E1
- 0..63    4.096 MHz Channelized
- 0..127   8.192 MHz Channelized





---

**Register Description**

**MASK**

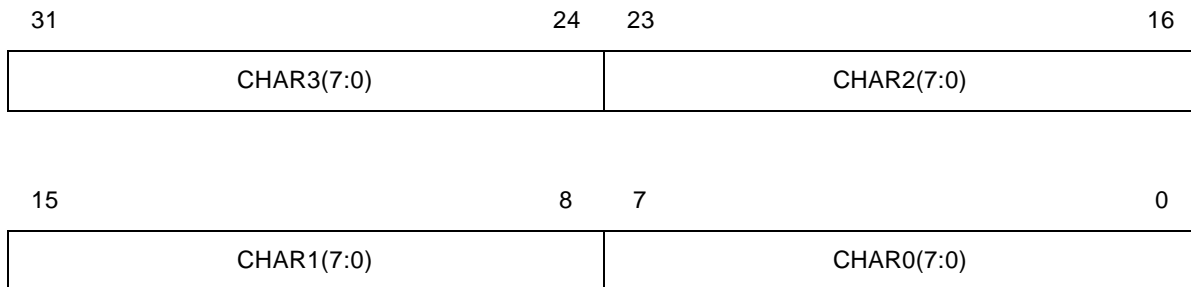
**Mask Bits**

Setting a bit in this bit field selects the corresponding bit in a time slot which is enabled for operation.

- 0 In receive direction the corresponding bit is discarded. In transmit direction the bit is tri-stated.
- 1 In receive direction the corresponding bit is forwarded to the protocol machine (via time slot assigner). In transmit direction data on the serial line is generated by the protocol machine.

**REC\_ACCMX**  
**Receive Extended ACCM Map Register**

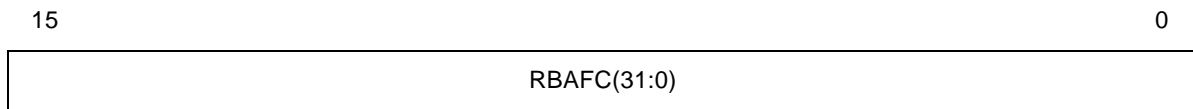
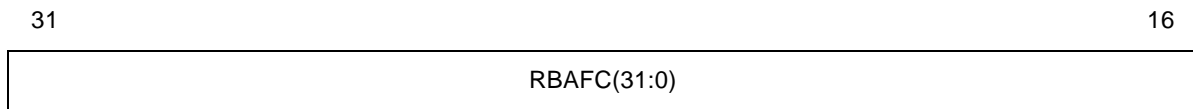
Access : read/write  
Address : 080<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



This register is only used by channels operated in octet synchronous PPP mode. A character written to this register is mapped with a control escape sequence, if the corresponding enable flag is set in the corresponding bit CSPEC\_MODE\_REC.ACCMX(3:0).

**RBAFC**  
**Receive Buffer Access Failed Counter Register**

Access : read  
Address : 084<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

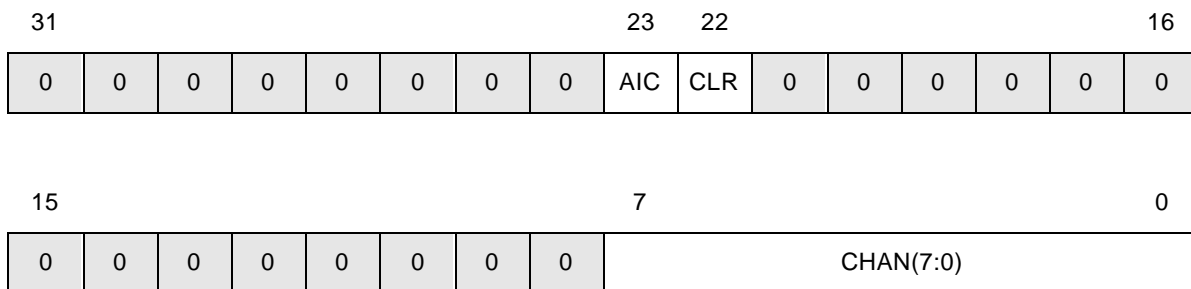


**RBAFC**      **Receive Buffer Access Failed Counter**

The read value of this register defines the number of packets which have been discarded due to inaccessibility of the internal receive buffer. A read access resets the counter to zero.

**SFDIA**  
**Small Frame Dropped Indirect Access Register**

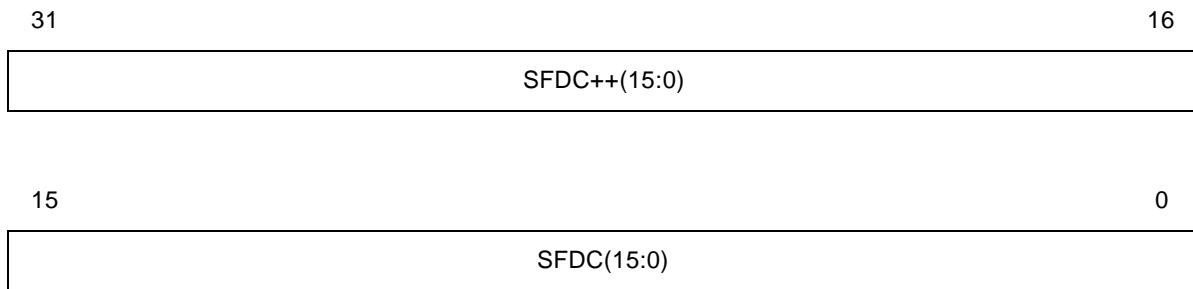
Access : read/write  
Address : 088<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



- AIC**      Auto Increment Channel  
This bit enables the auto increment function of bit field CHAN. Each read/write access to register SFD increments CHAN by two. This allows to read the status of multiple channels without accessing SFDIA again.  
0      Disable auto increment function.  
1      Enable auto increment function.
- CLR**      Clear  
This bit enables the counter mode on reads to register SFDC.  
0      Read of register SFDC does not affect the small frame dropped counter.  
1      After reading register SFDC the value of the small frame dropped counter will be reset to zero.
- CHAN**      Channel Number  
This bit field selects the channel, whose status can be read in register SFDC.  
0..255 Channel number

**SFDC**  
**Small Frame Dropped Counter Register**

Access : read  
Address : 08C<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

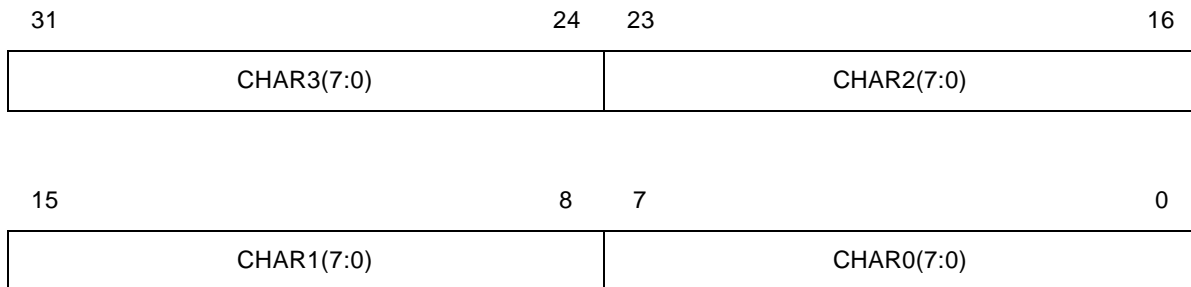


These both bit fields show the current value of the small frame dropped counter of the channel N and N+1 selected via SFDIA.CHAN. Dependent on bit field SFDIA.CLR the counter will be cleared after they are read.

- SFDC++      Small Frame Dropped Counter for Channel N+1  
                    The number of dropped, small frames of channel SFDIA.CHAN+1.
- SFDC         Small Frame Dropped Counter  
                    The number of dropped, small frames of channel SFDIA.CHAN.

**XMIT\_ACCMX**  
**Transmit Extended ACCM Map**

Access : read/write  
Address : 090<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



This register is only used by a channel in octet synchronous PPP mode. A character written to this register will be mapped with a Control Escape sequence, if the corresponding enable flag is set in the CSPEC\_MODE\_XMIT register (ACCMX(3:0)).







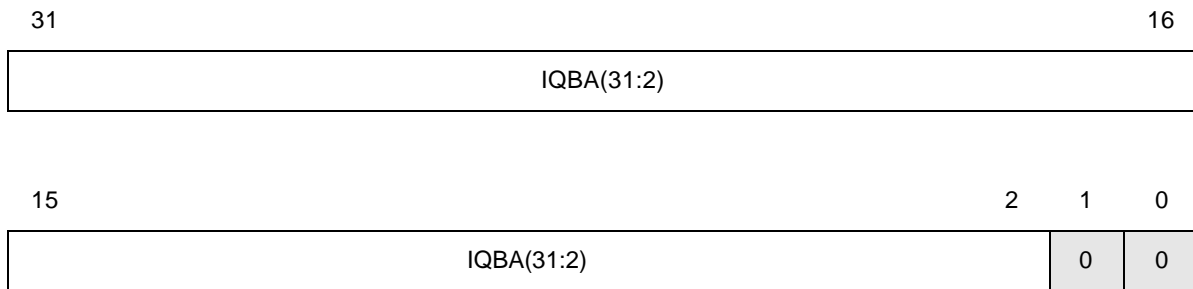


Register Description

SIQL	<p>Set Interrupt Queue Length</p> <p>This bit field enables setup of the interrupt queue length of queue Q. The value to be programmed has to be configured via register IQL prior to a write access to this bit.</p> <p>0      No operation</p> <p>1      Set interrupt queue length.</p>
SIQBA	<p>Set Interrupt Queue Base address</p> <p>This bit field enables setup of the interrupt queue base address of queue Q. The value to be programmed has to be configured via register IQBA prior to a write access to this bit.</p> <p>0      No operation</p> <p>1      Update interrupt queue base address with value programmed in register IQBA.</p>
Q	<p>Interrupt Queue Number</p> <p>This bit field determines the interrupt queue number for which programming is valid. The first eight (0..7) interrupt queues are used for channel, port and system interrupt vectors, while the last interrupt queue (8) is used for command interrupt vectors. Interrupt queue number seven is per default the high priority interrupt queue.</p> <p>System software may setup the interrupt queue high priority mask, the interrupt queue length and the interrupt queue base address simultaneously by setting SIQL, SIQBA and SIQM.</p> <p>The command interrupt queue has a fixed length of two times 256 DWORDs, that is one DWORD for each interrupt vector.</p> <p>It is possible to setup the interrupt queue high priority mask, the interrupt queue length and the interrupt queue base address concurrently by setting SIQBA, SIQL and SIQM to '1'.</p> <p><i>Note: Programming of interrupt queue length or interrupt queue high priority mask is not valid for the command interrupt queue (interrupt queue 8).</i></p> <p><i>Note: Programming of interrupt queue high priority mask is not valid for the high priority interrupt queue (interrupt queue 7).</i></p> <p>0..8    Interrupt Queue</p>

**IQBA**  
**Interrupt Queue Base Address Register**

Access : read/write  
Address : 0E4<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

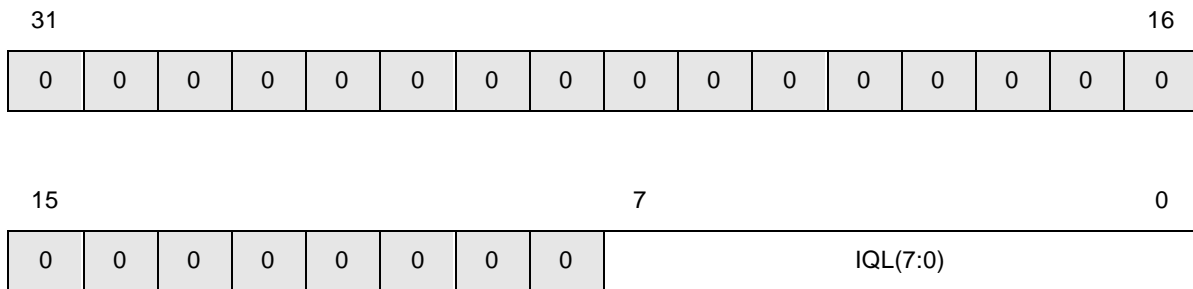


**IQBA**      Interrupt Queue Base Address

The interrupt queue base address register assigns a base address to the eight channel interrupt queues and the command interrupt queue. To set a new base address for a specific queue, system software must first program IQBA. Afterwards the value is released by selecting the associated queue via bit field IQIA.Q and setting of bit IQIA.SIQBA. The interrupt queue base address has to be DWORD aligned. Whenever the base address of a particular interrupt queue is modified, the next interrupt vector written to that queue is stored in the first location of the queue.

**IQL**  
**Interrupt Queue Length Register**

Access : read/write  
Address : 0E8<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>



**IQL**                      Interrupt Queue Length

This bit field assigns a interrupt queue length to the eight channel interrupt queues. To set the interrupt queue length of a specific queue, system software must first program IQL. Afterwards the value is released by selecting the associated queue via bit field IQIA.Q and setting of bit IQIA.SIQL. IQL specifies the interrupt queue length L (number of DWORDs) in the shared memory with  $L=(IQL+1)*16$  (maximum of 4092 DWORDs).

*Note: IQL = 255 equals a queue length of 1 DWORD.*

Whenever the length of a particular interrupt queue is modified, the next interrupt vector written to that queue is stored in the first location of the queue.

## IQMASK Interrupt Queue High Priority Mask

Access : read/write  
Address : 0EC<sub>H</sub>  
Reset Value : 00000000<sub>H</sub>

31	30		28					23	22						16
THI	TAB	0	HTAB	0	0	0	0	UR	TFE	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5		3	2		0
RHI	RAB	RFE	HRAB	MFL	RF0D	CRC	ILEN	RFOP	SF	IFTC	0	SFD	SD	0	0

For a description of the interrupt concept and interrupt vectors see [Chapter4.7.1](#).

In normal operation each channel interrupt vector is written to the interrupt queue associated with a specific channel, that is interrupt queue 0 to 7. The interrupt queue mask provides the functionality to forward selected channel interrupts to the high priority interrupt queue, which is hardwired as queue 7. Therefore a mask can be set for each of the interrupt queues, which specifies the channel interrupt vector to be forwarded to the high priority interrupt queue. To set the IQMASK for interrupt queues 0 to 6, system software must first program IQMASK. Afterwards the mask is released by selecting the affected interrupt queue via bit field IQIA.Q and setting of bit SIQM.

Those interrupt vectors which have an interrupt bit set, that is also masked in this high priority mask are forwarded to the high priority interrupt queue instead of the regular interrupt queue associated with a specific channel.

If a channel interrupt vector has at least one interrupt bit set, that is also masked in the high priority mask, the interrupt vector will be forwarded to the high priority interrupt queue.

In case that a channel interrupt vector has at least one bit set, that is not masked in the high priority mask, the interrupt vector is queued into the regular interrupt queue associated with the corresponding channel.



Register Description

as  $\overline{\text{LINT}}$  changes from an inactive to an active state the interrupt pin INTA will be asserted.

*Note: This bit does not clear by writing a '1'. This bit is set as long as the interrupt pin  $\overline{\text{LINT}}$  is asserted.*

0  $\overline{\text{LINT}}$  not asserted.

1  $\overline{\text{LINT}}$  asserted.

IF Interrupt FIFO

This bit indicates that there is an interrupt vector stored in the internal interrupt FIFO. The IF interrupt is available if the interrupt pin  $\overline{\text{LINT}}$  is switched to input mode (INTCTRL.ID = '1') and when the interrupt mask GMASK.IF is set to '0'.

*Note: This bit does not clear by writing a '1'. This bit is set as long as an interrupt vector is stored in the interrupt FIFO.*

0 No Interrupt vector in interrupt FIFO.

1 Interrupt vector stored in internal interrupt FIFO.

Q8..Q0 Interrupt Queue 8..0

On reads each bit flags one or more interrupt vectors that have been written to the corresponding interrupt queue. If one of the bits is set and the same bit is not masked in register GMASK, the interrupt pin INTA will be asserted. A bit is cleared, when an '1' is written to the specific bit.

0 No interrupt vector written.

1 Read: One or more interrupt vectors have been written to interrupt queue.

Write: Clear bit



**GMASK**  
**Global Interrupt Mask Register**

Access : read/write  
Address : 0F4<sub>H</sub>  
Reset Value : FFFFFFFF<sub>H</sub>

31													17	16	
INTOF	1	1	1	1	1	1	1	1	1	1	1	1	1	LINT	IF
							8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	Q8	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0

Each bit in this register mask the interrupts, which are flagged in register GISTA/GIACK.

- INTOF** Mask Interrupt Overflow  
This bit masks the interrupt overflow interrupt.
- LINT** Local Bus Interrupt  
This bit masks bridging of interrupt from the local bus to the PCI bus.  
0 Bridging of  $\overline{\text{LINT}}$  to  $\overline{\text{INTA}}$  enabled.  
1 Bridging of  $\overline{\text{LINT}}$  to  $\overline{\text{INTA}}$  disabled.
- IF** Interrupt FIFO  
This bit masks the internal mailbox/layer one interrupt FIFO.  
0 IF interrupt is enabled.  
1 IF interrupt is disabled.
- Q8..Q0** Mask Interrupt Queue 8..0  
Each of the bits Q8..Q0 masks an interrupt, which will be asserted, when an interrupt vector has been written to the corresponding interrupt queue 8..0. Masking an interrupt does not suppress generation of the interrupt vector itself.  
0 Enable interrupt, when interrupt vector has been written to selected interrupt queue.  
1 Mask (Disable) interrupt, when interrupt vector has been written to selected interrupt queue.

## 8.8.1 PCI and Local Bus Slave Register Set

### FCONF Configuration Register

Access : read/write  
Address : 100<sub>H</sub> (PCI), 00<sub>H</sub> (Local Bus)  
Reset Value : 8080<sub>H</sub>

15	14							7	6	5	4	3	2	1	0
IIP	0	0	0	0	0	0	0	MBID	WSE	BSD	P28	P18	P08	LAE	LME

- IIP** Initialization in Progress (Read Only)  
After reset (hardware reset or software reset) the internal RAM's are self initialized by the MUNICH256. During this time (approx. 250 μs) no other accesses to the device than reading register CONF1 or FCONF are allowed. This bit must be polled until it has been deasserted by the MUNICH256.
- 0 Self initialization has finished.
  - 1 Self initialization in progress.
- MBID** Mailbox Interrupt Vector Disable
- 0 Enable generation of mailbox interrupt vectors. As soon as system software on PCI side writes to register MBP2E0 an interrupt vector indicating a mailbox interrupt will be forwarded to the internal interrupt FIFO and can be read by the local CPU.
  - 1 Disable generation of mailbox interrupt vectors.
- WSE** Wait State Enable  
This bit enables the wait state controlled master mode.
- 0  $\overline{\text{LRDY}}$  (Intel),  $\overline{\text{LDTACK}}$  (Motorola) controlled bus mode.
  - 1 Wait state controlled bus mode. Wait states are defined in register MTIMER.WS.

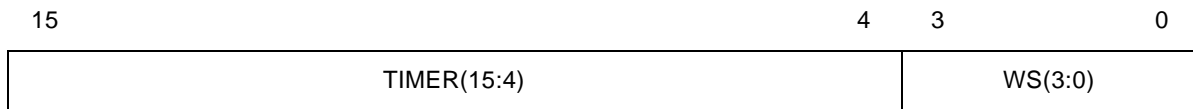
---

**Register Description**

BSD	<b>Byte Swap Disable</b> This bit disables byte swapping on 16-bit transfers when the local bus is operated in Motorola master mode. 0      Enable byte swap. 1      Disable byte swap.
P28..P08	<b>Switch Page 2..0 to 8-bit mode</b> The MUNICH256 maps three pages of 8 kByte each to the local bus in master mode. Each page accessed from the PCI side can be mapped in 8-bit mode or 16-bit mode. In 8-bit mode the data bits LD(15:8) are unused. 0      Set page mode to 16-bit mode. 1      Set page mode to 8-bit mode.
LAE	<b>Local Bus Arbiter Enable</b> This bit enables the local bus arbiter. In case that the local bus arbiter is enabled the MUNICH256 will arbitrate for each bus access on the local bus using the arbitration signals. If local bus arbiter functionality is disabled it assumes bus ownership and does not arbitrate for the local bus. 0      Disable the local bus arbiter. 1      Enable the local bus arbiter.
LME	<b>Local Bus Master Enable</b> This bit enables the local bus master functionality. As long as the local bus master functionality is disabled the MUNICH256 can be accessed from the local bus as slave only. 0      Disable Local Bus Master. 1      Enable Local Bus Master.

**MTIMER**  
**Master Local Bus Timer Register**

Access : read/write  
Address : 104<sub>H</sub> (PCI), 02<sub>H</sub> (Local Bus)  
Reset Value : 0000<sub>H</sub>



**TIMER**      Local Bus Latency Timer

TIMER\*16 determines the time in clock cycles the MUNICH256 holds the local bus as bus master after it was granted the bus. It holds the bus as long as the first transaction is in progress or the latency timer is counting. In case that the MUNICH256 shall release the bus after it each transaction the latency TIMER value must be set to zero.

**WS**            Wait State Timer

The value of this register determines the time in clock cycles the MUNICH256 asserts  $\overline{\text{LRD}}$ ,  $\overline{\text{LWR}}$  (Intel Mode) respectively  $\overline{\text{LDS}}$  (Motorola Bus Mode). See also FCONF.WSE.

**INTCTRL**  
**Interrupt Control Register**

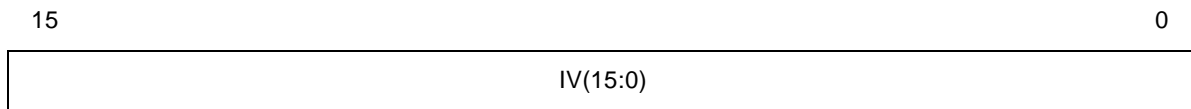
Access : read/write  
Address : 108<sub>H</sub> (PCI), 04<sub>H</sub> (Local Bus)  
Reset Value : 0001<sub>H</sub>

15													3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	ID	IP	CLIQ	IM

- ID**            Interrupt Direction  
This pin determines the direction of the interrupt pin  $\overline{\text{LINT}}$ .  
0         $\overline{\text{LINT}}$  is output.  
1         $\overline{\text{LINT}}$  is input.
- IP**            Interrupt Polarity  
0         $\overline{\text{LINT}}$  is active low.  
1         $\overline{\text{LINT}}$  is active high.
- CLIQ**        Clear Interrupt Queue  
Setting this bit will clear the internal interrupt FIFO. This effects mailbox interrupts to the local bus.  
0        No action  
1        Clear interrupt FIFO.
- IM**            Interrupt Mask  
This bit masks assertion of the pin  $\overline{\text{LINT}}$  when interrupts are stored in the internal interrupt FIFO. If the interrupt direction bit is set to output mode interrupt are flagged at interrupt pin  $\overline{\text{LINT}}$ . If the interrupt direction is set to input mode interrupts are flagged at pin  $\overline{\text{INTA}}$ .  
0        Enable assertion of interrupt pin  $\overline{\text{LINT}}$ .  
1        Disable assertion of interrupt pin  $\overline{\text{LINT}}$ .

**INTFIFO**  
**Interrupt FIFO**

Access : read  
Address : 10C<sub>H</sub> (PCI), 06<sub>H</sub> (Local Bus)  
Reset Value : FFFF<sub>H</sub>

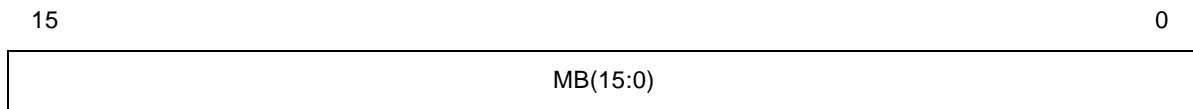


IV                    Interrupt Vector

After the MUNICH256 asserted interrupt pin  $\overline{\text{LINT}}$  on the local bus side, this bit field contains an interrupt vector containing interrupt information. Please refer to section **“Mailbox Interrupts to the Local Bus” on Page94** for a detailed description of interrupt vector contents.

**MBE2P0**  
**Mailbox Local Bus to PCI Command Register**

Access : read/write  
Address : 140<sub>H</sub> (PCI), 20<sub>H</sub> (Local Bus)  
Reset Value : 0000<sub>H</sub>

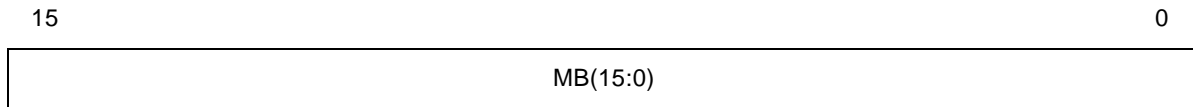


**MB** Mailbox Data register

This register can be written and read from local bus side. From PCI side this register should be used as read only in order to allow stable interprocessor communication. Write access to this register results in mailbox interrupt vectors on local bus side to the internal interrupt FIFO when FCONF.MBID is set to '0'.

**MBE2P1-7**  
**Mailbox Local Bus to PCI Data Register 1-7**

Access : read/write  
Address : 144<sub>H</sub>-15C<sub>H</sub> (PCI), 22<sub>H</sub>-2E<sub>H</sub> (Local Bus)  
Reset Value : 0000<sub>H</sub>

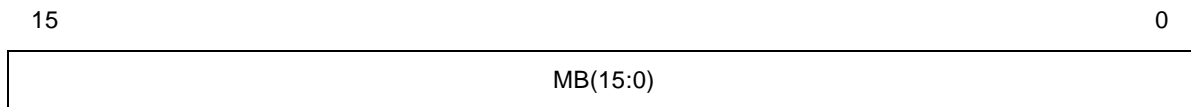


**MB** Mailbox Data register  
This register can be written and read from local bus side. From PCI side this register should be used as read only in order to allow stable interprocessor communication.



**MBP2E0**  
**Mailbox PCI to Local Bus Status Register**

Access : read/write  
Address : 160<sub>H</sub> (PCI), 30<sub>H</sub> (Local Bus)  
Reset Value : 0000<sub>H</sub>

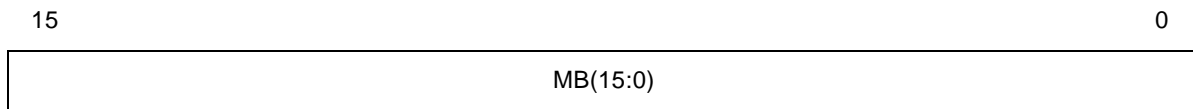


**MB** Mailbox Status Register

This register can be written and read from PCI side. From local bus side this register should be used as read only in order to allow stable interprocessor communication. Write access to this register results in mailbox interrupt vectors to PCI side when CONF1.MBIM is set to '0'.

**MBP2E1-7**  
**Mailbox PCI to Local Bus Data Register 1-7**

Access : read/write  
Address : 164<sub>H</sub>-17C<sub>H</sub> (PCI), 32<sub>H</sub>-3E<sub>H</sub> (Local Bus)  
Reset Value : 0000<sub>H</sub>



**MB** Mailbox Data Register  
This register can be written and read from PCI side. From local bus side this register should be used as read only in order to allow stable interprocessor communication.

Note - It is recommended to set this register to 00 after reset for normal operation.

## 9 Electrical Characteristics

### 9.1 Important Electrical Requirements

Both  $V_{DD3}$  and  $V_{DD25}$  can take on any power-on sequence. Within 50 milliseconds of power-up the voltages must be within their respective absolute voltage limits. At power-down, within 50 milliseconds of either voltage going outside its operational range, both voltages must be returned below 0.1V.

### 9.2 Absolute Maximum Ratings

Table 9-1 Absolute Maximum Ratings

Parameter	Symbol	Limit Values		Unit
		min	max	
Ambient temperature under bias PEB 20256 E PEF 20256 E	$T_A$	0 -40	70 85	°C
Junction temperature under bias	$T_J$		125	°C
Storage temperature	$T_{stg}$	-65	125	°C
Voltage on any pin with respect to ground	$V_S$	-0.5	$V_{DD3}+0.5$	V

*Note: Stresses above those listed here may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

### 9.3 DC Characteristics

#### a) Power Supply Pins

Table 9-2 DC Characteristics

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Core Supply Voltage	$V_{DD25}$	2.25	2.75	V	
I/O Supply Voltage	$V_{DD3}$	3.0	3.6	V	

## Electrical Characteristics

Parameter		Symbol	Limit Values		Unit	Test Condition
			min.	max.		
Core supply current $V_{DD25}$	operationa l	$I_{CC25}$		< 350	mA	
	power down (no clocks)	$I_{CCPD25}$		< 2	mA	
I/O supply current $V_{DD3}$	operationa l	$I_{CC3}$		< 200	mA	Inputs at $V_{SS}/V_{DD3}$ No output loads.
	power down (no clocks)	$I_{CCPD3}$		< 2	mA	
Sum of Input leakage current and Output leakage current (Outputs Hi-z)		$I_{LI}$ $I_{LO}$		< 10	$\mu$ A	
Power Dissipation		$P$		< 3	W	

**b) Non-PCI Interface Pins**
**Table 9-3 DC Characteristics (Non-PCI Interface Pins)**
 $T_A = -40 \text{ to } 85^\circ\text{C}$ ,  $V_{DD3} = 3.3 \text{ V} \pm 0.3 \text{ V}$ ,  $V_{DD25} = 2.5 \text{ V} \pm 0.25 \text{ V}$ ,  $V_{SS} = 0 \text{ V}$ 

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
L-input voltage	$V_{IL}$	-0.4	0.8	V	
H-input voltage	$V_{IH}$	2.0	$V_{DD3}+0.4$	V	
L-output voltage	$V_{OL}$		0.45	V	$I_{QL} = 2 \text{ mA}$
H-output voltage	$V_{OH}$	2.4		V	$I_{QH} = -400 \mu\text{A}$

c) PCI Interface Pins

**Table 9-4 DC Characteristics (PCI Interface Pins)**

$T_A = -40$  to  $85$  °C,  $V_{DD3} = 3.3$  V  $\pm$  0.3 V,  $V_{DD25} = 2.5$  V  $\pm$  0.25 V,  $V_{SS} = 0$  V

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
L-input voltage	$V_{IL}$	-0.5	$0.3V_{DD3} - 80\text{mV}$	V	
H-input voltage	$V_{IH}$	$0.5V_{DD3}$	$V_{DD3}+0.5$	V	
L-output voltage	$V_{OL}$		$0.1V_{DD3}$	V	$I_{QL} = 1500 \mu\text{A}$
H-output voltage	$V_{OH}$	$0.9V_{DD3}$		V	$I_{QH} = -500 \mu\text{A}$

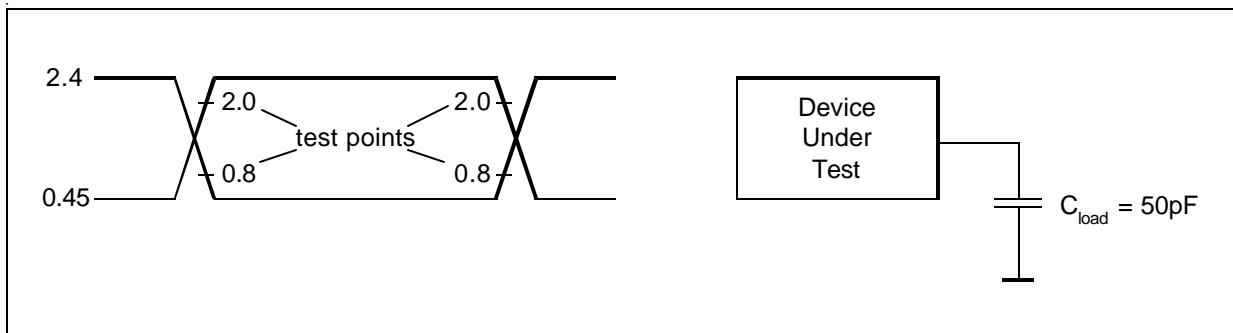
**9.4 AC Characteristics**

a) Non-PCI interface pins

$T_A = -40$  to  $85$  °C,  $V_{DD3} = 3.3$  V  $\pm$  0.3 V,  $V_{DD25} = 2.5$  V  $\pm$  0.25 V,  $V_{SS} = 0$  V

Inputs are driven to 2.4 V for a logical '1' and to 0.4 V for a logical '0'. Timing measurements are made at 2.0 V for a logical '1' and at 0.8 V for a logical '0'.

The AC testing input/output waveforms are shown below.



**Figure 9-1 Input/Output Waveform for AC Tests**

b) PCI interface pins

PCI interface pins are measured as pins compliant to the 3.3V signalling environment according to the PCI Specification Rev. 2.1.

### 9.4.1 PCI Bus Interface Timing

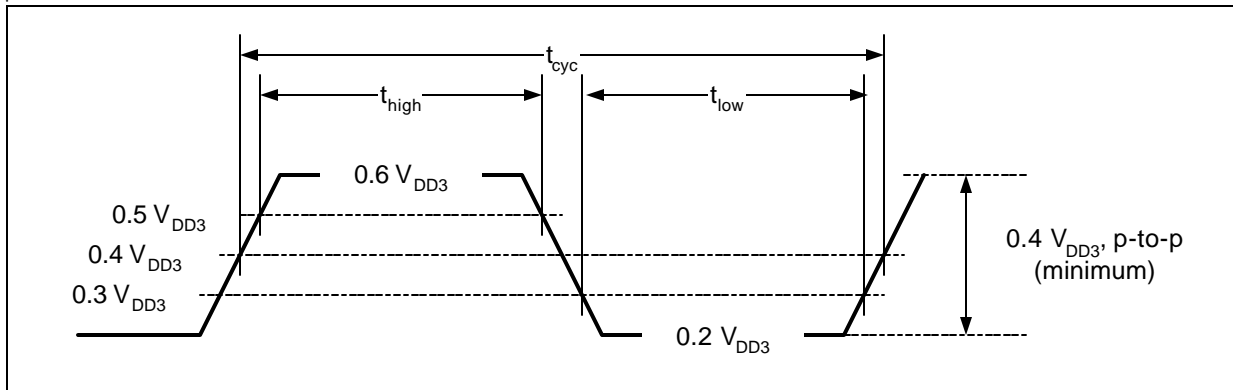


Figure 9-2 PCI Clock Cycle Timing

Table 9-5 PCI Clock Characteristics

Parameter	Symbol	Limit Values		Unit
		min.	max.	
CLK cycle time	$t_{cyc}$	15		ns
CLK high time	$t_{high}$	6		ns
CLK low time	$t_{low}$	7		ns
CLK slew rate (see note)		1.5	4	V/ns

Note: Rise and fall times are specified in terms of the edge rate measured in V/ns. This slew rate must be met across the minimum peak-to-peak portion of the clock waveform shown in [Figure 9-3](#).

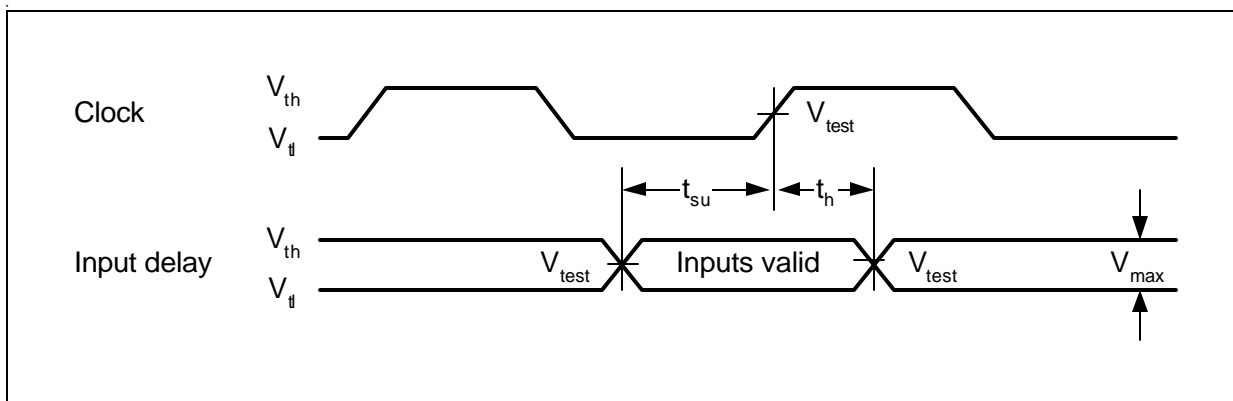


Figure 9-3 PCI Input Timing Measurement Conditions

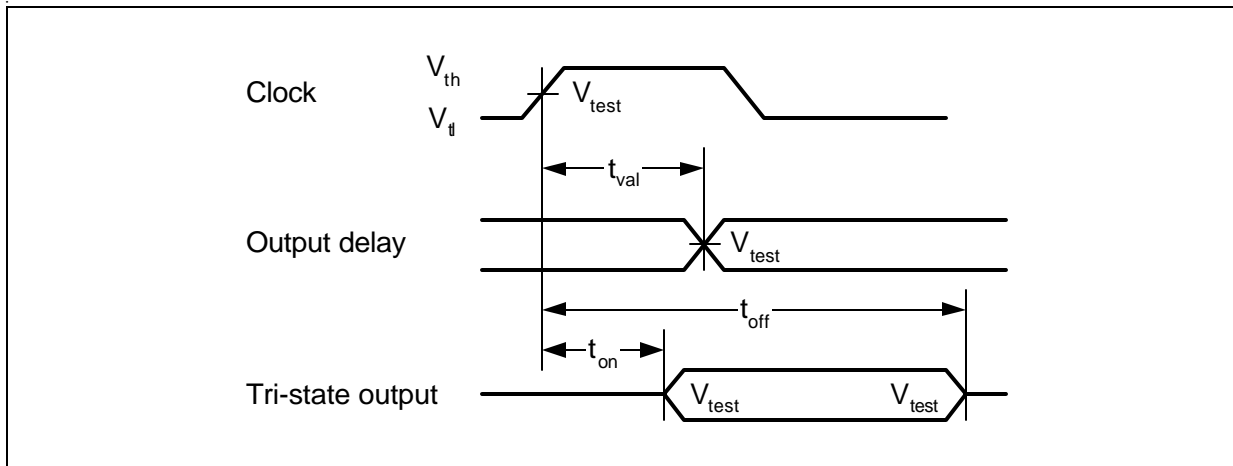


Figure 9-4 PCI Output Timing Measurement Conditions

Table 9-6 PCI Interface Signal Characteristics

Parameter	Symbol	Limit Values		Unit	Notes
		min.	max.		
CLK to signal valid - bussed signals	$t_{val}$	2	8	ns	1, 2
CLK to $\overline{REQ}$ valid	$t_{val}$	2	7	ns	1, 2
Float to active delay	$t_{on}$	2		ns	
Active to float delay	$t_{off}$		14		
Input setup time to CLK - bussed signals	$t_{su}$	4			2
Input setup time to CLK - $\overline{GNT}$	$t_{su}$	5			2
Input hold time from CLK	$t_h$	0.5			

Note:

1. Minimum times are measured for 3.3V signalling environment according to the PCI Specification Rev. 2.1.
2.  $\overline{REQ}$  and  $\overline{GNT}$  are point-to-point signals. All other signals are bussed.

### 9.4.2 SPI Interface Timing

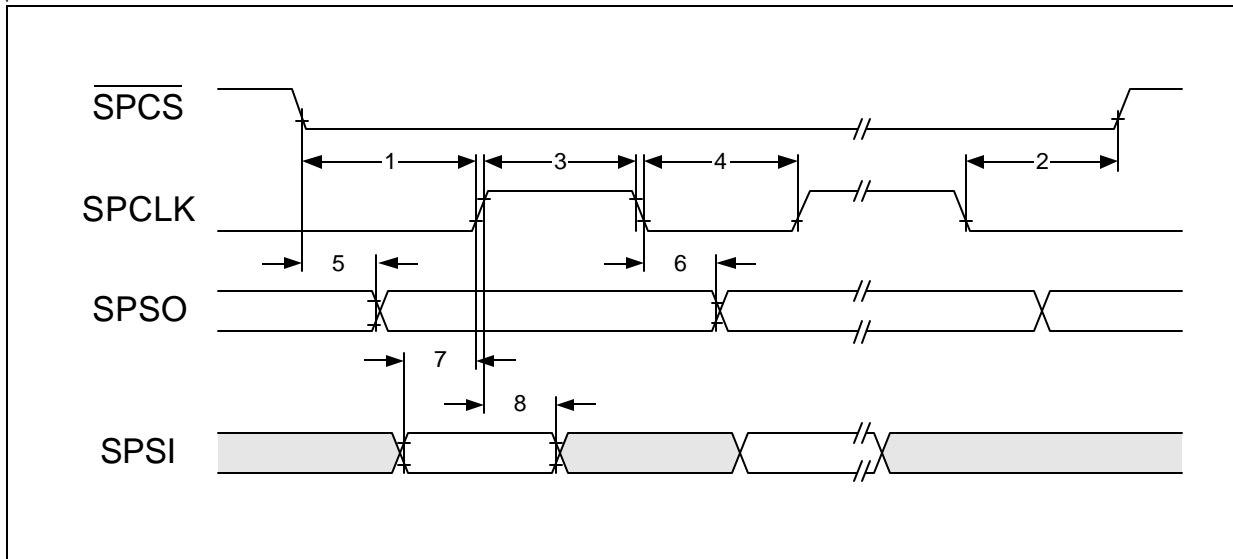


Figure 9-5 SPI Interface Timing

Table 9-7 SPI Interface Timing

No.	Parameter	Limit Values		Unit	Notes
		min.	max.		
1	SPCLK delay after $\overline{\text{SPCS}}$ low	500		ns	1
2	SPCLK delay before $\overline{\text{SPCS}}$ high	500		ns	
3	SPCLK high time	500		ns	
4	SPCLK low time	500		ns	
5	SPSO delay after $\overline{\text{SPCS}}$ low		100	ns	
6	SPSO delay after SPCLK		100	ns	
7	SPSI setup time before SPCLK	100		ns	
8	SPSI hold time after SPCLK	100		ns	

Note:

- 1 SPI clock is related to PCI clock where the SPI frequency is 1/78 of the PCI frequency. All timings for SPI interface are calculated with a PCI clock running at 33 MHz.



### 9.4.3 Local Microprocessor Interface Timing

#### 9.4.3.1 Intel Bus Interface Timing (Slave Mode)

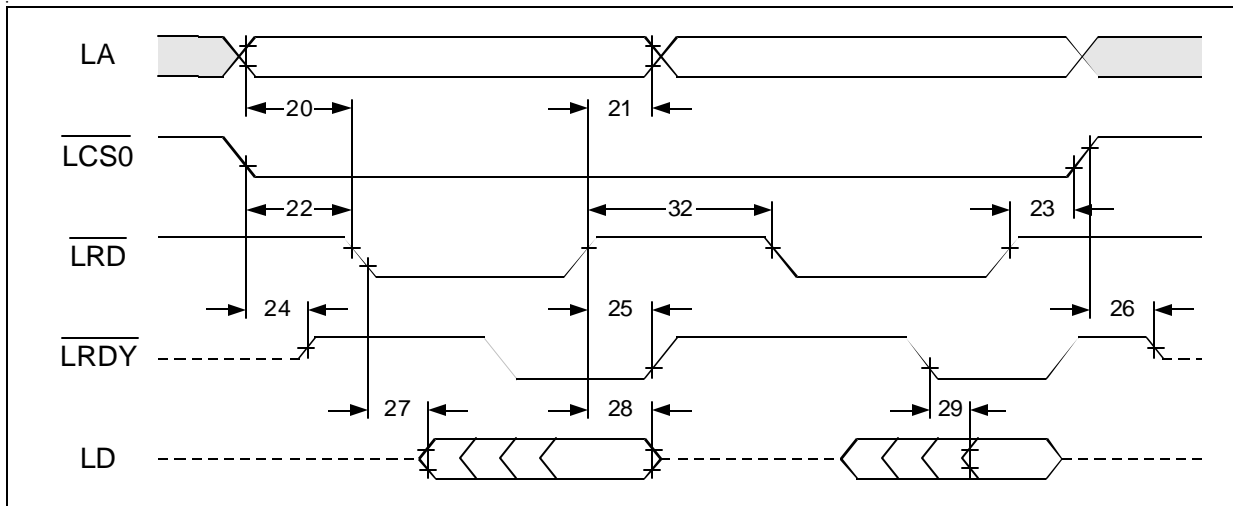


Figure 9-6 Intel Read Cycle Timing (Slave Mode)

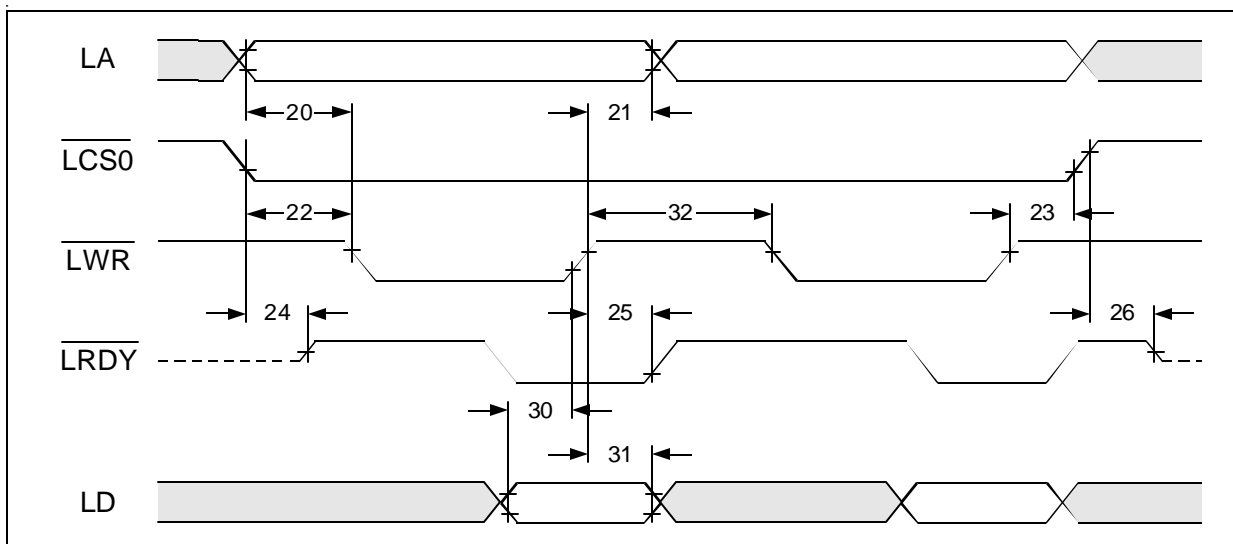


Figure 9-7 Intel Write Cycle Timing (Slave Mode)

**Table 9-8 Intel Bus Interface Timing**

No.	Parameter	Limit Values		Unit
		min.	max.	
20	LA to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ setup time	20		ns
21	LA to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ hold time	0		ns
22	$\overline{\text{LCS0}}$ to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ setup time	20		ns
23	$\overline{\text{LCS0}}$ to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ hold time	0		ns
24	$\overline{\text{LCS0}}$ low to $\overline{\text{LRDY}}$ active delay		20	ns
25	$\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ high to $\overline{\text{LRDY}}$ high delay		20	ns
26	$\overline{\text{LCS0}}$ high to $\overline{\text{LRDY}}$ float delay		20	ns
27	$\overline{\text{LRD}}$ low to LD active delay		20	ns
28	$\overline{\text{LRD}}$ high to LD float delay		20	ns
29	$\overline{\text{LRDY}}$ low to LD valid delay		20	ns
30	LD to $\overline{\text{LWR}}$ setup time	20		ns
31	LD to $\overline{\text{LWR}}$ hold time	0		ns
32	$\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ minimum high time	20		ns

### 9.4.3.2 Intel Bus Interface Timing (Master Mode)

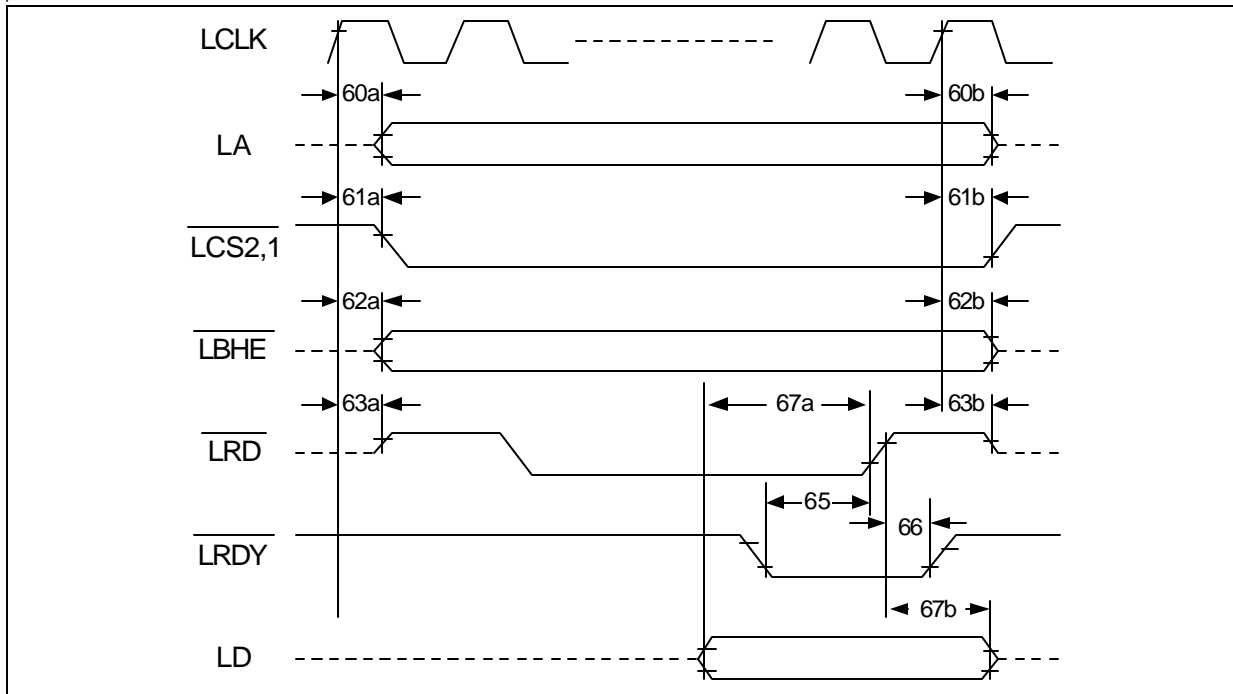


Figure 9-8 Intel Read Cycle Timing (Master Mode, LRDY controlled)

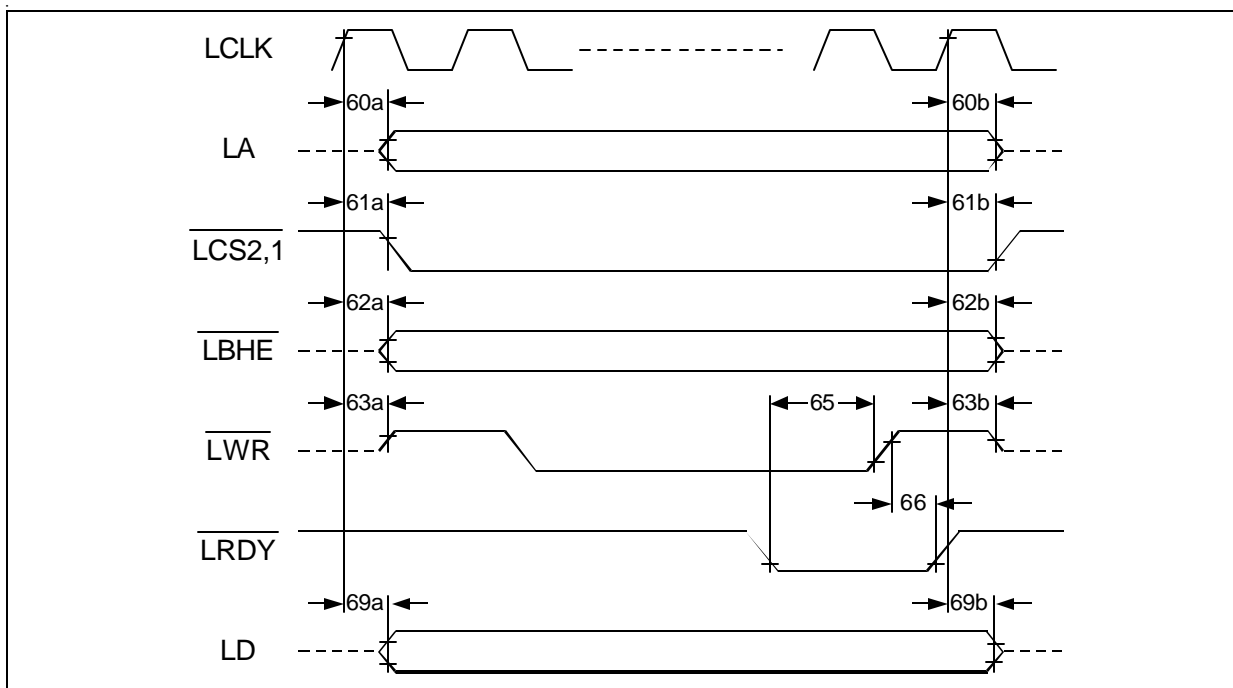


Figure 9-9 Intel Write Cycle Timing (Master Mode, LRDY controlled)

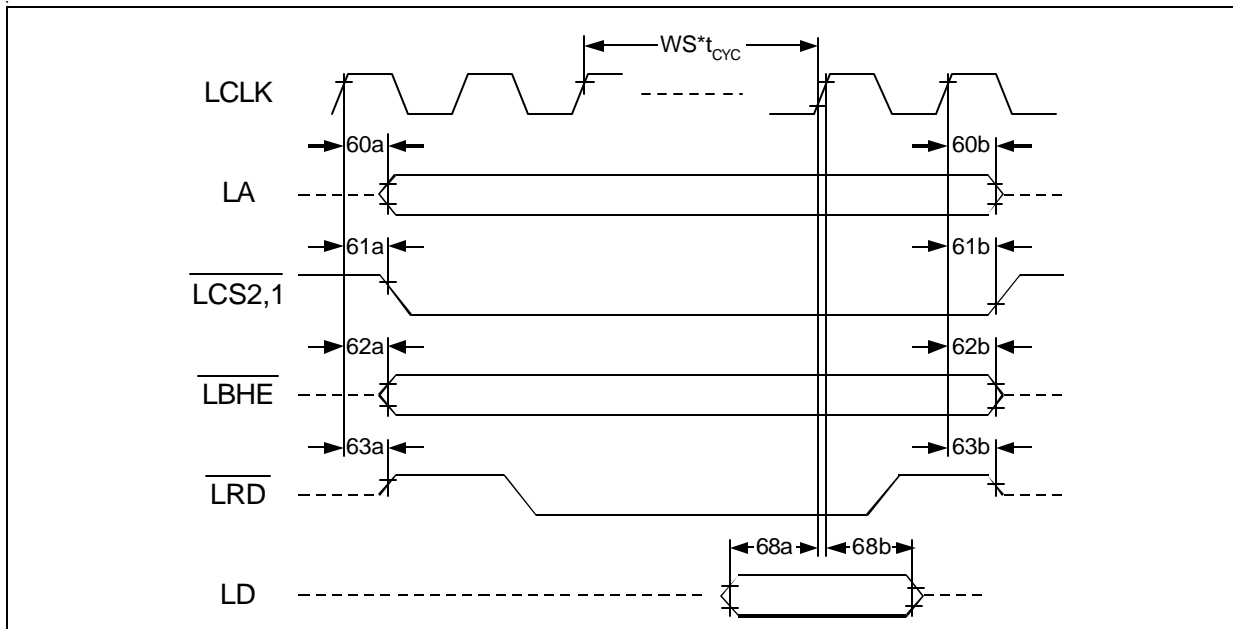


Figure 9-10 Intel Read Cycle Timing (Master Mode, Wait state controlled)

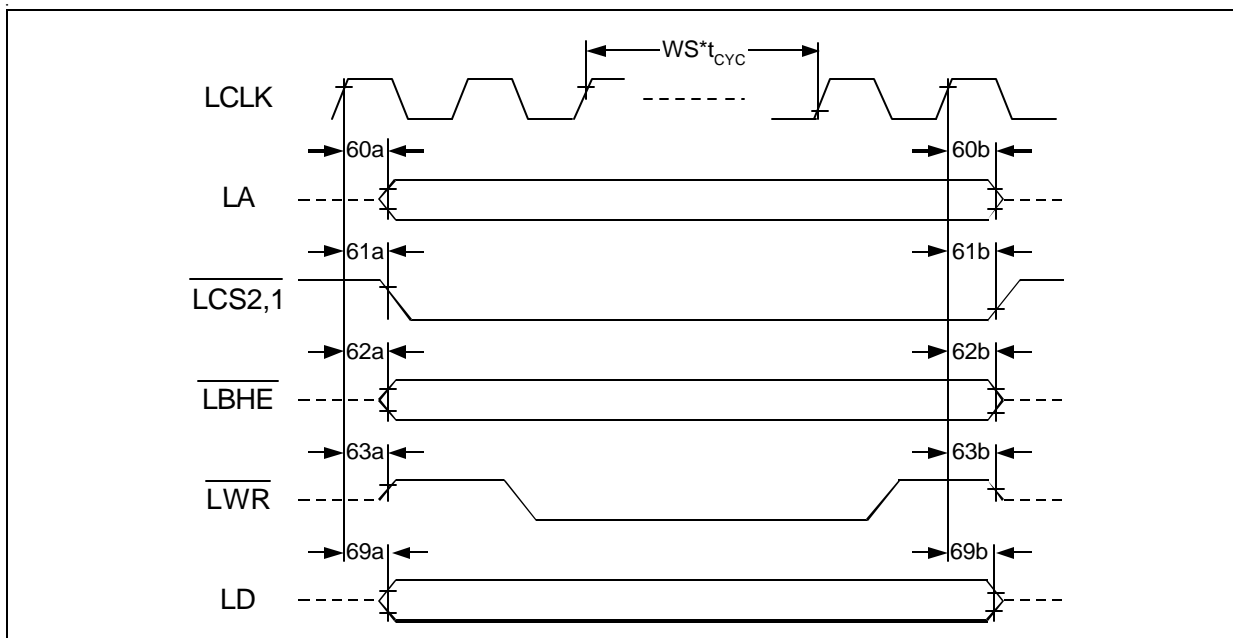


Figure 9-11 Intel Write Cycle Timing (Master Mode, Wait state controlled)

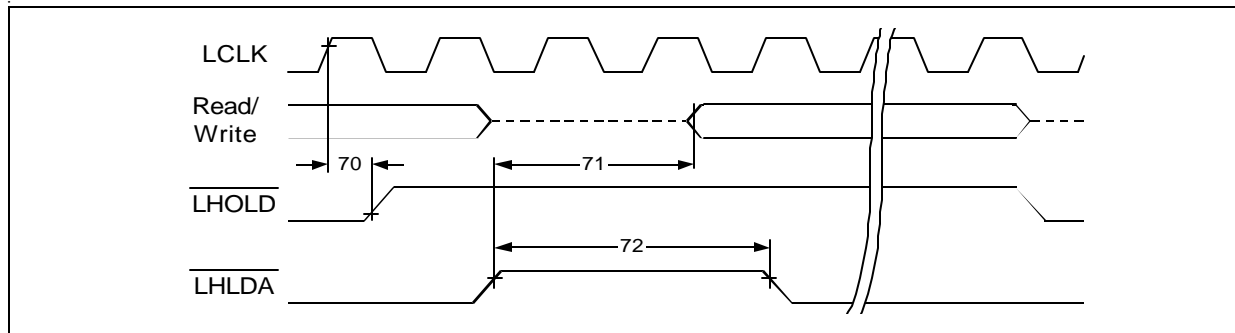


Figure 9-12 Intel Bus Arbitration Timing

Table 9-9 Intel Bus Interface Timing (Master Mode)

No.	Parameter	Limit Values		Unit
		min.	max.	
60a	LCLK to LA active delay	0	10	ns
60b	LCLK to LA float delay	0	10	ns
61a	LCLK to $\overline{\text{LCS2,1}}$ active delay	0	10	ns
61b	LCLK to $\overline{\text{LCS2,1}}$ float delay	0	10	ns
62a	LCLK to $\overline{\text{LBHE}}$ active delay	0	10	ns
62b	LCLK to $\overline{\text{LBHE}}$ float delay	0	10	ns
63a	LCLK to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ active delay	0	10	ns
63b	LCLK to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ float delay	0	10	ns
65	$\overline{\text{LRDY}}$ low to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ high delay	2		$t_{\text{CYC}}$
66	$\overline{\text{LRDY}}$ to $\overline{\text{LRD}}$ , $\overline{\text{LWR}}$ hold time	0		ns
67a	LD to $\overline{\text{LRD}}$ setup time	0		ns
67b	LD to $\overline{\text{LRD}}$ hold time	0		ns
68a	LD to LCLK setup time	10		ns
68b	LD to LCLK hold time	0		ns
69a	LCLK to LD delay	0	10	ns
69b	LCLK to LD float delay	0	10	ns
70	LCLK to LHOLD delay	0	10	ns
71	LHLDA asserted to Read/Write Cycle start	1		$t_{\text{CYC}}$
72	LHLDA minimum pulse width	2		$t_{\text{CYC}}$

Note:  $t_{\text{CYC}}$  is the clock period of the PCI clock.

### 9.4.3.3 Motorola Bus Interface Timing (Slave Mode)

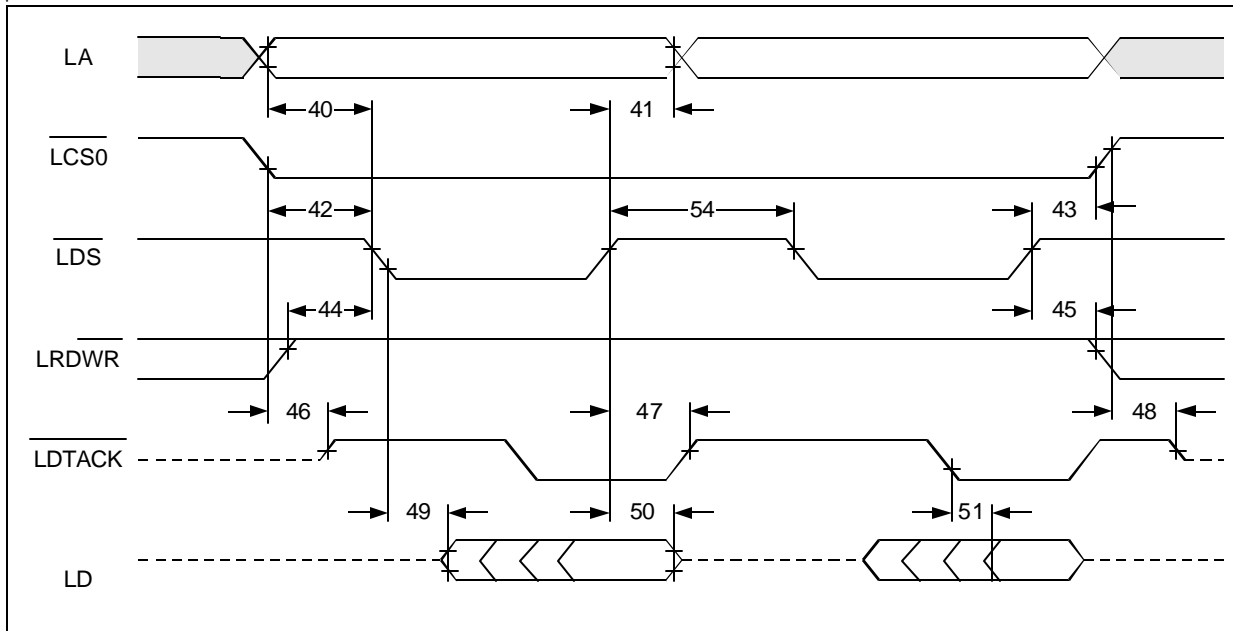


Figure 9-13 Motorola Read Cycle Timing (Slave Mode)

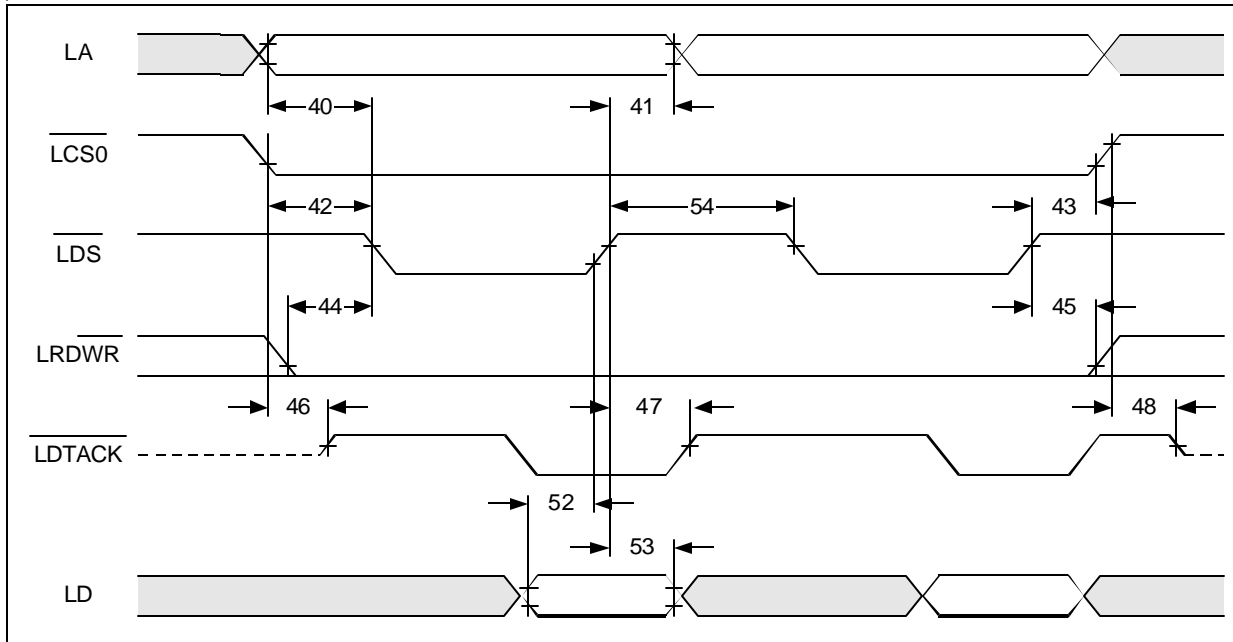


Figure 9-14 Motorola Write Cycle Timing (Slave Mode)

**Table 9-10 Motorola Bus Interface Timing**

No.	Parameter	Limit Values		Unit
		min.	max.	
40	LA to $\overline{\text{LDS}}$ setup time	20		ns
41	LA to $\overline{\text{LDS}}$ hold time	0		ns
42	$\overline{\text{LCS0}}$ to $\overline{\text{LDS}}$ setup time	20		ns
43	$\overline{\text{LCS0}}$ to $\overline{\text{LDS}}$ hold time	0		ns
44	$\overline{\text{LRDWR}}$ to $\overline{\text{LDS}}$ setup time	20		ns
45	$\overline{\text{LRDWR}}$ to $\overline{\text{LDS}}$ hold time	0		ns
46	$\overline{\text{LCS0}}$ low to $\overline{\text{LDTACK}}$ active delay		20	ns
47	$\overline{\text{LDS}}$ high to $\overline{\text{LDTACK}}$ high delay		20	ns
48	$\overline{\text{LCS0}}$ high to $\overline{\text{LDTACK}}$ float delay		20	ns
49	$\overline{\text{LDS}}$ low to LD active delay		20	ns
50	$\overline{\text{LDS}}$ high to LD float delay		20	ns
51	$\overline{\text{LDTACK}}$ low to LD valid delay		20	ns
52	LD to $\overline{\text{LDS}}$ setup time	20		ns
53	LD to $\overline{\text{LDS}}$ hold time	0		ns
54	$\overline{\text{LDS}}$ minimum high time	20		ns

### 9.4.3.4 Motorola Bus Interface Timing (Master Mode)

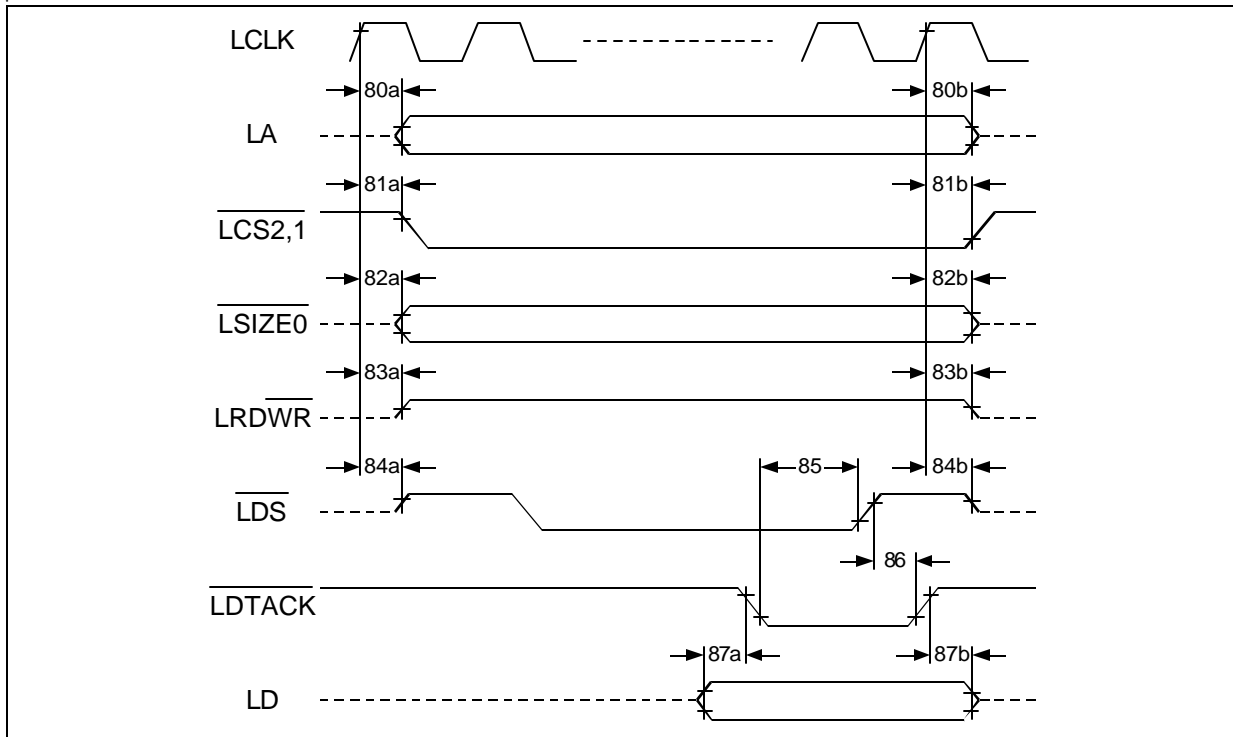


Figure 9-15 Motorola Read Cycle Timing (Master Mode, LDTACK controlled)

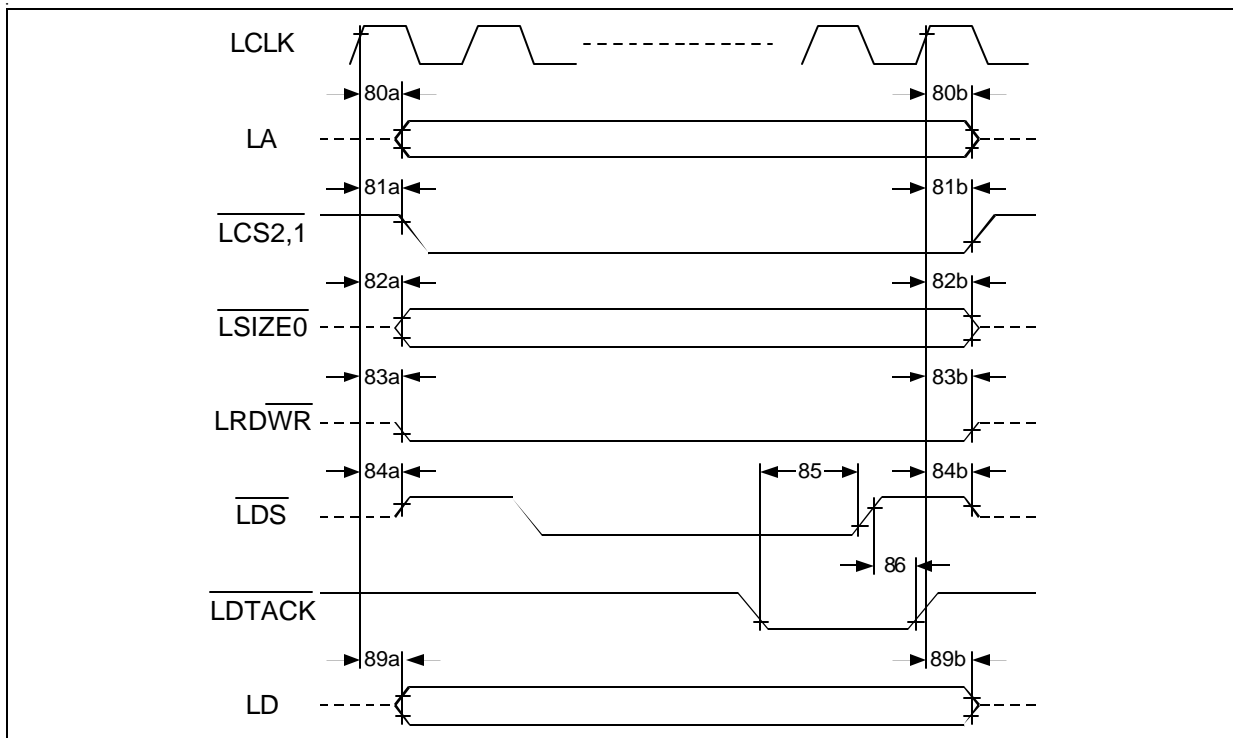


Figure 9-16 Motorola Write Cycle Timing (Master Mode, LDTACK controlled)



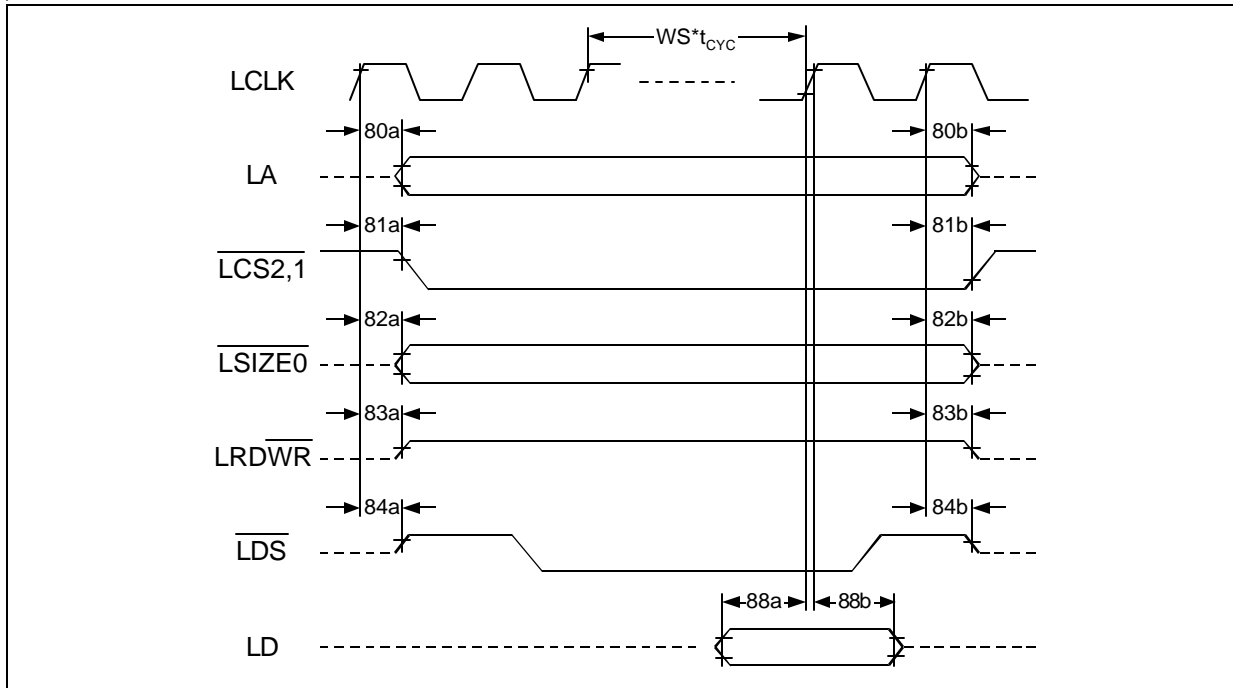


Figure 9-17 Motorola Read Cycle Timing (Master Mode, Wait state controlled)

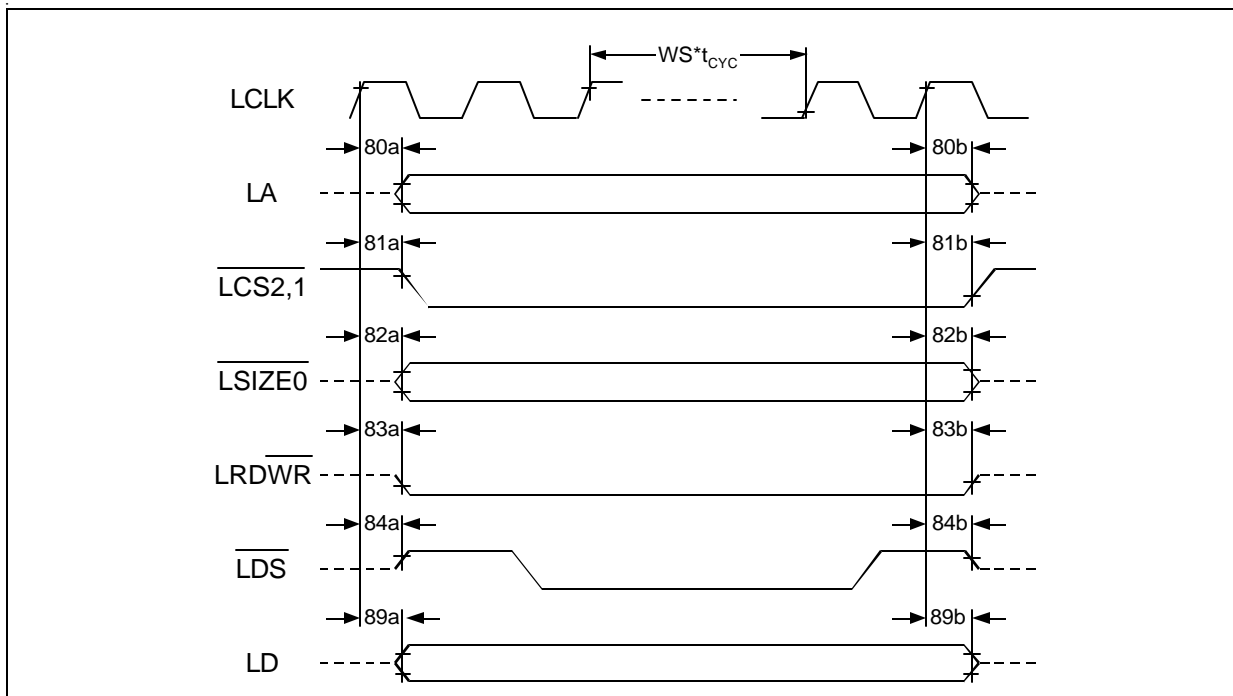


Figure 9-18 Motorola Write Cycle Timing (Master Mode, Wait state controlled)

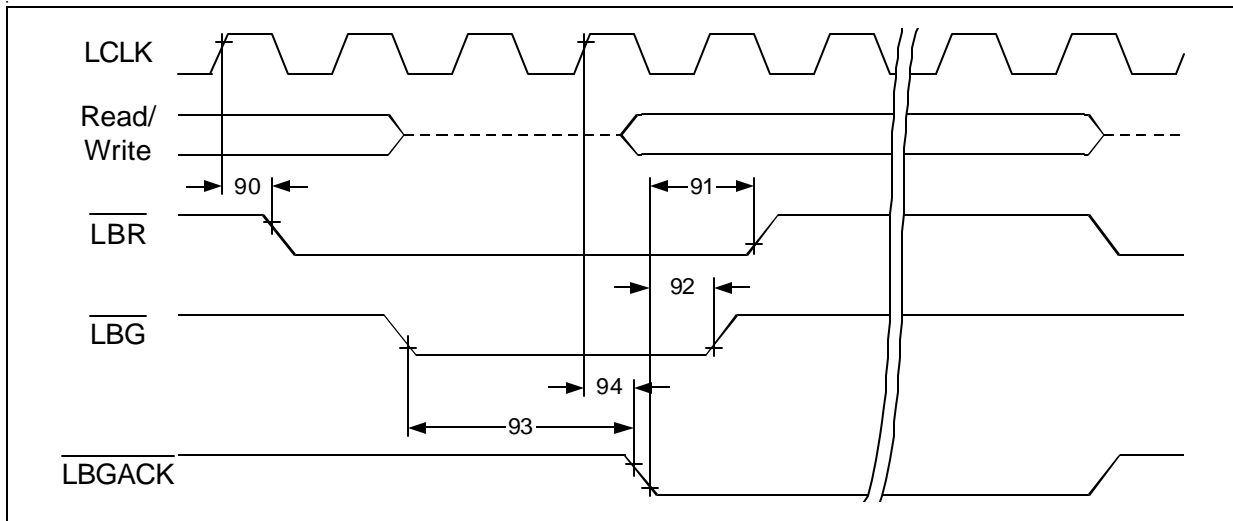


Figure 9-19 Motorola Bus Arbitration Timing

Table 9-11 Motorola Bus Interface Timing (Master Mode)

No.	Parameter	Limit Values		Unit
		min.	max.	
80a	LCLK to LA active delay	0	10	ns
80b	LCLK to LA float delay	0	10	ns
81a	LCLK to $\overline{\text{LCS2,1}}$ active delay	0	10	ns
81b	LCLK to $\overline{\text{LCS2,1}}$ float delay	0	10	ns
82a	LCLK to LSIZE0 active delay	0	10	ns
82b	LCLK to LSIZE0 float delay	0	10	ns
83a	LCLK to $\overline{\text{LRDWR}}$ active delay	0	10	ns
83b	LCLK to $\overline{\text{LRDWR}}$ float delay	0	10	ns
84a	LCLK to $\overline{\text{LDS}}$ active delay	0	10	ns
84b	LCLK to $\overline{\text{LDS}}$ float delay	0	10	ns
85	$\overline{\text{LDTACK}}$ low to $\overline{\text{LDS}}$ high delay	2		$t_{\text{CYC}}$
86	$\overline{\text{LDTACK}}$ to $\overline{\text{LDS}}$ hold time	0		ns
87a	LD to $\overline{\text{LDTACK}}$ setup time	0		ns
87b	LD to $\overline{\text{LDTACK}}$ hold time	0		ns
88a	LD to LCLK setup time	10		ns
88b	LD to LCLK hold time	0		ns
89a	LCLK to LD delay	0	10	ns

Electrical Characteristics

No.	Parameter	Limit Values		Unit
		min.	max.	
89b	LCLK to LD float delay	0	10	ns
90	LCLK to $\overline{\text{LBR}}$ delay	0	10	ns
91	$\overline{\text{LBGACK}}$ to $\overline{\text{LBR}}$ delay	1		$t_{\text{CYC}}$
92	LBG to $\overline{\text{LBGACK}}$ hold time	0		ns
93	$\overline{\text{LBG}}$ to $\overline{\text{LBGACK}}$ delay	1		$t_{\text{CYC}}$
94	LCLK to $\overline{\text{LBGACK}}$ delay	0	10	ns

Note:  $t_{\text{CYC}}$  is the clock period of the PCI clock.

Note: Status signals are generated synchronous to the PCI clock.

## 9.4.4 Serial Interface Timing

### 9.4.4.1 Clock Input Timing

Note: The clock input timings are calculated assuming PCI clock frequency of 33 MHz or more.

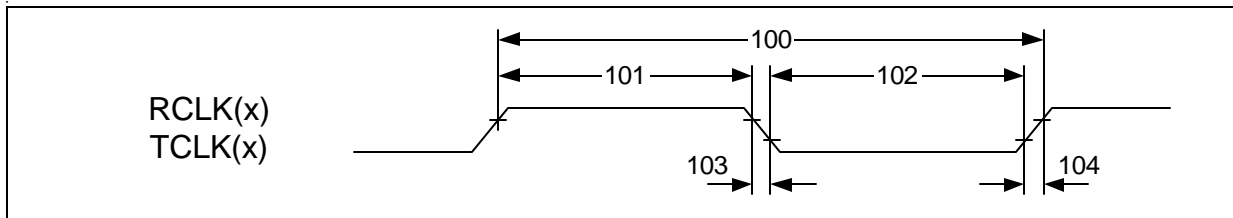


Figure 9-20 Clock Input Timing

Table 9-12 Clock Input Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
<b>Unchannelized Mode: High Speed Port 0</b>				
100	Clock period	20		ns
101	Clock high timing	7.5		ns
102	Clock low timing	7.5		ns
<b>Unchannelized Mode: Ports 1...27</b>				
100	Clock period	100		ns
101	Clock high timing	40		ns
102	Clock low timing	40		ns
<b>E1, T1 Mode: All Ports</b>				
100	Clock period	480		ns
101	Clock high timing	40		ns
102	Clock low timing	40		ns
<b>All Ports</b>				
103	Clock fall time		10	ns
104	Clock rise time		10	ns

### 9.4.4.2 Transmit Cycle Timing

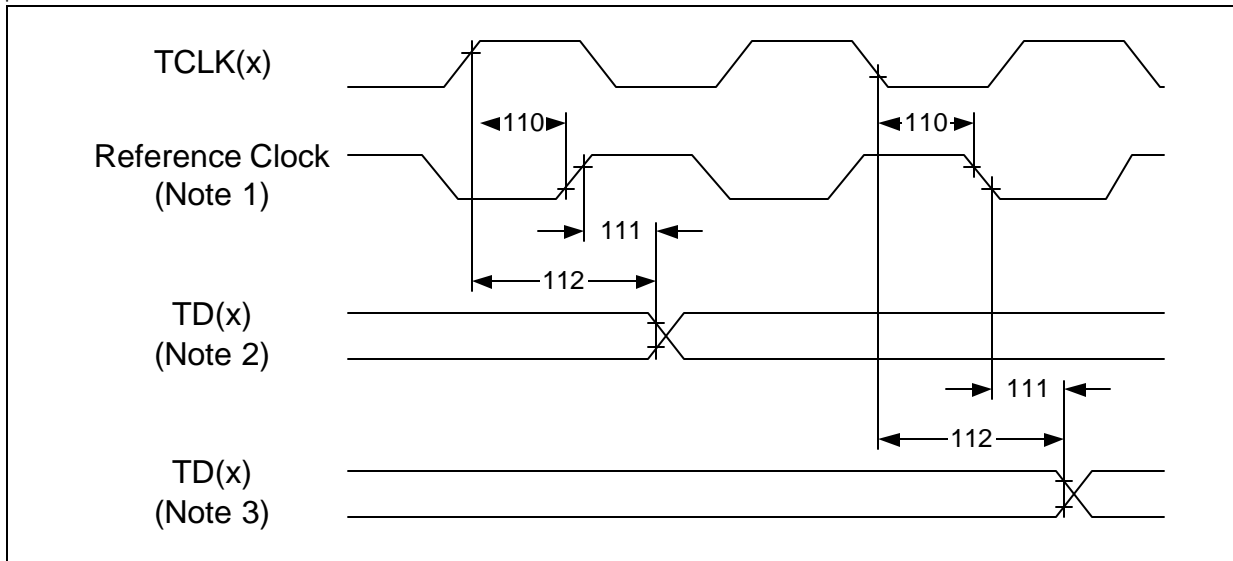


Figure 9-21 Transmit Cycle Timing

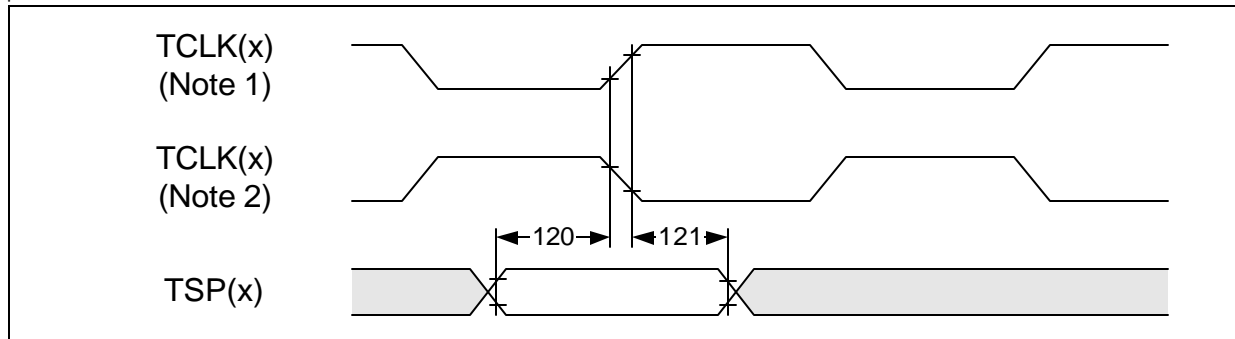
Note:

1. Reference Clock is only available in unchannelized mode for port zero. It is output on port TCLKO.
2. Timing for transmit data which is updated on the rising edge of the transmit clock.
3. Timing for transmit data which is updated on the falling edge of the transmit clock.

Table 9-13 Transmit Cycle Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
<b>Unchannelized Mode: High Speed Port 0</b>				
110	TCLK(0) to TCLKO delay		15	ns
111	TCLKO to TD(0) delay	0	5	ns
112	TCLK(0) to TD(0) delay		20	ns
<b>Unchannelized Mode: Ports 1..15 E1, T1 Mode: All Ports</b>				
112	TCLK(x) to TD(x) delay		25	ns

### 9.4.4.3 Transmit Synchronization Timing



**Figure 9-22 Transmit Synchronization Timing**

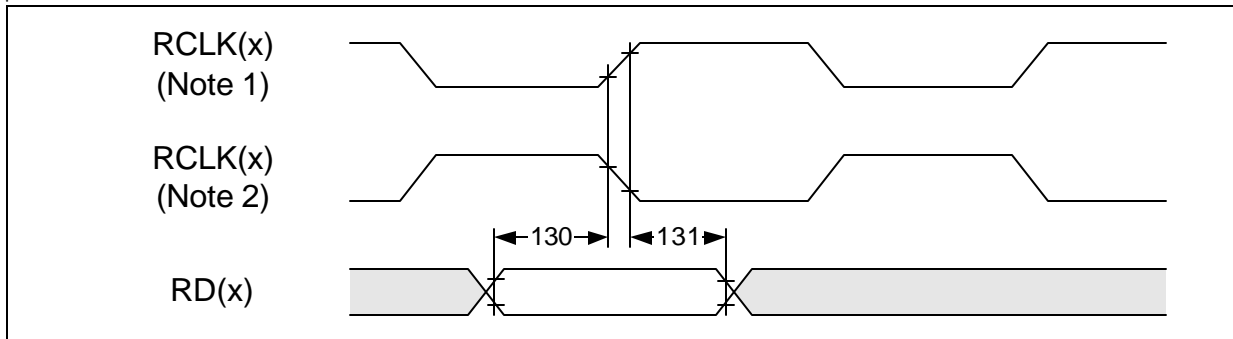
Note:

1. Timing for Transmit Synchronization Pulse updated on the rising edge of the transmit clock.
2. Timing for Transmit Synchronization Pulse updated on the falling edge of the transmit clock.

**Table 9-14 Transmit Synchronization Timing**

No.	Parameter	Limit Values		Unit
		min.	max.	
<b>E1, T1 Mode: All Ports</b>				
120	TSP(x) to TCLK(x) setup time	5		ns
121	TSP(x) to TCLK(x) hold time	5		ns

### 9.4.4.4 Receive Cycle Timing



**Figure 9-23 Receive Cycle Timing**

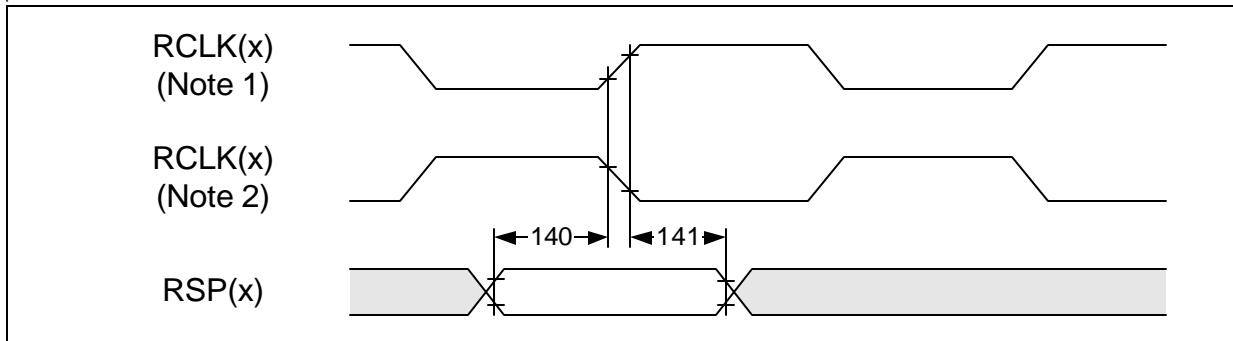
Note:

1. Timing for receive data sampled on the rising edge of the receive clock.
2. Timing for receive data sampled on the falling edge of the receive clock.

**Table 9-15 Receive Cycle Timing**

No.	Parameter	Limit Values		Unit
		min.	max.	
<b>E1, T1 Mode: All Ports</b>				
130	RD(x) to RCLK(x) setup time	5		ns
131	RD(x) to RCLK(x) hold time	5		ns

### 9.4.4.5 Receive Synchronization Timing



**Figure 9-24 Receive Synchronization Timing**

Note:

1. Timing for receive synchronization pulse sampled on the rising edge of the receive clock.
2. Timing for receive synchronization pulse sampled on the falling edge of the receive clock.

**Table 9-16 Receive Synchronization Timing**

No.	Parameter	Limit Values	
		min.	max.
<b>E1, T1 Mode: All Ports</b>			
140	RSP(x) to RCLK(x) setup time	5	
141	RSP(x) to RCLK(x) hold time	5	



### 9.4.5 JTAG Interface Timing

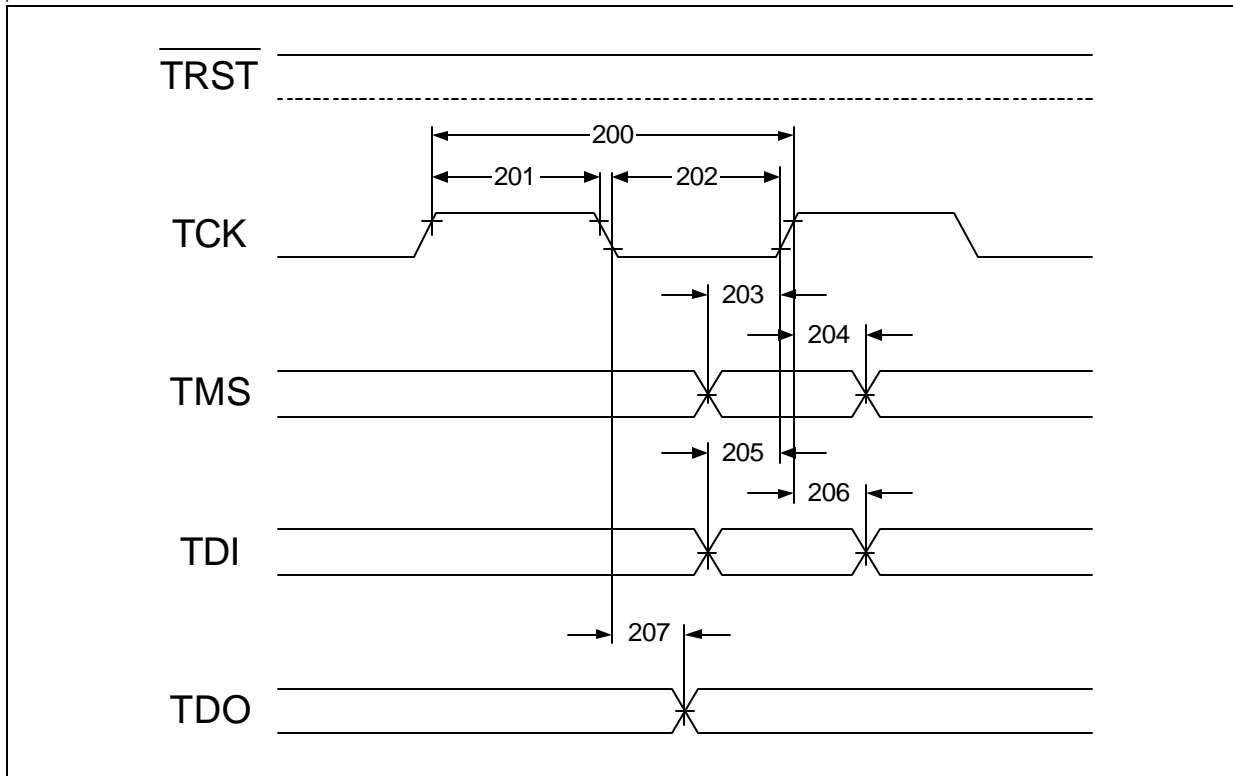


Figure 9-25 JTAG Interface Timing

Table 9-17 JTAG Interface Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
200	TCK period	120		ns
201	TCK high time	60		ns
202	TCK low time	60		ns
203	TMS setup time	20		ns
204	TMS hold time	20		ns
205	TDI setup time	20		ns
206	TDI hold time	20		ns
207	TDO valid time	50		ns

### 9.4.6 Reset Timing

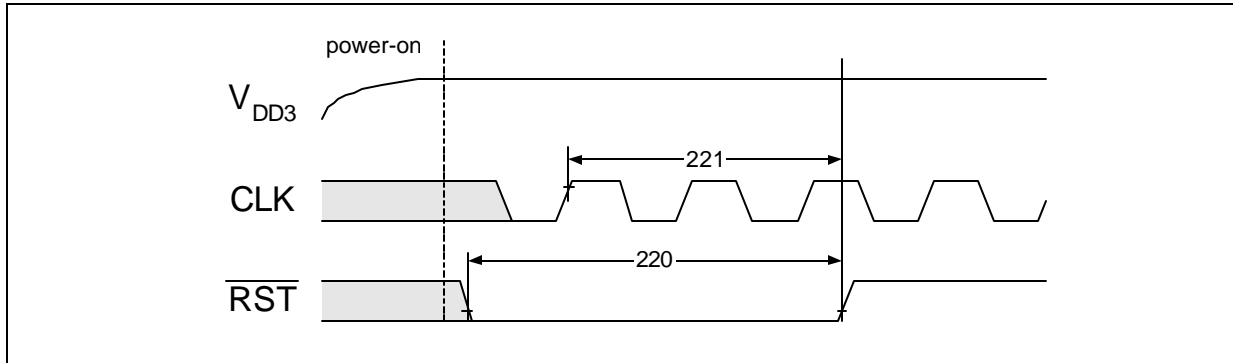
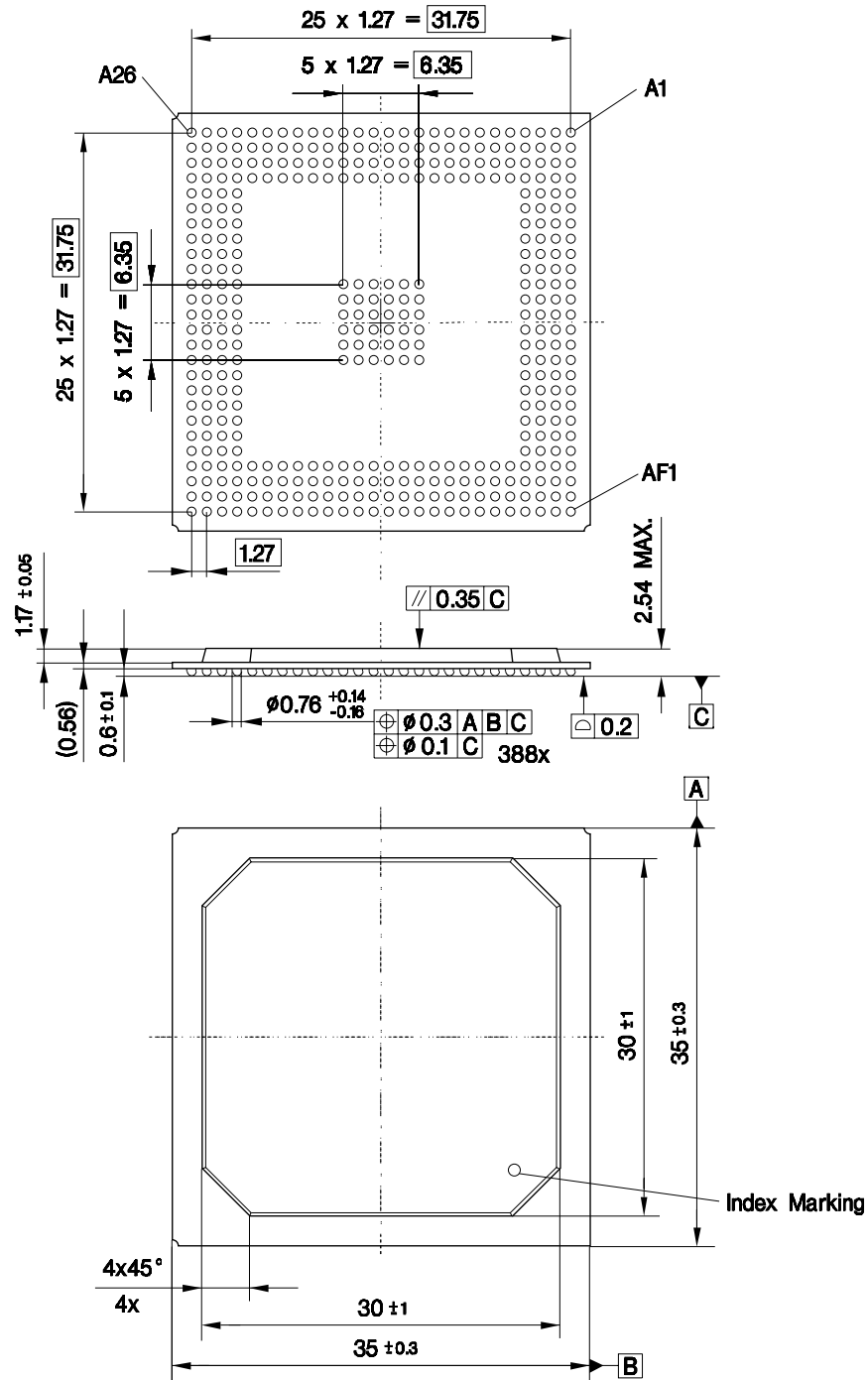


Figure 9-26 Reset Timing

Table 9-18 Reset Timing

No.	Parameter	Limit Values		Unit
		min.	max.	
220	$\overline{RST}$ pulse width	120		ns
221	Number of CLK cycles during $\overline{RST}$ active	2		CLK cycles

## 10 Package Outline





**PEB 20256 E**  
**PEF 20256 E**

---

**Package Outline**

## 11 List of Abbreviations

Abbreviation	Definition
A/C	Analogue to Digital
ADC	Analogue to Digital Converter
AIS	Alarm indication signal (blue alarm)
AGC	Automatic gain control
ALOS	Analog loss of signa
AMI	Alternate mark inversion
ANSI	American National Standards Institute
ATM	Asynchronous transfer mode
SDH	Synchornous Digital Hierarchy
SONET	Synchronous Optical Network
ESF	Extended Superframe
SF	Super Frame
HDLC	High Level Data Link Control
SDLC	Synchronous Level Data Link Control
PCI	Peripheral Component Interconnect.
DS3	Digital Signal Level 3
PLL	Phase Locked Loop
FDL	Facility Data link
SPI	Serial Peripheral Interface
BOM	Bit Oriented Massage
FIFO	First in First out
AUXP	Auxiliary pattern Line 0
B8ZS	Line coding to avoid too long strings of consecutive 0
BER	Bit error rate
BFA	Basic frame alignment
BOM	Bit orientated message
Bellcore	Bell Communications Research
BPV	Bipolar violation

---

**List of Abbreviations**

Abbreviation	Definition
A/C	Analogue to Digital
BSN	Backward sequence number
CAS	Channel associated signaling
CAS-BR	Channel associated signaling - bit robbing
CAS-CC	Channel associated signaling - common channel
CCS	Common channel signaling
CMI	coded mark inversion (also known as 1T2B code)
CR	Command/Response (special bit in PPR)
CRC	Cyclic redundancy check
CSU	Channel service unit
CVC	Code violation counter
DCO	Digitally controlled oscillator
DL	Digital loop
DPLL	Digitally controlled phase locked loop
DS1	Digital signal level 1
EA	Extended address (special bit in PPR)
PRBS	Pseudo Random Binary Sequence
LOS	Loss of Signal
LOF	Loss of Frame
WAN	Wide Area Network
DMA	Direct Memory Access
ACCM	Asynchronous Control Character Map
FCM	Frame Check Sum
DWORD	Double Word ( 4 bytes )
DMU	Data Management Unit



## Infineon goes for Business Excellence

“Business excellence means intelligent approaches and clearly defined processes, which are both constantly under review and ultimately lead to good operating results.

Better operating results and business excellence mean less idleness and wastefulness for all of us, more professional success, more accurate information, a better overview and, thereby, less frustration and more satisfaction.”

Dr. Ulrich Schumacher

<http://www.infineon.com>