SIEMENS

ICs for Communications

Multichannel Network Interface Controller for HDLC MUNICH32X

PEB 20321 Version 2.2

Data Sheet 1998-08-01

DS 1

PEB 20321		
Revision History:		**Current Version: 1998-08-01**
Previous Version:		Preliminary Data Sheet 01.98 (V 2.1)
Page (in previous Version)	Page (in current Version)	Subjects (major changes since last revision)
128 …	131 …	Chapter ‘Local Bus Interface (LBI)’ is reworked.
165 …	170 …	Chapter ‘IOM-2 Interface’ is reworked.
205	209	Description of register ‘LCONF’ modified
226 …	232 …	Description of IOM-2 registers modified.
280 …	292 …	Chapter ‘Electrical Characteristics’ modified.
–	304, 306	PCI Timings added, DEMUX interface timings added.
295	311	LBI Arbitration Timing modified.
–	316	SSC Serial Interface timings added.
–	317	Chapter ‘MUNICH32X Bus Utilization’ added.
8	8, 13, 325	TQFP-176-1 Package and Package Outline added.

Edition 1998-08-01

Published by Siemens AG,
HL SP,
Balanstraße 73,
81541 München

Attention please!

As far as patents or other rights of third parties are concerned, liability is only assumed for components, not for applications, processes and circuits implemented within components or assemblies.

The information describes the type of component and shall not be considered as assured characteristics.

Terms of delivery and rights to change design reserved.

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Siemens Office, Semiconductor Group.

Siemens AG is an approved CECC manufacturer.

Packing

Please use the recycling operators known to you. We can also help you – get in touch with your nearest sales office. By agreement we will take packing material back, if it is sorted. You must bear the costs of transport.

For packing material that is returned to us unsorted or which we are not obliged to accept, we shall have to invoice you for any costs incurred.

Components used in life-support devices or systems must be expressly authorized for such purpose!

Critical components[1] of the Semiconductor Group of Siemens AG, may only be used in life-support devices or systems[2] with the express written approval of the Semiconductor Group of Siemens AG.

1 A critical component is a component used in a life-support device or system whose failure can reasonably be expected to cause the failure of that life-support device or system, or to affect its safety or effectiveness of that device or system.

2 Life support devices or systems are intended (a) to be implanted in the human body, or (b) to support and/or maintain and sustain human life. If they fail, it is reasonable to assume that the health of the user may be endangered.

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

1 Introduction

The MUNICH32X is an enhanced version of the Multichannel Network Interface Controller for HDLC, MUNICH32 (PEB 20320, refer to the User's Manual 01.96).

Key enhancements include:

- a 33 MHz/32-bit PCI bus Master/Slave interface with integrated DMA controllers for higher performance, lower development effort and risk,
- symmetrical Rx and Tx buffer descriptor formats for faster switching,
- an improved Tx idle channel polling process for significantly reduced bus occupancy,
- an integrated Local Bus Interface (LBI) for connection to other peripherals that do not have a PCI bus interface with DMA capability,
- an SSC interface and
- an IOM®-2 interface.

The MUNICH32X provides capability for up to 32 full-duplex serial PCM channels. It performs layer 2 HDLC formatting/deformatting or V.110 or X.30 protocols up to a network data rate of 38.4 Kbit/s (V.110) or 64 Kbit/s (HDLC), as well as transparent transmission for the DMI mode 0, 1, and 2. Processed data are passed on to an external memory shared with one or more host processors.

The MUNICH32X is compatible with the LAPD ISDN (Integrated Services Digital Network) protocol specified by CCITT, as well as with HDLC, SDLC, LAPB and DMI protocols. It provides any rate adaption for time slot transmission data rate from 64 Kbit/s down to 8 Kbit/s and the concatenation of any time slots to data channels, supporting the ISDN H0, H11, H12 superchannels.

The MUNICH32X may be used in a wide range of telecommunication and networking applications, e.g.

- in switches to provide the connection to a PBX, to a host computer, or as a central D-channel controller for 32 D-channels,
- for connection of up to 4 MUNICH32Xs to one PCM highway to achieve a D-channel controller with 128 channels,
- in routers and bridges for LAN-WAN internetworking via channelized T1/E1 or multiple S/T interfaces,
- for wide area trunk cards in routers and switches (Frame Relay, ISDN PRI, Internet Protocols, etc.), and
- for centralized D- or B-channel packet processing in routers, switches (Frame Relay, Q.931 Signaling, X.25, etc.)

Note: In the course of the Data Sheet, the expression 'DWORD' always refers to 32-bit words in correspondence to the PCI specification.

Multichannel Network Interface Controller for HDLC with Extensions MUNICH32X

PEB 20321

Version 2.2

CMOS IC

1.1 Key Features

P-MQFP-160-1

P-TQFP-176-1

32-channel HDLC controller with PCI interface:

- **Serial PCM core**
  - Up to 32 independent full-duplex channels
  - Serial PCM traffic at 2.048, 4.096, 1.544, 1.536, 3.088, 6.176 or 8.192-Mbit/s
- Dynamic Programmable Channel Allocation
  - Compatible with T1/DS1 24-channel and CEPT 32-channel PCM byte format
  - Concatenation of any, not necessarily consecutive, time slots to superchannels independently for receive and transmit direction
  - Support of H0, H11, H12 ISDN-channels
  - Subchanneling on each time slot possible
- **Bit Processor Functions** (adjustable for each channel)
  - HDLC Protocol
    - Automatic flag detection
    - Shared opening and closing flag
    - Detection of interframe-time-fill change, generation of interframe-time-fill ‘1’s or flags
    - Zero bit insertion
    - Flag stuffing and flag adjustment for rate adaption
    - CRC generation and checking (16 or 32 bits)
    - Transparent CRC option per channel and/or per message
    - Error detection (abort, long frame, CRC error, 2 categories of short frames, non-octet frame content)
    - ABORT/IDLE flag generation

Type	Ordering Code	Package
PEB 20321	on request	P-MQFP-160-1
PEB 20321	on request	P-TQFP-176-1

  - V.110/X.30 Protocol
    - Automatic synchronization in receive direction, automatic generation of the synchronization pattern in transmit direction
    - E/S/X bits freely programmable in transmit direction, may be changed during transmission; changes monitored and reported in receive direction
    - Generation/detection of loss of synchronism
    - Bit framing with network data rates from 600 bit/s up to 38.4 Kbit/s
  - Transparent Mode A
    - Slot synchronous transparent transmission/reception without frame structure
    - Flag generation, flag stuffing, flag extraction, flag generation in the abort case with programmable flag
    - Synchronized data transfer for fractional T1/PRI channels
  - Transparent Mode B
    - Transparent transmission/reception in frames delimited by 00_H flags
    - Shared opening and closing flag
    - Flag stuffing, flag detection, flag generation in the abort case
    - Error detection (non octet frame content, short frame, long frame)
  - Transparent Mode R
    - Transparent transmission/reception with GSM 08.60 frame structure
    - Automatic 0000_H flag generation/detection
    - Support of 40, $39^1/_2$, $40^1/_2$ octet frames
    - Error detection (non octet frame contents, short frame, long frame)
  - Protocol Independent
    - Channel inversion (data, flags, IDLE code)
    - Format conventions as in CCITT Q.921 § 2.8
    - Data over- and underflow detected
- **Microprocessor Interface**
  - 32-bit PCI bus interface option, 33 MHz
  - 32-bit De-multiplexed bus interface option, 33 MHz
  - 68 channel DMA controller (64 for 32 serial channels, 4 for 2 LBI channels) with buffer chaining capability
  - Master 4-DWORD burst read and write capability
  - Slave single-DWORD read and write capability
  - Interrupt-circular buffers with variable sizes
  - Maskable interrupts for each channel
- **IOM®-2 Interface with on-chip C/I and monitor handlers**
- **Synchronous Serial Control (SSC) Interface**
- **8-/16-bit Local Bus Interface (LBI)**

- **General**
  - Connection of up to four MUNICH32X supporting a 128-channel basic access D-channel controller
  - On-chip Rx and Tx data buffers 256 bytes each
  - HDLC protocol or transparent mode, support of ECMA 102, CCITT I4.63 RA2, V.110, X.30, DMI mode 0, 1, 2 (bit rate adaption), GSM 08.60 TRAU frames
  - Loopback mode, complete loop as well as single channel loop
  - JTAG boundary scan test
  - 0.5 µm low-power CMOS technology
  - 3.3 V and 5 V voltage supply
  - TTL-compatible inputs/outputs
  - 160-pin P-MQFP package
  - 176-pin P-TQFP package

1.2 New or Changed from MUNICH32, PEB 20320

- Symmetrical Rx and Tx Buffer Descriptor formats for faster switching
- Improved Tx idle channel polling process, which significantly reduces bus occupancy of idle Tx channels
- Additional PCM modes supported: 3.088 Mbit/s, 6.176 Mbit/s, 8.192 Mbit/s
- 32-bit PCI bus Master/Slave interface (33 MHz) with integrated DMA controllers for higher performance, and lower development effort and risk
- Enhanced Interrupt Structure providing:
  separate serial PCM Rx and Tx Interrupt Queues in host memory,
  separate DMA related LBI Rx and Tx Interrupt Queues in host memory,
  dedicated LBI pass-through, SSC, General Purpose bus and IOM®-2 Peripheral Interrupt Queue in host memory
- Slave read capability of serial PCM core, LBI, SSC and IOM®-2 read/write registers
- Time Slot Shift capability
  programmable from -4 clock edges to +3 clock edges relative to synchronization pulse,
  programmable to sample Tx data at either clock falling or rising edge,
  programmable to sample Rx data at either clock falling or rising edge,
- Software initiated Action Request via a bit field in the Command register
- Tx End-of-Packet transmitted-on-wire interrupt capability per channel
- Tx packet size increased to 16 Kbytes
- Rx packet size 8 kbyte limit interrupt disable
- Rx Enable bit field of the MODE1 register
- Rx Interrupt Disable bit field of the MODE1 register
- Tx data tristate control line ($\overline{\text{TXDEN}}$)
- Synchronized data transfer in TMA mode for complete transparency when using fractional T1/PRI channels
- Integrated Local Bus Interface (LBI), which allows connection to peripherals that do not provide a PCI bus interface
- IOM®-2 interface with single and double data rate clock
- Collision control on S/T interface by QUAT-S (PEB 2084) via data ready control line (DRDY)
- Synchronous Serial Control (SSC) interface
- 16-bit General Purpose Bus (available, when LBI and SSC are not used)
- Internal Descriptor and Table Dump capability for software development purposes
- Little/Big Endian data formats selectable via a bit field in Configuration register

1.3 Pin Configuration

(top view)

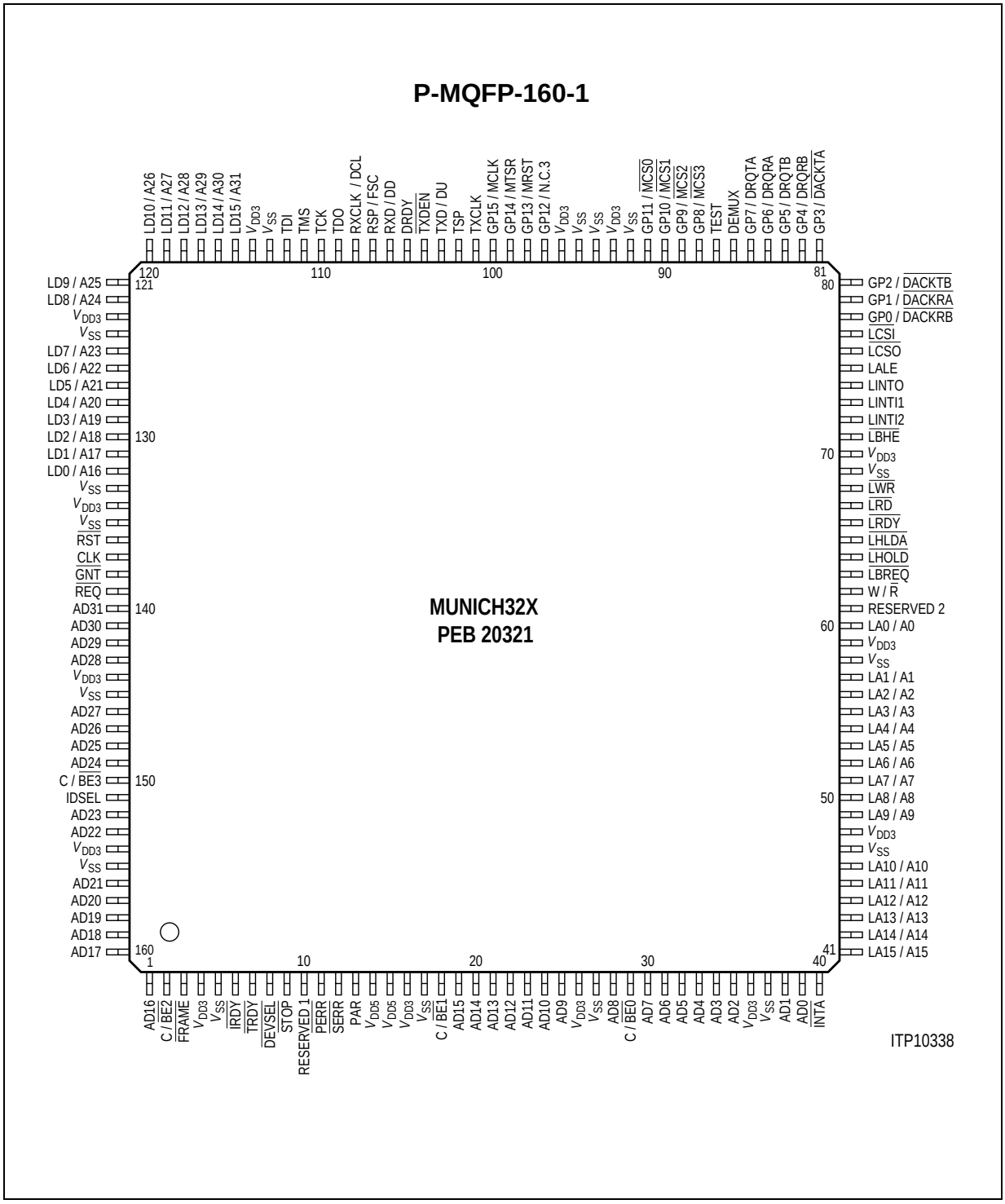


Figure 1

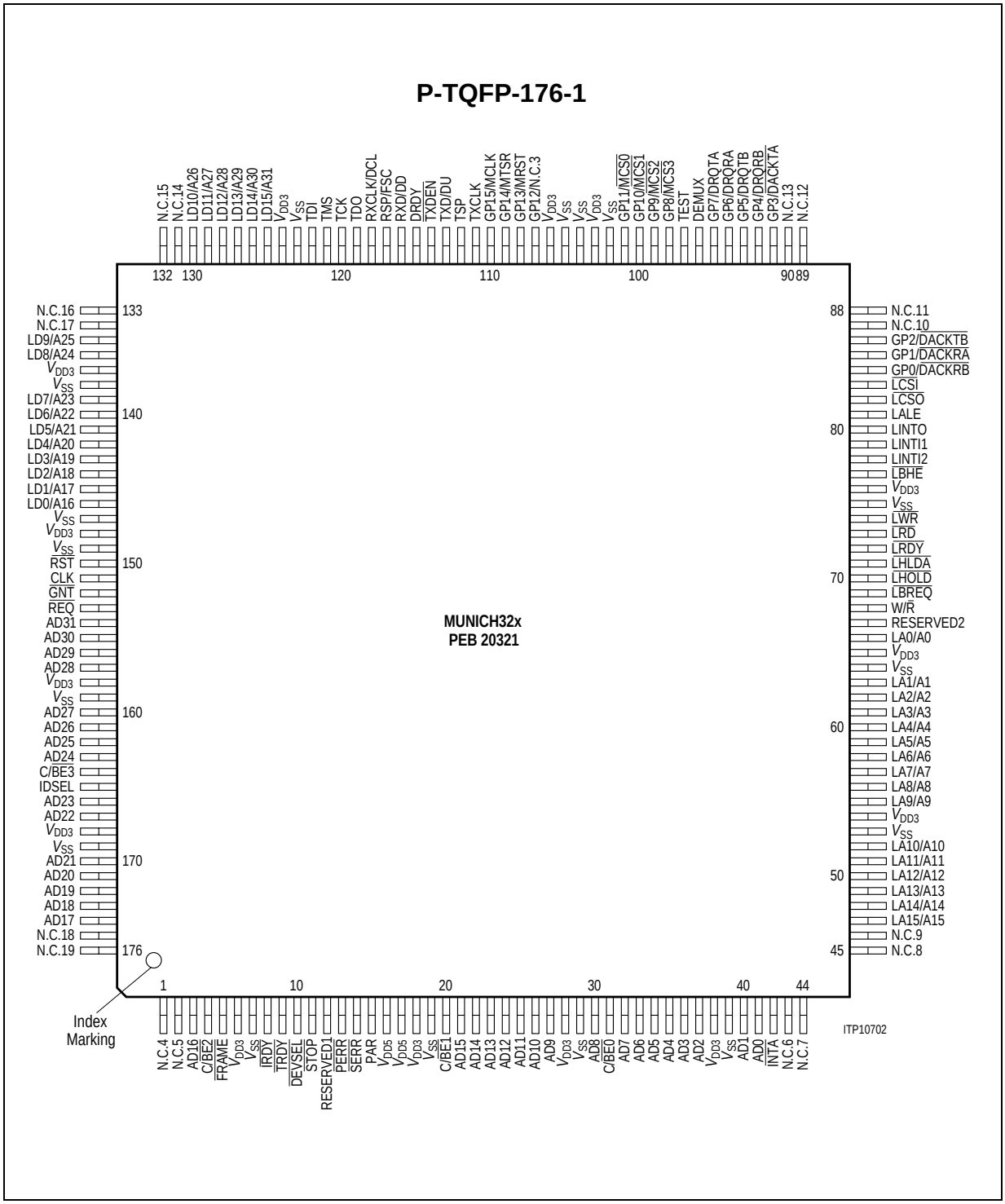


Figure 2

Signal Type Definition:

The following signal type definitions are mainly taken from the PCI Specification Revision 2.1:

in	*Input* is a standard input-only signal.
out	*Totem Pole Output* is a standard active driver.
t/s, I/O	*Tri-State* or *I/O* is a bi-directional, 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 one agent at a time. (For further information refer to the PCI Specification Revision 2.1)
o/d	*Open Drain* allows multiple devices to share 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.

Table 1
PCI Bus Interface Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
1, 28, 38, 140…143, 146…149, 152, 153, 156…160, 19…25, 30…35, 39	3, 30, 40, 154…158, 160…163, 166, 167, 170…174, 21…27, 32…37, 41	AD(31:0)	t/s	**Address/Data Bus** A bus transaction consists of an address phase followed by one or more data phases. When MUNICH32X is 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. When MUNICH32X is 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. AD(31:0) are updated and sampled on the rising edge of CLK.
150, 2, 18, 29	164, 4, 20, 31	C/$\overline{\text{BE}}$(3:0)	t/s	**Command/Byte Enable** During the address phase of a transaction, C/$\overline{\text{BE}}$(3:0) define the bus command. During the data phase, C/$\overline{\text{BE}}$(3:0) are used as Byte Enables. The Byte Enables are valid for the entire data phase and determine which byte lanes carry meaningful data. C/$\overline{\text{BE0}}$ applies to byte 0 (lsb) and C/$\overline{\text{BE3}}$ applies to byte 3 (msb). When MUNICH32X is Master, C/$\overline{\text{BE}}$(3:0) are outputs. When MUNICH32X is Slave, C/$\overline{\text{BE}}$(3:0) are inputs. C/$\overline{\text{BE}}$(3:0) are updated and sampled on the rising edge of CLK.

Table 1
PCI Bus Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
13	15	PAR	t/s	**Parity** PAR is even parity across AD(31:0) and C/$\overline{BE}$(3:0). PAR is stable and valid one clock after the address phase. PAR has the same timing as AD(31:0) but delayed by one clock. When MUNICH32X is Master, PAR is output during address phase and write data phases. When MUNICH32X is Slave, PAR is output during read data phases. Parity errors detected by the MUNICH32X are indicated on $\overline{PERR}$ output. PAR is updated and sampled on the rising edge of CLK.
3	5	$\overline{FRAME}$	s/t/s	**Frame** $\overline{FRAME}$ indicates the beginning and end of an access. $\overline{FRAME}$ is asserted to indicate a bus transaction is beginning. While $\overline{FRAME}$ is asserted, data transfers continue. When $\overline{FRAME}$ is deasserted, the transaction is in the final phase. When MUNICH32X is Master, $\overline{FRAME}$ is an output. When MUNICH32X is Slave, $\overline{FRAME}$ is an input. $\overline{FRAME}$ is updated and sampled on the rising edge of CLK.

Table 1
PCI Bus Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
6	8	$\overline{\text{IRDY}}$	s/t/s	**Initiator Ready** $\overline{\text{IRDY}}$ indicates the bus master's ability to complete the current data phase of the transaction. It is used in conjunction with $\overline{\text{TRDY}}$. A data phase is completed on any clock where both $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are sampled asserted. During a write, $\overline{\text{IRDY}}$ 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 $\overline{\text{IRDY}}$ and $\overline{\text{TRDY}}$ are asserted together. When MUNICH32X is Master, $\overline{\text{IRDY}}$ is an output. When MUNICH32X is Slave, $\overline{\text{IRDY}}$ is an input. $\overline{\text{IRDY}}$ is updated and sampled on the rising edge of CLK.
7	9	$\overline{\text{TRDY}}$	s/t/s	**Target Ready** $\overline{\text{TRDY}}$ indicates a slave's ability to complete the current data phase of the transaction. During a read, $\overline{\text{TRDY}}$ indicates that valid data is present on AD(31:0). During a write, it indicates the target is prepared to accept data. When MUNICH32X is Master, $\overline{\text{TRDY}}$ is an input. When MUNICH32X is Slave, $\overline{\text{TRDY}}$ is an output. $\overline{\text{TRDY}}$ is updated and sampled on the rising edge of CLK.

Table 1
PCI Bus Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
9	11	$\overline{\text{STOP}}$	s/t/s	**STOP** $\overline{\text{STOP}}$ is used by a slave to request the current master to stop the current bus transaction. When MUNICH32X is Master, $\overline{\text{STOP}}$ is an input. When MUNICH32X is Slave, $\overline{\text{STOP}}$ is an output. $\overline{\text{STOP}}$ is updated and sampled on the rising edge of CLK.
151	165	IDSEL	I	**Initialization Device Select** When MUNICH32X is slave in a transaction, if IDSEL is active in the address phase and C/$\overline{\text{BE}}$(3:0) indicates an I/O read or write, the MUNICH32X assumes a read or write to a configuration register. In response, the MUNICH32X asserts $\overline{\text{DEVSEL}}$ during the subsequent CLK cycle. IDSEL is sampled on the rising edge of CLK.
8	10	$\overline{\text{DEVSEL}}$	s/t/s	**Device Select** 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 transaction. If no bus slave activates $\overline{\text{DEVSEL}}$ within six bus CLK cycles, the master should abort the transaction. When MUNICH32X is master, $\overline{\text{DEVSEL}}$ is input. If $\overline{\text{DEVSEL}}$ is not activated within six clock cycles after an address is output on AD(31:0), the MUNICH32X aborts the transaction and generates an $\overline{\text{INTA}}$. When MUNICH32X is slave, $\overline{\text{DEVSEL}}$ is output.

Table 1
PCI Bus Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
11	13	$\overline{\text{PERR}}$	s/t/s	**Parity Error** When activated, indicates a parity error over the AD(31:0) and C/$\overline{\text{BE}}$(3:0) signals (compared to the PAR input). It has a delay of two CLK cycles with respect to AD and C/$\overline{\text{BE}}$(3:0) (i.e., it is valid for the cycle immediately following the corresponding PAR cycle). $\overline{\text{PERR}}$ is asserted relative to the rising edge of CLK.
12	14	$\overline{\text{SERR}}$	o/d	**System Error** The MUNICH32X asserts this signal to indicate a fatal system error. $\overline{\text{SERR}}$ is activated on the rising edge of CLK.
139	153	$\overline{\text{REQ}}$	t/s	**Request** Used by the MUNICH32X to request control of the PCI. $\overline{\text{REQ}}$ is activated on the rising edge of CLK.
138	152	$\overline{\text{GNT}}$	t/s	**Grant** This signal is asserted by the arbiter to grant control of the PCI to the MUNICH32X in response to a bus request via $\overline{\text{REQ}}$. After $\overline{\text{GNT}}$ is asserted, the MUNICH32X will begin a bus transaction only after the current bus Master has deasserted the $\overline{\text{FRAME}}$ signal. $\overline{\text{GNT}}$ is sampled on the rising edge of CLK.
137	151	CLK	I	**Clock** Provides timing for all PCI transactions. Most PCI signals are sampled or output relative to the rising edge of CLK. The actual clock frequency is either equal to the frequency of CLK, or CLK frequency divided by 2. The maximum CLK frequency is 33 MHz.

Table 1
PCI Bus Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
136	150	$\overline{\text{RST}}$	I	**Reset** An asynchronous active low $\overline{\text{RST}}$ signal brings all PCI registers, sequencers and signals into a consistent state. All PCI output signals are driven to their benign state. During RESET all output and I/O pins are in tristate condition with the following exception: TXDEN is active high during RESET.
40	42	$\overline{\text{INTA}}$	O (oD)	**Interrupt Request** When an interrupt status is active and unmasked, the MUNICH32X activates this open-drain output. Examples of interrupt sources are transmission/reception error, completion of transmit or receive packets etc. The MUNICH32X deactivates $\overline{\text{INTA}}$ when the interrupt status is acknowledged via an appropriate action (e.g., specific register write) and no other unmasked interrupt statuses are active. $\overline{\text{INTA}}$ is activated/ deactivated asynchronous to the CLK.

Note: PCI control signals always require pull-up resistors. For the system dependent pull-up recommendation please refer to PCI Specification Revision 2.1 chapter 4.3.3.

Table 2
Dedicated Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
86	96	DEMUX	I	**PCI/De-multiplexed Mode Select** DEMUX = 0 indicates normal PCI operation. DEMUX = 1 indicates that the MUNICH32X is operated in De-multiplexed mode.
62	68	W/$\overline{R}$	I/O	**Write/Read** This signal distinguishes write and read operations in the De-multiplexed mode. It is tristate when the MUNICH32X is in PCI mode. **A Pull-Up resistor to V_{DD3} is recommended if De-multiplexed mode is not used.**
10	12	RES. 1	I/O	Reserved. **A Pull-Up resistor to V_{DD3} is recommended.**
61	67	RES. 2	O	Reserved. **A Pull-Up resistor to V_{DD3} is recommended.**
5, 17, 27, 37, 47, 58, 69, 92, 94, 95, 113, 124, 133, 135, 145, 155	7, 19, 29, 39, 53, 64, 75, 102, 104, 105, 123, 138, 147, 149, 159, 169	V_{SS}	–	**Ground** (0 V) All pins must have the same level.
4, 16, 26, 36, 48, 59, 70, 93, 96, 114, 123, 134, 144, 154	6, 18, 28, 38, 54, 65, 76, 103, 106, 124, 137, 148, 158, 168	V_{DD3}	–	**Supply Voltage** 3.3 V ± 0.3 V All pins must have the same level.
14, 15	16, 17	V_{DD5}	–	**Supply Voltage** 5 V ± 0.25 V All pins must have the same level.
87	97	TEST	I	**Test Input** When set to V_{DD3} the MUNICH32X works in test mode. It must be set to V_{SS} for normal working mode.

Table 2
Dedicated Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
JTAG Test Port for Boundary Scan according to IEEE 1149.1				
110	120	TCK	I	**JTAG Test Clock** **A Pull-Up resistor to V_{DD3} is recommended if boundary scan unit is not used.**
111	121	TMS	I	**JTAG Test Mode Select** **A Pull-Up resistor to V_{DD3} is recommended if boundary scan unit is not used.**
112	122	TDI	I	**JTAG Test Data Input** **A Pull-Up resistor to V_{DD3} is recommended if boundary scan unit is not used.**
109	119	TDO	O	**JTAG Test Data Output**

Table 3
Local Bus Interface (LBI) Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
41 … 46, 49 … 57, 60	47 … 52, 55 … 63, 66	LA(15:0)/ A(15:0)	I/O I/O	**LBI Address** These pins provide the 16 bit Address bus for the Local Bus Interface. **A Pull-Down resistor to V_{SS} is recommended if LBI is not used.** **DEMUX Address** These pins provide the 16 least significant address lines for the De-multiplexed Interface, when DEMUX = 1.

Table 3
Local Bus Interface (LBI) Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
115…122, 125…132	125…130, 135, 136, 139…146	LD (15:0)/ A(31:16)	I/O I/O	**LBI Data** These pins provide the 16 bit Data bus for the Local Bus Interface. **A Pull-Down resistor to V_{SS} is recommended if LBI is not used.** **DEMUX Address** These pins provide the 16 most significant address lines for the De-multiplexed Interface, when DEMUX = 1.
64	70	$\overline{\text{LHOLD}}$	I	**LBI Hold Request** $\overline{\text{LHOLD}}$ = 1 is used for normal bus drive mode. $\overline{\text{LHOLD}}$ = 0 requests LBI to enter hold mode. **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
63	69	$\overline{\text{LBREQ}}$	O	**LBI Bus Request** Output $\overline{\text{LBREQ}}$ = 0 to request bus then set $\overline{\text{LBREQ}}$ = 1 after regaining bus.
65	71	$\overline{\text{LHLDA}}$	I/O	**LBI Hold Status** As an output, $\overline{\text{LHLDA}}$ = 0 confirms that the LBI bus is in HOLD mode. As an input, $\overline{\text{LHLDA}}$ = 1 means that MUNICH32X must remain in hold mode. **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
76	82	$\overline{\text{LCSO}}$	O	**LBI Chip Select Output** Used to select LBI external peripheral
77	83	$\overline{\text{LCSI}}$	I	**LBI Chip Select Input** Used to select MUNICH32X as LBI Slave. **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
75	81	LALE	O	**LBI Address Latch Enable** **A Pull-Down resistor to V_{SS} is recommended if LBI is not used.**

Table 3
Local Bus Interface (LBI) Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
67	73	$\overline{\text{LRD}}$	I/O	**LBI Read Strobe** **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
68	74	$\overline{\text{LWR}}$	I/O	**LBI Write Strobe** **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
71	77	$\overline{\text{LBHE}}$	I/O	**LBI Byte High Enable** **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
66	72	$\overline{\text{LRDY}}$	I/O	**LBI Ready Strobe to Extend Cycles** **A Pull-Up resistor to V_{DD3} is recommended if LBI is not used.**
73	79	LINTI1	I	**LBI Interrupt Input from Peripheral1** **In case of bit HE1 in register LCONF is set (HSCX register decoding selected) this pin must be connected to V_{DD3} if unused.** **In case of bit HE1 in register LCONF is reset (ESCC2 register decoding selected) this pin must be connected to V_{SS} if unused.**
72	78	LINTI2	I	**LBI Interrupt Input from Peripheral2** **In case of bit HE1 in register LCONF is set (HSCX register decoding selected) this pin must be connected to V_{DD3} if unused.** **In case of bit HE1 in register LCONF is reset (ESCC2 register decoding selected) this pin must be connected to V_{SS} if unused.**
74	80	LINTO	O	**LBI Interrupt Output to Local Microcontroller**

Table 4
LBI DMA Support/General Purpose Bus Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
85	95	DRQTA/	I	**DMA Request for Transmit Channel A**
		GP7	I/O	**On reset, pin is General Purpose Bus pin**
84	94	DRQRA/	I	**DMA Request for Receive Channel A**
		GP6	I/O	**On reset, pin is General Purpose Bus pin**
83	93	DRQTB/	I	**DMA Request for Transmit Channel B**
		GP5	I/O	**On reset, pin is General Purpose Bus pin**
82	92	DRQRB/	I	**DMA Request for Receive Channel B**
		GP4	I/O	**On reset, pin is General Purpose Bus pin**
81	91	$\overline{\text{DACKTA}}$/	O	**DMA Acknowledge for Transmit Channel A**
		GP3	I/O	**On reset, pin is General Purpose Bus pin**
80	86	$\overline{\text{DACKTB}}$/	O	**DMA Acknowledge for Transmit Channel B**
		GP2	I/O	**On reset, pin is General Purpose Bus pin**
79	85	$\overline{\text{DACKRA}}$/	O	**DMA Acknowledge for Receive Channel A**
		GP1	I/O	**On reset, pin is General Purpose Bus pin**
78	84	$\overline{\text{DACKRB}}$/	O	**DMA Acknowledge for Receive Channel B**
		GP0	I/O	**On reset, pin is General Purpose Bus pin**

Note: If bit 'LBI' is set to '1' in register CONF i.e. DMA support for LBI operation is selected controlled by pin numbers 78..85, all unused pins must be connected in accordance with the following recommendation:
DRQTA, DRQRA, DRQTB, DRQRB to V_{SS}
$\overline{\text{DACKTA}}$, $\overline{\text{DACKTB}}$, $\overline{\text{DACKRA}}$, $\overline{\text{DACKRB}}$ Pull-Up to V_{DD3}
If bit 'LBI' is set to '0' in register CONF (RESET value) pins 78..85 provide the General Purpose Port (GPP) pins 0..7. In this case a Pull-Up resistor to V_{DD3} is recommended for unused pins.

Table 5
Synchronous Serial Control (SSC) Interface/General Purpose Bus Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
100	110	MCLK/	I/O	**SSC Shift Clock Input/Output**
		GP15	I/O	**On reset, pin is General Purpose Bus pin**
99	109	MTSR/	I/O	**SSC Master Transmit/Slave Receive**
		GP14	I/O	**On reset, pin is General Purpose Bus pin**
98	108	MRST/	I/O	**SSC Master Receive/Slave Transmit**
		GP13	I/O	**On reset, pin is General Purpose Bus pin**
97	106	N.C.3/	I/O	Reserved when in SSC Mode
		GP12	I/O	**On reset, pin is General Purpose Bus pin**
91	101	$\overline{\text{MCS0}}$/	O	**SSC Chip select 0**
		GP11	I/O	**On reset, pin is General Purpose Bus pin**
90	100	$\overline{\text{MCS1}}$/	O	**SSC Chip select 1**
		GP10	I/O	**On reset, pin is General Purpose Bus pin**
89	99	$\overline{\text{MCS2}}$/	O	**SSC Chip select 2**
		GP9	I/O	**On reset, pin is General Purpose Bus pin**
88	98	$\overline{\text{MCS3}}$/	O	**SSC Chip select 3**
		GP8	I/O	**On reset, pin is General Purpose Bus pin**

Note: Pull-Up resistors to V_{DD3} are recommended for unused pins independent of whether they are configured as General Purpose Port (GPP) pins 8..15 (RESET value) or as Synchronous Serial Control (SSC) interface via bit 'SSC' in register CONF.

Table 6
PCM/IOM®-2 Interface Pins

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
108	118	RXCLK/	I	**Receive Clock** Provides the data clock for RXD T1/DS1 24-channel 1.544 MHz 24-channel 1.536 MHz CEPT 32-channel 2.048 MHz 32-channel 4.096 MHz Additional new PCM modes: 3.088 MHz, 6.176 MHz, 8.192 MHz (refer to MODE1 register description)
		DCL	I/O	**IOM®-2 Data Clock**
107	117	RSP/	I	**Receive Synchronization Pulse** This signal provides the reference for the receive PCM frame synchronization. It marks the first bit in the PCM frame.
		FSC	I/O	**IOM®-2 Frame Synchronization**
106	116	RXD/	I	**Receive Data** Serial data is received at this PCM input port. The MUNICH32X supports the T1/DS1 24-channel PCM format, the CEPT 32-channel PCM format as well as a 32-channel PCM format with 4.096-Mbit/s bit rate.
		DD		**IOM®-2 Data Downstream**
101	111	TXCLK	I	**Transmit Clock** Provides the data clock for TXD (refer to RXCLK).
102	112	TSP	I	**Transmit Synchronization Pulse** This signal provides the reference for the transmit frame synchronization. It marks the last bit in the PCM frame.
103	113	TXD/	O	**Transmit Data** Serial data sent by this PCM output port is push-pull for active bits in the PCM frame and tristate for inactive bits.
		DU		**IOM®-2 Data Upstream**

Table 6
PCM/IOM®-2 Interface Pins (cont'd)

Pin No. P-MQFP-160-1	Pin No. P-TQFP-176-1	Symbol	I/O	Function
104	114	$\overline{\text{TXDEN}}$	O	**Transmit Data Enable** Indicates tristate of TXD
105	115	DRDY	I	**Data Ready** **A Pull-Up resistor to V_{DD3} is recommended if not used.**
	1 2 43 44 45 46 87 88 89 90 131 132 133 134 175 176	N.C.4 N.C.5 N.C.6 N.C.7 N.C.8 N.C.9 N.C.10 N.C.11 N.C.12 N.C.13 N.C.14 N.C.15 N.C.16 N.C.17 N.C.18 N.C.19		**Not connected pins** It is recommended not to connect these pins.

Note: As a general recommendation 10 K Ohm resistors connected to V_{DD3} should be used as Pull-Ups and 10 K Ohm resistors connected to V_{SS} should be used as Pull-Downs.

1.4 Logic Symbol

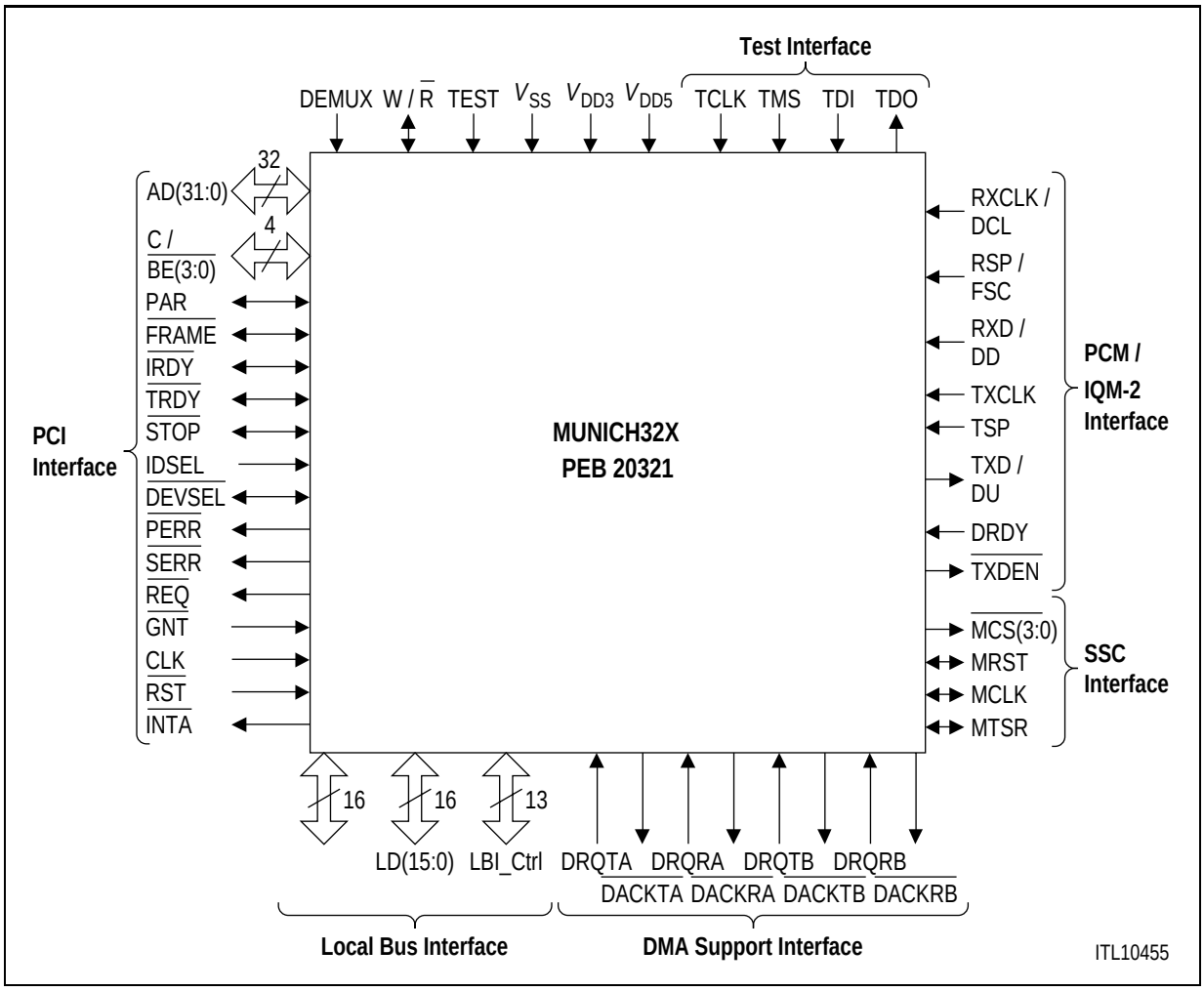


Figure 3
MUNICH32X Logic Symbol

Note: To reduce complexity, the De-multiplexed General Purpose Bus (refer to pin definition table) is not shown here.

1.5 Functional Block Diagram

The functional block diagrams are shown in **Figure 4** and **Figure 5**.

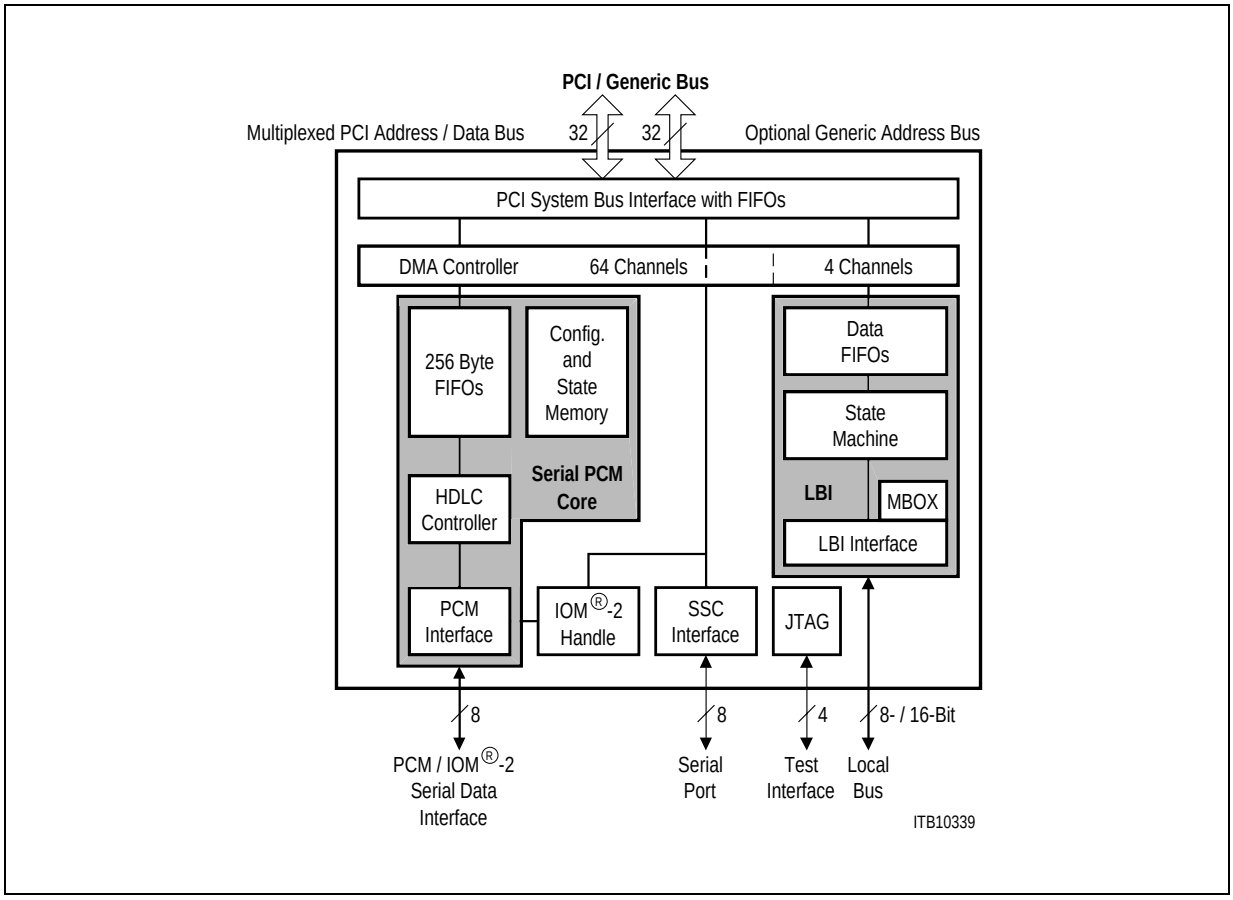


Figure 4
Functional Block Overview of MUNICH32X

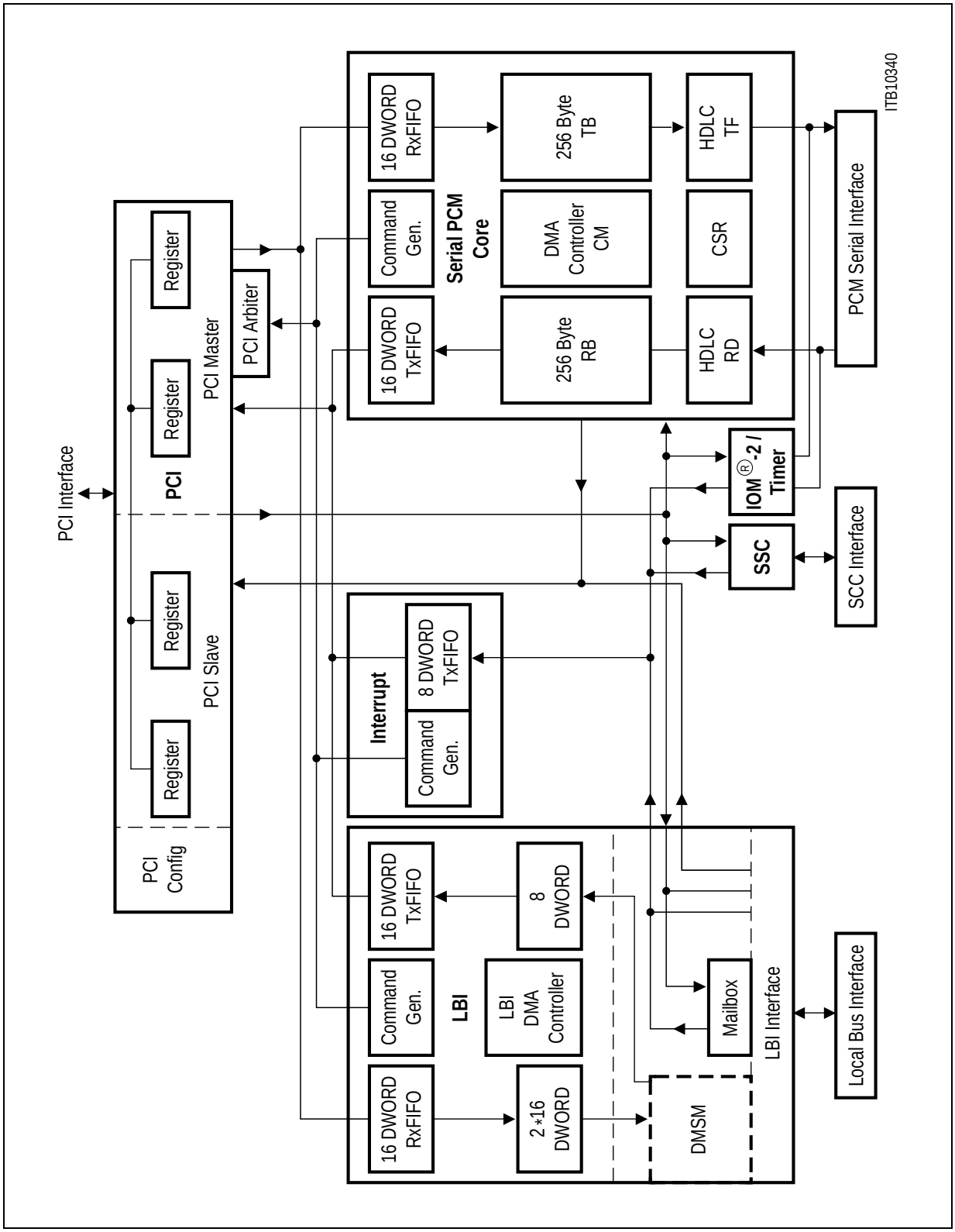


Figure 5
Block Diagram of MUNICH32X

The internal functions of the serial PCM core are partitioned into 8 major blocks:

1. PCM Serial Interface
   – Parallel-Serial conversion, PCM timing, switching of the test loops, controlling of the multiplex procedure.
2. Transmit Formatter TF
   – HDLC frame, bit stuffing, flag generation, flag stuffing and adjustment, CRC generation, transparent mode transmission and V.110, X.30 80 bit framing.
3. Transmit Buffer TB
   – Buffer size of 64 DWORDs allocated to the channels, i.e. eight PCM frames can be stored before transmission, individual channel capacity programmable.
4. Receive Deformatter RD
   – HDLC frame, zero-bit deletion, flag detection, CRC checking, transparent mode reception and V.110, X.30 80 bit framing.
5. Receive Buffer RB
   – Buffer size of 64 DWORDs allocated to the channels, i.e. eight PCM frames can be stored, individual DWORDs are freely accessible by each channel.
6. Configuration and State RAM CSR
   – Since the Transmit Formatter, Receive Deformatter are used in a multiplex manner, the state and configuration information of each channel has to be stored.
7. DMA Controller CM/DMAC
   – Interrupt processing, memory address calculation, chaining list handling, chip configuration.
8. Internal Bus Interface to PCI (IBUS)
   – On-chip interface, which connects the new functional blocks (LBI, SSC, IOM®-2, Global Registers) to the MUNICH32X core; 32-bit de-multiplexed address/data, control signals provided in little-endian format, 33 MHz, synchronous non-burst mode, bus arbitration provided.

Note that the structure and functionality of all other MUNICH32X blocks (IOM®-2, SSC, LBI) are described in the appropriate sections of this manual.

1.6 System Integration

The MUNICH32X is designed to handle up to 32 data channels of a PCM highway. It transfers the data between the PCM highway and a memory shared with a host processor via a 32-bit PCI bus interface (33 MHz). At the same time it performs protocol formatting and deformatting as well as rate adaption for each channel independently. The host sets the operating mode, bit rate adaption method and time slot allocation of each channel by writing the information into the shared memory.

Using subchanneling each time slot can be shared between up to four MUNICH32Xs; so that in one single time slot four different D-channels can be handled by four MUNICH32Xs.

Figure 6 gives a general overview of system integration of the MUNICH32X.

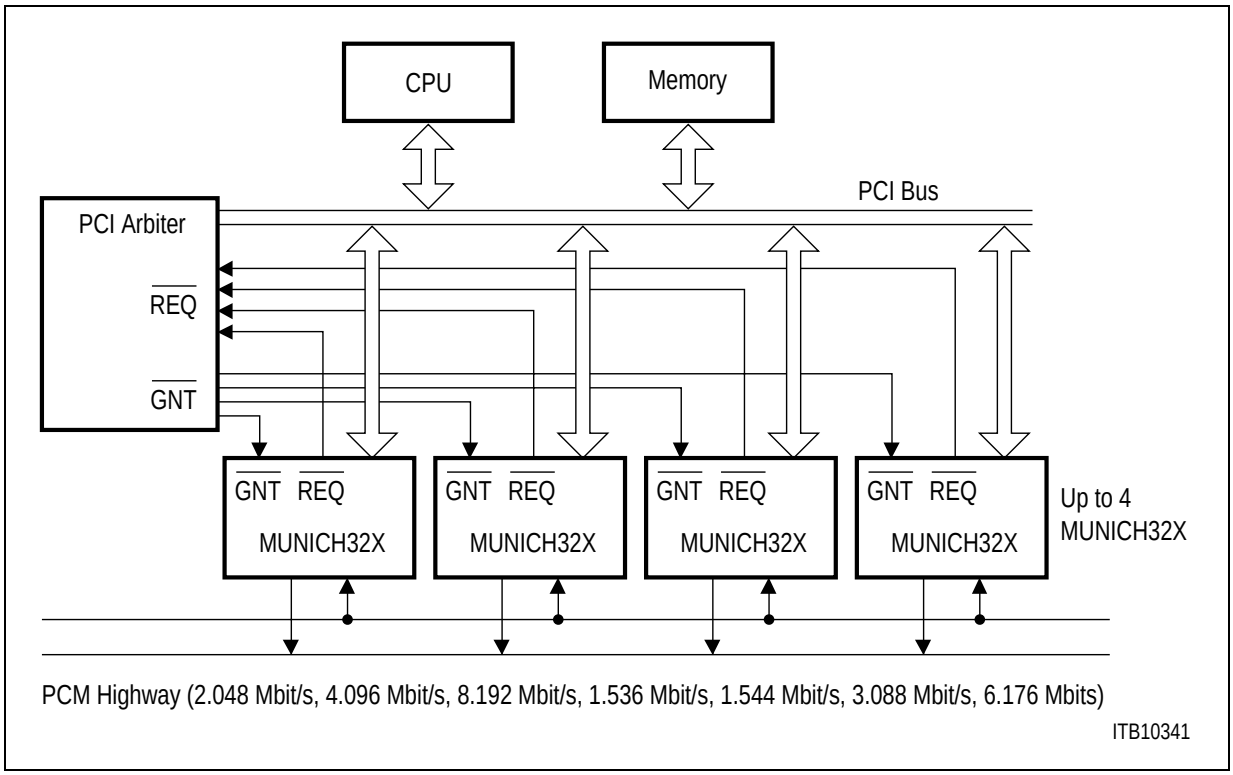


Figure 6
General System Integration

The PCI bus interface of the MUNICH32X consists of a 32 bit multiplexed data and address bus (AD31 … AD0), four command/byte enable lines C/$\overline{BE}$(3:0), PCI control and bus management lines (PAR, $\overline{FRAME}$, $\overline{IRDY}$, $\overline{TRDY}$, $\overline{STOP}$, IDSEL, $\overline{DEVSEL}$, $\overline{PERR}$, $\overline{SERR}$, $\overline{REQ}$, $\overline{GNT}$), one clock, one reset and one interrupt line.

The PCI bus traffic is controlled by the PCI arbiter. It manages the bus request/grant control for up to four independent PCI devices, hence supporting the connection of four MUNICH32X as shown above.

2 Serial PCM Core

The Serial PCM core provides up to 32 full-duplex channels. The serial PCM interface includes a Rx data (RXD) and a Tx data line (TXD) as well as the accompanying control signals (RXCLK = Receive Clock, RSP = Receive Synchronization Pulse, TXCLK = Transmit Clock, TSP = Transmit Synchronization Pulse). The timings of the receive and transmit PCM highway are independent of each other, i.e. the frame positions and clock phases are not correlated. Data is transmitted and received either at a rate of 2.048 Mbit/s for the CEPT 32-Channel European PCM format (**Figure 9**) or 1.544 Mbit/s or 1.536 Mbit/s for the T1/DS1 24-Channel American PCM format (**Figure 7** and **Figure 8**). The MUNICH32X may also be connected to a 4.096-Mbit/s PCM system (**Figure 10**), where it handles either the even- or odd-numbered time slots, so all 64 time slots can be covered by connecting two MUNICH32Xs to the PCM highway.

The MUNICH32X also supports three additional PCM highway modes: 3.088 Mbit/s, 6.176 Mbit/s and 8.192 Mbit/s (**Figure 11**).

The actual bit rate of a time slot can be varied from 64 Kbit/s down to 8 Kbit/s for the receive and transmit direction. A fill mask code specified in the time slot assignment determines the bit rate and which bits of a time slot should be ignored. Any of these time slots can be combined to a data channel allowing transmission rates from 8 Kbit/s up to 2.048 Mbit/s.

The frame alignment is programmable via register **MODE1**. Receive and transmit data may be sampled at either rising or falling clock edge, programmable in register **MODE2**. Note the MUNICH32X may be configured to be fully compatible to the MUNICH32, PEB 20320.

The F-bit for the 1.544 MHz T1/DS1 24-channel PCM format is ignored in receive direction, the corresponding bit is tristate in transmit direction. It is therefore assumed that this channel is handled by a different device.

For test purposes four different test loops can be switched. In a complete loop all logical channels are mirrored either from serial data output to input (internal loop) or vice versa (external loop).

In a channelwise loop one single logical channel is logically mirrored either from serial data output to input (internal loop) or vice versa (external loop).

For a more detailed description of the different loops see **Section 12.2**.

The following drawings show examples for transmit situations in different PCM modes. Note that **Figure 7 … Figure 10** address the case, in which the MUNICH32X is programmed in MUNICH32 mode.

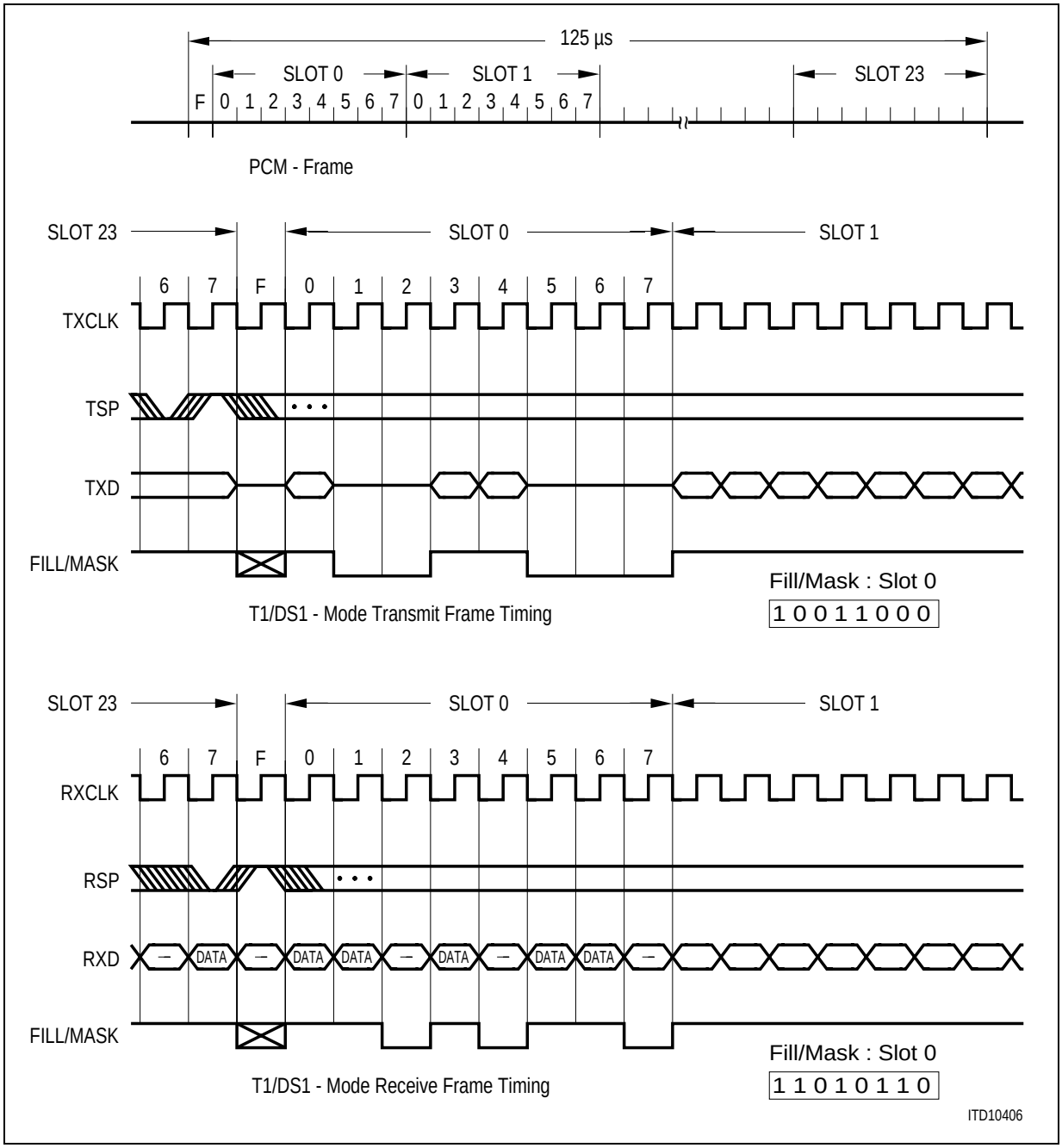


Figure 7
T1/DS1 Mode PCM Frame Timing 1.544 MHz

Note 9: A ⟨—⟩ box in a bit of the RXD line means that this bit is ignored.

Note 10: The fill/mask bit for the F-bit is not defined. TXD is tristate for the F-bit, and the F-bit is ignored in the receive direction.

Note 11: TSP and RSP must have one single rising and falling edge during a 125 µs PCM frame.

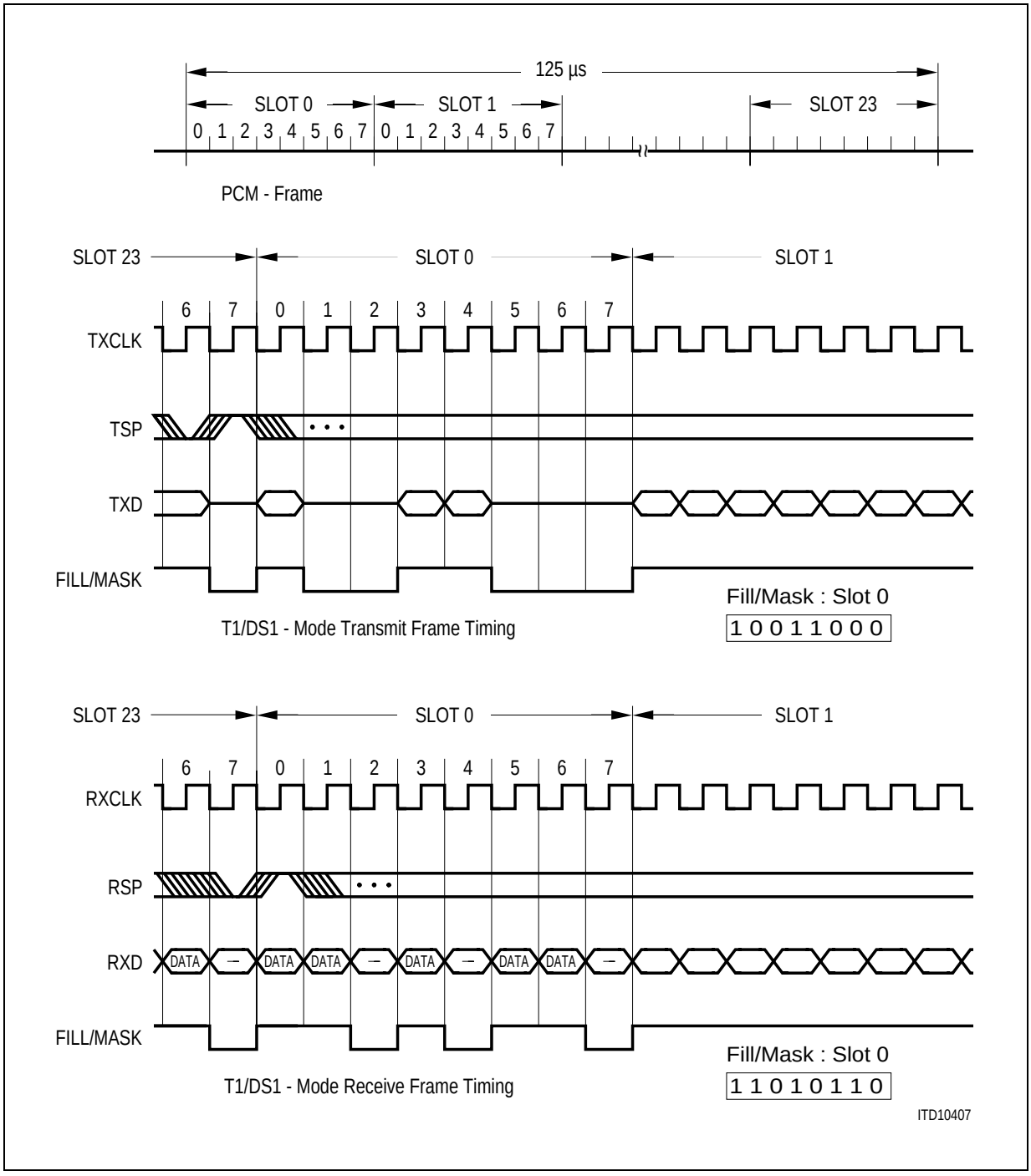


Figure 8
T1/DS1 Mode PCM Frame Timing 1.536 MHz

Note 1: A box in a bit of the RXD line means that this bit is ignored.
Note 2: TSP and RSP must have one single rising and falling edge during a 125 µs PCM frame.

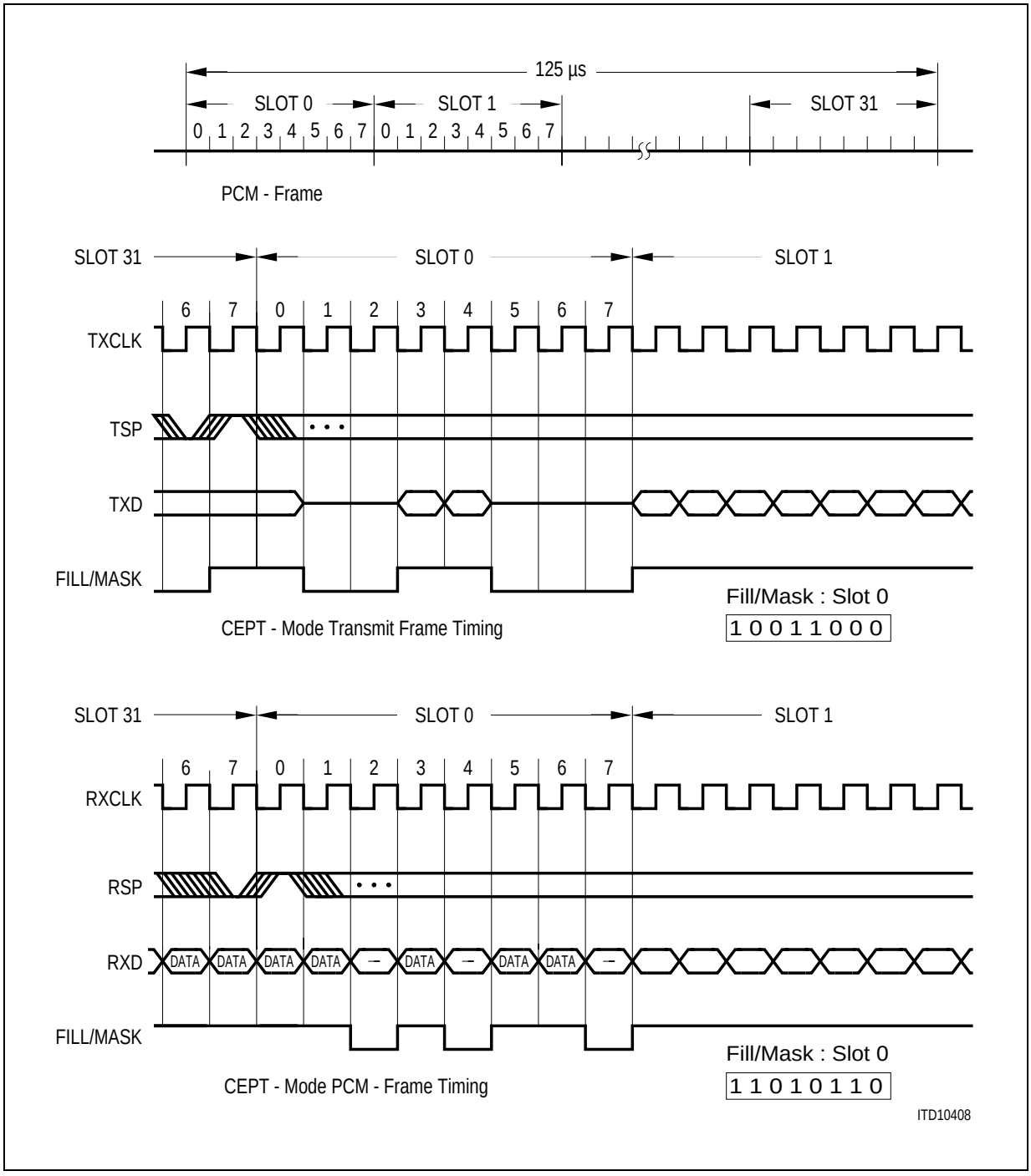


Figure 9
CEPT Mode PCM Frame Timing

Note 1: A box in a bit of the RXD line means that this bit is ignored.
Note 2: TSP and RSP must have one single rising and falling edge during a 125 µs PCM frame.

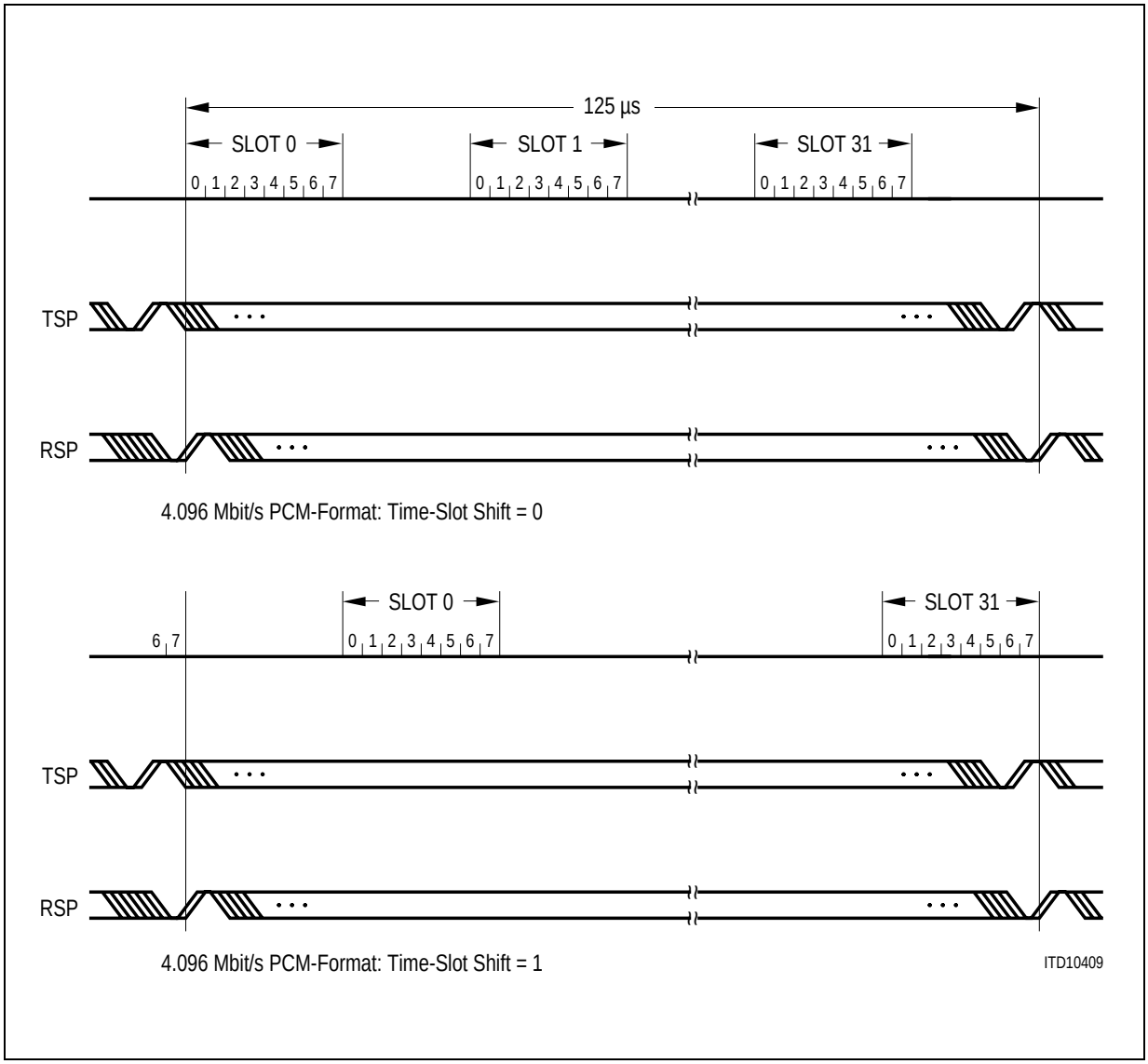


Figure 10
4.096 Mbit/s PCM Frame Timing

Note: TSP and RSP must have one single rising and falling edge during a 125 µs PCM frame.

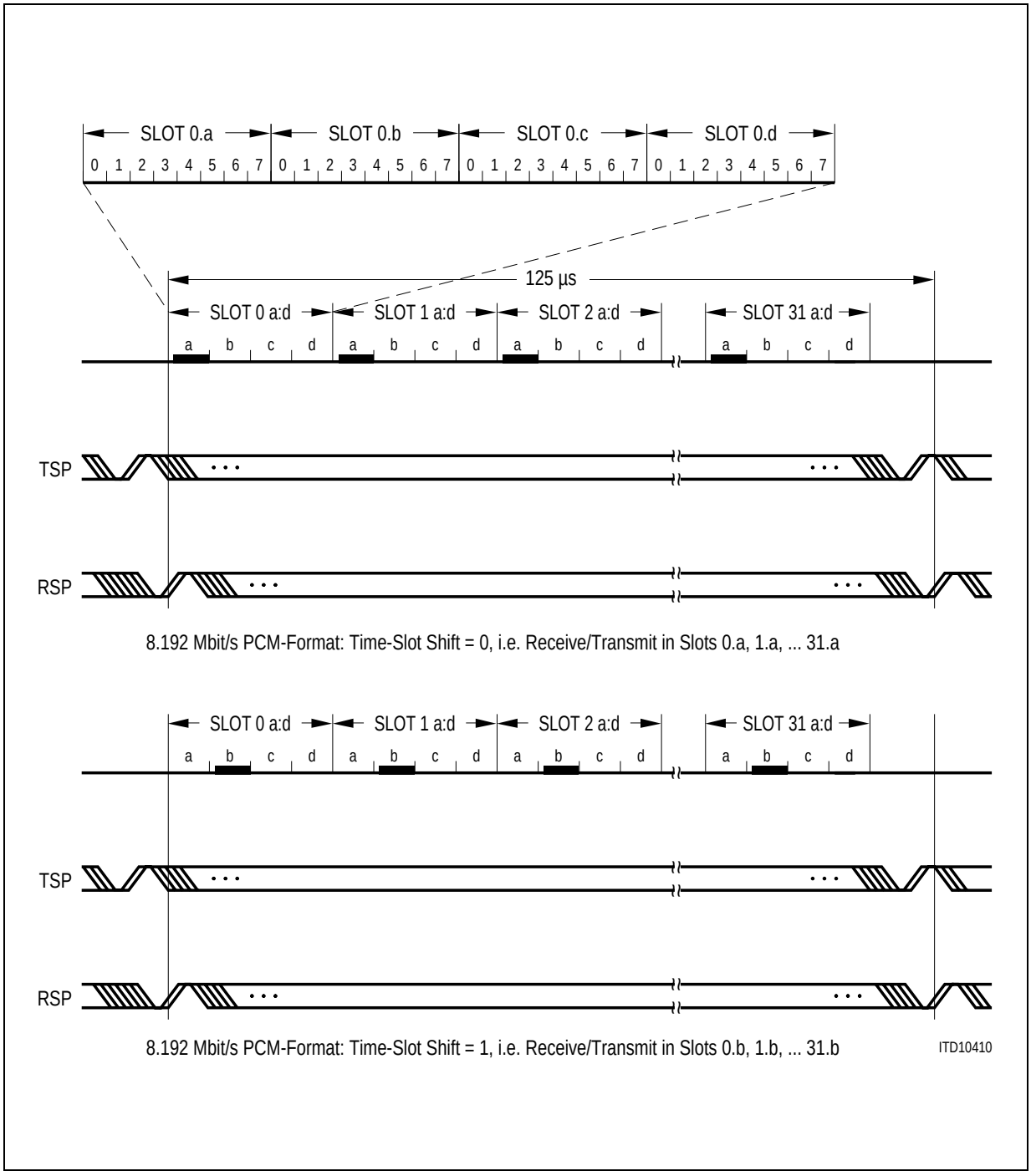


Figure 11
8.192 Mbit/s PCM Frame Timing

Note: TSP and RSP must have one single rising and falling edge during a 125 µs PCM frame.

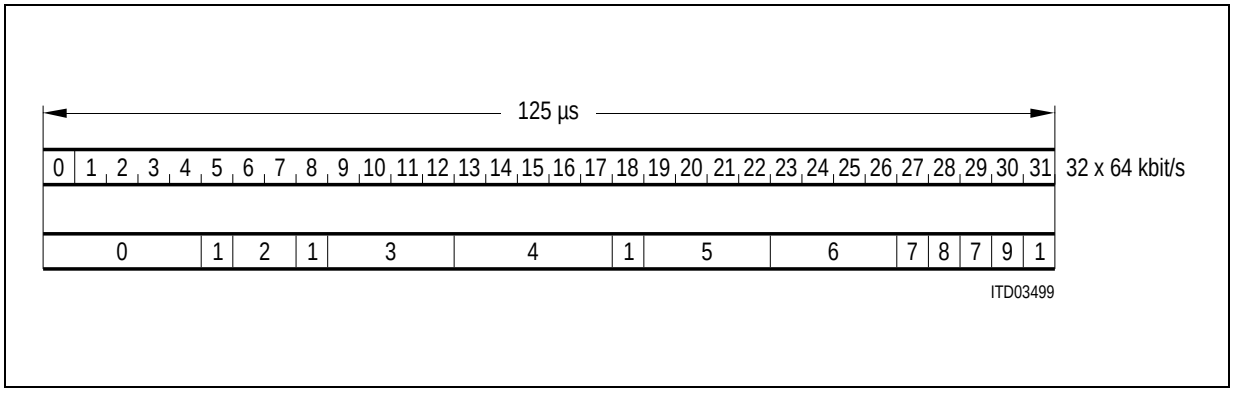


Figure 12
Example: Programmable Channel Allocation for 32 Time Slots

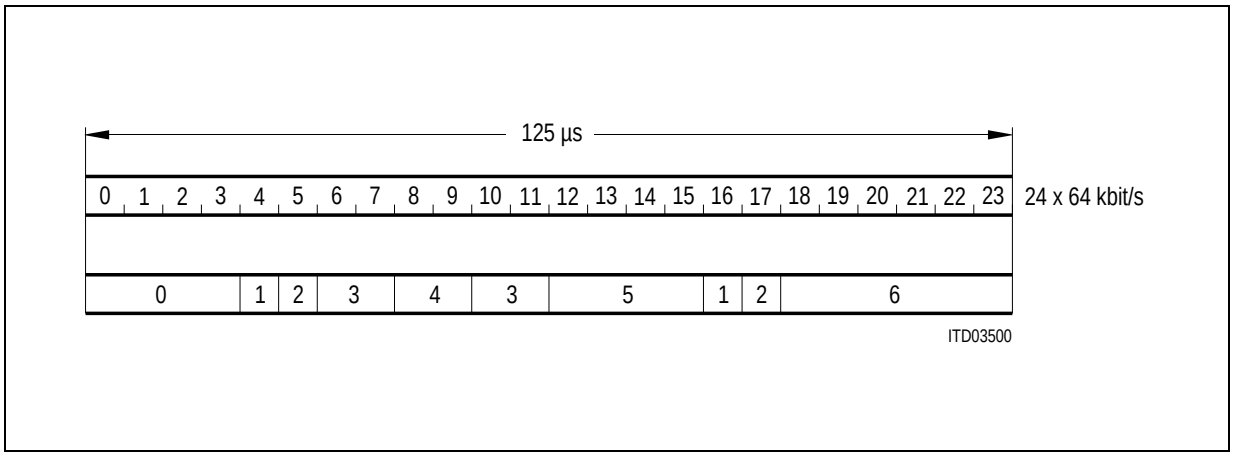


Figure 13
Example: Programmable Channel Allocation for 24 Time Slots

3 Basic Functional Principles

The MUNICH32X is a Multichannel Network Interface Controller for HDLC, offering a variety of additional features like subchanneling, data channels comprising of one or more time slots, DMI 0, 1, 2 transparent or V.110/X.30 transmission and programmable rate adaption. MUNICH32X performs formatting and deformatting operations in any network configuration, where it implements, together with a microprocessor and a shared memory, the bit oriented part (flag, bit stuffing, CRC check) of the layer 2 (data link protocol level) functions of the OSI reference model.

The block diagram is shown in **Figure 5**. The MUNICH32X is designed to handle up to 32 data channels of a 1.536/1.544 Mbit/s T1/DS1 24-channel, 2.048 Mbit/s CEPT 32-channel, 3.088/6.176 Mbit/s 24-channel or a 4.096/8.192 Mbit/s 32-channel PCM highway. The device provides transmission for all bit rates from 8 Kbit/s up to 2.048 Mbit/s of packed data in HDLC format or of data in a transparent format supporting the DMI mode (0, 1, 2) or V.110/X.30 mode. Tristating of the transmission line as well as switching a channelwise or complete loop are also possible. An on-chip 64-channel DMA generator controls the exchange of data and channel control information between the MUNICH32X and the external memory.

The MUNICH32X processes receive and transmit data independently for each time slot and transmission direction respectively (blocks TF = Transmit Formatter, RD = Receive Deformatter). The frame counters are reset by the rising edges of the RSP or TSP line. The processing units TF and RD work with a multiplex management, i.e. only one protocol handler exists, which is used by all channels in a time sharing manner (see **Figure 14** and **Figure 15**). The actual configuration, e.g. transmission mode, channel assignment, fill/mask code or state of the protocol handlers is retrieved from the Configuration and State RAM (CSR) at the beginning of the time slot and reloaded to the CSR at the end. In receive direction, 32 unpacked data bits are first accumulated and then stored into an on-chip receive buffer (RB) for transfer to the shared memory. As soon as the RB receives 32 bits for a channel it requests access to the parallel microprocessor bus. The on-chip transmit buffer (TB) is always kept full of data ready for transmission. The TB will request more data when 32 bits become available in the ITBS (refer to channel specification). These buffers allows a flexible access to the shared memory in order to prevent data underflow (Tx direction) and data overflow (Rx direction).

The transmit buffer (TB) has a size of 64 DWORDs (= 256 bytes). In this buffer, data of 8 PCM frames can be stored. In this case, the time between accesses to the shared memory and data supply to the Transmit Formatter is max. 1 ms. In order to meet these requirements, a variable and programmable part of the buffer (ITBS) must be allocated to each data channel (see **Figure 16**).

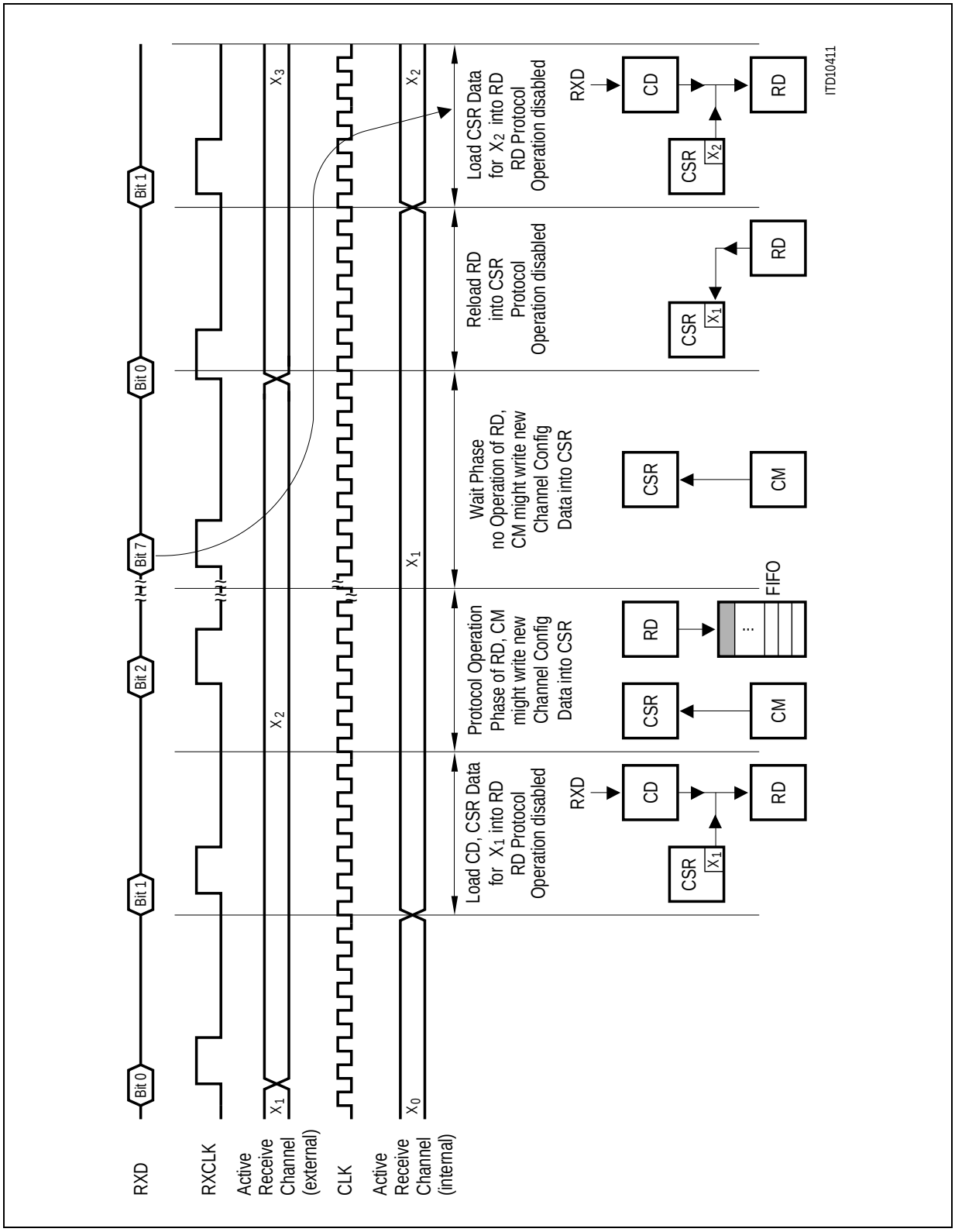


Figure 14
Multiplex Management Receive Direction

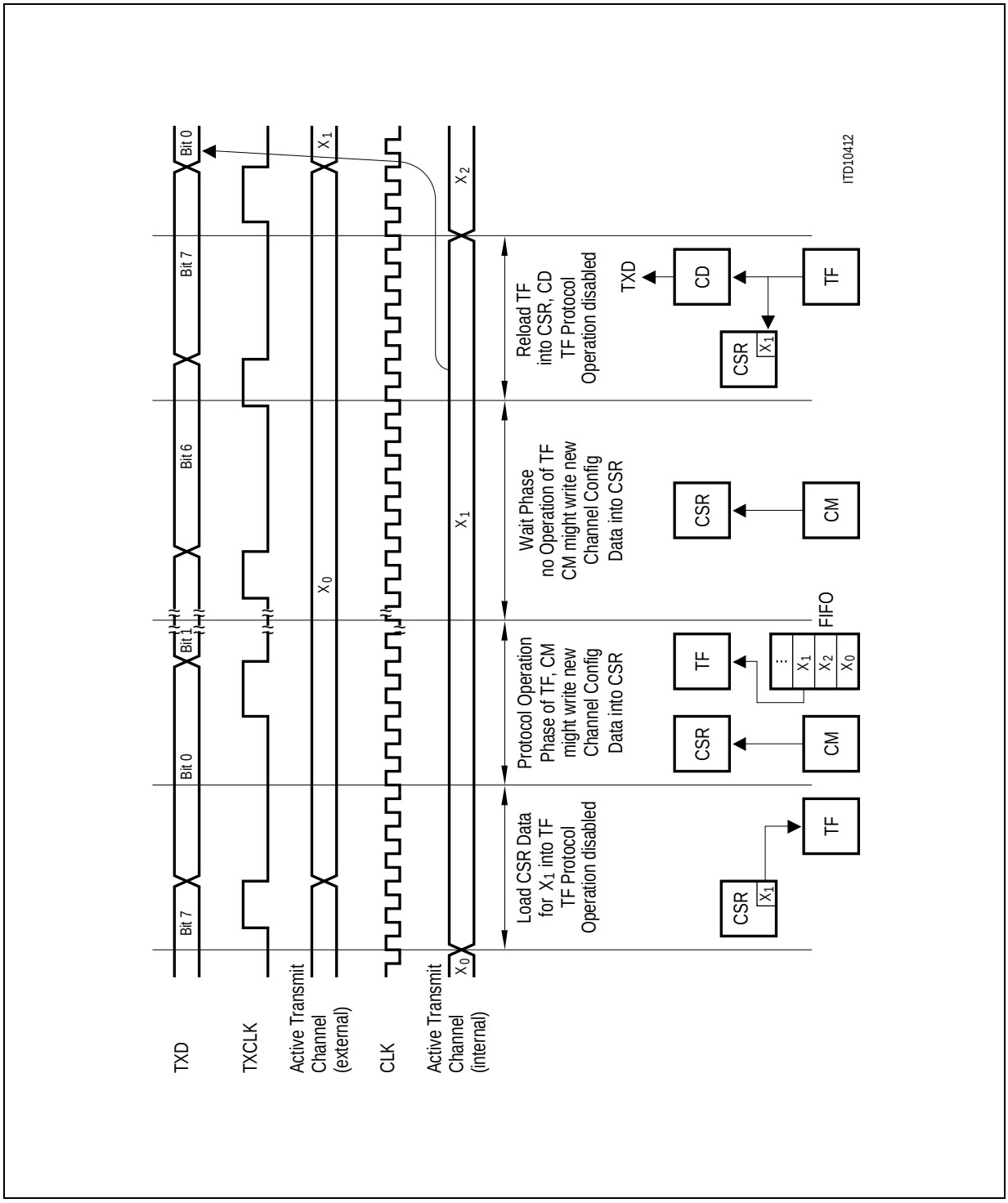


Figure 15
Multiplex Management Transmit Direction

For example:

a) 2.048-Mbit/s PCM highway
   32 × 64-Kbit/s data channels (8 bits are sent with each PCM frame). Two DWORDs of the buffer are allocated to each data channel.

b) 1 × 2.048-Kbit/s data channel
   The maximum buffer size for one channel (64 DWORDs) is allocated to this data channel.

c) 6 × 256 -Kbit/s and 8 × 64 Kbit/s data channels.
   Eight DWORDs of the buffer are allocated to each of the 6 data channels with 256 Kbit/s and two DWORDs are assigned to each of the 8 data channels with a transmission rate of 64 Kbit/s.

The choice of the individual buffer size of each data channel can be made in the channel specification (shared memory). The buffer size of one channel is changeable without disturbing the transmission of the other channels.

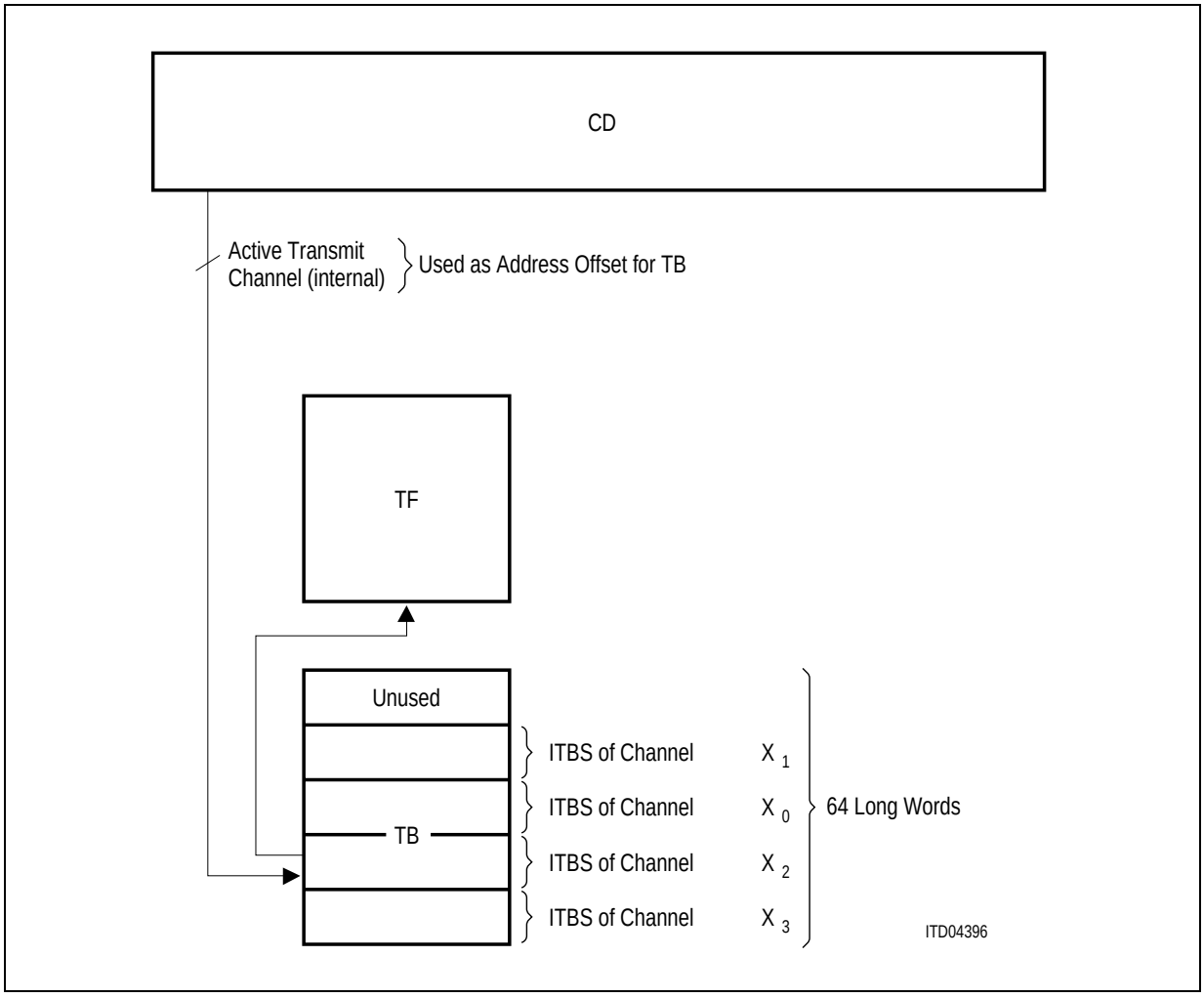


Figure 16
Partitioning of TB

The receive buffer (RB) is a FIFO buffer and has also a size of 64 DWORDs, which allows storing the data of eight complete PCM frames before transferring to the shared memory.

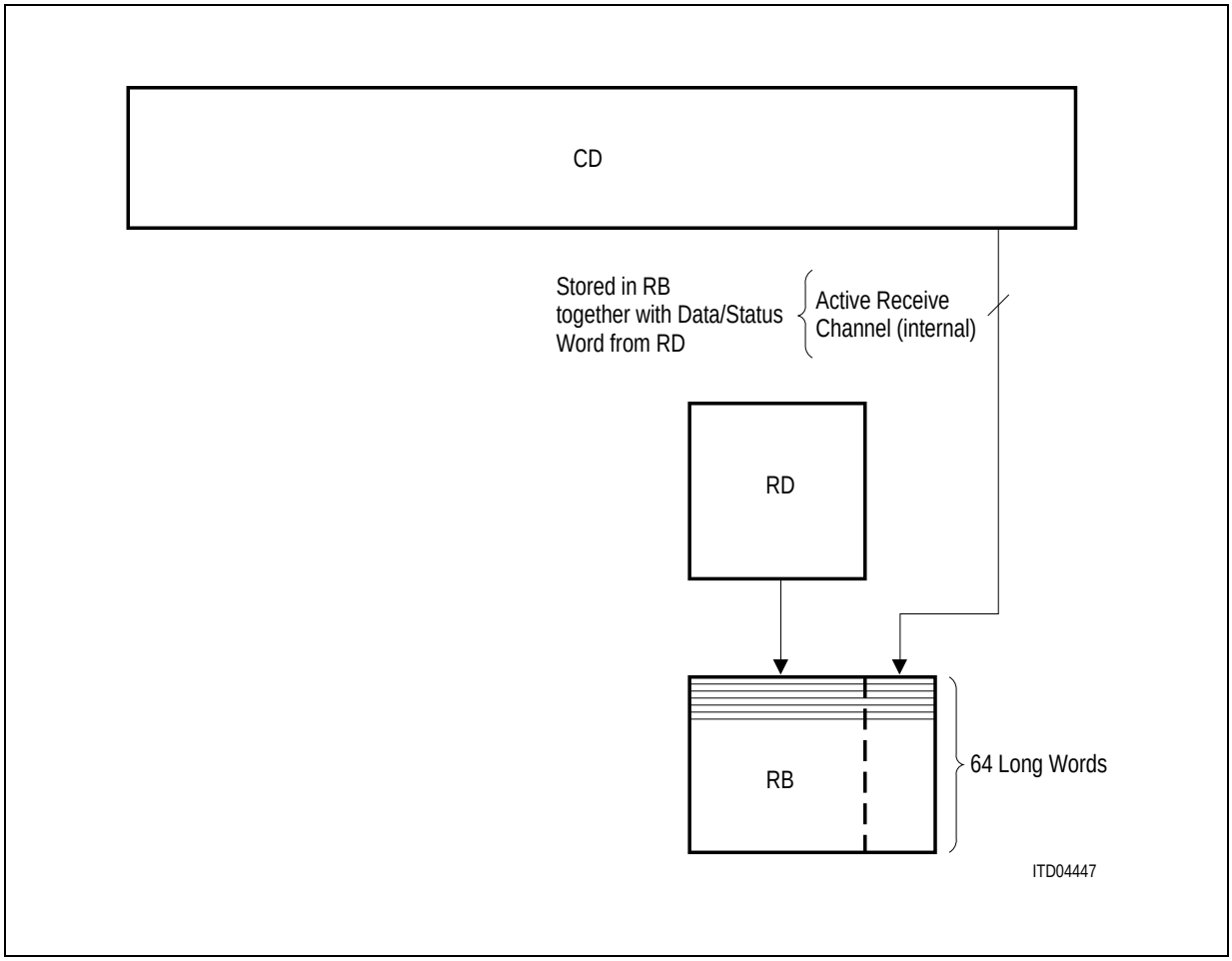


Figure 17
Partitioning of RB

The data transfer to the shared memory is performed via a 32-bit PCI interface. **Figure 18** shows the division of the shared memory required for each MUNICH32X when using the serial PCM interface:

– Configuration start address located at a programmable address in **Section CCBA** register
– Control and Configuration Block (CCB)
– Several interrupt circular queues with variable size for PCM Rx, PCM Tx, LBI Rx, LBI Tx, and peripherals on SSC, IOM®-2
– Descriptor and data sections for each channel

Note that the LBI Control and Configuration Block (LCCB) differs from the CCB. Please refer to **Chapter 12.1.2**.

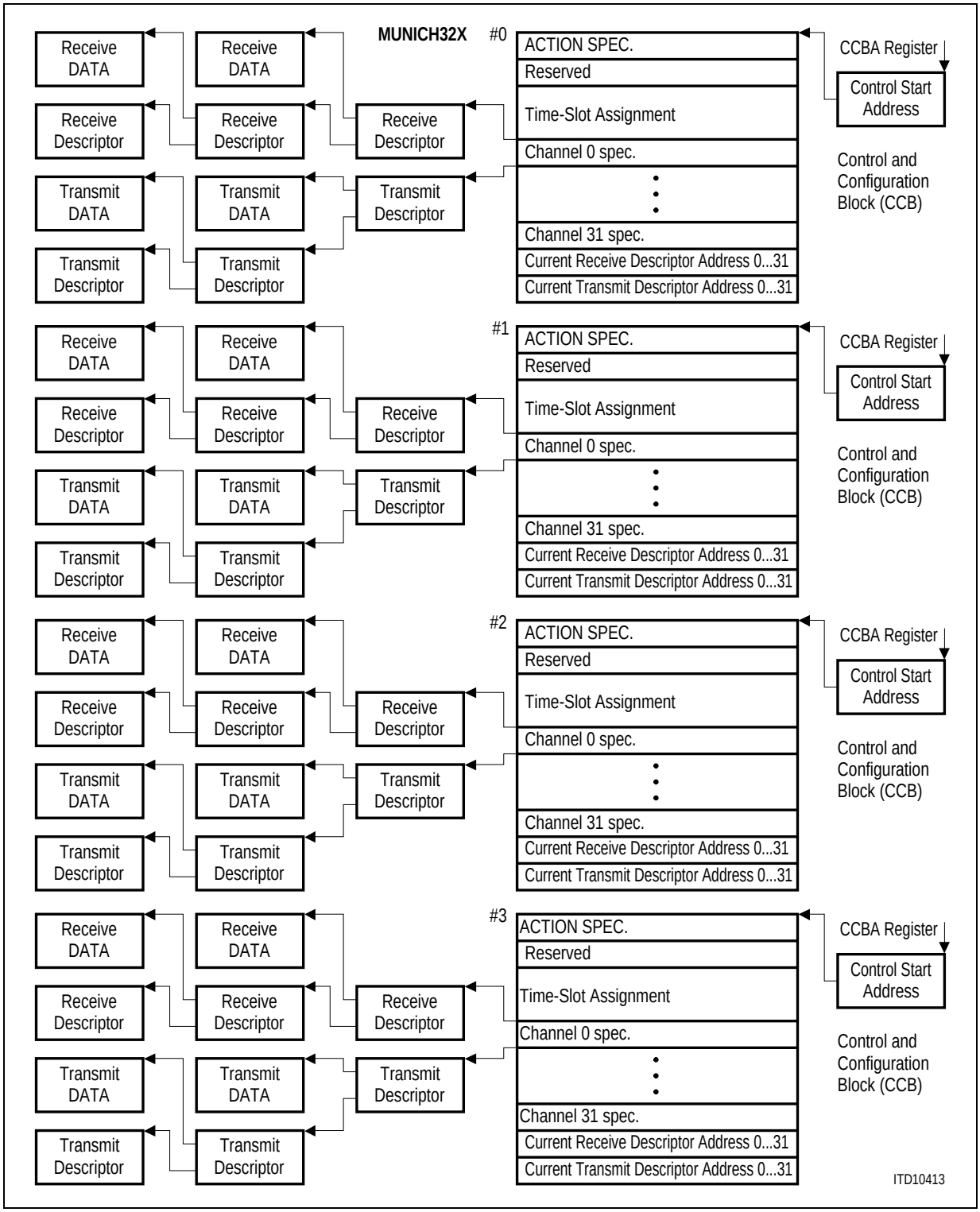


Figure 18
Memory Division (Serial PCM Core) for up to four MUNICH32X

Note: To reduce complexity, the interrupt queues are not shown here.

The shared memory allocated for each TX and Rx channel is organized as a chaining list of buffers set up by the host. Each chaining list is composed of descriptors and data sections. The descriptor contains the pointer to the next descriptor, the start address and the size of a data section. It also includes control information like frame end indication, transmission hold and rate adaption with interframe time-fill.

In the transmit direction the MUNICH32X reads a Tx descriptor, calculates the data address, writes the current Tx descriptor address into the CCB, and fills the on-chip Tx buffer. When the data transfer of the specified section is completed, the MUNICH32X releases the buffer, and branches to the next Tx descriptor.

If a frame end is indicated, the HDLC, TMB or TMR frame will be terminated and a specified number of the interframe time-fill bytes will be sent in order to perform rate adaption.

If frame end is found in a Tx descriptor of a TMA channel, the specified number of programmable TMA flags is appended to the data in the descriptor.

If frame end is found in a Tx descriptor of a V.110/X.30 channel, the frame is aborted (after the data in the descriptor are sent) by finishing the current 10-octet frame with ‘zeros’ and sending 2 more 10-octet frames with ‘zeros’ which leads to a loss of synchronism on the peer side. An adjustment for the inserted zeros in HDLC is programmable, which leads to a reduction of the specified number of interframe time-fill by $^{1}/_{8}^{th}$ of the number of zero insertions. This can be used to send long HDLC frames with a more or less fixed data rate in spite of the zero insertions. A maskable interrupt is generated before transmission is started again.

Examples of Typical Transmit Situations for the Individual Modes (refer to Chapter 12.1 … Chapter 12.9)

Note: These examples apply only to situations in which the MUNICH32X is operated in MUNICH32 mode, i.e. ***TXPOLL****.*$\overline{POLLn}$ *bit field for channel n is reset, while the* ***MODE2****.HPOLL bit field is set.*

Variable Size Frame Oriented Protocols (HDLC, TMB, TMR)

Normal operation, handling of frame end (FE) indication and hold (HOLD) indication.

Note: 1. FNUM0 must be set to zero.

2. Flag = $7E_H$ for HDLC
00_H for TMB, TMR

IC = $7E_H$ for HDLC and IFTF = 0
FF_H for HDLC and IFTF = 1
00_H for TMB, TMR

3. After sending the FNUM2 – 1 IC characters the device starts polling the HOLD bit in the Tx descriptor once for each further sent IC character. It also reads again the pointer to the next Tx descriptor once with each poll of the hold indication. The pointer to the next transmit descriptor can be changed while HOLD = 1 is set. The value of the pointer, which is read in each poll where HOLD = 0, is used as the next descriptor address.
If more than 6 IC characters will be sent, the use of the slow poll option provided in ***TXPOLL*** *register should be considered as an alternative to using the descriptor HOLD bit, or polling should be avoided with a new mode. Please refer to* ***Section 11.2.2*** *for a detailed description of the polling mechanism.*

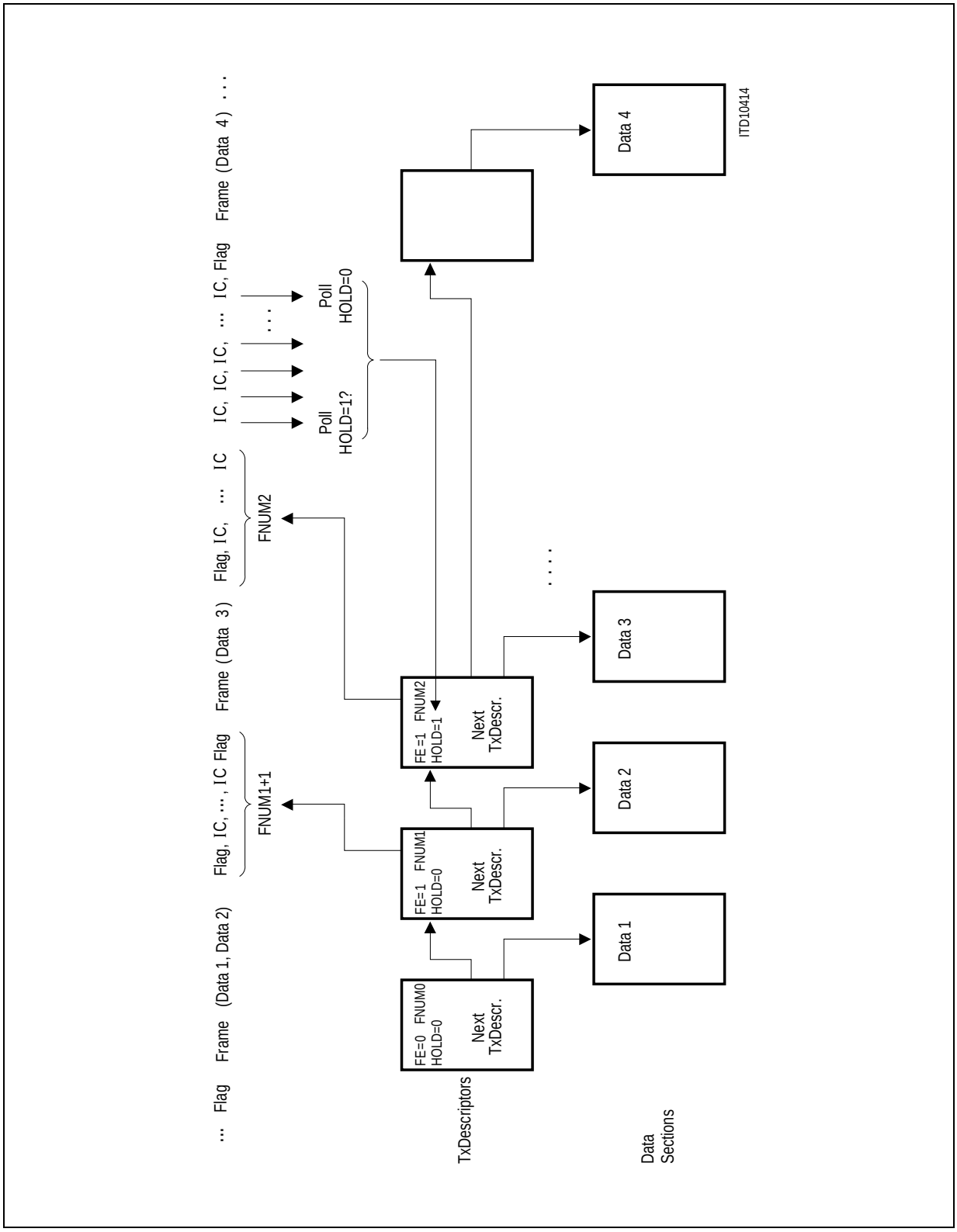


Figure 19
Handling of FE and HOLD Condition (Variable Size Frame Oriented Protocols)

Fixed Size Frame Oriented Protocols (V110/X.30)

Normal operation, E, S, X change (indicated by the V.110-bit in the transmit descriptor)

Example for TRV = ‘11’

Note: 1. FNUM must be 0 for all transmit descriptors.

2. *The actual E-, S-, X-bits have to be in the first transmit descriptor after reset.*
3. *As shown in the example the contiguous parts of a data section belonging to one descriptor are sent in contiguous frames (DATA 1(1) are the bytes 0 – 3 of DATA 1, DATA 1(2) are the bytes 4 – 7 of DATA 1). If the end of a data section is reached within a frame, the frame is continued with data from the next data section belonging to a transmit descriptor with the bit V.110 = 0 (DATA 2(2) = byte 4 of DATA 2, DATA 3(1) = byte 0 – 2 of DATA 3).*
4. *The E-, S-, X-bits are only changed from one frame to the next not within a frame. The change occurs in the first frame which does not contain data of the previous data section.*
5. *Neither FE nor HOLD may be set to 1 during a normal operation of the mode. They both lead to an abort of the serial interface.*

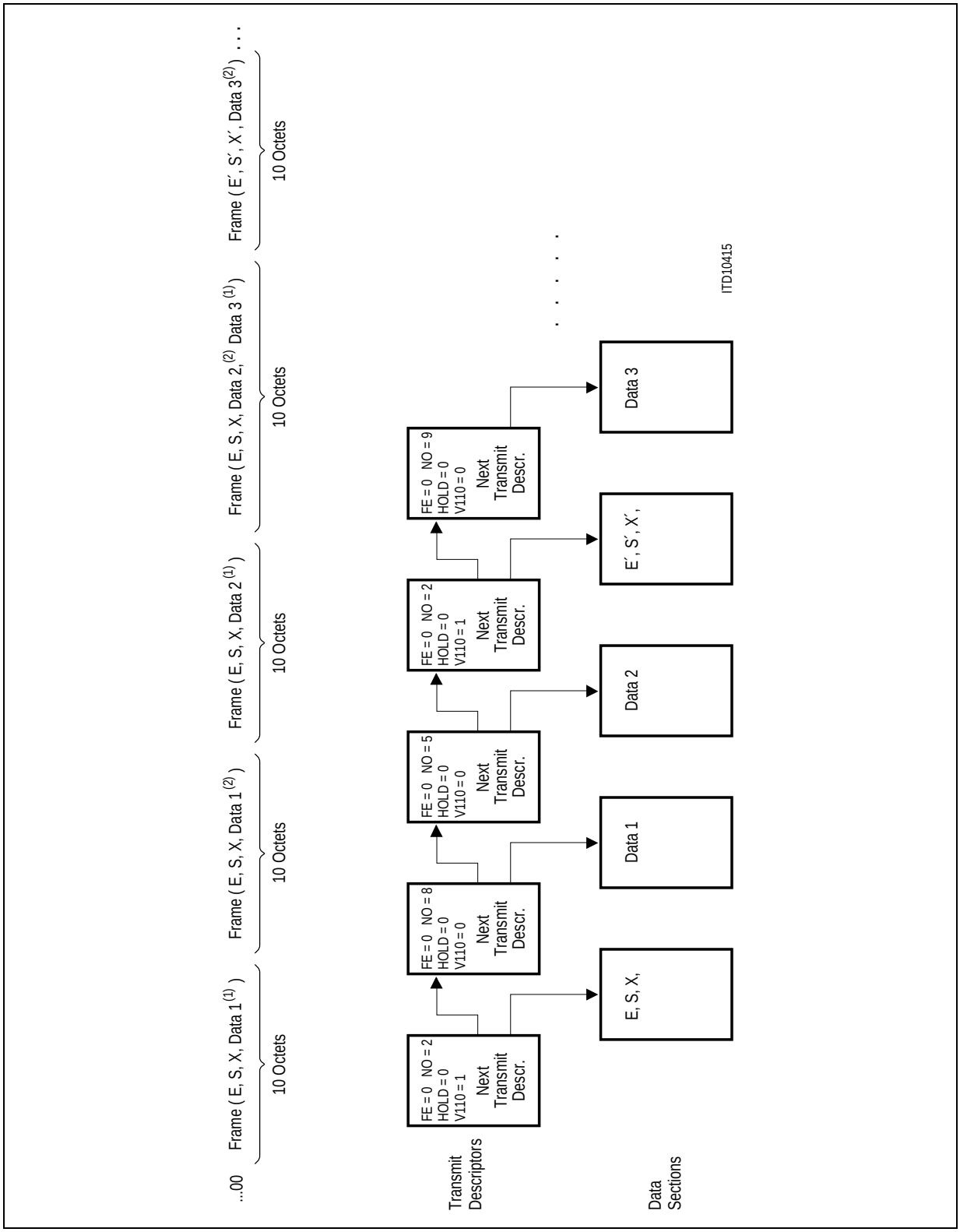


Figure 20
Handling of E, S, X Changes (Fixed Size Frame Oriented Protocols)

Fixed Size Frame Oriented Protocols (V.110/X.30)

Handling of frame end (FE) indication

Note: 1. FNUM must be '0' for all transmit descriptors.

2. The frame (E, S, X, DATA 2(2)) is the beginning of a 10-octet frame. It stops with the octet no. y, containing the last data bit of DATA 2 to be sent.

3. Since y = 1, …, 10 the 20 + y times 00_H characters sent afterwards cause the peer station to recognize 3 consecutive 10-octet frames with frame error which leads to a loss of synchronism in the peer station.

4. For y = 10 DATA 2 is identical to DATA 2(1) and 30 times 00_H characters are sent after frame (E, S, X, DATA 1(2), DATA 2(1)).

5. The E-, S-, X-bits are supposed to be loaded by an earlier transmit descriptor in the example. A descriptor changing them (with V.110-bit set) can be put between, before or after the descriptors in the example. It will change these bits according to the rules discussed previously.

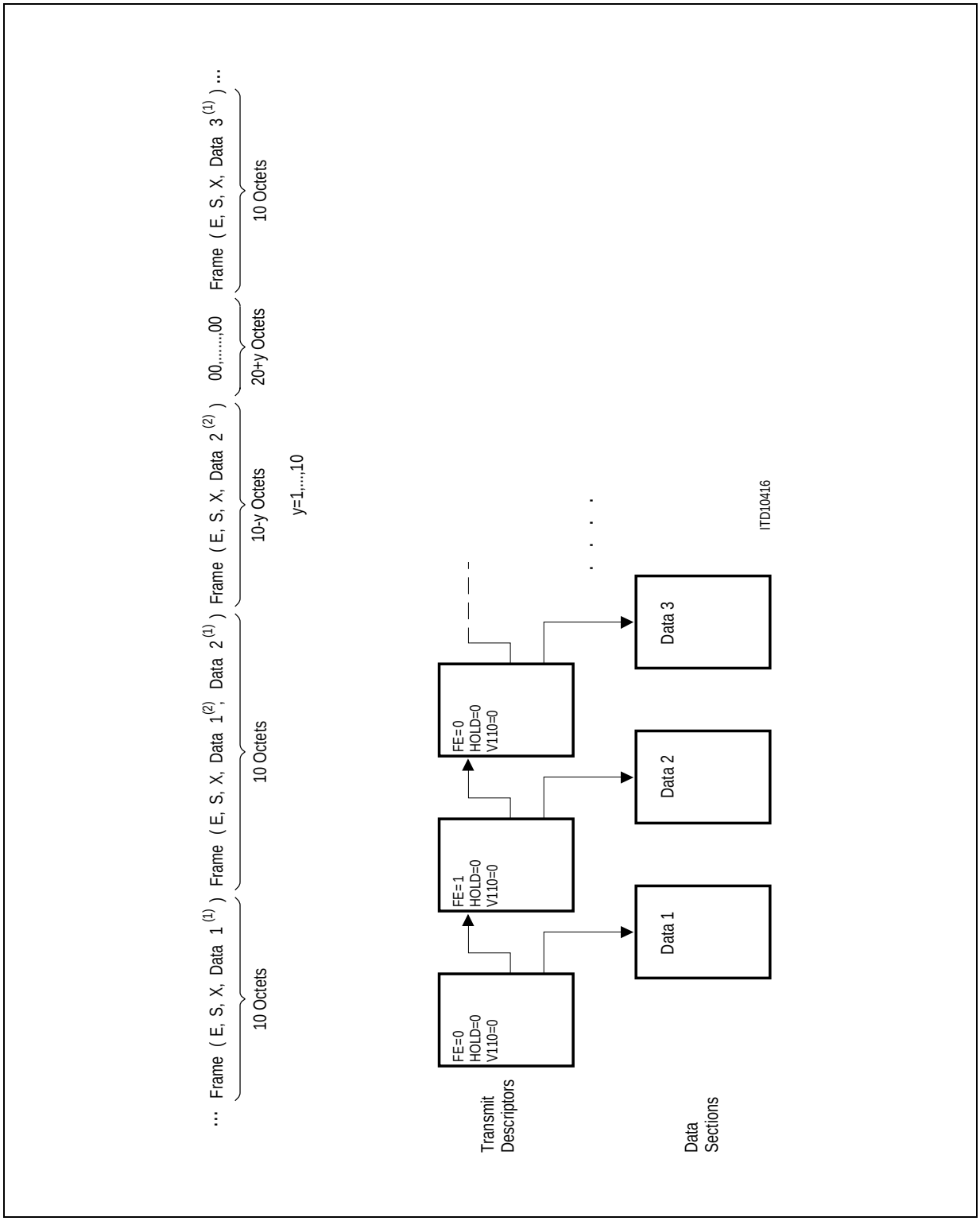


Figure 21
Handling of FE Condition (Fixed Size Frame Oriented Protocols)

Fixed Size Frame Oriented Protocols (V110/X.30)

Handling of hold (HOLD) indication

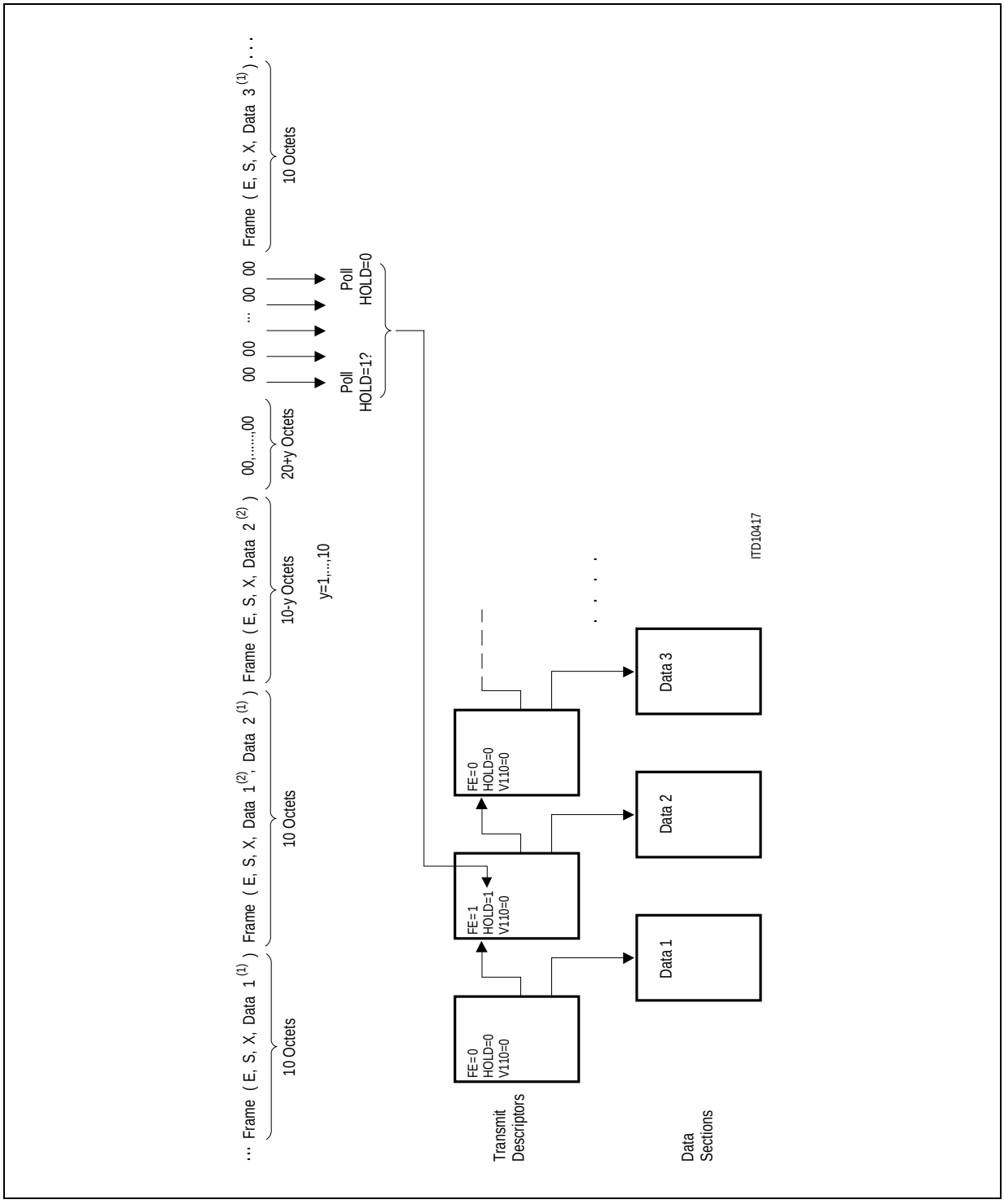


Figure 22
Handling of HOLD Condition (Fixed Size Frame Oriented Protocols)

Time Slot Oriented Protocol (TMA)

Normal operation, handling of frame end (FE) indication and hold (HOLD) indication.

Note: 1. FNUM must be set to zero.

2. TC = FF_H for TMA and FA = 0
the programmed flag with TMA and FA = 1

3. After sending the FNUM2 – 1 IC characters the device starts polling the HOLD bit in the Tx descriptor once for each further sent IC character. It also reads again the pointer to the next Tx descriptor once with each poll of the hold indication. The pointer to the next transmit descriptor can be changed while HOLD = 1 is set. The value of the pointer, which is read in each poll where HOLD = 0, is used as the next descriptor address.
*If more than 6 IC characters will be sent, the use of the slow poll option provided in **TXPOLL** register should be considered as an alternative to using the descriptor HOLD bit, or polling should be avoided with a new mode. Please refer to **Section 11.2.2** for a detailed description of the polling mechanism.*

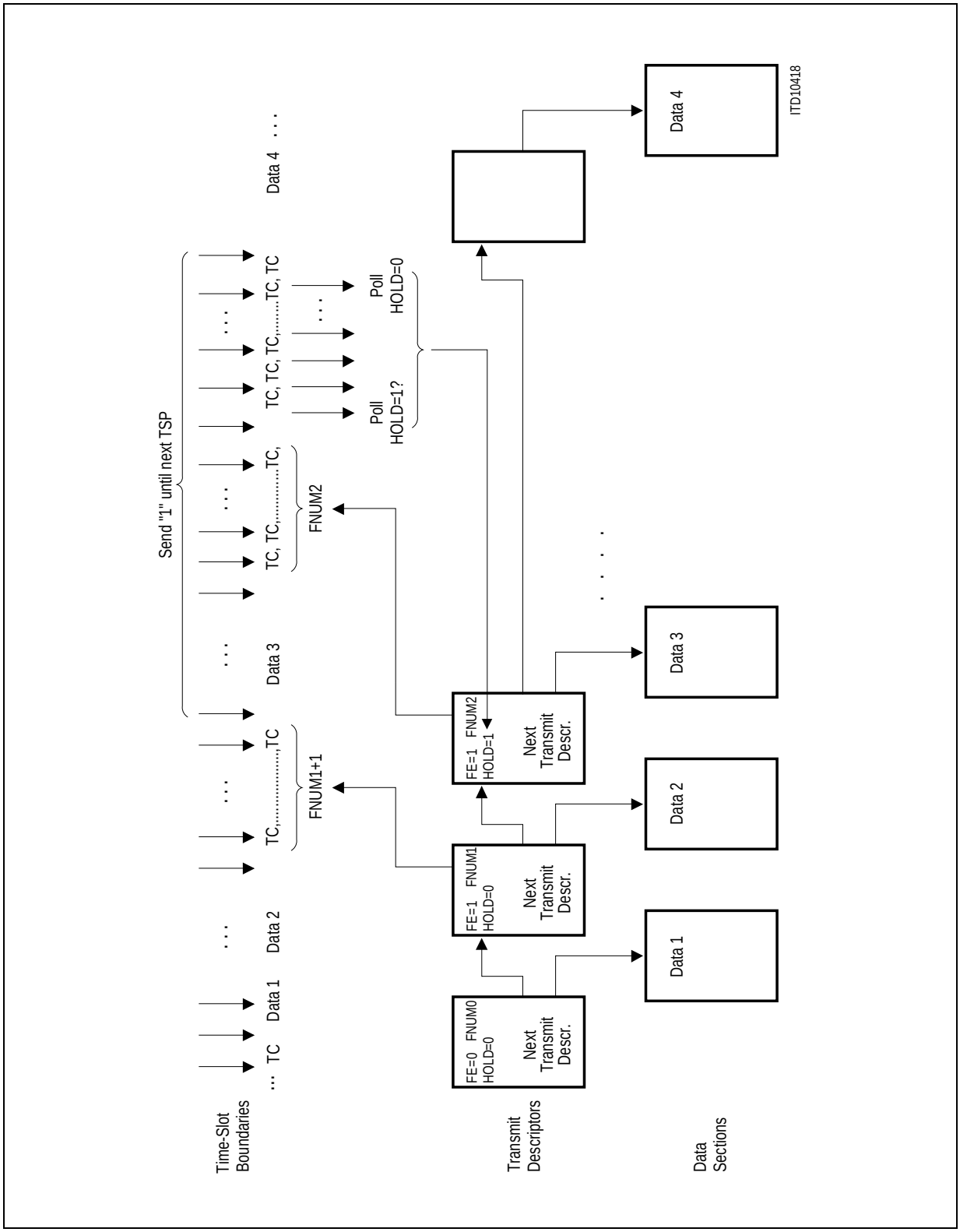


Figure 23
Handling of FE and HOLD Condition (Time Slot Oriented Protocol)

An activated transmission hold (HOLD bit in descriptor) prevents the MUNICH32X from sending more data. If a frame end has not occurred just before, the current frame will be aborted and an interrupt generated. Afterwards, the interframe time-fill bytes will be issued until the transmission hold indication is cleared. There is a further transmit hold (TH) bit in the Channel Specification in Control and Configuration Block (CCB) in addition to the HOLD bit in the descriptor. Setting the transmit hold (TH) bit by issuing a channel command will prevent further polling of the transmit descriptor.

This transmit hold bit is interpreted in the Formatter Controller CD (see **Figure 5**); it causes the Transmit Formatter (TF) to stay in the idle state and to send interframe time-fill after finishing the current frame. In the case of a very short frame (< ITBS), this frame will stay in the TF and not be sent until TH is removed. (In case of X.30/V.110 the current frame is aborted).

This means that the Transmit Buffer (TB) is not emptied from the TF side after the current frame, but still requests further data from the shared memory until it is filled. On the other hand, in the case of the descriptor HOLD bit set, the TF empties the TB and no further data requests from the shared memory occur until HOLD is withdrawn. Then TB is filled again and the TF is activated only after enough data have been stored in the TB to prevent a data underrun.

Reaction to Transmit HOLD for the Different Modes

Variable Size Frame Oriented Protocols (HDLC, TMB, TMR)

Reaction to a channel specification containing TH = 1

Normal operation

Note: 1. IC = $7E_H$ for HDLC and IFTF = 1
FF_H for HDLC and IFTF = 0
00_H for TMB or TMR

2. *flag = $7E_H$ for HDLC*
00_H for TMB or TMR
3. *FNUM2 is ignored. The number of interframe time-fills sent between the first frame and the second frame solely depends on the internal action request initiated by setting bit **CMD**.ARPCM = 1 and leading to the action with TH = 0.*
4. *The times Δt_1 and Δt_2 are statistical but typically only a few clock cycles.*
5. *The TH bit (as all channel commands) is **not** synchronized with TB! (as opposed to the HOLD-bit in the descriptor). TH acts on the frame currently being sent, not necessarily on the last frame currently stored in the TB. In the example, TB may or may not have stored DATA 3 before the action request with TH = 1 was issued. See **Chapter 12.6** for a further discussion of this issue.*
6. *If TH is handed over to CD outside of a frame, TH = 1 prevents the MUNICH32X from sending the next frame.*

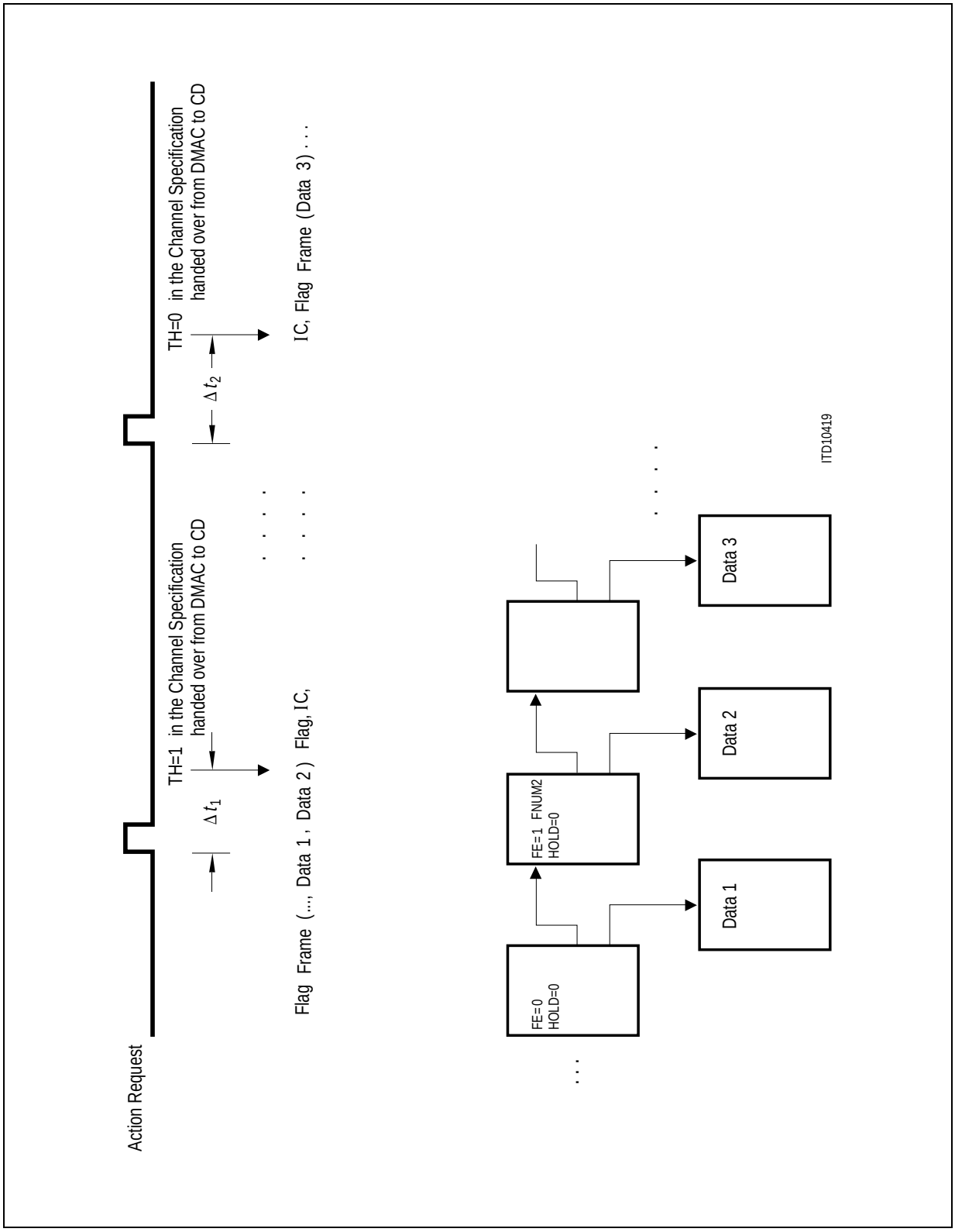


Figure 24
Handling of TH Condition (Variable Size Frame Oriented Protocols)

Fixed Size Frame Oriented Protocol (V.110/X.30)

Reaction to a channel specification containing TH = 1

Normal operation

Note: 1. The times Δt_1 and Δt_2 are statistical but typically only a few clock cycles.

2. The current processed frame is aborted, when TH = 1 is handed over to CD; only 10 – y, (y = 1, …, 10) octets of it are sent. The device then starts to send 20 + y 00_H characters regardless of how fast the TH bit is withdrawn. This ensures that the peer site is informed about the abort with a loss of synchronism.

3. The data section DATA 1 is split in the example; DATA $1^{(1)}$ is sent in the aborted frame, all bits that were read into the MUNICH32X with the same access are discarded (they would have been sent in the next frame(s) if TH = 1 was not issued) and the device starts the next frame with the bits DATA $1^{(3)}$ of the access to DATA 1 that follows the one getting the bits of DATA $1^{(1)}$.

*4. The TH (as all channel commands) is **not** synchronized with the Transmit Buffer. TH acts on the frame currently sent, not neccesarily on the last stored data.*

5. No frame will start, if TH is handed over to CD before a frame has started (after an abort or after a reset).

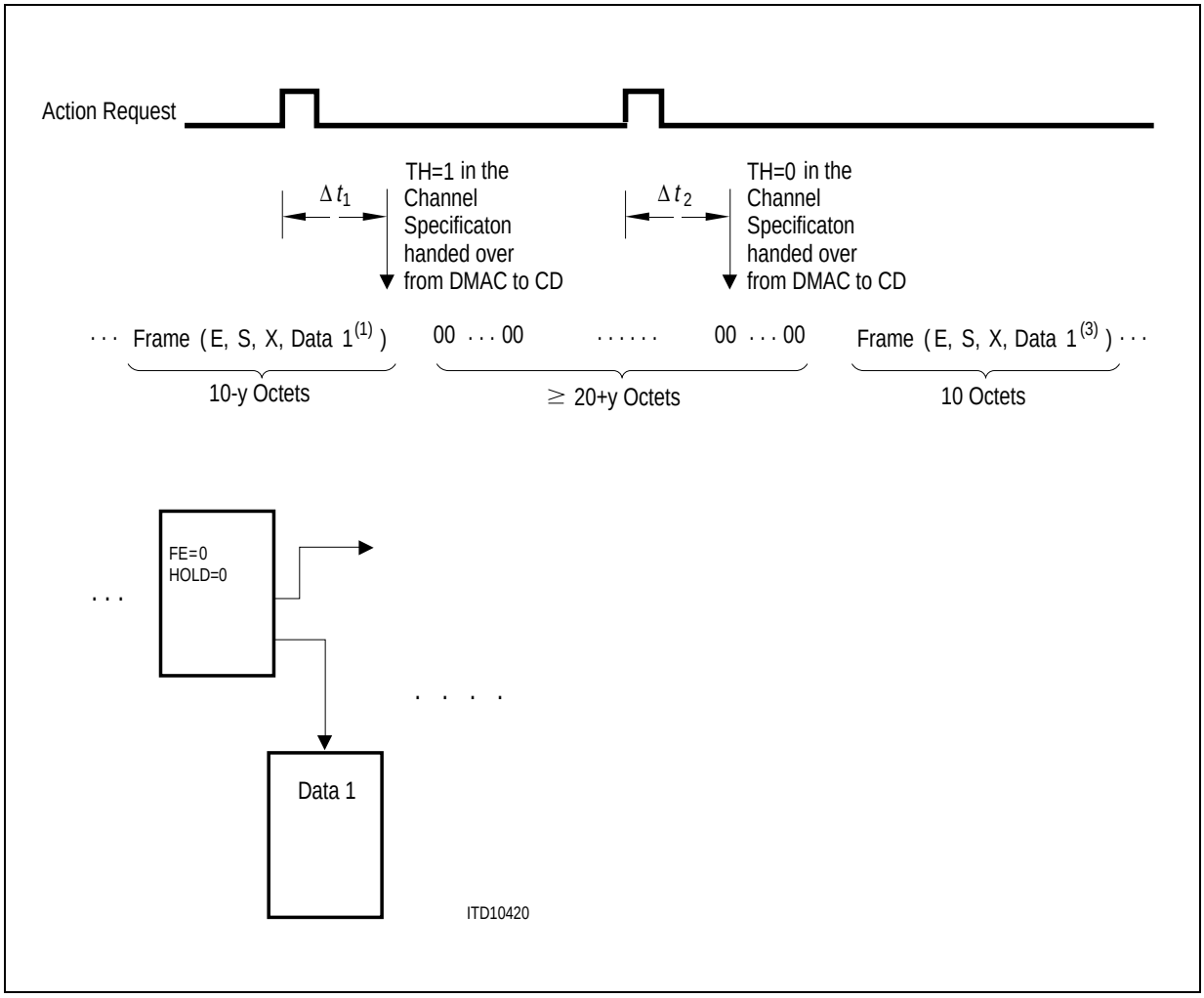


Figure 25
Handling of TH Condition (Fixed Size Frame Oriented Protocols)

Time Slot Oriented Protocol (TMA)

Reaction to a channel specification containing TH = 1

Note: 1. TC is the programmed TFLAG for FA = 1
FF_H for FA = 0

2. *The times Δt_1 and Δt_2 are statistical but typically only a few clock cycles.*
3. *The TH bit (as all channel commands) is **not** synchronized with the TB! (as opposed to the HOLD-bit in the descriptor) TH acts to the data stream currently sent.*

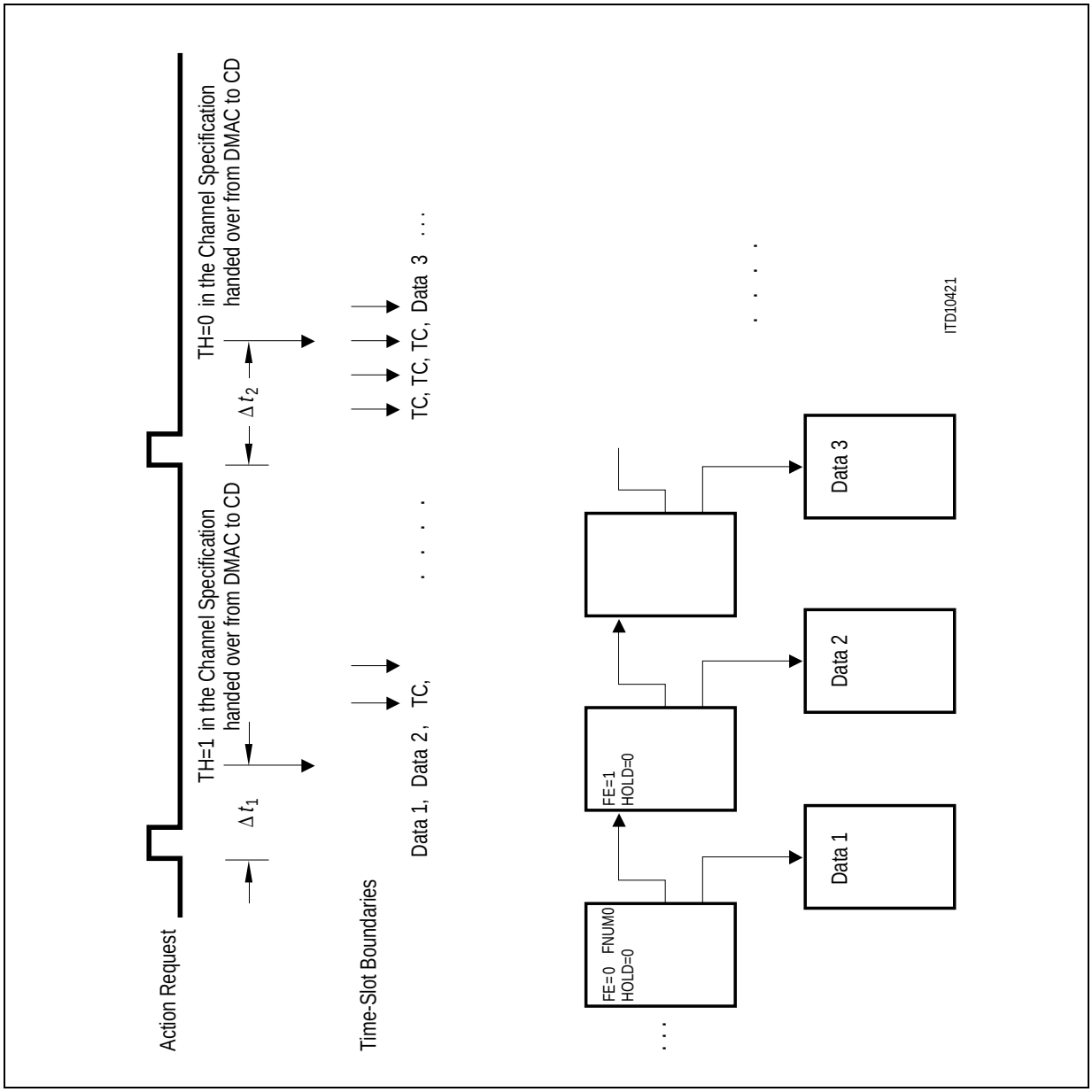


Figure 26
Handling of TH Condition (Time Slot Oriented Protocol)

Variable Size Frame Oriented Modes (HDLC, TMB, TMR)

Reaction to a channel specification containing TH = 1

Silencing of poll cycles for HOLD.

*Note: An action request initiated by setting bit **CMD**.ARPCM = 1 for an action specification leading to TH = 1 should be issued after (ITBS + 2) polls of the MUNICH32X, where ITBS is the previously programmed number of DWORDs in the TB reserved for this channel.*

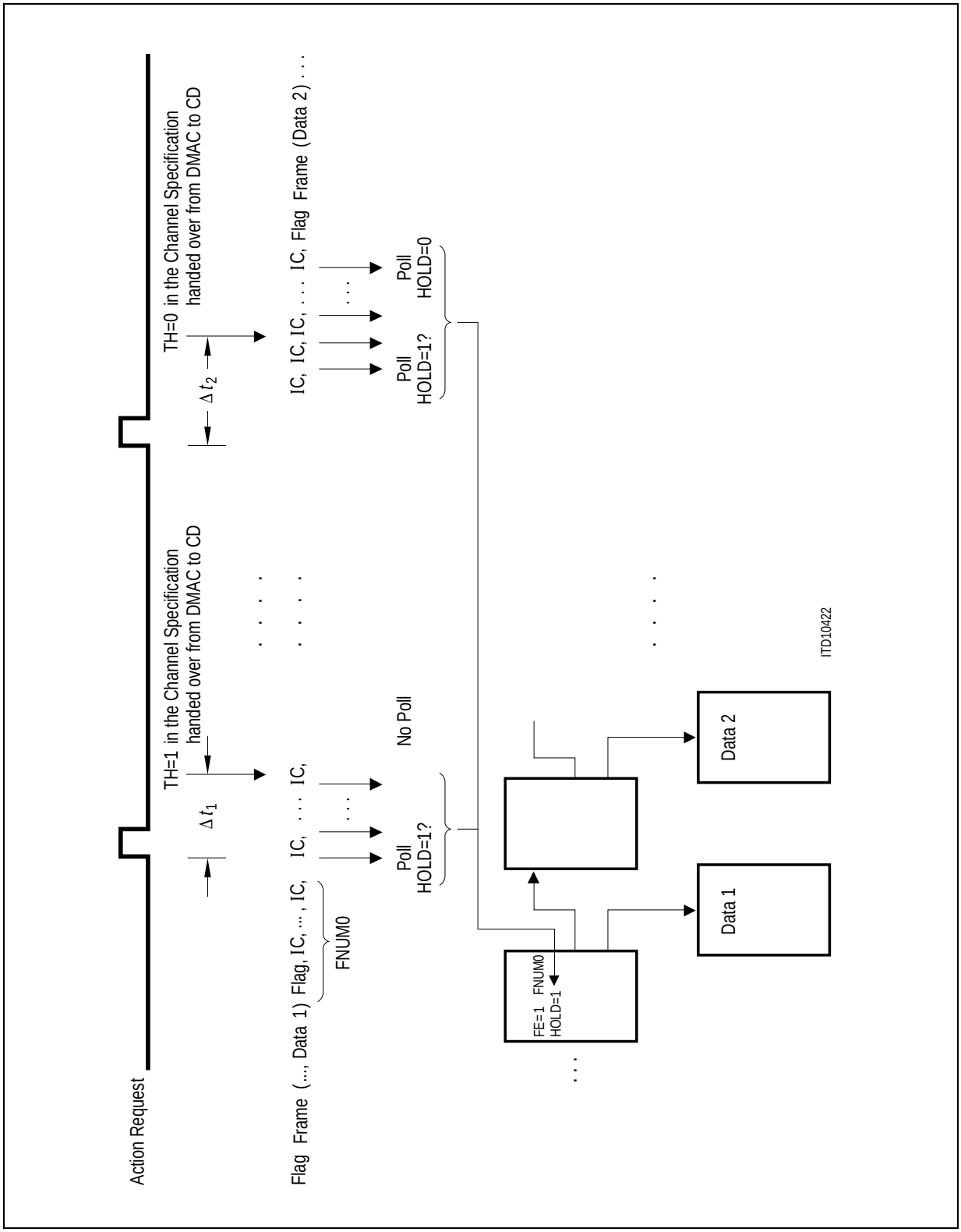


Figure 27
Handling of TH and HOLD Condition (Variable Size Frame Oriented Protocols)

Fixed Size Frame Oriented Protocol (V110/.30)

Silencing of poll cycles by TH = 1

Note: 1. The times Δt_1 and Δt_2 are statistical but typically only a few clock cycles.

*2. The TH bit (as all channel commands) is **not** synchronized with TB! (as opposed to the HOLD-bit in the descriptor) TH acts to the data stream currently sent.*

*3. In the example the proper use to silence a channel polling the HOLD bit of the transmit descriptor is illustrated. An action request initiated by setting bit **CMD**.ARPCM = 1 is issued **after** the polling has started and the HOLD-bit is not reset before polling has stopped by the TH bit.*

*4. An action request initiated by setting bit **CMD**.ARPCM = 1 for an action specification leading to TH = 1 should be issued after (ITBS + 2) polls of the MUNICH32X, where ITBS is previously programmed number of DWORDs in the TB reserved for this channel.*

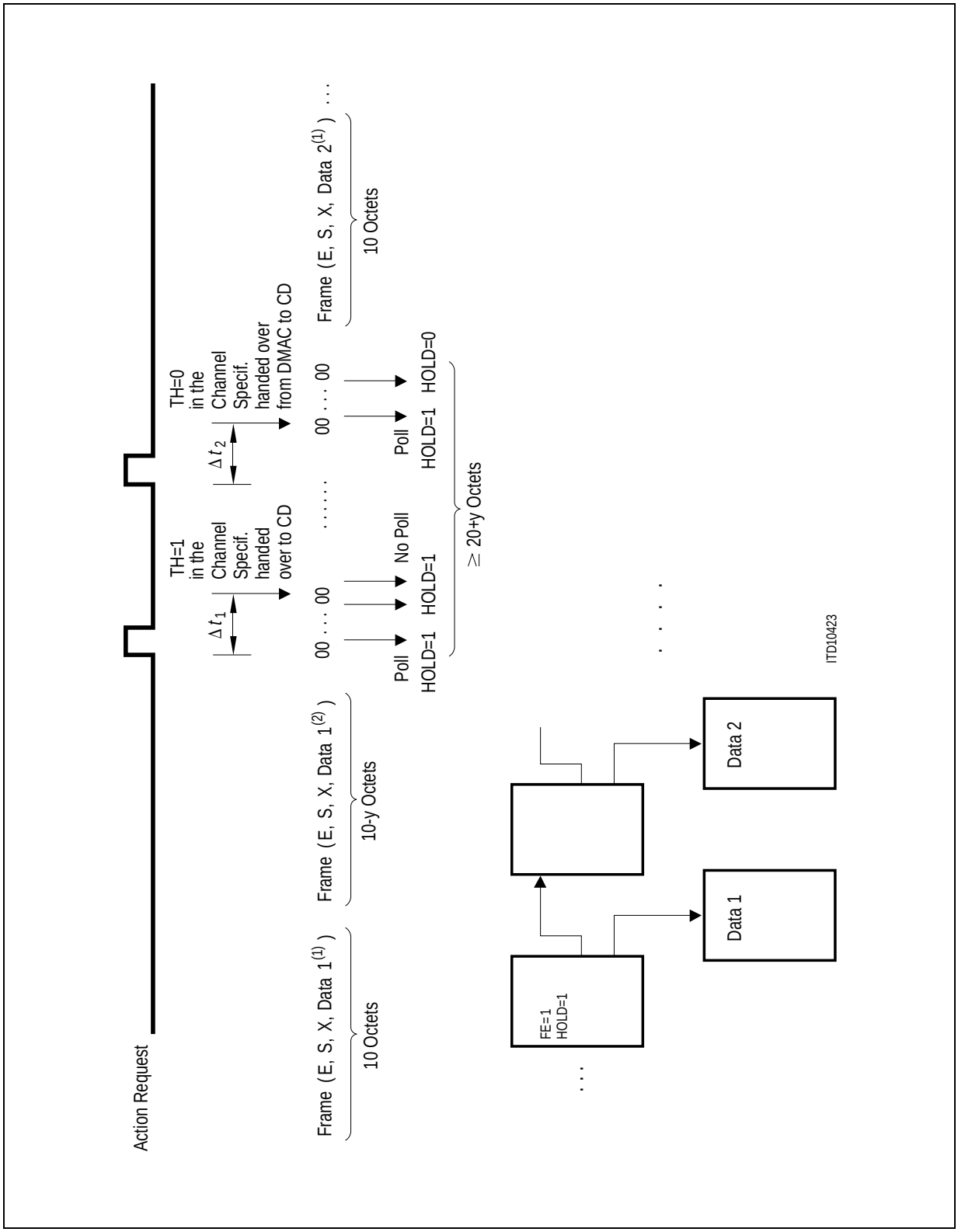


Figure 28
Handling of TH and HOLD Condition (Fixed Size Frame Oriented Protocols)

Time Slot Oriented Protocol (TMA)

Reaction to a channel specification containing TH = 1

Note: 1. TC = FF_H for TMA and FA = 0
the programmed flag for TMA and FA = 1

2. *FNUM2 is ignored. The number of interframe time-fills between the first frame and the second frame solely depends on the action request initiated by setting bit **CMD**.ARPCM = 1 leading to the action with a channel specification with TH = 0.*
3. *The times Δt_1 and Δt_2 are statistical but typically only a few clock cycles.*
4. *The TH bit (as all channel commands) is **not** synchronized with TB (as opposed to the HOLD-bit in the descriptor) TH acts on the data stream currently sent not neccessarily on the last data stored in TB. In the example TB may or may not have stored DATA 3 before action request with TH = 1 was issued.*
5. *The data stream is stopped and TC sent after the last byte of DATA 2 is sent. The stopping is triggered by the FE = 1 bit in the descriptor.*
6. *If TH is bonded over to CD during interframe time-fill (TC) it prevents the MUNICH32X from sending further data afterwards.*
7. *An action request initiated by setting bit **CMD**.ARPCM = 1 for an action specification leading to TH = 1 should be issued after (ITBS + 2) polls of the MUNICH32X, where ITBS is the previously programmed number of DWORDs in the TB reserved for this channel.*

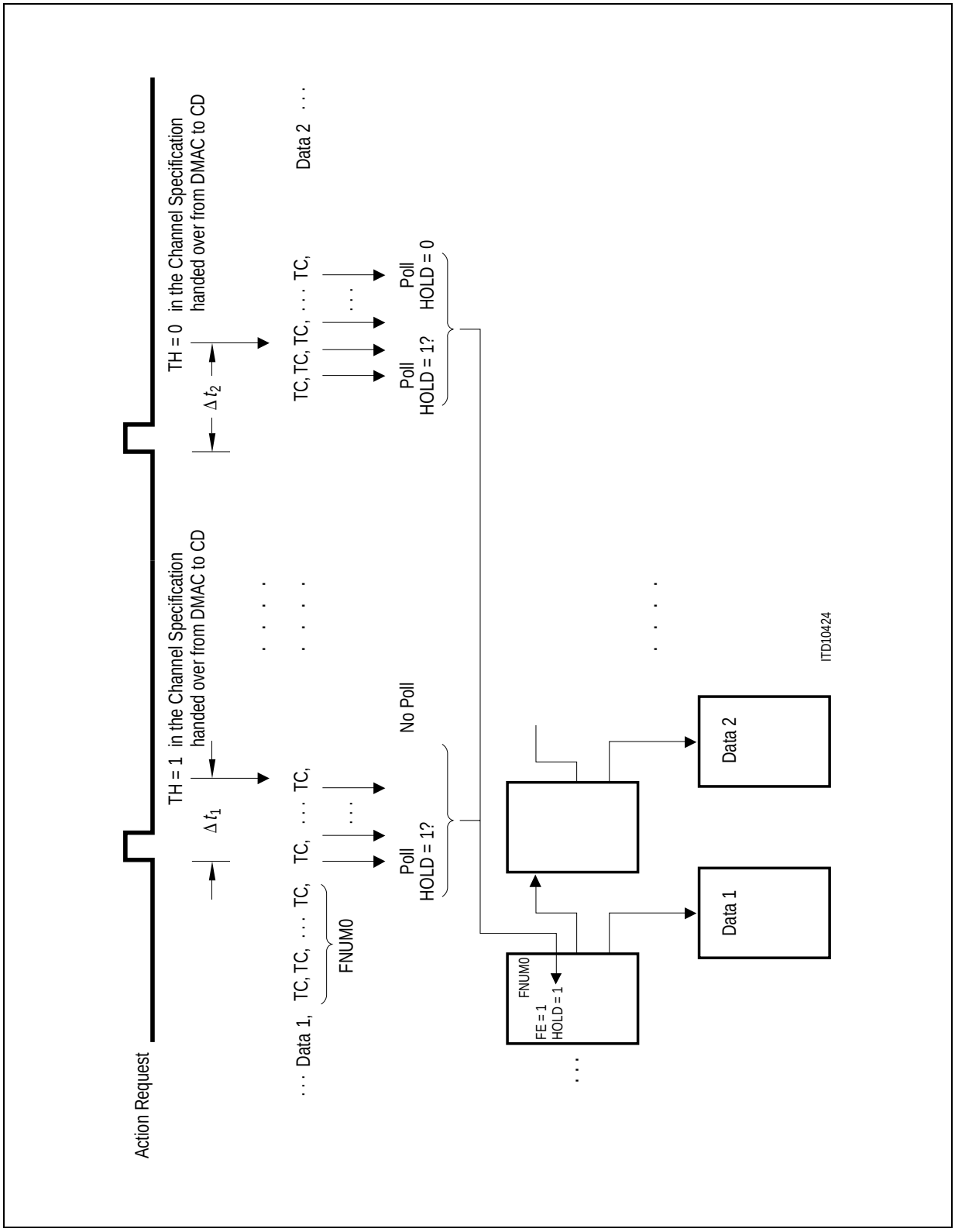


Figure 29
Handling of TH and HOLD Condition (Time Slot Oriented Protocol)

In receive direction, the MUNICH32X reads a receive descriptor, calculates the data address, writes the current receive descriptor address into the CCB, and exchanges data between the on-chip receive buffer and the external memory. After the data section has been filled, the MUNICH32X writes the number of stored bytes (BNO) into the descriptor. If a frame end has occurred, the frame status is written into the descriptor and an interrupt is generated.

The frame status includes the CRC check results and transmission error information like

– ‘non octet of bits’ (NOB),
– ‘aborted frame’ (RA),
– ‘data overflow’ (ROF),
– ‘maximum frame length exceeded’ (LFD) and
– ‘frames with less than or equal the CRC length, which equals 2 bytes for CRC16 and 4 bytes for CRC32’ (SF).

An activated reception-hold in the descriptor prevents the MUNICH32X from processing the receive data. The incoming frames are discarded until the hold is deactivated.

Because the MUNICH32X is divided into two non-synchronized parts by the on-chip buffers, two different kinds of aborting a channel transmission are implemented.

– Normal abort: This abort of a receive or transmit channel is processed in the formatters of the serial interface. The interframe time-fill code is sent after aborting the current issued frame. No accesses to the on-chip buffers are carried out, until the abort is withdrawn. The handling of the link lists and the processing of the buffers by the DMA controller are not affected by normal abort.
– Fast abort: A fast abort is performed by the DMA controller and does not disturb the transmission on the serial interface. If this abort is detected the current descriptor is suspended with an abort status immediately followed by a branching to the new descriptor defined in the channel specification of the CCB.

For initialization and control, the host sets up a **Control and Configuration Block (CCB)**, including the action specification, time slot assignment and the channel specification. The host initiates an action, e.g. reconfiguration, change of the channel mode, reset or switching of a test loop by updating the CCB and issuing an action request. This is done by writing a ‘1’ to the **CMD**.ARPCM bit field in Command register for the serial PCM core, or by writing a ‘1’ to the **CMD**.ARLBI bit field for LBI related action requests.

When the action request is detected by the MUNICH32X, it reads the control start address in **CCBA** register, then the action specification and (if necessary) additional information from the CCB. After execution, the action request is acknowledged by the **STAT**.PCMA or **STAT**.LBIA interrupt bit fields in Status register.

MUNICH32X indicates an interrupt by activating the interrupt line and storing the interrupt information (including the corresponding channel number) in the associated interrupt queue, which is indicated by a flag in Status register **STAT**. Interrupts may be masked in Interrupt Mask register **IMASK**.

The interrupt queues are implemented as circular buffers; the MUNICH32X starts to write status information into the queue and fills it successively in a circular manner. The host has to allocate sufficient buffer size and to empty the buffer fast enough in order to prevent overflow of the queue.

Monitoring functions are implemented in MUNICH32X to discover errors or condition changes, i.e.

- Receive frame end
- Receive frame abort by overflow of the receive buffer or hold condition or recognized ABORT flag
- Frame overflow, if a frame has to be discarded because of pending inaccessibility of the chip memory
- Transmit frame end
- Transmit frame abort (data underrun) by underrun of the transmit buffer or hold condition or bus cycle error
- Change of the interframe time-fill.
- Loss of synchronism or change of framing bits (V.110, X.30).
- Short frame with no data content detected.

An error or condition change is indicated by an interrupt. The host may react to the interrupt by either aborting or tristating the specific channel, or with a channel reconfiguration. To prevent underrun of the transmit buffer, sufficient buffer size has to be allocated to the channel.

A more detailed discussion of the receive procedure with examples is provided under the detailed protocol description in **Chapter 4**.

4 Detailed Protocol Description

In the following sections, the protocol support of the MUNICH32X is described in detail for transmit and receive direction.

Each section starts with a discussion of the general features, then proceeds with protocol variants and options from the channel specification, and closes with a description of interrupts and special topics.

4.1 HDLC

Transmit Direction

General Features

In transmit direction

– the starting and ending flag ($7E_H$ before and after a frame)
– the interframe time-fill between frames
– the zero insertions (a ‘0’-bit after 5 consecutive ‘1’s inserted within a frame)
– (optional) the Frame Check Sequence (FCS) at the end of a frame

is generated automatically.

Options

The different options for this mode are

– the value of the interframe time-fill character in the channel specification:
  – $7E_H$ for IFTF = 0
  – FF_H for IFTF = 1
– the number of interframe time-fill characters programmed by FNUM in the transmit descriptor. For the values FNUM = 0, 1, 2, the following sequences are used:
  – FNUM = 0: frame 1, $7E_H$, frame 2 (start flag = end flag)
  – FNUM = 1: frame 1, $7E_H$, $7E_H$, frame 2
  – FNUM = 2: frame 1, $7E_H$, IC, $7E_H$, frame 2
– the correction of the number of interframe time-fill characters by $^1/_8$ of the number of zero insertions by programming FA in the channel specification:
  – FA = 0: FNUM from the transmit descriptor is taken directly to determine the number of interframe time-fill characters as shown in **Figure 29**.
  – FA = 1: FNUM from the transmit descriptor is reduced by $^1/_8$ of the number of the zero insertions of the frame corresponding to the transmit descriptor as shown in **Figure 30**. This allows transmission of long HDLC frames for a constant bit rate

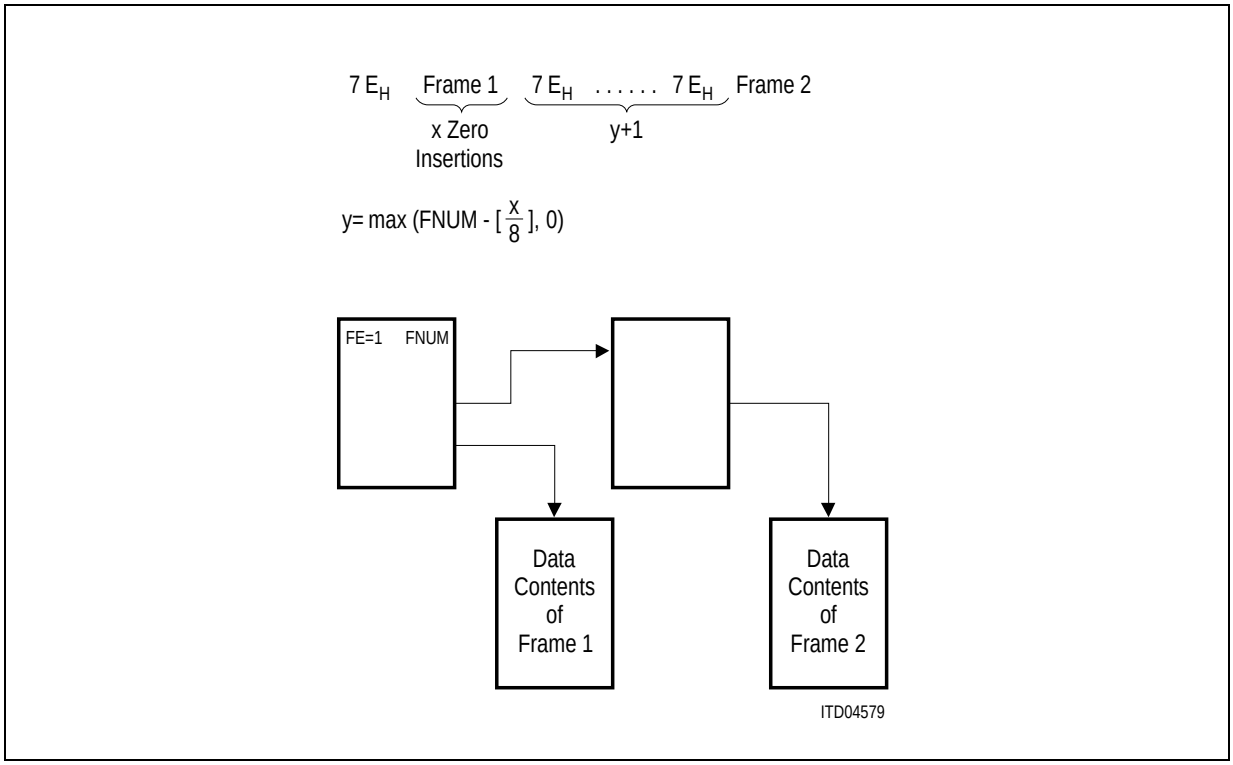


Figure 30
FNUM Reduction in HDLC Transmit Mode

Note: 1. $\left[\frac{x}{8}\right]$ ***is the biggest integer smaller than*** $\frac{x}{8}$.

2. For FNUM – $\left[\frac{x}{8}\right]$ *< 0, y = 0*

– *the type of Frame Check Sequence (FCS) is determined by the CRC bit in the channel specification.*
CRC = 0: the generator polynomial $x^{16} + {}^{12} + x^5 + 1$ *is used (2 byte FCS of CCITT Q.921)*
CRC = 1: the generator polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + \ldots$
$\ldots x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
(4 byte FCS) is used

– *the suppression of the automatic generation of the FCS is programmable in the channel specification:*

– *CS = 0: FCS generated automatically*
CS = 1: FCS generation suppressed
and in the transmit descriptor:
CSM = 0: FCS generated automatically if CS = 0 in the channel specification
CSM = 1: FCS generation suppressed

Interrupts

The possible interrupts for the mode in transmit direction are:

HI: issued if the HI bit is detected in the transmit descriptor
(not maskable)

FI: issued if the FE bit is detected in the transmit descriptor
(maskable by FIT in the channel specification)

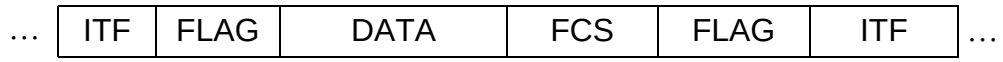
ERR: one of the following transmit errors has occurred:
– the last descriptor had H = 1 and FE = 0
– the last descriptor had NO = 0 and FE = 0
(maskable by TE in the channel specification)

FO: issued if the MUNICH32X was unable to access the shared memory in time either for new data to be sent or for a new transmit descriptor
(maskable by TE in the channel specification)

FE2: – data has been sent on the TXD line.

(maskable by FE2 in the channel specification)

A typical data stream has the form:



Example:

HDLC channel with

CS =0	(FCS generated automatically)
INV = 0	(no inversion)
CRC = 0	(CRC16)
TRV = 00	(required as unused in HDLC mode)
FA = 1	(flag adjustment)
MODE = 11	(HDLC)
IFTF = 1	(interframe time-fill ‘1’s)

Little Endian Data Format
Channel number 1A

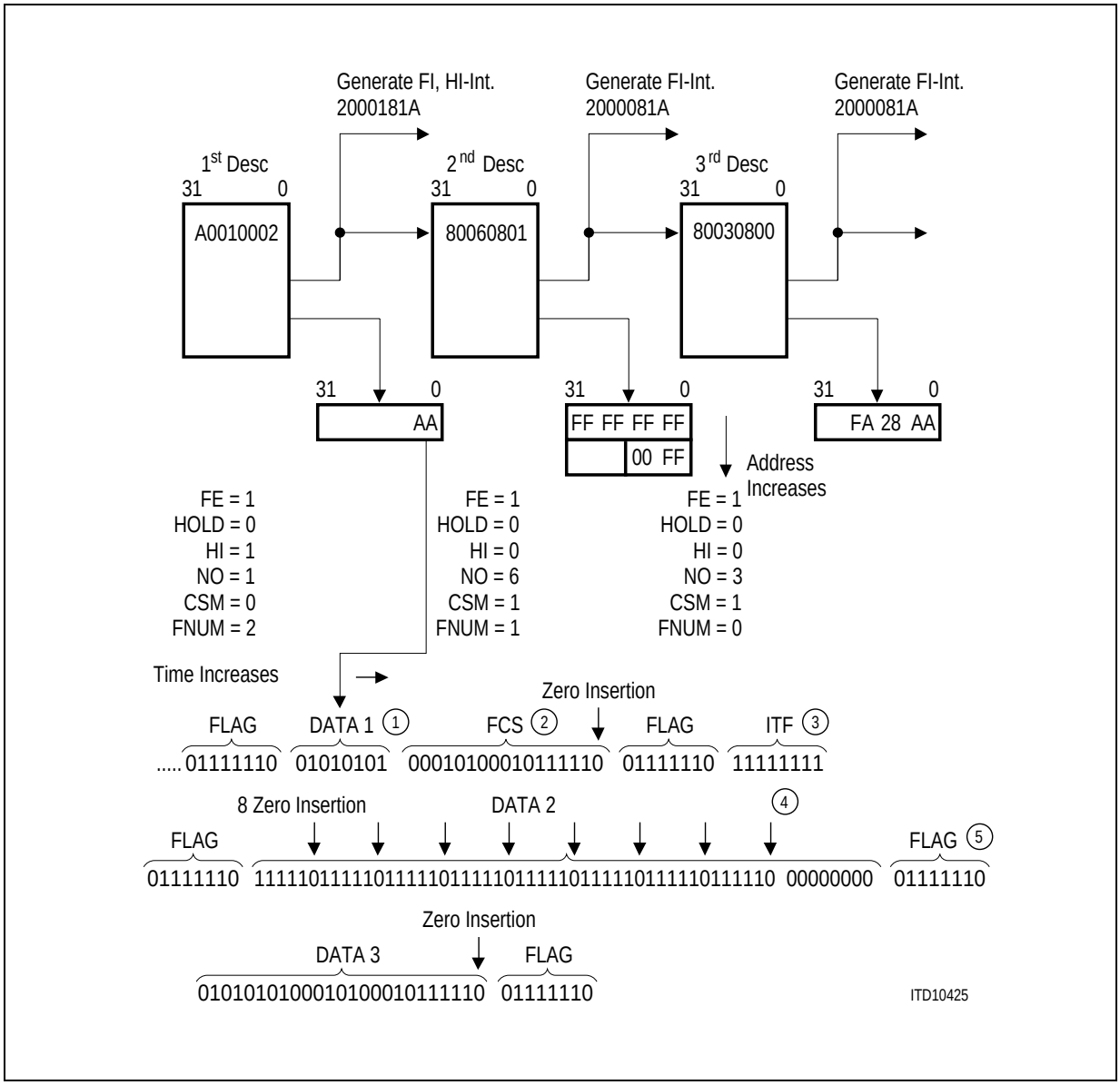


Figure 31
Example of HDLC Transmit Mode

Note: 1. Data is transmitted according to §2.8 of CCITT recommendation Q.921

2. Note: FCS in the data section is formatted as ordinary data!!! FCS is generated here automatically as CS = 0 and CSM = 0 for the 1st descriptor.

3. There was 1 zero insertion in the 1st frame, so FNUM – $\left[\frac{1}{8}\right]$ = FNUM = 2. Therefore between the first and the second frame we have the sequence 'FLAG ITF FLAG' (ITF = FF_H because IFTF = 1).

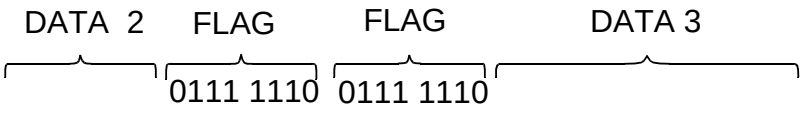
4. *No FCS is generated here as CSM is '1' for the second and third transmit descriptor. The FCS is supposed to be the last 2 bytes to be transmitted in this case, their validity is not checked internally.*
5. *There were 8 zero insertions in the 2nd frame, so FNUM – $\left[\frac{8}{8}\right]$ = FNUM–1 = 0. Therefore between the second and the third frame we have a shared FLAG.*

For CS = 1 (CRC select) the transmitted data stream would differ at FCS, FCS would just be omitted.

For INV = 1 (channel inversion) all bits of the data stream (including FLAG, DATA, FCS, ITF) would be inverted.

For CRC = 1 (CRC 32) the transmitted data stream would only differ in the FCS, the FCS would be 1101 0111 1010 0101 1000 0000 0010 0111.

For FA = 0 (no flag adjustment) the transmitted data stream would change only after DATA 2. The value FNUM = 1 in the second descriptor would alone determine the number of interframe time-fill characters, the scenario would look like



For IFTF = 0 (ITF flags) the transmitted data stream would only differ at ITF, the 8 ones would be replaced by 0111 1110.

In big endian mode the only difference is in the data section

For the first descriptor it ought to be

31	0
AA	

and for the second

31	0
FF FF FF FF	
FF 00	

and for the third

31	0
AA 28 FA	

Receive Direction

General Features

In receive direction:

1. The starting and ending flag ($7E_H$ before and after a frame) is recognized and extracted.
2. A change of the interframe time-fill is recognized and reported by an interrupt.
3. The zero insertions (a ‘0’-bit after five ‘1’s within a frame) are extracted.
4. The FCS at the end of a frame is checked, it is (optionally) transferred to the shared memory together with the data.
5. The number of the bits within a frame (without zero insertions) is checked to be divisible by 8.
6. The number of bytes within a frame is checked to be smaller than MFL + 1 (after extraction of ‘0’ insertions). The check is maskable by setting the bit field MFLD in **MODE1** register.
7. The number of bits within a frame after extraction of ‘0’ insertions is checked to be greater than check a) 16 for CRC = 0
   32 for CRC = 1
   (only for CS = 0) check b) 32 for CRC = 0
   48 for CRC = 1.
8. The occurrence of an abort flag ($7F_H$) ending a frame is checked.

More detailed description of the individual features:

1. a. A frame is supposed to have started if after a sequence of 0111 1110 in the receive data stream neither FC_H nor FD_H nor $7E_H$ has occurred. The frame is supposed to have started with the first bit after the closing ‘0’ of the sequence.
   b. A frame is supposed to have stopped if a sequence of 0111 1110 or 0111 1111 is found in the data stream after the frame has started. The last bit of the frame is supposed to be the bit preceding the ‘0’ in the above sequences. The cases of sequences 0111 1110 1111 111 and 0111 1110 0111 1111 are also supposed to be frames of bit length – 1 and 0 respectively.
   A frame is also supposed to have stopped if more than MFL bytes were received since the start of the frame.
   c. The ending flag of a frame may be the starting flag of the next frame (shared flags supported).
2. The receiver always remains in one of two possible interframe time-fill states: ‘F’ and ‘O’. **Figure 32** illustrates them.
   Note that a change from ‘F’ to ‘O’ and vice versa is reported by an IFC interrupt.

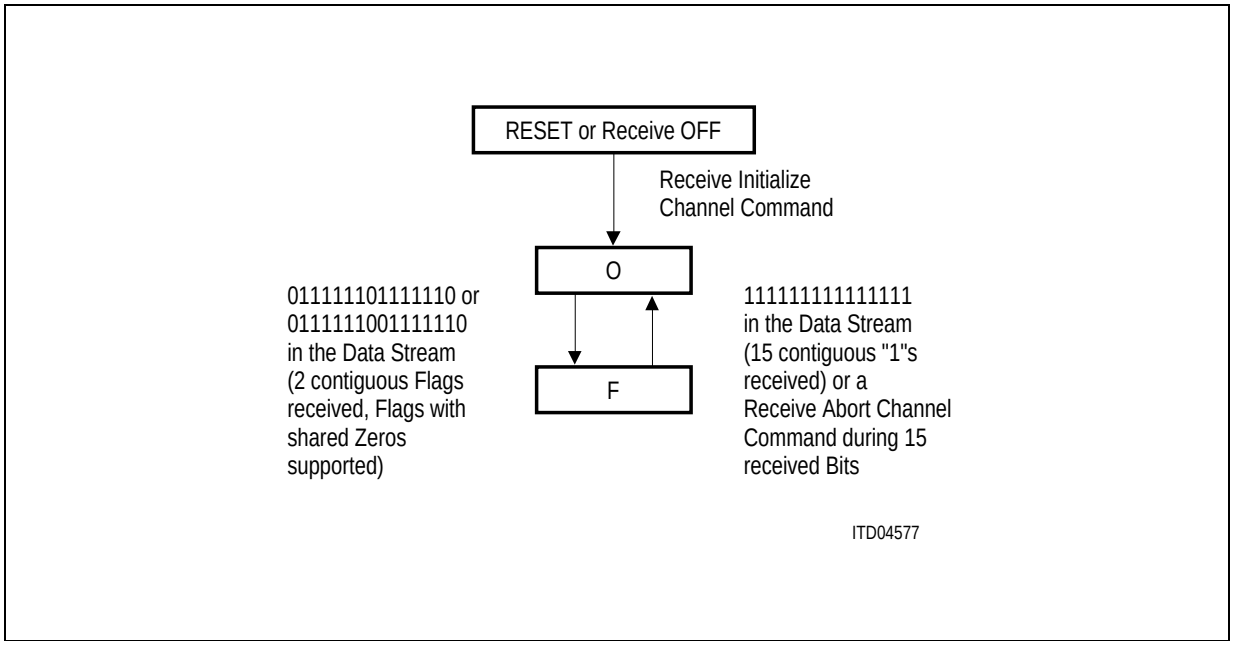


Figure 32
Receiver Interframe Time-Fill States in HDLC

3. The '0' extraction is also carried out for the last 6 bits before the stopping sequence.
4. The last 16 (CRC = 0) or 32 (CRC = 1) bits of a frame (after extraction of the zero insertions are supposed to be the FCS of the remaining bits of the frame. (For the case of a frame with less than or equal to 16 or 32 bits, respectively, see point 7). The FCS is always checked, the check is reported in the CRCO bit of the last receive descriptor of the frame.
   CRCO = 1: FCS was incorrect
   CRCO = 0: FCS was correct
5. The check is reported in the NOB bit in the last receive descriptor of the frame
   NOB = 1: The bit length of the frame was not divisible by 8.
   NOB = 0: The bit length of the frame was divisible by 8.
   If NOB = 1: The last access to a receive data section of the frame may contain erroneous bits and should not be evaluated.
6. The check is reported in the LFD bit in the last receive descriptor of the frame (if MFLOFF = 0).
   LFD = 1: The number of bytes was greater than MFL.
   LFD = 0: The number of bytes was smaller or equal to MFL.
   Only the bytes up to the
   MFL + 1st one for CS =1
   MFL – 1st one for CS = 0, CRC = 0
   MFL – 3rd one for CS = 0, CRC = 1
   are transferred to be stored memory. The bytes of the last access may be erroneous and should not be evaluated.

7. For frames not fulfilling check a) no data are transferred to the shared memory irrespective of CS.
   Only an interrupt with the bit FI, SF and (possibly) ERR is generated.
   For frames fulfilling check a) but not check b) data is transferred to the shared memory but the SF bit in the last receive descriptor is set.
8. The check is reported in the RA bit in the last receive descriptor of the frame
   RA = 1: The frame was stopped by the sequence $7F_H$
   RA = 0: The frame was not stopped by the sequence $7F_H$.
   Note: A receive descriptor with RA = 1 may also result from a fast receive abort or a receive abort channel command or from a receive descriptor with set HOLD bit.

Options

The different options for this mode are:

– The kind of Frame Check Sequence (FCS)
  Two kinds of FCS are implemented and can be chosen by CRC bit.
  CRC = 0: the generator polynomial $x^{16} + x^{12} + x^5 + 1$ is used (2 byte FCS of CCITT Q.921)
  CRC = 1: the generator polynomial
  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (4 byte FCS) is used.
– the transfer of the FCS together with the received data is programmable by the CS bit.
  CS = 0: FCS is not transferred to the data section
  CS = 1: FCS is transferred to the data section.
  *Note: FCS is **always** checked irrespective of the CS bit.*

Interrupts

The possible interrupts for the mode in receive direction are:

HI: issued if the HI bit is detected in the receive descriptor (not maskable)

FI: issued if a received frame has been finished as discussed in 1.b of the protocol features (also for frames which do not lead to data transfer as discussed in 7. of the protocol features)
(maskable by FIR in the channel spec.)

IFC: issued if a change of the interframe time-fill state as discussed in 2. has occurred. (maskable by IFC in the channel spec.)

SF: a frame not fulfilling check a) has been detected (maskable by SFE in the channel spec.)

ERR: issued if one of the following error conditions has occurred:
- FCS was incorrect
- the bit length was greater than MFL
- the frame was stopped by $7F_H$
- the frame could only be partly stored because of internal buffer overflow of RB
- a fast receive abort channel command was issued
- a receive abort channel command was detected during reception of a frame
- a frame could only be partly transferred to the shared memory because of a receive descriptor with HOLD bit set
  (maskable by RE in the channel spec.)

FO: issued if due to inaccessibility of internal buffer RB
- one ore more complete frames have been lost
- one ore more changes of interframe time-fill state were lost
  (maskable by RE in the channel spec.)

Note that all receive interrupts are maskable by setting the bit field RID in **MODE1** register.

Example:

HDLC channel with

CS = 1	(FCS transferred to shared memory)
INV = 0	(no inversion)
CRC = 1	(CRC 32)
TRV = 00	(required as unused in HDLC mode)
FA = x	(irrelevant in Rx direction)
MODE = 11	(HDLC)
IFTF = x	(irrelevant in Rx direction)

Big Endian Data Format
Channel No. 1D
MFL = 10

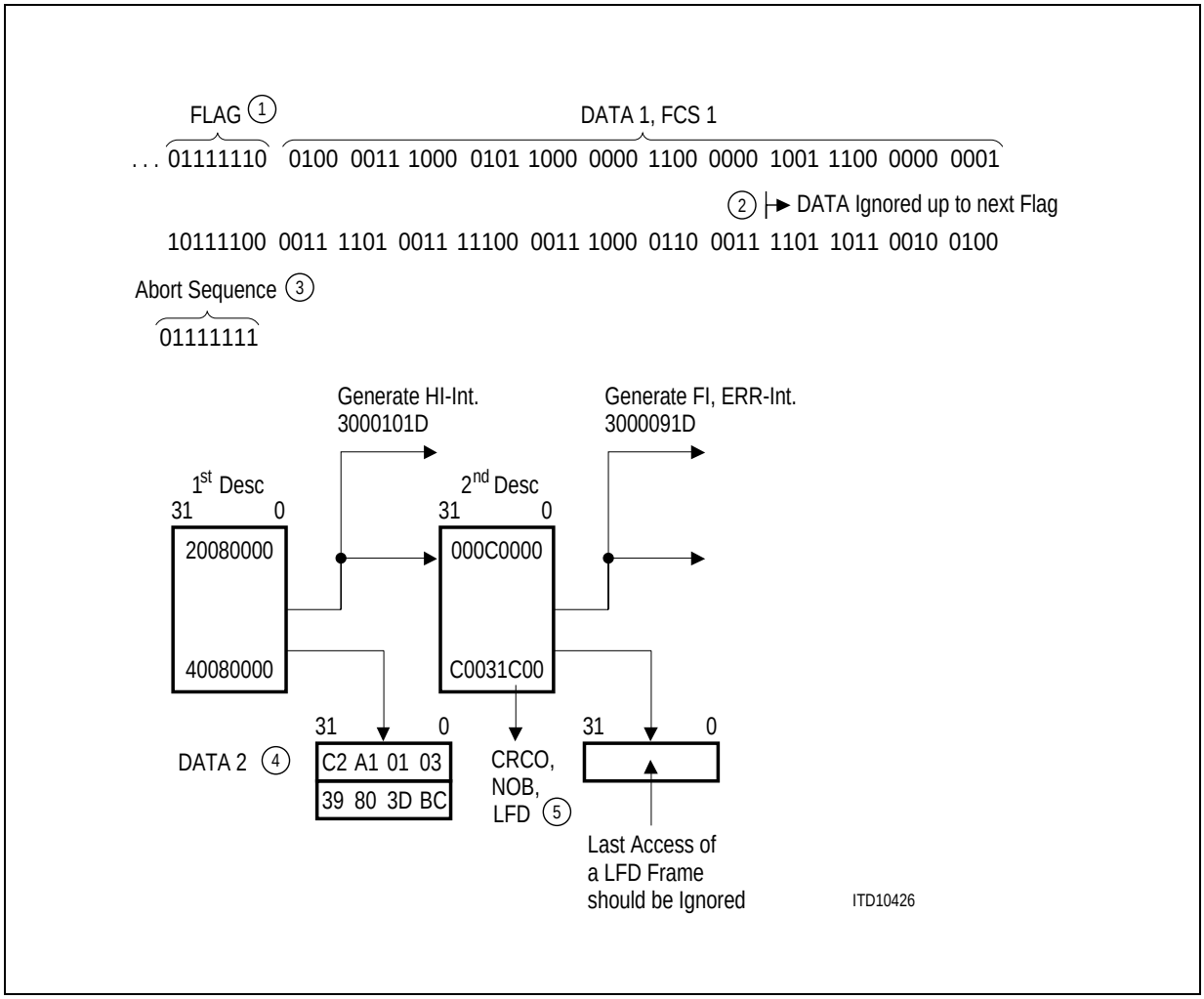


Figure 33a
Example of HDLC Receive Mode

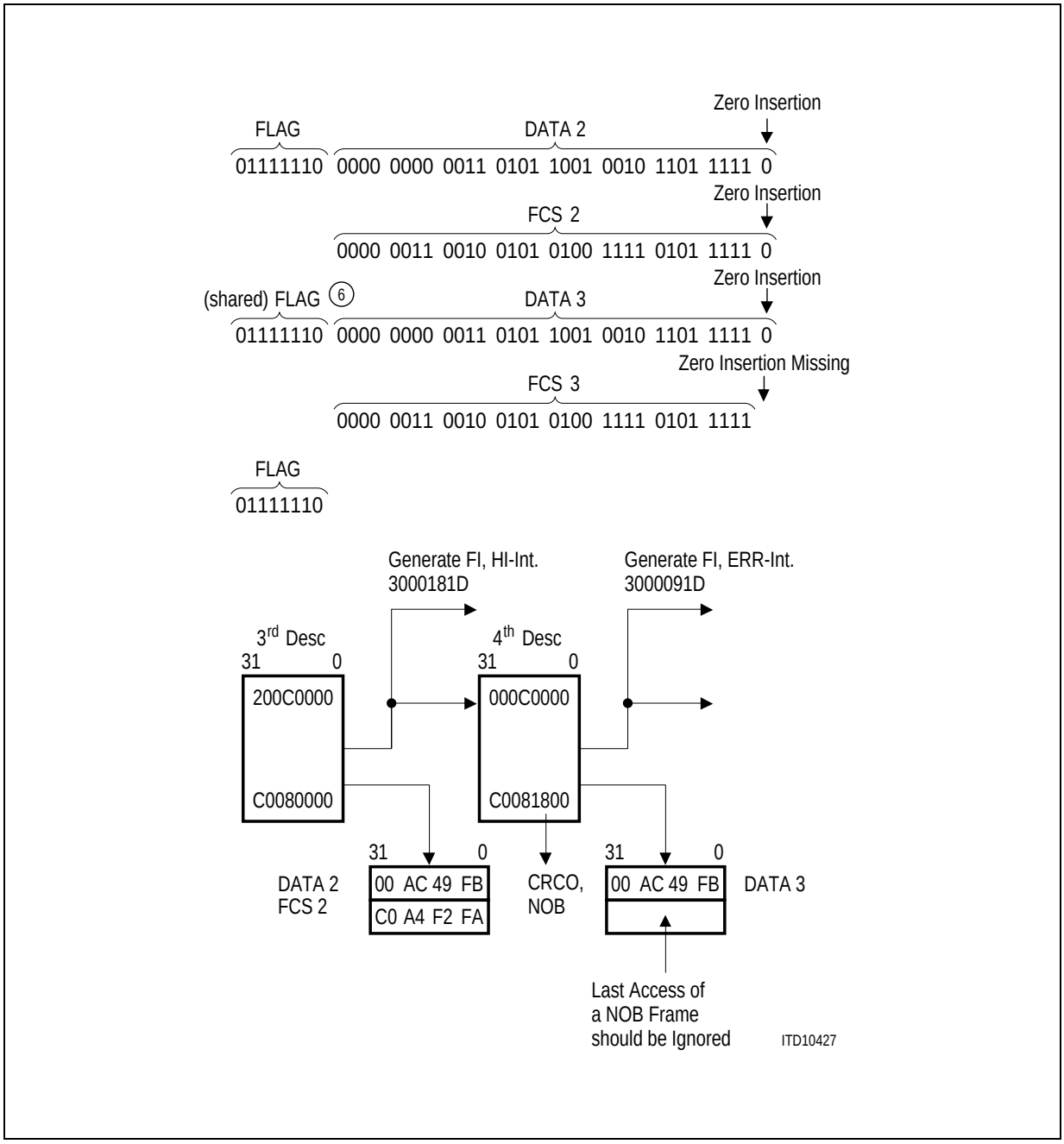


Figure 33b
Example of HDLC Receive Mode

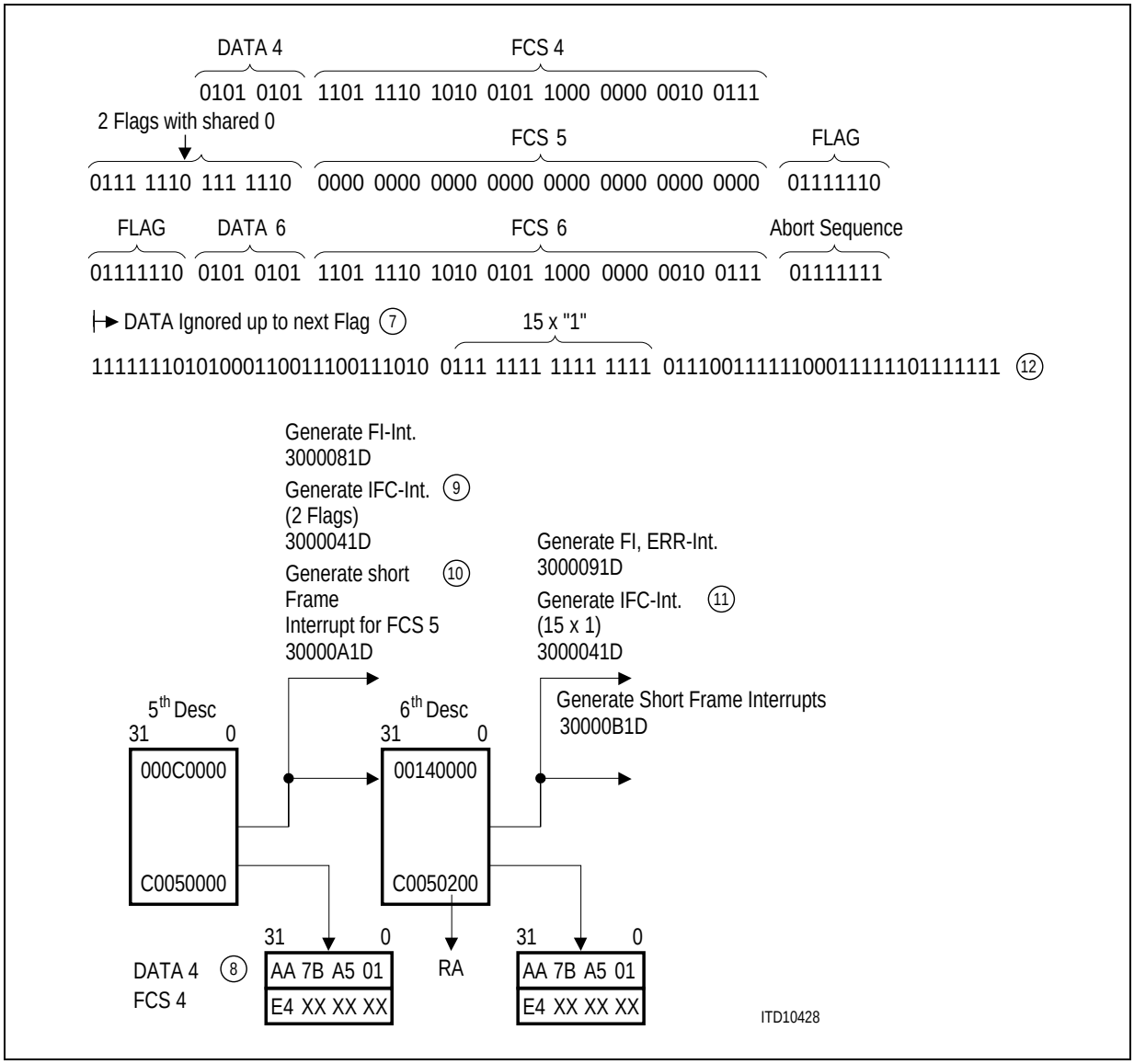


Figure 33c
Example of HDLC Receive Mode

Note: 1. After Receive Initialization is detected all data are ignored until a flag is received. The receiver is in the interframe time-fill state '0'.

2. After MFL + 1 data bytes are received the further data are ignored (except for a change of the interframe time-fill state) and are neither stored in the RB nor reported to the shared memory. The receiver waits for the next flag.

3. Even the abort sequence at the end of the frame will not lead to the RA bit in the descriptor to be set.

4. Data are formatted according to §2.8 of CCITT Q.921.

5. The FCS is formatted as ordinary data!

6. *LFD is issued and always accompanied by NOB. CRCO should not be interpreted for a LFD frame.*
7. *Here the ending flag of the second frame is the starting flag of the third frame.*
8. *After an abort sequence data is ignored until a flag is found (except for a change of the interframe time-fill state). They are neither stored in the RB nor reported to the shared memory.*
9. *The last 3 bytes in the last write access to the receive data section of the 5th descriptor have to be ignored.*
10. *The 2 flags with a shared 0 in the middle change the original interframe time-fill state '0' of the receiver to 'F'. The 2 flags following FCS 5 on the other hand do not change the interframe time-fill state, as it already was 'F'.*
11. *The frame consisting only of 32 times 0 between 2 flags does not pass check a). It only leads to an interrupt.*
12. *The 15 × '1' leads to a change of the interframe time-fill state from 'F' to '0' even through it is in a data ignored zone.*
13. *This frame of length – 1 leads to an interrupt.*

For CS = 0 (CRC not selected) the descriptors are shown in **Figure 34**.

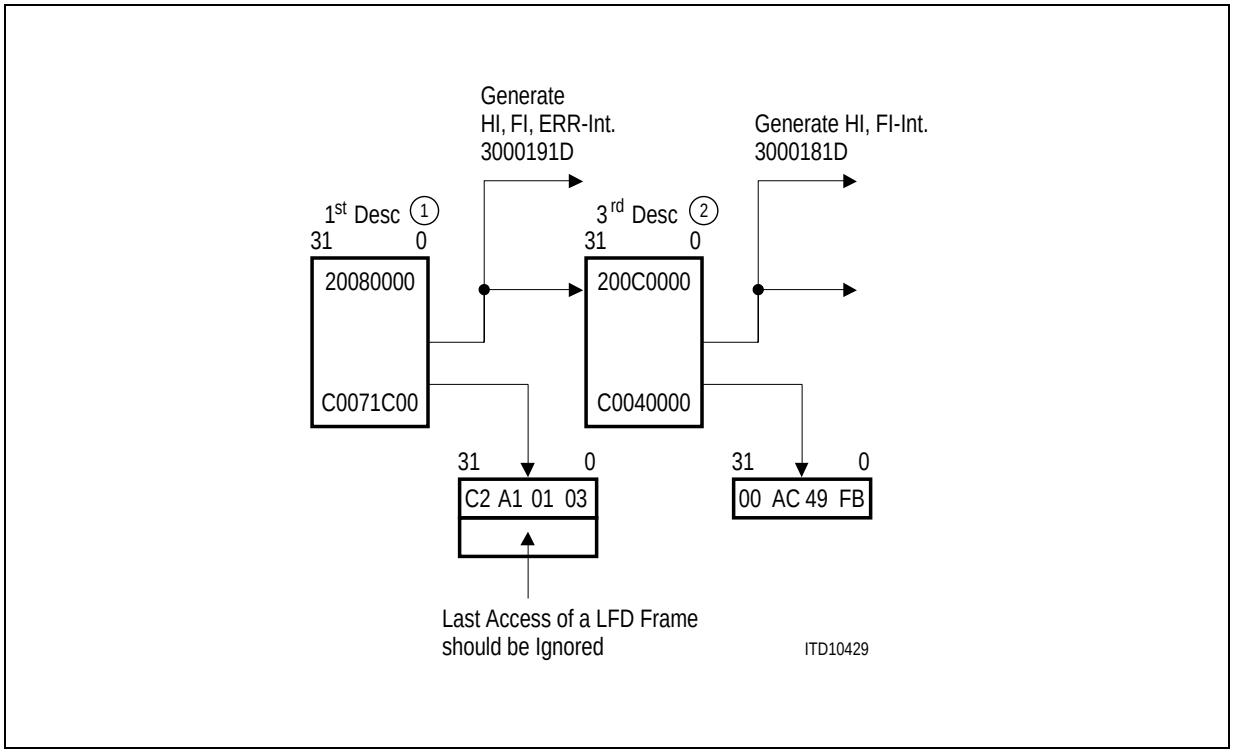


Figure 34a
Example of HDLC Reception with CS = 0

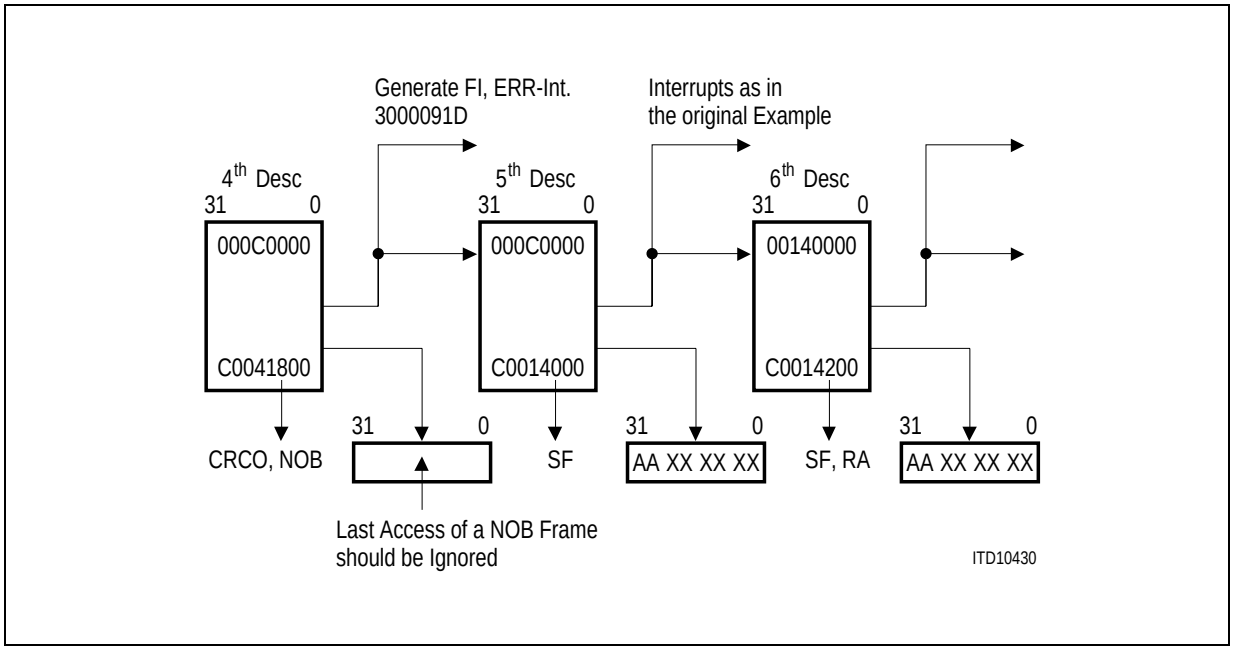


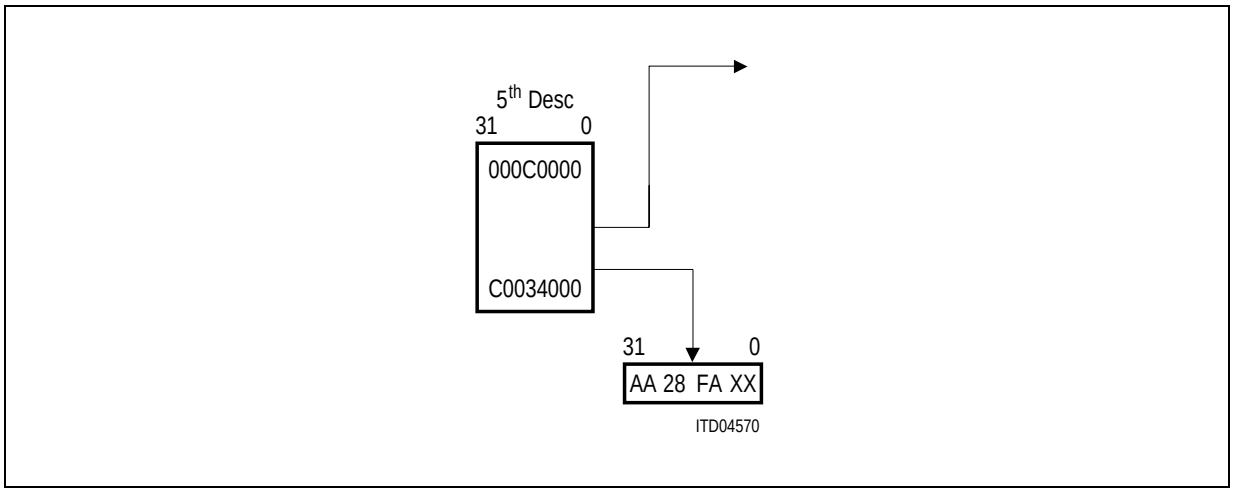
Figure 34b
Example of HDLC Reception with CS = 0

Note: 1. Only the 7 leading bytes are reported (the last 4 are supposed to be the FCS even in this case).

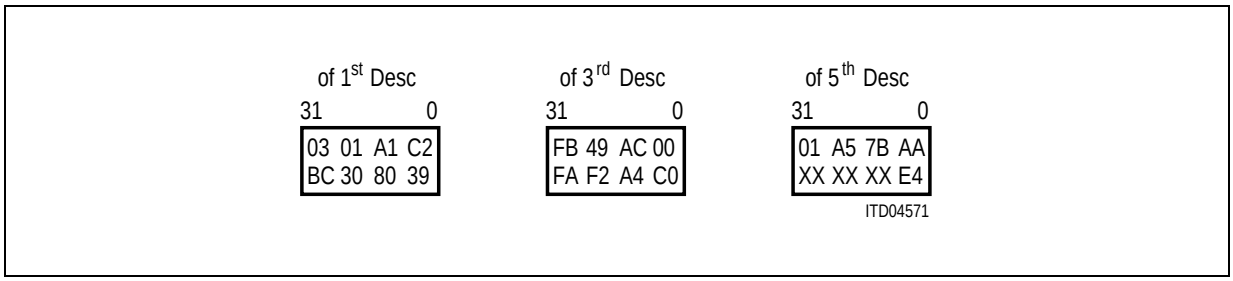
2. It is assumed here for convenience that the first descriptor points to the third and not to the second descriptor as in the original example.

For INV = 1 (channel inversion), all bits of the data stream (including DATA, FCS, flag, abort sequence 15 × '1') are interpreted inversely, e.g. '1000 0001' would be interpreted as a flag, and 15 × '0' would lead to a change from interframe time-fill state 'F' to '0' etc.

For CRC = 0 (CRC 16), the correct FCS (e.g., zeros for DATA 4) would be 00001 0100 0101 1110. The 5th descriptor would then appear as shown below.



In little endian mode, the only difference is in the receive data sections. They would be



4.2 TMB

Transmit Direction

General Features

In transmit direction:

– The starting and ending flag (00_H before and after a frame)
– The interframe time-fill between frames

is generated automatically.

Options

The different options for this mode are:

– The number of interframe time-fill characters (as shown in **Figure 27**) by choosing FNUM in the transmit descriptor.
  For the values FNUM = 0, 1, 2, the following sequences are used:

  FNUM = 0: … frame 1, 00_H, frame 2 … (start = end flag)
  FNUM = 1: … frame 1, 00_H, 00_H, frame 2 …
  FNUM = 2: … frame 1, 00_H, 00_H, 00_H, frame 2 …

Interrupts

The possible interrupts for the mode in transmit direction are identical to those of HDLC.

A typical data stream has the form: (ITF DATA ITF DATA).

Example

TMB channel with

INV = 0	(no inversion)
CRC = 0	(required)
TRV = 00	(required)
FA = 0	(required)
MODE = 01	(TMB)
IFTF = 0	(required)

Little Endian Data Format
Channel number 5

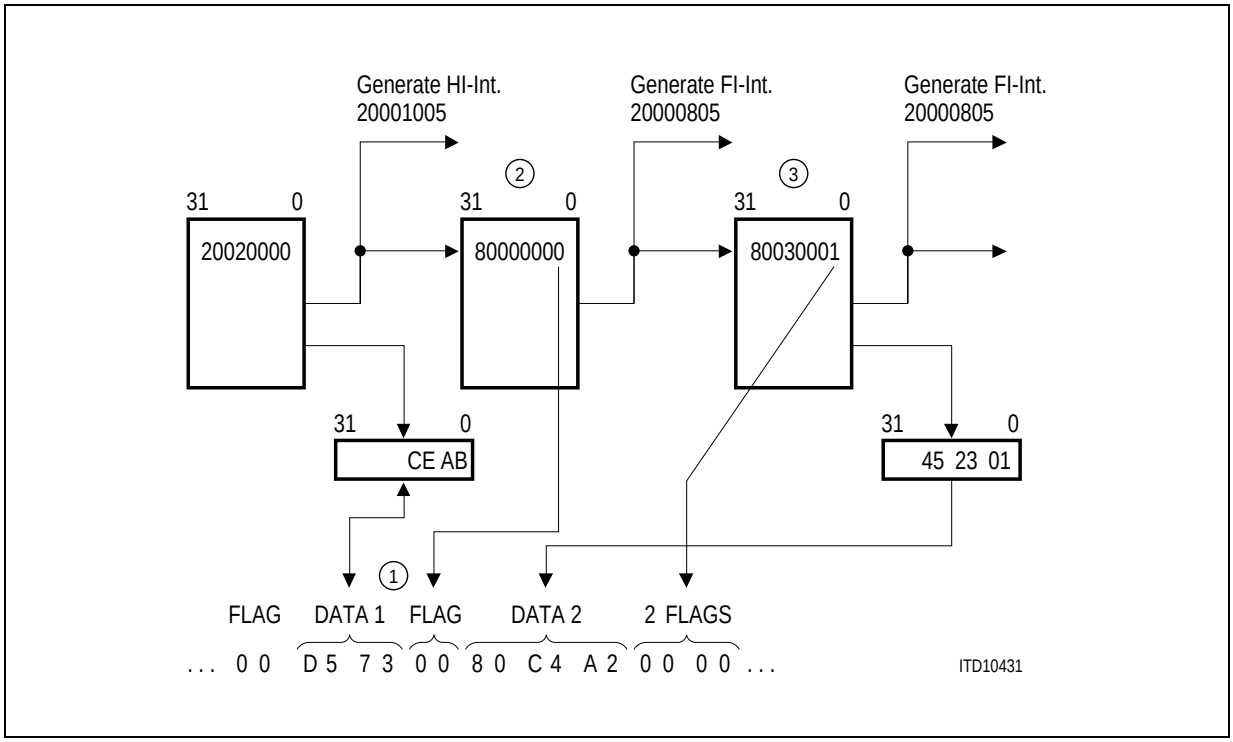


Figure 35
Example of TMB Transmission

Note: 1. Data is transmitted according to Q.921 §2.8 and fully transparent.

2. A transmit descriptor with NO = 0 and FE = 1 is allowed, one with NO = 0 and FE = 0 is not allowed.

3. FNUM = 1 leads to 2 FLAGS after DATA 2.

Receive Direction

General Features

1. The starting and ending flag (00_H before and after a frame) as well as interframe time-fill is recognized and extracted.
2. The number of bits within a frame is checked to be divisible by 8.
3. The number of bytes within a frame is checked to be smaller than MFL + 1.
4. A frame containing less than 8 bits may be ignored completely by the receiver.

More detailed description of the individual features:

1. a. A frame is supposed to have started if after a sequence ‘0000 0000’ a ‘1’-bit is recognized. The frame is supposed to have this ‘1’-bit as first bit.

   b. A frame is supposed to have stopped if
   - either a sequence 0000 0000 1 is found in the data stream after the frame has started
   - or a sequence 0000 0000 is found octet synchronous (i.e. the first bit of the sequence 00_H is the 8 m + 1^{st} bit since the starting ‘1’-bit of 1.a. for an integer m).

In both cases the last bit before the sequence 00_H is supposed to be the last bit of the frame.

2. The check is reported in the NOB bit in the last receive descriptor of the frame.
   NOB = 1: The bit length of the frame was not divisible by 8.
   NOB = 0: The bit length of the frame was divisible by 8.
3. The check is reported in the LFD bit in the last receive descriptor of the frame.
   LFD = 1: The number of bytes was greater than MFL.
   LFD = 0: The number of bytes was smaller or equal to MFL.
   Only the bytes up to the MFl + 1^{st} one are transferred to the shared memory. The bytes of the last access to the receive data section of the frame may contain erroneous bits and shouldn’t be evaluated. LFD is always accompanied by NOB.

Options

There are no options in receive direction for this mode.

Interrupts

The possible interrupts for the mode in receive direction are:

HI: issued if HI bit is detected in the receive descriptor (not maskable).

FI: issued if a received frame has been finished as discussed in 1b) of the protocol features or a receive abort channel command was detected during reception of a frame.
(maskable by FIR in the channel spec.)

ERR: issued if one of the following error conditions has occurred
- the bit length of the frame was not divisible by 8
- the byte length was greater than MFL
- the frame could only be partly stored because of internal buffer overflow of RB
- a fast receive abort channel command was issued
- the frame could only be partly transferred due to a receive descriptor with set HOLD bit.
  (maskable by RE in the channel specification)

FO: issued if due to inaccessibility of the internal buffer RB one or more complete frames have been lost. (maskable by RE in the channel spec.)

Example:

TMB channel with

INV = 0	(no inversion)
CRC = 0	(required)
TRV = 00	(required)
FA = 0	(required)
MODE = 01	(TMB)
IFTF = 0	(required)
MFL = 7	
Big Endian Data Format	
Channel No. A	

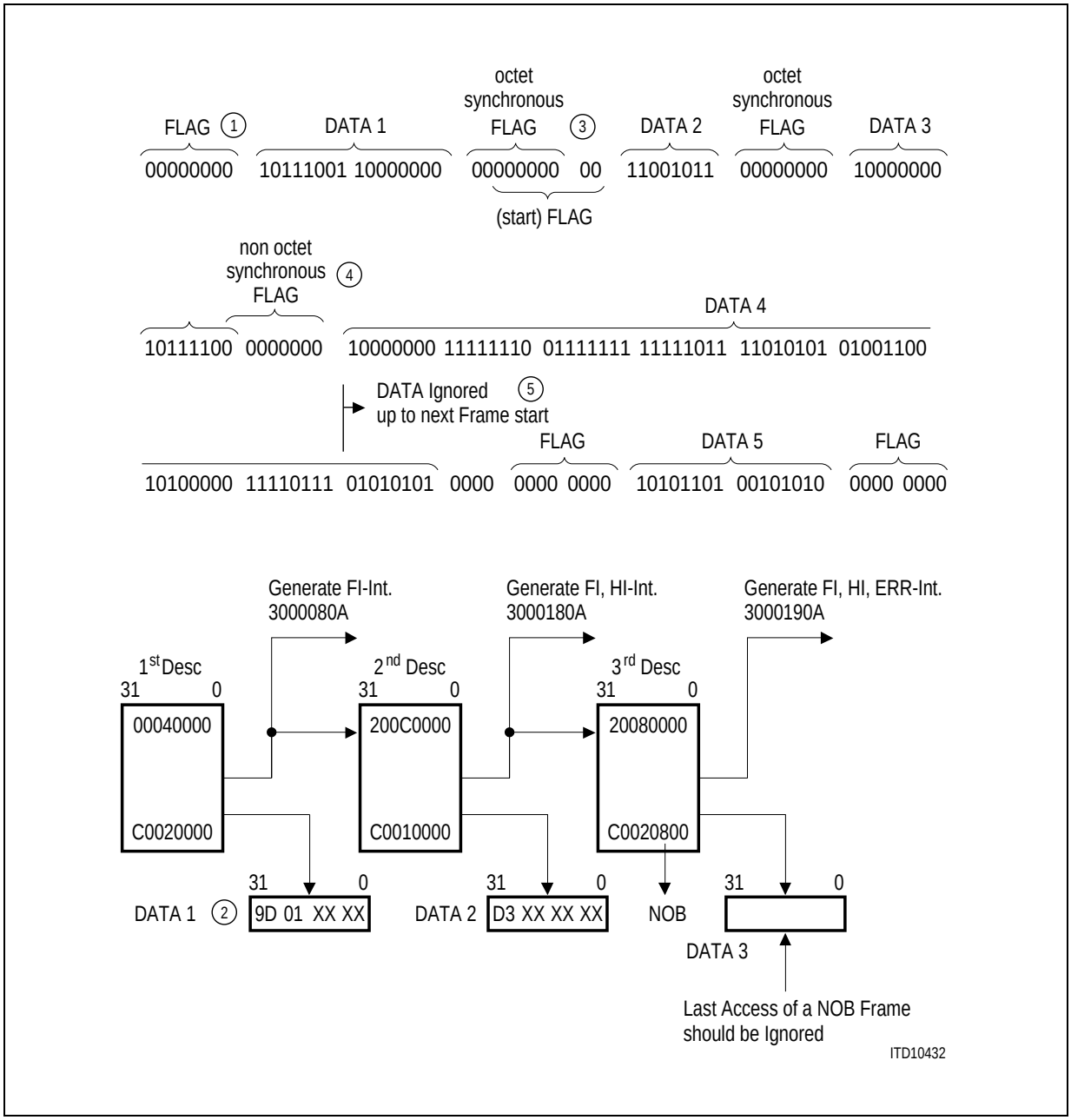


Figure 36a
Example of TMB Reception

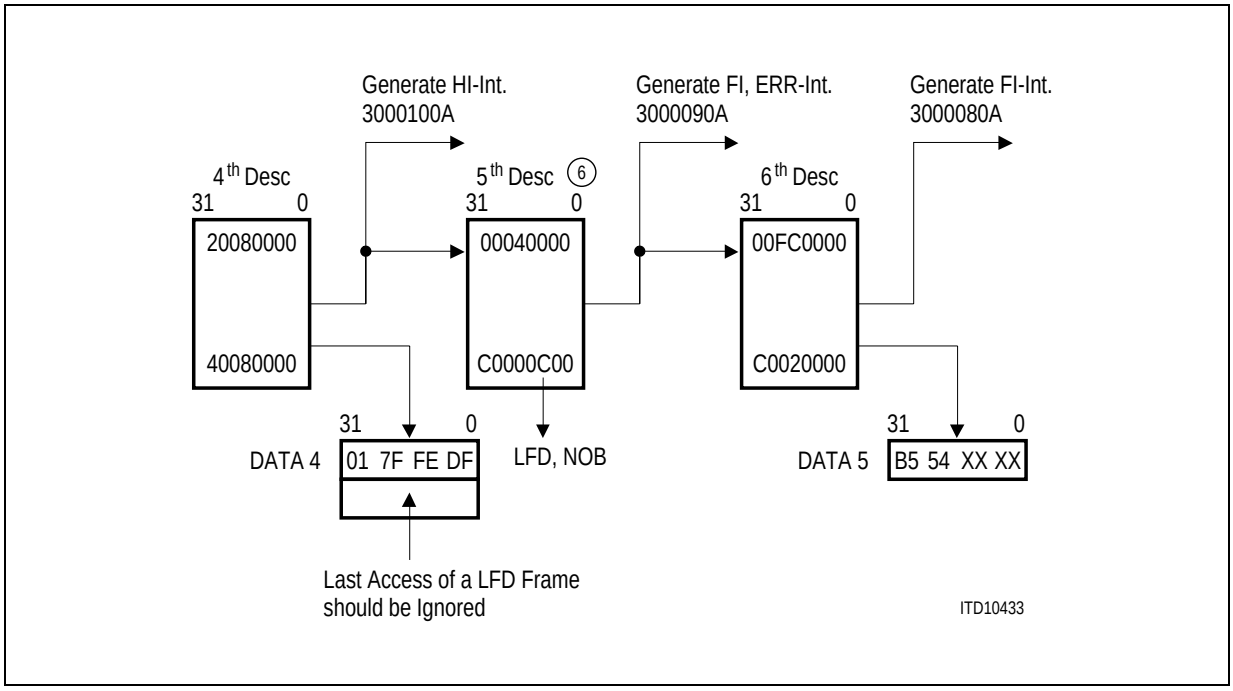


Figure 36b
Example of TMB Reception

Note: 1. After Receive Initialization is detected all data are ignored until the starting sequence 0000 0000 1 is detected.

2. Data are formatted according to §2.8 of CCITT Q.921.

3. The octet synchronous (end) flag of one frame can be part of the (start) flag of the next frame. Between DATA 1 and DATA 3 they are identical (shared flags supported).

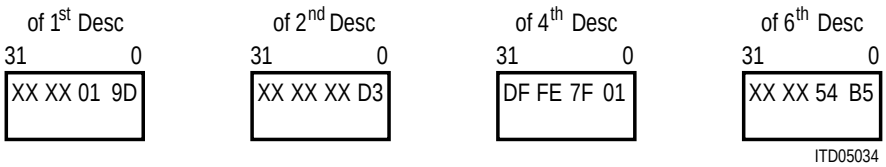
4. Here the sequence 0000 0000 1 is detected non-octet synchronously. Therefore the frame belonging to DATA 3 is supposed to have ended non-octet synchronously (NOB set in the 3rd descriptor).

5. After MFL + 1 data bytes the further data are ignored and are neither stored in the RB nor reported to the shared memory. The receiver waits for the next sequence 0000 0000 1 to come.

6. If a receive descriptor is full (4th desc.) the MUNICH32X branches to the next receive descriptor (5th desc.) even if no further data are to be given to the shared memory.

For INV = 1 (channel inversion) all bits of the data stream (including DATA, FLAG) are interpreted inversely, e.g., 1111 1111 0 would be interpreted as starting sequence.

In little endian format the only difference is in the receive data sections. They would be



4.3 TMR

Transmit Direction

General Features

In transmit direction

– the starting and ending flag (00 00_H or 0 00_H between frames) is generated automatically.

Options

The different options for this mode are

– the number of interframe time-fill characters as shown in **Figure 19** by choosing FNUM in the transmit descriptor. For the values 0, 1, 2, the following sequences are used:

FNUM = 0: … frame 1, 000_H, frame 2 …
FNUM = 1: … frame 1, 00_H, 00_H, frame 2 …
FNUM = 2: … frame 1, 00_H, 00_H, 00_H, frame 2 …

By choosing FNUM = 0 and setting the last transmitted nibble in the transmit data section to 0_H frames of effective length n + $^1/_2$ bytes can be sent as required by GSM 08.60.

Interrupts

The possible interrupts for the mode in the transmit direction are identical to those of HDLC.

A typical data stream has the form: (ITF DATA ITF DATA)

Example:

TMR channel with

INV = 0	(no inversion)
CRC = 1	(required)
TRV = 00	(required)
FA = 0	(required)
MODE = 01	(TMR)
IFTF = 0	(required)

Little Endian Data Format
Channel No. 5

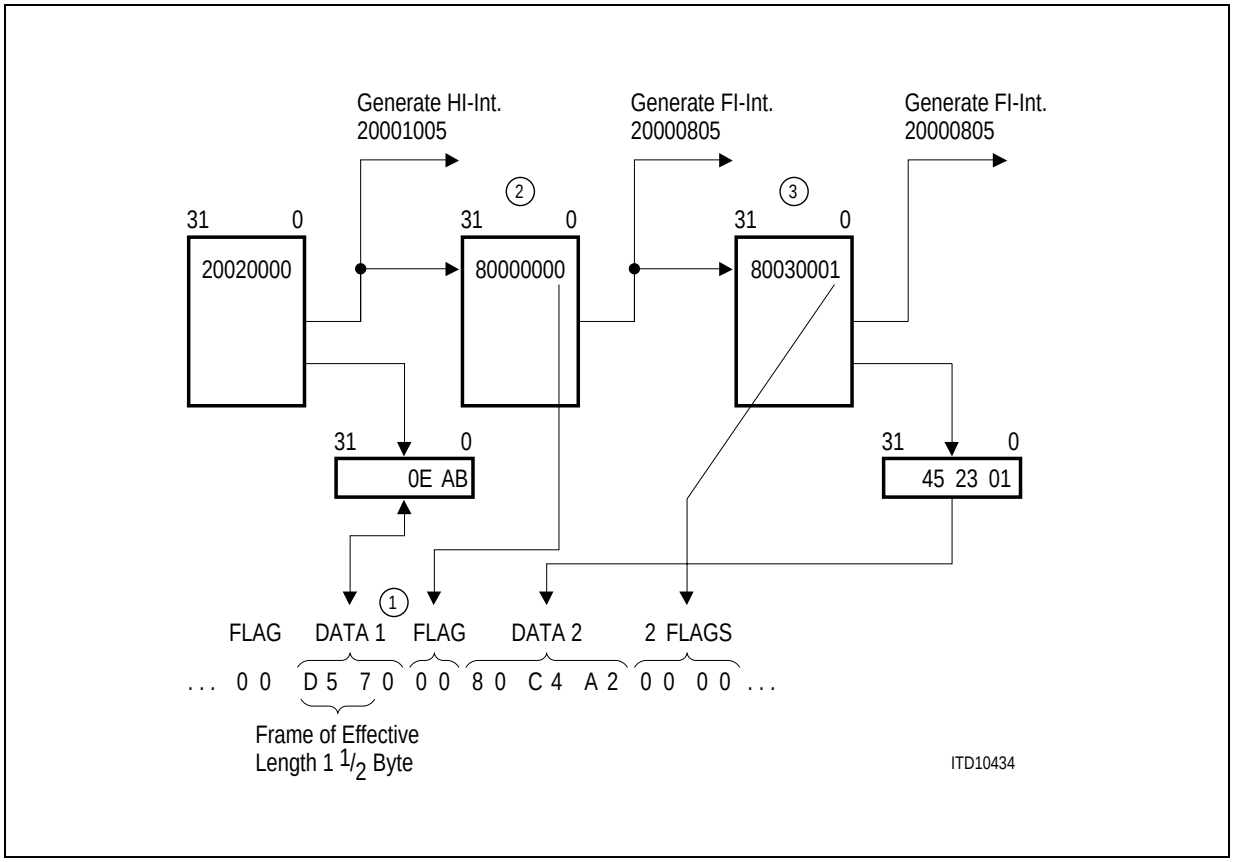


Figure 37
Example of TMR Transmission

Note: 1. Data is transmitted according to Q.921 §2.8 and fully transparent.

2. A transmit descriptor with NO = 0 and FE = 1 is allowed, one with NO = 0 and FE = 0 is forbidden.

3. FNUM = 1 leads to 2 FLAGS after DATA 2.

Receive Direction

General Features

1. The starting and the ending flag ($00\ 00_H$) is recognized. Interframe time-fill, both characters of the starting flag and the last character of the ending flag is extracted.
2. The number of bits within a frame is checked to be divisible by 8.
3. The number of bytes within a frame is checked to be smaller than MFL.

More detailed description of the individual features

1. a. A frame is supposed to have started after a sequence of 16 zeros a '1'-bit is recognized. The frame is supposed to have this '1'-bit as first bit.

   b. A frame is supposed to have stopped if
   - – either a sequence of 16 'zeros' and a 'one' is found in the data stream after the frame has started
   - – or a sequence of 16 zeros is found octet synchronous (i.e. the first bit of the sequence $00\ 00_H$ is the $8m + 1^{st}$ bit since the starting '1'-bit of 1.a. for an integer m).

   In both cases the eighth bit of the sequence $00\ 00_H$ is supposed to be the last bit of the frame.

2. The check is reported in the NOB bit in the last receive descriptor of the frame.
   NOB = 1 the bit length of the frame was not divisible by 8.
   NOB = 0 the bit length of the frame was divisible by 8.
   If NOB = 1 the last byte of the last access to a receive data section of the frame may contain erroneous bits and shouldn't be evaluated. This does **not** affect the reception of frames with $n + {}^1/_2$ octets
3. The check is reported in the LFD bit in the last receive descriptor of the frame.
   LFD = 1 the number of bytes was greater than MFL.
   LFD = 0 the number of bytes was smaller or equal to MFL.
   $MFL + 1^{st}$ one are transferred to the shared memory. The bytes of the last access to the receive data section of the frame may contain erroneous bits and should not be evaluated.
   LFD is always accompanied by NOB.

Options

There are no options in receive direction for this mode.

Interrupts

The possible interrupts for the mode in receive direction are identical to those of TMB.

Example:

TMR channel with

INV = 0 (no inversion)
CRC = 1 (required)
TRV = 00
FA = 0
MODE = 01 (TMR)
IFTF = 0 (required)
MFL = 7
Big Endian Data Format
Channel No. 15

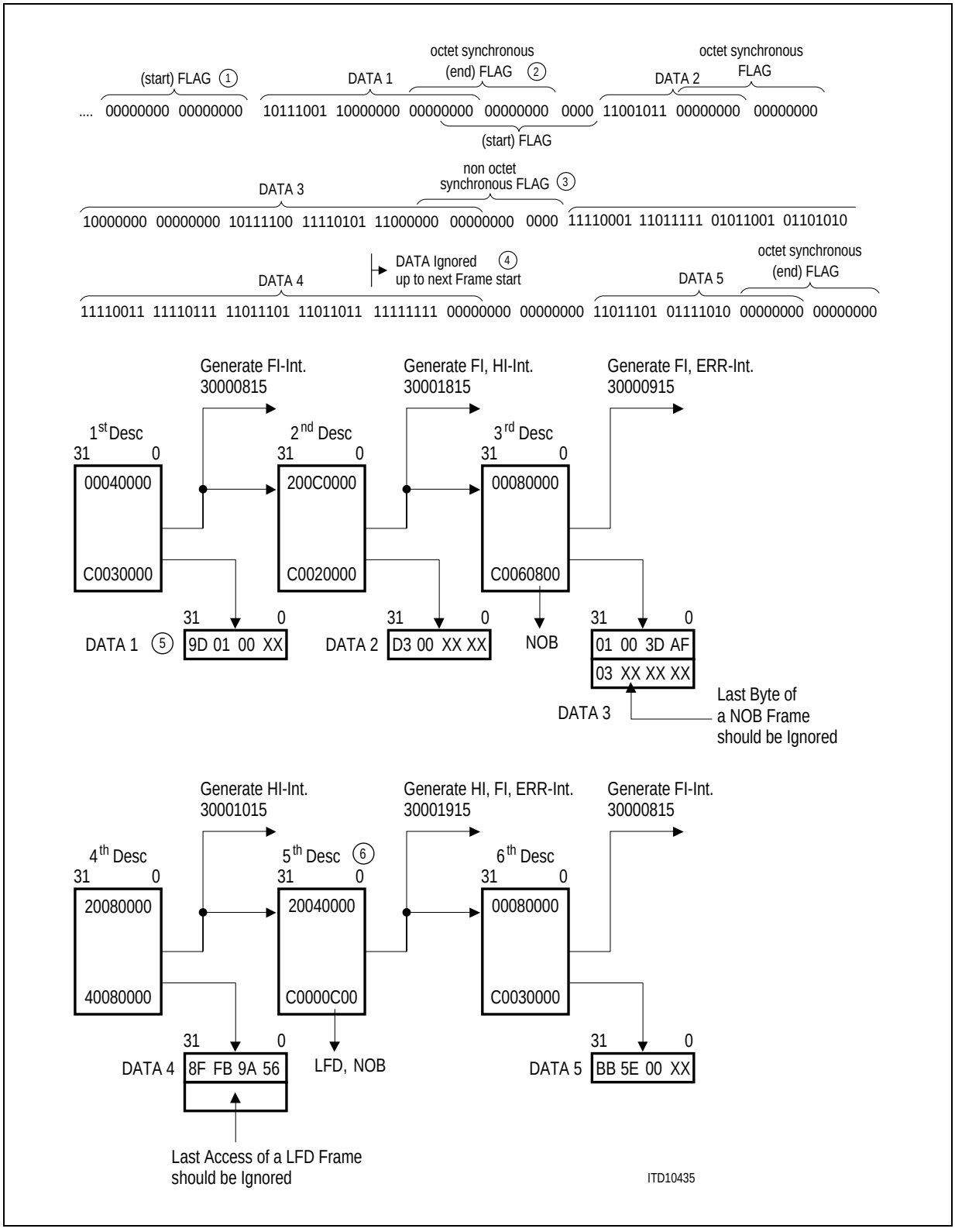
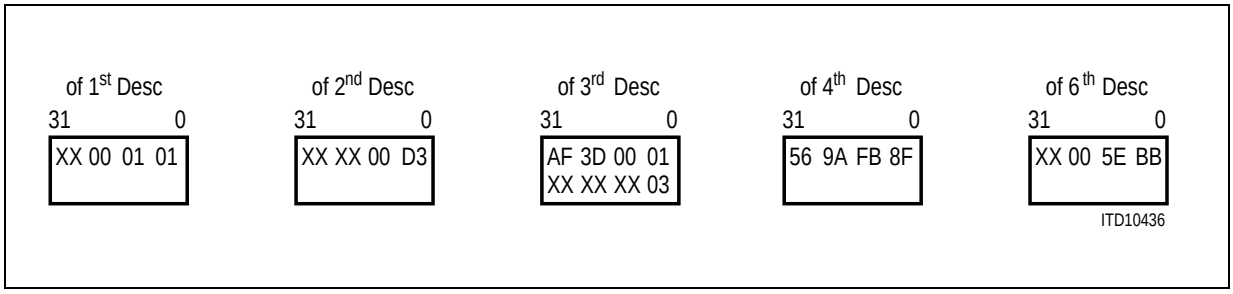


Figure 38
Example of TMR Reception

1. After receive initialization is detected all data are ignored until a starting sequence (16 ‘zeros’, ‘ones’) is detected.
2. The octet synchronous (end) flag of one frame can be part of the (start) flag of the next frame.
   Note that the first 00_H character of the end flag is stored in the receive data section as ordinary data and is included in BNO.
   Between DATA 2 and DATA 3 the start and end flag are identical (shared flags supported).
3. Here the start sequence is detected non-octet synchronously within a frame. Therefore the frame belonging to DATA 3 is supposed to have ended non-octet synchronously (NOB set in the 3rd descriptor).
4. After MFL + 1 data bytes the further data are ignored and are neither stored in the RB nor reported to the shared memory.
5. Data are formatted according to §2.8 of CCITT Q.921.
6. If a receive descriptor is full (4th descriptor) the MUNICH32X branches to the next receive descriptor (5th descriptor) even if no further data are to be given to the shared memory.

For INV = 1 (channel inversion) all bits of the data stream (including DATA, FLAG) are interpreted inversely e.g. 16 ‘ones’, ‘zeros’ is interpreted as starting sequence then.

In little endian mode the only difference is in the receive data sections. They would be



4.4 TMA

Transmit Direction

General Features

In the transmit direction

– a frame-synchronous transparent data transmission
– a programmable number of programmable fill characters after data

is generated automatically.

Synchronized Data Transfer

In order to transfer data over a PCM interface with a bandwidth greater than 64 Kbit/s, multiple time-slots must be concatenated into a single channel. E.g., video applications require 2 to 6 and more time slots; depending on the picture quality. In such applications, it is mandatory to know when the MUNICH32X starts to send real data (not only the flags).

The MUNICH32X supports fractional T1/PRI or full T1/PRI channels for high bandwidth services by synchronizing the START of the transparent data with the START of the 8 kHz frame. This allows complete T1 transparency end-to-end over the network.

Consider the case of an originating end in which in-band control information is intended to be carried in time-slot 0. By synchronizing the outgoing (transparent stream) to the Frame sync pulse, the time slot integrity can be maintained all the way to the far end.

For example, the time slots 0, 1, 5, 6, 21 and 23 are used for building a 384 Kbit/s data channel to be transferred via a fractional T1 interface (refer to **Figure 39**). As long as the MUNICH32X detects the HOLD bit = 1 (no transmit data is available) it transmits ‘FF’. Upon the detecting HOLD = 0 (in time slot 5) the MUNICH32X fetches data from shared memory and starts to transmit data in the first time slot of the assigned data channel in one of the next frames.

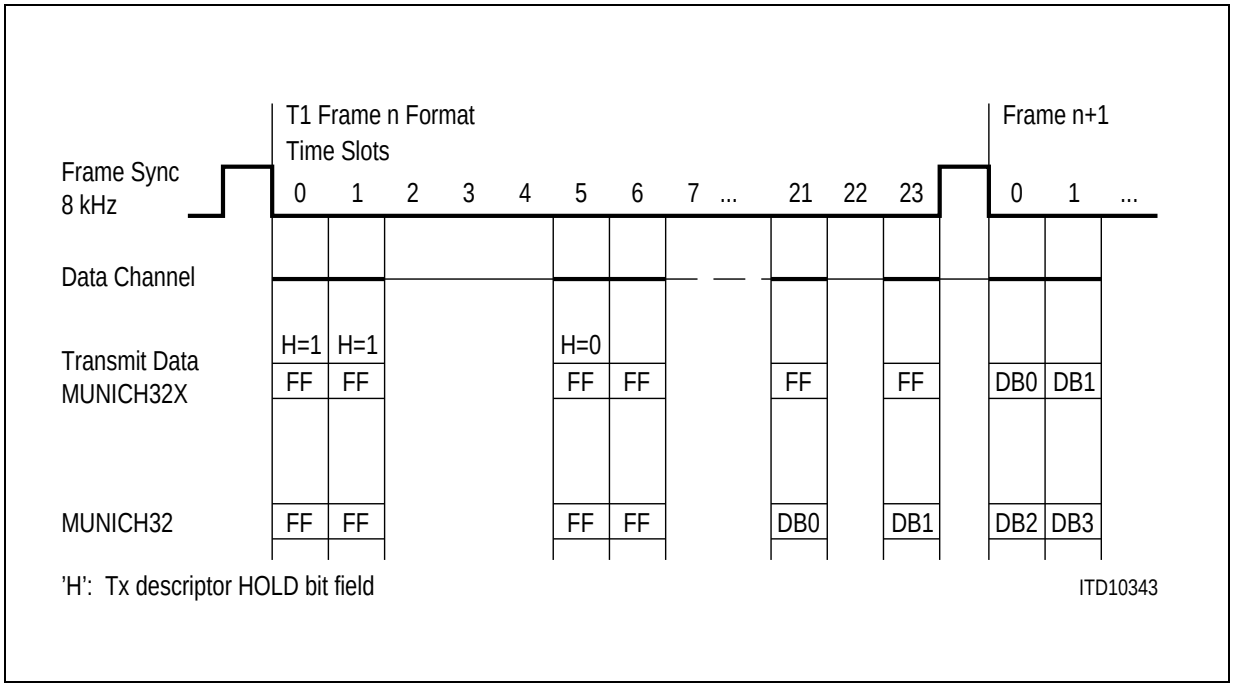


Figure 39
Example of Data Transfer Synchronization in TMA mode

The MUNICH32X starts to transmit real data in the frame n + x. The value x depends on the size of the assigned transmit buffer (ITBS value in the Channel Specification) and on the bandwidth provided for transmission. Generally, the following rule applies: The smaller ITBS and the larger the bandwidth, the earlier the MUNICH32X starts to send data.

For the above example, if ITBS was programmed to 12 DWORD, x would be 3. Thus the synchronized data transmission starts in the frame n + 3.

If the MUNICH32X has to send multiple data frames terminated by a Frame End (without descriptor HOLD bit = '1' between the frames), it synchronizes the beginning of every data frame to the frame sync pulse.

After the transmission of the last data frame byte the MUNICH32X sends inter-frame time fill flags (TFLAG programmed in the Channel Specification) according to the number programmed in the Transmit Descriptor (FNUM)

Note: MUNICH32 starts to transfer data immediately after detecting HOLD = 0.

Options

The different options for this mode are

– The value of the fill-character can be programmed for FA = 1 in the channel specification. The fill-character (TC) is then programmed in the TFLAG. For FA = 0 the fill character is FF_H and TFLAG has to be set to 00_H. If subchanneling is chosen (not all fill/mask bits of the channel are ‘1’) FA must be set to ‘0’.
– The number of inter-data time-fill characters as shown in **Figure 23**.
By setting the value of FNUM the following sequences result:

FNUM = 0… DATA 1, TC, DATA 2 …
FNUM = 1… DATA 1, TC, TC, DATA 2 …
FNUM = 2… DATA 1, TC, TC, TC, DATA 2 …

– DATA 2 starts at synchronization pulse TSP; ‘1’ are sent between last TC and DATA 2.
– In case of using subchanneling by fill masks the MUNICH32X supports two different modes of operation in Transparent Mode A (TMA). These modes are selected by bit 'CRC' in the channel configuration.
CRC = ‘0’:
Data is transmitted transparently only in bit positions selected by the transmit fill mask (corresponding fill mask bit equal ‘1’). Masked bit positions are driven Tristate ‘Z’. In receive direction bits are received from bit positions selected by the receive fill mask (corresponding fill mask bit equal ‘1’) only. Receive data is grouped to octets and stored in memory transparently (no gaps).
CRC = ‘1’:
In transmit direction each data octet is masked with the transmit fill mask. Masked bit positions are overwritten with Tristate ‘Z’ when transmitted. In receive direction the receive fill mask has to be set to 0xFF. The entire 8 bit time slot is received and stored byte aligned in memory. It is the software responsibility to mask received data octets as needed by the application.

Interrupts

The possible interrupts for this mode in transmit direction are identical to those of HDLC.

Example 1:

(no subchanneling by fill/mask bits)

TMA channel with

TFLAG = $B2_H$
INV = 0 (no data inversion)
CRC = 0
TRV = 00 (required)
FA = 1 (flag filtering)
MODE = 00 (TMA)
IFTF = 0 (required)

All fill-mask bits are '1' for this channel (no high impedance overwrite)
Little Endian Data Format
Channel no. D

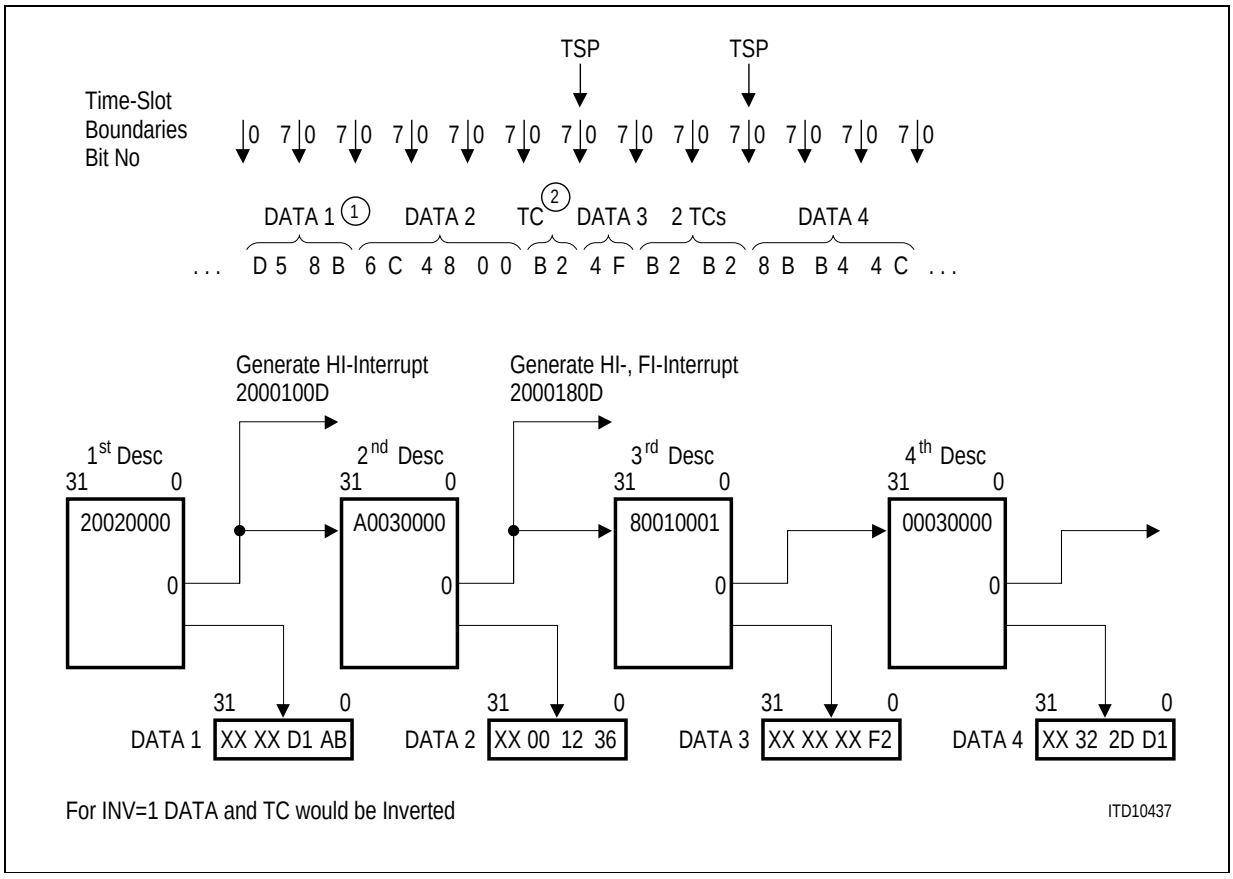


Figure 40

Example of TMA Transmission

Note: 1. Data are formatted according to §2.8 of Q.921. The TC is transmitted MSB (bit 15) first though!!!

2. FNUM = 0 in the second descriptor leads to the insertion of the TC after DATA 2, FNUM = 1 in the third descriptor to the insertion of 2 TCs.

3. A sync-pulse defined number of '1' is inserted between 'TC-DATA 3' and 'TC-DATA 4.

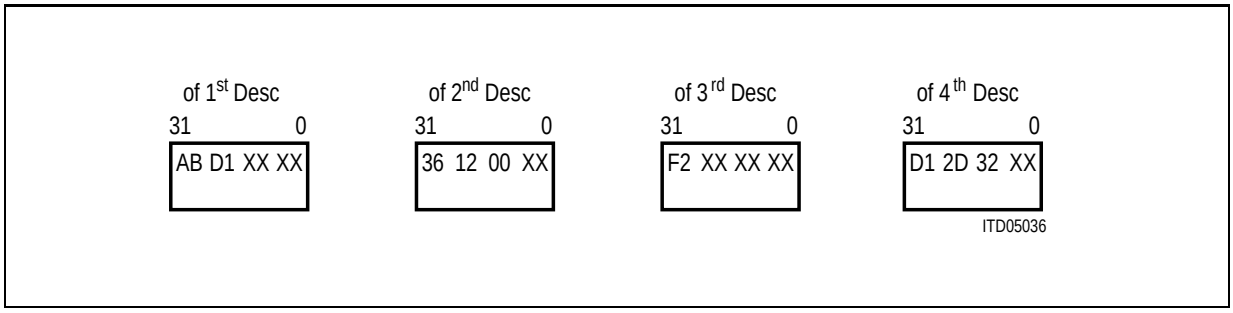
For INV = 1 the data stream would be inverted completely:

	DATA 1	DATA 2	TC	DATA 3	2 TCs	DATA 4	
…	2A 74	93 B7 FF	4D	B0	4D 4D	74 4B B3	…

For FA = 0 TFLAG has to be programmed to 00_H, resulting in a data stream of

DATA 1	DATA 2	TC	DATA 3	2 TCs	DATA 4
D5 8B	6C 48 00	FF	4F	FF FF	8B B4 4C

In big endian mode, the data sections for the same data stream would have been



Example 2:

(subchanneling by fill/mask bits)

TMA channel with

TFLAG = 00_H (required for this case)
INV = 0 (no data inversion)
CRC = 0
TRV = 00 (required)
FA = 0 (required for subchanneling)
MODE = 00 (TMA)
IFTF = 0 (required)
Little Endian Data Format
Channel no. D

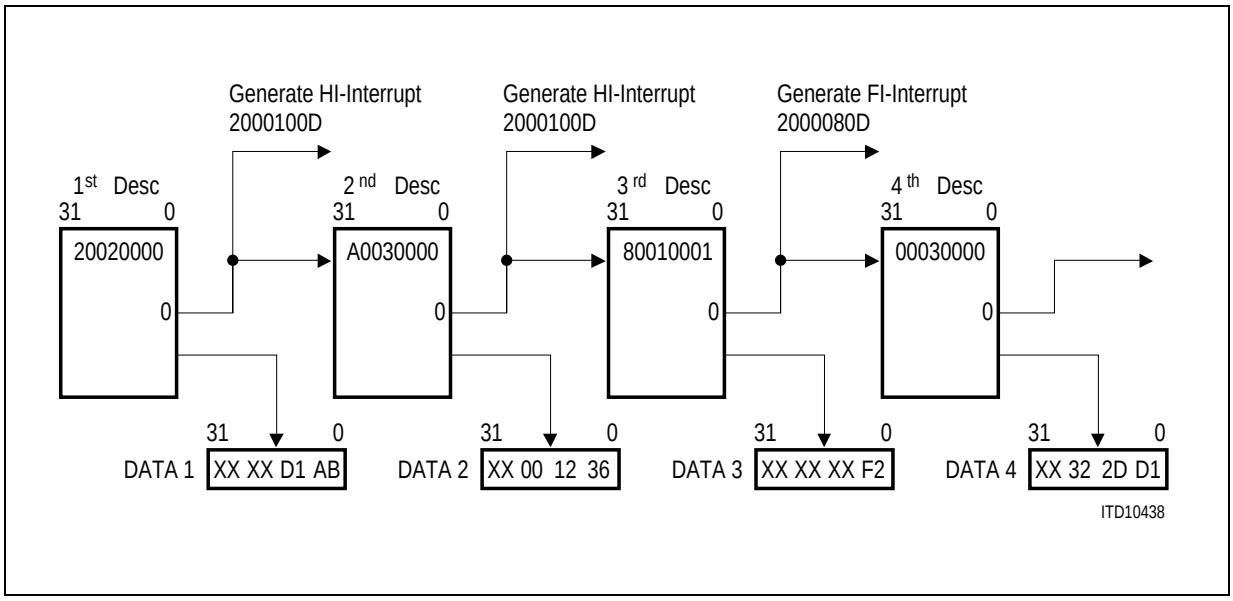


Figure 41
Example of TMA Transmission

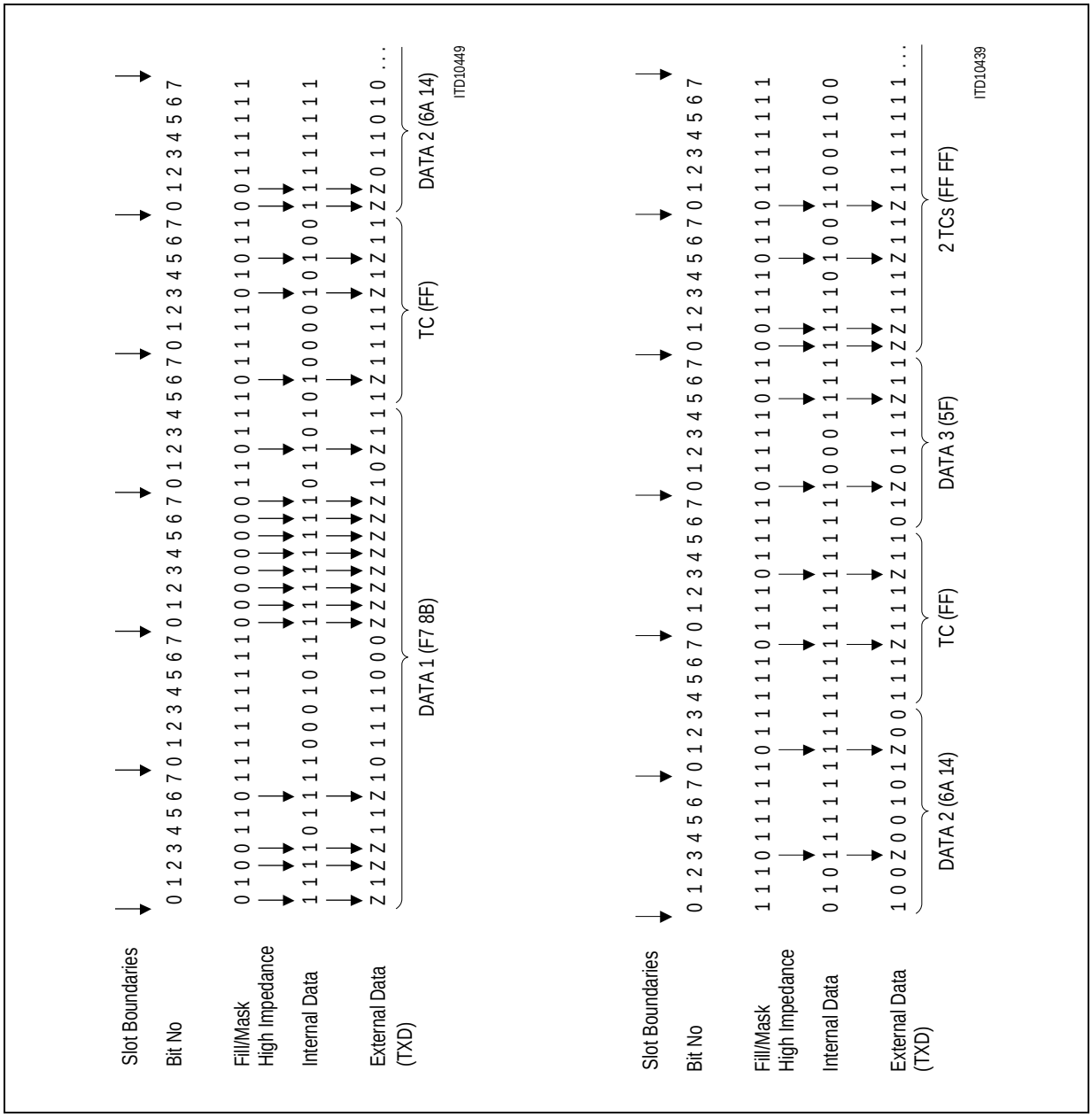


Figure 42

Receive Direction

General Features

In the receive direction

– a slot synchronous transparent data reception
– for FA = ‘1’ a slot synchronous programmable flag extraction

is performed automatically.

Options

The different options for this mode are:

– the programmable character TC to be extracted for FA = ‘1’ is TFLAG. For FA = ‘0’ nothing is extracted. If subchanneling is chosen (not all fill/mask bits of the channel are ‘1’) FA must be set to ‘0’.
– In case of using subchanneling by fill masks the MUNICH32X supports two different modes of operation in Transparent Mode A (TMA). These modes are selected by bit ‘CRC’ in the channel configuration.
  CRC = ‘0’:
  Receive data is grouped to octets and stored in memory transparently (no gaps).
  CRC = ‘0’:
  In receive direction the receive fill mask has to be set to 0xFF. The entire 8 bit time slot is received and stored byte aligned in memory. It is the software responsibility to mask received data octets as needed by the application.

Interrupts

The possible interrupts for the mode in receive direction are:

HI: issued if the HI bit is detected in the receive descriptor (not maskable).

ERR: issued if a fast receive abort channel command was issued.
(maskable by RE in the channel spec.)

FO: issued if data could only partially stored due to internal buffer overflow of RB.
(maskable by RE in the channel spec.)

Example 1:

(no subchanneling)

TMA channel with

TFLAG = D7
INV = 0 (no channel inversion)
CRC = 0
TRV = 00 (required)
FA = 1
MODE = 00 (TMA)
IFTF = 0
Big Endian Data Format
Channel No. E

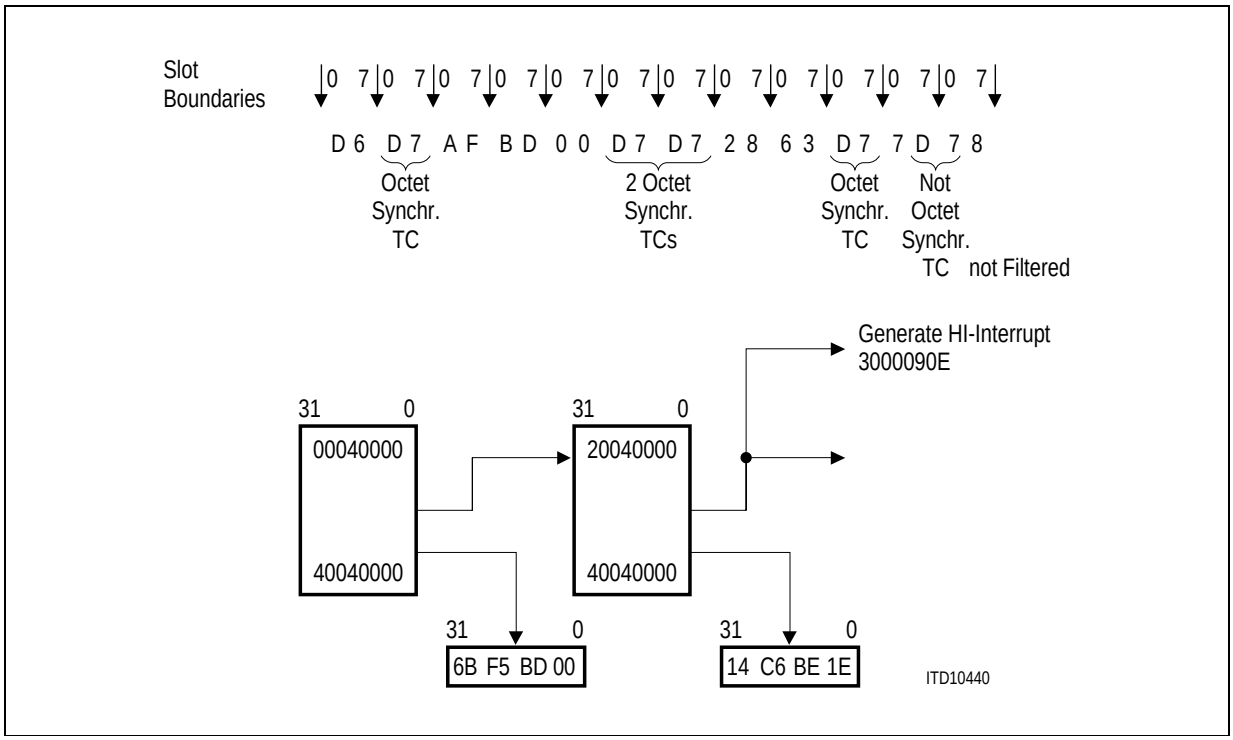


Figure 43a
Example of TMA Reception

*Note: The FE bit is **never** set in a receive descriptor.*
Data are formatted according to §2.8 Q.921.

For FA = 0 (and therefore TFLAG = 00_H), the descriptor would be

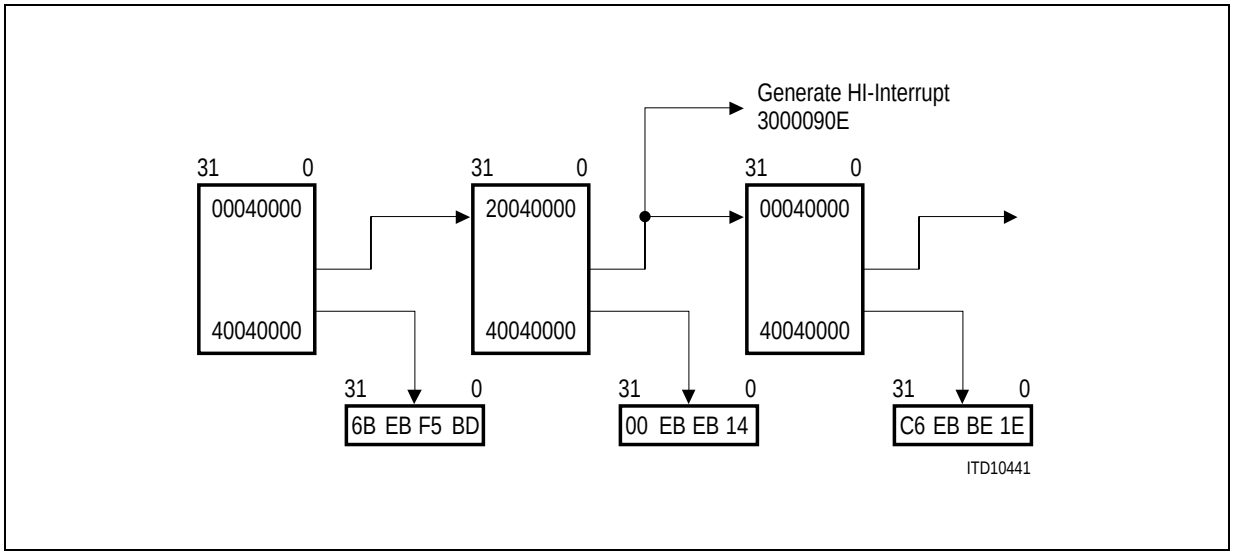


Figure 43b
Example of TMA Reception

For INV = 1 the receiver filters the inverse of the TFLAG as TC out of the data stream and inverts the data (only the octet synchronous 28_H would be filtered).

In little endian mode, the data sections for the first descriptor would be:

00 BD F5 6B

and for the second descriptor:

1E BE C6 14

4.5 V.110/X.30

Transmit Direction

General Features

In transmit direction

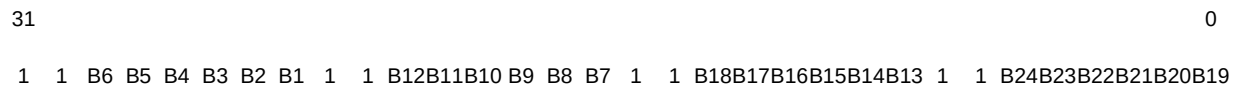
– the synchronization pattern for V.110/X.30 frame as shown in **Table 7**.
– the framing for the different data rates with programmable E-, S-, X-bits
– sending ‘0’ before all frames

is performed automatically.

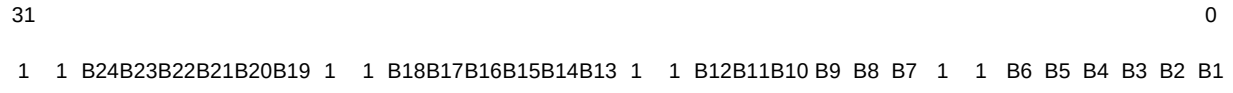
Table 7
Synchronization Pattern for V.110/X.30-Frames

Octet No.	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	1							
3	1							
4	1							
5	1							
6	1							
7	1							
8	1							
9	1							
10	1							

The E-, S-, X-bits are fed into the data stream by special transmit descriptor (as shown in **Figure 20**), they can only change from one 10-octet frame to the next, not within a 10-octet frame. The data from the data sections are supposed to come in the form:



(in big endian mode)



(in little endian mode),

where, in case of a transmission rate of 600 bit/s, for example B1 to B6 belong to the first 10-octet frame, B7 to B12 belong to the second 10-octet frame, etc.

Options

The different options for this mode are:

– the framing pattern, as shown in **Table 8** to **Table 11**, is programmed by the bits TRV.

Interrupts

HI: issued if the HI bit is detected in the transmit descriptor (not maskable)

ERR: if one of the following transmit errors has occurred
- the last descriptor had FE = 1 (leads to an abort of the transmit data, see **Figure 21**)
- the last descriptor had H = 1 (see **Figure 19**)
- the last descriptor had NO = 0 (maskable by TE in the channel spec.)

FO: issued if the MUNICH32X was unable to access the shared memory in time either for new data to be sent or for a new descriptor. (maskable by TE in the channel spec.)

Example

V.110/X.30 channel with

CS = 0 (required)
INV = 0
CRC = 0
TRV variable (all values shown in examples)
FA = 0 (required)
MODE = 10 (V.110/X.30)
Little Endian Data Format
Channel No. 1F

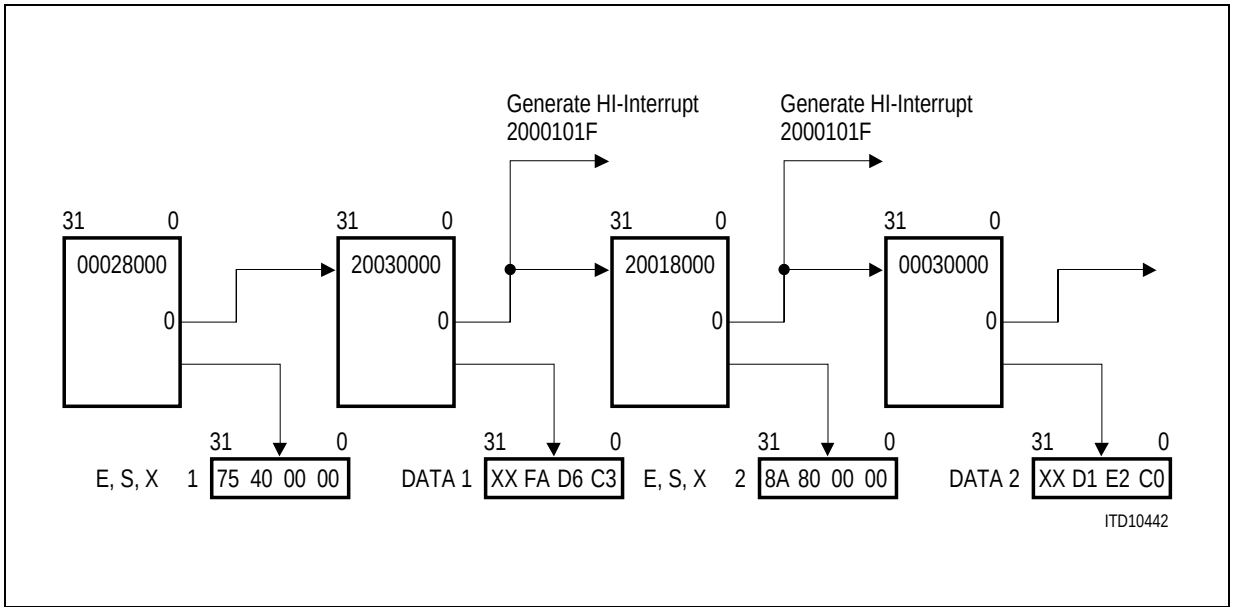
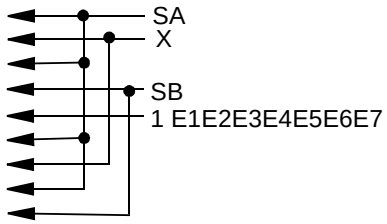


Figure 44
Example of V.110/X.30 Transmit Mode

*Note: The first transmit descriptor **must** have the V.110-bit set.*

TRV = 00

0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
1 1 1 1 1 0 0 0
1 0 0 0 0 0 0 1
1 0 1 0 1 1 1 0
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1



0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 1 1 1 1 1
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
1 0 1 0 1 1 1 0
1 0 0 0 0 0 0 0
1 0 0 1 1 1 1 1
1 1 1 1 1 0 0 0
1 0 0 0 0 0 0 1

D6 = 11 0 1 0 1 1 0
B6B5B4B3B2B1

0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
1 0 0 1 1 1 1 1
1 1 1 1 1 0 0 0
1 0 0 0 0 0 0 1
1 0 1 0 1 1 1 0
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1

FA = 11 1 1 1 0 1 0
B6B5B4B3B2B1

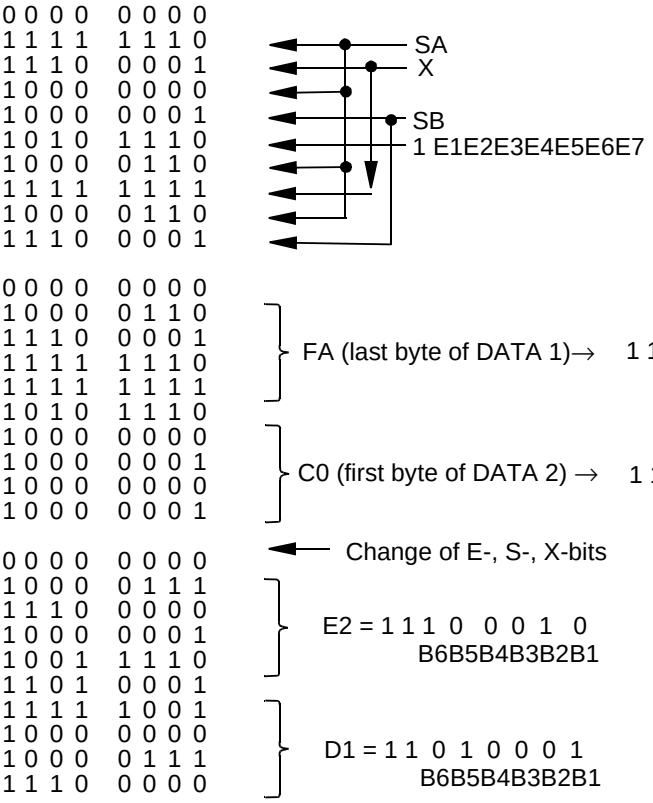
Change of E-, S-, X-bits

0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 1 0 1 0 0 0 1
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0

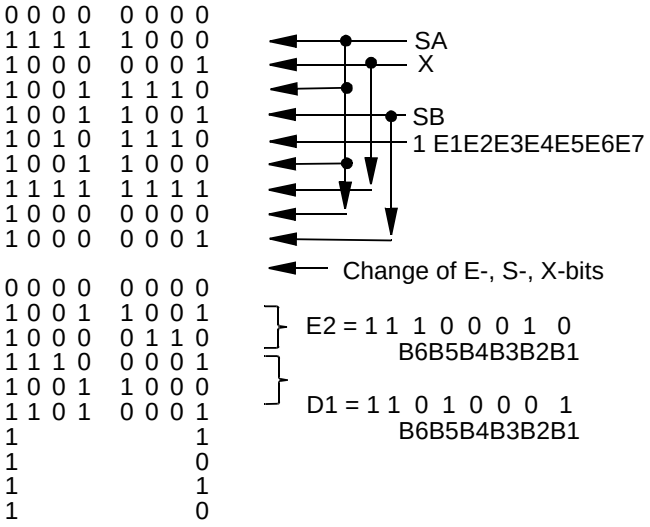
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 1 1 1 1 0
1 1 1 1 1 0 0 1
1 0 0 0 0 0 0 0
1 1 0 1 0 0 0 1
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 0 0 0 0 1 1 1
1 1 1 1 1 1 1 0

0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0
1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
1 1 0 1 0 0 0 1
1 0 0 0 0 0 0 1
1 0 0 1 1 1 1 0
1 1 1 1 1 0 0 1
1 0 0 0 0 0 0 0

TRV = 01



TRV = 10

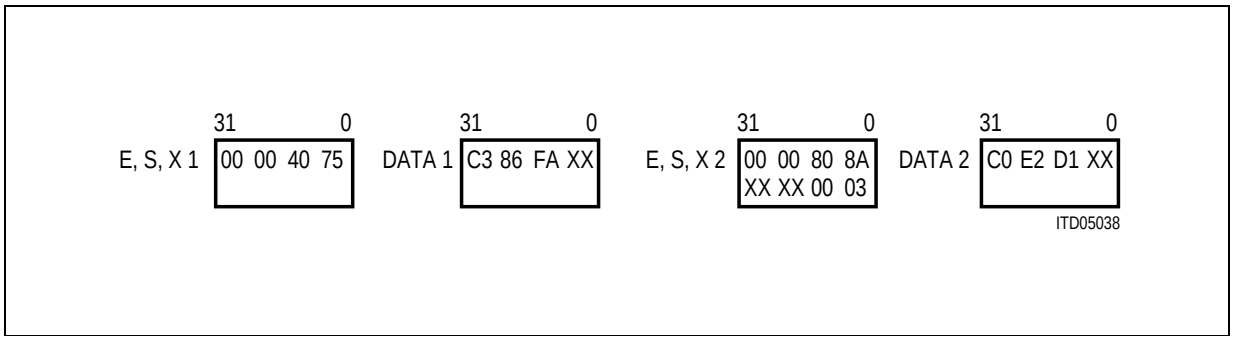


TRV = 11

```
0 0 0 0   0 0 0 0
1 1 1 0   0 0 0 0
1 0 1 1   0 1 0 1
1 0 1 0   1 1 1 0
1 0 0 0   0 0 0 1
1 0 1 0   1 1 1 0
1 0 1 0   0 0 1 0
1 1 0 0   0 1 0 1
1                 0
1                 1
```

← Change of E-, S-, X-bits

For INV = 1 (channel inversion) all bits are inverted. In big endian mode the data sections must have the following form to yield the same output data:



Receive Direction

General Features

In receive direction

- the starting sequence (00_H followed by a ‘1’-bit) after initialization of loss of synchronism is detected.
- the synchronization pattern is monitored, after 3 consecutive erroneous frames a loss of synchronism is detected.
- a change of E-, S-, X-bits is monitored and reported by an interrupt.
- the data bits are extracted and written into the data section.

More detailed description of the individual features:

The receiver can be in two different states:

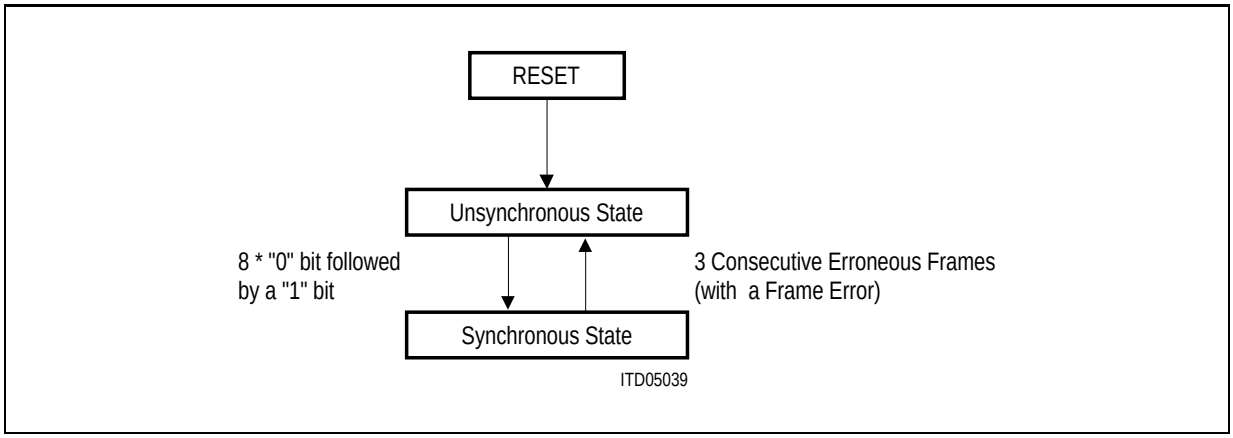


Figure 45
Receiver States in V.110/X.30

1. Data extraction and monitoring of a change of E-, S-, X-bits and synchronization pattern is only performed in synchronized state.
2. In the unsynchronized state the receiver waits for the synchronization pattern. The ‘1’-bit is then interpreted as bit 1 of octet 2.
3. During the synchronized state a change of E, S, X-bits from one frame to the next and even within a frame (for SA, SB bits) is monitored. Only one interrupt per frame is reported even if SA e.g. changes 3 times within the frame. The E-, S-, X-bits reported in the interrupt are S9 for SB and S8 for SA and the second occurrence of X for X.
4. The bits written into the data section are marked by O in **Table 8** to **Table 10**. As shown, bits repeated in the serial data are only strobed at their last instance.

Table 8
Framing for Networks with 600-bit/s Data Rate
Intermediate Rate = 8 Kbit/s, i.e. Subchannelling with Only 1 Fill/Mask Bit Set

Octet No.	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	1	B1	B1	B1	B1	B1	B1	S1
3	1	B1	B1	B2	B2	B2	B2	X
4	1	B2	B2	B2	B2	B3	B3	S3
5	1	B3	B3	B3	B3	B3	B3	S4
6	1	E1	E2	E3	E4	E5	E6	E7
7	1	B4	B4	B4	B4	B4	B4	S6
8	1	B4	B4	B5	B5	B5	B5	X
9	1	B5	B5	B5	B5	B6	B6	S8
10	1	B6	B6	B6	B6	B6	B6	S9

Table 9
Framing for Networks with 1200-bit/s Data Rate
Intermediate Rate = 8 Kbit/s, i.e. Subchannelling with Only 1 Fill/Mask Bit Set

Octet No.	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	1	B1	B1	B1	B1	B2	B2	S1
3	1	B2	B2	B3	B3	B3	B3	X
4	1	B4	B4	B4	B4	B5	B5	S3
5	1	B5	B5	B6	B6	B6	B6	S4
6	1	E1	E2	E3	E4	E5	E6	E7
7	1	B7	B7	B7	B7	B8	B8	S6
8	1	B8	B8	B9	B9	B9	B9	X
9	1	B10	B10	B10	B10	B11	B11	S8
10	1	B11	B11	B12	B12	B12	B12	S9

Table 10
Framing for Networks with 2400-bit/s Data Rate
Intermediate Rate = 8 Kbit/s, i.e. Subchannelling with Only 1 Fill/Mask Bit Set

Octet No.	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	1	B1	B1	B2	B2	B3	B3	S1
3	1	B4	B4	B5	B5	B6	B6	X
4	1	B7	B7	B8	B8	B9	B9	S3
5	1	B10	B10	B11	B11	B12	B12	S4
6	1	E1	E2	E3	E4	E5	E6	E7
7	1	B13	B13	B14	B14	B15	B15	S6
8	1	B16	B16	B17	B17	B18	B18	X
9	1	B19	B19	B20	B20	B21	B21	S8
10	1	B22	B22	B23	B23	B24	B24	S9

Table 11
Framing for Networks with 4800-, 9600-, 19200-, 38400-bit/s Data Rate
Intermediate Rate = 8, 16, 32, 64 Kbit/s, i.e. Subchannelling with 1, 2, 4, 8 Fill/Mask Bit Set

Octet No.	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	1	B1	B2	B3	B4	B5	B6	S1
3	1	B7	B8	B9	B10	B11	B12	X
4	1	B13	B14	B15	B16	B17	B18	S3
5	1	B19	B20	B21	B22	B23	B24	S4
6	1	E1	E2	E3	E4	E5	E6	E7
7	1	B25	B25	B27	B29	B29	B30	S6
8	1	B31	B32	B33	B35	B35	B36	X
9	1	B37	B36	B39	B41	B41	B42	S8
10	1	B43	B44	B45	B47	B47	B48	S9

They are grouped together in the form:

31 0

1 1 B6 B5 B4 B3 B2 B1 1 1 B12 B11 B10 B9 B8 B7 1 1 B18 B17 B16 B15 B14 B13 1 1 B24 B23 B22 B21 B20 B19

(in big endian mode)

31 0

1 1 B24 B23 B22 B21 B20 B19 1 1 B18 B17 B16 B15 B14 B13 1 1 B12 B11 B10 B9 B8 B7 1 1 B6 B5 B4 B3 B2 B1

(in little endian mode)

where for the 600 bit/s e.g. B1 to B6 belong to the first 10-octet frame, B7 to B12 belong to the second 10-octet frame etc.

Options

The different options for this mode are the framing pattern as shown in **Table 8** to **Table 11**. They are programmed by the bits TRV (Transmission Rate) in the Channel Specification.

Interrupts

The possible interrupts for this mode are

FRC: issued if the receiver has detected a change of S-, X-, E-bits; the value of the bits E7, …, E1, S8 for SA and S9 for SB and the second occurrence of X within the 10-octet frame is reported within the same interrupt.
(maskable by CH in the channel specification)

HI: issued if the HI bit is detected in the transmit descriptor (not maskable).

ERR: issued if one of the following receive errors has occurred:
- a fast receive abort channel command was issued (this leads to a setting of the RA bit in the status byte)
- data could only partly be stored due to internal buffer overflow of RB
- 3 consecutive frames had an error in the synchronization pattern (loss of synchronism)
- the HOLD bit in the receive descriptor was detected (this leads to a setting of the RA bit in status in the receive descriptor).
(maskable by RE in the channel specification)

FO: issued if due to inaccessibility of the internal buffer (RB) one or more changes of E-, S-, X-bits and/or loss of synchronism information have been lost.
(maskable by RE in the channel specification)

Example

V.110/X.30 channel with

CS = 0 (required)
INV = 0
CRC = 0
TRV = 00 (600 bit/s)
FA = 0
MODE = 10 (V.110/X.30)
Big Endian Data Format
Channel No. D

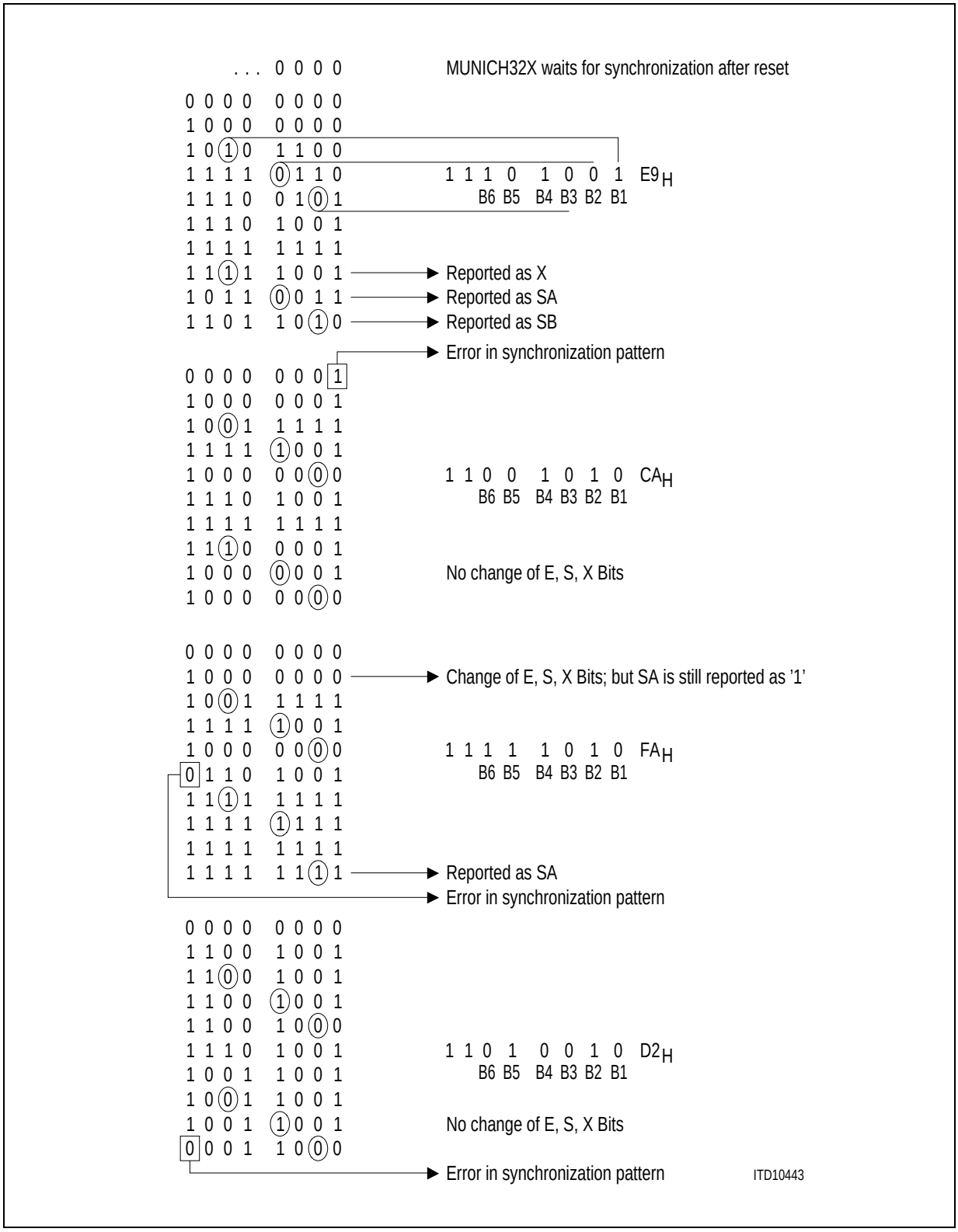


Figure 46a
Example of V.110/X.30 Receive Mode

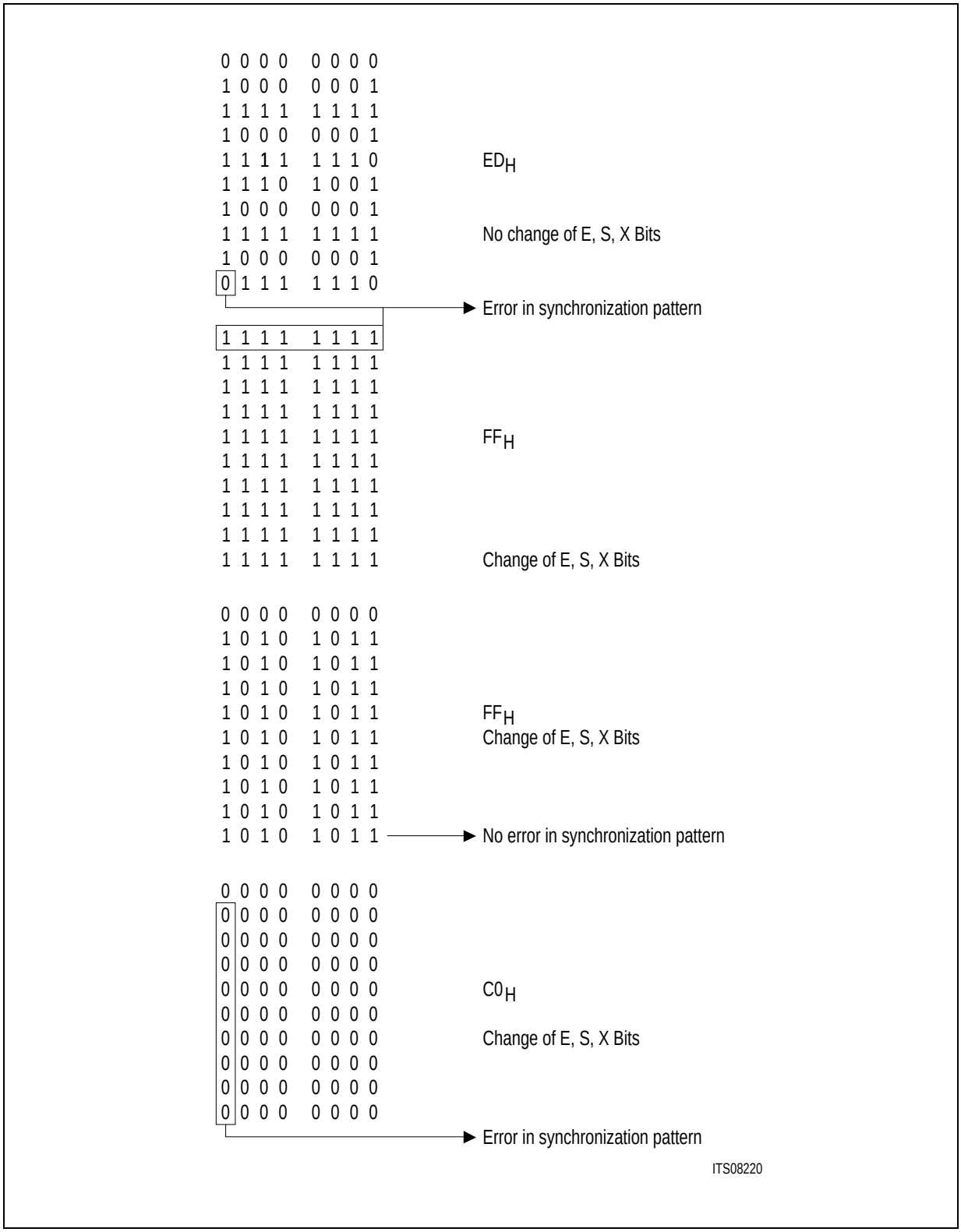


Figure 46b
Example of V.110/X.30 Receive Mode

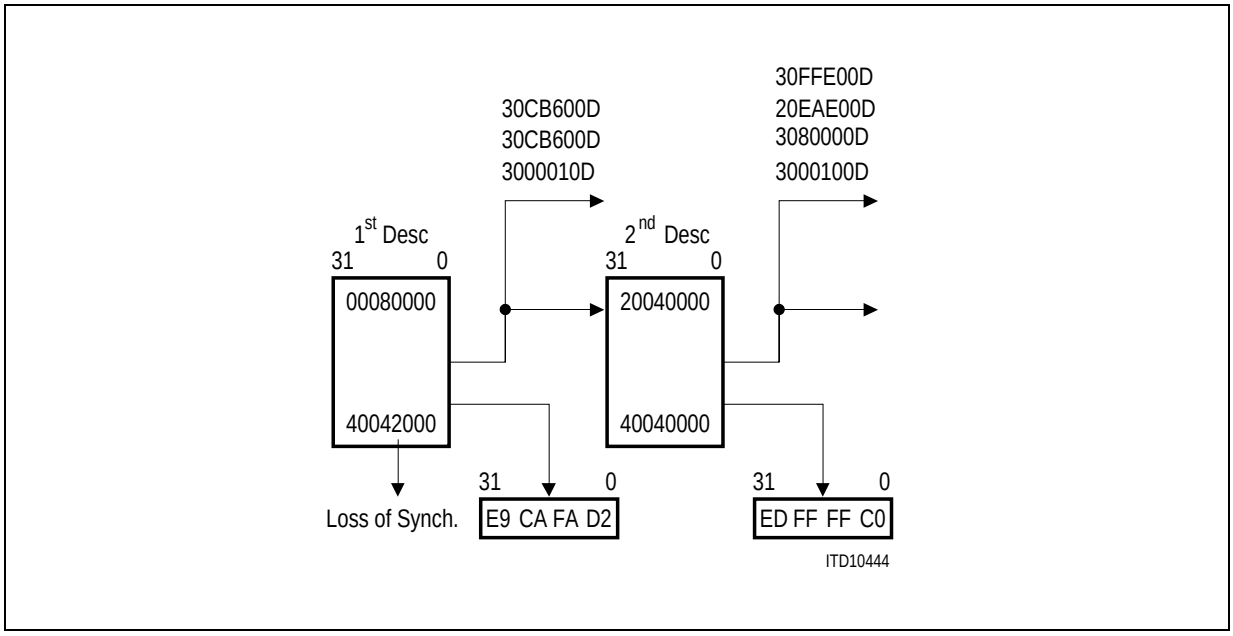
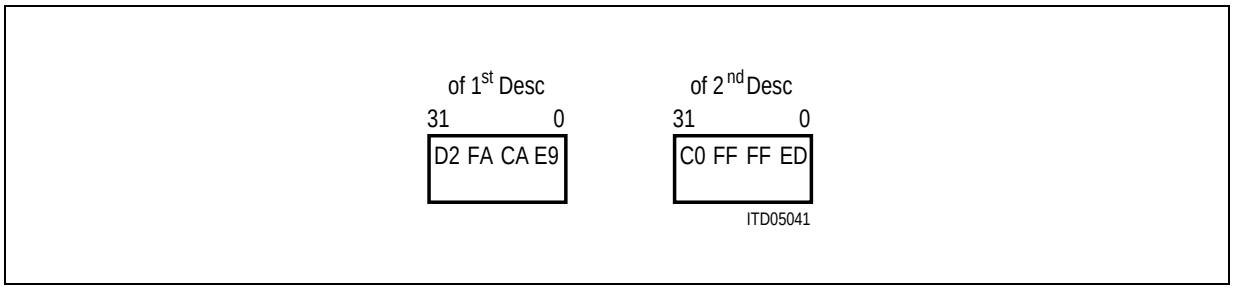


Figure 47
Loss of Synchronism in V.110/X.30

In little endian mode, the data sections have the form:



5 Microprocessor Bus Interface

The MUNICH32X may be configured either for 33 MHz/32-bit PCI operation, or for a 33 MHz/32-bit De-multiplexed bus. The MUNICH32X's DEMUX input pin is used to select the desired configuration ('0' = PCI, '1' = DEMUX).

The MUNICH32X provides identical DMA controller capability for both interfaces.

When in the PCI configuration, connection to other peripherals (e.g., ISDN transceivers, FALC54, ISAC-S or ESCC2) may be made through the MUNICH32X's Local Bus Interface (LBI).

5.1 PCI Bus Interface

In this configuration, the MUNICH32X interfaces directly to a 33 MHz/32-bit PCI bus. During run-time, the MUNICH32X operates mostly as a PCI Master; it may be accessed by the host processor as a PCI Slave. During device configuration, the MUNICH32X operates only as a slave device; memory transactions are used to configure the device. The MUNICH32X is compliant with the PCI specification 2.1 at up to 33 MHz.

In addition, the MUNICH32X supports little/big endian byte swapping for the data section, and unaligned-byte accesses for transmit data.

5.1.1 PCI Transactions Supported

Memory accesses as a PCI Master: The MUNICH32X supports both the PCI Memory Write and PCI Memory Read commands. For the PCI Memory Write command, it writes to an agent mapped in the memory access space, while for the PCI Read command, it reads from an agent mapped in the memory address space.

I/O accesses as a PCI Master: The MUNICH32X does not support the PCI I/O Write nor PCI I/O Read commands.

Memory accesses as a PCI Slave: The MUNICH32X supports both the PCI Memory Write and PCI Memory Read commands. For the PCI Memory Write command, the MUNICH32X is written to as an agent mapped in the memory address space, while for the PCI Memory Read command, the MUNICH32X is read from as an agent mapped in the memory address space.

I/O accesses as a PCI Slave: The MUNICH32X does not support the PCI I/O Write nor PCI I/O Read commands.

Burst Capability: Read/write descriptors: up to 3 DWORDs, read/write data for MUNICH32 core: 1 DWORD, read/write data for LBI interface: up to 8 DWORDs.

5.1.2 PCI Configuration Space Registers

The PCI Configuration Space Registers of the MUNICH32X are listed **Table 12**.

For a detailed description of the standard registers refer to **Chapter 6** (Configuration Space) of the PCI specification 2.1.

Table 12
PCI Configuration Space Registers

Register Name		Read/ Write	Absolute Address Pins IDSEL & AD(7:2)	Reset Value
Standard Configuration Space Registers				
Device ID/Vendor ID	DID/VID	R	00_H	$2101110A_H$
Status/Command	STA/CMD	R/W	04_H	02800000_H
Class Code/Revision ID V2.2	CC/RID	R	08_H	02800013_H
Builtin Self Test/Header Type/Latency Timer/ Cache Line Size	BIST/HEAD/ LATIM/ CLSIZ	R/W	$0C_H$	00000000_H
Base Address 1	BAR1	R/W	10_H	00000000_H
Base Address 2	BAR2	R/W	14_H	00000000_H
Base Address 3	BAR3	R/W	18_H	00000000_H
Base Address not used	BARX	R/W	$1C_H$-24_H	00000000_H
Cardbus CIS Pointer	CISP	R	28_H	00000000_H
Subsystem ID/ Subsystem Vendor ID	SSID/SSVID	R	$2C_H$	00000000_H
Expansion ROM Base Address	ERBAD	R/W	30_H	00000000_H
Reserved	RES34	R/W	34_H	00000000_H
Reserved	RES38	R/W	38_H	00000000_H
Maximum Latency/ Minimum Grant/ Interrupt Pin/ Interrupt Line	MAXLAT/ MINGNT/ INTPIN/ INTLIN	R/W	$3C_H$	10020100_H
User Defined Configuration Space Registers				
Reserved	RES40	R/W	40_H	00000000_H
Reserved	RES44	R/W	44_H	00000000_H
Reserved	RES48	R/W	48_H	00000000_H
PCI Configuration Space Reset	PCIRES	R/W	$4C_H$	00000000_H

The only non-standard PCI Configuration Space register is PCIRES, as described below.

5.1.3 PCI Configuration Space - Detailed Register Description

Status/Command register description.

Offset address: 04_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Status															
DPE	SSE	RMA	RTA	0	0	0	DPED	1	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Command															
0	0	0	0	0	0	FBBE	SERRE	0	PER	0	0	SC	BM	MS	IOS

Status and Command register bits

Bit Location	Symbol	Description
31	DPE	**Detected Parity Error** This bit is set by the device whenever it detects a parity error, even if parity error handling is disabled (as controlled by bit 6 in the Command register).
30	SSE	**Signaled System Error** This bit will be set when • the $\overline{\text{SERR}}$ Enable bit is set in the Command register and one of the following events occured: 1. A transaction in which the MUNICH32X acted as a master is terminated with master abort. 2. A transaction in which the MUNICH32X acted as a master is terminated with target abort by the involved target. 3. The transaction has an address parity error and the Parity Error Response bit is set.

Status and Command register bits

29	RMA	**Received Master Abort** This bit is set whenever the MUNICH32X aborts a transaction with master abort. This occurs when no device responds. *Note: In version 1.1 of the MUNICH32X the device does not properly abort the transaction.*
28	RTA	**Received Target Abort** This bit is set whenever a device responds to a master transaction of the MUNICH32X with a target abort. *Note: In version 1.1 of the MUNICH32X the device does not properly abort the transaction.*
27	0_B	Signaled Target Abort The MUNICH32X will never signal “Target Abort”.
26, 25	01_B	DEVSEL Timing The MUNICH32X is a medium device.
24	DPED	**Data Parity Error Detected** This bit is set when the following three conditions are met: 1. the device asserted PERR itself or observed PERR asserted 2. the device setting the bit acted as the bus master for the transaction in which the error occurred 3. and the Parity Error Response bit is set in the Command register
23	1_B	Fast Back-to-Back Capable The MUNICH32X is fast Back-to-Back capable.
22	0_B	UDF Supported No UDFs are supported by the MUNICH32X
21	0_B	66 MHz Capable The MUNICH32X is not 66 MHz capable
20 … 16	00000_B	Reserved
15 … 10	000000_B	Reserved
9	FBBE	**Fast Back-to-Back enable** A value of ‘1’ means the MUNICH32X is allowed to generate fast Back-to-Back transactions to different agents. A value of ‘0’ means the MUNICH32X is only allowed to generate fast Back-to-Back transaction to the same agent.
8	SERRE	**$\overline{\text{SERR}}$ Enable** A value of ‘1’ enables the $\overline{\text{SERR}}$ driver. A value of ‘0’ disables the $\overline{\text{SERR}}$ driver.

Status and Command register bits

7	0_B	Wait Cycle Control The MUNICH32X does never perform address/data stepping.
6	PER	**Parity Error Response** When this bit is set the MUNICH32X will take its normal action when a parity error is detected. When this bit is ‘0’ the MUNICH32X ignores any parity errors that it detects and continues normal operation.
5	0_B	VGA Palette Snoop The MUNICH32X is no VGA-Device.
4	0_B	Memory Write and Invalidate Enable The “Invalidate” command is not supported by the MUNICH32X.
3	SC	**Special Cycles** All special cycles are ignored. *Note: Although this bit can be set it has no effect.*
2	BM	**Bus Master** A value of ‘1’ enables the bus master capability. *Note: Before giving the first action request it is necessary to set this bit.*
1	MS	**Memory Space** A value of ‘1’ allows the MUNICH32X to respond to Memory Space Addresses. *Note: This bit must be set before the first read/write transactions to the MUNICH32X will be started.*
0	IOS	**IO Space** I/O Space accesses to the MUNICH32X are not supported. *Note: Although this bit can be set it has no effect.*

PCI Configuration Space Reset Register

Access : read/write

Offset Address : $4C_H$

Reset Value : 00000000_H

	31													16
PCIRES	0	0	0	0	0	0	0	0	0	0	0	0	RST(2:0)	0

	15	0
PCIRES	0000_H	

RST0 **Serial PCM Core Reset**

Setting this bit to ‘1’ has the same effect for the Serial PCM core as an external $\overline{RST}$; i.e., the PCM core is forced to go into standby mode.

Programming this bit to ‘0’ in turn corresponds to deasserting $\overline{RST}$ (refer to **Chapter 10.1**).

RST1 **LBI Reset**

Setting this bit to ‘1’ has the same effect for the LBI functional blocks, as an external $\overline{RST}$; i.e., the LBI EBC/DMSM/Mailbox modules are forced to go into reset mode and the DMAC is forced to go into standby mode.

Programming this bit to ‘0’ in turn corresponds to deasserting $\overline{RST}$ (refer to **Chapter 10.1**).

RST2 **SSC/IOM®-2 Reset**

Setting this bit to ‘1’ has the same effect for the SSC and IOM®-2 functions as an external $\overline{RST}$; i.e., both are forced to go into standby mode.

Programming this bit to ‘0’ in turn corresponds to deasserting $\overline{RST}$ (refer to **Chapter 10.1**).

5.2 De-multiplexed Bus Interface

The MUNICH32X may be configured for a 33 MHz/32-bit De-multiplexed bus for connection to systems with de-multiplexed processors such as the i960Hx or MC68EC0x0. The DEMUX input pin is used to select the desired configuration (‘0’ = PCI, ‘1’ = De-multiplexed mode).

The De-multiplexed bus interface is a synchronous interface very similar to the PCI bus with the following exceptions:

1. The W/$\overline{R}$ input/output signal replaces the function of the PCI command nibble of the C/$\overline{BE}$(3:0) bit field.
2. Note that in De-multiplexed mode, as in PCI mode, the MUNICH32X provides only the first address of a Master burst read or write transaction.

Table 13
Non-PCI Pins in the De-multiplexed Bus Configuration

Pins	Symbol	Input (I) Output (O)	Function
dedicated	DEMUX	I	**De-multiplexed Bus Enable** ‘0’ = PCI, ‘1’ = DEMUX
LBI address & data	A(31:2)	I/O	**Address Bus**
dedicated	W/$\overline{R}$	I/O	**Write/Read**

In this mode, 4-DWORD Master Read and Write Burst capability may be enabled via the DBE bit field in the Configuration register **CONF**; this bit is valid only if DEMUX = ‘1’.

Burst Capability:
DBE = 0: No burst capability, all transactions are 1 DWORD transfers.
DBE = 1: Burst capability up to 4 DWORDs, currently supported: read/write descriptors: up to 3 DWORDs, read/write data for MUNICH32 core: 1 DWORD

Even when burst capability has been selected, the target can request the MUNICH32X to stop the current transaction by asserting the $\overline{STOP}$ signal.

The following diagrams illustrate the timing waveforms for both single and burst transactions.

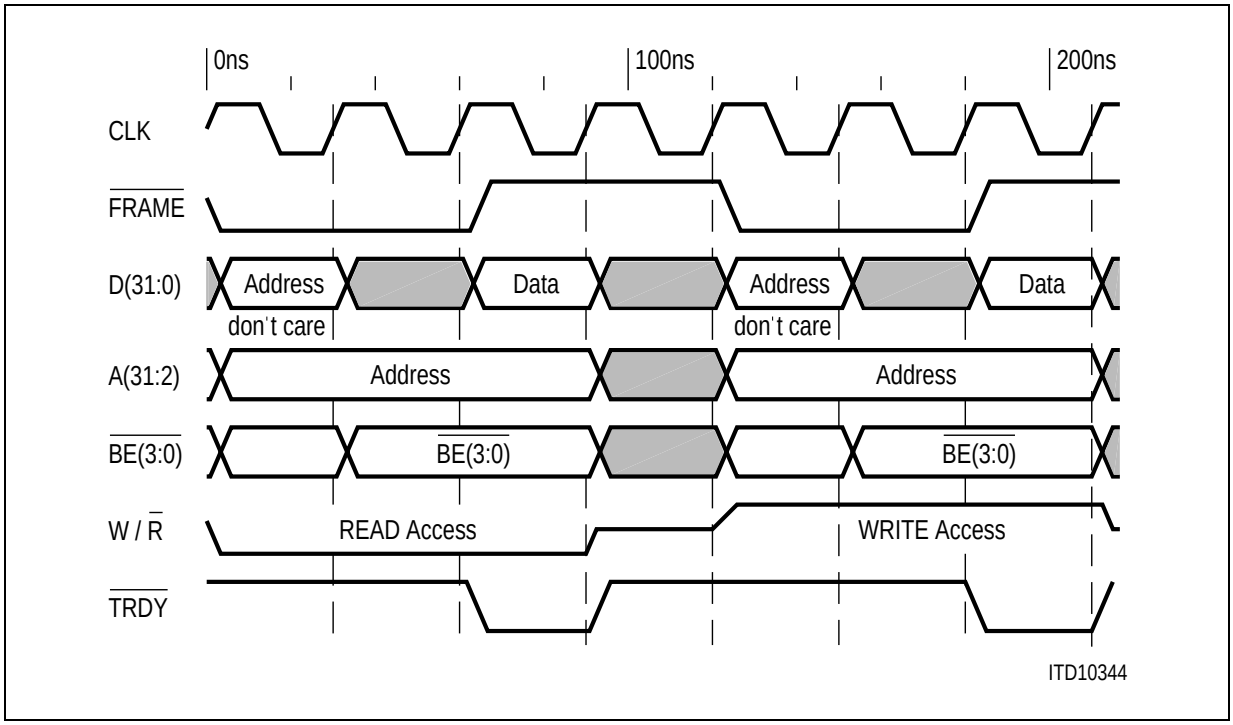


Figure 48
Master Single READ Transaction followed by a Master Single WRITE Transaction in De-multiplexed Bus Configuration

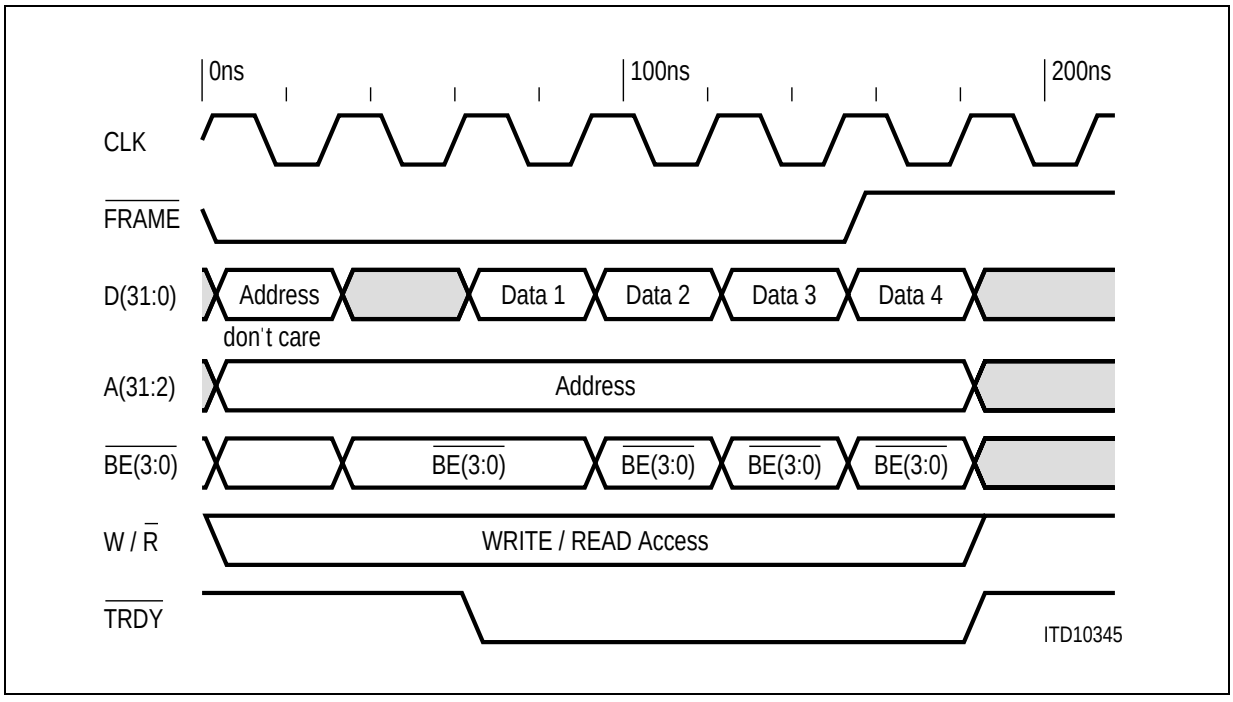


Figure 49
Master Burst WRITE/READ Access in De-multiplexed Bus Configuration

When in De-multiplexed configuration, the MUNICH32X adheres to the PCI bus protocol and timing specification, except for the address and command handling. In this mode, the addresses are provided on a separate address bus A(31:2) to eliminate the need for external de-multiplexing buffers. The address lines A(31:2) correspond to the address lines AD(31:2). The address becomes valid with the falling edge of $\overline{\text{FRAME}}$ and stays valid for the standard PCI address phase, the turn-around cycle and the entire data phase. The burst order, normally coded in AD(1:0), is not supported in De-multiplexed PCI mode. In burst mode the addresses have to be incremented externally for each single transfer.

Moreover, in De-multiplexed PCI mode the command signals are not used. Instead of the command signals a separate pin W/$\overline{\text{R}}$ (I/O) provides the Write/Read strobe signal. The Write/Read becomes valid with the falling edge of $\overline{\text{FRAME}}$ and stays valid for the standard PCI address phase, the turn-around cycle and the entire data phase.

The following four commands are supported:

Table 14
Supported Commands in De-multiplexed Mode

W/$\overline{\text{R}}$	IDSEL	Master Mode	Slave Mode
0	0	memory read	MUNICH32X register read
1	0	memory write	MUNICH32X register write
0	1	not supported	MUNICH32X PCI Configuration read
1	1	not supported	MUNICH32X PCI Configuration write

Note: When designing a de-multiplexed system with the MUNICH32X in De-multiplexed Bus mode it is the responsibility of the glue logic to meet the bus timing/protocol of the PCI specification and of the memory devices that are used in the system. When the MUNICH32X operates in master mode, the bus cycle, for example, can be delayed by the $\overline{\text{TRDY}}$ signal.

6 Local Bus Interface (LBI)

6.1 Overview

The MUNICH32X provides capability for the PCI host system to access LBI peripherals, as well as capability for an intelligent LBI peripheral (e.g., a CPU) to access the PCI host system.

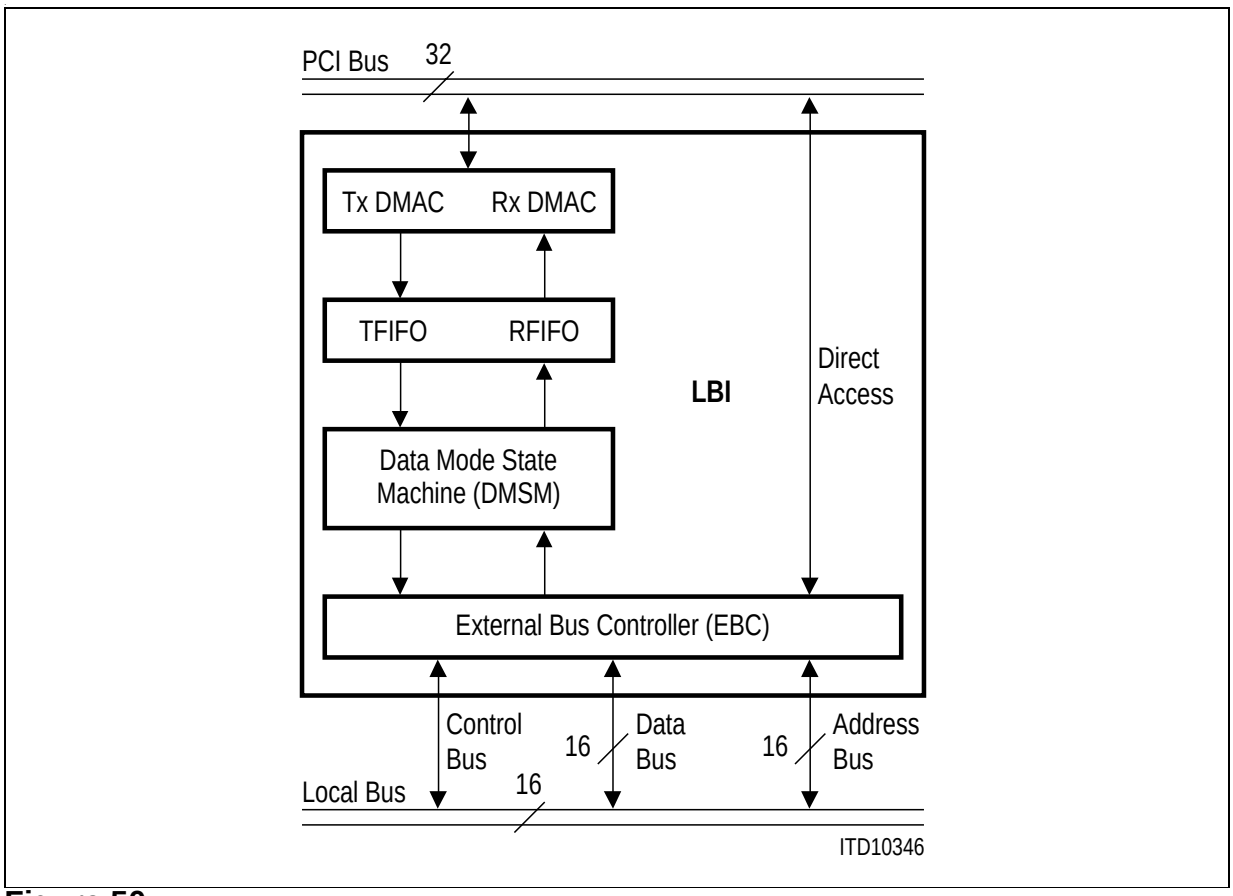


Figure 50
Local Bus Interface Block Structure

Note that the LBI is only available when the MUNICH32X is configured for the PCI mode. When in de-multiplexed mode, the LBI address and data pins interface to the system address bus.

Table 15
LBI Peripheral Transaction Options

Peripheral Type	PCI Transaction	Read	Write
Non-Intelligent	Slave	PCI Retry operation	PCI posted operation
Intelligent	Slave	via Mailbox registers	via Mailbox registers

6.1.1 Transactions with Non-intelligent Peripherals

Standard PCI Slave transactions are used when the PCI host system communicates with non-intelligent LBI peripherals.

For reads, a PCI Retry sequence of operations is performed, in which the MUNICH32X will immediately terminate the PCI transaction (and request a retry) until it terminates the transaction to the LBI. The MUNICH32X uses the retry procedure because the time to complete the data phase will require more than the maximum allowed 16 PCI clocks (from the assertion of $\overline{\text{FRAME}}$ to the completion of the first data phase). Data transfer will be successfully completed within a PCI retry cycle. The number of necessary PCI retry cycles depend on PCI arbitration behavior and the time it needs to terminate the transaction on the local bus; PCI $\overline{\text{TRDY}}$ wait states will not be added for the sequential retry read cycles unless the LBI arbitration time is excessive.

For write transactions, the MUNICH32X will store a single data DWORD and then immediately terminate the PCI transaction successfully. It will then arbitrate the local bus and perform the write transaction after being granted depending on the selected number of wait states and $\overline{\text{LREADY}}$ bus control signal.

Thus write accesses to LBI are performed as ‘posted write’ transactions from the PCI view. A consecutive write transaction results in PCI retry cycles in the case that the preceding write transaction is not yet finished on LBI.

Note that the MUNICH32X performs single word PCI Slave read or write transactions only; Slave burst transactions to LBI are not supported.

6.1.2 Transactions with Intelligent Peripherals

The MUNICH32X uses an ‘exclusive-access’ Mailbox Command Register **MBCMD** to control the transfer of information between the PCI host system and an intelligent LBI peripheral (e.g., a CPU). The PCI host system always reads the contents that was written to Mailbox Command Register by the LBI peripheral, while the intelligent LBI peripheral always reads the contents that was written to Mailbox Command Register by the PCI host system.

As an **example**, consider when the PCI host system wants to transfer data to an intelligent LBI peripheral. First, assuming it has ‘ownership’ of the Mailbox registers, it loads data into the Mailbox Data Registers, and then writes a ‘1’ to INPCI bit field of the Mailbox Command Register. This last action causes the LINTO output signal to become asserted, indicating to the intelligent LBI peripheral that data is ready.

The intelligent LBI peripheral will read Mailbox Command Register (which deasserts the LINTO output signal and resets the INPCI bit field of Mailbox Command Register), and then reads the data from the Mailbox Data Registers. Finally, it writes a ‘1’ to the INLBI bit field of Mailbox Command Register, which causes an interrupt to be generated to the PCI host system, informing the PCI host system that the data transfer is complete.

The PCI host system completes its participation of the transaction by reading the Status Register **STAT** (to determine the cause of the interrupt), writing a ‘1’ to the Status Acknowledge Register’s MBI bit field to deassert the PCI $\overline{\text{INTA}}$ signal.

Alternately, consider when the intelligent LBI peripheral wants to transfer data to the PCI host system. First, assuming it has ‘ownership’ of the Mailbox registers, it loads data into the Mailbox Data Registers, and then writes a ‘1’ to the INLBI bit field of Mailbox Command Register. This causes an interrupt to be generated to the PCI host system, indicating to the PCI host system that data is ready.

The PCI host system reads the Status Register **STAT** (to determine the cause of the interrupt), writes a ‘1’ to the Status Acknowledge Register’s MBI bit field to deassert the PCI $\overline{\text{INTA}}$ signal, and then reads the data from the Mailbox Data Registers. Next, it writes a ‘1’ to the INPCI bit field of the Mailbox Command Register, which asserts the LINTO signal to the LBI peripheral.

The intelligent LBI peripheral completes its participation of the transaction when it reads Mailbox Command Register, which deasserts the LINTO signal and resets the INLBI bit field of the Mailbox Command Register.

6.1.3 Software Arbiter/Data Transfer Control

The architecture of the Mailbox registers requires the PCI host system software to provide Mailbox arbitration. The primary data transfer control requirement is that only the current ‘owner’ of the Mailbox registers may write data into the Mailbox Data Registers.

Typically, upon exiting reset, the PCI host system becomes the Mailbox ‘owner’ and may transfer data to the LBI. If the LBI desires to transfer data to the PCI host system, it must generate an interrupt to the PCI host system (by writing a ‘1’ to INLBI bit field in Mailbox Command Register **MBCMD**), informing the PCI host system that it requests ‘ownership’ of the Mailbox registers. It is the responsibility of the PCI host system software arbiter to handle the request/grant protocol.

6.1.4 Mailbox Registers

The organization of the Mailbox registers is partioned into an ‘exclusive-access’ Mailbox Command Register, and into seven Mailbox Data Registers, as shown below.

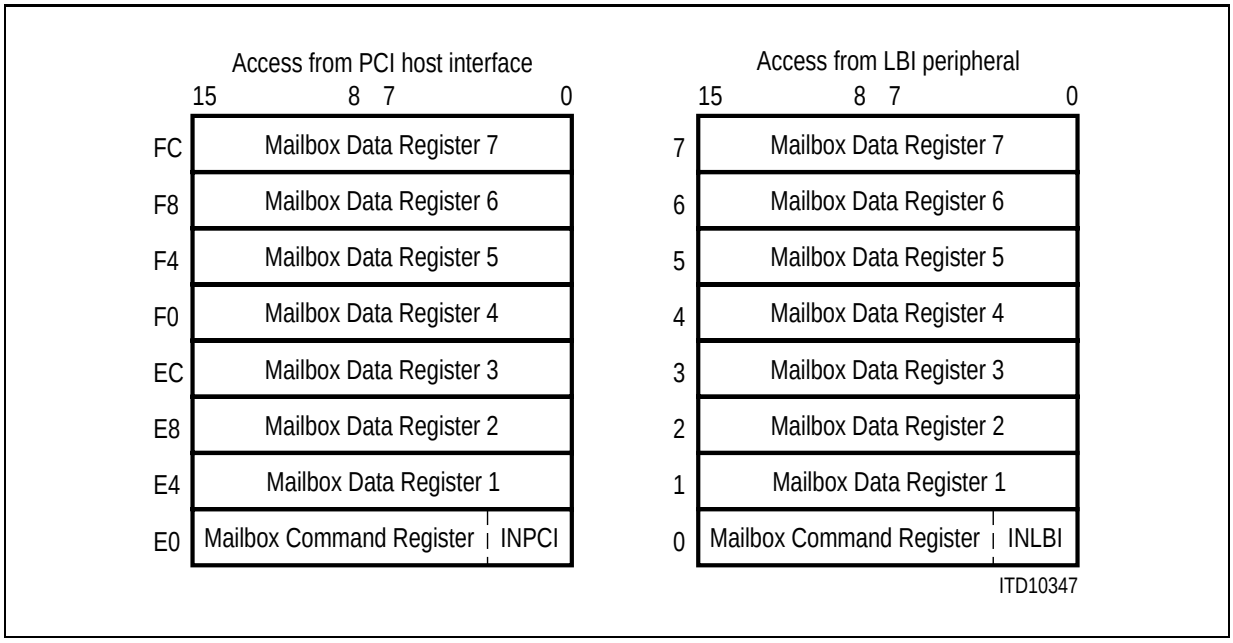


Figure 51
LBI Mailbox Structure

Note: The Mailbox registers should only be used for communication between the PCI host system and an intelligent LBI peripheral.

The Mailbox Command Register provides the PCI/LBI Mailbox registers exclusive access bit INPCI/INLBI and 15 bits for user defined interrupt information (refer to **Section 11.2.7**). It may for example perform the following functions:

- interrupt generation,
- deassertion of the LINTO interrupt signal by the intelligent LBI peripheral,
- end-of-data-transfer indication, and
- end-of-transaction indication.

Note that an intelligent LBI peripheral will deassert the LINTO interrupt signal by reading Mailbox Command Register **MBCMD**, while the PCI host system will deassert the PCI $\overline{\text{INTA}}$ interrupt signal by writing a ‘1’ to the MBI bit field in Status Acknowledge Register **STACK**.

When a mailbox interrupt from LBI peripheral is detected, the LBI Mailbox Interrupt Vector is generated and written to the host memory address specified in Peripheral Interrupt Queue Base Address (**PIQBA**) register.
Note that an interrupt vector is generated after a write access to bit field INPCI/ INLBI of **MBCMD**, even when this bit has not been reset between two write accesses.

The structure of the LBI Mailbox interrupt vector is as follows:

LBI Mailbox Interrupt Vector

	31								16
MB_IV	1	0	1	1	0	0	0	0	00_H

	15	1
MB_IV	IV(15:1)	1

IV **Interrupt Vector**

Contains the values of Mailbox Command Register's bit fields MBINT(15:1).

LBI Block Overview

Additionally to the Mailbox registers, the Local Bus Interface consists of three main functional blocks (refer to **Figure 50**):

- the External Bus Controller (EBC),
- the Data Mode State Machine (DMSM), and
- the DMA Controllers (DMAC).

They are described in detail in the following chapters.

6.2 LBI External Bus Controller (EBC)

The External Bus Controller (EBC) provides a flexible bus interface to connect a wide range of peripherals. In normal mode, this interface is master and drives peripheral devices. It provides the ability to select busses of different configuration: 8 bit multiplexed/de-multiplexed or 16 bit multiplexed/de-multiplexed. The configurable pins of DMA support/General Purpose Bus provide alternate functionality to support the LBI pins.

The EBC performs 'funneling' of data to or from the LBI FIFOs (as DWORDs) to the 8-/16-bit LBI bus. The EBC also supports bus arbitration. It inter-works with all other blocks of the LBI (FIFOs, DMSM and Mailbox registers), as well as supporting a 'Direct Access' path to the internal bus. It also provides the de-multiplexed address lines on the LBI address pins, if the MUNICH32X is operated in de-multiplexed mode.

The function of the EBC is controlled via the LBI Configuration register **LCONF**. It specifies the external bus cycles in terms of address (multiplexed/de-multiplexed), data (16-bit/8-bit) and control signal length (wait states).

6.2.1 External Bus Modes

Multiplexed Bus Modes

In the 16-bit multiplexed bus mode both the address and data lines use the pins LD(15:0). The address is time-multiplexed with the data and has to be latched externally. The width of the required latch depends on the selected data bus width, i.e. an 8-bit data bus requires a byte latch (the address bits LD15 … LD8 on the LBI port do not change, while on LD7 … LD0 address and data are multiplexed), a 16-bit data bus requires a word latch (the least significant address line LA0 is not relevant for word accesses).
In de-multiplexed mode, the address lines are permanently output on pins LA(15:0) and do not require latches.

The EBC initiates an external access by generating the Address Latch Enable signal (LALE) and then placing an address on the bus. The falling edge of LALE triggers an external latch to capture the address. After a period of time during which the address must have been latched externally, the address is removed from the bus. The EBC now activates the respective command signal ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LRDY}}$). Data is driven onto the bus either by the EBC (for write cycles) or by the external memory/peripheral (for read cycles). After a period of time, which is determined by the access time of the memory/peripheral, data become valid.

Read cycles: Input data is latched and the command signal is now deactivated. This causes the accessed device to remove its data from the bus which is then tri-stated again.

Write cycles: The command signal is now deactivated. The data remain valid on the bus until the next external bus cycle is started.

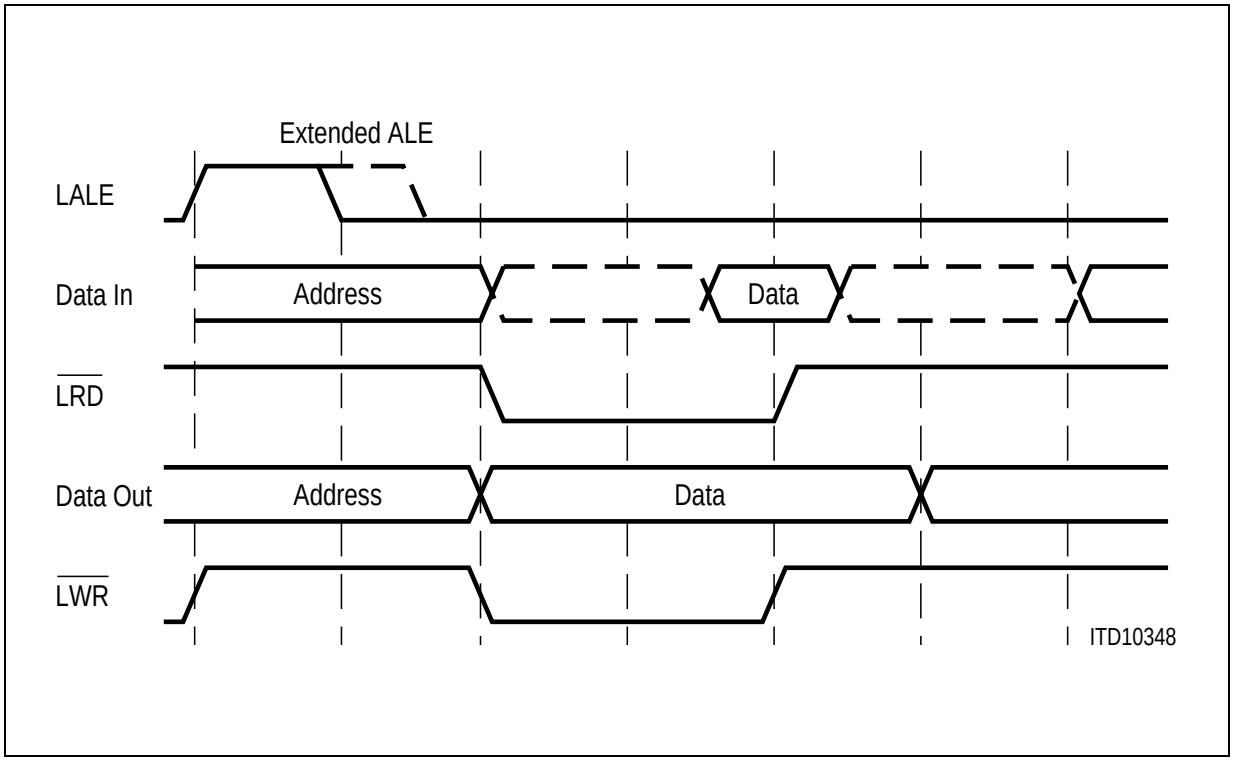


Figure 52
Multiplexed Bus Cycle

Demultiplexed Bus Modes

The de-multiplexed bus modes use the LBI port pins LA(15:0) for the 16-bit address and the LBI port pins LD(15:0) for 8/16-bit data. The EBC initiates an external access by placing an address on the address bus. The EBC then activates the respective command signal ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LBHE}}$). Data is driven onto the data bus either by the EBC (for write cycles) or by the external memory/peripheral (for read cycles). After a period of time, which is determined by the access time of the memory/peripheral, data become valid.

Read Cycles: Input data is latched and the command signal is now deactivated. This causes the accessed device to remove its data from the data bus which is then tri-stated again.

Write Cycles: The command signal is now deactivated. If a subsequent external bus cycle is required, the EBC places the respective address on the address bus. The data remain valid on the bus until the next external bus cycle is started.

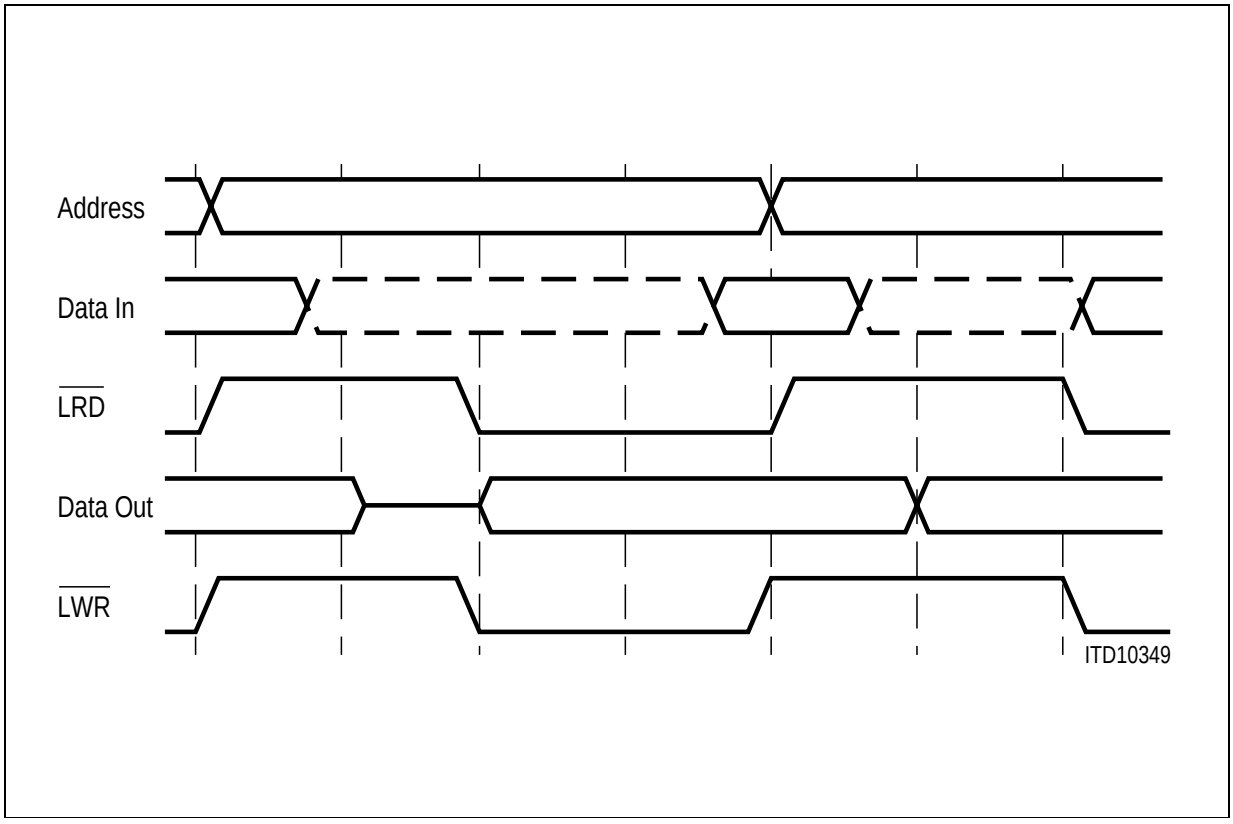


Figure 53
De-multiplexed Bus Cycle

External Data Bus Width

The EBC can operate on 8-bit or 16-bit wide external memory/peripherals. A 16-bit data bus uses the LBI port pins LD(15:0), while an 8-bit data bus only uses LD(7:0). This saves bus transceivers, bus routing and memory cost at the expense of transfer time. The EBC can control byte accesses on a 16-bit data bus.

Byte accesses on a 16-bit data bus require that the upper and lower half of the memory can be accessed individually. In this case the upper byte is selected with the $\overline{\text{LBHE}}$ signal, while the lower byte is selected with the AD0 signal. The two bytes of the memory can therefore be enabled independently from each other (or together when accessing words).
Devices such as the ESCC2 also provide a $\overline{\text{BHE}}$ input and hence allow byte accesses in 16-bit bus mode.

When reading bytes from an external 16-bit device, 16-bit words may be read and the EBC automatically selects the byte to be input and discards the other. However, care must be taken when reading devices that change their state when being read, like FIFOs, interrupt status registers, etc. In this case individual bytes should be selected using $\overline{\text{BHE}}$ and A0.

Switching between the Bus Modes

The EBC bus type can be switched dynamically by software. However, the user needs to keep track of the peripheral that is being addressed (multiplexed mode or de-multiplexed mode) with the selected bus type.
Master/Slave bus mode is also configured/arbitrated dynamically by the device itself.

6.2.2 Programmable Bus Characteristics

Important timing characteristics of the external bus interface are user programmable to adapt it to a wide range of different external bus and memory configurations with different types of memories and/or peripherals.

The following parameters of an external bus cycle are programmable:

- **Memory Cycle Time** (extendable with 1 … 15 wait states) defines the allowable access time.
- **READY Control** defines, if a bus cycle is terminated internally or externally.

Programmable Memory Cycle Time

The user can adjust the EBC external bus cycles to the access time of the respective memory or peripheral. This access time is the total time required to move the data to the destination. It represents the period of time during which the EBC's signals do not change.

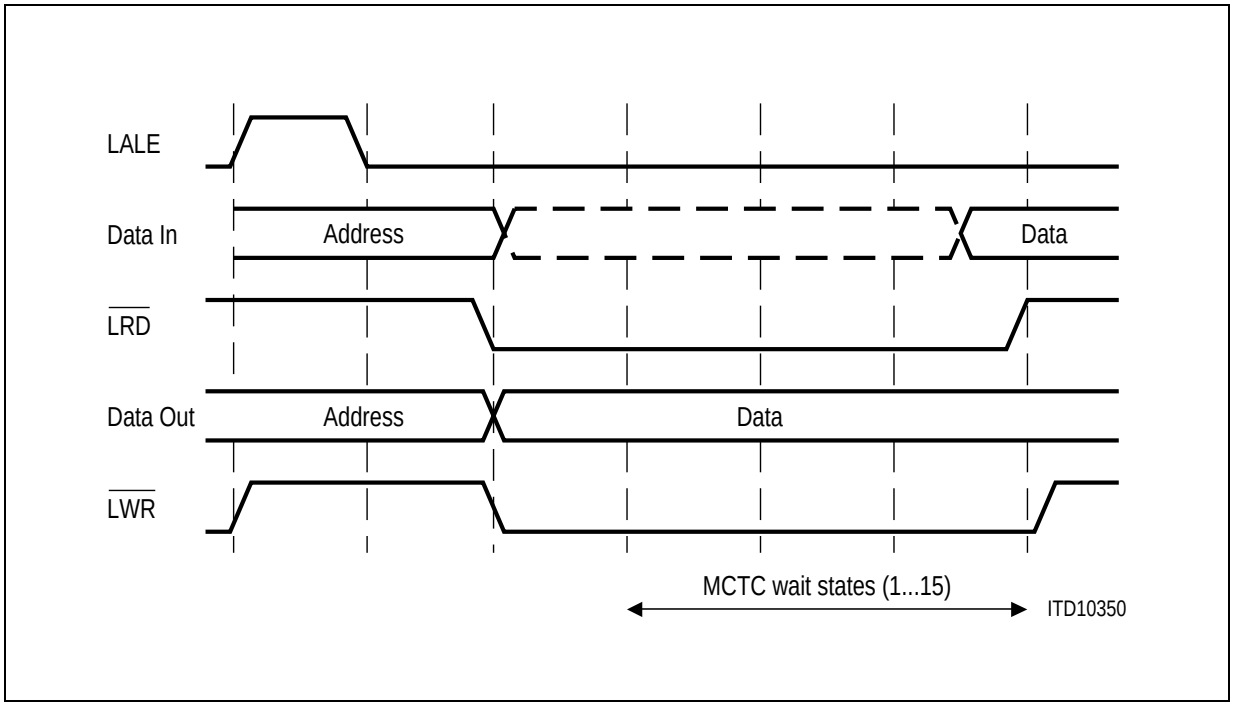


Figure 54
Memory Cycle Time

The external bus cycles of the EBC can be extended for a memory or peripheral, which cannot keep pace with the EBC's maximum speed, by introducing wait states during the access (see figure above).

The memory cycle time wait states can be programmed in increments of one EBC system clock (LCLKOUT) within a range from 0 … 15 (default after reset) via the MCTC bit fields of the LBI Configuration register **LCONF**. A number of (15 - <MCTC>) wait states will be inserted.

6.2.3 $\overline{\text{LRDY}}$ Controlled Bus Cycles

For situations, where the programmable wait states are not sufficient, or where the response (access) time of a peripheral is not constant, the MUNICH32X EBC interface provides external bus cycles that are terminated via a $\overline{\text{LRDY}}$ input signal. In this case the EBC first inserts a programmable number of waitstates (0 … 7) and then monitors the $\overline{\text{LRDY}}$ line to determine the actual end of the current bus cycle. The external device drives $\overline{\text{LRDY}}$ low in order to indicate that data either have been latched (write cycle) or are available (read cycle).

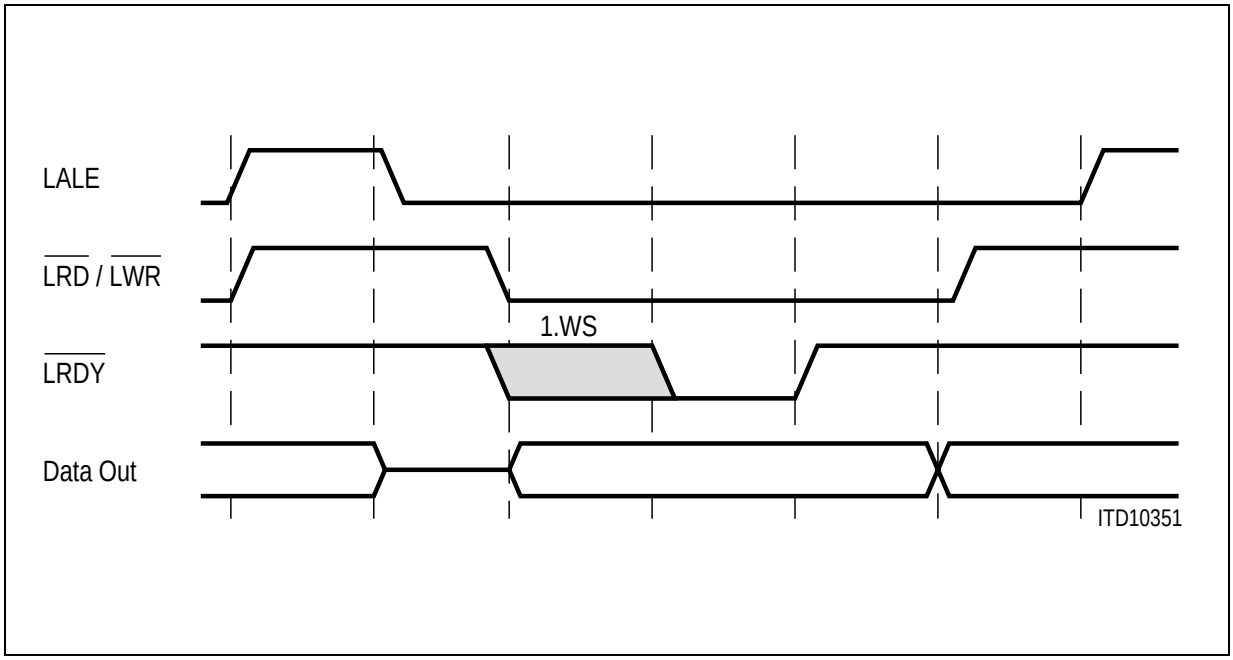


Figure 55
$\overline{\text{LRDY}}$ Controlled Bus Cycles

The $\overline{\text{LRDY}}$ function is enabled via the RDEN bit fields in the LBI Configuration register. When this function is selected (RDEN = '1'), only the lower 3 bits of the respective MCTC bit field define the number of inserted waitstates (0 … 7), while the MSB of bit field MCTC selects the $\overline{\text{LRDY}}$ operation.

The $\overline{\text{LRDY}}$ signal is always synchronized at the input port pin. An asynchronous $\overline{\text{LRDY}}$ signal that has been activated by an external device may be deactivated in response to the trailing (rising) edge of the respective command ($\overline{\text{LRD}}$ or $\overline{\text{LWR}}$).

Combining the $\overline{\text{LRDY}}$ function with predefined waitstates is advantageous in two cases. Memory components with a fixed access time and peripherals operating with $\overline{\text{LRDY}}$ may be grouped into the same address window. The (external) wait states control logic in this case would activate $\overline{\text{LRDY}}$ either upon the memory's chip select or with the peripheral's $\overline{\text{LRDY}}$ output. After the predefined number of wait states the EBC will check its $\overline{\text{LRDY}}$ line to determine the end of the bus cycle. For a memory access it will be low already, for a peripheral access it may be delayed. As memories tend to be faster than peripherals, there should be no impact on system performance.

When using the $\overline{\text{LRDY}}$ function with 'normally-ready' peripherals, it may lead to erroneous bus cycles, if the $\overline{\text{LRDY}}$ line is sampled too early. These peripherals pull their $\overline{\text{LRDY}}$ output low, while they are idle. When they are accessed, they deactivate $\overline{\text{LRDY}}$ until the bus cycle is complete, then drive it low again. By inserting predefined wait states, the first $\overline{\text{LRDY}}$ sample point can be shifted to a time by that the peripheral has safely controlled the $\overline{\text{LRDY}}$ line (e.g., after 2 wait states in the figure above).

6.2.4 Configuring the External Bus Controller

The properties of a bus cycle usage of $\overline{\text{LRDY}}$, external bus mode and wait states are controlled by LBI Configuration register **LCONF**. This allows the use of memory components or peripherals with different interfaces within the same system, while optimizing accesses to each of them.

The current interupt signal and bus arbitration status of the EBC is indicated by the LBI Status Register **LSTAT**:

- **LSTAT**.HLD indicates the hold mode of the EBC.
- **LSTAT**.INT1 indicates an interrupt on LINT1.
- **LSTAT**.INT2 indicates an interrupt on LINT2.

The reset control of the EBC is handled by the LBI Configuration Register:

- **LCONF**.$\overline{\text{EBCRES}}$ resets the EBC in an initial state (same as hardware reset state). For normal EBC operation bit $\overline{\text{LCONF.EBCRES}}$ must be set to '1' again.

6.2.5 EBC Idle State

Upon reset, the LBI is in bus slave mode with control strobes as inputs. The EBC can then be programmed to be master or slave by software.

When the EBC bus interface is enabled in arbitration master mode, but no external access is currently executed, the EBC is idle. During this idle state the external interface behaves in the following way:

- The data port LD(15:0) is in high impedance state (floating).
- The address port LA(15:0) drives the address used last.
- $\overline{\text{LRD}}$/$\overline{\text{LWR}}$ remain inactive (High).

6.2.6 External Bus Arbitration

In high performance systems it may be efficient to share external resources like memory banks or peripheral devices among more than one bus controller. The LBI's EBC block supports this approach with the possibility to arbitrate the access to its external bus, i.e. to the external devices.

This bus arbitration allows an external master to request the EBC's bus via the $\overline{\text{LHOLD}}$ input. The EBC acknowledges this request via the $\overline{\text{LHLDA}}$ output and will float its bus lines in this case. The new master may now access the peripheral devices or memory banks via the same interface lines as the EBC. During this time the MUNICH32X can continue executing internal processes, as long as it does not need access to the external bus.

When the EBC needs access to its external bus while it is occupied by another bus master, the bus is requested via the $\overline{\text{LBREQ}}$ output.

The external bus arbitration is enabled by setting bit HLDEN in the LBI Configuration register to '1'. This bit may be cleared during the execution of program sequences, where the external resources are required, but cannot be shared with other bus masters. In this case the EBC will not answer to $\overline{\text{LHOLD}}$ requests from other external masters.

Note: The pins $\overline{\text{LHOLD}}$, $\overline{\text{LHLDA}}$ and $\overline{\text{LBREQ}}$ maintain their functionality (bus arbitration) even after the arbitration function has been switched off by clearing HLDEN. All three pins are used for bus arbitration after bit HLDEN was set once.

Entering the Hold State

Access to the EBC's external bus is requested by driving its $\overline{\text{LHOLD}}$ input low. After synchronizing this signal the EBC will complete a current external bus cycle (if any is active), release the external bus and grant access to it by driving the $\overline{\text{LHLDA}}$ output low. During hold state the EBC manages the external bus interface as follows:

- Address and data bus(es) float to tri-state.
- Command lines become inputs ($\overline{\text{LRD}}$, $\overline{\text{LWR}}$, $\overline{\text{LBHE}}$).

Should the MUNICH32X require access to its external bus during hold mode, it activates its bus request output $\overline{\text{LBREQ}}$ to notify the arbitration circuitry. $\overline{\text{LHOLD}}$ is activated only during hold mode. It will be inactive during normal operation.

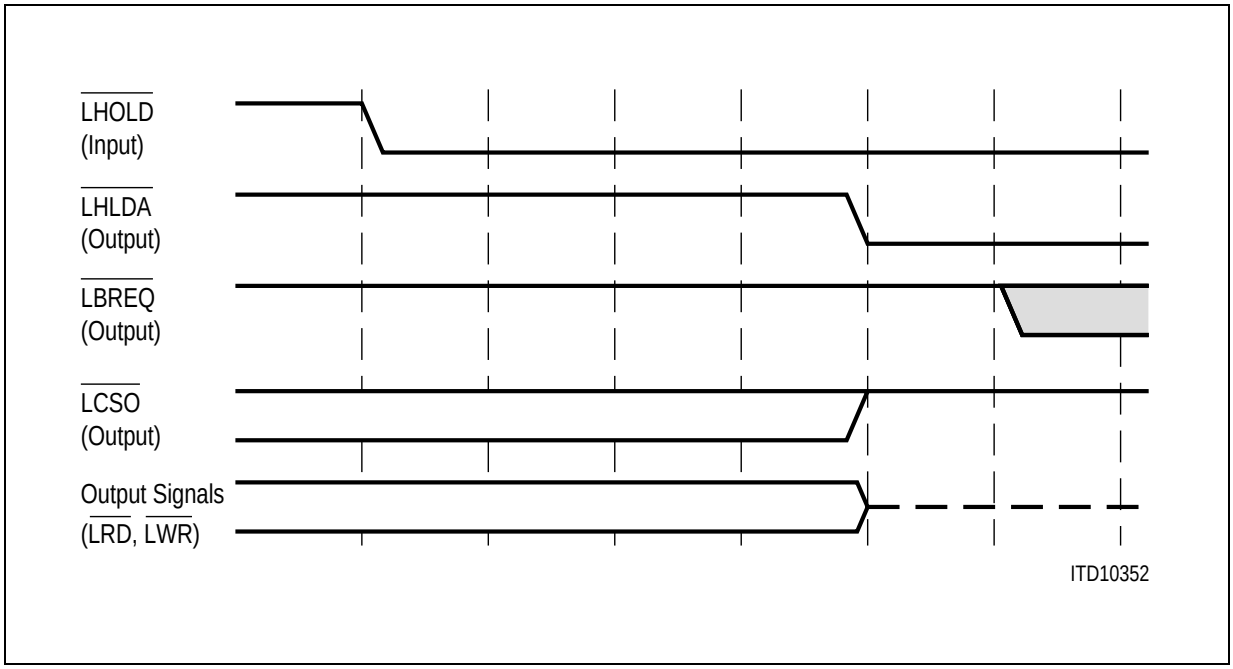


Figure 56
External Bus Arbitration (Releasing the Bus)

Note: The MUNICH32X will complete the currently running bus cycle before granting bus access as indicated by the dotted lines. This may delay hold acknowledge compared to this figure.
The figure above shows the first possibility for $\overline{\text{LBREQ}}$ to become active.

Exiting the Hold State

The external bus master returns the access rights to the MUNICH32X EBC by driving the $\overline{\text{LHOLD}}$ input high. After synchronizing this signal the EBC will drive the $\overline{\text{LHLDA}}$ output high, actively drive the control signals and resume executing external bus cycles if required.

Depending on the arbitration logic, the external bus can be returned to the EBC under two circumstances:

- The external master does not require access to the shared resources and gives up its own access rights, or
- The MUNICH32X EBC needs access to the shared resources and demands this by activating its $\overline{\text{LBREQ}}$ output. The arbitration logic may then deactivate the other master's $\overline{\text{LHLDA}}$ and hence free the external bus for the EBC, depending on the priority of the different masters.

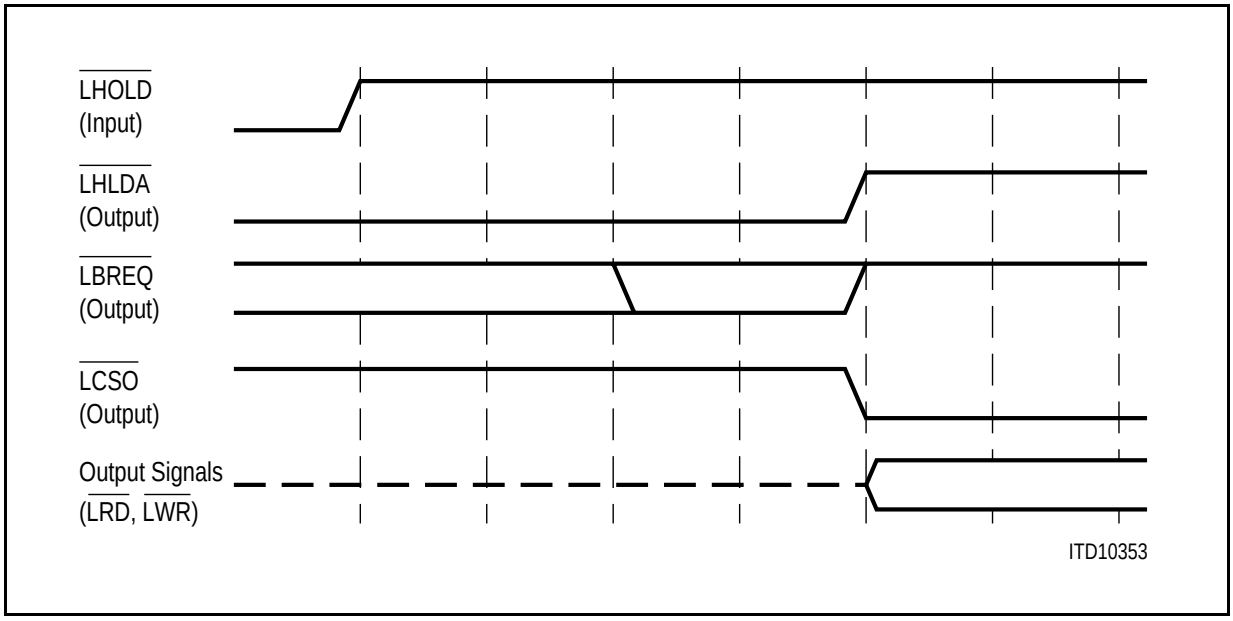


Figure 57
External Bus Arbitration (Regaining the Bus)

The falling LBREQ edge marks the last moment for LBREQ to trigger the indicated regain-sequence. Even if LBREQ is activated earlier the regain-sequence is initiated by LHOLD going high. LBREQ and LHOLD are connected via an external arbitration circuitry.
Note that LHOLD may also be deactivated without the EBC requesting the bus.

6.2.7 LBI Bus Arbitration

This section covers the LBI initialization when the MUNICH32X is operating in local bus Master mode or Slave mode, and the operation of bus mode transfers from hold to active state (and vice versa).

Note that **bus master mode** means that the device is **driving** the local bus and performing bus cycles, and **bus slave mode** means that the EBC bus is in **HOLD** mode, and that an external controller may read the LBI Mailbox registers.

On start-up, the MUNICH32X could be set to operate in LBI bus arbitration master mode (**LCONF**.ABM = ‘1’) or LBI bus arbitration slave mode (**LCONF**.ABM = ‘0’). The arbitration master mode is chosen if the device needs to output the LHLDA signal.

Note that **LCONF**.HDEN = ‘1’ allows responding to the arbitration signals, whereas **LCONF**.HDEN = ‘0’ causes this device not to give up its bus. This function can also be used by software or hardware during critical cycles.

6.2.7.1 Master/Slave Bus Arbitration

The master normally drives the LBI bus signals such as LALE (only in multiplexed mode), $\overline{\text{LWR}}$ and $\overline{\text{LRD}}$. The slave is defined to input the LBI bus control signals (LALE, $\overline{\text{LWR}}$ and $\overline{\text{LRD}}$). Bus arbitration is required when the slave also needs to access LBI bus peripherals.

For this purpose, the external busses of the master and the slave are directly connected together. However, it must always be assured that at one time only one of them, either the master or the slave, controls all external bus signals, while the other one drives its bus pins into an high-impedance state. This arbitration of the external bus is controlled by the low-level active pins $\overline{\text{LHOLD}}$ (hold request), $\overline{\text{LHLDA}}$ (hold acknowledge), and $\overline{\text{LBREQ}}$ (bus request) of the two devices.

Note that the definition and function of the bus arbitration signals is different in master and slave mode. The following table describes these differences.

Table 16
LBI Bus Arbitration Signals

Pin	Direction	Function in Master Mode
$\overline{\text{LHOLD}}$	Input	While $\overline{\text{LHOLD}}$ is high, the master operates in normal mode. Upon a high-to-low transition, the master issues a hold request. The master backs off the bus, activates $\overline{\text{LHLDA}}$ and goes into hold mode. A low-to-high transition issues the exit from hold mode. The master deactivates $\overline{\text{LHLDA}}$, takes over the bus and enters normal operation again.
$\overline{\text{LHLDA}}$	Output	High during normal operation. When the master enters hold mode, it sets $\overline{\text{LHLDA}}$ to low after releasing the bus. On exit of hold mode, the master first sets $\overline{\text{LHLDA}}$ to high and then goes onto the bus again.
$\overline{\text{LBREQ}}$	Output	High during normal operation. The master activates $\overline{\text{LBREQ}}$ by setting it to low earliest one TCL after activating $\overline{\text{LHLDA}}$ if it has to perform an external bus access. If the master has regained the bus, $\overline{\text{LBREQ}}$ is set to high one TCL after deactivation of $\overline{\text{LHLDA}}$.

6.2.7.2 Initialization of the Master/Slave Bus Arbitration

Figure 58 shows the correct connection of the bus arbitration signals between the master and the slave. In order to provide correct levels during initialization of the master and the slave, two external pull-up devices are required. One is connected to the master's $\overline{\text{LHOLD}}$ input, the other to the slave's $\overline{\text{LHLDA}}$ input.

Note: For compatibility reasons with existing applications, these pull-ups can not be integrated into the chip.

Pin	Direction	Function in Slave Mode
$\overline{\text{LHOLD}}$	Input	While both $\overline{\text{LHOLD}}$ and $\overline{\text{LHLDA}}$ are high, the slave is in hold mode, the bus interface is tristated. When the slave is released out of hold mode ($\overline{\text{LHLDA}}$ = 0) and has completely taken control over the external bus, a low level at this pins requests the slave to go into hold mode again. However, in any case the slave will perform at least one external bus cycle before going into hold mode again.
$\overline{\text{LHLDA}}$	Input	A high-to-low transition at this pin releases the slave from hold mode.
$\overline{\text{LBREQ}}$	Output	This signal is high as long as the slave operates from internal memory. When it detects that an external access is required, it sets $\overline{\text{LBREQ}}$ to low and waits for signal $\overline{\text{LHLDA}}$ to become low. $\overline{\text{LBREQ}}$ will go back to high when the slave has backed off the bus after it was requested to go into hold mode.

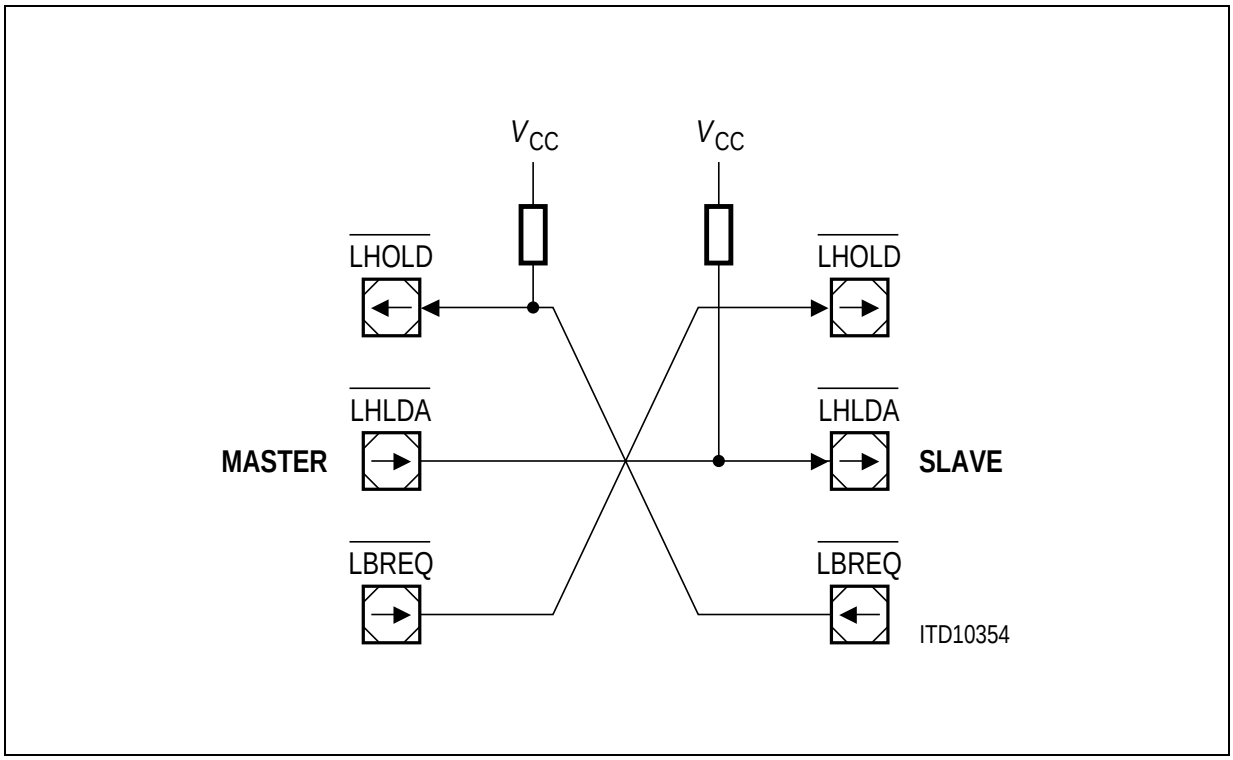


Figure 58
Connection of the Master and Slave Bus Arbitration Signals

Bus Arbitration Master Initialization

After reset, the master is normally starting execution out of external memory. During reset, the default is the arbitration slave mode. The master arbitration mode must first be selected done by setting the **LCONF**.ABM = '1'. During the initialization, the HDEN bit in register **LCONF** must be set. Since the $\overline{\text{LHOLD}}$ pin is held high through the external pull-up, no hold requests can occur, even when the slave has not been initialized yet.

Note that the HDEN bit of the master can be reset during normal operation to force the master to ignore hold requests from the slave until HDEN is set again. However, the pins $\overline{\text{LHOLD}}$, $\overline{\text{LHLDA}}$ and $\overline{\text{LBREQ}}$ are still reserved for the bus arbitration. This is intended to have the option to disable certain critical processes against interruption through hold requests.

Bus Arbitration Slave Initialization

The slave must start using internal resources only after reset. During reset, the default mode is the slave mode. This is also done by programming the **LCONF**.ABM = '0'. This enables the slave mode of the bus arbitration signals. After this, the HDEN bit in register **LCONF** must be set.

Note: 1. After setting the slave's HDEN bit, the $\overline{\text{LBREQ}}$ output of the slave might be activated to low for a period of 2TCL. If the master does not recognize this hold request (it depends on the master's transition detection time slot, whether this short pulse is detected), this pulse has no effect. If the master recognizes this pulse, it might go into hold mode for one cycle.

2. It is recommended to not reset the slave's HDEN bit after initialization.

6.2.7.3 Operation of the Master/Slave Bus Arbitration

The figure below shows the sequence of the bus arbitration signals in a master/slave system. The start-up condition is that the master is in normal mode and operating on the external bus, while the slave is in hold mode, operating from internal memory; the slave's bus interface is tristated. The marked time points in the diagram are explained in detail in the following.

1) The slave detects that it has to perform an external bus access. It activates $\overline{\text{LBREQ}}$ to low, which issues a hold request to the master.

2) The master activates $\overline{\text{LHLDA}}$ after releasing the bus. This initiates the slave's exit from hold sequence.

3a) When the master detects that it also has to perform external bus accesses, it activates $\overline{\text{LBREQ}}$ to low. The earliest time for the master to activate $\overline{\text{LBREQ}}$ is one TCL after the activation of the master's $\overline{\text{LHLDA}}$ signal. However, the slave will ignore this signal until it has completely taken over control of the external bus. In this way, it is assured that the slave will at least perform one complete external bus access.

3b) If the master can operate from internal memory while it is in hold mode, it leaves the $\overline{\text{LBREQ}}$ signal high until it detects that an external bus access has to be performed. The slave therefore can stay on the bus as long as the master does not request the bus again.

4) When the master has requested the bus again through activation of its $\overline{\text{LBREQ}}$ signal, the slave will complete the current access and go into hold mode again. After completely tristateing its bus interface, the slave deactivates its $\overline{\text{LBREQ}}$ signal, thus releasing the master out of hold mode.

5) The master has terminated its hold mode and deactivates its $\overline{\text{LHLDA}}$ signal again. Now the master again controls the external bus again.

6) The master deactivates its $\overline{\text{LBREQ}}$ signal again one TCL after deactivation of $\overline{\text{LHLDA}}$. From now on (and not earlier), the slave can generate a new hold request from the master. With this procedure it is assured that the master can perform at least one complete bus cycle before requested by the slave to go into hold mode again.

Also shown in **Figure 59** is the sequence of the bus control between the master and the slave.

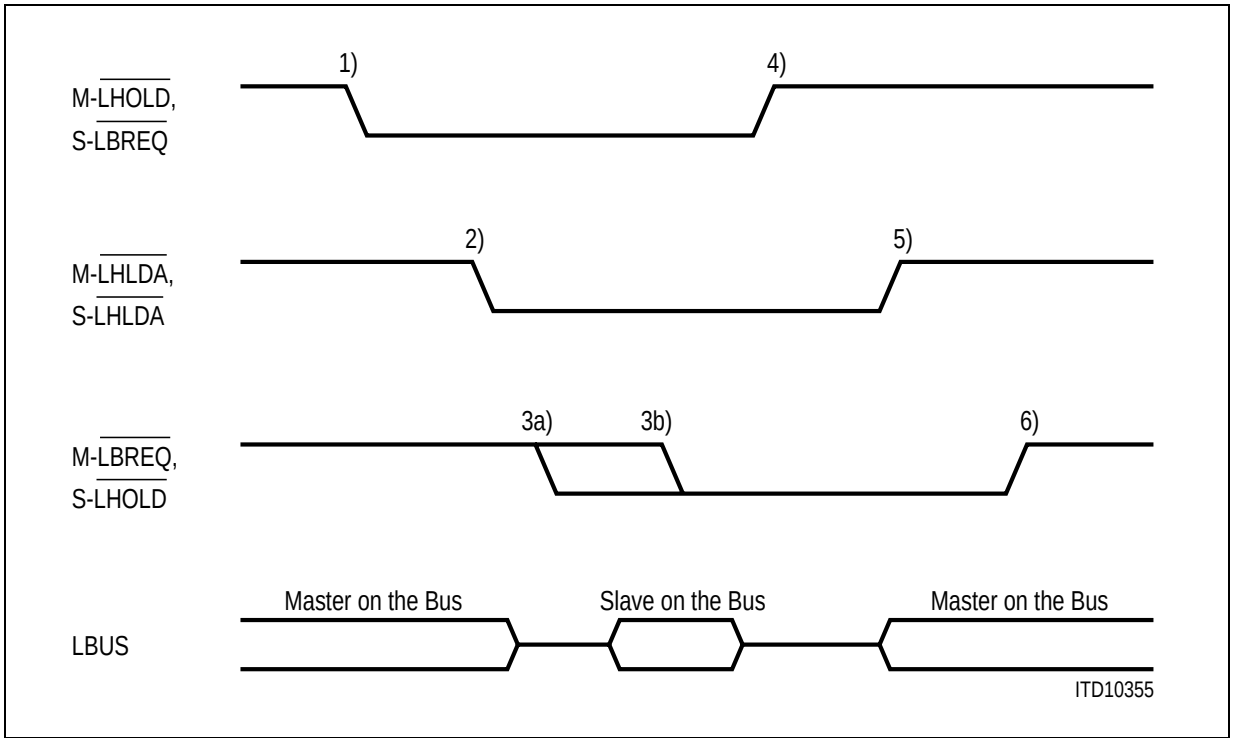


Figure 59
Bus Arbitration Sequence

6.3 LBI Data Mode State Machine (DMSM)

The Data Mode State Machine (DMSM) in the Local Bus Interface will service the FIFOs in specific devices such as the Siemens ESCC2 (SAB 82532), FALC54 (PEB 2254) or HSCX (SAB 82525, SAB 82526). The state machine has user-programmable registers to correctly handshake with the peripheral to transfer data. The DMSM registers are directly accessible from the PCI host side.

In the slave EBC mode, the Mailbox registers are accessible from the local bus side to facilitate communication between the PCI host system and the Local Bus host µC.

The MUNICH32X provides 4 DMA controllers to service two full duplex serial channels on the LBI (e.g., to connect an ESCC2).

The Tx DMACs deliver DWORDs from the memory to the LBI TFIFO, and the Rx DMACs transfer the DWORDs from LBI RFIFO to the host memory. The EBC is responsible for ‘funneling’ the DWORDs to the 8 or 16-bit local bus.

6.3.1 DMSM Function

The Data Mode State Machine (DMSM) services packet data from peripheral FIFOs and transfers them to the host memory via the DMACs.

The DMSM assists in transferring data from peripheral devices based on the Siemens HDLC controller family (HSCX, ESCC2, FALC54).

The procedure makes it easy for the software to transmit packets queued in the shared memory. Similarly, received packets are stored conveniently in the shared memory.

The data transfers via the LBI interface are processed in two different modes:

- Interrupt mode and
- DMA assisted mode.

The two modes can be selected for channel A/B by programming the bit fields **LCONF**.MDA/MDB of LBI Configuration register. Note that devices such as the ESCC2 and HSCX support both modes, whereas the FALC54 supports only the interrupt mode. The choice of a particular method will be application dependent.

6.3.2 Data Transfer in Interrupt Mode

In the interrupt mode of data transfer, the DMSM interrogates certain pre-defined interrupt status registers of the LBI peripherals (addressed by DMSM/LBI Indirect External Configuration registers **LREG0** … **LREG5**), and takes action based on the status of certain data FIFO related status bits. Note that all other status bits are ignored by the DMSM but passed on to the host via the interrupt queue.

When an interrupt from a LBI peripheral is detected and not masked, the LBI Pass Trough Interrupt Vector is generated and written to the address specified in Peripheral Interrupt Queue Base Address (**PIQBA**) register.

The structure of the LBI Pass Through Interrupt Vector is as follows:

LBI Pass Through Interrupt Vector

	31					16
LPT_IV	1	0	0	0	LPTID(3:0)	GIS(7:0)

	15	0
LPT_IV	IS(7:0)	LSTAT(7:0)

LSTAT **LBI Interrupt Status**

Contains the values of **LSTAT** LBI Status Register's bit field (7 … 0) in all modes with automatic DMSM interrupt processing.

In the case of no DMSM interrupt processing this bit field is constant '0' and the corresponding interrupt vector ID is LPTID = 0110_B

IS **Interrupt Status**

Contains the Interrupt Status registers bit fields of the LBI peripheral (see table below).

GIS **Global Interrupt Status**

Contains the Global Interrupt Status registers bit fields of the LBI peripheral (see table below).

LPTID **LBI Pass Through Interrupt Vector ID**

Specifies the ID code for the different interrupt vectors (see table below). LPTID = 0110_B is generated if no automatic interrupt processing by DMSM is selected in register **LCONF**. In this case bit field LSTAT is constant '0'.

LPTID(3:0) Coding	IS Contents	GIS Contents
0000_B	GIS	ISR0A
0001_B	GIS	ISR1A
0010_B	GIS	ISR0B
0011_B	GIS	ISR1B
0110_B	00_H	00_H

Data Transfer Description

As HDLC packets are received by the LBI peripheral, they fill into the RFIFO (threshold value must be programmed in register to be compliant to that of the external device). When the threshold is reached, the peripheral generates the RPF interrupt. The DMSM services then, depending on the threshold value, up to 32 data bytes, assembles them and alerts the DMA controller to transfer them to host memory. Each DMA access transfers as many DWORDs as possible (PCI burst size = 8 DWORD typically). At the end of the Rx packet, the RME interrupt is serviced, the RBCL byte count (bytes remaining in RFIFO) is determined and the receive bytes are serviced. Data is fetched and the valid number of bytes is indicated in the status word at the end of the DMA buffer.

Similarly on the transmit side, when an XPR interrupt is detected, the DMSM requests the LBI DMA controller to take the Tx packet data. The LBI DMAC then delivers the data and stores it in the LBI TFIFO. When 32 bytes are available in the TFIFO, the DMSM transfers the 32 bytes to the peripheral, and sets the XTF bit in the peripheral to start sending out the packet. When the next XPR is indicated, the LBI DMA is alerted to fetch the next set of data, and the DMSM transfers it to the peripheral. When the Tx packet is completed, the DMAC indicates the number of valid bytes transferred. The DMSM transfers the valid bytes and sets the XTF and the XME bits to indicate to the peripheral HDLC controller to close the packet with the trailer bytes (CRC and Flag).This is implemented by the on-chip logic.

Note that all LBI peripheral interrupts are maskable via the DMSM register **LREG6**.

6.3.3 Data Transfer in DMA Assisted Mode

Some devices such as ESCC2 and HSCX support DMA assisted data transfers from and to their internal FIFOs. If this function is chosen, the DMSM services the DMA request pins of the peripheral (DRQTA, DRQRA, DRQTB, DRQRB) and acknowledges the requests with the $\overline{\text{DACKTA}}$, $\overline{\text{DACKRA}}$, $\overline{\text{DACKTB}}$ and $\overline{\text{DACKRB}}$ pins.
Note that the bit field **LCONF**.CDP in LBI Configuration register enables combined DMA acknowledge pins for receive and transmit direction of the LBI channels A/B (refer to register description section).

The DMSM supports transferring of data to the LBI FIFO from the peripheral's FIFO. It also uses the DMSM registers (**LREG0** … **LREG5**) to set the peripheral's XTF and RMC bits to start and complete packet transfers. It recognizes RME and XPR interrupts and passes other interrupts.

6.3.4 DMSM Registers

If the DMSM has to control the interrupt driven data handling of only a single device such as the HSCX, the addresses of the device specific registers (e.g., XFIFOA, RMC bit location in CMDR register) could be fixed. However, the state machine is needed to handle *different* devices (HSCX, ESCC2, FALC54 etc.), which have the same registers (e.g., XFIFO, CMDR), but located at different addresses. Hence it is necessary that the state machine uses an indirect pointer mechanism to address the required registers of the peripheral on the LBI.

Refer to LBI register description **Section LBI Registers** for an overview of the DMSM register set.

DMSM Register Initialization

The user (PCI host CPU) initializes the DMSM registers with *addresses* of specific registers (e.g., XFIFOA address is 00_H), and control bit *positions* of the peripheral that is attached as part of the configuration instructions.
After initialization, no further software interaction with these registers is required.

The control of the DMSM is handled by the LBI Configuration Register:

- **LCONF**.$\overline{\text{LBIRES}}$ resets and keeps the DMSM in an initial state (same as hardware reset state). For normal DMSM operation bit $\overline{\text{LCONF.LBIRES}}$ must be set to '1' again.
- **LCONF**.DCA: Ignore processing of channel A interrupts and pass everything to the interrupt queue.
- **LCONF**.DCB: Ignore processing of channel B interrupts and pass everything to the interrupt queue.

DMSM Suspend Mode

PCI Direct Accesses to the peripheral registers are possible at any time (refer to **Figure 50**). If such an access is requested while the DMSM assisted data transfers are taking place, the DMSM will go into 'suspend' mode after completing its current bus cycle and give up the bus to the Direct Access path. The PCI Direct Access cycle (read/write) may be extended, e.g. by using the $\overline{\text{TRDY}}$ signal.

If the PCI Direct Access is requested while the LBI does not have 'ownership' of the local bus (i.e., it is in slave mode), then the LBI will extend the PCI cycle until it is able to get the bus back and complete the PCI access cycle. The status of the LBI (master/slave mode) is indicated by an interrupt (via line LINTI1 or LINTI1 & LINTI2) and the bit field **LCONF**.ABM in LBI Configuration register.

6.4 Peripheral Device Register Read/Write Operation

The local bus can work in 8/16-bit multiplexed/de-multiplexed mode. This 16-bit address space can be mapped into the host memory space using the base address initialized as part of device configuration. Other configuration parameters define the clock speed of the local bus, and the number of wait states to be used with the local bus, and also the number of wait states to be added to PCI cycles.

Register Write to Peripherals

A PCI write within the local bus address space causes the address and data to be transferred to the peripherals on the local bus. The $\overline{\text{TRDY}}$ (target ready) signal is delayed until the peripheral is ready to accept the write data. With this approach, consecutive PCI writes are possible to this address range.

Register Read from Peripherals

The local bus address space is mapped into the shared memory space, and hence a Read operation is similar to a read from memory or any memory mapped register.

From the PCI host, this is a PCI Read cycle with pre-programmed number of wait cycles (0-15) for access to this address range.

Within the local bus, the Read address is physically mapped into the 16-bit address of the local bus, and Read cycle is performed to the peripheral. A 8/16 bit data read takes place at the pre-programmed local bus speed, and the 8/16 bit data is then passed on to the PCI cycle with the correct number of $\overline{\text{BE}}$ (byte enable) bits set.

6.5 Connection to Common Peripherals

As described above, the DMSM is designed to work optimally with Siemens HDLC devices for efficient transfer of packet data. However, the PCI-to-local bus bridge may be used to connect any other peripheral device with a microprocessor interface.

In this case the only function of the DMSM is to check interrupt signals LINTI1 and LINTI2 and to generate LBI pass through interrupt vectors which are transferred into peripheral interrupt queue. These interrupt vectors are of the constant value 86000000_H and indicate that an LBI interrupt event occured (refer to **Chapter 6.3.2**)

In the case that both LINTI input signals are used (LINTI2 enabled by bit LCONF.SPINT) simultaneous interrupt events may generate only one interrupt vector.

The current status of signals LINTI1 and LINTI2 can be checked anytime by read access to LBI Status Register LSTAT. The signal polarity can be selected via bits HE1 and HE2 in LBI Configuration Register LCONF.

The following bit settings in registers CONF and LCONF are appropriate to connect common peripherals:

Configuration Register CONF Offset Address 00_H	
LBI = 0	**LBI MODE Select** Must be set to ‘0’ to disable DMSM DMA support functions of the LBI.
LCD = n	**LBI Timer/Clock Division** Any value required for local bus operation
LBI Configuration Register LCONF **Offset Address 40_H**	
IPA	It is recommended to set this bit to ‘0’
DCA = ‘1’	**Disregards the Interrupts for Channel A** Must be set to ‘1’; the DMSM will transfer all interrupt indications without any automatic evaluation of interrupt reasons.
DCB = ‘1’	**Disregards the Interrupts for Channel B** Must be set to ‘1’; the DMSM will transfer all interrupt indications without any automatic evaluation of interrupt reasons.
MDA = ‘0’	**Mode Channel A** Must be set to ‘0’ to disable DMSM DMA functions.
MDB = ‘0’	**Mode Channel B** Must be set to ‘0’ to disable DMSM DMA functions.
SDA	It is recommended to set this bit to ‘0’
DID	It is recommended to set this bit to ‘0’
CDP	It is recommended to set this bit to ‘0’
$\overline{\text{EBCRES}}$ = ‘1’	**Reset LBI EBC Block** Must be set to ‘1’ to enable EBC operation
$\overline{\text{LBIRES}}$ = ‘1’	**Reset LBI DMSM Block** Must be set to ‘1’ to enable DMSM operation. Otherwise no interrupt vectors will be generated on LINTI1 and LINTI2 activity.
DV(2:0)	It is recommended to set this bit field to 000_B
HE1	**LINTI1 Polarity** HE1 = ‘0’ configures input signal LINTI1 for active high polarity HE1 = ‘1’ configures input signal LINTI1 for active low polarity
HE2	**LINTI2 Polarity** HE2 = ‘0’ configures input signal LINTI2 for active high polarity HE2 = ‘1’ configures input signal LINTI2 for active low polarity

SPINT	**Seperate Interrupt Pins** Must be set to ‘1’ if LINTI2 interrupt input signal is used. Otherwise only signal LINTI1 will be evaluated by DMSM.
EALE	These bits are to be configured depending on the local bus requirements. Refer to LCONF register description for details.
HDEN	
BTYP(1:0)	
RDEN	
ABM	
MCTC(3:0)	

6.6 LBI DMA Controller (DMAC)

The LBI provides a 4-channel bi-directional DMA controller (2 channels Rx, 2 channels Tx). Note that for LBI channel A, the DMA signal lines in receive and transmit direction may be shared by setting the bit field **LCONF**.SDA. The direction is then controlled by the bit field **LCONF**.DID.

The polling mechanism of the LBI DMAC differs from the serial PCM core DMAC. The LBI DMAC does not include the slow poll and hold poll functions of the serial PCM core DMAC, which is advantageous in this case, since the serial data traffic on the LBI can be highly asynchronous. A functionality similar to the $\overline{\text{POLL}}$(31:0) bit fields of the **TXPOLL** register is implemented in the LBI Start Transfer register **LTRAN**, which provides two bit fields $\overline{\text{GOA}}$ and $\overline{\text{GOB}}$ for the two LBI channels.

To initiate the LBI DMAC mode, the user has to setup a dummy descriptor with bit fields HOLD = ‘1’, FE = ‘1’ and NO = ‘0’ for the required channel(s). If valid data are available in shared memory, the HOLD bit in that dummy descriptor must be reset and **LTRAN**.$\overline{\text{GOA}}$/$\overline{\text{GOB}}$ must be programmed to ‘0’.

From then on, each time the HOLD bit in Tx descriptor is reset by the host to continue a data transfer, the value of **LTRAN**.$\overline{\text{GOA}}$/$\overline{\text{GOB}}$ must also be programmed to ‘0’.

When a DMA related interrupt on the LBI is detected, the LBI DMA Interrupt Vector is generated and written to the address specified in LBI Tx Interrupt Queue Base Address (**LTIQBA**) register for transmit direction or LBI Rx Interrupt Queue Base Address (**LRIQBA**) register for receive direction.

The structure of the LBI DMA Interrupt Vector is as follows:

LBI DMA Interrupt Vector

	31								16
LDMA_IV	0	1	1	R/T	0	0	0	0	00_H

	15															0
LDMA_IV	0	0	0	HI	FI	0	0	ERR	0	FE2	0	0	0	0	0	CHN

R/T — **Rx/Tx Direction**

‘1’: LBI DMAC Receive Interrupt

‘0’: LBI DMAC Transmit Interrupt

CHN — **Channel Number**

‘1’: LBI DMAC Channel B Interrupt

‘0’: LBI DMAC Channel A Interrupt

HI, FI, ERR, FE2 (Tx only) — **Host Interrupt, Frame Interrupt, Error, Frame End**

For a detailed description, refer to **Section 12.4**: **Section Interrupt Bit Field Definitions**.

7 Synchronous Serial Control (SSC) Interface

7.1 Overview

The Synchronous Serial Control (SSC) interface provides a flexible high-speed serial communication link between the MUNICH32X and other microcontrollers or external peripherals.

The SSC supports full-duplex and half-duplex synchronous communication up to 8.25 MBaud (@ 33 MHz bus clock). The serial clock signal can be generated by the SSC itself (master mode) or be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible, or Microwire compatible devices. Transmission and reception of data is double-buffered. A 16-bit baud rate generator provides the SSC with a separate serial clock signal.

The high-speed synchronous serial interface can be configured very flexibly, so it can be used with other synchronous serial interfaces (e.g., the ASC0 in synchronous mode), serve for master/slave or multimaster interconnections or operate compatible with the popular SPI interface. It allows communicating with shift registers (I/O expansion), peripherals (e.g. EEPROMs) or other controllers (networking).

Data is transmitted or received on the pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output or input on pin MCLK. These pins are implemented as alternate functions of the General Purpose Bus.

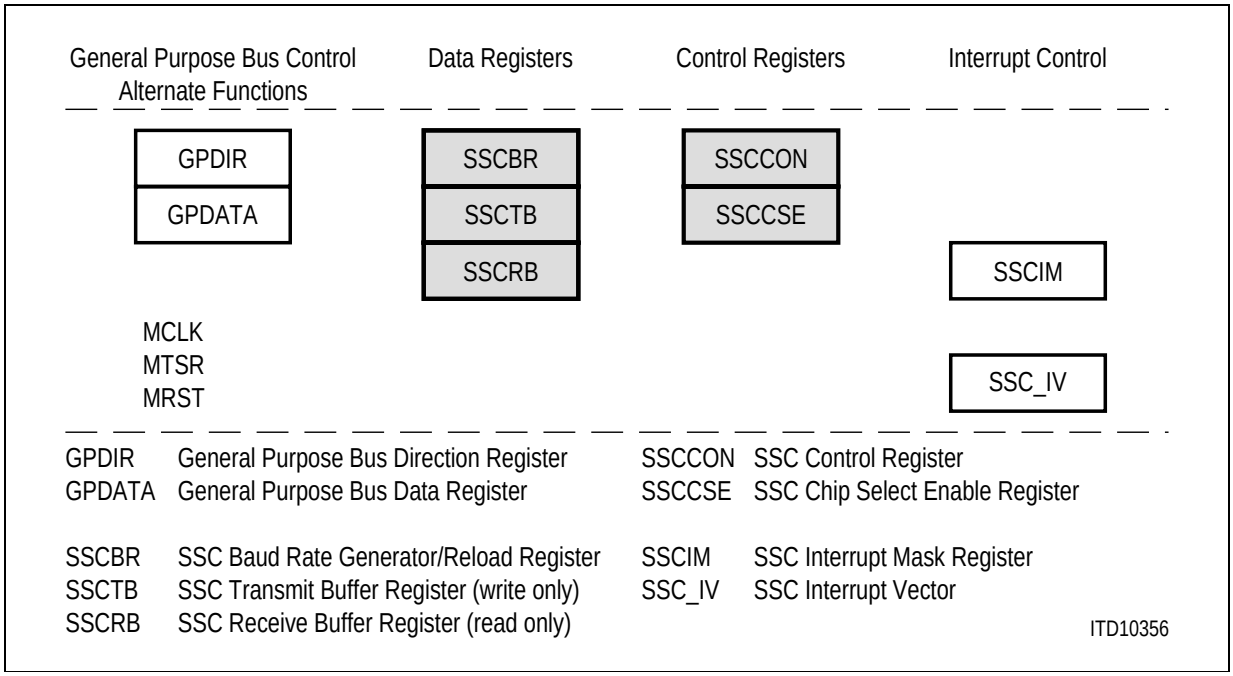


Figure 60
Registers and Port Pins Associated with the SSC

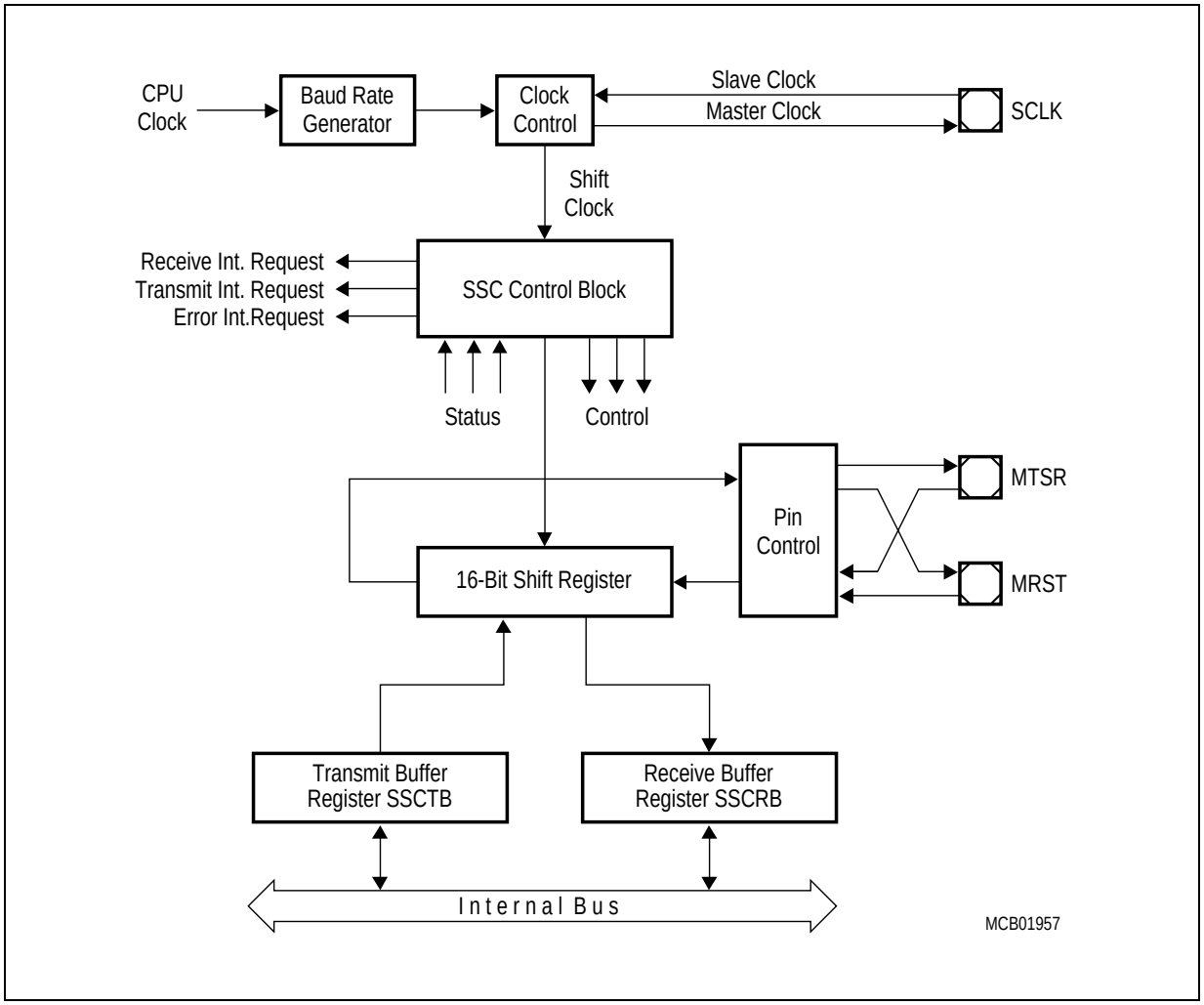


Figure 61
Synchronous Serial Channel SSC Block Diagram

The operating mode of the serial channel SSC is controlled by its bit-addressable control register SSCCON. This register serves for two purposes:

- During programming (SSC disabled by SSCEN = ‘0’) it provides access to a set of control bits,
- During operation (SSC enabled by SSCEN = ‘1’) it provides access to a set of status flags.

A detailed control register description for each of the two modes is provided in **Section 11.2.5**.

The shift register of the SSC is connected to both the transmit pin and the receive pin via the pin control logic (see block diagram). Transmission and reception of serial data is synchronized and takes place at the same time, i.e. the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer SSCTB. It is moved to the shift register as soon as this is empty. An SSC-master (SSCMS = ‘1’) immediately

starts transmitting, while an SSC-slave (SSCMS = ‘0’) waits for an active shift clock. When the transfer starts, the busy flag SSCBSY is set and a transmit interrupt request (SSCTXI) will be generated to indicate that SSCTB may be reloaded again. When the programmed number of bits (2 … 16) has been transferred, the contents of the shift register is moved to the Receive Buffer SSCRB and a receive interrupt request (SSCRXI) will be generated. If no further transfer is to take place (SSCTB is empty), SSCBSY will be cleared at the same time. Software should not modify SSCBSY, as this flag is hardware controlled.

Note: Only one SSC can be master at a given time.

The transfer of serial data bits may be programmed in many respects:

- The data width may be selected in a range between 2 bits and 16 bits.
- Transfer may start with the LSB or the MSB.
- The shift clock may be idle low or idle high.
- Data bits may be shifted with the leading or trailing edge of the clock signal.
- The baudrate may be set from 152 Baud up to 5 MBaud (@ 20 MHz CPU clock).
- The shift clock can be either generated (master) or received (slave).

This flexible programming allows to adapt the SSC to a wide range of applications, where serial data transfer is required.

The Data Width Selection allows to transfer frames of any length, from 2-bit ‘characters’ up to 16-bit ‘characters’. Starting with the LSB (SSCHB = ‘0’) allows communicating e.g. with ASC0 devices in synchronous mode (C166 family) or 8051 like serial interfaces. Starting with the MSB (SSCHB = ‘1’) allows to operate compatible with the SPI interface. Regardless which data width is selected and whether the MSB or the LSB is transmitted first, the transfer data is always right aligned in registers SSCTB and SSCRB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of SSCTB are ignored, the unselected bits of SSCRB will be not valid and should be ignored by the receiver service routine.

The Clock Control allows to adapt transmit and receive behaviour of the SSC to a variety of serial interfaces. A specific clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in receive data.
Bit SSCPH selects the leading edge or the trailing edge for each function. Bit SSCPO selects the level of the clock line in the idle state. Hence for an idle-high clock the leading edge is a falling one, a 1-to-0 transition. The figure below summarizes the clock control.

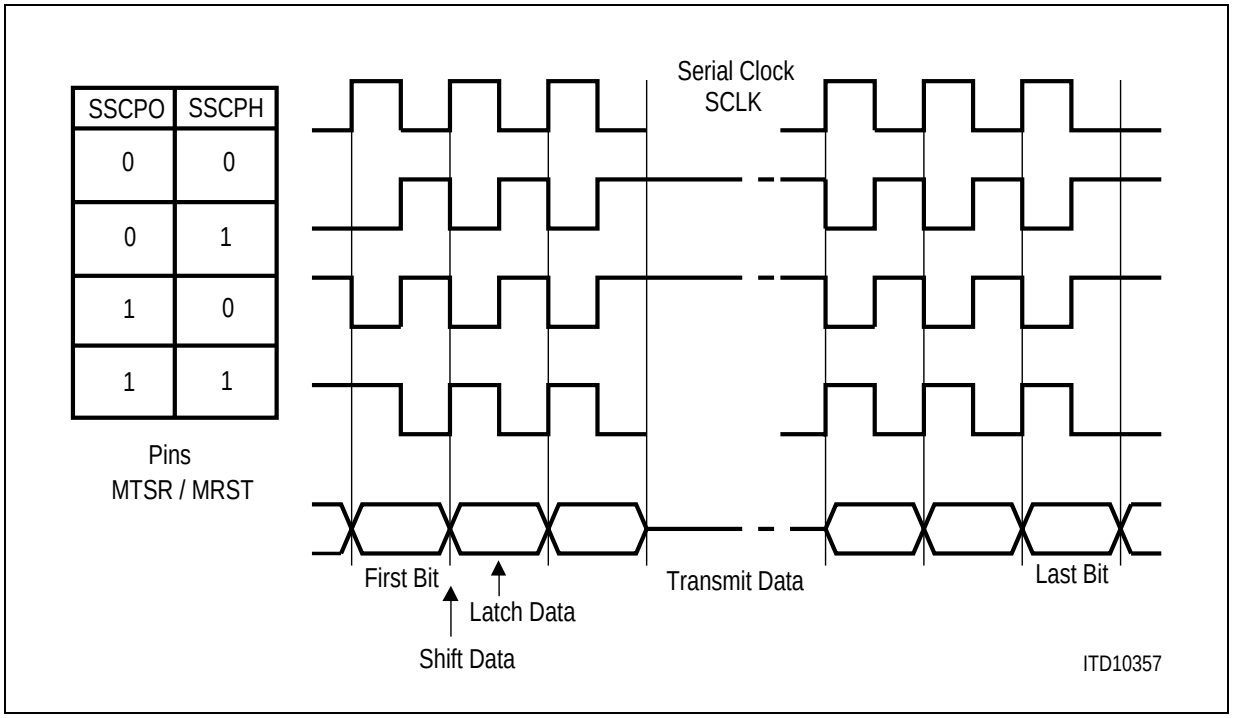


Figure 62
Serial Clock Phase and Polarity Options

7.2 Operational Mode

7.2.1 Full-Duplex Operation

The different devices are connected through three lines. The definition of these lines is always determined by the master: The line connected to the master's data output pin MTSR is the transmit line, the receive line is connected to its data input line MRST, and the clock line is connected to pin MCLK. Only the device selected for master operation generates and outputs the serial clock on pin MCLK. All slaves receive this clock, so their pin MCLK must be switched to input mode (GPDIR.p = '0'). The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

When initializing the devices in this configuration, select one device for master operation (SSCMS = '1'), all others must be programmed for slave operation (SSCMS = '0'). Initialization includes the operating mode of the device's SSC and also the function of the respective port lines (refer to section 'Port Control').

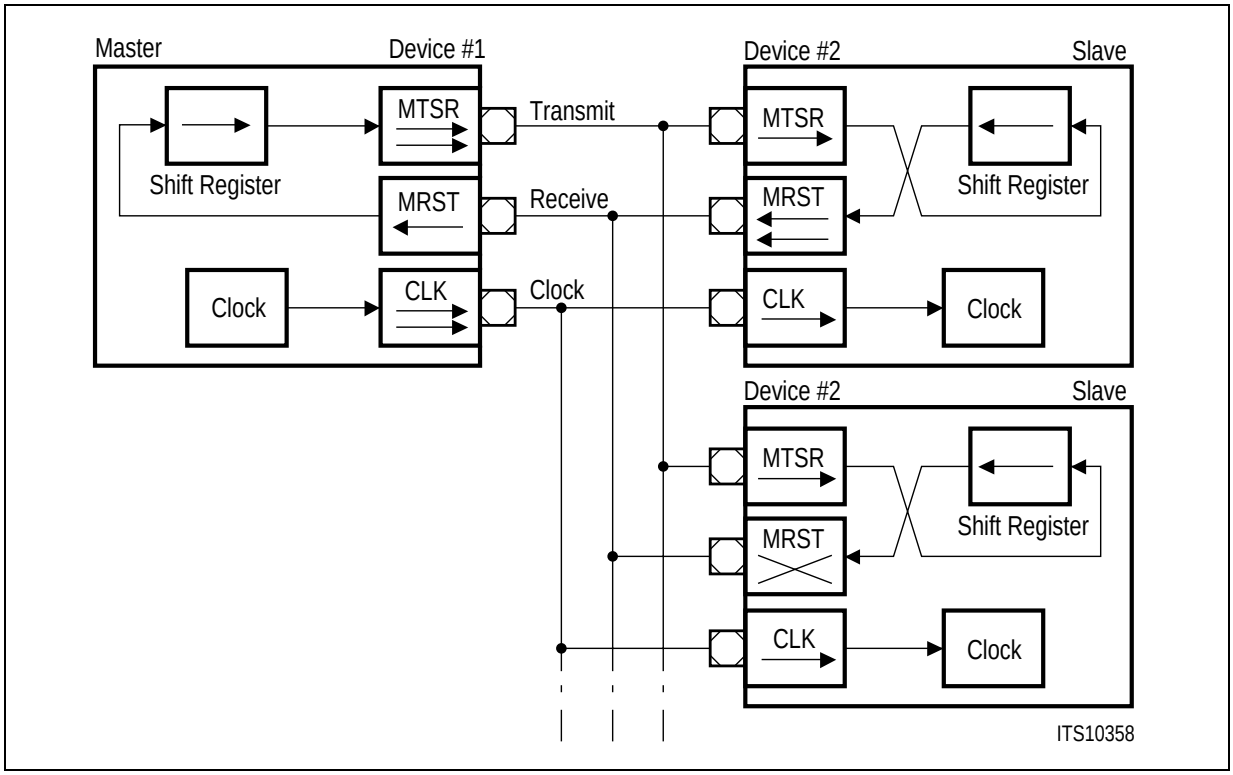


Figure 63
SSC Full Duplex Configuration

Note: The shift direction applies to MSB-first operation as well as to LSB-first operation.

The data output pins MRST of all slave devices are connected together onto the one receive line in this configuration. During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

1. **Only one slave drives the line**, i.e. enables the driver of its MRST pin. All the other slaves have to program there MRST pins to input. So only one slave can put its data onto the master's receive line. Only receiving of data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output, until it gets a deselection signal or command.
2. **The slaves use open drain output on MRST**. This forms a Wired-AND connection. The receive line needs an external pullup in this case. Corruption of the data on the receive line sent by the selected slave is avoided, when all slaves which are not selected for transmission to the master only send '1s'. Since this high level is not actively driven onto the line, but only held through the pullup device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device, from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initializations of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either ‘0’ or ‘1’, until the first transfer will start. After a transfer the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interface is enabled, the master device can initiate the first data transfer by writing the Tx data into Tx Buffer Register SSCTB. This value is copied into the shift register (which is assumed to be empty at this time), and the selected first bit of the Tx data will be placed onto the MTSR line on the next clock from the Baudrate Generator (transmission only starts, if SSCEN = ‘1’). Depending on the selected clock phase, a clock pulse will also be generated on the MCLK line. With the opposite clock edge the master at the same time latches and shifts in the data detected at its input line MRST. This ‘exchanges’ the Tx data with the Rx data. Since the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the preprogrammed number of clock pulses (via the data width selection) the data transmitted by the master is contained in all slaves’ shift registers, while the master’s shift register holds the data of the selected slave. The contents of the shift register of the master and all slaves are copied into the Rx Buffer Register SSCRB and the Rx interrupt flag SSCRXI is set.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin MRST, when the contents of the Tx buffer are copied into the slave’s shift register. It will not wait for the next clock from the baudrate generator, as the master does. The reason is that, depending on the selected clock phase, the first clock edge generated by the master may already be used to clock in the first data bit. Hence the slave’s first data bit must already be valid at this time.

*Note: On the SSC always a transmission **and** a reception takes place at the same time, regardless whether valid data has been transmitted or received. This is different, e.g., from asynchronous reception on ASC0.*

The initialization of the MCLK pin on the master requires some attention in order to avoid undesired clock transitions, which may disturb the other receivers. The state of the internal alternate output lines is ‘1’ as long as the SSC is disabled. This alternate output signal is ANDed with the respective port line output latch. Enabling the SSC with an idle-low clock (SSCPO = ‘0’) will drive the alternate data output and (via the AND) the port pin MCLK immediately low. To avoid this, use the following sequence:

- select the clock idle level (SSCPO = ‘x’),
- load the port output latch with the desired clock idle level (GPDATA.p = ‘x’),
- switch the pin to output (GPDIR.p = ‘1’),
- enable the SSC (SSCEN = ‘1’), and
- if SSCPO = ‘0’: enable alternate data output (GPDATA.p = ‘1’).

The same mechanism as for selecting a slave for transmission (separate select lines or special commands) may also be used to promote the role of the master to another device in the network. In this case the previous master and the future master (previous slave) will have to toggle their operating mode (SSCMS) and the direction of their port pins (see description above).

Chip Select Control

There are 4 chip select pins associated with the SSC port: $\overline{\text{MCS0}}$ to $\overline{\text{MCS3}}$. The four chip select lines are automatically activated at the beginning of a transfer and deactivated again after the transfer has ended. Activation of a chip enable line always begins one half bit time before the first data bit is output at the MTSR pin, and the deactivation (except for the continuous transfers) is performed one half bit time after the last bit of the transfer has been transmitted/received completely.

The chip select lines are selected by the control bits ASEL0 to ASEL3 of the SSC Chip Select Enable Register SSCCSE (refer to **Chapter 11.2.5**). By setting any of these bits to 0, the corresponding chip select port will be asserted when transmitting data. All other bits of the SSCCSE register have to be set to ‘0’.

7.2.2 Half Duplex Operation

In a half duplex configuration only one data line is necessary for both receiving **and** transmitting of data. The data exchange line is connected to both pins MTSR and MRST of each device, the clock line is connected to the MCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to full duplex mode there are **two ways to avoid collisions** on the data exchange line:

- only the transmitting device may enable its Tx pin driver
- the non-transmitting devices use open drain outputs and only send ones.

Since the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). This allows to detect any corruptions on the common data exchange line, where the Rx data is not equal to the Tx data.

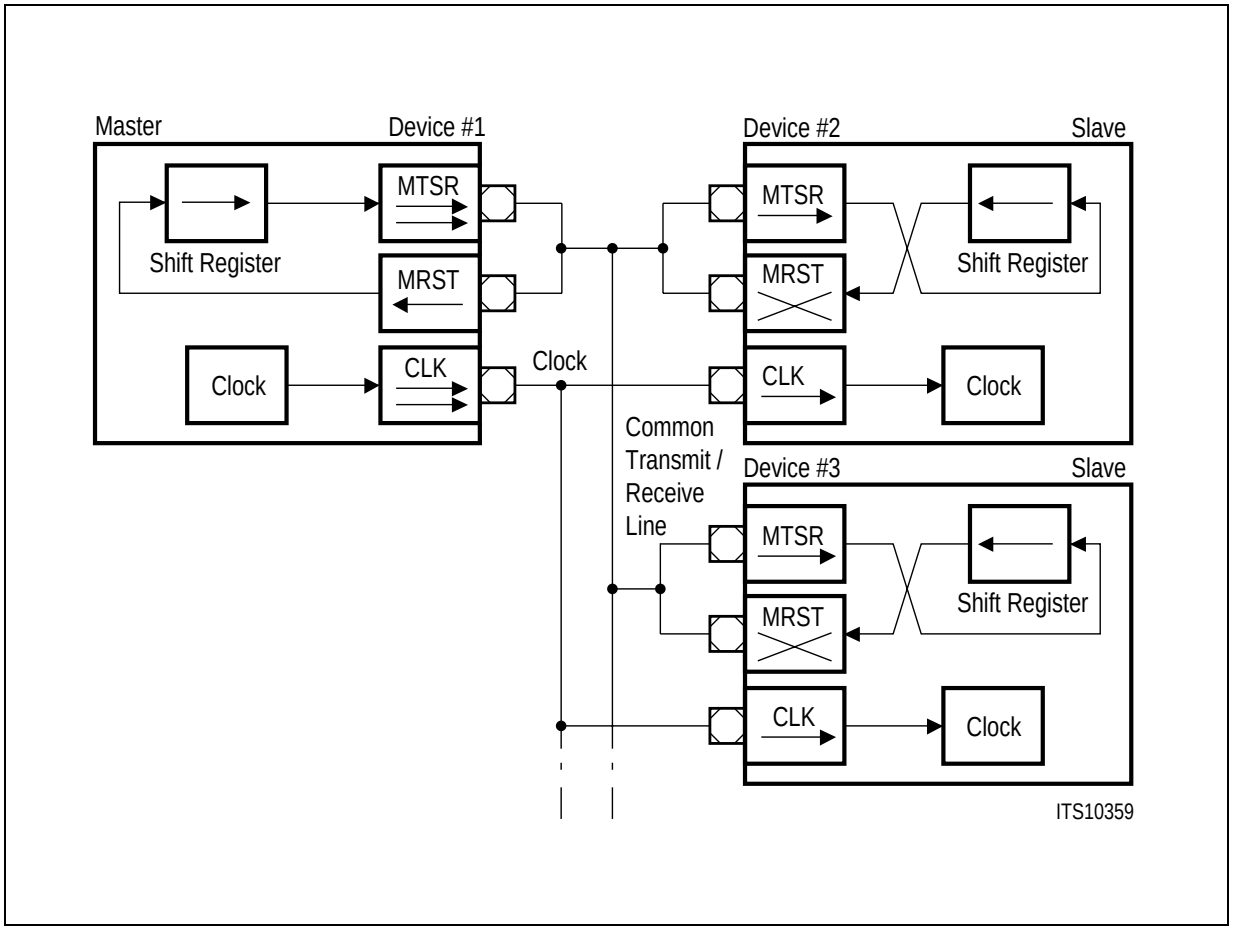


Figure 64
SSC Half Duplex Configuration

Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the Tx Buffer Register SSCTB is empty and ready to be loaded with the next transmit data. If SSCTB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line there is no gap between the two successive frames. For example, two byte transfers would look the same as one 16-bit word transfer. This feature can be used to interface with devices which can operate with or require more than 16 data bits per transfer. The length of a total data frame is up to the software. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus.

Note: This feature only applies to multiples of the selected basic data width, since it would require disabling/enabling of the SSC to re-program the basic data width on-the-fly.

Port Control

The SSC uses three pins of the General Purpose Bus pins to communicate with the external world. Pin GP15/MCLK serves as the clock line, while pins GP13/MRST (Master Receive/Slave Transmit) and GP14/MTSR (Master Transmit/Slave Receive) serve as the serial data I/O lines. The operation of these pins depends on the selected operating mode (master or slave).

In order to enable the alternate output functions (in this case SSC functions) of these pins instead of the general purpose I/O operation, the respective port latches of the General Purpose Bus registers (refer to **Chapter 11.2.1**) have to be set to '1', since the port latch outputs and the alternate output lines are ANDed. When an alternate data output line is not used (function disabled), it is held at a high level, allowing I/O operations via the port latch.

The direction of the port lines depends on the operating mode. The SSC will automatically use the correct alternate input or output line of the ports when switching the modes. The direction of the pins, however, must be programmed by the user, as shown in **Table 17**. Using the open drain output feature helps to avoid bus contention problems and reduces the need for hardwired hand-shaking or slave select lines. In this case it is not always necessary to switch the direction of a port pin. The table below summarizes the required values for the different modes and pins.

Table 17
Port Control of the SSC Interface

Pin	Master Mode		
	Function	**Port Latch**	**Direction**
GP15/MCLK	SSC Clock Output	GPDATA.15 = '1'	GPDIR.15 = '1'
GP14/MTSR	SSC Data Output	GPDATA.14 = '1'	GPDIR.14 = '1'
GP13/MRST	SSC Data Input	GPDATA.13 = 'x'	GPDIR.13 = '0'
	Slave Mode		
GP15/MCLK	SSC Clock Input	GPDATA.15 = 'x'	GPDIR.15 = '0'
GP14/MTSR	SSC Data Input	GPDATA.14 = 'x'	GPDIR.14 = '0'
GP13/MRST	SSC Data Output	GPDATA.13 = '1'	GPDIR.13 = '1'

Note: An 'x' means that the actual value is irrelevant in the respective mode. However, it is recommended to set these bits to '1' to make sure they are already in the correct state when switching between master and slave mode.

7.3 Baud Rate Generation

The SSC interface has its own dedicated 16-bit baud rate generator with 16-bit reload capability, allowing baud rate generation independent from timers.

The baud rate generator is clocked with the CPU clock divided by 2 (10 MHz @ 20 MHz bus clock). The timer is counting downwards and can be started or stopped through the global enable bit SSCEN in the SSC Control Register SSCCON.

The register SSCBR (refer to **Chapter 11.2.5**) is the dual-function Baud Rate Generation Register. Reading SSCBR, while the SSC is enabled, returns the contents of the timer. Reading SSCBR, while the SSC is disabled, returns the programmed reload value. In this mode the desired reload value can be written to SSCBR.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baudrate:

$$\text{Baudrate}_{SSC} = \frac{f_{CPU}}{2 \times (<SSCBR> + 1)} \qquad \text{SSCBR} = \left(\frac{f_{CPU}}{2 \times \text{Baudrate}_{SSC}} \right) - 1$$

<SSCBR> represents the contents of the reload register, taken as unsigned 16-bit integer.

7.4 Error Detection

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, while Transmit Error and Baudrate Error only apply to slave mode. When an error is detected, the respective error flag is set. When the corresponding error enable bit is set, also an error interrupt request will be generated by setting SSCERI (see **Figure 65**). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset automatically (like SSCERI), but rather must be cleared by software after servicing. This allows to service some error conditions via interrupt, while the others may be polled by software.

Note: The error interrupt handler must clear the associated (enabled) errorflag(s) to prevent repeated interrupt requests.

A **Receive Error** (Master or Slave mode) is detected, when a new data frame is completely received, but the previous data was not read out of the Receive Buffer register SSCRB. This condition sets the error flag SSCRE and, when enabled via SSCREN, the error interrupt request flag SSCERI. The old data in the receive buffer SSCRB will be overwritten with the new value and is unretrievably lost.

A **Phase Error** (Master or Slave mode) is detected, when the incoming data at pin MRST (master mode) or MTSR (slave mode), sampled with the same frequency as the CPU clock, changes in a range between one sample before and two samples after the latching

edge of the clock signal (refer to section ‘Clock Control’). This condition sets the error flag SSCPE and, when enabled via SSCPEN, the error interrupt request flag SSCERI.

A **Baud Rate Error** (Slave mode) is detected, when the incoming clock signal deviates from the programmed baud rate by more than 100%, i.e., it has a value of either more than double or less than half of the expected baud rate. This condition sets the error flag SSCBE and, when enabled via SSCBEN, the error interrupt request flag SSCEIR.

Using this error detection capability requires that the slave’s baud rate generator is programmed to the same baud rate as the master device. This feature allows to detect false additional, or missing pulses on the clock line (within a certain frame).

Note: If this error condition occurs and bit SSCAREN = ‘1’, an automatic reset of the SSC will be performed. This is done to reinitialize the SSC, when too few or too many clock pulses have been detected.

A **Transmit Error** (Slave mode) is detected, when a transfer was initiated by the master (shift clock gets active), but the transmit buffer SSCTB of the slave was not updated since the last transfer. This condition sets the error flag SSCTE and, when enabled via SSCTEN, the error interrupt request flag SSCERI. If a transfer starts while the transmit buffer is not updated, the slave will shift out the ‘old’ contents of the shift register, which are usually the data received during the last transfer.

This may lead to the corruption of the data on the transmit/receive line in half-duplex mode (open drain configuration), if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ‘ones’, i.e. their transmit buffers must be loaded with ‘$FFFF_H$’ prior to any transfer.

Note: A slave with push/pull output drivers, which is not selected for transmission, will normally have its output drivers switched. However, in order to avoid possible conflicts or misinterpretations, it is recommended to always load the slave’s transmit buffer prior to any transfer.

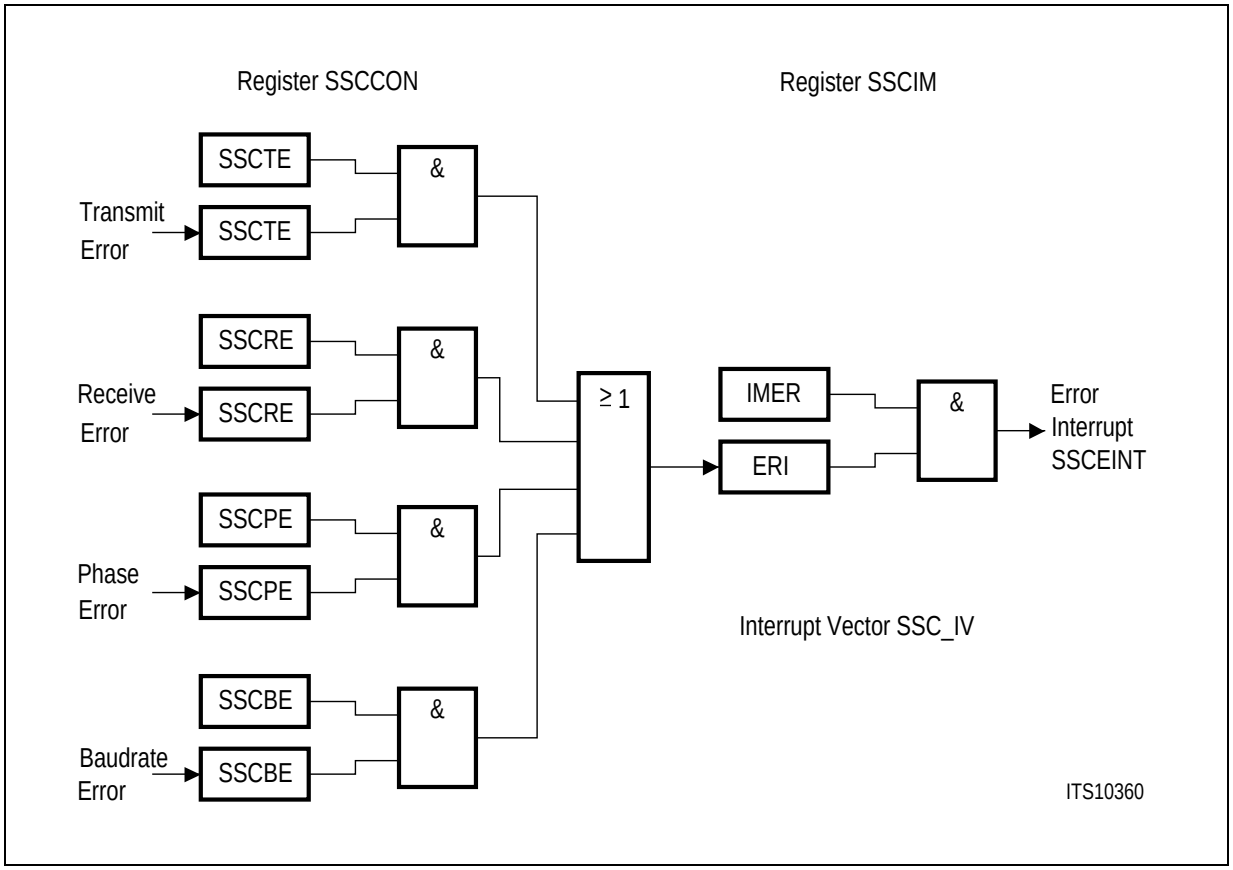


Figure 65
SSC Error Interrupt Control

7.5 SSC Interrupt Control

The SSC generates three types of interrupts: transmit, receive and error interrupts. Any of these interrupts can be enabled by setting the corresponding bit of the SSC Interrupt Mask Register SSCIM (refer to **Chapter 11.2.5**) to ‘1’.

All other bits of this register have to be set to zero.

SSC Interrupt Vector

	31															16
SSC_IV	1	0	1	0	0	0	0	R/T	D/E	0	0	0	0	ERI	RXI	TXI

	15	0
SSC_IV	IV(15:0)	

R/T **Receive/Transmit Direction**

0: Transmit direction

1: Receive direction

D/E **Data/Error Interrupt**

0: Error interrupt

1: Data interrupt

RXI **Rx Interrupt**

1: Indicates receive interrupt

ERI **Error Interrupt**

1: Indicates error interrupt

TXI **Tx Interrup**

1: Indicates transmit interrupt.

IV **Interrupt Vector**

Three different values:

RXI = 1: Contents of SSC Receive Buffer Register.

ERI = 1: Contents of SSC Control Register.

TXI = 1: 0000_H

Example:

1. Transmit: SSC_IV(31 : 0) = $A0010000_H$
2. Error: SSC_IV(31:16) = $A004_H$
   SSC_IV(15:0): contents of SSC Control Register SSCCON
3. Receive: SSC_IV(31:16) = $A002_H$
   SSC_IV(15:0): contents of SSC Rx Buffer Register SSCRB

8 IOM®-2 Interface

The MUNICH32X contains an integrated IOM®-2 handler for driving ISDN Layer-1 devices via the serial PCM interface. **Figure 66** shows an application with the MUNICH32X driving 8 S/T Basic Rate Interfaces via 2 QUAT®-S, PEB 2084, ISDN devices. All D-Channel and B-Channel packet processing is handled through the MUNICH32X using 24 HDLC channels operating directly on the host memory (no difference to normal HDLC operation).

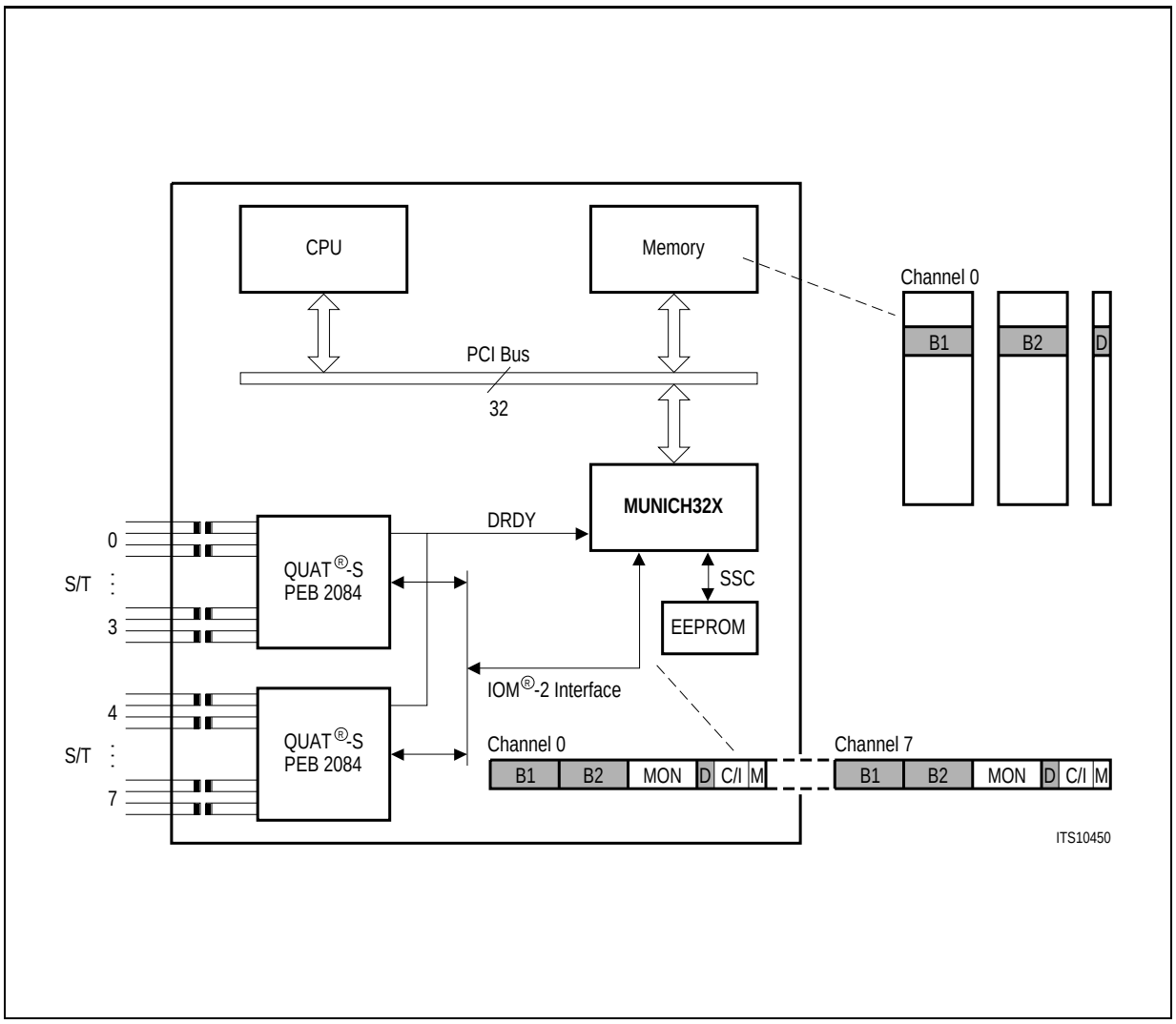


Figure 66
8 S/T Interfaces realized by one MUNICH32X and two QUAT®-S

8.1 General Features

The IOM®-Revision-2 (IOM®-2) standard defines an industry standard serial bus for interconnecting telecommunications ICs. The standard covers line card, NT1, and terminal architectures for ISDN and analog loop applications.

In digital telephone switches, an inter-chip communication bus is often used to connect the codec/filter ICs to the switch backbone. Typically, a separate serial bus is used between each codec/filter IC and a line-card controller IC. The line card controller provides the connection to the switch backbone, as well as an interface to the microprocessor that controls the line card. The inter-chip bus structures that were used in this single channel per line pre-ISDN (Integrated Services Digital Network) telephone equipment are not well-suited for the 2 B + D structure of ISDN.

To address this problem, four major European telephone equipment manufacturers jointly defined a new interface bus. These four companies, Alcatel, Italtel, Plessey, and Siemens invented a bus structure that satisfied the requirements of both ISDN and analog applications. The General Circuit Interface, or GCI, is an evolution of the ISDN Oriented Modular Interface (IOM®) invented by Siemens. The GCI was designed with the specific needs of interconnecting components on switch line cards. As such, it is not well-suited for terminal and NT1 applications. As a result, a terminal version of the interface has been designed. The terminal version has been designated the Special Circuit Interface T, or SCIT. The GCI line card and SCIT terminal bus specifications combined form the IOM® Revision 2 standard (IOM®-2).

The MUNICH32X contains an integrated IOM®-2 handler for driving ISDN Layer-1 devices via the serial PCM interface. In this mode, the PCM interface has a structure and features of the IOM®-2 interface:

- It is compatible with the IOM®-2 industry standard for line cards/trunk cards (supporting the Monitor and C/I communication channels), **Figure 67.**
- Its 4 lines correspond to TXD/DU, RXD/DD, DCL, FSC of other ISDN devices.
- The data rate is programmable up to 4.096 Mbit/s.
- The clock frequency is either equal to the data rate, or twice the data rate. By default it is equal to the data rate.

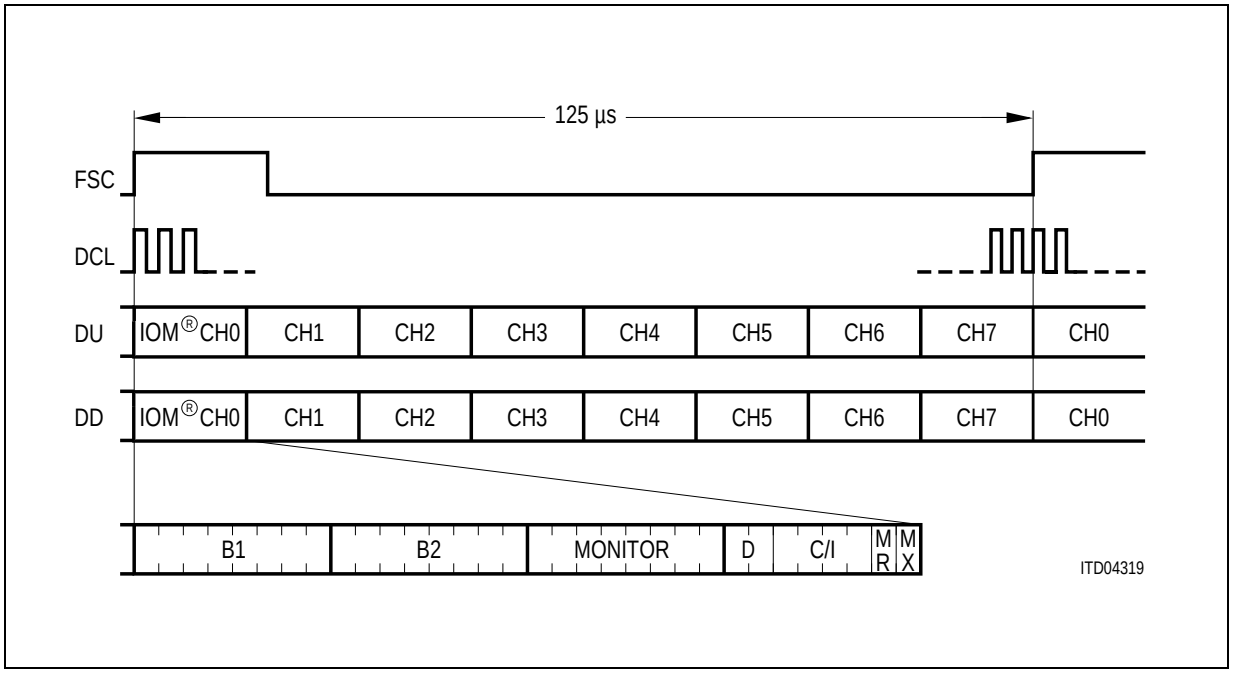


Figure 67
IOM®-2 Interface with 2.048 Mbit/s Data Rate

8.1.1 B-Channels

The B1- and B2-channels are physically the first two 8-bit time slots after the frame sync pulse. Each B-channel carries 64-Kbit/s of user data (or digitized voice).

8.1.2 D-Channel

The D-channel contains 2 bits per frame, providing a 16-Kbit/s channel for carrying ISDN D-channel data. (In analog line-card applications of IOM®-2, there is no D-channel).

8.1.3 Monitor Channel (including MX, MR bits)

The monitor channels provide an interface between the microprocessor, via the line-card controller (line-card applications) or the IOM®-2 bus master (terminal applications), and devices attached to the bus. This allows these devices to be designed without their own microprocessor interface. Each channel consists of 8 bits of data and two associated handshake bits, MX and MR (monitor transmit and receive). The handshake procedure is described in the **Chapter 8.2.1**.

8.1.4 Command/Indicate Channel

The Command/Indicate channel (C/I) carries real-time status information between a line transceiver and the MUNICH32X.

Status information transmitted over the C/I channel is “static” in the sense that a 4-bit word is repeatedly transmitted, every frame, as long as the status condition that it indicates is valid. In general, the receiver monitors the C/I channel for changes in status. The definition and usage of the 4-bit C/I codes is described in the **Chapter 8.2.2.**

8.2 IOM®-2 Handler

The built-in IOM®-2 handler processes the C/I and MONITOR channels of the IOM®-2 protocol; i.e. 8 independent IOM®-2 channels (subframes). It also supports the D-channel access (priority control) on the S/T interface via the DRDY input line (**Figure 66**).

8.2.1 Monitor Handler

Handshake Procedure

The monitor channel is full duplex and operates on a pseudo-asynchronous basis, that is, while data transfers on the bus take place synchronized to frame sync, the flow of data is controlled by a handshake procedure using the MX and MR bits. For example: data is placed onto the monitor channel and the MX bit is activated. This data will be transmitted repeatedly (once per 8-kHz frame) until the transfer is acknowledged via the MR bit. Thus, the data rate is not 8-kbytes/s.

Figure 68 illustrates the flow of events, and **Figure 69** and **Figure 70** show the timing.

Idle

The MX and MR pair being held inactive for two or more frames constitutes the channel being idle in that direction. The data (logical “high”, i.e. negative logic) received in the monitor channel is invalid and should be “11111111”.

Start of Transmission

The first byte of data is placed on the bus and MX is activated (low). MX remains active, and the data remains valid until an inactive-to-active transition of MR is received, indicating that the receiver has read the data off the bus.

Subsequent transmission (general case **Figure 69**). – In the case of the second byte transfer the transmitter detects the MR bit transition from the inactive to the active state before transmitting the new second byte. At the time that a new byte is transmitted, MX is returned inactive for one frame time only (MX inactive for more than one frame time indicates an end of message), the data is valid in the same frame. In the following frame MX returns active again and the same data is transmitted. Data is repeated in

subsequent frame and MX remains active until acknowledgement is detected (MR transition from inactive to active).

Only packets with a multiple of two bytes length can be transmitted via the monitor channel (refer to registers IOMCON1 and IOMTMO).

First Byte Reception

At the time the receiver sees the first byte, indicated by the inactive-to-active transition of MX, MR is by definition inactive. When the receiver is ready to acknowledge the first byte MR is activated. MR remains active until the next byte is received or an end of message is detected (MX held inactive for two or more frame times).

Subsequent Reception

The receiver acknowledges the receipt of a valid data by the transition of MR bit from the active to the inactive state for one frame followed by the transition to the active state in the next frame (**Note:** Validity of the data, including the first byte, can optionally be checked by the receiver applying a double-last-look criterion on each received data.)

The reception of data is terminated by the receipt of an end-of-transmission indication (MX remaining inactive for two or more frame times).

End of Transmission (EOM) – The transmitter after receiving a successful last byte acknowledge will indicate EOM by the transition of the MX bit from the active to inactive state followed by the persistence of the inactive state for at least one more frame. The contents of the monitor channel will become invalid in the same frame as the transition of the MX bit occurs. The invalid data should be “11111111”.
The sender is then in the idle state.

Abort – The abort is a signal from the receiver to the transmitter indicating that data has been missed. It is not an abort in the classical sense, which is an indication from the transmitter that the current message should be ignored. The receiver indicates an abort by holding MR inactive for two or more frames in response to MX going active.

Flow Control – The receiver can hold off the transmitter by keeping MR active until the receiver is ready for the next byte. The transmitter will not start the next transmission cycle until MR goes inactive.

Time-Out – For the cases where no device acknowledges a data, or where the transmitter is unable to resume transmission, an optional time-out may be implemented.

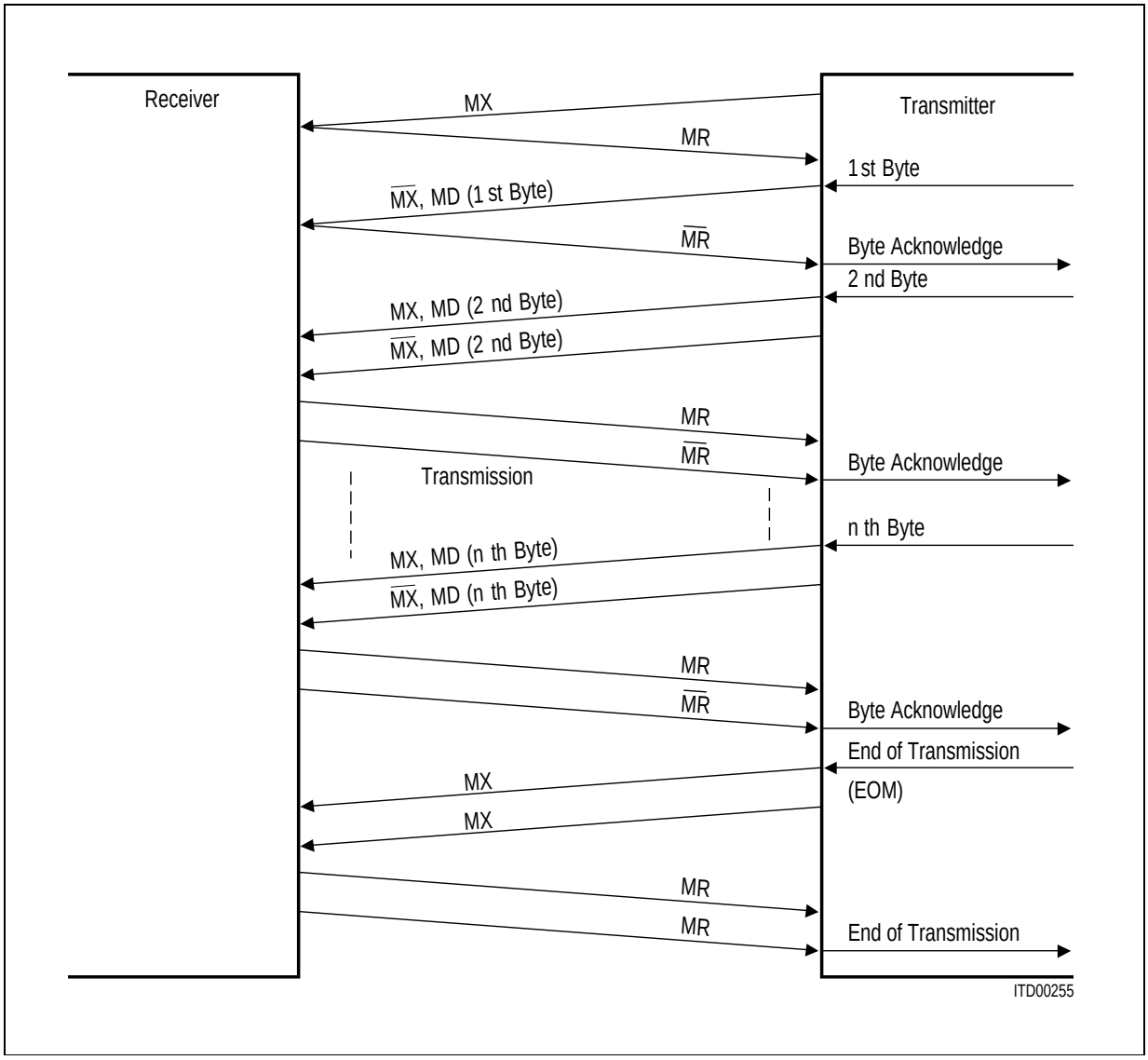


Figure 68
Monitor Handshake Procedure

MX: monitor transmit bit, active low
MR: monitor receive bit, active low
MD: monitor data

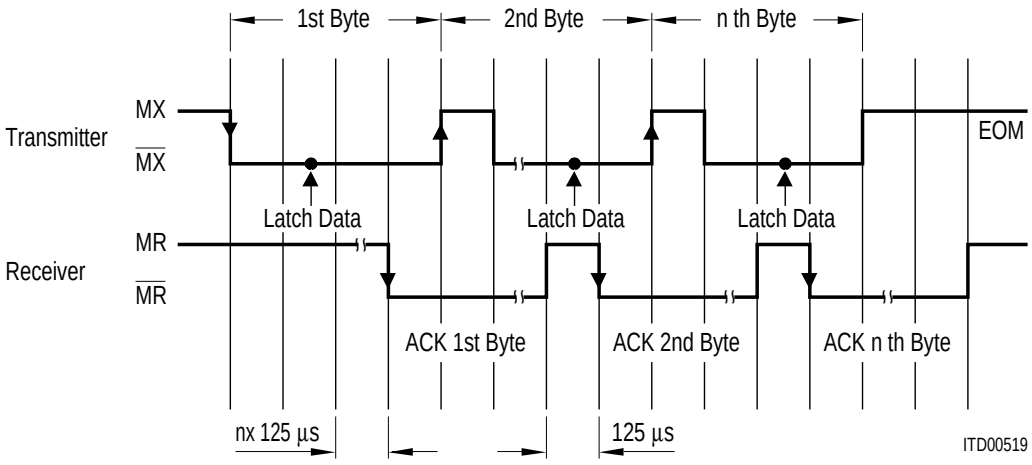


Figure 69
Monitor Handshake Timing in General

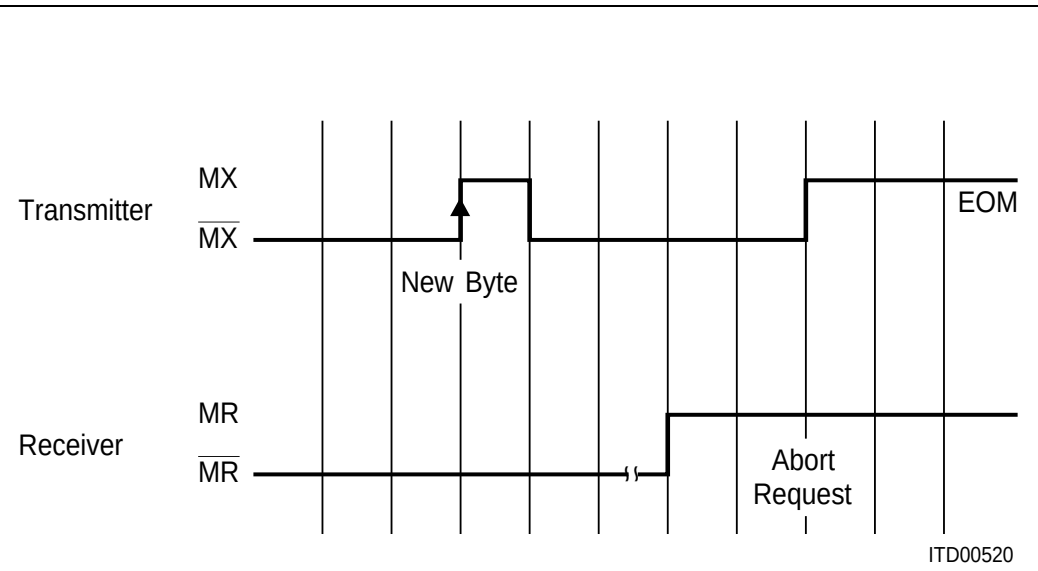


Figure 70
Abort Request from Receiver

8.2.2 C/I Channel Operation

The C/I channel is used to communicate real time status information and maintenance commands, such as loopback requests, link activation/deactivation procedures, and switch hook/ground key detection (voice channels). Data on the C/I channel is continuously transmitted in each frame until new data is to be sent. In this way, the C/I channel can be thought of as a set of static status lines that only change when the status changes.

Data Integrity

Where data integrity is a concern, a change in C/I channel data may be considered valid if it has been received in two consecutive frames.

Command/Indication Codes (C/I) Codes

The following is a description of the different commands and indications that are used over the IOM®-2 interface for signaling and control in ISDN systems.

Table 18
C/I Codes

Code	TXD (DD)	RXD (DU)
0000	DR	TIM
0001	RES	RES
0010	TM2	–
0011	TM1	–
0100	–	RSY
0101	–	–
0110	–	–
0111	UAR	UAI
1000	AR	AR
1001	AR2	–
1010	ARL	–
1011	AR4	–
1100	AI	AI
1101	–	–
1110	–	–
1111	DC	DI

Table 19
List of Commands and Indications

Commands and Indications	Abbreviation
Activation Indication	AI
Activation Indication priority 1	AI8
Activation Indication priority 2	AI10
Activation Indication test loop 2	AI2
Activation Indication local test loop	AIL
Activation Request	AR
Activation Request priority 1	AR8
Activation Request priority 2	AR10
Activation Request test loop 2	AR2
Activation Request local test loop	ARL
Activation Request test loop 4	AR4
Deactivation Confirmation	DC
Deactivation Indication	DI
Deactivation Request	DR
Power Up	PU
Reset	RES
Resynchronization (loss of framing)	RSY
Slip detected in framing	SLIP
Timing required (to activate IOM®-2)	TIM
Test Mode 1	TM1
Test Mode 2	TM2
U only Activation Indication	UAI
U only Activation Request	UAR

Note: The complete IOM®-2 register set is introduced in ***Section 11.2.6***

8.2.3 D-Channel Priority Control

In an application where the MUNICH32X represents a terminal (TE) interfaced to an S-bus via an S/T interface transceiver like the QUAT®-S (PEB 2081) and a multipoint configuration is implemented (i.e. more than one terminal is hooked up to one S-bus), the D-channel access has to be organized solving collision conditions. This D-channel access control is specified in ITU I.430. A data collision may occur during the MUNICH32X starts sending a packet over the D-channel. This collision is detected by the S/T interface transceiver and reported to the MUNICH32X by asserting the DRDY pin. Every time a D-channel collision occurs, this event is reported to the protocol controller. The transmit line (DU) sends continuous ‘1’s. A “Late Stop” interrupt is generated and stored in the PCM Tx Interrupt Queue, indicating, that the current frame has to be retransmitted (refer to PCM Core Interrupt Bit Field description, chapter 12.4) in case of destructive collision of transmit data.

8.3 IOM®-2 Interrupt Vector Description

8.3.1 Monitor Interrupt Vector

	31														16
IOMM_IV	1	0	0	1	0	0	0	DIR	Value(2:0)			1	0	CNO(2:0)	

Monitor Transmit FIFO Empty (MTFE)	0	0	0	0
Monitor Abort (MAB)	0	0	0	1
Monitor Receive FIFO Full (MRFF)	1	0	1	0
Monitor End of Message (MEM)	1	0	0	0
Active Monitor Channel Found (AMCF)	1	1	0	0

	15								7							0
IOMM_IV	Received Byte1								Received Byte0							

DIR **Interrupt upon Receive or Transmit Data Direction**

‘0’: Transmit

‘1’: Receive

CNO(2:0) **Channel Number**

8.3.2 C/I Interrupt Vector

	31														16
IOMCI_IV	1	0	0	1	0	0	0	1	0	0	0	0	1	CNO(2:0)	

	15													0
IOMCI_IV	0	0	0	0	0	0	0	0	0	0	0	0	CIC(3:0)	

CNO(2:0) **Channel Number**

CIC **Current Indication Code (C/I Value)**

9 General Purpose Port

This 16-bit port may be used for general purpose I/O. It also provides alternate functionality to support LBI and SSC operations.

The General Purpose Bus consists of two separate 8-bit ports, that can be individually selected to be general purpose or to provide the alternate function, i.e., each port line has a programmable alternate input or output function associated with it. The alternate function of the low 8-bit port is to provide DMA support pins (selected by setting Configuration register's bit field CONF.LBI), and the alternate function of high 8-bit port is to provide Synchronous Serial Control (SSC) interface support (selected by setting Configuration register's bit field CONF.SSC).
Additionally, if CONF.LBI is selected, the DMA Tx/Rx acknowledge pins may be combined for each channel A/B. This is performed by setting LBI Configuration register's bit field LCONF.CDP.
Note that the pin configuration for general purpose or alternate function mode is shown in **Figure 73** on **Page 212**.

Within each 8-bit port, all input/output lines are individually (bit-wise) programmable as inputs or outputs via the General Purpose Bus Direction register GPDIR.

The General Purpose Bus is a true bi-directional port which is switched to high impedance state when configured as input. The output drivers of the port are push/pull drivers. The logic level of a pin is clocked into the input latch once per state time, regardless whether the port is configured as input or output.

A write operation to a general purpose pin configured as an input (GPDIR.x = '0') causes the value to be written into the port output latch, while a read operation returns the latched state of the pin itself. A read-modify-write operation reads the value of the pin, modifies it, and writes it back to the output latch.

Writing to a general purpose pin configured as an output (GPDIR.x = '1') causes the output latch and the pin to have the written value, since the output buffer is enabled. Reading this pin returns the value of the output latch. A read-modify-write operation reads the value of the output latch, modifies it, and writes it back to the output latch, thus also modifying the level at the pin.

If an alternate output function of a General Purpose Bus pin is to be used, the direction of this pin must be programmed for output (GPDIR.x = '1'); otherwise the pin remains in high-impedance state and is not effected by the alternate output function. The respective port latch should hold a '1', because its output is ANDed with the alternate output data.

If an alternate input function of a General Purpose Bus pin is used, the direction of the pin must be programmed for input (GPDIR.x = '0'), if an external device is driving the pin. On reset, the input direction is default. If no external device is connected to the pin, however, the direction for this pin may also be set to output. In this case, the pin reflects the state of the port output latch. Thus, the alternate input function reads the value stored

in the port output latch. This can be used for testing purposes to allow a software trigger of an alternate input function by writing to the port output latch.

There is one basic structure for all port lines providing only an alternate **input** function. Port lines providing only an alternate **output** function, however, have different structures due to the way the direction of the pin is switched and depending on whether or not the pin is in alternate function mode accessible by user software.

All port lines that are not used for these alternate functions may be used as general purpose I/O lines. in order to avoid undesired transitions on the output pins when using port pins for general purpose output, the initial output value should be written to the port latch prior to enabling the output drivers.

Note that the General Purpose Bus has its own interrupt vector associated with it, which is generated when an interrupt of a peripheral on this bus is detected. Interrupts on the General Purpose Bus are enabled by setting Configuration register's bit CONF.GIEN.

General Purpose Bus Interrupt Vector

	31								16
GP_IV	1	0	0	0	0	1	0	1	00_H

	15	0
GP_IV	GPH(7:0)	GPL(7:0)

GPH **GP High Byte Interrupt Vector**

CONF.SSC = '0': Contains the user defined interrupt information of the high byte port of the General Purpose Bus (pins GP(15:8)).

CONF.SSC = '1': 00_H

GPL **GP Low Byte Interrupt Vector**

CONF.LBI = '0': Contains the user defined interrupt information of the low byte port of the General Purpose Bus (pins GP(7:0)).

CONF.LBI = '1': 00_H

10 Reset and Initialization

10.1 Reset

An external low signal on $\overline{\text{RST}}$ resets the MUNICH32X. It immediately switches all outputs to the high impedance state, with exception of pin $\overline{\text{TXDEN}}$, which is driven high in that case.

The alternate function pins are set to general purpose pins. All registers are set to their reset values. All functions are initialized to known states.

After $\overline{\text{RST}}$ is deasserted and all clock/frame signals are active, the subfunctions PCI, Global Registers, Serial PCM Core, LBI EBC/Mailbox, LBI DMSM, LBI DMAC, SSC and IOM®-2 are in reset or standby mode.

Status after Hardware Reset

Table 20
Subfunction Status after Hardware Reset

Subfunction	Reset Status
PCI interface	standby PCI config space registers accessible
Global Registers	standby Slave registers accessible; General Purpose Bus enabled!
Serial PCM Core	standby Slave registers accessible
LBI EBC/Mailbox/DMSM	reset Slave registers fixed to reset values
LBI DMA Controller	standby Slave registers accessible
SSC	standby Slave registers accessible
IOM®-2	standby Slave registers accessible; RXCLK (DCL), RXD (DD) and RSP (FSC) are connected; TXD (DU) is connected to Serial PCM core only

Accessing Subfunctions after Hardware Reset

The functions LBI EBC, LBI DMSM, SSC, and IOM®-2 must be made accessible by additional programming before they can be used.
Note that LBI functions are only available, if the DEMUX pin is set to ‘0’ (PCI Interface Mode).
The **Table 17** shows the programming required to access the MUNICH32X subfunctions.

Table 21
Programming after Hardware Reset

Subfunction	Required Register Programming
LBI EBC/ Mailbox	• bit LCONF.$\overline{\text{EBCRES}}$ must be set to ‘1’
LBI DMSM	• bit CONF.LBI must be set to ‘1’ • bit LCONF.$\overline{\text{LBIRES}}$ must be set to ‘1’
SSC	• bit CONF.SSC must be set to ‘1’ • GPDIR and GPDATA register values must be programmed to the desired I/O function (refer to SSC section)
IOM®-2	• bit CONF.IOM must be set to ‘1’ • special settings of registers MODE1 and MODE2 are necessary: bit fields MODE1.PCM(3:0) must be programmed to 8_H bit fields MODE1.TTS(2:0) must be programmed to 0_H bit fields MODE1.RTS(2:0) must be programmed to 0_H bit fields MODE1.TBS(2:0) must be programmed to 3_H bit fields MODE1.RBS(2:0) must be programmed to 5_H bit MODE2.RXF must be programmed to ‘0’ bit MODE2.TXR must be programmed to ‘0’ bit MODE2.RSF must be programmed to ‘0’ bit MODE2.TSF must be set to ‘1’ • bit IOMCON1.CLR must be set to ‘1’ bit IOMCON1.ENIH must be set to ‘1’

Software Reset

For the subfunctions, the states described in **Table 17** can also be reached by programming bits (19:17) in PCIRES register (refer to **Chapter 5.1.2**); i.e., setting those bits to ‘1’ has the same effect for the subfunction as an external $\overline{\text{RST}}$ = low. Resetting the bits to ‘0’ corresponds to deasserting $\overline{\text{RST}}$ for that function.

Channel Status

Channel processing is deactivated. After reset, all buffers are empty and no buffer size is allocated to the channels. The DMA controller state is set to the hold condition. The descriptor and data pointers remain at a random value.

On reset, the bits RO and TO are set to '1', whereas RA and TA are set to '0' for all logical channels. All time slots are connected to the logical channel 0 and the following configuration is set:

Action Specification

LOC = LOOP = LOOPI = 0

Time Slot Assignment

fill/mask = 00_H, i.e., all bits masked/set to '1'

RTI, TTI = 0

channel number = 00_H

Channel Specification

MODE = 00, i.e. TMA

FA = 0

IFTF = 0

CRC = 0

INV = 0

TRV = 00,

RO = 1

RA = 0

TO = 1

TA = 0

TH = 0

Transmit Descriptor

FNUM = 00_H, i.e. shared flags in HDLC, only eight zero bits between sent frames for TMB.

The E-, S-, X-bits are all set to zero internally by the reset. The receiver is set into the ITF/IDLE state for all channels, i.e. it assumes that on the line there are '1's as interframe time-fill for HDLC.

General

After reset, the valid PCM mode is T1/DS1 × 24-channel 1.536 Mbit/s PCM(3:0) = ‘0000’ in MODE1 register). The value for maximum frame length is ‘0’ (MODE1.MFL).
Note that in MUNICH32 (PEB 20320) both values are definied in the action specification.

10.2 Initialization

After reset, the MUNICH32X remains in the default state until the host processor generates an action request. The initialization sequence is defined in the action specification. The sequence can be split up into individual procedures for each channel or one procedure to initialize all channels simultaneously. For all procedures, the time slot assignment and the selected channel specifications are loaded into internal MUNICH32X memory. To prevent malfunction, the initialization of the link lists and the allocation of the buffer size to the channels has to be specified before transmission may be initiated. MUNICH32X assumes that timeslot 0 starts on the Rx and Tx lines. They may be resynchronized by 2 rising edges of TSP and RSP, respectively.

Before this resynchronization the host should neither remove RO = 1 or TO = 1 nor set LOOP or LOOPI to ‘1’ for any logical channel. During this time any incoming data is ignored, the Tx line is tristated.

For each action service the device first reads the control start address in the Control and Configuration Block (CCB), which is pointed to by the contents of the CCBA register.

The values of the CCBA register can be changed during operation.

11 Slave Register Descriptions

11.1 Register Set Overview

The following table provides a quick reference to the complete MUNICH32X slave register set. A detailed description of the registers related to each functional block can be found in **Chapter 11.2**.

Note that the PCI Configuration Space Registers are listed in **Section 5.1.2**.

Table 22
MUNICH32X Slave Register Set

Register Name		Read/ Write	Offset to PCI BAR1
Configuration	CONF	R/W	00_H
Command	CMD	W	04_H
Status Status Acknowledge	STAT STACK	R W	08_H
Interrupt Mask	IMASK	R/W	$0C_H$
Reserved	–	–	10_H
Peripheral Interrupt Queue Base Address	PIQBA	R/W	14_H
Peripheral Interrupt Queue Length	PIQL	R/W	18_H
Reserved	–	–	$1C_H$
Mode1	MODE1	R/W	20_H
Mode2	MODE2	R/W	24_H
CC Block Indirect Address	CCBA	R/W	28_H
Tx Poll	TXPOLL	R/W	$2C_H$
Tx Interrupt Queue Base Address	TIQBA	R/W	30_H
Tx Interrupt Queue Length	TIQL	R/W	34_H
Rx Interrupt Queue Base Address	RIQBA	R/W	38_H
Rx Interrupt Queue Length	RIQL	R/W	$3C_H$
LBI Configuration	LCONF	R/W	40_H
LBI CC Block Indirect Address	LCCBA	R/W	44_H
Reserved	–	–	48_H
LBI Start Transfer	LTRAN	W	$4C_H$
LBI Tx Interrupt Queue Base Address	LTIQBA	R/W	50_H

Table 22
MUNICH32X Slave Register Set (cont'd)

Register Name		Read/ Write	Offset to PCI BAR1
LBI Tx Interrupt Queue Length	LTIQL	R/W	54_H
LBI Rx Interrupt Queue Base Address	LRIQBA	R/W	58_H
LBI Rx Interrupt Queue Length	LRIQL	R/W	$5C_H$
LBI Indirect External Config 0	LREG0	R/W	60_H
LBI Indirect External Config 1	LREG1	R/W	64_H
LBI Indirect External Config 2	LREG2	R/W	68_H
LBI Indirect External Config 3	LREG3	R/W	$6C_H$
LBI Indirect External Config 4	LREG4	R/W	70_H
LBI Indirect External Config 5	LREG5	R/W	74_H
LBI Indirect External Config 6	LREG6	R/W	78_H
LBI Status	LSTAT	R	$7C_H$
General Purpose Bus Direction	GPDIR	R/W	80_H
General Purpose Bus Data	GPDATA	R/W	84_H
General Purpose Bus Open Drain	GPOD	R/W	88_H
Reserved	–	–	$8C_H$
SSC Control	SSCCON	R/W	90_H
SSC Baud Rate Generator	SSCBR	R/W	94_H
SSC Tx Buffer	SSCTB	R/W	98_H
SSC Rx Buffer	SSCRB	R	$9C_H$
SSC Chip Select Enable	SSCCSE	R/W	$A0_H$
SSC Interrupt Mask Register	SSCIM	R/W	$A4_H$
Reserved	–	–	$A8_H$
Reserved	–	–	AC_H
IOM®-2 Control 1	IOMCON1	R/W	$B0_H$
IOM®-2 Control 2	IOMCON2	R/W	$B4_H$
IOM®-2 Status	IOMSTAT	R	$B8_H$
IOM®-2 C/I Tx Channels 0-3	IOMCIT0	R/W	$C0_H$
IOM®-2 C/I Tx Channels 4-7	IOMCIT1	R/W	$C4_H$
IOM®-2 C/I Rx Channels 0-3	IOMCIR0	R	$C8_H$

Table 22
MUNICH32X Slave Register Set (cont'd)

Register Name		Read/ Write	Offset to PCI BAR1
IOM®-2 C/I Rx Channels 4-7	IOMCIR1	R	CC_H
IOM®-2 Tx Monitor	IOMTMO	R/W	$D0_H$
IOM®-2 Rx Monitor	IOMRMO	R	$D4_H$
Reserved	–	–	$D8_H$
Reserved	–	–	DC_H

Register Name		Read/ Write	Offset to PCI BAR1 (from PCI side)/ Pins LA(2:0) (from LBI side)
Mailbox Command	MBCMD	R/W	$E0_H/0_H$
Mailbox Data 1	MBDATA1	R/W	$E4_H/1_H$
Mailbox Data 2	MBDATA2	R/W	$E8_H/2_H$
Mailbox Data 3	MBDATA3	R/W	$EC_H/3_H$
Mailbox Data 4	MBDATA4	R/W	$F0_H/4_H$
Mailbox Data 5	MBDATA5	R/W	$F4_H/5_H$
Mailbox Data 6	MBDATA6	R/W	$F8_H/6_H$
Mailbox Data 7	MBDATA7	R/W	$FC_H/7_H$

Register Name		Read/ Write	Offset to PCI BAR2
PCI Direct LBI Access	PCILBI (XXXX)	R/W	0000_H - $FFFF_H$

Note: Read accesses to unused register addresses always returns the value '0'; the same applies to unused bit fields. An unused register cannot be written to.

11.2 Register Bit Field Definitions

11.2.1 MUNICH32X Global Registers

This section contains descriptions of all global MUNICH32X slave registers.

Configuration Register (CONF)

Access : read/write

Offset Address : 00_H

Reset Value : 00000000_H

	31															16
CONF	0000_H															

	15															0
CONF	0	0	0	0	0	0	0	CST	GPIE	SSC	LBI	IOM	LCD(1:0)		DBE	LBE

GPIE **General Purpose Bus Interrupt Enable**

‘1’: Interrupts on the General Purpose Bus are enabled.

CST **Clock Source Timer**

‘0’: The clock source is the LBI clock.

‘1’: The clock source is the RSP pin.

SSC **SSC Mode Select**

‘0’: These pins provide the high byte of the General Purpose Bus.
(refer to **Figure 73** on **Page 212**)

‘1’: The MUNICH32X provides the control functions of the Synchronous Serial Communication (SSC) interface via the pins specified below.

Pin No.	General Purpose Bus Pin	SSC Mode Function
100	GP15	MCLK
99	GP14	MTSR
98	GP13	MRST
97	GP12	not used
91	GP11	$\overline{\text{MCS0}}$
90	GP10	$\overline{\text{MCS1}}$
89	GP9	$\overline{\text{MCS2}}$
88	GP8	$\overline{\text{MCS3}}$

Important Note: Also, if the SSC Mode is selected, the values of General Purpose Bus Direction Register GPDIR(15:8) and General Purpose Bus Open Drain Register GPOD(15:8) must be programmed according to the desired function.

LBI **LBI Mode Select**

‘0’: These pins provide the low byte of the General Purpose Bus.

(refer to **Figure 73** on **Page 212**)

‘1’: The MUNICH32X provides the DMA support functions of the Local Bus Interface (LBI) via the pins specified below.

Pin No.	General Purpose Bus Pin	LBI DMA Mode Function
85	GP7	DRQTA
84	GP6	DRQRA
83	GP5	DRQTB
82	GP4	DRQRB
81	GP3	$\overline{\text{DACKTA}}$
80	GP2	$\overline{\text{DACKTB}}$
79	GP1	$\overline{\text{DACKRA}}$
78	GP0	$\overline{\text{DACKRB}}$

Note: After reset, the MUNICH32X operates in the General Purpose Bus mode.

The General Purpose Bus Direction Register GPDIR allows PCI host software to change the direction of the corresponding pins; after reset, all pins are inputs.

The General Purpose Bus Data Register GPDATA is a bi-directional register used for data transfer to/from devices connected to the pins of the General Purpose Bus.

IOM **IOM®-2 Mode Select**

‘1’: IOM®-2 Mode is selected (only applicable in 2 or 4 Mbit PCM mode)

LCD **LBI Timer/Clock Division**

The LBI operating clock is either identical to the PCI Clock, or divided by 2, 4 or 16 according to the following coding:

LCD(1:0)	LBi Clock Rate
00	PCI Clock
01	PCI Clock devided by 2
10	PCI Clock devided by 4
11	PCI Clock devided by 16

DBE **Demux Burst Enable**

If the DEMUX input pin = ‘1’, this bit field is valid, otherwise it is invalid. In De-multiplexed mode, if DBE is set to ‘1’, the MUNICH32X will perform Master read & write burst of descriptors up to a length of four.

LBE **Little/Big Endian Byte Swap**

‘0’: Data will be presented to the PCI bus or de-multiplexed bus in little-endian format for Rx operation, and data is expected in little-endian format for Tx operation.

‘1’: Big endian format is used.

Note that this applies only to data, the on-chip registers and the descriptors in CCB in host memory are in little endian format.

Command Register (CMD)

Access : write

Offset Address : 04_H

Reset Value : 00000000_H

	31															16
CMD	TIMV(15:0)															

	15															0
CMD	0	0	0	0	0	0	0	0	0	0	0	TIMR	0	0	ARLBI	ARPCM

TIMV **Timer Value**

A 16-bit value, programmed by host PCI software, which may be loaded into an on-chip down counter (via the TIMR bit field), to provide periodic interrupt generation to the host PCI system. The MUNICH32X generates an interrupt to the host PCI system with a cycle of 'TIMV + 2 clock cycles'.

TIMR **Timer Run**

'0': The timer is stopped.

'1': The MUNICH32X loads the value in the TIMV bit fields into the on-chip down counter. The timer runs at the frequency selected in bit field CONF.CST.

ARLBI **Action Request LBI**

'1': The MUNICH32X will immediately initiate an LBI action request, if no other action request is currently active. Bit is self resetting and will be reset by the LBIF or LBIA interrupt in Status Register STAT.

ARPCM **Action Request Serial PCM Core**

'1': The MUNICH32X will immediately initiate a serial PCM core action request, if no other action request is currently active. Bit is self resetting and will be reset by the PCMF or PCMA interrupt in Status Register STAT.

Status Register (STAT)

Access : read

Offset Address : 08_H

Reset Value : 00000000_H

	31															16
STAT	0000_H															

	15															0
STAT	PTI	PRI	LTI	LRI	IOMI	SSCI	LBII	MBI	0	TI	TSPA	RSPA	LBIF	LBIA	PCMF	PCMA

When an interrupt event occurs, the MUNICH32X sets the flag corresponding to the event.

If the interrupt event was non-masked via the Interrupt Mask register IMASK, the MUNICH32X will generate an interrupt to the PCI host system (i.e., it asserts the PCI $\overline{\text{INTA}}$ signal).

The host PCI system software may deassert the PCI $\overline{\text{INTA}}$ signal by writing a ‘1’ to the appropriate bit field.

PTI **Serial PCM Tx Interrupt**

‘1’: Indicates the MUNICH32X has written status information into the Tx Interrupt Queue in host memory.

PRI **Serial PCM Rx Interrupt**

‘1’: Indicates the MUNICH32X has written status information into the Rx Interrupt Queue in host memory.

LTI **LBI Tx Interrupt**

‘1’: Indicates the MUNICH32X has written status information into the LBI Tx Interrupt Queue in host memory.

LRI **LBI Rx Interrupt**

‘1’: Indicates the MUNICH32X has written status information into the LBI Rx Interrupt Queue in host memory.

IOMI **Peripheral IOM®-2 Interrupt**

‘1’: Indicates the MUNICH32X has written IOM®-2 status information into the Peripheral Interrupt Queue in host memory.

SSCI **Peripheral SSC interrupt**

‘1’: Indicates the MUNICH32X has written SSC status information into the Peripheral Interrupt Queue in host memory.

LBII **Peripheral LBI Interrupt**

‘1’: Indicates the MUNICH32X has written LBI status information into the Peripheral Interrupt Queue in host memory.

MBI **Peripheral Mailbox interrupt**

‘1’: Indicates the MUNICH32X has written Mailbox status information into the Peripheral Interrupt Queue in host memory.

TI **Timer Interrupt**

‘1’: Indicates the count-down timer has underflowed.

TSPA **PCM TSP Asynchronous**

‘1’: PCM frame signal TSP was asynchronous (frame did not match expected clock count)

RSPA **PCM RSP Asynchronous**

‘1’: PCM frame signal RSP was asynchronous (frame did not match expected clock count)

LBIF **LBI Fail**

‘1’: Indicates that LBI action request failed.

LBIA **LBI Acknowledgement**

‘1’: Indicates that LBI action request succeeded.

PCMF **Serial PCM Fail**

‘1’: Indicates that serial PCM core action request failed.

PCMA **Serial PCM Acknowledgement**

‘1’: Indicates that serial PCM action request succeeded.

Note: Bit fields (7:0) are standalone interrupt bits, i.e., they have no corresponding interrupt vector queue entry.

Status Acknowledge Register (STACK)

Access : write

Offset Address : 08_H

Reset Value : 00000000_H

	31															16
STACK	0000_H															

	15															0
STACK	PTI	PRI	LTI	LRI	IOMI	SSCI	LBII	MBI	0	TI	TSPA	RSPA	LBIF	LBIA	PCMF	PCMA

The PCI host system software may deassert the PCI $\overline{\text{INTA}}$ signal (independent of the IMASK bits) by writing a ‘1’ to the appropriate bit field and hence resetting the corresponding bit field in STAT.

Interrupt Mask Register (IMASK)

Access : read/write

Offset Address : $0C_H$

Reset Value : $0000FF7F_H$

	31															16
IMASK	0000_H															

	15															0
IMASK	PTI	PRI	LTI	LRI	IOMI	SSCI	LBII	MBI	0	TI	TSPA	RSPA	LBIF	LBIA	PCMF	PCMA

When set to ‘1’, an event which normally would cause the MUNICH32X to generate an interrupt to the PCI host system will now only trigger the appropriate Status Register bit field (flag) to become set to ‘1’.

Peripheral Interrupt Queue Base Address Register (PIQBA)

Access : write/read

Offset Address : 14_H

Reset Value : 00000000_H

	31															16
PIQBA	PIQBA(31:16)															

	15														1	0
PIQBA	PIQBA(15:2)														0	0

Specifies the host memory Peripheral Interrupt Queue base or start address; this address must be DWORD-aligned.

Peripheral Interrupt Queue Length Register (PIQL)

Access : read/write

Offset Address : 18_H

Reset Value : 00000000_H

	31															16
PIQL	0000_H															

	15								8	7						0
PIQL	0	0	0	0	0	0	0	0	PIQL (7:0)							

Specifies the length of the Peripheral Interrupt Queue in number of DWORDs:

Interrupt Queue length = 4 × PIQL (DWORDs).

The maximum length of 1024 DWORDs is achieved by setting PIQL(7:0) = 00_H.

11.2.2 Serial PCM Core Registers

This section contains descriptions of all serial PCM core slave registers.

Mode1 Register (MODE1)

Access : read/write

Offset Address : 20_H

Reset Value : 00000000_H

	31				16
MODE1	PCM(3:0)	TTS(2:0)	TBS(2:0)	RTS(2:0)	RBS(2:0)

	15			0
MODE1	REN	RID	MFLD	MFL(12:0)

PCM **PCM Transmission Rate**

Specifies the PCM transmission rate used by the Serial PCM core as indicated in the table below:

Bit 31	Bit 30	Bit 29	Bit 28	PCM Data Rate
0	0	0	0	1.536 Mbit/s
0	1	0	0	1.544 Mbit/s
0	1	0	1	3.088 Mbit/s
0	1	1	0	6.176 Mbit/s
1	0	0	0	2.048 Mbit/s
1	0	0	1	4.096 Mbit/s
1	0	1	0	8.192 Mbit/s

Note: All other values are illegal and will produce undefined results.

TBS/RBS **Tx/Rx Bit Shift**

Specifies the position of the Tx/Rx bits relative to the synch. pulses in transmit and receive direction.
Important note: In order to be compatible to the MUNICH32 (i.e., no bit shift), these bit fields must be programmed to ‘4’.

Bit 24	Bit 23	Bit 22	Shift of Tx Bit relative to Tx Sync Pulse	Bit 18	Bit 17	Bit 16	Shift of Rx Bit relative to Rx Sync Pulse
0	0	0	– 4	0	0	0	– 4
0	0	1	– 3	0	0	1	– 3
0	1	0	– 2	0	1	0	– 2
0	1	1	– 1	0	1	1	– 1
1	0	0	0	1	0	0	0
1	0	1	1	1	0	1	1
1	1	0	2	1	1	0	2
1	1	1	3	1	1	1	3

Note: To be consistent with the MUNICH32, PEB 20320, TBS(2:0) and RBS(2:0) must be programmed to '4'.

TTS/RTS **Tx/Rx Time Slot**

The MUNICH32X uses only valid timeslots according to the following table (n = 0 … 31; k = 1 in 2-Mbit mode, k = 2 in 4-Mbit mode;
k = 4 in 8-Mbit mode
n = 0 … 23; k = 1 in 1.5-Mbit mode, k = 2 in 3-Mbit mode,
k = 4 in 6-Mbit mode):

Bit 27	Bit 26	Bit 25	Valid Timeslots Transmit	Bit 21	Bit 20	Bit 19	Valid Timeslots Receive
0	0	0	k × n	0	0	0	k × n
0	0	1	(k × n) + 1	0	0	1	(k × n) + 1
0	1	0	(k × n) + 2	0	1	0	(k × n) + 2
0	1	1	(k × n) + 3	0	1	1	(k × n) + 3
1	0	0	reserved	1	0	0	reserved
1	0	1	reserved	1	0	1	reserved
1	1	0	reserved	1	1	0	reserved
1	1	1	reserved	1	1	1	reserved

Note: In 3-Mbit or 4-Mbit PCM mode, only the first 2 options (bit fields 21, 20, 27 and 26 equal to '0') are valid. In 6-Mbit or 8-Mbit PCM mode, all 4 options are valid.

REN **Rx Enable**

‘0’: Serial PCM core Rx operation is disabled.

‘1’: Serial PCM core Rx operation is enabled.

RID **Rx Interrupt Disable**

‘0’: A serial PCM Rx interrupt event will cause the MUNICH32X to generate an interrupt to the host PCI system, if the event was non-masked via the Interrupt Mask Register.

‘1’: A serial PCM Rx interrupt event will not cause the MUNICH32X to generate an interrupt to the host PCI system.

MFLD **Maximum frame Length Check Disabled**

‘0’: The MFL check is enabled.

‘1’: The MFL check is disabled.

MFL **Maximum Frame Length**

Maximum size of a received frame in HDLC, TMB and TMR mode (up to 8192 bytes). A received frame is aborted and an interrupt is generated if the size of a received frame exceeds the MFL value. MFL applies to all channels.

MODE2 Register (MODE2)

Access : read/write

Offset Address : 24_H

Reset Value : 00000000_H

	31															16
MODE2	0000_H															

	15															0
MODE2	0	0	0	0	0	0	0	HPOLL	SPOLL	LSIM	REIM	TEIM	RXF	TXR	RSF	TSF

HPOLL **Hold Poll**

The MODE2.HPOLL bit field provides capability for the MUNICH32X to be configured for Tx polling identical to that of the MUNICH32. In this case (i.e. HPOLL = '1' & TXPOLL(31:0) = 00000000_H), the MUNICH32X checks the status of the the TxHOLD bit field for each time slot assigned to the particular channel. When the TxHold bit field is reset, the MUNICH32X immediately resumes transmission (see TXPOLL register).

SPOLL **Slow Poll**

The MODE2.SPOLL bit field (slow poll) provides capability for the MUNICH32X to be configured for a Tx poll rate determined by the Tx synchronization pulses divided by 8 (125 µs × 8 = 1 ms). In this case (i.e. SPOLL = '1' & TXPOLL.$\overline{POLL}$(31:0) = $FFFFFFFF_H$), Tx polling per channel is performed every 8th PCM frame with four groups of interleaved DMA channels, as shown below:

Frame	DMA channels
0	0, 8, 16, 24
1	1, 9, 17, 25
n	n, n + 8, n + 16, n + 24
7	7, 15, 23, 31 (see TXPOLL register).

*Note: It is recommended not to use the Slow Poll option in combination with Sub Channeling on any one or more channels (for description of Sub Channeling refer to **Chapter 12.5**).*

LSIM **Late Stop Interrupt Mask**

'1': A Late Stop event (refer to serial PCM core interrupt vector description) will not cause the MUNICH32X to generate an interrupt to the host PCI system.

REIM **Rsync Error Interrupt Mask**

'1': An Rsync error condition will not cause the MUNICH32X to generate an interrupt to the host PCI system.

TEIM **Tsync Error Interrupt Mask**

'1': An Tsync error condition will not cause the MUNICH32X to generate an interrupt to the host PCI system.

RXF **Rx Data Falling Edge**

'0': RXD is sampled on the rising edge of RXCLK.

'1': RXD is sampled on the falling edge of RXCLK.

TXR **Tx Data Rising Edge**

‘0’: TXD is sampled on the falling edge of TXCLK.

‘1’: TXD is sampled on the rising edge of TXCLK.

RSF **RSP Falling Edge**

‘0’: RSP is sampled on the rising edge of RXCLK.

‘1’: RSP is sampled on the falling edge of RXCLK.

TSF **TSP Falling Edge**

‘0’: TSP is sampled on the falling edge of TXCLK.

‘1’: TSP is sampled on the rising edge of TXCLK.

CC Block Indirect Address Register (CCBA)

Access : read/write

Offset Address : 28_H

Reset Value : 00000000_H

	31	16
CCBA	CCBA(31:16)	

	15	0	
CCBA	CCBA(15:2)	0	0

The CC Block Indirect Address Register points to the location in host memory which contains the actual base address pointer to the Control and Configuration Block (for serial PCM core DMA). This address must be DWORD-aligned.

Tx Poll Register (TXPOLL)

Access : read/write

Offset Address : $2C_H$

Reset Value : 00000000_H

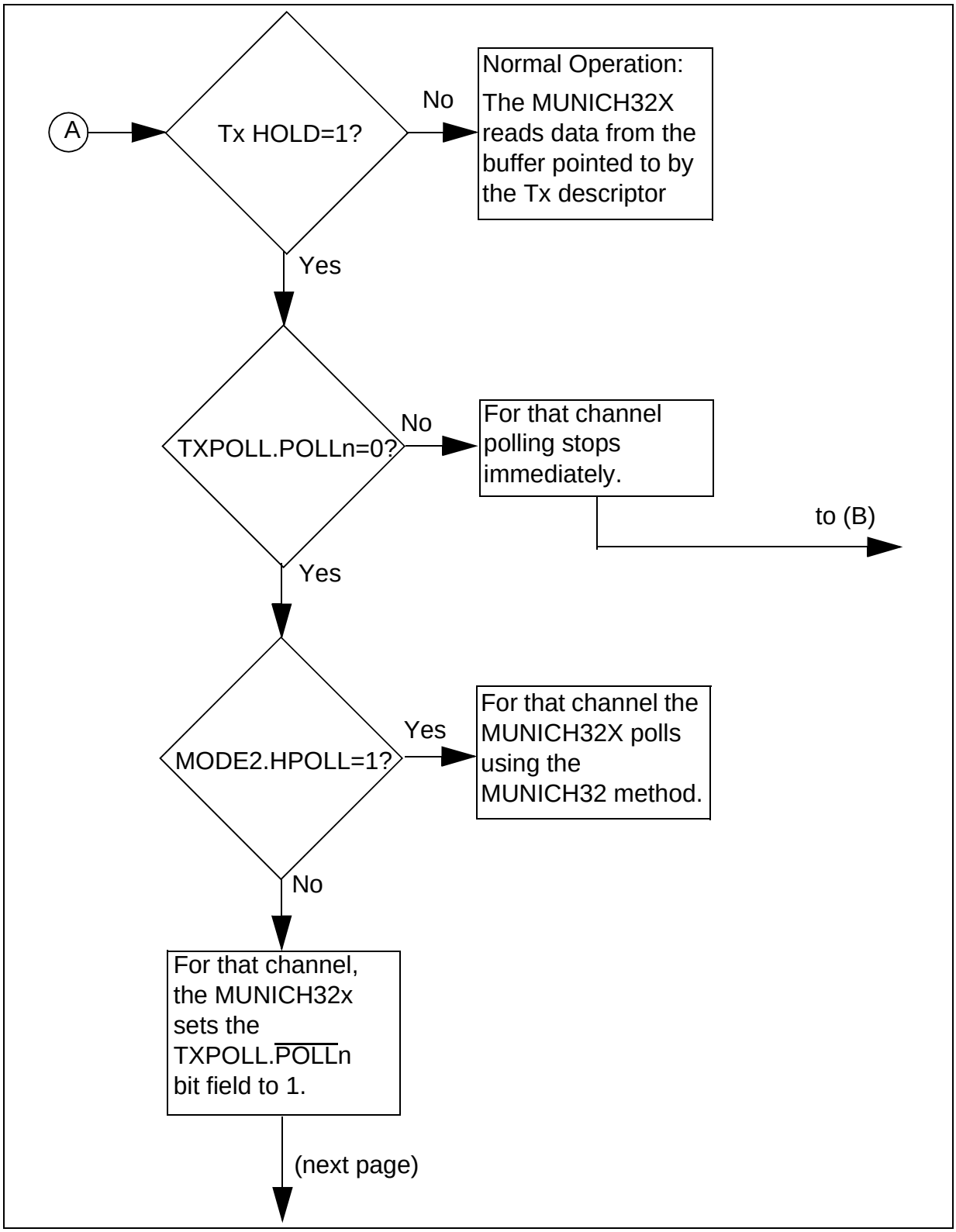
	31	16
TXPOLL	$\overline{POLL}$(31:16)	

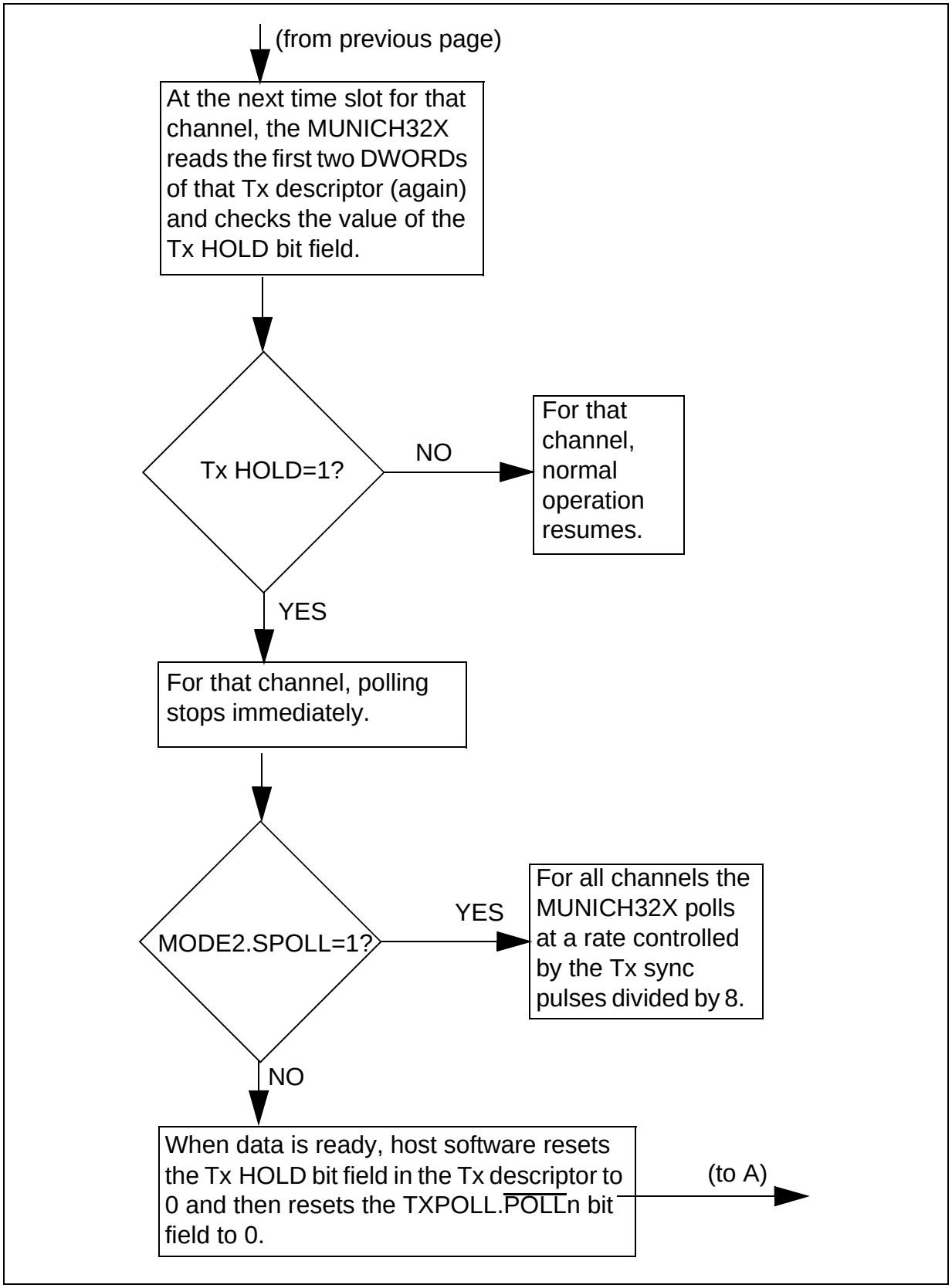
	15	0
TXPOLL	$\overline{POLL}$(15:0)	

The Tx Poll Register provides an asserted low $\overline{POLLn}$ bit field per Serial PCM core channel that allows host software to configure the MUNICH32X to respond in one of three ways to a Tx idle condition, depending upon the state of the TXPOLL.$\overline{POLLn}$ bit field (additionally, the states of the MODE2.SPOLL and MODE2.HPOLL bit fields are evaluated in the Tx polling process).
The flow diagram in **Figure 71** illustrates the Tx polling process.

1. Most applications will set the Tx HOLD bit field in Tx descriptor and have both the MODE2.SPOLL bit field and the MODE2.HPOLL bit field equal to zero. With this configuration, the MUNICH32X will disable polling either immediately or after one poll (depending upon the state of the TXPOLL.$\overline{POLLn}$ bit field). If TXPOLL.$\overline{POLLn}$ = 1, Tx polling stops immediately, while if TXPOLL.$\overline{POLLn}$ = 0, a single poll is performed. Software may enable a single poll for a particular channel by resetting the appropriate TXPOLL.$\overline{POLLn}$ bit field. With this technique, bus utilization of idle channels is very low, and startup from a temporary idle state is resumed after a simple write operation to the MUNICH32X's TXPOLL register.
2. If both the TXPOLL.$\overline{POLLn}$ bit field and the MODE2.SPOLL bit field are set, and the MUNICH32X detects the Tx HOLD bit field set for that particular channel, then the MUNICH32X will perform polling for all channels at a rate controlled by the Tx synch pulses divided-by-8. Operation of this mode is described in detail in the SPOLL (slow poll) section of the MODE2 Register.
3. If the TXPOLL.$\overline{POLLn}$ bit field is reset, while the MODE2.HPOLL bit field is set, and the MUNICH32X detects the Tx HOLD bit field set for that particular channel, then the MUNICH32X will perform Tx polling identical to that of the MUNICH32 (PEB 20320) for that particular channel. In this case, the MUNICH32X checks the status of the Tx HOLD bit field for each time slot assigned to this channel. In this way, if the bit has been cleared, the MUNICH32X will immediately resume transmission. Although this method is simpler (in concept) for the software design, it causes the MUNICH32X to consume higher than normal bus bandwidth. For this reason, this is the least desirable

of the methods. Operation of this mode is described in detail in the HPOLL section of the MODE2 register.





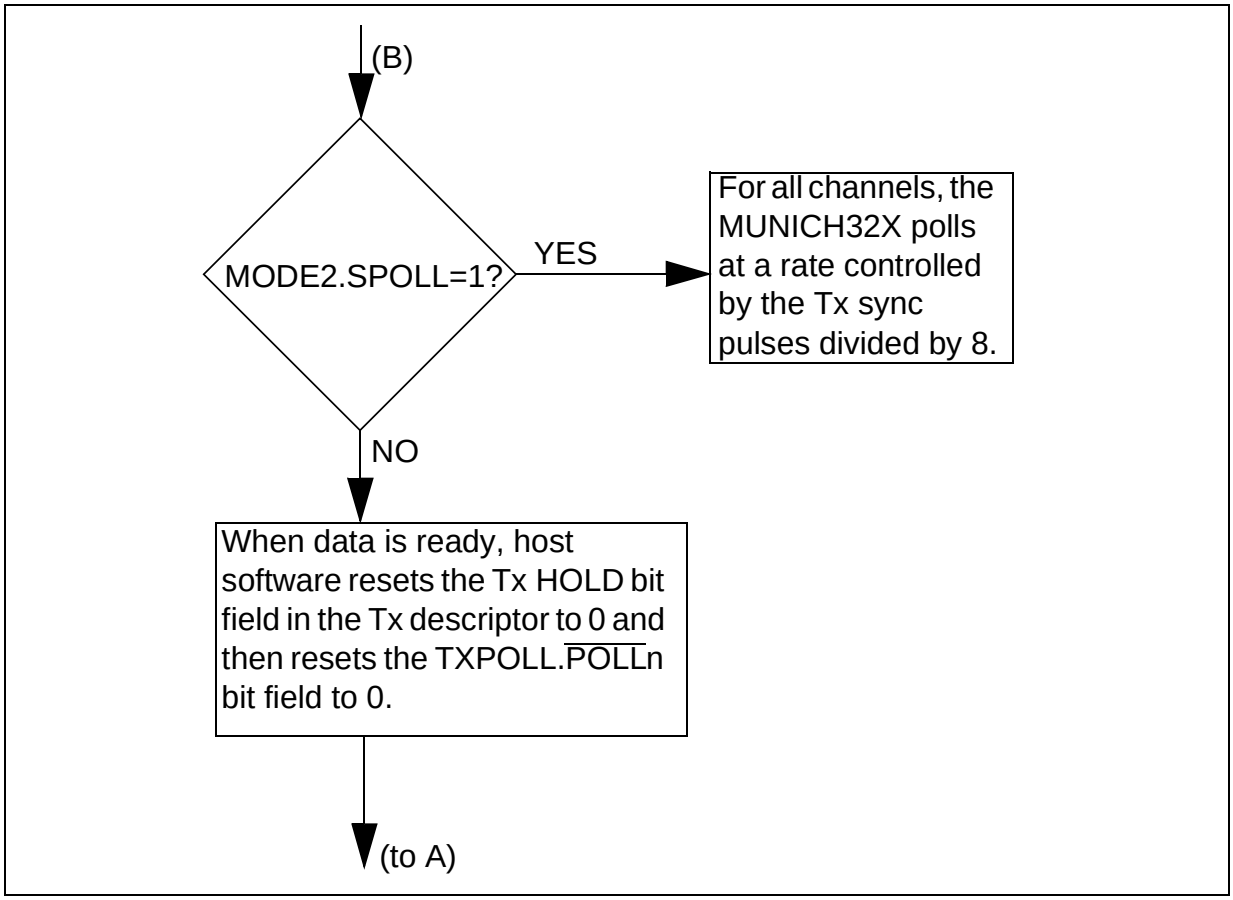


Figure 71
Tx Polling Procedure

Interupt Queue Registers

The MUNICH32X provides data status information (of the serial PCM core and the LBI peripheral) to the host PCI system via dedicated Rx and Tx Interrupt Queues in host memory, which have the same structure as shown in **Figure 72**. Non-data status information is provided via the serial PCM core Status Register STAT and the LBI status register LSTAT.

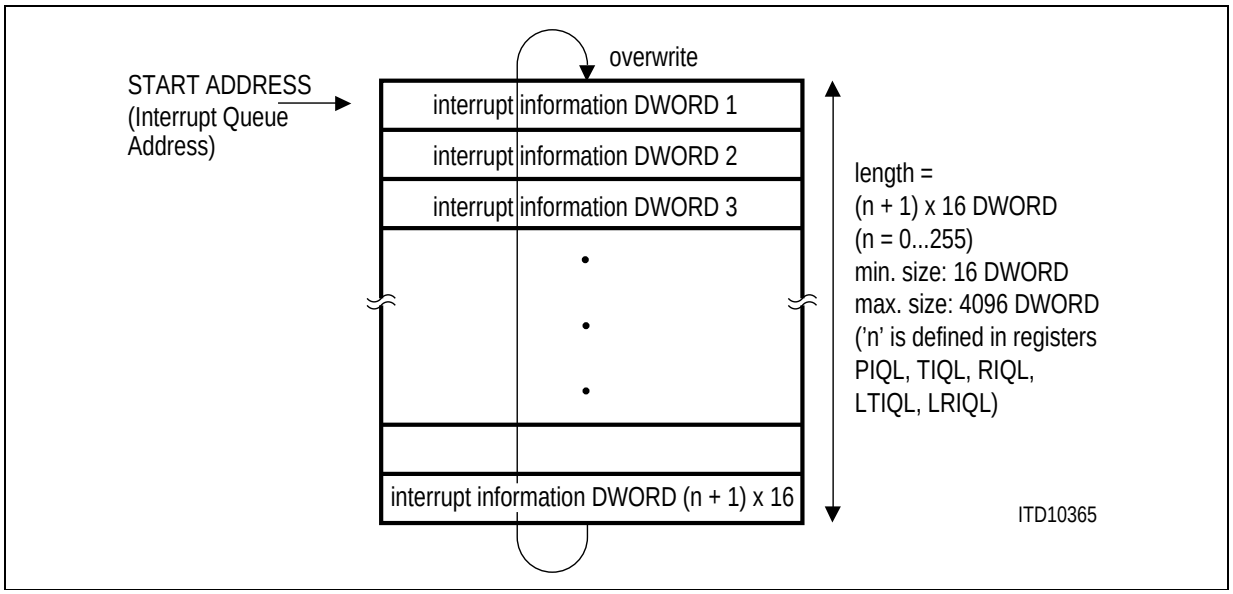


Figure 72
Interrupt Queue Structure

For each interrupt event, the MUNICH32X writes status information into the appropriate interrupt queue, increments the pointer to the next address in this block and generates an interrupt to the host PCI system, if non-masked.

It is the responsibility of the host PCI software to read the status information out of the appropriate interrupt queue. When the MUNICH32X arrives at the end of an interrupt queue, it will jump to the start address of that interrupt block again and overwrite the previous information.

If the start address or the length of a queue is changed during operation, an action request with the 'IA' bit set has to be initiated.

Tx Interrupt Queue Base Address (TIQBA)

Access : read/write

Offset Address : 30_H

Reset Value : 00000000_H

	31															16
TIQBA	TIQBA(31:16)															

	15														2	1	0
TIQBA	TIQBA(15:2)														0	0	

Specifies the host memory serial PCM core Tx Interrupt Queue base or start address; this address must be DWORD-aligned.

Tx Interrupt Queue Length (TIQL)

Access : read/write

Address relative to BAR1 : 34_H

Reset Value : 00000000_H

	31															16
TIQL	0000_H															

	15								8	7						0
TIQL	0	0	0	0	0	0	0	0	TIQL(7:0)							

Specifies the DWORD count of the serial PCM core Tx Interrupt Queue in host memory. The maximum size of the Queue is 4096 DWORDs ((n + 1) × 16 DWORDs, where n = TIQL(7:0); refer to **Figure 72**).

Rx Interrupt Queue Base Address (RIQBA)

Access : read/write

Offset Address : 38_H

Reset Value : 00000000_H

	31															16
RIQBA	RIQBA(31:16)															

	15														2	1	0
RIQBA	RIQBA(15:2)														0	0	

Specifies the host memory serial PCM core Rx Interrupt Queue base or start address; this address must be DWORD-aligned.

Rx Interrupt Queue Length (RIQL)

Access : read/write

Offset Address : $3C_H$

Reset Value : 00000000_H

	31															16
RIQL	0000_H															

	15								8							0
RIQL	0	0	0	0	0	0	0	0	RIQL(7:0)							

Specifies the DWORD count of the serial PCM core Rx Interrupt Queue in host memory. The maximum size of the Queue is 4096 DWORDs ((n + 1) × 16 DWORDs, where n = RIQL(7:0); refer to **Figure 72**).

11.2.3 LBI Registers

This section contains descriptions of all LBI registers.

LBI Configuration Register (LCONF)

Access: read/write

Offset address: 40_H

Reset Value: 00600000_H

	31												16
LCONF	IPA	DCA	DCB	MDA	MDB	SDA	DID	CDP	0	$\overline{\text{EBCRES}}$	$\overline{\text{LBIRES}}$	DV(2:0)	0	0

	15											0
LCONF	0	0	0	HE1	HE2	SPINT	EALE	HDEN	BTYP(1:0)	RDEN	ABM	MCTC(3:0)

MCTC **Memory Cycle Time Control**

(Number of memory cycle time wait states)

0 0 0 0: 15 waitstates (Number = 15 - <MCTC>)

…

1 1 1 1: No waitstates

BTYP **External Bus Configuration**

0 0: 8-bit De-multiplexed Bus
0 1: 8-bit Multiplexed Bus
1 0: 16-bit De-multiplexed Bus
1 1: 16-bit Multiplexed Bus

RDEN **$\overline{\text{LRDY}}$ Input Enable**

‘0’: External bus cycle is controlled by bit field MCTC only
‘1’: External bus cycle is controlled by the bit field MCTC and signal $\overline{\text{LRDY}}$

HDEN **HOLD Enable**

‘0’: Bus arbitration ignored
‘1’: LBI bus arbitration using $\overline{\text{LHOLD}}$, $\overline{\text{LHLDA}}$, $\overline{\text{LBREQ}}$ enabled

EALE **Extended ALE**

‘0’: Single LBI clock ALE pulse width

‘1’: 1.5 LBI clocks ALE pulse width

CDP **Combined DMA Pins** (valid only if CONF.LBI = '1')

'0': DMA Acknowledge pins are separated

'1': DMA Acknowledge pins are combined, i.e. $\overline{\text{DACKTA}}$ & $\overline{\text{DACKRA}}$ become $\overline{\text{DACKA}}$, and $\overline{\text{DACKTB}}$ & $\overline{\text{DACKRB}}$ become $\overline{\text{DACKB}}$.

In this case, the pin LCLKOUT1 (EBC system clock phase 1 output), as well as the pin PHI1 (additional PCI clock phase 1 output) is available, if the direction of the pins has been programmed to output by setting the bit fields GPDIR.0 and GPDIR.1 to '1' (see **Figure 73 (b)**).

ABM **Arbitration Master Function**

'0': MUNICH32X is arbitration slave device ($\overline{\text{LHLDA}}$ pin is input)

'1': MUNICH32X is arbitration master device ($\overline{\text{LHLDA}}$ pin is output)

$\overline{\text{LBIRES}}$ **Reset LBI DMSM Block**

'0': Resets and keeps the DMSM in its initial state (same as hardware reset).

'1': Normal DMSM operation.

$\overline{\text{EBCRES}}$ **Reset LBI EBC Block**

'0': Resets and keeps the External Bus Controller in its initial state (same as hardware reset).

'1': Normal EBC operation.

DCA **Disregard the Interrupts for Channel A**

'0': Normal DMSM operation regarding interrupt processing

'1': DMSM passes all Channel A interrupts to interrupt queue (including RPF, XPR, RMC)

DCB **Disregard the Interrupts for Channel B**

'0': Normal DMSM operation regarding interrupt processing

'1': DMSM passes all Channel B interrupts to interrupt queue (including RPF, XPR, RMC)

MDA **Mode Channel A**

‘0’: Interrupt mode

‘1’: DMA assisted mode

MDB **Mode Channel B**

‘0’: Interrupt mode

‘1’: DMA assisted mode

SDA **Shared DMA Channel A**

‘0’: Separate Tx and Rx DMA (default)

‘1’: Shared Tx and Rx signals. Requests on DRQTA (for both Tx and Rx), and Acknowldege on $\overline{\text{DACKA}}$ (for both Tx and Rx).

DID **Direction of DMA signals (Valid only if SDA = 1)**

‘0’: DMA request for Rx direction (from LBI peripheral)

‘1’: DMA request for Tx direction (from LBI peripheral)

HE1 **HSCX/ESCC Register Decoding on Pin LINTI1**

‘0’: ESCC2 register decoding
LINTI1 input signal polarity active high

‘1’: HSCX register decoding
LINTI1 input signal polarity active low

HE2 **HSCX/ESCC Register Decoding on Pin LINTI2**

‘0’: ESCC2 register decoding
LINTI2 input signal polarity active high

‘1’: HSCX register decoding
LINTI2 input signal polarity active low

SPINT **Separate Interrupt Pins**

‘0’: LINTI1 pin is used for both DMA channels; LINTI2 pin is disabled

‘1’: LINTI1 pin is used for DMA channel A, LINTI2 pin is used for DMA channel B; LINTI2 pin is enabled

DV **DMA Request Validation Period**

0 0 0: No delay

1 0 0: reserved

1 0 1: Validate DRQTA & DRQTB for 8 LCLKOUT periods (falling edge)

1 1 0: Validate DRQTA & DRQTB for 16 LCLKOUT periods (falling edge)

1 1 1: Validate DRQTA & DRQTB for 32 LCLKOUT periods (falling edge)

The DMA request is considered in-active only after the programmed delay. This is used to determine the end of packet indication.

IPA **Interrupt Pass**

This bit field is necessary when connecting a FALC54 (PEB 2254) to the LBI.

'0': All interrupts are interpreted by the Data Mode State Machine (DMSM).

'1': Interrupts related to registers ISR2, ISR3 of FALC54 are not interpreted by the DMSM, but passed to the LBI interrupt queue.

Note: Any change to register LCONF settings must be followed by an EBC and LBI reset to ensure proper operation based on the new settings. This can be achieved by writing '0' *to bits* $\overline{EBCRES}$ *and* $\overline{LBIRES}$ *with any write access which changes any settings and set these bits to* '1' *again with a second write access to register LCONF.*

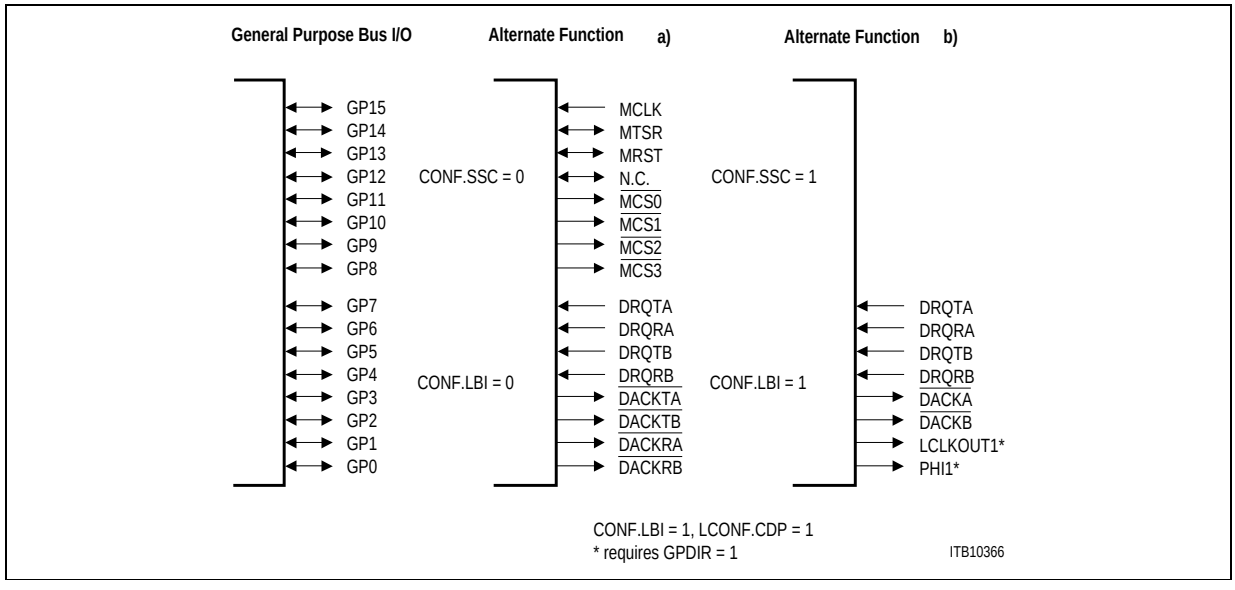


Figure 73
General Purpose Bus I/O and Alternate LBI/SSC Functions

LBI CC Block Indirect Address Register (LCCBA)

Access : read/write

Offset Address : 44_H

Reset Value : 00000000_H

	31															16
LCCBA	LCCBA(31:16)															

	15														2	1	0
LCCBA	LCCBA(15:2)														0	0	

The LBI CC Block Indirect Address Register points to the location in host memory which contains the actual base address pointer to the Control and Configuration Block (for LBI DMAC). This address must be DWORD-aligned.

LBI Start Transfer Register (LTRAN)

Access : read/write

Offset Address : $4C_H$

Reset Value : 00000003_H

	31															16
LTRAN	0000_H															

	15														1	0
LTRAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	GOB	GOA

GOB **LBI Start Transfer Channel B**

‘0’: Valid data in Next Tx Descriptor address is transferred on LBI channel B, if Tx HOLD for channel B is not set.

GOA **LBI Start Transfer Channel A**

‘0’: Valid data in Next Tx Descriptor address is transferred on LBI channel A, if Tx HOLD for channel A is not set.

LBI Tx Interrupt Queue Base Address Register (LTIQBA)

Access : read/write

Offset Address : 50_H

Reset Value : 00000000_H

	31															16
LTIQBA	LTIQBA(31:16)															

	15														1	0
LTIQBA	LTIQBA(15:2)														0	0

Specifies the host memory LBI DMAC Tx Interrupt Queue base or start address; this address must be DWORD-aligned.

LBI Tx Interrupt Queue Length Register (LTIQL)

Access : read/write

Offset Address : 54_H

Reset Value : 00000000_H

	31															16
LTIQL	0000_H															

	15															0
LTIQL	0	0	0	0	0	0	0	0	LTIQL(7:0)							

Specifies the DWORD count of the LBI DMA Controller Tx Interrupt Queue in host memory. The maximum size of the Queue is 4096 DWORDs ((n + 1) × 16 DWORDs, where n = LTIQL(7:0); refer to **Figure 72**).

LBI Rx Interrupt Queue Base Address Register (LRIQBA)

Access : read/write

Offset Address : 58_H

Reset Value : 00000000_H

	31															16
LRIQBA	LRIQBA(31:16)															

	15														1	0
LRIQBA	LRIQBA(15:2)														0	0

Specifies the host memory LBI DMAC Rx Interrupt Queue base or start address; this address must be DWORD-aligned.

LBI Rx Interrupt Queue Length Register (LRIQL)

Access : read/write

Offset Address : $5C_H$

Reset Value : 00000000_H

	31															16
LRIQL	0000_H															

	15															0
LRIQL	0	0	0	0	0	0	0	0	LRIQL(7:0)							

Specifies the DWORD count of the LBI Rx Interrupt Queue in host memory. The maximum size of the Queue is 4096 DWORDs ((n + 1) × 16 DWORDs, where n = LRIQL(7:0); refer to **Figure 72**).

LBI Indirect External Configuration Register 0 (LREG0)

Access : read/write

Offset Address : 60_H

Reset Value : 00000000_H

	31	16
LREG0	OFFSETA(7:0)	FIFOA(7:0)

	15	0
LREG0	CMDRA(7:0)	STARA(7:0)

Provides indirect pointers to the appropriate register of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG0 value to be programmed for ESCC2 operation is 00002020_H.

Note that the complete table of LREG0 … LREG5 settings for connecting other peripherals is shown after the LREG6 description.

LBI Indirect External Configuration Register 1 (LREG1)

Access : read/write

Offset Address : 64_H

Reset Value : 00000000_H

	31	16
LREG1	GISA(7:0)	ISR0A(7:0)

	15	0
LREG1	ISR1A(7:0)	RBCLA(7:0)

Provides indirect pointers to the appropriate register of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG1 value to be programmed for ESCC2 operation is $383A3B2A_H$.

LBI Indirect External Configuration Register 2 (LREG2)

Access : read/write

Offset Address : 68_H

Reset Value : 00000000_H

	31	16
LREG2	OFFSETB(7:0)	FIFOB(7:0)

	15	0
LREG2	CMDRB(7:0)	STARB(7:0)

Provides indirect pointers to the appropriate register of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG2 value to be programmed for ESCC2 operation is 00406060_H.

LBI Indirect External Configuration Register 3 (LREG3)

Access : read/write

Offset Address : $6C_H$

Reset Value : 00000000_H

	31	16
LREG3	GISB(7:0)	ISR0B(7:0)

	15	0
LREG3	ISR1B(7:0)	RBCLB(7:0)

Provides indirect pointers to the appropriate register of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG3 value to be programmed for ESCC2 operation is $387A7B6A_H$.

LBI Indirect External Configuration Register 4 (LREG4)

Access : read/write

Offset Address : 70_H

Reset Value : 00000000_H

	31															16
LREG4	OFFSETC(7:0)								FIFOC(7:0)							

	15															0
LREG4	OFFSETD(7:0)								FIFOD(7:0)							

Provides indirect pointers to the appropriate register of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG4 value to be programmed for ESCC2 operation is 00000040_H.

LBI Indirect External Configuration Register 5 (LREG5)

Access : read/write

Offset Address : 74_H

Reset Value : 00000000_H

	31															16
LREG5	0	0	RMC(2:0)			XTF/XHF(2:0)			XME(2:0)			0	0	0	0	0

	15															0
LREG5	0	ICA/EXI(2:0)			CEC(2:0)			RPF(2:0)			RME(2:0)			XPR(2:0)		

Provides indirect pointers to the appropriate register bit fields of the LBI peripheral, e.g., Siemens ESCCx, HSCX or FALC54 devices.
For example, the required LREG5 value to be programmed for ESCC2 operation is $3B200438_H$.

Note: The register bit fields XTF/XHF and ICA/EXI are not used in all LBI peripherals.

LBI Indirect External Configuration Register 6 (LREG6)

Access : read/write

Offset Address : 78_H

Reset Value : 00000000_H

	31															16
LREG6	0	0	RFTB(5:0)						0	0	RFTA(5:0)					

	15								0
LREG6	0000H	MB	MXB	MRMB	MRPB	MA	MXA	MRMA	MRPA

RFTB/ RFTA **RFIFO Threshold Level Channel B/A**

Controls the sizes of the accessible part of RFIFO of the LBI peripheral, e.g., Siemens ESCCx or FALC devices. Valid values are:

20_H: Size = 32 bytes,

10_H: Size = 16 bytes,

4_H: Size = 4 bytes,

2_H: Size = 2 bytes.

The value to be programmed depends on the corresponding register value of the peripheral (ESCC2: CCR4 register, FALC54: CCR1 register).

Note that for connection of HSCX on LBI, the value must be set to 32_H.

MB **Mask All Channel B**

‘1’: All interrupts on LBI channel B are masked.

MXB **Mask XPR Channel B**

‘1’: XPR interrupts on LBI channel B are masked.

MRMB **Mask RME Channel B**

‘1’: RME interrupts on LBI channel B are masked.

MRPB **Mask RPF Channel B**

‘1’: RPF interrupts on LBI channel B are masked.

MA **Mask All Channel A**

‘1’: All interrupts on LBI channel A are masked.

MXA **Mask XPR Channel A**

‘1’: XPR interrupts on LBI channel A are masked.

MRMA **Mask RME Channel A**

‘1’: RME interrupts on LBI channel A are masked.

MRPA **Mask RPF Channel A**

‘1’: RPF interrupts on LBI channel A are masked.

Examples of DMSM register values for inter-acting with different external peripherals that may be connected to the LBI and may be supported for automated data transfer using the DMSM are shown in the following tables (LREG6 is not considered since its value changes depending on the external interrupt bit fields that need to be masked):

Table 23
LBI External Configuration for ESCC2

Register Name	Byte3	Byte 2	Byte1	Byte 0
LREG0	00_H	00_H	20_H	20_H
LREG1	38_H	$3A_H$	$3B_H$	$2A_H$
LREG2	00_H	40_H	60_H	60_H
LREG3	38_H	$7A_H$	$7B_H$	$6A_H$
LREG4	00_H	00_H	00_H	40_H
LREG5	RMC(111_B), XTF(011_B), XME(001_B)		RPF(000_B), RME(111_B), XPR (000_B), CEC(010_B), ICA(000_B)	

Note: For Interrupt channel A mode use offset A & FIFOA pointers, and for Interrupt channel B mode use offset B & FIFOB pointers. For DMA A assisted transfers use offset C & FIFOC pointers, and for DMA B assisted transfers use offset D & FIFOD pointers in DMSM registers.

Table 24
LBI External Configuration for HSCX

Register Name	Byte3	Byte 2	Byte1	Byte 0
LREG0	00_H	00_H	21_H	21_H
LREG1	60_H	20_H	00_H	25_H
LREG2	00_H	40_H	61_H	61_H
LREG3	60_H	60_H	00_H	65_H
LREG4	00_H	00_H	00_H	40_H
LREG5	RMC(111_B), XTF(011_B), XME(001_B)		RPF(110_B), RME(111_B), XPR (100_B), CEC(010_B), ICA(010_B)	

Note: When a HSCX is connected to the LBI, the packet size in DMA mode must be a multiple of 32 bytes.

Table 25
LBI External Configuration for FALC54 (HDLC mode)

Register Name	Byte3	Byte 2	Byte1	Byte 0
LREG0	00_H	00_H	02_H	64_H
LREG1	$6E_H$	$6A_H$	$6B_H$	66_H
LREG2	00_H	00_H	02_H	64_H
LREG3	$6E_H$	68_H	69_H	66_H
LREG4	00_H	00_H	00_H	00_H
LREG5	RMC(111_B), XHF(011_B), XME(001_B)		RPF(000_B), RME(111_B), XPR (000_B), CEC(010_B), ICA(000_B)	

Note: When a FALC54 is connected to the LBI, only the LBI channel 'B' is used for data transfer. In this case, the interrupt mask bit fields for channel A (refer to LREG6 register description) are not valid.

LBI Status Register (LSTAT)

Access: read

Offset address: $7C_H$

Reset Value: 00000000_H

	31															16
LSTAT	0000_H															

	15															0
LSTAT	0	0	0	0	0	0	0	0	0	0	0	0	0	INT2	INT1	HLD

HLD **EBC HOLD Indicator**

Indicates EBC status

‘0’: The EBC is currently driving the bus.

‘1’: The EBC is currently in HOLD mode.

INT1 **EBC LINTI1 Indicator**

Indicates LINTI1 as interrupt source.

INT2 **EBC LINTI2 Indicator**

Indicates LINTI2 as interrupt source.

11.2.4 GPP Registers

General Purpose Bus Directon Register (GPDIR)

Access : read/write
Offset Address : 80_H
Reset Value : 00000000_H

	31	16
GPDIR	0000_H	

	15	0
GPDIR	GPDIR(15:0)	

GPDIR **General Purpose Bus Directon**

‘0’: General Purpose Bus GP0 … GP15 pins are input pins

‘1’: General Purpose Bus pins GP0 … GP15 are output pins

General Purpose Bus Data Register (GPDATA)

Access : read/write
Offset Address : 84_H
Reset Value : 00000000_H

	31	16
GPDATA	0000_H	

	15	0
GPDATA	GPDATA(15:0)	

16 bit data register for the General Purpose Bus.

General Purpose Bus Open Drain Register (GPOD)

Access : read/write

Offset Address : 88_H

Reset Value : 00000000_H

	31	16
GPOD	0000_H	

	15	0
GPOD	GPOD(15:0)	

GPOD **General Purpose Bus Open Drain**

'1': Corresponding General Purpose Bus pins GP0 … GP15 are open-drain pins

11.2.5 SSC Registers

This section contains descriptions of all SSC slave registers.

SSC Control Register (SSCCON)
a) Programming Mode (SSCEN = '0')

Access : read/write

Offset Address : 90_H

Reset Value : 00000000_H

	31															16
SSCCON	0000_H															

	15															0
SSCCON	SSC EN	SSC MS	0	0	SSC BEN	SSC PEN	SSC REN	SSC TEN	0	SSC PO	SSC PH	SSC HB	SSCBM(3:0)			

SSCBM **SSC Data Width Selection**

0_H: Reserved. Do not use this combination.
1 … 15_H: Transfer Data Width is 2 … 16 bit (<SSCBM>+1)

SSCHB **SSC Heading Control Bit**

'0': Transmit/Receive LSB First
'1': Transmit/Receive MSB First

SSCPH **SSC Clock Phase Control Bit**

'0': Shift transmit data on the leading clock edge, latch on trailing edge
'1': Latch receive data on leading clock edge, shift on trailing edge

SSCPO **SSC Clock Polarity Control Bit**

'0': Idle clock line is low, leading clock edge is low-to-high transition
'1': Idle clock line is high, leading clock edge is high-to-low transition

SSCTEN **SSC Transmit Error Enable Bit**

'0': Ignore transmit errors
'1': Check transmit errors

SSCREN **SSC Receive Error Enable Bit**

'0': Ignore receive errors
'1': Check receive errors

SSCPEN **SSC Phase Error Enable Bit**

‘0’: Ignore phase errors
‘1’: Check phase errors

SSCBEN **SSC Baudrate Error Enable Bit**

‘0’: Ignore baudrate errors
‘1’: Check baudrate errors

SSCMS **SSC Master Select Bit**

‘0’: Slave Mode. Operate on shift clock received via MCLK.
‘1’: Master Mode. Generate shift clock and output it via MCLK.

SSCEN **SSC Enable Bit = ‘0’**

Transmission and reception disabled. Access to control bits.

b) Operating Mode (SSCEN = ‘1’)

Access : read/write
Offset Address : 90_H
Reset Value : 00000000_H

	31												16
SSCCON	0000_H												

	15												0
SSCCON	SSC EN	SSC MS	0	SSC BSY	SSC BE	SSC PE	SSC RE	SSC TE	0	0	0	0	SSCBC(3:0)

SSCBC **SSC Bit Count Field**

Shift counter is updated with every shifted bit. **Do not write to!**

SSCTE **SSC Transmit Error Flag**

‘1’: Transfer starts with the slave’s transmit buffer not being updated

SSCRE **SSC Receive Error Flag**

‘1’: Reception completed before the receive buffer was read

SSCPE **SSC Phase Error Flag**

‘1’: Received data changes around sampling clock edge

SSCBE **SSC Baudrate Error Flag**

‘1’: More than factor 2 or 0.5 between Slave’s actual and expected baudrate

SSCBSY **SSC Busy Flag**

Set while a transfer is in progress. **Do not write to!**

SSCMS **SSC Master Select Bit**

‘0’: Slave Mode. Operate on shift clock received via MCLK.
‘1’: Master Mode. Generate shift clock and output it via MCLK.

SSCEN **SSC Enable Bit = ‘1’**

Transmission and reception enabled. Access to status flags and Master/Slave control.

SSC Baud Rate Generator Register (SSCBR)

Access : read/write

Offset Address : 94_H

Reset Value : 00000000_H

	31															16
SSCBR	0000_H															

	15															0
SSCBR	SSCBR(15:0)															

These bits define the baud rate used for data transfer via the SSC interface. Reading SSCBR (while SSC in enabled) returns the timer value. Reading SSCBR (while SSC in disabled) returns the programmed reload value. The desired reload value of the baud rate can be written to SSCBR when the SSC interface is disabled.
The table below lists some possible baud rates together with the required reload values, assuming a PCI clock of 20 MHz (33 MHz also supported).

SSCBR(15:0)	Baud Rate	Bit Time
0000_H	Reserved. Use a reload value > 0.	–
0001_H	5 MBaud	200 ns
0002_H	3.3 MBaud	300 ns
0003_H	2.5 MBaud	400 ns
0004_H	2.0 MBaud	500 ns
0009_H	1.0 MBaud	1 µs
0063_H	100 KBaud	10 µs
$03E7_H$	10 KBaud	100 µs
$270F_H$	1.0 KBaud	1 ms
$FFFF_H$	152.6 Baud	6.6 ms

Note 1: The contents of SSCBR must always be > 0.

Note 2: Never write to SSCBR, while the SSC is enabled.

SSC Tx Buffer Register (SSCTB)

Access : write
Offset Address : 98_H
Reset Value : 00000000_H

	31	16
SSCTB	0000_H	

	15	0
SSCTB	SSCTB(15:0)	

Contains the last of the SSC interface transmitted 16-bit word.

SSC Rx Buffer Register (SSCRB)

Access : write
Offset Address : $9C_H$
Reset Value : 00000000_H

	31	16
SSCRB	0000_H	

	15	0
SSCRB	SSCRB(15:0)	

Contains the last of the SSC interface received 16-bit word.

SSC Chip Select Enable Register (SSCCSE)

Access : read/write

Offset Address : $A0_H$

Reset Value : 00000000_H

	31															16
SSCCSE	0000_H															

	15															0
SSCCSE	0	0	0	0	0	0	0	0	AS3	AS2	AS1	AS0	0	0	0	0

AS3 **Pin $\overline{\text{MCS3}}$ Control**

‘0’: Activation of $\overline{\text{MCS3}}$ chip select pin

‘1’: Hardware automatically controls $\overline{\text{MCS3}}$

AS2 **Pin $\overline{\text{MCS2}}$ Control**

‘0’: Activation of $\overline{\text{MCS2}}$ chip select pin

‘1’: Hardware automatically controls $\overline{\text{MCS2}}$

AS1 **Pin $\overline{\text{MCS1}}$ Control**

‘0’: Activation of $\overline{\text{MCS1}}$ chip select pin

‘1’: Hardware automatically controls $\overline{\text{MCS1}}$

AS0 **Pin $\overline{\text{MCS0}}$ Control**

‘0’: Activation of $\overline{\text{MCS0}}$ chip select pin

‘1’: Hardware automatically controls $\overline{\text{MCS0}}$

SSC Interrupt Mask Register (SSCIM)

Access : read/write

Offset Address : $A4_H$

Reset Value : 00000000_H

	31															16
SSCIM	0000_H															

	15															0
SSCIM	0	0	0	0	0	0	0	0	0	0	0	0	0	IMRX	IMER	IMTX

IMRX **Interrupt Mask Rx**

‘0’: Disable SSC receive interrupt

‘1’: Enable SSC receive interrupt

IMER **Interrupt Mask Error**

‘0’: Disable SSC error interrupt

‘1’: Enable SSC error interrupt

IMTX **Interrupt Mask Tx**

‘0’: Disable SSC transmit interrupt

‘1’: Enable SSC transmit interrupt

Note 1: The transmit interrupt notifies the CPU about the start of a transmission.

Note 2: The receive interrupt transports the receive data to the shared memory.

Note 3: The error interrupt notifies the CPU about different error conditions of data transmission and reception. To further specify what sort of error interrupt the user wants to trace, the corresponding bit of the SSC control register SSCCON has to be set. The SSC error conditions that can be checked are transmit errors (SSCCON(8) = ‘1’), phase errors (SSCCON(10) = ‘1’) and baud rate errors (SSCCON(11) = ‘1’). If any of these error conditions shall not be checked, the corresponding bit has to be set to ‘0’.

Example

To check for transmit errors only:

SSCIM(1) = ‘1’, SSCCON(8) = ‘1’, SSCCON(10) = ‘0’,SSCCON(11) = ‘0’

Note: SSCCON(9) always has to be set to zero!

11.2.6 IOM®-2 Registers

This section contains descriptions of all IOM®-2 slave registers.

IOM®-2 Control Register 1 (IOMCON1)

Access : write/read

Offset Address : $B0_H$

Reset Value : $000000FF_H$

	31															16
IOMCON1	0	0	0	0	SWR	EXL	ENIH	CLR	0	0	0	EOM	MXR	MRR	0	0

	15						7	0
IOMCON1	AS	MSN(2:0)	MRIM	MRC	MXIM	MXC	IMCH (7:0)	

SWR **Software Reset**

‘1’: resets and keeps the IOM®-2 handler in its initial state.
‘0’: Normal operation.

EXL **External Loop**

‘0’: disables the external loop.

‘1’: enables an external test loop:
Incoming data (MON, C/I, MR and MX) are mirrored to the M32X output line.

ENIH **Enable IOM®-2 Handler**

‘1’: enables IOM®-2 handler.

Note: Set to ‘1’ as part of the initialisation whenever IOM®-2 operation is used.

CLR **Clock rate**

The IOM®-2 clock rate is related to the IOM®-2 data rate:

‘0’: single data rate.

‘1’: double data rate.

$\overline{\text{EOM}}$ **End of Monitor Data Stream**

'0': No further Data will be loaded into the Monitor Transmit FIFO. The IOM®-2 handler generates MX = '1' during 2 consecutive IOM® frames, after last data has been sent out of the transmit FIFO.

'1': Data loaded in the Monitor Transmit FIFO (2 bytes deep) belong to one large data packet (e.g. 8 bytes) and are sent directly after FIFO write access. No EOM condition is generated.

MXR **Monitor Transmit FIFO Reset**

'1': Resets the transmit FIFO and keeps the transmit part in its initial state.

'0': Normal operation.

MRR **Monitor Receive Reset**

'1': Resets and keeps the receiver in its initial state.

'0': Normal operation.

Note: MRR = '1' also affects the transmit part; set to '0' for normal operation even if monitor receive is not used.

AS **Auto Search**

'0': The monitor handler is allocated to one monitor channel which number is programmed in the MSN field.

'1': The monitor handler searches for an active monitor channel on the IOM® interface (receive data line). It generates an interrupt upon reception of a MX = '0'.
The interrupt vector contains the IOM® monitor subframe number of the selected monitor channel.

Note: Prior to starting the read procedure, this IOM® monitor subframe number must be loaded into the MSN bit field first.

MSN (2:0) **Monitor Subframe Number**

Address of the active monitor channel (one out of 8).

MRIM **Monitor Channel Receiver Interrupt Mask**

'0': Monitor receiver interrupts are not masked (are enabled)

'1': Monitor receiver interrupts are masked.

MRC **Monitor Channel Receiver Control**

'0': Monitor receiver is disabled (OFF)

'1': Monitor receiver is enabled (ON)

MXIM **Monitor Channel Transmitter Interrupt Mask**

‘0’: Monitor transmitter interrupts are not masked (are enabled)

‘1’: Monitor transmitter interrupts are masked.

MXC **Monitor Channel Transmitter Control**

‘0’: Monitor transmitter is disabled (OFF)

‘1’: Monitor transmitter is enabled (ON)

IMCH (7:0) **Interrupt Mask Channel-wise**

Interrupt on the IOM® subchannel corresponding to the bit field number is:

‘0’: enabled

‘1’: masked (Monitor and C/I interrupts).

IOM®-2 Control Register 2 (IOMCON2)

Access : write/read (bits 3 to 6 read only)

Offset Address : $B4_H$

Reset Value : 00000000_H

	31							16
IOMCON2	0000_H							

	15	7							0
IOMCON2	00_H	0	TMO	CIT1	CIT0	CON1	ASIM	MOIM	CIIM

TMO **IOMTMO Access Indicator**

A write access to the IOMTMO register is

‘0’: allowed

‘1’: not allowed.
This bit is readable only.

CIT1 **IOMCIT1 Access Indicator**

A write access to the IOMCIT1 register is

‘0’: allowed

‘1’: not allowed.
This bit is readable only.

CIT0 **IOMCIT0 Access Indicator**

A write access to the IOMCIT0 register is

‘0’: allowed

‘1’: not allowed.
This bit is readable only.

CON1 **IOMCON1 Access Indicator**

A write access to the IOMCON1 register is

‘0’: allowed

‘1’: not allowed.
This bit is readable only.

ASIM **Auto Search Interrupt Mask**

The monitor handler generates an interrupt upon Auto Search. This interrupt is:

‘0’: enabled

‘1’: masked.

MOIM **Monitor Interrupt Mask**

The monitor handler generates interrupts upon receiving or transmitting monitor data. The interrupts MEM, MRFF, MTFF and MAB are:

‘0’: enabled
(refer to the IOMSTAT register and **Section 8.3.1**)

‘1’: masked

CIIM **C/I Interrupt Mask**

The C/I handler generates an interrupt upon a detection of a change in one of 8 C/I fields (double last look). This interrupt is:

‘0’: enabled (refer to **Section 8.3.2**)

‘1’: masked

IOM®-2 Status Register (IOMSTAT)

Access : read

Offset Address : $B8_H$

Reset Value : 01008000_H

	31													16
IOMSTAT	0	0	0	0	0	0	0	RIP	0	0	0	AMCF	CSA	CNO(2:0)

	15							7	0
IOMSTAT	MFR	MFA	MTO	0	MEM	MRFF	MTFE	MAB	XX_H

RIP **Reset in Progress**

‘1’: A software reset or a monitor FIFO reset (SWR, MXR or MRR) are currently in progress

AMCF **Active Monitor Channel Found**

‘1’: Indicates, that autosearch function has found a channel with an active monitor transmit bit. The channel number is coded in CNO(2:0).

CSA **Channel Search Active**

‘1’: The autosearch function is active.

CNO **Channel Number**

Contains the coding for the IOM®-2 channel 0 … 7, in which the autosearch function has found an active monitor transmit bit.

MFR **Monitor Transmit FIFO Ready**

‘0’: The transmit FIFO is full

‘1’: The transmit FIFO is not full; 1 or 2 bytes may be storred in the transmit FIFO

MFA **Monitor Receive FIFO Data Available**

‘0’: The receive FIFO is empty

‘1’: The receive FIFO is not empty; 1 or 2 bytes may be read out of the FIFO

MTO **Monitor Timeout**

‘1’: A monitor timeout has occured.

MEM	**Monitor End of Message**
	'1': A complete monitor data message has been received.
MRFF	**Monitor Receive FIFO Full**
	'1': The receive FIFO is full
MTFE	**Monitor Transmit FIFO Empty**
	'1': The transmit FIFO is empty
MAB	**Monitor Abort**
	'1': A monitor abort was detected.
$\mathbf{XX_H}$	Do not care

IOM®-2 C/I Code Tx Register Channels 0 … 3 (IOMCIT0)

Access : write/read

Offset Address : $C0_H$

Reset Value : $FEFEFEFE_H$

	31									16
IOMCIT0	1	1	COM3	1	0	1	1	COM2	1	0

	15									0
IOMCIT0	1	1	COM1	1	0	1	1	COM0	1	0

COMn **Command in C/I Channel 3 … 0)**

Contains the IOM®-2 C/I information (4 bits) for channel 3 … 0 in transmit direction.

IOM®-2 C/I Code Tx Register Channels 4 … 7 (IOMCIT1)

Access : write/read

Offset Address : $C4_H$

Reset Value : $FEFEFEFE_H$

	31									16
IOMCIT1	1	1	COM7	1	0	1	1	COM6	1	0

	15									0
IOMCIT1	1	1	COM5	1	0	1	1	COM4	1	0

COMn **Command in C/I Channel 7 … 4 Tx**

Contains the IOM®-2 C/I information (4 bits) for channel 7 … 4 in transmit direction.

IOM®-2 C/I Code Rx Register Channels 0 … 3 (IOMCIR0)

Access : read

Offset Address : $C8_H$

Reset Value : 00000000_H

	31									16
IOMCIR0	0	0	IND3	X	0	0	0	IND2	X	0

	15									0
IOMCIR0	0	0	IND1	X	0	0	0	IND0	X	0

INDn **Indication in C/I Channel 3 … 0**

Contains the IOM®-2 C/I information (4 bits) for channel 3 … 0 in receive direction.

X Do not care

IOM®-2 C/I Code Rx Register Channels 4 … 7 (IOMCIR1)

Access : read

Offset Address : CC_H

Reset Value : 00000000_H

	31									16
IOMCIR1	0	0	IND7	X	0	0	0	IND6	X	0

	15									0
IOMCIR1	0	0	IND5	X	0	0	0	IND4	X	0

INDn **Indication in C/I Channel 7 … 4**

Contains the IOM®-2 C/I information (4 bits) for channel 7 … 4 in receive direction.

X Do not care

IOM®-2 Tx Monitor Register (IOMTMO)

Access : write/read

Offset Address : $D0_H$

Reset Value : $FFFFFFFF_H$

	31								16
IOMTMO	FF_H								FF_H

	15								0
IOMTMO	TMB1								TMB0

TMB1 **Transmit Monitor Byte 1**

Contains the byte 1 of the IOM®-2 transmit monitor channel information.

TMB0 **Transmit Monitor Byte 0**

Contains the byte 0 of the IOM®-2 transmit monitor channel information.

Note: Immediately after write access to TxMonitor Register IOMTMO the two bytes TMB0 and TMB1 are sent. The value of bit field MSN(2:0) in register IOMCON1 determins the monitor subframe number which is used.

IOM®-2 Rx Monitor Register (IOMRMO)

Access : read

Offset Address : $D4_H$

Reset Value : $FFFFFFFF_H$

	31								16
IOMRMO	FF_H								FF_H

	15								0
IOMRMO	RMB1								RMB0

RMB1 **Receive Monitor Byte 1**

Contains the byte 1 of the IOM®-2 receive monitor channel information.

RMB0 **Receive Monitor Byte 0**

Contains the byte 0 of the IOM®-2 receive monitor channel information.

11.2.7 Mailbox Registers

This section contains descriptions of all Mailbox slave and LBI accessible registers.

Mailbox Command Register (MBCMD)

Access : read/write

Offset Address
(slave register from PCI host system) : $E0_H$

Address coding
(pins LA(2:0) from LBI peripheral) : 000_B

Reset Value : 00000000_H

	31	16
MBCMD	0000_H	

	15	0
MBCMD	MBINT(15:1)	INPCI/ INLBI

INPCI **Mailbox Interrupt from PCI Host System**

This bit regulates the exclusive access of the Mailbox Data Registers from PCI host system.

Read access:
Returns the value written to the INLBI bit field from the LBI peripheral.

Write access:

‘1’: Generates interrupt on LBI side by assertion of LINTO signal (a mailbox interrupt vector with the contents of bits MBINT(15:1) is generated). The interrupt signal may be deasserted by writing a ‘1’ to the MBI bit field in Status Acknowledge Register STACK.

INLBI **Mailbox Interrupt from Intelligent LBI Peripheral**

This bit regulates the exclusive access of the Mailbox Data Registers from LBI peripheral.

Read access:
Returns the value written to the INPCI bit field from host PCI system.

Write access:

'1': Generates an interrupt on PCI side by assertion of $\overline{\text{INTA}}$ signal (a mailbox interrupt vector with the contents of bits MBINFO(15:1) is generated). The interrupt signal may be deasserted by reading MBCMD.

MBINT **Mailbox Interrupt Information from PCI/LBI**

User programmable bit fields to pass interrupt information from PCI host system on to LBI peripheral and vice versa. A read access from one side of the Mailbox returns the MBINT values that the other side had written to.

Mailbox Data Register 1 … **7 (MBDATA1** … **MBDATA7)**

Accesses	: read/write
Offset Addresses (slave registers from PCI host system)	: $E4_H$, $E8_H$, EC_H, $F0_H$, $F4_H$, $F8_H$, FC_H
Address codings (pins LA(2:0) from LBI peripheral)	: 001_B, 010_B, 011_B, 100_B, 101_B, 110_B, 111_B
Reset Values	: 00000000_H

	31 … 16	15 … 0
MBDATA1	0000_H	
MBDATA1		MBDATA1(15:0)

Used for data transfer between PCI interface and LBI and vice versa. For 16-bit LBI accesses, bits 15 … 0 are used to transfer data; whereas in case of 8-bit LBI accesses, only bits 7 … 0 are used.

*Note: The seven Mailbox Data Registers have the same structure (refer to **Section 6.1.4**).*

12 Host Memory Organization

12.1 Control and Configuration Block (CCB) in Host Memory

The architecture of the MUNICH32X uses two different Control and Configuration Blocks in host memory, as illustrated in **Figure 74** and **Figure 75**:

1. related to the serial PCM core (CCB)
2. related to the LBI (LCCB).

Note that each address in CCB/LCCB is DWORD aligned (i.e., the two least significant address bit fields must be set to ‘0’).

12.1.1 Serial PCM Core CCB

The **Table 26** shows the size of the CCB sections:

Table 26
Sizes of the Control and Configuration Block

Section	Number of DWORDs
Action Specification Command	1
Reserved	2
Time Slot Assignment	32
Channel Specification	128
Current Rx Descriptor Addresses	32
Current Tx Descriptor Addresses	32

The reserved section, located after the Action Specification, maintains backward compatibility with the MUNICH32, PEB 20320.

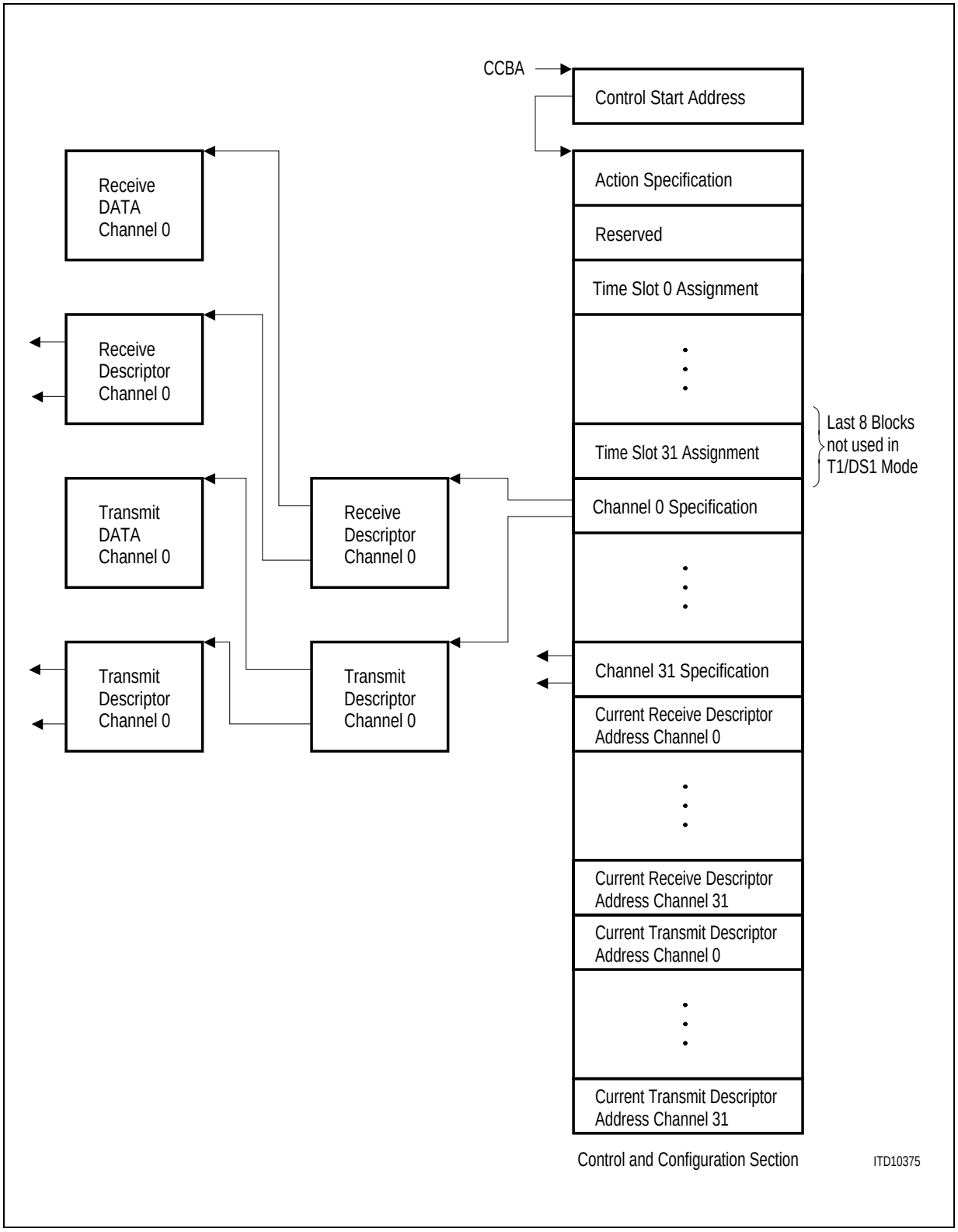


Figure 74
Serial PCM Core Control and Configuration Block (CCB)

12.1.2 LBI CCB

The following table shows the size of the LCCB sections:

Table 27
Sizes of the LBI Control and Configuration Block

Section	Number of DWORDs
Action Specification Command	1
Channel Specification	4
Current Rx Descriptor Addresses	2
Current Tx Descriptor Addresses	2

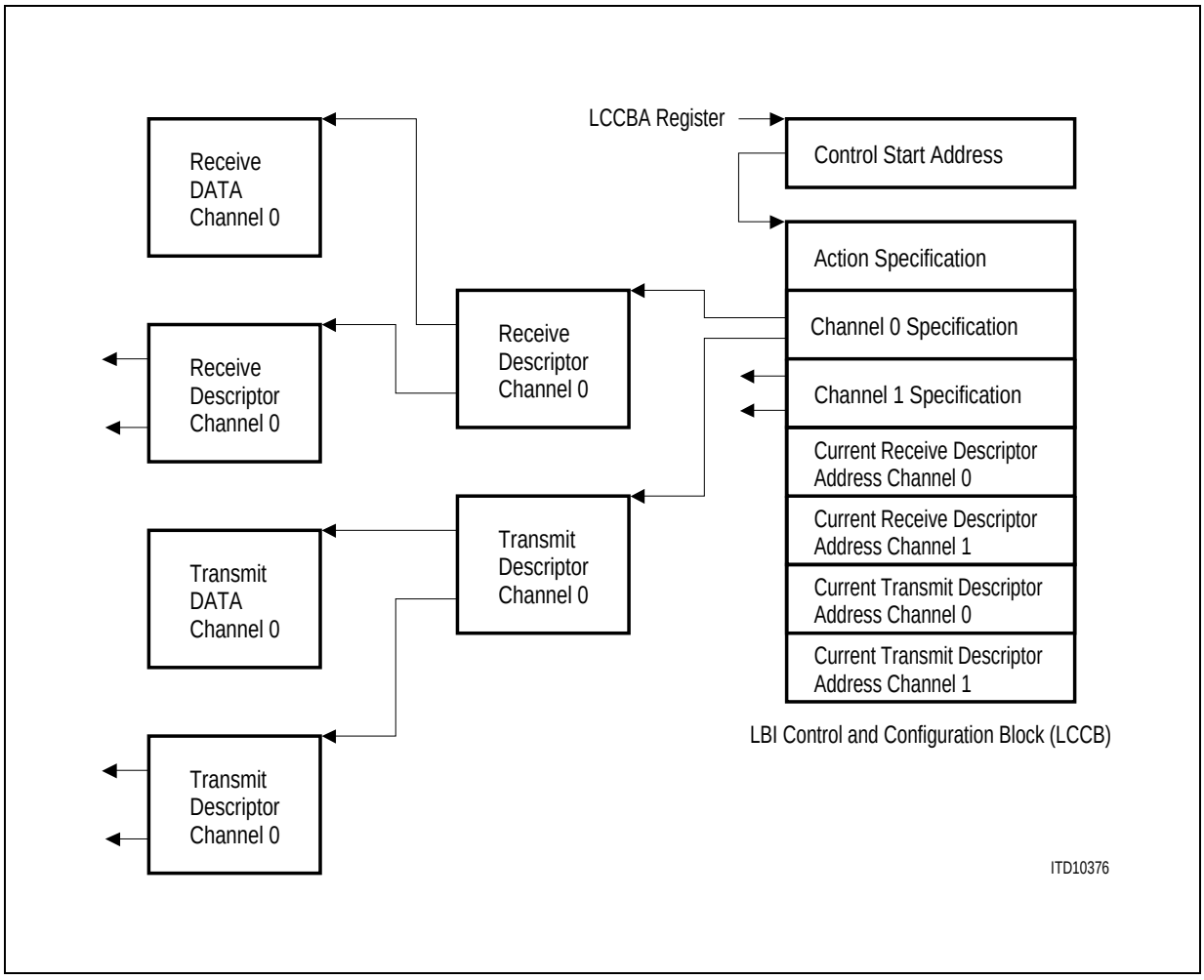


Figure 75
LBI Control and Configuration Block (LCCB)

12.2 Action Specification

The action specification is read once after each action request (initiated via bit field ARPCM for serial PCM core related action request or ARLBI for LBI related action request in Command Register CMD). All actions are selected by setting the corresponding action specification bit field to ‘1’.

12.2.1 Serial PCM Core Action Specification

31											16
0000_H											

15											0
IN	ICO	0	Channel Number		0	RES	LOC	LOOP	LOOPI	IA	0	0

IN: Initialization procedure; setting this bit to one causes MUNICH32X to fetch all the time slot assignments and the channel specification of the selected channel (channel number). To avoid collision all time slots being reinitialized should be in a deactivated mode, i.e. the receive and transmit channels must be switched off.

ICO: Initialize Channel Only; only the channel specification of the selected channel (channel number) is read and reconfigured.

RES: RESET; a single initialization procedure is performed. The time slot assignment and all channel specifications are written into the CSR. All time slots are reinitialized.

Note: The bits IN, ICO, RES are mutually exclusive within one action specification. They establish different ways of initializing, configuring and reconfiguring the channels and time slots of the MUNICH32X.

IA: Interrupt Attention; a new interrupt queue has been defined. The interrupt counter is reset.

For test purposes, four different loops can be switched at the serial interface using the following coding of the bit fields LOC, LOOP, LOOPI:

LOC	LOOP	LOOPI	Interpretation
0	0	0	no loop
0	0	1	complete internal loop
0	1	0	complete external loop
0	1	1	switch loops off
1	0	0	not allowed
1	0	1	channelwise internal loop
1	1	0	channelwise external loop
1	1	1	not allowed

The loops have the following functions:

– Complete external loop
  The serial data input is physically mirrored back to the serial data output. The time and strobe signals for receive and transmit direction must be identical.
– Complete internal loop
  The serial data output is physically mirrored back to the serial data input. The data on the external input line are ignored. The logical channels have to be programmed identically. The time and strobe signals for receive and transmit direction must be identical.
– Channelwise external loop
  One single logical channel is mirrored logically from serial data input to serial data output. The other channels are not affected by this operation. The data rate for this single logical channel must be identical for receive and transmit direction.
– Channelwise internal loop
  One single logical channel is mirrored logically from serial data output to serial data input. The other channels are not affected by this operation. The data rate for this single logical channel must be identical for receive and transmit direction.

All loops of the MUNICH32X are under complete software control. Loops can be closed and opened via software. Note that a more detailed description of the test loops will be provided later in the Application Notes section.

Handling of the Loops

1. Switch on loops:

RES = IN = ICO = ‘0’

LOC, LOOP, LOOPI	determine the selected loop type
PCM, MFL	do not change the previous values
CHANNEL NUMBER	in case of channelwise loops use the selected channel number; in case of complete loops use channel number of an active channel.

2. Switch off loops:

RES = IN = ICO = ‘0’
LOC = ‘0’, LOOP = LOOPI = ‘1’

afterwards:
RES = IN = ICO = ‘0’
LOC = LOOP = LOOPI = ‘0’

PCM, MFL	do not change the previous values
CHANNEL NUMBER	use channel number used with the ‘switch on loop’

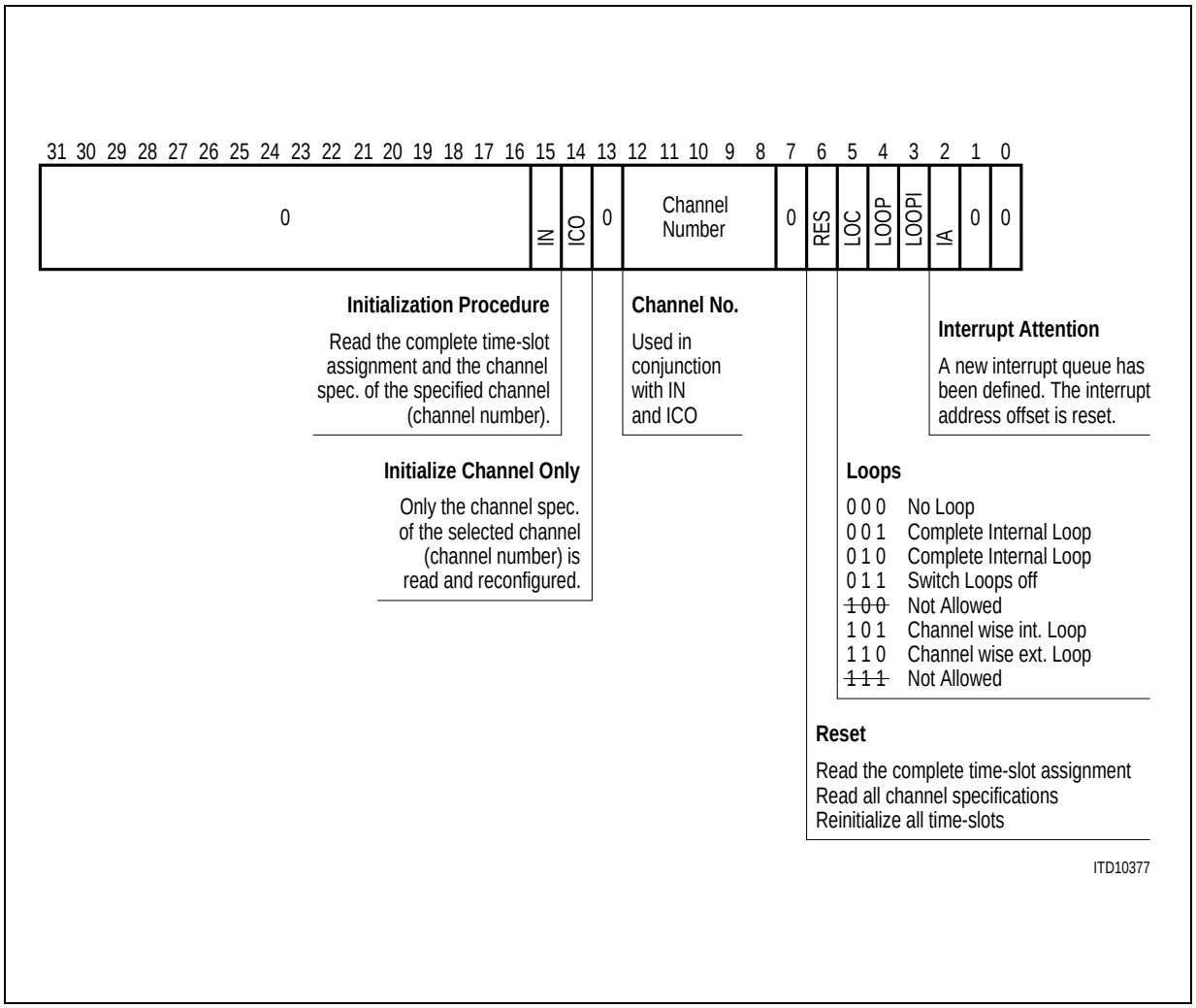


Figure 76
Serial PCM Core Action Specification

12.2.2 LBI Action Specification

The LBI action specification is different from the serial PCM core action specification; it provides only the bit fields ICO, IA (same functionality) and a single bit field CHN to address the LBI channel A (‘0’) or B (‘1’), replacing the 5 channel number bit fields of the PCM action specification.

31															16
0000_H															

15															0
0	ICO	0	0	0	0	0	CHN	0	0	0	0	0	IA	0	0

12.3 Serial PCM Core Interrupt Vector Structure

When an serial PCM core interrupt occurs, the MUNICH32X writes the interrupt information and the channel number into the interrupt circular buffer (either receive or transmit direction). At the same time it generates an interrupt pulse. The classes of error (e.g., host initiated interrupt or CRC error) of a channel in one direction are treated independently of each other. If several interrupt events coincide they will be indicated to the host with one shared interrupt.

Note that two seperate interrupt vectors for receive and transmit direction exist. In receive direction, the interrupt vector is located at the shared memory address of RIQBA register. In receive direction, the interrupt vector is located at the address of TIQBA register.

Serial PCM Core Interrupt Vector Rx Direction

	31															16
RX_IV	0	0	1	1	0	0	0	0	FRC	E7	E6	E5	E4	E3	E2	E1

	16															0
RX_IV	SB	SA	X	HI	FI	IFC	SF	ERR	FO	0	0	Channel Number				

Serial PCM Core Interrupt Vector Tx Direction

	31															16
TX_IV	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

	16												0
TX_IV	0	0	0	HI	FI	0	0	ERR	FO	FE2	Late Stop	Channel Number	

Note that the bit order has changed from the MUNICH32 (PEB 20320). The significant changes of the contents are the additional FE2 and Late Stop bits.

Bit Assignment For Interrupt Queue Registers

Channel specific interrupts indicate specific events in the channel encoded by 'Channel Number' in Rx or Tx Interrupt Queue Register.
The interpretation of these interrupts depends on the specification of the channel in which they occur.
The following table shows which interrupts can occur in which mode (unused bits are always '0').

	31															16
	0	0	1	R/T	0	0	0	0	FRC	E7	E6	E5	E4	E3	E2	E1
HDLC	0	0	1	RT	0	0	0	0	0	0	0	0	0	0	0	0
V.110/X.30	0	0	1	RT	0	0	0	0	R	R	R	R	R	R	R	R
TMA	0	0	1	RT	0	0	0	0	0	0	0	0	0	0	0	0
TMB/TMR	0	0	1	RT	0	0	0	0	0	0	0	0	0	0	0	0

	15															0
	SB	SA	X	HI	FI	IFC	SF	ERR	FO	FE2	Late Stop	Channel Number				
HDLC	0	0	0	RT	RT	R	R	RT	RT	T	TI	X	X	X	X	X
V.110/X.30	R	R	R	RT	0	0	0	RT	RT	0	0	X	X	X	X	X
TMA	0	0	0	RT	T	0	0	RT	RT	T	0	X	X	X	X	X
TMB/TMR	0	0	0	RT	RT	0	0	RT	RT	T	0	X	X	X	X	X

Where '0' means that the bit is always '0' for this mode
'R' means a bit that can only be set in the receive direction, i.e. may only be '1' if RT is '1'
'T' means a bit that can only occur in transmit direction, i.e. may only be '1' if RT is '0'
'RT' means a bit that can occur in receive or transmit direction
'X' means a bit fixed by the channel and direction (Rx, Tx) of the event it belongs to.
'TI' means transmit in conjunction with $\overline{\text{DRDY}}$ pin (in IOM®-2 mode and HDLC)

12.4 Interrupt Bit Field Definitions

The functions of the interrupt bits depend on the protocol mode. They are therefore discussed bit by bit, indicating the different meanings in each mode.

R/T: (all modes)
Determines the direction of the interrupt ('1' = Rx, '0' = Tx).

FRC: (V.110/X.30 mode, receive direction only)
Change of the framing (E, S, X) bits of the V.110/X.30 frame detected. This interrupt is generated whenever a change in the E-, S-, X-bits is detected, but at most one time within one frame of 10 octets, even if there is more than one change within the frame. After detecting a receive abort channel command for one 10-octet frame FRC is also issued.

Ex, Sx, X: (V.110/X.30 mode, receive direction only, only in conjunction with FRC)
The value of the bits Ex, Sx, X in the received V.110/X.30 frame. If a value changes e.g., two times within the same frame, only the final change is reported.
If the change was caused by a receive abort channel command all bits are 0.

HI: (all modes, all directions)
Host initiated Interrupt; this bit is set when the MUNICH32X detects the HI bit in the Rx or Tx descriptor and branches to the next descriptor, or starts polling the HOLD bit if set.

FI: 1.1 HDLC, TMB, TMR Receive Direction:
FI = 1 indicates, that a frame has been received completely or was stopped by a receive abort channel command or fast receive abort or a HOLD in a Rx descriptor. It is set when the MUNICH32X branches from the last descriptor belonging to the frame to the first descriptor of a new frame. It is also set when the descriptor in which the frame finished contained a HOLD bit, the interrupt is then issued when the MUNICH32X starts polling the HOLD bit.

1.2 HDLC, TMB, TMR, TMA Transmit Direction:
issued if the FE bit is detected in the Tx descriptor. It is set when the MUNICH32X branches to the next Tx descriptor, belonging to a new frame, or when it starts polling the HOLD bit if set in conjunction with the FE bit; ERR and FI are set if a Tx descriptor contains a HOLD bit, but no FE bit

IFC: (HDLC mode, Receive direction only)
Idle/Flag Change; an interrupt is generated in HDLC if the device changes the interframe time-fill (ITF) state. After reset, the device is in the ITF idle state. It changes to the ITF flag state if it receives two consecutive flags with or without shared zeros. It changes back to the ITF

idle state upon reception of 15 contiguous ‘1’-bits or when a receive abort channel command is active during 15 received bits.

SF: (HDLC mode, Receive direction only, always in conjunction with FI)
Short frame detected
A frame with ≤ 16 bits between start flag and end flag or end abort flag for CRC16
≤ 32 bits between start flag and end flag or end abort flag for CRC32
has been detected. The sequences 7E $7F_H$ and 7E FE_H and 7E FF_H are also short frames.
SF is always in conjunction with ERR except for the frames
7E00 $007E_H$ for CRC16
7E00 0000 $007E_H$ for CRC32

ERR: always in conjunction with FI = 1

1.1 HDLC mode Receive Direction
One of the following receive errors occurred
– FCS of the frame was incorrect
– the bit length of the frame was not divisible by 8
– the byte length exceeded MFL
– the frame was stopped by $7F_H$
– the frame could only be partly stored due to internal buffer overflow of RB
– the frame was ended by a receive abort channel command
– the frame could not be transferred to the shared memory completely because of a HOLD bit set in a Rx descriptor not providing enough bytes for the frame.
– the frame was aborted by a fast receive abort channel command
A more detailed error analysis can be performed by the status information in the Rx descriptor.

1.2 HDLC mode Transmit Direction
one of the following transmit errors occurred:
– the last descriptor had HOLD = 1 and FE = 0
– the last descriptor had NO = 0 and FE = 0

2.1 V.110/X.30 mode Receive Direction
one of the following receive errors occurred:
– data could only partly be stored due to internal buffer overflow of RB
– 3 consecutive frames had an error in the synchronization pattern (loss of synchronism)
– a fast receive abort channel command was issued
– the data could not be transferred to the shared memory completely because of a HOLD bit set in a Rx descriptor not providing enough bytes for the data

– a receive abort channel command was active for at least 3 consecutive frames

A more detailed error analysis can be performed by the status information in the Rx descriptor.

2.2 V.110/X.30 mode Transmit Direction
one of the following transmit errors occurred
– the last descriptor had a HOLD = 1 or FE = 1
– the last descriptor had FE = 0 and NO = 0

3.1 TMA mode Receive Direction
one of the following errors occurred
– the data could not be transferred to the shared memory completely because of a HOLD bit set in a Rx descriptor not providing enough bytes for the data
– a fast receive abort channel command was issued

3.2 TMA mode Transmit Direction
see **Chapter 1.2**

4.1 TMB/TMR mode Receive Direction
always in conjunction with FI = 1
one of the following receive errors occurred
– the bit length of the frame was not divisible by 8
– the frame could only be partly stored due to internal buffer overflow of RB
– the frame could not be transferred to the shared memory completely because of a HOLD bit set in a Rx descriptor not providing enough bytes for the frame
– the frame was aborted by a fast receive abort channel command

A more detailed error analysis can be performed by the status information in the Rx descriptor.

4.2 TMB/TMR mode Transmit Direction
see **Chapter 1.2**

FO: 1.1 HDLC, TMB, TMR Receive Direction
The MUNICH32X has discarded one or more whole frames or short frames or change of interframe time-fill informations due to inaccessibility of the internal buffer RB.

1.2 HDLC, TMB, TMR Transmit Direction
The MUNICH32X is unable to access the shared memory in time or has detected a fatal bus cycle error during a read access on the transmit data section. The current erroneous frame is aborted with a ‘0’ and 14 ‘1’ for HDLC, with ‘00’ for TMB and ‘0000’ for TMR; afterwards interframe time-fill is sent until the MUNICH32X can access again the shared

memory. The MUNICH32X will read the transmit data from the location which should be accessed before the Tx-FO happened and transmit the rest of the erroneous frame.

2.1 V.110/X.30 Receive Direction
The MUNICH32X has discarded a loss of synchronism information or a change of a E-, S-, X-bits information due to inaccessibility of the internal buffer RB.

2.2 V.110/X.30 Transmit Direction
The MUNICH32X is unable to access the shared memory in time. It generates three 10-octet frames with framing errors and restarts with the next error-free Tx data.

3.1 TMA Receive Direction
The MUNICH32X has discarded data due to inaccessibility of the internal buffer RB.

3.2 see **Chapter 1.2**

FE2: HDLC, TMA, TMB/TMR Transmit Direction

Indicates that data has been sent (including CRC)
Note that this provides a Tx End-of-Packet interrupt capability which allows host software to free-up Tx buffers after the contents have been completely transferred to the MUNICH32X.

Late Stop: HDLC Transmit Direction (IOM®-2 mode only)

Indicates that the DRDY pin has been asserted while transmission of data took place on an IOM®-2 D-channel (refer to IOM®-2 interface description in **Chapter 8**).

If DRDY is asserted after the MUNICH32X prepared at least the first data bit of a packet for sending or was already sending a packet, these prepared or sent data are lost; the current descriptor has to be re-transmitted. In order to clean up the transmit buffer a ‘Transmit Off’ command followed by a ‘Transmit Init’ command together with NITBS = ‘1’ must be issued for the corresponding channel (refer to Channel Specification description in **Chapter 12.6**).

The following table shows which interrupt bits fields may be masked by channel specification bits.

	31															16
	0	0	0	R/T	0	0	0	0	FRC	E7	E6	E5	E4	E3	E2	E1
Receive	–	–	–	–	–	–	–	–	CH	CH	CH	CH	CH	CH	CH	CH
Transmit	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	15															0
	SB	SA	X	HI	FI	IFC	SF	ERR	FO	FE2	Late Stop	Channel Number				
Receive	CH	CH	CH	–	FIR	IFC	SFE	RE	RE	–	–	–	–	–	–	–
Transmit	–	–	–	–	FIT	–	–	TE	TE	FE2	–	–	–	–	–	–

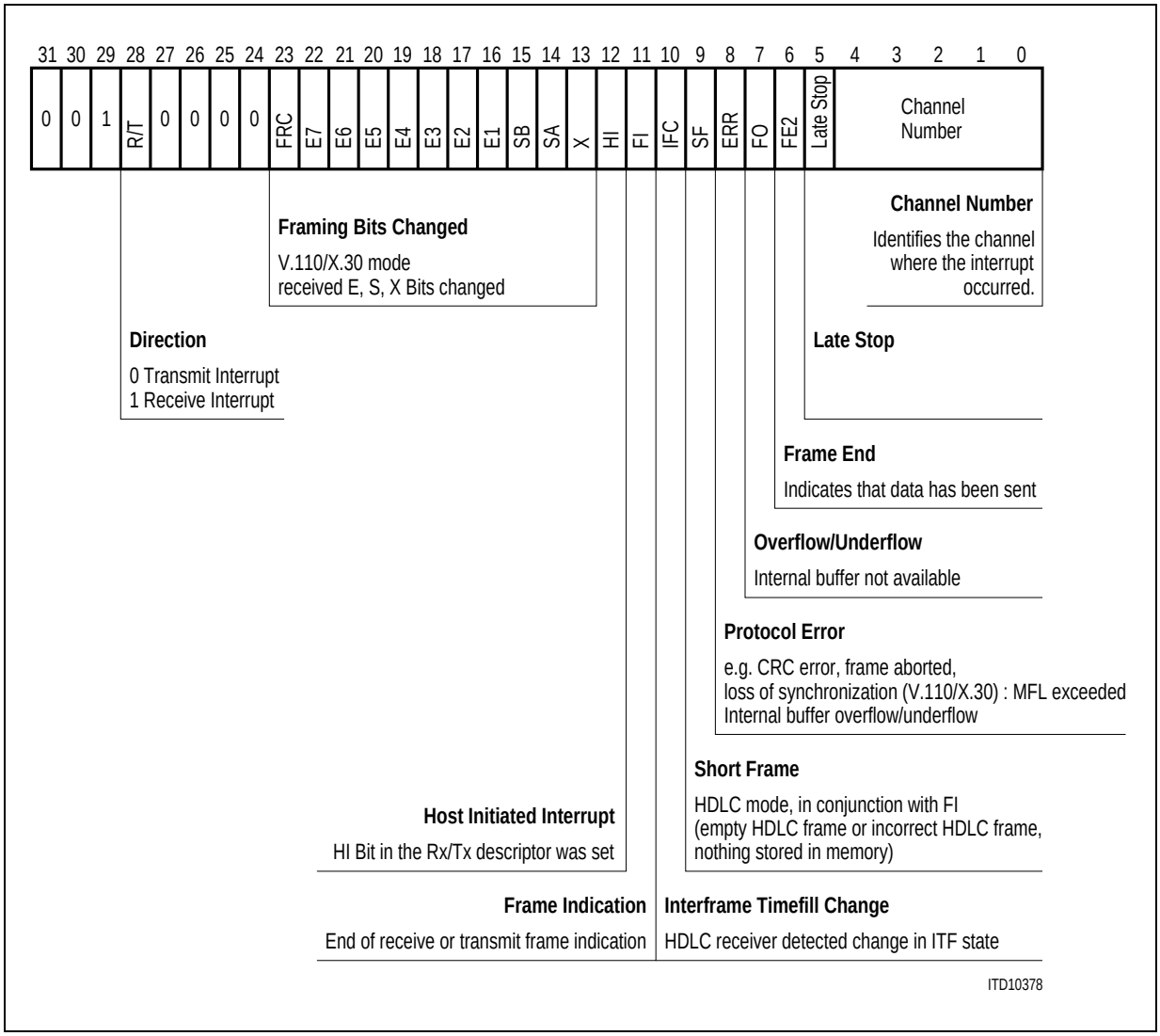


Figure 77
Interrupt Information

12.5 Time Slot Assignment

The time slot assignment is read once after each action request (initiated via bit ARPCM in Command Register CMD) having the action specification bit IN or RES set.

Note: The Time Slot Assignment is not applicable for the LBI block.

It provides the cross reference between the 32 (24) time slots of the PCM highway and the data channels (up to a maximum number of 32). The data channels can be composed of different Rx and Tx time slots, which have individual bit rates. With the concept of subchanneling, MUNICH32X can perform flexible transmission from 8 Kbit/s up to 2.048 Mbit/s per channel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0	0	TTI	Tx Channel Number					Tx Fill Mask								time slot 0
0	0	TTI	Tx Channel Number					Tx Fill Mask								time slot 1
																…
0	0	TTI	TX Channel Number					Tx Fill Mask								time slot 31

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	RTI	Rx Channel Number					Rx Fill Mask								time slot 0
0	0	RTI	RX Channel Number					RX Fill Mask								time slot 1
																…
0	0	RTI	Rx Channel Number					Rx Fill Mask								time slot 31

Fill/Mask Code: For bit rate adaption the fill/mask code determines the number of bits and the position of these bits within the time slot. For all modes the bits selected by Fill/Mask = ‘1’ in the slots of a channel are concatenated, those with Fill/Mask = ‘0’ are ignored/tristated in Rx/Tx direction.

Channel Number: The channel number identifies the data channel. Its transmission mode is described in the respective channel specification.

TTI: Tx Time slot Inhibit; setting this bit to ‘1’ causes MUNICH32X to tristate the Tx time slot. The data is not destroyed but sent in the next not tristated time slot allocated to this channel.

RTI: RX time Slot Inhibit; setting this bit to ‘1’ causes MUNICH32X to ignore the received data in the time slot. The channel is not processed in this time slot.

Note: It is recommended to set the inhibit bits ‘TTI’ and ‘RTI’ on any unused time slot.

12.6 Channel Specification

The channel specification is read once after each action request (initiated via bits ARPCM/ARLBI in Command Register CMD) having the action specification bit IN, RES or ICO set.
Note that RES applies to the channel specifications of all channels, IN and ICO only apply to the channel specification of the channel indicated in the action specification.

PCM Core:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt Mask								NITBS	Channel Command						
FRDA															
FTDA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFLAG							TFLAG/CS	INV	CRC	TRV		FA	Mode		IFTF
FRDA															
FTDA															
0	0	0	0	0	0	0	0	0	0	ITBS					

LBI:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Interrupt Mask								NITBS	Channel Command						
FRDA															
FTDA															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TFLAG							TFLAG/CS	INV	CRC	TRV		FA	**1**	**1**	IFTF
FRDA															
FTDA															
0	0	0	0	0	0	0	0	0	0	ITBS					

Interrupt Mask:

PCM Core:

31	30	29	28	27	26	25	24
FE2	SFE	IFC	CH	TE	RE	FIR	FIT

LBI:

31	30	29	28	27	26	25	24
FE2	0	0	0	TE	RE	FIR	FIT

These bits mask the bits in the interrupt information DWORD according to the table at the end of **Section 12.4** (interrupt bit fields definition).

If an event leads to an interrupt with several bits set (e.g. FI and ERR) masking only a proper subset of them (e.g. ERR) will lead to an interrupt with the nonmasked bits set (e.g., FI). If all bits of an event are masked, the interrupt is suppressed. The interrupt mask is therefore bit specific and not event specific.

NITBS: New ITBS value; if this bit is set the individual Tx buffer size ITBS is valid and a new buffer field of TB is assigned to the channel. In this process first the occupied buffer locations of the channel are released and then according to ITBS a new buffer area is allocated. If there is not enough buffer size in TB (occupied by other channels) the process will be aborted and an action request failure interrupt is generated. After aborting no buffer size is allocated to the channel. For preventing action request failure enough buffer locations must be available. This can be done by reducing the buffer size of the other channels. To avoid transmission errors all channels to be newly configured must be deactivated before processing.0

Note: ITBS has to be set to ‘0’ if NITBS = ‘0’.
NITBS should be set to ‘0’ in conjunction with a transmit abort channel command.

Note: For LBI channels ITBS has to be set to ‘10_H’, if NITBS is set to ‘1’.

Channel Command:

PCM Core:

22	21	20	19	18	17	16
RI	TI	TO	TA	TH	RO	RA

LBI:

22	21	20	19	18	17	16
RI	TI	0	0	TH	0	RA

These bits allow the channel to be initialized, aborted or reconfigured at the serial PCM side as well as at the µP side.

These bits can be decomposed in 3 independent command groups:

	RI, RO, RA	form the receive command group
	TO, TI, TA	the first transmit command group
and	TH	is the second transmit command group.

In the following section, the functionality of these bits is discussed according to the groups.

1. Receive Command Group

– **Receive Clear** (not supported by LBI)
RI = 0, RO = 0, RA = 0 (clears a previous receive abort or receive off condition, affects only the serial interface)
The effect of this command depends on the previous history of the channel
• if the channel was never initialized by a receive initialization command it has no effect
• if it was initialized previously it clears a receive off or receive abort condition set by a previous channel command
• if no receive off or receive abort condition is set it has no effect.

– **Fast Receive Abort** (not supported by LBI)
RI = 0, RO = 0, RA = 1 (clears a previous receive abort or receive off condition, affects only the DMA interface)
This abort is performed in the DMA controller and does not interfere with the reception on the serial interface and the transfer of the data into the Rx buffer. If this abort is detected the current Rx descriptor is suspended with an abort status (RA bit set to ‘1’) followed by a branching to the new descriptor (FRDA) defined in the channel specification of the CCB.
For HDLC, TMB, TMR the rest of a frame which was only partially transferred before suspension of the Rx descriptor is aborted, the new descriptor is related to the next

frame. An interrupt with FI, ERR is issued. For V.110/X.30 and TMA data bits might get lost. An interrupt with ERR is issued.

– **Receive Off** (not supported by LBI)
RI = '0', RO = '1', R = '0' (clears a previous receive abort condition, sets off condition, affects only the serial interface)
This channel command sets the receiver into the receive off condition. The receive channel is disabled completely at the serial interface, i.e. the receive deformatter RD is reset and the receive buffer RB is not accessed for this channel. A currently processed frame (HDLC, TMB, TMR mode) is not properly finished with any status information. The data stored in the RB at that time is still transferred to host memory. After the receive off condition is cleared by another channel command:
  • in HDLC, TMB, TMR (V.110/X.30, TMA) mode the device waits for a new frame (10-octet frame, nothing) to begin and then starts filling RB again. If the receive off command lead to an improper finishing of a frame (data, data), the new frame (data, data) is concatenated with the finished one. To avoid this problem there are two suggestions:
    a) issue a receive abort channel command and wait for 32 (240, 8) bits for this channel to be processed before issuing the receive off command.
    b) wait in the receive off condition until the RB is emptied for this channel (i.e. for at most 8 PCM frames if the MUNICH32X has sufficient access to the shared memory) and leave the receive off condition by a receive initialization command. The receive off channel command is ignored in case of any kind of loop.

– **Receive Abort** (not supported by LBI)
RI = '0', RO = '1', RA = '1' (clears a previous receive off condition, sets a receive abort condition, affects only the serial interface)
This receive channel command sets the receiver into the receive abort condition. In this condition it receives (instead of the normally received bits)
logical '1' bits for HDLC
logical '0' bits for V.110/X.30, TMB, TMR
logical '0' bits for TMA mode
irrespective of the INV bit.

This leads to

  • For HDLC: a currently processed frame is aborted after ≤ 7 received bits for this channel, leading to a RA set in the status of the frame and an interrupt with set FI and ERR bits only or to an interrupt with set SF, FI and ERR bits. If the receiver was in the flag interframe time-fill state it will lead to an interrupt with set IFC bit after ≤ 15 received bits.
  • For V.110/X.30: if the receiver was in the synchronized frame state it will go to the unsynchronized state after ≤ 240 bits and issue a LOSS bit in the status of the current Rx descriptor. It will also issue an interrupt with set ERR bit and (unless all E-, S-, X-bits were '0' previously) issue one or two interrupts with FRC set and having all E-, S-, X-bits at '0' in the last one.

- For TMB: a currently processed frame is aborted after ≤ 15 received bits for this channel, leading to an interrupt with FI set but ERR on 0, the status of this frame is always 00_H.
- For TMR: a currently processed frame is aborted after ≤ 31 received bits for this channel, leading to an interrupt with FI set but ERR on 0, the status of this frame is always 00_H.

Note 1: It is recommended to clear the receive abort condition via a receive off command for V.110/X.30 mode, the TMB and the TMR mode.

Note 2: After issuing a receive abort channel command it is advisable to stay in this condition during at least 16, 240, 16, 32, 8 bits of the channel for HDLC, V.110/X.30, TMB, TMR, TMA respectively.

– **Receive Jump**
RI = '1', RO = '0', RA = '0' (clears a previous receive abort or receive off condition, affects only the DMA interface)
During normal operation branching to a new descriptor (FRDA) is possible without interrupting the current descriptor and aborting the received frame (HDLC, TMB, TMR) or received data (V.110/X.30, TMA).
The DMA controller will proceed finishing the current receive descriptor as usual either with a frame end condition or with the corresponding data buffer completely filled and afterwards branch to the new descriptor specified by FRDA. Thus a received frame may be splitted on 'old' and 'new' descriptors.

– **Receive Initialization**
RI = '1', RO = '0', RA = '1' (clears a previous receive abort or receive off condition, affects the DMA and serial interface)
Before the MUNICH32X has got a receive initialization command it will not receive anything properly in a channel. This command should therefore be the first channel command after a reset for a channel to be used. FRDA is then the address of the starting point of the Rx descriptor chaining list.
If the command is issued during normal operation it only affects the DMA interface. The current Rx descriptor is suspended without writing the second DWORD with the status, no interrupt is generated. For HDLC, TMB, TMR the rest of a frame which was only partially transferred before the suspension of the Rx descriptor is aborted, the new descriptor (FRDA) is related to the next frame.
For V.110/X.30 and TMA data bits might get lost.

- General Notes on Receive Commands:

1. After a pulse at the reset pin a channel having a time slot with RTI = '0' should be issued receive off commands until it is used.
2. When the channel is intended to be used, a receive initialize command should be issued before using any other receive channel command.
3. To shut down a channel in receive direction, it should first be set into the receive abort condition for the time specified there and then set into the receive off condition.

4. Before changing the MODE, CRC, CS, TRV, INV, TFLAG bits of a channel or its RTI or time slot assignment or its fill/mask bits it should have been shut down. The bits should be changed while issuing the receive off command.
5. To revive a channel after it has been shut down, the receive initialization command should be used.
6. To switch to a new starting point of a Rx descriptor chain one should preferably use the receive jump command, only exceptionally the fast receive abort command and never the receive initialize command.
7. To issue channel commands not affecting the receive side one should issue
   – a receive clear command if neither a receive off nor a receive abort condition is set
   – a receive off command if a receive off condition is set
   – a receive abort command if a receive abort condition is set.
8. Combinations of the bits RI, RO, RA not in this description are reserved and are not allowed to be used.

2. First Transmit Command Group

– **Transmit Clear** (not supported by LBI)
  TI = ‘0’, TO = ‘0’, TA = ‘0’ (clears a previous transmit abort or transmit off condition, affects only the serial interface)
  • if the channel was never initialized by a transmit initialization command it has no effect
  • if it was initialized previously it clears a transmit off or transmit abort condition set by a previous channel command
  • if no transmit off or transmit abort condition is set it has no effect
– **Fast Transmit Abort** (not supported by LBI)
  TI = ‘0’, TO = ‘0’, TA = ‘1’ (clears a previous transmit abort or transmit off condition, affects only the DMA interface)
  This abort is performed in the DMA controller and does not interfere with the current transmission on the serial interface and the transfer between the TF and TB. If this abort is detected the current descriptor is suspended and the frame or data transferred to the TB is aborted. The next frame beginning in the Tx descriptor (FTDA) defined in the channel specification of the CCB will be started immediately.

  For HDLC, TMB, TMR the first part of the frame of the suspended descriptor is sent and append by
  011 1111 1111 1111 for HDLC
  at least 00_H for TMB
  at least $00\ 00_H$ for TMR
  Afterwards the next frame is started.
  For V.110/X.30 three 10-octet frames with errors in the synchronization pattern are sent after the data of the suspended descriptor, afterwards the next data are sent in correct frames.
  For TMA a TFLAG (FA = ‘1’) or FF_H (FA = ‘0’) is sent in at least one time slot after the data of the suspended descriptor, afterwards the next data are sent.

– **Transmit Off** (not supported by LBI)
TI = '0', TO = '1', TA = '0' (clears a previous transmit abort condition, sets a transmit off condition, effects only the serial interface)
The Tx channel is disabled immediately, i.e. the Tx formatter is reset and the Tx buffer is not accessed for this channel. The output time slots are tristated. Upon leaving the transmit off mode the transmit link list must be initialized by a transmit reinitialize command. Otherwise the transmission will be started with the remaining data still stored in TB and continue with the old link list. If a loop condition is set the transmit off does not reset the Tx formatter, it only tristates the serial output line.
After the transmit off condition is cleared by the transmit initialize command.
• In HDLC, TMB, TMR, V.110/X.30 the device starts with the interframe time-fill
7E for HDLC and IFTF = '0'
FF for HDLC and IFTF = '1'
00 for TMB, TMR, V.110/X.30
and then with the frame in the descriptor at FTDA. For V.110/X.30 this descriptor **must** have the V.110-bit set and point to the E-, S-, X-bits, the data are then at the next Tx descriptor.
• In TMA mode the device starts with the interframe time-fill
TFLAG for FA = '1'
FF_H for FA = '0'
and then with the data in the descriptor at the FTDA.

Note: It is recommended to set bit-field 'Mode' located in the first DWORD of the Channel Specification to '00' (Transparent Mode A) together with any 'Transmit Off ' channel command. This ensures octet to timeslot aligned transmission after re-initialising the channel again.

– **Transmit Abort** (not supported by LBI)
TI ='0', TO = '1', TA = '1' (clears transmit off condition, sets transmit abort condition, affects only the serial interface)
This abort is performed in the transmit formatter at the serial interface. The currently transmitted frame is aborted by the sequence:
011 1111 1111 1111 for HDLC
00_H for TMB
0000_H for TMR
3 frames with erroneous synchronization pattern for V.110/X.30
TFLAG for TMA, FA = '1'
FF for TMA, FA = '0'.

– Afterwards or, if no frame is currently sent, directly inter frame time fill:
7E for HDLC and IFTF = '0'
FF for HDLC and IFTF = '1'
00 for TMB, TMR, V.110/X.30
TFLAG for TMA, FA = '1'
FF for TMA, FA = '0'

is sent.
During transmit abort the TF does not access the Tx buffer. The handling of the link list is not affected by the transmit abort, i.e. the device keeps the TB full. When the transmit abort is withdrawn, the Tx formatter continues the transmission with the data stored in TB. In the case of HDLC or TMB or TMR mode the remaining data of the aborted HDLC or TMB frame is sent as a new independent frame. To avoid this problem the link list must be reinitialized by a transmit initialization command together with the revoking of the transmission abort.
Another proper use of the transmit abort command consists in setting the last descriptor of the last frame to be transmitted with HOLD = '1' and waiting for the device to poll the HOLD bit (ITBS + 2) times where ITBS is the number of DWORDs assigned to this channel currently. Afterwards TB is empty and the transmit abort then issued does not abort a currently sent frame. The same procedure can also be used for the transmit off command.

– **Transmit Jump**
TI = '1', TO = '0', TA = '0' (clears a transmit off and transmit abort condition, affects only the DMA interface)
This bit is set only during normal operation. Then the MUNICH32X branches to the transmit descriptor (FTDA) specified in the CCB after finishing the current Tx descriptor without interrupting or aborting the transmitted frame.
The DMA controller will proceed finishing the current transmit descriptor as usual and afterwards branch to the new descriptor specified by FTDA. If the current descriptor does not include a frame end (FE = 0) (HDLC, TMB, TMR) the DMA controller will link the following data section(s) of the 'new' descriptor chain to the opened frame. This may generate unexpected frames.

– **Transmit Initialization**
TI = '1', TO = '0', TA = '1' (clears a previous transmit abort condition, affects the DMA interface and the serial interface)
Before the MUNICH32X has received a transmit initialization command, it will not transmit correctly on the channel. This command should therefore be the first channel command after a pulse at the reset pin for a channel.
FTDA is then the address of the starting point of the Tx descriptor for chaining list. In this case the transmit initialize command should be accompanied by the NITBS bit set and a reasonable value for ITBS (0 < ITBS < 64).
If the command is issued during normal operation it only affects the DMA. The MUNICH32X stops processing of the current link list and branches to the Tx descriptor at the FTDA address. The data stored in the TB are discarded and the TB is filled with the data of the new descriptor.

3. Second Transmit Command Group

– **Transmit Hold** (not supported by LBI)
TH; setting this bit causes the device to finish transmission of the current frame (HDLC, TMB, TMR mode) the current data (TMA mode) or leads to an abort with 3 frames with '0' bits (V.110/X.30 mode). Afterwards

for		
	HDLC mode and IFTF = '1'	FF_H fill characters
	HDLC mode and IFTF = '0'	$7E_H$ fill characters
	V.110/X.30-mode	00_H fill characters
	TMA mode and FA ='1'	TFLAG fill characters, if no poll access was done; else: FF_H fill characters
	TMA mode and FA = '0'	FF_H fill characters
	TMB/TMR	00_H fill characters

are sent until TH is withdrawn by a further action specification affecting the channel specification of this channel.
Afterwards no further access to the TB from TF is done, therefore no further data are fetched from host memory and the polling of the HOLD bit in the Tx descriptor stops. In order to send the required frames/data before TH is active, the corresponding procedure as described for the transmit abort command should be used.

• General Notes on Transmit Commands:

1. After reset, a channel having a time slot with TTI = '0' should issue transmit off commands and TH = '1' until it is required to be used.
2. When it is supposed to be used it should be issued a transmit initialization command and TH = '0' before using any other Tx channel commands (together with NITBS = '1', ITBS ≠ '0').
3. To shut down a channel in transmit direction one should first set it into the transmit abort condition or use the TH bit with the proper procedure. One should leave it in that condition for 32, 240, 32, 32, 8 bits for HDLC, V.110/X.30,TMB, TMR, TMA respectively and then set it into the transmit off condition.
4. Before changing the MODE, CRC, CS, TRV, INV, TFLAG bits or TTI or time slot assignment or the fill/mask bits or the ITBS the channel should be shut down. The bits should be changed while issuing the transmit off command.
5. To revive a channel after it has been shut down one should use the transmit initialization command.
6. For V.110/X.30-mode the first descriptor after reviving from shut down or initialization after reset **must** have the V.110-bit set and contain the E-, S-, X-bits.
7. To switch to a new starting point of a Tx descriptor chain one should preferably use the transmit jump command, only exceptionally the fast transmit abort command and never the transmit initialize command.

8. To issue channel commands not affecting the transmit side one should issue
   – TH with the last set value
   – a transmit clear command if neither a transmit off nor a transmit abort condition is set
   – a transmit off if a transmit off condition is set
   – a transmit abort if a transmit abort condition is set.
9. Bit combinations in the first Tx command group not described are reserved.
10. Set NITBS = ‘1’ preferably in conjunction with a transmit initialize and transmit clear command if TB is to be newly configured, otherwise program NITBS = ‘0’.
11. For new configuration of a channel (‘de-allocation of ITBS’) consider two cases:
    a) if the channel has not been initialized yet: program NITBS = ‘0’, ITBS = ‘0’,
    b) after successful initialization: program NITBS = ‘1’, ITBS = ‘0’,
    both in conjunction with transmitt off

TFLAG: Transparent mode Flag; these bits are only used in the transparent mode A and constitute the fill code for flag stuffing and for flag filtering. These bits must be set to ‘0’ if subchanneling is used in transparent mode A. Bit No. 15 is the first bit of the flag to be received/transmitted.

CS: CRC Select; only used in HDLC mode. Setting this bit to ‘1’ causes the MUNICH32X to transfer the CRC bits to the data section in the shared memory. In receive direction the CRC check is carried out whereas in transmit direction the CRC generation is suppressed, see **Chapter 4: Detailed Protocol Description** for more details.

INV: Inversion; If this bit is set, all data of the channel transmitted or received by the MUNICH32X is inverted.

CRC: Cyclic Redundancy Check; in HDLC mode this bit determines the CRC generator polynomial: When the CRC bit is set to ‘1’, the 32-bit CRC is performed, otherwise the 16-bit CRC. For TMB/TMR mode this bit distinguishes:

TMB: CRC = ‘0’
TMR: CRC = ‘1’

Note: For all other modes this bit has to be set to ‘0’.

TRV: Transmission Rate of V.110/X.30. These signals determine the number of repeated D-bits in a V.110/X.30 frame.

Table 28

TRV	No. of Repetitions	Transmission Rate
00	7	600 bit/s
01	3	1200 bit/s
10	1	2400 bit/s
11	0	4.8, 9.6, 19.2, 38.4 Kbit/s

Note: In the other modes these bit fields must be programmed to '00'.

FA: Flag Adjustment selected (in HDLC mode) or flag filtering (selected in transparent mode A only if all fill/mask bits of the corresponding slots are '1'). In all other modes this bit must be set to '0'. If flag adjustment is selected in HDLC mode the number of interframe time-fill characters is FNUM minus one eighth of the number of zero insertions in the frame proceeding the interframe time-fill and belonging to the same transmit descriptor as FNUM.
If flag filtering is selected and fills a physical time slot in transparent mode A the flag specified in TFLAG is recognized and extracted from the data stream. In transmit direction the flag TFLAG is sent in all exception conditions, i.e. abort, idle state etc.; if flag filtering is not selected, '1'-bits are sent in this case. Flag filtering is only allowed if all fill/mask codes are set to '1', i.e. subchanneling is not allowed.
If flag filtering is not selected the bits in TFLAG have to be programmed to '0' for TMA.

MODE: Defines the transmission mode:
00: Transparent mode A
01: Transparent mode B or transparent mode R.
10: V.110/X.30 mode
11: HDLC mode

IFTF: Interframe Time-Fill; this bit determines the interframe time-fill for HDLC mode:
IFTF = '0': $7E_H$ characters are sent as interframe time-fill
IFTF = '1': FF_H characters are sent as interframe time-fill.

FRDA: First Rx Descriptor Address; points to the beginning of the Rx data chaining list.
This descriptor is only interpreted with a fast receive abort or a receive jump or a receive initialization command. It is read but ignored with any other receive channel command.

FTDA: First Tx Descriptor Address; points to the beginning of the Tx data chaining list.
This descriptor is only interpreted with a fast transmit abort or a transmit jump or a transmit initialization command. It is read but ignored with any other transmit channel command.

ITBS: Individual Tx Buffer Size; for undisturbed transmission the on-chip Tx buffer with a total size of 64 DWORDs stores the data before formatting and transmitting. The individual buffer size specifies the part of the on chip transmit buffer allocated to the channel. This allows a variable data buffer size if NITBS = '0', ITBS has to be programmed to '0' also; it is then read but ignored. (see **Chapter 3: Basic Functional Principles**).

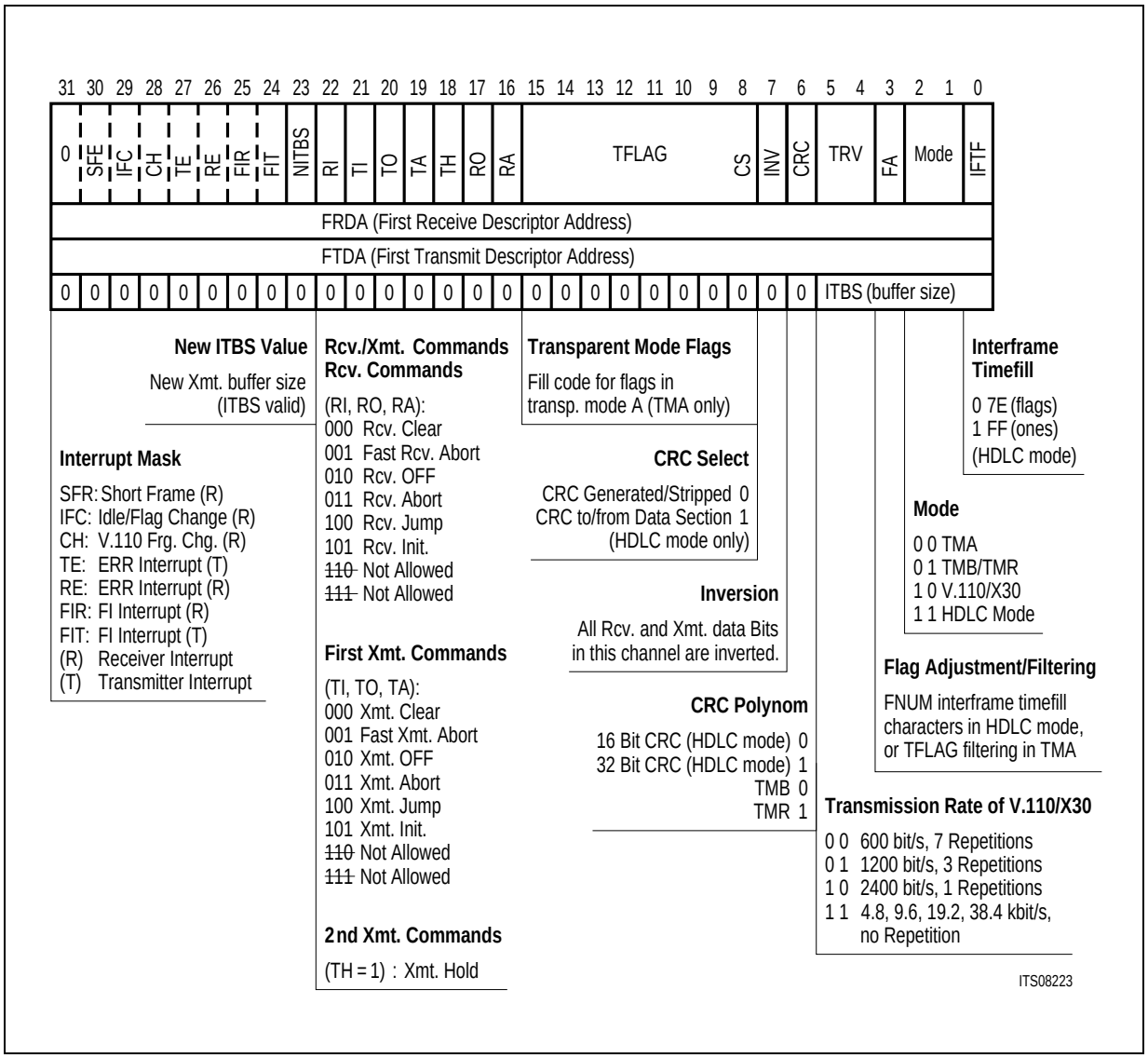


Figure 78
Channel Specification

12.7 Current Receive and Transmit Descriptor Addresses

31 16	15 0
Current Rx Descriptor Address Channel 0	
. . .	
Current Rx Descriptor Address Channel 31	
Current Tx Descriptor Address Channel 0	
. . .	
Current Tx Descriptor Address Channel 31	

For easier monitoring of the link lists the addresses of the just processed descriptors are written into the CCB. The MUNICH32X changes the current descriptor address at the same time when it branches to the next descriptor.

12.8 Receive Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	HOLD	HI	NO												
Next Rx Descriptor Pointer															
Rx Data Pointer															
FE	C	0	BNO												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000_H															
Next Rx Descriptor Pointer															
Rx Data Pointer															
Status								00_H							

The Rx Descriptor consists of 4 DWORDs. During run-time, the MUNICH32X reads the Rx Data Pointer. After the MUNICH32X accesses an Rx descriptor, it updates the appropriate current Rx descriptor address in the CC Block.

The Rx Descriptor is accessed in the following order (assuming a normal complete access with HOLD = ‘0’, i.e. no polling):

1. Access n: Read the first descriptor DWORD
2. Access n + 1: Read the next Rx descriptor pointer
3. Access n + 2: Read the Rx data pointer
4. Access n + 3: Write the current Rx descriptor address to Control and Configuration Block (CCB)
5. Access n + 4: Data Transfer
6. Access n + 5: Write the fourth descriptor DWORD (number of received data bytes, status information)
7. Access n + 5: Handle selected interrupts

*Note: The MUNICH32X does **not** update the fourth DWORD if the receive initialization command is used during normal operation (see **Chapter 12.5**)*

The descriptor bit fields have the following meaning:

HOLD: Setting the HOLD bit by the host prevents the device from branching to the next descriptor. The current data section is still filled.

– Afterwards the fourth descriptor DWORD is written by the MUNICH32X. For HDLC, TMB, TMR the FE and C bits are set. If the frame could not completely be stored into the data section the RA bit is set in the status. An interrupt with set FI bit is generated, and in case the frame was aborted, the ERR bit is also set.
For TMA, V.110/X.30 the C bit and the RA bit are set and an interrupt with set ERR but with FI = ‘0’ is generated.
– Afterwards the device starts polling the HOLD bit. Received data and received events normally leading to interrupts (with RT = 1) are discarded until HOLD = ‘0’ is detected. Each 1 … 4 byte data word or interrupt event normally leading to an access now results in a poll cycle.
Whenever HOLD = ‘1’ is detected the next Rx descriptor address is read but ignored.
– When HOLD = ‘0’ is detected
  • for HDLC, TMB, TMR the device continues to discard data until the end of a received frame or an event leading to an interrupt (with RT = ‘1’) is detected. Afterwards the next received frame is transferred into the next Rx descriptor. Interrupts are also generated again.
  • For V.110/X.30, TMA the device puts the next data into the next Rx descriptor. Interrupts are also generated again.

The HOLD condition is also discarded upon detection of a receive jump, fast receive abort or receive initialization command. The MUNICH32X then branches to the Rx descriptor determined by FRDA even though the HOLD bit in the current Rx descriptor may still be ‘1’.

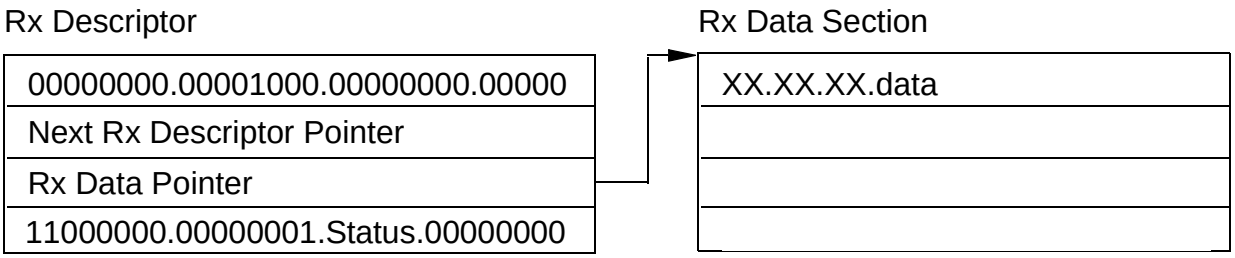
For a description of the complete polling process of the MUNICH32X refer to the descriptions of MODE2 and TXPOLL registers.

HI: Host initiated interrupt; if the HI bit is set, MUNICH32X generates an interrupt with set HI bit after receiving all data bytes.

NO: This defines the byte size of the receive data section allocated by the host. Because MUNICH32X always writes DWORDs the number of bytes (data section size) must be a multiple of 4 and greater or equal to 4. The maximum data section size is 8188 bytes.

After reception of an HDLC frame with a data byte number not divisible by 4 the MUNICH32X first transfers the greatest entire (number of data bytes/4) in DWORDs. Then the remainder of the data bytes is transferred in another DWORD, where the non-significant bytes are filled with random values. They should not be interpreted.

For example a HDLC frame with one data byte is received:



The data bytes are stored in the Rx data section in little endian format .

FE: Frame End: The frame end bit is set to ‘1’ only in HDLC, TMB, TMR mode and indicates that a receive frame has ended in this Rx descriptor. For TMA and V.110/X.30 the bit is always ‘0’.
FE = ‘0’ in HDLC, TMB, TMR mode means that frame continues in the next Rx descriptor or that it filled the current receive data section exactly (BNO = NO). In this case, the next Rx descriptor will have FE = ‘1’, C = ‘1’, BNO = ‘0’, and no data bytes are stored in the corresponding data section.

Note: For LBI FE is set to ‘1’ in general, since only HDLC is supported.

C: This bit field is set by MUNICH32X if
- it completes filling the data section (BNO = NO) ⇒ FE = '0', Status = 00_H
- it was aborted by a fast receive abort channel command ⇒ Status = 02_H
- for HDLC, TMB, TMR if the end of a frame was stored in the receive data section ⇒ FE = '1', status gives the receive status determined by Rx descriptor (interrupt with set FI bit is generated)
- for V.110/X.30 mode if the 3 contiguous frames with errors in the synchronization pattern are received ⇒ FE = '0', status = 20_H or status = 21_H (interrupt with set ERR bit)
- for V.110/X.30 mode if the data could not be transferred to the shared memory due to Rx buffer inaccessibility ⇒ FE = '0', Status = 01_H or Status = 21_H (interrupt with set ERR bit).

C indicates that the fourth DWORD of the Rx descriptor was written by the MUNICH32X. Afterwards the MUNICH32X writes the next Rx descriptor address into CCB. Then it branches to this descriptor immediately.

BNO: MUNICH32X writes the number of data bytes it has stored in the current data section into BNO.

Status: The MUNICH32X writes the status information into the Status byte whenever it sets the C bit field. If the status information does not equal 00_H or 40_H, an interrupt with ERR bit set is generated. The status then supports locating or analyzing the receive error.
The following table gives a general overview over the different status bits in relation to the channel modes.

Status Information

PCM Core:

	7							0
	0	SF	LOSS	CRCO	NOB	LFD	RA	ROF
HDLC (CS = 0)	0	NI	0	ILN	IL	I	I	I
HDLC (CS = 0)	0	0	0	ILN	IL	I	I	I
V.110/X.30	0	0	I	0	0	0	IF	I
TMB	0	0	0	0	IL	I	IF	I
TMR	0	0	0	0	IL	I	IF	I
TMA	0	0	0	0	0	0	IF	0

LBI:

	7							0
	0	0	0	0	0	0	RA	0
HDLC	0	0	0	0	0	0	I	0

Where ‘0’ means that in the corresponding mode the bit is always ‘0’.

NI means the bit may be ‘1’ or ‘0’ but does not cause an interrupt with ERR bit set.

ILN means that it may be ‘1’ or ‘0’ but should not be evaluated if LFD or NOB is also ‘1’.

IL means that it may be ‘1’ or ‘0’ but should not be evaluated if LFD = ‘1’.

I means that it may ‘1’ or ‘0’.

IF means that it may be ‘1’ only after a fast receive abort channel command or detection of a HOLD bit in the current receive descriptor.

I, IF, IL, ILN lead to an interrupt with ERR bit set.

Note: For HDLC, TMB, TMR the status word is only valid if the FE bit is set.

The meaning of the individual status bits is as follows:

SF = ‘1’ (HDLC mode with CS = ‘0’ only):
The device has received a frame which includes:
≤ 32 bits between start flag and end flag or end abort flag for CRC16
≤ 48 bits between start flag and end flag or end abort flag for CRC32,
i.e. BNO was 1 or 2.

LOSS = ‘1’ Three contiguous frames with errors in the synchronization pattern were detected.

CRCO = ‘1’ A frame with a CRC error was detected. CRCO = ‘0’ means the frame had no CRC error.

NOB = ‘1’ A frame whose bit contents were not divisible by 8 was detected.
NOB = ‘0’ means that the frame contents were divisible by 8.

LFD = ‘1’ Long frame detected. If this bit is set a frame whose bit contents were > MFL was detected and aborted. The reception will be continued as soon as a flag is recognized.

RA = ‘1’ Receive Abort; this bit indicates that
for HDLC: the frame was ended by an abort flag ($7F_H$) or by a receive abort command or a fast receive channel command or by a HOLD bit set in the current Rx descriptor.
for V.110/X.30, TMB, TMR, TMA: the frame or data were aborted by a fast receive abort channel command or a HOLD bit set in the current Rx descriptor.

ROF = ‘1’ An overflow of the internal receive buffer RB has occurred and lead to a loss of data.

Note: If ROF without FO interrupt is generated for a channel
- *for HDLC, TMB, TMR only the last part of one frame has been lost.*
- *For V.110/X.30 only data but no status information (change E-, S-, X-bits, Loss) has been lost.*

Note: In case of multiple errors all relevant bits are set.
In case of ROF = ‘1’ only the error conditions of the frame within which the overflow occurred are reported. Later frames that are aborted do not change the status.

Rx Data Pointer: This 32-bit pointer contains the start address of the receive data section.

Rx Descriptor Pointer: This 32-bit pointer contains the start of the next Rx descriptor. It is not used if a receive jump, fast receive abort or receive initialize command is detected while the MUNICH32X still writes data into the current receive descriptor or polls the HOLD bit. In this case FRDA is used as a pointer for the next Rx descriptor to be branched to.

12.9 Transmit Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE	HOLD	HI	NO												
Next Tx Descriptor Pointer															
Tx Data Pointer															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V.110	0	0	NO13	CSM	FNUM										
Next Tx Descriptor Pointer															
Tx Data Pointer															

A Transmit Descriptor consists of 3 DWORDs. During run-time, the MUNICH32X reads the Transmit Data Pointer. After accessing an Tx descriptor, it updates the appropriate current Tx descriptor address in the CC Block.
The Tx Descriptor is accessed in the following order (assuming a normal complete transfer with HOLD = ‘0’, i.e. no polling):

1. Access n: Read the first descriptor DWORD
2. Access n + 1: Read the next Tx descriptor pointer
3. Access n + 2: Read the Tx data pointer
4. Access n + 3: Write the current Tx descriptor address to Control and Configuration Block (CCB)
5. Access n + 4: Data Transfer
6. Access n + 5: Handle selected interrupts

The descriptor bit fields have the following meaning:

FE: Frame End; this bit is valid in all modes.
It indicates that after sending the data in the transmit data section
– the device generates an interrupt with FI bit set for HDLC, TMB, TMR, TMA
ERR bit set for V.110/X.30
– the device then sends (not supported by LBI)

• (FNUM + 1) × $7E_H$	for HDLC, IFTF = '0'
• $7E_H$, (FNUM – 1) × FF_H, $7E_H$	for HDLC, IFTF = '1', FNUM ≥ 1
• $7E_H$	for HDLC, IFTF = '1', FNUM = 0
• (FNUM + 1) × 00_H	for TMB, TMR (FNUM ≥ 1)
• 000_H	for TMR, FNUM = 0
• (FNUM + 1) × TFLAG	for TMA, FA = '1'
• (FNUM +1) × FF_H	for TMA, FA = '0'
• 3 frames with synchronization errors	for V.110/X.30

before starting with the data of the next Tx descriptor. If the data of the next Tx descriptor are not available in time (e.g., because the descriptor has FE **and** HOLD set) the device sends the interframe time-fill indefinitely.

HOLD: If the MUNICH32X detects a HOLD bit it
– generates an interrupt with bit ERR set in the case of frame end bit FE = '0' and/or V.110/X.30 mode was selected in the transmit descriptor;
in HDLC mode the corresponding data section is sent and terminated by an abort sequence;
In the case of FE = '1' and not V.110/X.30 mode selected:
– the device then sends at least (not supported by LBI)
• (FNUM + 1) × $7F_H$ for HDLC, IFTF = '0'
• $7E_H$, FNUM × FF_H for HDLC, IFTF = '1'
• (FNUM + 1) × 00_H or TMB, TMR (FNUM ≥ 1)
• 0000_H for TMR, FNUM = 0
• (FNUM + 1) × TFLAG for TMA, FA = '1'
• (FNUM + 1) × FF_H for TMA, FA = '0'
• (in TMA mode, after TFLAGs, '1's are sent until next sync. pulse)
• three frames with synchronization errors for V.110/X.30.

– It polls the HOLD bit and the next transmit descriptor address, but does not branch to a new descriptor until the HOLD bit is reset. The next transmit descriptor address is read but not interpreted as long as HOLD = '1'. Therefore it can be changed together with setting HOLD = '0'.
The poll mechanism depends on the settings in registers MODE2 and TXPOLL (automatic polling not supported by LBI).

– The device sends interframe time-fill until HOLD = '0' is polled.
The HOLD condition is also discarded if a transmit jump, fast transmit abort or transmit initialization command is detected during the polling. The

MUNICH32X then branches to the Tx descriptor determined by FTDA even though the HOLD bit of the current Tx descriptor may still be '1'.

HI: Host initiated Interrupt; if the HI bit is set, the MUNICH32X generates an interrupt with set HI bit after transferring all data bytes.

NO: This byte number (together with NO13) defines the number of bytes stored in the data section to be transmitted. A Tx descriptor and the corresponding data section must contain at least either one data byte or a frame end indication. Otherwise an interrupt with set ERR bit is generated.

V.110: This bit indicates that in the corresponding data section the E-, S- and X-bits of the following V.110/X.30 frame are stored. The MUNICH32X reads these bits and inserts them into the next possible V.110/X.30 frame. The data section may contain only the two bytes specified in the next figure (in little endian format).
The first Tx descriptor after a transmit initialization channel command **must** have the V.110 bit set if it revives the channel from a transmit off condition or after a reset.
Not supported by LBI.

NO13: This bit field is the MSB of NO.

Tx Descriptor Data Section in V.110 Mode (Not supported by LBI.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E7	E6	E5	E4	E3	E2	E1	SB	SA	X	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000_H															

CSM: CRC Select per Message: This bit is only valid in HDLC mode with CS = 0 and only in conjunction with the FE bit set. If set, it means that no FCS is generated automatically for the frame finished in this transmit descriptor.

FNUM: FNUM denotes the number of interframe time-fill characters between 2 HDLC or TMB frames. For X.30/V.110 these bits have to be programmed to '0'.

FNUM = 0 means that after the current frame only 1 character ($7E_H$ for HDLC and 00_H for TMB, 000_H for TMR, TFLAG, TFLAG for TMA, FA = '1'; FF_H for TMA, FA = '0') is sent before the following frame (shared flags).

FNUM = 1 means that after the current frame 2 characters ($7E_H$ $7E_H$ for HDLC and 00_H 00_H for TMB and TMR, TFLAG, TFLAG for TMA, FA = '1'; FF FF_H for TMA, FA = 0) are sent before the following frame (non shared flags).

FNUM = 2 means that after the current frame 3 characters ($7E_H$ $7E_H$ $7E_H$ (IFTF = '0') or $7E_H$ FF_H $7E_H$ (IFTF = '1') for HDLC and 00_H 00_H 00_H for TMB and TMR, TFLAG, TFLAG, TFLAG for TMA (FA = '1'); FF FF FF_H for TMA, (FA = 0)) are sent.

FNUM = k means that after the current frame k + 1 characters are sent

(k + 1) times $7E_H$	for ITFT = '0' and HDLC
$7E_H$, (k – 1) times FF_H, $7E_H$	for ITFT = '1' and HDLC
(k + 1) times 00_H	for TMB, TMR
(k + 1) times TFLAG	for TMA, FA = '1'
(k + 1) times FF_H	for TMA, FA = '0'.

For HDLC mode, FNUM is reduced by one eighth of the number of zero insertions, if FA is set. If the reduction would result in a negative number of interframe time-fill characters it is programmed to '0'.

Tx Data Pointer: This 32-bit pointer contains the start address of the Tx data section. Although MUNICH32X works only DWORD oriented, it is possible to begin a Tx data section at an odd address. The two least significant bits (ADD) of the Tx data pointer determine the beginning of the data section and the number of data bytes in the first DWORD of the data section, respectively.

ADD: 00 = 4 bytes
01 = 3 bytes
10 = 2 bytes
11 = 1 byte

MUNICH32X reads the first DWORD and discards the unused least significant bytes. The NO establishes (determines) the end of the data section, whereas the remainder of I (NO ADD)/4 I defines the number of bytes in the last DWORD of the data section.

MUNICH32X reads the last DWORD and discards the unused most significant bytes of the last DWORD.

If the first access is the same as the last access, ADD specifies the beginning of the data section and NO the number of data bytes in the DWORD. All unused bytes are discarded.

For example: 1) ADD = 01, NO = 8, byte swap bit field CONF.LBE = ‘0’

11	10	01	00	
byte 2	byte 1	byte 0	–	
byte 6	byte 5	byte 4	byte 3	3 DWORDs are read
–	–	–	byte 7	

2) ADD = 00, NO = 8

11	10	01	00	
byte 3	byte 2	byte 1	byte 0	
byte 7	byte 6	byte 5	byte 4	2 DWORDs are read
–	–	–	–	

3) ADD = 10, NO = 1

11	10	01	00	
–	byte 0	–	–	
–	–	–	–	1 DWORD is read!
–	–	–	–	

Next Tx Descriptor Pointer:

This 32-bit pointer contains the start address of the next Tx descriptor. After sending the indicated number of data bytes, MUNICH32X branches to the next Tx descriptor to continue transmission. The Tx descriptor is read entirely at the beginning of transmission and stored in an on-chip memory. Therefore all information in the next descriptor must be valid when MUNICH32X branches to this descriptor when HOLD = ‘0’. For HOLD = ‘1’ the next Tx descriptor pointer is polled together with HOLD; the next Tx descriptor must be valid, when HOLD = ‘0’ is polled.

It is not used if a transmit jump, fast transmit abort or transmit initialization channel command is detected while the MUNICH32X still reads data from the current Tx descriptor or polls the HOLD bit. In this case FTDA is used as a pointer for the next Tx descriptor to be branched to.

12.10 Serial PCM Core DMA Priorities

The following table shows the prioritization of queueing DMA cycles for serial PCM core accesses to the internal system bus.

Priority	Interrupt
Highest priority	Receive link list including accesses to the descriptors
–	Transmit link list including accesses to the descriptors
Lowest priority	Configuration of a channel (action requests)

12.11 Interrupt Queues Overview

This chapter provides an overview about all MUNICH32X interrupt vectors and the different groups of interrupt queues located in the host memory.

The different vector sources can be distinguished by decoding the most significant 8 bits of each vector.

12.11.1 Serial PCM Core Interrupts

The first group consists of the *Serial PCM Core Interrupts.* Different vectors for receive and transmit direction are written into dedicated queues.

Table 29
Serial PCM Core Interrupt Vectors

Interrupt Vector Name	Short Description	Reference Page
RX_IV	Serial PCM Core Interrupt Vector Rx Direction	249
TX_IV	Serial PCM Core Interrupt Vector Tx Direction	250

The corresponding queues are configured via registers listed below.

Table 30
Serial PCM Core Interrupt Queues

Queue Name	Control Registers, Register Name	Offset Address
RIQ Receive Interrupt Queue	**RIQBA** Receive Interrupt Queue Base Address **RIQL** Receive Interrupt Queue Length	38_H $3C_H$
TIQ Transmit Interrupt Queue	**TIQBA** Transmit Interrupt Queue Base Address **TIQL** Transmit Interrupt Queue Length	30_H 34_H

For detailed description of interrupt vectors refer to **Chapter 12.3**.

12.11.2 LBI DMA Controller Interrupts

This group consists of the *LBI DMA Controller Interrupts* which are generated by the LBI DMA Controller (refer to **Chapter 6.5**). Vectors for receive and transmit direction are written to dedicated queues.

Table 31
LBI DMA Controller Interrupt Vectors

Interrupt Vector Name	Short Description	Reference Page
LDMA_IV (receive)	LBI DMA Interrupt Vector Rx Direction	156
LDMA_IV (transmit)	LBI DMA Interrupt Vector Tx Direction	156

Receive and transmit direction is distinguished by bit ‘R/T’ in vector LDMA_IV.

The corresponding queues are configured via registers listed below.

Table 32
LBI DMA Controller Interrupt Queues

Queue Name	Control Registers, Register Name	Offset Address
LRIQ LBI Receive Interrupt Queue	**LRIQBA** LBI Receive Interrupt Queue Base Address **LRIQL** LBI Receive Interrupt Queue Length	58_H $5C_H$
LTIQ LBI Transmit Interrupt Queue	**LTIQBA** LBI Transmit Interrupt Queue Base Address **LTIQL** LBI Transmit Interrupt Queue Length	50_H 54_H

12.11.3 Peripheral Interrupts

This group consists of interrupt vectors generated by the following peripheral blocks:

- LBI Mailbox (MB)
- Synchronous Serial Control Interface (SSC)
- IOM(R)-2 Interface
- General Purpose Bus
- LBI Pass Through

Table 33
Peripheral Queue Interrupt Vectors

Interrupt Vector Name	Short Description	Reference Page
MB_IV	LBI Mailbox Interrupt Vector	135
SSC_IV (receive)	SSC Interrupt Vector (receive)	169
SSC_IV (transmit)	SSC Interrupt Vector (transmit)	169
IOMM_IV	Monitor Interrupt Vector	179
IOMCI_IV	C/I Interrupt Vector	179
GP_IV	General Purpose Bus Interrupt Vector	181
LPT_IV	LBI Pass Through Interrupt Vector	150

All peripheral interrupt vectors are written to one queue.

Peripheral Queue

Queue Name	Control Registers, Register Name	Offset Address
PIQ Peripheral Interrupt Queue	**PIQBA** Peripheral Interrupt Queue Base Address **PIQL** Peripheral Interrupt Queue Length	14_H 18_H

The different vector sources can be distinguished by decoding the most significant 8 bits as shown in **Figure 79**.

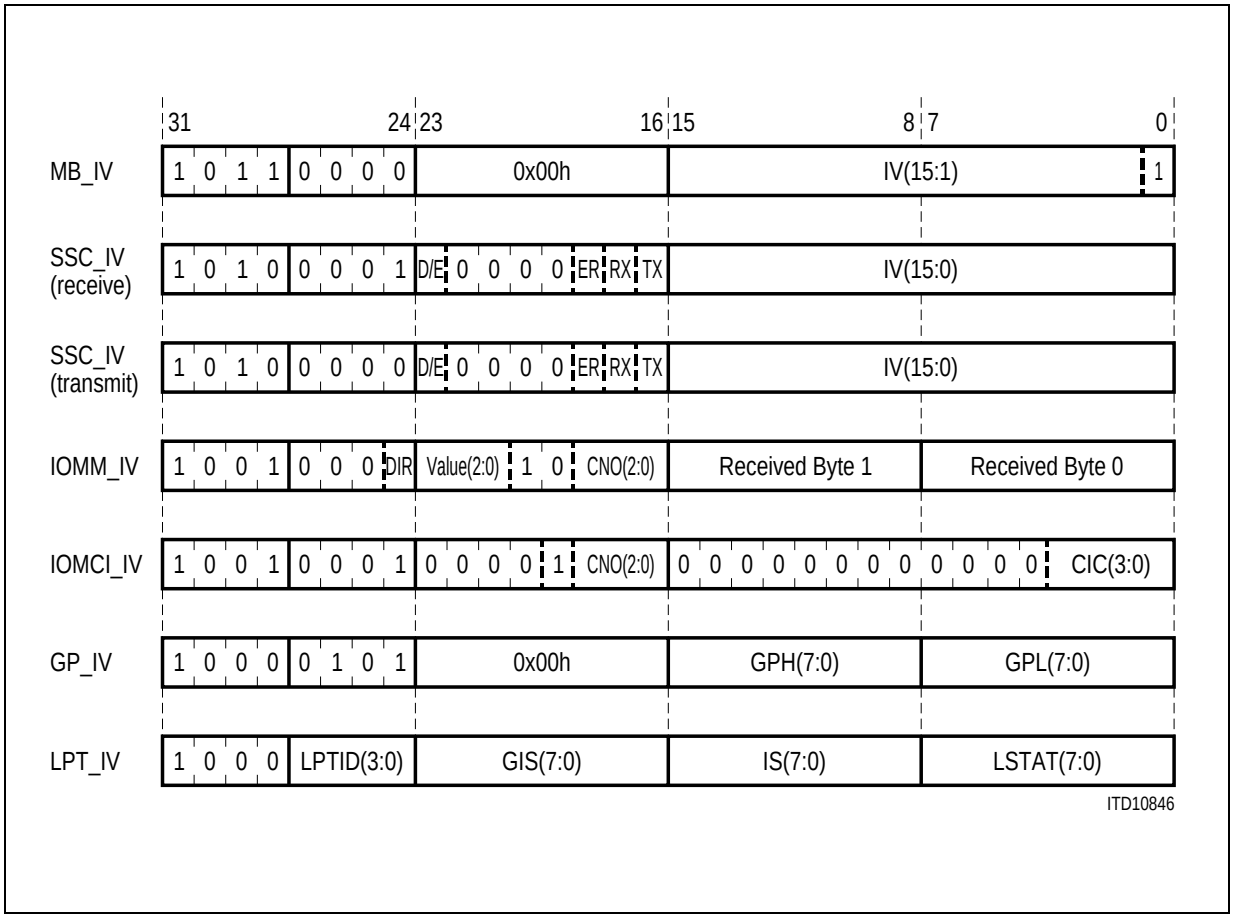


Figure 79
Decoding of Peripheral Queue Interrupt Vectors

13 Boundary Scan Unit

In MUNICH32X a Test Access Port (TAP) controller is implemented. 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 80** gives an overview.

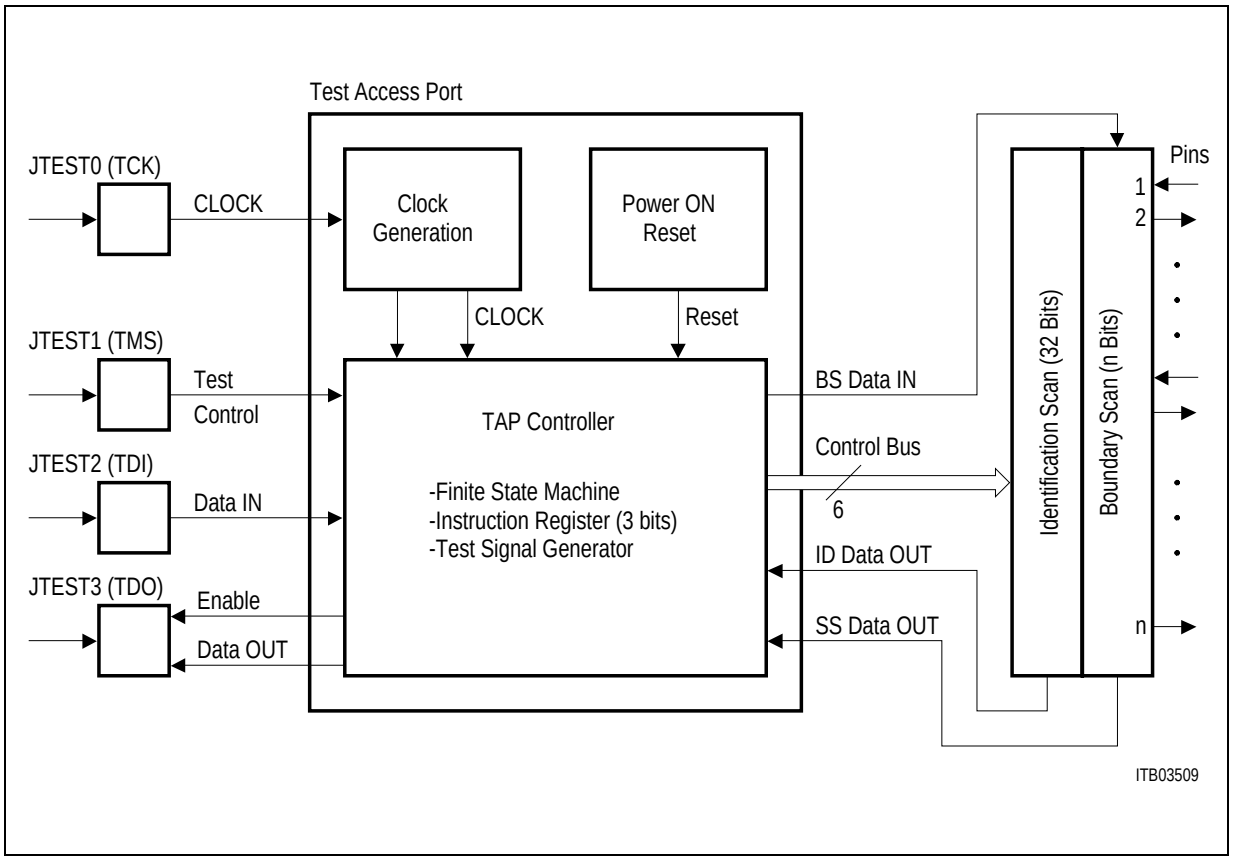


Figure 80
Block Diagram of Test Access Port and Boundary Scan

Test handling is performed via the pins TCK (Test Clock), TMS (Test Mode Select), TDI (Test Data Input) and TDO (Test Data Output). Test data at TDI are loaded with a 4-MHz clock signal connected to TCK. ‘1’ or ‘0’ on TMS causes a transition from one controller state to an other; constant ‘1’ on TMS leads to normal operation of the chip.

If no boundary scan testing is planned 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’.

After switching on the device (V_{DD} = 0 to 5 V) a power-on reset is generated which forces the TAP controller into test logic reset state.

Table 34
Boundary Scan Sequence in MUNICH32X

TDI →

Pin No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
1	LD15	I/O	3	001
2	LD14	I/O	3	100
3	LD13	I/O	3	000
4	LD12	I/O	3	000
5	LD11	I/O	3	001
6	LD10	I/O	3	111
7	LD9	I/O	3	000
8	LD8	I/O	3	000
9	LD7	I/O	3	100
10	LD6	I/O	3	000
11	LD5	I/O	3	110
12	LD4	I/O	3	000
13	LD3	I/O	3	000
14	LD2	I/O	3	000
15	LD1	I/O	3	000
16	LD0	I/O	3	000
17	$\overline{\text{RST}}$	I	1	0
18	CLK	I	1	0
19	$\overline{\text{GNT}}$	I/O	3	000
20	$\overline{\text{REQ}}$	I/O	3	000
21	AD31	I/O	3	000
22	AD30	I/O	3	000
23	AD29	I/O	3	000
24	AD28	I/O	3	000
25	AD27	I/O	3	000
26	AD26	I/O	3	000
27	AD25	I/O	3	000
28	AD24	I/O	3	000
29	C/$\overline{\text{BE}}$3	I/O	3	000
30	IDSEL	I/O	3	000
31	AD23	I/O	3	000
32	AD22	I/O	3	000
33	AD21	I/O	3	000
34	AD20	I/O	3	000
35	AD19	I/O	3	000
36	AD18	I/O	3	000

Table Table 34
Boundary Scan Sequence in MUNICH32X (cont'd)

TDI →

Pin No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
37	AD17	I/O	3	000
38	AD16	I/O	3	000
39	C/$\overline{BE}$2	I/O	3	000
40	$\overline{FRAME}$	I/O	3	000
41	$\overline{IRDY}$	I/O	3	000
42	$\overline{TRDY}$	I/O	3	000
43	$\overline{DEVSEL}$	I/O	3	000
44	$\overline{STOP}$	I/O	3	000
45	N.C.1	I/O	3	000
46	$\overline{PERR}$	I/O	3	000
47	$\overline{SERR}$	I/O	3	000
48	PAR	I/O	3	000
49	C/$\overline{BE}$1	I/O	3	000
50	AD15	I/O	3	000
51	AD14	I/O	3	000
52	AD13	I/O	3	000
53	AD12	I/O	3	000
54	AD11	I/O	3	000
55	AD10	I/O	3	000
56	AD9	I/O	3	000
57	AD8	I/O	3	000
58	C/$\overline{BE}$0	I/O	3	000
59	AD7	I/O	3	000
60	AD6	I/O	3	000
61	AD5	I/O	3	000
62	AD4	I/O	3	000
63	AD3	I/O	3	000
64	AD2	I/O	3	000
65	AD1	I/O	3	000
66	AD0	I/O	3	000
67	$\overline{INTA}$	I/O	3	000
68	LA15	I/O	3	000
69	LA14	I/O	3	000
70	LA13	I/O	3	000
71	LA12	I/O	3	000

Table Table 34
Boundary Scan Sequence in MUNICH32X (cont'd)

TDI →

Pin No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
72	LA11	I/O	3	000
73	LA10	I/O	3	000
74	LA9	I/O	3	000
75	LA8	I/O	3	000
76	LA7	I/O	3	000
77	LA6	I/O	3	000
78	LA5	I/O	3	000
79	LA4	I/O	3	000
80	LA3	I/O	3	000
81	LA2	I/O	3	000
82	LA1	I/O	3	000
83	LA0	I/O	3	000
84	N.C.2	I/O	3	000
85	W/$\overline{R}$	I/O	3	000
86	$\overline{LBREQ}$	I/O	3	000
87	$\overline{LHOLD}$	I	1	0
88	$\overline{LHLDA}$	I/O	3	000
89	$\overline{LRDY}$	I/O	3	000
90	$\overline{LRD}$	I/O	3	000
91	$\overline{LWR}$	I/O	3	000
92	$\overline{LBHE}$	I/O	3	000
93	LINTI2	I	1	0
94	LINTI1	I	1	0
95	LINTO	I/O	3	000
96	LALE	I/O	3	000
97	$\overline{LCSO}$	I/O	3	000
98	$\overline{LCSI}$	I	1	0
99	$\overline{DACKRB}$	I/O	3	000
100	$\overline{DACKRA}$	I/O	3	000
101	$\overline{DACKTB}$	I/O	3	000
102	$\overline{DACKTA}$	I/O	3	000
103	DRQRB	I/O	3	000
104	DRQTB	I/O	3	000
105	DRQRA	I/O	3	000
106	DRQTA	I/O	3	000

Table Table 34
Boundary Scan Sequence in MUNICH32X (cont'd)

TDI →

Pin No.	Pin	I/O	Number of Boundary Scan Cells	Constant Value In, Out, Enable
107	DEMUX	I	1	0
108	TEST	I	1	0
109	$\overline{\text{MCS3}}$	I/O	3	000
110	$\overline{\text{MCS2}}$	I/O	3	000
111	$\overline{\text{MCS1}}$	I/O	3	000
112	$\overline{\text{MCS0}}$	I/O	3	000
113	N.C.3	I/O	3	000
114	MRST	I/O	3	000
115	MTSR	I/O	3	000
116	MCLK	I/O	3	000
117	TXCLK	I	1	0
118	TSP	I	1	0
119	TXD	I/O	3	000
120	$\overline{\text{TXDEN}}$	I/O	3	000
121	$\overline{\text{DRDY}}$	I	1	0
122	RXD	I	1	0
123	RSP	I	1	0
124	RXCLK	I	1	0

→ TDO

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 some output and input pins of the MUNICH32X are tested as I/O pins in boundary scan, hence using three cells. The boundary scan unit of the MUNICH32X contains a total of n = 344 scan cells.

The right column of **Table 34** gives the initialization values of the cells.

The desired test mode is selected by serially loading a 3-bit instruction code into the instruction register via TDI (LSB first); see **Table 35.**

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', according to **Table 34**). 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

Table 35
Boundary Scan Test Modes

Instruction (Bit 2 … 0)	Test Mode
000	EXTEST (external testing)
001	INTEST (internal testing)
010	SAMPLE/PRELOAD (snap-shot testing)
011	IDCODE (reading ID code)
111	BYPASS (bypass operation)
others	handled like BYPASS

updated according to the new 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’, according to **Table 34**). 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.

Note: In capture IR-state the code ‘001’ is automatically loaded into the instruction register, i.e. if INTEST is wanted the shift IR-state does not need to be passed.

SAMPLE/PRELOAD is a test mode which provides a snap-shot 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’.

TDI →	0011	0000 0000 0011 1100	0000 1000 001	1	→ TDO

Note: Since in test logic reset state the code ‘011’ is automatically loaded into the instruction register, the ID code can easily be read out in shift DR state which is reached by TMS = 0, 1, 0, 0.

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

14 Electrical Characteristics

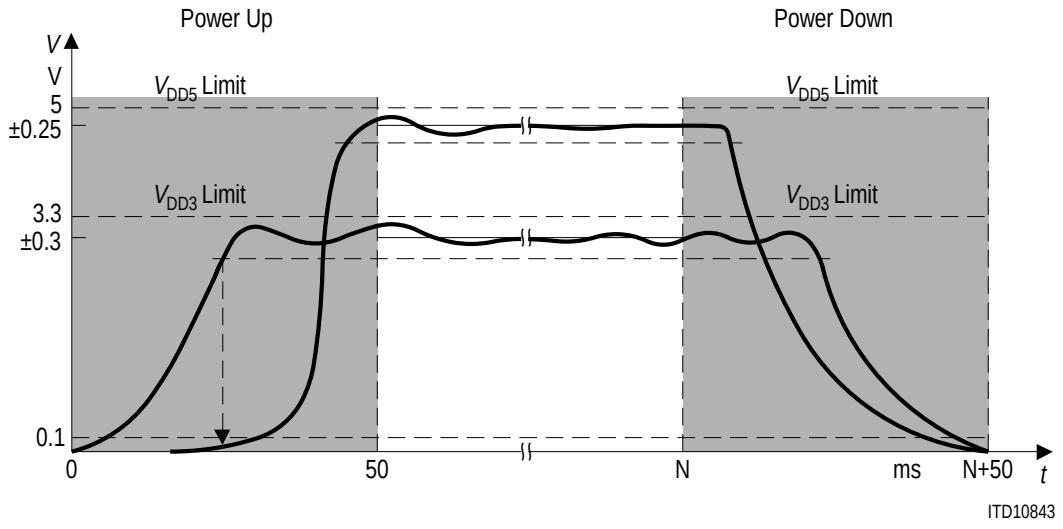
14.1 Important Electrical Requirements

V_{DD3} = 3.3 V ± 0.3 V **$V_{DD3\ max}$ = 3.6 V**

V_{DD5} = 5.0 V ± 0.25 V **$V_{DD5\ max}$ = 5.25 V**

During all MUNICH32X power-up and power-down situations the difference $|V_{DD5} - V_{DD3}|$ may not exceed 3.6 V. The absolute maximums of V_{DD5} and V_{DD3} should never be exceeded.

Figure 81 shows that both V_{DD3} and V_{DD5} can take on any time sequence, not exceeding the maximum slew rate for V_{DD3} mentioned later and not exceeding a voltage difference of 3.6 V, for up to 50 ms at power-up and power-down. Within 50 ms of power-up the voltages must be within their respective absolute voltage limits. At power-down, within 50 ms of either voltage going outside its operational range, the voltage difference should not exceed 3.6 V and both voltages must be returned below 0.1 V.



Within the grey boxes any shape of V_{DD3} and V_{DD5} signal is allowed with the requirements that the absolute limits of each signal are not exceeded, the slew rate recommendation for V_{DD3} is met to guarantee proper boundary scan reset and the voltage difference does not exceed 3.6 V.

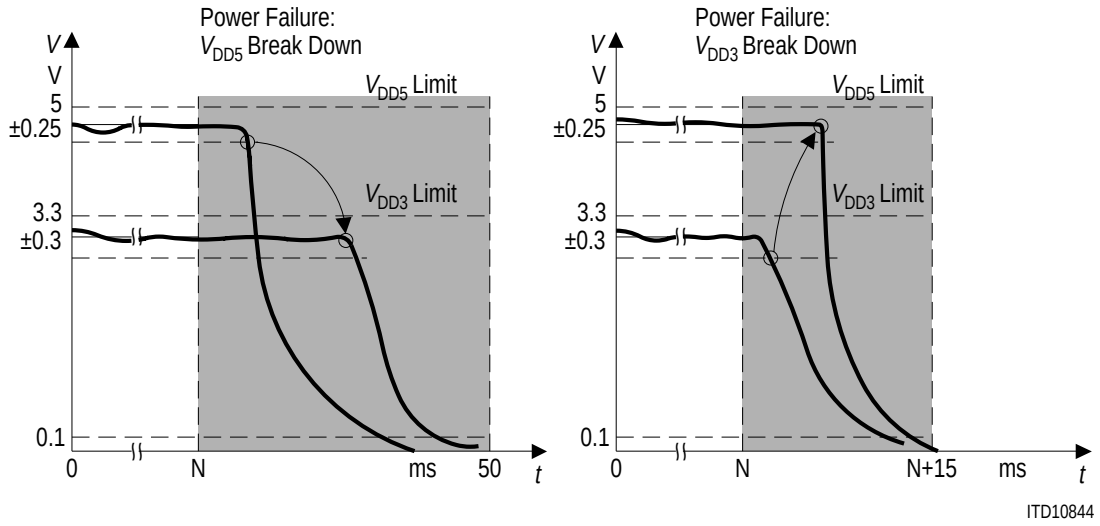
Outside the grey boxes the voltages provided to V_{DD3} and V_{DD5} should be inside the normal operation range.

In this power-up example V_{DD5} is enabled after V_{DD3} reached its minimum operation value which is a typical implementation.

For power-down V_{DD5} is switched off before V_{DD3}.

Figure 81
Power-up and Power-down Scenarios

Similar criteria also apply to power down in case of power failure situations:



Within the grey boxes any shape of V_{DD3} and V_{DD5} signal is allowed with the requirements that the absolute limits of each signal are not exceeded and the specified voltage differences are not exceeded.

a. In case of V_{DD5} break-down the 3.6V difference is not exceeded anyway. The voltages must return below 0.1V within 50 milliseconds.
b. In case of V_{DD3} break-down the maximum voltage difference must not exceed 4.5 V for a maximum of 15 milliseconds.The voltages must return below 0.1V within 50 milliseconds. This scenario is allowed for 2000 power failure cycles.

Figure 82
Power-Failure Scenarios

Note: Siemens recommends that the V_{DD3} power supply rise from 0 to 3.3 V over a duration between 5 and 50 ms (slew rate range 66 to 660 V/s).

The MUNICH32X contains an internal power-on reset generator, to reset the boundary scan state machine. A successful boundary scan reset is required for normal device operation.

For boundary scan operations **as well as for normal device operation** the following operational conditions should be observed:

A power-on reset to the boundary scan state machine is guaranteed to occur under these two conditions:

- V_{DD3} rising slope < 0.4 V/ms, or
- V_{DD3} voltage breakdown below 0.1 V or
  switch off > 20 ns combined with a V_{DD3} voltage breakdown below 0.1 V.

A power-on reset to the boundary scan state machine will not occur under these two conditions:

- V_{DD3} falls below 0.1 V for less than 5 ns, or
- V_{DD3} falls to no less than 2.6 V for any length of time.

The operation of the device and the boundary scan circuitry is unpredictable for any of the following conditions:

- V_{DD3} slope > 0.5 V/ms,or
- V_{DD3} falls below 0.1 V for a duration between 5 ns and 20 ns, or
- V_{DD3} falls to a voltage between 0 V and 2.6 V for any duration

Note: Siemens recommends that the V_{DD3} power supply rise from 0 to 3.3 V over a duration between 5 and 50 ms (slew rate range 66 to 660 V/second).

Additional recommendations:

If the pin DRDY is not used, it should be connected to V_{DD3} via a pull-up resistor. The pin TEST has to be tied to V_{SS} (refer to pin description table).

14.2 Absolute Maximum Ratings

Table 36

Parameter	Symbol	Limit Values		Unit
		min.	max.	
Ambient temperature under bias	T_A	0	70	°C
Junction temperature under bias	T_J	–	125	°C
Storage temperature	T_{stg}	– 65	125	°C
Voltage at any pin with respect to ground	V_S	– 0.4	V_{DD5} + 0.4	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.

14.3 Thermal Package Characteristics

Table 37

Parameter	Symbol	Value	Unit
Thermal Package Resistance Junction to Ambient (without air flow)	θ_{JA}	40.3	°C/W

14.4 DC Characteristics

a) Non-PCI Interface Pins

Table 38

T_A = 0 to + 70 °C; V_{DD5} = 5 V ± 5%, V_{DD3} = 3.3 V ± 0.3 V, V_{SS} = 0 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_{DD5} + 0.4	V	–
L-output voltage		V_{QL}	–	0.45	V	I_{QL} = 7 mA (pin TXD) I_{QL} = 2 mA (all others / non-PCI)
H-output voltage		V_{QH}	2.4	V_{DD3}	V	I_{QH} = – 400 µA
Power supply current V_{DD3}	operational	I_{CC3}	–	< 200	mA	V_{DD3} = 3.3 V, V_{DD5} = 5 V, inputs at V_{SS}/V_{DD}, no output loads
	power down (no clocks)	I_{CC3}	–	< 2	mA	
Power supply current V_{DD5}		I_{CC5}	–	< 2	mA	–
Power dissipation		P	–	< 800	mW	–
Input leakage current Output leakage current		I_{LI} I_{LQ}	–	10	µA	0 V < V_{IN} < V_{DD} to 0 V 0 V < V_{OUT} < V_{DD} to 0 V

Note: 1. The listed characteristics are ensured over the operating range of the integrated circuit. Typical characteristics specify mean values expected over the production spread. If not otherwise specified, typical characteristics apply at T_A = 25 °C and the given supply voltage.

*Note: 2. The electrical characteristics described in **Section 14.2** also apply here!*

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995.
(Chapter 4: Electrical Specification for 5 V signalling)

14.5 Capacitances

a) Non-PCI Interface Pins

Table 39

T_A = 25 °C; V_{DD5} = 5 V ± 5%, V_{DD3} = 3.3 V ± 0.3 V, V_{SS} = 0 V

Parameter	Symbol	Limit Values		Unit	Test Condition
		min.	max.		
Input capacitance	C_{IN}	1	5	pF	–
Output capacitance	C_{OUT}	5	10	pF	–
I/O-capacitance	C_{IO}	6	15	pF	–

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995 (Chapter 4: Electrical Specification for 5 V signalling)

14.6 AC Characteristics

a) Non-PCI Interface Pins

T_A = 0 to + 70 °C; V_{DD5} = 5 V ± 5%; V_{DD3} = 3.3 V ± 0.3 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.

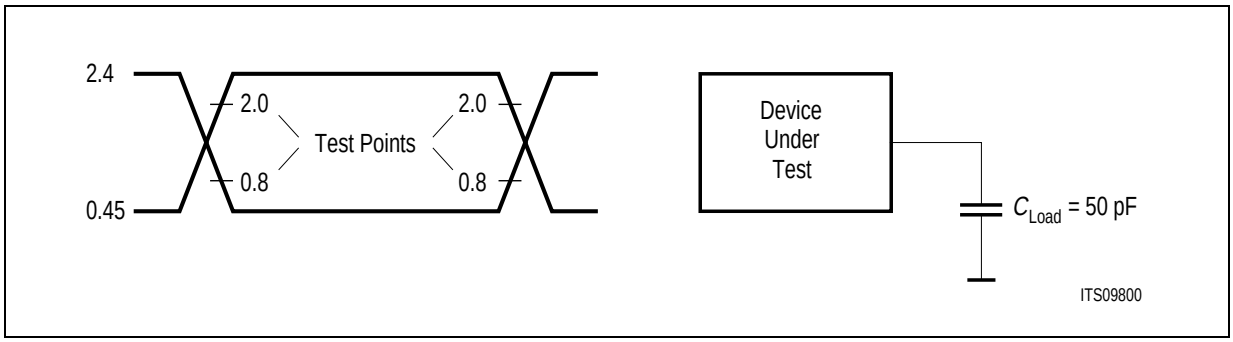


Figure 83
Input/Output Waveform for AC Tests

b) PCI Pins

According to the PCI specification V2.1 from June 1, 1995 (Chapter 4: Electrical Specification for 5 V signalling)

14.6.1 PCI Bus Interface Timing

The AC testing input/output waveforms are shown in **Figure 84** and **Figure 85**.

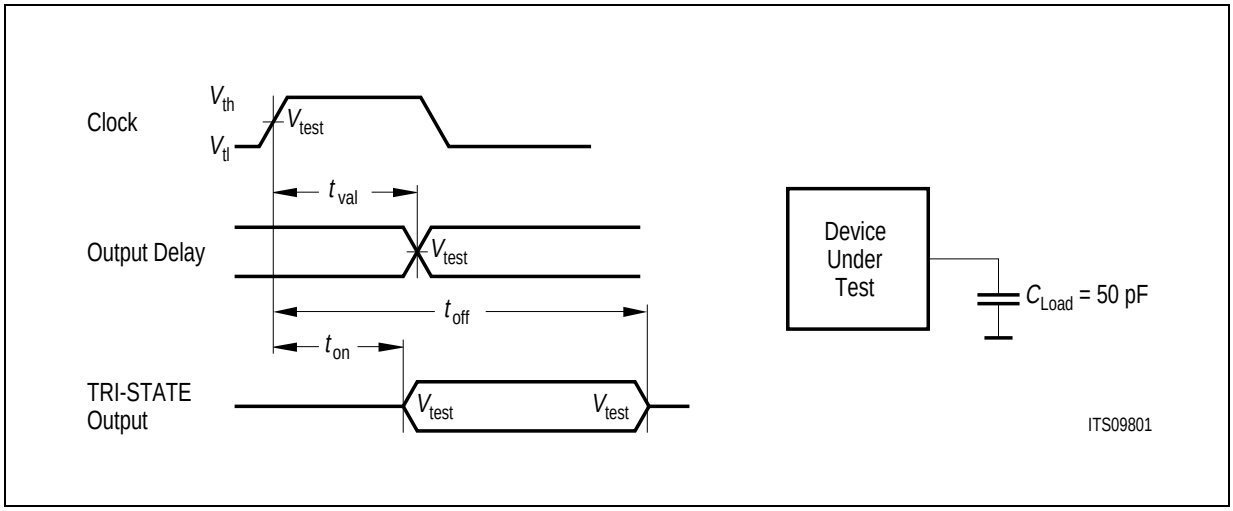


Figure 84
PCI Output Timing Measurement Waveforms

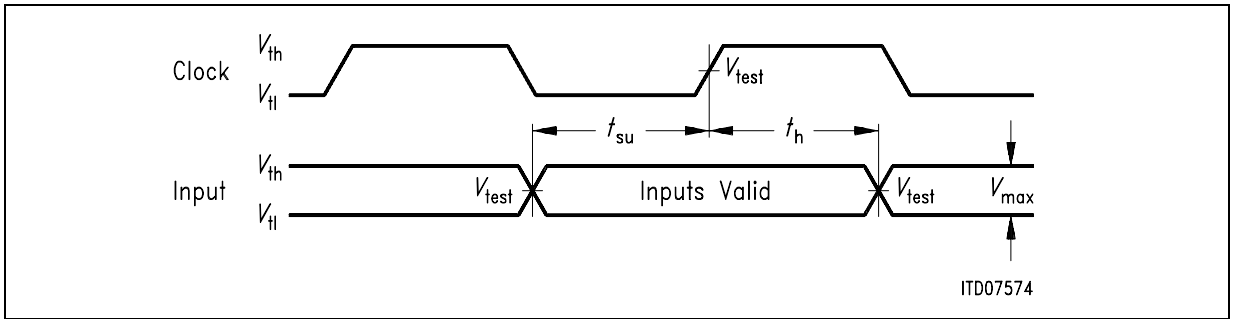


Figure 85
PCI Input Timing Measurement Waveforms

Table 40
PCI Input and Output Measurement Conditions

Symbol	Value	Unit
V_{th}	2.4	V
V_{tl}	0.4	V
V_{test}	1.5	V
V_{max}	2.0	V

The timings below show the basic read and write transaction between an initiator (Master) and a target (Slave) device. The MUNICH32X is able to work both as master and slave.

As a master the MUNICH32 reads/writes data from/to host memory using DMA and burst. The slave mode is used by an CPU to access the MUNICH32 PCI Configuration Space, the on-chip registers and to access peripherals connected to the MUNICH32 Local Bus Interface (LBI).

14.6.1.1 PCI Read Transaction

The transaction starts with an address phase which occurs during the first cycle when $\overline{\text{FRAME}}$ is activated (clock 2 in **Figure 86**). 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 turn-around cycle. In **Figure 86** 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 $\overline{IRDY}$ and $\overline{TRDY}$ 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 $\overline{TRDY}$ is deasserted. The last data phase is extended because $\overline{IRDY}$ 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 transfer, so $\overline{IRDY}$ is deasserted on clock 7, and $\overline{FRAME}$ stays asserted. Only when $\overline{IRDY}$ is asserted can $\overline{FRAME}$ be deasserted, which occurs on clock 8.

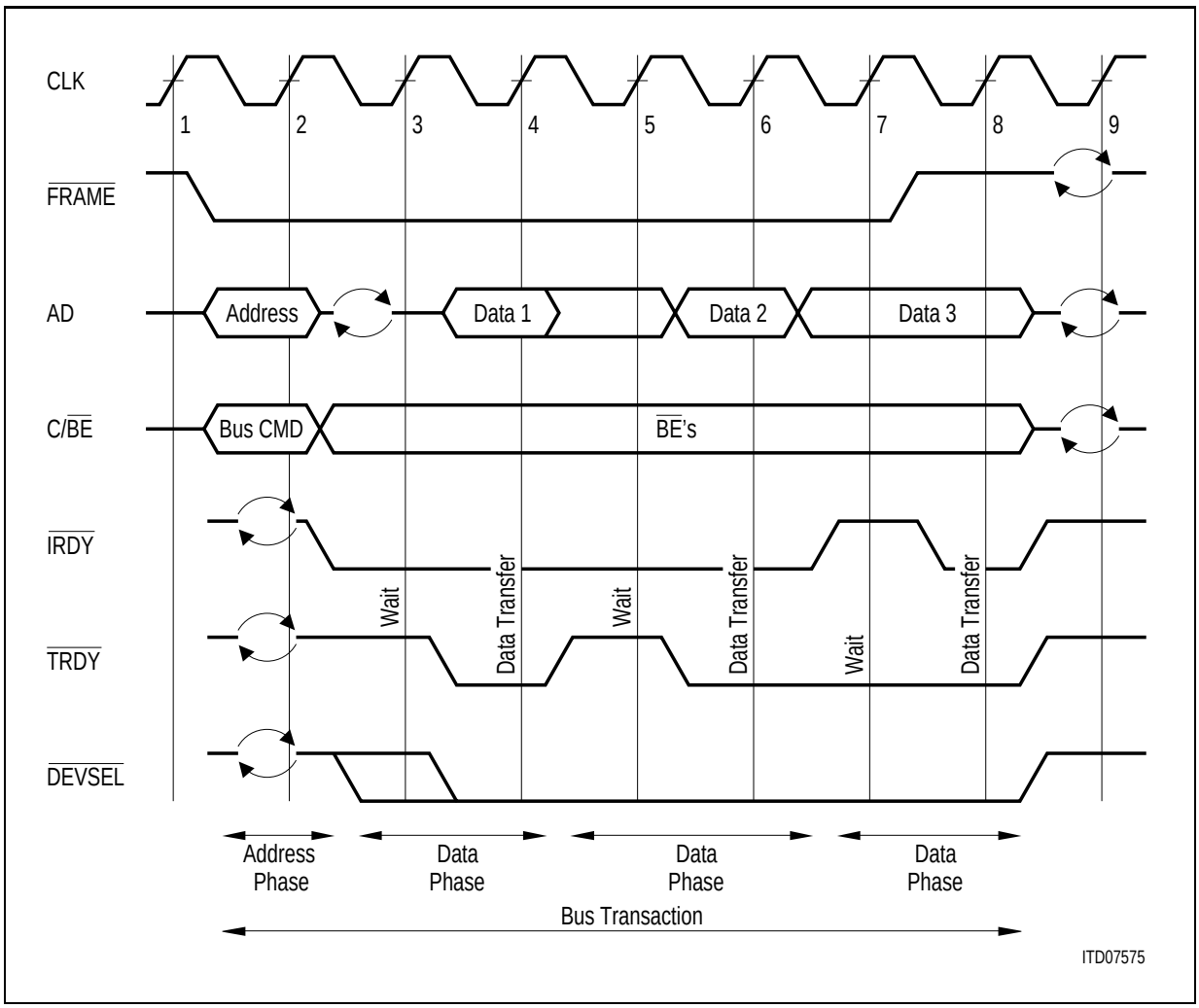


Figure 86
PCI Read Transaction

14.6.1.2 PCI Write Transaction

The transaction starts when $\overline{\text{FRAME}}$ is activated (clock 2 in **Figure 87**). 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).

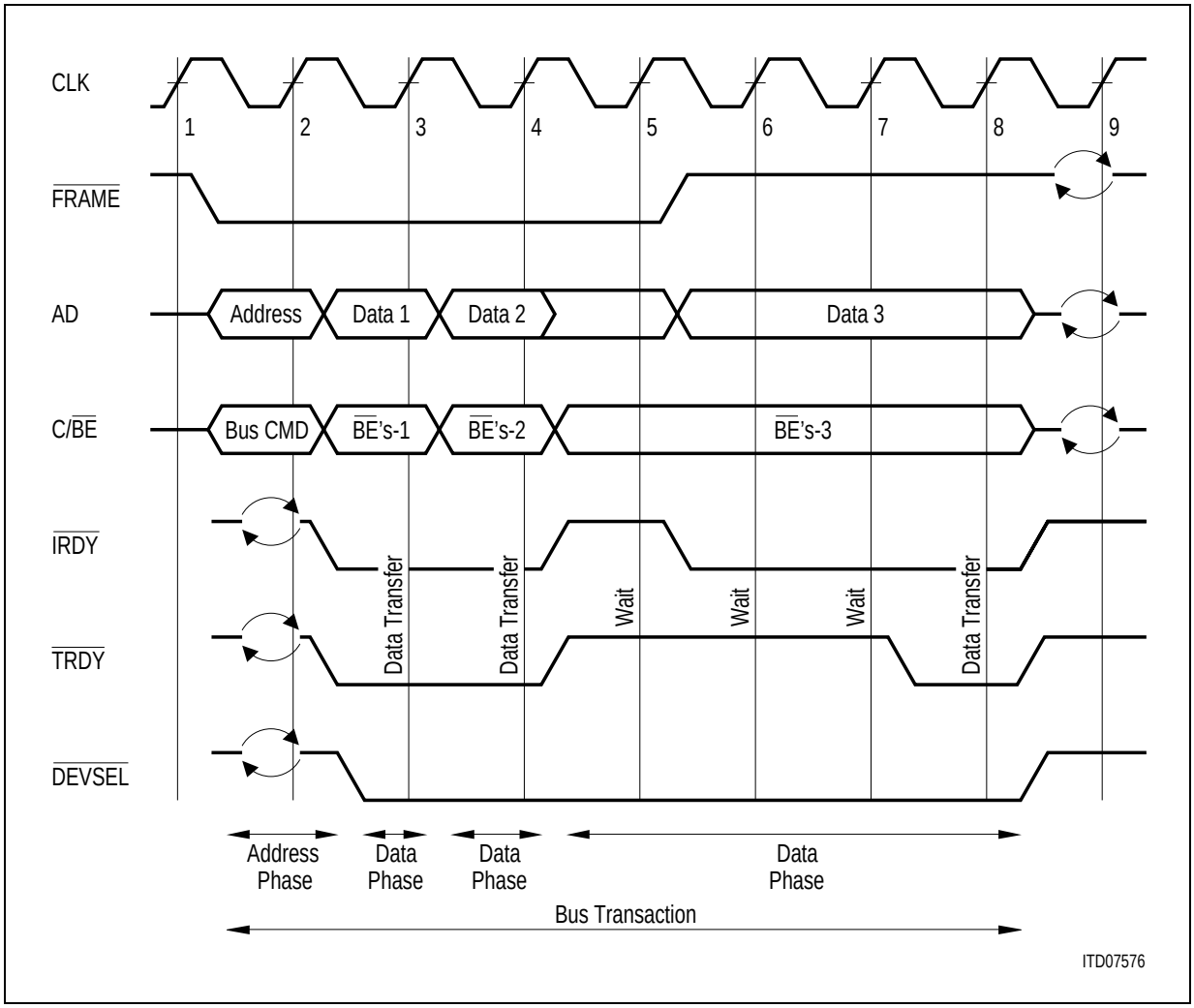


Figure 87
PCI Write Transaction

14.6.1.3 PCI Timing Characteristics

When the MUNICH32X operates as a PCI Master (initiator) and it either reads or writes a burst – as controlled by the on-chip DMA controller – it does not deactivate $\overline{\text{IRDY}}$ between consecutive data. In other words, no wait states are inserted by the MUNICH32X as a transaction initiator. The numbers of wait states, inserted by the MUNICH32X as initiator are listed in **Table 41**.

Table 41 Number of Wait States Inserted by the MUNICH32X as Initiator

Transaction	Number of Wait States	
	1st Data Cycle	2nd and Subsequent Data Cycles
Memory read burst	0	0
Memory write burst	0	0
Fast Back-to-back burst; 1st transaction	0	0
Fast Back-to-back burst; 2nd and subsequent transactions	1	0

When the MUNICH32X operates as a PCI Slave (target), it inserts wait cycles by deactivating $\overline{\text{TRDY}}$. The numbers of wait states, typically inserted by the MUNICH32X are listed in **Table 41**.

Table 42 Number of Wait States Inserted by the MUNICH32X as Target

Transaction	Number of Wait States
Configuration read	2
Configuration write	0
Register read	3
Register write	0
LBI read	3
LBI write	0

The number of wait states inserted by the MUNICH32X as target is not critical for two reasons. One, because accesses to/via the MUNICH32X are usually kept to a minimum in a system. And two, because they are dependent on the type of access (e.g. for an access to a peripheral device on the $\overline{\text{LRDY}}$ signal).

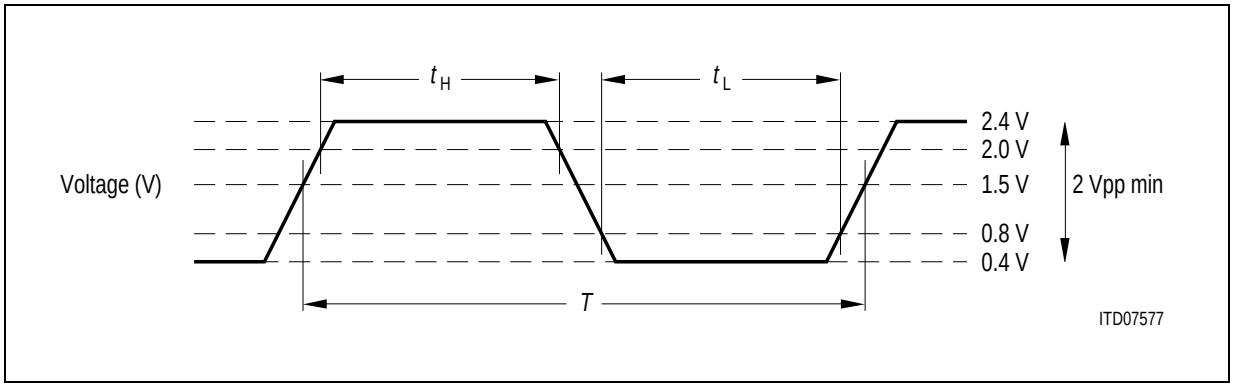


Figure 88
PCI Clock Specification

Table 43
PCI Clock Characteristics

Parameter	Symbol	Limit Values			Unit
		min.	typ.	max.	
CLK cycle time	T	30	–	–	ns
CLK high time	t_H	11	–	–	ns
CLK low time	t_L	11	–	–	ns
CLK slew rate (see note)	–	1	–	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 as shown in ***Figure 88****.*

Note: If f_T is the frequency of the clock TCLK, f_R the frequency of the clock RCLK and f_{CLK} the frequency of the clock CLK the equations
$7.996 \times \max(f_T, f_R) \leq f_{CLK} \leq 33.33$ MHz for CEPT, T1, E1 PCM mode
and
$3.998 \times \max(f_T, f_R) \leq f_{CLK} \leq 33.33$ MHz for 4.096 MHz PCM mode
and
$25\ \text{MHz} \leq f_{CLK} \leq 33.33$ MHz for 8.192 MHz PCM mode
describe the allowed range of frequencies for f_{CLK}.

Table 44
PCI Interface Signal Characteristics

Parameter	Limit Values			Unit	Remarks
	min.	typ.	max.		
CLK to signal valid delay bussed signals	–	–	11	ns	Notes 1, 2
CLK to signal valid delay point-to-point	–	–	12	ns	Notes 1, 2
Float to active delay	–	–	3	ns	–
Active to float delay	–	–	20	ns	–
Input setup time to CLK bussed signals	7	–	–	ns	Note 2
Input setup time to CLK point-to-point	8	–	–	ns	Note 2
Input hold time from CLK	0	–	–	ns	–

Note: Minimum times are measured with 0 pF equivalent load; maximum times are measured with 50 pF equivalent load.

Note: $\overline{REQ}$ and $\overline{GNT}$ are point-to-point signals. All other signals are bussed.

14.6.2 De-multiplexed Bus Interface

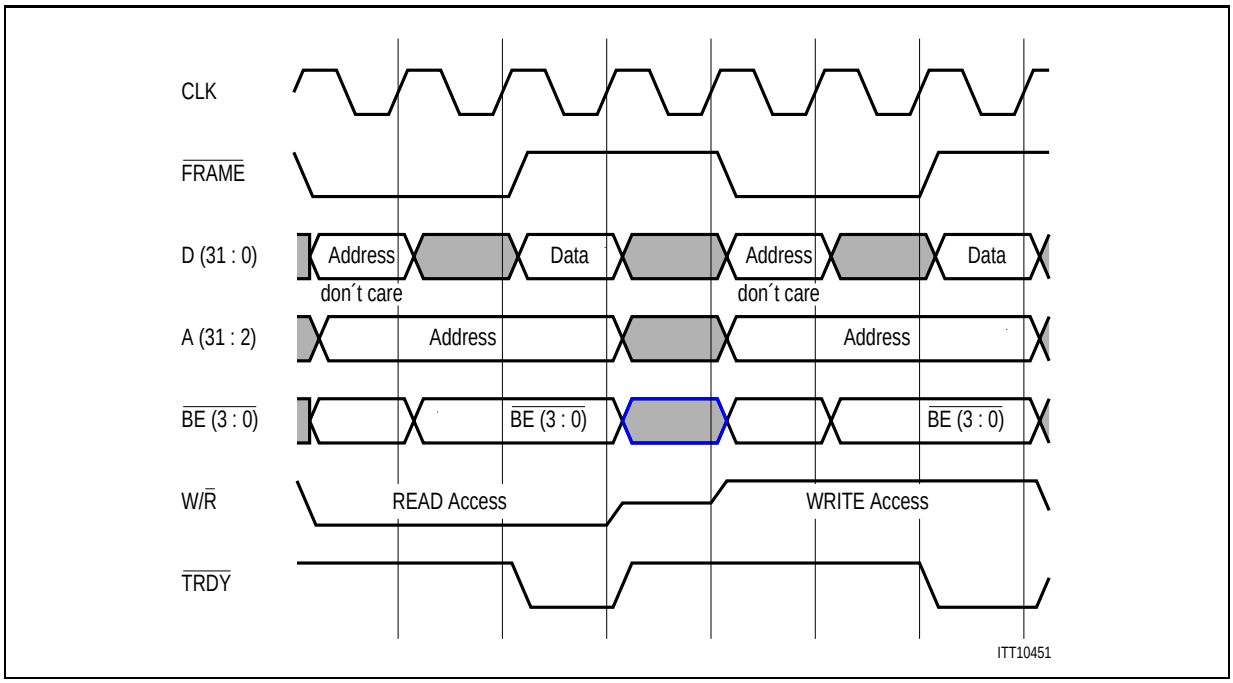


Figure 89
Master Single READ Transaction Followed by a Master Single WRITE Transaction in De-multiplexed Bus Configuration

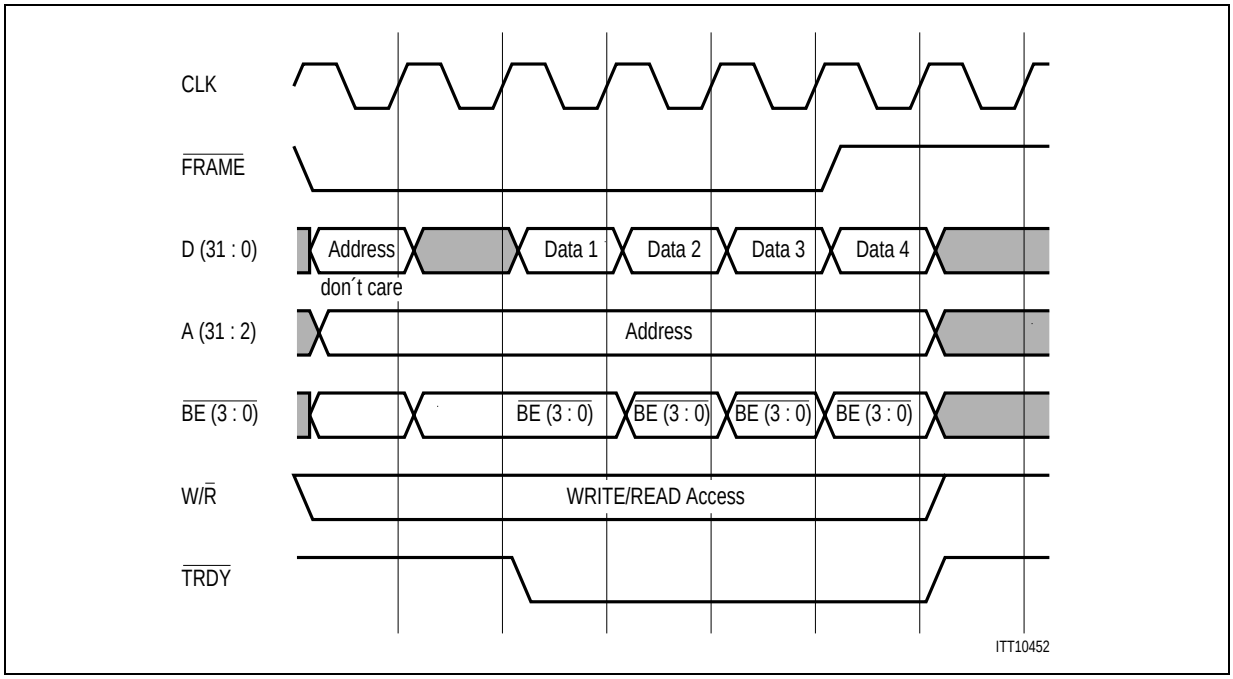


Figure 90
Master Burst WRITE/READ Access in De-multiplexed Bus Configuration

The timing provided in **Table 37** and **Table 38** can also be applied to the de-multiplexed bus interface.

Table 45
Additional De-multiplexed Interface Signal Characteristics

Parameter	Limit Values			Unit	Remarks
	min.	typ.	max.		
CLK to address bus signal valid delay	–	–	18	ns	–
CLK to W/$\overline{R}$ signal valid delay	–	–	15	ns	–
Address bus Input setup time to CLK	7	–	–	ns	–
Address bus Input hold time to CLK	0	–	–	ns	–
W/$\overline{R}$ signal Input setup time to CLK	7	–	–	ns	–
W/$\overline{R}$ signal Input hold time to CLK	0	–	–	ns	–

Note: The PCI parity signal PAR is not generated in de-multiplexed mode. It is driven active low by the MUNICH32X.

14.6.3 Local Bus Interface Timing

14.6.3.1 Local Bus Interface Timing in Slave Mode

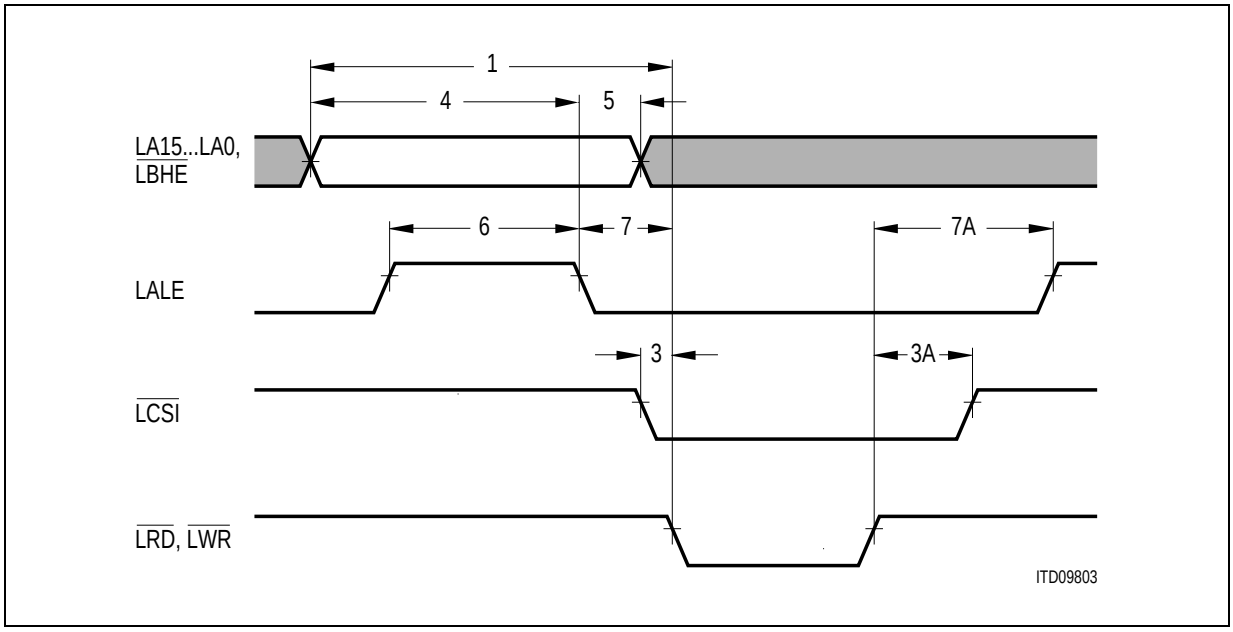


Figure 91
LBI Slave: Address Timing in Multiplexed Mode

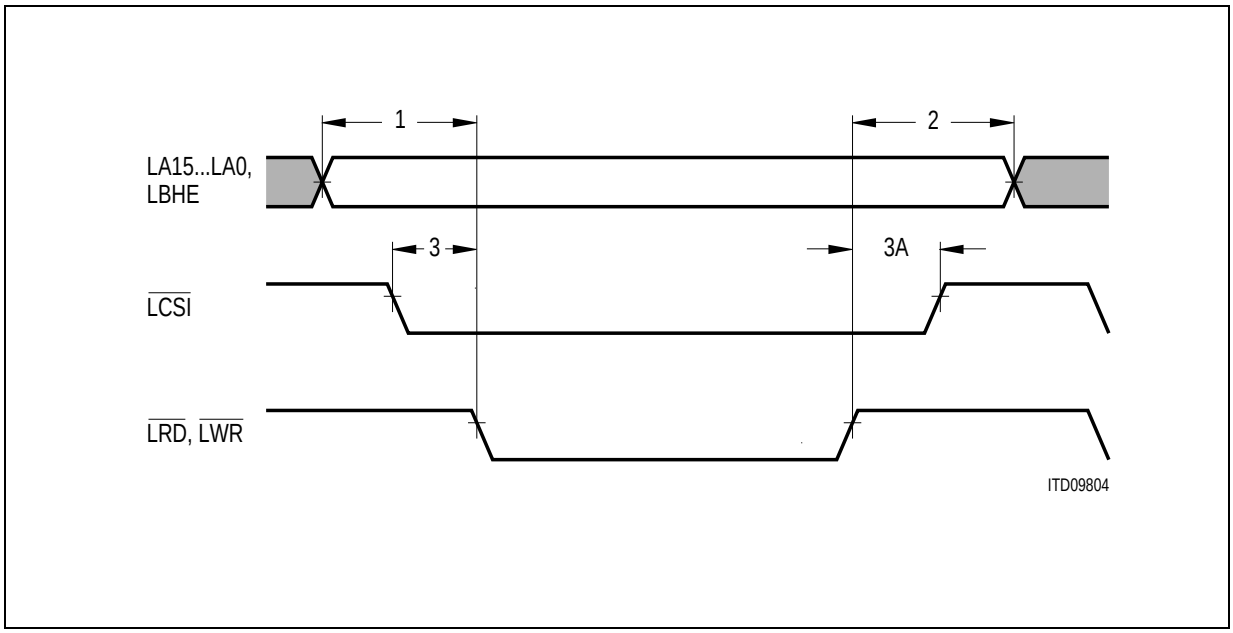


Figure 92
LBI Slave: Address Timing in Non-Multiplexed Mode

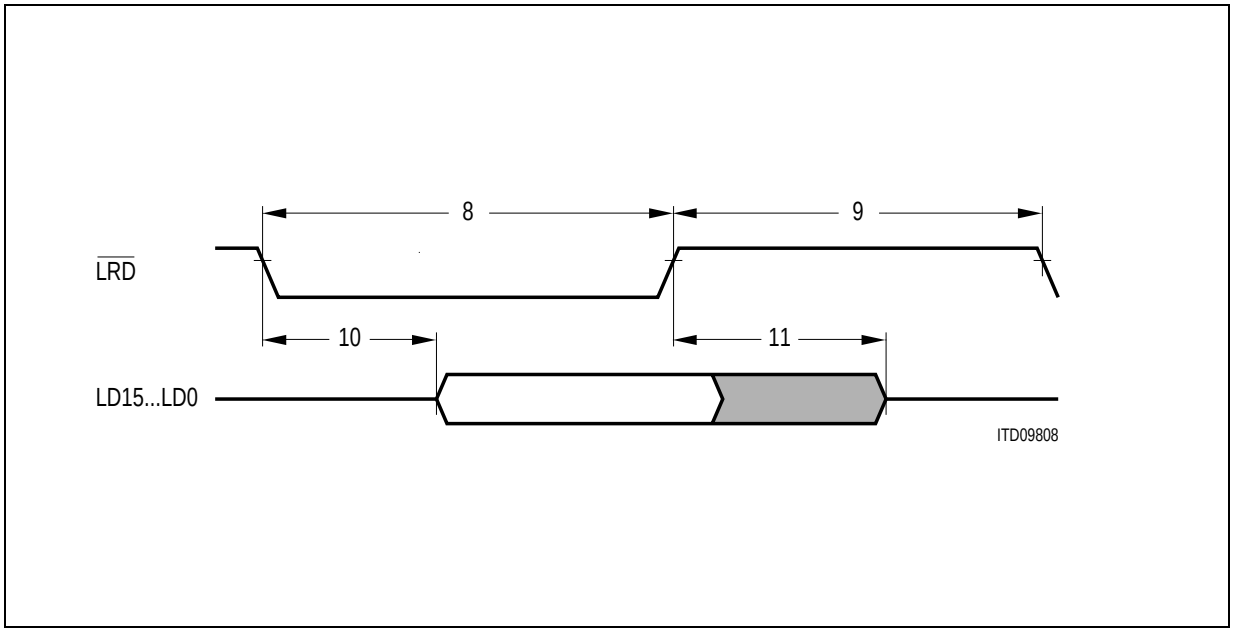


Figure 93
LBI Slave: Read Cycle Timing

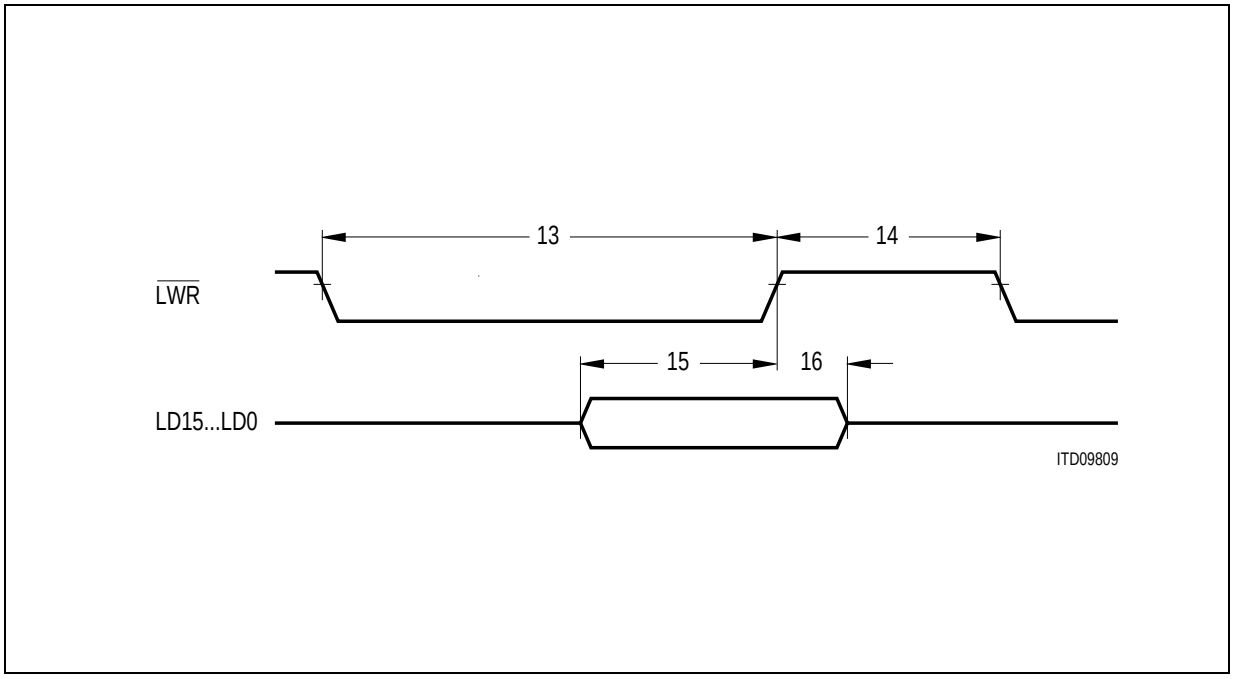


Figure 94
LBI Slave: Write Cycle Timing

Table 46

Parameter	No.	Symbol	Limit Values		Unit
			min.	max.	
LA, $\overline{\text{LBHE}}$, setup time (Slave)	1	$t_{\text{S-su(A)}}$	5	–	ns
LA, $\overline{\text{LBHE}}$, hold time (Slave)	2	$t_{\text{S-h(A)}}$	5	–	ns
$\overline{\text{LCSI}}$ setup time (Slave)	3	$t_{\text{S-su(A)}}$	0	–	ns
$\overline{\text{LCSI}}$ hold time (Slave)	3A	$t_{\text{S-h(A)}}$	0	–	ns
LA, $\overline{\text{LBHE}}$ stable before ALE inactive (Slave)	4	$t_{\text{S-su(A-ALE)}}$	1 T_{LBICLK}	–	(ns)
LA, $\overline{\text{LBHE}}$ hold after ALE inactive (Slave)	5	$t_{\text{S-h(A-ALE)}}$	5	–	ns
LALE pulse width (Slave)	6	$t_{\text{S-w(ALE)}}$	1 T_{LBICLK}	–	(ns)
Address latch setup time before command active (Slave)	7	$t_{\text{S-su(ALE)}}$	0	–	ns
LALE to command inactive delay (Slave)	7A	$t_{\text{S-rec(ALE)}}$	1 T_{LBICLK}	–	(ns)
$\overline{\text{LRD}}$ pulse width (Slave)	8	$t_{\text{S-w(RD)}}$	3 T_{LBICLK}	–	(ns)
$\overline{\text{LRD}}$ control interval (Slave)	9	$t_{\text{S-rec(RD)}}$	1 T_{LBICLK}	–	(ns)
LD valid after $\overline{\text{LRD}}$ active (Slave)	10	$t_{\text{S-a(RD)}}$	–	2 T_{LBICLK}	(ns)
LD hold after $\overline{\text{LRD}}$ inactive (Slave)	11	$t_{\text{S-h(RD)}}$	3 T_{LBICLK}	–	ns
$\overline{\text{LWR}}$ pulse width (Slave)	13	$t_{\text{S-w(WR)}}$	2 T_{LBICLK}	–	ns
$\overline{\text{LWR}}$ control interval (Slave)	14	$t_{\text{S-rec(WR)}}$	1 T_{LBICLK}	–	ns
LD stable before $\overline{\text{LWR}}$ inactive (Slave)	15	$t_{\text{S-su(WR)}}$	1 T_{LBICLK}	–	ns
LD hold after $\overline{\text{LWR}}$ inactive (Slave)	16	$t_{\text{S-h(WR)}}$	5	–	ns

Note: T_{LBICLK} is the LBI clock time period which depends on the LBI clock division factor.

14.6.3.2 Local Bus Interface Timing in Master Mode

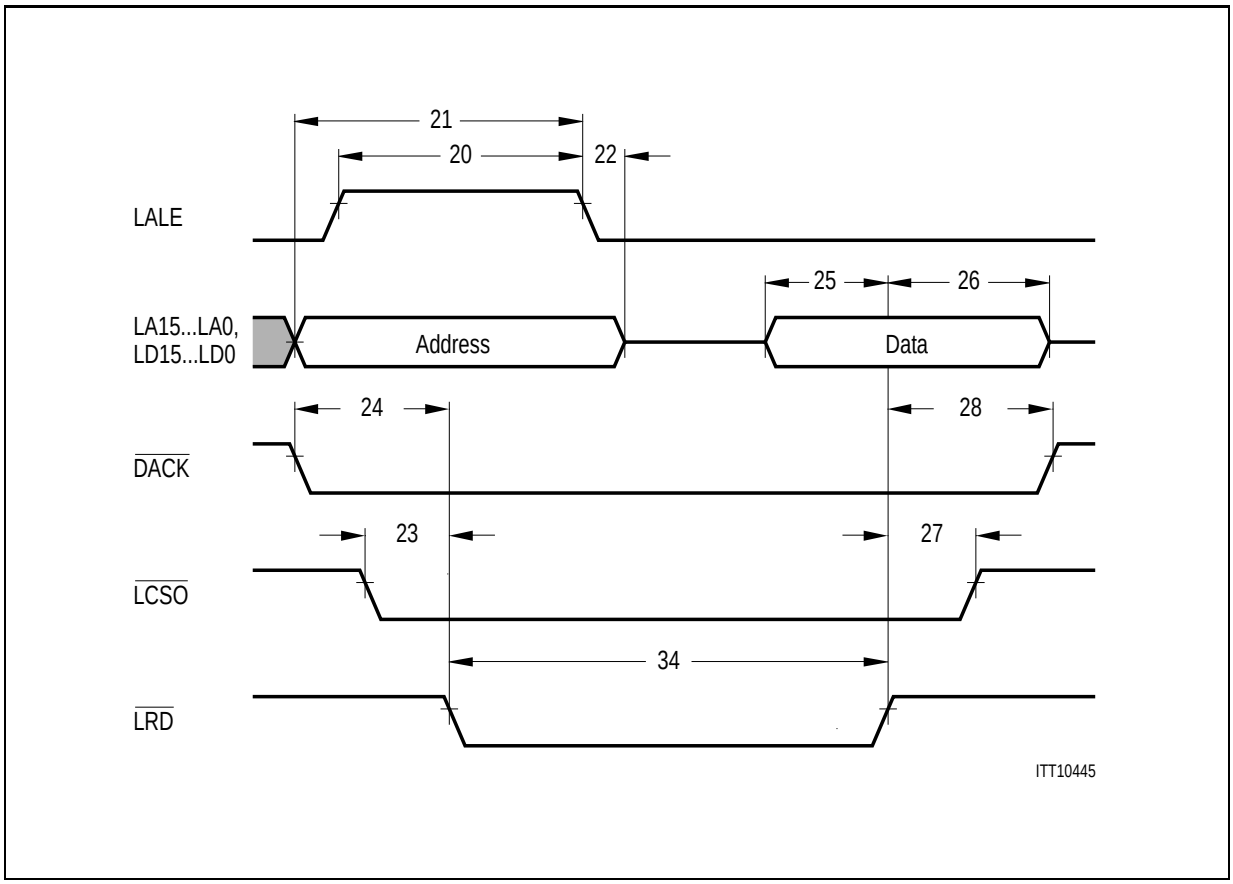


Figure 95
LBI Master: Read Cycle Timing in Multiplexed Mode

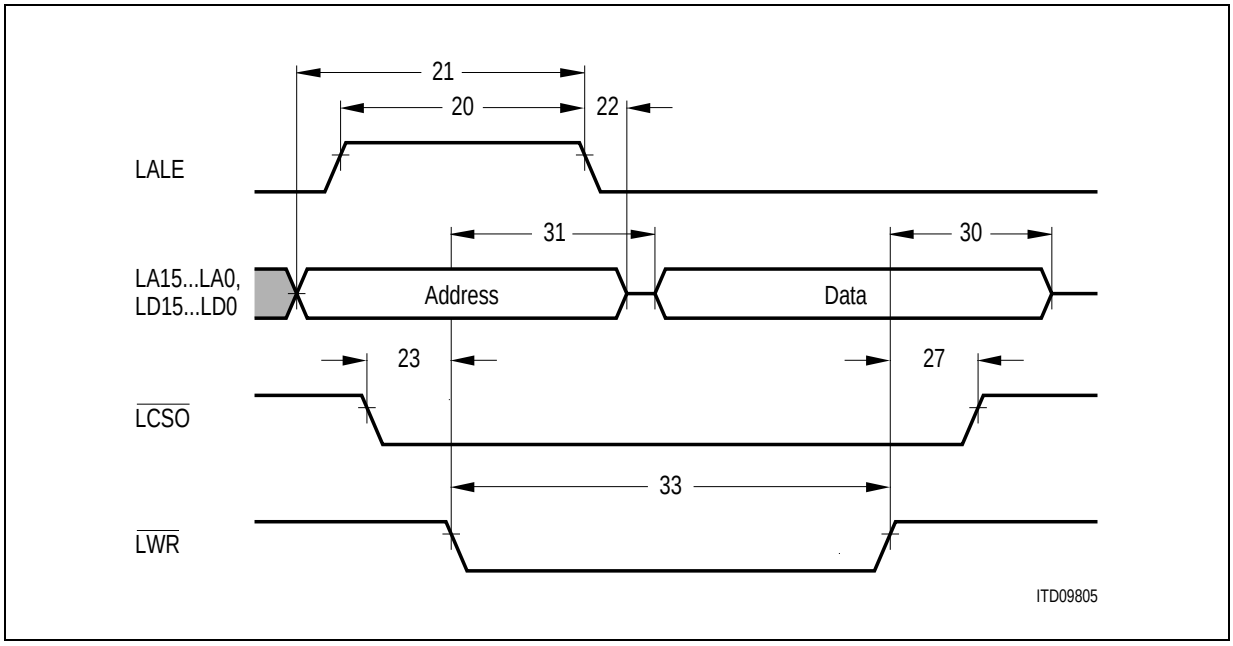


Figure 96
LBI Master: Write Cycle Timing

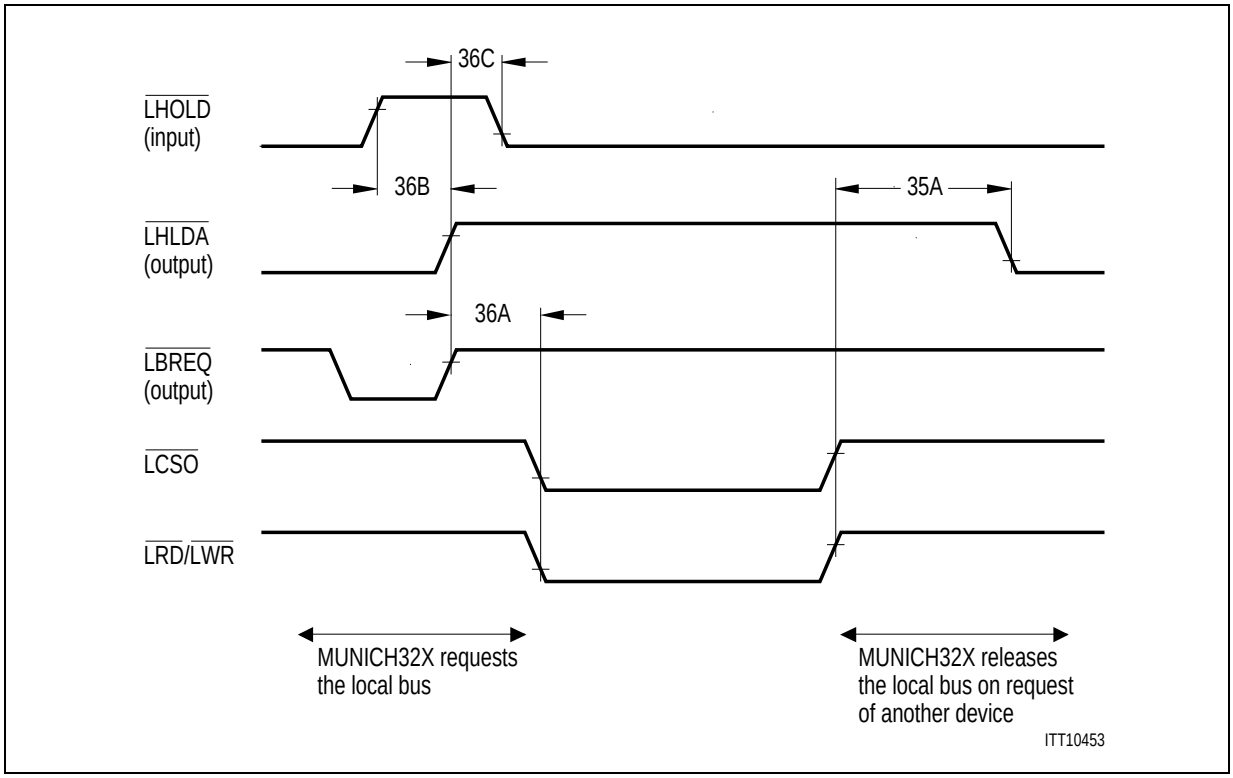


Figure 97
LBI Arbitration Timing

Table 47

Parameter	No.	Symbol	Limit Values		Unit
			min.	max.	
LA, $\overline{\text{DACK}}$, $\overline{\text{LBHE}}$, setup time (Master)	24	$t_{\text{M-su(A)}}$	0	–	ns
$\overline{\text{DACK}}$, hold time (Master)	28	$t_{\text{M-h(A)}}$	3 T_{LBICLK}		(ns)
$\overline{\text{LCSO}}$, setup time (Master)	23	$t_{\text{M-su(S)}}$	2 T_{LBICLK}		(ns)
$\overline{\text{LCSO}}$, hold time (Master)	27	$t_{\text{M-h(S)}}$	0	–	ns
LA, $\overline{\text{LBHE}}$ stable before ALE inactive (Master)	21	$t_{\text{M-su(A-ALE)}}$	1 T_{LBICLK}		(ns)
LA, $\overline{\text{LBHE}}$ hold after ALE inactive (Master)	22	$t_{\text{M-h(A-ALE)}}$	1 T_{LBICLK}		(ns)
LALE pulse width (Master)	20	$t_{\text{M-w(ALE)}}$	1 T_{LBICLK}	1.5 T_{LBICLK}	(ns)
$\overline{\text{LRD}}$ pulse width (Master)	34	$t_{\text{M-w(RD)}}$	n[1)]	–	ns
LD valid before $\overline{\text{LRD}}$ inactive (Master)	25	$t_{\text{M-su(RD)}}$	1 T_{LBICLK}	–	ns
LD hold after $\overline{\text{LRD}}$ inactive (Master)	26	$t_{\text{M-h(RD)}}$	5		(ns)
$\overline{\text{LWR}}$ pulse width (Master)	33	$t_{\text{M-w(WR)}}$	n[1)]	–	ns
LD stable after $\overline{\text{LWR}}$ active (Master)	31	$t_{\text{M-su(WR)}}$	5	–	ns
LD hold after $\overline{\text{LWR}}$ inactive (Master)	30	$t_{\text{M-h(WR)}}$	5	–	ns
Last Master Cycle inactive to $\overline{\text{LHLDA}}$ active	35A	$t_{\text{M-LA(ARB)}}$	5 T_{LBICLK}		ns
LHLDA inactive to First Master Cycle	36A	$t_{\text{LA-M(ARB)}}$	1 T_{LBICLK}		ns
LHOLD inactive to $\overline{\text{LBREQ}}$ inactive	36B	$t_{\text{LH-LR(ARB)}}$	2 T_{LBICLK}		(ns)
LHOLD inactive to $\overline{\text{LHLDA}}$ inactive	36B	$t_{\text{LH-LA(ARB)}}$	2 T_{LBICLK}		(ns)
LHLDA inactive to LHOLD active	36C	$t_{\text{LA-LH(ARB)}}$	10	–	ns

1) n depends on number of wait states: n = 2 + MCTC Wait State Cycles + additional $\overline{\text{LRDY}}$ Wait states

T_{LBICLK} is the LBI clock time period which depends on the LBI clock division factor.

14.6.4 PCM Serial Interface Timing

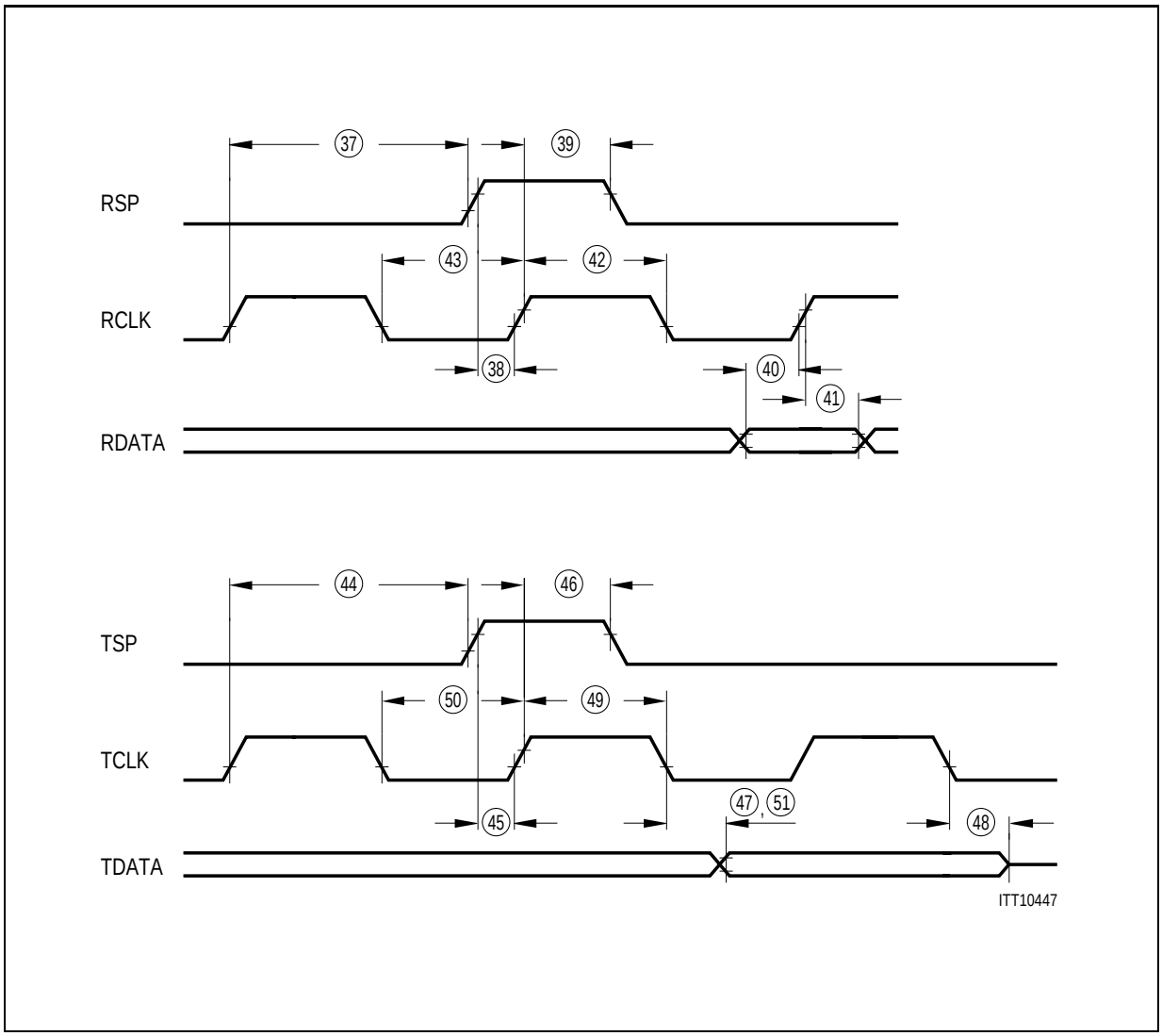


Figure 98

Table 48

No.	Parameter	Limit Values		Unit
		min.	max.	
37	Receive strobe guard time	10	–	ns
38	Receive strobe setup	5	–	ns
39	Receive strobe hold	5	–	ns
40	Receive data setup	5	–	ns
41	Receive data hold	5	–	ns

Table 48 (cont'd)

No.	Parameter	Limit Values		Unit
		min.	max.	
42	Receive clock high width	30	–	ns
43	Receive clock low width	30	–	ns
44	Transmit strobe guard time	20	–	ns
45	Transmit strobe setup	5	–	ns
46	Transmit strobe hold	5	–	ns
47	Transmit data delay	–	25	ns
48	Transmit clock to high impedance	–	25	ns
49	Transmit clock high width	30	–	ns
50	Transmit clock low width	30	–	ns
51	Transmit tristate delay	–	25	ns

Note: For complete internal or complete external loop t_{42} and t_{49} must be greater or equal to 3 times T.

System Interface Timing

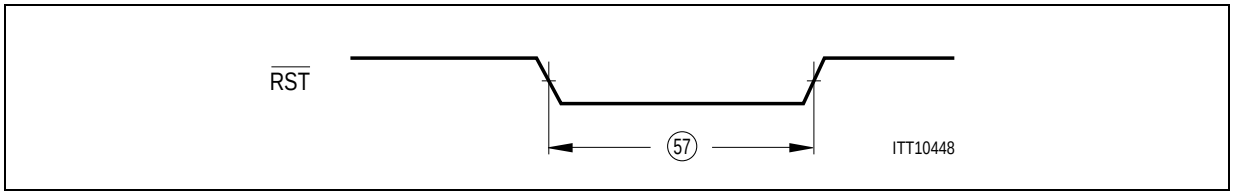


Figure 99

Table 49

No.	Parameter	Limit Values		Unit
		min.	max.	
57	RESET pulse width	10 CLK cycles	–	ns

Note: $\overline{RST}$ may be asynchronous to CLK when asserted or deasserted. Nevertheless deassertion must be clean, bounce-free edge as recommended by PCI Spec. Revision 2.1.

Note: $\overline{RST}$ signal timing is independent of whether PCI or De-multiplexed mode is selected via pin DEMUX.

14.6.5 JTAG-Boundary Scan Timing

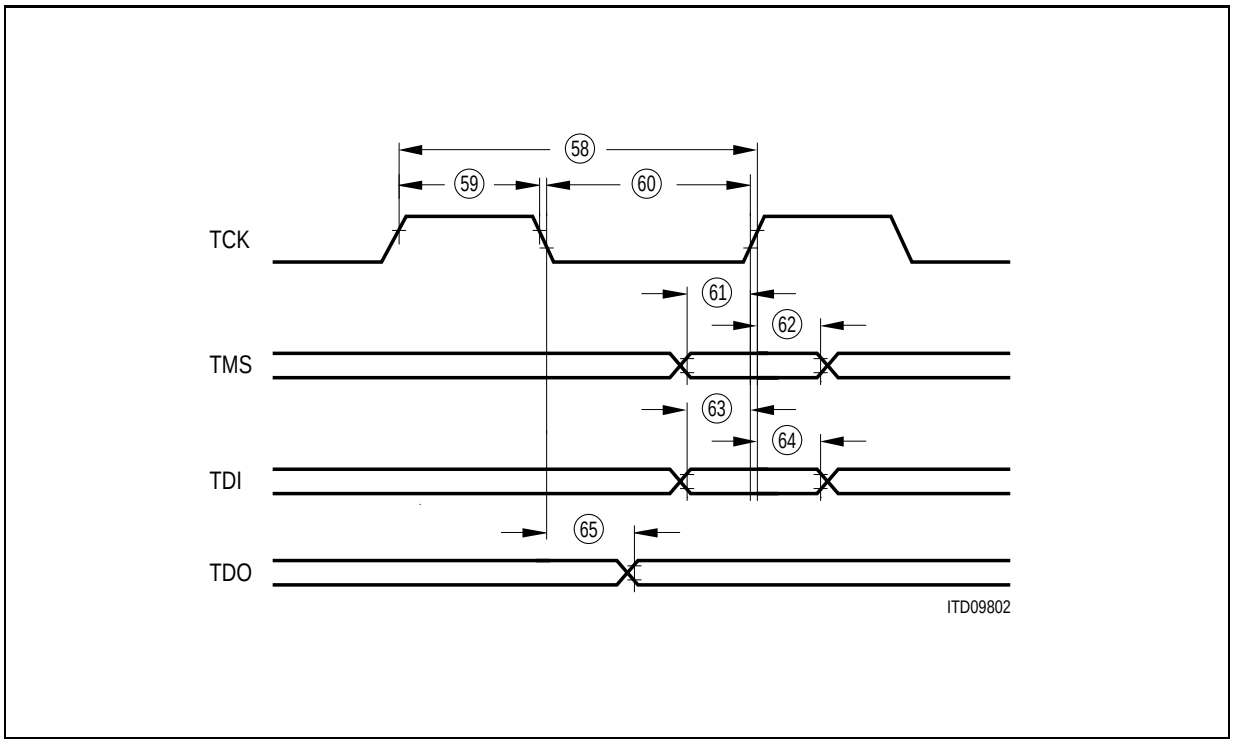


Figure 100
JTAG-Boundary Scan Timing

Table 50

No.	Parameter	Limit Values		Unit
		min.	max.	
58	TCK period	166	∞	ns
59	TCK high time	80	–	ns
60	TCK low time	80	–	ns
61	TMS setup time	30	–	ns
62	TMS hold time	10	–	ns
63	TDI setup time	30	–	ns
64	TDI hold time	20	–	ns
65	TDO valid delay	60	–	ns

14.6.6 SSC Serial Interface Timing

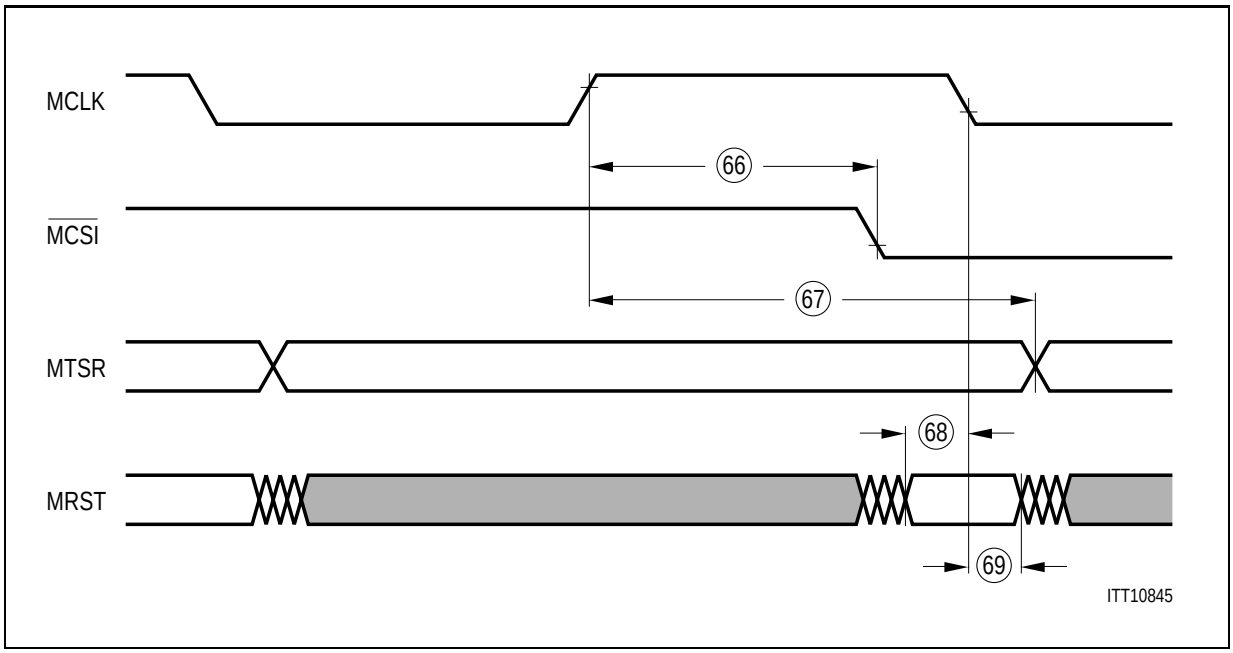


Figure 101
SSC Serial Interface Timing (Master)

Note: Figure 101 also applies to SSC slave operation. In this case 'MTSR' and 'MRST' are exchanged.

Table 51

No.	Parameter	Limit Values		Unit
		min.	max.	
66	MCLK high to $\overline{\text{MCSI}}$ active delay	–	2 T_{CLK} + 20	ns
67	MCLK high to MTSR delay (master)	–	2 T_{CLK} + 20	ns
	MCLK high to MRST delay (slave)	–	4 T_{CLK} + 20	ns
68	MRST setup time (master)	$1T_{CLK}$ + 20	–	ns
	MTSR setup time (slave)	$4T_{CLK}$ + 20	–	ns
69	MRST hold time	$1T_{CLK}$ + 20	–	ns
	MTSR hold time (slave)	$4T_{CLK}$ + 20	–	ns

15 MUNICH32X Bus Utilization

15.1 General

The MUNICH32X operates on linked lists within the shared memory. Handshaking with the software also operating on the linked lists is performed via status information in the linked descriptors and interrupt vectors which are written into queues also located in the shared memory. In addition current descriptor addresses are written to the Channel Configuration Block (CCB) which provide information about the MUNICH32X current position in the linked descriptor lists. Evaluation of this current addresses depend on the software implementation.

Configuration is handled via on-chip registers and through the Channel Configuration Block (CCB). The CCB is prepared by software in the shared memory and read by the MUNICH32X on request.

After initialization and configuration no further on-chip register access is neccessary other than handling of exception conditions or configuration changes.

- Transmit data flow:
  The Host CPU prepares transmit data in linked lists. The MUNICH32X gets the start address of this list via configuration procedure and starts processing the list by reading the first descriptor (3 DWORDs read burst) and the corresponding data section (multiple single DWORD read transfers depending on the packet size). After finishing one descriptor an interrupt vector is generated and written to the corresponding transmit interrupt queue. The MUNICH32X proceeds branching to the next descriptor and updating the current descriptor address in the CCB.
  For bus load calculation it is assumed that no linked list end condition occurs i.e. the CPU always attaches new descriptors to the chain and that one interrupt vector per packet is generated.
- Receive data flow.
  The Host CPU prepares a linked list of ‘empty’ receive descriptors and corresponding receive data buffers. The MUNICH32X gets the start address of this list via configuration procedure and starts reading the first descriptor (3 DWORDs read burst). Receive data is transferred to the receive data section by multiple single DWORD transfers depending on the packet size. After transfer of one complete packet the receive descriptor is finished when the MUNICH32X overwrites one DWORD in the descriptor with status and control information. One Interrupt vector is generated and written into the corresponding receive interrupt queue. The MUNICH32X proceeds branching to the next descriptor and updating the current descriptor address in the CCB.
  For bus load calculation it is assumed that no linked list end condition occurs i.e. the CPU always attaches new descriptors to the chain and that one interrupt vector per received packet is generated.

Data flow is illustrated in **Figure 102**.

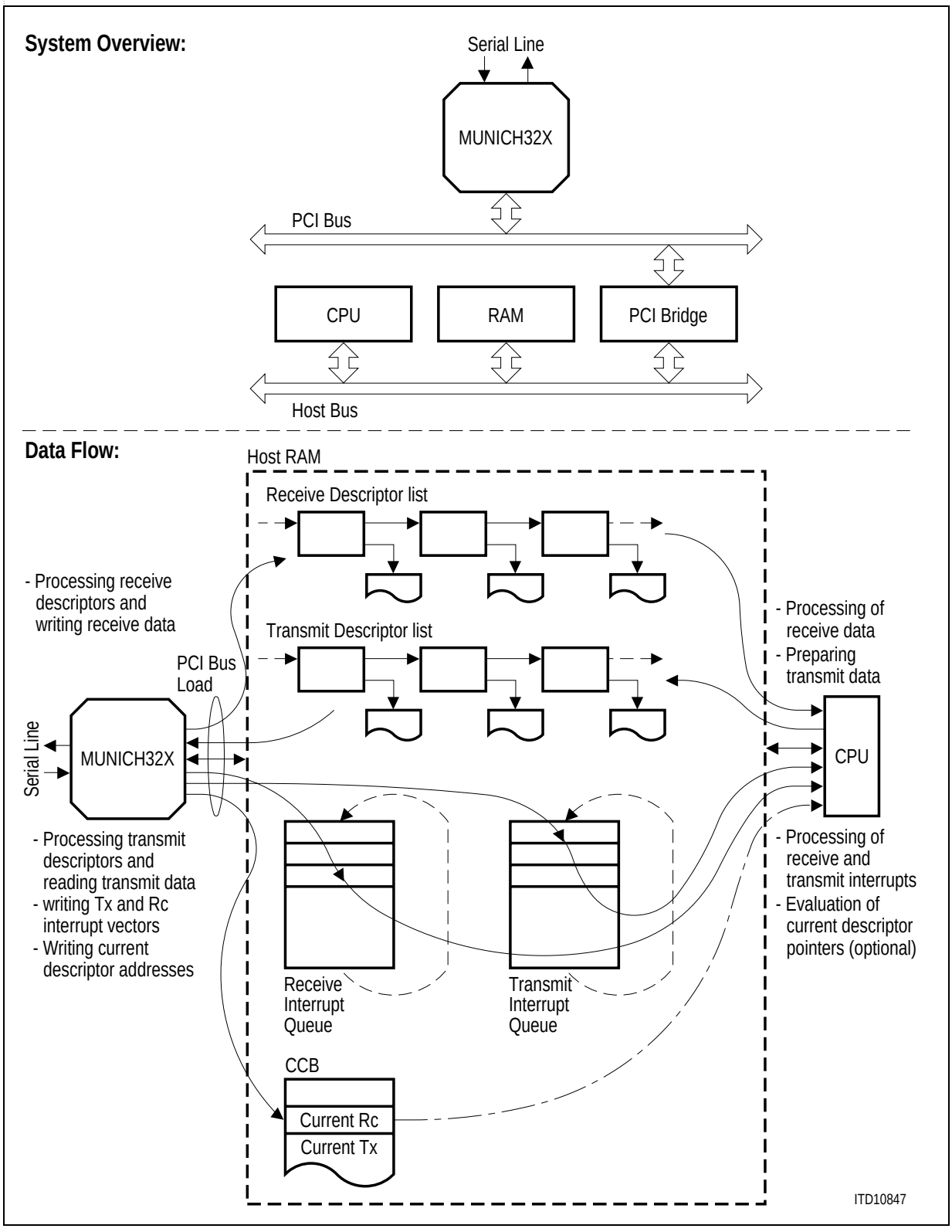


Figure 102
Data Flow

15.2 PCI Bus Cycle Calculations

This chapter provides a calculation of PCI clock cycles needed for all transactions neccessary for receiving and transmitting a data packet (HDLC mode).

List of abbreviations

Table 52

A	PCI Address cycle
T	PCI Turn arround cycle
W_{ir}	PCI Initial wait states on Read
W_r	PCI Wait states on consecutive Read cycles within a burst transfer
W_{iw}	PCI Initial wait states on Write
D	Data cycle
M	Packet size in number of bytes (payload bytes between start flag and CRC)
D_r	Serial line data rate (nominal)
N	Number of DWORDs per data packet

15.2.1 Transmit Descriptor and Data Processing

- Transmit descriptor read cycles T_d:

  $T_d = A + T + W_{ir} + 3 \times D + 2 \times W_r + T$

- Transmit data read cycles T_{data}:

  $T_{data} = N \times [A + T + W_{ir} + 1 \times D + T]$

  (N is the number of DWORDs equivalent to one transmit packet):

  $$N = \lfloor (M + 3)/4 \rfloor$$

- Current transmit descriptor address update cycle T_{dau}:

  $T_{dau} = A + W_{iw} + 1 \times D + T$

15.2.2 Transmit Interrupt Overhead

- Transmit interrupt vector write cycle T_{iv}:

  $T_{iv} = A + W_{iw} + 1 \times D + T$

15.2.3 Receive Descriptor and Data Processing

- Receive descriptor read cycles R_d:

  $R_d = A + T + W_{ir} + 3 \times D + 2 \times W_r + T$

- Receive data write cycles R_{data}:

  $R_{data} = N \times [A + W_{iw} + 1 \times D + T]$

  (N is the number of DWORDs equivalent to one receive packet.)

- Receive descriptor update cycle R_{du}:

  $R_{du} = A + W_{iw} + 1 \times D + T$

- Current receive descriptor address update cycle R_{dau}:

  $R_{dau} = A + W_{iw} + 1 \times D + T$

15.2.4 Receive Interrupt Overhead

- Receive interrupt vector write cycle R_{iv}:

  $R_{iv} = A + W_{iw} + 1 \times D + T$

15.3 PCI Bus Utilization Calculation

The effective serial data rate depends on the protocol mode. In HDLC mode with 16 bit CRC and shared flags (assuming back-to-back packet transmission and reception), the effective data rate D_{er} is

$$D_{er} = D_r \times M/(M + 3).$$

The bus utilization U can now be calculated

$$U = 1/f_{PCI} \times D_{er}/32 \times (T_d + T_{data} + T_{dau} + T_{iv} + R_d + R_{data} + R_{du} + R_{dau} + R_{iv}).$$

15.4 PCI Bus Utilization Example

The equations above are calculated for two cases:

1. Ideal system with parameter set:
   $W_{ir} = 0$
   $W_{iw} = 0$
   $W_r = 0$
2. Typical system with parameter set:
   $W_{ir} = 7$
   $W_{iw} = 1$
   $W_r = 0$
   (These parameters are typical for an INTEL 430 FX PCI chipset as used on many PC and workstation platforms. Thus the same results are achieved by using a PCI analyzer running in a real system)

For this calculation the serial line is operating in E1-Mode on all 32 channels with a total nominal bit rate of 2.048 MBit/s or in T1-Mode on all 24 channels with a total nominal bit rate of 1.536 MBit/s. The bus utilization is calculated depending on the packet size in number of bytes which is assumed to be constant in transmit and receive direction.

The figures below show the bus utilization and how bus utilization is composed distinguished by data transfer (data sections) and control overhead (descriptors, interrupts, current pointers).

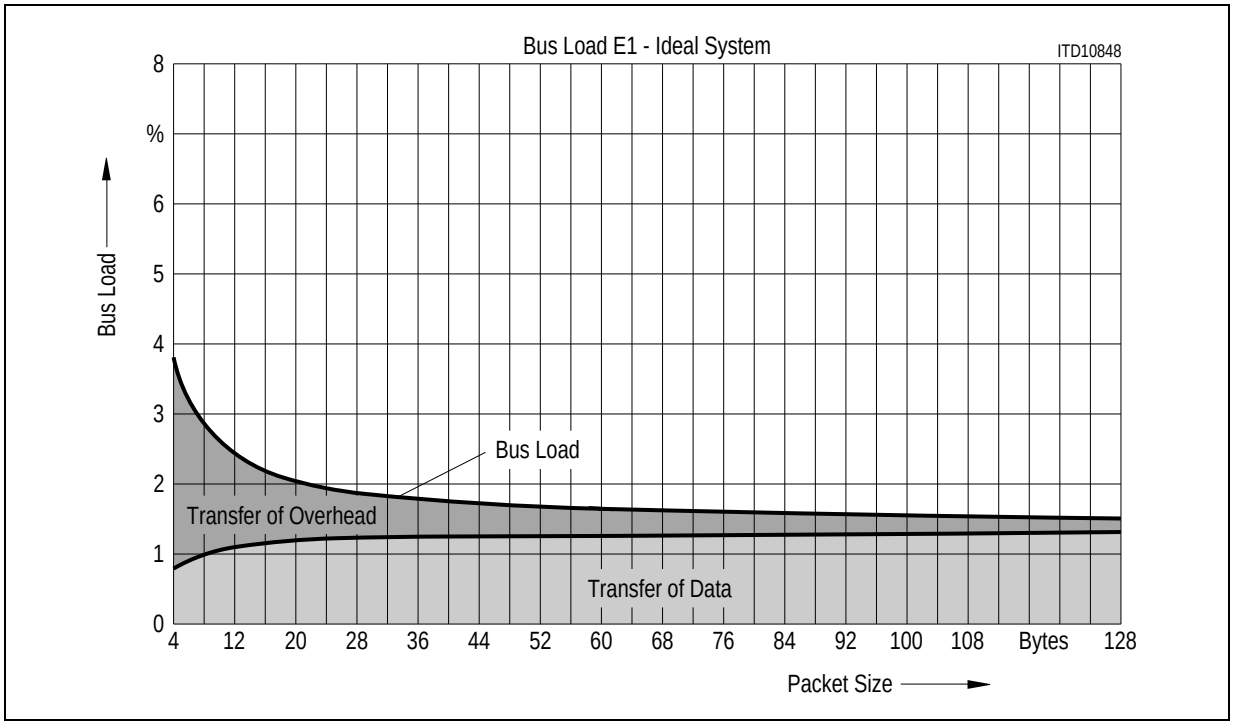


Figure 103
Ideal System, E1

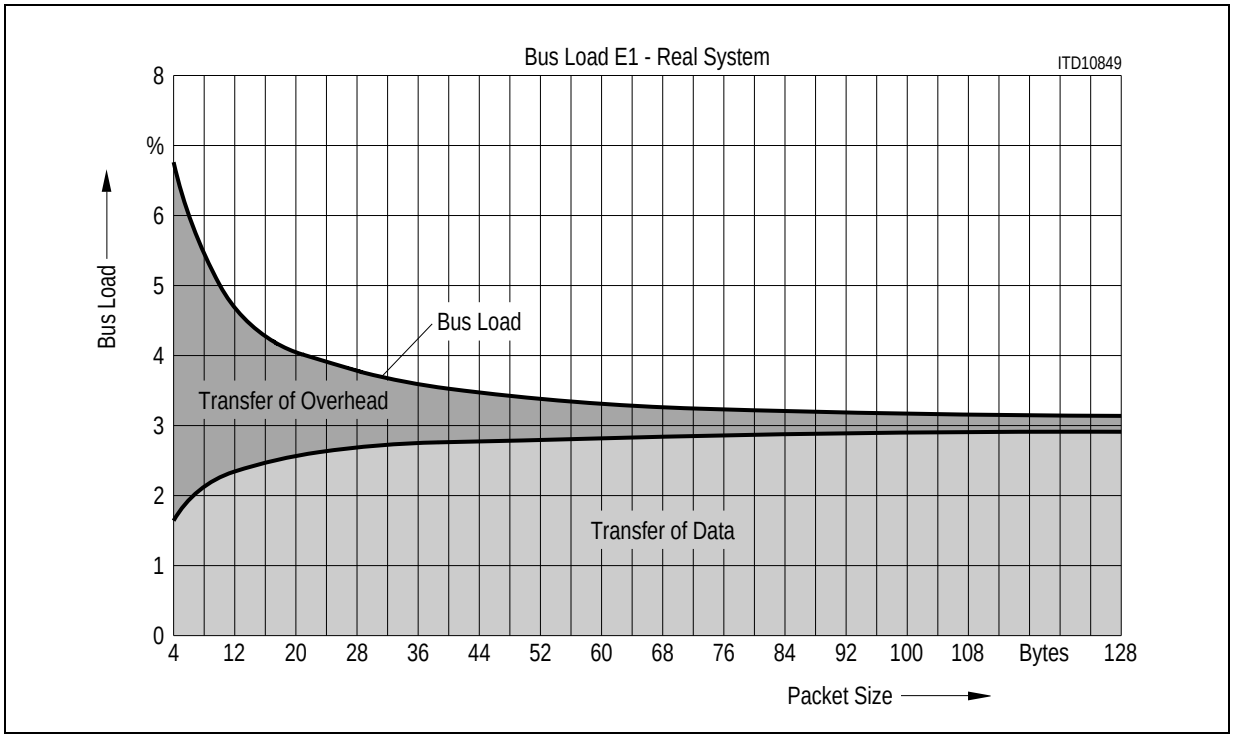


Figure 104
Typical System, E1

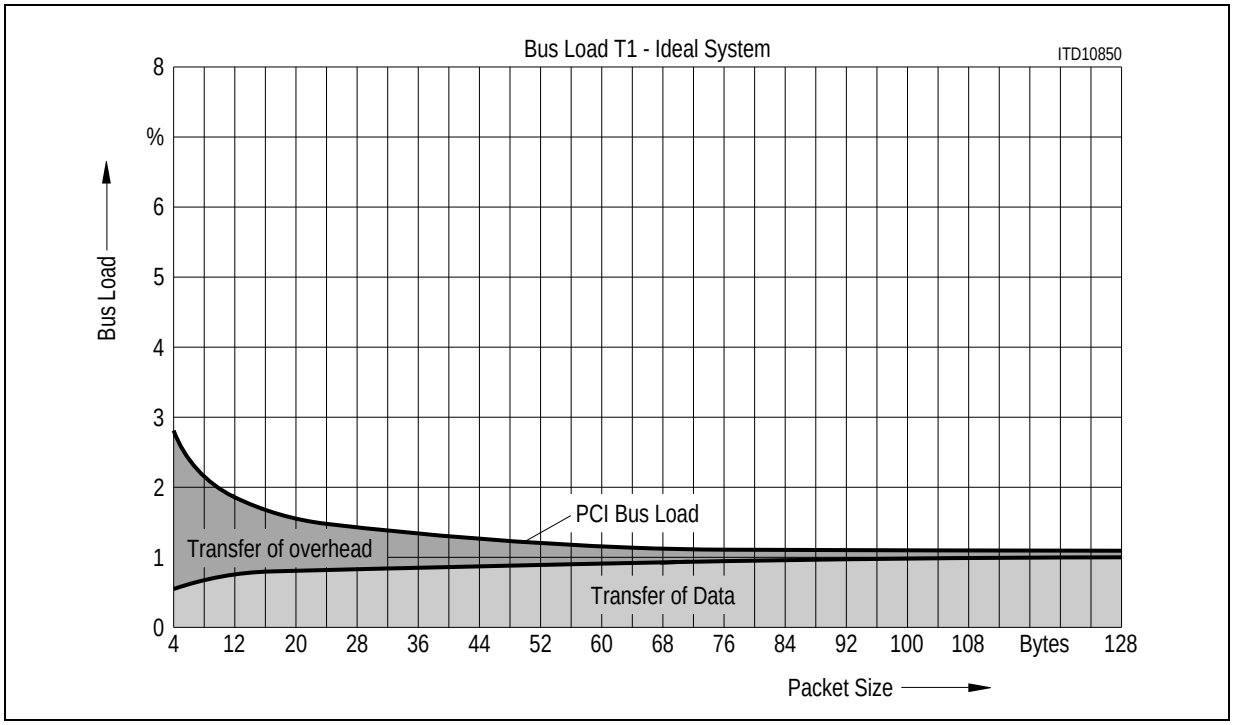


Figure 105
Ideal System, T1

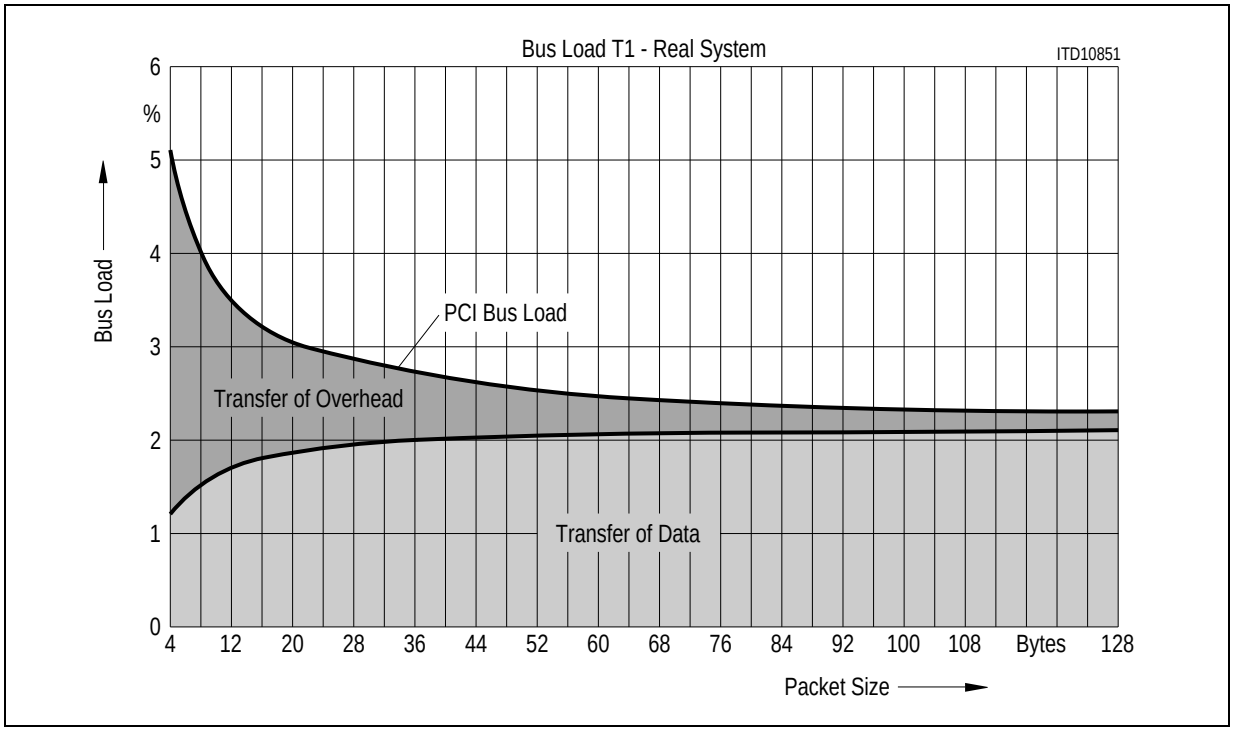


Figure 106
Typical System, T1

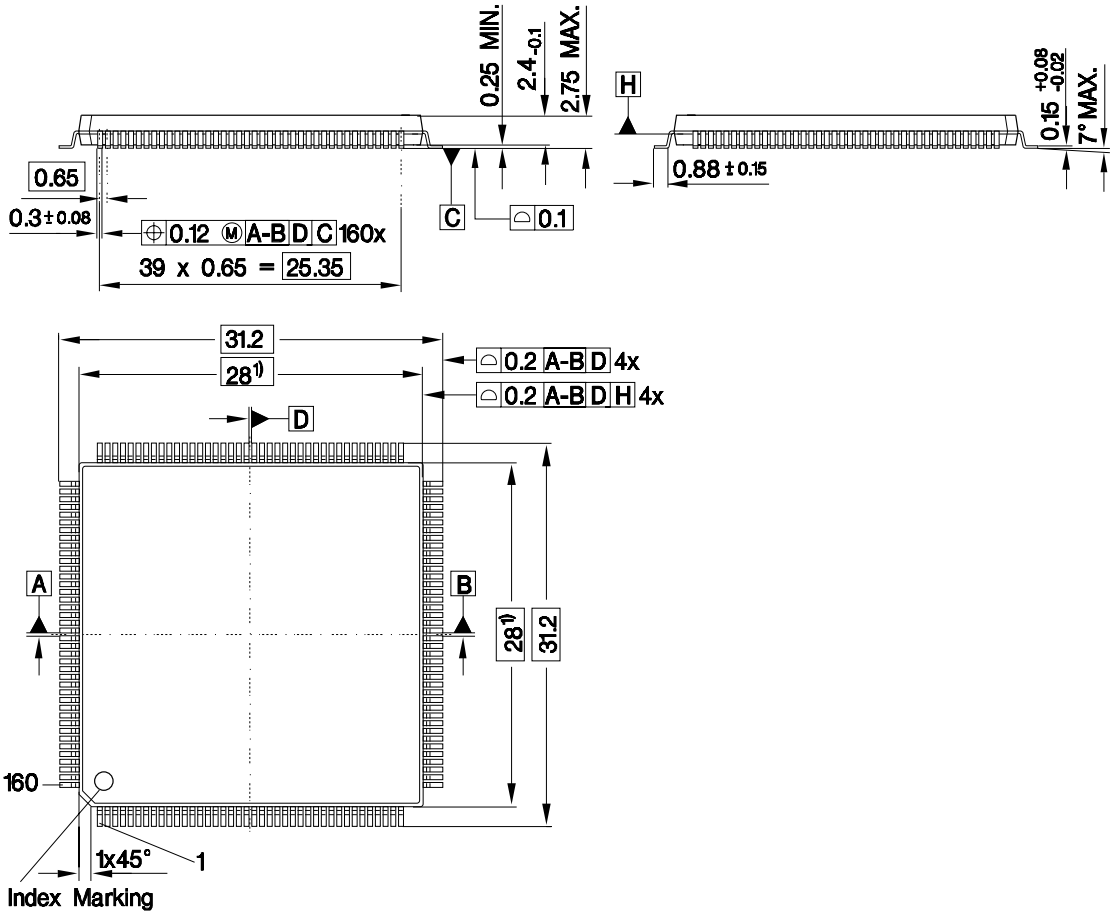
The control overhead per packet is constant which results in an increased bus load if the serial line data rate is achieved by very small packets. The overhead can be neglected for packet sizes greater than 32 bytes.

On the serial line the 16 bit CRC and one shared flag per packet also appears as a constant overhead which reduces the effective data rate for small packets.

The influence of bit-stuffing in HDLC protocol mode also reducing the effective data rate is neglected.

16 Package Outlines

P-MQFP-160-1
(Plastic Metric Quad Flat Package)



1) Does not include plastic or metal protrusion of 0.25 max. per side

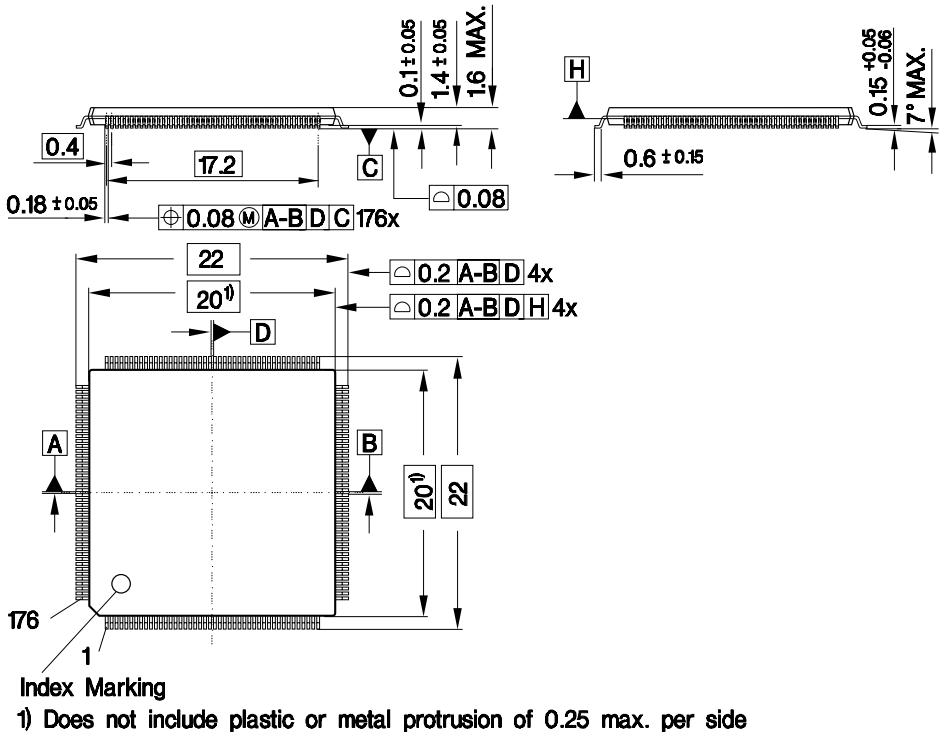
GPM05247

Sorts of Packing
Package outlines for tubes, trays etc. are contained in our Data Book “Package Information”.

SMD = Surface Mounted Device

Dimensions in mm

P-TQFP-176-1
(Plastic Thin Quad Flat Package)



Sorts of Packing
Package outlines for tubes, trays etc. are contained in our Data Book ‘Package Information’.

SMD = Surface Mounted Device

Dimensions in mm