CTDMA r01d0

User Manual

Document version	v1.1
Document status	Released
Date of issue	12 August 2026
Confidentiality Security Class	C-SC2 Confidential

Mobility Electronics
Robert Bosch GmbH

1 Cut Through Direct Memory Access (CTDMA) module

CTDMA is an add on module to the XS_CAN controller IP which enables the XS_CAN IP to perform cut through block copy operations thereby offloading the task from system DMA controllers. 4 instances of XS_CAN IPs can be connected to one CTDMA module and the data transfer from each XS_CAN IP is redirected back and forth to the system memory (SMEM) during block copy transfer by CTDMA. Apart from block copy transfers, CTDMA also supports host accesses to the LMEM space of XS_CAN IP for enqueue, dequeue, CREG read write etc via the AHB slave interface in CTDMA.

1.1 Key Features

-AXI master interface for block copy transfer between SMEM and XS_CAN IPs
-AHB slave interface for host accesses to XS_CAN IP.
-AHB Master interface per XS_CAN IP
-Supports 4 XS_CAN IPs.
-Supports burst transfers at all interfaces.
-One internal buffer of 64byte in CT_MANAGER.
-Internal arbiter to prioritize between host accesses and Cut through transfers
-AHB-APB bridge to convert CREG access at AHB slave to APB interface in CREG.

1.2 Block Diagram

The following block diagram shows the structure of CTDMA.

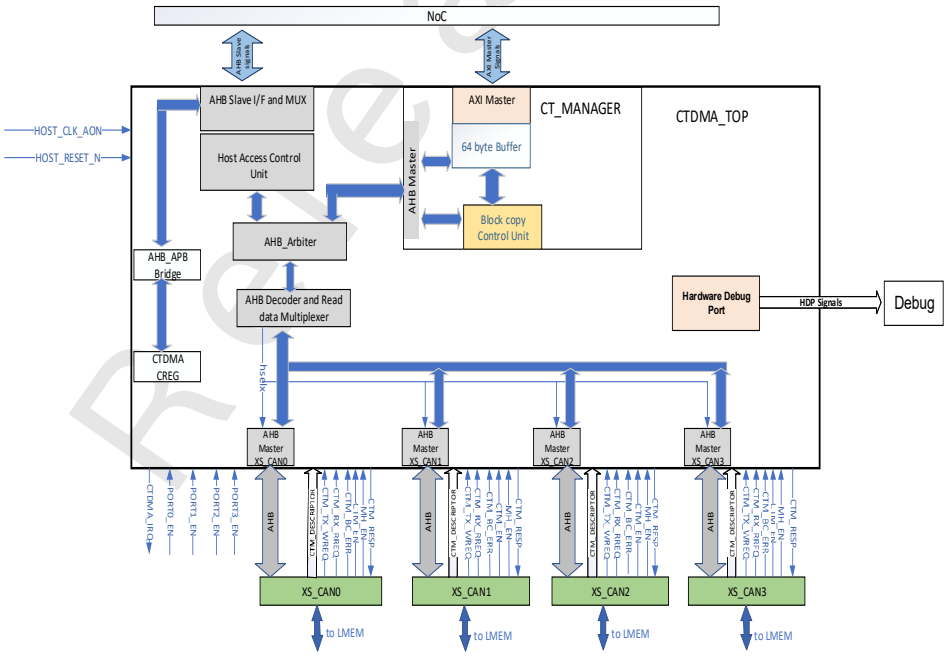

Figure 1. CTDMA Block Diagram

1.3 TOP - Top Level

1.3.1 Software Interface

1.3.1.1 CTDMA Memory Map

Table 1. CTDMA Memory Map

Start Location Address	End Location Address	Symbol	Name
0x000000	0x07FFFC	XSCAN 0	XSCAN 0 memory map
0x080000	0x0FFFFC	XSCAN 1	XSCAN 1 memory map
0x100000	0x17FFFC	XSCAN 2	XSCAN 2 memory map
0x180000	0x1FFFFC	XSCAN 3	XSCAN 3 memory map
0x200000	0x200B0C	CTDMA reg	Cut - Through DMA registers
0x200B10	0x3FFFFC	Reserved	Reserved Address

1.3.1.2 XS_CAN IP Memory Map

Table 2. XS_CAN IP Memory Map

Start Location Address	End Location Address	Symbol	Name
0x00000	0x008FC	MH reg	Message Handler registers
0x00900	0x009FC	PRT reg	Protocol Controller registers
0x00A00	0x00AFC	IRC reg	Interrupt Controller registers
0x00B00	0x00FFC	reserved	reserved
0x01000	0x01808	TXFQ	TX FIFO Queue
0x0180C	0x01FFC	reserved	reserved
0x02000	0x02808	TXPQ	TX Priority Queue
0x0280C	0x02FFC	reserved	reserved
0x03000	0x03810	RXFQ0	RX FIFO 0 Queue
0x03814	0x03FFC	reserved	reserved
0x04000	0x04810	RXFQ1	RX FIFO 1 Queue
0x04814	0x04FFC	reserved	reserved
0x05000	0x05010	TEFQ	TX Event FIFO Queue
0x05014	0x05FFC	reserved	reserved
0x06000	0x0603C	CTB	Cut Through Buffer
0x06040	0x3FFFC	reserved	reserved
0x40000	0x7FFFC	LMEM TA	LMEM Transparent Access

1.3.1.3 CTDMA MAP

1.3.1.3.1 Overview

Table 3. Address Blocks list

Address Block	Offset	Range	Description
ctdma_creg	0x0000	0x0FFF	Usage: register Global CTDMA control and status registers

Table 4. Registers overview (page 1 of 2)

Register name	Offset	Mode	Description
ctdma_creg			
CTDMA_VERSION	0x0000	R	Register size: 32 Release Identification Register
CTDMA_CFG	0x0004	RW	Register size: 32 Cut Through DMA Configuration register

Table 4. Registers overview (page 2 of 2)

Register name	Offset	Mode	Description
CTDMA_SFTY_CFG	0x0008	RW	Register size: 32 Cut Through DMA Safety Configuration register
CTDMA_STS0	0x000C	R	Register size: 32 Cut Through DMA Status register 0
CTDMA_STS1	0x0010	R	Register size: 32 Cut Through DMA Status register 1
CTDMA_STS2	0x0014	R	Register size: 32 Cut Through DMA Status register 2
CTDMA_STS3	0x0018	R	Register size: 32 Cut Through DMA Status register 3
STS0_STS1_CTRL	0x001C	RW	Register size: 32 Cut Through DMA STS0 STS1 Control register
DEBUG_TEST_CTRL	0x0020	RW	Register size: 32 Cut Through DMA Debug Test Control register
IRQ_RAW	0x0B00	R	Register size: 32 Interrupt RAW register
IRQ_ENA	0x0B04	RW	Register size: 32 Interrupt ENABLE register
IRQ_CLR	0x0B08	W	Register size: 32 Interrupt CLEAR register
CAPTURING_MODE	0x0B0C	R	Register size: 32 IRC configuration register. This register shows the hardware configuration of the IRC concerning the capturing mode of the event inputs. The IP internal events signals require an 'edge sensitive' capturing. That is why the value of this register is 0x1 and cannot be changed.

1.3.1.3.2 ctdma_creg Address Block

Table 5. CTDMA_VERSION register

Release Identification Register

Bit	Field	Mode	Initial Value	Description
11:8	MAJOR	R	0x1	Release name, used to identify the Major generation of the CTDMA.
7:4	MINOR	R	0x0	Release name, used to identify the Minor generation of the CTDMA.
3:0	PATCH	R	0x0	Release name, used to identify the Patch number for the CTDMA.

Table 6. CTDMA_CFG register

Cut Through DMA Configuration register

Bit	Field	Mode	Initial Value	Description
9:0	ARB_THROTTLE	R/W	0x1	Arbiter Throttle value Allowed values are 0 - No throttling 1 - 20% throttling 2 - 33% throttling 3 - 50% throttling 4 - 66% throttling 5 - 80% throttling 6 and above - Reserved.

Table 7. CTDMA_SFTY_CFG register

Cut Through DMA Safety Configuration register

Bit	Field	Mode	Initial Value	Description
27	PARITY_EN	R/W	0x0	Parity Enable When set to 1, the parity is enabled: Parity covers the data parity (on CTMANAGER side) and address parity (on host access side).
26:17	AHB_MST_TO_VAL	R/W	0x0	This value is used by the watchdog timer for the AHB Master interface and defines the maximum number of timer ticks until a read or write access has to be completed. This value must be configured to the expected maximum latency on the AHB interface. If this value is set to 0 and CTDMA_SFTY_CFG.AHB_MASTER_TO_EN = 1 then the AHB_TO_ERR interrupt is triggered.
16	AHB_MST_TO_EN	R/W	0x0	When set to 1, the watchdog for the AHB interface is enabled, otherwise disabled.
10:3	AXI_MST_TO_VAL	R/W	0x0	This value is used by the watchdog timer for the AXI Master interface and defines the maximum number of timer ticks until a read or write access has to be completed. This value must be configured to the expected maximum latency on the AXI interface. If this value is set to 0 and CTDMA_SFTY_CFG.AXI_MASTER_TO_EN = 1 then the AXI_TO_ERR interrupt is triggered right away when accessing the AXI Master.
2:1	AXI_MST_PRESCALER	R/W	0x0	Prescaler used to generate the timer ticks for the watchdogs. . According to the value a different clock ratio can be selected: 0: clk divided by 64 1: clk divided by 256 2: clk divided by 1024 3: clk divided by 4096
0	AXI_MST_TO_EN	R/W	0x0	When set to 1, the watchdog for the AXI interface is enabled, otherwise disabled.

Table 8. CTDMA_STS0 register

Cut Through DMA Status register 0

Bit	Field	Mode	Initial Value	Description
31:0	SMEM_ADDR_BC	R	0x0	Indicates the SMEM address issued at AXI Master for ongoing Block Copy transfer.

Table 9. CTDMA_STS1 register (page 1 of 2)

Cut Through DMA Status register 1

Bit	Field	Mode	Initial Value	Description
16	AHB_SRC	R	0x0	0 - Host AHB, 1 - CTMANAGER
15:14	AHB_XSCAN_INST	R	0x0	XS_CAN Slave that got accessed
13	AHB_ARB_ACTIVE	R	0x0	1 indicates Active request from arbiter towards XS_CANs
12	BC_AHB_TO_ACTIVE	R	0x0	1 indicates AHB TO is still active for the current block copy request to XS_CAN (No response received from XS_CAN)
11	BC_AXI_TO_ACTIVE	R	0x0	1 indicates AXI TO is still active for the current block copy request to SMEM (No response received from SMEM)
10:9	BC_XSCAN_INST	R	0x0	XS_CAN instance number for which block copy transfer is in progress at AXI Master. Identical to AXI AWUSER and ARUSER.
8:7	BC_TX_RX_REQ	R	0x0	Block copy request being executed. 0 indicates no TX or RX request, 1 indicates CTM_TX_WREQ is served for the XS_CAN instance in BC_XSCAN_INST, 2 indicates CTM_RX_RREQ is served for XS_CAN instance in BC_XSCAN_INST, 3- not used.

Table 9. CTDMA_STS1 register (page 2 of 2)

Cut Through DMA Status register 1

Bit	Field	Mode	Initial Value	Description
6:3	BC_BURST_LEN	R	0x0	Burst length of the block copy transfer in progress. Possible values are 0 to 15. Actual burst lenth = BC_BURST_LEN+1.
2:1	BC_SMEM_ACC_TYPE	R	0x0	SMEM access type which can be either read or write. 0-no block copy process, 1 indicates read and 2 indicates write; 3 not used.
0	BC_ACTIVE	R	0x0	1 indicates BC transfer is in progress in CTMANAGER. BC_XSCAN_INST and BC_SMEM_ACC_TYPE values are valid only if BC_ACTIVE is 1.

Table 10. CTDMA_STS2 register

Cut Through DMA Status register 2

Bit	Field	Mode	Initial Value	Description
31:24	BC_REQ	R	0x0	Block copy requests in the order: [XS_CAN3 TX, XS_CAN3 RX, XS_CAN2 TX, XS_CAN2 RX,XS_CAN1 TX, XS_CAN1 RX, XS_CAN0 TX, XS_CAN0 RX]
19:16	PORT_EN	R	0x0	[PORT_EN_3, PORT_EN_2, PORT_EN_1, PORT_EN_0]
11:8	MH_EN	R	0x0	[MH_EN_3, MH_EN_2, MH_EN_1, MH_EN_0]
3:0	CTM_EN	R	0x0	[CTM_EN_3, CTM_EN_2, CTM_EN_1, CTM_EN_0]

Table 11. CTDMA_STS3 register

Cut Through DMA Status register 3

Bit	Field	Mode	Initial Value	Description
22:1	AP_ERR_AHB_HADDR	R	0x0	Corrupted AHB address which resulted in address parity error.
0	AP_ERR_RW	R	0x0	Type of access during parity error. 0 - indicates read transfer and 1 indicates write transfer.

Table 12. STS0_STS1_CTRL register

Cut Through DMA STS0 STS1 Control register

Bit	Field	Mode	Initial Value	Description
0	STS_FROZEN	R/W	0x0	1 indicates that CTDMA_STS0 and CTDMA_STS1 register contents are frozen and not updated by CTDMA until this bit is reset. This bit is set by CTDMA when there is an AHB or AXI timeout. This bit must be reset by the host after reading CTDMA_STS0 and CTDMA_STS1 registers. Write 1 by host is ignored.

Table 13. DEBUG_TEST_CTRL register

Cut Through DMA Debug Test Control register

Bit	Field	Mode	Initial Value	Description
2:1	HDP_SEL	R/W	0x0	HDP select signal; 0 - selects set1, 1 - selects set 2. 2 and 3 are reserved
0	HDP_EN	R/W	0x0	HDP select signal; 1 - enables the HDP ports to monitor CTDMA signals

Table 14. IRQ_RAW register (page 1 of 2)

Interrupt RAW register

Bit	Field	Mode	Initial Value	Description
6	AXI_SLVERR	R	0x0	interrupt generated when slave error response is received at AXI Master interface for read and write to SMEM.
5	WR_ADDR_PARITY_ERR	R	0x0	Address parity error for AHB slave write address.
4	RD_ADDR_PARITY_ERR	R	0x0	Address parity error for AHB slave read address.
3	DATA_PARITY_ERR	R	0x0	Data parity error

Table 14. IRQ_RAW register (page 2 of 2)

Interrupt RAW register

Bit	Field	Mode	Initial Value	Description
2	AHB_TO	R	0x0	AHB Master Timeout interrupt
1	AXI_TO	R	0x0	AXI Master Timeout interrupt
0	ARA	R	0x0	Access to reserve address

Table 15. IRQ_ENA register

Interrupt ENABLE register

Bit	Field	Mode	Initial Value	Description
6	AXI_SLVERR	R/W	0x0	Writing 1 enables the bit IRQ_RAW.AXI_SLVERR to generate CTDMA_IRQ
5	WR_ADDR_PARITY_ERR	R/W	0x0	Writing 1 enables the bit IRQ_RAW.WR_ADDR_PARITY_ERR to generate CTDMA_IRQ
4	RD_ADDR_PARITY_ERR	R/W	0x0	Writing 1 enables the bit IRQ_RAW.RD_ADDR_PARITY_ERR to generate CTDMA_IRQ
3	DATA_PARITY_ERR	R/W	0x0	Writing 1 enables the bit IRQ_RAW.DATA_PARITY_ERR to generate CTDMA_IRQ
2	AHB_TO	R/W	0x0	Writing 1 enables the bit IRQ_RAW.AHB_TO to generate CTDMA_IRQ
1	AXI_TO	R/W	0x0	Writing 1 enables the bit IRQ_RAW.AXI_TO to generate CTDMA_IRQ
0	ARA	R/W	0x0	Writing 1 enables the bit IRQ_RAW.ARA to generate CTDMA_IRQ

Table 16. IRQ_CLR register

Interrupt CLEAR register

Bit	Field	Mode	Initial Value	Description
6	AXI_SLVERR	W	0x0	Writing 1 clears the bit IRQ_RAW.AXI_SLVERR
5	WR_ADDR_PARITY_ERR	W	0x0	Writing 1 clears the bit IRQ_RAW.WR_ADDR_PARITY_ERR
4	RD_ADDR_PARITY_ERR	W	0x0	Writing 1 clears the bit IRQ_RAW.RD_ADDR_PARITY_ERR
3	DATA_PARITY_ERR	W	0x0	Writing 1 clears the bit IRQ_RAW.DATA_PARITY_ERR
2	AHB_TO	W	0x0	Writing 1 clears the bit IRQ_RAW.AHB_TO
1	AXI_TO	W	0x0	Writing 1 clears the bit IRQ_RAW.AXI_TO
0	ARA	W	0x0	Writing 1 clears the bit IRQ_RAW.ARA

Table 17. CAPTURING_MODE register

IRC configuration register. This register shows the hardware configuration of the IRC concerning the capturing mode of the event inputs. The IP internal events signals require an 'edge sensitive' capturing. That is why the value of this register is 0x1 and cannot be changed.

Bit	Field	Mode	Initial Value	Description
0	IRQ_RAW	R	0x1	0 = Level Sensitive 1 = Edge Sensitive

1.3.2 Interrupt Controller

Interrupt controller is provided in CTDMA to generate interrupt to host. One interrupt *CTDMA_IRQ* is generated at the top which is a level triggered interrupt. Logical 1 on the line shows an active interrupt. The output interrupt lines are triggered after two host clock cycles after the corresponding source gets activated at IRC raw register.

The sources of the interrupt are stored in IRQ_RAW register. IRQ_ENA register consists of the enable bits for the raw flags. Only if IRQ_ENA.<flag> is set, interrupt gets generated at the top. Note that the flags in the IRQ_RAW register will be set irrespective of whether the bit is enabled or not. The interrupt output is the OR of all enabled IRQ raw bits. To clear the interrupt, host has to write 1 to the corresponding bit in the IRQ_CLR register. IRQ_CLR is auto clear register. This will clear the corresponding interrupt source flag in IRQ_RAW register. Overall, not more than 5 clock cycles are taken to trigger the interrupt to the system from the time of occurrence of the cause.

1.3.3 Functional Description

1.3.3.1 AXI Master Interface

CTDMA has one AXI master interface to perform block copy read and write to SMEM. The read and write channel data width is 32bits. The interface is compliant to AMBA4 protocol. In TX direction, AXI read transfers from SMEM are issued at AXI master interface. In RX direction, AXI write transfers to SMEM are issued at this interface. AXI transfers are always burst transfer. Refer [6] in references section for details on the protocol.

1.3.3.2 AHB Slave interface and Multiplexer

CTDMA implements an AMBA 2 AHB-compliant slave interface intended for point-to-point connectivity, means one master connects to one slave. Refer Integration Guideline section[12] for the requirements at the integration level.
Host can perform virtual buffer accesses of all XS_CANs for TXFQ/TXPQ enqueue, RXFQ dequeue, TX event FIFO dequeue, LMEM transparent accesses and XS_CAN CREG accesses using this interface. Host accesses to cut through buffers of XS_CAN IPs when a block copy request is active can result in corrupting of data or unpredictable behavior in CTDMA. Hence it is not recommended for host to access cut through buffers, but CTDMA does not block this access.

SINGLE, INCR4, INCR8 and INCR16 burst types are supported at this interface. WRAP4, WRAP8 and WRAP16 bursts and burst of infinite length are not supported. The supported read and write data width is always 32bits.i.e. HSIZE[2:0]=word. If the host tries to send unsupported accesses, error response is given back by the IP.

The Multiplexer present at the AHB slave interface redirects the accesses to CTDMA CREG and Host Access Control Unit.

1.3.3.3 Host Access Control Unit (HACU)

Host access control unit controls the accesses from host to the different XS_CAN IPs connected to the CTDMA. When host issues a transaction (read or write) to the connected XS_CAN IPs, HACU generates the request to AHB arbiter and waits for the grant. Once grant is received, the address and control signals are placed at the AHB decoder to generate the slave select signal.

1.3.3.4 AHB Master Interface

CTDMA supports one AHB master interface per XS_CAN IP connected to it. Data bus width is 32bits and address bus width is 19bits. The AHB slave interface of XS_CAN IP can be connected to one of the AHB master interfaces. These AHB master interfaces fanned out from one AHB master module inside the CTDMA. This is compliant to AMBA 2 protocol. All accesses from host and CTMANAGER are routed to XS_CAN IPs through AHB master interface ports.
Main features are:
i) SINGLE, INCR4, INCR8 and INCR16 burst accesses are supported.
ii) WRAP bursts are not supported.

1.3.3.5 AHB Arbiter

XS_CAN IPs connected to the CTDMA module can be accessed by either the host or the AHB master in the CTDMA module itself. The AHB arbiter present in CTDMA performs arbitration between the block copy requests from CTMANAGER and accesses from the host. Priority is given for block copy request from the CTMANAGER AHB Master module.

1.3.3.6 AHB Decoder and Read data Multiplexer

Host accesses at AHB slave interface of CTDMA must be routed to the corresponding XS_CAN IP connected to the CTDMA module. After AHB arbiter grants access to one of the requests, AHB decoder decodes the AHB address coming from the AHB arbiter and generates the corresponding slave select signal. For read accesses to XS_CAN IPs, the read data is redirected to the appropriate master via the read data multiplexer.

1.3.3.7 CTMANAGER internal buffer

CTDMA supports an internal buffer of 64bytes in the CTMANAGER. This buffer holds the block copy data which is being transferred to or from SMEM at the AXI interface. In TX path, block copy data read from SMEM by AXI master is stored into the buffer first. AHB master then reads this block data from the buffer and sends to the corresponding XS_CAN IP. In

RX path, AHB master reads out data from the respective XS_CAN IP and stores in the buffer. AXI master sends this block data to SMEM.

1.3.3.8 AHB-APB Bridge

CTDMA supports an AHB to APB bridge to convert AHB to APB transactions for CTDMA_CREG access by the host. Note that the module gives ERROR response to read from write only addresses, write to read only addresses and access to reserved addresses of CREG. In case of a write, the transaction is ignored and in case of a read, the default data 0xCAFECAFE is given. The interrupt ARA_IRQ is raised for illegal write and read.

1.3.3.9 Block Copy Control Unit

Block copy control unit handles the all the block copy transfer requests from the connected XS_CAN IPs. Based on the direction of the request, SMEM read write transactions are initiated at AXI master interface and XS_CAN IP read write transfers are generated at AHB master interface. Refer[3] for details on block copy transfer.

1.3.3.10 CTMANAGER

The blocks that are used for Cut through Block copy functions are grouped under CTMANAGER submodule. AXI master interface, internal buffer, Block Copy Control Unit, and AHB master interface are inside CTMANAGER module.

1.3.3.11 Cut Through Signals

The XS_CAN IP provides the necessary signals needed to operate the CTDMA add-on module. The CTDMA add on module can used by the integrator to perform block copy transfer in Cut Through Mode.
The following signals are provided to support the block copy operation between CTDMA and XS_CAN IP.
1) Cut Through descriptor signals- Source and destination addresses from the XS_CAN MH registers *CTM_DESCRIPTOR*.
2) The direction of transfer is known from the Cut Through Block copy DMA request signals *CTM_TX_WREQ* and *CTM_TX_RREQ*.
3)CTM block copy error signal to indicate if the ongoing block copy is canceled in the XS_CAN IP.
4) *MH_EN* signal taken from **XS_CAN.MH_CTRL.MH_EN** field.
5) *CTM_EN* signal taken from **XS_CAN.MH_CFG.CTME** field.

1.3.3.12 Sideband Signals

PORT_EN_x: For each AHB master interface, there is an input static signal *PORT_EN_x* associated. If an XS_CAN IP is connected to one of the AHB master interfaces, then the corresponding *PORT_EN_x* signal must be set to 1 by the user. If host tries to access an XS_CAN IP for which PORT_EN is disabled, CTDMA ignores the transaction and raises ARA interrupt. Write transactions are ignored with ERROR response and read transactions are responded with 0xCAFECAFE read data and ERROR response. PORT_EN signals are static signals that need to be set initially before reset and must not be changed during run time. Any update to the *PORT_EN_x* should be followed by a hard reset. CTDMA samples the PORT_EN_x values from the input after reset is deasserted and stores the value into the register **CTDMA_STS2.PORT_EN.** This registered value is used by CTDMA for all checks and not the input port value. Hence any change to the PORT_EN_x signal after reset is deasserted, is ignored by the IP.

1.3.4 Safety Measures

NA

1.4 Cut Through Manager (CTMANAGER)

CTMANAGER block handles the operations related to block copy transfer from all connected XS_CAN IPs. The block copy control unit sub module in CTMANAGER polls through the block copy TX and RX requests from XS_CAN IPs starting from PORT0 to PORT3 (if enabled) in a round robin manner. Based on the request which is currently active, AXI and AHB transactions are initiated at the corresponding masters respectively. There are 4 main sub blocks in CTMANAGER.
i) Block Copy Control Unit
ii) AXI Master
iii) AHB Master
iv) 64byte FIFO

1.4.1 Block Diagram

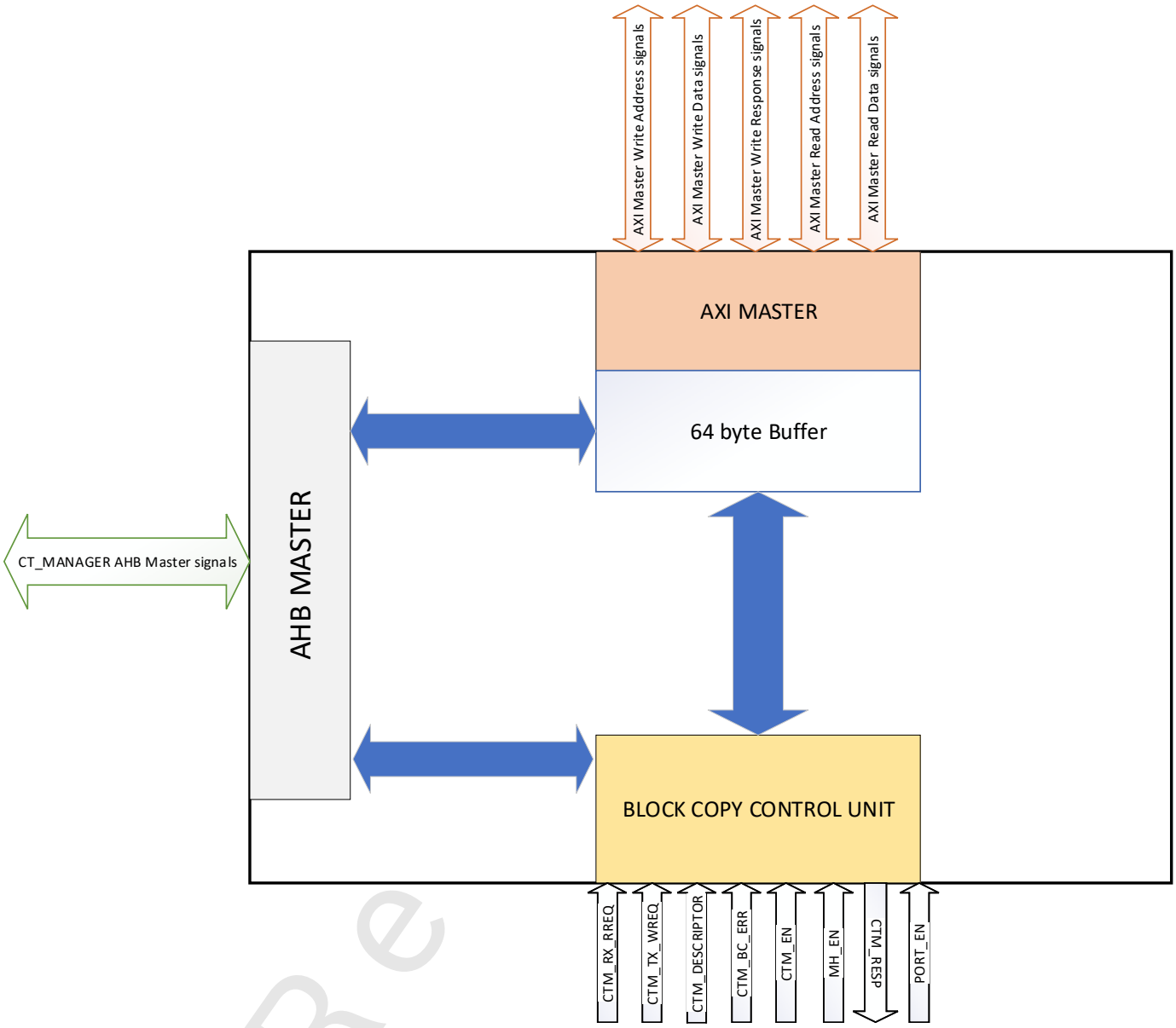

Figure 2. CT_MANAGER Block Diagram

1.4.2 Hardware Interface

Not Applicable

1.4.3 Software Interface

Not applicable

1.4.4 Functional Description

1.4.4.1 Block Copy Control Unit (BCCU)

The block copy control unit module monitors and controls all the block copy transfer requests from the connected XS_CAN IPs. Based on the block copy requests, AHB master transactions and AXI master transactions control signals are generated. The requests from the XS_CAN IPs are polled in a round robin manner starting from AHB MASTER PORT0 till port AHB MASTER PORT3. If PORT0 requests are served first (TX or RX), then PORT1 gets the slot, followed by PORT2 and PORT3 XS_CAN requests. In the next round, the XS_CAN request that just got served will not get a

chance until all other XS_CAN requests are served. If there are no active request from any other XS_CANs then the same XS_CAN request gets the chance.

Operation of block copy control unit is given below.

1. BCCU waits in the idle state after reset.
2. In idle state, during every clock edge, XS_CAN block copy signals are polled only if the corresponding PORT_EN signal is set. If **CTDMA_STS2.PORT_EN[0]**=1, then BCCU checks for CTM_EN_0 and MH_EN_0 signals. If either one is low, then CTM_RESP=ERROR is sent to the corresponding XS_CAN IP without initiating block copy transfer. If both MH_EN_0 and CTM_EN_0 are high, then *CTM_TX_WREQ_0* and *CTM_RX_RREQ_0* signals are polled.
3. If *CTM_TX_WREQ_0* is active, BCCU checks the value of *CTM_DESCRIPTOR_0*[31:0] given by XS_CAN_0. If this is not Cut Through Buffer start address, block copy transfer is not initiated by the CTMANAGER and *CTM_RESP_0* = ERROR is issued. If *CTM_DESCRIPTOR_0*[31:0] is the expected start address of the Cut Through Buffer, then BCCU issues control signals to AXI master module to perform AXI read from SMEM. The read address for starting the burst transfer is taken from *CTM_DESCRIPTOR_0*[63:32]. BCCU waits for AXI master to give back the acknowledgment after completion of the transfer along with a response. If the response is OK, then step 4 is performed, else if response is ERROR anytime while waiting for the ack, the same ERROR response is given at *CTM_RESP_0* to XS_CAN_0. This aborts the block copy transfer in BCCU and it goes back to idle state. Note that OK response from AXI Master is not given at *CTM_RESP_0* at this point. CTM_RESP_x is active for one host clock cycle.
4. After the acknowledgment is received from AXI master that read is completed successfully, BCCU issues control signal to AHB Master to perform write transaction to XS_CAN_0. The write address is also provided to AHB Master from *CTM_DESCRIPTOR_0*[31:0]. BCCU then waits for the ack and response from AHB Master. The response (OK or ERROR) is given back to *CTM_RESP_0*.
5. If *CTM_RX_RREQ_0* is active, the read address for starting the burst transfer is taken from *CTM_DESCRIPTOR_0*[63:32]. If this is not Cut Through Buffer start address, block copy transfer is not initiated by the AHB master and *CTM_RESP_0* = ERROR is issued. If *CTM_DESCRIPTOR_0*[63:32] is the expected start address of the Cut Through Buffer, BCCU issues control signals to AHB master module to perform AHB read from XS_CAN_0. BCCU waits for AHB master to give back the acknowledgment after completion of the transfer along with a response. If the response is OK, then step 6 is performed, else if response is ERROR anytime while waiting for the ack, the same ERROR response is given at *CTM_RESP_0* to XS_CAN0. This aborts the block copy transfer in BCCU and it goes back to idle state. Note that OK response from AHB Master is not given at *CTM_RESP_0* at this point.
6. After the acknowledgment is received from AHB master that read is completed successfully, BCCU issues control signal to AXI Master to perform write transaction to SMEM. The write address is also provided to AXI Master from *CTM_DESCRIPTOR_0*[31:0]. BCCU then waits for the ack and response from AXI Master. The response (OK or ERROR) is given back to *CTM_RESP_0* and goes to idle state
7. Steps 2 to 6 are repeated for the remaining enabled XS_CAN ports.If requests from other XS_CANs are not present, then the state machine moves to a delay state before starting the next round of polling. XS_CAN IP takes one clock to pull this request low after getting the response and the delay state is added to ensure that the request just got served is not read again as a new request.

1.4.4.2 AXI Master

The AXI Master sub module in CTMANAGER works on AMBA AXI4 protocol. Refer [6] for details on the protocol.
The master has 5 channels- Write address, write data, write response, read address and read data. The main features of the AXI master are mentioned below.

1. AXI master always issue burst transfer of type INCR for both read and write.i.e. *M_AXI_AWBURST*[1:0] and *M_AXI_ARBURST*[1:0] = 0b01
2. The Write and Read burst size is fixed at 4bytes. i.e. *M_AXI_AWSIZE*[2:0] and *M_AXI_ARSIZE*[2:0] = 0b010.
3. The length of each burst transfer, *M_AXI_AWLEN*[7:0] and *M_AXI_ARLEN*[7:0] is 0x0F. In case of 4KB address boundary violation, burst length is adjusted accordingly.
4. *M_AXI_AWID*[3:0] and *M_AXI_ARID*[3:0] are 0b0000 since only payload type data is read and written from SMEM and hence no need for different IDs.
5. Out of order transactions and outstanding transactions are not supported.

6. Exclusive accesses are not supported by CTDMA. *M_AXI_AWLOCK*=0b0 and *M_AXI_ARLOCK*=0b0 i.e. Normal access.
7. The memory device type requirement is non-buffer-able. *M_AXI_ARCACHE*[3:0] and *M_AXI_AWCACHE*[3:0] is 0b0000.
8. Quality of Service signaling feature is not used. *M_AXI_AWQOS*[3:0] and *M_AXI_ARQOS*[3:0] = 0b0000.
9. User signals are used to indicate the XS_CAN instance for which the block copy transfer is ongoing at the AXI master interface. For write operation, *M_AXI_AWUSER*[1:0], *M_AXI_WUSER*[1:0] are used.*M_AXI_ARUSER*[1:0] indicate the XS_CAN instance number which got selected for doing the block copy read from SMEM. *M_AXI_BUSER*[1:0] and *M_AXI_RUSER*[1:0] are not used.
10. Region decoding is not used. *M_AXI_AWREGION*[3:0] and *M_AXI_ARREGION*[3:0] are kept at default value 0b0000.

1.4.4.2.1 Working

AXI master has two independent channel state machines- one for the write channel and one for the read channel.
In the write channel FSM, AXI master waits for the write transfer request from BCCU. When the request is received, AXI master starts the write transfer of the data present in the 64byte FIFO to SMEM. The address and control signals are placed at the Write address channel. Data to be sent to SMEM is read from the 64byte FIFO word by word and issued in the data channel. The response from the slave at write response channel is taken and given to BCCU along with the acknowledgment signal indicating that the write transfer is completed. The supported responses at AXI are OKAY, SLVERR and DECERR. EXOKAY response is not supported.

In the read channel FSM, AXI master waits for the read transfer request from BCCU. When the request is received, AXI master issues the read address and control signals in the read address channel. Data read from SMEM is stored into the 64byte FIFO word by word. The response from the slave is taken and given to BCCU along with acknowledgment signal indicating that the read transfer is completed. Note that the master collects the read response for each of the beat and then gives the final response back to BCCU. If at least one word has given SLVERR or DECERR, then ERROR response is given to BCCU. Only if OKAY response is received for all the read words, OK is given back to BCCU.

4KB address boundary violations are handled by the AXI master for both read and write operations. In case of an address boundary violation, the burst of length 0x0F is split into two bursts and issued one after the other.

1.4.4.2.2 AXI Master Timeout

AXI Master timeout counter starts as soon as the request is placed on the read or write address channel (next clock after AWVALID and ARVALID is high). If the transaction is not completed by the time counter reaches the programmed timeout value, interrupt *AXI_TO* is generated. For write, timer is reset with BRESP and BVALID received for that transaction and for read, timer is reset with RLAST. RLAST should be asserted by the slave for the last data word in the transfer. If AXI master receives the last word without RLAST, then the timeout counter is reset. The timeout feature can be enabled/disabled by writing to the register field **CTDMA_SFTY_CFG.AXI_MST_TO_EN.** The timeout value and prescaler value can be programmed into **CTDMA_SFTY_CFG.AXI_MST_TO_VAL** and **CTDMA_SFTY_CFG.AXI_MST_PRESCALER**.

1.4.4.2.3 CTMANAGER DATA PARITY

Datapath parity checks are done in CTMANAGER for block copy data. For TX block copy requests from XS_CAN IPs, when data is read from SMEM, parity is calculated and inserted for each word (1 parity bit per byte). Parity appended data is then stored into the 64byte buffer. For RX block copy requests, parity is checked in AXI master before writing to SMEM. If address boundary violation is not there, then AXI transaction is completed even if parity error is detected in any of the data words. In case of address boundary violation, second burst is not initiated if parity error is detected for any word in the first burst.In case of parity mismatch, *DATA_PARITY_ERR* interrupt is triggered. Data parity can be enabled and disabled by writing to the register field **CTDMA_SFTY_CFG.PARITY_EN.**

1.4.4.3 AHB Master

AHB Master sub module in CTMANAGER works on AMBA2 AHB protocol. Refer [7] for details on protocol. Main features of AHB Master module is given below.

1. AHB Master always issues burst transfer of type INCR16. i.e *CT_MNGR_AHB_HBURST[2:0]* = 0b111.
2. AHB master issues only NONSEQ and SEQ type transfers. IDLE and BUSY are not used. i.e. *CT_MNGR_AHB_HTRANS[1:0]* = 0b10 / 0b11.

3. Transfer size is always 32bits. i.e. *CT_MNGR_AHB_HSIZE[2:0]* = 0b010.
4. Protection control is data access, user access, not bufferable and not cachable. *CT_MNGR_AHB_HPROT[3:0]*=0b0001.
5. Bus request (*CT_MNGR_AHB_HBUSREQ*) and grant (*CT_MNGR_AHB_HGRANT*) signals are used by AHB master to perform arbitration at the AHB arbiter module.
6. SPLIT transfers are not supported.
7. Early burst termination is not supported.

1.4.4.3.1 Working

AHB master sub module waits for the requests from BCCU in its idle state.

AHB Write
If BCCU raises write request to AHB master, the data present in the 64byte FIFO is written into the requested XS_CAN IP. AHB master first raises the HBUSREQ signal to the arbiter and waits for the grant. Once grant is received from AHB arbiter, in the next clock the address and control signals to perform the write are placed in the bus. This is the address phase. In the next clock cycle, the first data word is read from the 64byteFIFO and placed on the wdata bus along with the next address. Note that the write to XS_CAN IP in block copy transfer is always 64bytes data. Once the transfer is complete, ack and response is given to BCCU and the request to arbiter is deasserted.

AHB Read
If BCCU raises read request to AHB master, AHB master reads the data from the requested XS_CAN IPs and store into the 64byte FIFO. AHB master first raises the HBUSREQ signal to the arbiter and waits for the grant. Once grant is received from AHB arbiter, in the next clock the address and control signals to perform the read are placed on the bus. This is the address phase. In the next clock cycle, it starts to wait for the rdata and is written into 64byte FIFO. Note that the read from XS_CAN IP in block copy transfer is always 64bytes data. Once the transfer is complete, ack and response is given to BCCU.

1.4.4.3.2 AHB Master Timeout

AHB Master timeout counter starts as soon as the request is placed on the selected AHB master port. If the transaction is not completed by the time counter reaches the programmed timeout value, interrupt *AHB_TO* is generated. The timeout feature can be enabled/disabled by writing to the register field **CTDMA_SFTY_CFG.AHB_MST_TO_EN.** The timeout value and prescaler value can be programmed into **CTDMA_SFTY_CFG.AHB_MST_TO_VAL**. There is no prescaler associated with AHB Master timeout.

1.4.4.4 64 Byte Buffer

AXI master and AHB master in CTMANAGER share a common 64byte buffer to store the data used for read and write transfers. AXI master stores the read data from SMEM into this buffer which is then taken by AHB master to write to XS_CAN IPs. AHB master stores the read data from XS_CAN IPs into this buffer which is then used by AXI master to write to SMEM.

Buffer is implemented as a flipflop register with necessary control signals like write enable (wr_en), read enable (rd_en), data input (data_in), data output (data_out), full and empty.

1.4.4.5 Status Register updates

This section describes the status register fields updated by CTMANAGER.

The following status signals are updated by Block Copy Control Unit.

1. **CTDMA_STS2.BC_REQ** - Shows the values of all block copy requests from the XS_CAN IPs. BC_REQ[7]= CTM_TX_WREQ_3, BC_REQ[6]= CTM_RX_RREQ_3, BC_REQ[5]= CTM_TX_WREQ_2, BC_REQ[4]= CTM_RX_RREQ_2, BC_REQ[3]= CTM_TX_WREQ_1, BC_REQ[2]= CTM_RX_RREQ_1, C_REQ[1]= CTM_TX_WREQ_0, BC_REQ[0]= CTM_RX_RREQ_0.
2. **CTDMA_STS2.PORT_EN**- Shows the values of PORT Enable input signals. PORT_EN[3]= PORT_EN_3, PORT_EN[2]= PORT_EN_2, PORT_EN[1]= PORT_EN_1, PORT_EN[0]= PORT_EN_0.
3. **CTDMA_STS2.MH_EN-** Shows the values of MH enable input signals. MH_EN[3]= MH_EN_3, MH_EN[2]= MH_EN_2, MH_EN[1]= MH_EN_1, MH_EN[0]= MH_EN_0.

4. **CTDMA_STS2.CTM_EN-** Shows the values of CTM enable input signals. CTM_EN[3]= CTM_EN_3, CTM_EN[2]= CTM_EN_2, CTM_EN[1]= CTM_EN_1, CTM_EN[0]= CTM_EN_0.
5. **CTDMA_STS1.BC_ACTIVE**- Set to 1 if CTMANAGER is serving a block copy request from one of the XS_CAN IPs.
6. **CTDMA_STS1.BC_AXI_TO_ACTIVE** - Set to 1 in case of AXI Master timeout. Reset if response received and counter is reset.
7. **CTDMA_STS1.BC_AHB_TO_ACTIVE** - Set to 1 in case of AHB Master timeout. Reset if response received and counter is reset.
8. **CTDMA_STS1.BC_SMEM_ACC_TYPE** with the SMEM access type. 0- no access, 1-read access, 2-write access.
9. **CTDMA_STS1.BC_BURST_LEN** with the length of the ongoing block copy transfer.
10. **CTDMA_STS1.BC_TX_RX_REQ -** Updated with the type of block copy request which is being served currently.0- no request active,1 indicates CTM_TX_WREQ, 2 indicates CTM_RX_RREQ. The XS_CAN Instance number for which this request belongs to is known from the field **CTDMA_STS1.BC_XS_CAN_INST.**
11. **CTDMA_STS1.BC_XS_CAN_INST**- Indicates the XS_CAN instance number for which block copy is active.
12. **STS_FROZEN.STS_0_1_CTRL**- Set to 1when there is an AXI or AHB master timeout which indicates that the contents of registers **CTDMA_STS0** and **CTDMA_STS1** are not updated by the IP. **STS_0_1_CTRL.STS_FROZEN** field has to be reset by the host once the contents of **CTDMA_STS0** and **CTDMA_STS1** are read.
13. **CTDMA_STS0.SMEM_ADDR_BC** -Updated with the SMEM address (read or write) for which block copy transfer is being performed.

1.5 Host Access Control Unit (HACU)

1.5.1 Block diagram

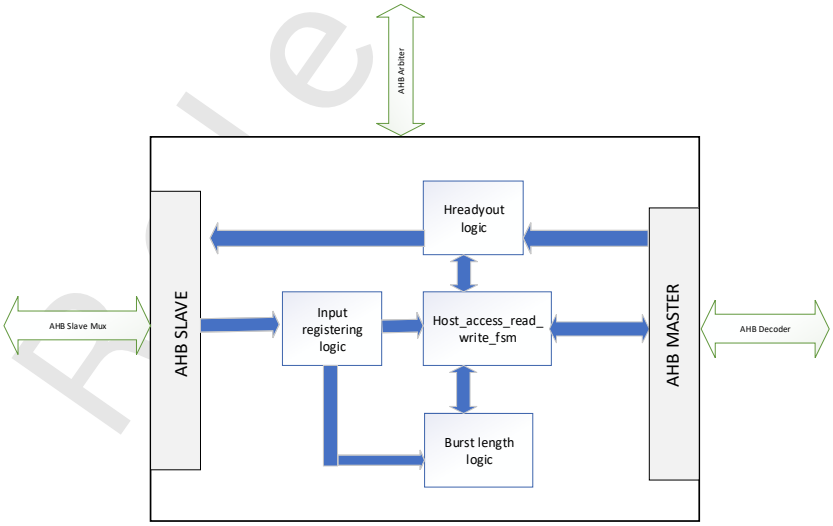

Figure 3. Host_Access_Control Block Diagram

1.5.2 Hardware Interface

Not Applicable

1.5.3 Software Interface

Not Applicable

1.5.4 Functional Description

Host Access Control Unit handles the host accesses to XS_CAN IPs. The AHB top mux decodes the host accesses directed to XS_CAN IPs and redirects to HACU. The inputs are registered once in HACU before sending to the

corresponding XS_CAN IP. HACU then raises request to the AHB ARBITER to get access to the targeted XS_CAN IP. Request to arbiter is kept active till the read or write transaction is finished. HACU gives default ready high to host S_AHB_HREADYOUT=1. After receiving the first address, ready is pulled low in the next clock cycle. Transaction is forwarded to the corresponding XS_CAN IP after grant is received. After receiving the grant, the outputs (read data, read response and ready) from the XS_CAN IP being accessed are forwarded to the host via HACU without registering.
In case of burst transfers from host, HACU inserts one busy state for every address except the start address in order to compensate for the registering of Host AHB input signals.
HACU also decides if throttling is applicable for the access issued by the host. If host accesses a register address in the XS_CAN IP, throttling is not performed for that access in the arbiter.

1.6 AHB Arbiter

1.6.1 Block diagram

Figure 4. AHB_ARBITER Block Diagram

1.6.2 Hardware Interface

Not Applicable

1.6.3 Software Interface

Not Applicable

1.6.4 Functional Description

AHB arbiter takes request signals from the CTMANAGER AHB master and the Host access control unit AHB master and grants access to one of them based on priority. CTMANAGER request has higher priority over Host access control unit. If both requests are active at the same time, CTMANAGER gets the grant. Arbiter takes a minimum of one clock cycle to assert the grant.

Master asserts the request signals till the entire transfer is completed. Grant signals from the arbiter stay high as long as there is request. If a master has to perform the next transaction request is asserted again and based on the priority, grant is given.

1.6.4.1 Throttling in AHB arbiter

AHB arbiter implements throttling feature in which no master is granted access for some clock cycles based on the configuration in the register **CTDMA_CFG.ARB_THROTTLE.** Throttling is applicable only for accesses to LMEM space of XS_CAN IPs. Accesses to registers in XS_CAN IPs are not throttled and are granted without any wait delay. For host

accesses, the decision whether throttling is needed or not is taken by HACU depending on the address received at the input. If the host access is in register address range (including the reserved addresses) of an XS_CAN IP, throttling is not performed.

Formula used for calculating wait cycles= (X*N) / (1-X) where X is the throttling percentage configured by the user, N is the number of clocks taken by the access, based on the look up table.

As soon as a master gets grant, the counter in arbiter is loaded with a predefined value from the table given below. Once the granted master finishes the access, counter starts counting down. No request is granted during this time when counter is active. The next request is granted only if the count value is 0 there by limiting the traffic to XS_CAN IPs connected to CTDMA. The status field **CTDMA_STS1.AHB_ARB_ACTIVE** is set to 1 if grant is given to one of the masters. The status field **CTDMA_STS1.AHB_SRC** is updated once grant is given and indicates the current master which got granted. Value 0-Host,1-CTMANAGER.

Table 18 indicates the number of clocks during which LMEM is occupied by one XS_CAN to perform the different types of accesses and Table 19 below shows the number of wait cycles based on the configuration in the register **CTDMA_CFG.ARB_THROTTLE** and the type of access granted by the arbiter.

Table 18. Arbiter Clock Cycles

Access Type	Clock Cycles (N)
Single Write	1
INCR4 Write	16
INCR8 Write	36
INCR16 Write	76
Single Read	2
INCR4 Read	14
INCR8 Read	30
INCR16 Read	62

Table 19. Throttling configuration (page 1 of 2)

Access Type	Throttling configuration				
	20	**33**	**50**	**66**	**80**
Single Write	1	1	1	2	4
INCR4 Write	4	8	16	32	64
INCR8 Write	9	18	36	70	144
INCR16 Write	19	38	76	148	304

Table 19. Throttling configuration (page 2 of 2)

Access Type	Throttling configuration				
Single Read	1	1	2	4	8
INCR4 Read	4	7	14	28	56
INCR8 Read	8	15	30	59	120
INCR16 Read	16	31	62	121	248

1.7 AHB Control Multiplexer and Decoder

1.7.1 Block diagram

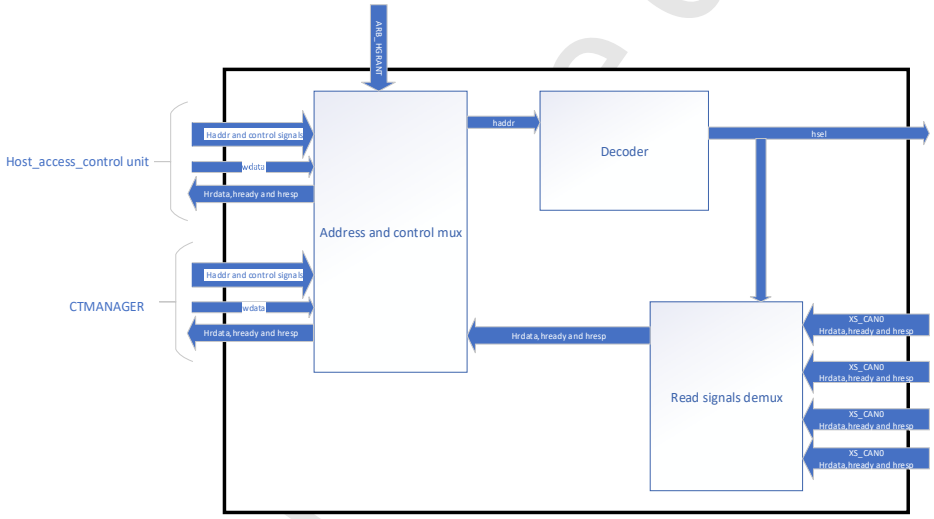

Figure 5. AHB Control Multiplexer and Decoder Block Diagram

1.7.2 Hardware Interface

Not Applicable

1.7.3 Software Interface

Not Applicable

1.7.4 Functional Description

The address and control multiplexer in this module handles the selection of the AHB master signals to be forwarded to the slave from the selected master based on the arbiter grant signal. The decoder generates the select line hselx for the slave based on the address issued by the granted master. The read data demultiplexer block in this module redirects the signals from the selected slave (AHB read data, read response and ready) back to the granted master.
Note that hsel is generated only if the status bit CTDMA_STS2.PORT_EN corresponding to that XS_CAN is enabled. If host is trying to communicate to an XS_CAN instance for which PORT_EN is disabled, then the behavior is such a way that write transaction is ignored with ERROR response and read access is responded with default read data 'hCAFECAFE with ERROR response. S_AHB_HREADYOUT will be high in both the cases. The status register field **CTDMA_STS1.AHB_XSCAN_INST** is updated with the XS_CAN instance number for which hsel is generated.

Data Parity Check for CTMANAGER TX requests and address parity checks for host access path are done in this module. In case of data parity check, it is performed for each of AHB master write data word before sending to XS_CAN IPs. In case of parity mismatch, AHB master gets a notification which in turn gives and ERROR response to block copy control

unit. The ongoing write burst is still completed at AHB master and is not terminated. *DATA_PARITY_ERR* IRQ is raised if there is a mismatch.

For address parity check in host access path, before sending the address to XS_CAN IP, parity is checked. In case of a mismatch, action is taken based on the access type. Refer Section 1.8 Error and Exception Handling for details. The status register fields **CTDMA_STS3.AP_ERR_AHB_HADDR** and **CTDMA_STS3.AP_ERR_RW** are updated during parity mismatch. In case of address parity mismatch for write address, *WR_ADDR_PARITY_ERR* IRQ is raised and for read address parity mismatch *RD_ADDR_PARITY_ERR* IRQ is raised.

1.8 Error and Exception Handling in CTDMA

This section describes the different error scenarios and the actions taken by CTDMA for the same.Note that the behavior of the IP when not configured properly by the host cannot be defined. It is up to the host to define proper values into the CREG to ensure correct working of the IP. Such error cases are not described here.

1.8.1 Timeout at CTMANAGER AXI Master interface

This section explains the causes and actions of CTDMA in case of a timeout at CTMANAGER AXI interface.

Cause: Delayed or no slave response from SMEM during AXI read and write transfer

AXI master at CTMANAGER issues read and write transfers to SMEM during block copy transfer phase of the BCU. The transfer is complete when SMEM sends the response back. As soon as the request is placed on the address bus, the timeout counter at AXI master starts counting up. If response is not received before the counter reaches the maximum value, interrupt is raised. For write channel, counter is reset with the BVALID and BRESP for that transaction and for read channel, counter is reset with RLAST.

Action: When AXI master gets a timeout, it continues to wait in the same state till host rests the CTDMA or the response is received. If a response is received before reset, AXI master continues to be operational. The timeout mechanism is designed to trigger at most once per transaction. Upon expiration of the timeout counter, a timeout interrupt is generated, and the counter is not restarted for the remainder of that transaction.
As a result, if a timeout occurs during any intermediate phase of the transaction (e.g., while waiting for AWREADY during a write), subsequent delays in later phases (e.g., delayed BVALID response) will not generate additional timeout events for the same transaction.

Interrupts and Flags raised:
IRQ_AXI_TO interrupt is raised to inform that a timeout has occurred.

1.8.2 Timeout at CTMANAGER AHB Master interface

This section explains the causes and actions of CTDMA in case of a timeout at CTMANAGER AHB interface.

Cause: Delayed or no slave response from XS_CAN IP during AHB block copy transfer.

Block copy transfer requests (read and write) to XS_CAN IPs issued at CTMANAGER AHB interface waits for response from the slave for completing the transfer. Once the CTMANAGER AHB master gets the grant, AHB timeout counter is loaded and starts counting down. If the response is not received from the corresponding XS_CAN IP before the timeout counter reaches 0, interrupt is raised.

Action: When AHB master gets a timeout, it continues to wait in the same state till host rests the CTDMA. Even if a response is received before reset, it is ignored by the master.

Interrupts and Flags raised:
IRQ_AHB_TO interrupt is raised to inform that a timeout has occurred.

1.8.3 Datapath Parity Error in CTMANAGER access path

This section explains the causes and actions of CTDMA in case of data path parity error in CTMANAGER access path.

Cause: i) Read data received from SMEM at CTMANAGER AXI interface is stored into the 64byte buffer after inserting parity bit per byte of each word. The appended parity is checked at AHB Control Multiplexer and Decoder before sending to the respective XS_CAN IP. Internal corruption of the data in this data path results in data path parity error.
ii) Parity is inserted for read data (1bit per byte of the word), received from XS_CAN IP at AHB Control Multiplexer and Decoder. This is stored into the 64byte buffer with the appended parity. Before issuing write to SMEM, parity is again calculated and checked. Internal corruption of the data in this data path results in data path parity error.

Action:
CTM_RESP= ERROR(0x02) is given to the XS_CAN IP for which the block copy transfer was ongoing.

Interrupts and Flags raised: Interrupt DATA_PARITY_ERR is generated for data path parity mismatch.

1.8.4 Address Parity Error in Host access path

This section explains the causes and actions of CTDMA if there is address parity mismatch in the host access path.

Cause: Parity mismatch in the read and write addresses issued by the host to XS_CAN IPs. CTDMA CREG accesses are not considered for address parity calculation.

Action: When CTDMA detects address parity error in any of the addresses issued by the host, following steps are taken:
i) If address parity error is detected for a single read or write (all accesses), access is not forwarded to XS_CAN IP.i.e. HSLEx is not generated. Error response is given back to host and 'hCAFECAFE as read data for read accesses.
ii) If address parity error is detected for the first address of a burst transfer (including Virtual buffer accesses and LMEM Transparent access), the entire burst is not forwarded to XS_CAN IP .i.e. HSLEx is not generated. Error response is given back to host and 'hCAFECAFE as read data for read accesses
iii) If address parity error is detected for any other address other than the start address of a burst for a virtual buffer accesses, the corrupted address is replaced by reserved address=0x3FFFC. XS_CAN IP response is forwarded to host in this case. CTDMA does not generate ERROR response for this scenario.
iv) In case of LMEM transparent accesses, address parity is not checked for subsequent burst addresses after the start address. The corrupted address can go to the selected XS_CAN in this case.

Interrupts and Flags raised: Interrupts RD_ADDR_PARITY_ERR and WR_ADDR_PARITY_ERR is generated for read address parity mismatch and write address parity mismatch respectively. The address issued by the host that resulted in address parity mismatch is stored in the register **CTDMA_STS_3.AP_ERR_AHB_HADDR.** The direction of transfer is indicated by the status bit CTDMA_STS_3.AP_ERR_RW.

1.8.5 Error response at AXI Master interface during block copy transfer to SMEM

This section explains the causes and actions of CTDMA in case of error response received at CTMANAGER AXI Master from SMEM input during a block copy transfer.

Cause: AXI read and write transactions to SMEM is completed when slave sends the response. If the response is DECERR or SLVERR, then this error scenario occurs.

Action: If the error is received during AXI transfer to SMEM , the block copy transfer is dropped and this information is sent to the XS_CAN IP via CTM_RESP signal as ERROR response.

Interrupts and Flags raised:
The interrupt IRQ_AXI_SLVERR is raised for both write and read error responses.

1.8.6 Error response from XS_CAN IPs at AHB Master ports

This section explains the causes and actions of CTDMA in case of error response received at AHB master port from XS_CAN IP.

Cause: XS_CAN IP gives back error response at AHB master port. The access can be from host or CTMANAGER

Action: If the error response at the AHB masters is received during host access to XS_CAN IPs, the same is forwarded to the host at the top AHB Slave interface.
If the error response at the AHB masters is received during a block copy transfer by CTMANAGER, the burst is finished with CTM_RESP=Error. Further block copy at AXI Master will not be initiated by Block Copy Control Unit and hence data path parity check in AXI master is not exercised.

Interrupts and Flags raised:
No IR or flag raised.

1.8.7 Access to disabled XS_CAN IPs by host

This section explains the cause and actions of CTDMA if host tries to access a disabled port at AHB Master.

Cause: 1. Host trying to access an XS_CAN IP address for which PORT_EN_x is disabled.
2. CTMANAGER block gets request from an XS_CAN IP for which PORT_EN_x is disabled.

Action: 1) Write transaction is ignored with ERROR response and read operation is responded with default data 0xCAFECAFE and ERROR response
2. CTMANAGER does not poll the request if PORT_EN is disabled.

Interrupts and Flags raised:
The interrupt *IRQ_HOST_ARA* is raised for host access to disabled PORT.

1.8.8 Illegal register accesses

This section explains the cause and actions of CTDMA in case if host accesses reserved addresses.

Cause: Host must not access any reserved addresses in CTDMA CREG. Any access to a reserved address (single or burst type) results in this error scenario. Reserved address range is 0x200B10 and 0x3FFFFC. Read to write only registers and write to read only registers also result in this error type.

Action: Write transaction is ignored with ERROR response and read operation from a reserved address is responded with default data 0xCAFECAFE and ERROR response.

Interrupts and Flags raised:
The interrupt *IRQ_HOST_ARA* is raised.

1.8.9 CTM_BC_ERR signal raised by XS_CAN IP

XS_CAN IP can raise CTM_BC_ERR signal for the following cases
Cause 1:
LMEM Uncorrected error during RX block copy by CTDMA.

Action: CTDMA finishes the ongoing AHB RX block copy transfer and polls for the next request. Response at the signal CTM_RESP is not given to the corresponding XS_CAN since the block copy transfer is aborted by XS_CAN itself. Further AXI write transaction to SMEM is not initiated.

Interrupts and Flags raised: NIL

Cause 2:
Non linear address issued to cut through buffer during TX or RX block copy.

Action: CTDMA finishes the ongoing AHB or AXI block copy transfer and polls for the next request. Response at the signal CTM_RESP is not given to the corresponding XS_CAN since the block copy transfer is aborted by XS_CAN itself. In case of RX, further AXI write transaction to SMEM is not initiated.

Interrupts and Flags raised: NIL

Cause 3:
Block copy timeout in XS_CAN resulting in TX Underrun or RX Overflow.

Action: CTDMA finishes the ongoing AHB or AXI block copy transfer and polls for the next request.In case of RX, if CTM_BC_ERR is received after AHB read is initiated, CTDMA finishes the AHB transactions and AXI write is not initiated. In case of TX, if CTM_BC_ERR is received after AXI read is initiated, CTDMA finishes the AXI transactions and AHB write is not initiated. In any case, CTM_RESP is not given to the corresponding XS_CAN since the block copy transfer is aborted by XS_CAN itself.

Interrupts and Flags raised: NIL

1.8.10 CTM_EN_x/MH_EN_x from XS_CAN_x going low during its block copy transfer.

Cause: Input signals CTM_EN_x and MH_EN_x from XS_CAN_x IP goes to 0 after raising request for a TX or RX Block copy transfer**.**

Case1:**TX Block Copy**
Action: If the CTM_EN_x and MH_EN_x signals go low during AXI read from SMEM, CTDMA waits for the AXI transfer to finish and gives error response CTM_RESP_x=Error. Further AHB write is not initiated.
If the CTM_EN_x and MH_EN_x signals go low during AHB write to LMEM of XS_CAN_x, CTDMA waits for the AHB transfer to finish or an AHB timeout which ever comes first and gives error response CTM_RESP_x=Error.
If the CTM_EN_x and MH_EN_x signals go low after both read at AXI and write at AHB are completed, then CTM_RESP=Error is given to XS_CAN_IP_x.

Case2:**RX Block Copy**
Action: If the CTM_EN_x and MH_EN_x signals go low during AXI write to SMEM, CTDMA waits for the AXI transfer to finish and gives error response CTM_RESP_x=Error. Further AHB read is not initiated at AHB master interface.
If the CTM_EN_x and MH_EN_x signals go low during AHB read from LMEM of XS_CAN_x, CTDMA waits for the AHB transfer to finish and gives error response CTM_RESP_x=Error. This may also result in an AHB timeout at the AHB master interface.
If the CTM_EN_x and MH_EN_x signals go low after both read at AHB and write at AXI are completed, then CTM_RESP_x=Error is given to XS_CAN_x.

If PORT_EN_x is high and the CTM_EN_x and MH_EN_x signals are low while polling XS_CAN_x request, then this request is not considered, CTM_RESP_x=ERROR is given to XS_CAN_x and the next XS_CAN's request is polled.

Interrupts and Flags raised: NIL

1.8.11 M_AXI_RLAST signal not received with last word at AXI Master

Cause: Protocol violation by slave resulting in RLAST not received along with the last word in AXI read channel.

Action: CTDMA gives CTM_RESP=ERROR to the corresponding XS_CAN IP's TX request and AHB write is not initiated. AXI timeout counter is also reset in this case.

Interrupts and Flags raised: NIL

1.8.12 M_AXI_BID/M_AXI_RID signals do not match to M_AXI_AWID/M_AXI_ARID at AXI Master

Cause: Mismatch in ID at write response channel or read channel compared to the ID sent out at Write address and Read address channels.

Action: CTDMA gives CTM_RESP=ERROR to the corresponding XS_CAN IP's TX/RX request and AHB write is not initiated in case of TX requests.

Interrupts and Flags raised: NIL

1.9 Clock Domains and Resets

1.9.1 Clock Domains

The CTDMA module has only one clock domain which is same as the HOST CLOCK_AON domain of the connected XS_CAN IPs. The clock HOST_CLK_AON is the clock input which is always on and not turned off. Only the rising edge of the clock is considered for operation.

These are required clock frequencies for the clock domain:
HOST: min. 40MHz, typ. 160MHz, max. 320MHz
Customers may also use other frequencies within this range.
Note that the frequency of the CTDMA must be same as the HOST CLK frequency of the connected XS_CAN IP.

1.9.1.1 Behavior While Not Clocked

Accessing the top AHB_SLAVE interface when the *HOST_CLK_AON* is inactive will result in a hang-up.

1.9.2 Resets

CTDMA supports one input reset *HOST_RESET_N* which is asserted (set to low) asynchronously and de-asserted (set to high) synchronously to *HOST_CLK_AON* clock. This reset is same as the *HOST_RESET_N* connected to all XS_CAN IPs and is synchronized for de-assertion with *HOST_CLK_AON* outside the CTDMA module.
The following figure shows the recommended reset circuit.

1.9.2.1 Behavior While Reset Active

S_AHB_HREADYOUT is set to high by CTDMA during reset. Access to the AHB_SLAVE interface while reset (*HOST_RESET_N*=0) will be ignored.

1.10 Trace and Debug

1.10.1 Hardware Debug Port

The 16 bit HDP bus provides some visibility to internal signals to debug the CTDMA. By default, there is not activity on the HDP bus.

To enable the toggling of the HW signal on the HDP bus, set the **DEBUG_TEST_CTRL.HDP_EN** bit to 1 and the selected set to be monitored on the HDP using the **DEBUG_TEST_CTRL.HDP_SEL[1:0]**.

To disable the Hardware Debug Port monitoring, do the previous set with **DEBUG_TEST_CTRL.HDP_EN** bit to 0.
Up to 2 sets can be defined using the **DEBUG_TEST_CTRL.HDP_SEL[1:0]** bit field. When the value is set to n the set n is selected in the HDP bus.

Here below are the description of sets available on the CTDMA HDP bus.

HDP [15:0]	HDP_SEL = 0	HDP_SEL = 1
15	HOST_CLK_AON	HOST_CLK_AON
14	AHB_SRC_S	IRQ_AXI_SLVERR_S
13	AHB_XSCAN_INST_S[1]	IRQ_WR_ADDR_PARITY_ERR_S
12	AHB_XSCAN_INST_S[0]	IRQ_RD_ADDR_PARITY_ERR_S
11	AHB_ARB_ACTIVE_S	DATA_PAR_ERR_IRQ
10	BC_AHB_TO_ACTIVE_S	AHB_TIMEOUT_ERR_S

HDP [15:0]	HDP_SEL = 0	HDP_SEL = 1
9	BC_AXI_TO_ACTIVE_S	AXI_TIMEOUT_ERR_S
8	BC_XSCAN_INST_S[1]	IRQ_ARA_S
7	BC_XSCAN_INST_S[0]	CTM_TX_WREQ_S(3)
6	BC_SMEM_ACC_TYPE_S[1]	CTM_RX_RREQ_S(3)
5	BC_SMEM_ACC_TYPE_S[0]	CTM_TX_WREQ_S(2)
4	BC_ACTIVE_S	CTM_RX_RREQ_S(2)
3	M_AHB_3_HWRITE_S	CTM_TX_WREQ_S(1)
2	M_AHB_2_HWRITE_S	CTM_RX_RREQ_S(1)
1	M_AHB_1_HWRITE_S	CTM_TX_WREQ_S(0)
0	M_AHB_0_HWRITE_S	CTM_RX_RREQ_S(0)

1.11 Timeout Settings

Two timeout counters are provided in CTDMA
1) AXI Master Interface timeout counter for SMEM block copy transfer.

A prescaler is used to set the reference clock for AXI timeout counter, see **CTDMA_SFTY_CFG.PRESCALER** register. According to the value defined in the **CTDMA_SFTY_CFG.PRESCALER**register, the timeout counter reference clock is either CLK/64, CLK/256, CLK/1024 or CLK/4096.

In order to set the timeout value, use the following registers:
• Set the value in the **CTDMA_SFTY_CFG.AXI_MST_TO _VAL** register for the interface and enable the counter by setting **CTDMA_SFTY_CFG.AXI_MST_TO _EN.**

The table below provides the possible range for different timeout (according to the CLK clock frequency and clock ratio).

HOST CLK	PRESCALER		Timeout Step (us)	LMEM Interface Timeout range (us)	
320/160/80/40 MHz	Value	CLK Ratio		Max	Min
				255	0
40	0	64	1.6	0	0
	1	256	6.4	0	0
	2	1024	25.6	0	0
	3	4096	102.4	0	0
80	0	64	0.8	0	0
	1	256	3.2	0	0
	2	1024	12.8	0	0
	3	4096	51.2	0	0

HOST CLK	PRESCALER		Timeout Step (us)	LMEM Interface Timeout range (us)	
160	0	64	0.4	0	0
	1	256	1.6	0	0
	2	1024	6.4	0	0
	3	4096	25.6	0	0
320	0	64	0.2	0	0
	1	256	0.8	0	0
	2	1024	3.2	0	0
	3	4096	12.8	0	0

Timeout Calculation

Timeout can get triggered anytime between (Timeout_val-1)*clock ratio to (Timeout_val) * clock ratio, measured in host clock cycles. This is because the divided clock gets generated as soon as IP is out of reset and the counter starts counting at the posedge of the divided clock whenever it comes after the request is raised on AXI address channel.

AXI Timeout value (us)= (1/(HOST_CLK (MHz)) * (ratio defined in **CTDMA_SFTY_CFG**.PRESCALER) * **CTDMA_SFTY_CFG.AXI_MST_TO _VAL**.

2) AHB Master Interface timeout counter for XS_CAN IP block copy transfer

AHB master in CTDMA timeout is based on the value programmed into the register **CTDMA_SFTY_CFG.AHB_MST_TO _VAL.** There is no prescaler value for AHB timeout counter. The value configured into this register **CTDMA_SFTY_CFG.AHB_MST_TO _VAL** is taken as is if **CTDMA_SFTY_CFG.AXI_MST_TO _EN** bit is set**.**

1.12 Detailed Design Information

1.12.1 Port Description

Table 20. Port List (page 1 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
HOST_RESET_N	1	I	Reset	High	HOST_CLK_AON
HOST_CLK_AON	1	I	Clock	High	HOST_CLK_AON
AHB Master for XS_CAN0					
M_AHB_0_HADDR	19	O	AHB Address	na	HOST_CLK_AON
M_AHB_0_HTRANS	2	O	AHB Transfer Type	na	HOST_CLK_AON
M_AHB_0_HWRITE	1	O	AHB Read or Write bit	High	HOST_CLK_AON
M_AHB_0_HSIZE	3	O	AHB Burst Size	na	HOST_CLK_AON
M_AHB_0_HBURST	3	O	AHB Burst Type	na	HOST_CLK_AON
M_AHB_0_HPROT	4	O	AHB Protection type	na	HOST_CLK_AON

Table 20. Port List (page 2 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
M_AHB_0_HMASTLOCK	1	O	AHB Lock	High	HOST_CLK_AON
M_AHB_0_HWDATA	32	O	AHB Write data	na	HOST_CLK_AON
M_AHB_0_HRDATA	32	I	AHB Read data	na	HOST_CLK_AON
M_AHB_0_HRESP	2	I	AHB Response	na	HOST_CLK_AON
M_AHB_0_HREADYOUT	1	I	AHB Ready	High	HOST_CLK_AON
M_AHB_0_HSEL	1	O	AHB Select	High	HOST_CLK_AON
AHB Master for XS_CAN1					
M_AHB_1_HADDR	19	O	AHB Address	na	HOST_CLK_AON
M_AHB_1_HTRANS	2	O	AHB Transfer Type	na	HOST_CLK_AON
M_AHB_1_HWRITE	1	O	AHB Read or Write bit	High	HOST_CLK_AON
M_AHB_1_HSIZE	3	O	AHB Burst Size	na	HOST_CLK_AON
M_AHB_1_HBURST	3	O	AHB Burst Type	na	HOST_CLK_AON
M_AHB_1_HPROT	4	O	AHB Protection type	na	HOST_CLK_AON
M_AHB_1_HMASTLOCK	1	O	AHB Lock	High	HOST_CLK_AON
M_AHB_1_HWDATA	32	O	AHB Write data	na	HOST_CLK_AON
M_AHB_1_HRDATA	32	I	AHB Read data	na	HOST_CLK_AON
M_AHB_1_HRESP	2	I	AHB Response	na	HOST_CLK_AON
M_AHB_1_HREADYOUT	1	I	AHB Ready	High	HOST_CLK_AON
M_AHB_1_HSEL	1	O	AHB Select	High	HOST_CLK_AON
AHB Master for XS_CAN2					
M_AHB_2_HADDR	19	O	AHB Address	na	HOST_CLK_AON
M_AHB_2_HTRANS	2	O	AHB Transfer Type	na	HOST_CLK_AON
M_AHB_2_HWRITE	1	O	AHB Read or Write bit	High	HOST_CLK_AON
M_AHB_2_HSIZE	3	O	AHB Burst Size	na	HOST_CLK_AON
M_AHB_2_HBURST	3	O	AHB Burst Type	na	HOST_CLK_AON
M_AHB_2_HPROT	4	O	AHB Protection type	na	HOST_CLK_AON

Table 20. Port List (page 3 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
M_AHB_2_HMASTLOCK	1	O	AHB Lock	High	HOST_CLK_AON
M_AHB_2_HWDATA	32	O	AHB Write data	na	HOST_CLK_AON
M_AHB_2_HRDATA	32	I	AHB Read data	na	HOST_CLK_AON
M_AHB_2_HRESP	2	I	AHB Response	na	HOST_CLK_AON
M_AHB_2_HREADYOUT	1	I	AHB Ready	High	HOST_CLK_AON
M_AHB_2_HSEL	1	O	AHB Select	High	HOST_CLK_AON
AHB Master for XS_CAN3					
M_AHB_3_HADDR	19	O	AHB Address	na	HOST_CLK_AON
M_AHB_3_HTRANS	2	O	AHB Transfer Type	na	HOST_CLK_AON
M_AHB_3_HWRITE	1	O	AHB Read or Write bit	High	HOST_CLK_AON
M_AHB_3_HSIZE	3	O	AHB Burst Size	na	HOST_CLK_AON
M_AHB_3_HBURST	3	O	AHB Burst Type	na	HOST_CLK_AON
M_AHB_3_HPROT	4	O	AHB Protection type	na	HOST_CLK_AON
M_AHB_3_HMASTLOCK	1	O	AHB Lock	High	HOST_CLK_AON
M_AHB_3_HWDATA	32	O	AHB Write data	na	HOST_CLK_AON
M_AHB_3_HRDATA	32	I	AHB Read data	na	HOST_CLK_AON
M_AHB_3_HRESP	2	I	AHB Response	na	HOST_CLK_AON
M_AHB_3_HREADYOUT	1	I	AHB Ready	High	HOST_CLK_AON
M_AHB_3_HSEL	1	O	AHB Select	High	HOST_CLK_AON
AXI Master connected to NOC					
AXI Master Write Address Channel					
M_AXI_AWID	1	O	Write address ID	High	HOST_CLK_AON
M_AXI_AWADDR	32	O	Write address	na	HOST_CLK_AON
M_AXI_AWLEN	8	O	Burst Length	na	HOST_CLK_AON
M_AXI_AWSIZE	3	O	Burst size	na	HOST_CLK_AON
M_AXI_AWBURST	2	O	Burst type	na	HOST_CLK_AON
M_AXI_AWLOCK	1	O	Lock type	High	HOST_CLK_AON

Table 20. Port List (page 4 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
M_AXI_AWCACHE	4	O	Memory type	na	HOST_CLK_AON
M_AXI_AWPROT	3	O	Write protection type	na	HOST_CLK_AON
M_AXI_AWREGION	4	O	Write region	na	HOST_CLK_AON
M_AXI_AWQOS	4	O	QoS value	na	HOST_CLK_AON
M_AXI_AWUSER	2	O	Write address channel User signals	na	HOST_CLK_AON
M_AXI_AWVALID	1	O	Write address valid	High	HOST_CLK_AON
M_AXI_AWREADY	1	I	Write address ready	High	HOST_CLK_AON
AXI Master Write Data Channel					
M_AXI_WDATA	32	O	write data	na	HOST_CLK_AON
M_AX_WSTRB	4	O	write data byte strobe	na	HOST_CLK_AON
M_AXI_WLAST	1	O	Write data last	High	HOST_CLK_AON
M_AXI_WUSER	2	O	Write data channel User signals	na	HOST_CLK_AON
M_AXI_WVALID	1	O	write data valid	High	HOST_CLK_AON
M_AXI_WREADY	1	I	write data ready	High	HOST_CLK_AON
AXI Master Write Response Channel					
M_AXI_BID	1	I	Write response ID	High	HOST_CLK_AON
M_AXI_BRESP	2	I	Response data valid	na	HOST_CLK_AON
M_AXI_BUSER	2	I	Write response channel User signals	na	HOST_CLK_AON
M_AXI_BVALID	1	I	Response data valid	High	HOST_CLK_AON
M_AXI_BREADY	1	O	Response data ready	High	HOST_CLK_AON
AXI Master Read Address Channel					
M_AXI_ARID	1	O	Read address ID	High	HOST_CLK_AON
M_AXI_ARADDR	32	O	Read address	na	HOST_CLK_AON
M_AXI_ARLEN	8	O	Burst length	na	HOST_CLK_AON
M_AXI_ARSIZE	3	O	Burst size	na	HOST_CLK_AON
M_AXI_ARBURST	2	O	Burst type	na	HOST_CLK_AON
M_AXI_ARLOCK	1	O	Lock type	High	HOST_CLK_AON

Table 20. Port List (page 5 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
M_AXI_ARCACHE	4	O	Memory type	na	HOST_CLK_AON
M_AXI_ARPROT	3	O	Read protection type	na	HOST_CLK_AON
M_AXI_ARREGION	4	O	Read region	na	HOST_CLK_AON
M_AXI_ARQOS	4	O	QoS value	na	HOST_CLK_AON
M_AXI_ARUSER	2	O	Read address channel User signals	na	HOST_CLK_AON
M_AXI_ARVALID	1	O	Read address valid	High	HOST_CLK_AON
M_AXI_ARREADY	1	I	Read address ready	High	HOST_CLK_AON
AXI Master Read Data Channel					
M_AXI_RID	1	I	Read data ID	High	HOST_CLK_AON
M_AXI_RDATA	32	I	Read data	na	HOST_CLK_AON
M_AXI_RRESP	2	I	Read data response	na	HOST_CLK_AON
M_AXI_RLAST	1	I	Read data last	High	HOST_CLK_AON
M_AXI_RUSER	2	I	Read data channel User signals	na	HOST_CLK_AON
M_AXI_RVALID	1	I	Read data valid	High	HOST_CLK_AON
M_AXI_RREADY	1	O	Read data ready	High	HOST_CLK_AON
AHB Slave					
S_AHB_HADDR	22	I	AHB address	na	HOST_CLK_AON
S_AHB_HTRANS	2	I	AHB Transfer type	na	HOST_CLK_AON
S_AHB_HWRITE	1	I	AHB read or write bit	High	HOST_CLK_AON
S_AHB_HSIZE	3	I	AHB burst size	na	HOST_CLK_AON
S_AHB_HBURST	3	I	AHB burst type	na	HOST_CLK_AON
S_AHB_HPROT	4	I	AHB protection type	na	HOST_CLK_AON
S_AHB_HMASTLOCK	1	I	AHB master lock	High	HOST_CLK_AON
S_AHB_HWDATA	32	I	AHB write data	na	HOST_CLK_AON
S_AHB_HRDATA	32	O	AHB read data	na	HOST_CLK_AON
S_AHB_HRESP	2	O	AHB response	na	HOST_CLK_AON

Table 20. Port List (page 6 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
S_AHB_HSELX	1	I	AHB select	High	HOST_CLK_AON
S_AHB_HREADYOUT	1	O	AHB ready	High	HOST_CLK_AON
CTM ports					
			CTM Read Requests from XS_CANs		
CTM_RX_RREQ_0	1	I	RX Cut-Through Buffer read request (XS_CAN0)	High	HOST_CLK_AON
CTM_RX_RREQ_1	1	I	RX Cut-Through Buffer read request (XS_CAN1)	High	HOST_CLK_AON
CTM_RX_RREQ_2	1	I	RX Cut-Through Buffer read request (XS_CAN2)	High	HOST_CLK_AON
CTM_RX_RREQ_3	1	I	RX Cut-Through Buffer read request (XS_CAN3)	High	HOST_CLK_AON
CTM Write Requests from XS_CANs					
CTM_TX_WREQ_0	1	I	TX Cut-Through Buffer write request (XS_CAN0)	High	HOST_CLK_AON
CTM_TX_WREQ_1	1	I	TX Cut-Through Buffer write request (XS_CAN1)	High	HOST_CLK_AON
CTM_TX_WREQ_2	1	I	TX Cut-Through Buffer write request (XS_CAN2)	High	HOST_CLK_AON
CTM_TX_WREQ_3	1	I	TX Cut-Through Buffer write request (XS_CAN3)	High	HOST_CLK_AON
CTM Responses for XS_CANs					
CTM_RESP_0	2	O	Cut-Through block copy competion response (XS_CAN0)	High	HOST_CLK_AON
CTM_RESP_1	2	O	Cut-Through block copy competion response (XS_CAN1)	High	HOST_CLK_AON
CTM_RESP_2	2	O	Cut-Through block copy competion response (XS_CAN2)	High	HOST_CLK_AON
CTM_RESP_3	2	O	Cut-Through block copy competion response (XS_CAN3)	High	HOST_CLK_AON
PORT ENABLE from XS_CANs					
PORT_EN_0	1	I	Port enable (XS_CAN0)	High	HOST_CLK_AON
PORT_EN_1	1	I	Port enable (XS_CAN1)	High	HOST_CLK_AON
PORT_EN_2	1	I	Port enable (XS_CAN2)	High	HOST_CLK_AON
PORT_EN_3	1	I	Port enable (XS_CAN3)	High	HOST_CLK_AON
CTM Descriptor from XS_CANs					
CTM_DESCRIPTOR_0	64	I	CT Descriptor Source and Destination address (XS_CAN0)	na	HOST_CLK_AON
CTM_DESCRIPTOR_1	64	I	CT Descriptor Source and Destination address (XS_CAN1)	na	HOST_CLK_AON
CTM_DESCRIPTOR_2	64	I	CT Descriptor Source and Destination address (XS_CAN2)	na	HOST_CLK_AON
CTM_DESCRIPTOR_3	64	I	CT Descriptor Source and Destination address (XS_CAN3)	na	HOST_CLK_AON

Table 20. Port List (page 7 of 7)

Signal	Bit Width	I/O	Description	Active Level	Clock Domain
CTM Enable from XS_CANs					
CTM_EN_0	1	I	CTM enable (XS_CAN0)	High	HOST_CLK_AON
CTM_EN_1	1	I	CTM enable (XS_CAN1)	High	HOST_CLK_AON
CTM_EN_2	1	I	CTM enable (XS_CAN2)	High	HOST_CLK_AON
CTM_EN_3	1	I	CTM enable (XS_CAN3)	High	HOST_CLK_AON
MH Enable from XS_CANs					
MH_EN_0	1	I	MH enable (XS_CAN0)	High	HOST_CLK_AON
MH_EN_1	1	I	MH enable (XS_CAN1)	High	HOST_CLK_AON
MH_EN_2	1	I	MH enable (XS_CAN2)	High	HOST_CLK_AON
MH_EN_3	1	I	MH enable (XS_CAN3)	High	HOST_CLK_AON
CTM Block Copy Error from XS_CANs					
CTM_BC_ERR_0	1	I	CTDMA Block Copy Error (XS_CAN0)	High	HOST_CLK_AON
CTM_BC_ERR_1	1	I	CTDMA Block Copy Error (XS_CAN1)	High	HOST_CLK_AON
CTM_BC_ERR_2	1	I	CTDMA Block Copy Error (XS_CAN2)	High	HOST_CLK_AON
CTM_BC_ERR_3	1	I	CTDMA Block Copy Error (XS_CAN3)	High	HOST_CLK_AON
Interrupt					
CTDMA_IRQ	1	O	CTDMA Interrrupt	High	HOST_CLK_AON
HARDWARE DEBUG PORT					
HDP	16	O	Hardware Debug Port	na	HOST_CLK_AON

1.13 Application Information

1.13.1 Configuration of XS_CAN Mode

CTDMA supports up to 4 XS_CANS, connected to the AHB Master ports. The connected XS_CANs can be configured in either Full Message Mode or Cut through Mode.
a) All connected XS_CAN IPs can work in FMM
b) Some XS_CAN IPs work in FMM and some in CTM
c) All XS_CAN IPs can work in CTM
Note that CAN FD Light Commander Mode (XS_CAN.PRT.MODE.LCHB = 1) is not supported when XS_CAN IPs are used in CTM with CTDMA. Refer section 1.10 Application Information in [3] for more details.

1.13.2 Constraints on Local Memory Sharing

All XS_CAN IPs connected to one CTDMA should not share the LMEM with other XS_CAN IPs that are not connected to the same CTDMA.

1.14 Programming Guidelines

1.14.1 Start Operation

Steps to program and start the IP.

1. Ensure that the clock is up and running and the reset is de-asserted. Wait for at least 20 clock cycles (HOST_CLK_AON) after reset before starting any operation.
2. Configure the registers in CTDMA. Refer section Software Interface in Chapter 1.3 TOP.
3. Start the transfers at AHB Slave Interface.

1.15 Glossary

Term/Acronym	Description
AF	Acceptance Field
ASIC	Application Specific IC
ASIL	Automotive Safety Integrity Level
BCD	Binary Coded Decimal
Buffer	1x Element in the FIFO
CAN	Controller Area Network
CAN CC	Classic CAN
CAN FD	CAN with Flexible Data rate
CAN FD Light	A commander / responder protocol based on CAN FD
CAN XL	CAN with extended frame Length
CAN XL Light	A commander / responder protocol based on CAN XL
CDC	Clock Domain Crossing
CiA	CAN in Automation
CiA 610-1	CiA CAN XL Protocol Standard
Commander	Commander (Master) in a Commander Responder Network
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check used for data consistency check
CRU	Clock Recovery Unit
CTM	Cut Through Mode (CTM) means the IP store messages only partly in the LMEM
DFA	Dependent Failure Analysis
DIA	Development Interface Agreement
DLC	Data Length Code
DLC-XL	Data Length Code with CAN XL encoding
DMA	Direct Memory Access
VBM	Virtual Buffer Manager
DST	Destination (data destination)
E_MEM	External Memory accessible off chip
EMI	Electro-Magnetic Interference
ETXP	Enhanced Traffic Pause
FEC	Filter Element Configuration
FF(s)	Flip-Flop(s)
FE	Filter Element
FIFO	First In First Out
FIM	Fault Injection Module

Term/Acronym	Description
FIR	Fault Injection Request
FM_PLL	Phase Lock Loop with Frequency Modulation to spread the noise spectrum
FMEDA	Failure Mode, Effects and Diagnostic Analysis
FMM	Full Message Mode (FMM) means the IP stores always full (complete) messages in LMEM
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input / Output
HDP	Hardware Debug Port
HFI	Header Format Invalid
HFI	Header Format Invalid
HOST	This is the CPU which is hosting the X_CAN
HW	Hardware
IP	Intellectual property e.g., the XS_CAN
IR	Interrupt
IRC	Interrupt Controller
IRQ	Interrupt Request
ISO	International Standardization Organization
ISO 11898-1:2015	ISO Standard for CAN
ISO 11898-1:2024	ISO CAN protocol specification (CAN, CAN FD, CAN XL and CAN FD Light Responder)
ISO 11898-1:2024 Annex A	ISO CAN FD Light Responder Standard
ISO 21434	ISO Standard for Cybersecurity
ISO 21434	This document specifies engineering requirements for cybersecurity risk management regarding concept, product development, production,. operation, maintenance and decommissioning of electrical and electronic (E/E) systems in road vehicles, including their components and interfaces
L_MEM	Local Memory accessible on chip
LMEM	Local Memory
LMEMPC	LMEM protection configuration
LSB	Least Significant Bit
MEM Memory	Memory
MESSAGE	The information package to be transported on the CAN bus
MH	Message Handler
MSB	Most Significant Bit
MSG	Message, see MESSAGE
MUX	Multiplexer
NA	Not Applicable
PARITY	Parity bit used for data consistency check
PLL	Phase Locked Loop
PRT	Protocol Controller
PWM	Pulse Width Modulation
PWME	Pulse Width Modulation Encoder
PWML	PWM long phase length
PWMO	PWM time offset parameter
PWMS	PWM short phase length
QUEUE	Buffer e.g., for Messages
REFP	Reference Pairs
Responder	Responder (Slave) in a Commander Responder Network
RTL	Register Transfer Level (design abstraction level)

Term/Acronym	Description
RX	Receive, e.g., reception of a frame on the CAN bus
RXFQ	Rx FIFO Queue
RxMsg	Rx Message
RxQE	RX FIFO Queue Element (RXQE)
S_MEM	System Memory. This memory could be an on-chip RAM or an E_MEM
SDT	SDU Type
SEC	Simple Extended Content
SEooC	Safety Element out of Context
SMEM	System Memory
SoC	System on Chip
SPI	Serial Peripheral Interface
SRAM	On chip Static RAM
SRC	Source (data source)
SW	Software
TEFQ	TX Event FIFO Queue
TEQE	TX Event Queue Element
TPE	TX Pause Enhanced
TSS	Transceiver Sharing Switch
Tx	Transmit
TX	Transmit, e.g., transmission of a frame on the CAN bus
Tx FIFO	dynamic message memory where the message are transmitted in the order of First in First out
Tx Pause	Delay of the next transmit message, see TSM
Tx Queue	dynamic message memory where the message are transmitted in the order of their CAN priority
TX SCAN	SCAN Process used to select the TX message having the highest priority before sending if to the PRT
TXFQ	TX FIFO Queue
TXPQ	Tx Priority Queue
TXQE	TX Queue Element
VCID	Virtual CAN Network ID
VHDL	VHSIC Hardware Description Language
VHSIC	Very High-Speed Integrated Circuit
WORD	Data word used by X_CAN architecture = 32 bit = DWORD
X_CAN	Family of CAN IP supporting CAN XL protocol
XCAND	X_CAN with embedded DMA
XS_CAN	Slim CAN XL IP

1.16 References

This document refers to the following documents:

Ref Author(s) Title
[1] ISO ISO 11898-1:2024 CAN data link layer and physical signaling
[2] CAN in Automation CiA 610-2 CAN XL Protocol Specification,
[3] XS_CAN IP User Manual
[4] CTDMA User Manual
[5] MS/EEJ calc_min_host_clk_freq v10
[6] AXI ARM 102202_0100_01_Introduction_to_AMBA_AXI
[7] AHB AMBA 2 ARM IHI 0011A
[8] APB AMBA 4 ARM IHI 0011A
[9] ISO ISO 26262-5:2018 Road vehicles - Functional safety
[10] IEC IEC/TR 62380:2004 Reliability data handbook - Universal model

for reliability prediction of electronics components, PCBs and equipment
[11] AIAG & VDA AIAG & VDA FMEA Handbook - 1st Edition - June 2019
[12] CTDMA Integration Guide

1.17 User Manual Version History

Table 21. Local Change History

Version	Date of Issue	Description
1.1	Aug 12th, 2026	Document status changed to "Released".
1.0	April 28th, 2026	Typo corrections, Section 1.3.3.2 AHB Slave Interface and Multiplexer updated
0.9	April 1st, 2026	Section 1.8.1 Timeout at CTMANAGER AXI Master interface description updated.
0.8	Mar 11th, 2026	Section 1.6 AHB Arbiter Throttling Values modified for the table numbers 18 and 19.
0.7	Feb 25th, 2026	Section 1.6 AHB Arbiter Throttling Values modified for the table numbers 18 and 19.
0.6	Feb 3rd, 2026	Section 1.7 AHB Control Multiplexer and Decoder updated.
0.5	Sept 25th, 2025	1) CT Manager functional description section 1.4.4 updated. 2) Error and Exception Handling scenarios section 1.8.3, 1.8.9, 1.8.10 updated and 1.8.12 added.
0.4	Aug 13th, 2025	Error and Exception Handling scenarios updated. ARB_HMASTER removed from arbiter
0.3	July 25th, 2025	CTDMA behavior related to Address parity error and accesses to disabled ports updated. AHB Slave interfaceand Multiplexer section updated. PORT_EN field description in CTDMA_SFTY_CFG register updated. Error and Exception Handling scenarios updated. AXI xUSER signal description updated
0.2	May 9th, 2025	Error scenarios updated, Register names updated, Programming Guidelines added, PORT_EN description updated.
0.1	March 7th, 2025	Initial version

1.18 Disclaimer

The content of this document is subject to continuous developments and improvements. All particulars and its use contained in this document are given by BOSCH in good faith.

NO WARRANTIES: TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON, EITHER EXPRESSLY OR IMPLICITLY, WARRANTS ANY ASPECT OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING ANY OUTPUT OR RESULTS OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO UNLESS AGREED TO IN WRITING. THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS BEING PROVIDED "AS IS", WITHOUT ANY WARRANTY OF ANY TYPE OR NATURE, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY THAT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS FREE FROM DEFECTS.
ASSUMPTION OF RISK: THE RISK OF ANY AND ALL LOSS, DAMAGE, OR UNSATISFACTORY PERFORMANCE OF THIS SPECIFICATION (RESPECTIVELY THE PRODUCTS MAKING USE OF IT IN PART OR AS A WHOLE), SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO RESTS WITH YOU AS THE USER. TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS,

COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON EITHER EXPRESSLY OR IMPLICITLY, MAKES ANY REPRESENTATION OR WARRANTY REGARDING THE APPROPRIATENESS OF THE USE, OUTPUT, OR RESULTS OF THE USE OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, BEING CURRENT OR OTHERWISE. NOR DO THEY HAVE ANY OBLIGATION TO CORRECT ERRORS, MAKE CHANGES, SUPPORT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, DISTRIBUTE UPDATES, OR PROVIDE NOTIFICATION OF ANY ERROR OR DEFECT, KNOWN OR UNKNOWN. IF YOU RELY UPON THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, YOU DO SO AT YOUR OWN RISK, AND YOU ASSUME THE RESPONSIBILITY FOR THE RESULTS. SHOULD THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL LOSSES, INCLUDING, BUT NOT LIMITED TO, ANY NECESSARY SERVICING, REPAIR OR CORRECTION OF ANY PROPERTY INVOLVED TO THE MAXIMUM EXTEND PERMITTED BY LAW.
DISCLAIMER: IN NO EVENT, UNLESS REQUIRED BY LAW OR AGREED TO IN WRITING, SHALL THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS OR ANY PERSON BE LIABLE FOR ANY LOSS, EXPENSE OR DAMAGE, OF ANY TYPE OR NATURE ARISING OUT OF THE USE OF, OR INABILITY TO USE THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING, BUT NOT LIMITED TO, CLAIMS, SUITS OR CAUSES OF ACTION INVOLVING ALLEGED INFRINGEMENT OF COPYRIGHTS, PATENTS, TRADEMARKS, TRADE SECRETS, OR UNFAIR COMPETITION.
INDEMNIFICATION: TO THE MAXIMUM EXTEND PERMITTED BY LAW YOU AGREE TO INDEMNIFY AND HOLD HARMLESS THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, AND EMPLOYEES, AND ANY PERSON FROM AND AGAINST ALL CLAIMS, LIABILITIES, LOSSES, CAUSES OF ACTION, DAMAGES, JUDGMENTS, AND EXPENSES, INCLUDING THE REASONABLE COST OF ATTORNEYS' FEES AND COURT COSTS, FOR INJURIES OR DAMAGES TO THE PERSON OR PROPERTY OF THIRD PARTIES, INCLUDING, WITHOUT LIMITATIONS, CONSEQUENTIAL, DIRECT AND INDIRECT DAMAGES AND ANY ECONOMIC LOSSES, THAT ARISE OUT OF OR IN CONNECTION WITH YOUR USE, MODIFICATION, OR DISTRIBUTION OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, ITS OUTPUT, OR ANY ACCOMPANYING DOCUMENTATION.
GOVERNING LAW: THE RELATIONSHIP BETWEEN YOU AND ROBERT BOSCH GMBH SHALL BE GOVERNED SOLELY BY THE LAWS OF THE FEDERAL REPUBLIC OF GERMANY. THE STIPULATIONS OF INTERNATIONAL CONVENTIONS REGARDING THE INTERNATIONAL SALE OF GOODS SHALL NOT BE APPLICABLE. THE EXCLUSIVE LEGAL VENUE SHALL BE DUESSELDORF, GERMANY.
MANDATORY LAW SHALL BE UNAFFECTED BY THE FOREGOING PARAGRAPHS.
INTELLECTUAL PROPERTY OWNERS/COPYRIGHT OWNERS/CONTRIBUTORS: ROBERT BOSCH GMBH, ROBERT BOSCH PLATZ 1, 70839 GERLINGEN, GERMANY AND ITS LICENSORS.

Robert Bosch GmbH
Robert-Bosch-Platz 1
70839 Gerlingen-Schillerhöhe