User Manual, V 1.0, Nov. 2005

CIC751

Companion IC

16bit

Microcontrollers

Never stop thinking

Edition 2005-11

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
81669 München, Germany

Attention please!

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

Terms of delivery and rights to technical change reserved.

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

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

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

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

CIC751

Companion IC

Microcontrollers

Never stop thinking

CIC751
CONFIDENTIAL

Revision History:	**2005-11**	V 1.0
Previous Version:	None.	

Page	Subjects (major changes since last revision)

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

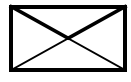


Table Of Contents

Table Of Contents

1 Introduction

The CIC751 is a companion IC for the Infineon AUDO-NG family of 32-bit microcontrollers. The major function of the CIC751 is to provide the AUDO-NG 32-bit microcontrollers with the capability of a 5 V Analog to Digital Converter (ADC). The interconnection of the CIC751 and the microcontroller is accomplished via either the Micro Link Interface (MLI) or the Synchronous Serial Interface (SSC). Internal operations of the CIC751 are supported by the very flexible on-chip DMA controller.

1.1 Overview

Figure 1-1 provides the block diagram of the CIC751 companion chip. This design allows access to the ADC by the host CPU without sacrificing any of the features of the ADC. This can be achieved because all registers of the ADC are mapped to the on-chip bus. This bus can be accessed via one of the two serial interfaces. Selection of the interface is made via pin MODE, which can be directly connected to the supply voltage or via pull-up/down resistors.

The bus domain is completely separated from the address domain on the CPU chip. The addresses of all modules on the companion chip are 32-bit addresses. Transactions between the CPU and the SSC are executed with the SSC transmission protocol; transactions between the MLI and the CPU use the MLI transmission protocol.

Each transaction via any of the two serial interfaces is defined by address, data, data width, and type of frame. The address from which data is read or written to, is related to the address domain. The data width may be 8, 16 or 32 bits for the MLI and 16 bits for the SSC. The ADC and the MLI may send request triggers to the DMA Controller.

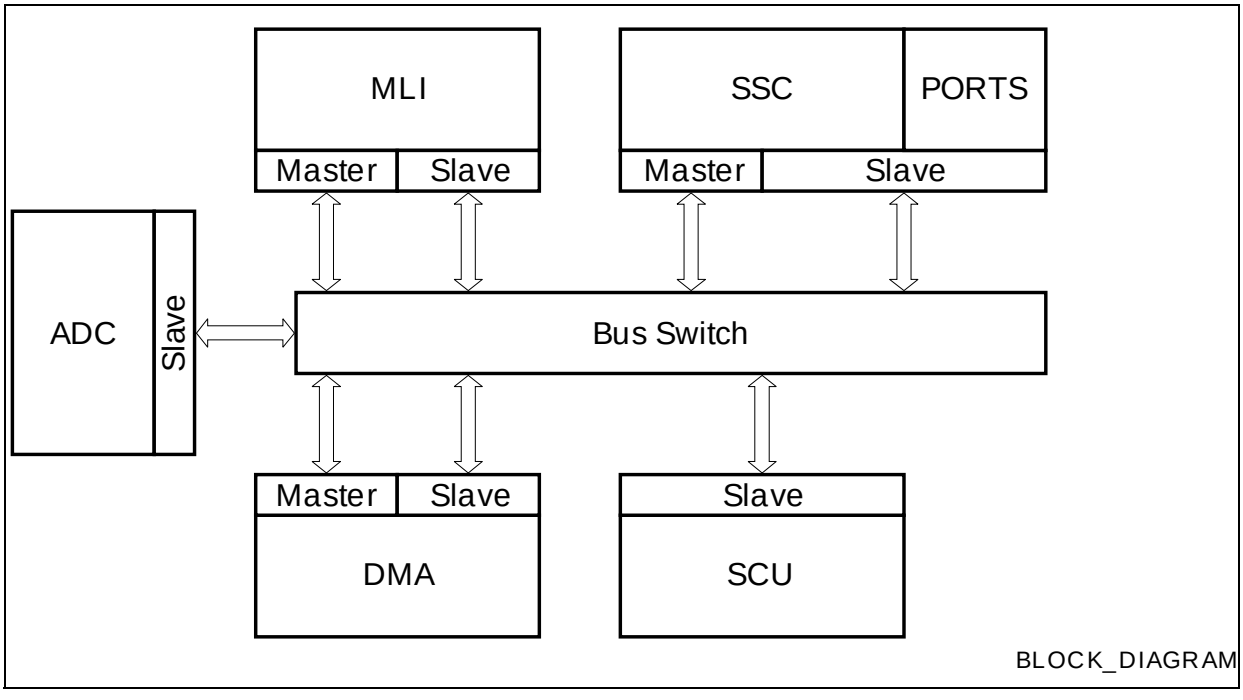


Figure 1-1 CIC751 Block Diagram

1.2 Features

This section provides a high-level description of the features on the CIC751.

- 5 V Analog to Digital Converter
- 16 analog input channels
- Internal low power oscillator
- Slave (SPI) SSC interface operating on 5 V or 3.3 V
- MLI Interface operating on 5 V or 3.3 V
- Maximum system frequency of 40 MHz
- Low-power design
- Single power supply concept design (for pad and core supply)
- Seperated ADC supply
- Input and output pins with 3.3 V and 5.0 V
- Flexible clocking concept
- Crossbar bus architecture

1.2.1 Detailed Features

The following sections provide detailed information about each of the on-chip modules.

1.2.1.1 ADC

The CIC751 provides an Analog/Digital Converter with 8-bit or 10-bit resolution and a sample & hold circuit on-chip. An input multiplexer selects between up to 16 analog input channels either via software (Fixed Channel Modes) or automatically (Auto Scan Modes).

To fulfill most requirements of embedded control applications, the ADC supports the following conversion modes:

- **Standard Conversions**
  - **Fixed Channel Single Conversion**
    produces just one result from the selected channel
  - **Fixed Channel Continuous Conversion**
    repeatedly converts the selected channel
  - **Auto Scan Single Conversion**
    produces one result from each of a selected group of channels
  - **Auto Scan Continuous Conversion**
    repeatedly converts the selected group of channels
  - **Wait for Read Mode**
    start a conversion automatically when the previous result was read
- **Channel Injection Mode**
  can insert the conversion of a specific channel into a group conversion (auto scan)

The key features of the ADC are:

- Use of Successive Approximation Method
- Integrated sample and hold functionality
- Analog Input Voltage Range from 0V to 5V
- 16 Analog Input Channels
- 16 ADC result registers
- Resolution:
  8-Bit or 10-Bit in Compatibility Mode
- Minimum Conversion Time:2.55 µs @ 10-Bit
- Total Unadjusted Error (TUE):±1 LSB @ 8-Bit, ± 2 LSB @10-Bit
- Support of several Conversion Modes
  Fixed Channel Single Conversion
  Fixed Channel Continuous Conversion
  Auto Scan Single Conversion
  Auto Scan Continuous Conversion
  Wait for Result Read and Start Next Conversion
  Channel Injection during Group Conversion
- Programmable Conversion and Sample Timing Scheme
- Automatic Self-Calibration to changing temperatures or process variations

1.2.1.2 MLI

The Micro Link Interface (MLI) is a fast synchronous serial interface that makes it possible to exchange data between microcontrollers or other devices.

The key features of the MLI are:

- Synchronous serial communication between an MLI transmitter and an MLI receiver
- Different system clock speeds are supported in the MLI transmitter and MLI receiver due to full handshake protocol (4 lines between a transmitter and a receiver)
- Fully transparent read/write access is supported (= remote programming)
- Complete address range of target device (Remote Controller) is available
- Specific frame protocol to transfer commands, addresses, and data
- Error detection by parity bit
- 32-bit, 16-bit, or 8-bit data transfers are supported
- Programmable baud rate: $f_{MLI}/2$ (max.: $f_{MLI} = f_{SYS}$)
- Multiple receiving devices are supported

1.2.1.3 SSC

The SSC supports full-duplex and half-duplex serial synchronous communication up to 10 Mbit/s (@ 40 MHz module clock). The serial clock signal is received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A shift clock generator provides the SSC with a separate serial clock signal.

This section describes only the use of the SSC module as a slave because the CIC751 always operates as a slave to a host.

Features

- Slave Mode operation
  - Full-duplex or half-duplex operation
  - Automatic pad control possible
- Flexible data format
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: Idle low or idle high state for the shift clock
  - Programmable clock/data phase: Data shift with leading or trailing edge of the shift clock
- Internal Master Function
  - Access to the all addresses
  - Automatic address handling
  - Automatic data handling

1.3 Signal Description

This section describes signals that connect off chip. **Table 1-1** gives a summery of the CIC751 external signals (pins).

Table 1-1 Pin Definitions and Functions

Symbol	Pin/Port	I/O	Function
AIN0	35 P1.0	I	Analog Input 0[1)] For this pin a Multiplexer Test Mode is available.
AIN1	36 P1.1	I	Analog Input 1[1)]
AIN2	37 P1.2	I	Analog Input 2[1)]
AIN3	38 P1.3	I	Analog Input 3[1)]
AIN4	1 P1.4	I	Analog Input 4[1)]
AIN5	2 P1.5	I	Analog Input 5[1)]
AIN6	7 P1.6	I	Analog Input 6[1)]
AIN7	8 P1.7	I	Analog Input 7[1)]
AIN8	5 P1.8	I	Analog Input 8[1)]
AIN9	6 P1.9	I	Analog Input 9[1)]
AIN10	3 P1.10	I	Analog Input 10[1)]
AIN11	4 P1.11	I	Analog Input 11[1)]
AIN12	11 P1.12	I	Analog Input 12[1)]
AIN13	12 P1.13	I	Analog Input 13[1)]
AIN14	13 P1.14	I	Analog Input 14[1)]

Table 1-1 Pin Definitions and Functions (cont'd)

Symbol	Pin/Port	I/O	Function
AIN15	14 P1.15	I	Analog Input 15[1)]
VAREF	9	I	Analog Reference Voltage
VAGND	10	I	Analog Ground
TCLK/SR3	17 P0.0	I/O	MODE = 0: MLI Transmit Channel Clock Output MODE = 1: Event output line 3
TREADY/SR4	19 P0.1	I/O	MODE = 0: MLI Transmit Channel Ready Input MODE = 1: Event request output line 4
TVALID/SCLK	20 P0.2	I/O	MODE = 0: MLI Transmit Channel Valid Output MODE = 1: SPI Serial Channel Clock
TDATA/MRST	21 P0.3	I/O	MODE = 0: MLI Transmit Channel Data Output MODE = 1: SPI Master Receive Slave Transmit
RCLK	22 P0.4	I/O	MODE = 0: MLI Receive Channel Clock Input MODE = 1: GPIO
RREADY/RDY	23 P0.5	I/O	MODE = 0: MLI Receive Channel Ready Output MODE = 1: SSC Ready Signal
RVALID/$\overline{\text{SLS}}$	24 P0.6	I/O	MODE = 0: MLI Receive Channel Valid Input MODE = 1**:** SSC Select Slave
RDATA/MTSR	25 P0.7	I/O	MODE = 0: MLI Receive Channel Data Input MODE = 1**:** SPI Master Transmit Slave Receive

Table 1-1 Pin Definitions and Functions (cont'd)

Symbol	Pin/Port	I/O	Function
MODE [2]	26 P0.8	I/O	Interface Selection Pin MODE selects whether the on-chip MLI or SSC are used to access the CIC751 device. 0: On-chip MLI 1: On-chip SSC Event request output line 5 (SR5) After latching the initial state with the rising edge of the $\overline{\text{PORST}}$ signal (see **Chapter 2.5**), this pin can be used as an additional general purpose or SR5 output line.
$\overline{\text{TESTMODE}}$ [3]	27 P0.9	I/O	Test Mode Selection 0: Reserved; do no use 1: Normal Mode After latching the initial state with the rising edge of the $\overline{\text{PORST}}$ signal, this pin can be used as an additional general purpose or special function I/O line (see **Chapter 2.5**).
SR0	28 P0.10	I/O	Event request output line 0
SR1	29 P0.11	I/O	External Trigger
SR2	30 P0.12	I/O	External Trigger
$\overline{\text{PORST}}$	31	I	Power-on Reset
V_{DDM}	34	+5 V	Power Supply, supply for ADC module
V_{DDP}	18, 33	+3.3 V or +5.0 V	Power Supply, supply for I/O pads
V_{DDC}	16	+2.5 V	Power Supply, supply for digital module cores
V_{SS}	15, 32	0 V	Ground

1) In addition to the analog input function of pin P1.x, a digital input stage is available. This input stage is activated while STCU_SYSCON.P1DIDIS = 0.

2) The initial logic state on pin MODE is latched while the $\overline{\text{PORST}}$ input is active. A weak pull-up can be disabled if used as the SR5 pin.

3) The initial logic state on pin $\overline{\text{TESTMODE}}$ is latched while the $\overline{\text{PORST}}$ input is active.

Figure 1-2 shows the pin-out for a 38-pin package.

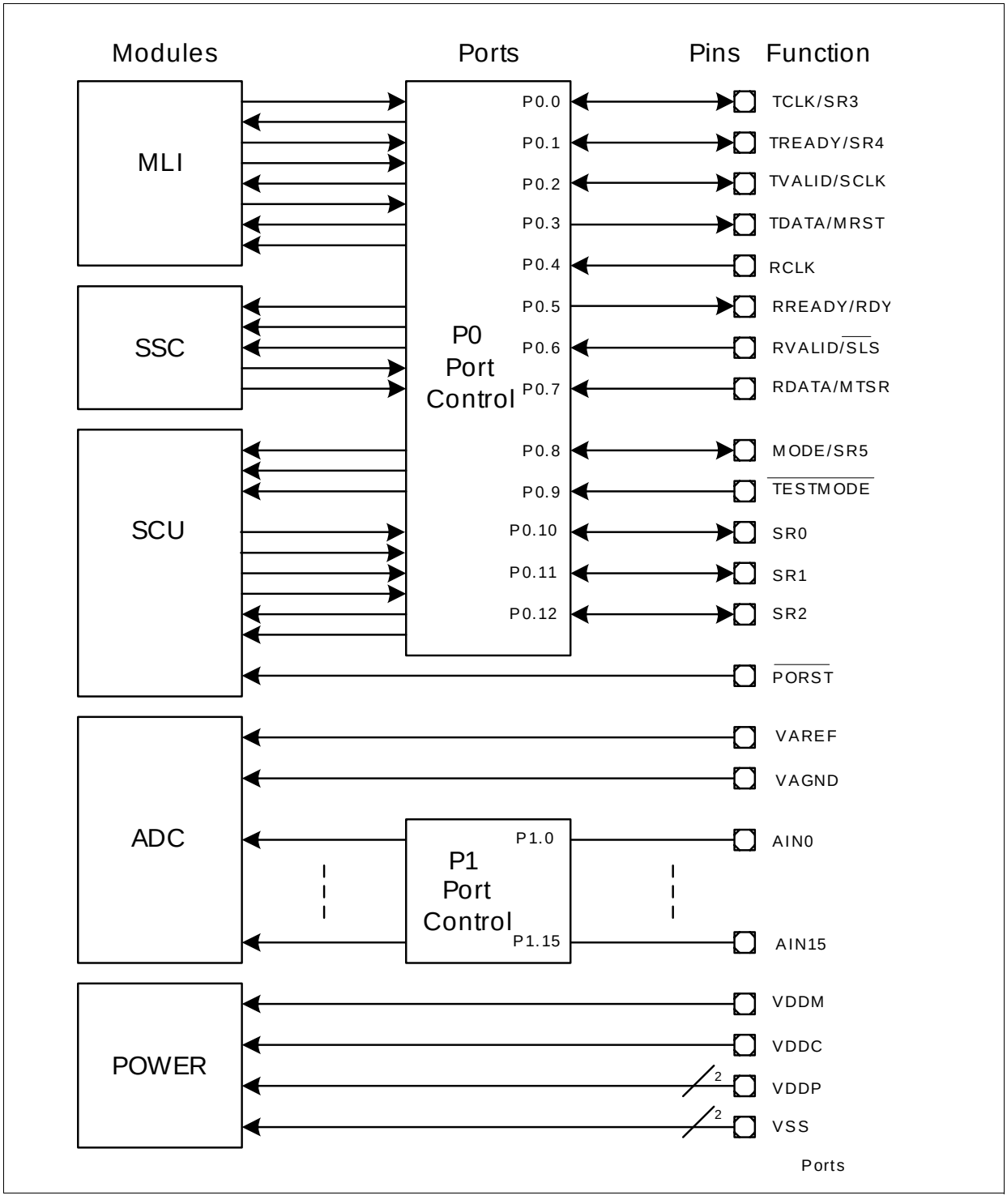


Figure 1-2 Pins for P/PG-TSSOP-38 Package

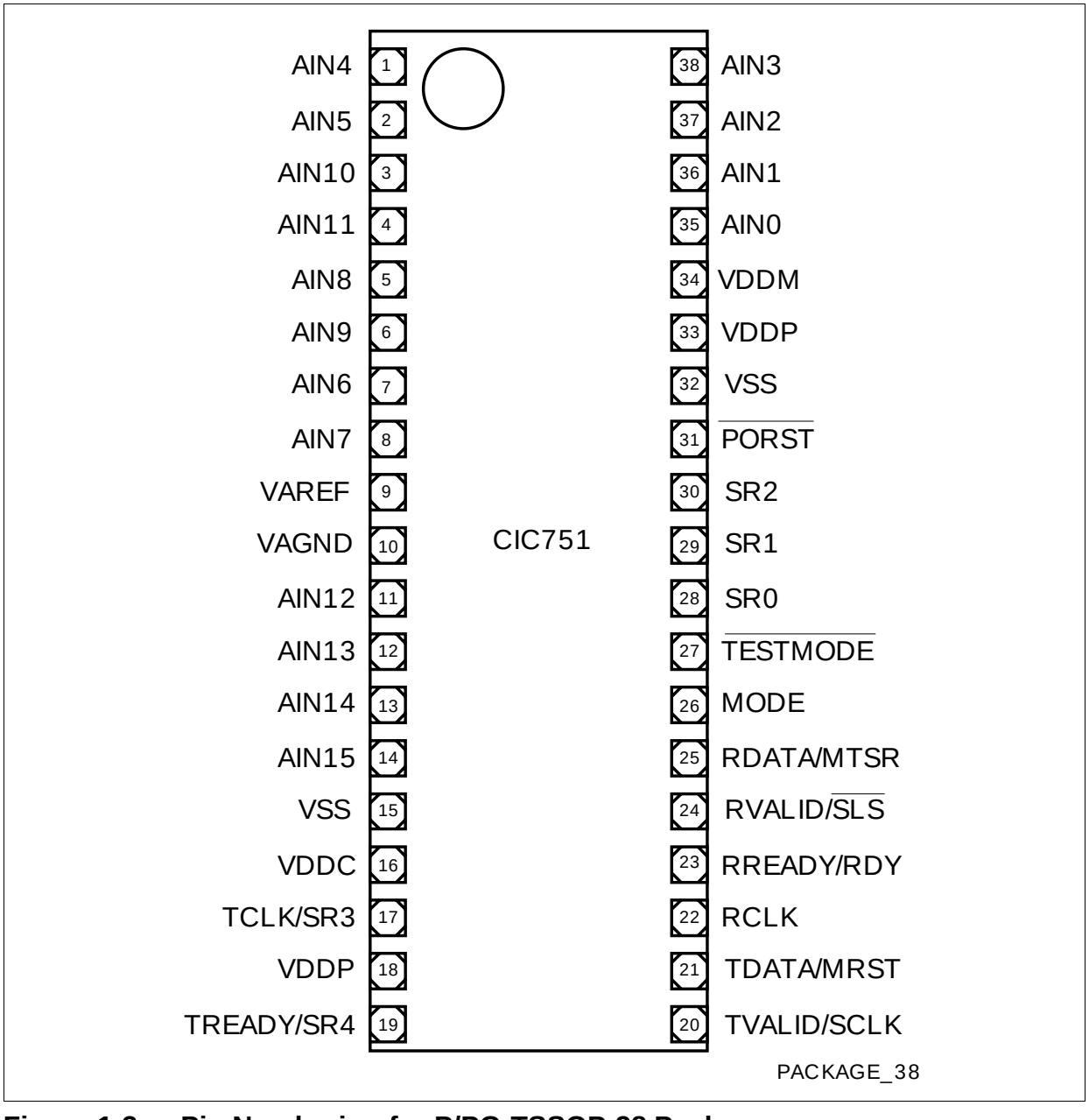


Figure 1-3 Pin Numbering for P/PG-TSSOP-38 Package

2 System and Control Unit (SCU)

The System and Control Unit (SCU) controls all system relevant tasks.

The system tasks of the SCU are:

- Reset operation (see **Chapter 2.1**)
- System clock control (see **Chapter 2.3**)
- Power supply system (see **Chapter 2.4**)
- System Interrupt control (see **Chapter 2.5**)

2.1 Reset Control Block

The single system reset function initializes the CIC751 into a defined default state and is invoked by any of the following trigger conditions:

- External $\overline{\text{PORST}}$ reset
  - Power-on reset; indicated by hardware reset input after power-on
- Internal EVR fail reset
  - The EVR encounters a problem and the required power supply levels are not longer guaranteed
- Setting bit SCU_SYSCON.SWRST
  - This generates a software trigger reset

The entire CIC751 is reset, regardless of the means by which the reset was generated. A reset always triggers a new mode selection phase with the exception that a software reset does not trigger a new Mode Selection. A software reset retains the mode currently selected.

2.1.1 Power-On Reset

The $\overline{\text{PORST}}$ input pin requests a power-on reset. Driving the $\overline{\text{PORST}}$ pin low causes a non-synchronized reset of the entire device.

$\overline{\text{PORST}}$ is equipped with a noise suppression filter which suppresses glitches below 10 ns pulse width. $\overline{\text{PORST}}$ pulses with a width above 100 ns are safely recognized.

2.1.2 Embedded Voltage Regulator (EVR) Reset

If the power supply does not reach the value required for proper functionality, a reset is applied. This ensures a reproducible behavior after power-on or in the case of power-fail.

2.2 Mode Selection

The pins $\overline{\text{TESTMODE}}$ and MODE should supplied with specific voltage levels to ensure correct configuration of the CIC751.

2.2.1 MODE Pin

The MODE pin defines whether the MLI interface or the SSC interface is activated for Normal Mode. For MODE = 0, the MLI interface is activated and configured as the only communication interface. For MODE = 1, the SSC interface is activated and configured as the only communication interface.

The value that is sampled and used for this decision must be held for 400 µs after V_{DDP} reached 1.5 V.

2.2.2 $\overline{\text{TESTMODE}}$ Pin

The pin must be tied to ’1’.

2.3 Clock System

This section describes the clock system of the CIC751. Topics include clock generation, clock domains, operation of clock circuitry, and clock control registers.

2.3.1 Overview

The CIC751 clock system performs the following functions:

- Uses the internal free running frequency of the VCO block to create a fast clock frequency f_{SYS}.
- Uses the internal oscillator of the VCO block to create a fast clock frequency f_{SYS}.
- Acquires and buffers the external clock signal (RCLK) to create a fast clock frequency f_{SYS}.
- Distributes the in-phase synchronized clock signal throughout the CIC751’s entire clock tree.

The clock system must be operational before the CIC751 can operate, so it contains special logic to handle power-up and reset operations. Its services are fundamental to the operation of the entire system, so it contains a special fail-safe logic.

2.3.2 Clock Generation Unit

The Clock Generation Unit (CGU) allows very flexible clock generation for CIC751.

The CGU in the CIC751 consists of an oscillator circuit (RCOSC), one Phase-Locked Loop (PLL) module, and a Clock Control Unit (CCU). The CGU can convert a low-frequency clock signal to a high-speed internal clock.

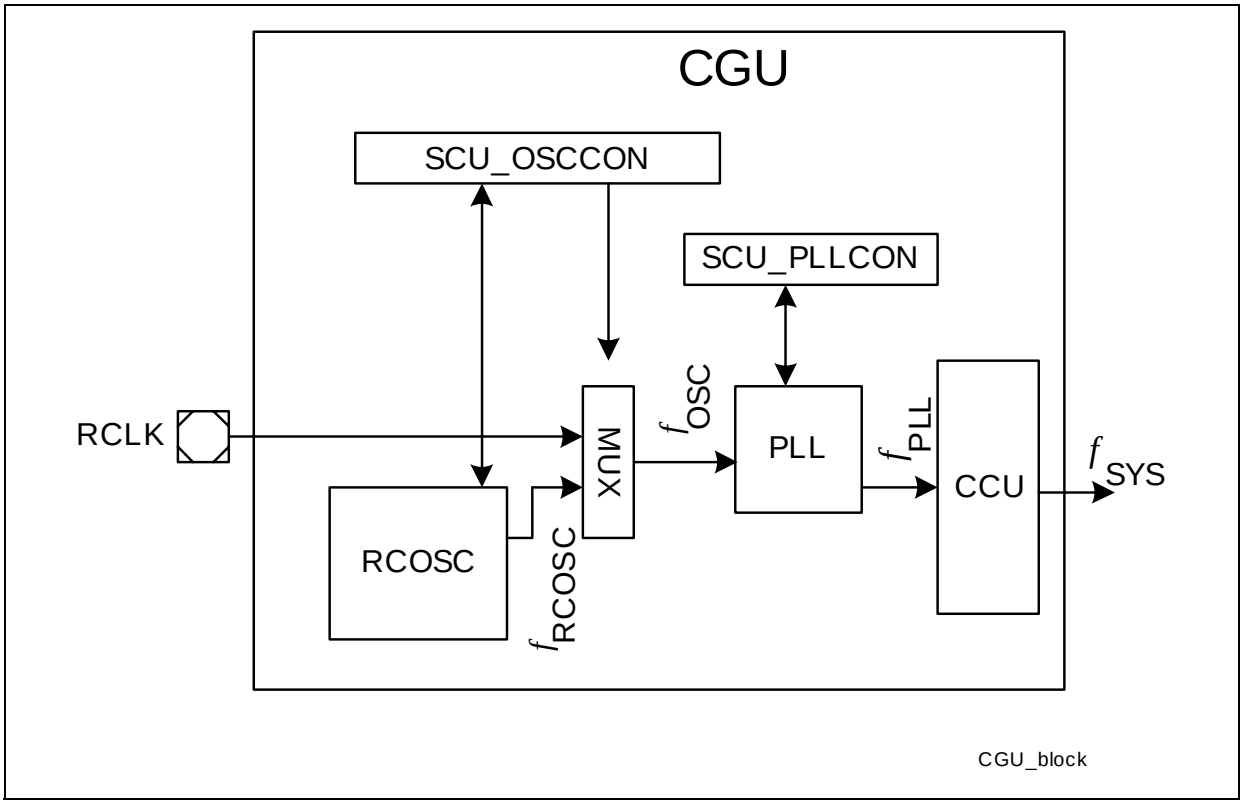


Figure 2-1 Clock Generation Unit Block Diagram

The following sections describe the various parts of the CGU:

2.3.2.1 RC Oscillator Circuit (RCOSC)

The RC Oscillator Circuit (RCOSC) is designed to work without an external crystal oscillator or an external stable clock source. The RCOSC consists of an Schmitt Trigger RC oscillator core and a standard current reference to provide a VDD-independent bias current.

Internal Clock Mode

When operating without an external crystal or clock source, the RC oscillator provides a stabile clock frequency of 9 MHz. The stability of this clock frequency is influenced by the temperature.

The system clock f_{SYS} (for Normal Mode equal to f_{PLL}) is generated from an oscillator clock f_{OSC} in one of four selectable ways:

- Bypass Mode (Direct Drive)
- Prescaler Mode
- Normal Mode
- Free running Mode

2.3.2.2 Phase-Locked Loop (PLL) Module

This section describes the PLL module of the CIC751. The PLL supplies the system with a single clock frequency.

Features

- Programmable clock generation PLL
- Loop filter
- Input frequency[1]:f_{OSC} = 3.1 to 37.5 MHz
- VCO frequency:f_{VCO} = 100 to 250 MHz (select by range)
- VCO lock detection
- Oscillator run detection
- Output frequency:f_{PLL} = 6.25 to 250 MHz
- 2bit input divider P:(divide by PDIV+1)
- 5bit feedback divider N:(multiply by NDIV+1, stability restrictions possible)
- 4bit output divider K:(divide by KDIV+1)
- Bypass Mode
- Prescaler Mode
- Freerunning Mode
- Normal Mode
- Glitchless switching between Normal Mode and Prescaler Mode

PLL Functional Description

The PLL provides the system with a clock generated from one of the various potential clock sources.

1) For P = 1, otherwise multiplied by P.

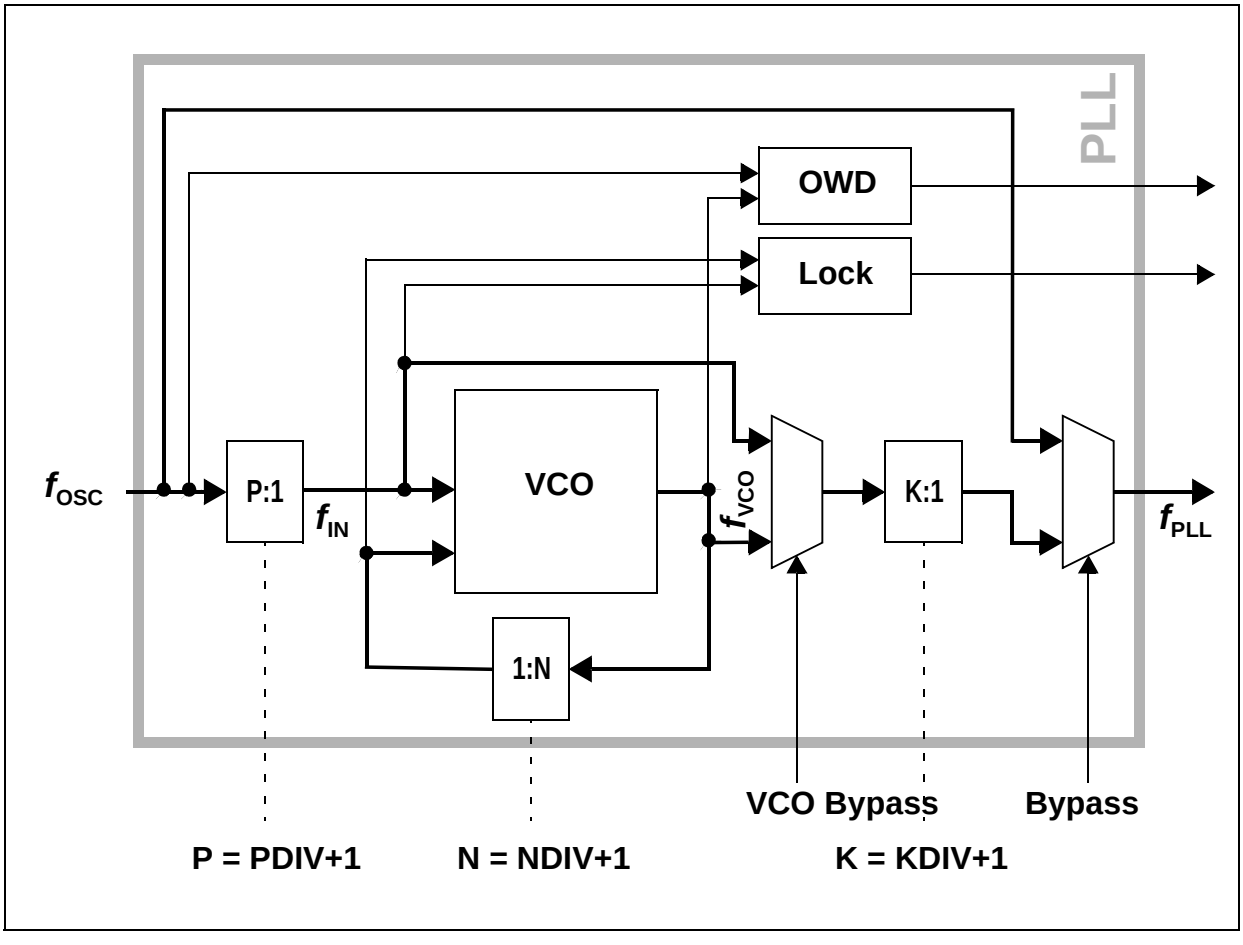


Figure 2-2 PLL Block Diagram

The PLL uses up to three dividers to manipulate the input frequency in a configurable way. Each of the three dividers can be bypassed in some way to define an operating mode

Bypassing P, N, and K divider; this defines the Bypass Mode

Bypassing N divider; this defines the Prescaler Mode

Bypassing no divider; this defines the Normal Mode

Ignoring the P divider; this defines the Freerunning Mode

Normal Mode

In Normal Mode, the input clock f_{OSC} is divided by a factor P, multiplied by a factor N, and then divided by a factor K.

So, the output frequency is given by

$$f_{PLL} = \frac{N}{P \cdot K} \cdot f_{OSC} \quad (2.1)$$

The Normal Mode is selected by setting SCU_PLLCON.PLLCTRL = 11_B.

Bypass Mode

In Bypass Mode, the input clock f_{OSC} is directly connected to the PLL output f_{PLL}.
So, the output frequency is given by

$$f_{PLL} = f_{OSC} \tag{2.2}$$

The Bypass Mode is selected by setting SCU_PLLCON.BY = 1_B.

Prescaler Mode

In Prescaler Mode, the input clock f_{OSC} is divided down by a factor P * K.
So, the output frequency is given by

$$f_{PLL} = \frac{f_{OSC}}{P \cdot K} \tag{2.3}$$

The Prescaler Mode is selected by setting SCU_PLLCON.PLLCTRL = 00_B.

Freerunning Mode

In Freerunning Mode, the base frequency output of the Voltage Controlled Oscillator (VCO) $f_{VCObase}$ is only divided by a factor K.
So, the output frequency is given by

$$f_{PLL} = \frac{f_{VCObase}}{K} \tag{2.4}$$

The Freerunning Mode is selected by setting SCU_PLLCON.PLLCTRL = 10_B.

General Configuration Overview

All three divider values and all necessary other values can be configured via the PLL configuration register **SCU_PLLCON**.

Table 2-1 lists a few possible values for the P factor and gives the valid output frequency range for the P divider dependent on P and the f_{OSC} frequency range:

Table 2-1 P-Divider Factors

P = PDIV+1	PDIV	f_{REF} for f_{OSC} =				
		4 MHz	**10 MHz**	**16 MHz**	**20 MHz**	**25 MHz**
1	0	4	10	16	20	25
2	1	not allowed	5	8	10	12.5
3	2		$3.3\overline{3}$	$5.3\overline{3}$	$6.6\overline{6}$	$8.3\overline{3}$
4	3	not allowed		4	5	6.25

Note: Of course, the entire range between two f_{OSC} columns in the above table is allowed. E.g. for a range f_{OSC} = 20 to 25, and P = 3, f_{REF} = $6.6\overline{6}$ to $8.3\overline{3}$ MHz.

The P-divider output frequency f_{REF} is fed to a Voltage Controlled Oscillator (VCO).The VCO is part of the PLL with a feedback path. A divider in the feedback path (N divider) divides the VCO frequency. As well as N, the correct range of f_{VCO} must be chosen by configuring SCU_PLLCON.PLLVB:

Table 2-2 VCO Ranges

PLLVB [1:0]	f_{VCOmin}	f_{VCOmax}	$f_{VCObase}$ [1)]	Unit
01	150	200	40…130	MHz
00	100	150	20…80	MHz
10	200	250	60…180	MHz
11	Reserved; do not use			

1) $f_{VCObase}$ is the free running operation frequency of the PLL, when no input clock is available. These values are only preliminary and are later updated with more exact simulation and measurement results from the PLL.

The VCO band (100...150 MHz, 150...200 MHz, 200...250 MHz) must be selected according to the desired VCO output frequency (100...250 MHz). **Figure 2-3** illustrates how this output frequency depends on the input frequency and the multiplication factor.

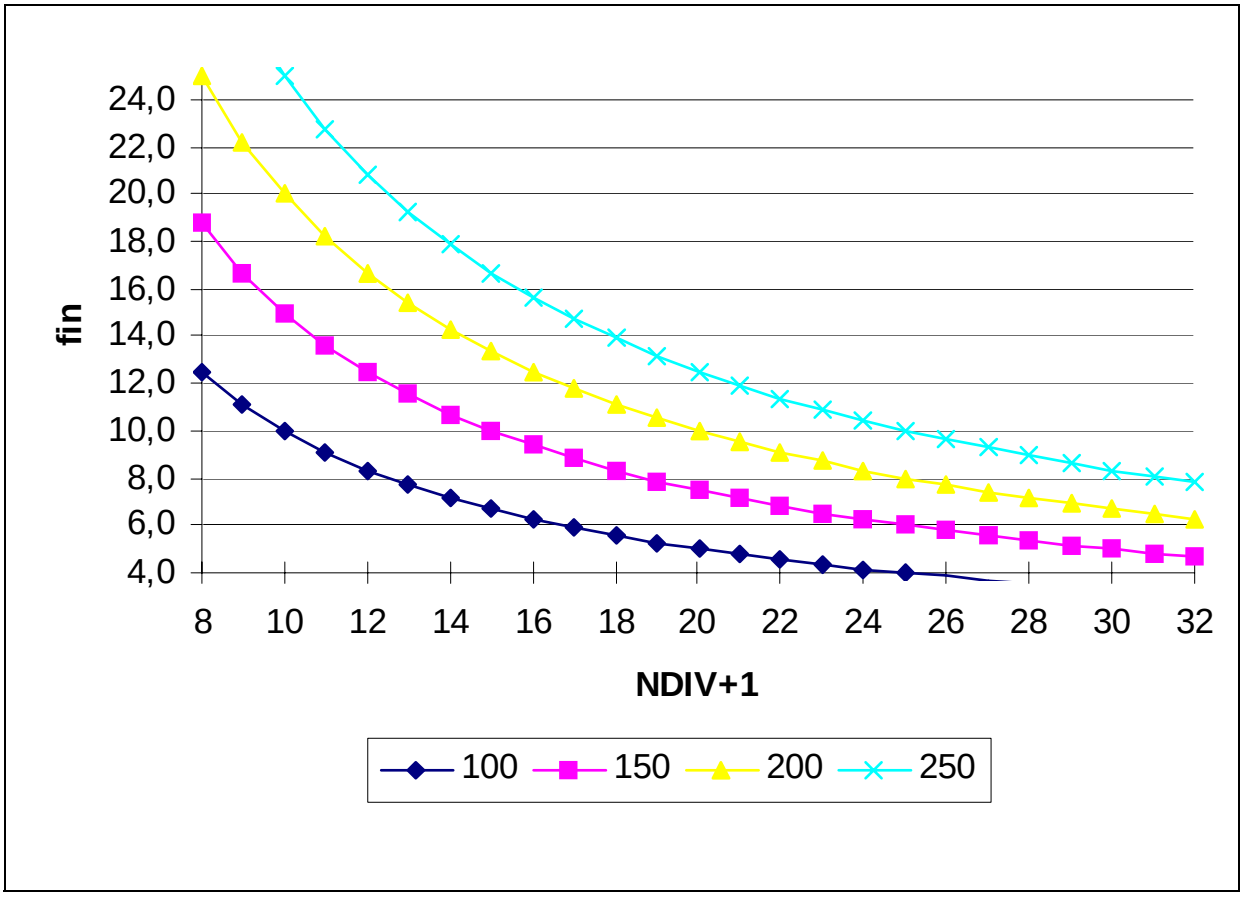


Figure 2-3 VCO Band Selection

Table 2-3 lists the possible N loop division rates and gives the valid output frequency range for f_{REF} depending on N and the VCO frequency range:

Table 2-3 N Loop Division Rates

N = NDIV+1	NDIV	f_{REF} for f_{VCO} =			
		100	**150**	**200**	**250**
≤ 7	≤ 6	not allowed[1)]			
8	7	12.5	18.75	25.00	31.25
9	8	$11.1\overline{1}$	$16.6\overline{6}$	$22.2\overline{2}$	$27.7\overline{7}$
10	9	10.00	15.00	20.00	25.00
11…30	10…29	…	…	…	…
31	30	$3.2\overline{2}$	4.84	6.45	8.06
32	31	3.13	4.69	6.25	7.81

1)Values in this range are allowed in Freerunning Mode, but have no impact there.

Note: Of course, the entire range between two f_{VCO} columns in the above table is allowed.

The N-divider output frequency f_{DIV} is then compared with f_{REF} in the phase detector logic, which is within the VCO logic. The phase detector determines the difference between the two clock signals and accordingly controls the output frequency of the VCO, f_{VCO}.

Note: Due to this operation, the VCO clock of the PLL has a frequency that is a multiple of f_{REF}. The factor for this is controlled through the value applied to the N-divider in the feedback path. For this reason, this factor is often called a multiplier, although it actually controls division.

The output frequency of the VCO, f_{VCO}, is divided by K to provide the final desired output frequency f_{PLL}. **Table 2-4** lists the output frequency range depending on the K divisor and the VCO frequency range:

Table 2-4 K Divisor Table

K = K2DIV+1	K2 DIV	f_{PLL} for f_{VCO}=				Duty Cycle [%]
		100	**150**	**200**	**250**	
1	0	100.0	150.0	200.0	250.0	45 - 55
2	1	50.0	75.0	100.0	125.0	50
3	2	$33.\overline{3}$	50.0	$66.\overline{6}$	$83.\overline{3}$	33
4	3	25.0	37.5	50.0	62.5	50
5	4	20.0	30.0	40.0	50.0	40
6…14	5…13	…	…	…	…	
15	14	$6.\overline{6}$	10.0	$13.\overline{3}$	$16.\overline{6}$	$46.\overline{6}$
16	15	6.25	9.38	12.5	18.75	50

Note: Note that the entire range between two f_{vco} columns in the above table is allowed.

Note: For divider factors that cause duty cycles far from 50% not only the cycle time has to be checked, but also the minimum clock pulse width.

Oscillator Run Detection

Oscillator Run Detection monitors the incoming clock from the oscillator and determines whether it is suitable for an operation in Normal Mode with the selected setting for the N-Divider. Only incoming frequencies that are too low to enable a stable operation of the VCO circuit are detected.

PLL Configuration and Status Registers

The PLL Configuration and Status Registers hold the hardware configuration bits of the PLL, and provide the control for the N, P and K-Dividers as well as the PLL status information.

The clock generation path is selected via the PLL control register SCU_PLLCON.

2.3.2.3 Clock Control Unit

The Clock Control Unit (CCU) receives the clock that is created by the PLL f_{PLL}. In Normal Mode the PLL output frequency f_{PLL} is always used directly as the system clock f_{SYS}.

2.4 Power Supply System

The power supply system is selected such that it offers maximum flexibility and requires a minimum of pins and system integration cost.

Features:

- 5 V supply for the ADC
- 5 V or 3.3 V supply for the I/O pads

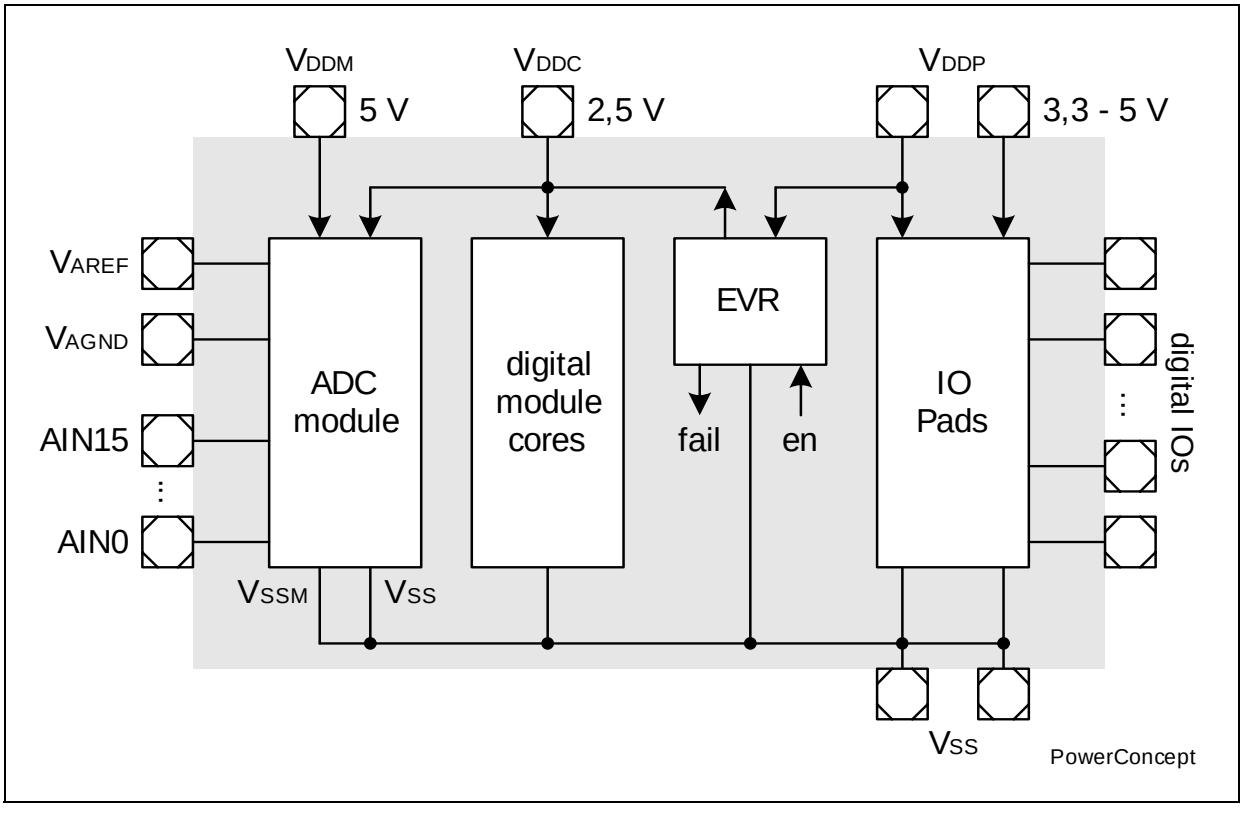


Figure 2-4 CIC751 Power Supply System

2.4.1 Embedded Voltage Regulator

The Embedded Validated Voltage Regulator (EVR) is used for the reduction of supply interfaces between PCB and CIC751. In addition to the I/O voltage VDDP, the voltage supply for the core VDD is necessary. The handling of two different supply voltages can have a large impact on application board design. Thus, it is highly appreciated to provide the on-chip voltage by an on-chip voltage regulator. This embedded voltage regulator further helps reduce the power consumption of the entire chip.

2.5 Event Control

Events or interrupts are generated towards the system by the ADC, SSC, MLI, DMA, and pins. In this chip, the term event is used because the term interrupt is normally linked with the interruption of a code execution but a code executing unit is not present within the CIC751.

2.5.1 Event Sources

There are 13 event sources available for the CIC751.

- ADC event 0; injection conversion interrupt of the ADC module
- ADC event 1; standard conversion interrupt of the ADC module
- ADC event 2; the OR-combination of all valid bits of the ADC_RESBn registers
- Doorbell event 0 that becomes active if the channel number written to INRES equals DBCTR.COMP0
- Doorbell event 1 that becomes active if the channel number written to INRES equals DBCTR.COMP1
- Doorbell event 2 that monitors the valid bit ADC_RESBn.V of the result register selected by DBCTR.COMP0
- Doorbell event 3 that monitors the valid bit ADC_RESBn.V of the result register selected by DBCTR.COMP1
- MLI Request 0 of the MLI module
- MLI Request 1 of the MLI module
- MLI Request 2 of the MLI module
- MLI Request 3 of the MLI module
- External trigger Input 0
- External trigger Input 1

2.5.2 External Trigger Inputs

The device supports external trigger sources to start ADC conversions or DMA transfers. The device has several input pins (SR0 - 2 for MLI Mode and SR0 - 4 for SSC Mode) capable of delivering a trigger input signal.

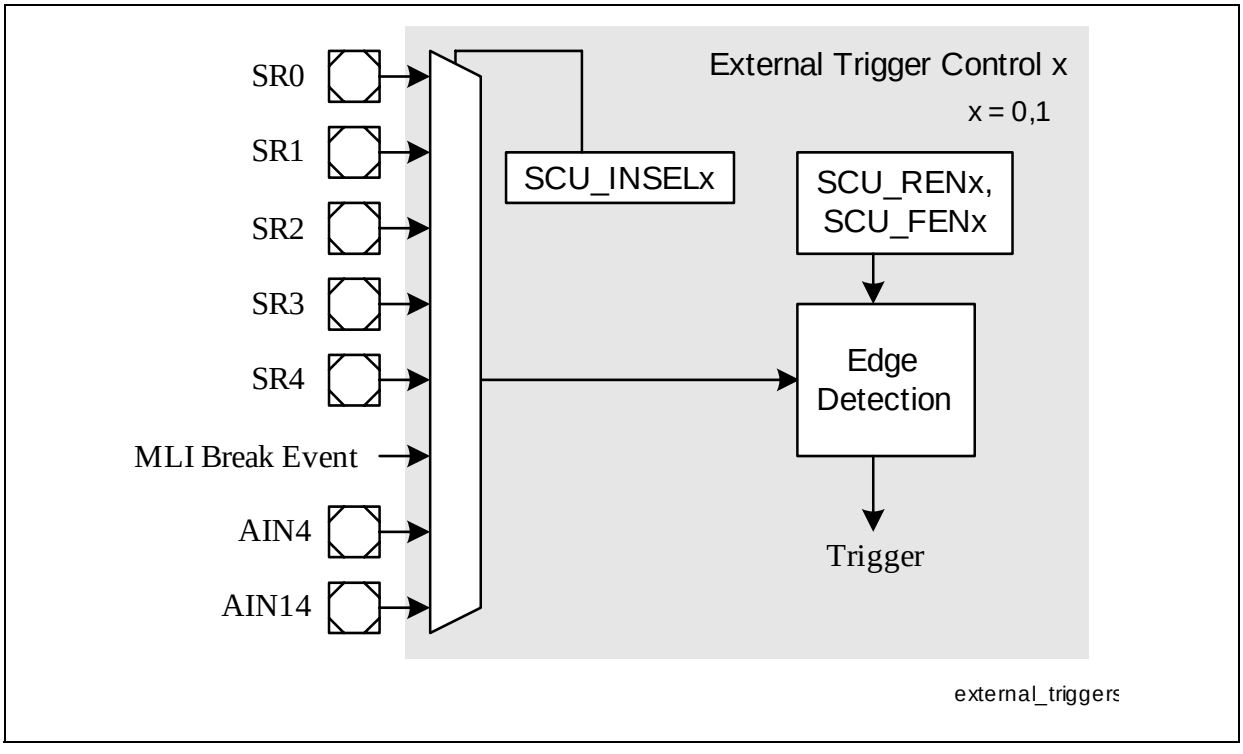


Figure 2-5 External Triggers

The rising edge and falling edge sensitivity of the selected input can be enabled individually. If both edge detections are enabled, an external trigger event is generated upon each change of the signal level (rising edge or falling edge).

The external trigger control register SCU_ETCTR contains the bits defining the behavior of the external trigger inputs.

2.5.3 Event Output Structure

The CIC751 allows output of internal status or notification events on output pins. In order to support different applications and pin usage, internal events are generated. These events are then distributed to the service request pins SRn (n = 0…5).

The following status events can be selected as the source for an output of an SRn pin:

- Doorbell event 2
- Doorbell event 3
- ADC event 2; the OR-combination of all valid bits of the ADC_RESBn registers
- ADC event 0
- ADC event 1

2.5.3.1 Service Request Routing

The service request routing allows the user to combine the various events as output for the pins SRx. The alternative data outputs of the SRx pins are connected as shown in **Figure 2-6**.

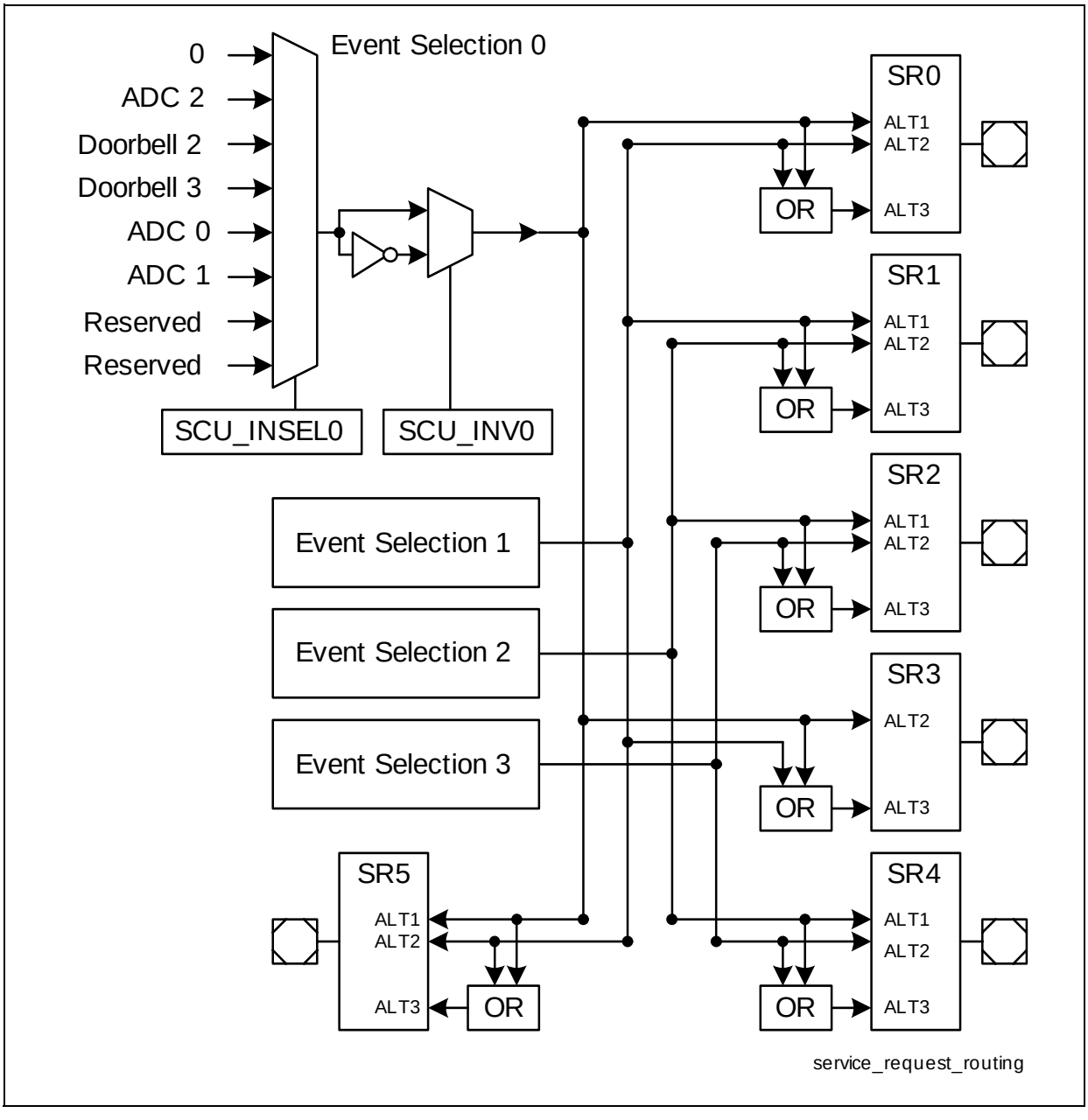


Figure 2-6 Service Request Routing

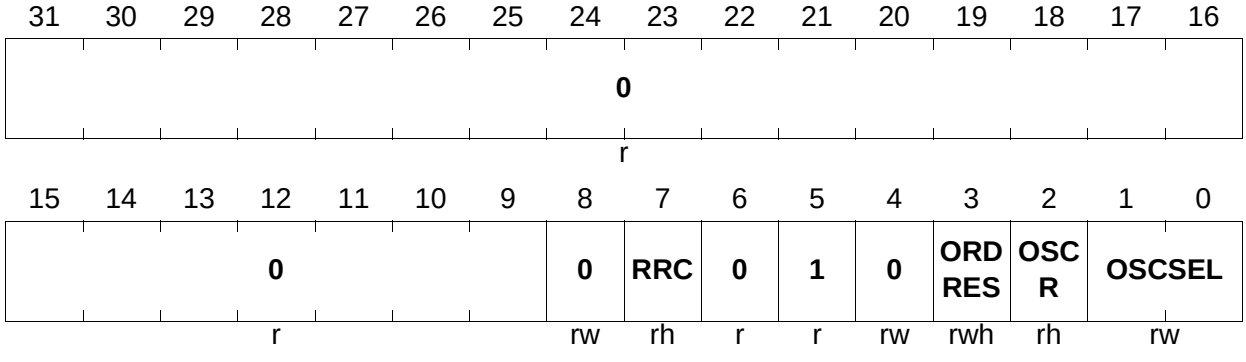
2.6 SCU Registers

2.6.1 Clock Control Registers

The following register controls the clock system of the CIC751.

SCU_OSCCON

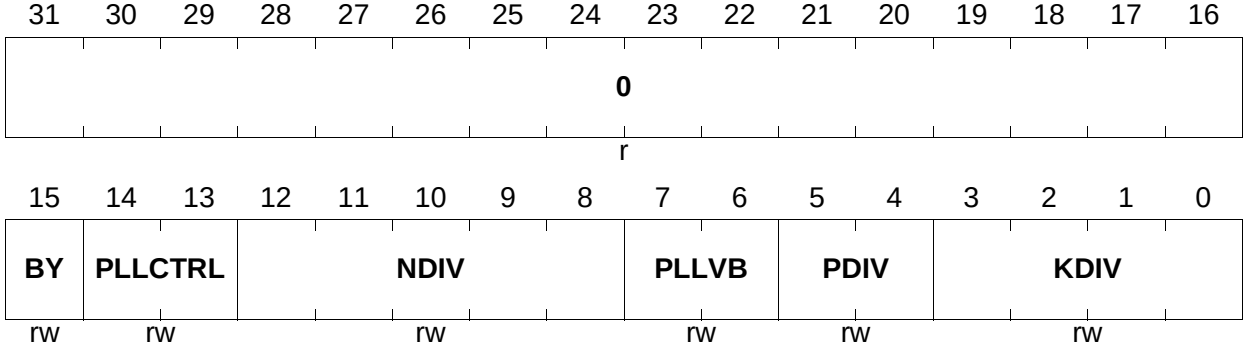
SCU Oscillator Control Register (800_H) Reset Value: $0000\ 0020_H$



Field	Bits	Type	Description
OSCSEL	[1:0]	rw	**Oscillator Select Configuration** This bit field selects the oscillator or clock input for the PLL 00 The RC oscillator is used 01 Reserved, do not use 10 RCLK is directly used 11 RCLK is directly used (same setting as 10_B)
OSCR	2	rh	**Oscillator Run Status Bit** This bit shows the state of the oscillator run state. 0 The oscillator is not running. 1 The oscillator is running.
ORDRES	3	rwh	**Oscillator Run Detection Reset** 0 No operation 1 The oscillator run detection logic is reset and restarted. When set, this bit is automatically cleared.
RRCOSC	7	rh	**RC Oscillator Status** 0 Nominal bias voltages is not reached 1 Nominal bias voltages is reached
0	4, 8	rw	**Reserved;** Read as 0; should be written with 0.

Field	Bits	Type	Description
1	5	r	**Reserved;** Read as 1; should be written with 1.
0	6, [31:9]	r	**Reserved;** Read as 0; should be written with 0.

SCU_PLLCON
SCU PLL Control Register **(804_H)** **Reset Value: 0000 6B02$_H$**



Field	Bits	Type	Description
KDIV	[3:0]	rw	**PLL K-Divider** Scales the PLL output frequency to the desired CPU frequency. $f_{PLL} = f_{VCO} / (KDIV+1)$
PDIV	[5:4]	rw	**PLL P-Divider** Adjusts the oscillator frequency to the defined input frequency range of the PLL $f_{IN} = f_{OSC} / (PDIV+1)$ Valid values: 11_B...00_B
PLLVB	[7:6]	rw	**PLL VCO Band Select** ValueVCO output frequencyBase frequency 00 100…150 MHz 20…80 MHz 01 150…200 MHz 40…130 MHz 10 200…250 MHz 60…180 MHz 11Reserved[1)]
NDIV	[12:8]	rw	**PLL N-Divider** ...by which the PLL multiplies its input frequency $f_{VCO} = f_{IN} * (NDIV+1)$ Valid values: 11111_B...00111_B

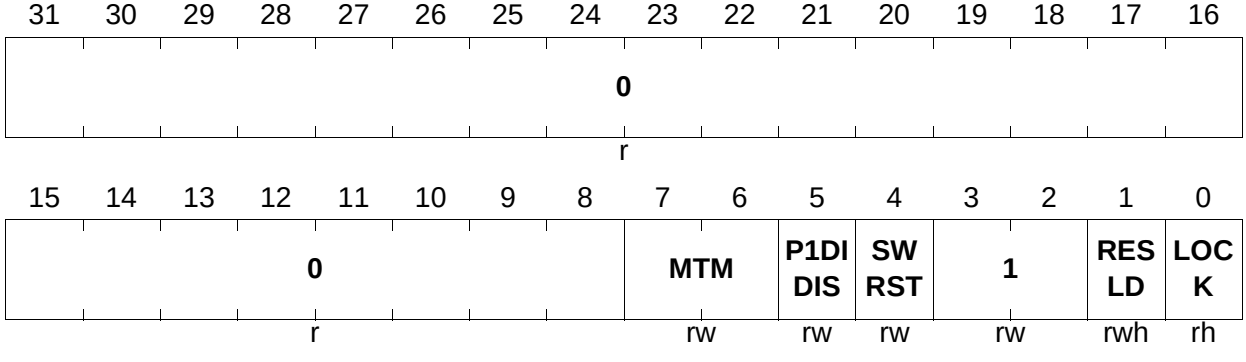
Field	Bits	Type	Description
PLLCTRL	[14:13]	rw	**PLL Operation Control** 00 Bypass PLL clock mult., the VCO is off; Prescaler Mode 01 Reserved, do not use this combination 10 VCO clock used, input clock switched off; Freerunning Mode 11 VCO clock used, input clock connected; Normal Mode
BY	15	rw	**PLL Bypass Control** 0 PLL operates as defined by bit field CTRL 1 PLL operates in Bypass Mode
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

1) Operation in the upper VCO band cannot be guaranteed because of a possible malfunction of the K-divider.

2.6.2 Miscellaneous SCU Registers

SCU_SYSCON

SCU System Control Register (820_H) Reset Value: 0000 000C$_H$



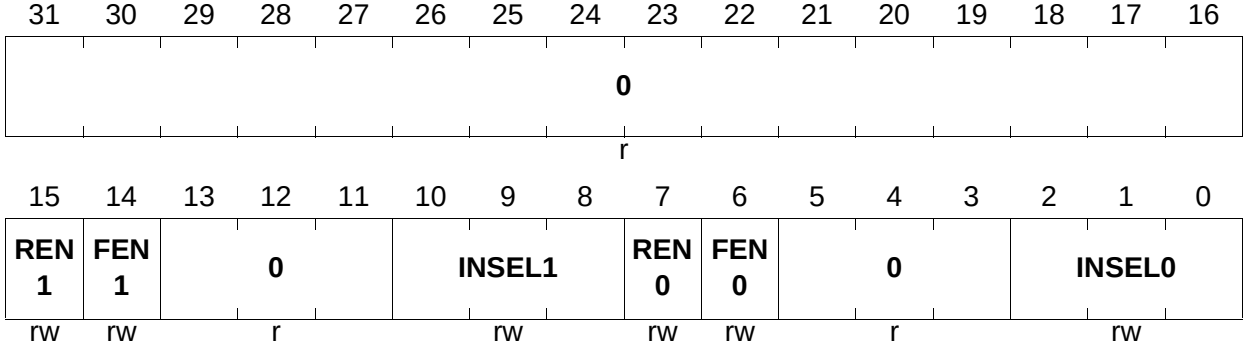
Field	Bits	Type	Description
LOCK	0	rh	**PLL Lock Status Flag** 0 PLL is not locked 1 PLL is locked

Field	Bits	Type	Description
RESLD	1	rwh	**Restart Lock Detection** Setting this bit will reset bit LOCK and restart the lock detection. When set, this bit is automatically cleared. 0 No effect 1 Reset LOCK and restart lock detection
SWRST	4	rw	**Software Reset Trigger** Setting this bit will automatically request and generate a reset. With the reset execution, this bit is automatically cleared.
P1DIDIS	5	rw	**Port 1 Digital Input Disable** This bit controls the digital input stage for all port 1 pins. 0 Digital input stage (Schmitt-trigger) is enabled 1 Digital input stage (Schmitt-trigger) is disabled. This is necessary if pins are used as analog input.
MTM	[7:6]	rw	**Multiplexer Test Mode for Channel 0** This bit enables/disables the Multiplexer Test Mode for the input channel 0. This feature is independent of the current mode of the analog part. If the Multiplexer Test Mode is enabled, the analog input is connected to ADC ground via an internal resistance[1)]. This structure creates a voltage divider to ground, so the measurement result becomes smaller. 00 The Multiplexer Test Mode is disabled. The analog input is not connected to ground and can be used for normal measurements. 01 The Multiplexer Test Mode is enabled. The internal resistance to ground is in the range of 300 Ohm. 10 The Multiplexer Test Mode is enabled. The internal resistance to ground is in the range of 70 Ohm. 11 Reserved, like 00
1	[3:2]	rw	**Reserved;** Should be written with 1.
0	[15:8]	r	**Reserved;** Read as 0; should be written with 0.

1) Please refer to the ACDC chapter for the current capability of the grounding resistor, especially when using RC input filters at the analog inputs.

SCU_ETCTR

SCU External Trigger Control Register(850_H) **Reset Value: $0000\ 0000_H$**

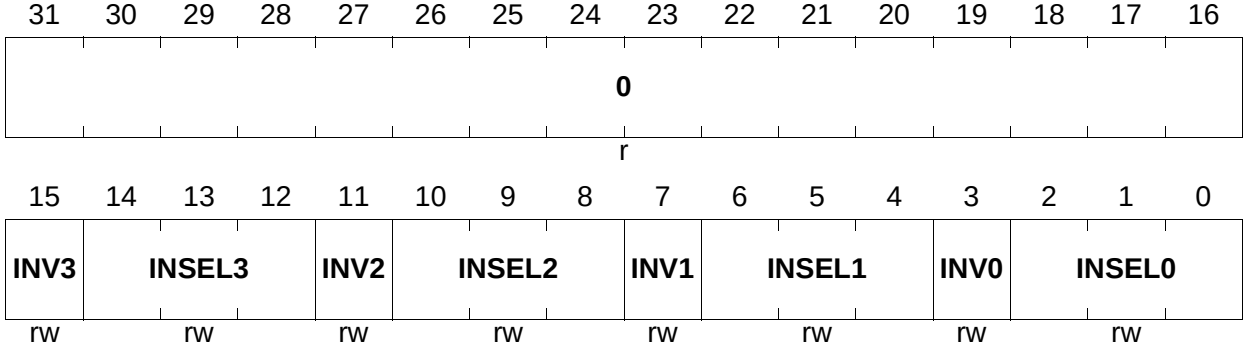


Field	Bits	Type	Description
INSEL0	[2:0]	rw	**External Trigger Input 0 Selection** This bit field defines the source for the external trigger input 0. 000 Input SR0 is selected 001 Input SR1 is selected 010 Input SR2 is selected 011 Input SR3 is selected 100 Input SR4 is selected 101 MLI Break Event is selected 110 Input AIN4 is selected 111 Input AIN14 is selected
FEN0	6	rw	**Falling Edge Enable for External Trigger Input 0** This bit enables/disables the activation of external trigger input 0 upon a falling edge at the selected input. 0 The trigger upon a falling edge is disabled 1 The trigger upon a falling edge is enabled
REN0	7	rw	**Rising Edge Enable for External Trigger Input 0** This bit enables/disables the activation of external trigger input 0 upon a rising edge at the selected input. 0 The trigger upon a rising edge is disabled 1 The trigger upon a rising edge is enabled

Field	Bits	Type	Description
INSEL1	[10:8]	rw	**External Trigger Input 1 Selection** This bit field defines the source for the external trigger input 1. 000 Input SR0 is selected 001 Input SR1 is selected 010 Input SR2 is selected 011 Input SR3 is selected 100 Input SR4 is selected 101 MLI Break Event is selected 110 Input AIN4 is selected 111 Input AIN14 is selected
FEN1	14	rw	**Falling Edge Enable for External Trigger Input 1** This bit enables/disables the activation of external trigger input 1 upon a falling edge at the selected input. 0 The trigger upon a falling edge is disabled 1 The trigger upon a falling edge is enabled
REN1	15	rw	**Rising Edge Enable for External Trigger Input 1** This bit enables/disables the activation of external trigger input 1 upon a rising edge at the selected input. 0 The trigger upon a rising edge is disabled 1 The trigger upon a rising edge is enabled
0	[5:3], [13:11] [31:16]	r	**Reserved;** Read as 0; should be written with 0.

SCU_SRCR

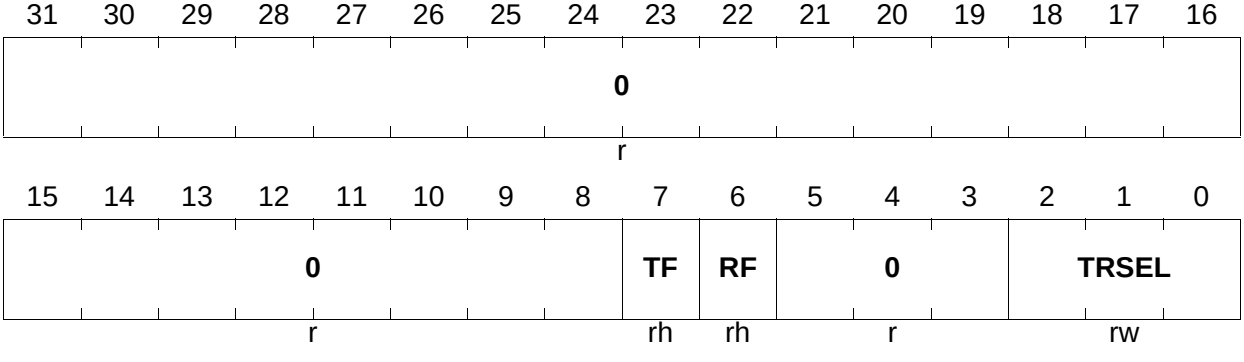
SCU Service Request Control Register(858_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
INSEL0	[2:0]	rw	**Input Selection for Event 0** 000 No Event is generated 001 ADC event 2 is used as source 010 Doorbell event 2 is used as source 011 Doorbell event 3 is used as source 100 ADC event 0 is used as source 101 ADC event 1 is used as source 110 Reserved, do not use this combination 111 Reserved, do not use this combination
INV0	3	rw	**Invert Source for Event 0** 0 The source is not inverted 1 The source is inverted
INSEL1	[6:4]	rw	**Input Selection for Event 1** 000 No Event is generated 001 ADC event 2 is used as source 010 Doorbell event 2 is used as source 011 Doorbell event 3 is used as source 100 ADC event 0 is used as source 101 ADC event 1 is used as source 110 Reserved, do not use this combination 111 Reserved, do not use this combination
INV1	7	rw	**Invert Source for Event 1** 0 The source is not inverted 1 The source is inverted
INSEL2	[10:8]	rw	**Input Selection for Event 2** 000 No Event is generated 001 ADC event 2 is used as source 010 Doorbell event 2 is used as source 011 Doorbell event 3 is used as source 100 ADC event 0 is used as source 101 ADC event 1 is used as source 110 Reserved, do not use this combination 111 Reserved, do not use this combination
INV2	11	rw	**Invert Source for Event 2** 0 The source is not inverted 1 The source is inverted

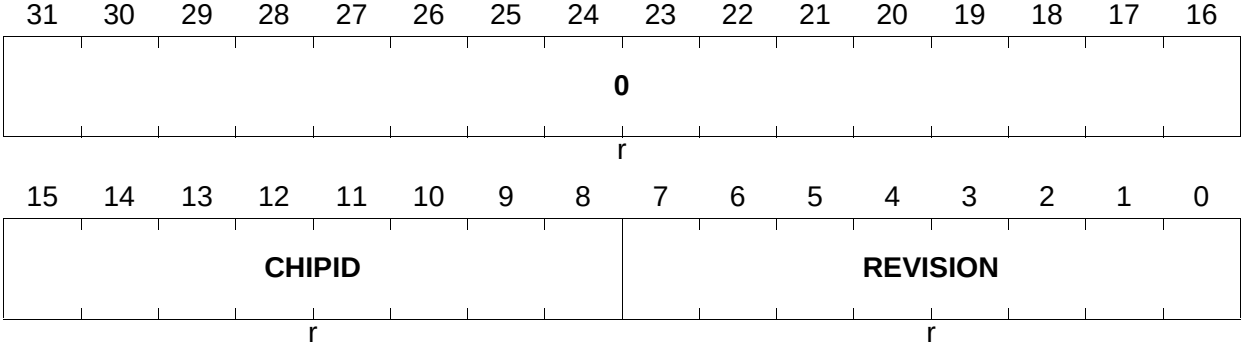
Field	Bits	Type	Description
INSEL3	[14:12]	rw	**Input Selection for Event 3** 000 No Event is generated 001 ADC event 2 is used as source 010 Doorbell event 2 is used as source 011 Doorbell event 3 is used as source 100 ADC event 0 is used as source 101 ADC event 1 is used as source 110 Reserved, do not use this combination 111 Reserved, do not use this combination
INV3	15	rw	**Invert Source for Event 3** 0 The source is not inverted 1 The source is inverted
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

SCU_CHTR0
SCU Channel Trigger 0 Register **(830_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR1
SCU Channel Trigger 1 Register **(834_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR2
SCU Channel Trigger 2 Register **(838_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR3
SCU Channel Trigger 3 Register **($83C_H$)** **Reset Value: 0000 0000$_H$**
SCU_CHTR4
SCU Channel Trigger 4 Register **(840_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR5
SCU Channel Trigger 5 Register **(844_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR6
SCU Channel Trigger 6 Register **(848_H)** **Reset Value: 0000 0000$_H$**
SCU_CHTR7
SCU Channel Trigger 7 Register **($84C_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TRSEL	[2:0]	rw	**Trigger Selection** This bit field defines the trigger source for the DMA channel n. 000 A constant 0 is selected. TF and RF are cleared. 001 ADC event 0 is selected as trigger source 010 ADC event 1 is selected as trigger source 011 Doorbell event 0 is selected as trigger source 100 Doorbell event 1 is selected as trigger source 101 External trigger input 0 is selected as trigger source 110 External trigger input 0 is selected as trigger source 111 Reserved, do not use this combination
RF	6	rh	**Ready Flag** This bit indicates if the MLI is ready for the next transfer. 0 The MLI is not yet ready for a new transfer (former transfer not yet finished) 1 The MLI is ready for a new transfer (former transfer is finished)
TF	7	rh	**Trigger Flag** This bit indicates that a channel trigger request is pending. 0 No channel trigger request is pending 1 A channel trigger request is pending
0	[5:3], [31:8]	r	**Reserved;** Read as 0; should be written with 0.

IDCHIP
Chip Identification Register (860_H) Reset Value: 0000 8EXX$_H$



Field	Bits	Type	Description
REVISION	[7:0]	r	**Device Revision Code** Identifies the device step.
CHIPID	[15:8]	r	**Device Identification** The value $8E_H$ identifies the device as CIC751.
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

2.7 SCU Register Overview

Table 2-5 SCU Registers

Register Short Name	Register Long Name	Address	Description see
SCU_OSCCON	SCU Oscillator Control Register	800_H	Page 2-14
SCU_PLLCON	SCU PLL Control Register	804_H	Page 2-15
SCU_SYSCON	SCU System Control Register	820_H	Page 2-16
SCU_ETCTR	SCU External Trigger Control Register	850_H	Page 2-18
SCU_SRCR	SCU Service Request Control Register	858_H	Page 2-19
SCU_CHTR0	SCU Channel Trigger 0 Register	830_H	Page 2-22
SCU_CHTR1	SCU Channel Trigger 1 Register	834_H	Page 2-22
SCU_CHTR2	SCU Channel Trigger 2 Register	838_H	Page 2-22
SCU_CHTR3	SCU Channel Trigger 3 Register	$83C_H$	Page 2-22

Table 2-5 SCU Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
SCU_CHTR4	SCU Channel Trigger 4 Register	840_H	Page 2-22
SCU_CHTR5	SCU Channel Trigger 5 Register	844_H	Page 2-22
SCU_CHTR6	SCU Channel Trigger 6 Register	848_H	Page 2-22
SCU_CHTR7	SCU Channel Trigger 7 Register	$84C_H$	Page 2-22
IDCHIP	Chip Identification Register	860_H	Page 2-24

3 Direct Memory Access Controller

This chapter describes the Direct Memory Access (DMA) Controller of the CIC751.

3.1 DMA Request Generation and Control

This section describes how a DMA Move / Transfer / Transaction is requested. The differnet request sources can be controlled to support the adaption for the required application.

3.1.1 Request Generation

Requests that trigger a DMA Transaction can be generated in several ways. This flexible request generation mechanism enables the software to configure the hardware to the exact needs of the application. After configuration, the DMA handles all requests without further software requirements. Each of the eight channels can be requested by one of six possible requests. The following request sources can trigger a DMA Transfer of a channel:

- MLI Request 0 (indirect)
- MLI Request 1 (indirect)
- MLI Request 2 (direct)
- MLI Request 3 (direct)
- ADC event 0 (indirect)
- ADC event 1 (indirect)
- Doorbell event 0 (indirect)
- Doorbell event 1 (indirect)
- Channel 0 Request (direct)
- Channel 1 Request (direct)
- Channel 2 Request (direct)
- Channel 3 Request (direct)
- Channel 4 Request (direct)
- Channel 5 Request (direct)
- Channel 6 Request (direct)
- Channel 7 Request (direct)

There are two classes of requests that are connected to the DMA; direct and indirect. Indirect requests need to be preselected on a system level in order to be mapped to the two additional direct requests

- Set Trigger Flag Request (direct)
- Trigger AND Ready Flag Request (direct)

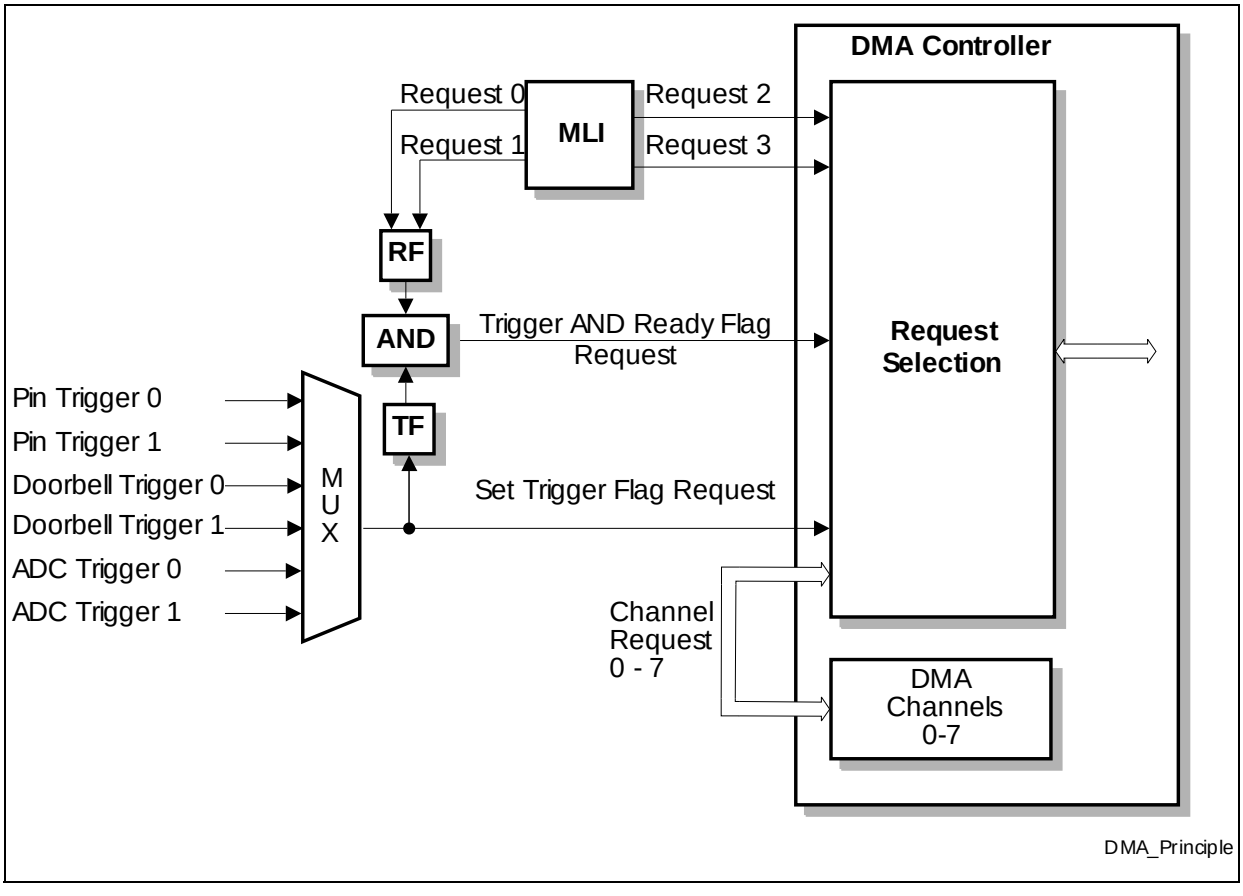


Figure 3-1 DMA Request Principle

3.1.1.1 Preselection of the Indirect Requests

There are two options for the requests.

Set Trigger Flag Request

The following requests can be mapped to the Set Trigger Request.

Pins SR0, SR1, SR2, SR3, SR4, AIN4, and AIN14

MLI Break Event

These eight sources are combined into two Pin Trigger Request sources at a first level. These eight sources represent all possible external Request Triggers (for more information about the MLI Break Event see **Chapter 4.2.1.5**). Which of the eight possible trigger sources is used can be configured via SCU_ETCTR.INSEL0 for the Pin Trigger Request 0 and SCU_ETCTR.INSEL1 for the Pin Trigger Request 1. An edge detection activates the trigger signal upon a event that is configurable via ETCTR.FENx and ETCTR.RENx.

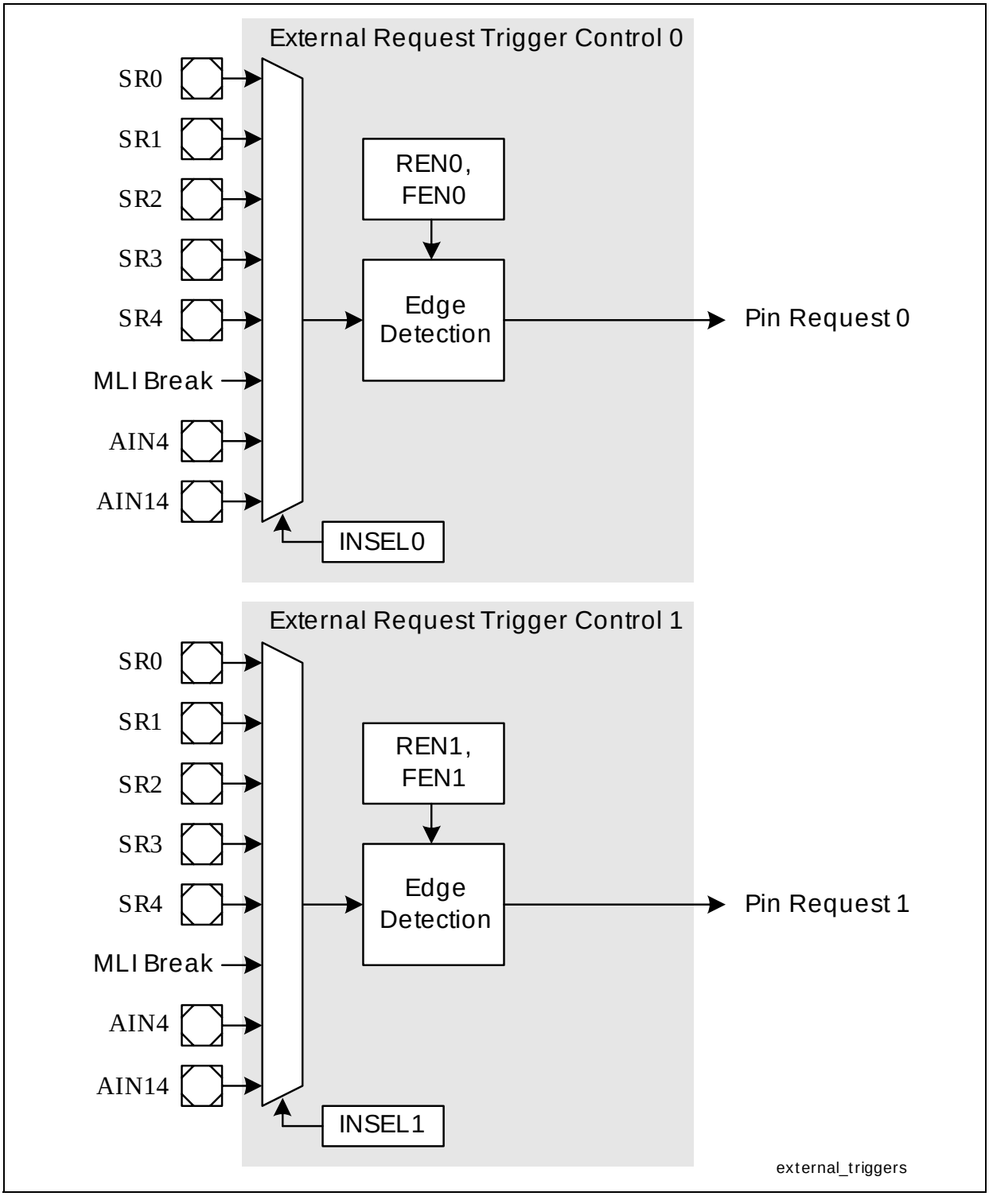


Figure 3-2 External Trigger Unit

The rising edge and falling edge sensitivity of the selected input can be enabled individually. If edge detection for both edges is enabled, a trigger signal is generated upon each change of the signal level (rising edge or falling edge).

Note: For MLI Mode, only the pins SR0, SR1, and SR2 are available as Request Trigger sources.

Trigger AND Ready Flag Request

This combined request enables the MLI interface and the ADC to quickly establish communication with the help of the DMA and minimal software requirements.

The Set Trigger Flag part signals that a new ADC conversion result is available. This can be from a standard conversion (indicated via ADC event 0) or from a injected conversion (indicated via ADC event 1). If the ADC conversion result was stored in the ADC extended result registers, the doorbell mechanism can be used (indicated via Doorbell event 0 or Doorbell event 1) for the communication.

Note: The Doorbell mechanism requires the use of at least one additional channel.

Typical Use Case Example:

This is combined with the MLI Request trigger 0 and 1. Both MLI Request triggers can be used to indicate when the MLI interface is ready to send the next data (ADC conversion result) to the host controller. Therefore, the combination indicates that a new ADC conversion result is available and the MLI interface is ready to transmit the result to the host controller.

3.1.2 DMA Request Assignment Matrix

The DMA requests input lines of the DMA are assigned as indicated in **Table 3-1**:

Table 3-1 DMA Request Assignment

DMA Channel	DMA Request Input	Selected by
0	Channel 7 Request	DMA_CHCR0.PRSEL = 000_B
	Channel 6 Request	DMA_CHCR0.PRSEL = 001_B
	MLI Request 2	DMA_CHCR0.PRSEL = 010_B
	MLI Request 3	DMA_CHCR0.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR0.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR0.PRSEL = 101_B
	not used, no Request	DMA_CHCR0.PRSEL = 110_B
		DMA_CHCR0.PRSEL = 111_B

Table 3-1 DMA Request Assignment (cont'd)

DMA Channel	DMA Request Input	Selected by
1	Channel 0 Request	DMA_CHCR1.PRSEL = 000_B
	Channel 7 Request	DMA_CHCR1.PRSEL = 001_B
	MLI Request 2	DMA_CHCR1.PRSEL = 010_B
	MLI Request 3	DMA_CHCR1.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR1.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR1.PRSEL = 101_B
	not used, no Request	DMA_CHCR1.PRSEL = 110_B
		DMA_CHCR1.PRSEL = 111_B
2	Channel 1 Request	DMA_CHCR2.PRSEL = 000_B
	Channel 0 Request	DMA_CHCR2.PRSEL = 001_B
	MLI Request 2	DMA_CHCR2.PRSEL = 010_B
	MLI Request 3	DMA_CHCR2.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR2.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR2.PRSEL = 101_B
	not used, no Request	DMA_CHCR2.PRSEL = 110_B
		DMA_CHCR2.PRSEL = 111_B
3	Channel 2 Request	DMA_CHCR3.PRSEL = 000_B
	Channel 1 Request	DMA_CHCR3.PRSEL = 001_B
	MLI Request 2	DMA_CHCR3.PRSEL = 010_B
	MLI Request 3	DMA_CHCR3.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR3.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR3.PRSEL = 101_B
	not used, no Request	DMA_CHCR3.PRSEL = 110_B
		DMA_CHCR3.PRSEL = 111_B

Table 3-1 DMA Request Assignment (cont'd)

DMA Channel	DMA Request Input	Selected by
4	Channel 3 Request	DMA_CHCR4.PRSEL = 000_B
	Channel 2 Request	DMA_CHCR4.PRSEL = 001_B
	MLI Request 2	DMA_CHCR4.PRSEL = 010_B
	MLI Request 3	DMA_CHCR4.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR4.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR4.PRSEL = 101_B
	not used, no Request	DMA_CHCR4.PRSEL = 110_B
		DMA_CHCR4.PRSEL = 111_B
5	Channel 4 Request	DMA_CHCR5.PRSEL = 000_B
	Channel 3 Request	DMA_CHCR5.PRSEL = 001_B
	MLI Request 2	DMA_CHCR5.PRSEL = 010_B
	MLI Request 3	DMA_CHCR5.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR5.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR5.PRSEL = 101_B
	not used, no Request	DMA_CHCR5.PRSEL = 110_B
		DMA_CHCR5.PRSEL = 111_B
6	Channel 5 Request	DMA_CHCR6.PRSEL = 000_B
	Channel 4 Request	DMA_CHCR6.PRSEL = 001_B
	MLI Request 2	DMA_CHCR6.PRSEL = 010_B
	MLI Request 3	DMA_CHCR6.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR6.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR6.PRSEL = 101_B
	not used, no Request	DMA_CHCR6.PRSEL = 110_B
		DMA_CHCR6.PRSEL = 111_B

Table 3-1 DMA Request Assignment (cont'd)

DMA Channel	DMA Request Input	Selected by
7	Channel 6 Request	DMA_CHCR7.PRSEL = 000_B
	Channel 5 Request	DMA_CHCR7.PRSEL = 001_B
	MLI Request 2	DMA_CHCR7.PRSEL = 010_B
	MLI Request 3	DMA_CHCR7.PRSEL = 011_B
	Trigger AND Ready Request	DMA_CHCR7.PRSEL = 100_B
	Set Trigger Flag Request	DMA_CHCR7.PRSEL = 101_B
	not used, no Request	DMA_CHCR7.PRSEL = 110_B
		DMA_CHCR7.PRSEL = 111_B

Note: Not all channel are connected to the exactly same direct channel Request.

3.2 DMA Controller Kernel Description

The DMA Controller of the CIC751 transfers data from data source locations to data destination locations without intervention of other on-chip devices. One data move operation is controlled by one DMA channel. Eight DMA channels are provided in one DMA Sub-Block.

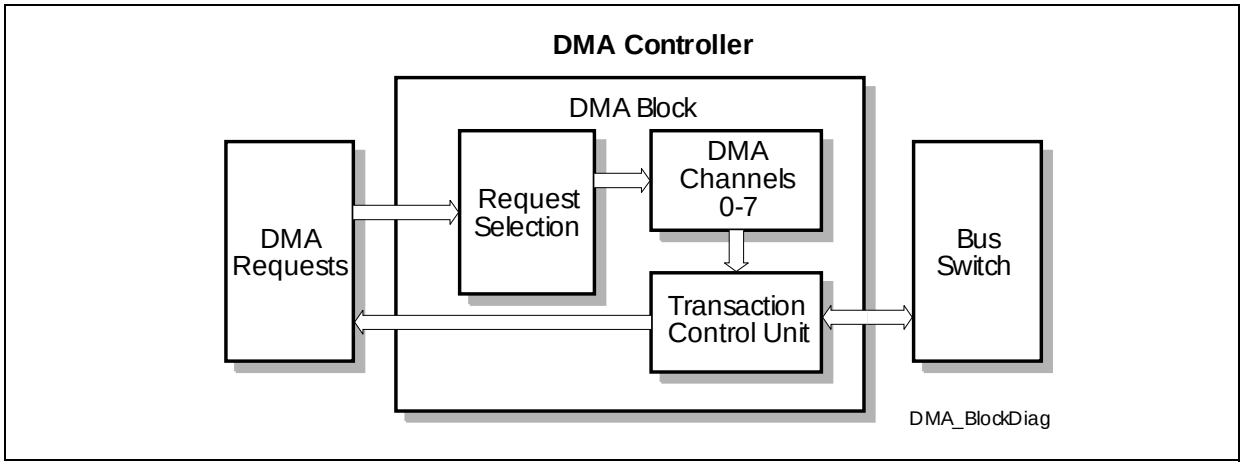


Figure 3-3 DMA Block Diagram

3.2.1 Features

The DMA controller has the following features:

- 8 independent DMA channels
  - 8 DMA channels in the DMA Sub-Block
  - Up to 8 selectable request inputs per DMA channel
  - 2-level programmable priority of DMA channels within the DMA Sub-Block
  - Software and hardware DMA request
  - Hardware requests by selected on-chip peripherals and external inputs
- Buffer capability for move actions on the buses (at least 1 move per bus is buffered)
- Individually programmable operation modes for each DMA channel
  - Single Mode: stops and disables DMA channel after a predefined number of DMA transfers
  - Continuous Mode: DMA channel remains enabled after a predefined number of DMA transfers; DMA transaction can be repeated
  - Programmable address modification
- Full 32-bit addressing capability of each DMA channel
  - 4 Gbyte address range
  - Support of circular buffer addressing mode
- Programmable data width of DMA transfer/transaction: 8-bit, 16-bit, or 32-bit
- Register set for each DMA channel
  - Source and destination address register
  - Channel control and status register
  - Transfer count register

3.2.2 Definition of Terms

Some basic terms must be defined for the functional description of the DMA controller.

DMA Move

A DMA move is an operation that always consists of two parts:

1. A read move that loads data from a data source into the DMA controller
2. A write move that puts data from the DMA controller to a data destination

Within a DMA move, data is always moved from the data source via the DMA controller to the data destination. Data is temporarily stored in the DMA controller. The data widths of read move and write move are always identical (8-bit, 16-bit or 32-bit). Data assembly or disassembly is not supported.

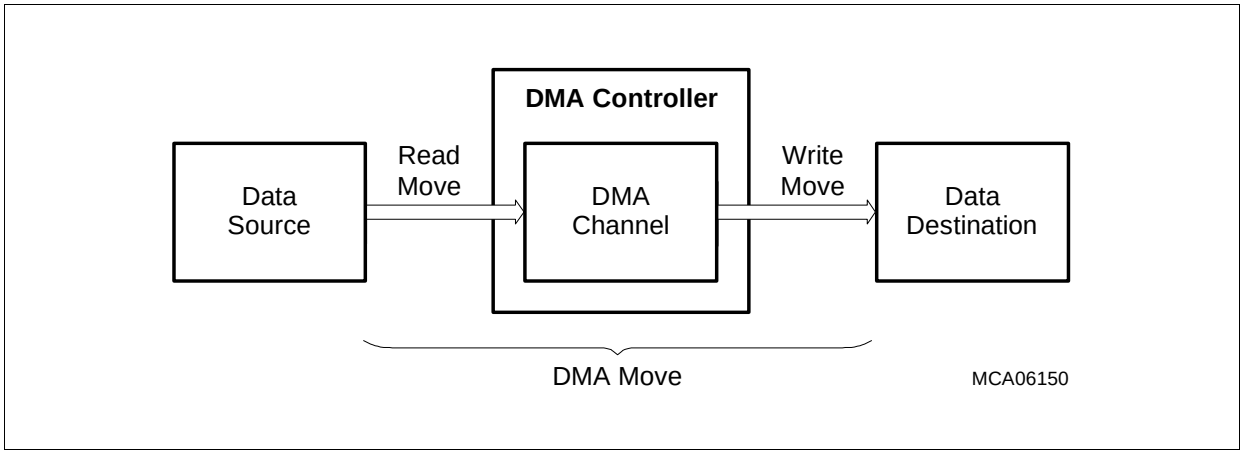


Figure 3-4 DMA Definition of Terms

DMA Transfer

A DMA transfer can be composed of 1, 2, 4, 8 or 16 DMA moves.

DMA Transaction

A DMA transaction is composed of several (at least one) DMA transfers. The Transfer Count determines the number of DMA transfers within one DMA transaction.

Example:
1024 word (32-bit wide) transactions can be composed of 256 transfers of four DMA word moves, or 128 transfers of eight DMA word moves.

3.2.3 DMA Principles

The DMA controller supports DMA moves from one address location to another one. DMA moves can be requested either by hardware or by software. DMA hardware requests are triggered by specific request lines from the peripheral modules or from other DMA channels. The number of available DMA request lines from a peripheral module varies depending on the module functionality. Typically, the occurrence of a receive or transmit data interrupts in a peripheral module are able to generate a DMA request.

3.2.4 DMA Channel Functionality

Each of the 8 DMA channels has one associated register set containing six 32-bit registers. These registers are numbered by one index to indicate the related DMA channel: Index “n” refers to the channel number (n = 0-7) within the DMA Sub-Block.

Example: CHCR04 is the Control Register of DMA channel 4 in Sub-Block 0.

The register set of a DMA channel register contains the following registers:

- Channel 0n Control Register CHCR0n (for details, see **Page 3-39**)
- Channel 0n Status Register CHSR0n (for details, see **Page 3-42**)
- Channel 0n Address Control Register ADRCR0n (for details, see **Page 3-43**)
- Channel 0n Source Address Register SADR0n (for details, see **Page 3-47**)
- Channel 0n Destination Address Register DADR0n (for details, see **Page 3-48**)
- Channel 0n Shadow Address Register SHADR0n (for details, see **Page 3-49**)

3.2.4.1 Shadowed Source or Destination Address

As a typical application, an SSC module that receives data (fixed source address) has to deliver it to a memory buffer using a DMA transaction (variable destination address). After a certain amount of data has been transferred, a new DMA transaction should be initiated to deliver further SSC data into another memory buffer. While the destination address register is updated during a running DMA transaction with the actual destination address, a shadow mechanism allows programming of a new destination address without disturbing the content of the destination address register. In this case, the new destination address is written into a buffer register, i.e. the shadow address register. At the start of the next DMA transaction, the new address is transferred from this shadow address register to the destination address register.

The shadow address register can be used also to store a source address. However, it cannot store source and destination address at the same time. This means that the shadow mechanism makes it possible to automatically update either a new source address, or a new destination address at the start of a DMA transaction. If both address registers (for source and destination address) have to be updated for the next DMA transaction, a running DMA transaction for this channel must be finished. After that, source and destination address registers can be written before the next DMA transaction is started.

Figure 3-5 shows the actions that take place when a source address register is updated. The update of a destination register happens in an equivalent manner.

When writing a new address to the (address of) the source or destination address register and no DMA transaction is running, the new address value is directly written into the source or destination address register. In this case, no buffering of the address is required. When writing a new address to the (address of) the source or destination address register and a DMA transaction is running, no transfer to an address register can take place and SHADR0n holds the new address value that was written. For this

operation, bit field ADRCR0n.SHCT must be set either to 01_B (address is a source address) or 10_B (new address is a destination address). At the start of the next DMA transaction, the shadow transfer takes place and the content of SHADR0n is written either into SADR0n or DADR0n (ADRCR0n.SHCT must be set accordingly). After the shadow transfer, SHADR0n is set to $0000\ 0000_H$. Therefore, the software can check by reading the shadow address register whether or not the shadow transfer has already taken place.

Only one address register can be shadowed while a transaction is running, because the shadow register can only be assigned either to the source or to the destination address register. Note that the shadow address register transfer has the same behavior in Single and Continuous Mode. When the shadow mechanism is disabled (ADRCR0n.SHCT = 00_B), SHADR0n is always read as $0000\ 0000_H$.

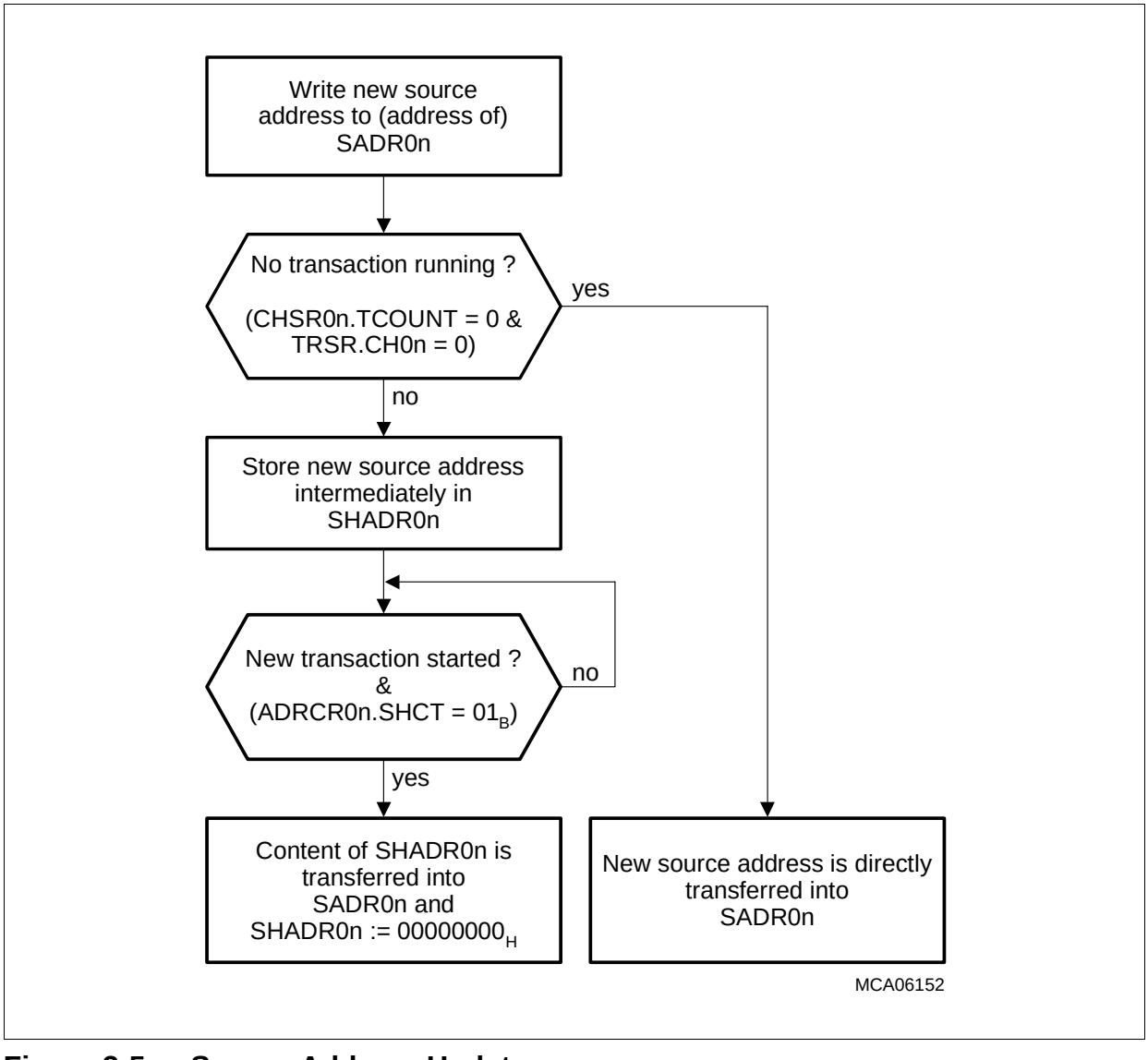


Figure 3-5 Source Address Update

The transfer count of a DMA transaction, stored in bit field CHCR0n.TREL, can also be programmed if the DMA transaction is running. At the start of a DMA transaction, TREL is transferred to bit field CHSR0n.TCOUNT, which is then updated during the DMA transaction.

No reload of address or counter will be done if TCOUNT is not equal to 0.

The reprogramming of channel specific values (except for the selected address shadow register) should be avoided while a DMA channel is active.

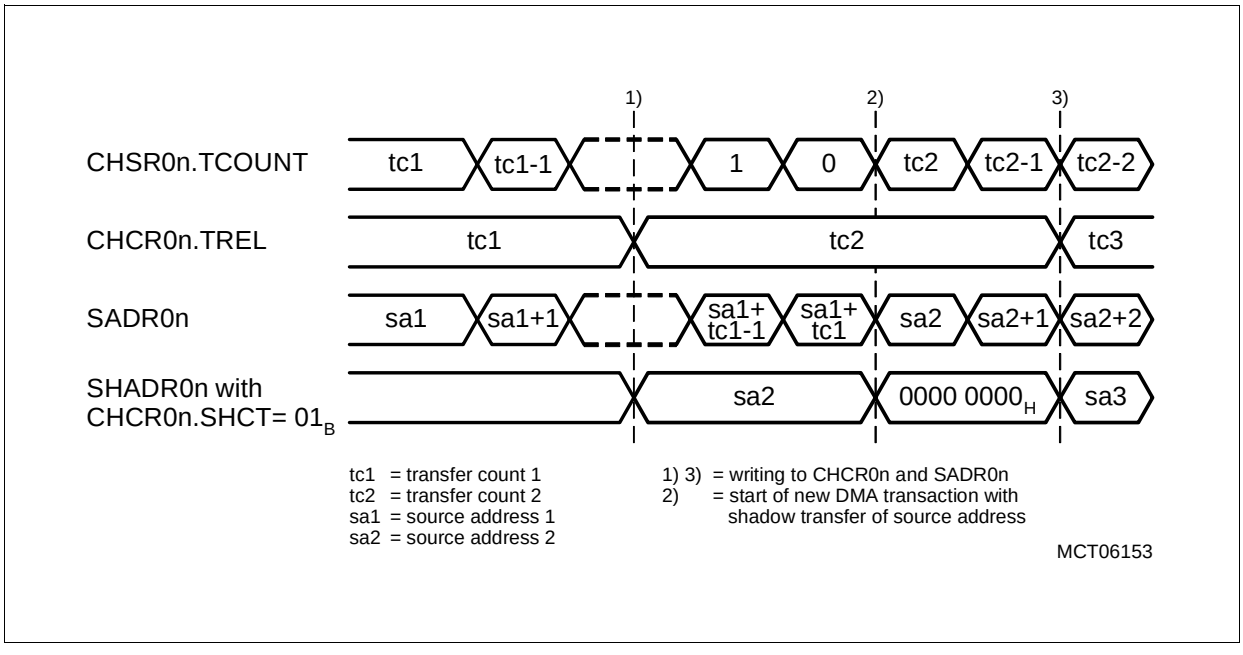


Figure 3-6 Shadow Source Address and Transfer Count Update

Figure 3-6 shows how the contents of the source address register SADR0n and the transfer count CHSR0n.TCOUNT are updated during two DMA transactions with a shadowed source address and transfer count update.

At reference point 2) the DMA transaction 1 is finished and DMA transaction 2 is started. At 1) the DMA channel is reprogrammed with two new parameters for the next DMA transaction: Transfer count tc2 and source address sa2. Source address sa2 is buffered in SADR0n and transferred to SADR0n when the new DMA transaction is started at 2). At this time, transfer count tc2 is also transferred to CHSR0n.TCOUNT.

3.2.4.2 DMA Channel Request Control

Figure 3-7 shows the control logic for DMA requests that is implemented for each DMA channel.

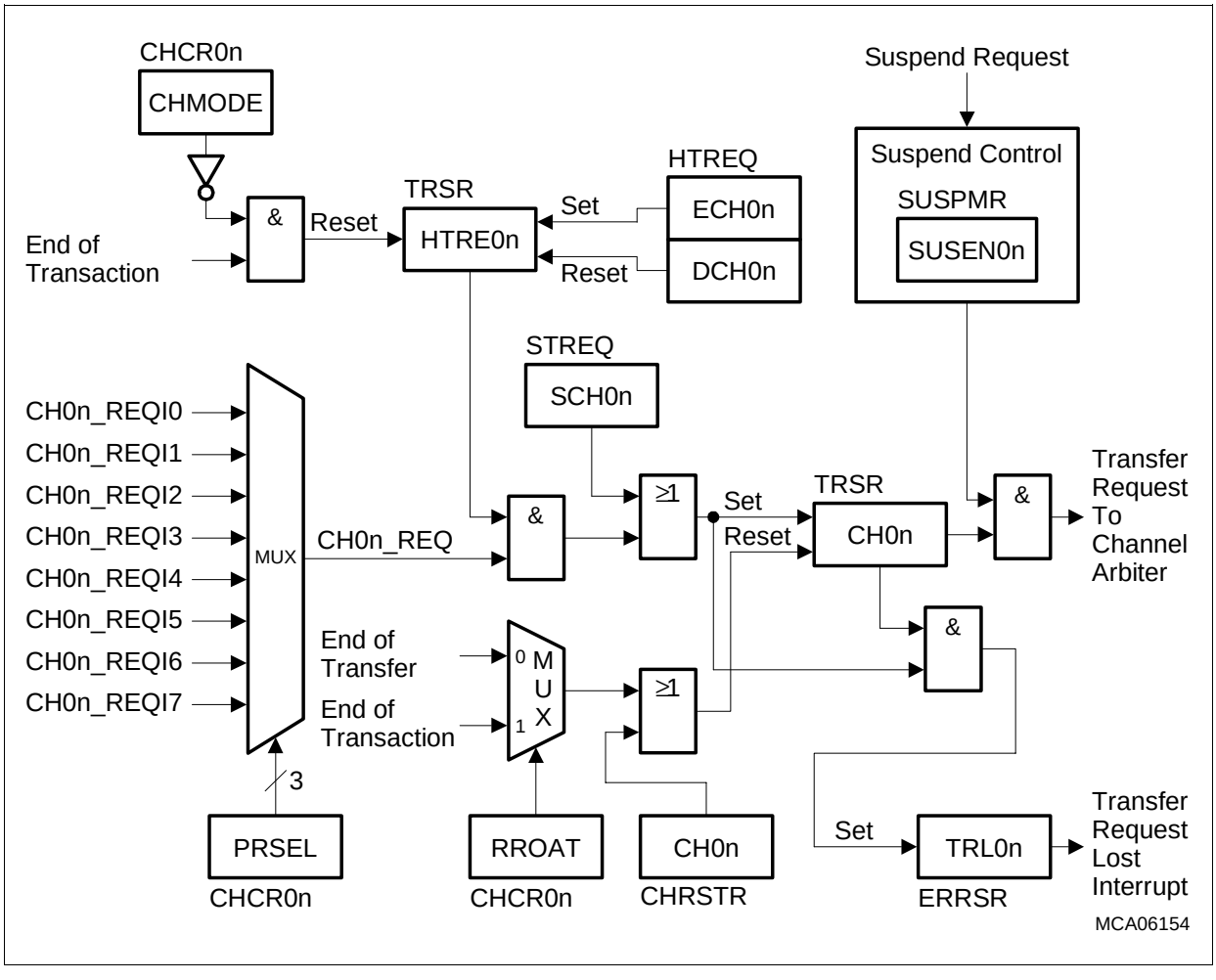


Figure 3-7 Channel Request Control

Two different types of DMA requests are possible:

- Hardware DMA requests
- Software DMA requests

The hardware request CH0n_REQ can be connected to one of eight possible hardware request input lines as selected by bit field CHCR0n.PRSEL. Hardware requests are enabled/disabled by status bit TRSR.HTRE0n. HTRE0n can be set/reset by software or by hardware in Single Mode at the end of a DMA transaction. A software request can be generated by setting bit STREQ.CH0n.

Status flag TRSR.CH0n indicates whether or not a software or hardware generated DMA request for DMA channel 0n is pending. TRSR.CH0n can be reset by software or by

hardware at the end of a DMA transfer (RROAT = 0) or at the end of a DMA transaction (RROAT = 1).

If a software or a hardware DMA request is detected for channel 0n while TRSR.CH0n is set, a request lost event occurs. This error event indicates that the DMA is already processing a transfer and that another transfer has been requested before the end of the previous one. In this case, bit ERRSR.TRL0n will be set.

3.2.4.3 DMA Channel Operation Modes

The operation mode of a DMA channel is individually programmable for each DMA channel 0n. Basically, a DMA channel can operate in the following modes:

- Software controlled mode
- Hardware controlled mode, in Single or Continuous Mode

In software-controlled mode, a DMA channel request is generated by setting a control bit. In hardware-controlled mode, a DMA channel request is generated by request signals typically generated by on-chip peripheral units.

In hardware-controlled Single Mode, a DMA channel 0n becomes disabled by hardware after the last DMA transfer of its DMA transaction. In hardware-controlled Continuous Mode, a DMA channel 0n remains enabled after the last DMA transfer of its DMA transaction.

In hardware- and software-controlled mode, a DMA request signal can be configured to trigger a complete DMA transaction or one single transfer.

Software-controlled Modes

In software-controlled mode, one software request starts one complete DMA transaction or one single DMA transfer. Software-controlled modes are selected by writing HTREQ.DCH0n = 1. This forces status flag TRSR.HTRE0n = 0 (hardware request of DMA channel 0n is disabled).

The software-controlled mode that initiates one complete DMA transaction to be executed is selected for DMA channel 0n by the following write operations:

- CHCR0n.RROAT = 1
- STREQ.SCH0n = 1

Setting STREQ.SCH0n to 1 (this is the software request) causes the DMA transaction of DMA channel 0n to be started and TRSR.CH0n to be set. At the start of the DMA transaction, the value of CHCR0n.TREL is loaded into CHSR0n.TCOUNT (transfer count or tc) and the DMA transfers are executed. After each DMA transfer, TCOUNT becomes decremented and next source and destination addresses are calculated. When TCOUNT reaches the 0, DMA channel 0n becomes disabled and status flag TRSR.CH0n is reset. Setting STREQ.SCH0n again starts a new DMA transaction of DMA channel 0n with the parameters as actually defined in the channel register set.

The software-controlled mode that initiates a single DMA transfer to be executed is selected for DMA channel 0n by the following write operations:

- CHCR0n.RROAT = 0
- STREQ.SCH0n = 1, repeated for each DMA transfer

When CHCR0n.RROAT = 0, TRSR.CH0n becomes reset after each DMA transfer of the DMA transaction and a new software request (writing STREQ.SCH0n = 1) must be generated for starting the next DMA transfer.

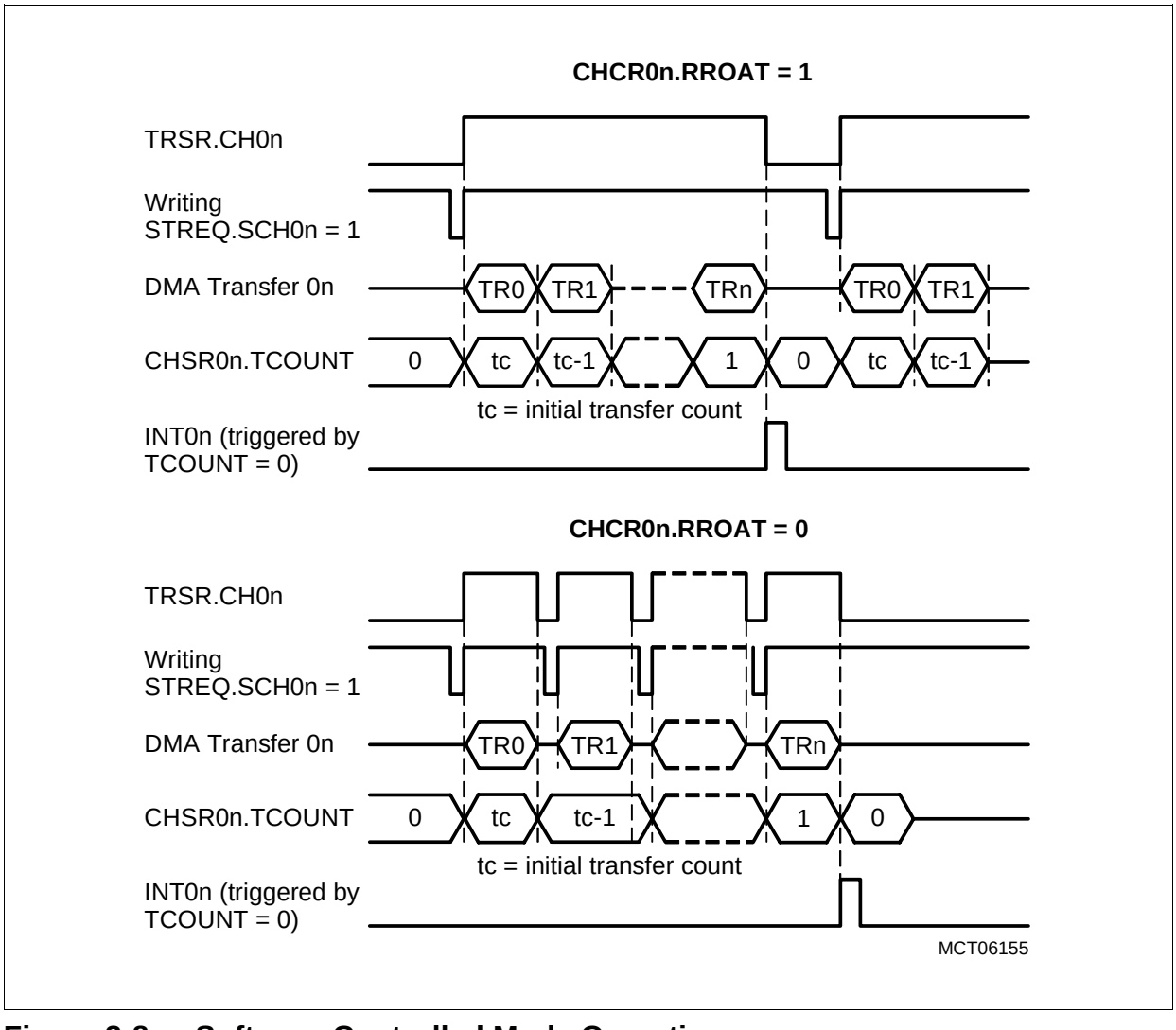


Figure 3-8 Software Controlled Mode Operation

Hardware-controlled Modes

In hardware-controlled modes, a hardware request signal starts a DMA transaction or a single DMA transfer. There are two hardware-controlled modes available:

- **Single Mode:**
  Hardware requests are disabled by hardware after a DMA transaction
- **Continuous Mode:**
  Hardware requests are not disabled by hardware after a DMA transaction

Hardware-controlled Single Mode

In hardware-controlled Single Modes, one hardware request starts one complete DMA transaction or one single DMA transfer. The hardware-controlled Single Mode that initiates one complete DMA transaction to be executed for DMA channel 0n is selected by the following operations:

- CHCR0n.CHMODE = 0
- CHCR0n.RROAT = 1
- Selecting one of the eight hardware request inputs via CHCR0n.PRSEL
- HTREQ.ECH0n = 1

Setting HTREQ.ECH0n to 1 causes the hardware request CH0n_REQ of channel 0n to be enabled (TRSR.HTRE0n = 1). Whenever the hardware request CH0n_REQ becomes active, the value of CHCR0n.TREL is loaded into CHSR0n.TCOUNT and the DMA transaction is started by executing its first DMA transfer. After each DMA transfer, TCOUNT becomes decremented and next source and destination addresses are calculated. When TCOUNT reaches the 0, DMA channel 0n becomes disabled and status flags TRSR.CH0n and TRSR.HTRE0n are reset. In order to start a new hardware-controlled DMA transaction, hardware requests must be enabled again by setting TRSR.HTRE0n through HTREQ.ECH0n = 1. The hardware request disable function in Single Mode is typically needed when a reprogramming of the DMA channel register set (addresses, transfer count) is required before the next hardware triggered DMA transaction is started.

The hardware-controlled Single Mode in which each single DMA transfer has to be requested by a hardware request signal is selected as described above, with one difference:

- CHCR0n.RROAT = 0

In this operation mode, TRSR.CH0n becomes reset after each DMA transfer of the DMA transaction, and a new hardware request at CH0n_REQ must be generated for starting the next DMA transfer.

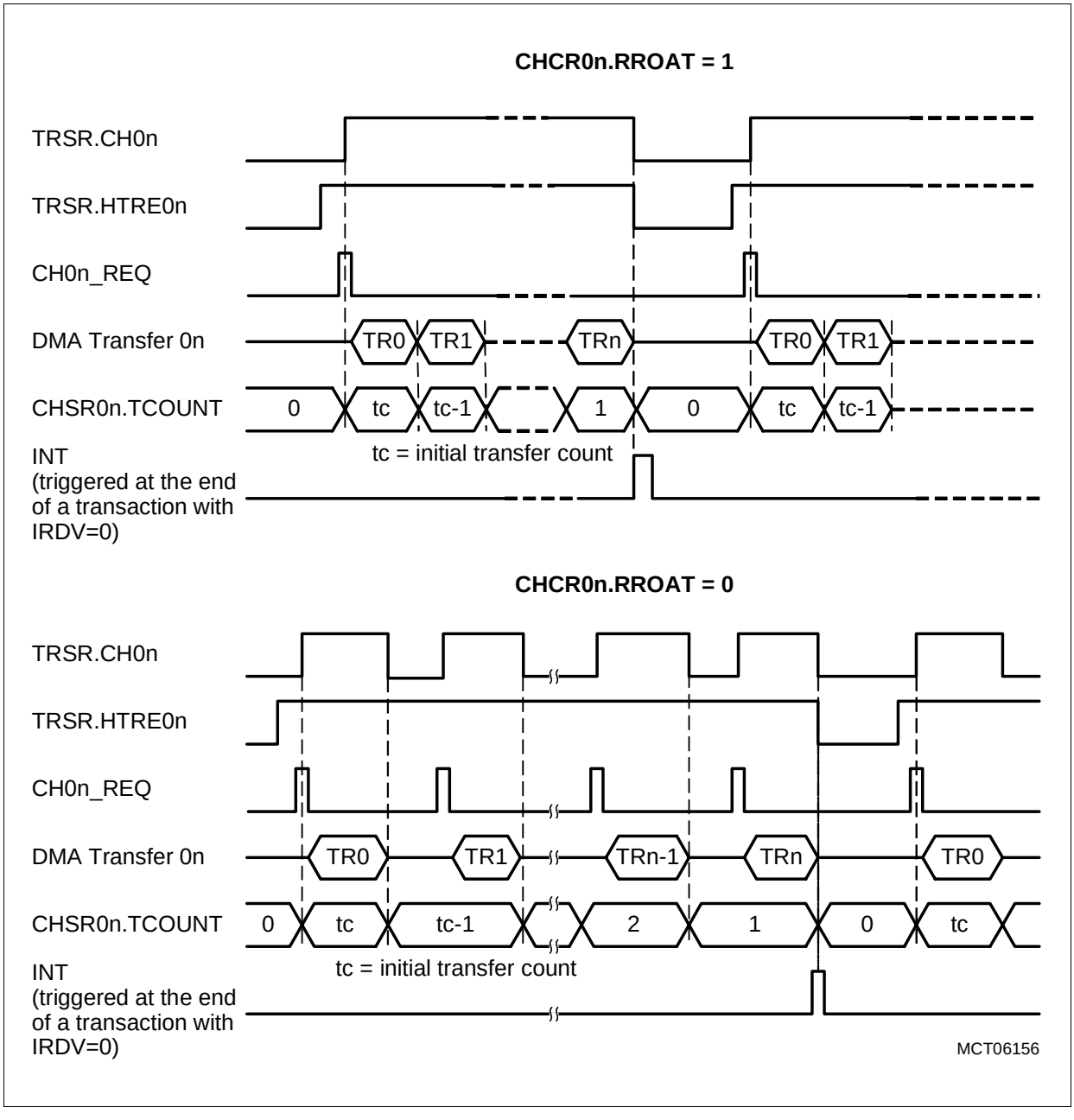


Figure 3-9 Hardware-controlled Single Mode Operation

Hardware-controlled Continuous Mode

In hardware-controlled Continuous Mode (CHCR0n.CHMODE = 1), the hardware transaction request enable bit HTRE0n is not reset at the end of a DMA transaction. A new transaction of DMA channel 0n with the parameters actually stored in the channel register set of DMA channel 0n is started each time when CHSR0n.TCOUNT reaches 000_H. No software re-enable for a hardware request at CH0n_REQ is required.

Combined Software/Hardware-controlled Mode

Figure 3-10 shows how software- and hardware-controlled modes can be combined. In the example, the first DMA transfer is triggered by software when setting STREQ.SCH0n. Hardware requests are still disabled. After hardware requests have been enabled by setting HTREQ.ECH0n, subsequent DMA transfers are triggered now by hardware request coming from the CH0n_REQ line.

In the example, DMA channel 0n operates in Single Mode (CHCR0n.CHMODE = 0). In this mode, TRSR.HTRE0n becomes reset by hardware when CHSR0n.TCOUNT reaches 0 at the end of the DMA transaction.

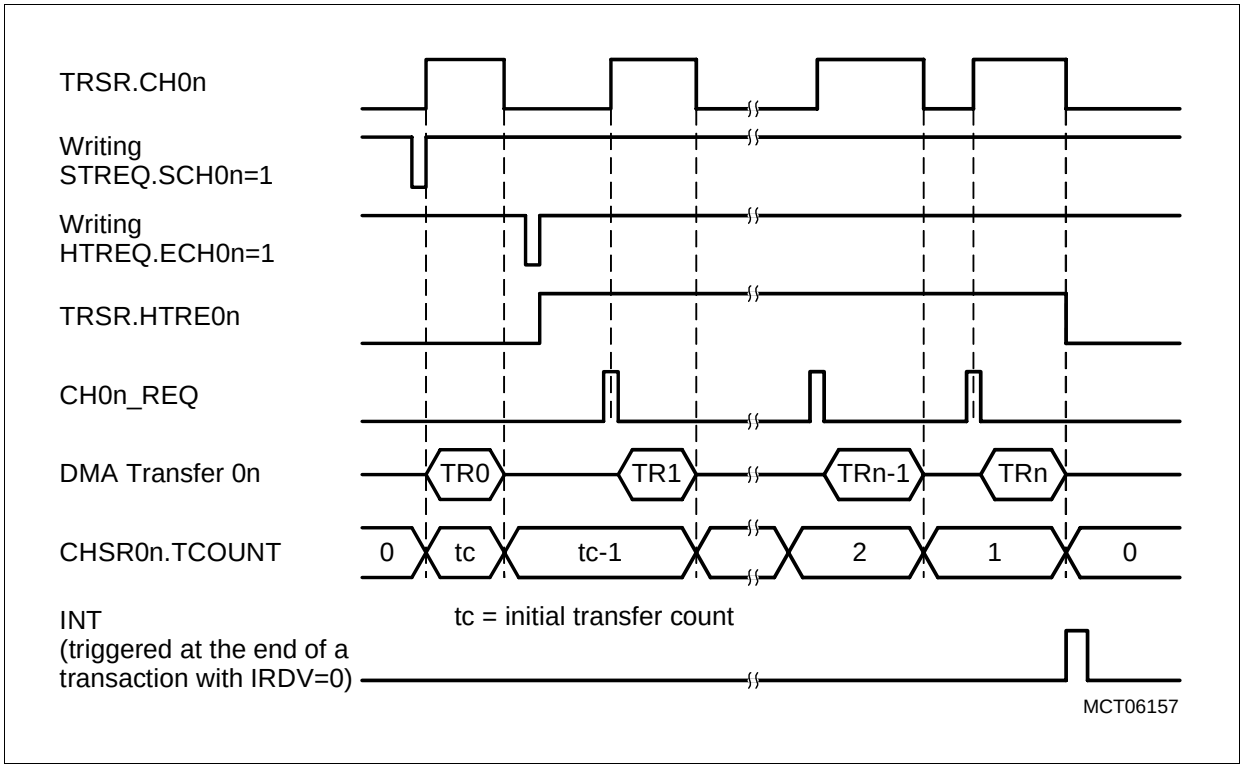


Figure 3-10 Transaction Start by Software, Continuation by Hardware

3.2.4.4 Channel Reset Operation

A DMA transaction of DMA channel 0n can be stopped (channel is reset) by setting bit CHRSTR.CH0n.

When CHRST.CH0n is set to 1:

- Bits TRSR.HTRE0n, TRSR.CH0n, ERRSR.TRL0n, INTSR.ICH0n, INTSR.IPM0n, WRPSR.WRPD0n, WRPSR.WRPS0n, CHSR0n.LXO, and bit field CHSR0n.TCOUNT are reset.
- If ADRCR0n.SHCT is 01_B or 10_B, either source or destination address register will be loaded with the value buffered in the shadow address register SHADR0n is cleared. SHADR0n will be cleared afterwards.

- All automatic functions are stopped for channel 0n.

A user program should execute the following steps for resetting and restarting a DMA channel:

1. Writing a 1 to CHRST.CH0n.
2. Waiting (polling) until CHRST.CH0n = 0.
3. Optionally (re-)configuring the address and other channel registers.
4. Restarting the DMA channel 0n by setting HTREQ.ECH0n = 1 for hardware requests or STREQ.SCH0n = 1 for software requests.

Bit field CHCR0n.TREL is copied to CHSR0n.TCOUNT when a new DMA transaction is requested.

3.2.4.5 Transfer Count and Move Count

The move count determines the number of moves (consisting of one read and one write each) to be done in each transfer. It allows the user to indicate to the DMA the number of moves to be done after one request. The number of moves per transfer is selected by the block mode settings (CHCR0n.BLKM).

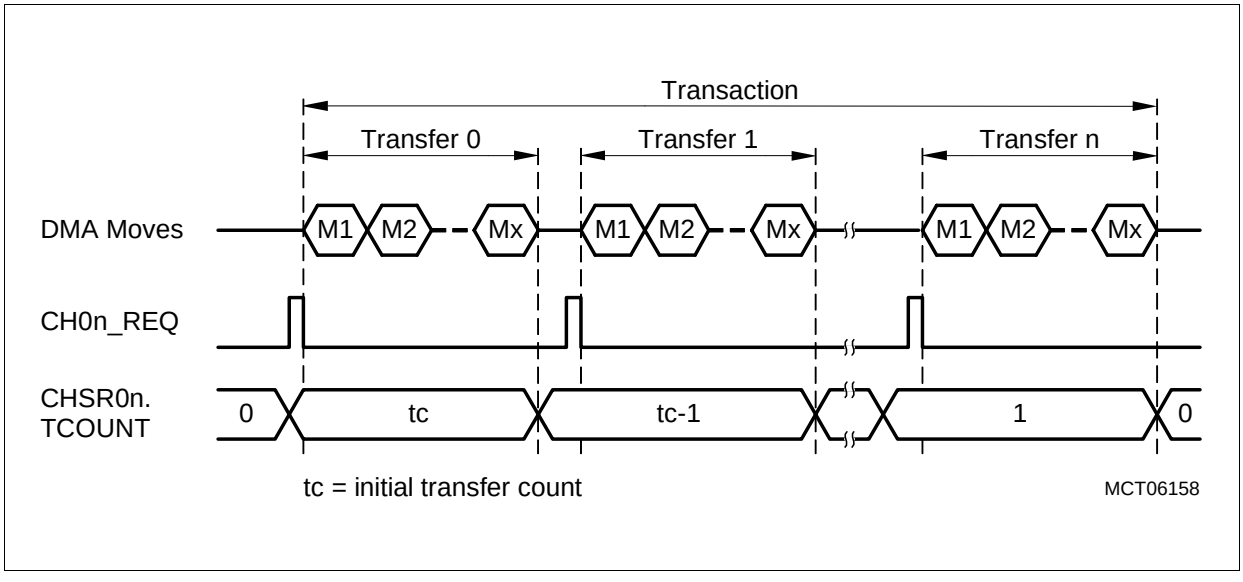


Figure 3-11 Transfer and Move Count

After a DMA move, the next source and destination addresses are calculated. Source and destination addresses are calculated independently of each other. The following address calculation parameters can be selected:

- The address offset, which is a multiple of the selected data width
- The offset direction: addition, subtraction, or none (unchanged address)

Control bits in address control register ADRCR0n determine how the addresses are incremented/decremented. Further, the data width as defined in CHCR0n.CHDW is taken into account for the address calculation.

Figure 3-12 and **Figure 3-13** show two examples of address calculation. In both examples, a data width of 16-bit (CHCR0n.CHDW = 01_B) is assumed.

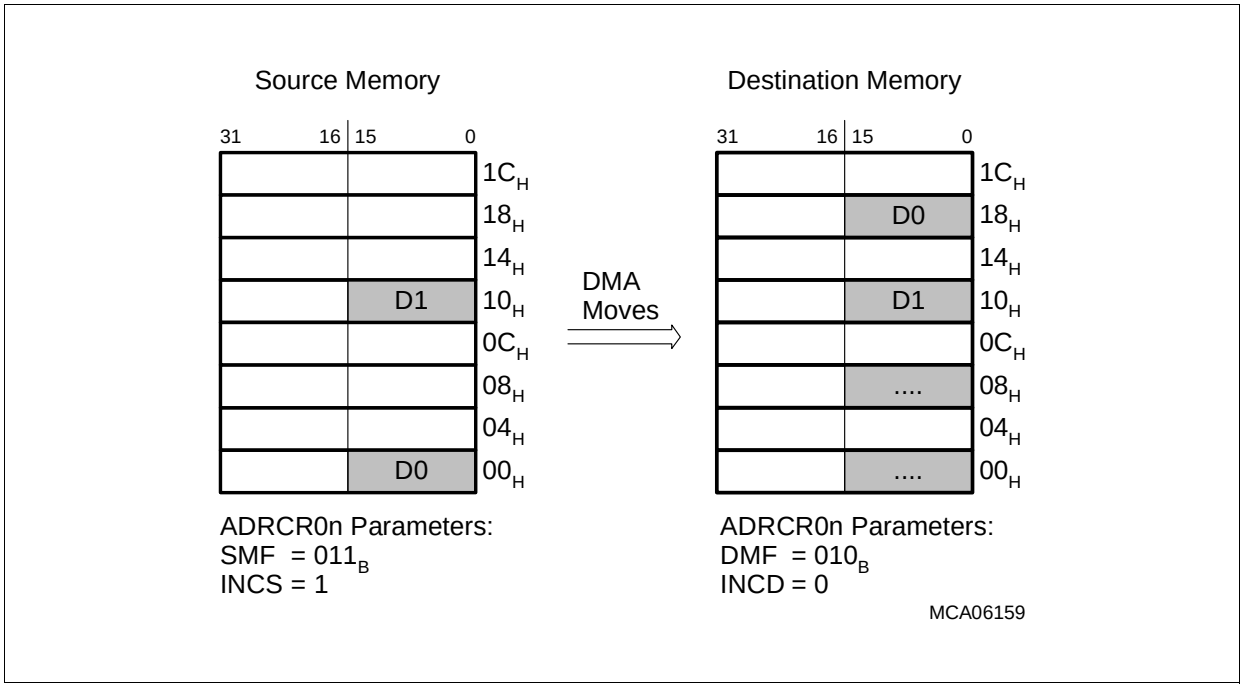


Figure 3-12 Programmable Address Modification - Example 1

In **Figure 3-12**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 10_H to a destination memory with decrementing destination addresses offset of 08_H.

In **Figure 3-13**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 02_H to a destination memory with incrementing destination addresses offset of 04_H.

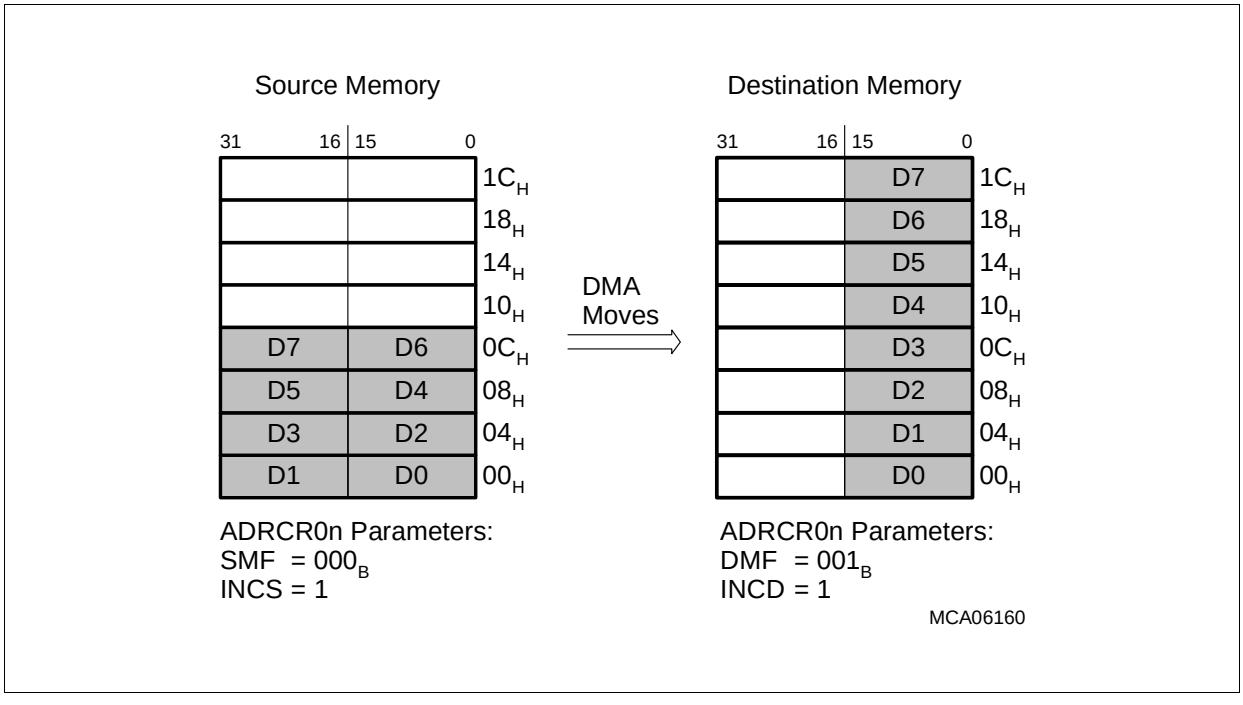


Figure 3-13 Programmable Address Modification - Example 2

3.2.4.6 Circular Buffer

Destination and source address can be configured to build a circular buffer separately for source and destination data. Within this circular buffer, addresses are updated as defined in **Figure 3-12** and **Figure 3-13** with a wrap-around at the buffer limits. The circular buffer length is determined by bit fields ADRCR0n.CBLS (for the source buffer) and ADRCR0n.CBLD (for the destination buffer). These 4-bit wide bit fields determine which bits of the 32-bit address remain unchanged at an address update. Possible buffer sizes of the circular buffers can be 2^{CBLS} or 2^{CBLD} bytes (= 1, 2, 4, 8, 16, … up to 32k bytes).

When source or destination addresses are updated (incremented or decremented) after a DMA move, all upper bits [31:CBLS] of source address and [31:CBLD] of destination address are frozen and remain unchanged, even if a wrap-around from the lower address bits [CBLS:0] or [CBLD:0] occurred. This address-freezing mechanism always causes the circular buffers to be aligned to a multiple integer value of its size.

If the circular buffer size is less or equal than the selected address offset (see **Table 3-5**), the same circular buffer address will always be accessed.

3.2.5 Transaction Control Engine

The Transaction Control Unit in the DMA Sub-Block, as shown in the DMA Controller block diagram in **Figure 3-3**, contains a Channel Arbiter and a Move Engine.

The Channel Arbiter arbitrates the transfer requests of the DMA channels, and submits the transfers parameters of the DMA channel with the highest channel priority that are needed for a DMA transfer to the Move Engine. DMA channels within a DMA Sub-Block have a two-level programmable channel priority as defined by bit CHCR0n.CHPRIO. When two transfer requests of two different DMA channels with identical channel priority become active at the same time, the DMA channel with the lowest channel number (n) is serviced first.

The Move Engine handles the execution of a DMA transfer that has been detected by the Channel Arbiter to be the next one. The Move Engine requests the required buses and loads or stores data according to the parameters of a DMA transfer. It is able to wait if a targeted bus is not available. In the Move Engine, a DMA transfer of a DMA transaction cannot be interrupted and always get finished. This means that a DMA transfer, which can also be composed of several data moves (read move and write move), cannot be interrupted by a transfer of another DMA channel.

After a DMA transfer is finished, the Move Engine will send back the actualized address register information to the related DMA channel. Possible error conditions are also reported.

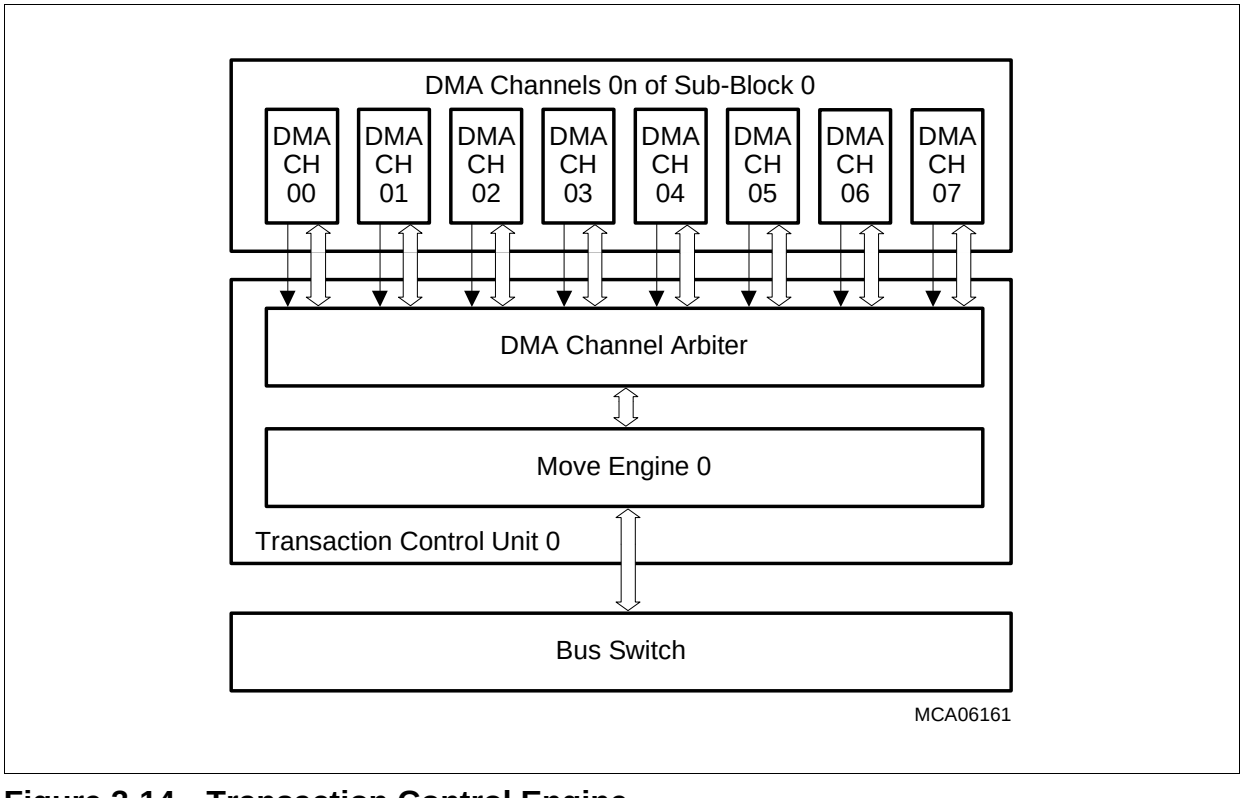


Figure 3-14 Transaction Control Engine

3.3 DMA Module Kernel Registers

Figure 3-15 and **Table 3-2** show all registers associated with the DMA Controller Kernel. All DMA kernel register names described in this section are also referenced in other parts of the CIC751 User Manual by the module name prefix “DMA_”.

DMA Registers Overview

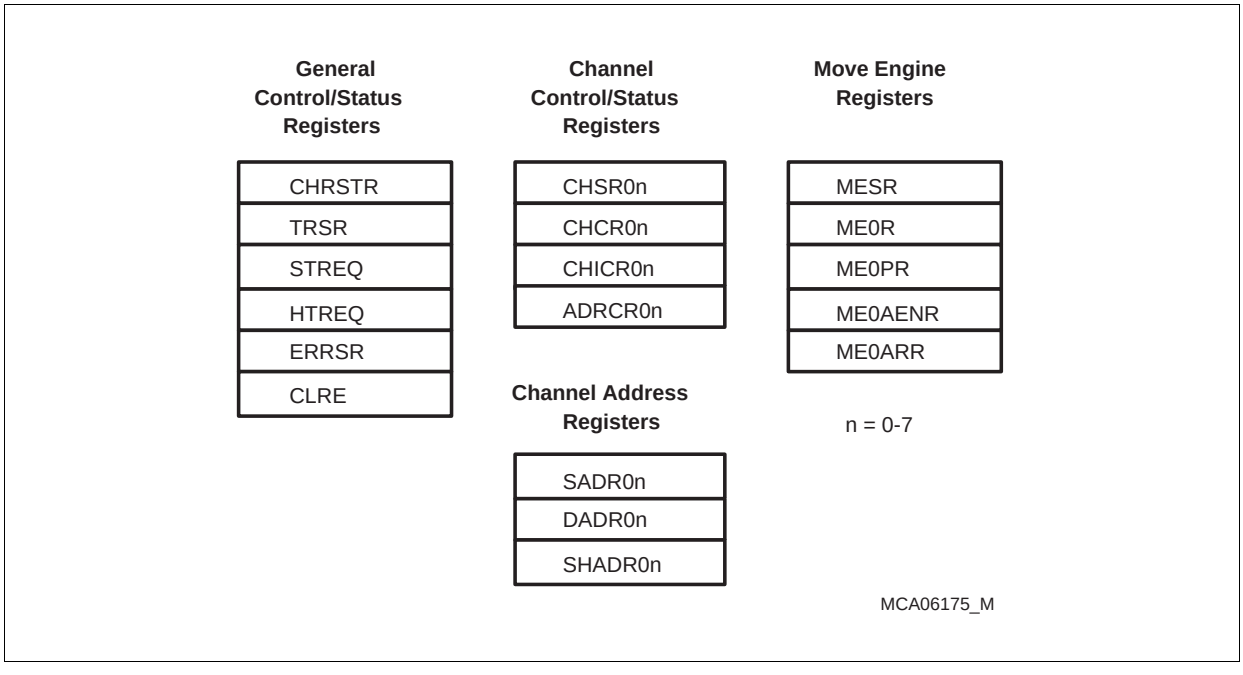


Figure 3-15 DMA Kernel Registers

Table 3-2 Registers Address Space - DMA Kernel Registers

Module	Base Address	End Address	Note
DMA	$0000\ 0400_H$	$0000\ 05FF_H$	-

Table 3-3 Registers Overview - DMA Kernel Registers

Register Short Name	Register Long Name	Address	Description see
DMA_CHRSTR	DMA Channel Reset Request Register	0410_H	**Page 3-29**
DMA_TRSR	DMA Transaction Request State Register	0414_H	**Page 3-31**
DMA_STREQ	DMA Software Transaction Request Register	0418_H	**Page 3-32**

Table 3-3 Registers Overview - DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_HTREQ	DMA Hardware Transaction Request Register	$041C_H$	Page 3-33
DMA_ERRSR	DMA Error Status Register	0424_H	Page 3-34
DMA_CLRE	DMA Clear Error Register	0428_H	Page 3-36
DMA_MESR	DMA Move Engine Status Register	0430_H	Page 3-37
DMA_ME0R	DMA Move Engine 0 Read Register	0434_H	Page 3-38
DMA_CHSR0n	DMA Channel 0n Status Register (n = 0-7)	$n \times 20_H + 0480_H$	Page 3-42
DMA_CHCR0n	DMA Channel 0n Control Register (n = 0-7)	$n \times 20_H + 0484_H$	Page 3-39
DMA_ ADRCR0n	DMA Channel 0n Address Control Register (n = 0-7)	$n \times 20_H + 048C_H$	Page 3-43
DMA_SADR0n	DMA Channel 0n Source Address Register (n = 0-7)	$n \times 20_H + 0490_H$	Page 3-47
DMA_DADR0n	DMA Channel 0n Destination Address Register (n = 0-7)	$n \times 20_H + 0494_H$	Page 3-48
DMA_SHADR0n	DMA Channel 0n Shadow Address Register (n = 0-7)	$n \times 20_H + 0498_H$	Page 3-49

3.3.1 General Control/Status Registers

The following registers are used to configure and control the request generation of the DMA from the system point of view.

SCU_ETCTR

External Trigger Control Register **850_H** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REN1	FEN1	0			INSEL1			REN0	FEN0	0			INSEL0		
rw	rw	r			rw			rw	rw	r			rw		

Field	Bits	Type	Description
INSEL0	[2:0]	rw	**Input Selection for Pin Trigger 0** This bit field defines the Trigger Source for Pin Trigger 0. 000 Pin SR0 is selected 001 Pin SR1 is selected 010 Pin SR2 is selected 011 Pin SR3 is selected 100 Pin SR4 is selected 101 MLI Break Event is selected 110 Pin AIN4 is selected 111 Pin AIN14 is selected
FEN0	6	rw	**Falling Edge Enable for Pin Trigger 0** This bit enables/disables the activation of Pin Trigger 0 upon a falling edge at the selected input. 0 The trigger upon a falling edge is disabled 1 The trigger upon a falling edge is enabled
REN0	7	rw	**Rising Edge Enable for Pin Trigger 0** This bit enables/disables the activation of Pin Trigger 0 upon a rising edge at the selected input. 0 The trigger upon a rising edge is disabled 1 The trigger upon a rising edge is enabled

Field	Bits	Type	Description
INSEL1	[10:8]	rw	**Input Selection for Pin Trigger 1** This bit field defines the Trigger Source for Pin Trigger 1. 000 Pin SR0 is selected 001 Pin SR1 is selected 010 Pin SR2 is selected 011 Pin SR3 is selected 100 Pin SR4 is selected 101 MLI Break Event is selected 110 Pin AIN4 is selected 111 Pin AIN14 is selected
FEN1	14	rw	**Falling Edge Enable for Pin Trigger 1** This bit enables/disables the activation of Pin Trigger 1 upon a falling edge at the selected input. 0 The trigger upon a falling edge is disabled 1 The trigger upon a falling edge is enabled
REN1	15	rw	**Rising Edge Enable for Pin Trigger 1** This bit enables/disables the activation of Pin Trigger 1 upon a rising edge at the selected input. 0 The trigger upon a rising edge is disabled 1 The trigger upon a rising edge is enabled
0	[5:3], [13:11]	r	**Reserved** Read as 0; should be written with 0.

The bits in the Channel Reset Request Register are used to reset DMA channel 0n.

DMA_CHRSTR
DMA Channel Reset Request Register

(410_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**CH 07**	**CH 06**	**CH 05**	**CH 04**	**CH 03**	**CH 02**	**CH 01**	**CH 00**
r							rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CH0n (n = 0-7)	n	rwh	**Channel 0n Reset** These bits force the DMA channel 0n to stop its current DMA transaction. Once set by software, this bit will be automatically cleared when the channel has been reset. Writing a 0 to CH0n has no effect. 0 No action (write) or the requested channel reset has been reset (read). 1 DMA channel 0n is stopped. More details see **Page 3-20**.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

The bits in the Transaction Request State Register indicates which DMA channel is processing a request, and which DMA channel has hardware transaction requests enabled.

DMA_TRSR
DMA Transaction Request State Register
(414_H) **Reset Value: $0000\ 0000_H$**

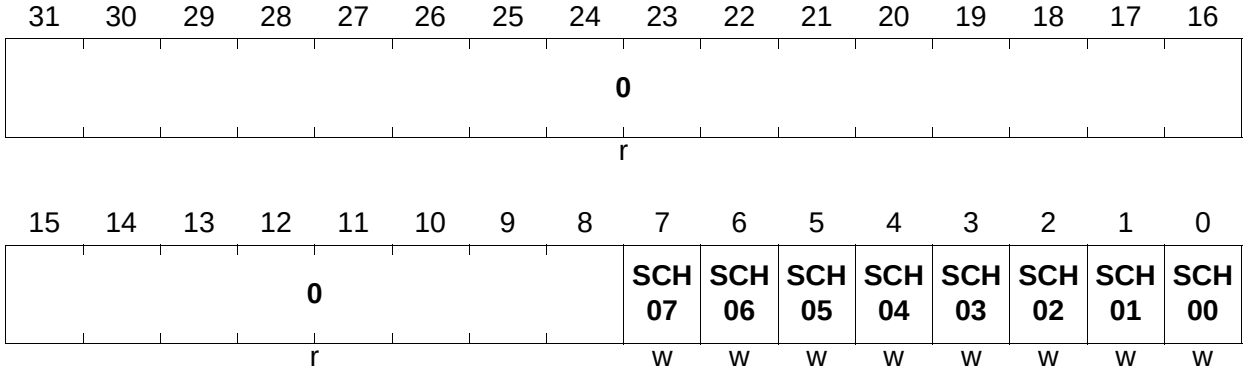
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								HTRE07	HTRE06	HTRE05	HTRE04	HTRE03	HTRE02	HTRE01	HTRE00
r								rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								CH07	CH06	CH05	CH04	CH03	CH02	CH01	CH00
r								rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CH0n (n = 0-7)	n	rh	**Transaction Request State of DMA Channel 0n** 0 No DMA request is pending for channel 0n. 1 A DMA request is pending for channel 0n.
HTRE0n (n = 0-7)	n+16	rh	**Hardware Transaction Request Enable State of DMA Channel 0n** 0 Hardware transaction request for DMA Channel 0n is disabled. An input DMA request will not trigger the channel 0n. 1 Hardware transaction request for DMA Channel 0n is enabled. The transfers of a DMA transaction are controlled by the corresponding channel request line of the DMA requesting source. HTRE0n is set to 0 when CHSR0n.TCOUNT is decremented and CHSR0n.TCOUNT = 0. HTRE0n can be enabled and disabled with HTREQ.ECH0n or HTREQ.DCH0n.
0	[31:24], [15:8]	r	**Reserved** Read as 0; should be written with 0.

The bits in the Software Transaction Request Register are used to generate a DMA transaction request by software.

DMA_STREQ
DMA Software Transaction Request Register
(418_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SCH0n (n = 0-7)	n	w	**Set Transaction Request for DMA Channel 0n** 0 No action. 1 A transaction for DMA channel 0n is requested. When setting SCH0n, TRSR.CH0n becomes set to indicate that a DMA request is pending for DMA channel 0n.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

The bits in the Hardware Transaction Request Register enable or disable DMA hardware requests.

DMA_HTREQ
DMA Hardware Transaction Request Register
($41C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								DCH 07	DCH 06	DCH 05	DCH 04	DCH 03	DCH 02	DCH 01	DCH 00
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								ECH 07	ECH 06	ECH 05	ECH 04	ECH 03	ECH 02	ECH 01	ECH 00
r								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
ECH0n (n = 0-7)	n	w	**Enable Hardware Transfer Request for DMA Channel 0n** see table below
DCH0n (n = 0-7)	n + 16	w	**Disable Hardware Transfer Request for DMA Channel 0n** see table below
0	[31:24], [15:8]	r	**Reserved** Read as 0; should be written with 0.

Set/Reset Bit Conditions

Table 3-4 Conditions to Set/Reset the Bits TRSR.HTRE0n

HTREQ.ECH0n	HTREQ.DCH0n	Transaction Finishes[1] for Channel 0n	Modification of TRSR.HTRE0n
0	0	0	unchanged
1	0	0	set
X	1	X	reset
X	X	1	reset

[1] In Single Mode only. In Continuous Mode, the end of a transaction has no impact.

The Error Status Register indicates if the DMA controller could not answer to a request because the previous request was not terminated.

DMA_ERRSR
DMA Error Status Register (424$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				MLI0		LEC ME0		0						ME0 DER	ME0 SER
r				rh		rh		r						rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								TRL 07	TRL 06	TRL 05	TRL 04	TRL 03	TRL 02	TRL 01	TRL 00
r								rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TRL0n (n = 0-7)	n	rh	**Transaction/Transfer Request Lost of DMA Channel 0n** 0 No request lost event has been detected for channel 0n. 1 A new DMA request was detected while TRSR.CH0n=1 (request lost event). This bit is reset by software when writing a 1 to CLRE.CTL0n, or by a channel reset (writing CHRSTR.CH0n = 1).
ME0SER	16	rh	**Move Engine 0 Source Error** This bit is set whenever a Move Engine 0 error occurred during a source (read) move of a DMA transfer, or a request could not been serviced due to the access protection. 0 No Move Engine 0 source error has occurred. 1 A Move Engine 0 source error has occurred.
ME0DER	17	rh	**Move Engine 0 Destination Error** This bit is set whenever a Move Engine 0 error occurred during a destination (write) move of a DMA transfer, or a request could not been serviced due to the access protection. 0 No Move Engine 0 destination error has occurred. 1 A Move Engine 0 destination error has occurred.

Field	Bits	Type	Description
LECME0	[26:24]	rh	**Last Error Channel Move Engine 0** This bit field indicates the channel number of the last channel of Move Engine 0 leading to an Bus error that has occurred.
MLI0	27	rh	**MLI0 Error Source** This bit is set whenever an Bus error occurred due to an action of MLI0. 0 No bus error occurred due to MLI0. 1 An bus error occurred due to MLI0.
0	[15:8], [23:18], [31:28]	r	**Reserved** Read as 0; should be written with 0.

The Clear Error contains bits that make it possible to clear the Transaction Request Lost flags or the Move Engine error flags.

DMA_CLRE
DMA Clear Error Register (428$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				CLRMLI0	0									CME0DER	CME0SER
r				w	r									w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								CTL07	CTL06	CTL05	CTL04	CTL03	CTL02	CTL01	CTL00
r								w	w	w	w	w	w	w	w

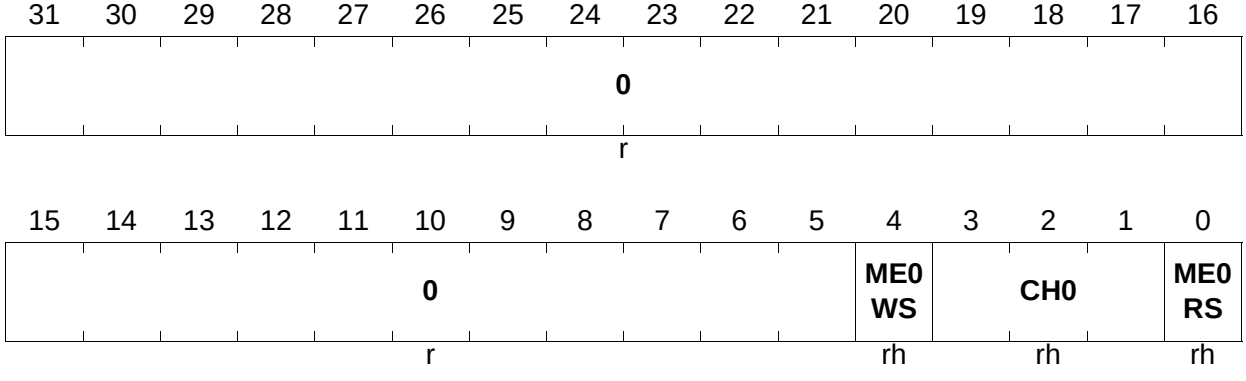
Field	Bits	Type	Description
CTL0n (n = 0-7)	n	w	**Clear Transaction Request Lost for DMA Channel 0n** 0 No action 1 Clear DMA channel 0n transaction request lost flag ERRSR.TRL0n
CME0SER	16	w	**Clear Move Engine 0 Source Error** 0 No action 1 Clear source error flag ERRSR.ME0SER.
CME0DER	17	w	**Clear Move Engine 0 Destination Error** 0 No action 1 Clear destination error flag ERRSR.ME0DER.
CLRMLI0	27	w	**Clear MLI0 Error** 0 No action 1 Clear error flag ERRSR.MLI0.
0	[15:8], [26:18], [31:28]	r	**Reserved** Read as 0; should be written with 0.

3.3.2 Move Engine Registers

The Move Engine Status Register is a read-only register that holds status information about the transaction handled by the Move Engines.

DMA_MESR

DMA Move Engine Status Register (430$_H$) Reset Value: 0000 0000$_H$

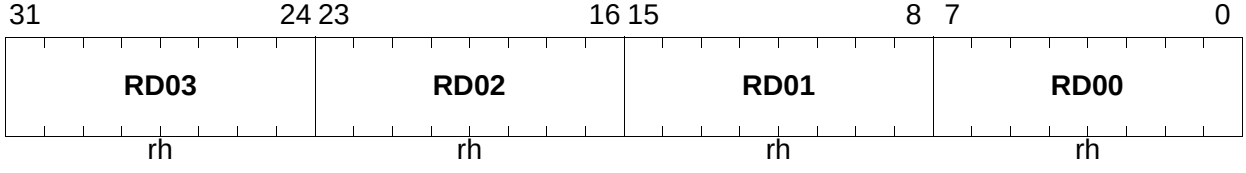


Field	Bits	Type	Description
ME0RS	0	rh	**Move Engine 0 Read Status** 0 Move Engine 0 is not performing a read. 1 Move Engine 0 is performing a read.
CH0	[3:1]	rh	**Reading Channel in Move Engine 0** This bit field indicates which channel number is currently being processed by the Move Engine 0.
ME0WS	4	rh	**Move Engine 0 Write Status** 0 Move Engine 0 is not performing a write. 1 Move Engine 0 is performing a write.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

The Move Engine 0 Read Register indicates the value that has just been read by Move Engine 0.

DMA_ME0R

DMA Move Engine 0 Read Register (434$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RD00, RD01, RD02, RD03	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine 0** Contains the 32-bit read data (four bytes RD0[3:0]) that is stored in the Move Engine 0 after each read move. The content of ME0R is overwritten after each read move of a DMA channel belonging to DMA Sub-block 0.

3.3.3 Channel Control/Status Registers

The Channel Control Register for DMA channel 0n contains its configuration and its control bits and bit fields.

DMA_CHCR0n (n = 0-7)
DMA Channel 0n Control Register (484_H+n*20_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**CH PRIO**	**0**		**0L**		**0**	**CHDW**		**CH MO DE**	**RRO AT**	**BLKM**		
r	rw	r	rw	r		rw		r	rw		rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRSEL			**0**				**TREL**								
rw			r				rw								

Field	Bits	Type	Description
TREL	[8:0]	rw	**Transfer Reload Value** This bit field contains the number of DMA transfers for s DMA transaction of DMA channel 0n. This 9-bit transfer count value is loaded into CHSR0n.TCOUNT at the start of a DMA transaction (when TRSR.CH00n becomes set and CHSR0n.TCOUNT = 0). TREL can be written during a running DMA transaction because TCOUNT will be updated (decremented) during the DMA transaction. If TREL = 0 or if TREL = 1, TCOUNT will be loaded with 1 when a new transaction is started (at least one DMA transfer must be executed per DMA transaction).

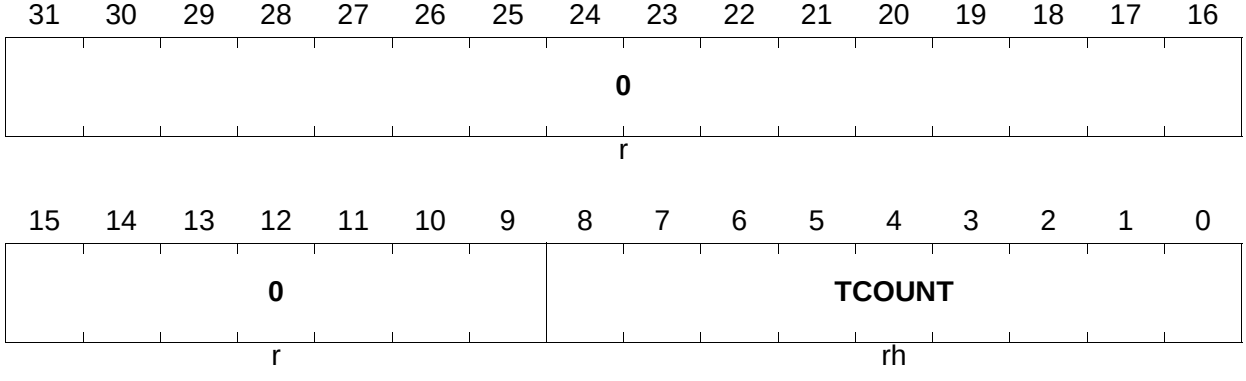
Field	Bits	Type	Description
PRSEL	[15:13]	rw	**Peripheral Request Select** This bit field controls the hardware request input multiplexer of DMA channel 0n (see **Figure 3-7** on **Page 3-15**). 000_BInput CH0n_REQI0 selected 001_BInput CH0n_REQI1 selected 010_BInput CH0n_REQI2 selected 011_BInput CH0n_REQI3 selected 100_BInput CH0n_REQI4 selected 101_BInput CH0n_REQI5 selected 110_BInput CH0n_REQI6 selected 111_BInput CH0n_REQI7 selected
BLKM	[18:16]	rw	**Block Mode** BLKM determines the number of DMA moves executed during one DMA transfer. 000_BOne DMA transfer has 1 DMA move 001_BOne DMA transfer has 2 DMA moves 010_BOne DMA transfer has 4 DMA moves 011_BOne DMA transfer has 8 DMA moves 100_BOne DMA transfer has 16 DMA moves OthersReserved; do not use these combinations. See also **Figure 3-11** on **Page 3-22**.
RROAT	19	rw	**Reset Request Only After Transaction** RROAT determines whether or not the TRSR.CH0n transfer request state flag is reset after each transfer. 0 TRSR.CH0n is reset after each transfer. A transfer request is required for each transfer. 1 TRSR.CH0n is reset when TCOUNT = 0 after a transfer. One transfer request starts a complete DMA transaction.

Field	Bits	Type	Description
CHMODE	20	rw	**Channel Operation Mode** CHMODE determines the reset condition for control bit TRSR.HTRE0n of DMA channel 0n. 0 Single Mode operation is selected for DMA channel 0n. After a transaction, DMA channel 0n is disabled for further hardware requests (TRSR.HTRE0n is reset by hardware). TRSR.HTRE0n must be set again by software for starting a new transaction. 1 Continuous Mode operation is selected for DMA channel 0n. After a transaction, bit TRSR.HTRE0n remains set.
CHDW	[22:21]	rw	**Channel Data Width** CHDW determines the data width for the read and write moves of DMA channel 0n. 00 8-bit (byte) data width for moves selected 01 16-bit (half-word) data width for moves selected 10 32-bit (word) data width for moves selected 11 Reserved
CHPRIO	28	rw	**Channel Priority** CHPRIO determines the priority of DMA channel 0n for the channel arbitration of Move Engine 0. 0 DMA channel 0n has a low channel priority. 1 DMA channel 0n has a high channel priority.
0	[25:24], 30	rw	**Reserved** Read as 0; have to be written with 0.
0	[12:9], 23, [27:26], 29, 31	r	**Reserved** Read as 0; should be written with 0.

The Channel Status Register contains the current transfer count and a pattern detection compare result.

DMA_CHSR0n (n = 0-7)

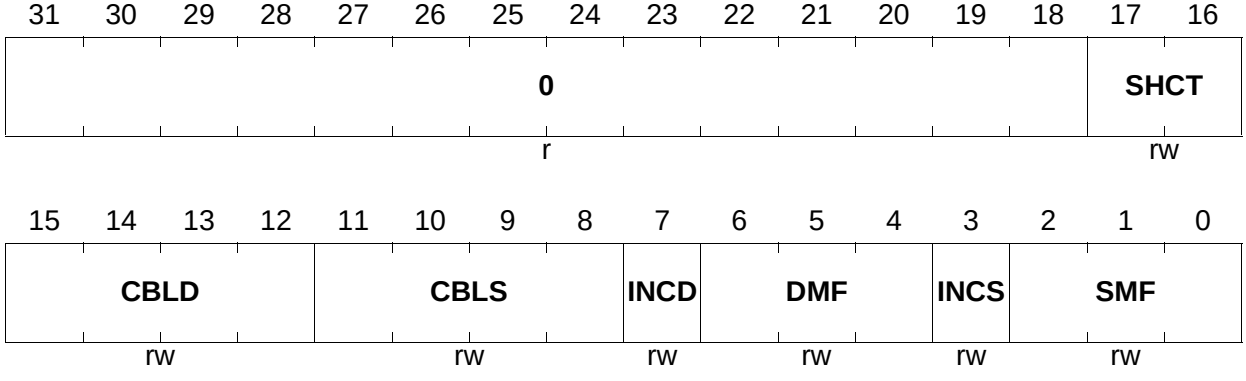
DMA Channel 0n Status Register (480$_H$+n*20$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TCOUNT	[8:0]	rh	**Transfer Count Status** TCOUNT holds the actual value of the DMA transfer count for DMA channel 0n. TCOUNT is loaded with the value of CHCR0n.TREL when TRSR.CH0n becomes set (and TCOUNT = 0). After each DMA transfer, TCOUNT is decremented by 1.
0	[31:9]	r	**Reserved** Read as 0; should be written with 0.

The Address Control Register controls how source and destination addresses are updated after a DMA move. Furthermore, it determines whether or not a source or destination address register update is shadowed.

DMA_ADRCR0n (n = 0-7)
DMA Channel 0n Address Control Register
($48C_H$+n*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SMF	[2:0]	rw	**Source Address Modification Factor** This bit field and the data width as defined in CHCR0n.CHDW determine an address offset value by which the source address is modified after each DMA move. See also **Table 3-5**. 000_B Address offset is 1 × CHCR0n.CHDW. 001_B Address offset is 2 × CHCR0n.CHDW. 010_B Address offset is 4 × CHCR0n.CHDW. 011_B Address offset is 8 × CHCR0n.CHDW. 100_B Address offset is 16 × CHCR0n.CHDW. 101_B Address offset is 32 × CHCR0n.CHDW. 110_B Address offset is 64 × CHCR0n.CHDW. 111_B Address offset is 128 × CHCR0n.CHDW.
INCS	3	rw	**Increment of Source Address** This bit determines whether the address offset as selected by SMF will be added to or subtracted from the source address after each DMA move. The source address is not modified if CBLS = 0000_B. 0 Address offset will be subtracted. 1 Address offset will be added.

Field	Bits	Type	Description
DMF	[6:4]	rw	**Destination Address Modification Factor** This bit field and the data width as defined in CHCR0n.CHDW determines an address offset value by which the destination address is modified after each DMA move. The destination address is not modified if CBLD = 0000_B. See also **Table 3-5**. 000_B Address offset is 1 × CHDW. 001_B Address offset is 2 × CHDW. 010_B Address offset is 4 × CHDW. 011_B Address offset is 8 × CHDW. 100_B Address offset is 16 × CHDW. 101_B Address offset is 32 × CHDW. 110_B Address offset is 64 × CHDW. 111_B Address offset is 128 × CHDW.
INCD	7	rw	**Increment of Destination Address** This bit determines whether the address offset as selected by DMF will be added to or subtracted from the destination address after each DMA move. The destination address is not modified if CBLD = 0000_B. 0 Address offset will be subtracted. 1 Address offset will be added.
CBLS	[11:8]	rw	**Circular Buffer Length Source** This bit field determines which part of the 32-bit source address register remains unchanged and is not updated after a DMA move operation (see also **Section 3.2.4.6**). Therefore, CBLS also determines the size of the circular source buffer. 0000_B Source address SADR[31:0] is not updated. 0001_B Source address SADR[31:1] is not updated. 0010_B Source address SADR[31:2] is not updated. 0011_B Source address SADR[31:3] is not updated. …… 1110_B Source address SADR[31:14] is not updated. 1111_B Source address SADR[31:15] is not updated.

Field	Bits	Type	Description
CBLD	[15:12]	rw	**Circular Buffer Length Destination** This bit field determines which part of the 32-bit destination address register remains unchanged and is not updated after a DMA move operation (see also Page 3-24). Therefore, CBLD also determines the size of the circular destination buffer. 0000_B Destination address DADR[31:0] is not updated. 0001_B Destination address DADR[31:1] is not updated. 0010_B Destination address DADR[31:2] is not updated. 0011_B Destination address DADR[31:3] is not updated. …… 1110_B Destination address DADR[31:14] is not updated. 1111_B Destination address DADR[31:15] is not updated.
SHCT	[17:16]	rw	**Shadow Control** This bit field determines whether an address is transferred into the shadow address register when writing to source or destination address register. 00 Shadow address register not used. Source and destination address register are written directly. 01 Shadow address register used for source address buffering. When writing to SADR0n, the address is buffered in SHADR0n and transferred to SADR0n with the start of the next DMA transaction. 10 Shadow address register used for destination address buffering. When writing to DADR0n, the address is buffered in SHADR0n and transferred to DADR0n with the start of the next DMA transaction. 11 Reserved In case of SHCT = 01_B or 10_B, SHCT must not be changed until the next DMA transaction has been started.
0	[31:18]	r	**Reserved** Read as 0; should be written with 0.

Table 3-5 shows the offset values that are added or subtracted to/from a source or destination address register after a DMA move. Bit field SMF and bit INCS determine the offset value for the source address. Bit field DMF and bit INCD determine the offset value for the destination address.

Table 3-5 Address Offset Calculation Table

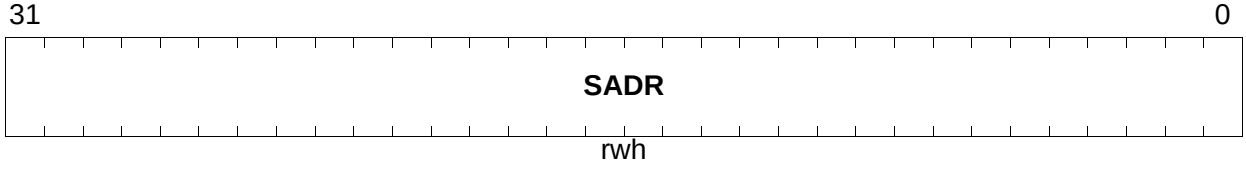
CHCR0n.CHDW = 00_B (8-bit Data Width)			CHCR0n.CHDW = 01_B (16-bit Data Width)			CHCR0n.CHDW = 10_B (32-bit Data Width)		
SMF DMF	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**
000_B	0	-1	000_B	0	-2	000_B	0	-4
	1	+1		1	+2		1	+4
001_B	0	-2	001_B	0	-4	001_B	0	-8
	1	+2		1	+4		1	+8
010_B	0	-4	010_B	0	-8	010_B	0	-16
	1	+4		1	+8		1	+16
011_B	0	-8	011_B	0	-16	011_B	0	-32
	1	+8		1	+16		1	+32
100_B	0	-16	100_B	0	-32	100_B	0	-64
	1	+16		1	+32		1	+64
101_B	0	-32	101_B	0	-64	101_B	0	-128
	1	+32		1	+64		1	+128
110_B	0	-64	110_B	0	-128	110_B	0	-256
	1	+64		1	+128		1	+256
111_B	0	-128	111_B	0	-256	111_B	0	-512
	1	+128		1	+256		1	+512

Note: CHCR0n.CHDW = 11_B is reserved and should not be used.

3.3.4 Channel Address Registers

The Source Address Register contains the 32-bit source address. If a DMA channel 0n is active, SADR0n is updated continuously (if programmed) and shows the actual source address that is used for read moves within DMA transfers.

DMA_SADR0n (n = 0-7)
DMA Channel 0n Source Address Register
(490_H+n*20_H) **Reset Value: 0000 0000_H**

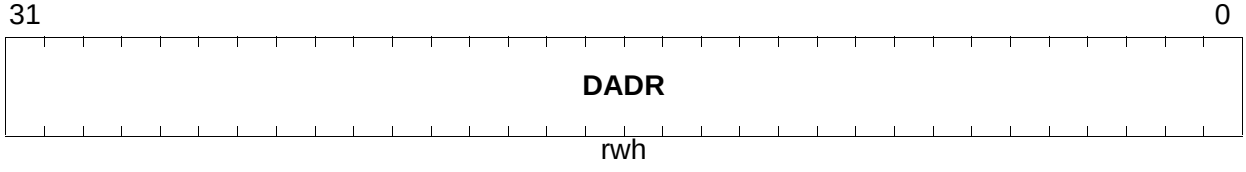


Field	Bits	Type	Description
SADR	[31:0]	rwh	**Source Start Address** This bit field holds the actual 32-bit source address of DMA channel 0n that is used for read moves.

A write to SADR0n is executed directly only when the DMA channel 0n is inactive (CHSR0n.TCOUNT = 0 and TRSR.CH0n = 0). If DMA channel 0n is active when writing to SADR0n, the source address will not be written into SADR0n directly but will be buffered in the shadow register SADR0n until the start of the next DMA transaction. During this shadowed address register operation, bit field ADRCR0n.SHCT must be set to 01_B.

The Destination Address Register contains the 32-bit destination address. If a DMA channel is active, DADR0n is updated continuously (if programmed) and shows the actual destination address that is used for write moves within DMA transfers.

DMA_DADR0n (n = 0-7)
DMA Channel 0n Destination Address Register
(494_H+n*20_H) **Reset Value: 0000 0000_H**

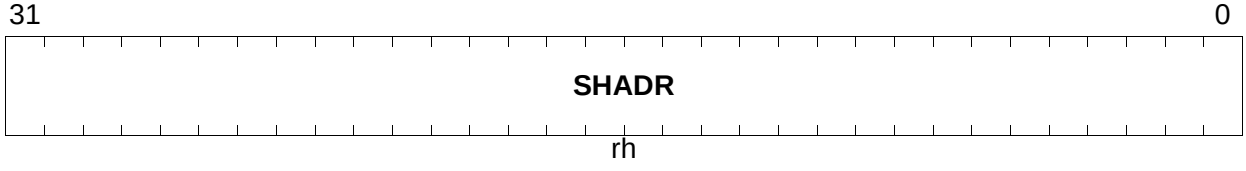


Field	Bits	Type	Description
DADR	[31:0]	rwh	**Destination Address** This bit field holds the actual 32-bit destination address of DMA channel 0n that is used for write moves.

A write to DADR0n is executed directly only when the DMA channel 0n is inactive (CHSR0n.TCOUNT = 0 and TRSR.CH0n = 0). If DMA channel 0n is active when writing to DADR0n, the source address will not be written into DADR0n directly but will be buffered in the shadow register SADR0n until the start of the next DMA transaction. During this shadowed address register operation, bit field ADRCR0n.SHCT must be set to 10_B.

The Shadow Address Register holds the shadowed source or destination address before it is written into the source or destination address register. SHADR0n can be read only.

DMA_SHADR0n (n = 0-7)
DMA Channel 0n Shadow Address Register
(498_H+n*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SHADR	[31:0]	rh	**Shadowed Address** This bit field holds the shadowed 32-bit source or destination address of DMA channel 0n.

SHADR0n is written when source or destination address buffering is selected (ADRCR0n.SHCT = 01_B or ADRCR0n.SHCT = 10_B) and a transaction is running. While the shadow mechanism is disabled, SHADR is set to 0000 0000_H.

The value stored in the SHADR is automatically set to 0000 0000_H when the shadow transfer takes place. The user can read the shadow register in order to detect if the shadow transfer has already taken place. If the value in SHADR is 0000 0000_H, no shadow transfer can take place and the corresponding address register is modified according to the circular buffer rules.

4 Micro Link Interface (MLI)

This chapter describes the Micro Link Interface (MLI) module and the MLI protocol.

4.1 MLI Protocol

This section describes the MLI protocol and its general usage, and defines the terms specific to the MLI.

4.1.1 Overview

The Micro Link Interface (MLI) is a fast synchronous serial interface that makes it possible to exchange data between microcontrollers or other devices. **Figure 4-1** shows how two microcontrollers are typically connected via their MLI interfaces. In this example, the MLI modules operate in both microcontrollers as bus masters on the internal system bus.

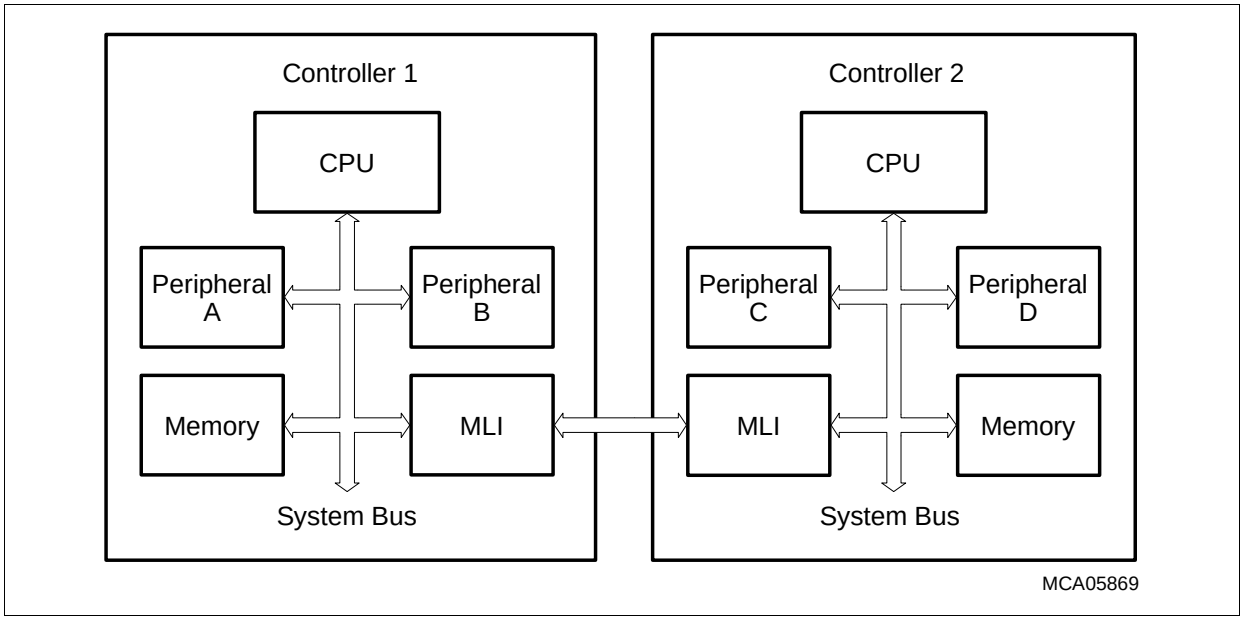


Figure 4-1 Typical Micro Link Interface Connection

Features:

- Synchronous serial communication between an MLI transmitter and an MLI receiver
- Different system clock speeds supported in MLI transmitter and MLI receiver due to full handshake protocol (4 lines between a transmitter and a receiver)
- Fully transparent read/write access supported (= remote programming)
- Complete address range of target device (Remote Controller) available
- Specific frame protocol to transfer commands, addresses and data
- Error detection by parity bit
- 32-bit, 16-bit, or 8-bit data transfers supported

- Programmable baud rate: $f_{MLI}/2$ (max.: $f_{MLI} = f_{SYS}$)
- Multiple receiving devices supported

4.1.2 MLI Specific Terms

Local and Remote Controller

The terms “local” and “remote” controller are assigned to the two partners (microcontrollers or other devices with MLI modules) in a serial MLI connection. The controller with an MLI module that operates as a transmitter in the serial MLI connection is defined as a Local Controller. A Local Controller handles data operations with Transfer Windows and also initiates all control tasks (control, address, and data transmissions) that are required for the data transfer/request between the Local Controller and the Remote Controller.

The controller with an MLI module that operates as a receiver in the serial MLI connection is defined as a Remote Controller. A Remote Controller handles data operations with Remote Windows and executes the tasks that have been assigned/ requested by the Local Controller.

Due to the full duplex operation capability of an MLI module, two serial MLI connections can be installed simultaneously. This means that each microcontroller with an MLI module is able to operate as a Local Controller (for transmission) as well as a Remote Controller (for reception) at the same time.

Transfer Window

A Transfer Window is an address space in the address map of the Local Controller. Transfer Windows are typically assigned to a fixed address space (base address and size) in a specific microcontroller. The transfers windows are the logical data inputs for the MLI transmitter. Data write actions are initiated by a write access to a Transfer Window, whereas data read actions are started by a read access from a Transfer Window.

Each MLI module supports up to eight independent Transfer Windows. Each Transfer Window can be accessed at two different address ranges with two different window sizes, leading to:

- Four small Transfer Windows, each with an 8 KByte address range
- Four large Transfer Windows, each with an 64 KByte address range

Remote Window

A Remote Window defines an area in the address space of the Remote Controller. Remote Window parameters (base address and window size) of the Remote Controller are programmable by the local microcontroller by MLI transfers. Each Remote Window of a Remote Controller is related to specific pair of small and large Transfer Windows of the Local Controller via one Pipe.

The Remote Windows are the logical data outputs of the MLI receiver. If enabled, the MLI module can automatically execute the requested data transfer to/from the defined address location in the Remote Window. If the automatic data handling is disabled, the offset and the data are available in the MLI receiver.

Pipe

A Pipe defines the logical connection between a Transfer Window in the Local Controller and the associated Remote Window in the Remote Controller. The MLI module supports up to four Pipes.

Frame

A frame is a contiguous set of bits forming a message sent by an MLI transmitter to an MLI receiver.

Normal Frame

A Normal Frame is the collective term for the following frame types:

- Write Offset and Data Frame
- Optimized Write Frame
- Discrete Read Frame
- Optimized Read Frame
- Answer Frame
- Copy Base Address Frame

Offset

The offset is defined by the accessed address location relative to the base address of the Transfer Window in the Local Controller. The access is transferred to the Remote Controller, where it is executed at the address location defined by the base address of the Remote Window plus the offset. For example, a write access to the 10th byte of the Transfer Window is transferred to a write to the 10th byte of the Remote Window.

The offset of a write action to a Transfer Window is also called a write offset, whereas a read offset is related to a read action.

4.1.3 MLI Communication Principles

The communication principle of the MLI allows data to be transferred between a local and a Remote Controller without intervention by a CPU in the Remote Controller. Data transfers are always triggered in the Local Controller by read or write operations to an address location in a Transfer Window. All control tasks (control, address, and data transmissions) that are required for the data transfer/request between local and Remote Controller are handled autonomously by the two connected MLI modules.

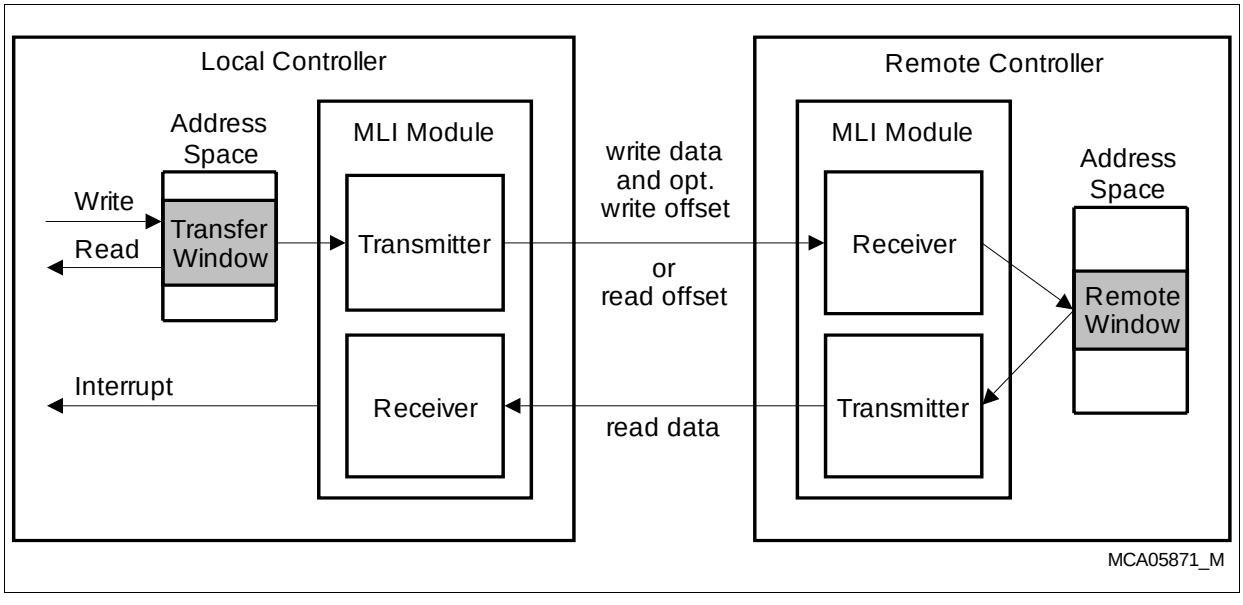


Figure 4-2 MLI Communication Principles

Transfer Window Organization

Figure 4-3 shows the organization of Transfer Windows and Remote Windows with a possible assignment in the Local and Remote Controller. Each of the four Pipes assigns one Transfer Window to one Remote Window with its base address and window size.

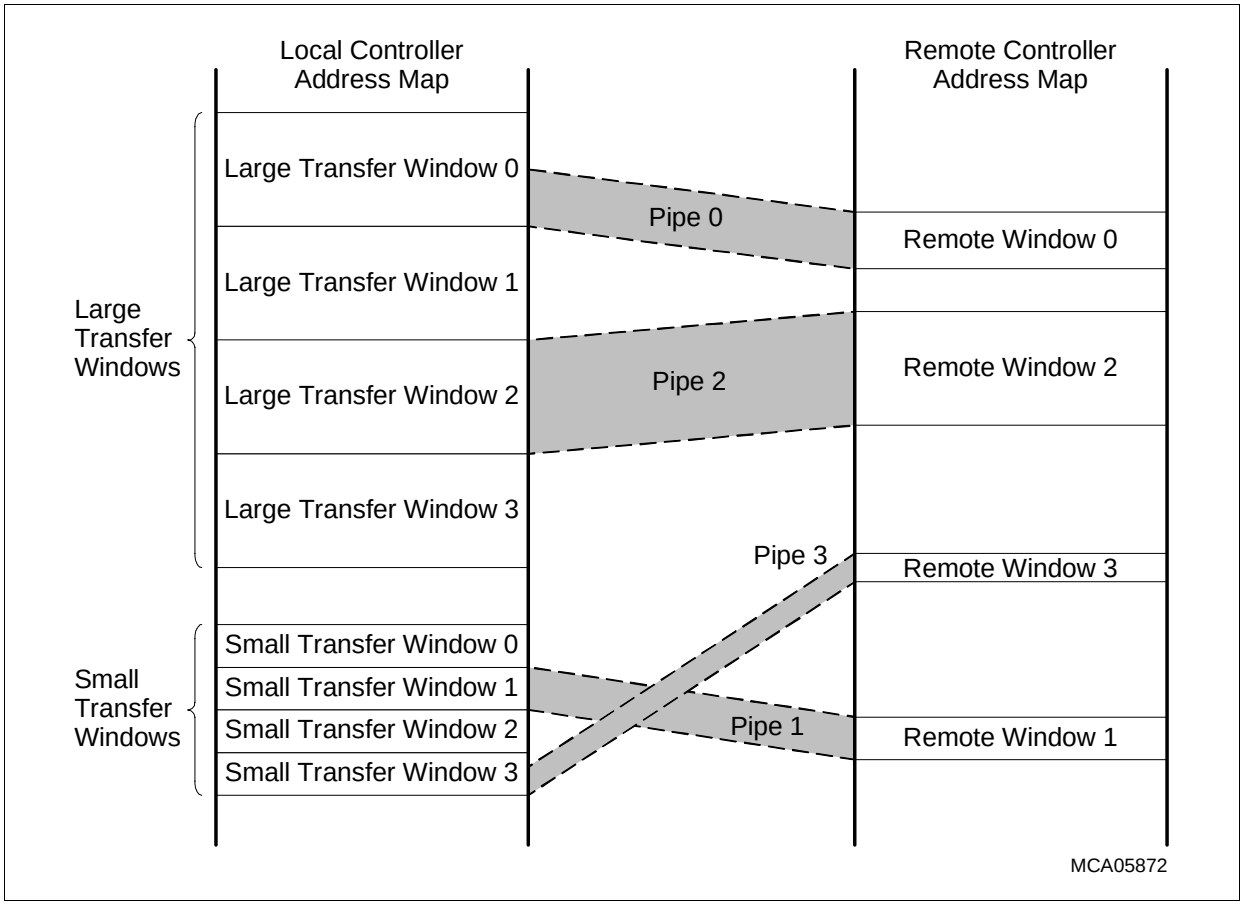


Figure 4-3 Transfer/Remote Window Assignment Example

During initialization of the Pipes, the base addresses and sizes of the Remote Windows are transmitted from the Local Controller to the Remote Controller. In the example of **Figure 4-3**, two large Transfer Windows and two small Transfer Windows are assigned to Remote Windows. Pipe 1 and Pipe 2 cover the full range of their transfer and Remote Windows. Pipe 0 and Pipe 3 address only sub-areas of the related Transfer Windows.

Remote Windows can be freely moved and located within the complete address space of the Remote Controller. They are used to overlay address ranges of peripheral modules or internal memories.

Remote Window Address Definition

A Remote Window is defined for each Pipe. Each Remote Window is defined by a programmable base address and a size for each Pipe. Frames transfer only the address offset information to define an address within a Remote Window if no prediction is possible. A Remote Window can have a size of up to 64 KBytes.

4.1.4 MLI Frame Types

A frame is a message sent by an MLI transmitter to an MLI receiver. The fame type depends on the desired behavior:

The MLI protocol offers seven different frame types for communication.

- Copy Base Address Frame—defines the base address and size of a Remote Window
- Write Offset and Data Frame—transmits the write offset and the write data
- Discrete Read Frame—transmits read request with the read offset
- Command Frame—transmits a command (e.g. setup information or MLI Request generation)
- Optimized Write Frame—transmits write data without a write offset (in case of an address prediction match)
- Optimized Read Frame—transmits the read request without a read offset (in case of an address prediction match)
- Answer Frame—transmits the data previously requested by a read frame

The local/remote structure of an MLI connection between two microcontrollers requires a transmitter unit and a receiver unit in both MLI modules (local and remote) for communication. Physically, the communication is performed via two separate serial MLI buses. Logically, the information flow of each MLI frame can be assigned to one of the MLI-Buses (see **Figure 4-4**).

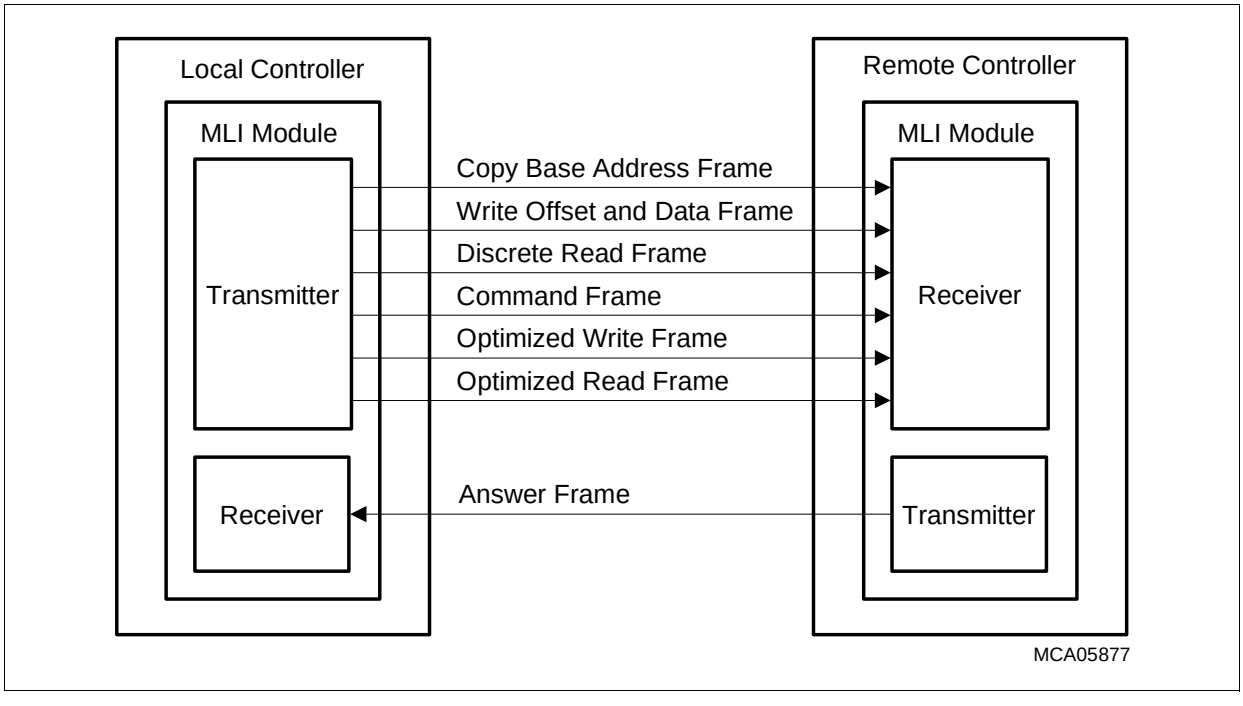


Figure 4-4 Logic Frame Assignment to Local/Remote Controller

The general layout of a frame is shown in **Figure 4-5**. It contains the following parts:

- A frame starts with a 4-bit header field that contains a 2-bit Frame Code (FC) and a 2-bit Pipe Number (PN).
- The data field can contain address, data, or control information. The length of the data field depends on the frame type.
- The frame is terminated by a parity bit (P) with even parity (see **Page 4-17**), which is calculated over header and data field bits.

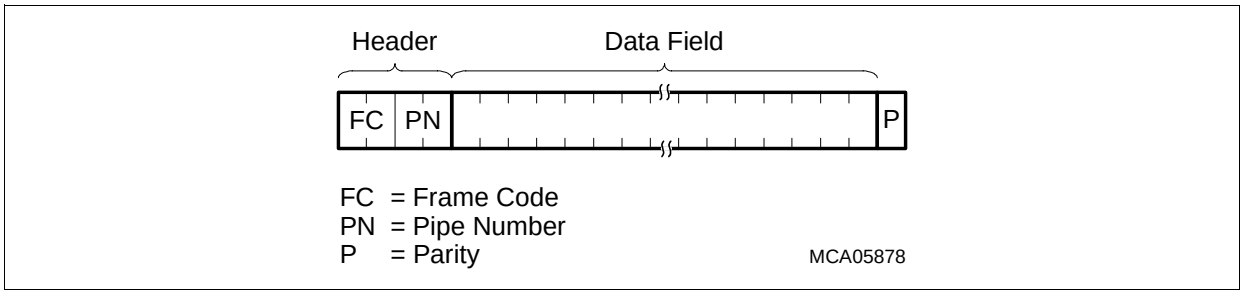


Figure 4-5 General Frame Layout

The Frame Code (FC) and the length of the data field determines the frame type of the transmitted frame.

The Pipe Number (PN) indicates the Pipe that is related to the frame content (the value of PN is defined as 00_B for Pipe 0, 01_B for Pipe 1, 10_B for Pipe 2, and 11_B for Pipe 3).

The FC parameter is coded according to **Table 4-1**.

Table 4-1 Frame Code Definition

Frame Code FC	Data Field Length	Frame Type	Description
00_B	32 Bits	Copy Base Address Frame	**Page 4-8**
01_B	8 + m, 16 + m, or 32 + m Bits	Write Offset and Data Frame	**Page 4-9**
	6 + m Bits	Discrete Read Frame	**Page 4-11**
10_B	4 Bits	Command Frame	**Page 4-13**
	8, 16, or 32 Bits	Answer Frame	**Page 4-14**
11_B	8, 16, or 32 Bits	Optimized Write Frame	**Page 4-10**
	2 Bits	Optimized Read Frame	**Page 4-12**

4.1.4.1 Copy Base Address Frame

With a Copy Base Address Frame, the two parameters (base address and size) of a Remote Window are transferred from the Local Controller to the Remote Controller to initialize or to update the Remote Window.

The Copy Base Address Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 00_B followed by the Pipe number PN of the Pipe to which the transmitted base address bits and the size are assigned.
- Remote Window base address:
  The 28 most significant bits of the 32 bit base address bits can be programmed by the Local Controller (the four LSBs are considered as 0). The base address of a Remote Window must be aligned to its size, e.g. a window of 1 KByte to start at 1 KByte address boundaries.
- Remote Window size:
  The size is defined by the 4-bit coded size BS. The maximum size is 64 KBytes.
- Parity bit P

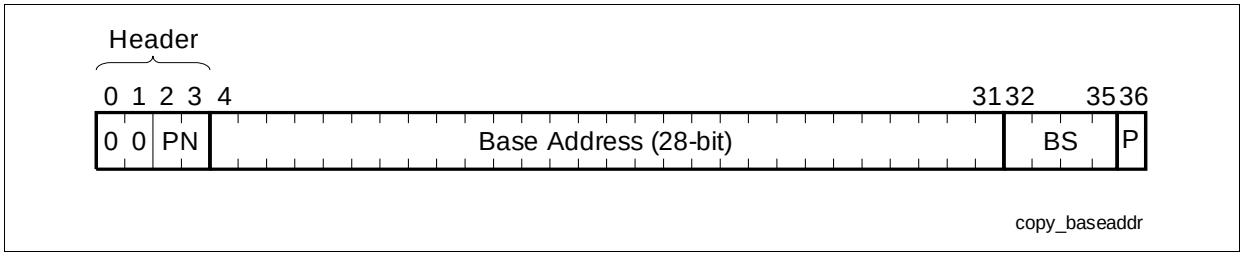


Figure 4-6 Copy Base Address Frame

Table 4-2 Size Coding

BS	Remote Window Size	Number of Offset Bits
0000_B	2 bytes	1
0001_B	4 bytes	2
...	...	...
1110_B	32 KBytes	15
1111_B	64 KBytes	16

Details about the Copy Base Address Frame handling of the CIC751 are provided in **Chapter 4.2.1.1**.

4.1.4.2 Write Offset and Data Frame

A Write Offset and Data Frame are used by the Local Controller to send an address offset and data to the Remote Controller.

The Write Offset and Data Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 00_B followed by the Pipe number PN of the Transfer Window that has been the target of the write operation.
- m-Bits of write offset:
  These bits define the write offset. The value of m depends on the size of the Remote Window, defined by the Copy Base Address Frame (m = 1-16).
- Write data field:
  The write data field can be 8, 16, or 32 bit wide, depending on the data width of the write access to the Transfer Window.
- Parity bit P.

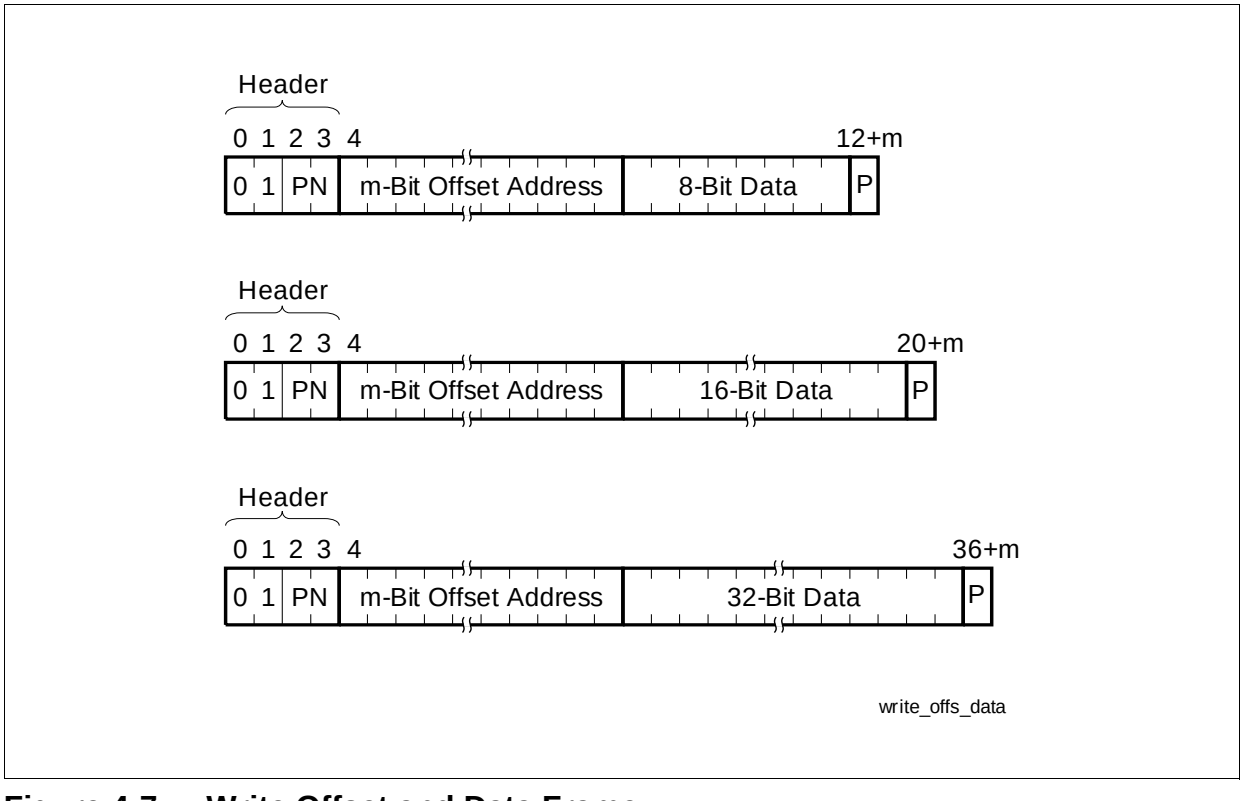


Figure 4-7 Write Offset and Data Frame

Details about the Write Offset and Data Frame handling of the CIC751 are provided in **Chapter 4.2.1.2**.

4.1.4.3 Optimized Write Frame

An Optimized Write Frame is used by the Local Controller to send 8-bit, 16-bit, or 32-bit wide data to the Remote Controller. In contrast to a Write Offset and Data Frame, no write offset is transmitted because the offset address for the write data is predicted and calculated by the MLI receiver of the Remote Controller. An Optimized Write Frame allows a higher data bandwidth than Write Offset and Data Frames. An optimized frame is only send if the predicted address matches with the actually written address within the Local Controller. Otherwise, an Write Offset and Data Frame is generated.

The Write Offset and Data Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 11_B followed by the Pipe number PN of the Transfer Window that has been the target of the write operation.
- Write data field:
  The write data field can be 8, 16, or 32 bit wide, depending on the data width of the write access to the Transfer Window.
- Parity bit P.

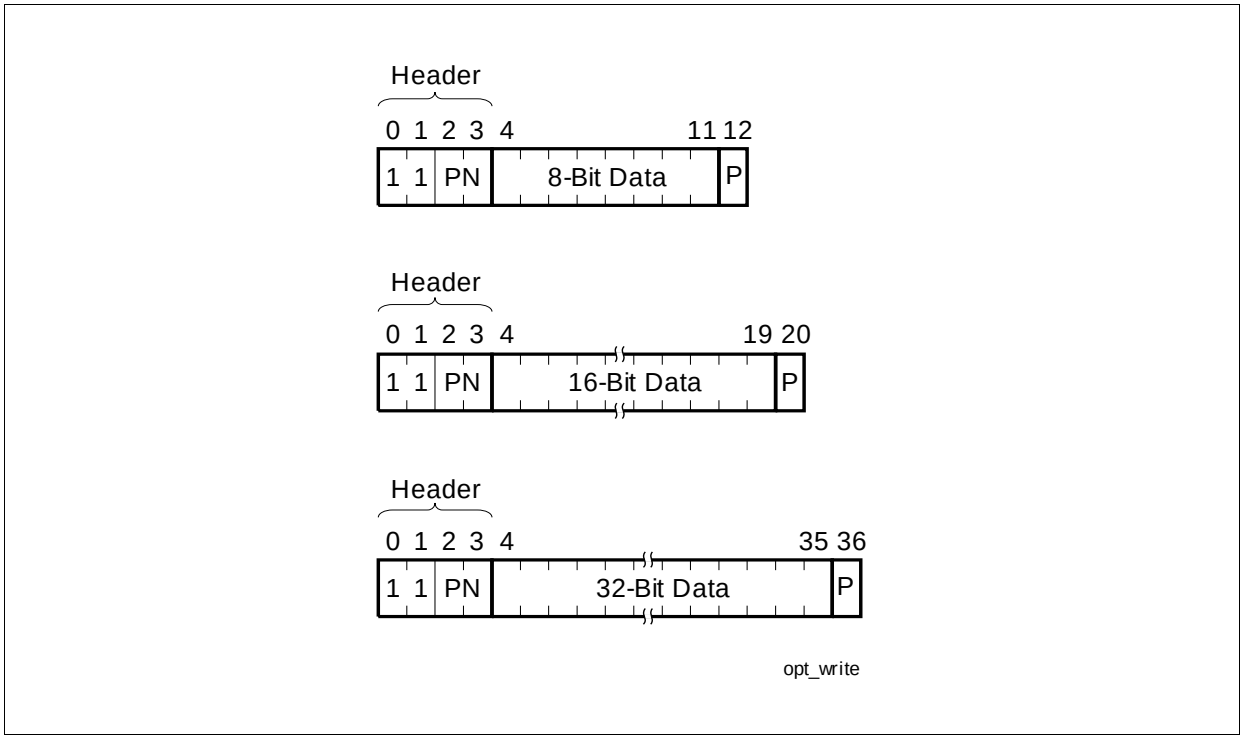


Figure 4-8 Optimized Write Frame

Details about the Optimized Write Frame handling of the CIC751 are provided in **Chapter 4.2.1.2**.

4.1.4.4 Discrete Read Frame

A Discrete Read Frame is used by the Local Controller to request data to be read from the Remote Window in the Remote Controller. If the read data is available, the Remote Controller responds to this request by sending an Answer Frame with the requested read data back to the Local Controller.

The Discrete Read Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 01_B followed by the Pipe number PN of the Transfer Window that has been the target of the read operation.
- m-Bits of read offset:
  These bits define the read offset. The value of m depends on the size of the Remote Window, defined by the Copy Base Address Frame (m = 1-16).
- Data Width DW:
  The data width DW indicates if the read from the Remote Window was an 8-, 16-, or 32-bit read action. It defines how many bytes must be delivered to the Local Controller by the Answer Frame.
- Parity bit P.

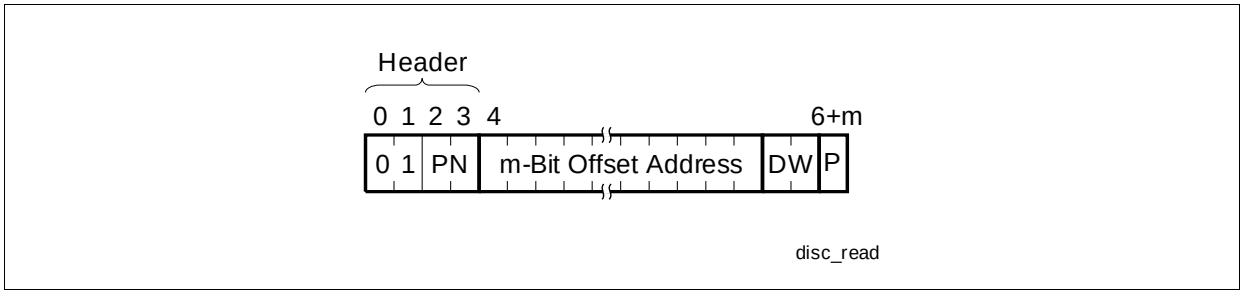


Figure 4-9 Discrete Read Frame

Table 4-3 Data Width DW Coding

Data Width DW	Number of Data Bits to be transferred
00_B	8-bit read access
01_B	16-bit read access
10_B	32-bit read access
11_B	Reserved

Details about the Discrete Read Frame handling of the CIC751 are provided in **Chapter 4.2.1.3**.

4.1.4.5 Optimized Read Frame

An Optimized Read Frame is used by the Local Controller to request 8-bit, 16-bit, or 32-bit wide data from the Remote Controller without sending any offset address. The address for the requested data is predicted and calculated by the MLI receiver of the Remote Controller.

The Optimized Read Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 11_B followed by the Pipe number PN of the Transfer Window that has been the target of the read operation.
- Data Width DW:
  The data width DW indicates if the read from the Transfer Window was an 8-, 16-, or 32-bit read action. It defines how many bytes must be delivered to the Local Controller by the Answer Frame. Same coding as for the Discrete Read Frame.
- Parity bit P.

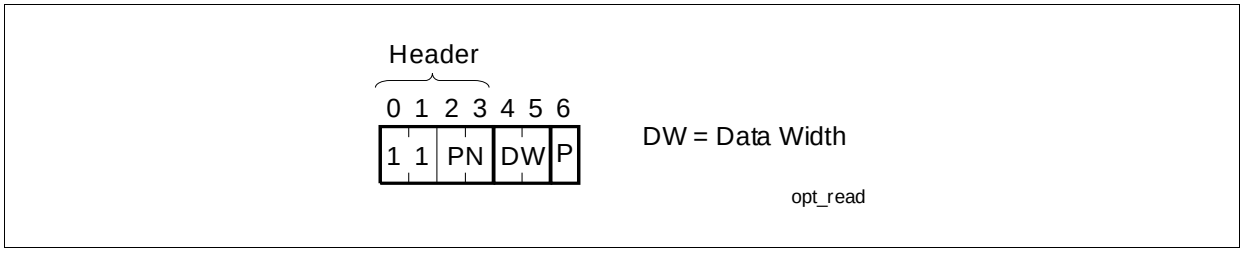


Figure 4-10 Optimized Read Frame

Table 4-4 Data Width DW Coding

Data Width DW	Number of Data Bits to be transferred
00_B	8-bit read access
01_B	16-bit read access
10_B	32-bit read access
11_B	Reserved

Details about the Optimized Read Frame handling of the CIC751 are provided in **Chapter 4.2.1.2**.

4.1.4.6 Command Frame

The Local Controller is able to initiate control actions to be executed by the Remote Controller by sending a Command Frame.

The Command Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 10_B followed by the Pipe number PN. The Pipe number defines the type of command to be executed.
- Command Code CMD:
  Pipe number PN and a 4-bit CMD field are used for command coding. The command coding of some control actions is fixed, but free programmable software commands can also be defined (with PN = 11_B). The coding of the command bit field is Pipe-specific and depends on the transmitted Pipe Number n.
- Parity bit P.

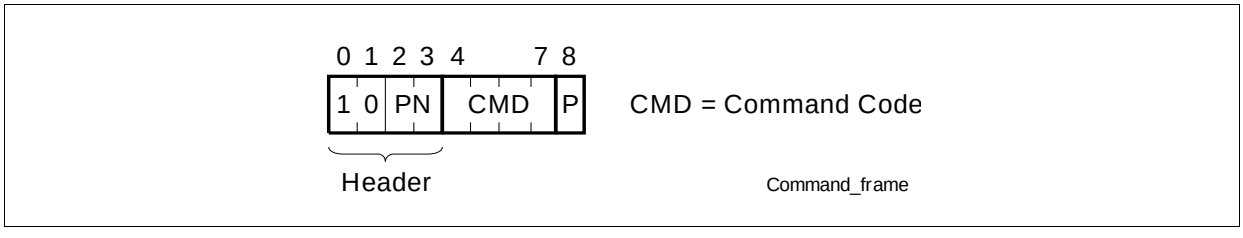


Figure 4-11 Command Frame

Table 4-5 PN for Command Coding

Pipe Number PN	Command Type
00_B	Activate MLI Request generation or other control signal(s) of the Remote Controller. The identity of which signal becomes activated is defined by CMD. The use of these lines depends on the product.
01_B	Define delay for parity error indication in the Remote Controller. The delay in RCLK cycles is defined by the value of CMD.
10_B	Control of internal functions of the Remote Controller. The value of CMD indicates which function is controlled. The coding of CMD and the control mechanisms depend on the product.
11_B	Freely programmable software command.

Details about the Command Frame handling of the CIC751 are provided in **Chapter 4.2.1.5**.

4.1.4.7 Answer Frame

An Answer Frame is used by the Remote Controller to send 8-bit, 16-bit, or 32-bit wide data to the Local Controller. The Answer Frame is the only frame that is transmitted within a logic Local/Remote Controller assignment from the Remote Controller to the Local Controller. It is the answer to a Discrete Read Frame or an Optimized Read Frame that has been sent by the Local Controller to request data from the Remote Controller.

The Answer Frame contains the following parts:

- Header:
  The header starts with Frame Code FC = 10_B followed by the Pipe number PN. The value of PN is taken from the read frame that has triggered the Answer Frame.
- Read data field:
  The read data field can be 8, 16, or 32bits wide, depending on the data width requested by the read frame that triggered the Answer Frame.
- Parity bit P.

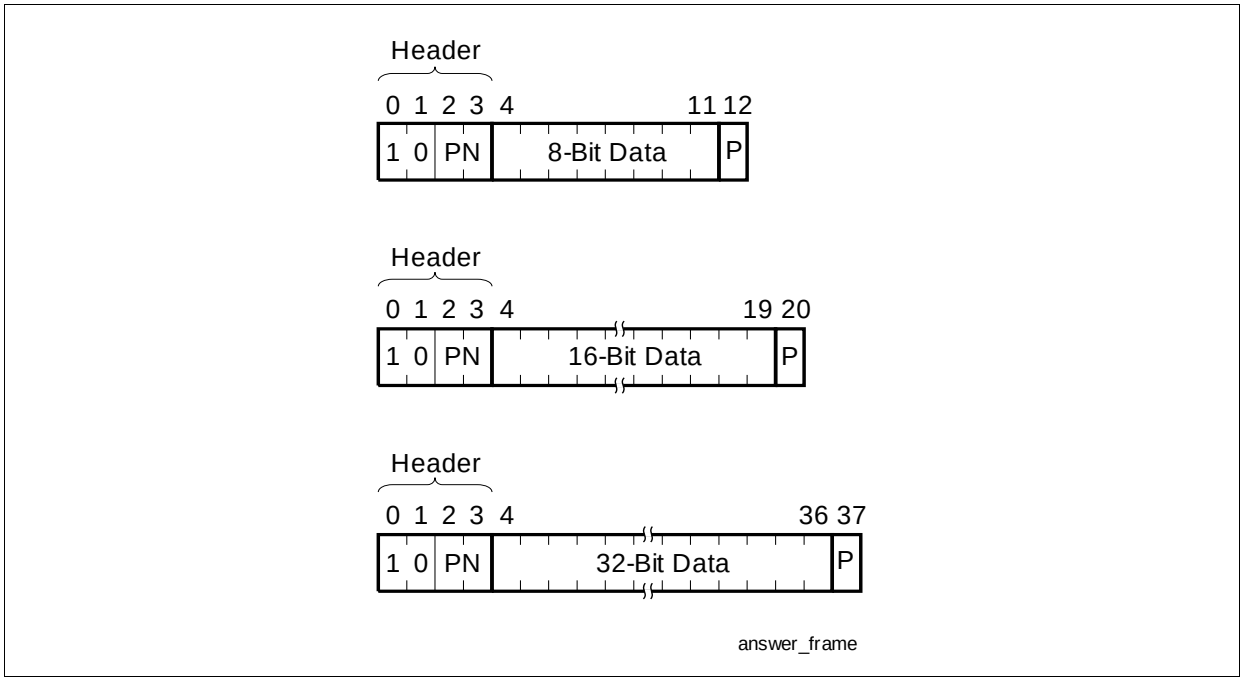


Figure 4-12 Answer Frame

Details about the Answer Frame handling of the CIC751 are provided in **Chapter 4.2.1.4**.

4.1.5 Naming Conventions

MLI module transmitter I/O signals are indicated with the prefix “T” and MLI receiver I/O signals are indicated with the prefix “R”.

The 4-line MLI-Bus between a transmitter and a receiver outside the controllers uses signal names without any prefix, as referred to in the timing diagrams of this section.

In order to emphasize where a signal is generated or sampled, actions taken by the transmitter are described by referring to signals with the prefix “T”, whereas receiver actions are referred to by signals with the prefix “R”.

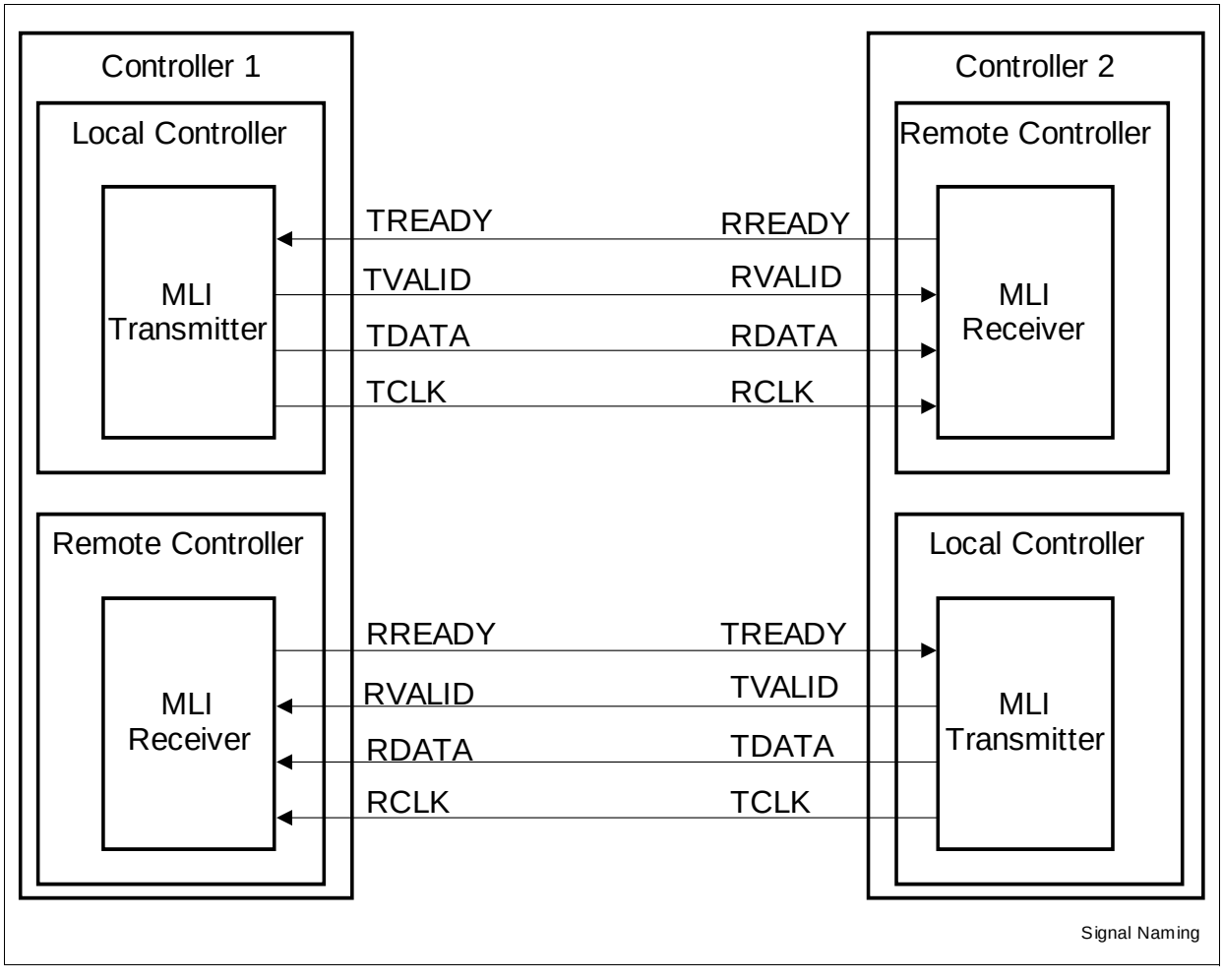


Figure 4-13 Transmitter/Receiver Signal Definitions

4.1.6 MLI Communication Examples

The following section provides some basic example of the MLI communication from the point of view of the transmitter.

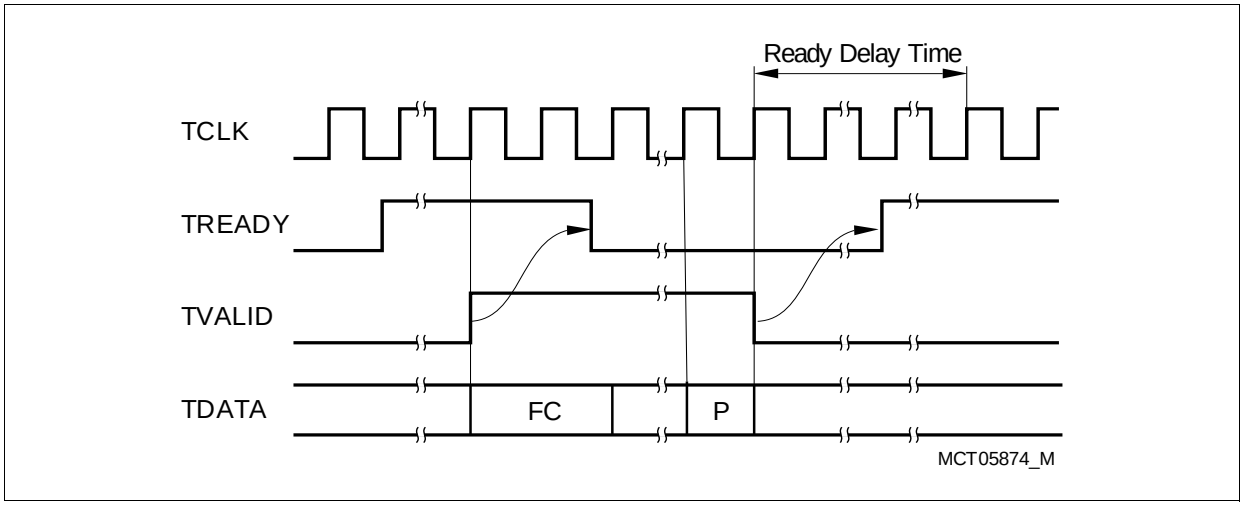


Figure 4-14 MLI Communication without Error (Transmitter View)

A transmission can be started by an MLI transmitter when the MLI receiver is ready to receive frames, which is indicated by TREADY = 1. When the MLI transmitter detects TREADY = 1 and starts its transmission, TVALID is asserted and is held as long as frame data is sent out. When the MLI receiver has detected the falling edge of the RVALID signal, it will de-assert RREADY (transmission start acknowledged by receiver). At the end of the frame transmission, the MLI transmitter de-asserts the TVALID signal and checks if the TREADY signal is also de-asserted. This check is used as the life-sign of the receiver and the MLI transmitter can detect whether the receiver is able to react in-time to the transmitter actions.

4.1.6.1 Ready Delay Time

In order to support significant propagation delays, the signal TREADY is evaluated with respect to TVALID and TCLK in a time interval called Ready Delay Time (see **Figure 4-14**).

When a transmission is finished (RVALID becomes 0), the MLI receiver checks the received frame for correct reception (parity error). In the case of correct reception, it asserts RREADY to indicate the correct reception with the next falling edge of RCLK. The MLI transmitter checks TREADY with respect to TVALID becoming 0 by counting TCLK periods with a ready delay counter. The ready delay counter is started from 0 at the end of a frame transmission (TVALID becomes 0). If the TREADY = 1 is detected and the ready delay counter value is less than a programmed value, it is assumed that the MLI receiver has received the frame without a parity error and a new frame can be transmitted by the MLI transmitter. An MLI transfer without a parity error condition is shown in **Figure 4-14**.

Figure 4-23 shows a transfer with a parity error detected by the MLI receiver. In this case, the receiver waits a programmed number of RCLK clock cycles before setting RREADY to 1. If TREADY = 1 is detected by the transmitter and the ready delay counter

value is greater than the programmed value, an reception error condition has been signalled. For this case, it is assumed that the MLI receiver has received the frame with a parity error and has discarded the frame. In this case, the transmitter automatically sends the last frame again.

4.1.6.2 Non-Acknowledge Error

The transmitter of the Local Controller is able to detect an inoperable receiver in the Remote Controller. Such a non-acknowledge error condition is detected by the transmitter when at the end of a frame transmission the TREADY signal is still at high level (TREADY = 1 when TVALID becomes 0). **Figure 4-15** shows the non-acknowledge error case. In this case, the transmitter automatically sends the last frame again.

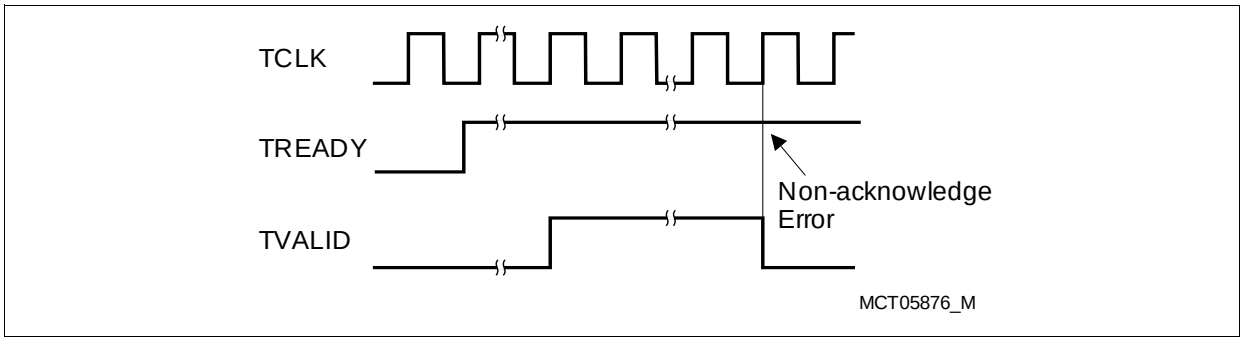


Figure 4-15 Non-Acknowledge Error

4.1.7 Parity Generation

For parity generation, the number of transmitted bits with the value of 1 is counted over the header and the complete data field of a frame. For even parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 1. For error-free MLI traffic, even parity generation and checking is defined.

Details about the parity handling of the CIC751 are provided in **Chapter 4.2.2.1**.

4.1.8 Address Prediction

An address prediction mechanism supports communication between the MLI transmitter and the MLI receiver without sending address offset information in the frames. This feature reduces the required bandwidth for MLI communication. Both of the communication partners, the MLI transmitter and the receiver, are able to detect regular offset differences of consecutive window accesses to the same window. The address prediction mechanism operates independently for each Pipe; different prediction values can be handled in parallel for the different Pipes.

The MLI transmitter can compare the offset of each Transfer Window read or write access with the offset of the previous access to the same Transfer Window. Between

accesses to a specific window, other windows can be accessed without disturbing the prediction. Offset differences larger than 512 bytes are not supported by the address prediction.
If, in at least two accesses, the offset differences are identical, an address prediction is possible and Optimized Write Frames or Optimized Read Frames can be sent to the receiver in the Remote Controller for this Pipe. If the offset difference of the next access to this Transfer Window does not match the previous ones (predicted offset), address prediction is not possible. In this case, a Normal Frame for writing or reading (Write Offset and Data Frame or Discrete Read Frame) is started.
Identical address prediction mechanisms are used by both the transmitter and the receiver. As a result, the receiver can elaborate on the original offset value in the transmitter when receiving an optimized frame for any Pipe.

Details about the prediction mechanism of the CIC751 are provided in **Chapter 4.2.2.3**.

4.2 Functional Description

This chapter describes the functionality of the MLI interface and how frame handling can be done by software.

- Frame handling (see **Page 4-19**)
- MLI features (see **Page 4-34**)
- MLI Request structure (see **Page 4-41**)
- MLI transmitter interrupts (see **Page 4-41**)
- MLI receiver interrupts (see **Page 4-43**)
- Baud rate generation (see **Page 4-45**)

4.2.1 Frame Handling

Frame handling is based on receiver and transmitter registers and the Transfer Windows. Depending on the type of access to the Transfer Windows, different actions take place inside the MLI interface. Please refer to the following sections for the handling of specific frame types, see the pages indicated:

- Copy Base Address Frame (see **Page 4-19**)
- Data frames (see **Page 4-22**)
- Read frames (see **Page 4-26**)
- Answer Frame (see **Page 4-29**)
- Command Frame (see **Page 4-30**)

4.2.1.1 Copy Base Address Frame

A Copy Base Address Frame defines the base address and the size of a Remote Window.

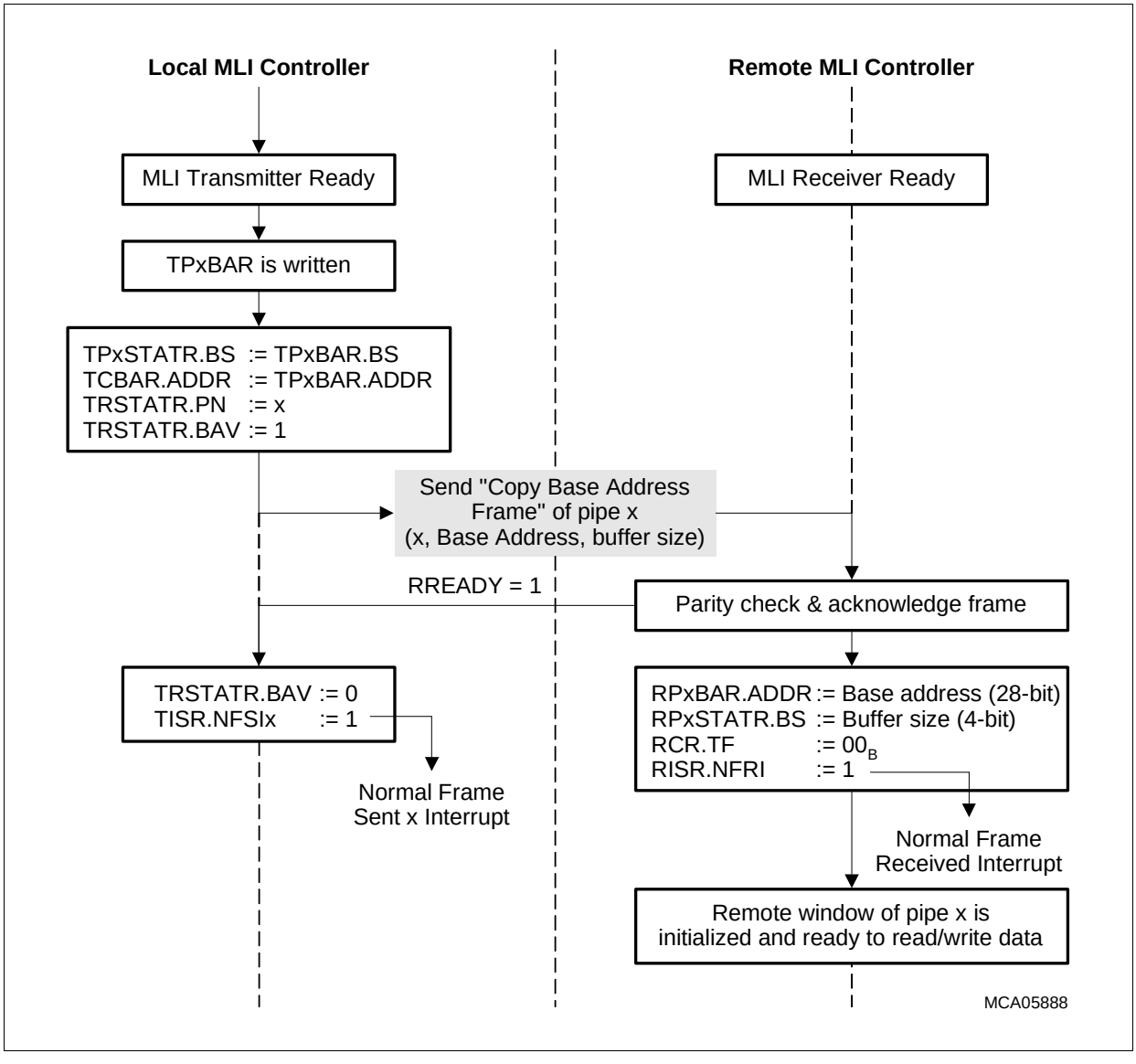


Figure 4-16 Copy Base Address Frame Transaction Flow

Local Controller

The transmission of a Copy Base Address Frame is started, each time a transmitter Pipe x base address register MLI_TPxBAR (x = 0-3) is written, triggering the following actions for Pipe x.

- Bit field MLI_TPxBAR.BS (x = 0-3) is written to bit field MLI_TPxSTATR.BS (x = 0-3)
- Bit field MLI_TPxBAR.ADDR (x = 0-3) is written to bit field **MLI_TCBAR**.ADDR.
- Status bit field **MLI_TRSTATR**.PN is updated with Pipe Number x (for example x = 2 when MLI_TP2BAR has been written).
- Status flag **MLI_TRSTATR**.BAV (base address valid) is set.
- The transmission of a Copy Base Address Frame with the two parameters **MLI_TCBAR**.ADDR and MLI_TPxSTATR.BS is started for Pipe x.

- Status flag **MLI_TRSTATR**.BAV (in the Local Controller) is cleared after the Copy Base Address Frame has been finished and correctly acknowledged by the MLI receiver of the Remote Controller.
- Interrupt status flag **MLI_TISR**.NFSIx is set and an MLI Request is generated if enabled by **MLI_TIER**.NFSIEx = 1.

Note: After the transfer of a Copy Base Address Frame, the optimized mode will be suppressed automatically for the next two data frames. This ensures a correct offset prediction afterwards.

Remote Controller

When a Copy Base Address Frame for Pipe x has been received correctly and acknowledged, the following actions are executed in the MLI receiver of the Remote Controller.

- The received address bits are stored in the receiver Pipe x base address register bit field **MLI_RP0BAR**.ADDR. This bit field determines the base address of the Pipe x Remote Window.
- The received size is stored in the receiver Pipe x status register bit field **MLI_RP0STATR**.BS. This bit field determines the number of offset address bits of the Pipe x Remote Window.
- The information about the received frame type (= 00_B for Copy Base Address Frame) is stored in the receiver control register bit field **MLI_RCR**.TF.
- Interrupt status flag **MLI_RISR**.NFRI (Normal Frame received) is set and an MLI Request is generated if enabled by **MLI_RIER**.NFRIE = 01_B or 10_B.

4.2.1.2 Data Frames

Data frames transmit the write data and (optionally) the write offset.

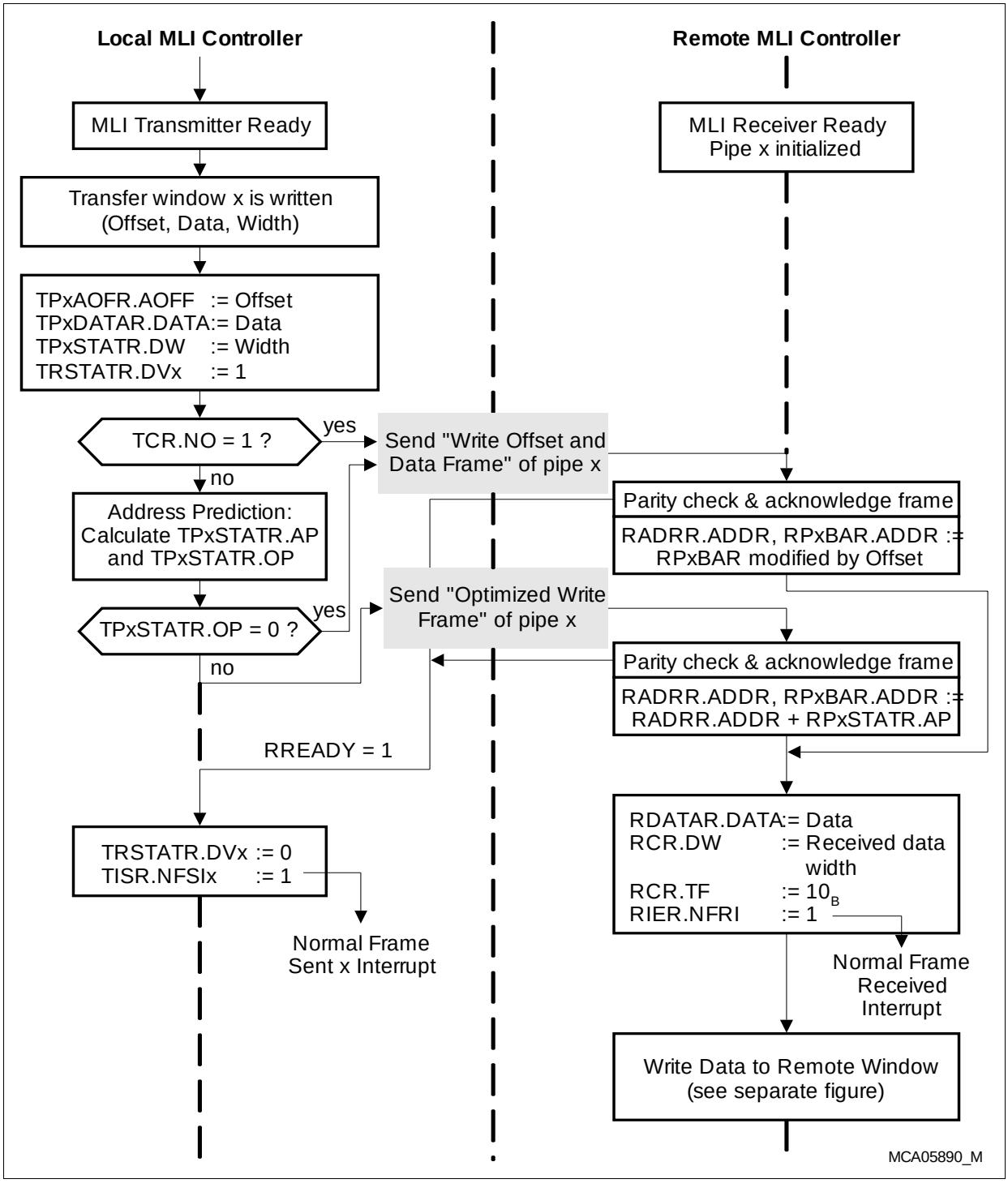


Figure 4-17 Write Frame Transaction Flow

Local Controller

In the Local Controller, a write operation to a Transfer Window address defines the address, the data, and the data size and triggers the following actions in the MLI transmitter.

- The 16 least significant address bits of the Transfer Window write access are stored in MLI_TPxAOFR.AOFF (x = 0-3) as write offset address.
- The data of the write access is stored in MLI_TPxDATAR.DATA (x = 0-3).
- The data width of the Transfer Window write access (8-bit, 16-bit, or 32-bit) is stored in bit field MLI_TPxSTATR.DW (x = 0-3).
- Status flag **MLI_TRSTATR**.DVx is set, indicating that the Pipe contains valid data for transmission.
- If the address prediction is disabled (**MLI_TCR**.NO = 1), the transmission of a Write Offset and Data Frame is started as soon as the MLI transmitter is idle.
  If the address prediction is enabled (**MLI_TCR**.NO = 0), a Write Offset and Data Frame is started only if an address prediction is not possible (indicated by **MLI_TP0STATR**.OP = 0). If **MLI_TP0STATR**.OP = 1, an address prediction is possible in the MLI transmitter and an Optimized Write Frame is started. The address prediction is described in **Chapter 4.2.2.3**.
- Status flag **MLI_TRSTATR**.DVx is cleared after the Write Offset and Data Frame or the Optimized Write Frame has been finished and correctly acknowledged by the MLI receiver of the Remote Controller.
- Interrupt status flag **MLI_TISR**.NFSIx is set and an MLI Request is generated if enabled by **MLI_TIER**.NFSIEx = 1.

The number of offset address bits that are transmitted at a Write Offset and Data Frame is determined by the size of the Remote Window that has been previously initialized by the transmission of a Copy Base Address Frame.

Remote Controller

After a data frame has been received correctly and acknowledged, the following actions are executed in the MLI receiver of the Remote Controller:

- In the case of a Write Offset and Data Frame:
  The result of the internal address prediction is not taken into account. The received offset address is added to the base address of the Pipe x Transfer Window and the result is stored in MLI_RPxBAR.ADDR (x = 0-3) and MLI_RADRR.ADDR.
  In the case of an Optimized Write Frame:
  The result of the internal address prediction is taken into account. The next address in the Remote Controller to that data are written is calculated by adding the detected receiver address prediction value MLI_RPxSTATR.AP (x = 0-3) to the actual address (MLI_RPxBAR.ADDR (x = 0-3)) and the result is stored in MLI_RPxBAR.ADDR (x = 0-3) and in MLI_RADRR.ADDR.
- The received data is stored in the receiver data register **MLI_RDATAR** (right aligned, unused bits are 0).

- The data width of the received data is stored in bit field **MLI_RCR**.DW.
- The information about the received frame type (= 10_B for a write frame) is stored into bit field **MLI_RCR**.TF.
- Interrupt status flag **MLI_RISR**.NFRI is set and an MLI Request is generated if enabled by **MLI_RIER**.NFRIE = 01_B or 10_B.

After all these actions related to the reception of a write frame by the remote receiver are performed, the data that has been received from the Local Controller is ready to be written into the Remote Window related to the receiving Pipe.
This write operation can be executed in two ways:

- **MLI_RCR**.MOD = 0: Automatic Data Mode is disabled.
  In this mode, the DMA is request by an MLI Request generated for the Normal Frame received interrupt **MLI_RISR**.NFRI (if enabled by **MLI_RIER**.NFRIE = 10) to transfer the received write data from the MLI receiver to the Remote Window address. Therefore, it must read the data from **MLI_RDATAR**, together with width **MLI_RCR**.DW and the address stored in **MLI_RADRR** and write it to the indicated address location.
- **MLI_RCR**.MOD = 1: Automatic Data Mode is selected.
  In this mode, the MLI automatically writes the received write data to the Remote Window address and sets interrupt status flag **MLI_RISR**.MEI when the access is terminated. An MLI Request is generated if enabled by **MLI_RIER**.MEIE = 1.

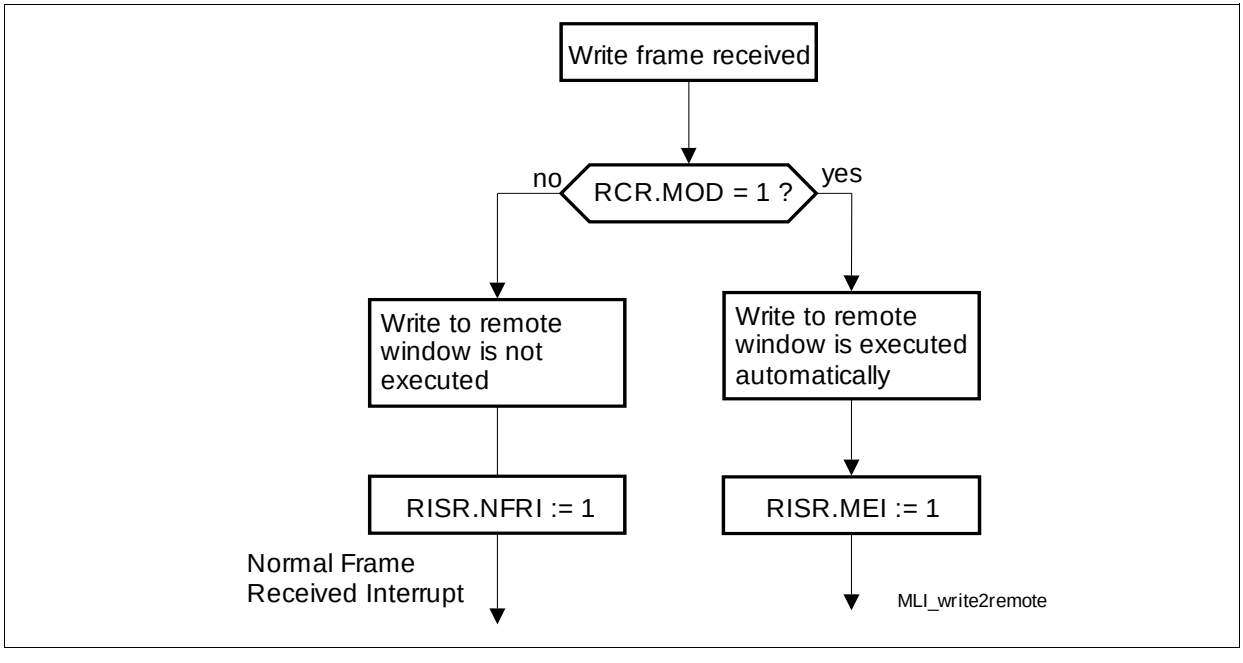


Figure 4-18 Write Frame Handling on Remote Side

Note: In Automatic Data Mode, write frames lead to a write action executed by the MLI. During the move operation, only one new MLI frame can be received (stored in a waiting position to be executed). Then the reception of more frames is blocked by

non-acknowledge errors. If the move operation is finished, frame execution and reception continue normally.
If Automatic Data Mode is no selected, no blocking mechanism is present.

4.2.1.3 Read Frames

Read frames transmit the read request and (optionally) the read offset.

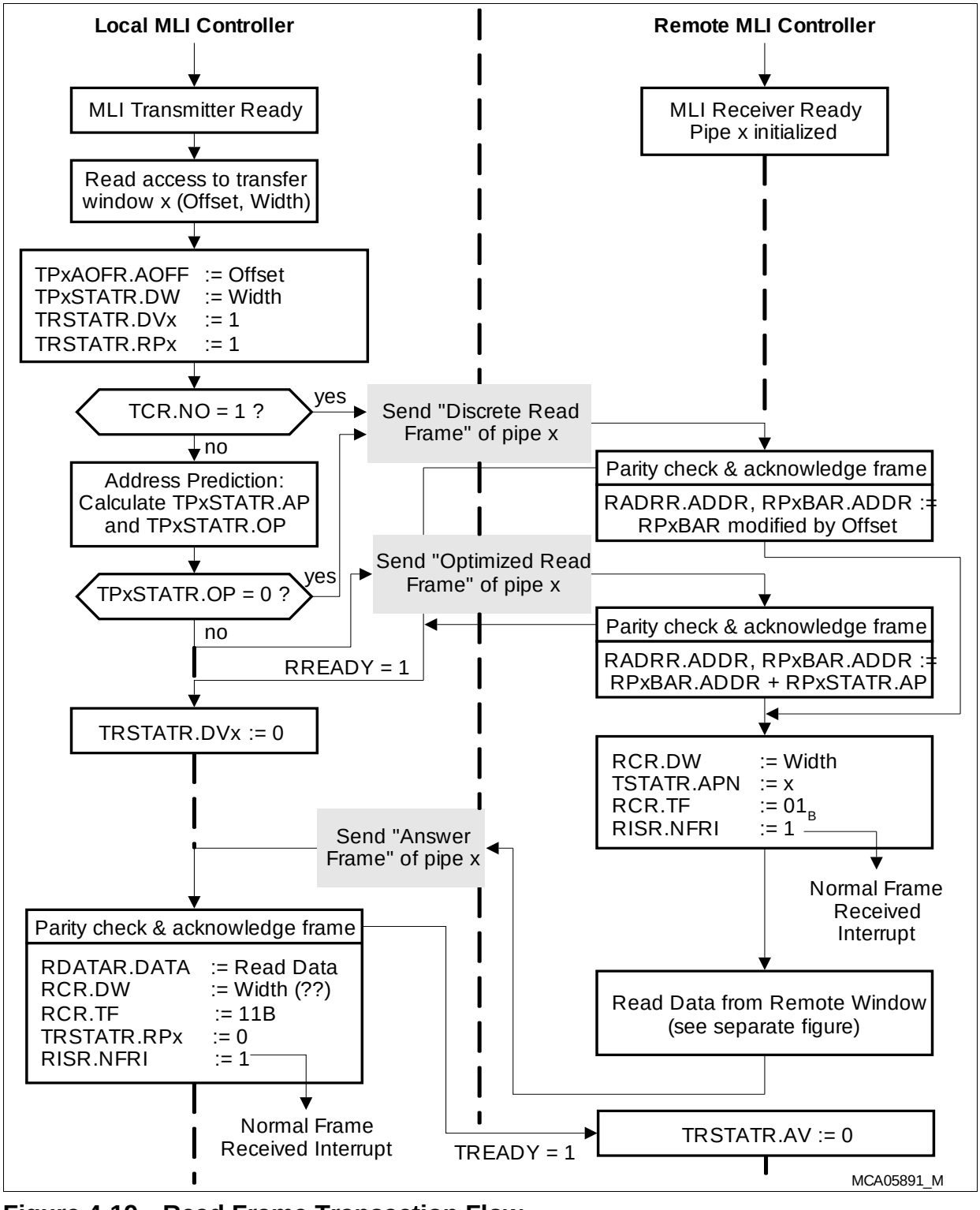


Figure 4-19 Read Frame Transaction Flow

Local Controller

A read operation from a location within a Transfer Window delivers a dummy value as result of the read action and triggers the transmission of a read frame from the Local to the Remote Controller.

- The 16 least significant address bits of the Transfer Window read access are stored in MLI_TPxAOFR.AOFF (x = 0-3) as read offset address.
- The data width of the Transfer Window read access (8-bit, 16-bit, or 32-bit) is stored in bit field MLI_TPxSTATR.DW (x = 0-3).
- Status flag **MLI_TRSTATR**.DVx is set.
- Status flag **MLI_TRSTATR**.RPx is set. This bit is cleared when an Answer Frame has been received correctly.
- If the address prediction is not enabled (**MLI_TCR**.NO = 1), transmission of a Discrete Read Frame is started. If the address prediction is enabled (**MLI_TCR**.NO = 0), a Discrete Read Frame is started only if an address prediction is not possible (indicated by MLI_TPxSTATR.OP = 0). If MLI_TPxSTATR.OP = 1, an address prediction is possible and an Optimized Read Frame is started.
- Status flag **MLI_TRSTATR**.DVx is cleared after the read frame has been finished and correctly acknowledged by the MLI receiver of the Remote Controller.

The number of offset address bits that are transmitted by a Discrete Read Frame is determined by the size of the Remote Window in the Remote Controller that has been previously initialized.

After the transmission of a read frame, the MLI expects the reception of an Answer Frame.

The Answer Frame is introduced, with the highest priority, into the data flow of the transmitter of the Remote Controller.

Remote Controller

After a read frame has been correctly received and acknowledged, the following actions are executed in the MLI receiver of the Remote Controller:

- In the case of a Discrete Read Frame:
  The result of the address prediction is not taken into account. The received offset address is added to the base address of the Pipe x (stored in MLI_RPxBAR.ADDR (x = 0-3)). The result of this addition is stored in both **MLI_RADRR**.ADDR and MLI_RPxBAR.ADDR and represents the source address of the Remote Controller where data should be read.
  In the case of an Optimized Read Frame:
  The result of the address prediction is taken into account. The next address in the Remote Controller where data is read is calculated by adding the detected receiver address prediction value MLI_RPxSTATR.AP (x = 0-3) to the actual address stored in MLI_RPxBAR.ADDR (x = 0-3). The result of this addition is stored in **MLI_RADRR**.ADDR and MLI_RPxBAR.ADDR and represents the destination address in the Remote Controller.

- The transmitted data width DW is stored in bit field **MLI_RCR**.DW.
- The information about the received frame type is stored in bit field **MLI_RCR**.TF.
- Interrupt status flag **MLI_RISR**.NFRI is set and an MLI Request is generated if enabled by **MLI_RIER**.NFRIE = 01_B or 10_B.

After correct reception of a read frame by the Remote Controller, the data requested by the Local Controller can be read by the Remote Controller and sent back to the Local Controller by an Answer Frame.
This read operation can be executed in two ways:

- **MLI_RCR**.MOD = 0:
  Automatic Data Mode is disabled. The DMA is requested by an MLI Request generated by the Normal Frame received interrupt to read the requested read data and transfer it to the MLI receiver. Therefore, it must read data with width **MLI_RCR**.DW from the address stored in **MLI_RADRR** and write the data into **MLI_TDRAR**.DATA.
- **MLI_RCR**.MOD = 1:
  Automatic Data Mode is enabled. In this mode, the MLI automatically reads the read data from the Remote Window and sets interrupt status flag **MLI_RISR**.MEI. An MLI Request is generated if enabled by **MLI_RIER**.MEIE = 1.
- After **MLI_TDRAR**.DATA has been updated, status flag **MLI_TRSTATR**.AV of the Remote Controller is set and the transmission of an Answer Frame is started.

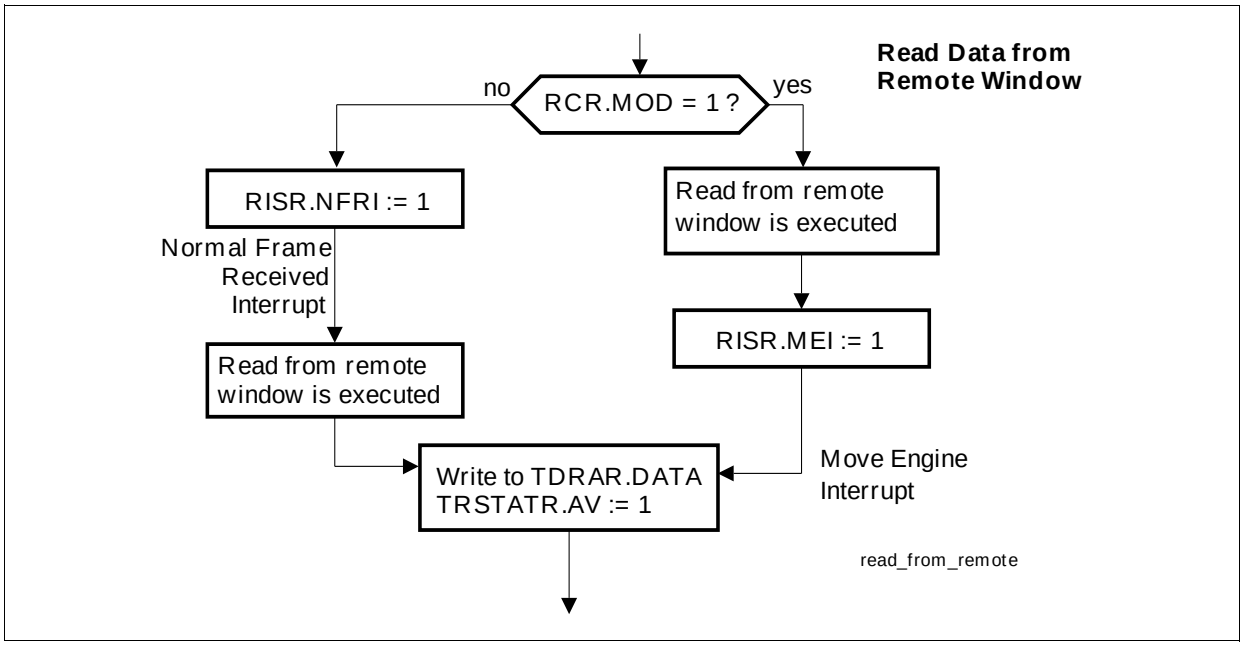


Figure 4-20 Read Frame Handling on Remote Side

Note: In Automatic Data Mode, read frames lead to a read action executed by the MLI. During the move operation, only one more MLI frame can be received (stored in a waiting position to be executed). Then, the reception of more frames is blocked by

non-acknowledge error. If the move operation is finished, frame execution and reception continue normally. If Automatic Data Mode is disabled, no blocking mechanism is present.

4.2.1.4 Answer Frame

Please note that only one Answer Frame can be handled at a time. No additional Read Frame should be requested while any **MLI_TRSTATR**.RPx bit is set. To ensure this a certain time-out criterion has to be defined and handled by Local Controller software. The Remote Controller should take care that no Answer Frame is delivered after the time-out criterion has been detected (e.g. by a software triggered Command Frame). The length of the time-out depends on the application and has to be defined accordingly on a case by case base (e.g. the transfer rates between Local Controller and Remote Controller etc. have to be considered). In the case a time-out has been detected, the Local Controller has to clear the **MLI_TRSTATR**.RPx bit by writing 1 to **MLI_SCR**.CDVx a new Read Frame can be started. If no time-out handling is supported Answer Frame data can be lost or corrupted.

Remote Controller

The Answer Frame is the only frame sent from the Remote Controller to the Local Controller. The transmitter part of the Remote Controller is used to generate the Answer Frame.

Every time the transmitter data read answer register **MLI_TDRAR** is updated in the Remote Controller, the transmission of an Answer Frame is started and the following actions are triggered.

- Status flag **MLI_TRSTATR**.AV is set to trigger the transmission of an Answer Frame.
- Status flag **MLI_TRSTATR**.AV is cleared after the Answer Frame has been finished and correctly acknowledged by the MLI receiver of the Local Controller.

An Answer Frame is sent through the same Pipe that was used by the read frame.

Local Controller

If an Answer Frame has been received correctly and acknowledged, the following actions are executed in the MLI receiver of the Local Controller:

- The **MLI_TRSTATR**.RPx flag is cleared.
- The received data is stored into the receiver data register **MLI_RDATAR**.
  If 8 data bits are received, they are duplicated to all 4 bytes in **MLI_RDATAR**.
  If 16 data bits are received, they are duplicated to both half-words in **MLI_RDATAR**.
- The data width of the received data is written to bit field **MLI_RCR**.DW.
- The received Pipe Number x represents the answer Pipe Number and is stored in bit field **MLI_TSTATR**.APN.
- The information about the received frame type (= 11_B for an Answer Frame) is stored in bit field **MLI_RCR**.TF.

- Interrupt status flag **MLI_RISR**.NFRI is set and an MLI Request is generated if enabled by **MLI_RIER**.NFRIE = 01_B or 10_B.
- The data that has been previously requested from the Remote Controller by a read frame is available in **MLI_RDATAR**.
- If an Answer Frame is received while the corresponding **MLI_TRSTATR**.RPx bit is cleared, the reception is declared as unintended and a discarded read answer event is generated (see **Page 4-43**).

Note: If an Answer Frame has been correctly received in the Local Controller, the software must read it. As long as at least one byte of this data has not yet been read out, only one more MLI frame can be received (stored in a waiting position to be executed). Then, the reception of more frames is blocked by a non-acknowledge error. If the received data has been read out, frame execution and reception continue normally.

4.2.1.5 Command Frame

Command Frames transmit a command (e.g. setup information).

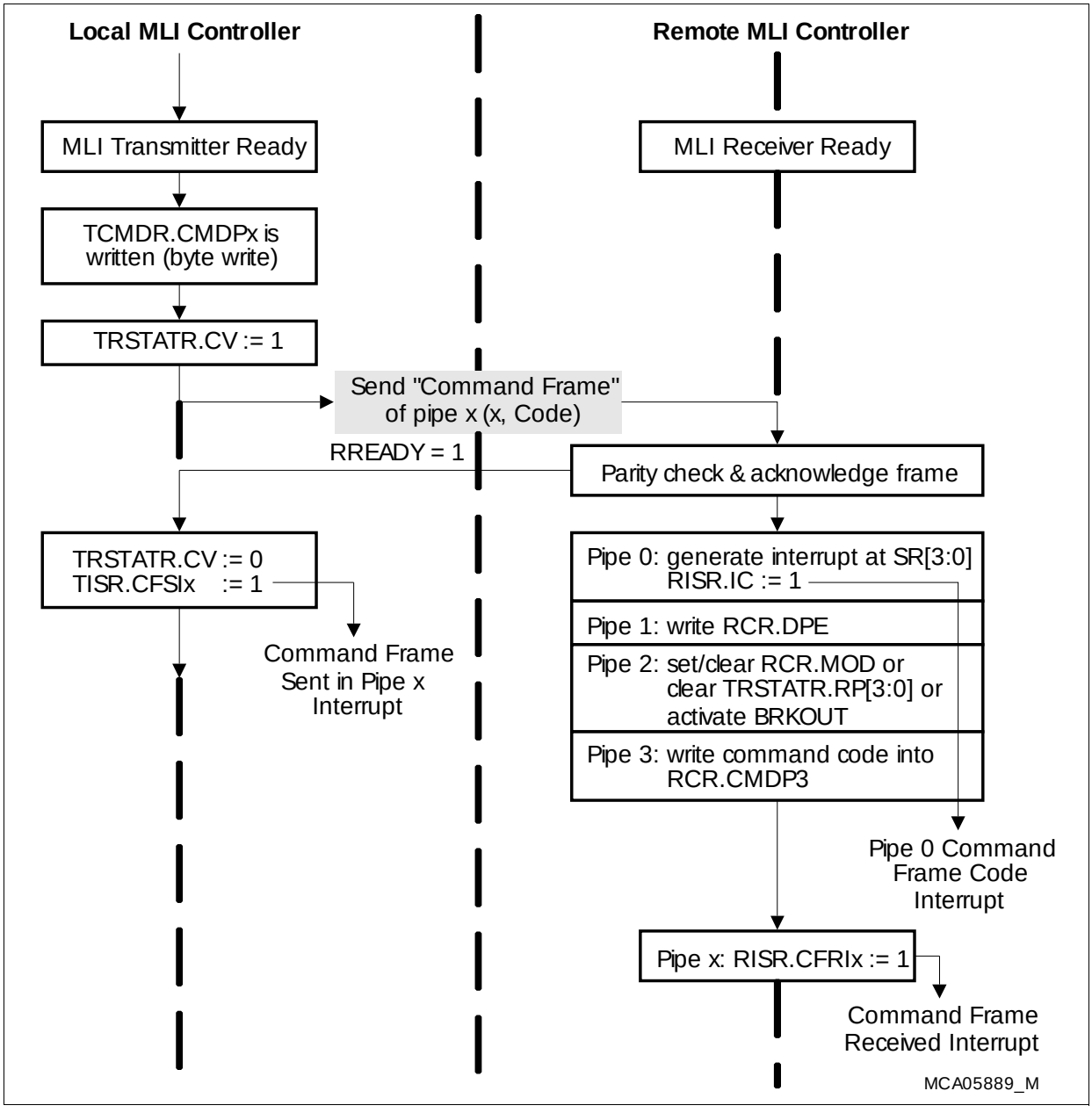


Figure 4-21 Command Frame Transaction Flow

Local Controller

The transmission of a Command Frame is initiated by writing one of the four Pipe x related command code bit fields in register **MLI_TCMDR**.CMDPx, triggering the following actions:

- Status flag MLI_TRSTATR.CVx is set and the Command Frame transmission is started using x as Pipe number PN and the command code stored in **MLI_TCMDR**.CMDPx as parameter.
- **MLI_TRSTATR**.CVx is cleared after the Command Frame has been finished and correctly acknowledged by the Remote Controller.
- Interrupt status flag **MLI_TISR**.CFSIx is set and an MLI Request is generated if enabled by **MLI_TIER**.CFSIEx = 1.

Remote Controller

Depending on the Pipe x related command code that is transmitted by a Command Frame, different actions are triggered in the Remote Controller. **Table 4-6** describes the actions that are triggered by a Command Frame.

- The received PN value is checked and the corresponding control actions are executed according to **Table 4-6**.
- Independent of the received Pipe Number, interrupt status flag **MLI_RISR**.CFRIx is set and an MLI Request is generated if enabled by **MLI_RIER**.CFRIEx = 1.

If a Command Frame is received for Pipe 2 with command code 1111_B, the MLI Break Event is generated if enabled (**MLI_RCR**.BEN = 1).

Table 4-6 Command Frame Actions for the Remote Controller

PN	CMD	Command Description
00_B	0001_B	Generate an MLI Request 0
	0010_B	Generate an MLI Request 1
	0011_B	Generate an MLI Request 2
	0100_B	Generate an MLI Request 3
	Others	No effect
01_B	0000_B	Set **MLI_RCR**.DPE to 0000_B
	0001_B	Set **MLI_RCR**.DPE to 0001_B
	0010_B	Set **MLI_RCR**.DPE to 0010_B
	...	...
	1111_B	Set **MLI_RCR**.DPE to 1111_B

Table 4-6 Command Frame Actions for the Remote Controller (cont'd)

PN	CMD	Command Description
10_B	0001_B	Select Automatic Data Mode (set **MLI_RCR**.MOD = 1)
	0010_B	Select Manual Remote Data Transfer Mode (set **MLI_RCR**.MOD = 0)
	0100_B	Clear bit **MLI_TRSTATR**.RP0
	0101_B	Clear bit **MLI_TRSTATR**.RP1
	0110_B	Clear bit **MLI_TRSTATR**.RP2
	0111_B	Clear bit **MLI_TRSTATR**.RP3
	1111_B	Generate MLI Break Event; (if enabled by **MLI_RCR**.BEN = 1)
	others	No effect
11_B	Any	Free programmable command, stored in the remote MLI receiver bit field **MLI_RCR**.CMDP3

4.2.2 General MLI Features

The following general features comprise the MLI:

- Parity generation and checking (see **Page 4-34**)
- Non-acknowledge error (see **Page 4-37**)
- Address prediction (see **Page 4-38**)
- Automatic data transfers (see **Page 4-39**)
- Transmit priority (see **Page 4-39**)

4.2.2.1 Parity Generation and Checking

For parity generation, the number of transmitted bits with the value of 1 is counted over the header and the complete data field of a frame. For even parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 1. For odd parity, the parity bit is set if the result of a modulo-2 division of the elaborated number is 0.

For a parity error-free MLI connection, even parity must be selected in the transmitter because the receiver operates only with even parity detection. The capability to select odd parity can be used by the transmitter to force a parity error reply from the receiver during the startup procedure of the MLI connection. This can be used to measure the propagation delay and to optimize the ready delay time.

Note: There is no protection against frames where more than one bit is corrupted (e.g. shortened frames). In such a case, unpredicted behavior of the MLI module may occur.

Local Controller

The MLI transmitter is able to count parity errors and to generate a parity error interrupt when a programmable number (max. 16) of parity errors has occurred. A parity error condition is indicated by Remote Controller after the transmission of a frame (see **Figure 4-23**). The transmitter parity error condition is detected when the TREADY signal is sampled at low level during a programmable number (**MLI_TCR**.MDP = maximum delay for parity errors) of TCLK clock cycles after TVALID is de-asserted.

When a transmitter parity error condition is detected, the MLI transmitter sets the parity error flag **MLI_TSTATR**.PE and also decreases the maximum parity error counter **MLI_TCR**.MPE by 1. The maximum parity error counter of the transmitter **MLI_TCR**.MPE determines the number of transmit parity error conditions that can be still detected until a transmitter parity error interrupt event is generated. If a transmitter parity error condition is detected and **MLI_TCR**.MPE is becoming 0 or while it is 0, a transmitter parity error interrupt event is generated by setting bit **MLI_TISR**.PEI and an interrupt is generated if enabled by **MLI_TIER**.PEIE = 1. After a transmitter parity error event occurred, **MLI_TCR**.MPE can be set again by software to a value greater 0001_B. Otherwise, each additional transmitter parity error condition will generate an MLI Request.

The transmitter parity error flag **MLI_TSTATR**.PE is cleared when a correct frame transmission and TREADY has been sampled with 1 within the ready delay time. It can be cleared by software by writing a 1 to bit **MLI_SCR**.CTPE. If for example, each transmitter parity error condition should generate a transmitter parity error event, **MLI_TCR**.MPE should be set to 0001_B. The software can check for accumulated parity error conditions by reading **MLI_TCR**.MPE or **MLI_TISR**.PEI, for the status of the latest received frame, it can check **MLI_TSTATR**.PE.

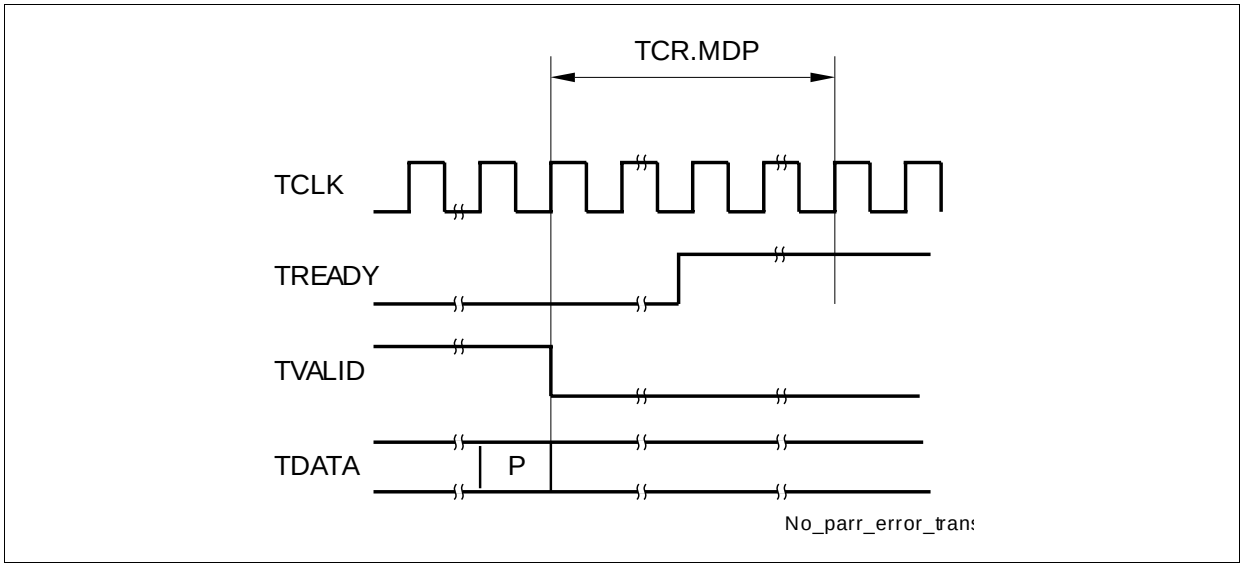


Figure 4-22 MLI Communication without Parity Error Indicator (Transmitter View)

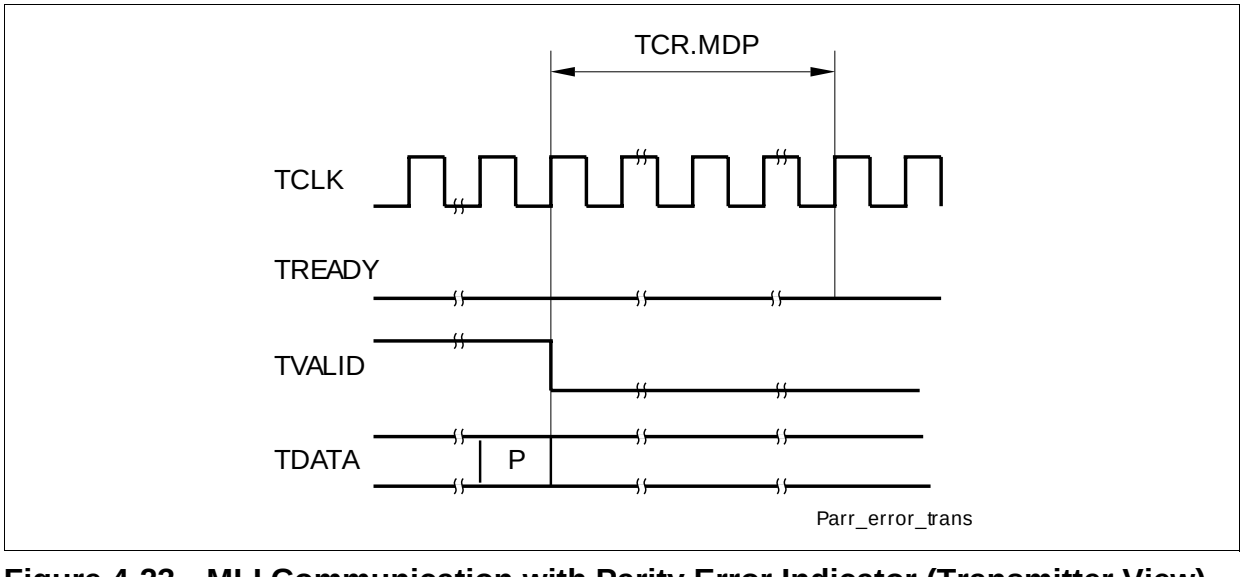


Figure 4-23 MLI Communication with Parity Error Indicator (Transmitter View)

Remote Controller

The receiver always checks the parity bit of a received frame for even parity. A receiver parity condition is detected if the received parity bit does not match with the internally calculated one. If no receiver parity error condition is found after the reception of a frame, RREADY is immediately set to 1; otherwise, RREADY is kept at 0 until a defined number of RCLK cycles (as determined by bit field **MLI_RCR**.DPE = delay for parity error) has been elapsed. Then, RREADY is asserted high.

If a receiver parity condition is found, the MLI receiver sets the parity error flag **MLI_RCR**.PE and additionally decreases the maximum parity error counter of the receiver **MLI_RCR**.MPE by 1. The maximum parity error counter **MLI_RCR**.MPE determines the number of receiver parity error conditions that can be still detected until the next receiver parity error event is generated. If a receiver parity error condition is detected and **MLI_RCR**.MPE is becoming 0 or while it is already 0, a receiver parity error interrupt event is generated by setting bit **MLI_RISR**.PEI and an interrupt is generated if enabled by **MLI_RIER**.PEIE = 1. After a receiver parity error event has occurred, **MLI_RCR**.MPE can set again by software to a value greater 0001_B. If, for example, each receiver parity error condition should generate a receiver parity error interrupt, **MLI_RCR**.MPE can be set to 0001_B after a receiver parity error interrupt has occurred.

The receiver parity error flag **MLI_RCR**.PE is cleared when a correct frame transmission has occurred. **MLI_RCR**.PE can be cleared by software when writing a 1 to bit **MLI_SCR**.CRPE.

The receiver parity error flag **MLI_RCR**.PE is cleared when a correct frame transmission and TREADY has been sampled with 1 within the ready delay time. It can be cleared by software by writing a 1 to bit **MLI_SCR**.CRPE. If for example, each receiver parity error condition should generate a receiver parity error event, **MLI_RCR**.MPE should be set to 0001_B. The software can check for accumulated parity error conditions by reading **MLI_RCR**.MPE or **MLI_RISR**.PEI, for the status of the latest received frame, it can check **MLI_RCR**.PE.

The delay for parity error bit field **MLI_RCR**.DPE is a read-only bit field in the receiver that can be written only by hardware if a Command Frame for Pipe 1 is received. With this frame type, the transmitter in the Local Controller transfers a value for **MLI_RCR**.DPE to the receiver in the Remote Controller.

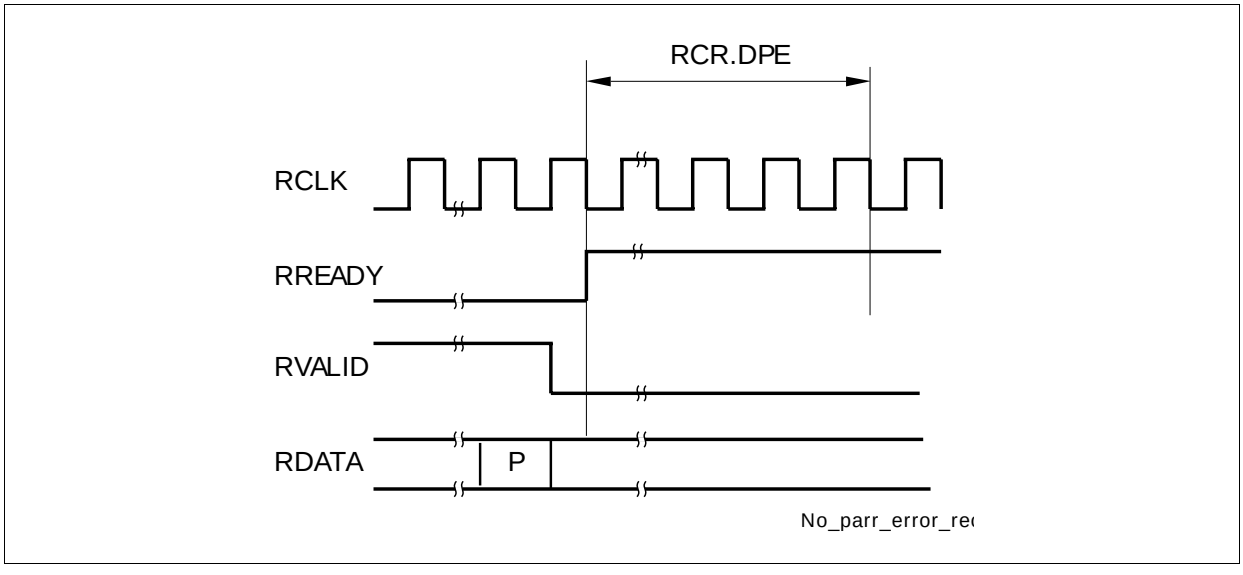


Figure 4-24 MLI Communication without Parity Error Indicator (Receiver View)

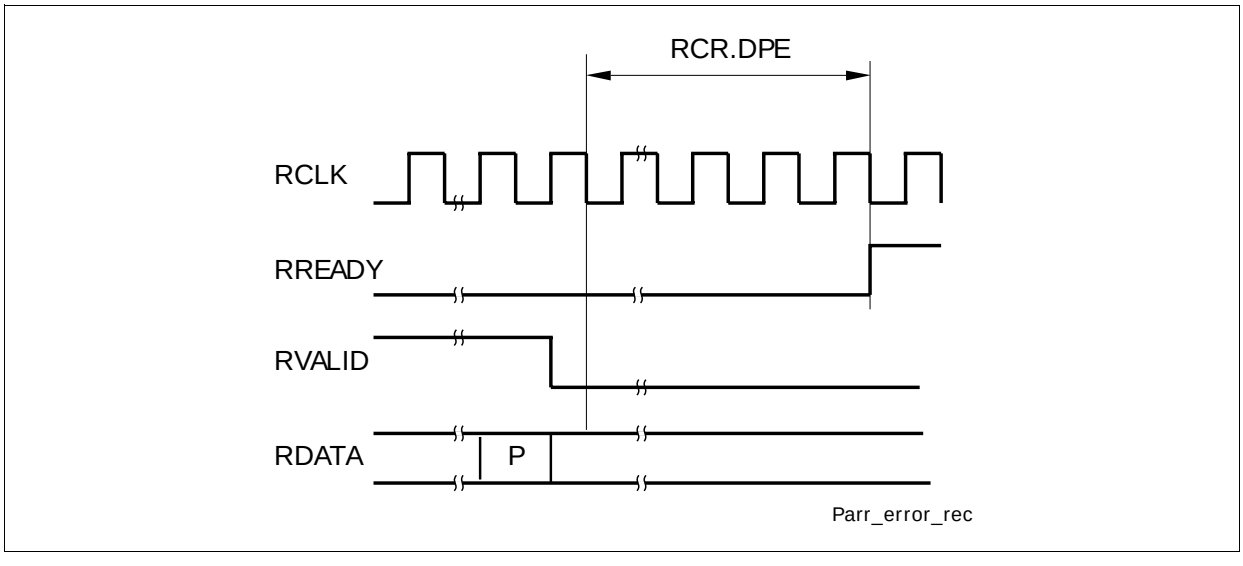


Figure 4-25 MLI Communication with Parity Error Indicator (Receiver View)

4.2.2.2 Non-Acknowledge Error

A non-acknowledge error condition is detected by the transmitter when, at the end of a frame transmission, the TREADY signal is still at high level (TREADY = 1 when TVALID becomes 0). In this case, the error flag **MLI_TSTATR**.NAE is set and the maximum non-acknowledge error counter **MLI_TCR**.MNAE is decremented by 1. If a non-acknowledge error condition is detected and **MLI_TCR**.MNAE is becoming 0 or while it is already 0, a time-out interrupt event is generated by setting bit **MLI_TISR**.TEI and an MLI Request is generated if enabled by **MLI_TIER**.TEIE = 1. The non-acknowledge error flag

MLI_TSTATR.NAE is cleared when a frame transmission has been acknowledged correctly. It can also be cleared by software when writing a 1 to bit **MLI_SCR**.CNAE.

The non-acknowledge error counter **MLI_TCR**.MNAE is automatically set to 11_B when a frame has been acknowledged correctly. It can be read and written by software, allowing a limited number of consecutive non-acknowledge errors to be defined that can be detected until a non-acknowledge error interrupt event is generated. If, for example, the first occurrence of a non-acknowledge error should lead to an non-acknowledge interrupt, bit **MLI_TCR**.MNAE has to be written by software with 01_B after each correctly received frame.

4.2.2.3 Address Prediction

Address prediction can be enabled to support communication between the MLI transmitter and MLI receiver without sending address offset information in the frames. This feature reduces the required bandwidth for MLI communication. Both of the communication partners, the MLI transmitter and the receiver, are able to detect regular offset differences of consecutive window accesses to the same window. The address prediction mechanism operates independently for each Pipe, so different prediction values can be handled in parallel for the different Pipes.

Local Controller

If the address prediction mechanism is enabled (**MLI_TCR**.NO = 0), the MLI transmitter compares the offset of each Transfer Window read or write access with the offset of the previous access to the same Transfer Window (stored in MLI_TPxAOFR.AOFF). The result of this comparison is stored in two's complement representation in MLI_TPxSTATR.AP (limited to 9 bits, otherwise prediction is not possible). Between the accesses to a specific window, other windows can be accessed without disturbing the prediction.

If the offset differences in at least two accesses are identical to the same Transfer Window, an address prediction is possible (flag MLI_TPxSTATR.OP is set) and optimized frames can be sent to the receiver in the Remote Controller for this Pipe. If the offset difference of the next access to the same Transfer Window does not match the calculated value in MLI_TPxSTATR.AP, flag MLI_TPxSTATR.OP is cleared and address prediction is not possible. In this case, a Normal Frame for writing or reading (Write Offset and Data Frame or Discrete Read Frame) is started.

Remote Controller

The MLI receiver operates with the same address prediction as the MLI transmitter. This means that after receiving at least two consecutive Write Offset and Data Frames and/ or Discrete Read Frames that include address information, the MLI receiver is able to follow the address prediction used by the MLI transmitter.
Each received offset is compared in the MLI receiver with the offset of the previously received frame of the same Pipe. The result of this comparison is stored in two's

complement representation in MLI_RPxSTATR.AP (limited 9 bits).
If an optimized frame is received by the MLI receiver, it calculates the next address by adding the value stored in MLI_RPxSTATR.AP to the contents of the receiver address register **MLI_RADRR**.
In case of a Write Offset and Data Frame or a Discrete Read Frame, the receiver address registers **MLI_RADRR** and MLI_RPxBAR are always updated with an address. This address is calculated by replacing the offset bit positions in MLI_RPxBAR with the received offset value. In this case, the address delta value stored in MLI_RPxSTATR.AP is not taken into account. The programmed size of the Remote Window and the number of offset bits are given by MLI_RPxSTATR.BS. The non-offset bit positions in register MLI_RPxBAR are kept constant, whereas the offset bit positions are replaced.

4.2.2.4 Automatic Data Mode

The MLI module supports automatic data transfer for read or write frames in the Remote Controller. The Automatic Data Mode in the Remote Controller can be enabled either via setting bit **MLI_RCR**.MOD or by a Command Frame sent by the Local Controller. The Automatic Data Mode in the Remote Controller can be disabled by clearing bit **MLI_RCR**.MOD.
If the Automatic Data Mode is disabled, the DMA has to execute the requested data transfers.

Note: For the CIC751 the Automatic Data Mode should also be used.

4.2.2.5 Transmit Priority

In the case that several requests for frame transmission are pending at the same time in the MLI transmitter of the Local Controller, the following priority scheme is applied, starting with the highest priority:

- Answer Frame
- Software driven Command Frames (CCV0 before CCV1 before CCV2 before CCV3)
- Read or Write Frames (DV0 before DV1 before DV2 before DV3)
- Base Address Copy Frame (BAV0 before BAV1 before BAV2 before BAV3)

4.2.3 MLI Interface Control

Each of the MLI transmitter and MLI receiver communicate with other MLI receivers and MLI transmitters via a four-line serial connection. Each input/output signal used for MLI communication between a transmitter and a receiver can be disabled and inverted in its polarity. The control is achieved via register **MLI_OICR**.

4.2.4 MLI Request Generation

MLI Request generation is based on the interrupt event cases that can be created by the different interrupt sources.
Each interrupt source is provided with a status flag and an enable bit with software clear capability. Several interrupt sources can be combined into one MLI Request using a common interrupt node pointer. An interrupt event, generated by an interrupt source, is always stored in an interrupt status flag that is located in the interrupt status registers **MLI_TISR** (for transmitter interrupts) or **MLI_RISR** (for receiver interrupts). All interrupt event flags can be cleared individually by write actions to bits located in the interrupt enable registers **MLI_TIER** (for transmitter interrupts) or **MLI_RIER** (for receiver interrupts). These two registers also contain the enable control bits that allow each interrupt source to be enabled/disabled individually. Some interrupt events are combined to one common interrupt. Each interrupt is connected to exactly one of the four MLI interrupt node pointer.

One additional register, the Global Interrupt Set Register **MLI_GINTR**, allows each MLI Request to be activated separately without setting the request flags of the interrupt sources. This feature is sometimes helpful for software test purposes.

Interrupt Registers

MLI interrupt sources are controlled by several registers (see **Table 4-7** and **Page 4-83**). The register name prefixes “T” and “R” indicate whether an interrupt register is assigned to the MLI transmitter or to the MLI receiver.

Table 4-7 Interrupt Registers

Unit	Registers with		
	Request Flags	Enable Bits/ Req. Flag Clear Bits	Node Pointer
MLI Transmitter	MLI_TISR	MLI_TIER	MLI_TINPR
MLI Receiver	MLI_RISR	MLI_RIER	MLI_RINPR

Interrupt Request Compressor

Interrupt control of the MLI uses an interrupt compressing scheme that allows great flexibility in interrupt processing. Eleven interrupts (six transmitter interrupts and four of the five receiver interrupts) are directed via a interrupt node pointer to one of the four MLI Request. One receiver interrupt, the interrupt Command Frame interrupt, has a special characteristic: its node pointer is controlled by the received CMD value directly.

4.2.5 Transmitter Interrupts

The MLI transmitter can generate the following interrupts:

Table 4-8 MLI Transmitter Interrupts

Interrupt Events	Interrupt	See
Parity Error	Parity/Time-out Error	Page 4-43
Time-out Error		
Normal Frame Sent in Pipe 0	Normal Frame Sent in Pipe 0	Page 4-43
Normal Frame Sent in Pipe 1	Normal Frame Sent in Pipe 1	
Normal Frame Sent in Pipe 2	Normal Frame Sent in Pipe 2	
Normal Frame Sent in Pipe 3	Normal Frame Sent in Pipe 3	
Command Frame Sent in Pipe 0	Command Frame Sent	Page 4-43
Command Frame Sent in Pipe 1		
Command Frame Sent in Pipe 2		
Command Frame Sent in Pipe 3		

4.2.5.1 Parity/Time-out Error Interrupt

A parity/time-out error interrupt is generated when a programmable maximum number of parity errors or a programmable maximum number of non-acknowledge errors has been reached. Both interrupt events have separate status/control bits but are concatenated to one common error interrupt.

4.2.5.2 Normal Frame Sent x Interrupt

A Normal Frame sent x (x = 0-3) interrupt is generated when a Normal Frame has been sent and correctly received in Pipe x.

4.2.5.3 Command Frame Sent Interrupt

A Command Frame sent interrupt is generated when the MLI transmitter has sent a Command Frame through Pipe x (x = 0-3) that has been correctly received. Separate status/control bits are assigned to each Pipe. All four Pipe related Command Frame sent interrupt events are concatenated to one common Command Frame sent interrupt.

4.2.6 Receiver Interrupts

The MLI receiver can generate the following interrupts:

Table 4-9 MLI Receiver Interrupts

Interrupt Events	Interrupt	See
Discarded Read Answer	Discarded Read Answer	Page 4-43
Parity Error	Parity Error	Page 4-44
Normal Frame Correctly Received	Normal Frame Received	Page 4-44
Move Engine Access Terminated		
Interrupt Command Frame	Interrupt Command Frame	Page 4-44
Command Frame Received on Pipe 0	Command Frame Received	Page 4-44
Command Frame Received on Pipe 1		
Command Frame Received on Pipe 2		
Command Frame Received on Pipe 3		

4.2.6.1 Discarded Read Answer Interrupt

A discarded read answer received interrupt is generated when an Answer Frame has been received and the read pending flag **MLI_TRSTATR**.RPx of its correspondent Pipe is 0. Although named “discarded”, the received data is available in the receiver data register until it is overwritten by the next incoming data.

4.2.6.2 Parity Error Interrupt

A parity error interrupt is generated when a programmable maximum number of receiver parity errors is reached.

4.2.6.3 Normal Frame Received/Move Engine Terminated Interrupt

A Normal Frame received interrupt is generated when the MLI receiver has correctly received a Normal Frame (a read or a write frame, not a Command Frame or Copy Base Address Frame) correctly or when the MLI has terminated its read or write access. Both interrupt sources have separate status/control bits but are concatenated to one common frame receive interrupt.

4.2.6.4 Interrupt Command Frame Interrupt

An interrupt command frame interrupt is generated when a Command Frame is received correctly on Pipe 0 with a valid command code for remote interrupt generation (CMD = 0000_B to 0011_B). The received command code determines which of the service request output lines SR[3:0] should be activated.

4.2.6.5 Command Frame Received Interrupt

A command frame received interrupt is generated when the MLI receiver has correctly received a Command Frame through Pipe Number x (x = 0-3). Separate interrupt status/ control bits are assigned to each Pipe. All four Pipe related Command Frame received in Pipe x interrupt events are concatenated to one common Command Frame received interrupt.

4.2.7 Baud Rate Generation

The MLI transmitter baud rate is given by $f_{MLI}/2$. The MLI shift clock output signal TCLK of the transmitter toggles with each clock cycle of f_{MLI} in order to obtain a 50% duty cycle (the 50% duty cycle can vary up to one clock cycle of f_{SYS} in Fractional Divider Mode). The MLI receiver automatically adapts to the incoming receive shift clock signal RCLK. The received baud rate is determined by the connected transmitter and has no direct relation to f_{SYS} except that it should not exceed f_{SYS}.
The frequency f_{MLI} is generated by the fractional divider FDIV, programmable by register **MLI_FDR**.

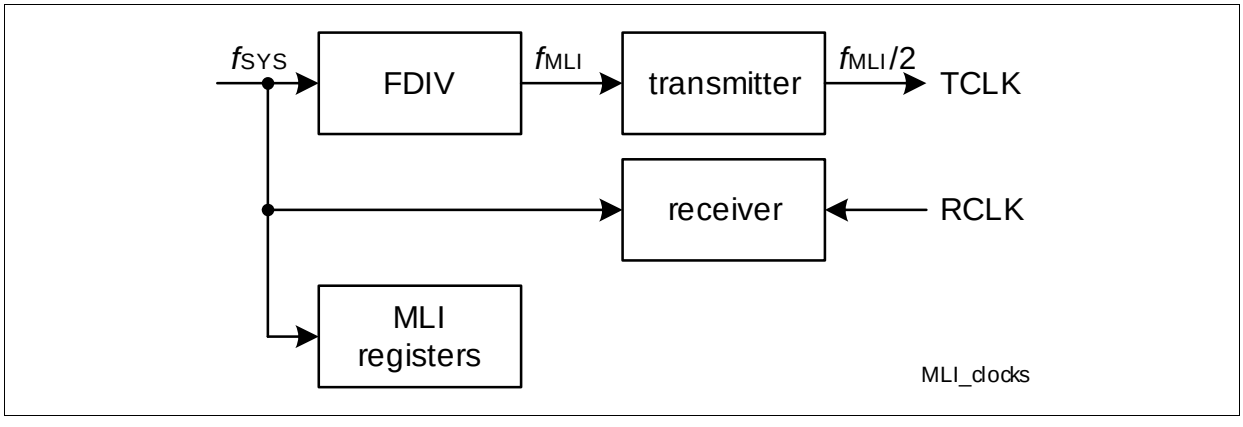


Figure 4-26 MLI Baud Rate Generation

Normal Divider Mode

In Normal Divider Mode (**MLI_FDR**.DM = 01_B) the fractional divider behaves like a reload counter (addition of +1) that generates a clock f_{MLI} on the transition from $3FF_H$ to 000_H. **MLI_FDR**.RESULT represents the counter value and **MLI_FDR**.STEP defines the reload value. In order to achieve $f_{MLI} = f_{SYS}$, **MLI_FDR**.STEP must be programmed with $3FF_H$. The output frequency in Normal Divider Mode is defined according the following equation:

$$f_{MLI} = f_{SYS} \times \frac{1}{1024 - STEP} \tag{4.1}$$

Fractional Divider Mode

If the Fractional Divider Mode is selected (**MLI_FDR**.DM = 10_B), the clock f_{MLI} is derived from the input clock f_{SYS} by division of a fraction of STEP/1024 for any value of STEP from 0 to 1023. In general, the Fractional Divider Mode allows to program the average clock frequency with a higher accuracy than in Normal Divider Mode. In Fractional Divider Mode, a clock pulse f_{MLI} is generated based on the result of the addition **MLI_FDR**.RESULT + **MLI_FDR**.STEP. The frequency f_{MLI} corresponds to the overflows

over $3FF_H$. Note that in Fractional Divider Mode, the clock f_{MLI} can have a maximum period jitter of one f_{SYS} clock period. This jitter is not accumulated over several cycles and does not exceed one cycle of f_{SYS}.
The frequency in Fractional Divider Mode is defined according to the following equation:

$$f_{MLI} = f_{SYS} \times \frac{STEP}{1024} \tag{4.2}$$

The baud rate of MLI transmissions equals f_{TCLK}, which is defined by the frequency of clock signal f_{MLI} divided by 2 to create the 50% duty cycle of the shift clock signal TCLK. The signal TCLK toggling with each period of f_{MLI}, a jitter due to fractional dividing is propagated to TCLK.

$$f_{TCLK} = \frac{f_{MLI}}{2} \tag{4.3}$$

4.3 MLI Kernel Registers

Table 4-10 lists all of the registers associated with the MLI.

All registers can be accessed with 8-bit, 16-bit or 32-bit write or read operations. Accesses to address locations inside the MLI address range not targeting the indicated registers are not allowed.

The base address of the MLI is 0000 0200. A register address is computed by adding the base address to the register offset address.

Table 4-10 MLI Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
MLI_FDR	Fractional Divider Register	$000C_H$	Page 4-49
MLI_TCR	Transmitter Control Register	0010_H	Page 4-59
MLI_TSTATR	Transmitter Status Register	0014_H	Page 4-62
MLI_TP0STATR	Transmitter Pipe x Status Register	0018_H + (x * 4)	Page 4-64
MLI_TCMDR	Transmitter Command Register	0028_H	Page 4-66
MLI_TRSTATR	Transmitter Receiver Status Register	$002C_H$	Page 4-68
MLI_TP0AOFR	Transmitter Pipe x Address Offset Register	0030_H + (x * 4)	Page 4-70
MLI_TP0DATAR	Transmitter Pipe x Data Register	0040_H + (x * 4)	Page 4-71
MLI_TDRAR	Transmitter Data Read Answer Register	0050_H	Page 4-72
MLI_TP0BAR	Transmitter Pipe x Base Address Register	0054_H + (x * 4)	Page 4-73
MLI_TCBAR	Transmitter Copy Base Address Register	0064_H	Page 4-74
MLI_RCR	Receiver Control Register	0068_H	Page 4-75
MLI_RP0BAR	Receiver Pipe x Base Address Register	$006C_H$ + (x * 4)	Page 4-78
MLI_RP0STATR	Receiver Pipe x Status Register	$007C_H$ + (x * 4)	Page 4-74
MLI_RADRR	Receiver Address Register	$008C_H$	Page 4-81
MLI_RDATAR	Receiver Data Register	0090_H	Page 4-82
MLI_SCR	Set Clear Register	0094_H	Page 4-51

Table 4-10 MLI Kernel Registers (cont'd)

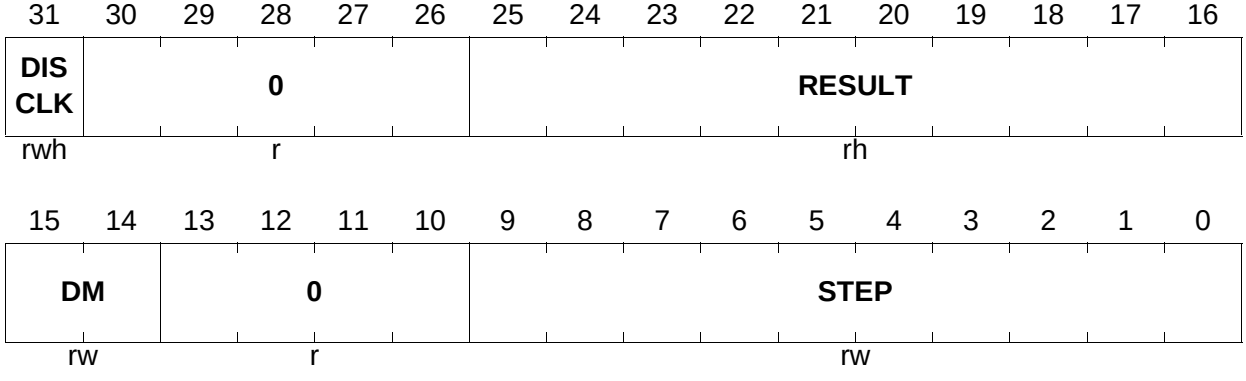
Register Short Name	Register Long Name	Offset Address	Description see
MLI_TIER	Transmitter Interrupt Enable Register	0098_H	Page 4-83
MLI_TISR	Transmitter Interrupt Status Register	$009C_H$	Page 4-85
MLI_TINPR	Transmitter Interrupt Node Pointer Register	$00A0_H$	Page 4-86
MLI_RIER	Receiver Interrupt Enable Register	$00A4_H$	Page 4-88
MLI_RISR	Receiver Interrupt Status Register	$00A8_H$	Page 4-90
MLI_RINPR	Receiver Interrupt Node Pointer Register	$00AC_H$	Page 4-92
MLI_GINTR	Global Interrupt Set Register	$00B0_H$	Page 4-53
MLI_OICR	Output Input Control Register	$00B4_H$	Page 4-54

4.3.1 General Registers

4.3.1.1 Fractional Divider Register

The fractional divider register allows programming of the frequency f_{MLI} to generate the baud rate of the of the 50% duty cycle transmitter shift clock TCLK.

MLI_FDR
MLI Fractional Divider Register (20C$_H$) Reset Value: 03FF 43FF$_H$



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** In Normal Divider Mode, STEP contains the reload value for RESULT. In Fractional Divider Mode, this bit field defines the 10-bit value that is added to the RESULT with each input clock cycle.
DM	[15:14]	rw	**Divider Mode** This bit field defines the functionality of the fractional divider block. 00 Fractional divider is switched off; no output clock is generated. RESULT is not updated. 01 Normal Divider Mode selected. 10 Fractional Divider Mode selected. 11 Fractional divider is switched off; no output clock is generated. RESULT is not updated.

Field	Bits	Type	Description
RESULT	[25:16]	rh	**Result Value** In Normal Divider Mode, RESULT acts as reload counter (addition +1). In Fractional Divider Mode, this bit field contains the result of the addition RESULT+STEP. If DM is written with 01_B or 10_B, RESULT is loaded with $3FF_H$.
DISCLK	31	rwh	**Disable Clock** 0 Clock generation of f_{MLI} is enabled according to the setting of bit field DM. 1 Fractional divider is stopped. Signal f_{MLI} becomes inactive. No change except when writing bit field DM.
0	[13:10] [30:26]	r	**Reserved** Read as 0; should be written with 0.

4.3.1.2 Set Clear Register

The Set Clear Register **MLI_SCR** is a write-only register that makes it possible to set or clear several status flags located in registers **MLI_TSTATR**, **MLI_TRSTATR** and **MLI_RCR** under software control. Reading register **MLI_SCR** always returns zero.

MLI_SCR
MLI Set Clear Register **(294_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				C NAE	C TPE	C RPE	C AV	0						C BAV	C MOD
w				w	w	w	w				w			w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C CV3	C CV2	C CV1	C CV0	C DV3	C DV2	C DV1	C DV0	0			S MOD	S CV3	S CV2	S CV1	S CV0
w	w	w	w	w	w	w	w		w		w	w	w	w	w

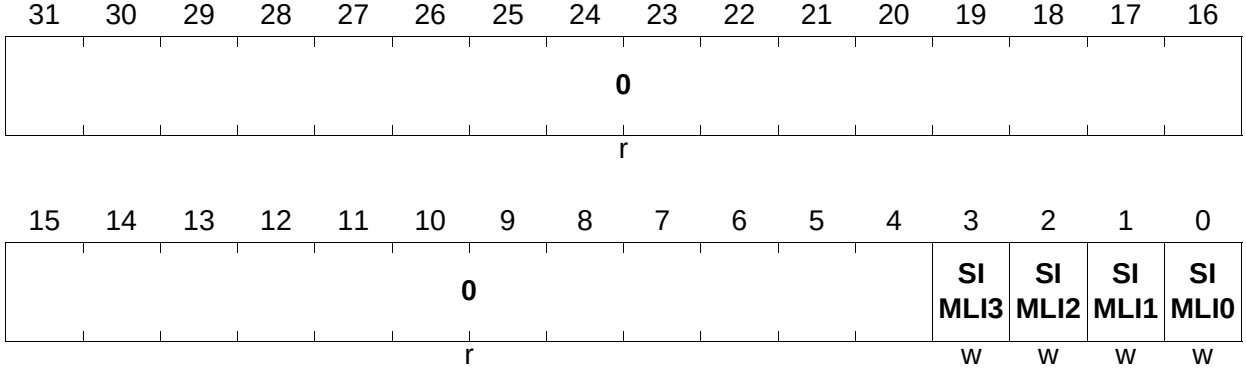
Field	Bits	Type	Description
SCV0, **SCV1,** **SCV2,** **SCV3**	0, 1, 2, 3	w	**Set Command Valid** 0 No effect 1 Bit **MLI_TRSTATR**.CVx is set
SMOD	4	w	**Set MOD Flag** 0 No effect 1 If CMOD = 0, **MLI_RCR** is set. If CMOD = 1, **MLI_RCR**.MOD is cleared.
CDV0, **CDV1,** **CDV2,** **CDV3**	8, 9, 10, 11	w	**Clear Data Valid x Flag** 0 No effect 1 If SCVx = 0, bits **MLI_TRSTATR**.DVx and **MLI_TRSTATR**.RPx are cleared. If SCVx = 1, bit **MLI_TRSTATR**.DVx is set.
CCV0, **CCV1,** **CCV2,** **CCV3**	12, 13, 14, 15	w	**Clear Command Valid x Flag** 0 No effect 1 Bit **MLI_TRSTATR**.CVx is cleared
CMOD	16	w	**Clear MOD Flag** 0 No effect 1 Bit **MLI_RCR**.MOD is cleared

Field	Bits	Type	Description
CBAV	17	w	**Clear BAV Flag** 0 No effect 1 Bit **MLI_TRSTATR**.BAV is cleared
CAV	24	w	**Clear AV Flag** 0 No effect 1 Bit **MLI_TRSTATR**.AV is cleared
CRPE	25	w	**Clear Receiver PE Flag** 0 No effect 1 Bit **MLI_RCR**.PE is cleared
CTPE	26	w	**Clear Transmitter PE Flag** 0 No effect 1 Bit **MLI_TSTATR**.PE is cleared
CNAE	27	w	**Clear NAE Flag** 0 No effect 1 Bit **MLI_TSTATR**.NAE is cleared
0	[7:5], [23:18], [31:28]	w	**Reserved** Read as 0; should be written with 0.

4.3.1.3 Global Interrupt Set Register

The Global Interrupt Set Register **MLI_GINTR** is a write-only register (always reads 0) that allows each of the MLI Requests to be activated under software control.

MLI_GINTR
MLI Global Interrupt Set Register (2B0$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SIMLIx (x = 0-3)	x	w	**Set MLI Service Request Output Line x** 0 No action 1 MLI Request x is activated
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.

4.3.1.4 Output Input Control Register

The Output Input Control Register **MLI_OICR** determines the functionality of the MLI transmitter and MLI receiver I/O control logic.

MLI_OICR
MLI Output Input Control Register (2B4$_H$) **Reset Value: 1000 8000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDP	RDS		RCE	RCP	RCS		RVP	RVS		0		RRPB	0	RRS	
rw	rw		rw	rw	rw		rw	rw		rw		rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RVE	TDP	TCP	TCE	TRE	TRP	TRS		0		TVPB	0			TVEB	0
rw	rw	rw	rw	rw	rw	rw		rw		rw		rw		rw	rw

Field	Bits	Type	Description
TVEB	1	rw	**Transmitter Valid Enable** This bit enables the MLI transmitter output signal TVALID. 0 TVALID is disabled and remains at passive level (as selected by TVPB) 1 Transmitter output signal TVALID is enabled and driven
TVPB	5	rw	**Transmitter Valid Polarity** This bit determines the polarity of the transmitter output signals TVALID. 0 Non-inverted polarity for TVALID selected: TVALID is passive when driving a 0. TVALID is active when driving a 1. 1 Inverted polarity for TVALID selected: TVALID is passive when driving a 1. TVALID is active when driving a 0.

Field	Bits	Type	Description
TRS	[9:8]	rw	**Transmitter Ready Selection** This bit field determines the input TREADY that is used as MLI transmitter input. 00_B TREADY is not connected to the MLI 01_B TREADY is selected 10_B TREADY is not connected to the MLI 11_B TREADY is not connected to the MLI
TRP	10	rw	**Transmitter Ready Polarity** This bit determines the polarity of TREADY. 0 Non-inverted polarity for TREADY selected: TREADY is passive if 0. TREADY is active if 1. 1 Inverted polarity for TREADY selected: TREADY is passive if 1. TREADY if 0.
TRE	11	rw	**Transmitter Ready Enable** This bit enables the MLI transmitter input signal TREADY. 0 TREADY signal is disabled (always at 0 level) 1 TREADY signal is enabled and driven by TREADY according to the settings of TRS and TRP
TCE	12	rw	**Transmitter Clock Enable** This bit enables the module output signal TCLK. 0 TCLK is disabled and remains at passive level (as selected by TCP) 1 TCLK is enabled and driven according to the setting of TCP
TCP	13	rw	**Transmitter Clock Polarity** This bit determines the polarity of the module output clock signal TCLK. 0 Non-inverted polarity for TCLK selected: TCLK is driving a 0 when it is passive. 1 Inverted polarity for TCLK selected: TCLK is driving a 1 when it is passive.

Field	Bits	Type	Description
TDP	14	rw	**Transmitter Data Polarity** This bit determines the polarity of the module output clock signal TDATA. 0 TDATA is directly driven by MLI transmitter output signal TDATA (non-inverted) 1 TDATA is directly driven by the inverted MLI transmitter output signal TDATA
RVE	15	rw	**Receiver Valid Enable** This bit enables the MLI receiver input signal RVALID. 0 RVALID signal is disabled (always at 0 level) 1 RVALID signal is enabled and driven by RVALID according to the settings of RVS and RVP
RRS	[17:16]	rw	**Receiver Ready Selector** This bit field determines whether RREADY is driven by the MLI receiver or is tied to passive level according to the setting of RRP. 00 RREADY is at passive level 01 RREADY is selected 10 RREADY is at passive level 11 RREADY is at passive level
RRPB	19	rw	**Receiver Ready Polarity** This bit determines the polarity of the receiver output RREADY. 0 Non-inverted polarity for RREADY selected: RREADY is passive if 0. RREADY is active if 1. 1 Inverted polarity for RREADYx selected: RREADY is passive if 1. RREADY is active if 0.
RVS	[23:22]	rw	**Receiver Valid Selector** This bit field determines whether the MLI is connected to pin RVALID or not. 00_B RREADY is not connected to the MLI 01_B RREADY is connected to the MLI 10_B RREADY is not connected to the MLI 11_B RREADY is not connected to the MLI

Field	Bits	Type	Description
RVP	24	rw	**Receiver Valid Polarity** This bit determines the polarity of RVALID. 0 Non-inverted polarity for RVALID selected: RVALID is passive if 0. RVALID is active if 1. 1 Inverted polarity for RVALID selected: RVALID is passive if 1. RVALID is active if 0.
RCS	[26:25]	rw	**Receiver Clock Selector** This bit field determines whether the MLI is connected to pin RCLK or not. 00_B RCLK is not connected to the MLI 01_B RCLK is connected to the MLI 10_B RCLK is not connected to the MLI 11_B RCLK is not connected to the MLI
RCP	27	rw	**Receiver Clock Polarity** This bit determines the polarity of RCLK. 0 Non-inverted polarity for RCLK selected: RCLK is at 0 level in passive state. 1 Inverted polarity for TCLK selected: RCLK is at 1 level in passive state.
RCE	28	rw	**Receiver Clock Enable** This bit enables the MLI receiver input clock RCLK. 0 RCLK signal is disabled (always at 0 level). 1 RCLK signal is enabled and driven by RCLK according to the settings of RCS and RCP.
RDS	[30:29]	rw	**Receiver Data Selector** This bit field determines whether the MLI is connected to pin RDATA or not. 00_B RDATA is not connected to the MLI 01_B RDATA is connected to the MLI 10_B RDATA is not connected to the MLI 11_B RDATA is not connected to the MLI
RDP	31	rw	**Receiver Data Polarity** This bit determines the polarity of RDATA. 0 Non-inverted polarity for RDATA selected: RDATA is passive if 0. RDATA is active if 1. 1 Inverted polarity for RDATA selected: RDATA is passive if 1. RDATA is active if 0.

Field	Bits	Type	Description
0	[4:2], [7:6], 18, [21:20]	rw	**Reserved** Should be written with 0.

4.3.2 MLI Transmitter Registers

4.3.2.1 Transmitter Control Register

The Transmitter Control Register **MLI_TCR** includes transmitter related control bits and bit fields that are used for parity/acknowledge, address optimization, TDATA idle polarity, retry, and transmitter enable/disable control.

MLI_TCR
MLI Transmitter Control Register (210$_H$) Reset Value: 0000 0110$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP	NO	MDP				MNAE		MPE				0	RTY	DNT	MOD
rw	rw	rw				rwh		rwh				r	rw	rw	rw

Field	Bits	Type	Description
MOD	0	rw	**Mode of Operation** This bit enables the MLI transmitter. 0 The MLI transmitter is disabled 1 The MLI transmitter is enabled
DNT	1	rw	**Data in Not Transmission** This bit determines the level of the transmitter data line TDATA when no transmission is in progress. 0 TDATA is at low level if no transmission is running 1 TDATA is at high level if no transmission is running

Field	Bits	Type	Description
RTY	2	rw	**Retry** This bit enables the retry mechanism for the Transfer Windows. This bit is only relevant for system bus architectures supporting a retry mechanism, otherwise it is ignored. 0 The retry mechanism is disabled. Any access while the transmitter is busy is discarded without additional action. 1 The retry mechanism is enabled. Any access while the transmitter is busy is acknowledged with a retry. In this case, the requesting bus master sends the requested access again until the request is accepted.
MPE	[7:4]	rwh	**Maximum Parity Errors** This bit field determines the maximum number of transmitter parity error conditions that can be still detected until a transmitter parity error event is generated (see **Page 4-34**). With each condition detected, MPE is decremented down to 0. 0000 A parity error interrupt event can be generated if a transmitter parity error condition is detected. 0001 A parity error interrupt event can be generated if a transmitter parity error condition is detected. 0010 A parity error interrupt event can be generated if 2 transmitter parity error conditions are detected. 0011 A parity error interrupt event can be generated if 3 transmitter parity error conditions are detected. ... ... 1111 A parity error interrupt event can be generated if 15 transmitter parity error conditions are detected.

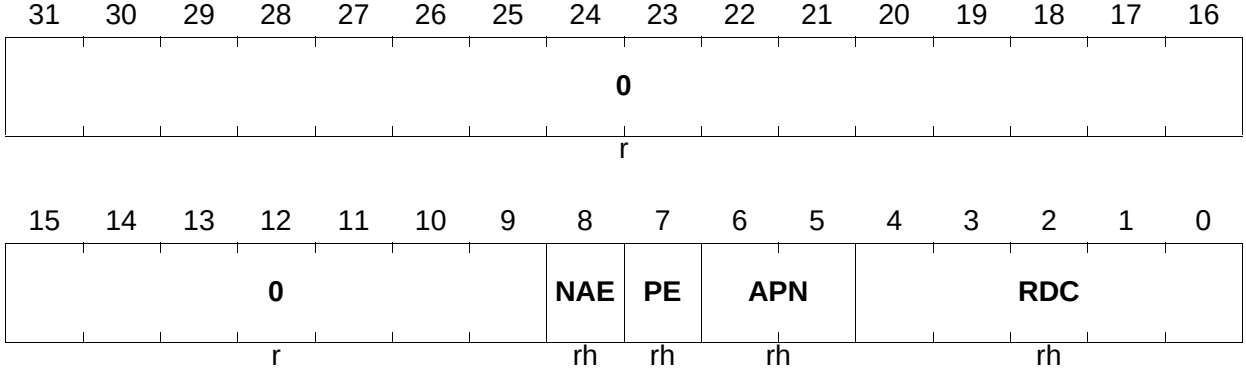
Field	Bits	Type	Description
MNAE	[9:8]	rwh	**Maximum Non Acknowledge Errors** This bit field determines the maximum number of consecutive non-acknowledge error conditions that can be still detected in the transmitter until a time-out interrupt is generated. MNAE is decremented down to 0 at each non-acknowledge error condition. When MNAE = 0 or becoming 0, a time-out interrupt event is generated. MNAE is automatically set to 11_B after a successful frame transmission (see **Page 4-37**). 00 A time-out interrupt can be generated if a non-ack condition is detected. 01 A time-out interrupt can be generated if a non-ack condition is detected. 10 A time-out interrupt can be generated if 2 consecutive non-ack conditions are detected. 11 A time-out interrupt can be generated if 3 consecutive non-ack conditions are detected.
MDP	[13:10]	rw	**Maximum Delay for Parity Error** This bit field determines a window for the transmitter in number of TCLK clock periods in which a TREADY low-to-high signal transition signal is considered as “correctly received” condition. 0000 Zero clock periods selected (not useful) 0001 1 clock period selected ... 1110 14 clock periods selected 1111 15 clock periods selected
NO	14	rw	**No Address Prediction** This bit field enables/disables the address prediction for read or write frames (see **Page 4-38**). 0 Address prediction is enabled 1 Address prediction is disabled
TP	15	rw	**Type of Parity** This bit determines the type of parity used in frame transmissions. For correct data transfers, TP = 0 must be programmed. The value TP = 1 can be selected to force parity errors to analyze the propagation delay (see **Page 4-17**). 0 Even parity is selected 1 Odd parity selected

Field	Bits	Type	Description
0	3, [31:16]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.2 Transmitter Status Register

The Transmitter Status Register **MLI_TSTATR** contains transmitter specific status information.

MLI_TSTATR
MLI Transmitter Status Register (214$_H$) Reset Value: 0000 0000$_H$



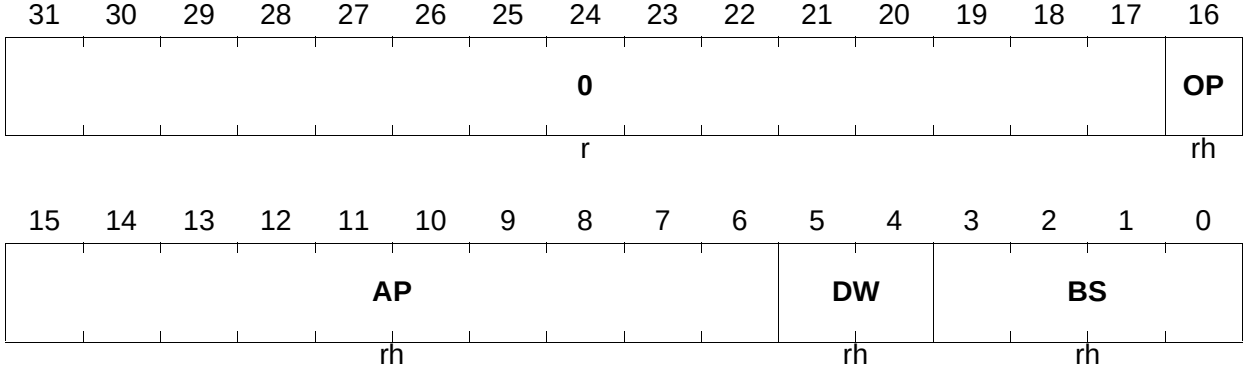
Field	Bits	Type	Description
RDC	[4:0]	rh	**Ready Delay Counter** This bit field counts TCLK periods after the end of a frame transmission. When the TVALID signal goes to low level, RDC is cleared to zero and starts counting up the TCLK clock periods until a TREADY high level is detected.
APN	[6:5]	rh	**Answer Pipe Number** This bit field is written by the MLI receiver with the Pipe Number of a received read frame. APN is used by an Answer Frame that is transmitted as response to the read frame. 00 Pipe 0 is used in Answer Frame 01 Pipe 1 is used in Answer Frame 10 Pipe 2 is used in Answer Frame 11 Pipe 3 is used in Answer Frame

Field	Bits	Type	Description
PE	7	rh	**Parity Error Flag** This bit is set if a transmitter parity error condition is detected by the transmitter after a frame transmission. PE is cleared by hardware when a frame has been transmitted without a parity error (see **Page 4-34**). Bit PE can be cleared via bit **MLI_SCR**.CTPE.
NAE	8	rh	**Non Acknowledge Error Flag** This bit is set when a non-acknowledge error condition is detected by the MLI transmitter after a frame transmission (see **Page 4-37**). NAE is cleared by hardware if a transmitted frame has been acknowledged correctly. Bit NAE can be cleared via bit **MLI_SCR**.CNAE.
0	[31:9]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.3 Transmitter Pipe x Status Registers

The Transmitter Pipe x Status Registers MLI_TPxSTATR contain Pipe-specific status information related to address optimization and prediction, data width for transmit data, and Remote Window size.

MLI_TP0STATR
MLI Transmitter Pipe 0 Status Register (218_H) **Reset Value: $0000\ 0000_H$**
MLI_TP1STATR
MLI Transmitter Pipe 1 Status Register ($21C_H$) **Reset Value: $0000\ 0000_H$**
MLI_TP2STATR
MLI Transmitter Pipe 2 Status Register (220_H) **Reset Value: $0000\ 0000_H$**
MLI_TP3STATR
MLI Transmitter Pipe 3 Status Register (224_H) **Reset Value: $0000\ 0000_H$**



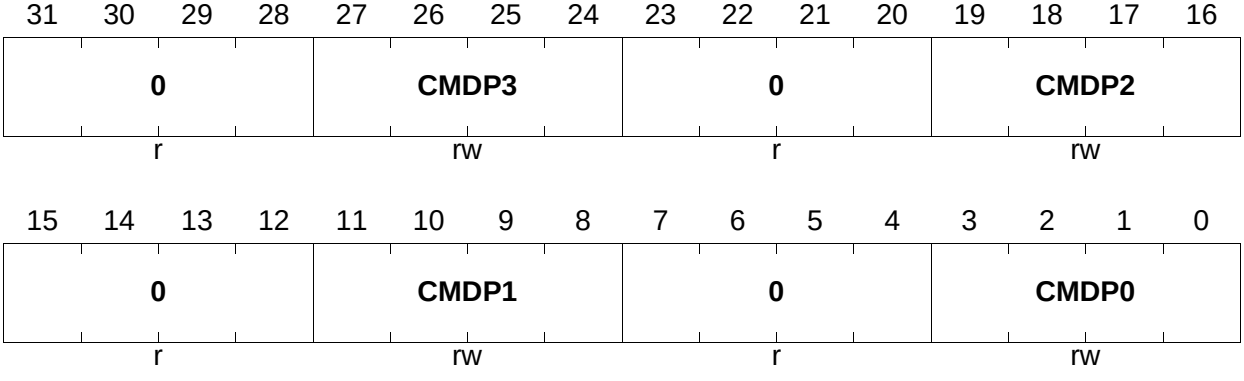
Field	Bits	Type	Description
BS	[3:0]	rh	**Size** This bit field indicates the coded size of the Pipe x Remote Window in the Remote Controller. BS further determines how many address offset bits are transmitted in a Write Offset and Data Frame or in a Discrete Read Frame. When register MLI_TPxBAR is written for generation of a Copy Base Address Frame, BS is updated by the Copy Base Address Frame (see Page 4-19). 0000 1-bit offset address of Remote Window 0001 2-bit offset address of Remote Window 0010 3-bit offset address of Remote Window ... ... 1111 16-bit offset address of Remote Window

Field	Bits	Type	Description
DW	[5:4]	rh	**Data Width** This bit field indicates the data width that has been detected for a read or write access to a Transfer Window of Pipe x (see **Page 4-22** and **Page 4-26**). 00 8-bit data width detected 01 16-bit data width detected 10 32-bit data width detected 11 Reserved
AP	[15:6]	rh	**Address Prediction Factor** This bit field indicates the delta value (positive or negative number) of offset address used by the MLI transmitter for the next address prediction. AP is a signed 9-bit number (10th bit is the sign bit) that is written with each transmitter address prediction calculation (see **Page 4-17** and **Page 4-38**).
OP	16	rh	**Use Optimized Frame** When address optimization is enabled with **MLI_TCR**.NO = 0, this bit indicates if address prediction is possible in the transmitter. OP is written with each transmitter address prediction calculation (see **Page 4-17** and **Page 4-38**). 0 No address prediction is possible. A Write Offset and Data Frame or a Discrete Read Frame are used for transmission. 1 Address prediction is possible. An Optimized Write Frame or an Optimized Read Frame are used for transmission.
0	[31:17]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.4 Transmitter Command Register

The Transmitter Command Register **MLI_TCMDR** contains the command codes that are used during Command Frame transmission (see **Page 4-30**). Each time one of the MLI_TCMDR.CMDPx bit fields is written, a Command Frame transmission is triggered. Therefore, only byte write accesses may be used when writing to **MLI_TCMDR** (only one Command Frame can be sent at a time).

MLI_TCMDR
MLI Transmitter Command Register (228_H) **Reset Value: $0000\ 0000_H$**



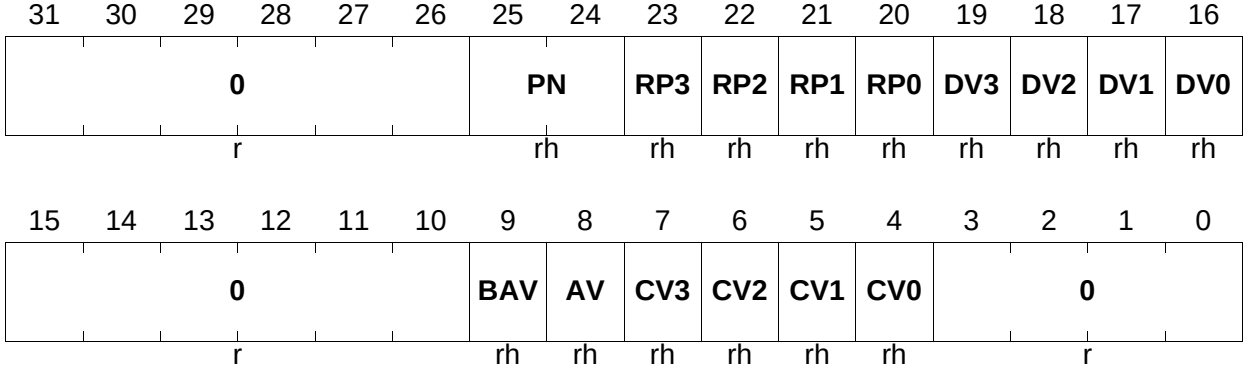
Field	Bits	Type	Description
CMDP0	[3:0]	rw	**Command Code for Pipe 0** This bit field contains the command code related to Pipe 0. The Pipe 0 command codes allow an activation (pulse) of one of the MLI Requests in the Remote Controller. 0001 Activate MLI Request 0 0010 Activate MLI Request 1 0011 Activate MLI Request 2 0100 Activate MLI Request 3 else no action

Field	Bits	Type	Description
CMDP1	[11:8]	rw	**Command Code for Pipe 1** This bit field contains the command code related to Pipe 1. The Pipe 1 command codes allow to adjust the receiver delay for the parity error condition (see **MLI_RCR**.DPE) in the MLI receiver of the Remote Controller. 0000 Set **MLI_RCR**.DPE = 0000_B 0001 Set **MLI_RCR**.DPE = 0001_B ... ... 1111 Set **MLI_RCR**.DPE = 1111_B
CMDP2	[19:16]	rw	**Command Code for Pipe 2** This bit field contains the command code related to Pipe 2. The Pipe 2 command codes allow to control the MLI receiver in the Remote Controller. 0001 Select Automatic Data Mode (set **MLI_RCR**.MOD = 1) 0010 Automatic Data Mode is disabled (set **MLI_RCR**.MOD = 0) 0100 Clear bit **MLI_TRSTATR**.RP0 0101 Clear bit **MLI_TRSTATR**.RP1 0110 Clear bit **MLI_TRSTATR**.RP2 0111 Clear bit **MLI_TRSTATR**.RP3 1111 Activate MLI break event else No action
CMDP3	[27:24]	rw	**Command Code for Pipe 3** This bit field contains the command code related to Pipe 3. The command codes for Pipe 3 are free programmable.
0	[7:4], [15:12] [23:20] [31:28]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.5 Transmitter-Receiver Status Register

The Transmitter-Receiver Status Register **MLI_TRSTATR** contains read-only flags that indicate the status of MLI operations.

MLI_TRSTATR
MLI Transmitter-Receiver Status Register ($22C_H$) **Reset Value: $0000\ 0000_H$**



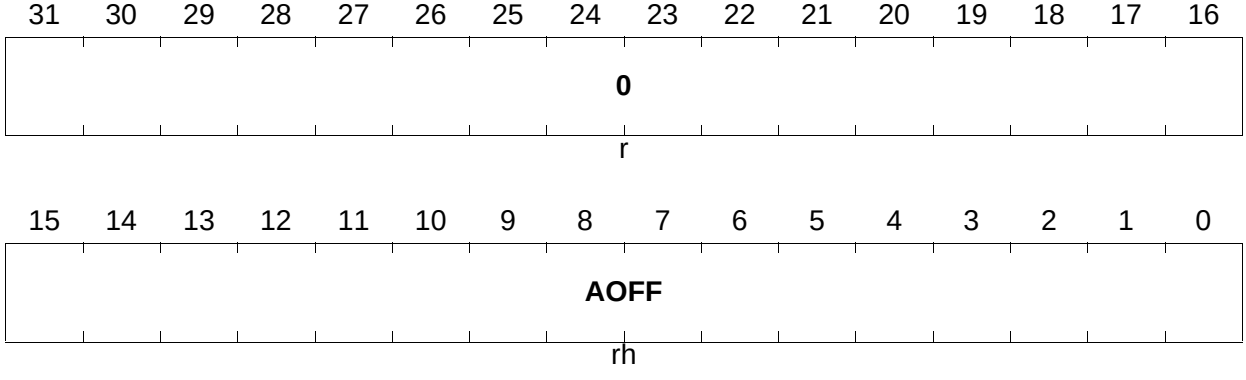
Field	Bits	Type	Description
CV0, **CV1,** **CV2,** **CV3**	4, 5, 6, 7	rh	**Command Valid** Bit is set when a **MLI_TCMDR**.CMDPx bit field is written. It is cleared when the Command Frame has been correctly transmitted. CVx can be set or cleared via bits **MLI_SCR**.SCVx or **MLI_SCR**.CCVx.
AV	8	rh	**Answer Valid** Bit is set when the **MLI_TDRAR** register in the MLI transmitter (in the Remote Controller) is written. AV is cleared when the Answer Frame has been correctly sent. AV can be cleared via bit **MLI_SCR**.CAV.
BAV	9	rh	**Base Address Valid** Bit is set when the **MLI_TCBAR** register in the MLI transmitter is written. BAV is cleared when the Copy Base Address Frame has been correctly sent. BAV can be cleared via bit **MLI_SCR**.CBAV.
DV0, **DV1,** **DV2,** **DV3**	16, 17, 18, 19	rh	**Data Valid** Bit is set when the **MLI_TP0DATAR** and/or the **MLI_TP0AOFR** registers of the MLI transmitter are updated after an access to a Transfer Window of Pipe x. DVx is cleared again when the read or write frame has been correctly sent. DVx can be cleared via bit **MLI_SCR**.CDVx.

Field	Bits	Type	Description
RP0, **RP1,** **RP2,** **RP3**	20, 21, 22, 23	rh	**Read Pending** Bit is set when the **MLI_TP0AOFR** register of the MLI transmitter is updated after a read access to a Transfer Window of Pipe x. RPx is cleared when the MLI receiver in the Local Controller receives an Answer Frame for Pipe x from the Remote Controller. RPx can be cleared via bit **MLI_SCR**.CDVx.
PN	[25:24]	rh	**Pipe Number** This bit field indicates the Pipe Number x of the base address that has been written into register MLI_TPxBAR. 00 Register MLI_TP0BAR has been written last 01 Register MLI_TP1BAR has been written last 01 Register MLI_TP2BAR has been written last 11 Register MLI_TP3BAR has been written last
0	[3:0], [15:10], [31:26]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.6 Transmitter Pipe x Address Offset Register

The Transmitter Pipe x Address Offset Register MLI_TPxAOFR (x = 0-3) is a read-only register that stores the offset address that has been used by the last read or write access to a Transfer Window of Pipe x.

MLI_TP0AOFR
MLI Transmitter Pipe 0 Address Offset Reg. (230_H) **Reset Value: $0000\ 0000_H$**
MLI_TP1AOFR
MLI Transmitter Pipe 1 Address Offset Reg. (234_H) **Reset Value: $0000\ 0000_H$**
MLI_TP2AOFR
MLI Transmitter Pipe 2 Address Offset Reg. (238_H) **Reset Value: $0000\ 0000_H$**
MLI_TP3AOFR
MLI Transmitter Pipe 3 Address Offset Reg. ($23C_H$) **Reset Value: $0000\ 0000_H$**

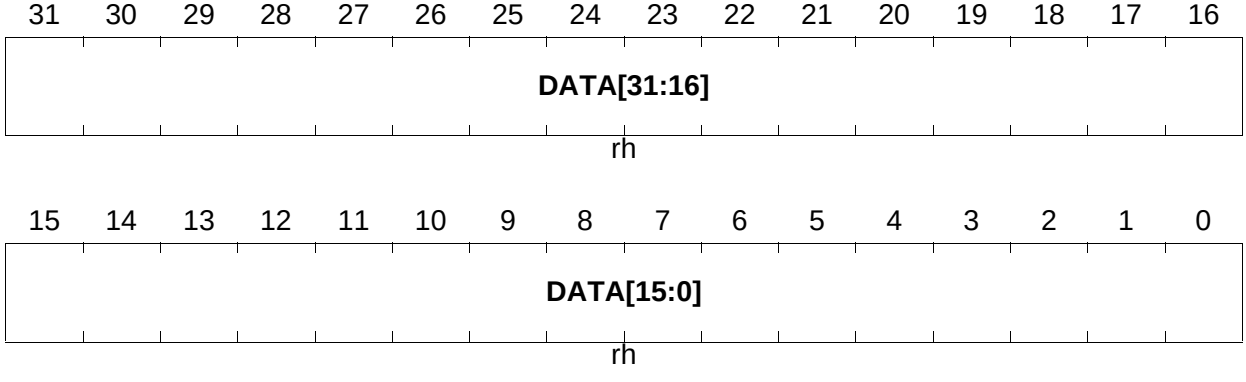


Field	Bits	Type	Description
AOFF	[15:0]	rh	**Address Offset** Whenever a location within a Transfer Window is accessed (read or written) AOFF is loaded with the lowest 16 address bits of the access. Also in the case of a small Transfer Window access, all AOFF bits are loaded, but AOFF[15:13] are not taken into account for further actions.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

4.3.2.7 Transmitter Pipe x Data Register

The Transmitter Pipe x Data Register MLI_TPxDATAR (x = 0-3) is a read-only register that stores the data that has been written during the last write access to a Transfer Window of Pipe x.

MLI_TP0DATAR
MLI Transmitter Pipe 0 Data Register (240_H) **Reset Value: 0000 0000_H**
MLI_TP1DATAR
MLI Transmitter Pipe 1 Data Register (244_H) **Reset Value: 0000 0000_H**
MLI_TP2DATAR
MLI Transmitter Pipe 2 Data Register (248_H) **Reset Value: 0000 0000_H**
MLI_TP3DATAR
MLI Transmitter Pipe 3 Data Register ($24C_H$) **Reset Value: 0000 0000_H**



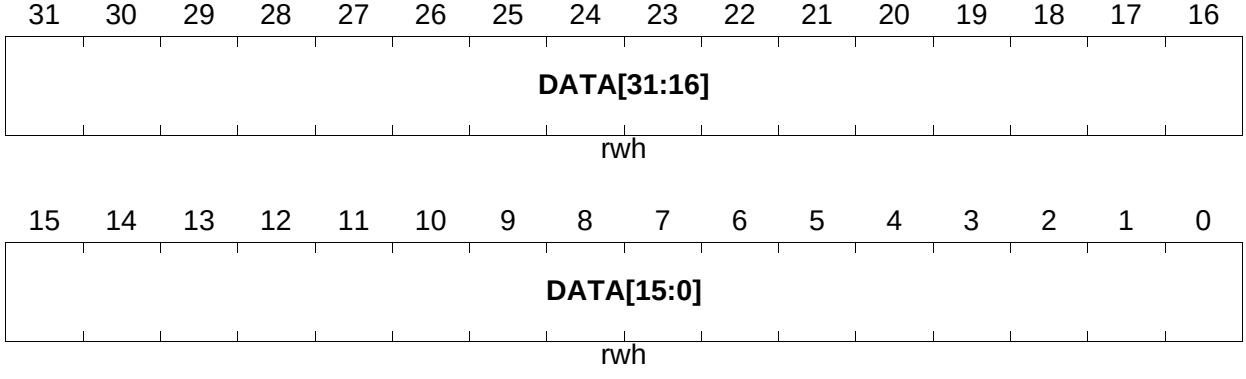
Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** Whenever a location within a Transfer Window is written, the data is loaded in this bit field.

4.3.2.8 Transmitter Data Read Answer Register

The Transmitter Data Read Answer Register **MLI_TDRAR** contains the read data for the transmission of an Answer Frame.

MLI_TDRAR

MLI Transmitter Data Read Answer Register (250_H) **Reset Value: $0000\ 0000_H$**

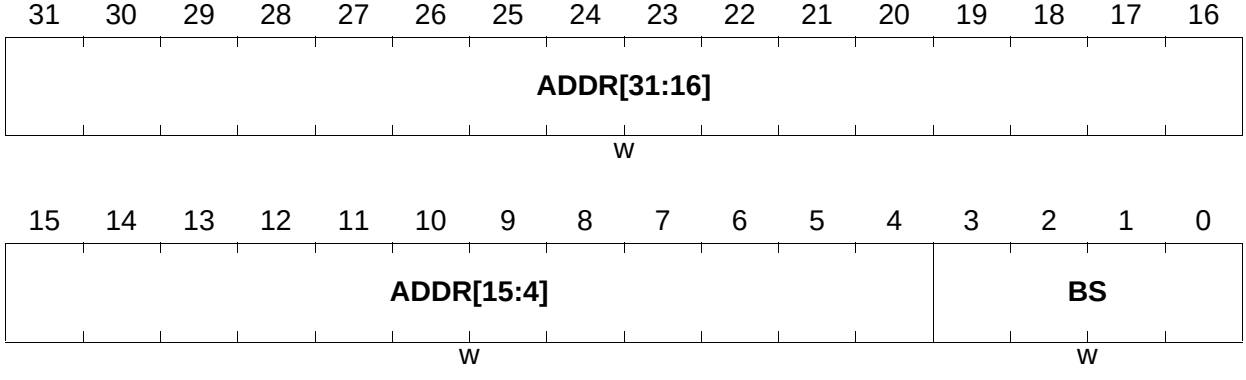


Field	Bits	Type	Description
DATA	[31:0]	rwh	**Data** This bit field is loaded with data that is read from the address requested by a read frame. An update of this bit field triggers the start of an Answer Frame with DATA used as content of the Answer Frame.

4.3.2.9 Transmitter Pipe x Base Address Register

The write-only Transmitter Pipe x Base Address Register MLI_TPxBAR (x = 0-3) represents the 28-bit Pipe x Remote Window base address that is transmitted to the Remote Controller via a Copy Base Address Frame.

MLI_TP0BAR
MLI Transmitter Pipe 0 Base Address Register (254_H) **Reset Value: 0000 0000_H**
MLI_TP1BAR
MLI Transmitter Pipe x Base Address Register (258_H) **Reset Value: 0000 0000_H**
MLI_TP2BAR
MLI Transmitter Pipe x Base Address Register ($25C_H$) **Reset Value: 0000 0000_H**
MLI_TP3BAR
MLI Transmitter Pipe x Base Address Register (260_H) **Reset Value: 0000 0000_H**



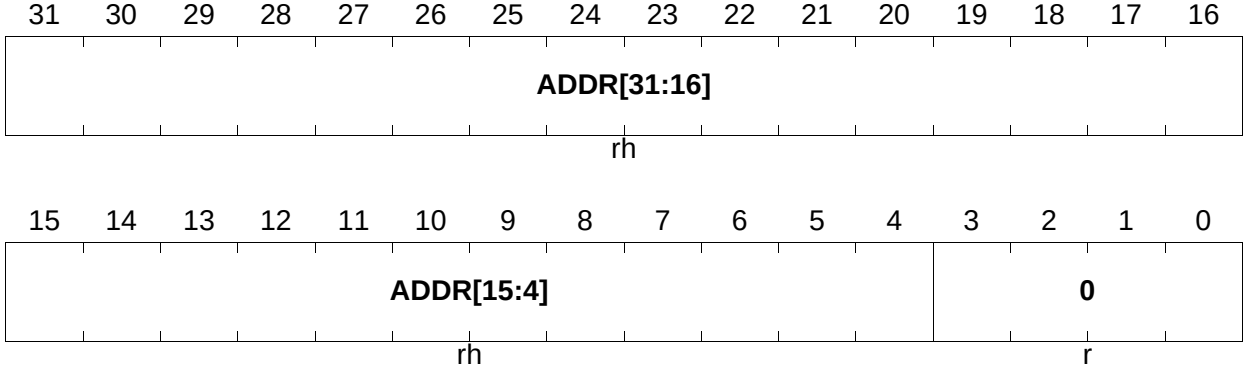
Field	Bits	Type	Description
BS	[3:0]	w	**Size** This bit field determines the coded size of the Pipe x Remote Window in the Remote Controller. When writing **MLI_TP0BAR**, BS is copied into bit field **MLI_TP0STATR**.BS. 0000 1-bit offset address of Remote Window 0001 2-bit offset address of Remote Window 0010 3-bit offset address of Remote Window ... 1111 16-bit offset address of Remote Window
ADDR	[31:4]	w	**Address** This bit field determines the most significant 28 bits of the Pipe x Remote Window base address. When writing **MLI_TP0BAR**, ADDR is copied into bit field **MLI_TCBAR**.ADDR[31:4].

4.3.2.10 Transmitter Copy Base Address Register

The Transmitter Copy Base Address Register **MLI_TCBAR** contains the 28-bit Pipe x Remote Window base address of the latest write access to **MLI_TP0BAR**.ADDR.

MLI_TCBAR

MLI Transmitter Copy Base Address Register (64_H) **Reset Value: 0000 0000$_H$**



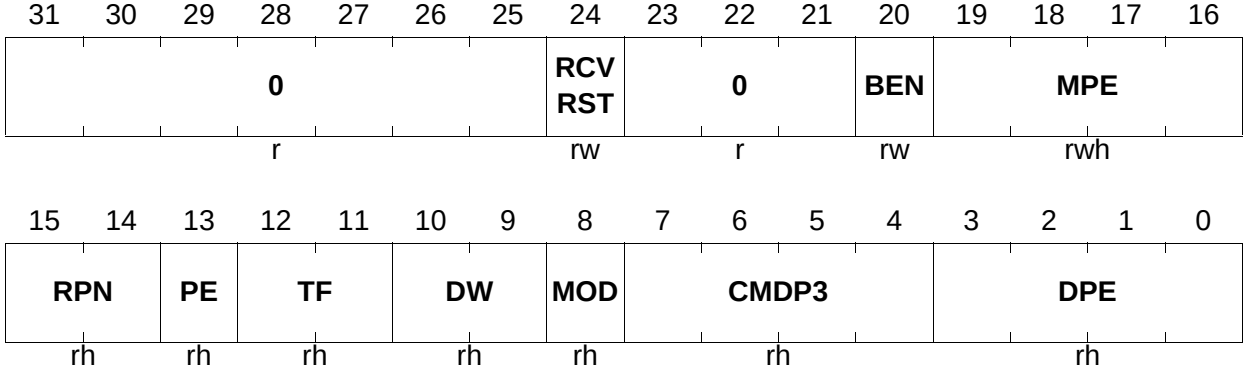
Field	Bits	Type	Description
ADDR	[31:4]	rh	**Address** This bit field contains the 28 address bits written to **MLI_TP0BAR**.ADDR. This value will be transferred to the Remote Controller to define the base address of the Remote Window for Pipe x.
0	[3:0]	r	**Reserved** Read as 0; should be written with 0.

4.3.3 MLI Receiver Registers

4.3.3.1 Receiver Control Register

The Receiver Control Register **MLI_RCR** contains control and status bits/bit fields that are related to the MLI receiver operation.

MLI_RCR
MLI Receiver Control Register (268$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DPE	[3:0]	rh	**Delay for Parity Error** DPE determines the number of RCLK clock periods that the MLI receiver waits before the RREADY signal is raised again when it has detected a parity error. When a Pipe 1 Command Frame is received by the MLI receiver, the command code is stored in this bit field (see **Page 4-30**). 0000 Zero RCLK clock period delay is selected 0001 One RCLK clock period delay is selected 0010 Two RCLK clock periods delay is selected ... ... 1110 Fourteen RCLK clock periods delay is selected 1111 Fifteen RCLK clock periods delay is selected
CMDP3	[7:4]	rh	**Command From Pipe 3** When a Pipe 3 Command Frame is received by the MLI receiver, the command code is stored in this bit field. Pipe 3 commands are available for software use.

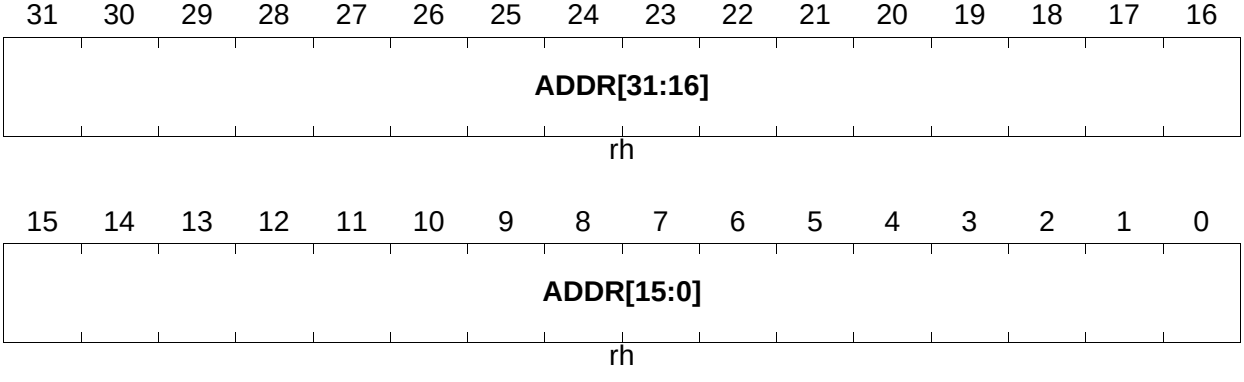
Field	Bits	Type	Description
MOD	8	rh	**Mode of Operation** This bit determines the data transfer operation mode of the MLI receiver. Bit MOD can be set with the reception of a Pipe 2 Command Frame (see **Page 4-67**). It can be set or cleared via bits **MLI_SCR**.SMOD or **MLI_SCR**.CMOD. 0 Automatic Data Mode is disabled. Data read/write operations from/to a Remote Window must be executed by the DMA. 1 Automatic Data Mode selected. Data read/write operations from/to a Remote Window are executed by the MLI.
DW	[10:9]	rh	**Data Width** This bit field is updated by the MLI receiver whenever new data is received in the **MLI_RDATAR** register. It indicates the relevant data width. 00 8-bit relevant data width in **MLI_RDATAR** 01 16-bit relevant data width in **MLI_RDATAR** 10 32-bit relevant data width in **MLI_RDATAR** 11 Reserved
TF	[12:11]	rh	**Type of Frame** This bit field determines the frame type that has most recently been received by the MLI receiver. It is updated whenever the MLI receiver updates **MLI_RDATAR**, **MLI_RADRR**, or MLI_RPxBAR. The most recently received frame was a: 00 Copy Base Address Frame 01 Discrete Read Frame or Optimized Read Frame 10 Write Offset and Data Frame or Optimized Write Frame 11 Answer Frame
PE	13	rh	**Parity Error** PE is set when a parity error is detected by the receiver in a received frame (see **Page 4-34**). PE is cleared by hardware when a frame has been received without parity error. PE can be cleared via bit **MLI_SCR**.CRPE.
RPN	[15:14]	rh	**Received Pipe Number** This bit field contains the Pipe Number that was indicated by the Pipe Number bit field of the latest received frame. It is updated by any received frame.

Field	Bits	Type	Description
MPE	[19:16]	rwh	**Maximum Parity Errors** This bit field indicates the number of receive parity error conditions after which a receiver parity error interrupt event will be generated. It is set to a desired value by software and it is decremented down to 0 automatically by the MLI each time it detects a receiver parity error condition. If a receiver parity error condition is detected and MPE becomes 0 or is already 0, a receiver parity error event is generated (see **Page 4-34**). 0000 A receiver parity event is generated if a receiver error condition is detected 0001 A receiver parity event is generated if a receiver error condition is detected 0010 A receiver parity event is generated if two receiver error conditions are detected ... ... 1111 A receiver parity event is generated if 15 receiver error conditions are detected
BEN	20	rw	**Break Out Enable** When setting BEN = 1, the MLI receiver generates an MLI Break Event when a Pipe Command Frame with command code CMD = 1111_B is received. 0 MLI Break Event generation is disabled. 1 MLI Break Event is enabled.
RCVRST	24	rw	**Receiver Reset** This bit forces the receiver to be reset in order to be able to change **MLI_OICR** settings without affecting the receiver registers. 0 The MLI receiver is in operating mode. 1 The MLI receiver is held in reset state and OICR can be modified without unintentional actions in the receiver.
0	[23:21], [31:25]	r	**Reserved** Read as 0; should be written with 0.

4.3.3.2 Receiver Pipe x Base Address Register

The Receiver Pipe x Base Address Register MLI_RPxBAR (x = 0-3) is a read-only register that contains the complete target address in the Remote Window of Pipe x.

MLI_RP0BAR
MLI Receiver Pipe 0 Base Address Register ($26C_H$) **Reset Value: 0000 0000_H**
MLI_RP1BAR
MLI Receiver Pipe 1 Base Address Register (270_H) **Reset Value: 0000 0000_H**
MLI_RP2BAR
MLI Receiver Pipe 2 Base Address Register (274_H) **Reset Value: 0000 0000_H**
MLI_RP3BAR
MLI Receiver Pipe 3 Base Address Register (278_H) **Reset Value: 0000 0000_H**

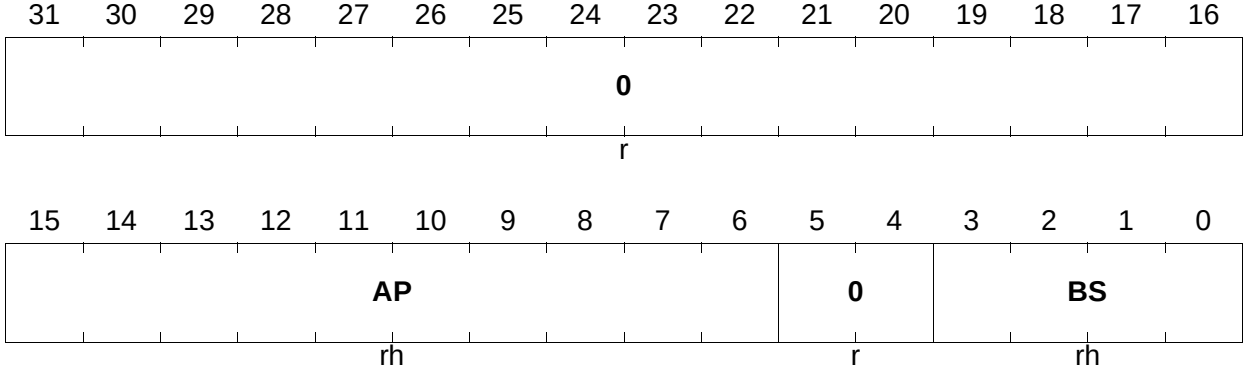


Field	Bits	Type	Description
ADDR	[31:0]	rh	**Address** ADDR indicates the complete target address for the Pipe x Remote Window. If a Pipe x Copy Base Address Frame is received, ADDR[31:4] becomes loaded with the transmitted 28-bit address and bits [3:0] are cleared. If a write or read frame with m bits of address offset is received, bits ADDR[31:m] are held constant and bits ADDR[m-1:0] are replaced by the received offset. If an optimized read or data frame is received, the address prediction mechanism adds the predicted address offset MLI_RPxSTATR.AP to ADDR and stores the result in ADDR. If an Answer Frame is received, ADDR is not changed.

4.3.3.3 Receiver Pipe x Status Register

The Receiver Pipe x Status Register MLI_RPxSTATR (x = 0-3) indicates the data width (8-, 16-, or 32-bit) of the last access to the Pipe x Remote Window and the address prediction factor that has been calculated for Pipe x in the receiver of the Remote Controller.

MLI_RP0STATR
MLI Receiver Pipe 0 Status Register ($27C_H$) Reset Value: 0000 0000_H
MLI_RP1STATR
MLI Receiver Pipe 1 Status Register (280_H) Reset Value: 0000 0000_H
MLI_RP2STATR
MLI Receiver Pipe 2 Status Register (284_H) Reset Value: 0000 0000_H
MLI_RP3STATR
MLI Receiver Pipe 3 Status Register (288_H) Reset Value: 0000 0000_H



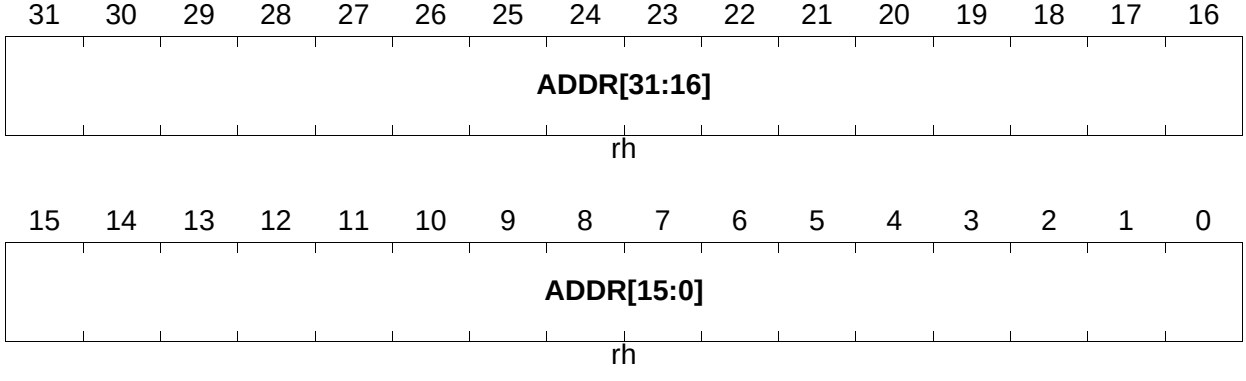
Field	Bits	Type	Description
BS	[3:0]	rh	**Size** This bit field indicates the size of Pipe x Remote Window in the Remote Controller. It is updated by hardware when a Copy Base Address Frame has been received (see **Page 4-19**). 0000 1-bit offset address of Remote Window 0001 2-bit offset address of Remote Window 0010 3-bit offset address of Remote Window ... 1111 16-bit offset address of Remote Window

Field	Bits	Type	Description
AP	[15:6]	rh	**Address Prediction Factor** AP contains the address prediction factor that has been calculated for Pipe x in the receiver of the Remote Controller. It is a signed 9-bit number with the sign in its most significant bit (see **Page 4-38**).
0	[5:4], [31:16]	r	**Reserved** Read as 0; should be written with 0.

4.3.3.4 Receiver Address Register

The Receiver Address Register **MLI_RADRR** is a read-only register storing the complete address of the most recently (or currently) targeted Remote Window.

MLI_RADRR
MLI Receiver Address Register (28C$_H$) Reset Value: 0000 0000$_H$

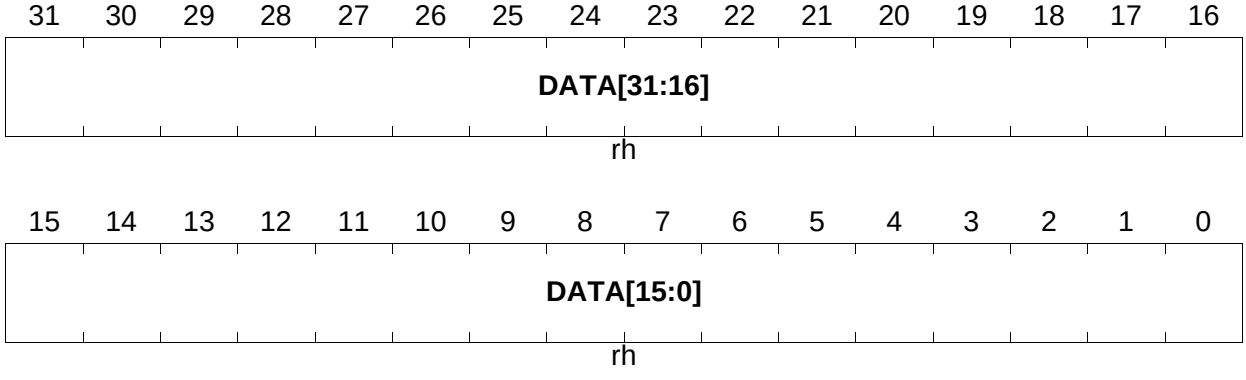


Field	Bits	Type	Description
ADDR	[31:0]	rh	**Address** ADDR indicates the complete target address for the most recently (or currently) targeted Remote Window (Pipe x). If a Copy Base Address Frame is received, ADDR is unchanged. If a write or read frame with m bits of address offset is received, bits ADDR[31:m] replaced by the bits **MLI_RP0BAR**.ADDR[31:m] and bits ADDR[m-1:0] are replaced by the received offset. If an optimized read or data frame is received, the address prediction mechanism adds the predicted address offset **MLI_RP0STATR**.AP to **MLI_RP0BAR**.ADDR and stores the result in ADDR. If an Answer Frame is received, ADDR becomes invalid.

4.3.3.5 Receiver Data Register

The Receiver Data Register **MLI_RDATAR** is a read-only register that stores data received by a write frame or an Answer Frame.

MLI_RDATAR
MLI Receiver Data Register **(290_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** In the Remote Controller, DATA contains the data received by a write frame or an Answer Frame. Bit field **MLI_RCR**.DW determines the width of the relevant data that is stored in **MLI_RDATAR**. **MLI_RCR**.DW = 00:**MLI_RDATAR**[7:0] are relevant (8-bit) **MLI_RCR**.DW = 01:**MLI_RDATAR**[15:0] are relevant (16-bit) **MLI_RCR**.DW = 10:**MLI_RDATAR**[31:0] are relevant (32-bit)

4.3.4 Transmitter Interrupt Registers

4.3.4.1 Transmitter Interrupt Enable Register

The Transmitter Interrupt Enable Register **MLI_TIER** contains the interrupt enable bits and the interrupt request enable flag clear bits for all transmitter interrupt request events. The bits marked w are always read as 0.

MLI_TIER
MLI Transmitter Interrupt Enable Register (298_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						TEIR	PEIR	CFSIR3	CFSIR2	CFSIR1	CFSIR0	NFSIR3	NFSIR2	NFSIR1	NFSIR0
r						w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TEIE	PEIE	CFSIE3	CFSIE2	CFSIE1	CFSIE0	NFSIE3	NFSIE2	NFSIE1	NFSIE0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NFSIE0, **NFSIE1,** **NFSIE2,** **NFSIE3**	0, 1, 2, 3	rw	**Normal Frame Sent in Pipe x Interrupt Enable** 0 Normal Frame sent in Pipe x interrupt source is disabled 1 Normal Frame sent in Pipe x interrupt source is enabled
CFSIE0, **CFSIE1,** **CFSIE2,** **CFSIE3**	4, 5, 6, 7	rw	**Command Frame Sent in Pipe x Interrupt Enable** 0 Command Frame sent in Pipe x interrupt source is disabled 1 Command Frame sent in Pipe x interrupt source is enabled
PEIE	8	rw	**Parity Error Interrupt Enable** 0 Parity error interrupt source is disabled 1 Parity error interrupt source is enabled
TEIE	9	rw	**Time-Out Error Interrupt Enable** 0 Time-out error interrupt source is disabled 1 Time-out error interrupt source is enabled

Field	Bits	Type	Description
NFSIR0 NFSIR1, NFSIR2, NFSIR3	16, 17, 18, 19	w	**Normal Frame Sent in Pipe x Flag Clear** 0 No action 1 Clear **MLI_TISR**.NFSIx
CFSIR0, CFSIR1, CFSIR2, CFSIR3	20, 21, 22, 23	w	**Command Frame Sent in Pipe x Flag Clear** 0 No action 1 Clear **MLI_TISR**.CFSIx
PEIR	24	w	**Parity or Time Out Error Flag Clear** 0 No action 1 Clear **MLI_TISR**.PEIx
TEIR	25	w	**Time Out Error Flag Clear** 0 No action 1 Clear **MLI_TISR**.TEIx
0	[15:10], [31:26]	r	**Reserved** Read as 0; should be written with 0.

4.3.4.2 Transmitter Interrupt Register

The Transmitter Interrupt Status Register **MLI_TISR** contains all of the interrupt request flags of the MLI transmitter. These interrupt request flags can be cleared by software when writing the appropriate bits in the **MLI_TIER** register.

MLI_TISR

MLI Transmitter Interrupt Status Register ($29C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TEI	PEI	CFSI3	CFSI2	CFSI1	CFSI0	NFSI3	NFSI2	NFSI1	NFSI0
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

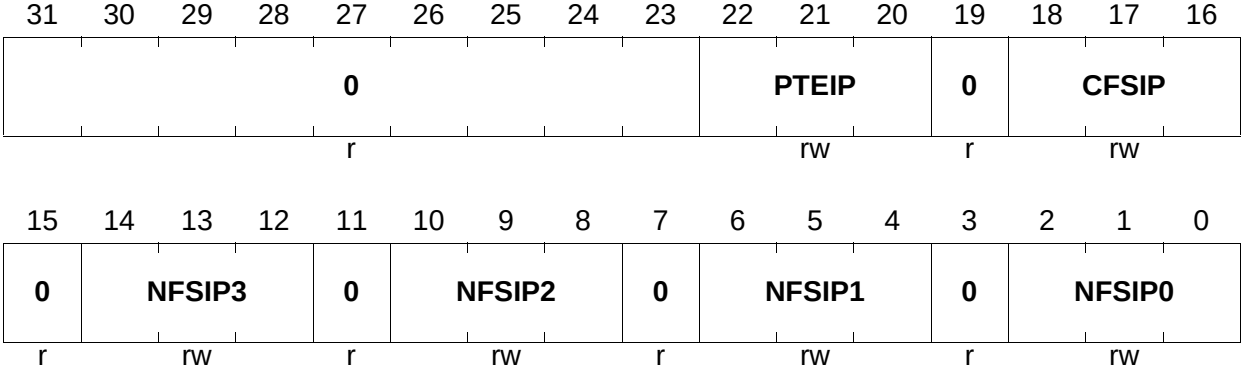
Field	Bits	Type	Description
NFSI0, **NFSI1,** **NFSI2,** **NFSI3**	0, 1, 2, 3	rh	**Normal Frame Sent in Pipe x Flag** This flag becomes set if a write or read frame has been correctly sent and acknowledged for Pipe x. The service request output that is activated is defined by **MLI_TINPR**.NFSIPx.
CFSI0, **CFSI1,** **CFSI2,** **CFSI3**	4, 5, 6, 7	rh	**Command Frame Sent in Pipe x Flag** This flag becomes set if a Command Frame has been correctly sent and acknowledged for Pipe x. The service request output that is activated is defined by **MLI_TINPR**.CFSIP.
PEI	8	rh	**Parity Error Flag** This flag becomes set if a transmitter parity error interrupt event has been detected. The service request output that is activated is defined by **MLI_TINPR**.PTEIPx.
TEI	9	rh	**Time-Out Error Flag** This flag becomes set if a time-out error interrupt event has been detected The service request output that is activated is defined by **MLI_TINPR**.PTEIPx.

Field	Bits	Type	Description
0	[31:10]	r	**Reserved** Read as 0; should be written with 0.

4.3.4.3 Transmitter Interrupt Node Pointer Register

The Transmitter Interrupt Node Pointer Register **MLI_TINPR** contains the interrupt node pointers for the MLI transmitter interrupts events.

MLI_TINPR
MLI Transmitter Interrupt Node Pointer Register ($2A0_H$) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
NFSIP0	[2:0]	rw	**Normal Frame Sent in Pipe 0 Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Normal Frame sent in Pipe 0 interrupt occurs. 000 MLI Request0 is selected 001 MLI Request1 is selected 010 MLI Request2 is selected 011 MLI Request3 is selected 100 Reserved, do not use 101 Reserved, do not use 110 Reserved, do not use 111 Reserved, do not use
NFSIP1	[6:4]	rw	**Normal Frame Sent in Pipe 1 Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Normal Frame sent in Pipe 1 interrupt occurs. Coding see NFSIP0.

Field	Bits	Type	Description
NFSIP2	[10:8]	rw	**Normal Frame Sent in Pipe 2 Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Normal Frame sent in Pipe 2 interrupt occurs. Coding see NFSIP0.
NFSIP3	[14:12]	rw	**Normal Frame Sent in Pipe 3 Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Normal Frame sent in Pipe 3 interrupt occurs. Coding see NFSIP0.
CFSIP	[18:16]	rw	**Command Frame Sent Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Command Frame sent interrupt occurs. Coding see NFSIP0.
PTEIP	[22:20]	rw	**Parity or Time Out Interrupt Pointer** This bit field determines which MLI Request x becomes active when a parity/time-out interrupt occurs. Coding see NFSIP0.
0	3, 7, 11, 15, 19, [31:23]	r	**Reserved** Read as 0; should be written with 0.

4.3.5 Receiver Interrupt Registers

4.3.5.1 Receiver Interrupt Enable Register

The Receiver Interrupt Enable Register **MLI_RIER** contains the interrupt enable bits and the interrupt request enable flag clear bits for all receiver interrupt request sources. The bits marked w are always read as 0.

MLI_RIER
MLI Receiver Interrupt Enable Register (2A4$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DRAIR	0	PEIR	ICER	CFRIR3	CFRIR2	CFRIR1	CFRIR0	MEIR	NFRIR
r						w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DRAIE	0	PEIE	ICE	CFRIE3	CFRIE2	CFRIE1	CFRIE0	NFRIE	
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
NFRIE	[1:0]	rw	**Normal Frame Received Interrupt Enable** This bit field defines whether an interrupt is generated when a Normal Frame is correctly received. 00 The interrupt generation is disabled 01 The interrupt is generated each time a Normal Frame is correctly received 10 The interrupt is generated each time a Normal Frame is correctly received that is not handled automatically by the MLI (e.g. an Answer Frame) 11 Reserved
CFRIE0, CFRIE1, CFRIE2, CFRIE3	2, 3, 4, 5	rw	**Command Received in Pipe x Interrupt Enable** This bit determines whether an interrupt is generated when a Command Frame for Pipe x has been received correctly. 0 Command received in Pipe x interrupt is disabled 1 Command received in Pipe x interrupt is enabled

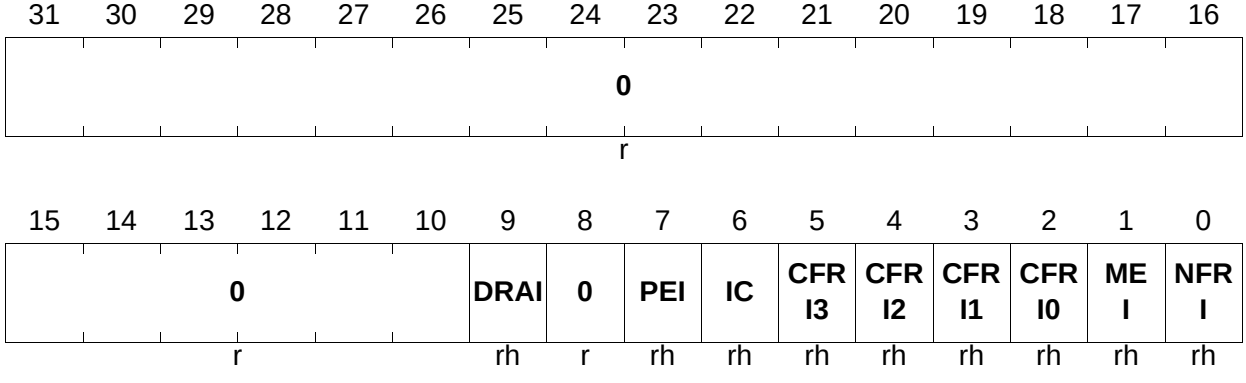
Field	Bits	Type	Description
ICE	6	rw	**Interrupt Command Enable** This bit determines whether an interrupt is generated when a Command Frame is received in Pipe 0. 0 Command Frame received in Pipe 0 interrupt is disabled 1 Command Frame received in Pipe 0 interrupt is enabled
PEIE	7	rw	**Parity Error Interrupt Enable** This bit enables the interrupt generated if receiver a parity error event is detected. 0 Parity error interrupt is disabled 1 Parity error interrupt is enabled
DRAIE	9	rw	**Discarded Read Answer Interrupt Enable** This bit enables the interrupt generated if a discarded read Answer Frame condition is detected. 0 Discarded read answer interrupt is disabled 1 Discarded read answer interrupt is enabled
NFRIR	16	w	**Normal Frame Received Interrupt Flag Clear** 0 No action 1 Clear **MLI_RISR**.NFRI
MEIR	17	w	**MLI Move Engine Interrupt Flag Clear** 0 No action 1 Clear **MLI_RISR**.MEI
CFRIR0 CFRIR1, CFRIR2, CFRIR3	18, 19, 20, 21	w	**Command Frame Received through Pipe x Interrupt Flag Clear** 0 No action 1 Clear **MLI_RISR**.CFRIx
ICER	22	w	**Interrupt Command Flag Clear** 0 No action 1 Clear **MLI_RISR**.ICE
PEIR	23	w	**Parity Error Interrupt Flag Clear** 0 No action 1 Clear **MLI_RISR**.PEI
DRAIR	25	w	**Discarded Read Answer Interrupt Flag Clear** 0 No action 1 Clear **MLI_RISR**.DRAI
0	8, 24	w	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
0	[15:10] [31:26]	r	**Reserved** Read as 0; should be written with 0.

4.3.5.2 Receiver Interrupt Status Register

The Receiver Interrupt Status Register **MLI_RISR** contains all of the interrupt request flags of the MLI receiver. These interrupt request flags can be cleared by software when writing the appropriate bits in the **MLI_RIER** register.

MLI_RISR
MLI Receiver Interrupt Status Register($2A8_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
NFRI	0	rh	**Normal Frame Received Interrupt Flag** This flag is set when a write or a read frame has been received. The MLI Request that is activated is defined by **MLI_RINPR**.NFRIP.
MEI	1	rh	**MLI Move Engine Interrupt Flag** This flag is set when the MLI has finished an operation (read or write, depending on received frame). The MLI Request that is activated is defined by **MLI_RINPR**.MPPEIP.
CFRI0 **CFRI1,** **CFRI2,** **CFRI3**	2, 3, 4, 5	rh	**Command Frame Received through Pipe x Interrupt Flag** This flag is set when a Command Frame has been received in Pipe x. The MLI Request that is activated is defined by **MLI_RINPR**.CFRIP.

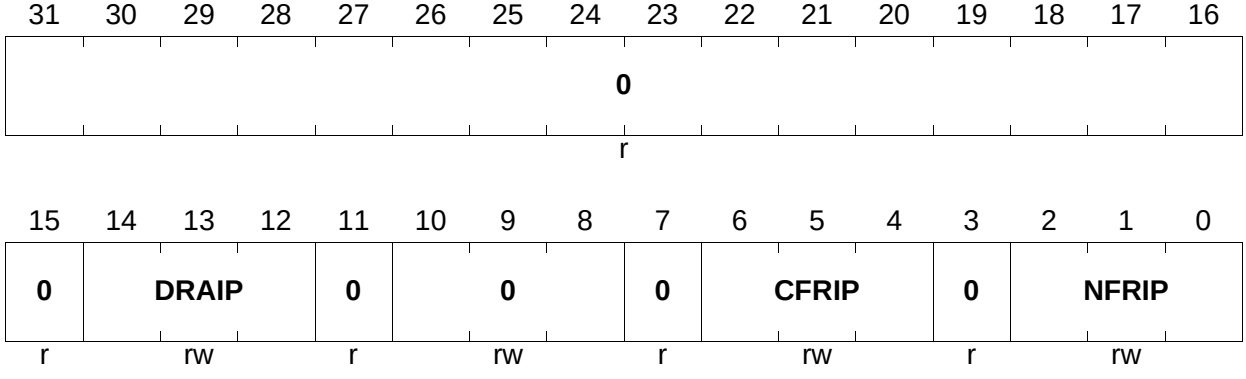
Field	Bits	Type	Description
IC	6	rh	**Interrupt Command Flag** This flag is set when a Command Frame has been received in Pipe 0 leading to an activation of one of the MLI Requests. The MLI Request that is activated is defined by the received command CMD.
PEI	7	rh	**Parity Error Interrupt Flag** This flag is set when a parity error interrupt event has occurred. The MLI Request that is activated is defined by **MLI_RINPR**.MPPEIP.
DRAI	9	rh	**Discarded Read Answer Interrupt Flag** This flag is set when the discarded read answer interrupt event has occurred. This condition occurs if an Answer Frame is received while none of the **MLI_TRSTATR**.RPx bits is set (the Answer Frame was not expected). The MLI Request that is activated is defined by **MLI_RINPR**.DRAIP.
0	8, [31:10]	r	**Reserved** Read as 0; should be written with 0.

4.3.5.3 Receiver Interrupt Node Pointer Register

The Receiver Interrupt Node Pointer Register **MLI_RINPR** contains the interrupt node pointers for the MLI receiver interrupts.

MLI_RINPR

MLI Receiver Interrupt Node Pointer Register ($2AC_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
NFRIP	[2:0]	rw	**Normal Frame Received Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Normal Frame received interrupt occurs. 000 MLI Request 0 is selected 001 MLI Request 1 is selected 010 MLI Request 2 is selected 011 MLI Request 3 is selected 100 Reserved, do not use 101 Reserved, do not use 110 Reserved, do not use 111 Reserved, do not use
CFRIP	[6:4]	rw	**Command Frame Received Interrupt Pointer** This bit field determines which MLI Request x becomes active when a Command Frame received interrupt occurs. Coding see NFRIP.
DRAIP	[14:12]	rw	**Discarded Read Answer Interrupt Pointer** This bit field determines which MLI Request x becomes active when a discarded read answer interrupt occurs. Coding see NFRIP.
0	[10:8]	r	**Reserved** Should be written with 0.

Field	Bits	Type	Description
0	3, 7, 11 [31:15]	r	**Reserved** Read as 0; should be written with 0.

4.4 MLI Address Map

The MLI module supports four Small Transfer Windows (STW)—one for each Pipe—and four Large Transfer Windows (LTW)—one for each Pipe.

Transfer Window Areas and MLI Register Address Space

Table 4-11 Transfer Window Areas

Module	Base Address	End Address	Note
STW Pipe 0	0000 8000_H	0000 $9FFF_H$	8 kBytes max.
STW Pipe 1	0000 $A000_H$	0000 $BFFF_H$	8 kBytes max.
STW Pipe 2	0000 $C000_H$	0000 $DFFF_H$	8 kBytes max.
STW Pipe 3	0000 $E000_H$	0000 $FFFF_H$	8 kBytes max.
LTW Pipe 0	0001 0000_H	0001 $FFFF_H$	64 kBytes max.
LTW Pipe 1	0002 0000_H	0002 $FFFF_H$	64 kBytes max.
LTW Pipe 2	0003 0000_H	0003 $FFFF_H$	64 kBytes max.
LTW Pipe 3	0004 0000_H	0004 $FFFF_H$	64 kBytes max.

5 Synchronous Serial Interface (SSC)

This chapter describes how the SSC interface is used in the CIC751.

5.1 Overview

The SSC supports full-duplex and half-duplex serial synchronous communication up to 10 Mbit/s (@ 40 MHz module clock). The serial clock signal is received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A shift clock generator provides the SSC with a separate serial clock signal.

This chapter describes only the use of the SSC module as a slave because the CIC751 always operates as a slave to a host.

Features

- Slave Mode operation
  - Full-duplex or half-duplex operation
  - Automatic pad control possible
- Flexible data format
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: Idle low or idle high state for the shift clock
  - Programmable clock/data phase: Data shift with leading or trailing edge of the shift clock
- Internal Master Function
  - Access to the all addresses
  - Automatic address handling
  - Automatic data handling

5.2 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 10 Mbit/s (@ 40 MHz module clock). The serial clock signal is received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A shift clock generator provides the SSC with a separate serial clock signal.

Configuration of the high-speed synchronous serial interface is very flexible, so it can work with other synchronous serial interfaces, can serve for master/slave or multi-master interconnections, or can operate compatibly with the popular SPI interface. The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is received via pin SCLK (Serial Clock). Using the RDY pin, the CIC751 signals the master that SCLK can be activated. The SSC can be selected from a master via the Slave Select input Line ($\overline{SLS}$).

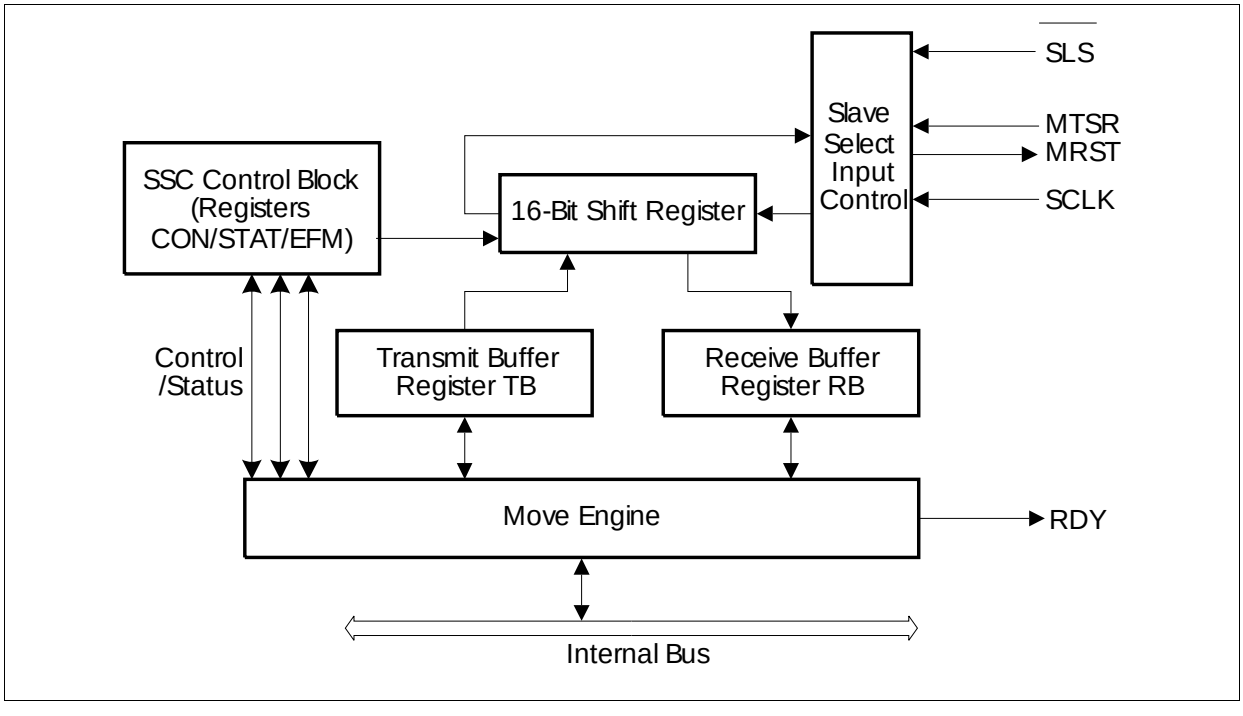


Figure 5-1 Synchronous Serial Channel SSC Block Diagram

5.2.1 SPI Communication Basics

There are two principal modes of operation for SPI communication: Full-Duplex and Half-Duplex.

5.2.1.1 Full-Duplex Operation

The description in this section assumes that the SSC is used with software controlled bi-directional GPIO port lines that have an open-drain capability.

The various 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 (Master Transmit Slave Receive) is the transmit line, the receive line is connected to its data input line MRST (Master Receive Slave Transmit), and the clock line is connected to pin SCLK. Only the device selected for master operation generates and outputs the serial clock on pin SCLK. All slaves receive this clock, so their pin SCLK must be switched to input mode. 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, with the function and direction of these pins determined by the master or slave operation of the individual device.

*Note: The shift direction shown in **Figure 5-2** applies to both MSB-first and LSB-first operation.*

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

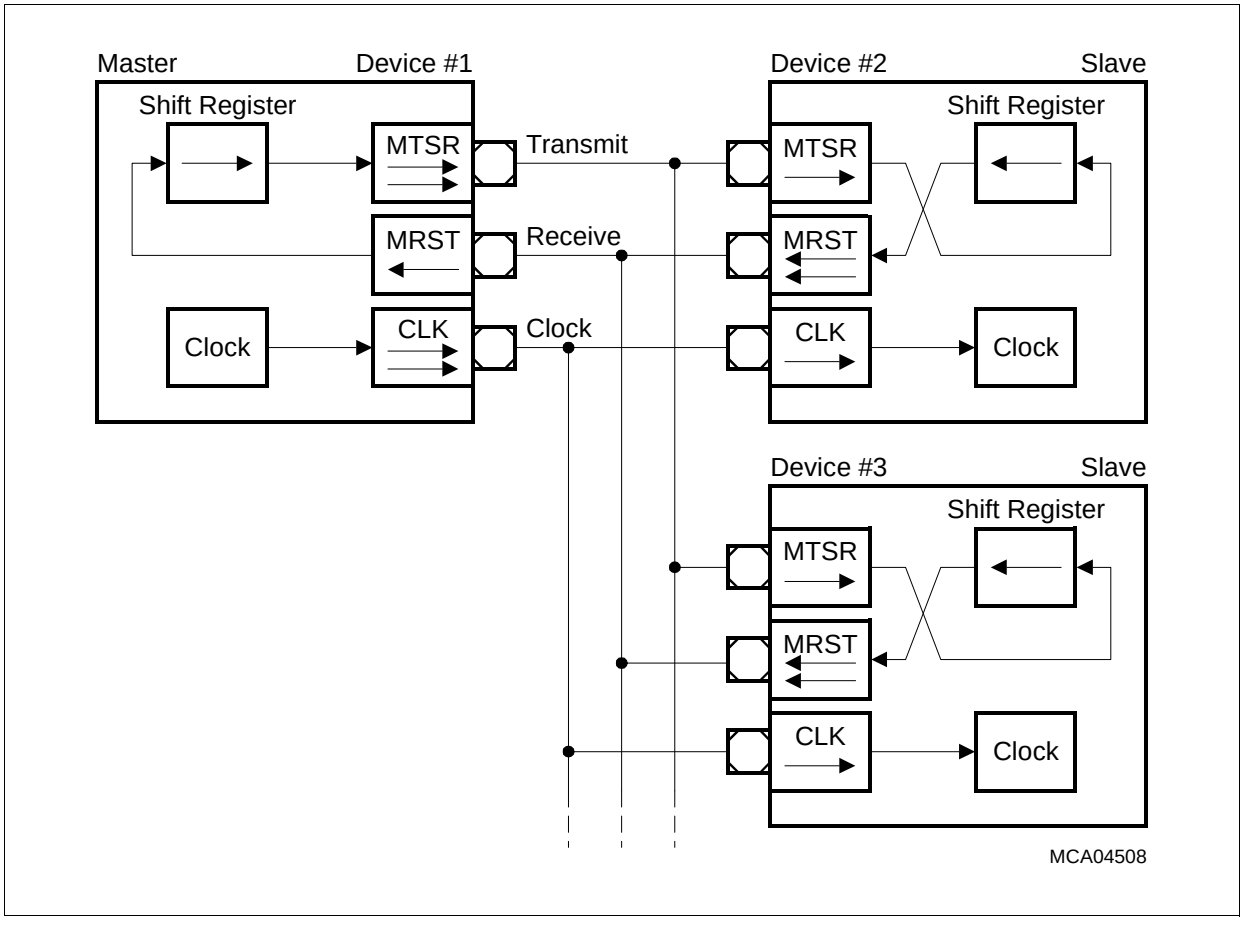


Figure 5-2 SSC Full-Duplex Configuration

The data output pins (MRST) of all slave devices are connected to 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:

- **Only one slave drives the line** and enables the driver of its MRST pin. All the other slaves must program their MRST pins to input. Therefore, only one slave can input its data to the master's receive line. Only reception 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 de-selection signal or command.
- **The slaves use an open drain output on MRST**. This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master send only 1s. Since this high level is not actively driven onto the line, but is only held through the pull-up device, the selected slave can pull this line actively to a low level when transmitting a 0 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 initialization tasks for 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 starts. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the MTSR line on the next clock from the shift clock generator. Depending on the selected clock phase, a clock pulse is generated on the SCLK line. With the opposite clock edge, the master simultaneously latches and shifts in the data detected at its input line MRST. This “exchanges” the transmit data with the receive data. Because 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 pre-programmed 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. In the master and all slaves, the content of the shift register is copied into the Receive Buffer (SSC_RB).

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 transmit buffer is copied into the slave’s shift register. Bit SSC_STAT.BSY is not set until the first clock edge at SCLK appears. The slave device will not wait for the next clock from the shift clock generator—as the master does—because the first clock edge generated by the master may be already used to clock in the first data bit, depending on the selected clock phase. So, the slave’s first data bit must already be valid at this time.

Note: On the SSC, a transmission and a reception always takes place at the same time, regardless of whether valid data has been transmitted or received.

5.2.1.2 Half-Duplex Operation

The description in this section assumes that the SSC is used with software controlled bi-directional GPIO port lines that provide an open-drain capability.

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

As in full-duplex mode, there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver
- The non-transmitting devices use open-drain output and only send 1’s

Because 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). In this way, any corruption is detected on the common data exchange line when the received data is not equal to the transmitted data.

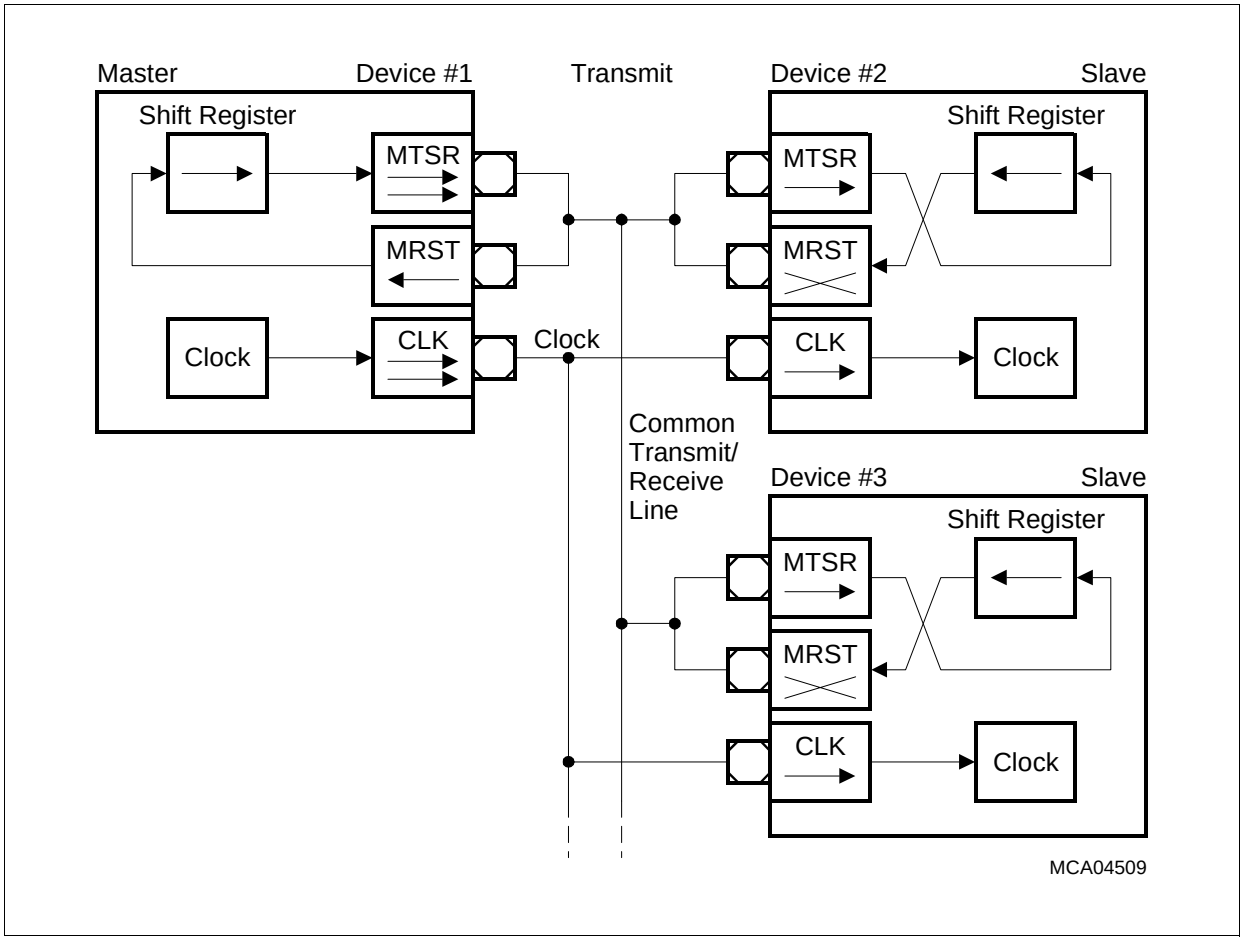


Figure 5-3 SSC Half-Duplex Configuration

5.2.2 Operating the SSC

The following sections explain how the SSC is best used for operation of the CIC751.

The basic task of the SSC is to communicate with a host controller to which the CIC751 is connect. Therefore, the CIC751 SSC always operates as a slave within the serial communication. The communication requirements can be split into two categories:

- Configuration of the CIC751
  - This requires write access from the host to the CIC751 to update the various control registers.
- Transfer of the conversion result back to the host
  - The conversion results need to be transferred back to the requesting host. The CIC751 provides all required hardware support so that the conversion results can be communicated back to the host in a semi-automatic way.

5.2.2.1 SSC Transaction Header

The first halfword that is send by the host is interpreted as the transaction header and is composed of the CMD bit, the INCE bit and an address ADDR of 14 bits. Therefore, the master (host) must first send the 16-bit header information before sending each communication block. A communication block is composed of the transaction header and one or more 16-bit communication data blocks.

The following table defines the SSC transaction header.

Table 5-1 Transaction Header

Name	Description	Bit Position
CMD	**Transfer Type Identifier** 0 A read transfer is selected 1 A write transfer is selected	15
INCE	**Address Increment Enable** 0 The destination address is not increment after each transaction 1 The destination address is increment after each transaction	14
ADDR	**Destination Address**	13…0

CMD determines if the following portion of the communication indicates a write or read operation.

INCE determines whether or not the address is automatically incremented by two after the first data block.

Example: INCE = 1

Setup for the source and destination addresses of DMA Channel 5 for additional data transfers must be updated. Therefore, the DMA channel registers DMA_ADRCR5, DMA_SADR5, and DMA_DADR5 must be updated. Beginning with register DMA_ADRCR5, all three registers can be updated with a single SSC communication block.

5.2.2.2 SSC Data Flow Model

As the SSC operates only as slave, the master must follow some rules to establish a working communication link.

Communication Rules

- An SSC communication block is composed out of one Transaction Header and several data blocks.
- An SSC communication block is started by the master with an assertion of $\overline{SLS}$
- An SSC communication block is stopped by the master with a de-assertion of $\overline{SLS}$
- The duration of a communication block should always be a multiple of 16 SCLK cycles
- A data block is the data that is transmitted during the 16 SCLK cycles
- Only 16-bit data blocks are legal for an SSC communication
- The first 16-bit data block is always used as the transaction header
- All 16-bit data blocks after the transaction header are used as write or read data

SSC Write Operation (CMD = 1)

For each SSC transfer that is received via the SSC interface, the transaction header information is extracted from the first transmitted halfword—which is the command CMD, the increment indicator INCE, and the address ADDR. The following halfwords are then used as data. With CMD, INCE, and ADDR available, the data is copied to the destination address according to the setting of INCE and ADDR.

The pin RDY is provided for additional synchronization between the host and slave. This pin or information is required due to the fact that the master does not know how much time (system cycles) is exactly consumed by the CIC751 to move the transmitted write data to the desired destination. The amount of time depends on the frequency of the CIC751 and the currently active CIC751 register accesses performed by the DMA. Therefore, the RDY pin is introduced to show that the SSC interface is ready for the next part of the transmission (RDY is asserted).

The RDY pin should be used by the master in the following way:

- If RDY changes from de-asserted (’0’) to asserted (’1’), the master starts to generate 16 clock cycles for SCLK
- If RDY changes from asserted (’1’) to de-asserted (’0’), the master takes no action

The default level of RDY is ’0’. If a master selects the CIC751 for an SSC communication, SLS is asserted (’0’) and RDY is changed to asserted (’1’). This allows the master to start with the transmission of the transaction header (the first 16 SCLK cycles). Meanwhile, RDY is de-asserted again to be ready for the next use and to signal the master that the first SCLK cycle was received. Using the first address, the first write data is forwarded to the destination register. This is indicated by the assertion of RDY again. Thereafter, the master can generate the next 16 SCLK required for the next write data to transmit. With the first cycle of SCLK, RDY is de-asserted again to be ready for the next use and to signal the master that the first SCLK cycle was received. This sequence is repeated until SLS is de-asserted (’1’) by the master after RDY was asserted.

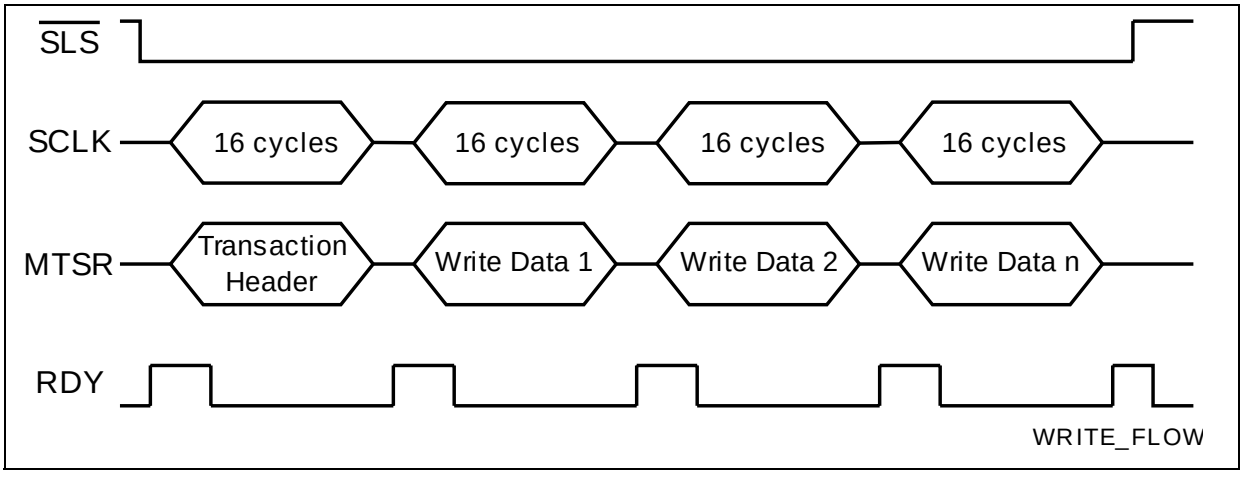


Figure 5-4 Consecutive Writes

SSC Read Operation (CMD = 0)

For each SSC transfer that is received via the SSC interface, the transaction header information is extracted from the first transmitted halfword—which is the command CMD, the increment indicator INCE, and the address ADDR. With this header, the data from the source address on the CIC751 reads automatically and sends them back to the host via the SSC transmit buffer SSC_TB.

The pin RDY is provided for additional synchronization between the host and slave. This pin or information is required due to the fact that the master does not know how much time (system cycles) is exactly consumed by the CIC751 for fetching the requested read data. This depends on the frequency of the CIC751 and the currently active CIC751 register accesses performed by the DMA. Therefore, the RDY pin is introduced to ensure that the read data is ready for transmission (RDY is asserted).

The RDY pin should be used by the master in the following way:

- If RDY changes from de-asserted (’0’) to asserted (’1’), the master starts to generate 16 clock cycles for SCLK
- If RDY changes from asserted (’1’) to de-asserted (’0’), the master takes no action

The default level of RDY is ’0’. If a master selects the CIC751 for an SSC communication, SLS is asserted (’0’) and RDY is changed to asserted (’1’). This allows the master to start with the transmission of the transaction header (the first 16 SCLK cycles). Meanwhile, RDY is de-asserted again to be ready for the next use and to signal the master that the first SCLK cycle was received. Using the first address, the first read data is fetched and is ready for transmission. This is indicated by the assertion of RDY again. Thereafter, the master can generate the next 16 SCLK required for the slave to transmit the 16-bit read data. With the first cycle of SCLK, RDY is de-asserted again to be ready for the next use and to signal the master that the first SCLK cycle was received and the transmission of the first read data is started. In parallel with the transmission of the read data, the next read is fetched and prepared for transmission either from address ADDR (INCE = 0) or from address ADDR + 2 (INCE = 1). This sequence is repeated until SLS is de-asserted (’1’) by the master.

This automatic read process is optimized for several consecutive read data transfers within one communication block. Therefore, the next to transmit read data is prefetched during the transmission of the currently processed data. This leads to a minimum dead time between the transmission of two 16-bit read data parts and an increase of the maximum usable bandwidth for communication. But, on the other hand, this mechanism has a negative impact for starting a read access.

After the first read access to the CIC751, the last read data that was prefetched is still read for transmission and will be automatically transmitted in parallel with the reception of the next transaction header. Automatically, during the transmission of this prefetched data, a new prefetch is started before the new transaction header is taken into account.

Therefore, the following rules must be considered for read accesses by the master:

- The read data that is received in parallel with sending the transaction header should be ignored
- The first read data that is received after sending the transaction header should be ignored
- The above mentioned rules does not apply to the first read access after a reset (PORST or SW reset) or a write access

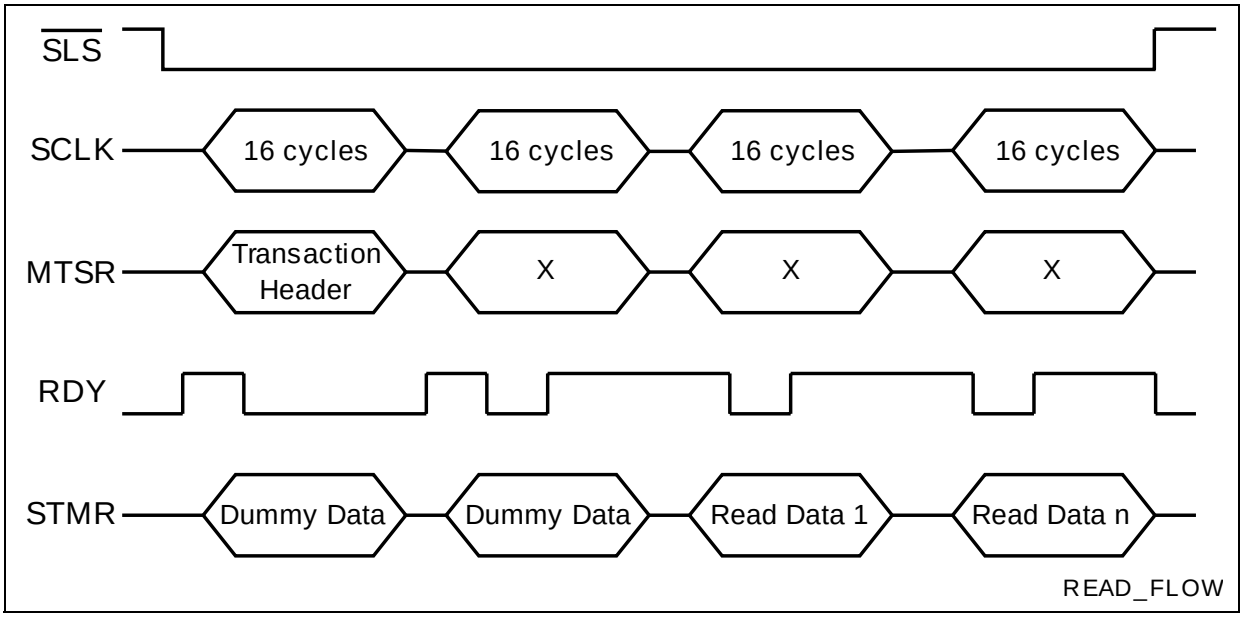


Figure 5-5 Consecutive Reads

5.2.3 Operating Mode Selection

The following features of the serial data bit transfer can be programmed:

- A transfer may start with the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading or trailing edge of the clock signal
- The baud rate (shift clock) can be set from 0.15 bit/s up to 10 Mbit/s (@ 40 MHz module clock)

These features allow the SSC to be adapted to a wide range of applications that require serial data transfer.

Regardless of whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers **SSC_TB** and **SSC_RB**, 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 Clock Control allows the adaptation of the transmit and receive behavior 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 **SSC_CON**.PH selects the leading edge or the trailing edge for each function. Bit **SSC_CON**.PO selects the level of the clock line in the idle state. For an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see **Figure 5-6**).

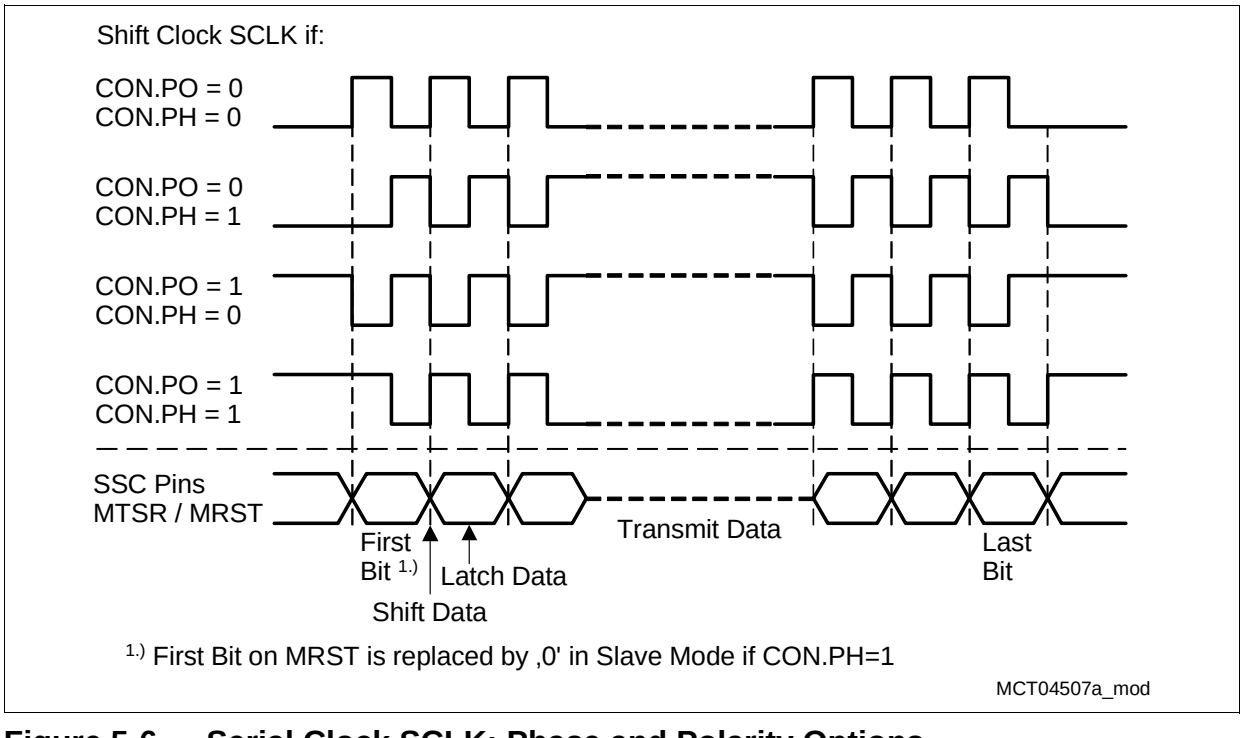


Figure 5-6 Serial Clock SCLK: Phase and Polarity Options

5.2.4 Error Detection Mechanisms

The SSC is able to detect three different error conditions. Receive Error, Phase Error, and Transmit Error. When an error is detected, the respective error flag is always set. The error flags are not cleared automatically, but must be cleared via register SSC_EFM after servicing. The error status flags can be set and cleared by software via the error flag modification register SSC_EFM.

A **Receive Error** is detected when a new data frame is completely received, but the previous data was not read out of the receive buffer register RB. This condition sets the error flags STAT.RE and SCU_ERRCUM.RE. The old data in the receive buffer RB will be overwritten with the new value and is unrecoverable lost.

A **Phase Error** is detected when the incoming data at pin MTSR (Slave Mode), sampled with the same frequency as the system clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error status flags STAT.PE and SCU_ERRCUM.PE.

Note: When using the setting SSC_CON.PH = 1, phase errors can occur due to the fact that the slave select signal change can result in a change of the data signal. The slave select signal always changes with the leading clock edge.

A **Transmit Error** is detected when a transfer was initiated by the master (shift clock becomes active), but the transmit buffer TB of the slave was not updated since the last transfer. This condition sets the error status flags STAT.TE and SCU_ERRCUM.TE. If a transfer starts while the transmit buffer is not updated, the slave will shift out the ‘old’ contents of the shift register, which is normally 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.

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

5.3 Register Descriptions

Table 5-2 identifies all of the SSC registers.

The base address of the SSC is 0000 0900. A register address is computed by adding the base address to the register offset address.

Table 5-2 Registers Overview

Register Short Name	Register Long Name	Offset Address	Page Number
SSC_CON	Control Register	10_H	**Page 5-15**
SSC_STAT	Status Register	28_H	**Page 5-18**
SSC_EFM	Error Flag Modification Register	$2C_H$	**Page 5-19**
SSC_TB	Transmit Buffer Register	20_H	**Page 5-21**
SSC_RB	Receive Buffer Register	24_H	**Page 5-22**
SSC_BR	Baud Rate Timer Reload Register	14_H	**Page 5-22**

SSC_CON
SSC Control Register **(10_H)** **Reset Value: 0000 875F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	MS	0	0		PEN	REN	TEN	LB	PO	PH	HB	BM			
rw	rw	r	rw		rw	rw	rw	rw	rw	rw	rw	rw			

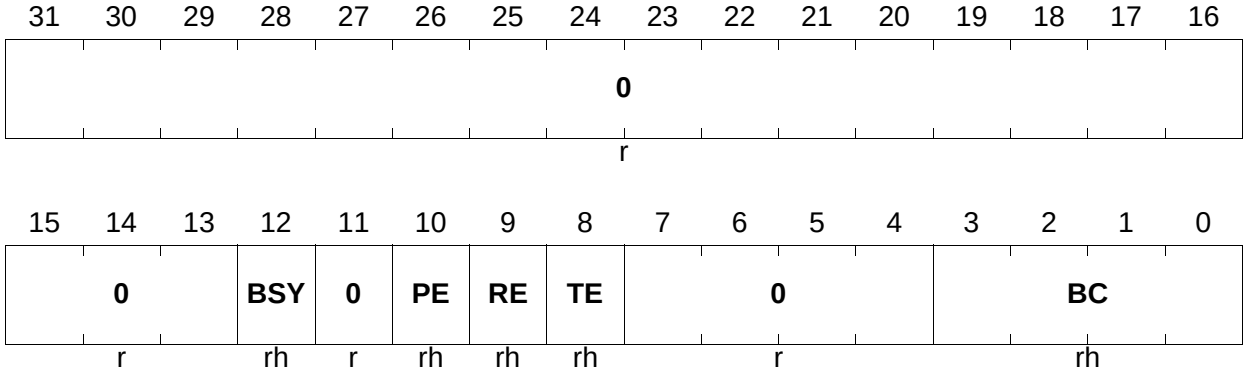
Field	Bits	Type	Description
BM	[3:0]	rw	**Data Width Selection** BM determines the number of data bits of the serial frame. The data width is set to 16-bit and should never be changed. Always write 1111_B to this bit field when this register is updated. Otherwise, the communication is corrupted.

Field	Bits	Type	Description
HB	4	rw	**Heading Bit Control** 0 Transmit/Receive LSB First 1 Transmit/Receive MSB First
PH	5	rw	**Clock Phase Control** 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
PO	6	rw	**Clock Polarity Control** 0 Idle clock line is low, the leading clock edge is low-to-high transition 1 Idle clock line is high, the leading clock edge is high-to-low transition
LB	7	rw	**Loop-Back Control** 0 Normal output 1 Receive input is connected to transmit output (Half-duplex Mode)
TEN	8	rw	**Transmit Error Enable** 0 Ignore transmit errors 1 Check transmit errors
REN	9	rw	**Receive Error Enable** 0 Ignore receive errors 1 Check receive errors
PEN	10	rw	**Phase Error Enable** 0 Ignore phase errors 1 Check phase errors
MS	14	rw	**Master Select** 0 Slave Mode. Operate on shift clock received via SCLK 1 Master Mode. This mode should not be selected. The module should also operated in Slave Mode only. Always set this bit when this register is updated. Otherwise, the communication is corrupted.

Field	Bits	Type	Description
EN	15	rw	**Enable Bit** 0 Transmission and reception are disabled. 1 Transmission and reception are enabled. The module should also be enabled. Always set this bit when this register is updated. Otherwise, the communication is corrupted.
0	[12:11]	rw	**Reserved** Returns 0_B if read; has to be written with 0_B. Writing something different than 0_B could lead to a corruption of the communication.
0	13, [31:16]	r	**Reserved** Read as 0; should be written with 0.

The Status Register SSC_STAT contains status flags for error identification, the busy flag, and a bit field that indicates the current shift counter status.

SSC_STAT
SSC Status Register **(28_H)** **Reset Value: 0000 0000$_H$**

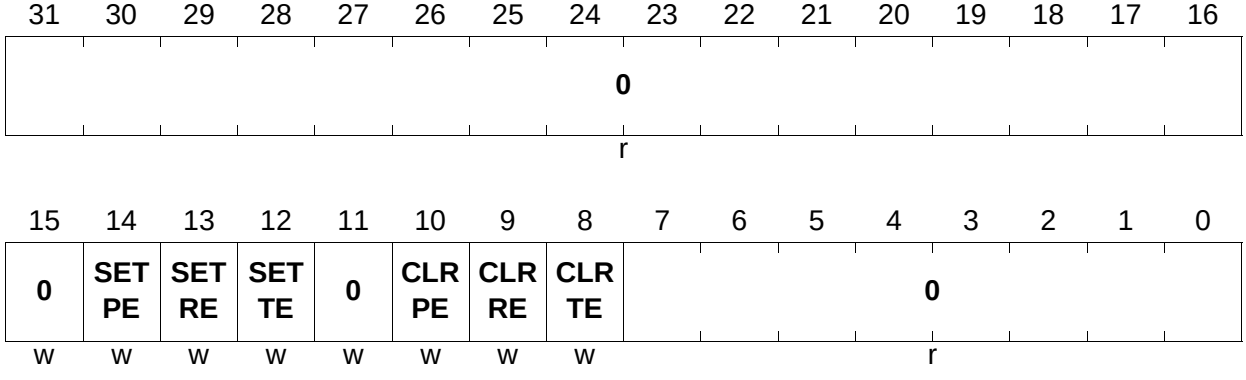


Field	Bits	Type	Description
BC	[3:0]	rh	**Bit Count Status** BC indicates the current status of the shift counter. The shift counter is updated with every shifted bit.
TE	8	rh	**Transmit Error Flag** 0 No error 1 Transfer starts with the slave's transmit buffer not being updated
RE	9	rh	**Receive Error Flag** 0 No error 1 Reception completed before the receive buffer was read
PE	10	rh	**Phase Error Flag** 0 No error 1 Received data changes during the sampling clock edge
BSY	12	rh	**Busy Flag** BSY is set while a transfer is in progress.
0	[7:4],11 [31:13]	r	**Reserved** Read as 0; should be written with 0.

The Error Flag Modification Register SSC_EFM is required for resetting or setting the four error flags that are located in register SSC_STAT.

SSC_EFM
SSC Error Flag Modification Register ($2C_H$)

Reset Value: 0000 0000$_H$

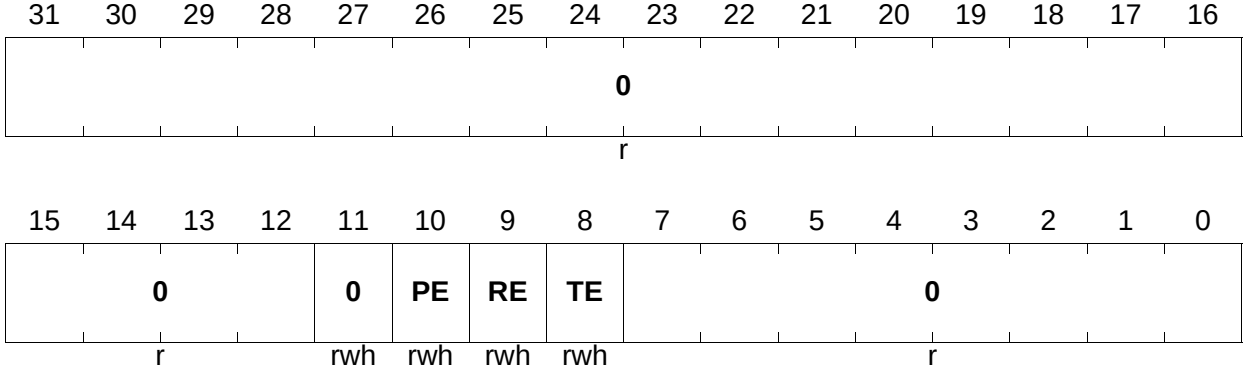


Field	Bits	Type	Description
CLRTE	8	w	**Clear Transmit Error Flag** 0 No effect 1 Bit SSC_STAT.TE is cleared Bit is always read as 0.
CLRRE	9	w	**Clear Receive Error Flag** 0 No effect 1 Bit SSC_STAT.RE is cleared Bit is always read as 0.
CLRPE	10	w	**Clear Phase Error Flag** 0 No effect 1 Bit SSC_STAT.PE is cleared Bit is always read as 0.
SETTE	12	w	**Set Transmit Error Flag** 0 No effect 1 Bit SSC_STAT.TE is set Bit is always read as 0.
SETRE	13	w	**Set Receive Error Flag** 0 No effect 1 Bit SSC_STAT.RE is set Bit is always read as 0.
SETPE	14	w	**Set Phase Error Flag** 0 No effect 1 Bit SSC_STAT.PE is set Bit is always read as 0.

Field	Bits	Type	Description
0	11, 15	w	**Reserved** Read as 0; have to be written with 0.
0	[7:0], [31:16]	r	**Reserved** Read as 0; should be written with 0.

SCU_ERRCUM

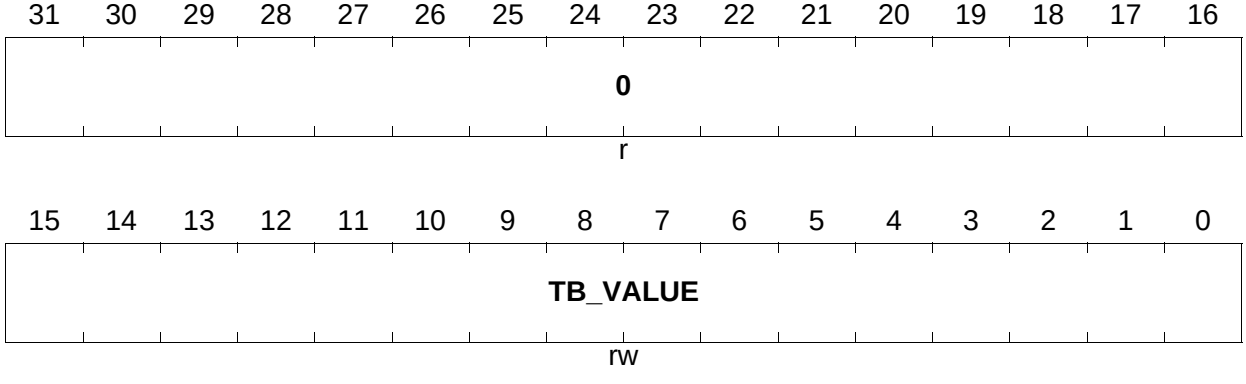
SCU Cumulative Error Register ($85C_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TE	8	rwh	**Transmit Error Flag** 0 No error 1 Transfer starts with transmit buffer not being updated This bit is set in case of a Transmit Error event (see **Chapter 5.2.4**). This bit has to be cleared by software.
RE	9	rwh	**Receive Error Flag** 0 No error 1 Reception completed before the receive buffer was read This bit is set in case of a Receive Error event (see **Chapter 5.2.4**). This bit has to be cleared by software.

Field	Bits	Type	Description
PE	10	w	**Phase Error Flag** 0 No error 1 Received data changes around the sampling clock edge This bit is set in case of a Phase Error event (see **Chapter 5.2.4**). This bit has to be cleared by software.
0	15	rwh	**Reserved** Read as 0; have to be written with 0.
0	[7:0], [31:12]	r	**Reserved** Read as 0; should be written with 0.

SSC_TB
SSC Transmit Buffer Register (20$_H$) Reset Value: 0000 0000$_H$

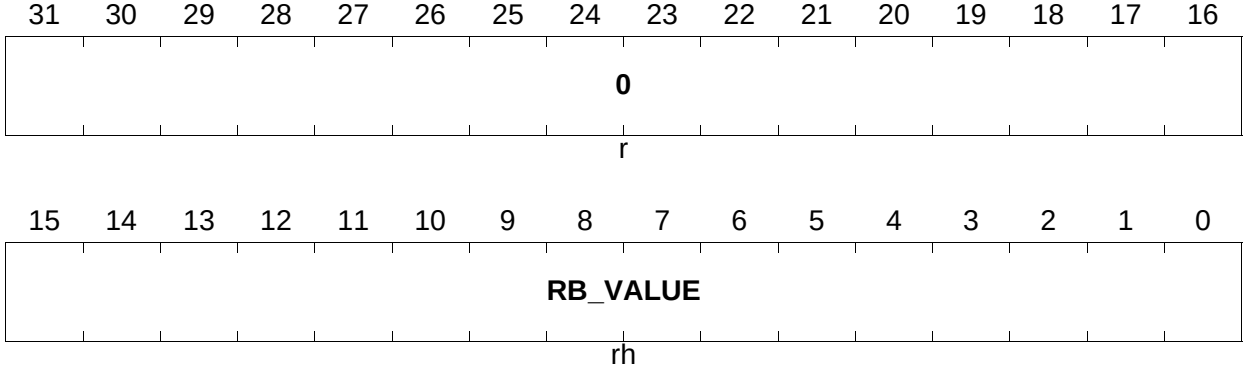


Field	Bits	Type	Description
TB_VALUE	[15:0]	rw	**Transmit Data Register Value** Register SSC_TB stores the data value to be transmitted TB_VALUE.
0	[31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

The Receive Buffer Register RB contains the receive data value.

SSC_RB

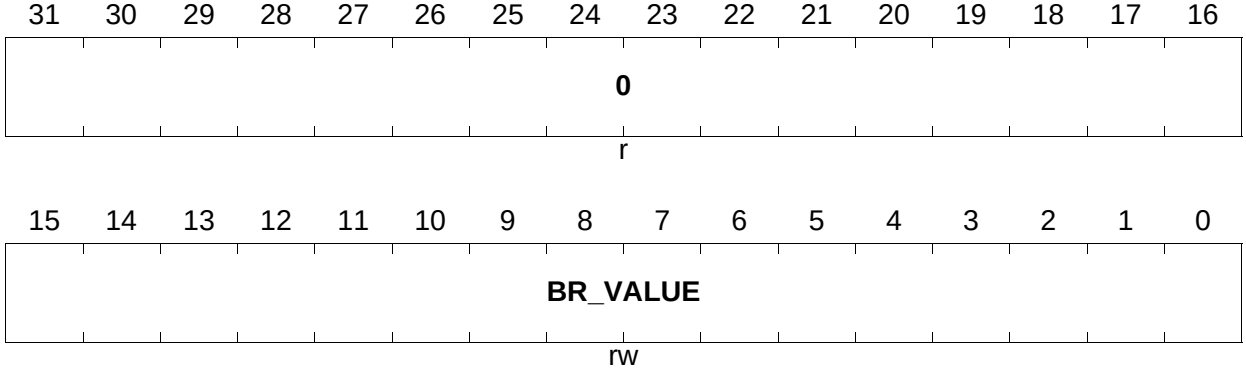
SSC Receive Buffer Register (24_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RB_VALUE	[15:0]	rh	**Receive Data Register Value** Register RB contains the received data value RB_VALUE.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

SSC_BR

SSC Baud Rate Timer Reload Register (14_H) Reset Value: 0000 0063$_H$



Field	Bits	Type	Description
BR_VALUE	[15:0]	rw	**Baud Rate Timer Reload Value** This bit field has to be set to FFFF by the set-up software to ensure a error free communication. The reset value is set in a way that a communication with a bandwidth greater than 400 KBit/s is possible.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

5.4 Port Control

If the SSC was selected as the communication interface (pin MODE was latched after PORST with '1') for the CIC751, the port control registers of port 0 are automatically configured in a way that a communication via SPI is possible.

Port 0 control registers P0_IOCR0, P0_IOCR4, P0_IOCR8, and P0_IOCR12 are initialized with the following values:

- P0_IOCR0 = $A0202020_H$
- P0_IOCR4 = $0020A020_H$
- P0_IOCR8 = 20202020_H
- P0_IOCR12 = 00000020_H

Port pins that can be used for either SSC or MLI communication are automatically configured in way that the MLI part is inactive and does not generate any action that can cause any harm to an SSC communication.

Using the open-drain output feature of port lines helps avoid bus contention problems and reduces the need for hard-wired hand-shaking or slave-select lines. The open-drain output feature can be selected for pin MRST via bit field P0_IOCR0.PC3

5.4.1 Connecting 2 or more CIC751 SSC Slaves to 1 Host

If more than one slave is connected to an SSC master, only one of them may be selected at a time. The master enables one of the connected slaves with the associated slave select output signals. The other output signals of the master (MTSR and SCLK) are broadcast and connected to each slave. The output signals of the slaves (MRST and RDY), which are inputs of the master, must be activated by the selected slave and must be deactivated by all other slaves. In this way, it is possible to directly connect the incoming signals at the master's input terminal. The deactivation of the outputs (MRST and RDY) of the deselected slaves is accomplished by deactivating the output stage on the associated pads. This deactivation takes place, if MODE = 1 and SLS = 1, i.e. SSC is selected and slave is not selected.

Two additional bit fields are defined: PC3B in register P0_IOCR0 for pin P0.3 and PC5B in register P0_IOCR4 for pin P0.5. If the upper condition is true, then the new bit fields, PC3B and PC5B, define the GPIO port behavior; otherwise, the regular PC3 and PC5 define the GPIO port behavior. These new bit fields are programmable and should be programmed to an input-function in this case. The port thus becomes input whenever SLS goes high and the slave is deselected.

6 The Analog/Digital Converter

The CIC751 provides an Analog/Digital Converter with 8-bit or 10-bit resolution and a sample & hold circuit on-chip. An input multiplexer selects from up to 16 analog input channels either via software (Fixed Channel Modes) or automatically (Auto Scan Modes).

To fulfill most requirements of embedded control applications, the ADC supports the following conversion modes:

- **Standard Conversions**
  - **Fixed Channel Single Conversion**
    produces just one result from the selected channel
  - **Fixed Channel Continuous Conversion**
    repeatedly converts the selected channel
  - **Auto Scan Single Conversion**
    produces one result from each of a selected group of channels
  - **Auto Scan Continuous Conversion**
    repeatedly converts the selected group of channels
  - **Wait for Read Mode**
    starts a conversion automatically when the previous result has been read
- **Channel Injection Mode**
  can insert the conversion of a specific channel into a group conversion (auto scan)

A set of SFRs provide access to control functions and results of the ADC. The Enhanced Mode registers provide more detailed control functions for the ADC.

The external analog reference voltages V_{AREF} and V_{AGND} are not programmable. The separate supply for the ADC reduces interference with other digital signals. The reference voltages must be stable during the reset calibration phase and during an entire conversion, to achieve maximum accuracy.

The sample time as well as the conversion time are programmable, so the ADC can be adjusted to the internal resistances of the analog sources and/or the analog reference voltage supply.

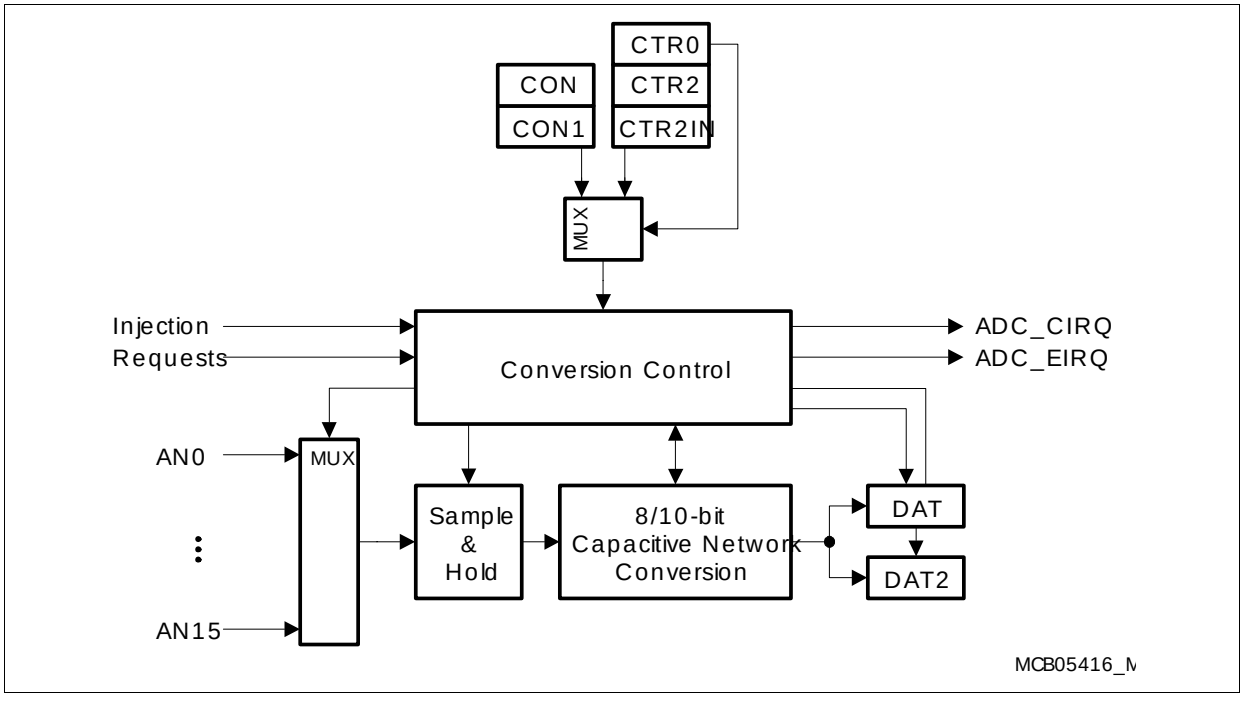


Figure 6-1 Analog/Digital Converter Block Diagram

The ADC is implemented as a capacitive network using successive approximation conversion. A conversion consists of three phases.

- During the sample phase, the capacitive network is connected to the selected analog input and is charged or discharged to the voltage of the analog signal.
- During the actual conversion phase, the network is disconnected from the analog input and is repeatedly charged or discharged via V_{AREF} during the steps of successive approximation.
- After the (optional) post-calibration phase (to adjust the network to changing conditions such as temperature), the result is written to the result register and an interrupt request is generated.

There are two sets of control, data, and status registers: one set for Compatibility Mode and one set for Enhanced Mode. Only one of these register sets may be active at a given time. As most of the bits and bit fields of the registers of the two sets control the same functionality or control the functionality in a very similar way, the following description is organized according to the functionality, not according to the two register sets.

6.1 Mode Selection

It is recommended that the digital input stage should be disabled via register STCU_SYSCON.P1DIDIS if the ADC is used. This avoids undesired cross currents and switching noise when the (analog) input signal level is between V_{IL} and V_{IH}.

The functions of the A/D Converter are controlled by two sets of control registers. In Compatibility Mode, registers ADC_CON and ADC_CON1 are used. In Enhanced Mode, registers ADC_CTR0, ADC_CTR2, and ADC_CTR2IN are used. Their bit fields specify the analog channel to be acted upon, specify the conversion mode, and also reflect the status of the converter.

6.1.1 Compatibility Mode

In Compatibility Mode (ADC_CTR0.MD = 0), registers ADC_CON and ADC_CON1 select the basic functions.

6.1.2 Enhanced Mode

In Enhanced Mode (ADC_CTR0.MD = 1), registers ADC_CTR0, ADC_CTR2, and ADC_CTR2IN select the basic functions. The register layout for Enhanced Mode differs from the Compatibility Mode layout, but this mode provides more options.

Conversion timing is selected via registers ADC_CTR2IN, where ADC_CTR2 controls standard conversions and ADC_CTR2IN controls injected conversions.

6.2 ADC Operation

This section describes the various control mechanisms of the ADC.

6.2.1 Channel Selection

Bit field ADC_CON.ADCH or ADC_CTR0.ADCH controls the input channel multiplexer logic. In the Single Channel Modes, it specifies the analog input channel which is to be converted. In the Auto Scan Modes, it specifies the highest channel number to be converted in the auto scan round.

ADC_CON.ADCH or ADC_CTR0.ADCH may be changed while a conversion is in progress. The new value will go into effect after the current conversion is finished in the Fixed Channel Modes, or after the current conversion round is finished in the Auto Scan Modes.

6.2.2 ADC Status Flags

The ADC busy status flag (ADC_CON.ADBSY or ADC.CTR0.ADBSY) is set when the ADC is started (by setting ADC_CON.ADST or ADC_CTR0.ADST) and remains set as long as the ADC performs conversions or calibration cycles.

ADC_CON.ADBSY and ADC_CTR0.ADBSY are cleared when the ADC is idle, meaning there are no conversion or calibration operations in progress.

Bit ADC_CON1.SAMPLE or ADC_CTR0.SAMPLE is set during the sample phase.

6.2.3 ADC Start/Stop Control

Bit ADC_CON.ADST or ADC_CTR0.ADST is used to start or to stop the ADC. A single conversion or a conversion sequence is started by setting bit ADC_CON.ADST or ADC_CTR0.ADST.

The busy flag ADC_CON.ADBSY or ADC.CTR0.ADBSY will be set and the converter then selects and samples the input channel, which is specified by the channel selection bit field ADC_CON.ADCH or ADC.CTR0.ADCH. The sampled level will then be held internally during the conversion. When the conversion of this channel is complete, the result is transferred into the result register together with the number of the converted channel and the interrupt request is generated. The conversion result is placed into bit field ADC_DAT.ADRES.

ADC_CON.ADST or ADC_CTR0.ADST remains set until cleared either by hardware or by software. Hardware clears the bit dependent on the conversion mode:

- In Fixed Channel Single Conversion Mode, ADC_CON.ADST or ADC_CTR0.ADST is cleared after the conversion of the specified channel is finished.
- In Auto Scan Single Conversion Mode, ADC_CON.ADST or ADC_CTR0.ADST is cleared after the conversion of channel 0 is finished.

Note: In the Continuous Conversion Modes, ADC_CON.ADST or ADC_CTR0.ADST is never cleared by hardware.

Stopping the ADC via software is performed by clearing bit ADC_CON.ADST or bit ADC_CTR0.ADST. The reaction of the ADC depends on the conversion mode:

- In Fixed Channel Single Conversion Mode, the ADC finishes the conversion and then stops. There is no difference to the operation if ADC_CON.ADST or ADC_CTR0.ADST was not cleared by software.
- In Fixed Channel Continuous Conversion Mode, the ADC finishes the current conversion and then stops. This is the usual way to terminate this conversion mode.
- In Auto Scan Single Conversion Mode, the ADC continues the auto scan round until the conversion of channel 0 is finished, then it stops. There is no difference to the operation if ADST was not cleared by software.
- In Auto Scan Continuous Conversion Mode, the ADC continues the auto scan round until the conversion of channel 0 is finished, then it stops. This is the usual way to terminate this conversion mode.

A restart of the ADC can be performed by clearing and then setting bit ADC_CON.ADST or ADC_CTR0.ADST. This sequence will abort the current conversion and restart the ADC with the new parameters given in the control registers.

6.2.4 Conversion Mode Selection

Bit field ADC_CON.ADM or ADC_CTR0.ADM selects the conversion mode of the A/D Converter, as listed in **Table 6-1**.

Table 6-1 A/D Converter Conversion Mode

ADM	Description
00	Fixed Channel Single Conversion Mode
01	Fixed Channel Continuous Conversion Mode
10	Auto Scan Single Conversion Mode
11	Auto Scan Continuous Conversion Mode

While a conversion is in progress, the mode selection bit field ADC_CON.ADM or ADC_CTR0.ADM and the channel selection bit field ADC_CON.ADCH or ADC_CTR0.ADCH may be changed. ADC_CON.ADM or ADC_CTR0.ADM will be evaluated after the current conversion. ADC_CON.ADCH or ADC_CTR0.ADCH will be evaluated after the current conversion (Fixed Channel Modes) or after the current conversion sequence (Auto Scan Modes).

6.2.5 Conversion Resolution Control

The ADC can produce either a 10-bit result (ADC_CON1.RES = 0 or ADC_CTR2.RES = 00_B) or an 8-bit result (ADC_CON1.RES = 1 or ADC_CTR2.RES = 01_B). Depending on the application's requirements, a higher conversion speed (an 8-bit conversion requires less conversion time) or a higher resolution can be chosen.

6.2.5.1 Conversion Result

The result of a conversion is stored in the result register ADC_DAT, or in register ADC_DAT2 for an injected conversion.

The position of the result depends on the basic operating mode (compatibility or enhanced) and on the selected resolution (8-bit or 10-bit).

Note: Bit field CHNR of register ADC_DAT is loaded by the ADC to indicate the channel to which the result refers. Bit field CHNR of register ADC_DAT2 is loaded by software to select the analog channel, which is to be injected.

6.2.6 Fixed Channel Conversion Modes

These modes are selected by programming the mode selection bit field ADC_CON.ADM or ADC_CTR0.ADM to 00_B (single conversion) or to 01_B (continuous conversion). After starting the converter through setting bit ADC_CON.ADST or ADC_CTR0.ADST the busy flag ADC_CON.ADBSY or ADC_CTR0.ADBSY will be set and the channel specified in bit field ADC_CON.ADCH or ADC_CTR0.ADCH will be converted. After the conversion is complete, an interrupt request trigger (ADC event 0) is generated that can be used to trigger the DMA.

In Single Conversion Mode the converter will automatically stop and clears bits ADC_CON.ADBSY or ADC_CTR0.ADBSY and ADC_CON.ADST or ADC_CTR0.ADST.

In Continuous Conversion Mode the converter will automatically start a new conversion of the channel specified in bit field ADC_CON.ADCH or ADC_CTR0.ADCH. An interrupt request trigger (ADC event 0) is generated that can be used to trigger the DMA after each completed conversion.

When bit ADC_CON.ADST or ADC_CTR0.ADST is cleared by software, while a conversion is in progress, the converter will complete the current conversion and then stop and clear bit ADC_CON.ADBSY or ADC_CTR0.ADBSY.

6.2.7 Auto Scan Conversion Modes

These modes are selected by programming the mode selection bit field ADC_CON.ADM or ADC_CTR0.ADM to 10_B (single conversion) or to 11_B (continuous conversion). Auto Scan Modes automatically convert a sequence of analog channels, beginning with the

channel specified in bit field ADC_CON.ADCH or ADC_CTR0.ADCH and ending with channel 0, without requiring software to change the channel number.
After starting the converter through bit ADC_CON.ADST or ADC_CTR0.ADST, the busy flag ADC_CON.ADBSY or ADC_CTR0.ADBSY will be set and the channel specified in bit field ADC_CON.ADCH or ADC_CTR0.ADCH will be converted. After the conversion is completed, an interrupt request trigger (ADC event 0) is generated that can be used to trigger the DMA and the converter will automatically start a new conversion of the next lower channel. After each completed conversion an interrupt trigger is generated. After conversion of channel 0, the current sequence is complete.

In Single Conversion Mode, the converter will automatically stop and clear bits ADC_CON.ADBSY or ADC_CTR0.ADBSY and ADC_CON.ADST or ADC_CTR0.ADST.

In Continuous Conversion Mode, the converter will automatically start a new sequence beginning with the conversion of the channel specified in ADC_CON.ADCH or ADC_CTR0.ADCH.

When bit ADC_CON.ADST or ADC_CTR0.ADST is cleared by software while a conversion is in progress, the converter will complete the current sequence (including conversion of channel 0) and then stop and clear bit ADC_CON.ADBSY or ADC_CTR0.ADBSY.

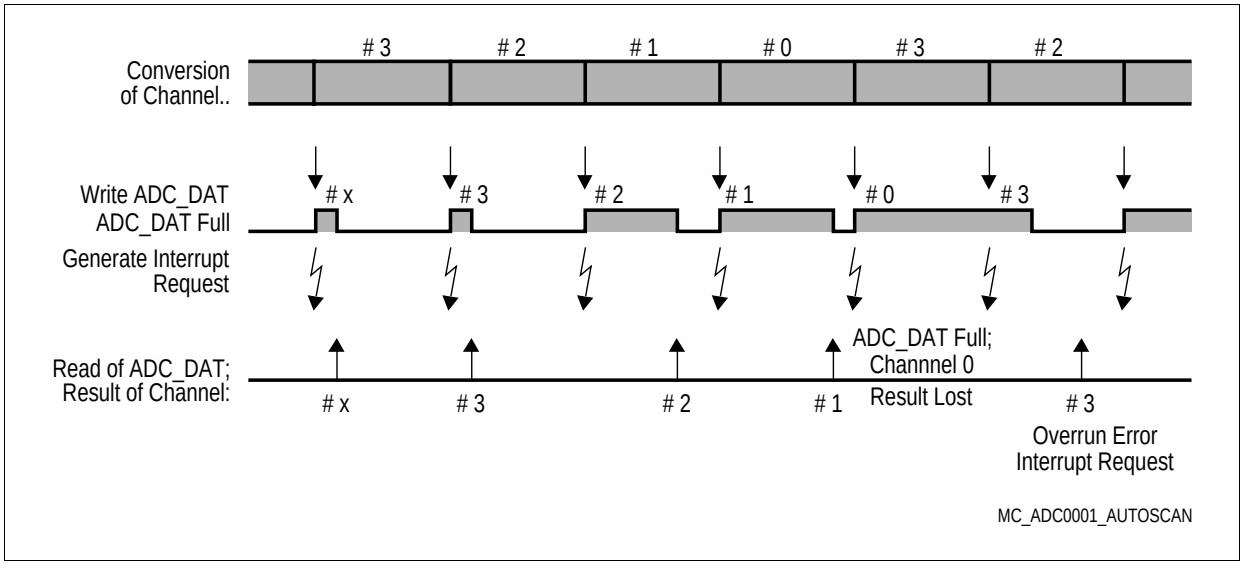


Figure 6-2 Auto Scan Conversion Mode Example

6.2.8 Wait for Read Mode

If a previous conversion result has not been read out of the result register by the time a new conversion is complete, the previous result is lost because it is overwritten by the new value, and the error/injection interrupt request trigger is generated.

In order to avoid error/injection interrupts and the loss of conversion results especially when using Continuous Conversion Modes, the ADC can be switched to “Wait for Read Mode” by setting bit ADC_CON.ADWR or ADC_CTR0.ADWR.

If the result value has not been read by the time the current conversion is completed, the new result is stored in a temporary buffer and the next conversion is suspended (ADST and ADC_CON.ADBSY or ADC_CTR0.ADBSY will remain set in the meantime, but no interrupt will be generated). After reading the previous value, the temporary buffer is copied into ADC_DAT (generating an interrupt) and the suspended conversion is started. This mechanism applies to both single and Continuous Conversion Modes.

Note: In Standard Mode, continuous conversions are executed at a fixed rate (determined by the conversion time), but, in “Wait for Read Mode” there may be delays due to suspended conversions.

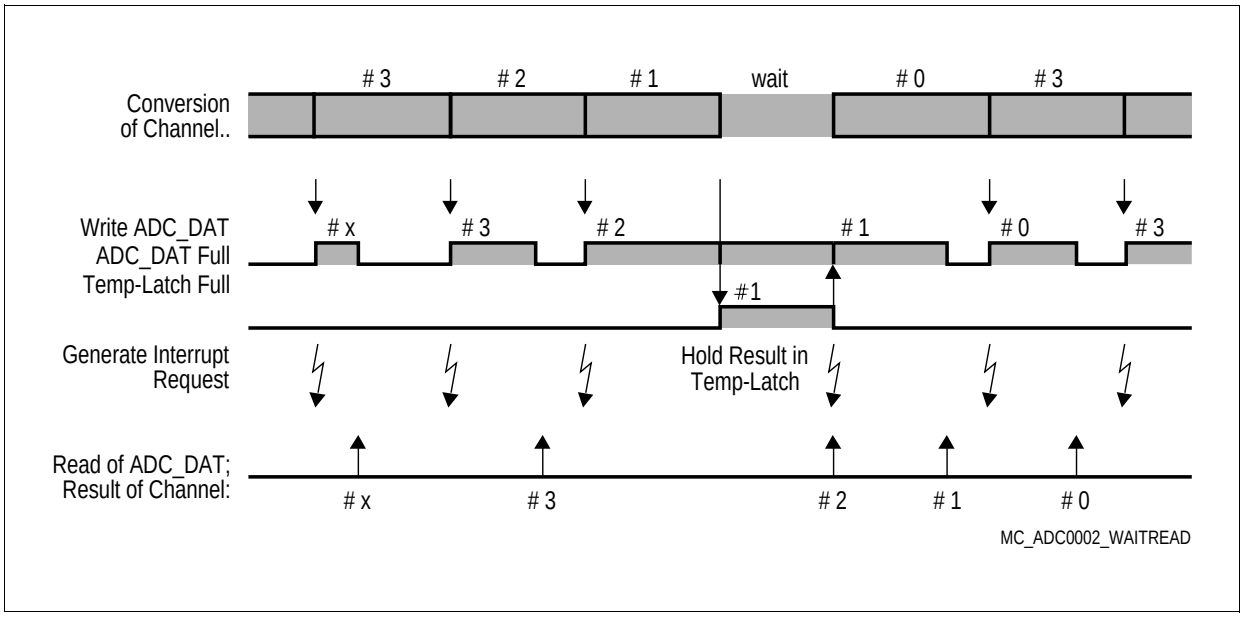


Figure 6-3 Wait for Read Mode Example

6.2.9 Channel Injection Mode

Channel Injection Mode allows the conversion of a specific analog channel (also while the ADC is running in a Continuous or Auto Scan Mode) without changing the current operating mode. After the conversion of this specific channel, the ADC continues with the original operating mode.

Channel Injection Mode is enabled by setting bit ADC_CON.ADCIN or ADC_CTR0.ADCIN and requires the Wait for Read Mode (ADC_CON.ADWR = 1 or ADC_CTR0.ADWR = 1). The channel to be converted in this mode is specified in bit field CHNR of register ADC_DAT2.

Note: Since the channel number for an injected conversion is not buffered, bit field CHNR of ADC_DAT2 must never be modified during the sample phase of an injected conversion, otherwise the input multiplexer will switch to the new channel. It is recommended to change the channel number only when no injected conversion is running.

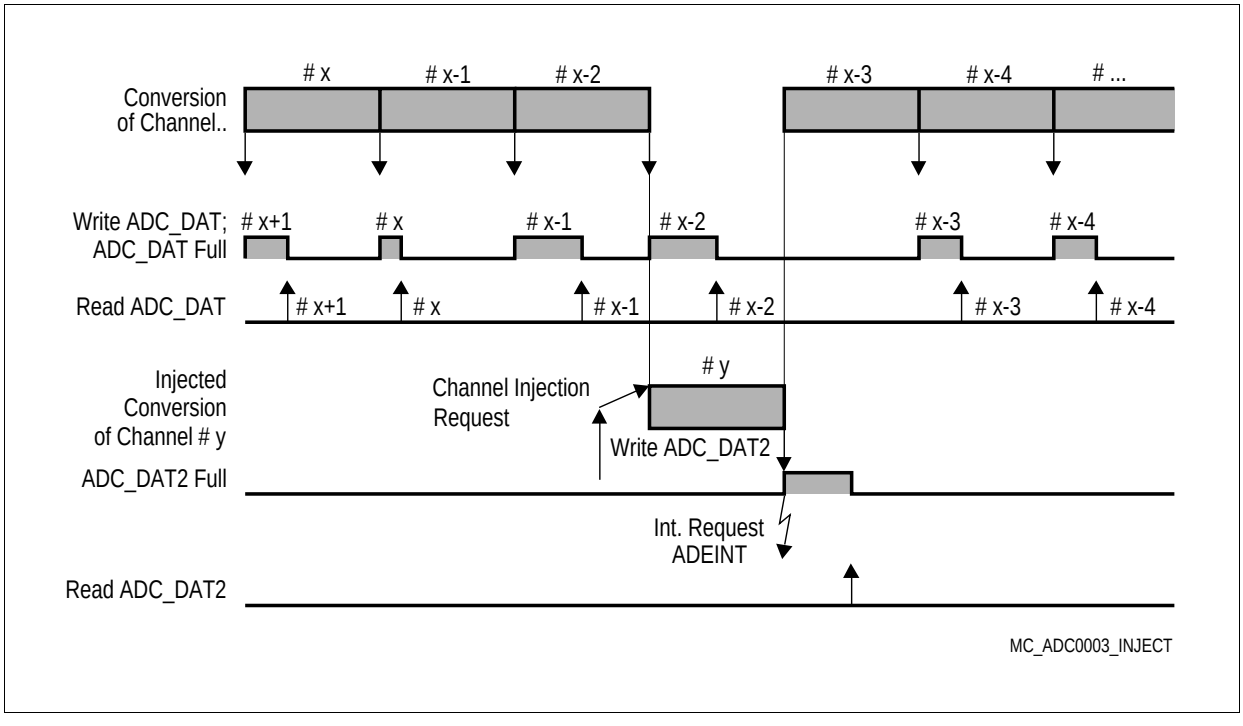


Figure 6-4 Channel Injection Example

A Channel Injection can be triggered in the following way:

- setting of the Channel Injection Request bit ADC_CON.ADCRQ or ADC_CTR0.ADCRQ via software

Note: The channel injection request bit ADC_CON.ADCRQ or ADC_CTR0.ADCRQ will be set regardless of whether or not the Channel Injection Mode is enabled. It is recommended to always clear bit ADC_CON.ADCRQ or ADC_CTR0.ADCRQ before enabling the Channel Injection Mode.

After the completion of the current conversion (if any is in progress), the converter will start (inject) the conversion of the specified channel. When the conversion of this channel is completed, the result will be placed into the alternate result register ADC_DAT2, and a Channel Injection Complete interrupt request trigger (ADC event 1) is generated that can be used to trigger the DMA.

Note: The result of an injected conversion is directly written to ADC_DAT2. If the previous result has not been read in the meantime, it is overwritten.

6.2.10 Arbitration of Conversions

Conversion requests that are activated while the ADC is idle immediately trigger the respective conversion. If a conversion is requested while another conversion is currently in progress, the operation of the A/D converter depends on the type of conversions involved (standard or injected).

Note: A conversion request is activated if the respective control bit (ADC_CON.ADST / ADC:CTR0.ADST or ADC:CON.ADCRQ / ADC_CTR0.ADCRQ) is toggled from 0 to 1, i.e. the bit must have been zero before being set.

Table 6-2 summarizes the ADC operation in the possible situations.

Table 6-2 Conversion Arbitration

Conversion in Progress	New Requested Conversion	
	Standard	**Injected**
Standard	Abort running conversion, and start requested new conversion.[1)]	Complete running conversion, start requested conversion after that.
Injected	Complete running conversion, start requested conversion after that.	Complete running conversion, start requested conversion after that. Bit ADC_CON.ADCRQ or ADC_CTR0.ADCRQ will be 0 for the second conversion, however.

1) If an injected conversion is pending when a Standard Conversion is re-started, the injected conversion is executed before the newly started Standard Conversion.

6.3 Automatic Calibration

The ADC of the CIC751 features automatic self-calibration. This calibration corrects gain errors, which are mainly due to process variation, and offset errors, which are mainly due to temperature changes.

Two types of calibration are supported:

- Reset calibration performs a thorough basic calibration of the ADC after a power-on reset.
- Post-calibration performs one small calibration step after each conversion.

Reset Calibration

After a reset, a thorough power-up calibration is performed automatically to correct gain and offset errors of the A/D converter. To achieve the best calibration results, the reference voltages as well as the supply voltages must be stable during the power-up calibration. During the calibration sequence, a series of calibration cycles is executed, where the step width for adjustments is reduced gradually. The total number of executed calibration cycles depends on the actual properties of the respective ADC module. The maximum duration of the power-up calibration is 11,696 cycles of the basic clock f_{BC}.

Status flag ADC_CON1.CAL is set as long as this power-up calibration takes place.

Post-Calibration

After each conversion, a small calibration step can be executed. For 8-bit and 10-bit conversions, post-calibration is not mandatory in order not to exceed the total unadjusted error (TUE) specified in the data sheet. Post-calibration can be disabled by setting bit CALOFF in register ADC_CTR0. When disabled, the post-calibration cycles are skipped, which reduces the total conversion time.

Note: Calibration may be disabled only after the reset calibration is completed.

6.4 Multiplexer Test Mode

For analog channel 0, a Multiplexer Test Mode (MTM) is also available. This function is controlled via bit field SCU_SYSCON.MTM. This feature is independent of the currently selected conversion mode. If the MTM is enabled, the analog input is connected to ADC ground via an internal resistor. This structure creates a voltage divider to ground. Therefore, conversion delivers a smaller result if MTM is enabled.

6.5 Conversion Timing Control

When a conversion is started, first the capacitances of the converter are loaded via the respective analog input pin to the current analog input voltage. The time to load the capacitances is referred to as sample time. Next, the sampled voltage is converted to a digital value in successive steps, which correspond to the resolution of the ADC. During these phases (except for the sample time), the internal capacitances are repeatedly charged and discharged via pins V_{AREF} and V_{AGND}.

The current that must be drawn from the sources for sampling and changing charges depends on the time required for each respective step, because the capacitors must reach their final voltage level within the given time, at least with a certain approximation. The maximum current, however, that a source can deliver, depends on its internal resistance.

The time that the two different actions during conversion take (sampling, and converting) can be programmed within a certain range in the CIC751 relative to the system clock. The absolute time that is consumed by the different conversion steps therefore is independent from the general speed of the device. This allows the A/D converter of the CIC751 to be adjusted to the properties of the system:

Fast Conversion can be achieved by programming the respective times to their absolute possible minimum. This is preferable for scanning high frequency signals. The internal resistance of analog source and analog supply must, however, be sufficiently low.

High Internal Resistance can be achieved by programming the respective times to a higher value, or to the possible maximum. This is preferable when using analog sources and supply with a high internal resistance in order to keep the current as low as possible. The conversion rate in this case may, however, be considerably lower.

Control Bit Fields

Two mechanisms are provided for timing control of the conversion and the sample phase:

- **Standard timing control** uses two 2-bit fields in register ADC_CON to select prescaler values for the general conversion timing and the duration of the sample phase. This provides compact control, while limiting the prescaler factors to a few steps.
- **Improved timing control** uses two 6-bit fields in register ADC_CON1 (Compatibility Mode) or register ADC_CTR2/ADC_CTR2IN (Enhanced Mode). This provides a wide range of prescaler factors, so the ADC can be better adjusted to the internal and external system circumstances.

Improved timing control is selected by setting bit ICST in register ADC_CON1 in Compatibility Mode, or by selecting Enhanced Mode.

Note: The conversion clock f_{BC} must not exceed 20 MHz.

Standard Timing Control

Standard timing control is performed by using two 2-bit fields in register ADC_CON. Bit field ADCTC (conversion time control) selects the basic conversion clock (f_{BC}), used for the operation of the A/D Converter. The sample time is derived from this conversion clock and controlled by bit field ADSTC. The sample time is always a multiple of 8 f_{BC} periods. **Table 6-3** lists the possible combinations.

Table 6-3 Standard Conversion and Sample Timing Control

ADC_CON.ADCTC	A/D Converter Basic Clock f_{BC}	ADC_CON.ADSTC	Sample Time t_S
00_B	$f_{ADC}/4$	00_B	$t_{BC} \times 8$
01_B	$f_{ADC}/2$	01_B	$t_{BC} \times 16$
10_B	$f_{ADC}/16$	10_B	$t_{BC} \times 32$
11_B	$f_{ADC}/8$	11_B	$t_{BC} \times 64$

Improved Timing Control

To provide a finer resolution for programming of the timing parameters, wider bit fields have been implemented for timing control (the 2-bit bit fields in register ADC_CON are disregarded in all cases).

In Compatibility Mode (with bit ADC_CON1.ICST = 1), the bit fields in register ADC_CON1 are used for all conversions.
In Enhanced Mode (bit ADC_CTR0.MD = 1), the bit fields in register ADC_CTR2 are used for Standard Conversions. Injected conversions use the bit fields in register ADC_CTR2IN.

Bit field ADCTC (conversion time control) selects the basic conversion clock (f_{BC}), used for the operation of the A/D Converter. The sample time is derived from this conversion clock and is controlled by bit field ADC_CTR2.ADSTC. The sample time is always a multiple of 4 f_{BC} periods. **Table 6-4** lists the possible combinations.

Table 6-4 Improved Conversion and Sample Timing Control

ADCTC	A/D Converter Basic Clock f_{BC} [1)]	ADSTC	Sample Time t_S
$00\ 0000_B = 00_H$	$f_{SYS}/1$	$00\ 0000_B = 00_H$	$t_{BC} \times 8$
$00\ 0001_B = 01_H$	$f_{SYS}/2$	$00\ 0001_B = 01_H$	$t_{BC} \times 12$
$00\ 0010_B = 02_H$	$f_{SYS}/3$	$00\ 0010_B = 02_H$	$t_{BC} \times 16$
…	$f_{SYS}/(ADCTC + 1)$	…	$t_{BC} \times 4 \times (ADSTC + 2)$
$11\ 1111_B = 3F_H$	$f_{SYS}/64$	$11\ 1111_B = 3F_H$	$t_{BC} \times 260$

1) The limit values for f_{BC} (see data sheet) must not be exceeded when selecting ADC_CTR2.ADCTC and f_{SYS}.

Total Conversion Time Examples

The time for a complete conversion includes the sample time t_S, the conversion itself (successive approximation and calibration), and the time required to transfer the digital value to the result register as shown in the example below (Standard Conversion timing).

The timings refer to module clock cycles, where $t_{SYS} = 1/f_{SYS}$.

- Assumptions: f_{SYS} = 40 MHz (i.e. t_{SYS} = 25 ns), ADC_CON.ADCTC = 01_B, ADC_CON.ADSTC = 00_B
- Basic clock: $f_{BC} = f_{SYS}/2$ = 20 MHz, i.e. t_{BC} = 50 ns
- Sample time: $t_S = t_{BC} \times 8$ = 400 ns

Conversion 10-bit:

- With post-calibr.: $t_{C10P} = t_S + 52 \times t_{BC} + 6 \times t_{SYS}$ = (2600 + 400 + 150) ns = 3.15 µs
- Post-calibr. off: $t_{C10} = t_S + 40 \times t_{BC} + 6 \times t_{SYS}$ = (2000 + 400 + 150) ns = 2.55 µs

Conversion 8-bit:

- With post-calibr.: $t_{C8P} = t_S + 44 \times t_{BC} + 6 \times t_{SYS}$ = (2200 + 400 + 150) ns = 2.75 µs
- Post-calibr. off: $t_{C8} = t_S + 32 \times t_{BC} + 6 \times t_{SYS}$ = (1600 + 400 + 150) ns = 2.15 µs

Note: For the exact specification, refer to the data sheet of the selected derivative.

6.6 A/D Converter Interrupt Operation

The ADC offers different interrupts request triggers that can occur due to different cases:

- End-of-conversion interrupt (ADC event 0)
  – the result of a conversion is placed into register ADC_DAT
- Error/injection interrupt (ADC event 1)
  – a conversion result overwrites a previous value in register ADC_DAT (error interrupt in standard mode)
  – the result of an injected conversion has been stored into ADC_DAT2 (end-of-injected-conversion interrupt)
- ADC event 2; the OR-combination of all valid bits of the ADC_RESBn registers

6.6.1 Interrupt Event Handling

Interrupt events can be handled in three different ways:

- Trigger an DMA action
- Forward the interrupt to pin SRn

6.6.1.1 Trigger an DMA Action

Both ADC interrupts can be used to trigger a DMA transfer. This mechanism can be used to store the conversion result within a extended result register.

*Note: For more information about triggering a DMA transfer, see **Chapter 3.1**.*

6.6.1.2 Forward to an SRn Pin

An ADC interrupt can be forwarded to a host controller via an SRn pin of the CIC751.

The following events can be selected as the source for an output of an SRn pin:

- An end-of-conversion interrupt was triggered
- An error/injection interrupt was triggered

*Note: For information about routing an interrupt to an SRn pin, see **Chapter 2.5.3.1**.*

6.7 ADC Buffer Registers

This section describes the extended result register, including the control and input register and the doorbell mechanism.

6.7.1 Overview

The ADC module offers the possibility of storing two results in its data registers. The result of the injected conversion is stored in register DAT2, whereas all other results (auto scan or single programmed conversions) are stored in register DAT.
In order to make all results available at the same time, additional result registers are added. As the ADC has 16 analog input channels, the same number of result registers is necessary to store the results of each single channel.
The data transfer between the standard ADC result registers and the extended result registers is performed by an interrupt request to the DMA.

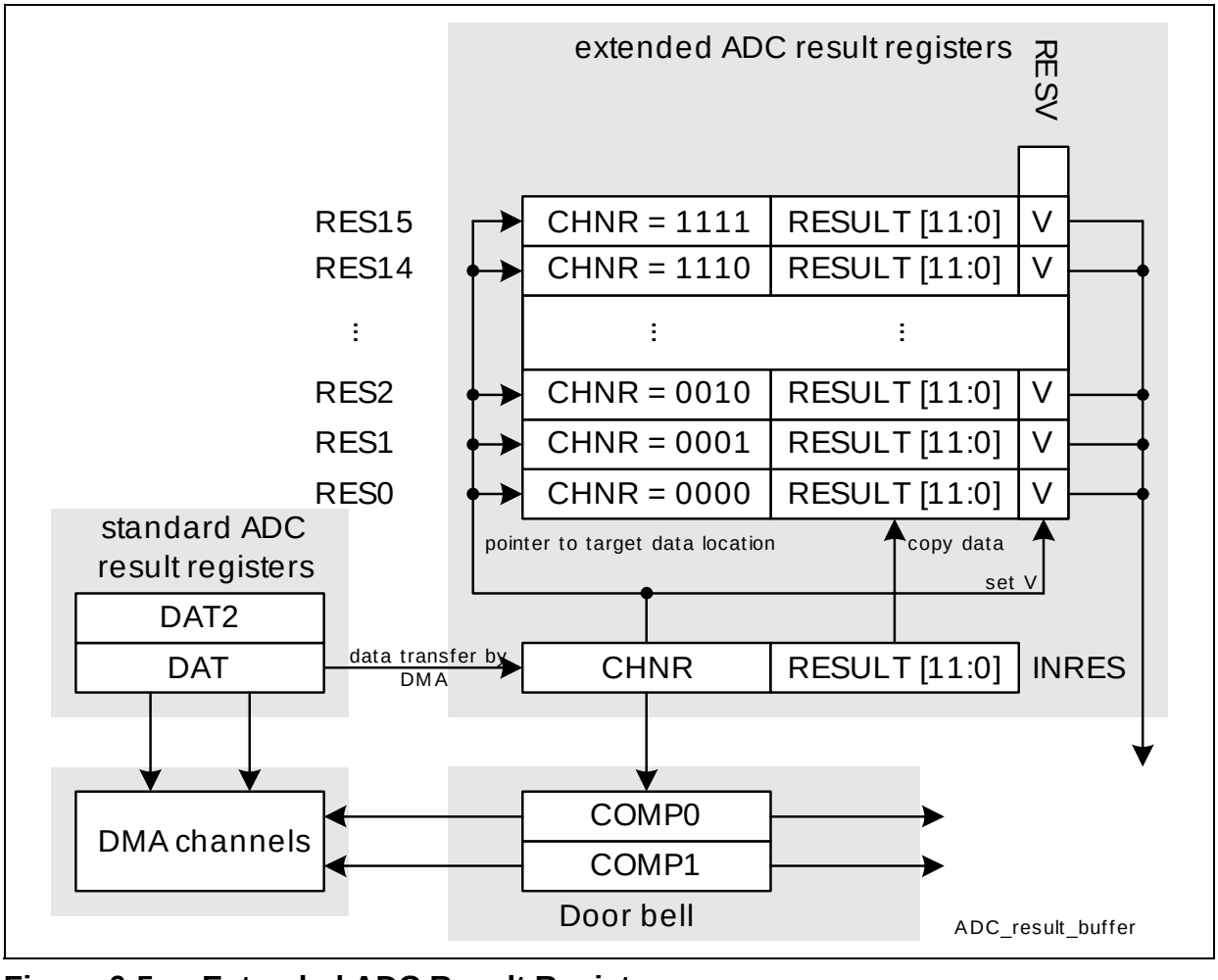


Figure 6-5 Extended ADC Result Registers

Each of the 16 extended result registers contains a RESULT bit field with the conversion result delivered by the ADC. The ADC extended result registers also indicate the channel number belonging to the result.

6.7.2 Extended Result Registers

After being triggered, the DMA reads a 16-bit data word from the register DAT in the ADC and writes it to the input result register INRES. A 16-bit data word written to this register is transferred automatically to one of the 16 result registers. The upper 4 bits written to INRES indicate the number of the target result register.

Each result register has an associated valid bit RESBn.V. The valid bit of a result register is automatically set if data is transferred to it. The valid bit is automatically cleared if the result register is read out. As a result, the valid bit indicates that new data is available, that has not yet been read out.

In order to allow different read modes, two different 16-bit read views exist, selected by two different read address (one view (view A) shows the same bit positions as the original ADC register, the other view (view B) shows the valid bit instead of the channel number) so the user can do polling to check for new data in view B. This view B can be accessed when reading the second set of addresses.

The input register for the 16 result registers is INRES. Any write access to this address will lead to an update of the corresponding result register. The bit positions [15:12] of the written data are used as pointer to indicate the targeted result register.

All 16 valid bits of the different result registers RESBn are additionally accessible by a single status register RESV.This enable the host to verify easily which result register has valid date and which not.

6.7.3 Doorbell Mechanism

The doorbell mechanism offers a monitoring system for the extended result registers. Two doorbell channels are implemented, each with an individual sensitivity level. The sensitivity level can be configured to monitor one of the 16 extended result registers via ADC_DBCTR.COMP0 or ADC_DBCTR.COMP0.

The doorbell mechanism can be used to trigger either a DMA transfer or to stimulate an SRn pin.

6.7.3.1 Trigger an DMA Transfer

If a extended result register that is monitored by a doorbell channel is updated via register ADC_INRES, the doorbell channel generates a trigger that can be used to request an DMA transfer.

Note: Please note that writing to register ADC:INRES updates both views of a extended result register.

*Note: For more information about the control of DMA transfer triggers, see **Chapter 3.1**.*

This mechanism can be used to create a sensitivity level for the start of the block ADC conversion result data download to the host controller for Auto Scan conversions.

6.7.3.2 Stimulate SRn Pins

The status valid bit of the extended result register that is monitored by a doorbell channel can be forwarded to an SRn pin. This offers an additional opportunity for supervision of the extended result registers.

*Note: For information about routing a valid bit to an SRn pin, see **Chapter 2.5.3.1**.*

6.8 ADC Registers

Table 6-5 summarizes all ADC registers.

The base address of the ADC is 0000 1000. A register address is computed by adding the base address to the register offset address.

Table 6-5 Registers Overview

Register Short Name	Register Long Name	Offset Address	Page Number
ADC_CON	ADC Control Register	10_H	Page 6-21
ADC_CON1	ADC Control 1 Register	12_H	Page 6-23
ADC_CTR0	ADC Control 0 Register	24_H	Page 6-24
ADC_CTR2	ADC Control 2 Register	20_H	Page 6-25
ADC_ CTR2IN	ADC Injection Control 2 Register	22_H	Page 6-26
ADC_DAT	ADC Result Register	30_H	Page 6-27
ADC_DAT2	ADC Result 2 Register	32_H	Page 6-27
ADC_ RESA0	ADC Extended Result 0 View A Register	100_H	Page 6-30
ADC_ RESA1	ADC Extended Result 1 View A Register	104_H	Page 6-30
ADC_ RESA2	ADC Extended Result 2 View A Register	108_H	Page 6-30
ADC_ RESA3	ADC Extended Result 3 View A Register	$10C_H$	Page 6-30
ADC_ RESA4	ADC Extended Result 4 View A Register	110_H	Page 6-30
ADC_ RESA5	ADC Extended Result 5 View A Register	114_H	Page 6-30
ADC_ RESA6	ADC Extended Result 6 View A Register	118_H	Page 6-30
ADC_ RESA7	ADC Extended Result 7 View A Register	$11C_H$	Page 6-30
ADC_ RESA8	ADC Extended Result 8 View A Register	120_H	Page 6-30
ADC_ RESA9	ADC Extended Result 9 View A Register	124_H	Page 6-30

Table 6-5 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Page Number
ADC_ RESA10	ADC Extended Result 10 View A Register	128_H	Page 6-30
ADC_ RESA11	ADC Extended Result 11 View A Register	$12C_H$	Page 6-30
ADC_ RESA12	ADC Extended Result 12 View A Register	130_H	Page 6-30
ADC_ RESA13	ADC Extended Result 13 View A Register	134_H	Page 6-30
ADC_ RESA14	ADC Extended Result 14 View A Register	138_H	Page 6-30
ADC_ RESA15	ADC Extended Result 15 View A Register	$13C_H$	Page 6-30
ADC_ RESB0	ADC Extended Result 0 View B Register	140_H	Page 6-32
ADC_ RESB1	ADC Extended Result 1 View B Register	144_H	Page 6-32
ADC_ RESB2	ADC Extended Result 2 View B Register	148_H	Page 6-32
ADC_ RESB3	ADC Extended Result 3 View B Register	$14C_H$	Page 6-32
ADC_ RESB4	ADC Extended Result 4 View B Register	150_H	Page 6-32
ADC_ RESB5	ADC Extended Result 5 View B Register	154_H	Page 6-32
ADC_ RESB6	ADC Extended Result 6 View B Register	158_H	Page 6-32
ADC_ RESB7	ADC Extended Result 7 View B Register	$15C_H$	Page 6-32
ADC_ RESB8	ADC Extended Result 8 View B Register	160_H	Page 6-32
ADC_ RESB9	ADC Extended Result 9 View B Register	164_H	Page 6-32
ADC_ RESB10	ADC Extended Result 10 View B Register	168_H	Page 6-32

Table 6-5 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Page Number
ADC_ RESB11	ADC Extended Result 11 View B Register	$16C_H$	Page 6-32
ADC_ RESB12	ADC Extended Result 12 View B Register	170_H	Page 6-32
ADC_ RESB13	ADC Extended Result 13 View B Register	174_H	Page 6-32
ADC_ RESB14	ADC Extended Result 14 View B Register	178_H	Page 6-32
ADC_ RESB15	ADC Extended Result 15 View B Register	$17C_H$	Page 6-32
ADC_ INRES	ADC Input Result Register	180_H	Page 6-33
ADC_ RESV	ADC Result Valid Register	188_H	Page 6-34
ADC_ DBCTR	ADC Doorbell Control Register	184_H	Page 6-35

6.8.1 ADC Control Registers for Compatibility Mode

The following registers are used in the Compatibility Mode to configure the ADC module.

ADC_CON
ADC Control Register **(010_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCTC		ADSTC		AD CRQ	AD CIN	AD WR	AD BSY	AD ST	0	ADM		ADCH			
rw		rw		rwh	rw	rw	rh	rwh	r	rw		rw			

Field	Bits	Type	Description
ADCH	[3:0]	rw	**ADC Analog Channel Input Selection** Selects the (first) ADC channel which is to be converted.

Field	Bits	Type	Description
ADM	[5:4]	rw	**ADC Mode Selection** 00 Fixed Channel Single Conversion 01 Fixed Channel Continuous Conversion 10 Auto Scan Single Conversion 11 Auto Scan Continuous Conversion
ADST	7	rwh	**ADC Start Bit** 0 Stop a running conversion 1 Start conversion(s)
ADBSY	8	rh	**ADC Busy Flag** 0 ADC is idle 1 A conversion is active
ADWR	9	rw	**ADC Wait for Read Control** 0 Wait for Read Mode is deactivated 1 Wait for Read Mode is activated
ADCIN	10	rw	**ADC Channel Injection Enable** 0 Channel Injection is disabled 1 Channel Injection is enabled
ADCRQ	11	rwh	**ADC Channel Injection Request Flag** 0 No Channel Injection request is pending 1 A Channel Injection request is pending This bit is automatically cleared if a Channel Injection conversion is started.
ADSTC	[13:12]	rw	**ADC Sample Time Control** (Defines the ADC sample time in a certain range) 00 $t_{BC} \times 8$ 01 $t_{BC} \times 16$ 10 $t_{BC} \times 32$ 11 $t_{BC} \times 64$
ADCTC	[15:14]	rw	**ADC Conversion Time Control** (Defines the ADC basic conversion clock f_{BC}) 00 $f_{BC} = f_{SYS}/4$ 01 $f_{BC} = f_{SYS}/2$ 10 $f_{BC} = f_{SYS}/16$ 11 $f_{BC} = f_{SYS}/8$
0	6	r	**Reserved;** Read as 0; should be written with 0.

ADC_CON1
ADC Control 1 Register **(012_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICST	SAMPLE	CAL	RES	ADCTC						ADSTC					
rw	rh	rh	rw	rw						rw					

Field	Bits	Type	Description
ADSTC	[5:0]	rw	**ADC Sample Time Control** Defines the ADC sample time: $t_S = t_{BC} \times 4 \times$ (<ADSTC> + 1)
ADCTC	[11:6]	rw	**ADC Conversion Time Control** Defines the ADC basic conversion clock: $f_{BC} = f_{SYS}$ / (<ADCTC> + 1)
RES	12	rw	**Conversion Resolution Control** 0 10-bit resolution (default after reset) 1 8-bit resolution
CAL	13	rh	**Reset Calibration Phase Status Flag** 0 A/D Converter is not in calibration phase 1 A/D Converter is in calibration phase
SAMPLE	14	rh	**Sample Phase Status Flag** 0 A/D Converter is not in sampling 1 A/D Converter is currently in the sample phase
ICST	15	rw	**Improved Conversion and Sample Timing** Selects the active timing control bit fields 0 Standard Conversion and sample time control, controlled by the two bit fields ADC_CON.ADCH and ADC_CON.ADM 1 Improved conversion and sample time control, controlled by the two bit fields ADC_CON1.ADSTC and ADC_CON1.ADCTC

Note: The limit values for f_{BC} (see data sheet) must not be exceeded when selecting ADCTC and f_{SYS}.

6.8.2 ADC Control Registers for Enhanced Mode

The following registers are used in the Enhanced Mode to configure the ADC module.

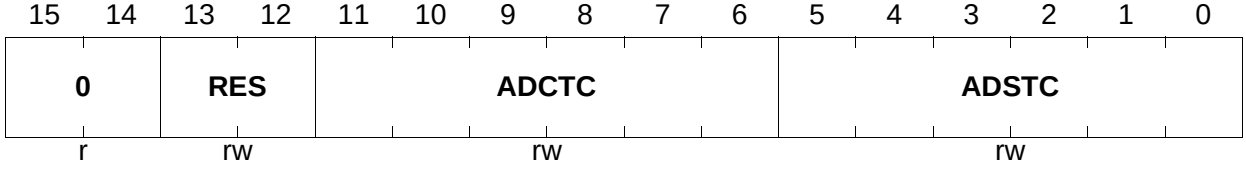
ADC_CTR0
ADC Control 0 Register **(024_H)** **Reset Value: 1000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MD	SAMPLE	ADCTS		ADCRQ	ADCIN	ADWR	ADBSY	ADST	ADM		CALOFF	ADCH			
rw	rh	rw		rwh	rw	rw	rh	rwh	rw		rw	rw			

Field	Bits	Type	Description
ADCH	[3:0]	rw	**Analog Input Channel Selection** Selects the (first) ADC channel which is to be converted
CALOFF	4	rw	**Calibration Disable Control** 0 Calibration cycles are executed 1 Calibration is disabled (off) *Note: This control bit is active in both compatibility and Enhanced Mode.*
ADM	[6:5]	rw	**Mode Selection Control** 00 Fixed Channel Single Conversion 01 Fixed Channel Continuous Conversion 10 Auto Scan Single Conversion 11 Auto Scan Continuous Conversion
ADST	7	rwh	**ADC Start/Stop Control** 0 Stop a running conversion 1 Start conversion(s)
ADBSY	8	rh	**Busy Flag** 0 ADC is idle 1 A conversion is active
ADWR	9	rw	**ADC Wait for Read Control** 0 Wait for Read Mode is deactivated 1 Wait for Read Mode is activated
ADCIN	10	rw	**ADC Channel Injection Enable** 0 Channel Injection is disabled 1 Channel Injection is enabled

Field	Bits	Type	Description
ADCRQ	11	rwh	**ADC Channel Injection Request Flag** 0 No Channel Injection request is pending 1 A Channel Injection request is pending This bit is automatically cleared if a Channel Injection conversion is started.
ADCTS	[13:12]	rw	**Channel Injection Trigger Input Select** 00 Channel injection trigger input disabled 01 Trigger input CAPCOM2 selected 10 Trigger input CAPCOM6 selected 11 Reserved *Note: Reset value of bit field ADCTS is 01_B for compatibility purposes.*
SAMPLE	14	rh	**Sample Phase Status Flag** 0 A/D Converter is not in sample phase 1 A/D Converter in sample phase
MD	15	rw	**Mode Control** 0 Compatibility Mode 1 Enhanced Mode *Note: Any modification of control bit MD is forbidden while a conversion is currently running. User software must take care.*

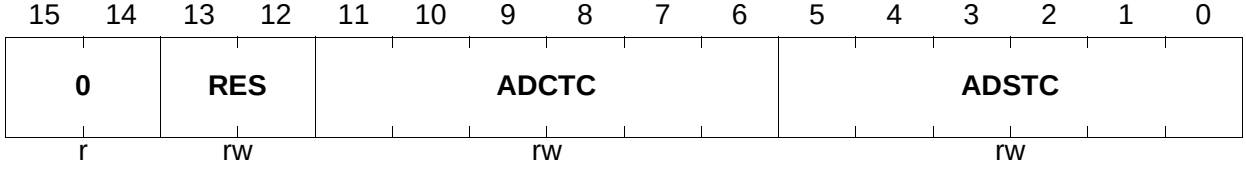
ADC_CTR2
ADC Control 2 Register **(020_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
ADSTC	[5:0]	rw	**ADC Sample Time Control** Defines the ADC sample time: $t_S = t_{BC} \times 4 \times (<ADSTC> + 1)$
ADCTC	[11:6]	rw	**ADC Conversion Time Control** Defines the ADC basic conversion clock: $f_{BC} = f_{SYS} / (<ADCTC> + 1)$

Field	Bits	Type	Description
RES	[13:12]	rw	**Converter Resolution Control** 00 10-bit resolution 01 8-bit resolution 1x Reserved
0	[15:14]	r	**Reserved;** Read as 0; should be written with 0.

ADC_CTR2IN
ADC Injection Control 2 Register (022$_H$) Reset Value: 0000$_H$



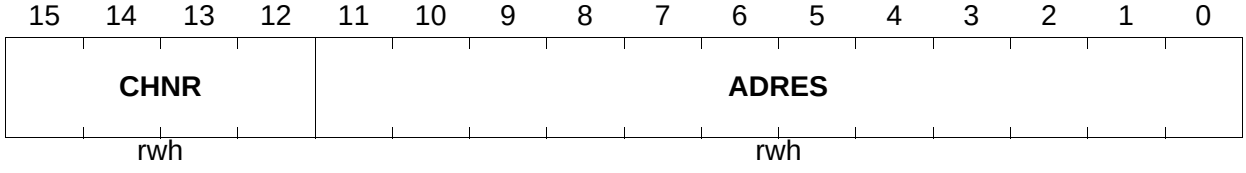
Field	Bits	Type	Description
ADSTC	[5:0]	rw	**ADC Sample Time Control** Defines the ADC sample time: $t_S = t_{BC} \times 4 \times$ (<ADSTC> + 1)
ADCTC	[11:6]	rw	**ADC Conversion Time Control** Defines the ADC basic conversion clock: $f_{BC} = f_{SYS}$ / (<ADCTC> + 1)
RES	[13:12]	rw	**Converter Resolution Control** 00 10-bit resolution 01 8-bit resolution 1x Reserved
0	[15:14]	r	**Reserved;** Read as 0; should be written with 0.

Note: The limit values for f_{BC} (see data sheet) must not be exceeded when selecting ADCTC and f_{SYS}.

6.8.3 ADC Result Registers

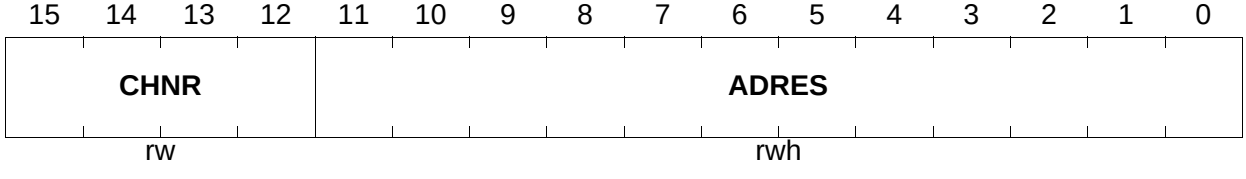
The following registers are used in the Compatibility Mode and Enhanced Mode for storage of the last conversion result.

ADC_DAT
ADC Result Register **(030_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
ADRES	[11:0]	rwh	**A/D Conversion Result** The digital result of the most recent conversion. In Compatibility Mode, the result is placed as follows: 8-bit: ADRES[9:2] 10-bit: ADRES[9:0] In Enhanced Mode, the result is placed as follows: 8-bit: ADRES[11:4] 10-bit: ADRES[11:2] *Note: Unused bits of ADRES are always set to 0.*
CHNR	[15:12]	rwh	**Channel Number** This bit field identifies the converted analog channel.

ADC_DAT2
ADC Result 2 Register **(032_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
ADRES	[11:0]	rwh	**A/D Conversion Result** The digital result of the most recent conversion. In Compatibility Mode, the result is placed as follows: 8-bit: ADRES[9:2] 10-bit: ADRES[9:0] In Enhanced Mode, the result is placed as follows: 8-bit: ADRES[11:4] 10-bit: ADRES[11:2] *Note: Unused bits of ADRES are always set to 0.*
CHNR	[15:12]	rw	**Channel Number** This bit field identifies the converted analog channel.

6.8.4 ADC Extended Result Registers

The following registers can be used in the Compatibility Mode and Enhanced Mode for storage of the conversion results.

ADC_RESA0
ADC Extended Result 0 View A Register(100_H) **Reset Value: 0000 0000_H**

ADC_RESA1
ADC Extended Result 1 View A Register(104_H) **Reset Value: 0000 1000_H**

ADC_RESA2
ADC Extended Result 2 View A Register(108_H) **Reset Value: 0000 2000_H**

ADC_RESA3
ADC Extended Result 3 View A Register($10C_H$) **Reset Value: 0000 3000_H**

ADC_RESA4
ADC Extended Result 4 View A Register(110_H) **Reset Value: 0000 4000_H**

ADC_RESA5
ADC Extended Result 5 View A Register(114_H) **Reset Value: 0000 5000_H**

ADC_RESA6
ADC Extended Result 6 View A Register(118_H) **Reset Value: 0000 6000_H**

ADC_RESA7
ADC Extended Result 7 View A Register($11C_H$) **Reset Value: 0000 7000_H**

ADC_RESA8
ADC Extended Result 8 View A Register(120_H) **Reset Value: 0000 8000_H**

ADC_RESA9
ADC Extended Result 9 View A Register(124_H) **Reset Value: 0000 9000_H**

ADC_RESA10
ADC Extended Result 10 View A Register(128_H) **Reset Value: 0000 $A000_H$**

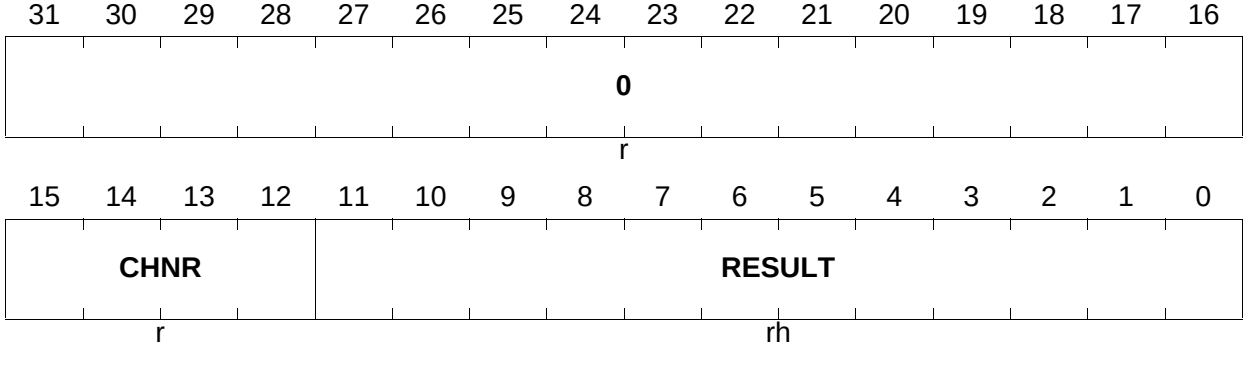
ADC_RESA11
ADC Extended Result 11 View A Register($12C_H$) **Reset Value: 0000 $B000_H$**

ADC_RESA12
ADC Extended Result 12 View A Register(130_H) **Reset Value: 0000 $C000_H$**

ADC_RESA13
ADC Extended Result 13 View A Register(134_H) **Reset Value: 0000 $D000_H$**

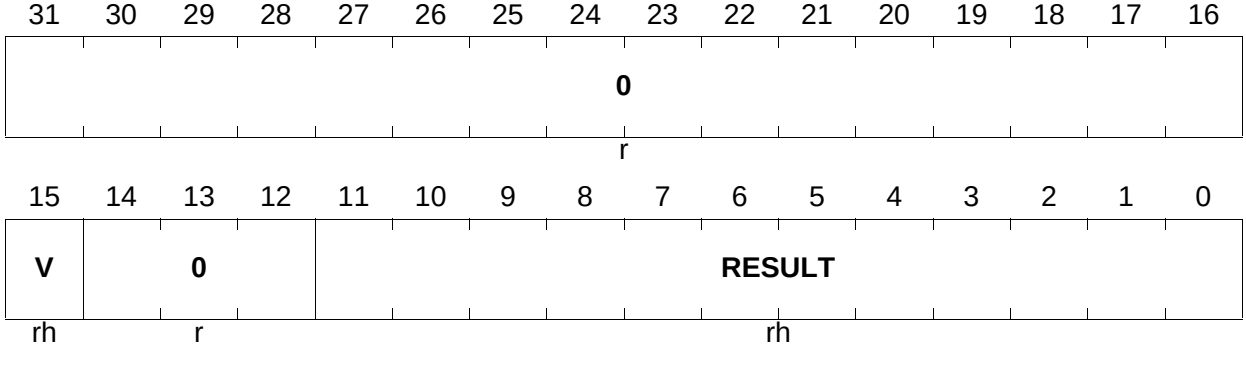
ADC_RESA14
ADC Extended Result 14 View A Register(138_H) **Reset Value: 0000 $E000_H$**

ADC_RESA15
ADC Extended Result 15 View A Register($13C_H$) **Reset Value: 0000 $F000_H$**



Field	Bits	Type	Description
RESULT	[11:0]	rh	**Conversion Result** This bit field represents the conversion result for the selected channel. If the conversion result is smaller than 10 bits, the result always starts with its MSB at bit position 11 and the unused bit positions are filled with 0.
CHNR	[15:12]	r	**Channel Number** This bit field indicates the channel number.
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

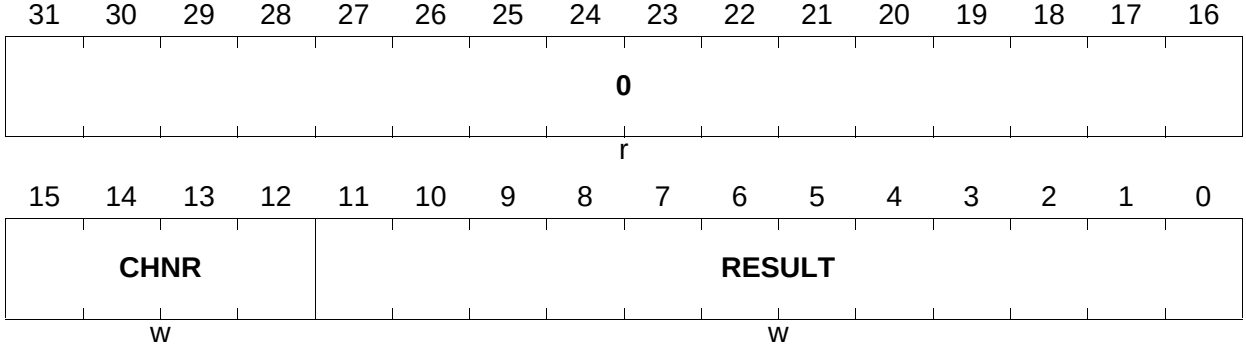
ADC_RESB0
ADC Extended Result 0 View B Register(140_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB1
ADC Extended Result 1 View B Register(144_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB2
ADC Extended Result 2 View B Register(148_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB3
ADC Extended Result 3 View B Register($14C_H$) **Reset Value: $0000\ 0000_H$**
ADC_RESB4
ADC Extended Result 4 View B Register(150_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB5
ADC Extended Result 5 View B Register(154_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB6
ADC Extended Result 6 View B Register(158_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB7
ADC Extended Result 7 View B Register($15C_H$) **Reset Value: $0000\ 0000_H$**
ADC_RESB8
ADC Extended Result 8 View B Register(160_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB9
ADC Extended Result 9 View B Register(164_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB10
ADC Extended Result 10 View B Register(168_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB11
ADC Extended Result 11 View B Register($16C_H$) **Reset Value: $0000\ 0000_H$**
ADC_RESB12
ADC Extended Result 12 View B Register(170_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB13
ADC Extended Result 13 View B Register(174_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB14
ADC Extended Result 14 View B Register(178_H) **Reset Value: $0000\ 0000_H$**
ADC_RESB15
ADC Extended Result 15 View B Register(178_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RESULT	[11:0]	rh	**Conversion Result** This bit field represents the conversion result for the selected channel. If the conversion result is smaller than 10 bits, the result always starts with its MSB at bit position 11 and the unused bit positions are filled with 0.
V	15	rh	**Valid** This bit indicates that the result has been written with a new value since the last read from this location. It becomes set with the write action of a new result and cleared when at least the low byte of the result is read out. 0 The result is not new (has already been read out). 1 The result is new (has not yet been read out).
0	[14:12] [31:16]	r	**Reserved;** Read as 0; should be written with 0.

ADC_INRES

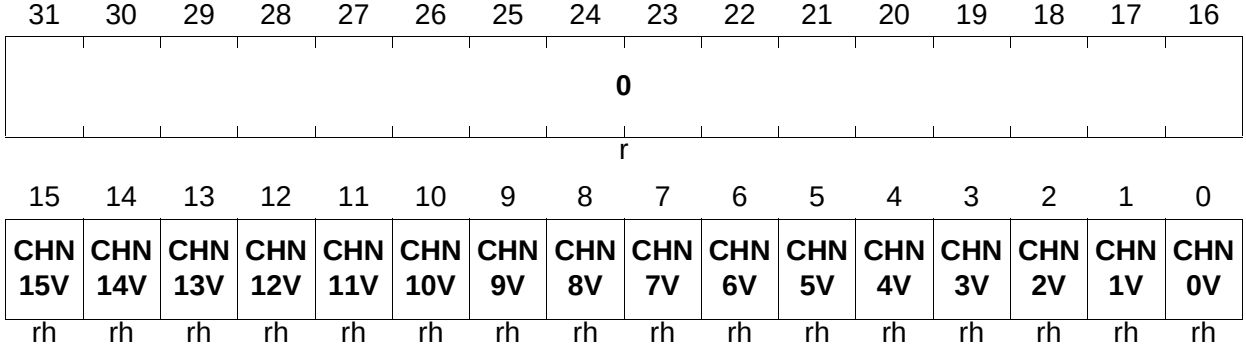
ADC Input Result Register **(180_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	[11:0]	w	**Conversion Result** This bit field updates both bit fields for the registers RESAn.RESULT and RESBn.RESULT. n is equal the value written to CHNR.

Field	Bits	Type	Description
CHNR	[15:12]	w	**Channel Number** This bit field defines the extended result registers that are updated.
0	[31:15]	r	**Reserved;** Read as 0; should be written with 0.

ADC_RESV
ADC Result Valid Register **(188_H)** **Reset Value: 0000 0000_H**

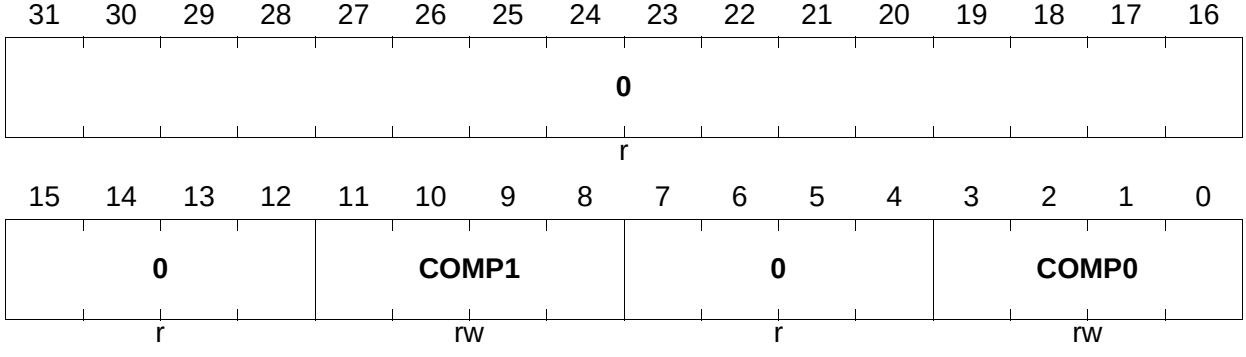


Field	Bits	Type	Description
CHNnV **(n = 0 to 15)**	n	rh	**Channel n Valid Status** This bit indicates that the result has been written with a new value since the last read from this location. It becomes set with the write action of a new result and cleared when at least the low byte of the result is read out. 0 The result is not new (has already been read out). 1 The result is new (has not yet been read out).
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

6.8.5 ADC Doorbell Register

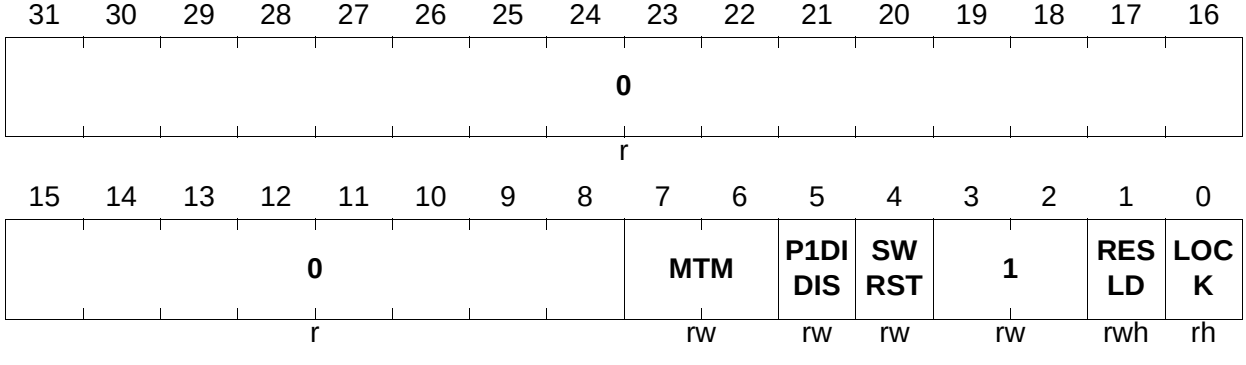
The following register control the doorbell mechanism of the extended result registers.

ADC_DBCTR
ADC Doorbell Control Register (184_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
COMP0	[3:0]	rw	**Compare Value 0** This bit field defines the compare value for the doorbell mechanism channel 0.
COMP1	[11:8]	rw	**Compare Value 1** This bit field defines the compare value for the doorbell mechanism channel 1.
0	[7:4], [31:12]	r	**Reserved;** Read as 0; should be written with 0.

SCU_SYSCON
SCU System Control Register (820_H) Reset Value: 0000 000C$_H$



Field	Bits	Type	Description
LOCK	0	rh	**PLL Lock Status Flag** 0 PLL is not locked 1 PLL is locked
RESLD	1	rwh	**Restart Lock Detection** Setting this bit will reset bit LOCK and restart the lock detection. When set, this bit is automatically cleared. 0 No effect 1 Reset LOCK and restart lock detection
SWRST	4	rw	**Software Reset Trigger** Setting this bit will automatically request and generate a reset. With the reset execution, this bit is automatically cleared.
P1DIDIS	5	rw	**Port 1 Digital Input Disable** This bit controls the digital input stage for all port 1 pins. 0 Digital input stage (Schmitt-trigger) is enabled 1 Digital input stage (Schmitt-trigger) is disabled. This is necessary if pins are used as analog input.
MTM	[7:6]	rw	**Multiplexer Test Mode for Channel 0** This bit enables/disables the Multiplexer Test Mode for the input channel 0. This feature is independent of the current mode of the analog part. If the Multiplexer Test Mode is enabled, the analog input is connected to ADC ground via an internal resistance[1)]. This structure creates a voltage divider to ground, so the measurement result becomes smaller. 00 The Multiplexer Test Mode is disabled. The analog input is not connected to ground and can be used for normal measurements. 01 The Multiplexer Test Mode is enabled. The internal resistance to ground is in the range of 300 Ohm. 10 The Multiplexer Test Mode is enabled. The internal resistance to ground is in the range of 70 Ohm. 11 Reserved, like 00

Field	Bits	Type	Description
1	[3:2]	rw	**Reserved;** Should be written with 1.
0	[15:8]	r	**Reserved;** Read as 0; should be written with 0.

1) Please refer to the ACDC chapter for the current capability of the grounding resistor, especially when using RC input filters at the analog inputs.

7 Parallel Ports

The CIC751 has two parallel ports, port 0 and port 1. Port 0 controls all pins for the communication (MLI/SSC, Service Requests). Port 1 controls 16 inputs of the 16 ADC channels.

Each port line has a number of control and data bits, enabling very flexible usage of the line. Each port pin can be configured for input or output operation. In Input Mode, the output driver is switched off (high-impedance). The actual voltage level present at the port pin is translated into a logical 0 or 1 via a Schmitt-Trigger device and can be read via the read-only register Pn_IN. The input can also be connected directly to the various inputs of the peripheral units (Alternate Input). The function of the input line from the pin to the input register Pn_IN and to the alternate input is independent of whether the port pin operates as input or output. This means that when the port is in Output Mode, the level of the pin can be read via Pn_IN or a peripheral can use the pin level as an input.

In Output Mode, the output driver is activated and drives the value supplied through the multiplexer to the port pin. Switching between Input and Output Mode is accomplished through the Pn_IOCRx registers, which enables or disables the output driver. If a peripheral unit uses a GPIO port line as a bi-directional I/O line, register Pn_IOCR has to be written for input or output selection. The Pn_IOCRx registers further controls the driver type of the output driver, and determines whether an internal weak pull-up or pull-down device is alternatively connected to the pin when used as an input. This offers additional advantages in an application.

The output multiplexer in front of the output driver selects the source for the GPIO line when used as output. If the pin is used as general-purpose output, the multiplexer is switched (by Pn_IOCRx register) to the Output Data Register Pn_OUT. If the on-chip peripheral units use the pin for output, the alternate output lines ALT1 to ALT3 can be switched via the multiplexer to the output driver. The data written into the output register Pn_OUT can be used as input data to an on-chip peripheral.

When selected as general-purpose output line, the logic state of each port pin can be changed individually by programming the pin-related bits in the Output Modification Register Pn_OMR. The bits in Pn_OMR make it possible to set, reset, toggle, or leave the bits in the Pn_OUT register unchanged.

When selected as general-purpose output line, the actual logic level at the pin can be examined through reading latch Pn_IN and compared against the applied output level. This can be used to detect some electrical failures at the pin caused through external circuitry. Collisions on the external communication lines can be detected when a high level (1) is output, but a low level (0) is seen when reading the pin value via the input register Pn_IN.

7.1 Port 0

This section describes the control mechanisms of all pins other than the ADC analog channels and the reset pin PORST.

7.1.1 Block Diagram

Figure 7-1 shows the different options for the control of port 0.

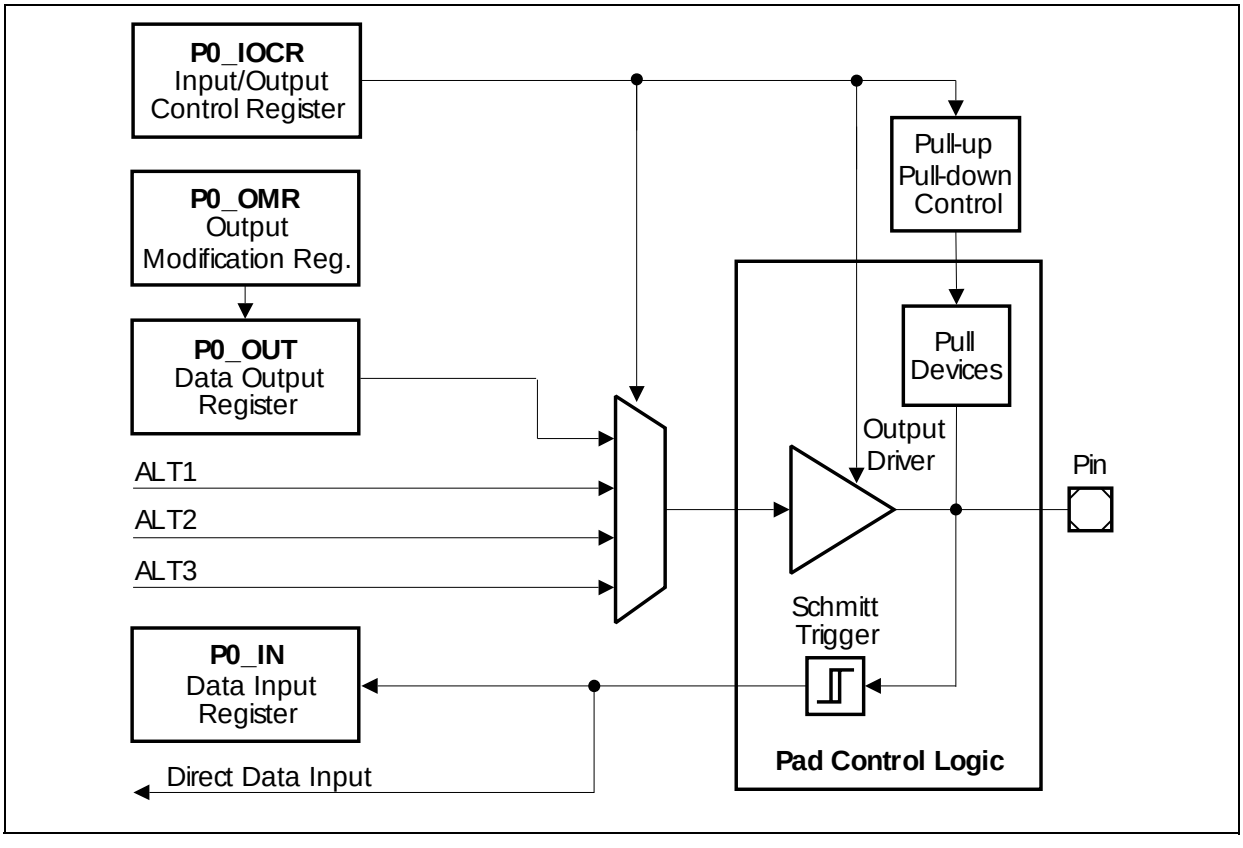


Figure 7-1 Port 0 Control Structure

7.1.2 Input Stage

The input value of each pin can be used in two different ways:

1. The input value of pin 0.x is always available at bit P0_IN.Px
2. The input can be used directly by peripheral if connected
   a) pins P0.1, P0.3, P0.6, and P0.7 are connected to the MLI
   b) pins P0.2, P0.6, and P0.7 are connected to the SSC
   c) pins P0.3, P0.8, P0.9, P0.10, P0.11, and P0.12 are used for general purpose

Note: Not all pins are directly connected for functional reasons to a peripheral.

7.1.3 Port 0 Routing

The following table describes the mapping of the pins of Port 0 and the related I/O functions.

Table 7-1 Port 0 Input/Output Functions

Port Pin	I/O	Select	Connected Function	From / to Module
P0.0	Input		Not used	
	Output	GPIO	Port Output Register P0_OUT.P0	Port
		ALT1	TCLK	MLI
		ALT2	SR3	SCU
		ALT3	SR3	SCU
P0.1	Input		TREADY	MLI
	Output	GPIO	Port Output Register P0_OUT.P1	Port
		ALT1	SR4	SCU
		ALT2	SR4	SCU
		ALT3	SR4	SCU
P0.2	Input		SCLK	SSC
	Output	GPIO	Port Output Register P0_OUT.P2	
		ALT1	TVAILD	MLI
		ALT2	Not used	
		ALT3	Not used	
P0.3	Input		Not used	
	Output	GPIO	Port Output Register P0_OUT.P3	
		ALT1	TDAT	MLI
		ALT2	MRST	SSC
		ALT3	Not used	
P0.4	Input		RCLK	MLI
			RCLK	SCU
	Output	GPIO	Port Output Register P0_OUT.P4	
		ALT1	Not used	
		ALT2	Not used	
		ALT3	Not used	

Table 7-1 Port 0 Input/Output Functions (cont'd)

Port Pin	I/O	Select	Connected Function	From / to Module
P0.5	Input		Not used	
	Output	GPIO	Port Output Register P0_OUT.P5	
		ALT1	RREADY	MLI
		ALT2	RDY	SSC
		ALT3	Not used	
P0.6	Input		RVALID	MLI
			$\overline{\text{SLS}}$	SSC
	Output	GPIO	Port Output Register P0_OUT.P6	
		ALT1	Not used	
		ALT2	Not used	
		ALT3	Not used	
P0.7	Input		RDATA	MLI
			MTSR	SSC
	Output	GPIO	Port Output Register P0_OUT.P7	
		ALT1	Not used	
		ALT2	Not used	
		ALT3	Not used	
P0.8	Input		MODE	SCU
	Output	GPIO	Port Output Register P0_OUT.P8	
		ALT1	SR5	SCU
		ALT2	SR5	SCU
		ALT3	SR5	SCU
P0.9	Input		$\overline{\text{TESTMODE}}$	SCU
	Output	GPIO	Port Output Register P0_OUT.P9	
		ALT1	Not used	
		ALT2	Not used	
		ALT3	Not used	

Table 7-1 Port 0 Input/Output Functions (cont'd)

Port Pin	I/O	Select	Connected Function	From / to Module
P0.10	Input		SR0	SCU
	Output	GPIO	Port Output Register P0_OUT.P10	
		ALT1	SR0	SCU
		ALT2	SR0	SCU
		ALT3	SR0	SCU
P0.11	Input		SR1	SCU
	Output	GPIO	Port Output Register P0_OUT.P11	
		ALT1	SR1	SCU
		ALT2	SR1	SCU
		ALT3	SR1	SCU
P0.12	Input		SR2	SCU
	Output	GPIO	Port Output Register P0_OUT.P12	
		ALT1	SR2	SCU
		ALT2	SR2	SCU
		ALT3	SR2	SCU

7.1.4 Port 0 Register Description

7.1.4.1 Port 0 Control Register

P0_IN
Port 0 Input Register **($A24_H$)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
r			rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px **(x = 0-12)**	x	rwh	**Port 0 Input Bit x** This bit indicates the level at the input pin of port P0, pin x. 0 The input level of P0.x is 0 1 The input level of P0.x is 1
0	[31:13]	r	**Reserved;** Read as 0; should be written with 0.

P0_OUT
Port 0 Output Register **($A00_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
r			rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px **(x = 0-12)**	x	rwh	**Port Output Bit x** This bit defines the level at the output pin of port 0, pin x if the output is selected as GPIO output. 0 The output level of P0.x is 0 1 The output level of P0.x is 1
0	[31:13]	r	**Reserved;** Read as 0; should be written with 0.

P0_OMR
Port 0 Output Modification Register (A04$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0		**PR 12**	**PR 11**	**PR 10**	**PR 9**	**PR 8**	**PR 7**	**PR 6**	**PR 5**	**PR 4**	**PR 3**	**PR 2**	**PR 1**	**PR 0**
	r		w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0		**PS 12**	**PS 11**	**PS 10**	**PS 9**	**PS 8**	**PS 7**	**PS 6**	**PS 5**	**PS 4**	**PS 3**	**PS 2**	**PS 1**	**PS 0**
	r		w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-12)	x	w	**Port Set Bit x** Setting this bit sets or toggles the corresponding bit in the port output register P0_OUT (see **Table 7-2**). On a read access, this bit returns 0.
PCx (x = 0-12)	x + 16	w	**Port Clear Bit x** Setting this bit clears or toggles the corresponding bit in the port output register P0_OUT. (see **Table 7-2**). On a read access, this bit returns 0.
0	[15:13] [31:29]	r	**Reserved;** Read as 0; should be written with 0.

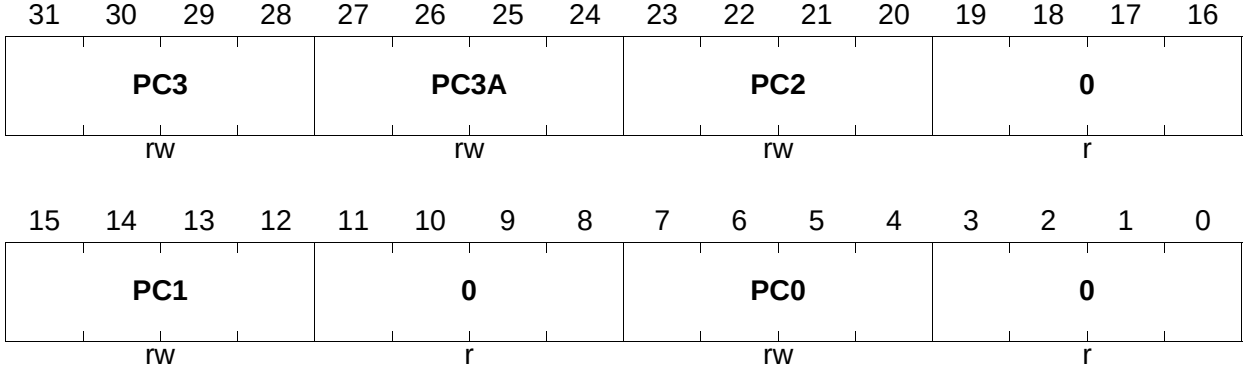
Function of the PCx and PSx Bit fields

Table 7-2 Function of the Bits PCx and PSx

PCx	PSx	Function
0 or no write access	0 or no write access	Bit P0_OUT.Px is not changed
0 or no write access	1	Bit P0_OUT.Px is set
1	0 or no write access	Bit P0_OUT.Px is cleared
1	1	Bit P0_OUT.Px is toggled

Note: If a bit position is not written (one out of two bytes not targeted by a byte write), the corresponding value is considered as 0. Toggling a bit requires one 16-bit write.

P0_IOCR0
Port 0 Input/Output Control 0 Register ($A10_H$) **Reset Value: Table 7-3**



Field	Bits	Type	Description
PC0	[7:4]	rw	**Port Input/Output Control Bit 0** see **Table 7-4**
PC1	[15:12]	rw	**Port Input/Output Control Bit 1** see **Table 7-4**
PC2	[23:20]	rw	**Port Input/Output Control Bit 2** see **Table 7-4**
PC3A	[27:24]	rw	**Port Input/Output Control Bit 3** see **Table 7-4**
PC3	[31:28]	rw	**Port Input/Output Control Bit 3** see **Table 7-4**
0	[3:0], [11:8], [19:16]	r	**Reserved;** Read as 0; should be written with 0.

Table 7-3 Register Reset Values

Register Reset Type	Reset Values	Reset Short Name	Reset Mode	Note
SSC Mode Selected	A020 2020_H	-	Asynchronous	
MLI Mode Selected	9290 0090_H	-	Asynchronous	

Coding of the PCx Bit field

The coding of the GPIO port behavior is done by the bit fields in the port control registers P0_IOCRx. There's a control bit field PCx for each port pin. The bit fields PCx are located

in separate control registers in order to allow modifying a port pin (without influencing the others) with simple move operations.

Table 7-4 PCx Coding

PCx[3:0]	I/O	Selected Pull-up/down / Selected Output Function
0000_B	Input Mode	No pull device connected
0001_B		Pull-down device connected
0010_B		Pull-up device connected
0011_B		No pull device connected
0100_B		No pull device connected
0101_B		Pull-down device connected
0110_B		Pull-up device connected
0111_B		No pull device connected
1000_B	Output Mode (Direct input) Push-pull	General purpose Output
1001_B		Output function ALT1
1010_B		Output function ALT2
1011_B		Output function ALT3
1100_B	Output Mode (Direct input) Open-drain	General purpose Output
1101_B		Output function ALT1
1110_B		Output function ALT2
1111_B		Output function ALT3

P0_IOCR4

Port 0 Input/Output Control 4 Register ($A14_H$) **Reset Value: Table 7-5**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7				0				PC6				0			
rw				r				rw				r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5				PC5A				PC4				0			
rw				rw				rw				r			

Field	Bits	Type	Description
PC4	[7:4]	rw	**Port Input/Output Control Bit 4** see **Table 7-4**
PC5A	[11:8]	rw	**Port Input/Output Control Bit 5** see **Table 7-4**
PC5	[15:12]	rw	**Port Input/Output Control Bit 5** see **Table 7-4**
PC6	[23:20]	rw	**Port Input/Output Control Bit 6** see **Table 7-4**
PC7	[31:28]	rw	**Port Input/Output Control Bit 7** see **Table 7-4**
0	[3:0], [19:16] [27:24]	r	**Reserved;** Read as 0; should be written with 0.

Table 7-5 Register Reset Values

Register Reset Type	Reset Values	Reset Short Name	Reset Mode	Note
SSC Mode Selected	0020 A020$_H$	-	Asynchronous	
MLI Mode Selected	0020 9020$_H$	-	Asynchronous	

P0_IOCR8

Port 0 Input/Output Control 8 Register (A18$_H$) **Reset Value: Table 7-6**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC11				**0**				**PC10**				**0**			
rw				r				rw				r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC9				**0**				**PC8**				**0**			
rw				r				rw				r			

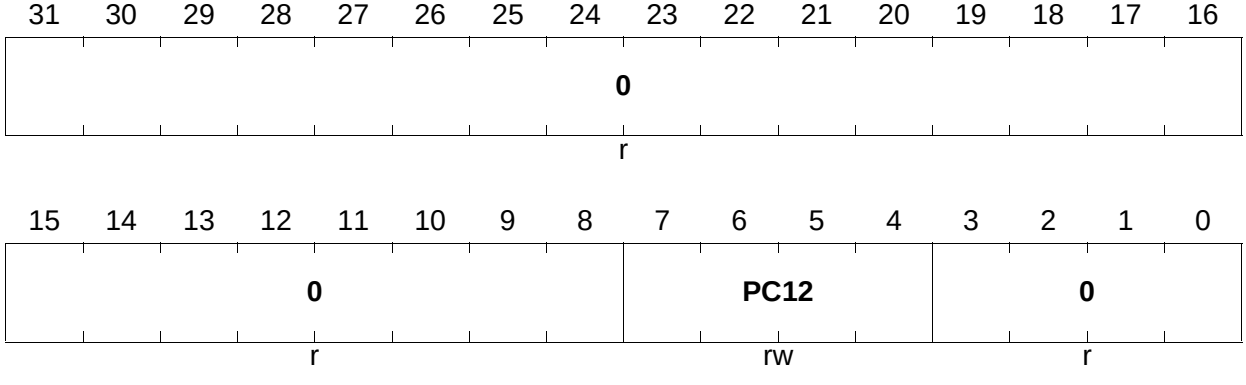
Field	Bits	Type	Description
PC8	[7:4]	rw	**Port Input/Output Control Bit 8** see **Table 7-4**
PC9	[15:12]	rw	**Port Input/Output Control Bit 9** see **Table 7-4**
PC10	[23:20]	rw	**Port Input/Output Control Bit 10** see **Table 7-4**
PC11	[31:28]	rw	**Port Input/Output Control Bit 11** see **Table 7-4**
0	[3:0], [11:8], [19:16] [27:24]	r	**Reserved;** Read as 0; should be written with 0.

Table 7-6 Register Reset Values

Register Reset Type	Reset Values	Reset Short Name	Reset Mode	Note
SSC Mode Selected	2020 2020_H	-	Asynchronous	
MLI Mode Selected	2020 2020_H	-	Asynchronous	

P0_IOCR12

Port 0 Input/Output Control 12 Register ($A1C_H$) **Reset Value: Table 7-7**



Field	Bits	Type	Description
PC12	[7:4]	rw	**Port Input/Output Control Bit 12** see **Table 7-4**

Field	Bits	Type	Description
0	[3:0], [31:8]	r	**Reserved;** Read as 0; should be written with 0.

Table 7-7 Register Reset Values

Register Reset Type	Reset Values	Reset Short Name	Reset Mode	Note
SSC Mode Selected	$0000\ 0020_H$	-	Asynchronous	
MLI Mode Selected	$0000\ 0020_H$	-	Asynchronous	

7.2 Port 1

This section describes the control mechanisms of all pins used as ADC analog channels.

7.2.1 Block Diagram

Figure 7-1 shows the different options for the control of port 1.

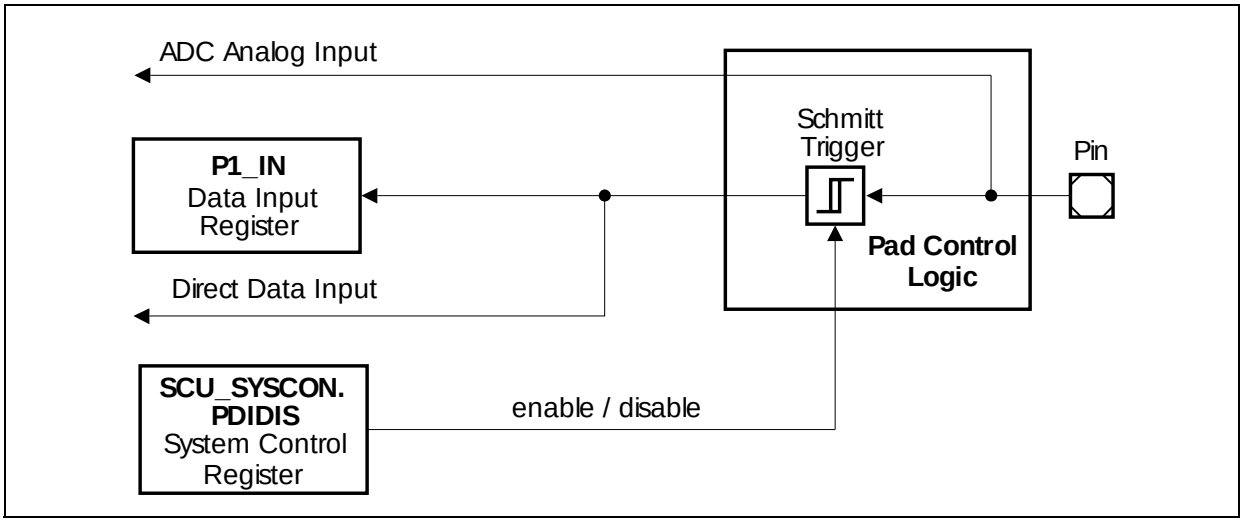


Figure 7-2 Port 1 Control Structure

7.2.2 Input Stage

The input value of each pin can be used in up to three different ways:

1. The input value of pin 0.x is always available at bit P0_IN.Px
2. The input can be used directly as External Trigger Input
   a) pins P1.4, and P1.14 are usable as trigger inputs
3. The pins are used as analog inputs for the ADC
   a) pin P1.0 is equipped with the Multiplexer Test Mode Feature.

7.2.3 Port 1 Routing

The following table describes the mapping of the pins of Port 0 and the related I/O functions.

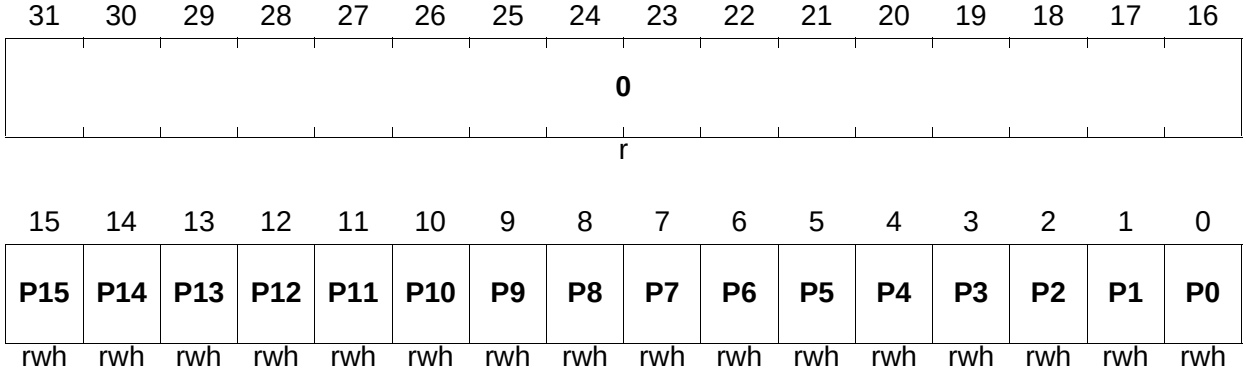
Table 7-8 Port 1 Input/Output Functions

Port Pin	I/O	Select	Connected Function	From / to Module
P1.0	Input Mode		Analog Input 0	ADC
P1.1			Analog Input 1	ADC
P1.2			Analog Input 2	ADC
P1.3			Analog Input 3	ADC
P1.4			Analog Input 4	ADC
			Trigger	SCU/DMA
P1.5			Analog Input 5	ADC
P1.6			Analog Input 6	ADC
P1.7			Analog Input 7	ADC
P1.8			Analog Input 8	ADC
P1.9			Analog Input 9	ADC
P1.10			Analog Input 10	ADC
P1.11			Analog Input 11	ADC
P1.12			Analog Input 12	ADC
P1.13			Analog Input 13	ADC
P1.14			Analog Input 14	ADC
			Trigger	SCU/DMA
P1.15			Analog Input 15	ADC

7.2.4 Port 1 Register Description

7.2.4.1 Port Input Register

P1_IN
Port 1 Input Register **($A64_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
Px **(x = 15-0)**	x	rwh	**Port 1 Input Bit x** This bit indicates the level at the input pin of port P1, pin x. 0 The input level of P1.x is 0 1 The input level of P1.x is 1
0	[31:16]	r	**Reserved;** Read as 0; should be written with 0.

7.3 Ports Register Overview

All port register can only be accessed by 32-bit accesses. 8-bit or 16-bit accesses are not allowed and lead to errors.

Table 7-9 Port 0 Registers

Register Short Name	Register Long Name	Address	Description see
P0_OUT	Port 0 Output Register	$A00_H$	Page 7-6
P0_OMR	Port 0 Output Modification Register	$A04_H$	Page 7-7
P0_IOCR0	Port 0 Input/Output Control Register 0	$A10_H$	Page 7-8
P0_IOCR4	Port 0 Input/Output Control Register 4	$A14_H$	Page 7-9
P0_IOCR8	Port 0 Input/Output Control Register 8	$A18_H$	Page 7-10

Table 7-9 Port 0 Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
P0_IOCR12	Port 0 Input/Output Control Register 12	$A1C_H$	Page 7-11
P0_IN	Port 0 Input Register	$A24_H$	Page 7-5

Table 7-10 Port 1 Registers

Register Short Name	Register Long Name	Address	Description see
P1_IN	Port 1 Input Register	$A64_H$	Page 7-15

8 Register Overview

This chapter describes all registers of the CIC751. It also describes the read/write access rights of the specific address ranges/registers.

8.1 Address Map of CIC751

Table 8-1 shows the block address map of CIC751.

Table 8-1 Block Address Map of CIC751

Unit	Address Range
Reserved	$0000\ 0000_H$ - $0000\ 01FF_H$
Micro Link Interface (MLI)	$0000\ 0200_H$ - $0000\ 02FF_H$
Reserved	$0000\ 0300_H$ - $0000\ 03FF_H$
Direct Memory Access Controller (DMA)	$0000\ 0400_H$ - $0000\ 05FF_H$
Reserved	$0000\ 0600_H$ - $0000\ 07FF_H$
System Control Unit (SCU)	$0000\ 0800_H$ - $0000\ 08FF_H$
Synchronous Serial Interface (SSC)	$0000\ 0900_H$ - $0000\ 09FF_H$
Ports	$0000\ 0A00_H$ - $0000\ 0AFF_H$
Reserved	$0000\ 0B00_H$ - $0000\ 0FFF_H$

Table 8-1 Block Address Map of CIC751 (cont'd)

Unit		Address Range
Analog-to-Digital Converter (ADC)		$0000\ 1000_H$ - $0000\ 11FF_H$
Reserved		$0000\ 1200_H$ - $0000\ 7FFF_H$
MLI Small Transfer Windows	Pipe 0	$0000\ 8000_H$ - $0000\ 9FFF_H$
	Pipe 1	$0000\ A000_H$ - $0000\ BFFF_H$
	Pipe 2	$0000\ C000_H$ - $0000\ DFFF_H$
	Pipe 3	$0000\ E000_H$ - $0000\ FFFF_H$
MLI Large Transfer Windows	Pipe 0	$0001\ 0000_H$ - $0001\ FFFF_H$
	Pipe 1	$0002\ 0000_H$ - $0002\ FFFF_H$
	Pipe 2	$0003\ 0000_H$ - $0003\ FFFF_H$
	Pipe 3	$0004\ 0000_H$ - $0004\ FFFF_H$
Reserved		$0005\ 0000_H$ - $FFFF\ FFFF_H$

8.1.1 Access Rules

The following rules apply for all accesses:

- All registers read and write conditions can be found in the different module chapters
- Accesses to reserved marked addresses are forbidden and lead to an undefined behavior
- All port register can only be accesses by 32-bit accesses
- The MLI master automatically generates 32-bit accesses for 32-bit data transfers
- The MLI master automatically generates 16-bit accesses for 16-bit data transfers
- The MLI master automatically generates 8-bit accesses for 8-bit data transfers
- The SSC master automatically generates 32-bit accesses to all registers beside the ADC register
- The SSC master automatically generates 16-bit accesses to all ADC registers
- Accesses to not configured parts of an MLI Remote Window are forbidden and lead to an undefined behavior
- All registers beside the ADC core register are 32-bit registers
- The ADC core registers are 16-bit registers; 32-bit accesses are forbidden and lead to an undefined behavior
- The registers are ADC core registers
  - ADC_CON; ADC Control Register
  - ADC_CON1; ADC Control 1 Register
  - ADC_CTR0; ADC Control 0 Register
  - ADC_CTR2; ADC Control 2 Register
  - ADC_CTR2IN; ADC Injection Control 2 Register
  - ADC_DAT; ADC Result Register
  - ADC_DAT2; ADC Result 2 Register

8.2 Registers Tables

Table 8-2 MLI Kernel Registers

Register Short Name	Register Long Name	Address	Description see
Reserved	Reserved	0000 0200$_H$ - 0000 0208$_H$	-
MLI_FDR	Fractional Divider Register	0000 020C$_H$	**Page 4-49**
MLI_TCR	Transmitter Control Register	0000 0210$_H$	**Page 4-59**
MLI_TSTATR	Transmitter Status Register	0000 0214$_H$	**Page 4-62**
MLI_TP0STATR	Transmitter Pipe 0 Status Register	0000 0218$_H$	**Page 4-64**
MLI_TP1STATR	Transmitter Pipe 1 Status Register	0000 021C$_H$	**Page 4-64**
MLI_TP2STATR	Transmitter Pipe 2 Status Register	0000 0220$_H$	**Page 4-64**
MLI_TP3STATR	Transmitter Pipe 3 Status Register	0000 0224$_H$	**Page 4-64**
MLI_TCMDR	Transmitter Command Register	0000 0228$_H$	**Page 4-66**
MLI_TSTATR	Transmitter Receiver Status Register	0000 022C$_H$	**Page 4-62**
MLI_TP0AOFR	Transmitter Pipe 0 Address Offset Register	0000 0230$_H$	**Page 4-70**
MLI_TP1AOFR	Transmitter Pipe 1 Address Offset Register	0000 0234$_H$	**Page 4-70**
MLI_TP2AOFR	Transmitter Pipe 2 Address Offset Register	0000 0238$_H$	**Page 4-70**
MLI_TP3AOFR	Transmitter Pipe 3 Address Offset Register	0000 023C$_H$	**Page 4-70**
MLI_TP0DATAR	Transmitter Pipe 0 Data Register	0000 0240$_H$	**Page 4-71**
MLI_TP1DATAR	Transmitter Pipe 1 Data Register	0000 0244$_H$	**Page 4-71**
MLI_TP2DATAR	Transmitter Pipe 2 Data Register	0000 0248$_H$	**Page 4-71**
MLI_TP3DATAR	Transmitter Pipe 3 Data Register	0000 024C$_H$	**Page 4-71**
MLI_TDRAR	Transmitter Data Read Answer Register	0000 0250$_H$	**Page 4-72**
MLI_TP0BAR	Transmitter Pipe 0 Base Address Register	0000 0254$_H$	**Page 4-73**

Table 8-2 MLI Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
MLI_TP1BAR	Transmitter Pipe 1 Base Address Register	0000 0258$_H$	Page 4-73
MLI_TP2BAR	Transmitter Pipe 2 Base Address Register	0000 025C$_H$	Page 4-73
MLI_TP3BAR	Transmitter Pipe 3 Base Address Register	0000 0260$_H$	Page 4-73
MLI_TCBAR	Transmitter Copy Base Address Register	0000 0264$_H$	Page 4-74
MLI_RCR	Receiver Control Register	0000 0268$_H$	Page 4-75
MLI_RP0BAR	Receiver Pipe 0 Base Address Register	0000 026C$_H$	Page 4-78
MLI_RP1BAR	Receiver Pipe 1 Base Address Register	0000 0270$_H$	Page 4-78
MLI_RP2BAR	Receiver Pipe 2 Base Address Register	0000 0274$_H$	Page 4-78
MLI_RP3BAR	Receiver Pipe 3 Base Address Register	0000 0278$_H$	Page 4-78
MLI_RP0STATR	Receiver Pipe 0 Status Register	0000 027C$_H$	Page 4-79
MLI_RP1STATR	Receiver Pipe 1 Status Register	0000 0280$_H$	Page 4-79
MLI_RP2STATR	Receiver Pipe 2 Status Register	0000 0284$_H$	Page 4-79
MLI_RP3STATR	Receiver Pipe 3 Status Register	0000 0288$_H$	Page 4-79
MLI_RADRR	Receiver Address Register	0000 028C$_H$	Page 4-81
MLI_RDATAR	Receiver Data Register	0000 0290$_H$	Page 4-82
MLI_SCR	Set Clear Register	0000 0294$_H$	Page 4-51
MLI_TIER	Transmitter Interrupt Enable Register	0000 0298$_H$	Page 4-83
MLI_TISR	Transmitter Interrupt Status Register	0000 029C$_H$	Page 4-85
MLI_TINPR	Transmitter Interrupt Node Pointer Register	0000 02A0$_H$	Page 4-86
MLI_RIER	Receiver Interrupt Enable Register	0000 02A4$_H$	Page 4-88
MLI_RISR	Receiver Interrupt Status Register	0000 02A8$_H$	Page 4-90

Table 8-2 MLI Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
MLI_RINPR	Receiver Interrupt Node Pointer Register	0000 02AC$_H$	Page 4-92
MLI_GINTR	Global Interrupt Set Register	0000 02B0$_H$	Page 4-53
MLI_OICR	Output Input Control Register	0000 02B4$_H$	Page 4-54
MLI_MEM0	MLI Memory 0 Register	0000 02B8$_H$	-
MLI_MEM1	MLI Memory 1 Register	0000 02BC$_H$	-
Reserved	Reserved	0000 02C0$_H$ - 0000 02FF$_H$	-

Table 8-3 DMA Kernel Registers

Register Short Name	Register Long Name	Address	Description see
Reserved	Reserved	0000 0400$_H$ - 0000 040C$_H$	-
DMA_CHRSTR	DMA Channel Request Register	0000 0410$_H$	Page 3-29
DMA_TRSR	DMA Transaction Request State Register	0000 0414$_H$	Page 3-31
DMA_STREQ	DMA Software Transaction Request Register	0000 0418$_H$	Page 3-32
DMA_HTREQ	DMA Hardware Transaction Request Register	0000 041C$_H$	Page 3-33
Reserved	Reserved	0000 0420$_H$	-
DMA_ERRSR	DMA Error Status Register	0000 0424$_H$	Page 3-34
DMA_CLRE	DMA Clear Error Register	0000 0428$_H$	Page 3-36
Reserved	Reserved	0000 042C$_H$	-
DMA_MESR	DMA Move Engine Status Register	0000 0430$_H$	Page 3-37
DMA_ME0R	DMA Move Engine 0 Read Register	0000 0434$_H$	Page 3-38
Reserved	Reserved	0000 0438$_H$	-
DMA_MEM0	DMA Memory 0 Register	0000 043C$_H$	-

Table 8-3 DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_MEM1	DMA Memory 1 Register	0000 0440$_H$	-
DMA_MEM2	DMA Memory 2 Register	0000 0444$_H$	-
DMA_MEM3	DMA Memory 3 Register	0000 0448$_H$	-
DMA_MEM4	DMA Memory 4 Register	0000 044C$_H$	-
DMA_MEM5	DMA Memory 5 Register	0000 0450$_H$	-
Reserved	Reserved	0000 0454$_H$ - 0000 0464$_H$	-
DMA_MEM14	DMA Memory 14 Register	0000 0468$_H$	-
Reserved	Reserved	0000 046C$_H$ - 0000 047C$_H$	-
DMA_CHSR00	DMA Channel 0 Status Register	0000 0480$_H$	Page 3-42
DMA_CHCR00	DMA Channel 0 Control Register	0000 0484$_H$	Page 3-39
DMA_MEM6	DMA Memory 6 Register	0000 0488$_H$	-
DMA_ADRCR00	DMA Channel 0 Address Control Register	0000 048C$_H$	Page 3-43
DMA_SADR00	DMA Channel 0 Source Address Register	0000 0490$_H$	Page 3-47
DMA_DADR00	DMA Channel 0 Destination Address Register	0000 0494$_H$	Page 3-36
DMA_SHADR00	DMA Channel 0 Shadow Address Register	0000 0498$_H$	Page 3-49
DMA_CHSR01	DMA Channel 1 Status Register	0000 04A0$_H$	Page 3-42
DMA_CHCR01	DMA Channel 1 Control Register	0000 04A4$_H$	Page 3-39
DMA_MEM7	DMA Memory 7 Register	0000 04A8$_H$	-
DMA_ADRCR01	DMA Channel 1 Address Control Register	0000 04AC$_H$	Page 3-43
DMA_SADR01	DMA Channel 1 Source Address Register	0000 04B0$_H$	Page 3-47
DMA_DADR01	DMA Channel 1 Destination Address Register	0000 04B4$_H$	Page 3-48

Table 8-3 DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_SHADR01	DMA Channel 1 Shadow Address Register	0000 04B8$_H$	Page 3-49
DMA_CHSR02	DMA Channel 2 Status Register	0000 04C0$_H$	Page 3-42
DMA_CHCR02	DMA Channel 2 Control Register	0000 04C4$_H$	Page 3-39
DMA_MEM8	DMA Memory 8 Register	0000 04C8$_H$	-
DMA_ADRCR02	DMA Channel 2 Address Control Register	0000 04CC$_H$	Page 3-43
DMA_SADR02	DMA Channel 2 Source Address Register	0000 04D0$_H$	Page 3-47
DMA_DADR02	DMA Channel 2 Destination Address Register	0000 04D4$_H$	Page 3-48
DMA_SHADR02	DMA Channel 2 Shadow Address Register	0000 04D8$_H$	Page 3-49
DMA_CHSR03	DMA Channel 3 Status Register	0000 04E0$_H$	Page 3-42
DMA_CHCR03	DMA Channel 3 Control Register	0000 04E4$_H$	Page 3-39
DMA_MEM9	DMA Memory 9 Register	0000 04E8$_H$	-
DMA_ADRCR03	DMA Channel 3 Address Control Register	0000 04EC$_H$	Page 3-43
DMA_SADR03	DMA Channel 3 Source Address Register	0000 04F0$_H$	Page 3-47
DMA_DADR03	DMA Channel 3 Destination Address Register	0000 04F4$_H$	Page 3-48
DMA_SHADR03	DMA Channel 3 Shadow Address Register	0000 04F8$_H$	Page 3-49
DMA_CHSR04	DMA Channel 4 Status Register	0000 0500$_H$	Page 3-42
DMA_CHCR04	DMA Channel 4 Control Register	0000 0504$_H$	Page 3-39
DMA_MEM10	DMA Memory 10 Register	0000 0508$_H$	-
DMA_ADRCR04	DMA Channel 4 Address Control Register	0000 050C$_H$	Page 3-43
DMA_SADR04	DMA Channel 4 Source Address Register	0000 0510$_H$	Page 3-47
DMA_DADR04	DMA Channel 4 Destination Address Register	0000 0514$_H$	Page 3-48

Table 8-3 DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_SHADR04	DMA Channel 4 Shadow Address Register	0000 0518$_H$	Page 3-49
DMA_CHSR05	DMA Channel 5 Status Register	0000 0520$_H$	Page 3-42
DMA_CHCR05	DMA Channel 5 Control Register	0000 0524$_H$	Page 3-39
DMA_MEM11	DMA Memory 11 Register	0000 0528$_H$	-
DMA_ADRCR05	DMA Channel 5 Address Control Register	0000 052C$_H$	Page 3-43
DMA_SADR05	DMA Channel 5 Source Address Register	0000 0530$_H$	Page 3-47
DMA_DADR05	DMA Channel 5 Destination Address Register	0000 0534$_H$	Page 3-48
DMA_SHADR05	DMA Channel 5 Shadow Address Register	0000 0538$_H$	Page 3-49
DMA_CHSR06	DMA Channel 6 Status Register	0000 0540$_H$	Page 3-42
DMA_CHCR06	DMA Channel 6 Control Register	0000 0544$_H$	Page 3-39
DMA_MEM12	DMA Memory 12 Register	0000 0548$_H$	-
DMA_ADRCR06	DMA Channel 6 Address Control Register	0000 054C$_H$	Page 3-43
DMA_SADR06	DMA Channel 6 Source Address Register	0000 0550$_H$	Page 3-47
DMA_DADR06	DMA Channel 6 Destination Address Register	0000 0554$_H$	Page 3-48
DMA_SHADR06	DMA Channel 6 Shadow Address Register	0000 0558$_H$	Page 3-49
DMA_CHSR07	DMA Channel 7 Status Register	0000 0560$_H$	Page 3-42
DMA_CHCR07	DMA Channel 7 Control Register	0000 0564$_H$	Page 3-39
DMA_MEM13	DMA Memory 13 Register	0000 0568$_H$	-
DMA_ADRCR07	DMA Channel 7 Address Control Register	0000 056C$_H$	Page 3-43
DMA_SADR07	DMA Channel 7 Source Address Register	0000 0570$_H$	Page 3-47
DMA_DADR07	DMA Channel 7 Destination Address Register	0000 0574$_H$	Page 3-48

Table 8-3 DMA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_SHADR07	DMA Channel 7 Shadow Address Register	$0000\ 0578_H$	Page 3-49
Reserved	Reserved	$0000\ 0580_H$ - $0000\ 05FF_H$	-

Table 8-4 SCU Registers

Register Short Name	Register Long Name	Address	Description see
SCU_OSCCON	SCU Oscillator Control Register	$0000\ 0800_H$	Page 2-14
SCU_PLLCON	SCU PLL Control Register	$0000\ 0804_H$	Page 2-15
Reserved	Reserved	$0000\ 0808_H$ - $0000\ 081C_H$	-
SCU_SYSCON	SCU System Control Register	$0000\ 0820_H$	Page 2-16
Reserved	Reserved	$0000\ 0824_H$ - $0000\ 082C_H$	-
SCU_CHTR0	SCU Channel Trigger 0 Register	$0000\ 0830_H$	Page 2-22
SCU_CHTR1	SCU Channel Trigger 1 Register	$0000\ 0834_H$	Page 2-22
SCU_CHTR2	SCU Channel Trigger 2 Register	$0000\ 0838_H$	Page 2-22
SCU_CHTR3	SCU Channel Trigger 3 Register	$0000\ 083C_H$	Page 2-22
SCU_CHTR4	SCU Channel Trigger 4 Register	$0000\ 0840_H$	Page 2-22
SCU_CHTR5	SCU Channel Trigger 5 Register	$0000\ 0844_H$	Page 2-22
SCU_CHTR6	SCU Channel Trigger 6 Register	$0000\ 0848_H$	Page 2-22
SCU_CHTR7	SCU Channel Trigger 7 Register	$0000\ 084C_H$	Page 2-22
SCU_ETCTR	SCU External Trigger Control Register	$0000\ 0850_H$	Page 2-18
Reserved	Reserved	$0000\ 0854_H$	-
SCU_SRCR	SCU Service Request Control Register	$0000\ 0858_H$	Page 2-19
SCU_ERRCUM	SCU Cumulative Error Register	$0000\ 085C_H$	Page 5-20

Table 8-4 SCU Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
IDCHIP	Chip Identification Register	0000 0860$_H$	Page 2-24
Reserved	Reserved	0000 0864$_H$ - 0000 08FF$_H$	-

Table 8-5 SSC Registers

Register Short Name	Register Long Name	Address	Description see
Reserved	Reserved	0000 0900$_H$ - 0000 090C$_H$	-
SSC_CON	SSC Control Register	0000 0910$_H$	Page 5-15
SSC_BR	SSC Baud Rate Timer Reload Register	0000 0914$_H$	Page 5-22
Reserved	Reserved	0000 0918$_H$ - 0000 091C$_H$	-
SSC_TB	SSC Transmit Buffer Register	0000 0920$_H$	Page 5-21
SSC_RB	SSC Receive Buffer Register	0000 0924$_H$	Page 5-22
SSC_STAT	SSC Status Register	0000 0928$_H$	Page 5-18
SSC_EFM	SSC Error Flag Modification Register	0000 092C$_H$	Page 5-19
Reserved	Reserved	0000 0930$_H$ - 0000 09FF$_H$	-

Table 8-6 Port Registers

Register Short Name	Register Long Name	Address	Description see
P0_OUT	Port 0 Output Register	0000 0A00$_H$	Page 7-6
P0_OMR	Port 0 Output Modification Register	0000 0A04$_H$	Page 7-7

Table 8-6 Port Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
Reserved	Reserved	0000 0A08$_H$ - 0000 0A0C$_H$	-
P0_IOCR0	Port 0 Input/Output Control Register 0	0000 0A10$_H$	**Page 7-8**
P0_IOCR4	Port 0 Input/Output Control Register 4	0000 0A14$_H$	**Page 7-9**
P0_IOCR8	Port 0 Input/Output Control Register 8	0000 0A18$_H$	**Page 7-10**
P0_IOCR12	Port 0 Input/Output Control Register 12	0000 0A1C$_H$	**Page 7-11**
Reserved	Reserved	0000 0A20$_H$	-
P0_IN	Port 0 Input Register	0000 0A24$_H$	**Page 7-5**
Reserved	Reserved	0000 0A28$_H$ - 0000 0A60$_H$	-
P1_IN	Port 1 Input Register	0000 0A64$_H$	**Page 7-15**
Reserved	Reserved	0000 0A68$_H$ - 0000 0AFF$_H$	-

Table 8-7 ADC Registers

Register Short Name	Register Long Name	Address	Description see
Reserved	Reserved	0000 1000$_H$ - 0000 100E$_H$	-
ADC_CON	ADC Control Register	0000 1010$_H$	**Page 6-21**
ADC_CON1	ADC Control 1 Register	0000 1012$_H$	**Page 6-23**
Reserved	Reserved	0000 1014$_H$ - 0000 101E$_H$	-
ADC_CTR2	ADC Control 2 Register	0000 1020$_H$	**Page 6-25**

Table 8-7 ADC Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
ADC_CTR2IN	ADC Injection Control 2 Register	0000 1022_H	Page 6-26
ADC_CTR0	ADC Control 0 Register	0000 1024_H	Page 6-24
Reserved	Reserved	0000 1026_H - 0000 $101E_H$	-
ADC_DAT	ADC Result Register	0000 1030_H	Page 6-27
ADC_DAT2	ADC Result 2 Register	0000 1032_H	Page 6-27
Reserved	Reserved	0000 1034_H - 0000 $10FE_H$	-
ADC_RESA0	ADC Extended Result 0 View A Register	0000 1100_H	Page 6-30
ADC_RESA1	ADC Extended Result 1 View A Register	0000 1104_H	Page 6-30
ADC_RESA2	ADC Extended Result 2 View A Register	0000 1108_H	Page 6-30
ADC_RESA3	ADC Extended Result 3 View A Register	0000 $110C_H$	Page 6-30
ADC_RESA4	ADC Extended Result 4 View A Register	0000 1110_H	Page 6-30
ADC_RESA5	ADC Extended Result 5 View A Register	0000 1114_H	Page 6-30
ADC_RESA6	ADC Extended Result 6 View A Register	0000 1118_H	Page 6-30
ADC_RESA7	ADC Extended Result 7 View A Register	0000 $111C_H$	Page 6-30
ADC_RESA8	ADC Extended Result 8 View A Register	0000 1120_H	Page 6-30
ADC_RESA9	ADC Extended Result 9 View A Register	0000 1124_H	Page 6-30
ADC_RESA10	ADC Extended Result 10 View A Register	0000 1128_H	Page 6-30

Table 8-7 ADC Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
ADC_RESA11	ADC Extended Result 11 View A Register	0000 112C$_H$	Page 6-30
ADC_RESA12	ADC Extended Result 12 View A Register	0000 1130$_H$	Page 6-30
ADC_RESA13	ADC Extended Result 13 View A Register	0000 1134$_H$	Page 6-30
ADC_RESA14	ADC Extended Result 14 View A Register	0000 1138$_H$	Page 6-30
ADC_RESA15	ADC Extended Result 15 View A Register	0000 113C$_H$	Page 6-30
ADC_RESB0	ADC Extended Result 0 View B Register	0000 1140$_H$	Page 6-32
ADC_RESB1	ADC Extended Result 1 View B Register	0000 1144$_H$	Page 6-32
ADC_RESB2	ADC Extended Result 2 View B Register	0000 1148$_H$	Page 6-32
ADC_RESB3	ADC Extended Result 3 View B Register	0000 114C$_H$	Page 6-32
ADC_RESB4	ADC Extended Result 4 View B Register	0000 1150$_H$	Page 6-32
ADC_RESB5	ADC Extended Result 5 View B Register	0000 1154$_H$	Page 6-32
ADC_RESB6	ADC Extended Result 6 View B Register	0000 1158$_H$	Page 6-32
ADC_RESB7	ADC Extended Result 7 View B Register	0000 115C$_H$	Page 6-32
ADC_RESB8	ADC Extended Result 8 View B Register	0000 1160$_H$	Page 6-32
ADC_RESB9	ADC Extended Result 9 View B Register	0000 1164$_H$	Page 6-32
ADC_RESB10	ADC Extended Result 10 View B Register	0000 1168$_H$	Page 6-32
ADC_RESB11	ADC Extended Result 11 View B Register	0000 116C$_H$	Page 6-32

Table 8-7 ADC Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
ADC_RESB12	ADC Extended Result 12 View B Register	0000 1170$_H$	Page 6-32
ADC_RESB13	ADC Extended Result 13 View B Register	0000 1174$_H$	Page 6-32
ADC_RESB14	ADC Extended Result 14 View B Register	0000 1178$_H$	Page 6-32
ADC_RESB15	ADC Extended Result 15 View B Register	0000 117C$_H$	Page 6-32
ADC_INRES	ADC Input Result Register	0000 1180$_H$	Page 6-33
ADC_DBCTR	ADC Doorbell Control Register	0000 1184$_H$	Page 6-35
ADC_RESV	ADC Result Valid Register	0000 1188$_H$	Page 6-34
Reserved	Reserved	0000 118C$_H$ - 0000 11FF$_H$	-

8.3 Memory Registers

Within the DMA and the MLI address area there are some memory register defined in Table 8-2 and Table 8-3. These registers can be used as memory registers if needed. All memory registers in Table 8-8 are 32-bit register that can be read and written from all masters. All memory registers in Table 8-9 are 16-bit register that can be read and written from all masters. Register DMA_MEM14 in Table 8-10 is a 8-bit register that can be read and written from all masters.

Table 8-8 32-Bit Memory Registers

Register Short Name	Register Long Name	Address	Description see
MLI_MEM0	MLI Memory 0 Register	0000 02B8$_H$	-
MLI_MEM1	MLI Memory 1 Register	0000 02BC$_H$	-
DMA_MEM0	DMA Memory 0 Register	0000 043C$_H$	-
DMA_MEM1	DMA Memory 1 Register	0000 0440$_H$	-
DMA_MEM2	DMA Memory 2 Register	0000 0444$_H$	-
DMA_MEM3	DMA Memory 3 Register	0000 0448$_H$	-

Table 8-8 32-Bit Memory Registers (cont'd)

Register Short Name	Register Long Name	Address	Description see
DMA_MEM4	DMA Memory 4 Register	0000 044C$_H$	-
DMA_MEM5	DMA Memory 5 Register	0000 0450$_H$	-

Table 8-9 16-Bit Memory Registers

Register Short Name	Register Long Name	Address	Description see
DMA_MEM6	DMA Memory 6 Register	0000 0488$_H$	-
DMA_MEM7	DMA Memory 7 Register	0000 04A8$_H$	-
DMA_MEM8	DMA Memory 8 Register	0000 04C8$_H$	-
DMA_MEM9	DMA Memory 9 Register	0000 04E8$_H$	-
DMA_MEM10	DMA Memory 10 Register	0000 0508$_H$	-
DMA_MEM11	DMA Memory 11 Register	0000 0528$_H$	-
DMA_MEM12	DMA Memory 12 Register	0000 0548$_H$	-
DMA_MEM13	DMA Memory 13 Register	0000 0568$_H$	-

Table 8-10 8-Bit Memory Registers

Register Short Name	Register Long Name	Address	Description see
DMA_MEM14	DMA Memory 14 Register	0000 0468$_H$	-

Keyword Index

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG