Features

- High Performance, Low Power AVR®32 UC 32-Bit Microcontroller
  - Compact Single-cycle RISC Instruction Set Including DSP Instruction Set
  - Read-Modify-Write Instructions and Atomic Bit Manipulation
  - Up to 60 MHz Clock Frequency with 1.24 DMIPS/MHz
  - Memory Protection Unit
- Multi-hierarchy Bus System
  - High-Performance Data Transfers on Separate Buses for Increased Performance
  - 7 Peripheral DMA Channels for Automatic Data Transfer
- Internal High-Speed Flash
  - 256K Bytes, 128K Bytes, 64K Bytes Versions
  - Single Cycle Access up to 30 MHz
  - Prefetch Buffer Optimizing Instruction Execution at Maximum Speed
  - 1ms Page Programming Time and 2ms Full-Chip Erase Time
  - 10,000 Write Cycles, 10-year Data Retention Capability
  - Flash Security Locks and User Defined Configuration Area
- Internal High-Speed SRAM, Single-Cycle Access at Full Speed
  - 32K Bytes (256KB and 128KB Flash), 16K Bytes (64KB Flash)
- Interrupt Controller
  - Autovectored Low Latency Interrupt Service with Programmable Priority
- System Functions
  - Power and Clock Manager Including Internal RC Clock and One 32KHz Oscillator
  - Two Multipurpose Oscillators and Two Phase-Lock-Loop (PLL)
  - Watchdog Timer, Real-Time Clock Timer
- Universal Serial Bus (USB)
  - Device 2.0 Full/Low Speed and On-The-Go (OTG)
  - Flexible End-Point Configuration and Management with Dedicated DMA Channels
  - On-chip Transceivers Including Pull-Ups
  - USB Wake Up functionality
- One Three-Channel 16-bit Timer/Counter (TC)
  - Three External Clock Inputs, PWM, Capture and Various Counting Capabilities
- One 7-Channel 16-bit Pulse Width Modulation Controller (PWM)
- Three Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Independant Baudrate Generator, Support for IrDA and ISO7816 interfaces
  - Support for Hardware Handshaking, RS485 Interfaces and Modem Line
- One Master/Slave Serial Peripheral Interfaces (SPI) with Chip Select Signals
- One Synchronous Serial Protocol Controller
  - Supports I2S and Generic Frame-Based Protocols
- One Master/Slave Two-Wire Interface (TWI), 400kbit/s I2C-compatible
- One 8-channel 10-bit Analog-To-Digital Converter
- On-Chip Debug System (JTAG interface)
  - Nexus Class 2+, Runtime Control, Non-Intrusive Data and Program Trace
- 64-pin TQFP/QFN (44 GPIO pins), 48-pin TQFP/QFN (28 GPIO pins)
- 5V Input Tolerant I/Os, including 4 high-drive pins.
- Single 3.3V Power Supply

AVR®32 32-Bit Microcontroller

AT32UC3B0256
AT32UC3B0128
AT32UC3B064
AT32UC3B1256
AT32UC3B1128
AT32UC3B164

Preliminary Datasheet

32059B–AVR32–07/07

1. Description

The AT32UC3B is a complete System-On-Chip microcontroller based on the AVR32 UC RISC processor running at frequencies up to 60 MHz. AVR32 UC is a high-performance 32-bit RISC microprocessor core, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption, high code density and high performance.

The processor implements a Memory Protection Unit (MPU) and a fast and flexible interrupt controller for supporting modern operating systems and real-time operating systems.

Higher computation capabilities are achievable using a rich set of DSP instructions.

The AT32UC3B incorporates on-chip Flash and SRAM memories for secure and fast access.

The Peripheral Direct Memory Access controller enables data transfers between peripherals and memories without processor involvement. PDCA drastically reduces processing overhead when transferring continuous and large data streams between modules within the MCU.

The Power Manager improves design flexibility and security: the on-chip Brown-Out Detector monitors the power supply, the CPU runs from the on-chip RC oscillator or from one of external oscillator sources, a Real-Time Clock and its associated timer keeps track of the time.

The Timer/Counter includes three identical 16-bit timer/counter channels. Each channel can be independently programmed to perform frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

The PWM modules provides seven independent channels with many configuration options including polarity, edge alignment and waveform non overlap control. One PWM channel can trigger ADC conversions for more accurate close loop control implementations.

The AT32UC3B also features many communication interfaces for communication intensive applications. In addition to standard serial interfaces like UART, SPI or TWI, other interfaces like flexible Synchronous Serial Controller and USB are available.

The Synchronous Serial Controller provides easy access to serial communication protocols and audio standards like I2S.

The Full-Speed USB 2.0 Device interface supports several USB Classes at the same time thanks to the rich End-Point configuration. The On-The-GO (OTG) Host interface allows device like a USB Flash disk or a USB printer to be directly connected to the processor.

AT32UC3B integrates a class 2+ Nexus 2.0 On-Chip Debug (OCD) System, with non-intrusive real-time trace, full-speed read/write memory access in addition to basic runtime control.

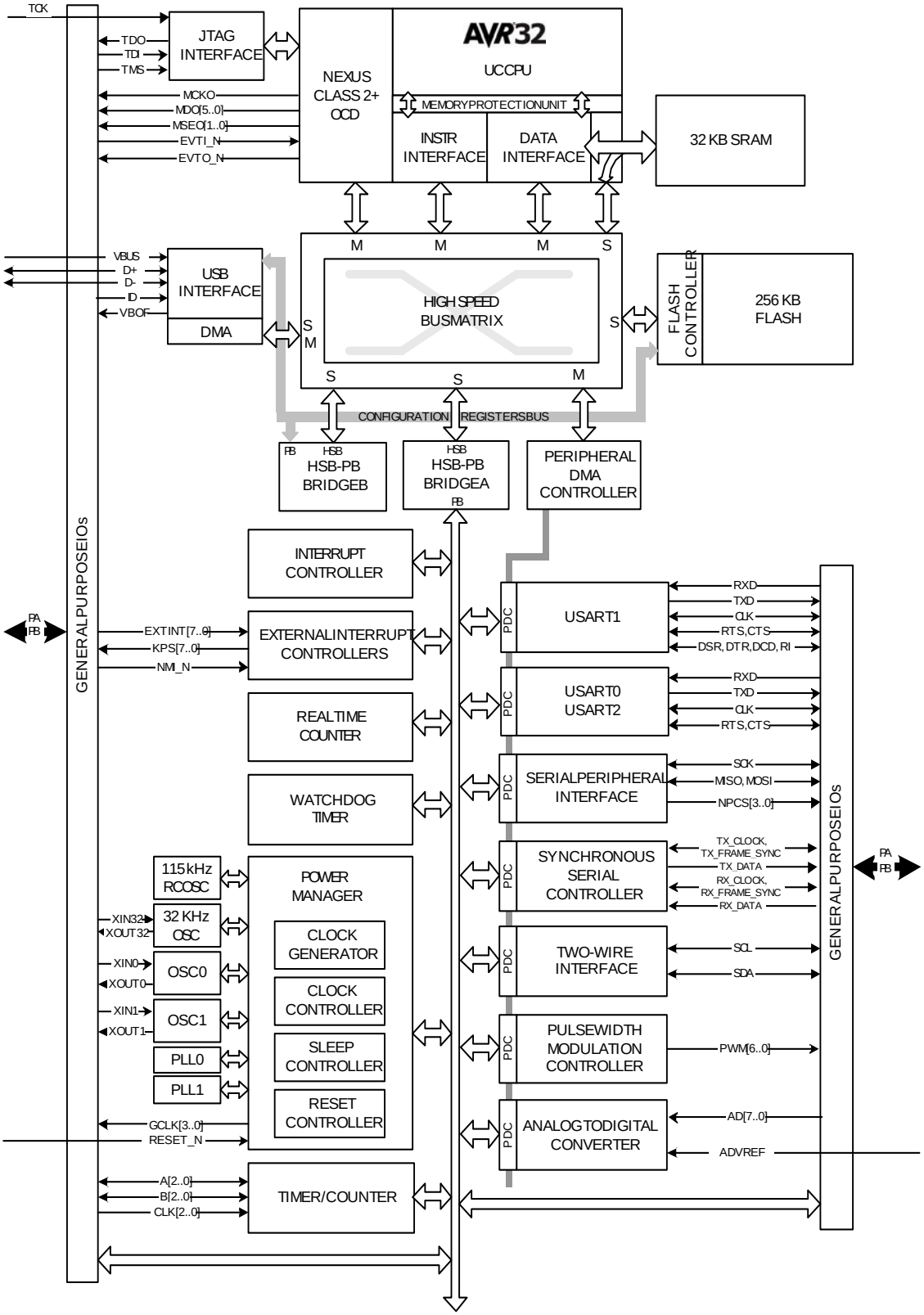
2. Configuration Summary

The table below lists all AT32UC3B memory and package configurations:

Device	Flash	SRAM	USART	SSC	ADC	OSC	USB Configuration	Package
AT32UC3B0256	256 Kbytes	32 Kbytes	3	1	8	2	Mini-Host + Device	64 lead TQFP/QFN
AT32UC3B0128	128 Kbytes	32 Kbytes	3	1	8	2	Mini-Host + Device	64 lead TQFP/QFN
AT32UC3B064	64 Kbytes	16 Kbytes	3	1	8	2	Mini-Host + Device	64 lead TQFP/QFN
AT32UC3B1256	256 Kbytes	32 Kbytes	2	0	6	1	Device	48 lead TQFP/QFN
AT32UC3B1128	128 Kbytes	32 Kbytes	2	0	6	1	Device	48 lead TQFP/QFN
AT32UC3B164	64 Kbytes	16 Kbytes	2	0	6	1	Device	48 lead TQFP/QFN

3. Blockdiagram

Figure 3-1. Block diagram



3.1 Processor and architecture

3.1.1 AVR32uC CPU

- 32-bit load/store AVR32B RISC architecture.
  - 15 general-purpose 32-bit registers.
  - 32-bit Stack Pointer, Program Counter and Link Register reside in register file.
  - Fully orthogonal instruction set.
  - Privileged and unprivileged modes enabling efficient and secure Operating Systems.
  - Innovative instruction set together with variable instruction length ensuring industry leading code density.
  - DSP extention with saturating arithmetic, and a wide variety of multiply instructions.
- 3 stage pipeline allows one instruction per clock cycle for most instructions.
  - Byte, half-word, word and double word memory access.
  - Multiple interrupt priority levels.
- MPU allows for operating systems with memory protection.

3.1.2 Debug and Test system

- IEEE®1149.1 compliant JTAG and boundary scan
- Direct memory access and programming capabilities through JTAG interface
- Extensive On-Chip Debug features in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+
  - Low-cost NanoTrace supported.
- Auxiliary port for high-speed trace information
- Hardware support for 6 Program and 2 data breakpoints
- Unlimited number of software breakpoints supported
- Advanced Program, Data, Ownership, and Watchpoint trace supported

3.1.3 Peripheral DMA Controller

- Transfers from/to peripheral to/from any memory space without intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- 7 channels that can be dynamically attributed to
  - all USARTs
  - the Serial Synchronous Controller
  - the Serial Peripheral Interface
  - the ADC
  - the TWI Interface

3.1.4 Bus system

- High Speed Bus (HSB) matrix with 5 Masters and 5 Slaves handled
  - Handles Requests from the CPU Data Fetch, CPU Instruction Fetch, PDCA, USBB, CPU SAB, and to internal Flash, internal SRAM, Peripheral Bus A, Peripheral Bus B, USBB.
  - Round-Robin Arbitration (three modes supported: no default master, last accessed default master, fixed default master)
  - Burst Breaking with Slot Cycle Limit
  - One Address Decoder Provided per Master
  - Peripheral Bus A able to run on at divided bus speeds compared to the High Speed Bus
  - All modules connected to the same bus use the same clock, but the clock to each module can be individually shut off by the Power Manager.

4. Signals Description

The following table gives details on the signal name classified by peripheral

The signals are multiplexed with GPIO pins as described in ”Peripheral Multiplexing on IO lines” on page 39.

Table 4-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power				
VDDSYS	Power supply for PLL and ADC	Power		1.65 to 1.95 V
VDDCORE	Core Power Supply	Power		1.65 to 1.95 V
VDDIO	I/O Power Supply	Power		3.0 to 3.6V
VDDANA	Analog Power Supply	Power		3.0 to 3.6V
VDDIN	Voltage Regulator Input Supply	Power		3.0 to 3.6V
VDDOUT	Voltage Regulator Output	Power Output		1.65 to 1.95 V
GNDANA	Analog Ground	Ground		
GND	Ground	Ground		
Clocks, Oscillators, and PLL’s				
XIN0, XIN1, XIN32	Crystal 0, 1, 32 Input	Analog		
XOUT0, XOUT1, XOUT32	Crystal 0, 1, 32 Output	Analog		
JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		
Auxiliary Port - AUX				
MCKO	Trace Data Output Clock	Output		
MDO0 - MDO5	Trace Data Output	Output		
MSEO0 - MSEO1	Trace Frame Control	Output		
EVTI_N	Event In	Output	Low	
EVTO_N	Event Out	Output	Low	

Table 4-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power Manager - PM				
GCLK0 - GCLK2	Generic Clock Pins	Output		
RESET_N	Reset Pin	Input	Low	
External Interrupt Module - EIM				
EXTINT0 - EXTINT7	External Interrupt Pins	Input		
KPS0 - KPS7	Keypad Scan Pins	Output		
NMI_N	Non-Maskable Interrupt Pin	Input	Low	
General Purpose Input/Output 2 - GPIOA, GPIOB				
P0 - P31	Parallel I/O Controller GPIOA	I/O		
P0 - P11	Parallel I/O Controller GPIOB	I/O		
Serial Peripheral Interface - SPI0				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
NPCS0 - NPCS3	SPI Peripheral Chip Select	I/O	Low	
SCK	Clock	Output		
Synchronous Serial Controller - SSC				
RX_CLOCK	SSC Receive Clock	I/O		
RX_DATA	SSC Receive Data	Input		
RX_FRAME_SYNC	SSC Receive Frame Sync	I/O		
TX_CLOCK	SSC Transmit Clock	I/O		
TX_DATA	SSC Transmit Data	Output		
TX_FRAME_SYNC	SSC Transmit Frame Sync	I/O		
Timer/Counter - TIMER				
A0	Channel 0 Line A	I/O		
A1	Channel 1 Line A	I/O		
A2	Channel 2 Line A	I/O		
B0	Channel 0 Line B	I/O		
B1	Channel 1 Line B	I/O		

Table 4-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
B2	Channel 2 Line B	I/O		
CLK0	Channel 0 External Clock Input	Input		
CLK1	Channel 1 External Clock Input	Input		
CLK2	Channel 2 External Clock Input	Input		
Two-wire Interface - TWI				
SCL	Serial Clock	I/O		
SDA	Serial Data	I/O		
Universal Synchronous Asynchronous Receiver Transmitter - USART0, USART1, USART2				
CLK	Clock	I/O		
CTS	Clear To Send	Input		
DCD	Data Carrier Detect			Only USART1
DSR	Data Set Ready			Only USART1
DTR	Data Terminal Ready			Only USART1
RI	Ring Indicator			Only USART1
RTS	Request To Send	Output		
RXD	Receive Data	Input		
RXDN	Inverted Receive Data	Input	Low	
TXD	Transmit Data	Output		
TXDN	Inverted Transmit Data	Output	Low	
Analog to Digital Converter - ADC				
AD0 - AD7	Analog input pins	Analog input		
ADVREF	Analog positive reference voltage input	Analog input		2.6 to 3.6V
Pulse Width Modulator - PWM				
PWM0 - PWM6	PWM Output Pins	Output		
Universal Serial Bus Device - USB				
DDM	USB Device Port Data -	Analog		
DDP	USB Device Port Data +	Analog		

5. Power Considerations

5.1 Power Supplies

The AT32UC3B has several types of power supply pins:

- **VDDIO: Powers I/O lines. Voltage is 3.3V nominal.**
- **VDDANA: Powers the ADC Voltage is 3.3V nominal.**
- **VDDIN: Input voltage for the voltage regulator. Voltage is 3.3V nominal.**
- **VDDCORE: Powers the core, memories, and peripherals. Voltage is 1.8V nominal.**
- **VDDSYS: Powers the PLL and ADC. Voltage is 1.8V nominal.**

The ground pins GND are common to VDDCORE and VDDIO. The ground pin for VDDANA and VDDSYS is GNDANA.

See "Electrical Characteristics" on page 554 for power consumption on the various supply pins.

5.2 Voltage Regulator

The AT32UC3B embeds a voltage regulator that converts from 3.3V to 1.8V with a load of up to 100 mA. The regulator takes its input voltage from VDDIN, and supplies the output voltage on VDDOUT. VDDOUT should be externally connected to the 1.8V domains to be powered.

Adequate output supply decoupling is mandatory for VDDOUT to reduce ripple and avoid oscillations. The best way to achieve this is to use two capacitors in parallel: one external 470 pF (or 1 nF) NPO capacitor should be connected between VDDOUT and GND as close to the chip as possible. One external 2.2 µF (or 3.3 µF) X7R capacitor should be connected between VDDOUT and GND.

Adequate input supply decoupling is mandatory for VDDIN in order to improve startup stability and reduce source voltage drop. The input decoupling capacitor should be placed close to the chip. For example, two capacitors can be used in parallel: 100 nF NPO and 4.7 µF X7R.

6. Package and Pinout

The device pins are multiplexed with peripheral functions as described in "Peripheral Multiplexing on IO lines" on page 39.

Figure 6-1. QFP64 Pinout

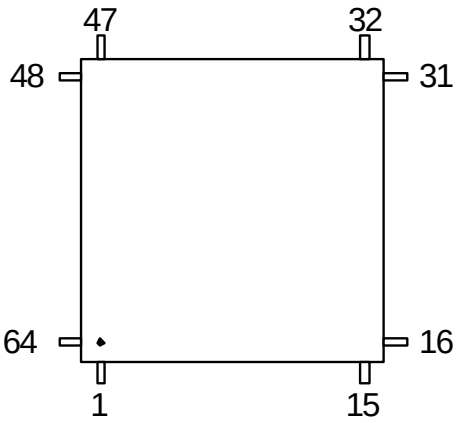


Table 6-1. QFP64 Package Pinout

1	GND	**17**	GND	**33**	PA13	**49**	GND
2	TCK	**18**	ADVREF	**34**	PA14	**50**	DP
3	TDI	**19**	VDDANA	**35**	PA15	**51**	DM
4	TDO	**20**	VDDOUT	**36**	PA16	**52**	VBUS
5	TMS	**21**	VDDIN	**37**	PA17	**53**	VDDPLL1
6	PB00	**22**	VDDCORE	**38**	PB06	**54**	PB08
7	PB01	**23**	GND	**39**	PA18	**55**	PB09
8	VDDCORE	**24**	PB02	**40**	PA19	**56**	VDDCORE
9	PA03	**25**	PB03	**41**	PA28	**57**	PB10
10	PA04	**26**	PB04	**42**	PA29	**58**	PB11
11	PA05	**27**	PB05	**43**	PB07	**59**	PA24
12	PA06	**28**	PA09	**44**	PA20	**60**	PA25
13	PA07	**29**	PA10	**45**	PA21	**61**	PA26
14	PA08	**30**	PA11	**46**	PA22	**62**	PA27
15	PA30	**31**	PA12	**47**	PA23	**63**	RESET_N
16	PA31	**32**	VDDIO	**48**	VDDIO	**64**	VDDIO

Figure 6-2. QFP48 Pinout

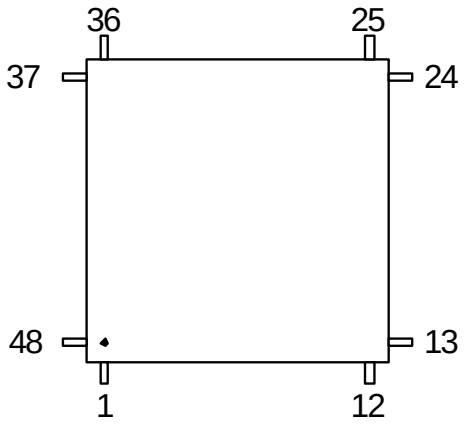


Table 6-2. QFP48 Package Pinout

1	GND	**13**	GND	**25**	PA13	**37**	GND
2	TCK	**14**	ADVREF	**26**	PA14	**38**	DP
3	TDI	**15**	VDDANA	**27**	PA15	**39**	DM
4	TDO	**16**	VDDOUT	**28**	PA16	**40**	VBUS
5	TMS	**17**	VDDIN	**29**	PA17	**41**	VDDPLL1
6	VDDCORE	**18**	VDDCORE	**30**	PA18	**42**	VDDCORE
7	PA03	**19**	GND	**31**	PA19	**43**	PA24
8	PA04	**20**	PA09	**32**	PA20	**44**	PA25
9	PA05	**21**	PA10	**33**	PA21	**45**	PA26
10	PA06	**22**	PA11	**34**	PA22	**46**	PA27
11	PA07	**23**	PA12	**35**	PA23	**47**	RESET_N
12	PA08	**24**	VDDIO	**36**	VDDIO	**48**	VDDIO

7. I/O Line Considerations

7.1 JTAG pins

TMS and TDI pins have pull-up resistors. TDO pin is an output, driven at up to VDDIO, and has no pull-up resistor. These 3 pins can be used as GPIO-pins. At reset state, these pins are in GPIO mode.

TCK pin cannot be used as GPIO pin. JTAG interface is enabled when TCK pin is tied low. This pins must be pulled-up externally on application board.

7.2 RESET_N pin

The RESET_N pin is a schmitt input and integrates a permanent pull-up resistor to VDDIO. As the product integrates a power-on reset cell, the RESET_N pin can be left unconnected in case no reset from the system needs to be applied to the product.

7.3 TWI pins

When these pins are used for TWI, the pins are open-drain outputs with slew-rate limitation and inputs with inputs with spike-filtering. When used as GPIO-pins or used for other peripherals, the pins have the same characteristics as PIO pins.

7.4 GPIO pins

All the I/O lines integrate a programmable pull-up resistor. Programming of this pull-up resistor is performed independently for each I/O line through the GPIO Controllers. After reset, I/O lines default as inputs with pull-up resistors disabled, except when indicated otherwise in the column “Reset State” of the GPIO Controller multiplexing tables.

7.5 High drive pins

The four pins PA20, PA21, PA22, PA23 have high drive output capabilities.

8. Processor and Architecture

This chapter gives an overview of the AVR32 UC CPU. AVR32 UC is an implementation of the AVR32 architecture. A summary of the programming model, instruction set and MPU is presented. For further details, see the *AVR32 Architecture Manual* and the *AVR32 UC Technical Reference Manual*.

8.1 AVR32 Architecture

AVR32 is a new, high-performance 32-bit RISC microprocessor architecture, designed for cost-sensitive embedded applications, with particular emphasis on low power consumption and high code density. In addition, the instruction set architecture has been tuned to allow a variety of microarchitectures, enabling the AVR32 to be implemented as low-, mid- or high-performance processors. AVR32 extends the AVR family into the world of 32- and 64-bit applications.

Through a quantitative approach, a large set of industry recognized benchmarks has been compiled and analyzed to achieve the best code density in its class. In addition to lowering the memory requirements, a compact code size also contributes to the core's low power characteristics. The processor supports byte and half-word data types without penalty in code size and performance.

Memory load and store operations are provided for byte, half-word, word and double word data with automatic sign- or zero extension of half-word and byte data. The C-compiler is closely linked to the architecture and is able to exploit code optimization features, both for size and speed.

In order to reduce code size to a minimum, some instructions have multiple addressing modes. As an example, instructions with immediates often have a compact format with a smaller immediate, and an extended format with a larger immediate. In this way, the compiler is able to use the format giving the smallest code size.

Another feature of the instruction set is that frequently used instructions, like add, have a compact format with two operands as well as an extended format with three operands. The larger format increases performance, allowing an addition and a data move in the same instruction in a single cycle. Load and store instructions have several different formats in order to reduce code size and speed up execution.

The register file is organized as sixteen 32-bit registers and includes the Program Counter, the Link Register, and the Stack Pointer. In addition, register R12 is designed to hold return values from function calls and is used implicitly by some instructions.

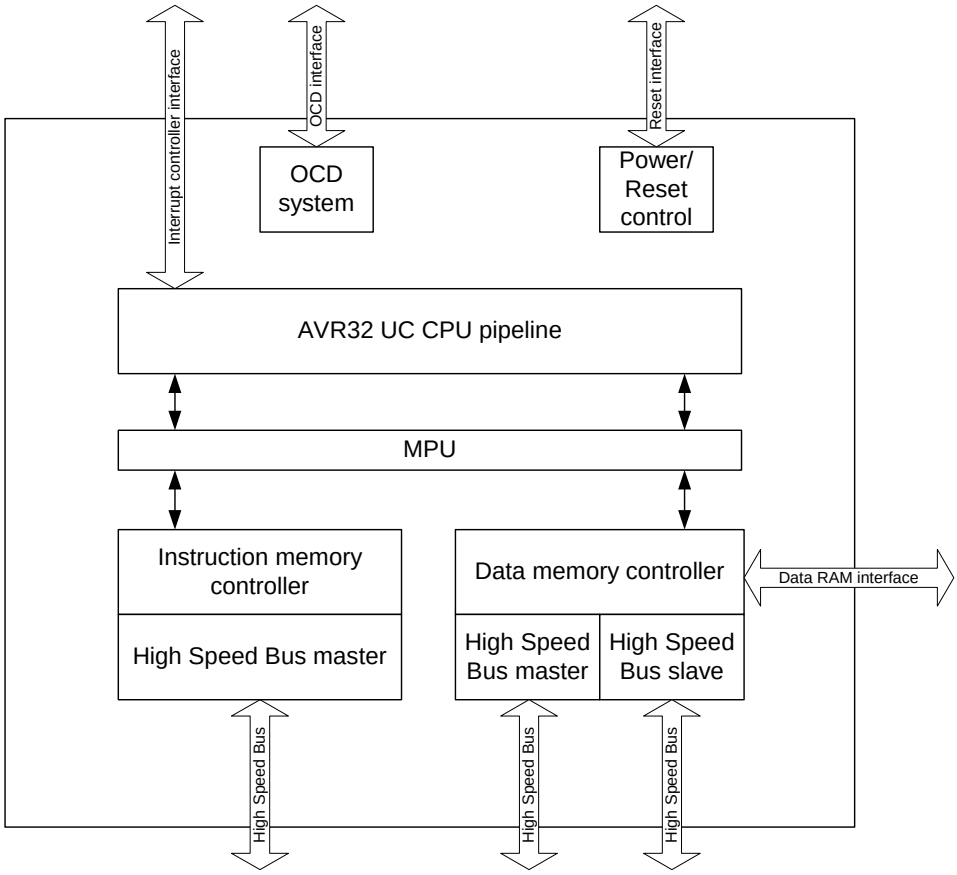
8.2 The AVR32 UC CPU

AVR32 UC implements the AVR32A architecture. This implementation targets low- and medium-performance applications, and provides an advanced OCD system, no caches, and a Memory Protection Unit (MPU). Java acceleration hardware is not implemented.

AVR32 UC provides three memory interfaces, one High Speed Bus master for instruction fetch, one High Speed Bus master for data access, and one High Speed Bus slave interface allowing other bus masters to access data RAMs internal to the CPU. Keeping data RAMs internal to the CPU allows fast access to the RAMs, reduces latency and guarantees deterministic timing. Also, power consumption is reduced by not needing a full High Speed Bus access for memory accesses. A dedicated data RAM interface is provided for communicating with the internal data RAMs.

Figure 8-1 on page 16 displays the contents of AVR32 UC.

Figure 8-1. Overview of the AVR32 UC CPU



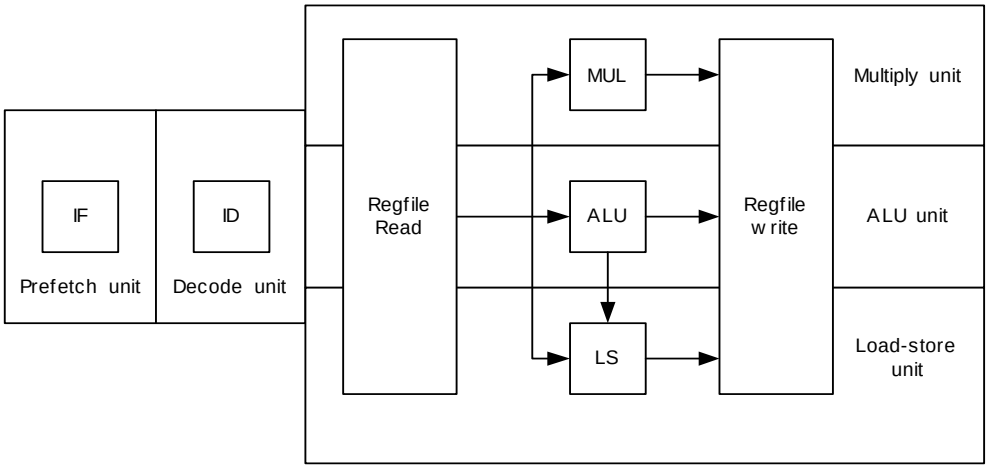
8.2.1 Pipeline Overview

AVR32 UC is a pipelined processor with three pipeline stages. There are three pipeline stages, Instruction Fetch (IF), Instruction Decode (ID) and Instruction Execute (EX). The EX stage is split into three parallel subsections, one arithmetic/logic (ALU) section, one multiply (MUL) section and one load/store (LS) section.

Instructions are issued and complete in order. Certain operations require several clock cycles to complete, and in this case, the instruction resides in the ID and EX stages for the required number of clock cycles. Since there is only three pipeline stages, no internal data forwarding is required, and no data dependencies can arise in the pipeline.

Figure 8-2 on page 17 shows an overview of the AVR32 UC pipeline stages.

Figure 8-2. The AVR32 UC Pipeline



8.2.2 AVR32A Microarchitecture Compliance

AVR32 UC implements an AVR32A microarchitecture. The AVR32A microarchitecture is targeted at cost-sensitive, lower-end applications like smaller microcontrollers. This microarchitecture does not provide dedicated hardware registers for shadowing of register file registers in interrupt contexts. Additionally, it does not provide hardware registers for the return address registers and return status registers. Instead, all this information is stored on the system stack. This saves chip area at the expense of slower interrupt handling.

Upon interrupt initiation, registers R8-R12 are automatically pushed to the system stack. These registers are pushed regardless of the priority level of the pending interrupt. The return address and status register are also automatically pushed to stack. The interrupt handler can therefore use R8-R12 freely. Upon interrupt completion, the old R8-R12 registers and status register are restored, and execution continues at the return address stored popped from stack.

The stack is also used to store the status register and return address for exceptions and *scall*. Executing the *rete* or *rets* instruction at the completion of an exception or system call will pop this status register and continue execution at the popped return address.

8.2.3 Java Support

AVR32 UC does not provide Java hardware acceleration.

8.2.4 Memory protection

The MPU allows the user to check all memory accesses for privilege violations. If an access is attempted to an illegal memory address, the access is aborted and an exception is taken. The MPU in AVR32 UC adheres to the MPU defined in the AVR32 architecture, and implements the following:

- 8 MPUAR registers, shared by both instruction and data accesses
- No MPUCRD/MPUCRI registers
- No MPUBRD register
- One MPUAPR register, shared by both instruction and data accesses
- One MPUCR register

8.2.5 Unaligned reference handling

AVR32 UC does not support unaligned accesses, except for doubleword accesses. AVR32 UC is able to perform word-aligned *st.d* and *ld.d*. Any other unaligned memory access will cause an address exception. Doubleword-sized accesses with word-aligned pointers will automatically be performed as two word-sized accesses.

The following table shows the instructions with support for unaligned addresses. All other instructions require aligned addresses.

Table 8-1. Instructions with unaligned reference support

Instruction	Supported alignment
ld.d	Word
st.d	Word

8.2.6 Unimplemented instructions

The following instructions are unimplemented in AVR32 UC, and will cause an Unimplemented Instruction Exception if executed:

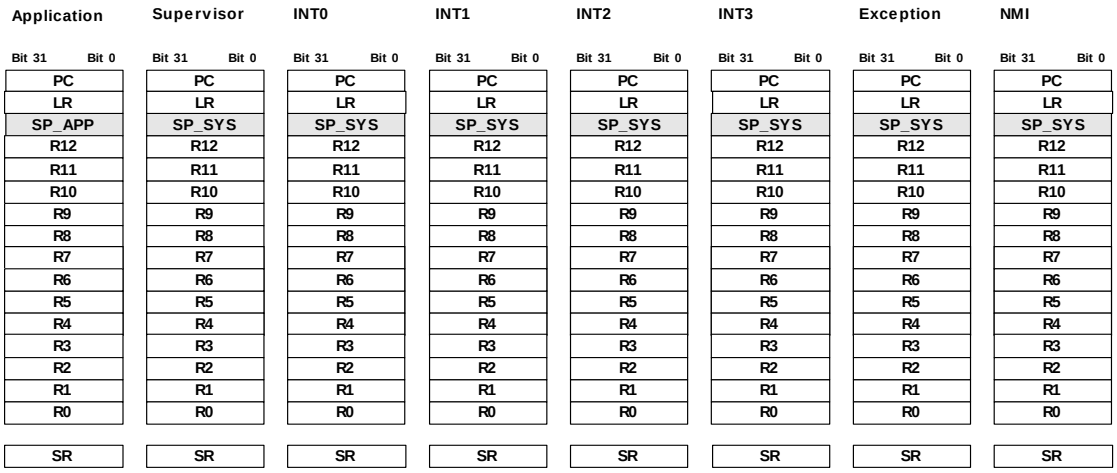
- All SIMD instructions
- All coprocessor instructions
- retj, incjosp, popjc, pushjc
- tlbr, tlbs, tlbw
- cache

8.3 Programming Model

8.3.1 Register file configuration

The AVR32A architecture dictates a specific register file implementation, reproduced below.

Figure 8-3. The AVR32 UC Register File



8.3.2 Status register configuration

The Status Register (SR) is split into two halfwords, one upper and one lower, see Figure 8-4 on page 19 and Figure 8-5 on page 20. The lower word contains the C, Z, N, V and Q condition code flags and the R, T and L bits, while the upper halfword contains information about the mode and state the processor executes in. Refer to the *AVR32 Architecture Manual* for details.

Figure 8-4. The Status Register High Halfword

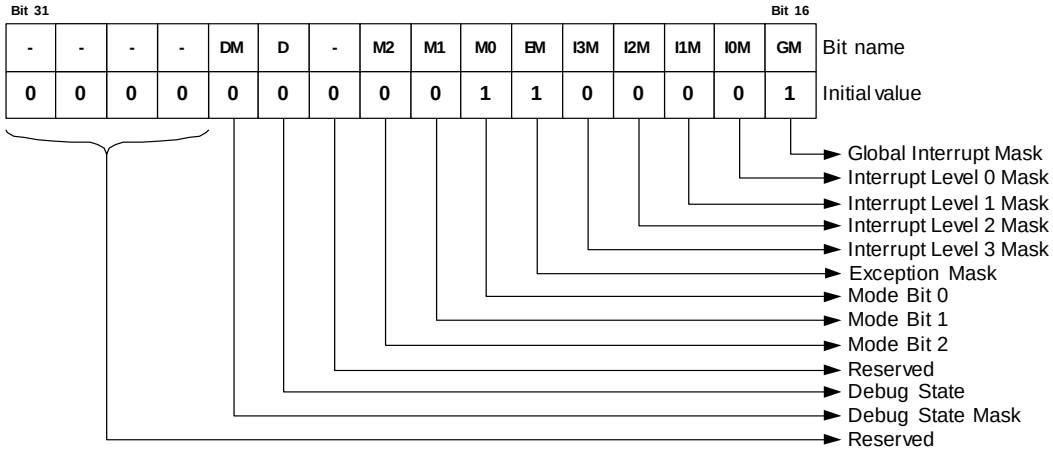
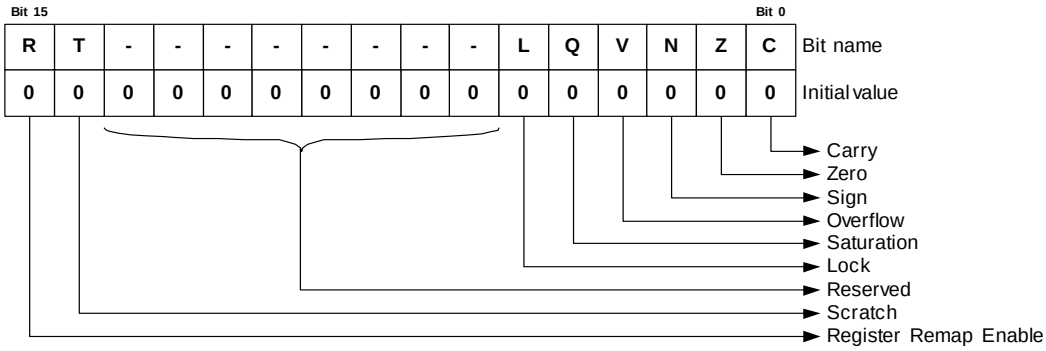


Figure 8-5. The Status Register Low Halfword



8.3.3 Processor States

8.3.3.1 Normal RISC State

The AVR32 processor supports several different execution contexts as shown in Table 8-2 on page 20.

Table 8-2. Overview of execution modes, their priorities and privilege levels.

Priority	Mode	Security	Description
1	Non Maskable Interrupt	Privileged	Non Maskable high priority interrupt mode
2	Exception	Privileged	Execute exceptions
3	Interrupt 3	Privileged	General purpose interrupt mode
4	Interrupt 2	Privileged	General purpose interrupt mode
5	Interrupt 1	Privileged	General purpose interrupt mode
6	Interrupt 0	Privileged	General purpose interrupt mode
N/A	Supervisor	Privileged	Runs supervisor calls
N/A	Application	Unprivileged	Normal program execution mode

Mode changes can be made under software control, or can be caused by external interrupts or exception processing. A mode can be interrupted by a higher priority mode, but never by one with lower priority. Nested exceptions can be supported with a minimal software overhead.

When running an operating system on the AVR32, user processes will typically execute in the application mode. The programs executed in this mode are restricted from executing certain instructions. Furthermore, most system registers together with the upper halfword of the status register cannot be accessed. Protected memory areas are also not available. All other operating modes are privileged and are collectively called System Modes. They have full access to all privileged and unprivileged resources. After a reset, the processor will be in supervisor mode.

8.3.3.2 Debug State

The AVR32 can be set in a debug state, which allows implementation of software monitor routines that can read out and alter system information for use during application development. This implies that all system and application registers, including the status registers and program counters, are accessible in debug state. The privileged instructions are also available.

All interrupt levels are by default disabled when debug state is entered, but they can individually be switched on by the monitor routine by clearing the respective mask bit in the status register.

Debug state can be entered as described in the AVR32 UC *Technical Reference Manual*.

Debug state is exited by the *retd* instruction.

8.3.4 System registers

The system registers are placed outside of the virtual memory space, and are only accessible using the privileged *mfsr* and *mtsr* instructions. The table below lists the system registers specified in the AVR32 architecture, some of which are unused in AVR32 UC. The programmer is responsible for maintaining correct sequencing of any instructions following a *mtsr* instruction. For detail on the system registers, refer to the AVR32 UC *Technical Reference Manual*.

Table 8-3. System Registers

Reg #	Address	Name	Function
0	0	SR	Status Register
1	4	EVBA	Exception Vector Base Address
2	8	ACBA	Application Call Base Address
3	12	CPUCR	CPU Control Register
4	16	ECR	Exception Cause Register
5	20	RSR_SUP	Unused in AVR32 UC
6	24	RSR_INT0	Unused in AVR32 UC
7	28	RSR_INT1	Unused in AVR32 UC
8	32	RSR_INT2	Unused in AVR32 UC
9	36	RSR_INT3	Unused in AVR32 UC
10	40	RSR_EX	Unused in AVR32 UC
11	44	RSR_NMI	Unused in AVR32 UC
12	48	RSR_DBG	Return Status Register for Debug Mode
13	52	RAR_SUP	Unused in AVR32 UC
14	56	RAR_INT0	Unused in AVR32 UC
15	60	RAR_INT1	Unused in AVR32 UC
16	64	RAR_INT2	Unused in AVR32 UC
17	68	RAR_INT3	Unused in AVR32 UC
18	72	RAR_EX	Unused in AVR32 UC
19	76	RAR_NMI	Unused in AVR32 UC
20	80	RAR_DBG	Return Address Register for Debug Mode
21	84	JECR	Unused in AVR32 UC
22	88	JOSP	Unused in AVR32 UC
23	92	JAVA_LV0	Unused in AVR32 UC
24	96	JAVA_LV1	Unused in AVR32 UC
25	100	JAVA_LV2	Unused in AVR32 UC

Table 8-3. System Registers (Continued)

Reg #	Address	Name	Function
26	104	JAVA_LV3	Unused in AVR32 UC
27	108	JAVA_LV4	Unused in AVR32 UC
28	112	JAVA_LV5	Unused in AVR32 UC
29	116	JAVA_LV6	Unused in AVR32 UC
30	120	JAVA_LV7	Unused in AVR32 UC
31	124	JTBA	Unused in AVR32 UC
32	128	JBCR	Unused in AVR32 UC
33-63	132-252	Reserved	Reserved for future use
64	256	CONFIG0	Configuration register 0
65	260	CONFIG1	Configuration register 1
66	264	COUNT	Cycle Counter register
67	268	COMPARE	Compare register
68	272	TLBEHI	Unused in AVR32 UC
69	276	TLBELO	Unused in AVR32 UC
70	280	PTBR	Unused in AVR32 UC
71	284	TLBEAR	Unused in AVR32 UC
72	288	MMUCR	Unused in AVR32 UC
73	292	TLBARLO	Unused in AVR32 UC
74	296	TLBARHI	Unused in AVR32 UC
75	300	PCCNT	Unused in AVR32 UC
76	304	PCNT0	Unused in AVR32 UC
77	308	PCNT1	Unused in AVR32 UC
78	312	PCCR	Unused in AVR32 UC
79	316	BEAR	Bus Error Address Register
80	320	MPUARI0	MPU Address Register Instruction region 0
81	324	MPUARI1	MPU Address Register Instruction region 1
82	328	MPUARI2	MPU Address Register Instruction region 2
83	332	MPUARI3	MPU Address Register Instruction region 3
84	336	MPUARI4	MPU Address Register Instruction region 4
85	340	MPUARI5	MPU Address Register Instruction region 5
86	344	MPUARI6	MPU Address Register Instruction region 6
87	348	MPUARI7	MPU Address Register Instruction region 7
88	352	MPUARD0	Unused in AVR32 UC
89	356	MPUARD1	Unused in AVR32 UC
90	360	MPUARD2	Unused in AVR32 UC
91	364	MPUARD3	Unused in AVR32 UC

Table 8-3. System Registers (Continued)

Reg #	Address	Name	Function
92	368	MPUARD4	Unused in AVR32 UC
93	372	MPUARD5	Unused in AVR32 UC
94	376	MPUARD6	Unused in AVR32 UC
95	380	MPUARD7	Unused in AVR32 UC
96	384	MPUCRI	Unused in AVR32 UC
97	388	MPUCRD	Unused in AVR32 UC
98	392	MPUBRD	Unused in AVR32 UC
99	396	MPUAPRI	MPU Access Permission Register Instruction regions
100	400	MPUAPRD	Unused in AVR32 UC
101	404	MPUCR	MPU Control Register
102-191	408-764	Reserved	Reserved for future use
192-255	768-1020	IMPL	IMPLEMENTATION DEFINED

8.4 Exceptions and Interrupts

AVR32 UC incorporates a powerful exception handling scheme. The different exception sources, like Illegal Op-code and external interrupt requests, have different priority levels, ensuring a well-defined behavior when multiple exceptions are received simultaneously. Additionally, pending exceptions of a higher priority class may preempt handling of ongoing exceptions of a lower priority class.

When an event occurs, the execution of the instruction stream is halted, and execution control is passed to an event handler at an address specified in Table 8-4 on page 26. Most of the handlers are placed sequentially in the code space starting at the address specified by EVBA, with four bytes between each handler. This gives ample space for a jump instruction to be placed there, jumping to the event routine itself. A few critical handlers have larger spacing between them, allowing the entire event routine to be placed directly at the address specified by the EVBA-relative offset generated by hardware. All external interrupt sources have autovectored interrupt service routine (ISR) addresses. This allows the interrupt controller to directly specify the ISR address as an address relative to EVBA. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes. The target address of the event handler is calculated as (EVBA | event_handler_offset), not (EVBA + event_handler_offset), so EVBA and exception code segments must be set up appropriately. The same mechanisms are used to service all different types of events, including external interrupt requests, yielding a uniform event handling scheme.

An interrupt controller does the priority handling of the external interrupts and provides the autovector offset to the CPU.

8.4.1 System stack issues

Event handling in AVR32 UC, like in all AVR32A architectures, uses the system stack pointed to by the system stack pointer, SP_SYS, for pushing and popping R8-R12, LR, status register and return address. Since event code may be timing-critical, SP_SYS should point to memory addresses in the IRAM section, since the timing of accesses to this memory section is both fast and deterministic.

The user must also make sure that the system stack is large enough so that any event is able to push the required registers to stack. If the system stack is full, and an event occurs, the system will enter an UNDEFINED state.

8.4.2 Exceptions and interrupt requests

When an event other than *scall* or debug request is received by the core, the following actions are performed atomically:

1. The pending event will not be accepted if it is masked. The I3M, I2M, I1M, I0M, EM and GM bits in the Status Register are used to mask different events. Not all events can be masked. A few critical events (NMI, Unrecoverable Exception, TLB Multiple Hit and Bus Error) can not be masked. When an event is accepted, hardware automatically sets the mask bits corresponding to all sources with equal or lower priority. This inhibits acceptance of other events of the same or lower priority, except for the critical events listed above. Software may choose to clear some or all of these bits after saving the necessary state if other priority schemes are desired. It is the event source's responsability to ensure that their events are left pending until accepted by the CPU.
2. When a request is accepted, the Status Register and Program Counter of the current context is stored to the system stack. If the event is an INT0, INT1, INT2 or INT3, registers R8-R12 and LR are also automatically stored to stack. Storing the Status Register ensures that the core is returned to the previous execution mode when the current event handling is completed. When exceptions occur, both the EM and GM bits are set, and the application may manually enable nested exceptions if desired by clearing the appropriate bit. Each exception handler has a dedicated handler address, and this address uniquely identifies the exception source.
3. The Mode bits are set to reflect the priority of the accepted event, and the correct register file bank is selected. The address of the event handler, as shown in Table 8-4, is loaded into the Program Counter.

The execution of the event handler routine then continues from the effective address calculated.

The *rete* instruction signals the end of the event. When encountered, the Return Status Register and Return Address Register are popped from the system stack and restored to the Status Register and Program Counter. If the *rete* instruction returns from INT0, INT1, INT2 or INT3, registers R8-R12 and LR are also popped from the system stack. The restored Status Register contains information allowing the core to resume operation in the previous execution mode. This concludes the event handling.

8.4.3 Supervisor calls

The AVR32 instruction set provides a supervisor mode call instruction. The *scall* instruction is designed so that privileged routines can be called from any context. This facilitates sharing of code between different execution modes. The *scall* mechanism is designed so that a minimal execution cycle overhead is experienced when performing supervisor routine calls from time-critical event handlers.

The *scall* instruction behaves differently depending on which mode it is called from. The behaviour is detailed in the instruction set reference. In order to allow the *scall* routine to return to the correct context, a return from supervisor call instruction, *rets*, is implemented. In the AVR32A microarchitecture, *scall* and *rets* uses the system stack to store the return address and the status register.

8.4.4 Debug requests

The AVR32 architecture defines a dedicated debug mode. When a debug request is received by the core, Debug mode is entered. Entry into Debug mode can be masked by the DM bit in the status register. Upon entry into Debug mode, hardware sets the SR[D] bit and jumps to the Debug Exception handler. By default, debug mode executes in the exception context, but with dedicated Return Address Register and Return Status Register. These dedicated registers remove the need for storing this data to the system stack, thereby improving debuggability. The mode bits in the status register can freely be manipulated in Debug mode, to observe registers in all contexts, while retaining full privileges.

Debug mode is exited by executing the *retd* instruction. This returns to the previous context.

8.4.5 Entry points for events

Several different event handler entry points exists. In AVR32 UC, the reset address is 0x8000_0000. This places the reset address in the boot flash memory area.

TLB miss exceptions (which are unused in AVR32 UC since it has no MMU) and *scall* have a dedicated space relative to EVBA where their event handler can be placed. This speeds up execution by removing the need for a jump instruction placed at the program address jumped to by the event hardware. All other exceptions have a dedicated event routine entry point located relative to EVBA. The handler routine address identifies the exception source directly.

AVR32 UC uses the ITLB and DTLB protection exceptions to signal a MPU protection violation.

All external interrupt requests have entry points located at an offset relative to EVBA. This autovector offset is specified by an external Interrupt Controller. The programmer must make sure that none of the autovector offsets interfere with the placement of other code. The autovector offset has 14 address bits, giving an offset of maximum 16384 bytes.

Special considerations should be made when loading EVBA with a pointer. Due to security considerations, the event handlers should be located in non-writeable flash memory, or optionally in a privileged memory protection region if an MPU is present.

If several events occur on the same instruction, they are handled in a prioritized way. The priority ordering is presented in Table 8-4. If events occur on several instructions at different locations in the pipeline, the events on the oldest instruction are always handled before any events on any younger instruction, even if the younger instruction has events of higher priority than the oldest instruction. An instruction B is younger than an instruction A if it was sent down the pipeline later than A.

The addresses and priority of simultaneous events are shown in Table 8-4. Some of the exceptions are unused in AVR32 UC since it has no MMU, coprocessor interface or floating-point unit.

Table 8-4. Priority and handler addresses for events

Priority	Handler Address	Name	Event source	Stored Return Address
1	0x8000_0000	Reset	External input	Undefined
2	Provided by OCD system	OCD Stop CPU	OCD system	First non-completed instruction
3	EVBA+0x00	Unrecoverable exception	Internal	PC of offending instruction
4	EVBA+0x04	TLB multiple hit	UNUSED	
5	EVBA+0x08	Bus error data fetch	Data bus	First non-completed instruction
6	EVBA+0x0C	Bus error instruction fetch	Data bus	First non-completed instruction
7	EVBA+0x10	NMI	External input	First non-completed instruction
8	Autovectored	Interrupt 3 request	External input	First non-completed instruction
9	Autovectored	Interrupt 2 request	External input	First non-completed instruction
10	Autovectored	Interrupt 1 request	External input	First non-completed instruction
11	Autovectored	Interrupt 0 request	External input	First non-completed instruction
12	EVBA+0x14	Instruction Address	CPU	PC of offending instruction
13	EVBA+0x50	ITLB Miss	UNUSED	
14	EVBA+0x18	ITLB Protection	MPU	PC of offending instruction
15	EVBA+0x1C	Breakpoint	OCD system	First non-completed instruction
16	EVBA+0x20	Illegal Opcode	Instruction	PC of offending instruction
17	EVBA+0x24	Unimplemented instruction	Instruction	PC of offending instruction
18	EVBA+0x28	Privilege violation	Instruction	PC of offending instruction
19	EVBA+0x2C	Floating-point	UNUSED	
20	EVBA+0x30	Coprocessor absent	UNUSED	
21	EVBA+0x100	Supervisor call	Instruction	PC(Supervisor Call) +2
22	EVBA+0x34	Data Address (Read)	CPU	PC of offending instruction
23	EVBA+0x38	Data Address (Write)	CPU	PC of offending instruction
24	EVBA+0x60	DTLB Miss (Read)	UNUSED	
25	EVBA+0x70	DTLB Miss (Write)	UNUSED	
26	EVBA+0x3C	DTLB Protection (Read)	MPU	PC of offending instruction
27	EVBA+0x40	DTLB Protection (Write)	MPU	PC of offending instruction
28	EVBA+0x44	DTLB Modified	UNUSED	

9. Memories

9.1 Embedded Memories

- **Internal High-Speed Flash**
  - **256 KBytes (AT32UC3B0256, AT32UC3B1256)**
  - **128 KBytes (AT32UC3B0128, AT32UC3B1128)**
  - **64 KBytes (AT32UC3B064, AT32UC3B164)**
    - **0 Wait State Access at up to 30 MHz in Worst Case Conditions**
    - **1 Wait State Access at up to 60 MHz in Worst Case Conditions**
    - **Pipelined Flash Architecture, allowing burst reads from sequential Flash locations, hiding penalty of 1 wait state access**
    - **Pipelined Flash Architecture typically reduces the cycle penalty of 1 wait state operation to only 8% compared to 0 wait state operation**
    - **10 000 Write Cycles, 10-year Data Retention Capability**
    - **1 ms Page Programming Time, 2 ms Chip Erase Time**
    - **Sector Lock Capabilities, Bootloader Protection, Security Bit**
    - **64 Fuses, 32 Of Which Are Preserved During Chip Erase**
    - **User Page For Data To Be Preserved During Chip Erase**
- **Internal High-Speed SRAM, Single-cycle access at full speed**
  - **32KBytes (AT32UC3B0256, AT32UC3B0128, AT32UC3B1256 and AT32UC3B1128)**
  - **16KBytes (AT32UC3B064 and AT32UC3B164)**

9.2 Physical Memory Map

The system bus is implemented as a bus matrix. All system bus addresses are fixed, and they are never remapped in any way, not even in boot. Note that AVR32 UC CPU uses unsegmented translation, as described in the AVR32 Architecture Manual. The 32-bit physical address space is mapped as follows:

Table 9-1. AT32UC3B Physical Memory Map

Start Address	Size						Device
	AT32UC3B0256	AT32UC3B1256	AT32UC3B0128	AT32UC3B1128	AT32UC3B064	AT32UC3B164	
0x0000_0000	32 Kbytes	32 Kbytes	32 Kbytes	32 Kbytes	16 Kbytes	16 Kbytes	Embedded SRAM
0x8000_0000	256 Kbytes	256 Kbytes	128 Kbytes	128 Kbytes	64 Kbytes	64 Kbytes	Embedded Flash
0xD000_0000	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	USB Configuration
0xFFFE_0000	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	64 Kbytes	HSB-PB Bridge A
0xFFFF_0000	64 Kbytes	64 Kbytes	64 kBytes	64 kBytes	64 Kbytes	64 Kbytes	HSB-PB Bridge B

Accesses to unused areas returns an error result to the master requesting such an access.

The bus matrix has the several masters and slaves. Each master has its own bus and its own decoder, thus allowing a different memory mapping per master. The master number in the table

below can be used to index the HMATRIX control registers. For example, MCFG0 is associated with the CPU Data master interface.

Table 9-2. High Speed Bus masters

Master 0	CPU Data
Master 1	CPU Instruction
Master 2	CPU SAB
Master 3	PDCA
Master 4	USBB DMA

Each slave has its own arbiter, thus allowing a different arbitration per slave. The slave number in the table below can be used to index the HMATRIX control registers. For example, SCFG3 is associated with the Internal SRAM Slave Interface.

Table 9-3. High Speed Bus slaves

Slave 0	Internal Flash
Slave 1	HSB-PB Bridge 0
Slave 2	HSB-PB Bridge 1
Slave 3	Internal SRAM Slave
Slave 4	USBB Slave

10. JTAG ID codes

The different device configurations have different JTAG ID codes (see the table below). If the flash controller is in reset the ID code will be undefined. Note that if the flash controller is in reset, the security bit in the device is set and JTAG instructions using the Service Access Bus will be blocked.

Device name	JTAG ID code (r is the revision number)
AT32UC3B0256	0xr1EE403F
AT32UC3B1256	0xr1EE503F
AT32UC3B0128	0xr1EE603F
AT32UC3B1128	0xr1EE903F
AT32UC3B064	0xr1EEA03F
AT32UC3B064	0xr1EEB03F

11. General purpose fuses

The flash block contains a number of general purpose fuses. Some of these fuses have defined meanings outside the flash controller and are described in this section.

11.1 Flash General Purpose Fuse Register (FGPFR)

31	30	29	28	27	26	25	24
GPF31	GPF30	GPF29	BODEN		BODHYST	BODLEVEL[5:4]	

23	22	21	20	19	18	17	16
BODLEVEL[3:0]				BOOTPROT			EPFL

15	14	13	12	11	10	9	8
LOCK[15:8]							

7	6	5	4	3	2	1	0
LOCK[7:0]							

BODEN: Brown Out Detector Enable

BODEN	Description
0x0	BOD disabled
0x1	BOD enabled, BOD reset enabled
0x2	BOD enabled, BOD reset disabled
0x3	BOD disabled

BODHYST: Brown Out Detector Hystersis

0: The Brown out detector hystersis is disabled

1: The Brown out detector hystersis is enabled

BODLEVEL: Brown Out Detector Trigger Level

This controls the voltage trigger level for the Brown out detector.

LOCK, EPFL, BOOTPROT

These are Flash controller fuses and are described in the FLASHC section

12. Peripherals

12.1 Peripheral Address Map

Table 12-1. Peripheral Address Mapping

Address		Peripheral Name	Bus
0xFFFE0000	USBB	USB 2.0 OTG - USBB	PBB
0xFFFE1000	HMATRIX	HMATRIX Configuration Interface - HMATRIX	PBB
0xFFFE1400	FLASHC	Flash controller - FLASHC	PBB
0xFFFF0000	PDCA	Peripheral Direct Memory Access - PDCA	PBA
0xFFFF0800	INTC	Interrupt controller - INTC	PBA
0xFFFF0C00	PM	Power Manager - PM	PBA
0xFFFF0D00	RTC	Real Time Counter - RTC	PBA
0xFFFF0D30	WDT	Watchdog Timer - WDT	PBA
0xFFFF0D80	EIC	External Interrupt Controller - EIC	PBA
0xFFFF1000	GPIO	General Purpose Input/Output - GPIO	PBA
0xFFFF1400	USART0	Universal Synchronous Asynchronous Receiver Transmitter - USART0	PBA
0xFFFF1800	USART1	Universal Synchronous Asynchronous Receiver Transmitter - USART1	PBA
0xFFFF1C00	USART2	Universal Synchronous Asynchronous Receiver Transmitter - USART2	PBA
0xFFFF2400	SPI	Serial Peripheral Interface - SPI	PBA
0xFFFF2C00	TWI	Two-wire Interface - TWI	PBA
0xFFFF3000	PWM	Pulse Width Modulation Controller - PWM	PBA
0xFFFF3400	SSC	Synchronous Serial Controller - SSC	PBA

Table 12-1. Peripheral Address Mapping

0xFFFF3800	TC	Timer/Counter - TC	PBA
0xFFFF3C00	ADC	Analog to Digital Converter - ADC	PBA

12.2 CPU Local Bus Mapping

Some of the registers in the GPIO module are mapped onto the CPU local bus, in addition to being mapped on the Peripheral Bus. These registers can therefore be reached both by accesses on the Peripheral Bus, and by accesses on the local bus.

Mapping these registers on the local bus allows cycle-deterministic toggling of GPIO pins since the CPU and GPIO are the only modules connected to this bus. Also, since the local bus runs at CPU speed, one write or read operation can be performed per clock cycle to the local bus-mapped GPIO registers.

The following GPIO registers are mapped on the local bus:

Table 12-2. Local bus mapped GPIO registers

Port	Register	Mode	Local Bus Address	Access
0	Output Driver Enable Register (ODER)	WRITE	0x4000_0040	Write-only
		SET	0x4000_0044	Write-only
		CLEAR	0x4000_0048	Write-only
		TOGGLE	0x4000_004C	Write-only
	Output Value Register (OVR)	WRITE	0x4000_0050	Write-only
		SET	0x4000_0054	Write-only
		CLEAR	0x4000_0058	Write-only
		TOGGLE	0x4000_005C	Write-only
	Pin Value Register (PVR)	-	0x4000_0060	Read-only
1	Output Driver Enable Register (ODER)	WRITE	0x4000_0140	Write-only
		SET	0x4000_0144	Write-only
		CLEAR	0x4000_0148	Write-only
		TOGGLE	0x4000_014C	Write-only
	Output Value Register (OVR)	WRITE	0x4000_0150	Write-only
		SET	0x4000_0154	Write-only
		CLEAR	0x4000_0158	Write-only
		TOGGLE	0x4000_015C	Write-only
	Pin Value Register (PVR)	-	0x4000_0160	Read-only

12.3 Interrupt Request Signal Map

The various modules may output Interrupt request signals. These signals are routed to the Interrupt Controller (INTC), described in a later chapter. The Interrupt Controller supports up to 64

groups of interrupt requests. Each group can have up to 32 interrupt request signals. All interrupt signals in the same group share the same autovector address and priority level. Refer to the documentation for the individual submodules for a description of the semantic of the different interrupt requests.

The interrupt request signals are connected to the INTC as follows.

Table 12-3. Interrupt Request Signal Map

Group	Line	Module	Signal
0	0	Stiletto CPU with optional MPU and optional OCD	SYSBLOCK COMPARE
1	0	External Interrupt Controller	EIC 0
	1	External Interrupt Controller	EIC 1
	2	External Interrupt Controller	EIC 2
	3	External Interrupt Controller	EIC 3
	4	External Interrupt Controller	EIC 4
	5	External Interrupt Controller	EIC 5
	6	External Interrupt Controller	EIC 6
	7	External Interrupt Controller	EIC 7
	8	Real Time Counter	RTC
	9	Power Manager	PM
2	0	General Purpose Input/Output	GPIO 0
	1	General Purpose Input/Output	GPIO 1
	2	General Purpose Input/Output	GPIO 2
	3	General Purpose Input/Output	GPIO 3
	4	General Purpose Input/Output	GPIO 4
	5	General Purpose Input/Output	GPIO 5
3	0	Peripheral Direct Memory Access	PDCA 0
	1	Peripheral Direct Memory Access	PDCA 1
	2	Peripheral Direct Memory Access	PDCA 2
	3	Peripheral Direct Memory Access	PDCA 3
	4	Peripheral Direct Memory Access	PDCA 4
	5	Peripheral Direct Memory Access	PDCA 5
	6	Peripheral Direct Memory Access	PDCA 6
4	0	Flash controller	FLASHC
5	0	Universal Synchronous Asynchronous Receiver Transmitter	USART0
6	0	Universal Synchronous Asynchronous Receiver Transmitter	USART1
7	0	Universal Synchronous Asynchronous Receiver Transmitter	USART2

Table 12-3. Interrupt Request Signal Map

9	0	Serial Peripheral Interface	SPI
11	0	Two-wire Interface	TWI
12	0	Pulse Width Modulation Controller	PWM
13	0	Synchronous Serial Controller	SSC
14	0	Timer/Counter	TC0
	1	Timer/Counter	TC1
	2	Timer/Counter	TC2
15	0	Analog to Digital Converter	ADC
17	0	USB 2.0 OTG	USBB

12.4 Clock Connections

12.4.1 Timer/Counters

Each Timer/Counter channel can independently select an internal or external clock source for its counter:

Table 12-4. Timer/Counter clock connections

Source	Name	Connection
Internal	TIMER_CLOCK1	Slow Clock (Internal RC oscillator)
	TIMER_CLOCK2	PBA Clock / 4
	TIMER_CLOCK3	PBA Clock / 8
	TIMER_CLOCK4	PBA Clock / 16
	TIMER_CLOCK5	PBA Clock / 32
External	XC0	See Section 12.8
	XC1	
	XC2	

12.4.2 USARTs

Each USART can be connected to an internally divided clock:

Table 12-5. USART clock connections

USART	Source	Name	Connection
0	Internal	CLK_DIV	PBA Clock / 8
1			
2			

12.4.3 SPIs

SPI can be connected to an internally divided clock:

Table 12-6. SPI clock connections

SPI	Source	Name	Connection
0	Internal	CLK_DIV	PBA clock or PBA clock / 32

12.5 Nexus OCD AUX port connections

If the OCD trace system is enabled, the trace system will take control over a number of pins, irrespectively of the PIO configuration. Two different OCD trace pin mappings are possible, depending on the configuration of the OCD AXS register. For details, see the AVR32 UCTechnical Reference Manual.

Table 12-7. Nexus OCD AUX port connections

Pin	AXS=0	AXS=1
EVTI_N	PB05	PA14
MDO[5]	PB04	PA08
MDO[4]	PB03	PA07
MDO[3]	PB02	PA06
MDO[2]	PB01	PA05
MDO[1]	PB00	PA03
MDO[0]	PA31	PA02
EVTO_N	PA15	PA15
MCKO	PA30	PA13
MSEO[1]	PB06	PA09
MSEO[0]	PB07	PA10

12.6 DMA handshake signals

The PDCA and the peripheral modules communicate through a set of handshake signals. The following table defines the valid settings for the Peripheral Identifier (PID) in the PDCA Peripheral Select Register (PSR).

Table 12-8. PDCA Handshake Signals

PID Value	Peripheral module & direction
0	ADC
1	SSC - RX
2	USART0 - RX
3	USART1 - RX
4	USART2 - RX

Table 12-8. PDCA Handshake Signals

PID Value	Peripheral module & direction
5	TWI - RX
6	SPI0 - RX
7	SSC - TX
8	USART0 - TX
9	USART1 - TX
10	USART2 - TX
11	TWI - TX
12	SPI0 - TX

12.7 High Drive Current GPIO

Ones of GPIOs can be used to drive twice current than other GPIO capability (see Electrical Characteristics chapter). The list of those GPIOs is shown in Table 12-9.

Table 12-9. High Dry Current GPIO

GPIO Name
GPIO/0/P21
GPIO/0/P22
GPIO/0/P23
GPIO/0/P24

12.8 Peripheral Multiplexing on IO lines

Each GPIO line can be assigned to one of 3 peripherals functions: A,B or C. The following table define how the I/O lines on peripherals A,B and C are multiplexed by the GPIO.

Table 12-10. GPIO Controller Function Multiplexing

QFP48 pin	QFP64 pin	Pin	GPIO Pin	Function A	Function B	Function C
7	9	PA03	GPIO 3	ADC/0/AD[0]	PM/0/GCLK[0]	USB/0/USB_ID
8	10	PA04	GPIO 4	ADC/1/AD[1]	PM/1/GCLK[1]	USB/0/USB_VBOF
9	11	PA05	GPIO 5	EIC/0/EXTINT[0]	ADC/0/ADC[2]	USART3/1/DCD
10	12	PA06	GPIO 6	EIC/O/EXTINT[1]	ADC/0/ADC[3]	USART/1/DSR
11	13	PA 07	GPIO 7	PWM/0/PWM[0]	ADC/0/AD[4]	USART/1/DSR
12	14	PA 08	GPIO 8	PWM/1/PWM[1]	ADC/0/ADC[5]	USART/1/RI
20	28	PA 09	GPIO 9	TWI/0/SCL	SPI/0/NPCS[2]	USART/1/CTS
21	29	PA 10	GPIO 10	TWI/0/SDA	SPI/0/NPCS[3]	USART/1/RTS
22	30	PA11	GPIO 11	USART/0/RTS	TC/0/A2	PWM/0/PWM[0]
23	31	PA12	GPIO 12	USART/0/CTS	TC/0/B2	PWM/1/PWM[1]

25	33	PA 13	GPIO 13	EIC/0/EXTINT[8]	PWM/0/PWM[2]	USART/0/CLK
26	34	PA 14	GPIO 14	SPI/0/MOSI	PWM/0/PWM[3]	EIC/0/EXTINT[2
27	35	PA 15	GPIO 15	SPI/0/SCK	PWM/0/PWM[4]	USART/2/CLK
28	36	PA 16	GPIO 16	SPI/0/NPCS[0]	TC/0/CLK1	
29	37	PA 17	GPIO 17	SPI/0/NPCS[1]	TC/0/CLK2	SPI/0/SCK
30	39	PA 18	GPIO 18	USART/0/RXD	PWM/0/PWM[5]	SPI/0/MISO
31	40	PA 19	GPIO 19	USART/0/TXD	PWM/0/PWM[6]	SPI/0/MOSI
32	44	PA20	GPIO 20	USART/1/CLK	TC/0/CLK0	USART/2/RXD
33	45	PA 21	GPIO 21	PWM/0/PWM[2]	TC/0/A1	USART/2/TXD
34	46	PA 22	GPIO 22	PWM/0/PWM[6]	TC/0/B1	ADC/0/TRIGGER
35	47	PA 23	GPIO 23	USART/1/TXD	SPI/0/NPCS[1]	EIC/0/EXTINT[3]
43	59	PA 24	GPIO 24	USART/1/RXD	SPI/0/NPCS[0]	EIC/0/EXTINT[4]
44	60	PA 25	GPIO 25	SPI/0/MISO	PWM/0/PWM3	EIC/0/EXTINT[5]
45	61	PA 26	GPIO 26	USB/0/USB_ID	USART/2/TXD	TC/0/0A0
46	62	PA 27	GPIO 27	USB/0/USB_VBOF	USART/2/RXD	TC/0/B0
	41	PA 28	GPIO 28	USART/0/CLK	PWM/0/PWM[4]	SPI/0/MISO
	42	PA 29	GPIO 29	TC/0/CLK0	TC/0/CLK1	SPI/0/MOSI
	15	PA 30	GPIO 30	ADC/0/AD[6]	EIC/0/SCAN[0]	PM/0/GCLK[2]
	16	PA 31	GPIO 31	ADC/0/ADC[7]	EIC/0/SCAN[1]	
	6	PB 00	GPIO 32	TC/0/A0	EIC/0/SCAN[2]	USART/2/CTS
	7	PB 01	GPIO 33	TC/0/B0	EIC/0/SCAN[3]	USART/2/RTS
	24	PB 02	GPIO 34	EIC/0/EXTINT[6]	TC/0/A1	USART/1/TXD
	25	PB 03	GPIO 35	EIC/0/EXTINT[7]	TC/0/B1	USART/1/RXD
	26	PB 04	GPIO 36	USART/1/CTS/	SPI/0/NPCS[3]	TC/0/CLK2
	27	PB 05	GPIO 37	USART/1/RTS	SPI/0/NPCS[2]	PWM/0/PWM[5]
	38	PB 06	GPIO 38	SSC/0/RX_CLOCK	USART/1/DCD	EIC/0/SCAN[4]
	43	PB 07	GPIO 39	SSC/0/RX_DATA	USART/1/DSR	EIC/0/SCAN[5]
	54	PB 08	GPIO 40	SSC/0/RX_FRAME	USART/1/DTR	EIC/0/SCAN[6]
	55	PB 09	GPIO 41	SSC/0/TX_CLOCK	USART/1/RI	EIC/0/SCAN[7]
	57	PB 10	GPIO 42	SSC/0/TX_DATA	TC/0/A2	USART/0/RXD
	58	PB 11	GPIO 43	SSC/0/TX_FRAME	TC/0/B2	USART/0/TXD

12.9 Oscillator Pinout

The oscillators are not mapped to the normal A,B or C functions and their muxings are controlled by registers in the Power Manager (PM). Please refer to the power manager chapter for more information about this.

Table 12-11. Oscillator pinout

QFP48 pin	QFP64 pin	Pad	Oscillator pin
30	39	PA18	xin0
	41	PA28	xin1
22	30	PA11	xin32
31	40	PA19	xout0
	42	PA29	xout1
23	31	PA12	xout32

12.10 Peripheral overview

12.10.1 USB Controller

- **USB 2.0 Compliant, Full-/Low-Speed (FS/LS) and On-The-Go (OTG), 12 Mbit/s**
- **7 Pipes/Endpoints**
- **960 bytes of Embedded Dual-Port RAM (DPRAM) for Pipes/Endpoints**
- **Up to 2 Memory Banks per Pipe/Endpoint (Not for Control Pipe/Endpoint)**
- **Flexible Pipe/Endpoint Configuration and Management with Dedicated DMA Channels**
- **On-Chip Transceivers Including Pull-Ups**
- **System wake-up on USB line activity**

12.10.2 Serial Peripheral Interface

- **Supports communication with serial external devices**
  - **Four chip selects with external decoder support allow communication with up to 15 peripherals**
  - **Serial memories, such as DataFlash and 3-wire EEPROMs**
  - **Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors**
  - **External co-processors**
- **Master or slave serial peripheral bus interface**
  - **8- to 16-bit programmable data length per chip select**
  - **Programmable phase and polarity per chip select**
  - **Programmable transfer delays between consecutive transfers and between clock and data per chip select**
  - **Programmable delay between consecutive transfers**
  - **Selectable mode fault detection**
- **Very fast transfers supported**
  - **Transfers with baud rates up to MCK**
  - **The chip select line may be left active to speed up transfers on the same device**

12.10.3 Two-wire Interface

- High speed up to 400kbit/s
- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations

12.10.4 USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode or 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB- or LSB-first
  - Optional break generation and detection
  - By 8 or by-16 over-sampling receiver frequency
  - Hardware handshaking RTS-CTS
  - Receiver time-out and transmitter timeguard
  - Optional Multi-drop Mode with address generation and detection
  - Optional Manchester Encoding
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- IrDA modulation and demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo

12.10.5 Serial Synchronous Controller

- Provides serial synchronous communication links used in audio and telecom applications (with CODECs in Master or Slave Modes, I2S, TDM Buses, Magnetic Card Reader, etc.)
- Contains an independent receiver and transmitter and a common clock divider
- Offers a configurable frame sync and data length
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal

12.10.6 Timer Counter

- Three 16-bit Timer Counter Channels
- Wide range of functions including:
  - Frequency Measurement
  - Event Counting
  - Interval Measurement
  - Pulse Generation
  - Delay Timing
  - Pulse Width Modulation
  - Up/down Capabilities
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five internal clock inputs
  - Two multi-purpose input/output signals

- Two global registers that act on all three TC Channels

12.10.7 Pulse Width Modulation Controller

- 7 channels, one 16-bit counter per channel
- Common clock generator, providing Thirteen Different Clocks
  - A Modulo n counter providing eleven clocks
  - Two independent Linear Dividers working on modulo n counter outputs
- Independent channel programming
  - Independent Enable Disable Commands
  - Independent Clock
  - Independent Period and Duty Cycle, with Double Bufferization
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

13. Power Manager (PM)

Rev: 2.0.0.1

13.1 Features

- **Controls integrated oscillators and PLLs**
- **Generates clocks and resets for digital logic**
- **Supports 2 crystal oscillators 450 kHz-16 MHz**
- **Supports 2 PLLs 80-240 MHz**
- **Supports 32 KHz ultra-low power oscillator**
- **Integrated low-power RC oscillator**
- **On-the fly frequency change of CPU, HSB, PBA, and PBB clocks**
- **Sleep modes allow simple disabling of logic clocks, PLLs, and oscillators**
- **Module-level clock gating through maskable peripheral clocks**
- **Wake-up from internal or external interrupts**
- **Generic clocks with wide frequency range provided**
- **Automatic identification of reset sources**
- **Controls brownout detector (BOD), RC oscillator, and bandgap voltage reference through control and calibration registers**

13.2 Description

The Power Manager (PM) controls the oscillators and PLLs, and generates the clocks and resets in the device. The PM controls two fast crystal oscillators, as well as two PLLs, which can multiply the clock from either oscillator to provide higher frequencies. Additionally, a low-power 32 KHz oscillator is used to generate the real-time counter clock for high accuracy real-time measurements. The PM also contains a low-power RC oscillator with fast start-up time, which can be used to clock the digital logic.

The provided clocks are divided into synchronous and generic clocks. The synchronous clocks are used to clock the main digital logic in the device, namely the CPU, and the modules and peripherals connected to the HSB, PBA, and PBB buses. The generic clocks are asynchronous clocks, which can be tuned precisely within a wide frequency range, which makes them suitable for peripherals that require specific frequencies, such as timers and communication modules.

The PM also contains advanced power-saving features, allowing the user to optimize the power consumption for an application. The synchronous clocks are divided into three clock domains, one for the CPU and HSB, one for modules on the PBA bus, and one for modules on the PBB bus.The three clocks can run at different speeds, so the user can save power by running peripherals at a relatively low clock, while maintaining a high CPU performance. Additionally, the clocks can be independently changed on-the-fly, without halting any peripherals. This enables the user to adjust the speed of the CPU and memories to the dynamic load of the application, without disturbing or re-configuring active peripherals.

Each module also has a separate clock, enabling the user to switch off the clock for inactive modules, to save further power. Additionally, clocks and oscillators can be automatically switched off during idle periods by using the sleep instruction on the CPU. The system will return to normal on occurrence of interrupts.

The Power Manager also contains a Reset Controller, which collects all possible reset sources, generates hard and soft resets, and allows the reset source to be identified by software.

13.3 Block Diagram

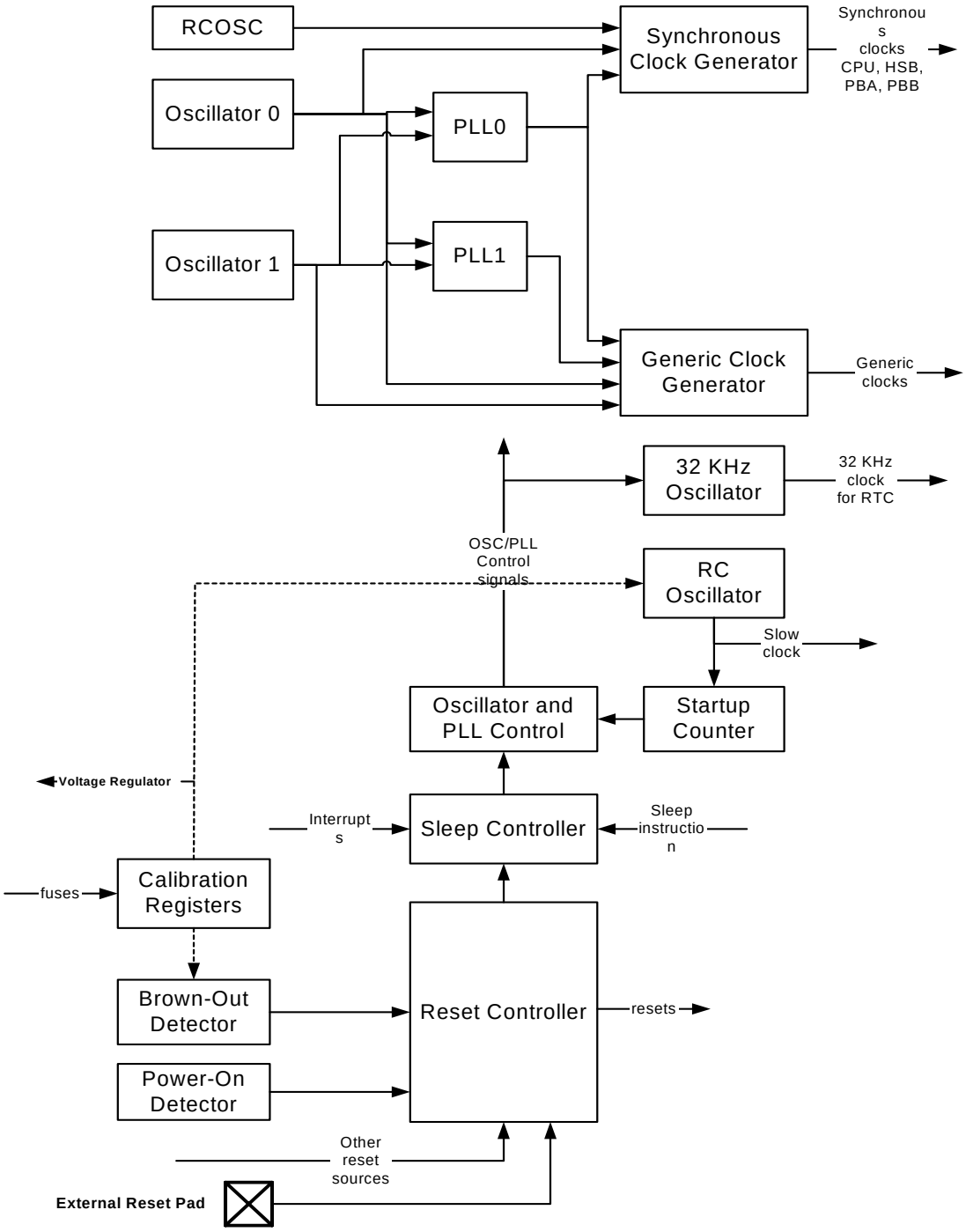


Figure 13-1. Power Manager block diagram

13.4 Product Dependencies

13.4.1 I/O Lines

The PM provides a number of generic clock outputs, which can be connected to output pins, multiplexed with GPIO lines. The programmer must first program the GPIO controller to assign these pins to their peripheral function. If the I/O pins of the PM are not used by the application, they can be used for other purposes by the GPIO controller.

13.4.2 Interrupt

The PM interrupt line is connected to one of the internal sources of the interrupt controller. Using the PM interrupt requires the interrupt controller to be programmed first.

13.4.3 Clock implementation

In AT32UC3B, the HSB shares the source clock with the CPU. This means that writing to the HSBDIV and HSBSEL bits in CKSEL has no effect. These bits will always read the same as CPUDIV and CPUSEL.

13.5 Functional Description

13.5.1 Slow clock

The slow clock is generated from an internal RC oscillator which is always running, except in Static mode. The slow clock can be used for the main clock in the device, as described in "Synchronous clocks" on page 49. The slow clock is also used for the Watchdog Timer and measuring various delays in the Power Manager.

The RC oscillator has a 3 cycles startup time, and is always available when the CPU is running. The RC oscillator operates at approximately 115 kHz, and can be calibrated to a narrow range by the RCOSCCAL fuses. Software can also change RC oscillator calibration through the use of the RCCR register. Please see the Electrical Characteristics section for details.

RC oscillator can also be used as the RTC clock when crystal accuracy is not required.

13.5.2 Oscillator 0 and 1 operation

The two main oscillators are designed to be used with an external 450 kHz to 16 MHz crystal and two biasing capacitors, as shown in Figure 13-2. Oscillator 0 can be used for the main clock in the device, as described in "Synchronous clocks" on page 49. Both oscillators can be used as source for the generic clocks, as described in "Generic clocks" on page 52.

The oscillators are disabled by default after reset. When the oscillators are disabled, the XIN and XOUT pins can be used as general purpose I/Os. When the oscillators are configured to use an external clock, the clock must be applied to the XIN pin while the XOUT pin can be used as a general purpose I/O.

The oscillators can be enabled by writing to the OSCnEN bits in MCCTRL. Operation mode (external clock or crystal) is chosen by writing to the MODE field in OSCCTRLn. Oscillators are automatically switched off in certain sleep modes to reduce power consumption, as described in Section 13.5.7 on page 51.

After a hard reset, or when waking up from a sleep mode that disabled the oscillators, the oscillators may need a certain amount of time to stabilize on the correct frequency. This start-up time can be set in the OSCCTRLn register.

The PM masks the oscillator outputs during the start-up time, to ensure that no unstable clocks propagate to the digital logic. The OSCnRDY bits in POSCSR are automatically set and cleared according to the status of the oscillators. A zero to one transition on these bits can also be configured to generate an interrupt, as described in ”Interrupt Enable/Disable/Mask/Status/Clear” on page 67.

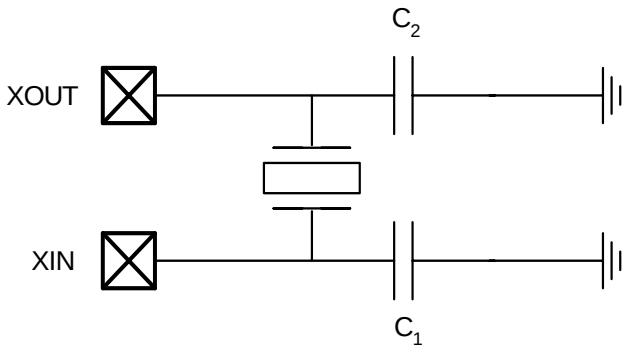


Figure 13-2. Oscillator connections

13.5.3 32 KHz oscillator operation

The 32 KHz oscillator operates as described for Oscillator 0 and 1 above. The 32 KHz oscillator is used as source clock for the Real-Time Counter.

The oscillator is disabled by default, but can be enabled by writing OSC32EN in OSCCTRL32. The oscillator is an ultra-low power design and remains enabled in all sleep modes except Static mode.

While the 32 KHz oscillator is disabled, the XIN32 and XOUT32 pins are available as general purpose I/Os. When the oscillator is configured to work with an external clock (MODE field in OSCCTRL32 register), the external clock must be connected to XIN32 while the XOUT32 pin can be used as a general purpose I/O.

The startup time of the 32 KHz oscillator can be set in the OSCCTRL32, after which OSC32RDY in POSCSR is set. An interrupt can be generated on a zero to one transition of OSC32RDY.

As a crystal oscillator usually requires a very long startup time (up to 1 second), the 32 KHz oscillator will keep running across resets, except Power-On-Reset.

13.5.4 PLL operation

The device contains two PLLs, PLL0 and PLL1. These are disabled by default, but can be enabled to provide high frequency source clocks for synchronous or generic clocks. The PLLs can take either Oscillator 0 or 1 as reference clock. The PLL output is divided by a multiplication factor, and the PLL compares the resulting clock to the reference clock. The PLL will adjust its output frequency until the two compared clocks are equal, thus locking the output frequency to a multiple of the reference clock frequency.

The Voltage Controlled Oscillator inside the PLL can generate frequencies from 80 to 240 MHz. To make the PLL output frequencies under 80 MHz the OTP[1] bitfield could be set. This will

divide the output of the PLL by two and bring the clock in range of the max frequency of the CPU.

When the PLL is switched on, or when changing the clock source or multiplication factor for the PLL, the PLL is unlocked and the output frequency is undefined. The PLL clock for the digital logic is automatically masked when the PLL is unlocked, to prevent connected digital logic from receiving a too high frequency and thus become unstable.

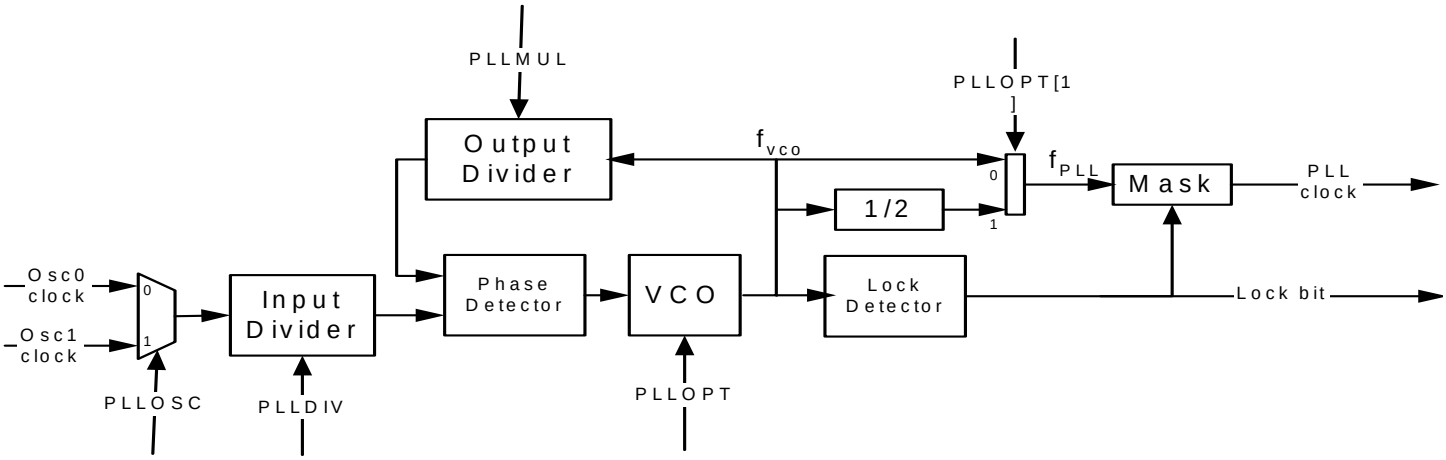


Figure 13-3. PLL with control logic and filters

13.5.4.1 Enabling the PLL

PLLn is enabled by writing the PLLEN bit in the PLLn register. PLLOSC selects Oscillator 0 or 1 as clock source. The PLLMUL and PLLDIV bitfields must be written with the multiplication and division factors, respectively, creating the voltage controlled ocillator frequency f_{VCO} and the PLL frequency f_{PLL} :

$f_{VCO} = (PLLMUL+1)/(PLLDIV) \bullet f_{OSC}$ if PLLDIV > 0.

$f_{VCO} = 2*(PLLMUL+1) \bullet f_{OSC}$ if PLLDIV = 0.

If PLLOPT[1] field is set to 0:

$f_{PLL} = f_{VCO}$.

If PLLOPT[1] field is set to 1:

$f_{PLL} = f_{VCO} / 2$.

The PLLn:PLLOPT field should be set to proper values according to the PLL operating frequency. The PLLOPT field can also be set to divide the output frequency of the PLLs by 2.

The lock signal for each PLL is available as a LOCKn flag in POSCSR. An interrupt can be generated on a 0 to 1 transition of these bits.

13.5.5 Synchronous clocks

The slow clock (default), Oscillator 0, or PLL0 provide the source for the main clock, which is the common root for the synchronous clocks for the CPU/HSB, PBA, and PBB modules. The main clock is divided by an 8-bit prescaler, and each of these four synchronous clocks can run from any tapping of this prescaler, or the undivided main clock, as long as $f_{CPU} \geq f_{PBA,B,}$. The synchronous clock source can be changed on-the fly, responding to varying load in the application. The clock domains can be shut down in sleep mode, as described in "Sleep modes" on page 51. Additionally, the clocks for each module in the four domains can be individually masked, to avoid power consumption in inactive modules.

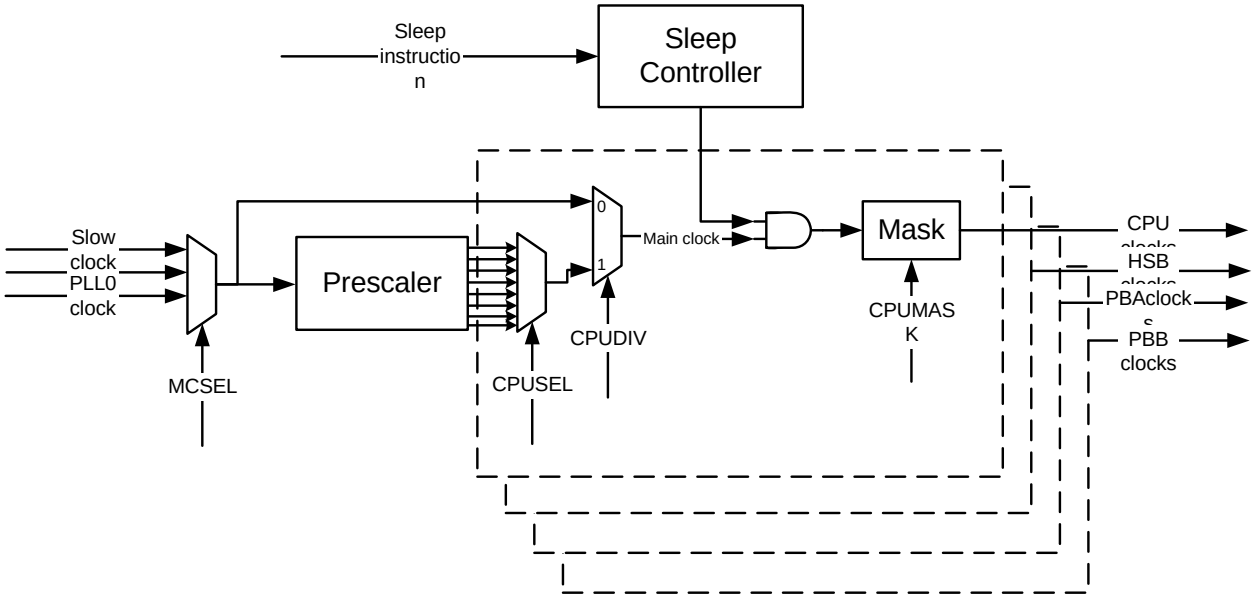


Figure 13-4. Synchronous clock generation

13.5.5.1 *Selecting PLL or oscillator for the main clock*

The common main clock can be connected to the slow clock, Oscillator 0, or PLL0. By default, the main clock will be connected to the slow clock. The user can connect the main clock to Oscillator 0 or PLL0 by writing the MCSEL bitfield in the Main Clock Control Register (MCCTRL). This must only be done after that unit has been enabled, otherwise a deadlock will occur. Care should also be taken that the new frequency of the synchronous clocks does not exceed the maximum frequency for each clock domain.

13.5.5.2 *Selecting synchronous clock division ratio*

The main clock feeds an 8-bit prescaler, which can be used to generate the synchronous clocks. By default, the synchronous clocks run on the undivided main clock. The user can select a pres-

caler division for the CPU clock by writing CKSEL:CPUDIV to 1 and CPUSEL to the prescaling value, resulting in a CPU clock frequency:

$$f_{CPU} = f_{main} / 2^{(CPUSEL+1)}$$

Similarly, the clock for the PBA, and PBB can be divided by writing their respective bitfields. To ensure correct operation, frequencies must be selected so that $f_{CPU} \geq f_{PBA,B}$. Also, frequencies must never exceed the specified maximum frequency for each clock domain.

CKSEL can be written without halting or disabling peripheral modules. Writing CKSEL allows a new clock setting to be written to all synchronous clocks at the same time. It is possible to keep one or more clocks unchanged by writing the same value a before to the xxxDIV and xxxSEL bitfields. This way, it is possible to e.g. scale CPU and HSB speed according to the required performance, while keeping the PBA and PBB frequency constant.

13.5.5.3 Clock Ready flag

There is a slight delay from CKSEL is written and the new clock setting becomes effective. During this interval, the Clock Ready (CKRDY) flag in ISR will read as 0. If IER:CKRDY is written to 1, the Power Manager interrupt can be triggered when the new clock setting is effective. CKSEL must not be re-written while CKRDY is 0, or the system may become unstable or hang.

13.5.6 Peripheral clock masking

By default, the clock for all modules are enabled, regardless of which modules are actually being used. It is possible to disable the clock for a module in the CPU, HSB, PBA, or PBB clock domain by writing the corresponding bit in the Clock Mask register (CPU/HSB/PBA/PBB) to 0. When a module is not clocked, it will cease operation, and its registers cannot be read or written. The module can be re-enabled later by writing the corresponding mask bit to 1.

A module may be connected to several clock domains, in which case it will have several mask bits.

Table 13-5 contains a list of implemented maskable clocks.

13.5.6.1 Cautionary note

Note that clocks should only be switched off if it is certain that the module will not be used. Switching off the clock for the internal RAM will cause a problem if the stack is mapped there. Switching off the clock to the Power Manager (PM), which contains the mask registers, or the corresponding PBx bridge, will make it impossible to write the mask registers again. In this case, they can only be re-enabled by a system reset.

13.5.6.2 Mask Ready flag

Due to synchronization in the clock generator, there is a slight delay from a mask register is written until the new mask setting goes into effect. When clearing mask bits, this delay can usually be ignored. However, when setting mask bits, the registers in the corresponding module must not be written until the clock has actually be re-enabled. The status flag MSKRDY in ISR provides the required mask status information. When writing either mask register with any value, this bit is cleared. The bit is set when the clocks have been enabled and disabled according to the new mask setting. Optionally, the Power Manager interrupt can be enabled by writing the MSKRDY bit in IER.

13.5.7 Sleep modes

In normal operation, all clock domains are active, allowing software execution and peripheral operation. When the CPU is idle, it is possible to switch off the CPU clock and optionally other clock domains to save power. This is activated by the sleep instruction, which takes the sleep mode index number as argument.

13.5.7.1 Entering and exiting sleep modes

The sleep instruction will halt the CPU and all modules belonging to the stopped clock domains. The modules will be halted regardless of the bit settings of the mask registers.

Oscillators and PLLs can also be switched off to save power. Some of these modules have a relatively long start-up time, and are only switched off when very low power consumption is required.

The CPU and affected modules are restarted when the sleep mode is exited. This occurs when an interrupt triggers. Note that even if an interrupt is enabled in sleep mode, it may not trigger if the source module is not clocked.

13.5.7.2 Supported sleep modes

The following sleep modes are supported. These are detailed in Table 13-1.

•Idle: The CPU is stopped, the rest of the chip is operating. Wake-up sources are any interrupt.

•Frozen: The CPU and HSB modules are stopped, peripherals are operating. Wake-up sources are any interrupt from PB modules.

•Standby: All synchronous clocks are stopped, but oscillators and PLLs are running, allowing quick wake-up to normal mode. Wake-up sources are RTC or external interrupt.

•Stop: As Standby, but Oscillator 0 and 1, and the PLLs are stopped. 32 KHz (if enabled) and RC oscillators and RTC/WDT still operate. Wake-up sources are RTC, external interrupt, or external reset pin.

•DeepStop: All synchronous clocks, Oscillator 0 and 1 and PLL 0 and 1 are stopped. 32 KHz oscillator can run if enabled. RC oscillator still operates. Bandgap voltage reference and BOD is turned off.

•Static: All oscillators, including 32 KHz and RC oscillator are stopped. Bandgap voltage reference BOD detector is turned off.

Table 13-1. Sleep modes

Index	Sleep Mode	CPU	HSB	PBA,B GCLK	Osc0,1 PLL0,1	Osc32	RCOsc	BOD & Bandgap	Voltage Regulator
0	**Idle**	Stop	Run	Run	Run	Run	Run	On	Full power
1	**Frozen**	Stop	Stop	Run	Run	Run	Run	On	Full power
2	**Standby**	Stop	Stop	Stop	Run	Run	Run	On	Full power
3	**Stop**	Stop	Stop	Stop	Stop	Run	Run	On	Low power
4	**DeepStop**	Stop	Stop	Stop	Stop	Run	Run	Off	Low power
5	**Static**	Stop	Stop	Stop	Stop	Stop	Stop	Off	Low power

The power level of the internal voltage regulator is also adjusted according to the sleep mode to reduce the internal regulator power consumption.

13.5.7.3 Precautions when entering sleep mode

Modules communicating with external circuits should normally be disabled before entering a sleep mode that will stop the module operation. This prevents erratic behavior when entering or exiting sleep mode. Please refer to the relevant module documentation for recommended actions.

Communication between the synchronous clock domains is disturbed when entering and exiting sleep modes. This means that bus transactions are not allowed between clock domains affected by the sleep mode. The system may hang if the bus clocks are stopped in the middle of a bus transaction.

The CPU is automatically stopped in a safe state to ensure that all CPU bus operations are complete when the sleep mode goes into effect. Thus, when entering Idle mode, no further action is necessary.

When entering a sleep mode (except Idle mode), all HSB masters must be stopped before entering the sleep mode. Also, if there is a chance that any PB write operations are incomplete, the CPU should perform a read operation from any register on the PB bus before executing the sleep instruction. This will stall the CPU while waiting for any pending PB operations to complete.

13.5.7.4 Wake Up

The USB can be used to wake up the part from sleep modes through register PM_AWEN of the Power Manager.

13.5.8 Generic clocks

Timers, communication modules, and other modules connected to external circuitry may require specific clock frequencies to operate correctly. The Power Manager contains an implementation defined number of generic clocks that can provide a wide range of accurate clock frequencies.

Each generic clock module runs from either Oscillator 0 or 1, or PLL0 or 1. The selected source can optionally be divided by any even integer up to 512. Each clock can be independently enabled and disabled, and is also automatically disabled along with peripheral clocks by the Sleep Controller.

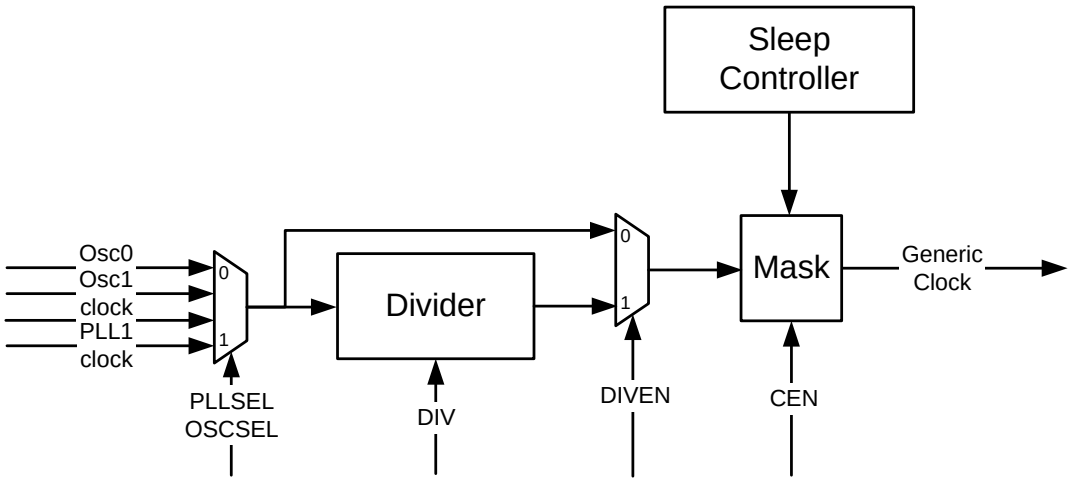


Figure 13-5. Generic clock generation

13.5.8.1 *Enabling a generic clock*

A generic clock is enabled by writing the CEN bit in GCCTRL to 1. Each generic clock can use either Oscillator 0 or 1 or PLL0 or 1 as source, as selected by the PLLSEL and OSCSEL bits. The source clock can optionally be divided by writing DIVEN to 1 and the division factor to DIV, resulting in the output frequency:

$$f_{GCLK} = f_{SRC} / (2*(DIV+1))$$

13.5.8.2 *Disabling a generic clock*

The generic clock can be disabled by writing CEN to 0 or entering a sleep mode that disables the PB clocks. In either case, the generic clock will be switched off on the first falling edge after the disabling event, to ensure that no glitches occur. If CEN is written to 0, the bit will still read as 1 until the next falling edge occurs, and the clock is actually switched off. When writing CEN to 0, the other bits in GCCTRL should not be changed until CEN reads as 0, to avoid glitches on the generic clock.

When the clock is disabled, both the prescaler and output are reset.

13.5.8.3 *Changing clock frequency*

When changing generic clock frequency by writing GCCTRL, the clock should be switched off by the procedure above, before being re-enabled with the new clock source or division setting. This prevents glitches during the transition.

13.5.8.4 Generic clock implementation

In AT32UC3B, there are 4 generic clocks. These are allocated to different functions as shown in Table 13-2.

Table 13-2. Generic clock allocation

Clock number	Function
0	GCLK0 pin
1	GCLK1 pin
2	GCLK2 pin
3	USBB

13.5.9 Divided PB clocks

The clock generator in the Power Manager provides divided PBA and PBB clocks for use by peripherals that require a prescaled PBx clock. This is described in the documentation for the relevant modules.

The divided clocks are not directly maskable, but are stopped in sleep modes where the PBx clocks are stopped.

13.5.10 Debug operation

During a debug session, the user may need to halt the system to inspect memory and CPU registers. The clocks normally keep running during this debug operation, but some peripherals may require the clocks to be stopped, e.g. to prevent timer overflow, which would cause the program to fail. For this reason, peripherals on the PBA and PBB buses may use “debug qualified” PBx clocks. This is described in the documentation for the relevant modules. The divided PBx clocks are always debug qualified clocks.

Debug qualified PB clocks are stopped during debug operation. The debug system can optionally keep these clocks running during the debug operation. This is described in the documentation for the On-Chip Debug system.

13.5.11 Reset Controller

The Reset Controller collects the various reset sources in the system and generates hard and soft resets for the digital logic.

The device contains a Power-On Detector, which keeps the system reset until power is stable. This eliminates the need for external reset circuitry to guarantee stable operation when powering up the device.

It is also possible to reset the device by asserting the RESET_N pin. This pin has an internal pull-up, and does not need to be driven externally when negated. Table 13-4 lists these and other reset sources supported by the Reset Controller.

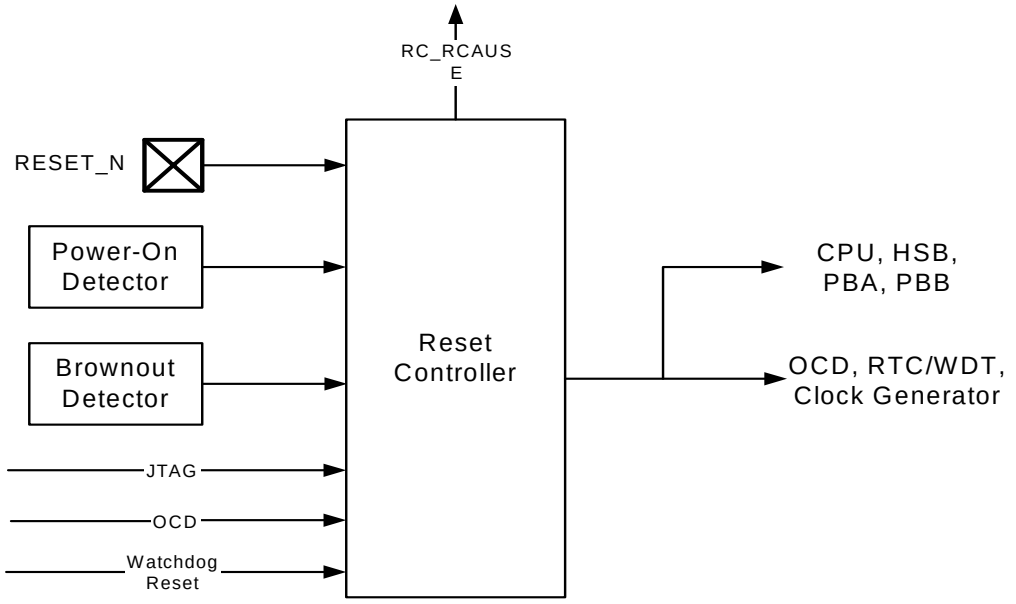


Figure 13-6. Reset Controller block diagram

In addition to the listed reset types, the JTAG can keep parts of the device statically reset through the JTAG Reset Register. See JTAG documentation for details.

Table 13-3. Reset description

Reset source	Description
Power-on Reset	Supply voltage below the power-on reset detector threshold voltage
External Reset	RESET_N pin asserted
Brownout Reset	Supply voltage below the brownout reset detector threshold voltage
CPU Error	Caused by an illegal CPU access to external memory while in Supervisor mode
Watchdog Timer	See watchdog timer documentation.
OCD	See On-Chip Debug documentation

When a Reset occurs, some parts of the chip are not necessarily reset, depending on the reset source. Only the Power On Reset (POR) will force a reset of the whole chip.

Table 13-4 lists parts of the device that are reset, depending on the reset source.

Table 13-4. Effect of the different reset events

	Power-On Reset	External Reset	Watchdog Reset	BOD Reset	CPU Error Reset	OCD Reset
CPU/HSB/PBA/PBB (excluding Power Manager)	Y	Y	Y	Y	Y	Y
32 KHz oscillator	Y	N	N	N	N	N
RTC control register	Y	N	N	N	N	N
GPLP registers	Y	N	N	N	N	N
Watchdog control register	Y	Y	N	Y	Y	Y
Voltage Calibration register	Y	N	N	N	N	N
RC Oscillator Calibration register	Y	N	N	N	N	N
BOD control register	Y	Y	N	N	N	N
Bandgap control register	Y	Y	N	N	N	N
Clock control registers	Y	Y	Y	Y	Y	Y
Osc0/Osc1 and control registers	Y	Y	Y	Y	Y	Y
PLL0/PLL1 and control registers	Y	Y	Y	Y	Y	Y
OCD system and OCD registers	Y	Y	N	Y	Y	N

The cause of the last reset can be read from the RCAUSE register. This register contains one bit for each reset source, and can be read during the boot sequence of an application to determine the proper action to be taken.

13.5.11.1 Power-On Detector

The Power-On Detector monitors the VDDCORE supply pin and generates a reset when the device is powered on. The reset is active until the supply voltage from the linear regulator is above the power-on threshold level. The reset will be re-activated if the voltage drops below the power-on threshold level. See Electrical Characteristics for parametric details.

13.5.11.2 Brown-Out Detector

The Brown-Out Detector (BOD) monitors the VDDCORE supply pin and compares the supply voltage to the brown-out detection level, as set in BOD:LEVEL. The BOD is disabled by default, but can be enabled either by software or by flash fuses. The Brown-Out Detector can either generate an interrupt or a reset when the supply voltage is below the brown-out detection level. In any case, the BOD output is available in bit POSCR:BODET bit.

Note 1 :

Any change to the BOD:LEVEL field of the BOD register should be done with the BOD deactivated to avoid spurious reset or interrupt.

Note 2 :

If the BOD level is set to a value higher than VDDCORE and enabled by fuses, the part will be in constant reset. In order to leave reset state, an external voltage higher than the BOD level should be applied. Thus, it is possible to disable BOD.

See Electrical Characteristics for parametric details.

13.5.11.3 External Reset

The external reset detector monitors the state of the RESET_N pin. By default, a low level on this pin will generate a reset.

13.5.12 Calibration registers

The Power Manager controls the calibration of the RC oscillator, voltage regulator, bandgap voltage reference through several calibrations registers.

Those calibration registers are loaded after a Power On Reset with default values stored in factory-programmed flash fuses.

Although it is not recommended to override default factory settings, it is still possible to override these default values by writing to those registers. To prevent unexpected writes due to software bugs, write access to these registers is protected by a “key”. First, a write to the register must be made with the field “KEY” equal to 0x55 then a second write must be issued with the “KEY” field equal to 0xAA

13.6 User Interface

Offset	Register	Name	Access	Reset State
0x0000	Main Clock Control	MCCTRL	Read/Write	0x00000000
0x0004	Clock Select	CKSEL	Read/Write	0x00000000
0x0008	CPU Mask	CPUMASK	Read/Write	0x00000000
0x000C	HSB Mask	HSBMASK	Read/Write	0x00000000
0x0010	PBA Mask	PBAMASK	Read/Write	0x00000000
0x0014	PBB Mask	PBBMASK	Read/Write	0x00000000
0x0018 - 0x001C	Reserved			
0x0020	PLL0 Control	PLL0	Read/Write	0x00000000
0x0024	PLL1 Control	PLL1	Read/Write	0x00000000
0x0028	Oscillator 0 Control Register	OSCCTRL0	Read/Write	0x00000000
0x002C	Oscillator 1 Control Register	OSCCTRL1	Read/Write	0x00000000
0x0030	Oscillator 32 Control Register	OSCCTRL32	Read/Write	0x00000000

0x0034	Reserved			
0x0038	Reserved			
0x003C	Reserved			
0x0040	PM Interrupt Enable Register	IER	Write Only	0x00000000
0x0044	PM Interrupt Disable Register	IDR	Write Only	0x00000000
0x0048	PM Interrupt Mask Register	IMR	Read Only	0x00000000
0x004C	PM Interrupt Status Register	ISR	Read Only	0x00000000
0x0050	PM Interrupt Clear Register	ICR	Write Only	0x00000000
0x0054	Power and Oscillators Status Register	POSCSR	Read/Write	0x00000000
0x0058 - 0x005C	Reserved			
0x0060	Generic Clock Control	GCCTRL	Read/Write	0x00000000
0x0064 - 0x00BC	Reserved			
0x00C0	RC Oscillator Calibration Register	RCCR	Read/Write	Factory settings
0x00C4	Bandgap Calibration Register	BGCR	Read/Write	Factory settings
0x00C8	Linear Regulator Calibration Register	VREGCR	Read/Write	Factory settings
0x00CC	Reserved			
0x00D0	BOD Level Register	BOD	Read/Write	BOD fuses in Flash
0x00D4 - 0x013C	Reserved			
0x0140	Reset Cause Register	RCAUSE	Read Only	Latest Reset Source
0x0144 - 0x01FC	Reserved			
0x0200	General Purpose Low-Power register	GPLP	Read/Write	0x00000000

13.6.1 Main Clock Control

Name: MCCTRL

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-			OSC1EN	OSC0EN	MCSEL	

- **MCSEL: Main Clock Select**
  - 0: The slow clock is the source for the main clock
  - 1: Oscillator 0 is source for the main clock
  - 2: PLL0 is source for the main clock
  - 3: Reserved
- **OSC0EN: Oscillator 0 Enable**
  - 0: Oscillator 0 is disabled
  - 1: Oscillator 0 is enabled
- **OSC1EN: Oscillator 1 Enable**
  - 0: Oscillator 1is disabled
  - 1: Oscillator 1is enabled

13.6.2 Clock Select

Name: CKSEL

Access Type: Read/Write

31	30	29	28	27	26	25	24
PBBDIV	-	-	-	-	PBBSEL		

23	22	21	20	19	18	17	16
PBADIV	-	-	-	-	PBASEL		

15	14	13	12	11	10	9	8
HSBDIV	-	-	-	-	HSBSEL		

7	6	5	4	3	2	1	0
CPUDIV	-	-	-	-	CPUSEL		

- **PBBDIV, PBBSEL: PBB Division and Clock Select**
  PBBDIV = 0: PBB clock equals main clock.
  PBBDIV = 1: PBB clock equals main clock divided by $2^{(PBBSEL+1)}$.
- **PBADIV, PBASEL: PBA Division and Clock Select**
  PBADIV = 0: PBA clock equals main clock.
  PBADIV = 1: PBA clock equals main clock divided by $2^{(PBASEL+1)}$.
- **HSBDIV, HSBSEL: HSB Division and Clock Select**
  For the AT32UC3B, HSBDIV always equals CPUDIV, and HSBSEL always equals CPUSEL, as the HSB clock is always equal to the CPU clock.
- **CPUDIV, CPUSEL: CPU Division and Clock Select**
  CPUDIV = 0: CPU clock equals main clock.
  CPUDIV = 1: CPU clock equals main clock divided by $2^{(CPUSEL+1)}$.

Note that if xxxDIV is written to 0, xxxSEL should also be written to 0 to ensure correct operation.

Also note that writing this register clears POSCSR:CKRDY. The register must not be re-written until CKRDY goes high.

13.6.3 Clock Mask

Name: CPU/HSB/PBA/PBBMASK

Access Type: Read/Write

31	30	29	28	27	26	25	24
MASK[31:24]							

23	22	21	20	19	18	17	16
MASK[23:16]							

15	14	13	12	11	10	9	8
MASK[15:8]							

7	6	5	4	3	2	1	0
MASK[7:0]							

- **MASK: Clock Mask**

  If bit n is cleared, the clock for module n is stopped. If bit n is set, the clock for module n is enabled according to the current power mode. The number of implemented bits in each mask register, as well as which module clock is controlled by each bit, is shown in Table 13-5.

Table 13-5. Maskable module clocks in AT32UC3B.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
0		FLASHC	INTC	HMATRIX
1	OCD	PBA bridge	GPIO	USBB
2	-	PBB bridge	PDCA	FLASHC
3	-	USBB	PM/RTC/EIC	MACB
4	-	MACB	ADC	SMC
5	-	PDCA	SPI0	SDRAMC
6	-	EBI	SPI1	-
7	-	-	TWI	-
8	-	-	USART0	-
9	-	-	USART1	-
10	-	-	USART2	-
11	-	-	USART3	-
12	-	-	PWM	-

Table 13-5. Maskable module clocks in AT32UC3B.

Bit	CPUMASK	HSBMASK	PBAMASK	PBBMASK
13	-	-	SSC	-
14	-	-	TC	-
31:15	-	-	-	-

13.6.4 PLL Control

Name: PLL0,1

Access Type: Read/Write

31	30	29	28	27	26	25	24
RESERVED		PLLCOUNT					

23	22	21	20	19	18	17	16
PLLMUL							

15	14	13	12	11	10	9	8
PLLDIV							

7	6	5	4	3	2	1	0
-	-	-	PLLOPT			PLLOSC	PLLEN

- **RESERVED: Reserved bitfields**

  Reserved for internal use. Always write to 0.
- **PLLCOUNT: PLL Count**

  Specifies the number of slow clock cycles before ISR:LOCKn will be set after PLLn has been written, or after PLLn has been automatically re-enabled after exiting a sleep mode.
- **PLLMUL: PLL Multiply Factor**
- **PLLDIV: PLL Division Factor**

  These bitfields determine the ratio of the PLL output frequency (voltage controlled oscillator frequency f_{VCO}) to the source oscillator frequency:

  $f_{VCO} = (PLLMUL+1)/(PLLDIV) \bullet f_{OSC}$ if PLLDIV > 0.

  $f_{VCO} = 2*(PLLMUL+1) \bullet f_{OSC}$ if PLLDIV = 0.

  If PLLOPT[1] field is set to 0:

  $f_{PLL} = f_{VCO.}$

  If PLLOPT[1] field is set to 1:

  $f_{PLL} = f_{VCO} / 2.$

  Note that the MUL field cannot be equal to 0 or 1, or the behavior of the PLL will be undefined.

- **PLLOPT: PLL Option**

  Select the operating range for the PLL.

  PLLOPT[0]: Select the VCO frequency range.

  PLLOPT[1]: Enable the extra output divider.

  PLLOPT[2]: Disable the Wide-Bandwidth mode (Wide-Bandwidth mode allows a faster startup time and out-of-lock time).

Table 13-6. PLLOPT Fields Description in AT32UC3B

	Description
PLLOPT[0]: VCO frequency	
0	160MHz<f_{vco}<240MHz
1	80MHz<f_{vco}<180MHz
PLLOPT[1]: Output divider	
0	$f_{PLL} = f_{vco}$
1	$f_{PLL} = f_{vco}/2$
PLLOPT[2]	
0	Wide Bandwidth Mode enabled
1	Wide Bandwidth Mode disabled

- **PLLOSC: PLL Oscillator Select**
  0: Oscillator 0 is the source for the PLL.
  1: Oscillator 1 is the source for the PLL.
- **PLLEN: PLL Enable**
  0: PLL is disabled.
  1: PLL is enabled.

13.6.5 PM Oscillator 0/1 Control

Register name OSCCTRL0,1

Register access Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	STARTUP		

7	6	5	4	3	2	1	0
-	-	-	-	-	MODE		

- **MODE: Oscillator Mode**

Choose between crystal, or external clock

0: External clock connected on XIN, XOUT can be used as an I/O (no crystal)
1: Crystal is connected to XIN/XOUT - Oscillator is used with automatic gain control
2 to 7: reserved

- **STARTUP: Oscillator Startup Time**

Select startup time for the oscillator.

Table 13-7. Startup time for oscillators 0 and 1

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCOsc = 115 kHz)
0	0	0
1	64	560 us
2	128	1.1 ms
3	2048	18 ms
4	4096	36 ms
5	8192	71 ms
6	16384	142 ms
7	*Reserved*	*Reserved*

13.6.6 PM 32 KHz Oscillator Control Register

Register name OSCCTRL32

Register access Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	STARTUP		

15	14	13	12	11	10	9	8
-	-	-	-	-	MODE		

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	OSC32EN

Note: This register is only reset by Power-On Reset

- **OSC32EN: Enable the 32 KHz oscillator**
  0: 32 KHz Oscillator is disabled
  1: 32 KHz Oscillator is enabled
- **MODE: Oscillator Mode**

Choose between crystal, or external clock

0: External clock connected on XIN32, XOUT32 can be used as a I/O (no crystal)
1: Crystal is connected to XIN32/XOUT32 - Oscillator is used with automatic gain control
2 to 7: reserved

- **STARTUP: Oscillator Startup Time**

Select startup time for 32 KHz oscillator

Table 13-8. Startup time for 32 KHz oscillator

STARTUP	Number of RC oscillator clock cycle	Approximative Equivalent time (RCOsc = 115 kHz)
0	0	0
1	128	1.1 ms
2	8192	72.3 ms
3	16384	143 ms
4	65536	570 ms
5	131072	1.1 s
6	262144	2.3 s
7	524288	4.6 s

13.6.7 Interrupt Enable/Disable/Mask/Status/Clear

Name: IER/IDR/IMR/ISR/ICR

Access Type: IER/IDR/ICR: Write-only

IMR/ISR: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

- **BODDET: Brown out detection**
  Set to 1 when 0 to 1 transition on POSCSR:BODDET bit is detected: BOD has detected that power supply is going below BOD reference value.
- **OSC32RDY: 32 KHz oscillator Ready**
  Set to 1 when 0 to 1 transition on the POSCSR:OSC32RDY bit is detected: The 32 KHz oscillator is stable and ready to be used as clock source.
- **OSC1RDY: Oscillator 1 Ready**
  Set to 1 when 0 to 1 transition on the POSCSR:OSC1RDY bit is detected: Oscillator 1 is stable and ready to be used as clock source.
- **OSC0RDY: Oscillator 0 Ready**
  Set to 1 when 0 to 1 transition on the POSCSR:OSC1RDY bit is detected: Oscillator 1 is stable and ready to be used as clock source.
- **MSKRDY: Mask Ready**
  Set to 1 when 0 to 1 transition on the POSCSR:MSKRDY bit is detected: Clocks are now masked according to the (CPU/HSB/PBA/PBB)_MASK registers.
- **CKRDY: Clock Ready**
  0: The CKSEL register has been written, and the new clock setting is not yet effective.
  1: The synchronous clocks have frequencies as indicated in the CKSEL register.
  Note: Writing ICR:CKRDY to 1 has no effect.
- **LOCK1: PLL1 locked**
  Set to 1 when 0 to 1 transition on the POSCSR:LOCK1 bit is detected: PLL 1 is locked and ready to be selected as clock source.
- **LOCK0: PLL0 locked**
  Set to 1 when 0 to 1 transition on the POSCSR:LOCK0 bit is detected: PLL 0 is locked and ready to be selected as clock source.

The effect of writing or reading the bits listed above depends on which register is being accessed:

- **IER (Write-only)**
  - 0: No effect
  - 1: Enable Interrupt
- **IDR (Write-only)**
  - 0: No effect
  - 1: Disable Interrupt
- **IMR (Read-only)**
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
- **ISR (Read-only)**
  - 0: An interrupt event has not occurred or has been previously cleared
  - 1: An interrupt event has not occurred
- **ICR (Write-only)**
  - 0: No effect
  - 1: Clear corresponding event

13.6.8 Power and Oscillators Status

Name: POSCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	BODDET

15	14	13	12	11	10	9	8
-	-	-	-	-	-	OSC32RDY	OSC1RDY

7	6	5	4	3	2	1	0
OSC0RDY	MSKRDY	CKRDY	-	-	-	LOCK1	LOCK0

- **BODDET: Brown out detection**
  0: No BOD event
  1: BOD has detected that power supply is going below BOD reference value.
- **OSC32RDY: 32 KHz oscillator Ready**
  0: The 32 KHz oscillator is not enabled or not ready.
  1: The 32 KHz oscillator is stable and ready to be used as clock source.
- **OSC1RDY: OSC1 ready**
  0: Oscillator 1 not enabled or not ready.
  1: Oscillator 1 is stable and ready to be used as clock source.
- **OSC0RDY: OSC0 ready**
  0: Oscillator 0 not enabled or not ready.
  1: Oscillator 0 is stable and ready to be used as clock source.
- **MSKRDY: Mask ready**
  0: Mask register has been changed, masking in progress.
  1: Clock are masked according to xxx_MASK
- **CKRDY:**
  0: The CKSEL register has been written, and the new clock setting is not yet effective.
  1: The synchronous clocks have frequencies as indicated in the CKSEL register.
- **LOCK1: PLL 1 locked**
  0:PLL 1 is unlocked
  1:PLL 1 is locked, and ready to be selected as clock source.
- **LOCK0: PLL 0 locked**
  0: PLL 0 is unlocked
  1: PLL 0 is locked, and ready to be selected as clock source.

13.6.9 Generic Clock Control

Name: GCCTRL

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
DIV[7:0]							

7	6	5	4	3	2	1	0
-	-	-	DIVEN	-	CEN	PLLSEL	OSCSEL

There is one GCCTRL register per generic clock in the design.

- **DIV: Division Factor**
- **DIVEN: Divide Enable**
  - 0: The generic clock equals the undivided source clock.
  - 1: The generic clock equals the source clock divided by 2*(DIV+1).
- **CEN: Clock Enable**
  - 0: Clock is stopped.
  - 1: Clock is running.
- **PLLSEL: PLL Select**
  - 0: Oscillator is source for the generic clock.
  - 1: PLL is source for the generic clock.
- **OSCSEL: Oscillator Select**
  - 0: Oscillator (or PLL) 0 is source for the generic clock.
  - 1: Oscillator (or PLL) 1 is source for the generic clock.

13.6.10 Reset Cause

Name: RCAUSE

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	JTAGHARD	OCDRST

7	6	5	4	3	2	1	0
CPUERR	-	-	JTAG	WDT	EXT	BOD	POR

- **POR Power-on Reset**
  The CPU was reset due to the supply voltage being lower than the power-on threshold level.
- **BOD: Brown-out Reset**
  The CPU was reset due to the supply voltage being lower than the brown-out threshold level.
- **EXT: External Reset Pin**
  The CPU was reset due to the RESET pin being asserted.
- **WDT: Watchdog Reset**
  The CPU was reset because of a watchdog timeout.
- **JTAG: JTAG reset**
  The CPU was reset by setting the bit RC_CPU in the JTAG reset register.
- **CPUERR: CPU Error**
  The CPU was reset because it had detected an illegal access.
- **OCDRST: OCD Reset**
  The CPU was reset because the RES strobe in the OCD Development Control register has been written to one.
- **JTAGHARD: JTAG Hard Reset**
  The chip was reset by setting the bit RC_OCD in the JTAG reset register or by using the JTAG HALT instruction.

13.6.11 BOD Control

BOD Level register

Register name BOD

Register access Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CTRL	

7	6	5	4	3	2	1	0
-	HYST	LEVEL					

- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
- **FCD: BOD Fuse calibration done**
  Set to 1 when CTRL, HYST and LEVEL fields has been updated by the Flash fuses after power-on reset or Flash fuses update
  If one, the CTRL, HYST and LEVEL values will not be updated again by Flash fuses
  Can be cleared to allow subsequent overwriting of the value by Flash fuses
- **CTRL: BOD Control**
  0: BOD is off
  1: BOD is enabled and can reset the chip
  2: BOD is enabled and but cannot reset the chip. Only interrupt will be sent to interrupt controller, if enabled in the IMR register.
  3: BOD is off
- **HYST: BOD Hysteresis**
  0: No hysteresis
  1: Hysteresis On
- **LEVEL: BOD Level**
  This field sets the triggering threshold of the BOD. See Electrical Characteristics for actual voltage levels.
  Note that any change to the LEVEL field of the BOD register should be done with the BOD deactivated to avoid spurious reset or interrupt.

13.6.12 RC Oscillator Calibration

Register name RCCR

Register access Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	CALIB	

7	6	5	4	3	2	1	0
CALIB							

- **CALIB: Calibration Value**
  Calibration Value for the RC oscillator.
- **FCD: Flash Calibration Done**
  Set to 1 when CTRL, HYST, and LEVEL fields have been updated by the Flash fuses after power-on reset, or after Flash fuses are reprogrammed. The CTRL, HYST and LEVEL values will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.
- **KEY: Register Write protection**
  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.

13.6.13 Bandgap Calibration

Register name BGCR

Register access Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	CALIB		

- **KEY: Register Write protection**

This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.

- **CALIB: Calibration value**

Calibration value for Bandgap. See Electrical Characteristics for voltage values.

- **FCD: Flash Calibration Done**

  Set to 1 when the CALIB field has been updated by the Flash fuses after power-on reset or when the Flash fuses are reprogrammed. The CALIB field will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.

13.6.14 PM Voltage Regulator Calibration Register

Register name VREGCR

Register access Read/Write

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	FCD

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	CALIB		

- **KEY: Register Write protection**

  This field must be written twice, first with key value 0x55, then 0xAA, for a write operation to have an effect.
- **CALIB: Calibration value**

  Calibration value for Voltage Regulator. See Electrical Characteristics for voltage values.
- **FCD: Flash Calibration Done**

  Set to 1 when the CALIB field has been updated by the Flash fuses after power-on reset or when the Flash fuses are reprogrammed. The CALIB field will not be updated again by the Flash fuses until a new power-on reset or the FCD field is written to zero.

13.6.15 AWEN USB Wake Up Enable

Register name PM_AWEN

Register access Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	USB_WAKEN

- **USB_WAKEN : Wake Up Enable Register**
  Set to 1 to enable USB wake up.

13.6.16 General Purpose Low-power register

Register name GPLP

Register access Read/Write

31	30	29	28	27	26	25	24
GPLP							

23	22	21	20	19	18	17	16
GPLP							

15	14	13	12	11	10	9	8
GPLP							

7	6	5	4	3	2	1	0
GPLP							

These registers are general purpose 32-bit registers that are reset only by power-on-reset. Any other reset will keep the content of these registers untouched.

14. Real Time Counter (RTC)

Rev: 2.0.0.1

14.1 Features

- **32-bit real-time counter with 16-bit prescaler**
- **Clocked from RC oscillator or 32 KHz oscillator**
- **High resolution: Max count frequency 16 KHz**
- **Long delays**
  - **Max timeout 272 years**
- **Extremely low power consumption**
- **Available in all sleep modes except Static**
- **Optional wrap at max value**
- **Interrupt on wrap**

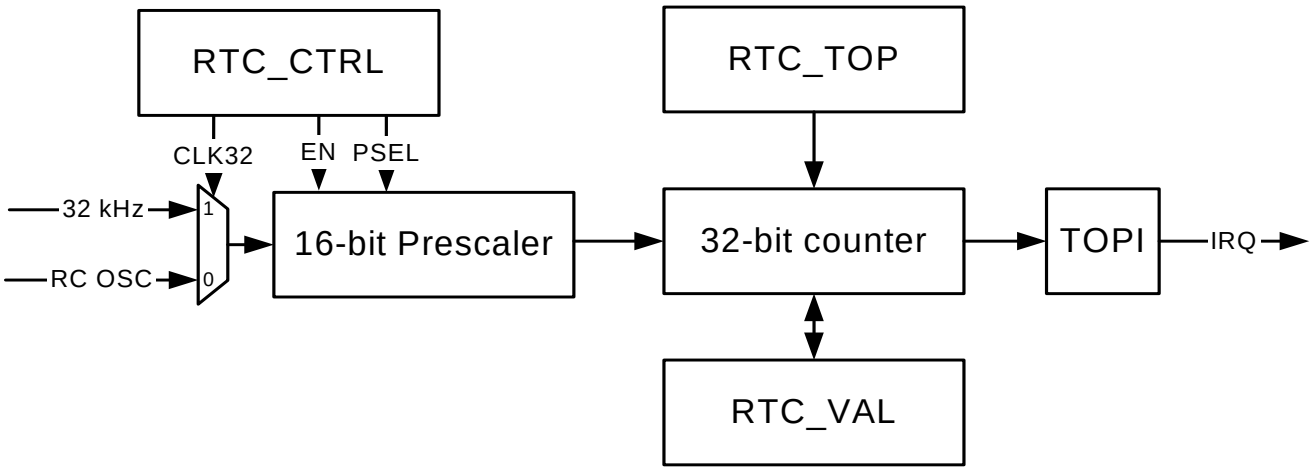
14.2 Description

The Real Time Counter (RTC) enables periodic interrupts at long intervals, or accurate measurement of real-time sequences. The RTC is fed from a 16-bit prescaler, which is clocked from the RC oscillator or the 32 KHz oscillator. Any tapping of the prescaler can be selected as clock source for the RTC, enabling both high resolution and long timeouts. The prescaler cannot be written directly, but can be cleared by the user.

The RTC can generate an interrupt when the counter wraps around the value stored in the top register, producing accurate periodic interrupts.

14.3 Block Diagram

Figure 14-1. Real Time Counter module block diagram



14.4 Product Dependencies

14.4.1 Power Management

The RTC is continuously clocked, and remains operating in all sleep modes except Static. Interrupts are not available in DeepStop mode.

14.4.2 Interrupt

The RTC interrupt line is connected to one of the internal sources of the interrupt controller. Using the RTC interrupt requires the interrupt controller to be programmed first.

14.4.3 Debug Operation

The RTC prescaler is frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

14.4.4 Clocks

The RTC can use the internal RC oscillator as clock source. This oscillator is always enabled whenever these modules are active. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator (f_{RC}).

The RTC can also use the 32 KHz crystal oscillator as clock source. This oscillator must be enabled before use. Please refer to the Power Manager chapter for details.

14.5 Functional Description

14.5.1 RTC operation

14.5.1.1 Source clock

The RTC is enabled by writing the EN bit in the CTRL register to 1. The 16-bit prescaler will then increment on the selected clock. The prescaler cannot be read or written, but it can be reset by writing the PCLR strobe.

The CLK32 bit selects either the RC oscillator or the 32 KHz oscillator as clock source for the prescaler.

The PSEL bitfield selects the prescaler tapping, selecting the source clock for the RTC:

$$f_{RTC} = 2^{-(PSEL+1)} * (f_{RC} \text{ or } 32\ KHz)$$

14.5.1.2 Counter operation

When enabled, the RTC will increment until it reaches TOP, and then wrap to 0x0. The status bit TOPI in ISR is set when this occurs.

The RTC count value can be read from or written to the register VAL. Due to synchronization, continuous reading of the VAL with the lowest prescaler setting will skip every other value.

14.5.1.3 RTC Interrupt

Writing the TOPI bit in IER enables the RTC interrupt, while writing the corresponding bit in IDR disables the RTC interrupt. IMR can be read to see whether or not the interrupt is enabled. If enabled, an interrupt will be generated if the TOPI flag in ISR is set. The flag can be cleared by writing TOPI in ICR to one.

The RTC interrupt can wake the CPU from all sleep modes except DeepStop and Static mode.

14.5.1.4 RTC wakeup

The RTC can also wake up the CPU directly without triggering an interrupt when the TOPI flag in ISR is set. In this case, the CPU will continue executing from the instruction following the sleep instruction.

This direct RTC wakeup is enabled by writing the WAKE_EN bit in the CTRL register to one. When the CPU wakes from sleep, the WAKE_EN bit must be written to zero to clear the internal wake signal to the sleep controller, otherwise a new sleep instruction will have no effect.

The RTC wakeup is available in all sleep modes except Static mode. The RTC wakeup can be configured independently of the RTC interrupt.

14.5.1.5 Busy bit

Due to the crossing of clock domains, the RTC uses a few clock cycles to propagate the values stored in CTRL, TOP, and VAL to the RTC. The BUSY bit in CTRL indicates that a register write is still going on and all writes to TOP, CTRL, and VAL will be discarded until BUSY goes low again.

14.6 User Interface

Offset	Register	Register Name	Access	Reset
0x00	RTC Control	CTRL	Read/Write	0x0
0x04	RTC Value	VAL	Read/Write	0x0
0x08	RTC Top	TOP	Read/Write	0x0
0x10	RTC Interrupt Enable	IER	Write-only	0x0
0x14	RTC Interrupt Disable	IDR	Write-only	0x0
0x18	RTC Interrupt Mask	IMR	Read-only	0x0
0x1C	RTC Interrupt Status	ISR	Read-only	0x0
0x20	RTC Interrupt Clear	ICR	Write-only	0x0

14.6.1 RTC Control

Name: CTRL

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	PSEL			

7	6	5	4	3	2	1	0
-	-	-	BUSY	CLK32	WAKE_EN	PCLR	EN

- **PSEL: Prescale Select**
  Selects prescaler bit PSEL as source clock for the RTC.
- **BUSY: RTC busy**
  0: The RTC accepts writes to TOP, VAL, and CTRL.
  1: The RTC is busy and will discard writes to TOP, VAL, and CTRL.
- **CLK32: 32 KHz oscillator select**
  0: The RTC uses the RC oscillator as clock source
  1: The RTC uses the 32 KHz oscillator as clock source
- **WAKE_EN: Wakeup enable**
  0: The RTC does not wake up the CPU from sleep modes
  1: The RTC wakes up the CPU from sleep modes.

- **PCLR: Prescaler Clear**
  Writing 1 to this strobe clears the prescaler.
- **EN: Enable**
  0: The RTC is disabled
  1: The RTC is enabled

14.6.2 RTC Value

Name: VAL

Access Type: Read/Write

31	30	29	28	27	26	25	24
VAL[31:24]							

23	22	21	20	19	18	17	16
VAL[23:16]							

15	14	13	12	11	10	9	8
VAL[15:8]							

7	6	5	4	3	2	1	0
VAL[7:0]							

- **VAL: RTC Value**

  This value is incremented on every rising edge of the source clock.

14.6.3 RTC Top

Name: TOP

Access Type: Read/Write

31	30	29	28	27	26	25	24
TOP[31:24]							

23	22	21	20	19	18	17	16
TOP[23:16]							

15	14	13	12	11	10	9	8
TOP[15:8]							

7	6	5	4	3	2	1	0
TOP[7:0]							

- **TOP: RTC Top Value**
  VAL wraps at this value.

14.6.4 RTC Interrupt Enable/Disable/Mask/Status/Clear

Name: IER/IDR/IMR/ISR/ICR

Access Type: IER/IDR/ICR: Write-only

IMR/ISR: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	TOPI

- **TOPI: Top Interrupt**
  VAL has wrapped at its top value.

The effect of writing or reading this bit depends on which register is being accessed:

- **IER (Write-only)**
  0: No effect
  1: Enable Interrupt
- **IDR (Write-only)**
  0: No effect
  1: Disable Interrupt
- **IMR (Read-only)**
  0: Interrupt is disabled
  1: Interrupt is enabled
- **ISR (Read-only)**
  0: An interrupt event has occurred
  1: An interrupt even has not occurred
- **ICR (Write-only)**
  0: No effect
  1: Clear interrupt even

15. Watchdog Timer (WDT)

Rev: 2.0.0.0

15.1 Features

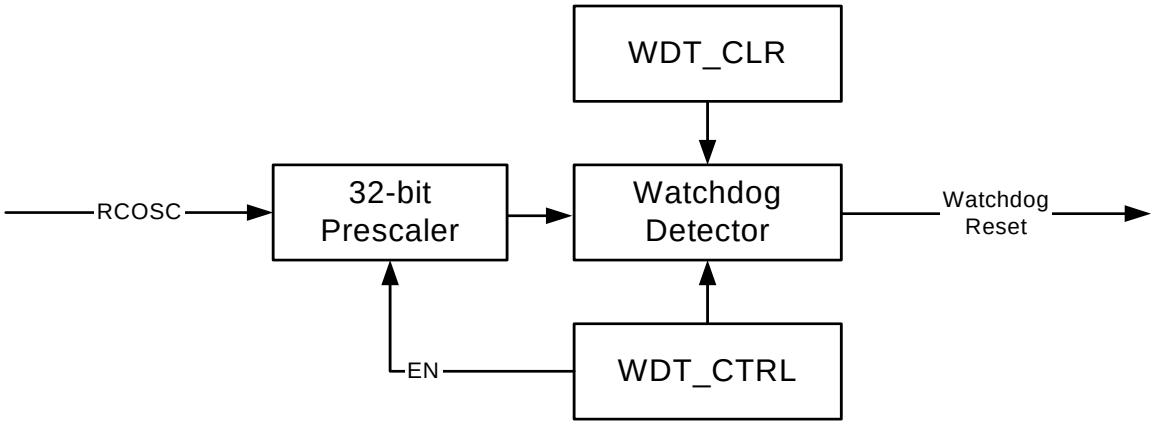
- **Watchdog Timer counter with 16-bit prescaler**
- **Clocked from RC oscillator**

15.2 Description

The Watchdog Timer (WDT) has a prescaler generating a timeout period. This prescaler is clocked from the RC oscillator. The watchdog timer must be periodically reset by software within the timeout period, otherwise, the device is reset and starts executing from the boot vector. This allows the device to recover from a condition that has caused the system to be unstable.

15.3 Block Diagram

Figure 15-1. Watchdog Timer module block diagram



15.4 Product Dependencies

15.4.1 Power Management

When the WDT is enabled, the WDT remains clocked in all sleep modes, and it is not possible to enter Static mode.

15.4.2 Debug Operation

The WDT prescaler is frozen during debug operation, unless the OCD system keeps peripherals running in debug operation.

15.4.3 Clocks

The WDT can use the internal RC oscillator as clock source. This oscillator is always enabled whenever these modules are active. Please refer to the Electrical Characteristics chapter for the characteristic frequency of this oscillator (f_{RC}).

15.5 Functional Description

The WDT is enabled by writing the EN bit in the CTRL register to one. This also enables the RC clock for the prescaler. The PSEL bitfield in the same register selects the watchdog timeout period:

$$T_{WDT} = 2^{(PSEL+1)} / f_{RC}$$

To avoid accidental disabling of the watchdog, the CTRL register must be written twice, first with the KEY field set to 0x55, then 0xAA without changing the other bitfields. Failure to do so will cause the write operation to be ignored, and CTRL does not change value.

The CLR register must be written with any value with regular intervals shorter than the watchdog timeout period. Otherwise, the device will receive a soft reset, and the code will start executing from the boot vector.

When the WDT is enabled, it is not possible to enter Static mode. Attempting to do so will result in entering Shutdown mode, leaving the WDT operational.

15.6 User Interface

Offset	Register	Register Name	Access	Reset
0x00	WDT Control	CTRL	Read/Write	0x0
0x04	WDT Clear	CLR	Write-only	0x0

15.6.1 WDT Control

Name: CTRL

Access Type: Read/Write

31	30	29	28	27	26	25	24
KEY[7:0]							
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	PSEL				
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **KEY**

  This bitfield must be written twice, first with key value 0x55, then 0xAA, for a write operation to be effective. This bitfield always reads as zero.
- **PSEL: Prescale Select**

  Prescaler bit PSEL is used as watchdog timeout period.
- **EN: WDT Enable**

  0: WDT is disabled.

  1: WDT is enabled.

15.6.2 WDT Clear

Name: CLR

Access Type: Write-only

When the watchdog timer is enabled, this register must be periodically written, with any value, within the watchdog timeout period, to prevent a watchdog reset.

16. Interrupt Controller (INTC)

Rev: 1.0.1.0

16.1 Description

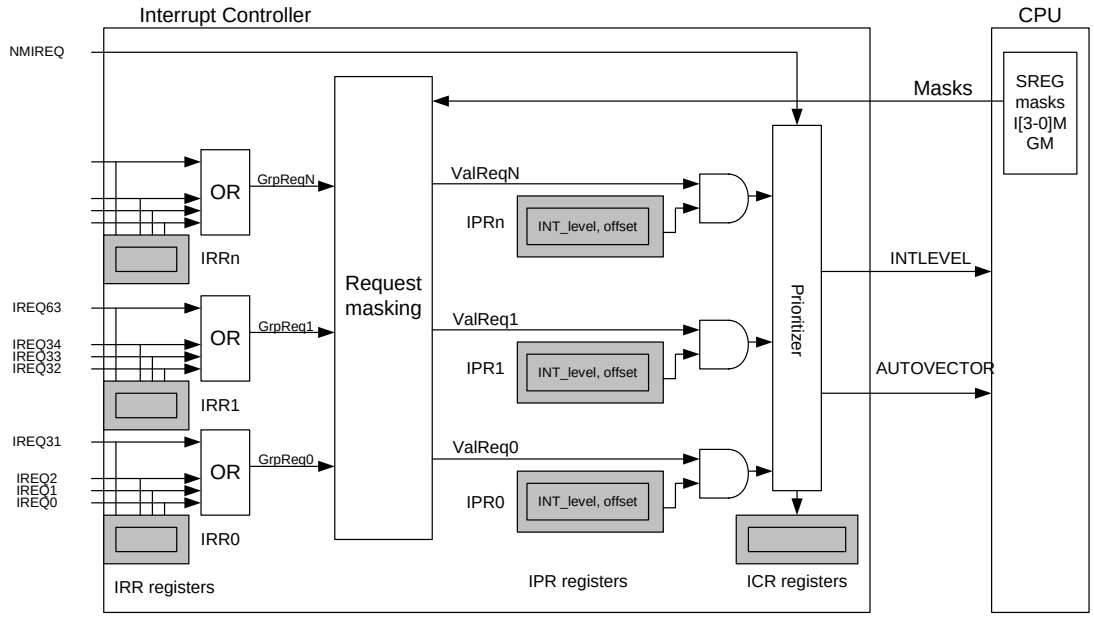
The INTC collects interrupt requests from the peripherals, prioritizes them, and delivers an interrupt request and an autovector to the CPU. The AVR32 architecture supports 4 priority levels for regular, maskable interrupts, and a Non-Maskable Interrupt (NMI).

The INTC supports up to 64 groups of interrupts. Each group can have up to 32 interrupt request lines, these lines are connected to the peripherals. Each group has an Interrupt Priority Register (IPR) and an Interrupt Request Register (IRR). The IPRs are used to assign a priority level and an autovector to each group, and the IRRs are used to identify the active interrupt request within each group. If a group has only one interrupt request line, an active interrupt group uniquely identifies the active interrupt request line, and the corresponding IRR is not needed. The INTC also provides one Interrupt Cause Register (ICR) per priority level. These registers identify the group that has a pending interrupt of the corresponding priority level. If several groups have an pending interrupt of the same level, the group with the highest number takes priority.

16.2 Block Diagram

Figure 16-1 on page 91 gives an overview of the INTC. The grey boxes represent registers that can be accessed via the Peripheral Bus (PB). The interrupt requests from the peripherals (IREQn) and the NMI are input on the left side of the figure. Signals to and from the CPU are on the right side of the figure.

Figure 16-1. Overview of the Interrupt Controller



16.3 Operation

All of the incoming interrupt requests (IREQs) are sampled into the corresponding Interrupt Request Register (IRR). The IRRs must be accessed to identify which IREQ within a group that is active. If several IREQs within the same group is active, the interrupt service routine must pri-

oritize between them. All of the input lines in each group are logically-ORed together to form the GrpReqN lines, indicating if there is a pending interrupt in the corresponding group.

The Request Masking hardware maps each of the GrpReq lines to a priority level from INT0 to INT3 by associating each group with the INTLEVEL field in the corresponding IPR register. The GrpReq inputs are then masked by the I0M, I1M, I2M, I3M and GM mask bits from the CPU status register. Any interrupt group that has a pending interrupt of a priority level that is not masked by the CPU status register, gets its corresponding ValReq line asserted.

The Prioritizer hardware uses the ValReq lines and the INTLEVEL field in the IPRs to select the pending interrupt of the highest priority. If a NMI interrupt is pending, it automatically gets highest priority of any pending interrupt. If several interrupt groups of the highest pending interrupt level have pending interrupts, the interrupt group with the highest number is selected.

Interrupt level (INTLEVEL) and handler autovector offset (AUTOVECTOR) of the selected interrupt are transmitted to the CPU for interrupt handling and context switching. The CPU doesn't need to know which interrupt is requesting handling, but only the level and the offset of the handler address. The IRR registers contain the interrupt request lines of the groups and can be read via PB for checking which interrupts of the group are actually active.

Masking of the interrupt requests is done based on five interrupt mask bits of the CPU status register, namely interrupt level 3 mask (I3M) to interrupt level 0 mask (I0M), and Global interrupt mask (GM). An interrupt request is masked if either the Global interrupt mask or the corresponding interrupt level mask bit is set.

16.3.1 Non maskable interrupts

A NMI request has priority over all other interrupt requests. NMI has a dedicated exception vector address defined by the AVR32 architecture, so AUTOVECTOR is undefined when INTLEVEL indicates that an NMI is pending.

16.3.2 CPU response

When the CPU receives an interrupt request it checks if any other exceptions are pending. If no exceptions of higher priority are pending, interrupt handling is initiated. When initiating interrupt handling, the corresponding interrupt mask bit is set automatically for this and lower levels in status register. E.g, if interrupt on level 3 is approved for handling the interrupt mask bits I3M, I2M, I1M, and I0M are set in status register. If interrupt on level 1 is approved the masking bits I1M, and I0M are set in status register. The handler offset is calculated from AUTOVECTOR and EVBA and a change-of-flow to this address is performed.

Setting of the interrupt mask bits prevents the interrupts from the same and lower levels to be passed trough the interrupt controller. Setting of the same level mask bit prevents also multiple request of the same interrupt to happen.

It is the responsibility of the handler software to clear the interrupt request that caused the interrupt before returning from the interrupt handler. If the conditions that caused the interrupt are not cleared, the interrupt request remains active.

16.3.3 Clearing an interrupt request

Clearing of the interrupt request is done by writing to registers in the corresponding peripheral module, which then clears the corresponding NMIREQ/IREQ signal.

The recommended way of clearing an interrupt request is a store operation to the controlling peripheral register, followed by a dummy load operation from the same register. This causes a

pipeline stall, which prevents the interrupt from accidentally re-triggering in case the handler is exited and the interrupt mask is cleared before the interrupt request is cleared.

16.4 User Interface

This chapter lists the INTC registers are accessible through the PB bus. The registers are used to control the behaviour and read the status of the INTC.

16.4.1 Memory Map

The following table shows the address map of the INTC registers, relative to the base address of the INTC.

Table 16-1. INTC address map

Offset	Register	Name	Access	Reset Value
0	Interrupt Priority Register 0	IPR0	Read/Write	0x0000_0000
4	Interrupt Priority Register 1	IPR1	Read/Write	0x0000_0000
...	...	...	...	...
252	Interrupt Priority Register 63	IPR63	Read/Write	0x0000_0000
256	Interrupt Request Register 0	IRR0	Read-only	N/A
260	Interrupt Request Register 1	IRR1	Read-only	N/A
...	...	...	...	...
508	Interrupt Request Register 63	IRR63	Read-only	N/A
512	Interrupt Cause Register 3	ICR3	Read-only	N/A
516	Interrupt Cause Register 2	ICR2	Read-only	N/A
520	Interrupt Cause Register 1	ICR1	Read-only	N/A
524	Interrupt Cause Register 0	ICR0	Read-only	N/A

16.4.2 Interrupt Request Map

The mapping of interrupt requests from peripherals to INTREQs is presented in the Peripherals Section.

16.4.3 Interrupt Request Registers

Register Name: IRR0...IRR63

Access Type: Read-only

31	30	29	28	27	26	25	24
IRR(32*x+31)	IRR(32*x+30)	IRR(32*x+29)	IRR(32*x+28)	IRR(32*x+27)	IRR(32*x+26)	IRR(32*x+25)	IRR(32*x+24)
23	22	21	20	19	18	17	16
IRR(32*x+23)	IRR(32*x+22)	IRR(32*x+21)	IRR(32*x+20)	IRR(32*x+19)	IRR(32*x+18)	IRR(32*x+17)	IRR(32*x+16)
15	14	13	12	11	10	9	8
IRR(32*x+15)	IRR(32*x+14)	IRR(32*x+13)	IRR(32*x+12)	IRR(32*x+11)	IRR(32*x+10)	IRR(32*x+9)	IRR(32*x+8)
7	6	5	4	3	2	1	0
IRR(32*x+7)	IRR(32*x+6)	IRR(32*x+5)	IRR(32*x+4)	IRR(32*x+3)	IRR(32*x+2)	IRR(32*x+1)	IRR(32*x+0)

- **IRR: Interrupt Request line**

0 = No interrupt request is pending on this input request input.

1 = An interrupt request is pending on this input request input.

The are 64 IRRs, one for each group. Each IRR has 32 bits, one for each possible interrupt request, for a total of 2048 possible input lines. The IRRs are read by the software interrupt handler in order to determine which interrupt request is pending. The IRRs are sampled continuously, and are read-only.

16.4.4 Interrupt Priority Registers

Register Name: IPR0...IPR63

Access Type: Read/Write

31	30	29	28	27	26	25	24
INTLEVEL[1:0]		-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	AUTOVECTOR[13:8]					
7	6	5	4	3	2	1	0
AUTOVECTOR[7:0]							

- **INTLEVEL: Interrupt level associated with this group**

Indicates the EVBA-relative offset of the interrupt handler of the corresponding group:

INTLEVEL[1:0]		Priority
0	0	INT0
0	1	INT1
1	0	INT2
1	1	INT3

- **AUTOVECTOR: Autovector address for this group**

Handler offset is used to give the address of the interrupt handler. The LSB should be written to zero to give halfword alignment

16.4.5 Interrupt Cause Registers

Register Name: ICR0...ICR3

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	CAUSE					

- **CAUSE: Interrupt group causing interrupt of priority n**

ICRn identifies the group with the highest priority that has a pending interrupt of level n. If no interrupts of level n are pending, or the priority level is masked, the value of ICRn is UNDEFINED.

17. External Interrupts Controller (EIC)

Rev: 2.0.0.1

17.1 Features

- **Dedicated interrupt requests for each interrupt**
- **Individually maskable interrupts**
- **Interrupt on rising or falling edge**
- **Interrupt on high or low level**
- **Asynchronous interrupts for sleep modes without clock**
- **Filtering of interrupt lines**
- **Keypad scan support**
- **Maskable NMI interrupt**

17.2 Description

The External Interrupt Module allows pins to be configured as external interrupts. Each pin has its own interrupt request and can be individually masked. Each pin can generate an interrupt on rising or falling edge, or high or low level. Every line has a configurable filter too remove spikes on the interrupt lines. Every interrupt pin can also be configured to be asynchronous to wake up the part from sleep modes where the clock has been disabled.

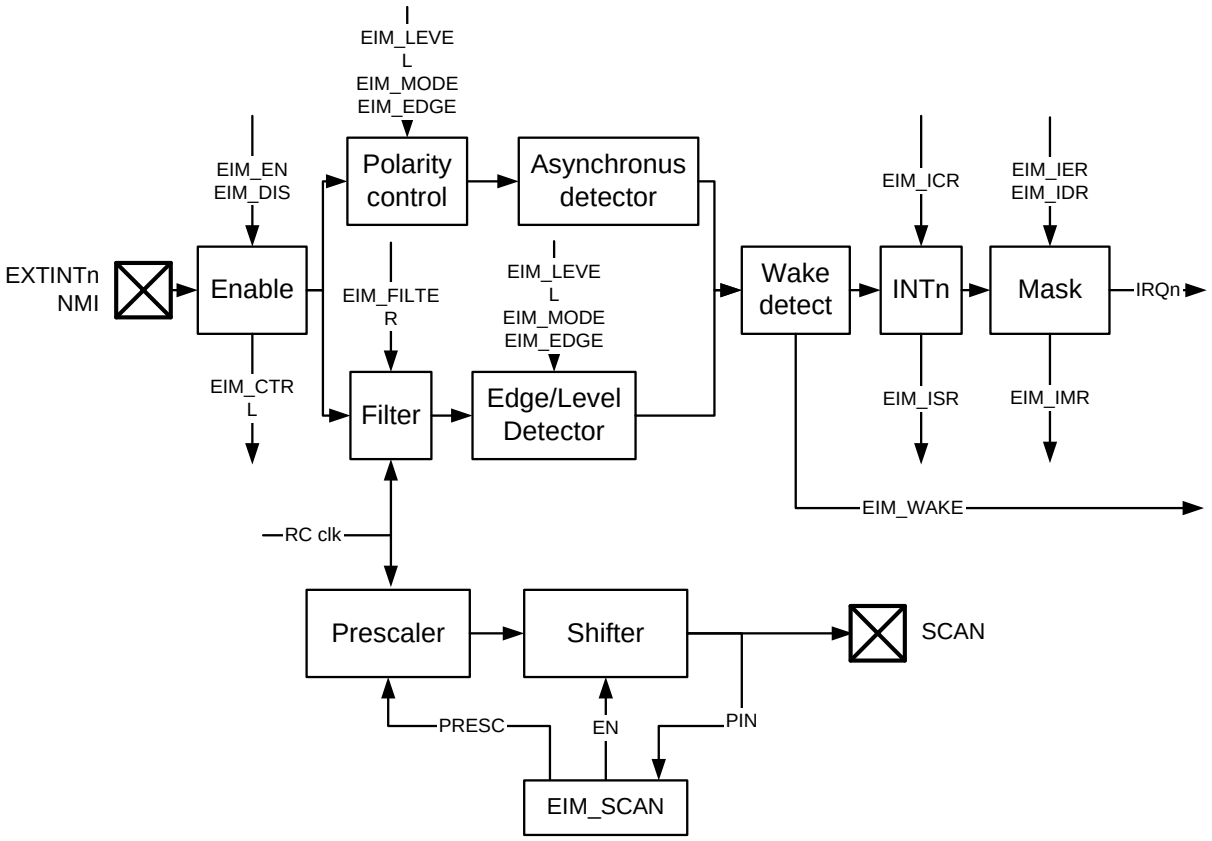
A Non-Maskable Interrupt (NMI) is also supported. This has the same properties as the other external interrupts, but is connected to the NMI request of the CPU, enabling it to interrupt any other interrupt mode.

The External Interrupt Module has support for keypad scanning for keypads laid out in rows and columns. Columns are driven by a separate set of scanning outputs, while rows are sensed by the external interrupt lines. The pressed key will trigger an interrupt, which can be identified through the user registers of the module.

The External Interrupt Module can wake up the part from sleep modes without triggering an interrupt. In this mode, code execution starts from the instruction following the sleep instruction.

17.3 Block Diagram

Figure 17-1. External Interrupt Module block diagram



17.4 Product Dependencies

17.4.1 I/O Lines

The External Interrupt and keypad scan pins are multiplexed with PIO lines. To act as external interrupts, these pins must be configured as inputs pins by the PIO controller. It is also possible to trigger the interrupt by driving these pins from registers in the PIO controller, or another peripheral output connected to the same pin.

17.4.2 Power Management

All interrupts are available in every sleep mode. However, in sleep modes where the clock is stopped, asynchronous interrupts must be selected.

17.4.3 Interrupt

The external interrupt lines are connected to internal sources of the interrupt controller. Using the external interrutps requires the interrupt controller to be programmed first.

Using the Non-Maskable Interrupt does not require the interrupt controller to be programmed.

17.5 Functional Description

17.5.1 External Interrupts

To enable an external interrupt EXTINTn must be written to 1 in register EN. Similarly, writing EXTINTn to 1 in register DIS disables the interrupt. The status of each Interrupt line can be observed in the CTRL register.

Each external interrupt pin EXTINTn can be configured to produce an interrupt on rising or falling edge, or high or low level. External interrupts are configured by the MODE, EDGE, and LEVEL registers. Each interrupt n has a bit INTn in each of these registers.

Similarly, each interrupt has a corresponding bit in each of the interrupt control and status registers. Writing 1 to the INTn strobe in IER enables the external interrupt on pin EXTINTn, while writing 1 to INTn in IDR disables the external interrupt. IMR can be read to check which interrupts are enabled. When the interrupt triggers, the corresponding bit in ISR will be set. The flag remains set until the corresponding strobe bit in ICR is written to 1.

Writing INTn in MODE to 0 enables edge triggered interrupts, while writing the bit to 1 enables level triggered interrupts.

If EXTINTn is configured as an edge triggered interrupt, writing INTn in EDGE to 0 will trigger the interrupt on falling edge, while writing the bit to 1 will trigger the interrupt on rising edge.

If EXTINTn is configured as a level triggered interrupt, writing INTn in LEVEL to 0 will trigger the interrupt on low level, while writing the bit to 1 will trigger the interrupt on high level.

To remove spikes that are longer than the clock period in the current mode each external interrupt contains a filter that can be enabled by writing 1 to INTn to FILTER.

Each interrupt line can be made asynchronous by writing 1 to INTn in the ASYNC register. This will route the interrupt signal through the asynchronous path of the module. All edge interrupts will be interpreted as level interrupts and the filter is disabled.

17.5.1.1 Synchronization of external interrupts

The pin value of the EXTINTn pins is normally synchronized to the CPU clock, so spikes shorter than a CPU clock cycle are not guaranteed to produce an interrupt. In Stop mode, spikes shorter than a 32 KHz clock cycle are not guaranteed to produce an interrupt.

In Static mode, only unsynchronized interrupts remain active, and any short spike on this interrupt will wake up the device.

17.5.1.2 Wakeup

The External interrupts can be used to wake up the part from sleep modes. The wakeup can be interpreted in two ways. If the corresponding bit in IMR is set, then the execution starts at the interrupt handler for this interrupt. If the bit in IMR is not set, then the execution starts from the next instruction after the sleep instruction.

17.5.2 Non-Maskable Interrupt

The NMI supports the same features as the external interrupts, and is accessed through the same registers. The description in Section 17.5.1 should be followed, accessing the NMI bit instead of the INTn bits.

The NMI is non-maskable within the CPU in the sense that it can interrupt any other execution mode. Still, as for the other external interrupts, the actual NMI input line can be enabled and disabled by accessing the registers in the External Interrupt Module. These interrupts are not enabled by default, allowing the proper interrupt vectors to be set up by the CPU before the interrupts are enabled.

17.5.3 Keypad scan support

The External Interrupt Module also includes support for keypad scanning. The keypad scan feature is compatible with keypads organized as rows and columns, where a row is shorted against a column when a key is pressed.

The rows should be connected to the external interrupt pins with pullups enabled in the GPIO module. These external interrupts should be enabled as low level or falling edge interrupts. The columns should be connected to the available scan pins. The GPIO must be configured to let the required scan pins be controlled by the EIC module. Unused external interrupt or scan pins can be left controlled by the GPIO or other peripherals.

The Keypad Scan function is enabled by writing SCAN:EN to 1, which starts the keypad scan counter. The SCAN outputs are tristated, except SCAN[0], which is driven to zero. After $2^{(SCAN:PRESC+1)}$ RC clock cycles this pattern is left shifted, so that SCAN[1] is driven to zero while the other outputs are tristated. This sequence repeats infinitely, wrapping from the most significant SCAN pin to SCAN[0].

When a key is pressed, the pulled-up row is driven to zero by the column, and an external interrupt triggers. The scanning stops, and the software can then identify the key pressed by the interrupt status register and the SCAN:PINS value.

The scanning stops whenever there is an active interrupt request from the EIC to the CPU. When the CPU clears the interrupt flags, scanning resumes.

17.6 User Interface

Offset	Register	Register Name	Access	Reset
0x00	EIC Interrupt Enable	IER	Write-only	0x0
0x04	EIC Interrupt Disable	IDR	Write-only	0x0
0x08	EIC Interrupt Mask	IMR	Read-only	0x0
0x0C	EIC Interrupt Status	ISR	Read-only	0x0
0x10	EIC Interrupt Clear	ICR	Write-only	0x0
0x14	External Interrupt Mode	MODE	Read/Write	0x0
0x18	External Interrupt Edge	EDGE	Read/Write	0x0
0x1C	External Interrupt Level	LEVEL	Read/Write	0x0
0x20	External Interrupt Filter	FILTER	Read/Write	0x0
0x24	External Interrupt Test	TEST	Read/Write	0x0
0x28	External Interrupt Asynchronous	ASYNC	Read/Write	0x0
0x2C	External Interrupt Scan	SCAN	Read/Write	0x0
0x30	External Interrupt Enable	EN	Write-only	0x0
0x34	External Interrupt Disable	DIS	Write-only	0x0
0x38	External Interrupt Control	CTRL	Read/Write	0x0

17.6.1 EIC Interrupt Enable/Disable/Mask/Status/Clear

Name: IER/IDR/IMR/ISR/ICR

Access Type: IER/IDR/ICR: Write-only

IMR/ISR: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	NMI
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

The effect of writing or reading the bits listed above depends on which register is being accessed:

- **IER (Write-only)**
  - 0: No effect
  - 1: Enable Interrupt
- **IDR (Write-only)**
  - 0: No effect
  - 1: Disable Interrupt
- **IMR (Read-only)**
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
- **ISR (Read-only)**
  - 0: An interrupt event has occurred
  - 1: An interrupt even has not occurred
- **ICR (Write-only)**
  - 0: No effect
  - 1: Clear interrupt event

17.6.2 External Interrupt Mode/Edge/Level/Filter/Async

Name: MODE/EDGE/LEVEL/FILTER/ASYNC

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	NMI
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

The bit interpretation is register specific:

- **MODE**
  - 0: Interrupt is edge triggered
  - 1: Interrupt is level triggered
- **EDGE**
  - 0: Interrupt triggers on falling edge
  - 1: Interrupt triggers on rising edge
- **LEVEL**
  - 0: Interrupt triggers on low level
  - 1: Interrupt triggers on high level
- **FILTER**
  - 0: Interrupt is not filtered
  - 1: Interrupt is filtered
- **ASYNC**
  - 0: Interrupt is synchronized to the clock
  - 1: Interrupt is asynchronous

17.6.3 External Interrupt Test

Name: TEST

Access Type: Read/Write

31	30	29	28	27	26	25	24
TEST_EN	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	NMI
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

- **NMI**
  - 0: No effect
  - 1: NMI is triggered if TEST_EN is 1
- **INTn**
  - 0: No effect
  - 1: External Interrupt n is triggered if TEST_EN is 1
- **TEST_EN**
  - 0: External interrupt test is disabled
  - 1: External interrupt test is enabled

17.6.4 External Interrupt Scan

Name: SCAN

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	PIN[2:0]		

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	PRESC[4:0]				

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	EN

- **EN**

  0: Keypad scanning is disabled

  1: Keypad scanning is enabled

- **PRESC**

  Prescale select for the keypad scan rate:

  Scan rate = $2^{(SCAN:PRESC+1)}$ T_{RC}

  The RC clock period can be found in the Electrical Characteristics section.

- **PIN**

  The index of the currently active scan pin. Writing to this bitfield has no effect.

17.6.5 External Interrupt Enable/Disable/Control

Name: EN/DIS/CTRL

Access Type: EN/DIS: Write-only

CTRL: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	NMI
7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0

The bit interpretation is register specific:

- **EN**
  - 0: No effect
  - 1: Interrupt is enabled
- **DIS**
  - 0: No effect
  - 1: Interrupt is disabled
- **CTRL**
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled

18. Flash Controller (HFLASHC)

Rev: 1.0.0.0

18.1 Features

- **Controls flash block with dual read ports allowing staggered reads.**
- **Supports 0 and 1 wait state bus access.**
- **Allows interleaved burst reads for systems with one wait state, outputting one 32-bit word per clock cycle.**
- **32-bit HSB interface for reads from flash array and writes to page buffer.**
- **32-bit PB interface for issuing commands to and configuration of the controller.**
- **16 lock bits, each protecting a region consisting of (total number of pages in the flash block / 16) pages.**
- **Regions can be individually protected or unprotected.**
- **Additional protection of the Boot Loader pages.**
- **Supports reads and writes of general-purpose NVM bits.**
- **Supports reads and writes of additional NVM pages.**
- **Supports device protection through a security bit.**
- **Dedicated command for chip-erase, first erasing all on-chip volatile memories before erasing flash and clearing security bit.**
- **Interface to Power Manager for power-down of flash-blocks in sleep mode.**
- **Interface to Power Manager for automatically selecting the correct wait state setting as a function of clock frequency.**

18.2 Description

The flash controller (HFLASHC) interfaces a flash block with the 32-bit internal HSB bus. Performance for uncached systems with high clock-frequency and one wait state is increased by placing words with sequential addresses in alternating flash subblocks. Having one read interface per subblock allows them to be read in parallel. While data from one flash subblock is being output on the bus, the sequential address is being read from the other flash subblock and will be ready in the next clock cycle.

The controller also manages the programming, erasing, locking and unlocking sequences with dedicated commands.

18.3 Functional description

18.3.1 Bus interfaces

The HFLASHC has two bus interfaces, one High-Speed Bus (HSB) interface for reads from the flash array and writes to the page buffer, and one Peripheral Bus (PB) interface for writing commands and control to and reading status from the controller.

18.3.2 Memory organization

To maximize performance for high clock-frequency systems, HFLASHC interfaces to a flash block with two read ports. The flash block has the following parameters, given by the design of the flash block:

- *p* pages
- *w* words in each page and in the page buffer
- *pw* words in total

- *f* general-purpose fuse bits
- 1 security fuse bit
- 1 User Page

18.3.3 User page

The User page is an additional page, outside the regular flash array, that can be used to store various data, like calibration data and serial numbers. This page is not erased by regular chip erase. The User page can only be written and erased by proprietary commands. Read accesses to the User page is performed just as any other read access to the flash. The address map of the User page is given in Figure 18-1.

18.3.4 Read operations

The HFLASHC provides two different read modes:

- 0 wait state (0ws) for clock frequencies < (access time of the flash plus the bus delay)
- 1 wait state (1ws) for clock frequencies < (access time of the flash plus the bus delay)/2

Higher clock frequencies that would require more wait states are not supported by the flash controller.

The programmer can select the wait states required by writing to the FWS field in the flash control register (FCR). The actual wait states used by the flash controller is a function of both the FWS field, and also the DIVIDED_CLOCK input line from the Power Manager, in the following way:

- If the Power Manager is set up so that the input clock to the flash controller is a divided clock, i.e. the clock is divided by a factor 2 or more from the base clock source, the setting of FWS is disregarded and the flash controller will use 0ws mode.
- If the input clock to the flash controller is not divided, i.e. the clock is identical to the output from the base clock source, the flash controller will use the number of wait states dictated by the FWS field.

It is the responsibility of the programmer to select a number of wait states compatible with the clock frequency and timing characteristics of the flash block.

In 0ws mode, only one of the two flash read ports is accessed. The other flash read port is idle. In 1ws mode, both flash read ports are active. One read port reading the addressed word, and the other reading the next sequential word.

If the clock frequency allows, the user should use 0ws mode, because this gives the lowest power consumption for low-frequency systems as only one flash read port is read. Using 1ws mode has a power/performance ratio approaching 0ws mode as the clock frequency approaches twice the max frequency of 0ws mode. Using two flash read ports use twice the power, but also give twice the performance.

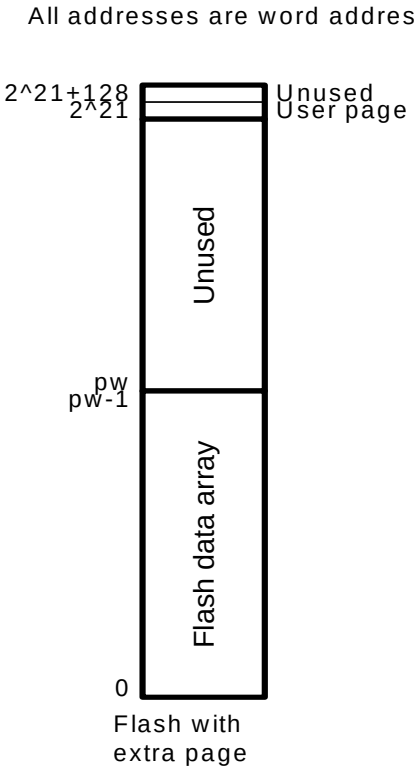
The flash controller supports flash blocks with up to 2^21 word addresses, as displayed in Figure 18-1. Reading the memory space between address *pw* and 2^21-1 returns an undefined result. The User page is permanently mapped to addresses starting at word address 2^21. This page is alternately mapped into the entire memory range between 2^21 and 2^22-1, so

that the first word in the user page is present on word address 2^21, 2^21 + 256, 2^21 + 512 and so on.

Table 18-1. User row addresses

Memory type	Start address, byte sized	Size
Main array	0	*pw* words = 4*pw* bytes
User	2^23 = 8388608	128 words = 512 bytes

Figure 18-1. Memory map for the Flash memories



18.3.5 Quick Page Read

A dedicated command, Quick Page Read (QPR), is provided to read all words in an addressed page. All bits in all words in this page are AND'ed together, returning a 1-bit result. This result is placed in the Quick Page Read Result (QPRR) bit in Flash Status Register (FSR). The QPR command is useful to check that a page is in an erased state. The QPR instruction is much faster than performing the erased-page check using a regular software subroutine.

18.3.6 Write page buffer operations

The internal memory area reserved for the embedded flash can also be written through a write-only page buffer. The page buffer is addressed only by the address bits required to address *w* words (since the page buffer is word addressable) and thus wrap around within the internal memory area address space and appear to be repeated within it.

When writing to the page buffer, the PAGEN field in the FCMD register is updated with the page number corresponding to page address of the latest word written into the page buffer.

The page buffer is also used for writes to the User page.

Write operations can be prevented by programming the Memory Protection Unit of the CPU. Writing 8-bit and 16-bit data to the page buffer is not allowed and may lead to unpredictable data corruption.

Page buffer write operations are performed with 4 wait states.

Writing to the page buffer can only change page buffer bits from one to zero, ie writing 0xaaaaaaaa to a page buffer location that has the value 0x00000000, will not change the page buffer value. The only way to change a bit from zero to one, is to reset the entire page buffer with the Clear Page Buffer command.

The page buffer is not automatically reset after a page write. The programmer should do this manually by issuing the Clear Page Buffer flash command. This can be done after a page write, or before the page buffer is loaded with data to be stored to the flash page.

Example: Writing a word into word address 130 of a flash with 128 words in the page buffer. PAGEN will be updated with the value 1, and the word will be written into word 2 in the page buffer.

18.4 Flash commands

The HFLASHC offers a command set to manage programming of the flash memory, locking and unlocking of regions, and full flash erasing. See chapter 18.7.3 for a complete list of commands.

To run a command, the field FCMD of the Flash Command Register (FCMD) has to be written with the command number. As soon as the FCMD register is written, the FRDY flag is automatically cleared. Once the current command is complete, the FRDY flag is automatically set. If an interrupt has been enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is activated. All flash commands except for Quick Page Read (QPR) will generate an interrupt request upon completion if FRDY is set.

All the commands are protected by the same keyword, which has to be written in the eight highest bits of the FCMD register. Writing FCMD with data that does not contain the correct key and/or with an invalid command has no effect on the flash memory; however, the PROGE flag is set in the Flash Status Register (FSR). This flag is automatically cleared by a read access to the FSR register.

Writing a command to FCMD while another command is being executed has no effect on the flash memory; however, the PROGE flag is set in the Flash Status Register (FSR). This flag is automatically cleared by a read access to the FSR register.

If the current command writes or erases a page in a locked region, or a page protected by the BOOTPROT fuses, the command has no effect on the flash memory; however, the LOCKE flag is set in the FSR register. This flag is automatically cleared by a read access to the FSR register.

18.4.1 Write/erase page operation

Flash technology requires that an erase must be done before programming. The entire flash can be erased by an Erase All command. Alternatively, pages can be individually erased by the Erase Page command.

The User page can be written and erased using the mechanisms described in this chapter.

After programming, the page can be locked to prevent miscellaneous write or erase sequences. Locking is performed on a per-region basis, so locking a region locks all pages inside the region. Additional protection is provided for the lowermost address space of the flash. This address space is allocated for the Boot Loader, and is protected both by the lock bit(s) corresponding to this address space, and the BOOTPROT[2:0] fuses.

Data to be written are stored in an internal buffer called page buffer. The page buffer contains *w* words. The page buffer wraps around within the internal memory area address space and appears to be repeated by the number of pages in it. Writing of 8-bit and 16-bit data to the page buffer is not allowed and may lead to unpredictable data corruption.

Data must be written to the page buffer before the programming command is written to the Flash Command Register FCMD. The sequence is as follows:

- Reset the page buffer with the Clear Page Buffer command.
- Fill the page buffer with the desired contents, using only 32-bit access.
- Programming starts as soon as the programming key and the programming command are written to the Flash Command Register. The PAGEN field in the Flash Command Register (FCMD) must contain the address of the page to write. PAGEN is automatically updated when writing to the page buffer, but can also be written to directly. The FRDY bit in the Flash Status Register (FSR) is automatically cleared when the page write operation starts.
- When programming is completed, the bit FRDY in the Flash Status Register (FSR) is set. If an interrupt was enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is set.

Two errors can be detected in the FSR register after a programming sequence:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: The page to be programmed belongs to a locked region. A command must be executed to unlock the corresponding region before programming can start. A Lock Error is also issued if a page erase/write is attempted to one of the pages in the Unused address space shown in Figure 18-1.

18.4.2 Erase All operation

The entire memory is erased if the Erase All command (EA) is written to the Flash Command Register (FCMD). Erase All erases all bits in the flash array. The User page is not erased. All flash memory locations, the lock and general-purpose fuse bits, and the security bit are erased (reset to 0xFF) after an Erase All.

The EA command also ensures that all volatile memories, such as register file and RAMs, are erased before the security bit is erased.

Erase All operation is allowed only if no regions are locked, and the BOOTPROT fuses are set to 0. Thus, if at least one region is locked, the bit LOCKE in FSR is set and the command is cancelled. If the bit LOCKE has been written to 1 in FCR, the interrupt line rises.

When the command is complete, the bit FRDY bit in the Flash Status Register (FSR) is set. If an interrupt has been enabled by setting the bit FRDY in FCR, the interrupt line of the flash controller is set. Two errors can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: At least one lock region to be erased is protected, or BOOTPROT is different from 0. The erase command has been refused and no page has been erased. A Clear Lock Bit command must be executed previously to unlock the corresponding lock regions.

18.4.3 Region lock bits

The flash block has *p* pages, and these pages are grouped into 16 lock regions, each region containing *p*/16 pages. Each region has a dedicated lock bit preventing writing and erasing pages in the region. After production, the device may have some regions locked. These locked regions are reserved for a boot or default application. Locked regions can be unlocked to be erased and then programmed with another application or other data.

To lock or unlock a region, the commands Lock Region Containing Page (LP) and Unlock Region Containing Page (UP) are provided. Writing one of these commands, together with the number of the page whose region should be locked/unlocked, performs the desired operation.

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that lock bits can also be set/cleared using the commands for writing/erasing general-purpose fuse bits, see chapter 18.5. The general-purpose bit being in an erased (1) state means that the region is unlocked.

The lowermost pages in the Flash can additionally be protected by the BOOTPROT fuses, see Section 18.5.

18.5 General-purpose fuse bits

Each flash block has a number of general-purpose fuse bits that the application programmer can use freely. The fuse bits can be written and erased using dedicated commands, and read

through a dedicated Peripheral Bus address. Some of the general-purpose fuse bits are reserved for special purposes, and should not be used for other functions.:

Table 18-2. General-purpose fuses with special functions

General-Purpose fuse number	Name	Usage
15:0	LOCK	Region lock bits.
16	EPFL	External Privileged Fetch Lock. Used to prevent the CPU from fetching instructions from external memories when in privileged mode. This bit can only be changed when the security bit is cleared. The address range corresponding to external memories is device-specific, and not known to the flash controller. This fuse bit is simply routed out of the CPU or bus system, the flash controller does not treat this fuse in any special way, except that it can not be altered when the security bit is set. If the security bit is set, only an external JTAG Chip Erase can clear EPFL. No internal commands can alter EPFL if the security bit is set.
19:17	BOOTPROT	Used to select one of four different bootloader sizes. Pages included in the bootloader area can not be erased or programmed except by a JTAG chip erase. BOOTPROT can only be changed when the security bit is cleared. If the security bit is set, only an external JTAG Chip Erase can clear BOOTPROT, and thereby allow the pages protected by BOOTPROT to be programmed. No internal commands can alter BOOTPROT or the pages protected by BOOTPROT if the security bit is set.

The BOOTPROT fuses protects the following address space for the Boot Loader:

Table 18-3. Boot Loader area specified by BOOTPROT

BOOTPROT	Pages protected by BOOTPROT	Size of protected memory
7	None	0
6	0-1	1kByte
5	0-3	2kByte
4	0-7	4kByte
3	0-15	8kByte
2	0-31	16kByte
1	0-63	32kByte
0	0-127	64kByte

To erase or write a general-purpose fuse bit, the commands Write General-Purpose Fuse Bit (WGPB) and Erase General-Purpose Fuse Bit (EGPB) are provided. Writing one of these commands, together with the number of the fuse to write/erase, performs the desired operation.

An entire General-Purpose Fuse byte can be written at a time by using the Program GP Fuse Byte (PGPFB) instruction. A PGPFB to GP fuse byte 2 is not allowed if the flash is locked by the security bit. The PFB command is issued with a parameter in the PAGEN field:

- PAGEN[1:0] - byte to write
- PAGEN[9:2] - Fuse value to write

All General-Purpose fuses can be erased by the Erase All General-Purpose fuses (EAGP) command. An EAGP command is not allowed if the flash is locked by the security bit.

Two errors can be detected in the FSR register after issuing these commands:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.
- Lock Error: A write or erase of any of the special-function fuse bits in Table 18-3 was attempted while the flash is locked by the security bit.

The lock bits are implemented using the lowest 16 general-purpose fuse bits. This means that the 16 lowest general-purpose fuse bits can also be written/erased using the commands for locking/unlocking regions, see Section 18.4.3.

18.6 Security bit

The security bit allows the entire chip to be locked from external JTAG or other debug access for code security. The security bit can be written by a dedicated command, Set Security Bit (SSB). Once set, the only way to clear the security bit is through the JTAG Chip Erase command.

Once the Security bit is set, the following Flash controller commands will be unavailable and return a lock error if attempted:

- Write General-Purpose Fuse Bit (WGPB) to BOOTPROT or EPFL fuses
- Erase General-Purpose Fuse Bit (EGPB) to BOOTPROT or EPFL fuses
- Program General-Purpose Fuse Byte (PGPFB)
- Erase All General-Purpose Fuses (EAGPF)

One error can be detected in the FSR register after issuing the command:

- Programming Error: A bad keyword and/or an invalid command have been written in the FCMD register.

18.7 User interface

18.7.1 Address map

The following addresses are used by the HFLASHC. All offsets are relative to the base address allocated to the flash controller.

Table 18-4. Flash controller register mapping

Offset	Register	Name	Access	Reset state
0x0	Flash Control Register	FCR	R/W	0

Table 18-4. Flash controller register mapping

Offset	Register	Name	Access	Reset state
0x4	Flash Command Register	FCMD	R/W	0
0x8	Flash Status Register	FSR	R/W	0 (*)
0xc	Flash General Purpose Fuse Register	FGPFR	R	NA (*)

(*) The value of the Lock bits is dependent of their programmed state. All other bits in FSR are 0. All bits in FGPFR and FCFR are dependent on the programmed state of the fuses they map to. Any bits in these registers not mapped to a fuse read 0.

18.7.2 Flash Control Register (FCR)

Offset: 0x0

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	FWS	-	-	PROGE	LOCKE	-	FRDY

FRDY: Flash Ready Interrupt Enable

0: Flash Ready does not generate an interrupt.

1: Flash Ready generates an interrupt.

LOCKE: Lock Error Interrupt Enable

0: Lock Error does not generate an interrupt.

1: Lock Error generates an interrupt.

PROGE: Programming Error Interrupt Enable

0: Programming Error does not generate an interrupt.

1: Programming Error generates an interrupt.

FWS: Flash Wait State

0: The flash is read with 0 wait states.

1: The flash is read with 1 wait state.

18.7.3 Flash Command Register (FCMD)

Offset: 0x4

The FCMD can not be written if the flash is in the process of performing a flash command. Doing so will cause the FCR write to be ignored, and the PROGE bit to be set.

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
PAGEN [15:8]							

15	14	13	12	11	10	9	8
PAGEN [7:0]							

7	6	5	4	3	2	1	0
-	-	-	FCMD				

FCMD: Flash command

This field defines the flash command. Issuing any unused command will cause the Programming Error flag to be set, and the corresponding interrupt to be requested if the PROGE bit in FCR is set.

Table 18-5. Set of commands

Command	Value	Mnemonic
No operation	0	NOP
Write Page	1	WP
Erase Page	2	EP
Clear Page Buffer	3	CPB
Lock region containing given Page	4	LP
Unlock region containing given Page	5	UP
Erase All	6	EA
Write General-Purpose Fuse Bit	7	WGPB
Erase General-Purpose Fuse Bit	8	EGPB
Set Security Bit	9	SSB
Program GP Fuse Byte	10	PGPFB
Erase All GPFuses	11	EAGPF
Quick Page Read	12	QPR
Write User Page	13	WUP
Erase User Page	14	EUP

PAGEN: Page number

The PAGEN field is used to address a page or fuse bit for certain operations. In order to simplify programming, the PAGEN field is automatically updated every time the page buffer is written to. For every page buffer write, the PAGEN field is updated with the page number of the address being written to.

Table 18-6. Semantic of PAGEN field in different commands

Command	PAGEN description
No operation	Not used
Write Page	The number of the page to write
Clear Page Buffer	Not used
Lock region containing given Page	Page number whose region should be locked
Unlock region containing given Page	Page number whose region should be unlocked
Erase All	Not used
Write General-Purpose Fuse Bit	GPFUSE #
Erase General-Purpose Fuse Bit	GPFUSE #
Set Security Bit	Not used
Program GP Fuse Byte	WriteData[7:0], ByteAddress[1:0]
Erase All GP Fuses	Not used
Quick Page Read	Page number
Write User Page	Not used
Erase User Page	Not used

KEY: Write protection key

This field should be written with the value 0xA5 to enable the command defined by the bits of the register. If the field is written with a different value, the write is not performed and no action is started.

This field always reads as 0.

18.7.4 Flash Status Register (FSR)

Offset: 0x08

31	30	29	28	27	26	25	24
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8

23	22	21	20	19	18	17	16
LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0

15	14	13	12	11	10	9	8
FSZ			-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	QPRR	SECURITY	PROGE	LOCKE	-	FRDY

FRDY: Flash Ready Status

0: The flash controller is busy and the application must wait before running a new command.

1: The flash controller is ready to run a new command.

LOCKE: Lock Error Status

Automatically cleared when FSR is read.

0: No programming of at least one locked lock region has happened since the last read of FSR.

1: Programming of at least one locked lock region has happened since the last read of FSR.

PROGE: Programming Error Status

Automatically cleared when FSR is read.

0: No invalid commands and no bad keywords were written in the Flash Command Register FCMD.

1: An invalid command and/or a bad keyword was/were written in the Flash Command Register FCMD.

SECURITY: Security Bit Status

0: The security bit is inactive.

1: The security bit is active.

QPRR: Quick Page Read Result

0: The result is zero, i.e. the page is not erased.

1: The result is one, i.e. the page is erased.

FSZ: Flash Size

The size of the flash. Not all device families will provide all flash sizes indicated in the table.

FSZ	Flash Size
0	32 KByte
1	64 kByte
2	128 kByte
3	256 kByte
4	384 kByte
5	512 kByte
6	768 kByte
7	1024 kByte

LOCKx: Lock Region x Lock Status

0: The corresponding lock region is not locked.

1: The corresponding lock region is locked.

18.7.5 Flash General Purpose Fuse Register (FGPFR)

Offset: 0x0c

31	30	29	28	27	26	25	24
GPF31	GPF30	GPF29	GPF28	GPF27	GPF26	GPF25	GPF24

23	22	21	20	19	18	17	16
GPF23	GPF22	GPF21	GPF20	GPF19	GPF18	GPF17	GPF16

15	14	13	12	11	10	9	8
GPF15	GPF14	GPF13	GPF12	GPF11	GPF10	GPF09	GPF08

7	6	5	4	3	2	1	0
GPF07	GPF06	GPF05	GPF04	GPF03	GPF02	GPF01	GPF00

GPFxx: General Purpose Fuse xx

0: The fuse has a written/programmed state.

1: The fuse has an erased state.

19. HSB Bus Matrix (HMATRIX)

Rev: 2.2.0.0

19.1 Features

- **User Interface on peripheral bus**
- **Configurable Number of Masters (Up to sixteen)**
- **Configurable Number of Slaves (Up to sixteen)**
- **One Decoder for Each Master**
- **Three Different Memory Mappings for Each Master (Internal and External boot, Remap)**
- **One Remap Function for Each Master**
- **Programmable Arbitration for Each Slave**
  - **Round-Robin**
  - **Fixed Priority**
- **Programmable Default Master for Each Slave**
  - **No Default Master**
  - **Last Accessed Default Master**
  - **Fixed Default Master**
- **One Cycle Latency for the First Access of a Burst**
- **Zero Cycle Latency for Default Master**
- **One Special Function Register for Each Slave (Not dedicated)**

19.2 Description

The Bus Matrix implements a multi-layer bus structure, that enables parallel access paths between multiple High Speed Bus (HSB) masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 16 HSB Masters to up to 16 HSB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency). The Bus Matrix provides 16 Special Function Registers (SFR) that allow the Bus Matrix to support application specific features.

19.3 Memory Mapping

The Bus Matrix provides one decoder for every HSB Master Interface. The decoder offers each HSB Master several memory mappings. In fact, depending on the product, each memory area may be assigned to several slaves. Booting at the same address while using different HSB slaves (i.e. external RAM, internal ROM or internal Flash, etc.) becomes possible.

The Bus Matrix user interface provides Master Remap Control Register (MRCR) that performs remap action for every master independently.

19.4 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first access of a burst or single transfer. This bus granting mechanism sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master and fixed default master.

19.5 No Default Master

At the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

19.6 Last Access Master

At the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

19.7 Fixed Default Master

At the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike last access master, the fixed master does not change unless the user modifies it by a software action (field FIXED_DEFMSTR of the related SCFG).

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

19.8 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflict cases occur, i.e. when two or more masters try to access the same slave at the same time. One arbiter per HSB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user with the possibility of choosing between 2 arbitration types for each slave:

1. Round-Robin Arbitration (default)
2. Fixed Priority Arbitration

This choice is made via the field ARBT of the Slave Configuration Registers (SCFG).

Each algorithm may be complemented by selecting a default master configuration for each slave.

When a re-arbitration must be done, specific conditions apply. See Section 19.8.1 ”Arbitration Rules” on page 124.

19.8.1 Arbitration Rules

Each arbiter has the ability to arbitrate between two or more different master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See Section “19.8.1.1” on page 125.

4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See Section “19.8.1.2” on page 125.

19.8.1.1 Undefined Length Burst Arbitration

In order to avoid long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected from among the following five possibilities:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. One beat bursts: Predicted end of burst is generated at each single transfer inside the INCP transfer.
3. Four beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
4. Eight beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
5. Sixteen beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the field ULBT of the Master Configuration Registers (MCFG).

19.8.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, half word or word transfer.

19.8.2 Round-Robin Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is first serviced, then the others are serviced in a round-robin manner.

There are three round-robin algorithms implemented:

- Round-Robin arbitration without default master
- Round-Robin arbitration with last default master
- Round-Robin arbitration with fixed default master

19.8.2.1 Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

19.8.2.2 *Round-Robin Arbitration with Last Default Master*

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. In fact, at the end of the current transfer, if no other master request is pending, the slave remains connected to the last master that performed the access. Other non privileged masters still get one latency cycle if they want to access the same slave. This technique can be used for masters that mainly perform single accesses.

19.8.2.3 *Round-Robin Arbitration with Fixed Default Master*

This is another biased round-robin algorithm. It allows the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Every request attempted by this fixed default master will not cause any latency whereas other non privileged masters will still get one latency cycle. This technique can be used for masters that mainly perform single accesses.

19.8.3 Fixed Priority Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If two or more master requests are active at the same time, the master with the highest priority number is serviced first. If two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority Registers for Slaves (PRAS and PRBS).

19.9 HSB Generic Bus Matrix User Interface

Table 19-1. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register 0	MCFG0	Read/Write	0x00000002
0x0004	Master Configuration Register 1	MCFG1	Read/Write	0x00000002
0x0008	Master Configuration Register 2	MCFG2	Read/Write	0x00000002
0x000C	Master Configuration Register 3	MCFG3	Read/Write	0x00000002
0x0010	Master Configuration Register 4	MCFG4	Read/Write	0x00000002
0x0014	Master Configuration Register 5	MCFG5	Read/Write	0x00000002
0x0018	Master Configuration Register 6	MCFG6	Read/Write	0x00000002
0x001C	Master Configuration Register 7	MCFG7	Read/Write	0x00000002
0x0020	Master Configuration Register 8	MCFG8	Read/Write	0x00000002
0x0024	Master Configuration Register 9	MCFG9	Read/Write	0x00000002
0x0028	Master Configuration Register 10	MCFG10	Read/Write	0x00000002
0x002C	Master Configuration Register 11	MCFG11	Read/Write	0x00000002
0x0030	Master Configuration Register 12	MCFG12	Read/Write	0x00000002
0x0034	Master Configuration Register 13	MCFG13	Read/Write	0x00000002
0x0038	Master Configuration Register 14	MCFG14	Read/Write	0x00000002
0x003C	Master Configuration Register 15	MCFG15	Read/Write	0x00000002
0x0040	Slave Configuration Register 0	SCFG0	Read/Write	0x00000010
0x0044	Slave Configuration Register 1	SCFG1	Read/Write	0x00000010
0x0048	Slave Configuration Register 2	SCFG2	Read/Write	0x00000010
0x004C	Slave Configuration Register 3	SCFG3	Read/Write	0x00000010
0x0050	Slave Configuration Register 4	SCFG4	Read/Write	0x00000010
0x0054	Slave Configuration Register 5	SCFG5	Read/Write	0x00000010
0x0058	Slave Configuration Register 6	SCFG6	Read/Write	0x00000010
0x005C	Slave Configuration Register 7	SCFG7	Read/Write	0x00000010
0x0060	Slave Configuration Register 8	SCFG8	Read/Write	0x00000010
0x0064	Slave Configuration Register 9	SCFG9	Read/Write	0x00000010
0x0068	Slave Configuration Register 10	SCFG10	Read/Write	0x00000010
0x006C	Slave Configuration Register 11	SCFG11	Read/Write	0x00000010
0x0070	Slave Configuration Register 12	SCFG12	Read/Write	0x00000010
0x0074	Slave Configuration Register 13	SCFG13	Read/Write	0x00000010
0x0078	Slave Configuration Register 14	SCFG14	Read/Write	0x00000010
0x007C	Slave Configuration Register 15	SCFG15	Read/Write	0x00000010
0x0080	Priority Register A for Slave 0	PRAS0	Read/Write	0x00000000
0x0084	Priority Register B for Slave 0	PRBS0	Read/Write	0x00000000
0x0088	Priority Register A for Slave 1	PRAS1	Read/Write	0x00000000

Table 19-1. Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x008C	Priority Register B for Slave 1	PRBS1	Read/Write	0x00000000
0x0090	Priority Register A for Slave 2	PRAS2	Read/Write	0x00000000
0x0094	Priority Register B for Slave 2	PRBS2	Read/Write	0x00000000
0x0098	Priority Register A for Slave 3	PRAS3	Read/Write	0x00000000
0x009C	Priority Register B for Slave 3	PRBS3	Read/Write	0x00000000
0x00A0	Priority Register A for Slave 4	PRAS4	Read/Write	0x00000000
0x00A4	Priority Register B for Slave 4	PRBS4	Read/Write	0x00000000
0x00A8	Priority Register A for Slave 5	PRAS5	Read/Write	0x00000000
0x00AC	Priority Register B for Slave 5	PRBS5	Read/Write	0x00000000
0x00B0	Priority Register A for Slave 6	PRAS6	Read/Write	0x00000000
0x00B4	Priority Register B for Slave 6	PRBS6	Read/Write	0x00000000
0x00B8	Priority Register A for Slave 7	PRAS7	Read/Write	0x00000000
0x00BC	Priority Register B for Slave 7	PRBS7	Read/Write	0x00000000
0x00C0	Priority Register A for Slave 8	PRAS8	Read/Write	0x00000000
0x00C4	Priority Register B for Slave 8	PRBS8	Read/Write	0x00000000
0x00C8	Priority Register A for Slave 9	PRAS9	Read/Write	0x00000000
0x00CC	Priority Register B for Slave 9	PRBS9	Read/Write	0x00000000
0x00D0	Priority Register A for Slave 10	PRAS10	Read/Write	0x00000000
0x00D4	Priority Register B for Slave 10	PRBS10	Read/Write	0x00000000
0x00D8	Priority Register A for Slave 11	PRAS11	Read/Write	0x00000000
0x00DC	Priority Register B for Slave 11	PRBS11	Read/Write	0x00000000
0x00E0	Priority Register A for Slave 12	PRAS12	Read/Write	0x00000000
0x00E4	Priority Register B for Slave 12	PRBS12	Read/Write	0x00000000
0x00E8	Priority Register A for Slave 13	PRAS13	Read/Write	0x00000000
0x00EC	Priority Register B for Slave 13	PRBS13	Read/Write	0x00000000
0x00F0	Priority Register A for Slave 14	PRAS14	Read/Write	0x00000000
0x00F4	Priority Register B for Slave 14	PRBS14	Read/Write	0x00000000
0x00F8	Priority Register A for Slave 15	PRAS15	Read/Write	0x00000000
0x00FC	Priority Register B for Slave 15	PRBS15	Read/Write	0x00000000
0x0100	Master Remap Control Register	MRCR	Read/Write	0x00000000
0x0104 - 0x010C	Reserved	–	–	–
0x0110	Special Function Register 0	SFR0	Read/Write	–
0x0114	Special Function Register 1	SFR1	Read/Write	–
0x0118	Special Function Register 2	SFR2	Read/Write	–
0x011C	Special Function Register 3	SFR3	Read/Write	–
0x0120	Special Function Register 4	SFR4	Read/Write	–

Table 19-1. Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x0124	Special Function Register 5	SFR5	Read/Write	–
0x0128	Special Function Register 6	SFR6	Read/Write	–
0x012C	Special Function Register 7	SFR7	Read/Write	–
0x0130	Special Function Register 8	SFR8	Read/Write	–
0x0134	Special Function Register 9	SFR9	Read/Write	–
0x0138	Special Function Register 10	SFR10	Read/Write	–
0x013C	Special Function Register 11	SFR11	Read/Write	–
0x0140	Special Function Register 12	SFR12	Read/Write	–
0x0144	Special Function Register 13	SFR13	Read/Write	–
0x0148	Special Function Register 14	SFR14	Read/Write	–
0x014C	Special Function Register 15	SFR15	Read/Write	–
0x0150 - 0x01F8	Reserved	–	–	–

19.10 Bus Matrix Master Configuration Registers

Register Name: MCFG0...MCFG15

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	ULBT		

- **ULBT: Undefined Length Burst Type**

0: Infinite Length Burst

No predicted end of burst is generated and therefore INCR bursts coming from this master cannot be broken.

1: Single Access

The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst.

2: Four Beat Burst

The undefined length burst is split into a four-beat burst, allowing re-arbitration at each four-beat burst end.

3: Eight Beat Burst

The undefined length burst is split into an eight-beat burst, allowing re-arbitration at each eight-beat burst end.

4: Sixteen Beat Burst

The undefined length burst is split into a sixteen-beat burst, allowing re-arbitration at each sixteen-beat burst end.

19.11 Bus Matrix Slave Configuration Registers

Register Name: SCFG0...SCFG15

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	ARBT

23	22	21	20	19	18	17	16
–	–	FIXED_DEFMSTR				DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

- **SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst**

When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.

This limit has been placed to avoid locking a very slow slave when very long bursts are used.

This limit must not be very small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

- **DEFMSTR_TYPE: Default Master Type**

0: No Default Master

At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.

This results in a one cycle latency for the first access of a burst transfer or for a single access.

1: Last Default Master

At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it.

This results in not having one cycle latency when the last master tries to access the slave again.

2: Fixed Default Master

At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field.

This results in not having one cycle latency when the fixed master tries to access the slave again.

- **FIXED_DEFMSTR: Fixed Default Master**

This is the number of the Default Master for this slave. Only used if DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.

- **ARBT: Arbitration Type**

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

19.12 Bus Matrix Priority Registers A For Slaves

Register Name: PRAS0...PRAS15

Access Type: Read/Write

31	30	29	28	27	26	25	24
M7PR				M6PR			

23	22	21	20	19	18	17	16
M5PR				M4PR			

15	14	13	12	11	10	9	8
M3PR				M2PR			

7	6	5	4	3	2	1	0
M1PR				M0PR			

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

19.13 Bus Matrix Priority Registers B For Slaves

Register Name: PRBS0...PRBS15

Access Type: Read/Write

31	30	29	28	27	26	25	24
M15PR				M14PR			

23	22	21	20	19	18	17	16
M13PR				M12PR			

15	14	13	12	11	10	9	8
M11PR				M10PR			

7	6	5	4	3	2	1	0
M9PR				M8PR			

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

19.14 Bus Matrix Master Remap Control Register

Register Name: MRCR

Access Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RCB15	RCB14	RCB13	RCB12	RCB11	RCB10	RCB9	RCB8

7	6	5	4	3	2	1	0
RCB7	RCB6	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

- **RCB: Remap Command Bit for Master x**

0: Disable remapped address decoding for the selected Master

1: Enable remapped address decoding for the selected Master

19.15 Bus Matrix Special Function Registers

Register Name: SFR0...SFR15

Access Type: Read/Write

Reset:

31	30	29	28	27	26	25	24
SFR							

23	22	21	20	19	18	17	16
SFR							

15	14	13	12	11	10	9	8
SFR							

7	6	5	4	3	2	1	0
SFR							

- **SFR: Special Function Register Fields**

The bitfields of these registers are described in the Peripherals chapter.

20. Peripheral DMA Controller (PDCA)

rev: 1.0.0.0

20.1 Features

- **Generates Transfers to/from Peripherals such as USART, SSC and SPI**
- **Two address pointers/counters per channel allowing double buffering**

20.2 Overview

The Peripheral DMA controller (PDCA) transfers data between on-chip peripheral modules such as USART, SPI, SSC and on- and off-chip memories. Using the PDCA avoids CPU intervention for data transfers, improving the performance of the microcontroller. The PDCA can transfer data from memory to a peripheral or from a peripheral to memory.

The PDCA consists of a number of DMA channels. Each channel has:

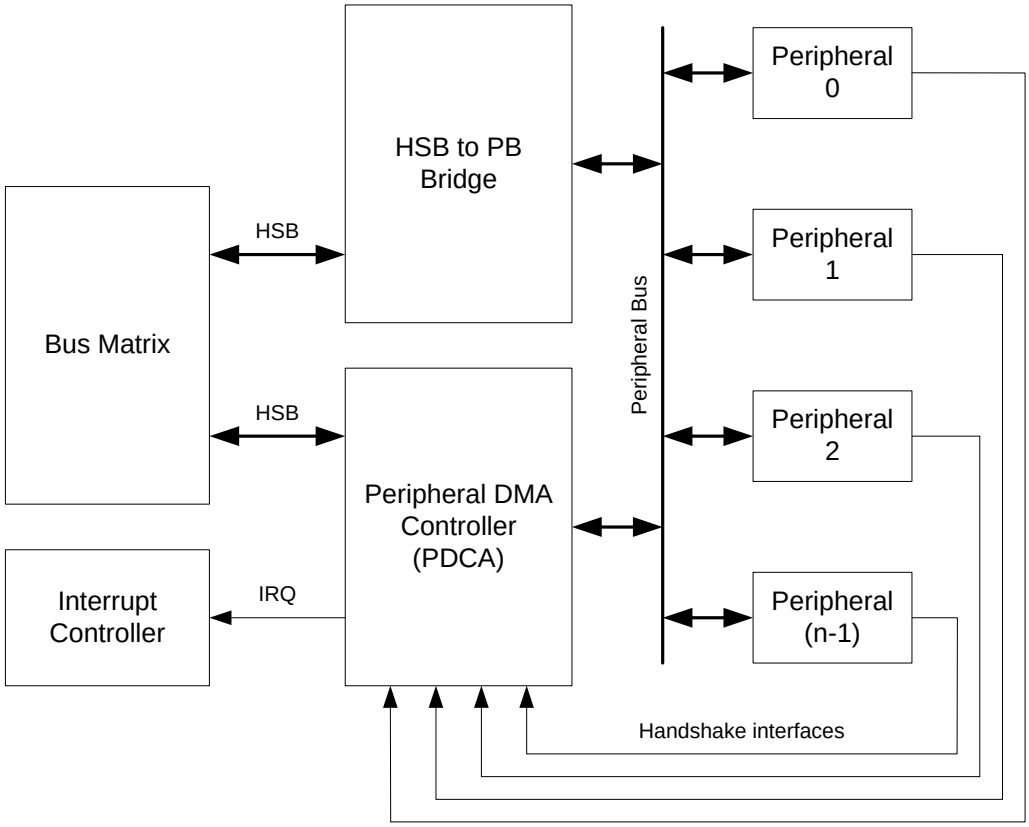
- A 32-bit memory pointer
- A 16-bit transfer counter
- A 32-bit memory pointer reload value
- A 16-bit transfer counter reload value

The PDCA communicates with the peripheral modules over a number of handshake interfaces. The peripheral signals to the PDCA when it is ready to receive or transmit data. The PDCA acknowledges the request when the transmission has started.

The number of handshake-interfaces may be higher than the number of DMA channels. If this is the case, the DMA channel must be programmed to use the desired interface.

When a transmit buffer is empty or a receive buffer is full, an interrupt request can be signalled.

20.3 Block Diagram



20.4 Functional Description

20.4.1 Configuration

Each channel in the PDCA has a set of configuration registers. Among these are the Memory Address Register (MAR), the Peripheral Select Register (PSR) and the Transfer Counter Register (TCR). The 32-bit Memory Address Register must be programmed with the start address of the memory buffer. The register will be automatically updated after each transfer to point to the next location in memory. The Peripheral Select Register must be programmed to select the desired peripheral/handshake interface. The Transfer Counter Register determines the number of data items to be transferred. The counter will be decreased by one for each data item that has been transferred.

Both the Memory Address Register and the Transfer Counter Register can be read at any time to check the progress of the transfer.

Each channel has also reload registers for the Memory Address Register and the Transfer Counter Register. When the TCR reaches zero, the values in the reload registers are loaded into MAR and TCR. In this way, the PDCA can operate on two buffers for each channel.

20.4.2 Memory Pointer

Each channel has a 32-bit Memory Pointer Register (MAR). This register holds the memory address for the next transfer to be performed. The register is automatically updated after each

transfer. The address will be increased by either 1, 2 or 4 depending on the size of the DMA transfer (Byte, Half-Word or Word). The Memory Address Register can be read at any time during transfer.

20.4.3 Transfer Counter

Each channel has a 16-bit Transfer Counter Register (TCR). This register must be programmed with the number of transferred to be performed. TCR should contain the number of data items to be transferred independently of the transfer size. The Transfer Counter Register can be read at any time during transfer to see the number of remaining transfers.

20.4.4 Reload Registers

Both the Memory Address Register and the Transfer Counter Register have a reload register, respectively Memory Address Reload Register (MARR) and Transfer Counter Reload Register (TCRR). These registers provide the possibility for the PDCA to work on two memory buffers for each channel. When one buffer has completed, MAR and TCR will be reloaded with the values in MARR and TCRR. The reload logic is always enabled and will trigger if the TCR reaches zero while TCRR holds a non-zero value.

20.4.5 Peripheral Selection

The Peripheral Select Register decides which peripheral should be connected to the PDCA channel. Configuring PSR will both select the direction of the transfer (memory to peripheral or peripheral to memory), which handshake interface to use, and the address of the peripheral holding register.

20.4.6 Transfer Size

The transfer size can be set individually for each channel to be either Byte, Half-Word or Word (8-bit, 16-bit or 32-bit respectively). Transfer size is set by programming the SIZE bit-field in the Mode Register (MR).

20.4.7 Enabling and Disabling

Each DMA channel is enabled by writing ‘1’ to the Transfer Enable bit (TEN) in the Control Register (CR) and disabled by writing ‘1’ to the Transfer Disable bit (TDIS). The current status can be read from the Status Register (SR).

20.4.8 Interrupts

Interrupts can be enabled by writing to the Interrupt Enable Register (IER) and disabled by writing to Interrupt Disable Register (IDR). The Interrupt Mask Register (IMR) can be read to see whether an interrupt is enabled or not. The current status of an interrupt source can be read through the Interrupt Status Register (ISR).

The PDCA has three interrupt sources:

- Reload Counter Zero - The Transfer Counter Reload Register is zero.
- Transfer Finished - Both the Transfer Counter Register and Transfer Counter Reload Register are zero.
- Transfer Error - An error has occurred in accessing memory.

20.4.9 Priority

If more then one PDCA channel is requesting transfer at a given time, the PDCA channels are prioritized by their channel number. Channels with lower numbers have priority over channels with higher numbers, giving channel 0 the highest priority.

20.4.10 Error Handling

If the memory address is set to point to an invalid location in memory, an error will occur when the PDCA tries to perform a transfer. When an error occurs, the Transfer Error flag (TERR) in the Interrupt Status Register will be set and the DMA channel that caused the error will be stopped. In order to restart the channel, the user must program the Memory Address Register to a valid address and then write the Error Clear bit (ECLR) in the Control Register (CR) to '1'. An interrupt can optionally be triggered on errors by writing the TERR-bit in the Interrupt Enable Register (IER) to '1'.

20.5 User Interface

20.5.1 Memory Map Overview

Table 20-1. Register Map Overview

Address Range	Contents
0x0000 - 0x003F	DMA channel 0 configuration registers
0x0040 - 0x007F	DMA channel 1 configuration registers
0x0080 - 0x00BF	DMA channel 2 configuration registers
0x00C0 - 0x00FF	DMA channel 3 configuration registers
0x0100 - 0x013F	DMA channel 4 configuration registers
-	-
-	DMA channel n-1 configuration registers

Note: The number of channels is implementation specific. See part documentation for details.

20.5.2 Channel Memory Map

Offset	Register	Register Name	Access	Reset
0x00	Memory Address Register	MAR	Read/Write	0x00000000
0x04	Peripheral Select Register	PSR	Read/Write	*
0x08	Transfer Counter Register	TCR	Read/Write	0x00000000
0x0C	Memory Address Reload Register	MARR	Read/Write	0x00000000
0x10	Transfer Counter Reload Register	TCRR	Read/Write	0x00000000
0x14	Control Register	CR	Write-only	-
0x18	Mode Register	MR	Read/Write	0x00000000
0x1C	Status Register	SR	Read-only	0x00000000
0x20	Interrupt Enable Register	IER	Write-only	-

Offset	Register	Register Name	Access	Reset
0x24	Interrupt Disable Register	IDR	Write-only	-
0x28	Interrupt Mask Register	IMR	Read-only	0x00000000
0x2C	Interrupt Status Register	ISR	Read-only	0x00000000

20.5.3 PDCA Memory Address Register

Name: MAR

Access Type: Read/Write

31	30	29	28	27	26	25	24
MADDR							

23	22	21	20	19	18	17	16
MADDR							

15	14	13	12	11	10	9	8
MADDR							

7	6	5	4	3	2	1	0
MADDR							

- **MADDR: Memory Address**

Address of memory buffer. MADDR should be programmed to point to the start of the memory buffer when configuring the PDCA. During transfer, MADDR will point to the next memory location to be read/written.

20.5.4 PDCA Peripheral Select Register

Name: PSR

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
PID							

- **PID: Peripheral Identifier**

The Peripheral Identifier selects which peripheral should be connected to the DMA channel. Programming PID will select both which handshake interface to use, the direction of the transfer and also the address of the Receive/Transfer Holding Register for the peripheral. The PID values for the different peripheral modules are implementation specific. See the part specific documentation for details.

The width of the PID bitfield is implementation specific and dependent on the number of peripheral modules in the microcontroller.

20.5.5PDCA Transfer Counter Register

Name: TCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TCV							

7	6	5	4	3	2	1	0
TCV							

- **TCV: Transfer Counter Value**

Number of data items to be transferred by PDCA. TCV must be programmed with the total number of transfers to be made. During transfer, TCV contains the number of remaining transfers to be done.

20.5.6 PDCA Memory Address Reload Register

Name: MARR

Access Type: Read/Write

31	30	29	28	27	26	25	24
MARV							

23	22	21	20	19	18	17	16
MARV							

15	14	13	12	11	10	9	8
MARV							

7	6	5	4	3	2	1	0
MARV							

- **MARV: Memory Address Reload Value**

Reload Value for the Memory Address Register (MAR). This value will be loaded into MAR when TCR reaches zero if the TCRR has a non-zero value.

20.5.7 PDCA Transfer Counter Reload Register

Name: TCRR

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
TCRV							

7	6	5	4	3	2	1	0
TCRV							

- **TCRV: Transfer Counter Reload Value**

Reload value for the Transfer Counter Register (TCR). When TCR reaches zero, it will be reloaded with TCRV if TCRV has a positive value. If TCRV is zero, no more transfers will be performed for the channel. When TCR is realoaded, the Transfer Counter Reload Register is cleared.

20.5.8 PDCA Control Register

Name: CR

Access Type: Write-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	ECLR

7	6	5	4	3	2	1	0
-	-	-	-	-	-	TDIS	TEN

- **ECLR: Error Clear**

0 = No Effect.

1 = Clear Transfer Error (TERR) flag in the Status Register (SR). Clearing the Transfer Error flag will allow the channel to transmit data. The memory address must first be set to point to a valid location.

- **TEN: Transfer Enable**

0 = No Effect.

1 = Enable transfer for DMA channel.

- **TDIS: Transfer Disable**

0 = No Effect.

1 = Disable transfer for DMA channel.

20.5.9 PDCA Mode Register

Name: MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	SIZE	

- **SIZE: Size of transfer**

SIZE		Size of Transfer
0	0	Byte
0	1	Half-Word
1	0	Word
1	1	Reserved

20.5.10 PDCA Status Register

Name: SR

Access Type: Read

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	TEN

- **TEN: Transfer Enabled**

0 = Transfer is disabled for the DMA channel

1 = Transfer is enabled for the DMA channel.

20.5.11 PDCA Interrupt Enable Register

Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**

0 = No effect.

1 = Enable Transfer Error interrupt.

- **TRC: Transfer Complete**

0 = No effect.

1 = Enable Transfer Complete interrupt.

- **RCZ: Reload Counter Zero**

0 = No effect.

1 = Enable Reload Counter Zero interrupt.

20.5.12 PDCA Interrupt Disable Register

Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**

0 = No effect.

1 = Disable Transfer Error interrupt.

- **TRC: Transfer Complete**

0 = No effect.

1 = Disable Transfer Complete interrupt.

- **RCZ: Reload Counter Zero**

0 = No effect.

1 = Disable Reload Counter Zero interrupt.

20.5.13 PDCA Interrupt Mask Register

Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**

0 = Transfer Error interrupt is disabled.

1 = Transfer Error interrupt is enabled.

- **TRC: Transfer Complete**

0 = Transfer Complete interrupt is disabled.

1 = Transfer Complete interrupt is enabled.

- **RCZ: Reload Counter Zero**

0 = Reload Counter Zero interrupt is disabled.

1 = Reload Counter Zero interrupt is enabled.

20.5.14 PDCA Interrupt Status Register

Name: ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-

23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-

7	6	5	4	3	2	1	0
-	-	-	-	-	TERR	TRC	RCZ

- **TERR: Transfer Error**

0 = No transfer errors have occurred.

1 = A transfer error has occurred.

- **TRC: Transfer Complete**

0 = The Transfer Counter Register (TCR) and/or the Transfer Counter Reload Register (TCRR) hold a non-zero value.

1 = Both the Transfer Counter Register (TCR) and the Transfer Counter Reload Register (TCRR) are zero.

- **RCZ: Reload Counter Zero**

0 = The Transfer Counter Reload Register (TCRR) holds a non-zero value.

1 = The Transfer Counter Reload Register (TCRR) is zero.

21. General-Purpose Input/Output Controller (GPIO)

Rev: 1.0.0.0

21.1 Features

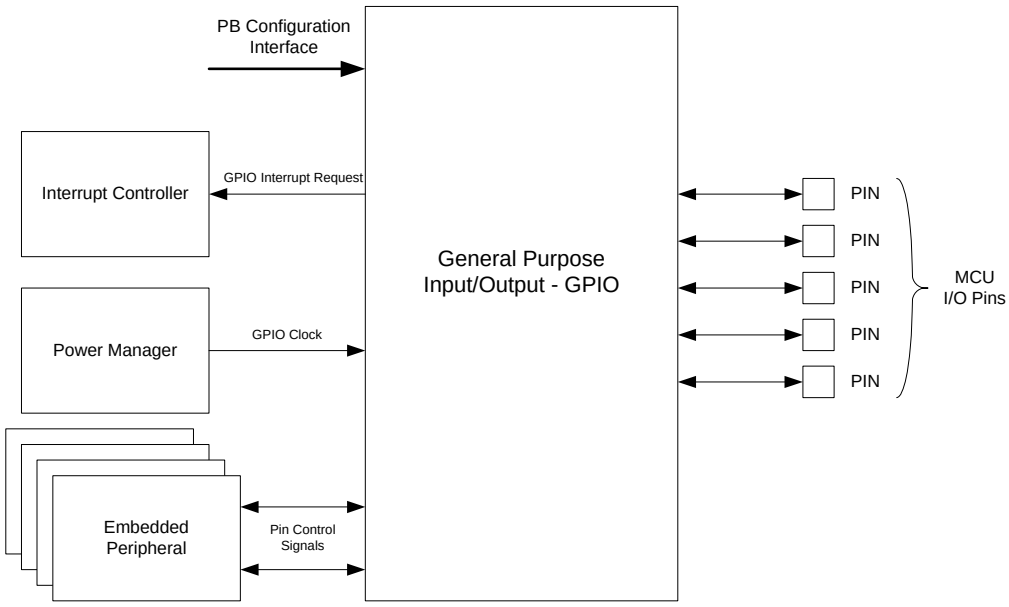
Each I/O line of the GPIO features:

- **Configurable pin-change, rising-edge or falling-edge interrupt on any I/O line.**
- **A glitch filter providing rejection of pulses shorter than one clock cycle.**
- **Open Drain mode enabling sharing of an I/O line between the MCU and external components.**
- **Input visibility and output control.**
- **Multiplexing of up to four peripheral functions per I/O line.**
- **Programmable internal pull-up resistor.**

21.2 Overview

The General Purpose Input/Output manages the I/O pins of the microcontroller. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Figure 21-1. Overview of the GPIO system



21.3 Product dependencies

21.3.1 Module Configuration

Most of the features of the GPIO are configurable for each product. The programmer must refer to the Peripherals Section for these settings.

Product specific settings includes:

- **Number of I/O pins.**
- **Functions implemented on each pin.**
- **Peripheral function(s) multiplexed on each I/O pin.**
- **Reset state of registers.**

21.3.2 Interrupt Lines

The GPIO interrupt lines are connected to the interrupt controller. Using the GPIO interrupt requires the interrupt controller to be programmed first.

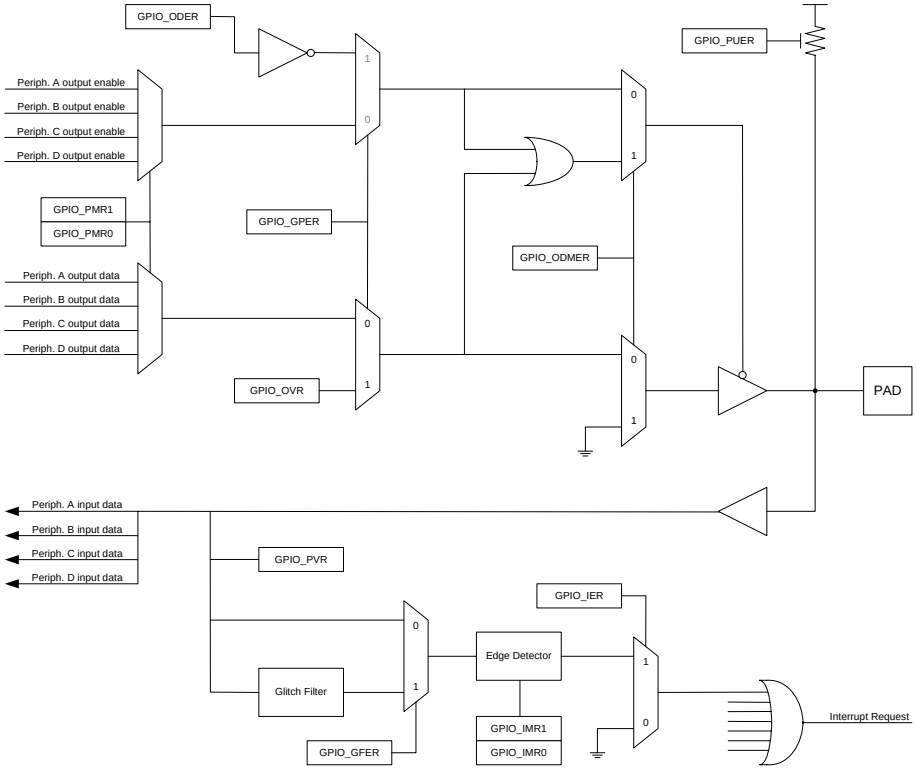
21.3.3 Power and Clock Management

The clock for the GPIO is controlled by the power manager. The programmer must ensure that the GPIO clock is enabled in the power manager before using the GPIO. The clock must be enabled in order to access the configuration registers of the GPIO and when interrupts are enabled. After configuring the GPIO, the clock can be disabled if interrupts are not enabled.

21.4 Functional Description

The GPIO controls the I/O lines of the microcontroller. The control logic associated with each pin is represented in the figure below:

Figure 21-2. Overview of the GPIO pad connections



21.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by accessing the corresponding bit in PUER (Pull-up Enable Register). Control of the pull-up resistor is possible whether an I/O line is controlled by a peripheral or the GPIO.

21.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or more peripheral functions, the selection is controlled with the register GPER. If a bit in the register is set, the corresponding pin is controlled by the GPIO. If a bit is cleared, the corresponding pin is controlled by a peripheral function.

21.4.3 Peripheral Selection

The GPIO provides multiplexing of up to four peripheral functions on a single pin. The selection is performed by accessing PMR0 (Peripheral Mux Register 0) and PMR1 (Peripheral Mux Register 1).

21.4.4 Output Control

When the I/O line is assigned to a peripheral function, i.e. the corresponding bit in GPER is at 0, the drive of the I/O line is controlled by the peripheral. The peripheral, depending on the value in PMR0 and PMR1, determines whether the pin is driven or not.

When the I/O line is controlled by the GPIO, the value of ODER (Output Driver Enable Register) determines if the pin is driven or not. When a bit in this register is at 1, the corresponding I/O line is driven by the GPIO. When the bit is at 0, the GPIO does not drive the line.

The level driven on an I/O line can be determined by writing OVR (Output Value Register).

21.4.5 Open Drain Mode

Each I/O line can be independently programmed to operate in open drain mode. This feature permits several drivers to be connected on the I/O line. The drivers should only actively drive the line low. An external pull-up resistor (or enabling the internal one) is generally required to guarantee a high level on the line when no driver is active.

The Open Drain feature is controlled by ODMER (Open Drain Mode Enable Register). The Open Drain mode can be selected whether the I/O line is controlled by the GPIO or assigned to a peripheral function.

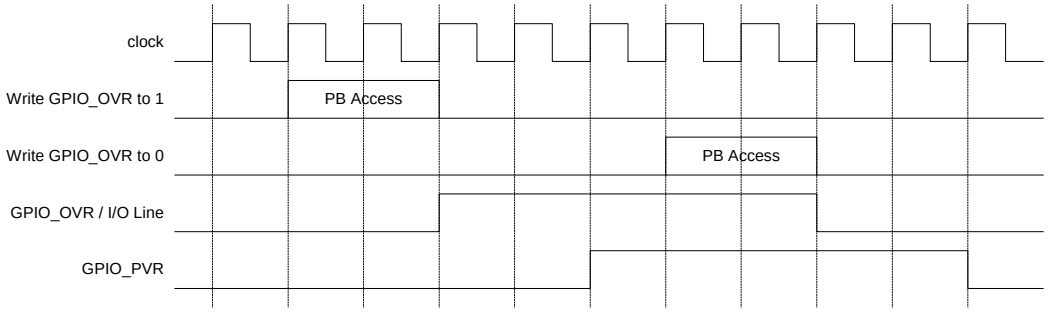
21.4.6 Inputs

The level on each I/O line can be read through PVR (Pin Value Register). This register indicates the level of the I/O lines regardless of whether the lines are driven by the GPIO or by an external component. Note that due to power saving measures, PVR register can only be read when GPER is set for the corresponding pin or if interrupt is enabled for the pin.

Output Line Timings

The figure below shows the timing of the I/O line when setting and clearing the Output Value Register by accessing OVR. The same timing applies when performing a ‘set’ or ‘clear’ access i.e. writing to OVRS or OVRC. The timing of PVR (Pin Value Register) is also shown.

Figure 21-3. Output line timings



21.4.7 Interrupts

The GPIO can be programmed to generate an interrupt when it detects an input change on an I/O line. The module can be configured to signal an interrupt whenever a pin changes value or only to trigger on rising edges or falling edges. Interrupt is enabled on a pin by setting the corresponding bit in IER (Interrupt Enable Register). The interrupt mode is set by accessing IMR0 (Interrupt Mode Register 0) and IMR1 (Interrupt Mode Register 1). Interrupt can be enabled on a pin, regardless of the configuration the I/O line, i.e. controlled by the GPIO or assigned to a peripheral function.

In every port there are four interrupt lines connected to the interrupt controller. Every eigth interrupts in the port are ored together to form an interrupt line.

When an interrupt event is detected on an I/O line, and the corresponding bit in IER is set, the GPIO interrupt request line is asserted. A number of interrupt signals are ORed-wired together to generate a single interrupt signal to the interrupt controller.

IFR (Interrupt Flag Register) can by read by software to determine which pin(s) caused the interrupt. The interrupt flag must be manually cleared by writing to IFR.

GPIO interrupts can only be triggered when the GPIO clock is enabled.

21.4.8 Input Glitch Filter

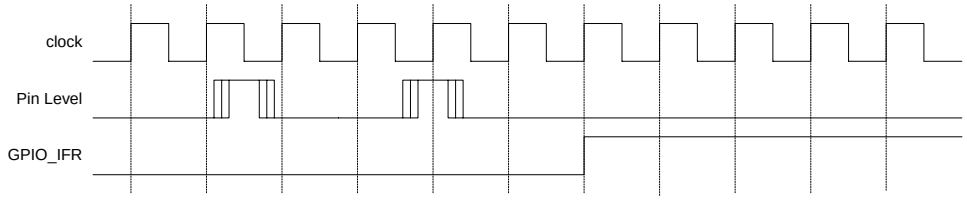
Optional input glitch filters can be enabled on each I/O line. When the glitch filter is enabled, a glitch with duration of less than 1 clock cycle is automatically rejected, while a pulse with duration of 2 clock cycles or more is accepted. For pulse durations between 1 clock cycle and 2 clock cycles, the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be guaranteed visible it must exceed 2 clock cycles, whereas for a glitch to be reliably filtered out, its duration must not exceed 1 clock cycle. The filter introduces 2 clock cycles latency.

The glitch filters are controlled by the register GFER (Glitch Filter Enable Register). When a bit is set in GFER, the glitch filter on the corresponding pin is enabled. The glitch filter affects only interrupt inputs. Inputs to peripherals or the value read through PVR are not affected by the glitch filters.

21.4.9 Interrupt Timings

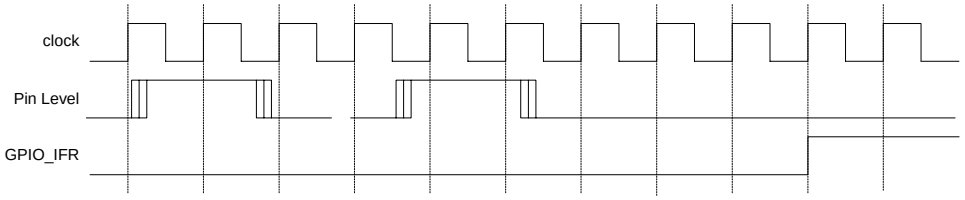
The figure below shows the timing for rising edge (or pin-change) interrupts when the glitch filter is disabled. For the pulse to be registered, it must be sampled at the rising edge of the clock. In this example, this is not the case for the first pulse. The second pulse is however sampled on a rising edge and will trigger an interrupt request.

Figure 21-4. Interrupt timing with glitch filter disabled



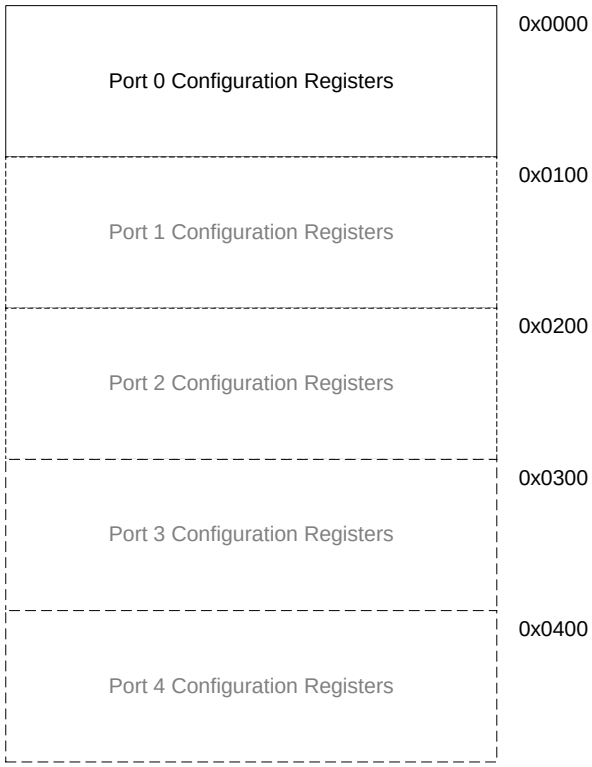
The figure below shows the timing for rising edge (or pin-change) interrupts when the glitch filter is enabled. For the pulse to be registered, it must be sampled on two subsequent rising edges. In the example, the first pulse is rejected while the second pulse is accepted and causes an interrupt request.

Figure 21-5. Interrupt timing with glitch filter enabled



21.5 General Purpose Input/Output (GPIO) User Interface

The GPIO controls all the I/O pins on the AVR32 microcontroller. The pins are managed as 32-bit ports that are configurable through an PB interface. Each port has a set of configuration registers. The overall memory map of the GPIO is shown below. The number of pins and hence the number of ports is product specific.



In the Peripheral muxing table in the Peripherals chapter each GPIO line has a unique number. Note that the PA, PB, PC and PX ports do not directly correspond to the GPIO ports. To find the corresponding port and pin the following formulas can be used:

GPIO port = floor((GPIO number) / 32), example: floor((36)/32) = 1

GPIO pin = GPIO number mod 32, example: 36 mod 32 = 4

The table below shows the configuration registers for one port. Addresses shown are relative to the port address offset. The specific address of a configuration register is found by adding the register offset and the port offset to the GPIO start address. One bit in each of the configuration registers corresponds to an I/O pin.

Table 21-1. GPIO Register Map

Offset	Register	Function	Name	Access	Reset value
0x00	GPIO Enable Register	Read/Write	GPER	Read/Write	0x00000000
0x04	GPIO Enable Register	Set	GPERS	Write-Only	
0x08	GPIO Enable Register	Clear	GPERC	Write-Only	
0x0C	GPIO Enable Register	Toggle	GPERT	Write-Only	
0x10	Peripheral Mux Register 0	Read/Write	PMR0	Read/Write	0x00000000
0x14	Peripheral Mux Register 0	Set	PMR0S	Write-Only	

Table 21-1. GPIO Register Map

Offset	Register	Function	Name	Access	Reset value
0x18	Peripheral Mux Register 0	Clear	PMR0C	Write-Only	
0x1C	Peripheral Mux Register 0	Toggle	PMR0T	Write-Only	
0x20	Peripheral Mux Register 1	Read/Write	PMR1	Read/Write	0x00000000
0x24	Peripheral Mux Register 1	Set	PMR1S	Write-Only	
0x28	Peripheral Mux Register 1	Clear	PMR1C	Write-Only	
0x2C	Peripheral Mux Register 1	Toggle	PMR1T	Write-Only	
0x30	RESERVED	-	-	-	
0x34	RESERVED	-	-	-	
0x38	RESERVED	-	-	-	
0x3C	RESERVED	-	-	-	
0x40	Output Driver Enable Register	Read/Write	ODER	Read/Write	0x00000000
0x44	Output Driver Enable Register	Set	ODERS	Write-Only	
0x48	Output Driver Enable Register	Clear	ODERC	Write-Only	
0x4C	Output Driver Enable Register	Toggle	ODERT	Write-Only	
0x50	Output Value Register	Read/Write	OVR	Read/Write	0x00000000
0x54	Output Value Register	Set	OVRS	Write-Only	
0x58	Output Value Register	Clear	OVRC	Write-Only	
0x5c	Output Value Register	Toggle	OVRT	Write-Only	
0x60	Pin Value Register	Read	PVR	Read-Only	depending on pin states
0x64	Pin Value Register	-	-	-	
0x68	Pin Value Register	-	-	-	
0x6c	Pin Value Register	-	-	-	
0x70	Pull-up Enable Register	Read/Write	PUER	Read/Write	0x00000000
0x74	Pull-up Enable Register	Set	PUERS	Write-Only	
0x78	Pull-up Enable Register	Clear	PUERC	Write-Only	
0x7C	Pull-up Enable Register	Toggle	PUERT	Write-Only	
0x80	Open Drain Mode Enable Register	Read/Write	ODMER	Read/Write	0x00000000
0x84	Open Drain Mode Enable Register	Set	ODMERS	Write-Only	
0x88	Open Drain Mode Enable Register	Clear	ODMERC	Write-Only	
0x8C	Open Drain Mode Enable Register	Toggle	ODMERT	Write-Only	
0x90	Interrupt Enable Register	Read/Write	IER	Read/Write	0x00000000
0x94	Interrupt Enable Register	Set	IERS	Write-Only	
0x98	Interrupt Enable Register	Clear	IERC	Write-Only	
0x9C	Interrupt Enable Register	Toggle	IERT	Write-Only	
0xA0	Interrupt Mode Register 0	Read/Write	IMR0	Read/Write	0x00000000

Table 21-1. GPIO Register Map

Offset	Register	Function	Name	Access	Reset value
0xA4	Interrupt Mode Register 0	Set	IMR0S	Write-Only	
0xA8	Interrupt Mode Register 0	Clear	IMR0C	Write-Only	
0xAC	Interrupt Mode Register 0	Toggle	IMR0T	Write-Only	
0xB0	Interrupt Mode Register 1	Read/Write	IMR1	Read/Write	0x00000000
0xB4	Interrupt Mode Register 1	Set	IMR1S	Write-Only	
0xB8	Interrupt Mode Register 1	Clear	IMR1C	Write-Only	
0xBC	Interrupt Mode Register 1	Toggle	IMR1T	Write-Only	
0xC0	Glitch Filter Enable Register	Read/Write	GFER	Read/Write	0x00000000
0xC4	Glitch Filter Enable Register	Set	GFERS	Write-Only	
0xC8	Glitch Filter Enable Register	Clear	GFERC	Write-Only	
0xCC	Glitch Filter Enable Register	Toggle	GFERT	Write-Only	
0xD0	Interrupt Flag Register	Read	IFR	Read-Only	0x00000000
0xD4	Interrupt Flag Register	-	-	-	
0xD8	Interrupt Flag Register	Clear	IFRC	Write-Only	
0xDC	Interrupt Flag Register	-	-	-	
0xE0-0xFF	RESERVED	-	-	-	

21.5.1 Access Types

Each configuration register can be accessed in four different ways. The first address location can be used to write the register directly. This address can also be used to read the register value. The following addresses facilitate three different types of write access to the register. Performing a “set” access, all bits written to ‘1’ will be set. Bits written to ‘0’ will be unchanged by the operation. Performing a “clear” access, all bits written to ‘1’ will be cleared. Bits written to ‘0’ will be unchanged by the operation. Finally, a toggle access will toggle the value of all bits written to ‘1’. Again all bits written to ‘0’ remain unchanged. Note that for some registers (e.g. IFR), not all access methods are permitted.

Note that for ports with less than 32 bits, the corresponding control registers will have unused bits. This is also the case for features that are not implemented for a specific pin. Writing to an unused bit will have no effect. Reading unused bits will always return 0.

21.5.2 GPIO Enable Register

Name: GPER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: GPIO Enable**

0 = A peripheral function controls the corresponding pin.

1 = The GPIO controls the corresponding pin.

21.5.3 Peripheral Mux Register 0

Name: PMR0

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 0**

21.5.4 Peripheral Mux Register 1

Name: PMR1

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Peripheral Multiplexer Select bit 1**

{PMR1, PMR0}	Selected Peripheral Function
00	A
01	B
10	C
11	D

21.5.5 Output Driver Enable Register

Name: ODER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Driver Enable**

0 = The output driver is disabled for the corresponding pin.

1 = The output driver is enabled for the corresponding pin.

21.5.6 Output Value Register

Name: OVR

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Output Value**

0 = The value to be driven on the I/O line is 0.

1 = The value to be driven on the I/O line is 1.

21.5.7 Pin Value Register

Name: PVR

Access: Read

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Pin Value**

0 = The I/O line is at level ‘0’.

1 = The I/O line is at level ‘1’.

Note that the level of a pin can only be read when GPER is set or interrupt is enabled for the pin.

21.5.8 Pull-up Enable Register

Name: PUER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Pull-up Enable**

0 = The internal pull-up resistor is disabled for the corresponding pin.

1 = The internal pull-up resistor is enabled for the corresponding pin.

21.5.9 Open Drain Mode Enable Register

Name: ODMER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Open Drain Mode Enable**

0 = Open drain mode is disabled for the corresponding pin.

1 = Open drain mode is enabled for the corresponding pin.

21.5.10 Interrupt Enable Register

Name: IER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Enable**

0 = Interrupt is disabled for the corresponding pin.

1 = Interrupt is enabled for the corresponding pin.

21.5.11 Interrupt Mode Register 0

Name: IMR0

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Mode Bit 0**

21.5.12 Interrupt Mode Register 1

Name: IMR1

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Mode Bit 1**

{IMR1, IMR0}	Interrupt Mode
00	Pin Change
01	Rising Edge
10	Falling Edge
11	Reserved

21.5.13 Glitch Filter Enable Register

Name: GFER

Access: Read, Write, Set, Clear, Toggle

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Glitch Filter Enable**

0 = Glitch filter is disabled for the corresponding pin.

1 = Glitch filter is enabled for the corresponding pin.

NOTE! The value of this register should only be changed when IER is ‘0’. Updating this GFER while interrupt on the corresponding pin is enabled can cause an unintentional interrupt to be triggered.

21.5.14 Interrupt Flag Register

Name: IFR

Access: Read, Clear

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-31: Interrupt Flag**

0 = An interrupt condition has been detected on the corresponding pin.

1 = No interrupt condition has been detected on the corresponding pin.

The number of interrupt request lines is dependant on the number of I/O pins on the MCU. Refer to the product specific data for details. Note also that a bit in the Interrupt Flag register is only valid if the corresponding bit in IER is set.

21.6 Programming Examples

21.6.1 8-bit LED-Chaser

```
                    // Set R0 to GPIO base address
                    mov    R0, LO(AVR32_GPIO_ADDRESS)
                    orh    R0, HI(AVR32_GPIO_ADDRESS)

                    // Enable GPIO control of pin 0-8
                    mov    R1, 0xFF
                    st.w   R0[AVR32_GPIO_GPERS], R1

                    // Set initial value of port
                    mov    R2, 0x01
                    st.w   R0[AVR32_GPIO_OVRS], R2

                    // Set up toggle value. Two pins are toggled
                    // in each round. The bit that is currently set,
                    // and the next bit to be set.
                    mov    R2, 0x0303
                    orh    R2, 0x0303


                loop:
                    // Only change 8 LSB
                    mov    R3, 0x00FF
                    and    R3, R2
                    st.w   R0[AVR32_GPIO_OVRT], R3
                    rol    R2
                    rcall  delay
                    rjmp   loop
```

It is assumed in this example that a subroutine "delay" exists that returns after a given time.

21.6.2 Configuration of USART pins

The example below shows how to configure a peripheral module to control I/O pins. It assumed in this example that the USART receive pin (RXD) is connected to PC16 and that the USART transmit pin (TXD) is connected to PC17. For both pins, the USART is peripheral B. In this example, the state of the GPIO registers is assumed to be unknown. The two USART pins are therefore first set to be controlled by the GPIO with output drivers disabled. The pins can then be assured to be tri-stated while changing the Peripheral Mux Registers.

```
                    // Set up pointer to GPIO, PORTC
                    mov    R0, LO(AVR32_GPIO_ADDRESS + PORTC_OFFSET)
                    orh    R0, HI(AVR32_GPIO_ADDRESS + PORTC_OFFSET)

                    // Disable output drivers
```

```
mov     R1, 0x0000
orh     R1, 0x0003
st.w    R0[AVR32_GPIO_ODERC], R1

// Make the GPIO control the pins
st.w    R0[AVR32_GPIO_GPERS], R1

// Select peripheral B on PC16-PC17
st.w    R0[AVR32_GPIO_PMR0S], R1
st.w    R0[AVR32_GPIO_PMR1C], R1

// Enable peripheral control
st.w    R0[AVR32_GPIO_GPERC], R1
```

22. Serial Peripheral Interface (SPI)

Rev: 1.9.2.0

22.1 Features

- **Supports Communication with Serial External Devices**
  - **Four Chip Selects with External Decoder Support Allow Communication with Up to 15 Peripherals**
  - **Serial Memories, such as DataFlash and 3-wire EEPROMs**
  - **Serial Peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors**
  - **External Co-processors**
- **Master or Slave Serial Peripheral Bus Interface**
  - **8- to 16-bit Programmable Data Length Per Chip Select**
  - **Programmable Phase and Polarity Per Chip Select**
  - **Programmable Transfer Delays Between Consecutive Transfers and Between Clock and Data Per Chip Select**
  - **Programmable Delay Between Consecutive Transfers**
  - **Selectable Mode Fault Detection**
- **Connection to PDC Channel Capabilities Optimizes Data Transfers**
  - **One Channel for the Receiver, One Channel for the Transmitter**
  - **Next Buffer Support**

22.2 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

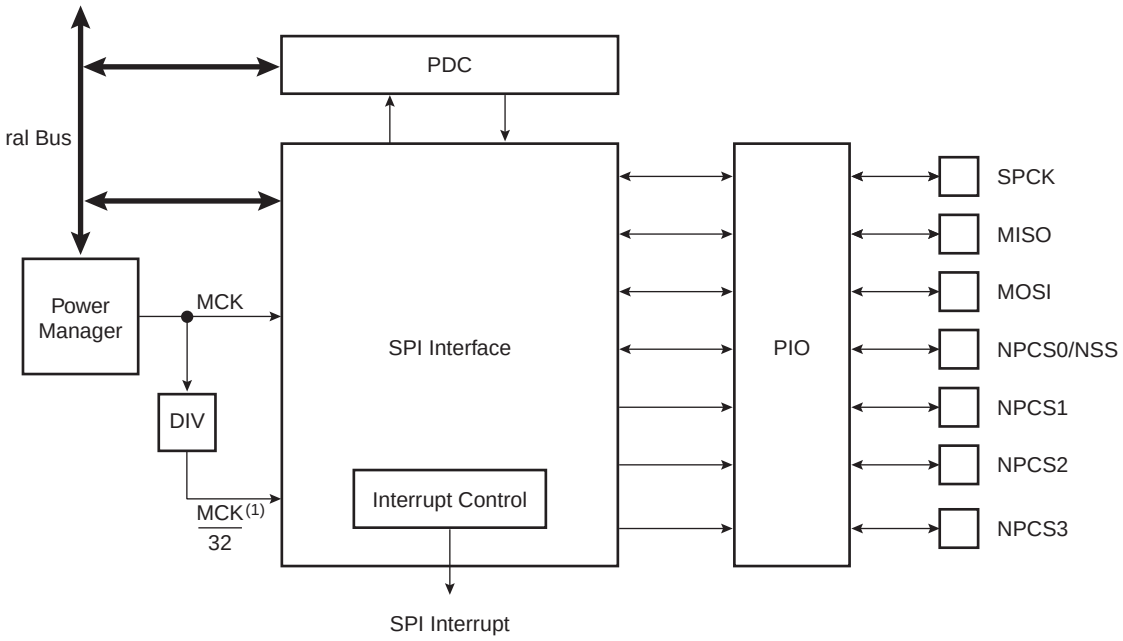
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

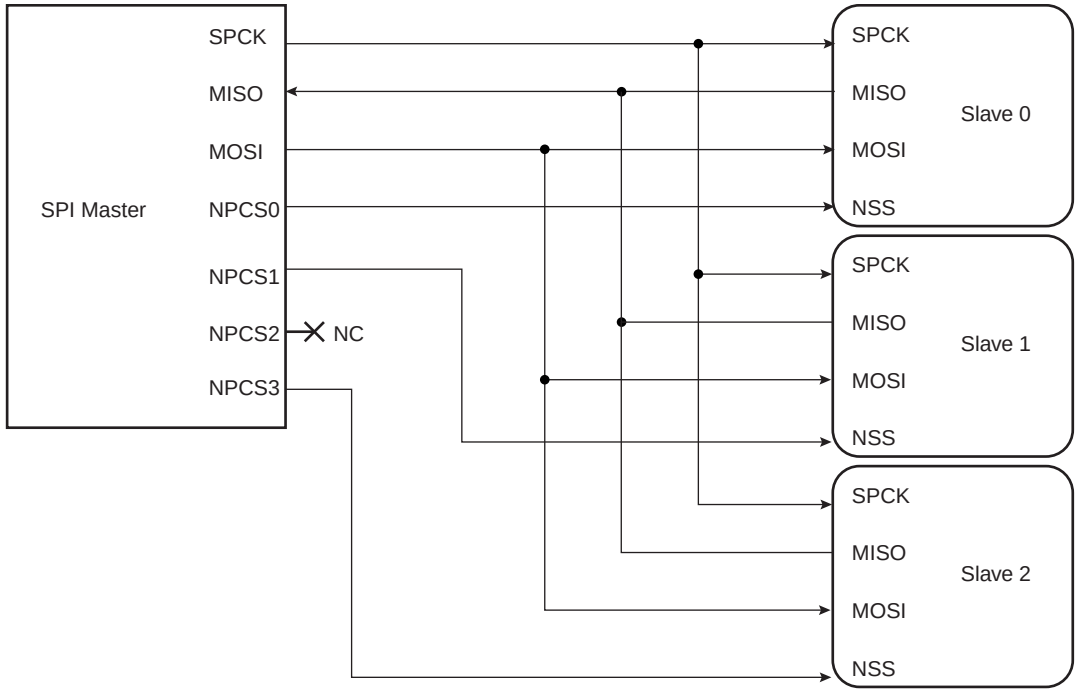
22.3 Block Diagram

Figure 22-1. Block Diagram



22.4 Application Block Diagram

Figure 22-2. Application Block Diagram: Single Master/Multiple Slave Implementation



22.5 Signal Description

Table 22-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1-NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

22.6 Product Dependencies

22.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

22.6.2 Power Management

The SPI clock is generated by the Power Manager. Before using the SPI, the programmer must ensure that the SPI clock is enabled in the Power Manager.

In the SPI description, Master Clock (MCK) is the clock of the peripheral bus to which the SPI is connected.

22.6.3 Interrupt

The SPI interface has an interrupt line connected to the Interrupt Controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

22.7 Functional Description

22.7.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

22.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

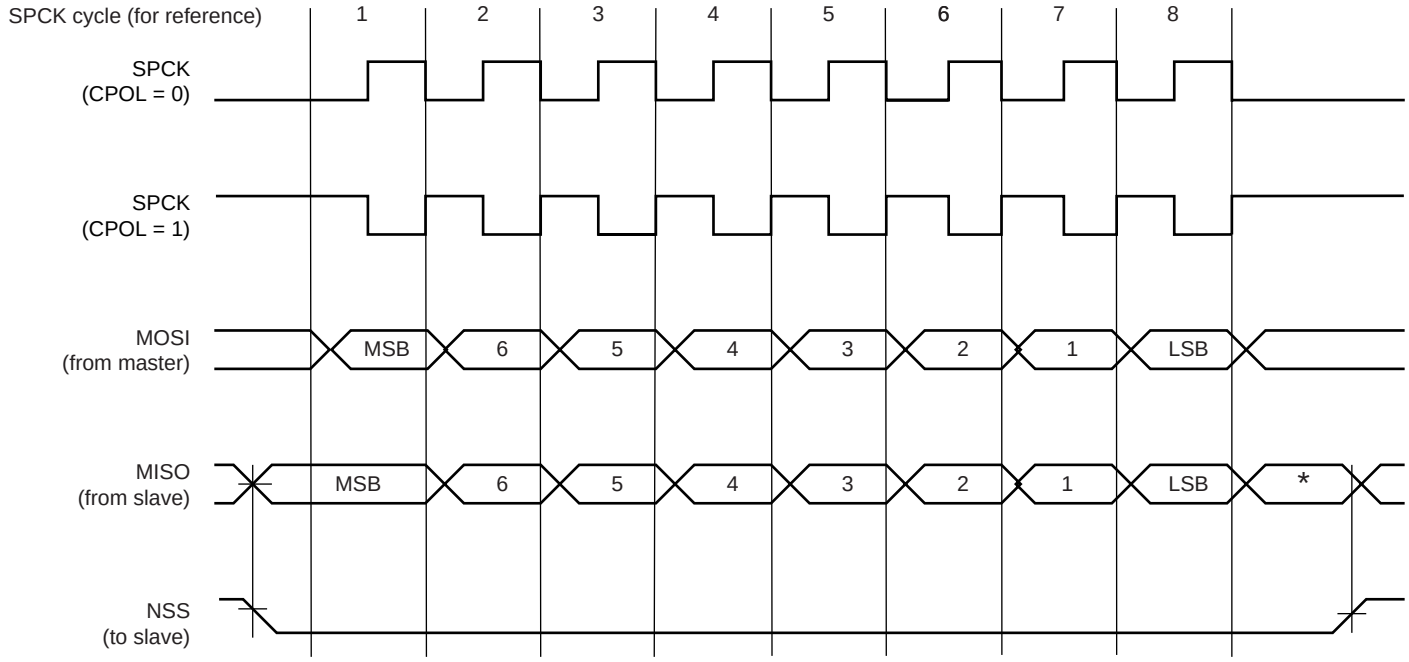
Table 22-2 shows the four modes and corresponding parameter settings.

Table 22-2. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

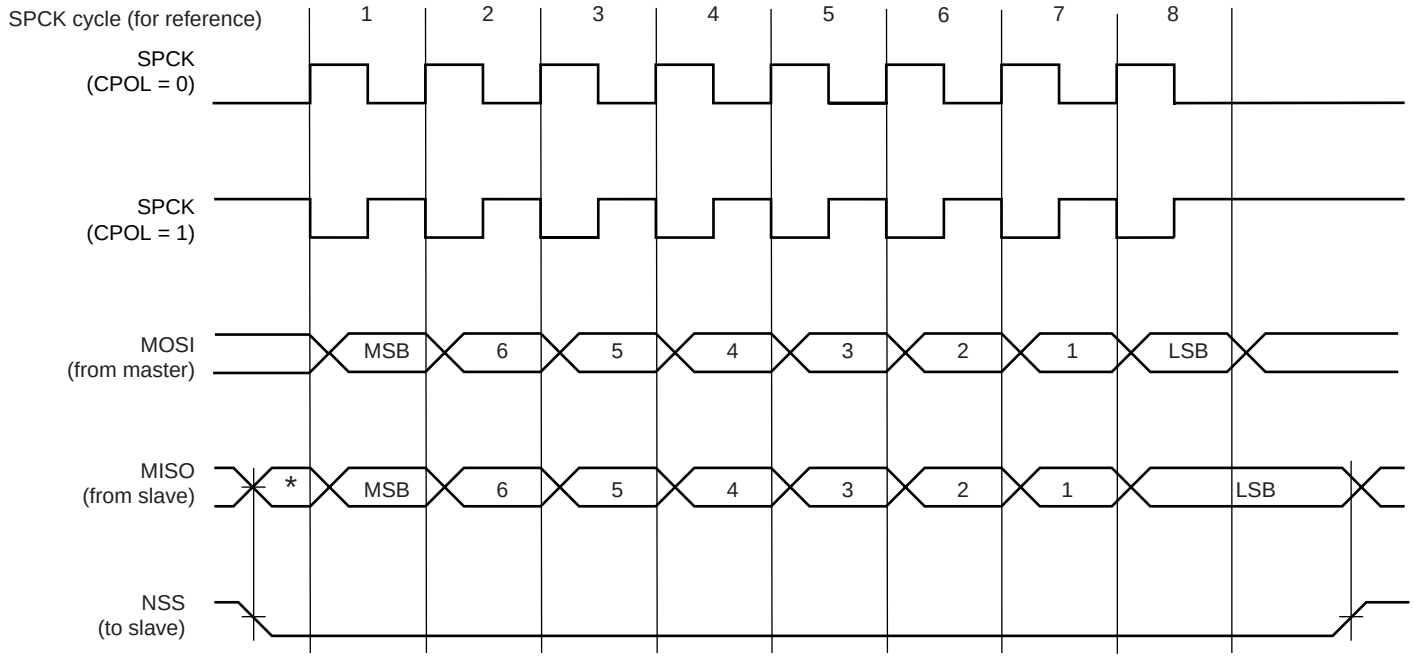
Figure 22-3 and Figure 22-4 show examples of data transfers.

Figure 22-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 22-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

22.7.3 Master Mode Operations

When configured in Master Mode, the SPI uses the internal programmable baud rate generator as clock source. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writing the TDR, the PCS field must be set in order to select a slave.

If new data is written in TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to RDR, the data in TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SR). When new data is written in TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

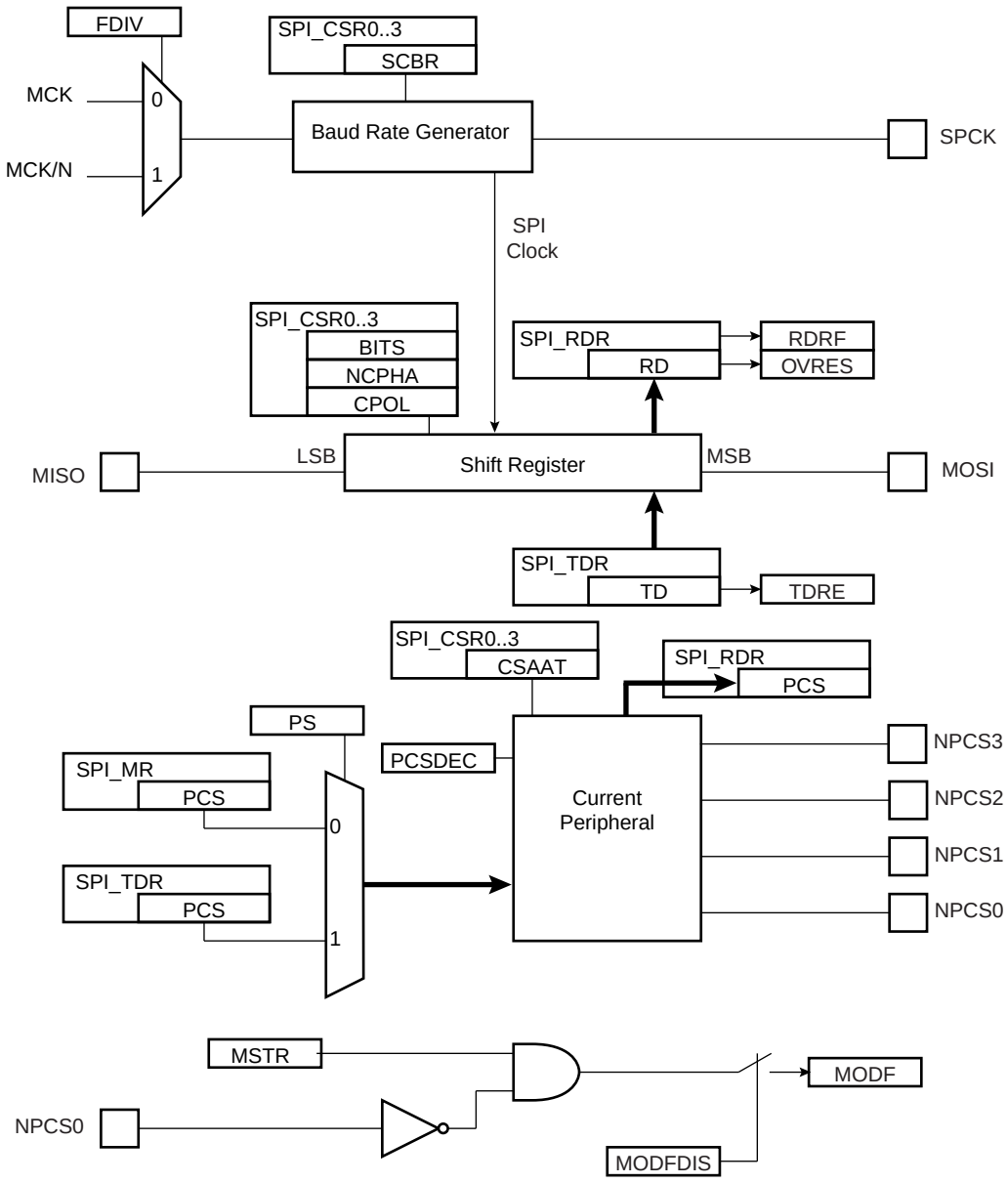
The transfer of received data from the Shift Register in RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SR). When the received data is read, the RDRF bit is cleared.

If the RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SR is set. As long as this flag is set, data is loaded in RDR. The user has to read the status register to clear the OVRES bit.

Figure 22-5 on page 182 shows a block diagram of the SPI when operating in Master Mode. Figure 22-6 on page 183 shows a flow chart describing how transfers are handled.

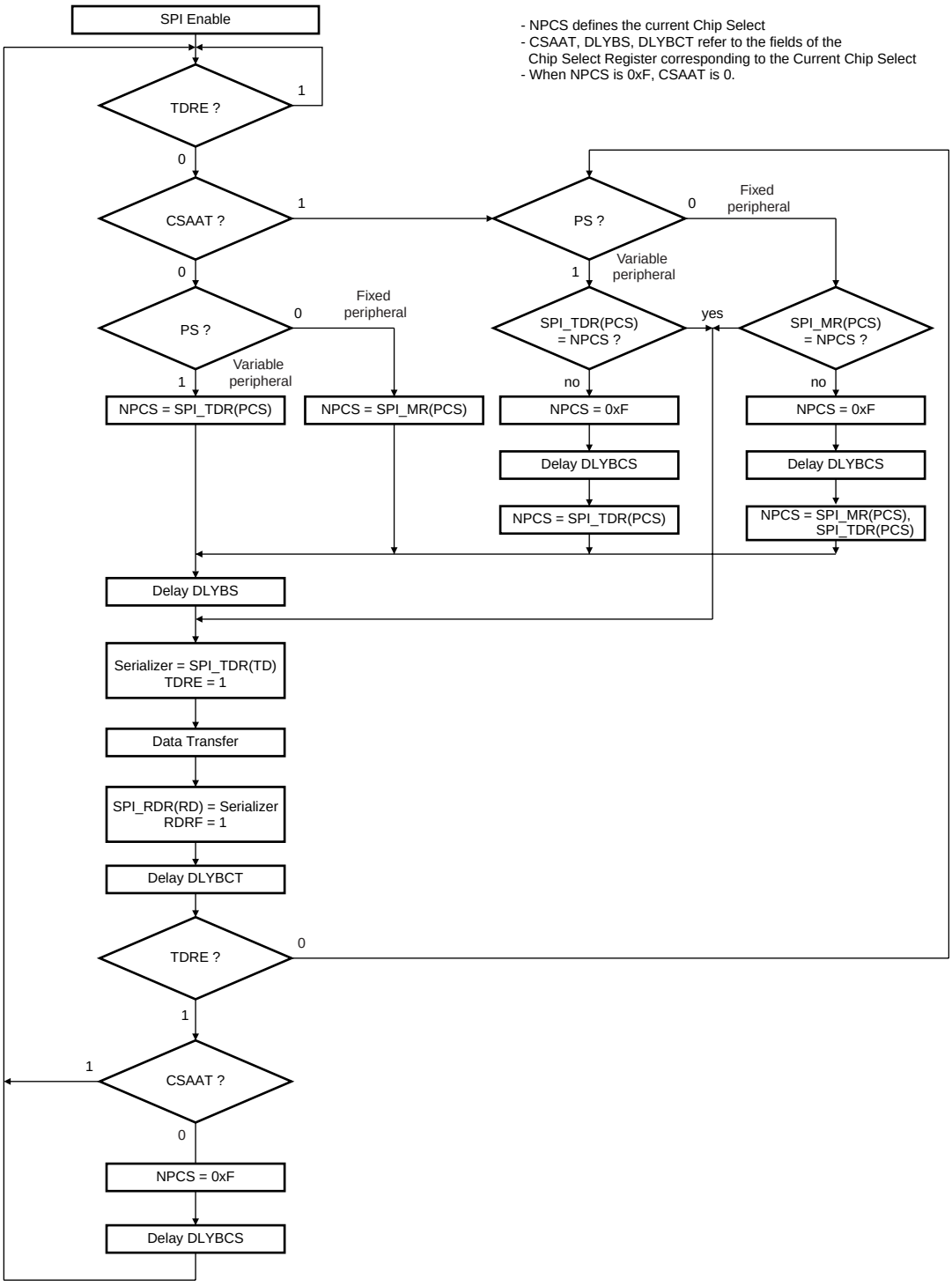
22.7.3.1 Master Mode Block Diagram

Figure 22-5. Master Mode Block Diagram



22.7.3.2 Master Mode Flow Diagram

Figure 22-6. Master Mode Flow Diagram S



22.7.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK) or the Master Clock divided by 32, by a value between 1 and 255. The selection between Master Clock or Master Clock divided by 32 is done by the FDIV value set in the Mode Register

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255*32.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

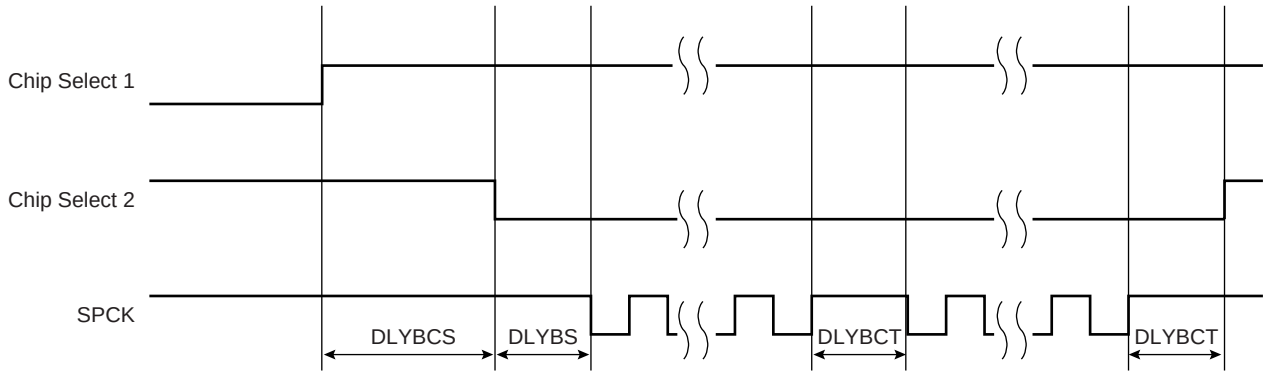
22.7.3.4 Transfer Delays

Figure 22-7 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 22-7. Programmable Delays



22.7.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral
- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing the PS bit to zero in MR (Mode Register). In this case, the current peripheral is defined by the PCS field in MR and the PCS field in TDR have no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

22.7.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

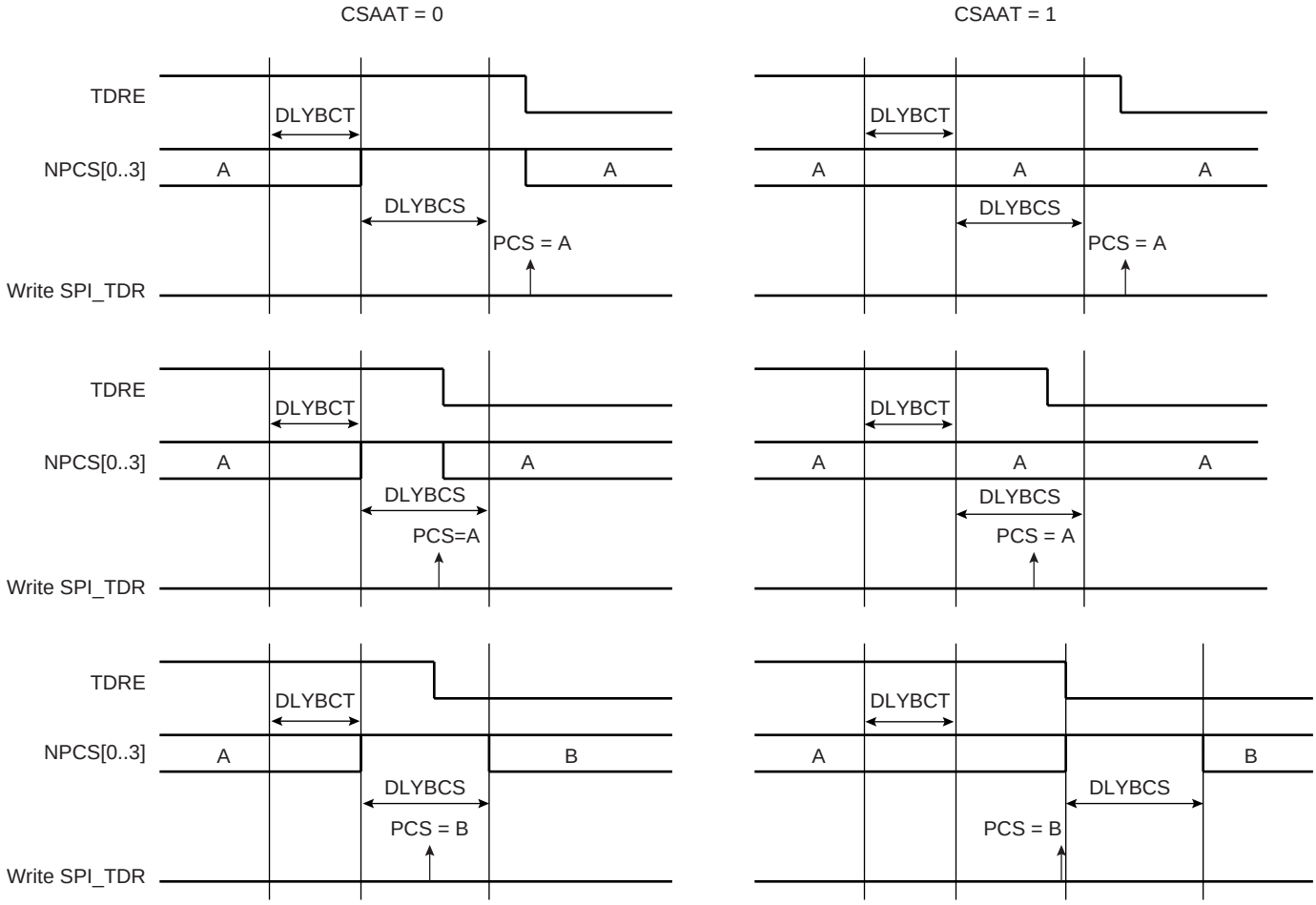
22.7.3.7 Peripheral Deselection

When operating normally, as soon as the transfer of the last data written in TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

Figure 22-8 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 22-8. Peripheral Deselection



22.7.3.8 Mode Fault Detection

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal. NPCS0, MOSI, MISO and SPCK must be configured in open-drain through the PIO controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the MODF bit in the SR is set until the SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (MR).

22.7.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the BITS field of the Chip Select Register 0 (CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If RDRF is already high when the data is transferred, the Overrun bit rises and the data transfer to RDR is aborted.

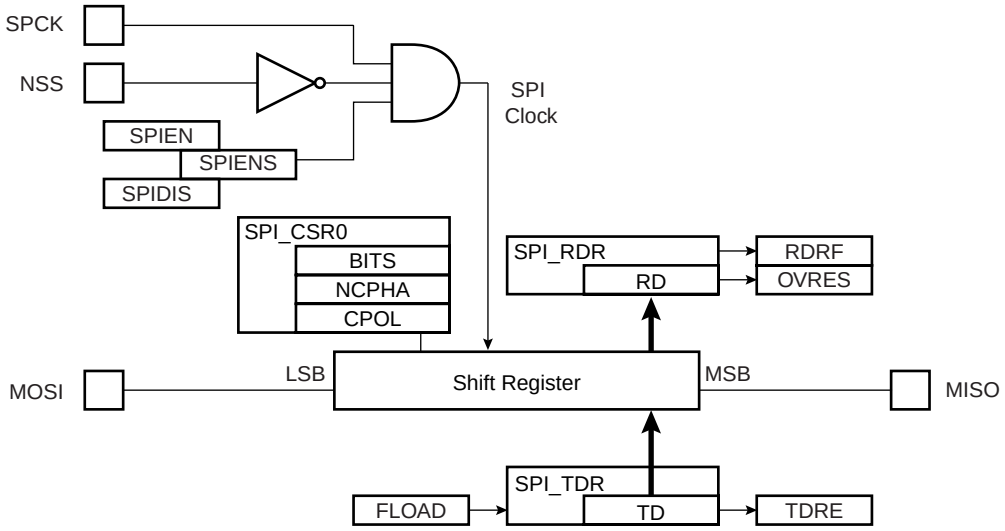
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in TDR since the last load from TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 22-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 22-9. Slave Mode Functional Block Diagram



22.8 Serial Peripheral Interface (SPI) User Interface

Table 22-3. SPI Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	Control Register	CR	Write-only	---
0x04	Mode Register	MR	Read/Write	0x0
0x08	Receive Data Register	RDR	Read-only	0x0
0x0C	Transmit Data Register	TDR	Write-only	---
0x10	Status Register	SR	Read-only	0x000000F0
0x14	Interrupt Enable Register	IER	Write-only	---
0x18	Interrupt Disable Register	IDR	Write-only	---
0x1C	Interrupt Mask Register	IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	CSR0	Read/Write	0x0
0x34	Chip Select Register 1	CSR1	Read/Write	0x0
0x38	Chip Select Register 2	CSR2	Read/Write	0x0
0x3C	Chip Select Register 3	CSR3	Read/Write	0x0
0x100 - 0x124	Reserved for the PDC			

22.8.1 SPI Control Register

Name: CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPDIS is set, SPI finishes its transfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after a software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

22.8.2 SPI Mode Register

Name: MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	–	MODFDIS	FDIV	PCSDEC	PS	MSTR

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

CSR0 defines peripheral chip select signals 0 to 3.

CSR1 defines peripheral chip select signals 4 to 7.

CSR2 defines peripheral chip select signals 8 to 11.

CSR3 defines peripheral chip select signals 12 to 14.

- **FDIV: Clock Selection**

0 = The SPI operates at MCK.

1 = The SPI operates at MCK/32.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled.

LLB controls the local loopback on the data serializer for testing in Master Mode only. MISO is internally connected to MOSI.

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don't care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is less than or equal to six, six MCK periods (or 6*N MCK periods if FDIV is set) will be inserted by default.

Otherwise, the following equation determines the delay:

If FDIV is 0:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

If FDIV is 1:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS \times N}{MCK}$$

22.8.3 SPI Receive Data Register

Name: RDR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
RD							
7	6	5	4	3	2	1	0
RD							

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

22.8.4 SPI Transmit Data Register

Name: TDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

PCS: Peripheral Chip Select

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

This field is only used if Variable Peripheral Select is active (PS = 1).

22.8.5 SPI Status Register

Name: SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of RDR

1 = Data has been received and the received data has been transferred from the serializer to RDR since the last read of RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SR.

1 = A Mode Fault occurred since the last read of the SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SR.

1 = An overrun has occurred since the last read of SR.

An overrun occurs when RDR is loaded at least twice from the serializer since the last read of the RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in RCR or RNCR.

1 = The Receive Counter Register has reached 0 since the last write in RCR or RNCR.

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in TCR or TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in TCR or TNCR.

- **RXBUFF: RX Buffer Full**

0 = RCR or RNCR has a value other than 0.

1 = Both RCR and RNCR has a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = TCR or TNCR has a value other than 0.

1 = Both TCR and TNCR has a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in TDR.

1 = TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

22.8.6 SPI Interrupt Enable Register

Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **NSSR: NSS Rising Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

22.8.7 SPI Interrupt Disable Register

Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**
- **NSSR: NSS Rising Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

22.8.8 SPI Interrupt Mask Register

Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**
- **NSSR: NSS Rising Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

22.8.9 SPI Chip Select Register

Name: CSR0... CSR3

Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							
23	22	21	20	19	18	17	16
DLYBS							
15	14	13	12	11	10	9	8
SCBR							
7	6	5	4	3	2	1	0
BITS				CSAAT	–	NCPHA	CPOL

- **CPOL: Clock Polarity**

0 = The inactive state value of SPCK is logic level zero.

1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSAAT: Chip Select Active After Transfer**

0 = The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1 = The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used, see Table 22-4 on page 201.

Table 22-4. BITS, Bits Per Transfer

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

If FDIV is 0:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

If FDIV is 1:

$$\text{SPCK Baudrate} = \frac{MCK}{(N \times SCBR)}$$

Note: N = 32

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

If FDIV is 0:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

If FDIV is 1:

$$\text{Delay Before SPCK} = \frac{N \times DLYBS}{MCK}$$

Note: N = 32

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT equals zero, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

If FDIV is 0:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK} + \frac{SCBR}{2MCK}$$

If FDIV is 1:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times N \times DLYBCT}{MCK} + \frac{N \times SCBR}{2MCK}$$

N = 32

23. Two-wire Interface (TWI)

Rev: 2.0.2.1

23.1 Features

- **Compatible with Standard Two-wire Serial Memories**
- **One, Two or Three Bytes for Slave Address**
- **Sequential Read/Write Operations**
- **Master, Multi-master and Slave Mode Operation**
- **Bit Rate: Up to 400 Kbits**
- **General Call Supported in Slave mode**
- **Connection to Peripheral DMA Controller (PDC) Channel Capabilities Optimizes Data Transfers in Master Mode Only**
  - **One Channel for the Receiver, One Channel for the Transmitter**
  - **Next Buffer Support**

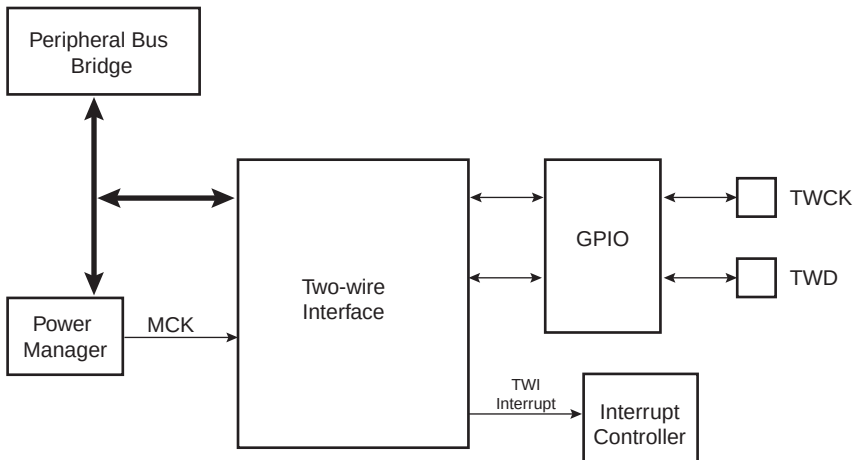
23.2 Description

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel two-wire bus Serial EEPROM. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported. Arbitration of the bus is performed internally and puts the TWI in slave mode automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

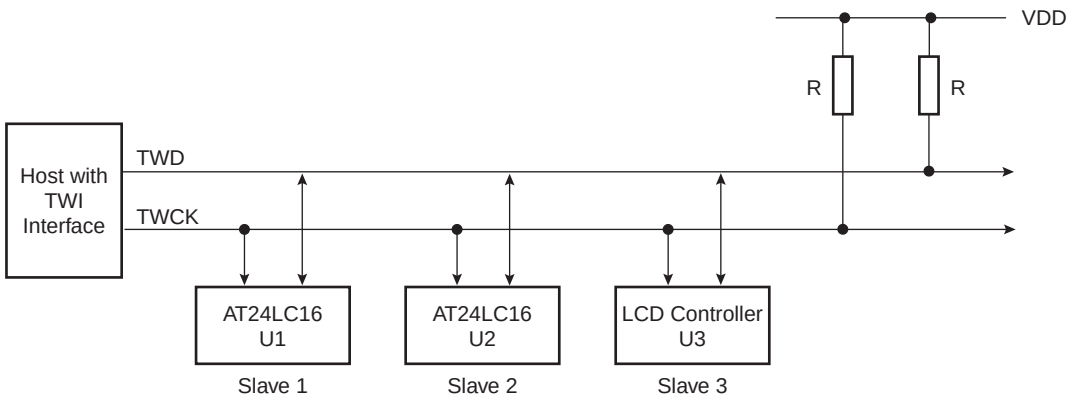
23.3 Block Diagram

Figure 23-1. Block Diagram



23.4 Application Block Diagram

Figure 23-2. Application Block Diagram



23.4.1 I/O Lines Description

Table 23-1. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

23.5 Product Dependencies

23.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 23-2 on page 204). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with GPIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the GPIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

23.5.2 Power Management

The TWI clock is generated by the power manager. Before using the TWI, the programmer must ensure that the TWI clock is enabled in the power manager.

In the TWI description, Master Clock (MCK) is the clock of the peripheral bus to which the TWI is connected.

23.5.3 Interrupt

The TWI interface has an interrupt line connected to the interrupt controller. In order to handle interrupts, the interrupt controller must be programmed before configuring the TWI.

23.6 List of Abbreviations

Table 23-2. Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
RS	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

23.7 Functional Description

23.7.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 23-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 23-3).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 23-3. START and STOP Conditions

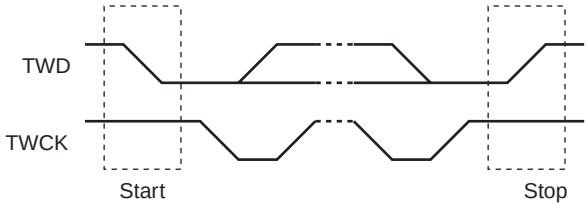
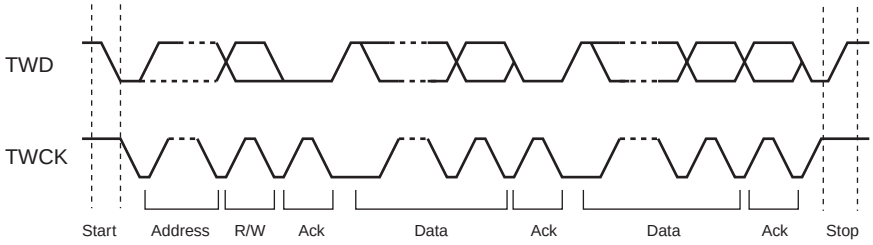


Figure 23-4. Transfer Format



23.7.2 Modes of Operation

The TWI has six modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following chapters.

23.8 Master Mode

23.8.1 Definition

The Master is the device which starts a transfer, generates a clock and stops it.

23.8.2 Application Block Diagram

Figure 23-5. Master Mode Typical Application Block Diagram

23.8.3 Programming Master Mode

The following registers have to be programmed before entering Master mode:

1. DADR (+ IADRSZ + IADR if a 10 bit device is addressed): The device address is used to access slave devices in read or write mode.
2. CKDIV + CHDIV + CLDIV: Clock Waveform.
3. SVDIS: Disable the slave mode.
4. MSEN: Enable the master mode.

23.8.4 Transmitting Data

After the master initiates a Start condition, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction (write or read). If this bit is 0, it indicates a write operation (transmit operation). If the bit is 1, it indicates a request for data read (receive operation).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse, the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the NACK bit in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI_IER). After writing in the transmit-holding register (TWI_THR), the transmission will start as soon as the bus is free. The data is shifted in the internal shifter and when an acknowledge is detected, the TXRDY bit is set until a new write in the TWI_THR (see Figure 23-7 on page 208). The master generates a stop condition to end the transfer.

TXRDY is used as Transmit Ready for the PDC transmit channel.

The read sequence begins by setting the START bit. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

The TWI interface performs various transfer formats (7-bit slave address, 10-bit slave address). The three internal address bytes are configurable through the Master Mode register (TWI_MMR). If the slave device supports only a 7-bit address, the IADRSZ must be set to 0. For slave address higher than 7 bits, the user must configure the address size **(**IADRSZ) and set the other slave address bits in the internal address register (TWI_IADR).

RXRDY is used as Receive Ready for the PDC receive channel.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10):

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 23-6. Master Write with One, Two or Three Bytes Internal Address and One Data Byte

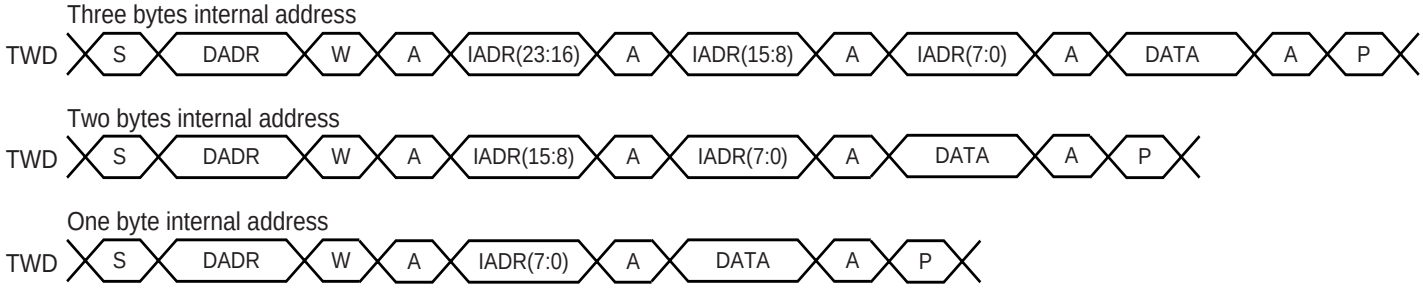


Figure 23-7. Master Write with One Byte Internal Address and Multiple Data Bytes

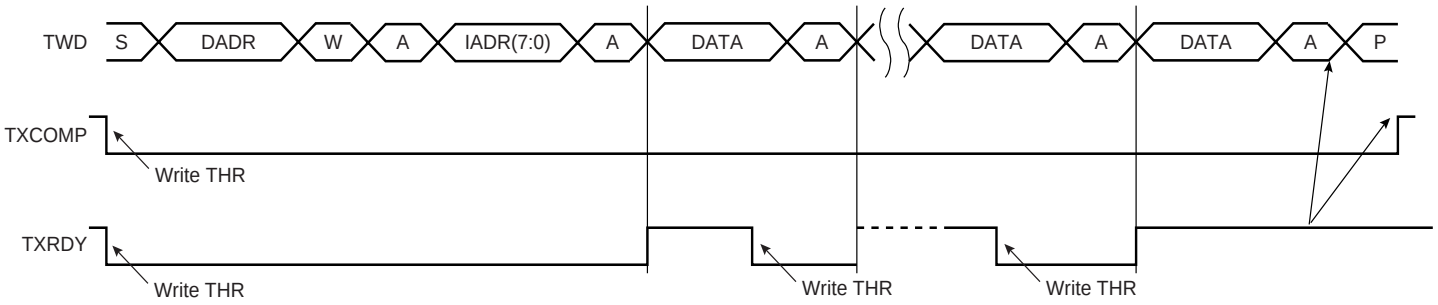


Figure 23-8. Master Read with One, Two or Three Bytes Internal Address and One Data Byte

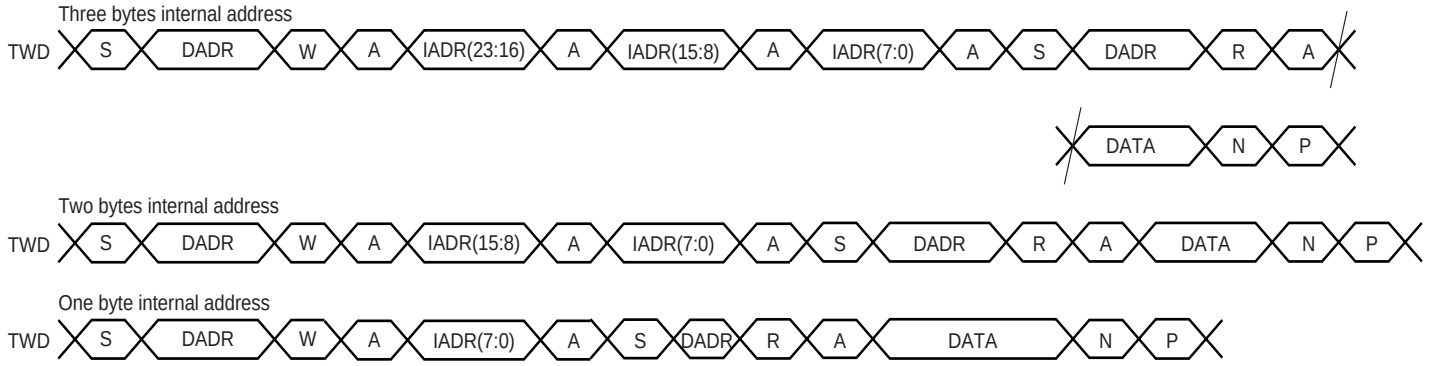


Figure 23-9. Master Read with One Byte Internal Address and Multiple Data Bytes

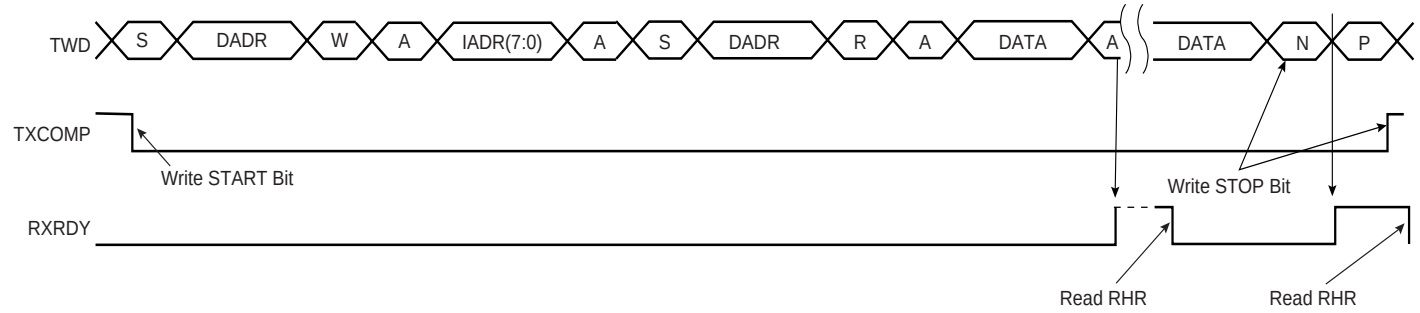
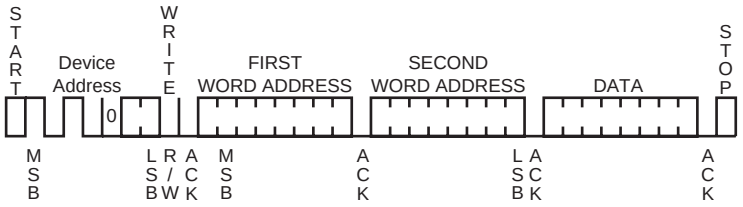


Figure 23-10. Internal Address Usage



23.9 Using the Peripheral DMA Controller (PDC)

The use of the PDC significantly reduces the CPU load. Refer to the PDC chapter for details on programing the PDC.

23.9.1 Read/Write Flowcharts

The flowcharts shown in Figure 23-11 on page 210 and in Figure 23-12 on page 211 give examples for read and write operations in Master Mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 23-11. TWI Write in Master Mode

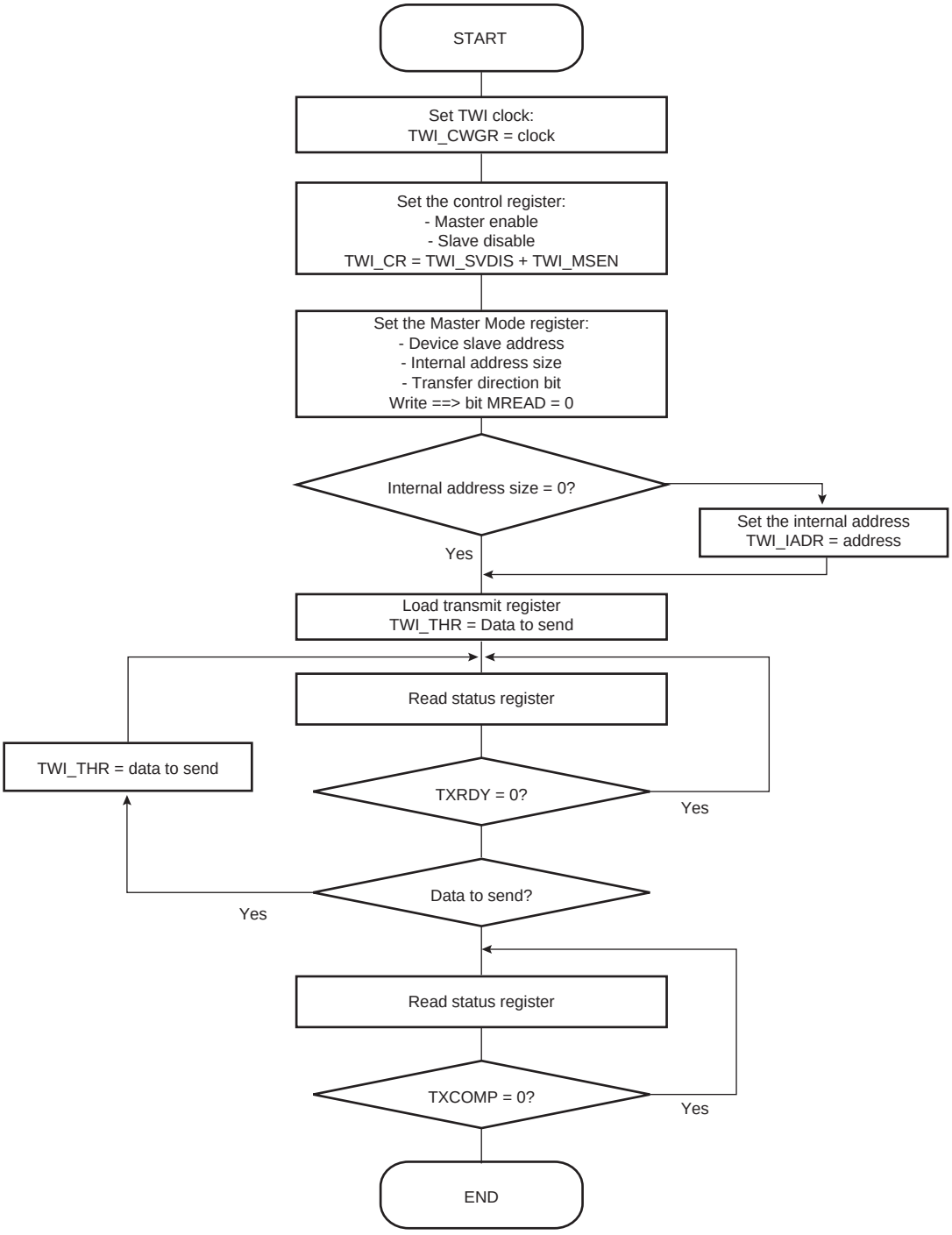
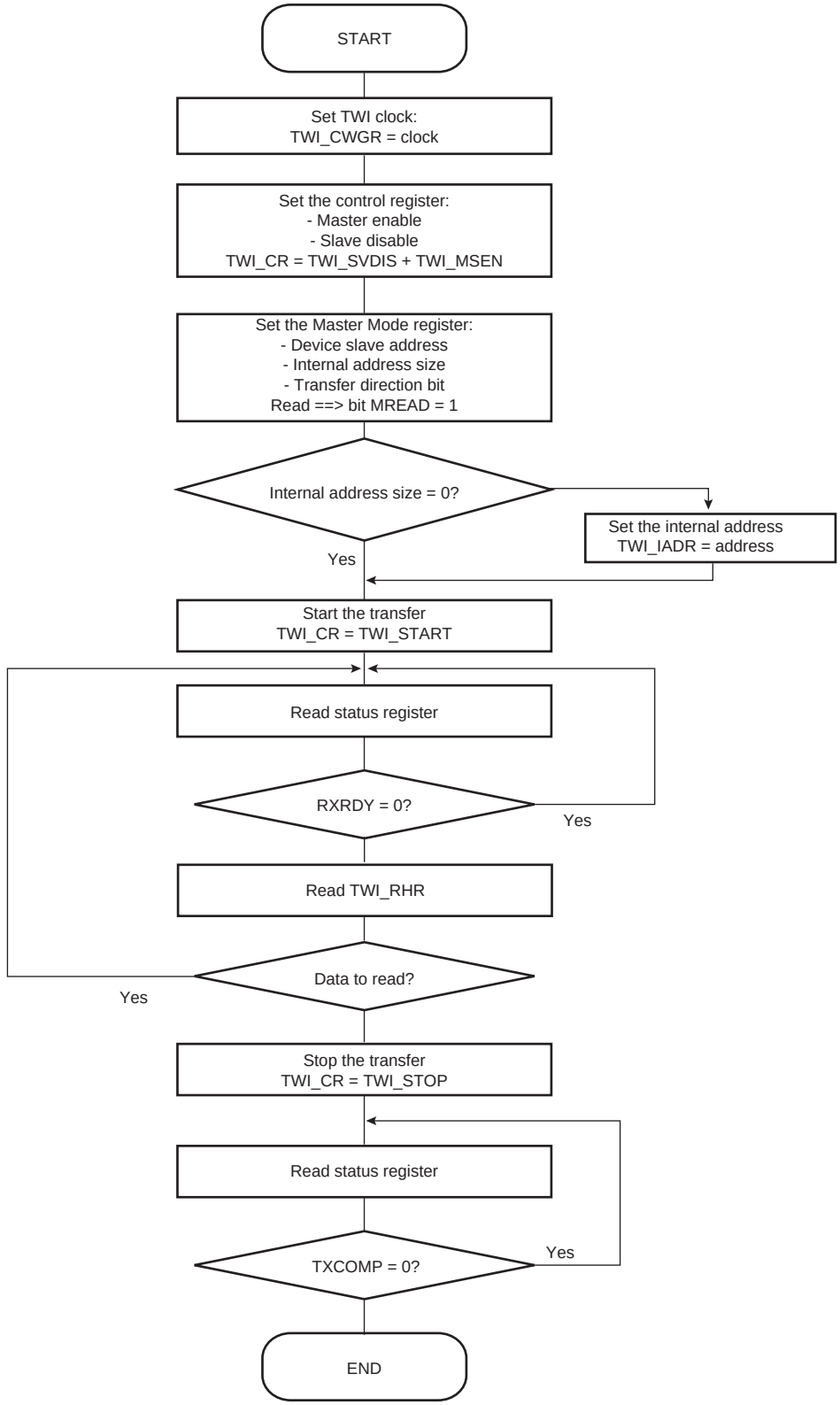


Figure 23-12. TWI Read in Master Mode



23.10 Multi-master Mode

23.10.1 Definition

More than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master who has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in Figure 23-14 on page 213.

23.10.2 Different Multi-master Modes

Two multi-master modes may be distinguished:

1. TWI is considered as a Master only and will never be addressed.
2. TWI may be either a Master or a Slave and may be addressed.

Note: In both Multi-master modes arbitration is supported.

23.10.2.1 TWI as Master Only

In this mode, TWI is considered as a Master only (MSEN is always at one) and must be driven like a Master with the ARBLST (ARBitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the programmer must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see Figure 23-13 on page 213).

Note: The state of the bus (busy or free) is not indicated in the user interface.

23.10.2.2 TWI as Master or Slave

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

Then, in the case where TWI may be either a Master or a Slave, the programmer must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform Slave Access (if TWI is addressed).
2. If TWI has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, TWI scans the bus in order to detect if it is busy or free. When the bus is considered as free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in the case where the Master that won the arbitration wanted to access the TWI.

7. If TWI has to be set in Slave mode, wait until TXCOMP flag is at 1 and then program the Slave mode.

Note: In the case where the arbitration is lost and TWI is addressed, TWI will not acknowledge even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then, the Master must repeat SADR.

Figure 23-13. Programmer Sends Data While the Bus is Busy

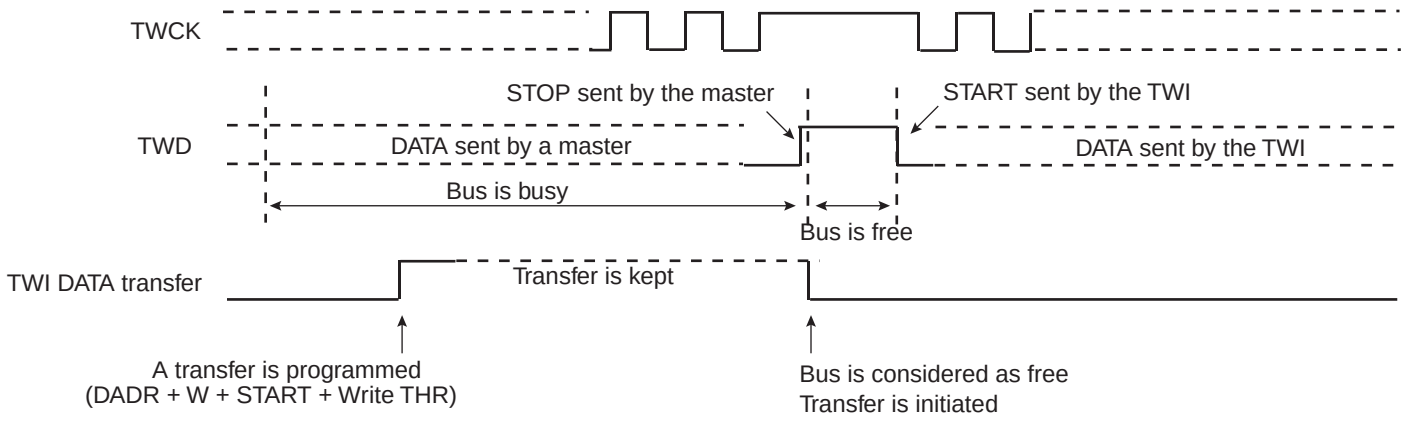
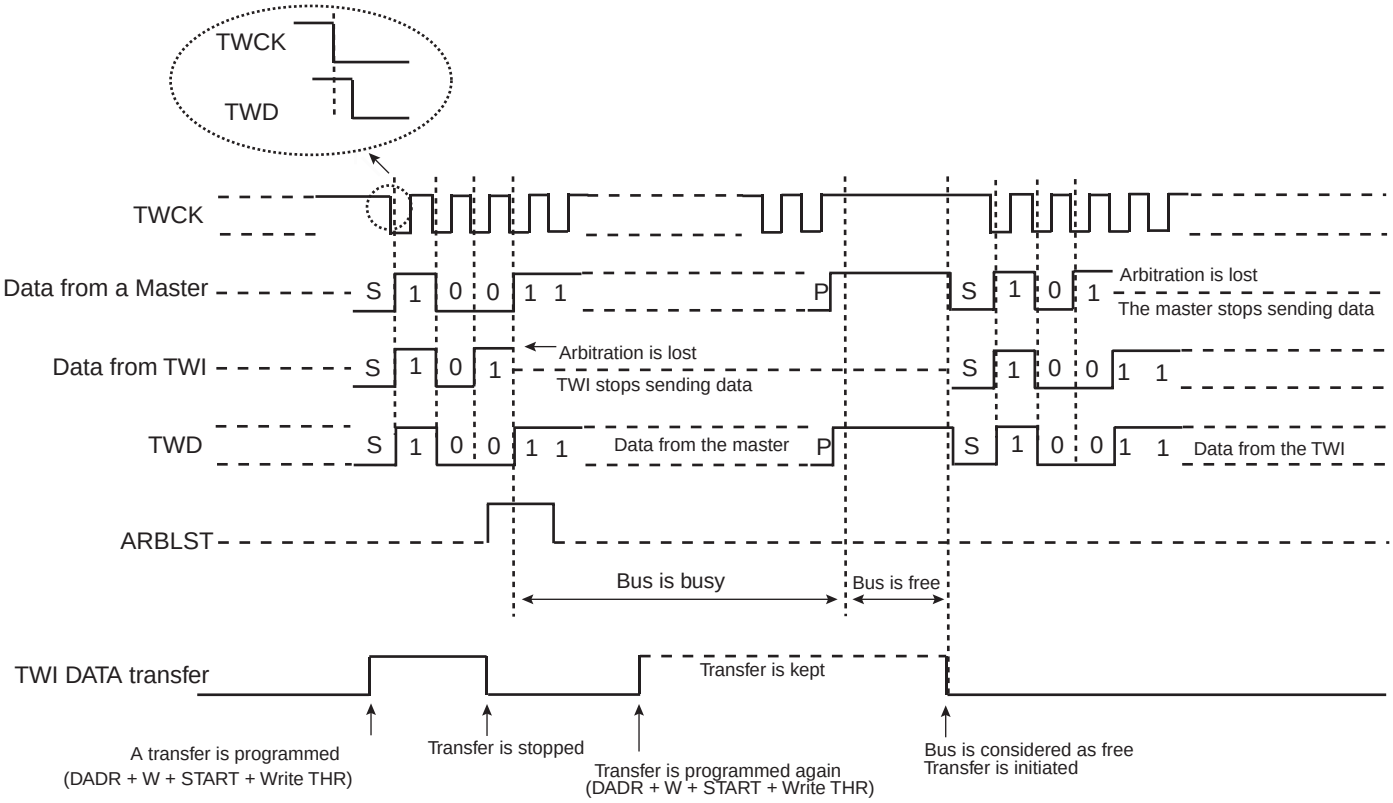
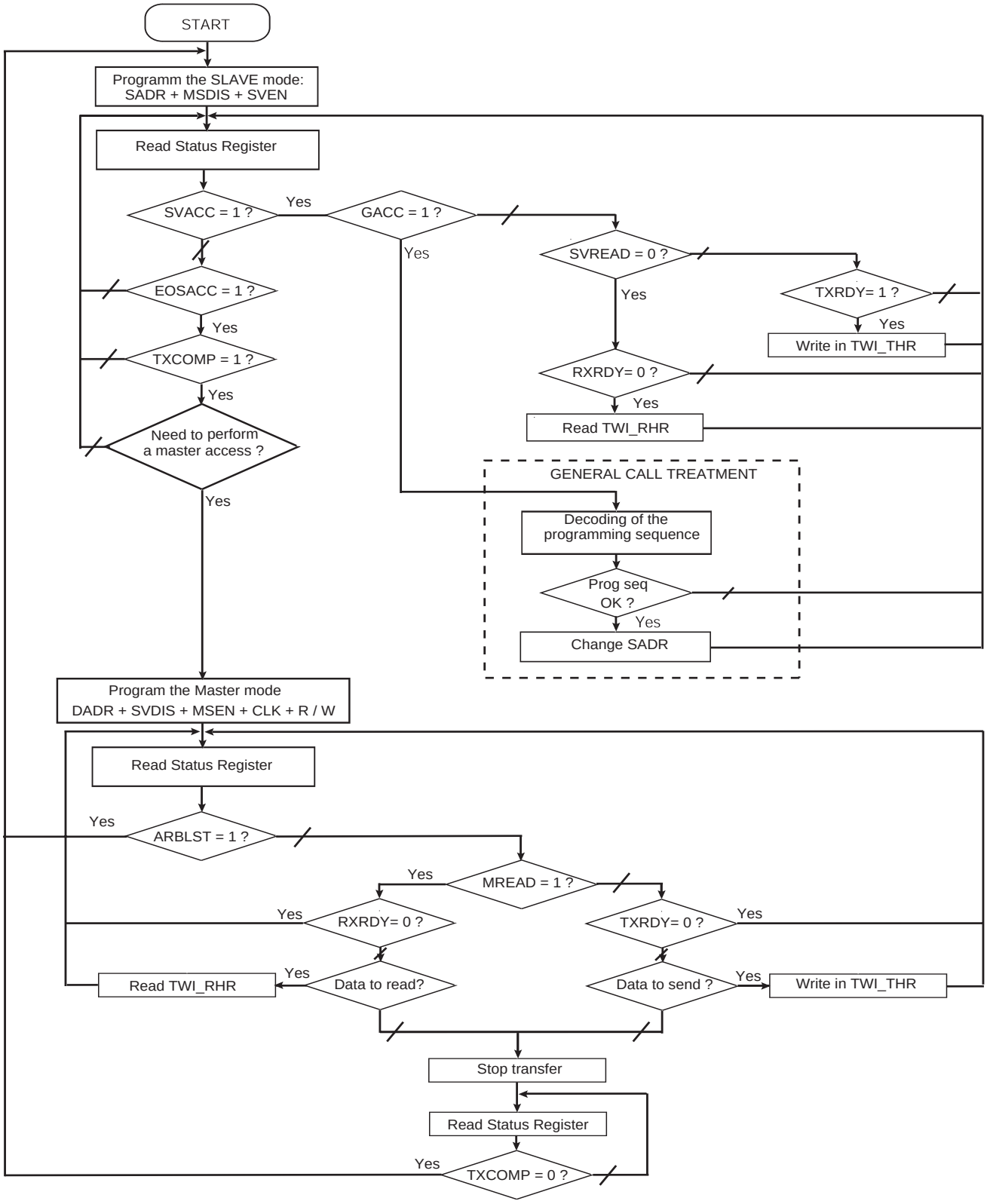


Figure 23-14. Arbitration Cases



The flowchart shown in Figure 23-15 on page 214 gives an example of read and write operations in Multi-master mode.

Figure 23-15. Multi-master Flowchart



23.11 Slave Mode

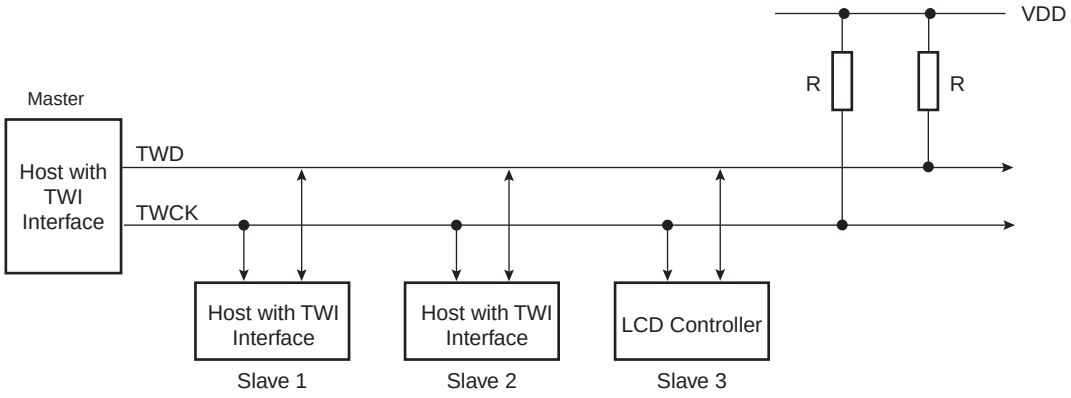
23.11.1 Definition

The Slave Mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED_START and STOP conditions are always provided by the master).

23.11.2 Application Block Diagram

Figure 23-16. Slave Mode Typical Application Block Diagram



23.11.3 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. SADR (TWI_SMR): The slave device address is used in order to be accessed by master devices in read or write mode.
2. MSDIS (TWI_CR): Disable the master mode.
3. SVEN (TWI_CR): Enable the slave mode.

As the device receives the clock, values written in TWI_CWGR are not taken into account.

23.11.4 Receiving Data

After a Start or Repeated Start condition is detected and if the address sent by the Master matches with the Slave address programmed in the SADR (Slave ADdress) field, SVACC (Slave ACCess) flag is set and SVREAD (Slave READ) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a repeated START is detected. When such a condition is detected, EOSACC (End Of Slave ACCess) flag is set.

23.11.4.1 Read Sequence

In the case of a Read sequence (SVREAD is high), TWI transfers data written in the TWI_THR (TWI Transmit Holding Register) until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as a data is written in the TWI_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the shift register is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a repeated START always follows a NACK.

See Figure 23-17 on page 217.

23.11.4.2 Write Sequence

In the case of a Write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI_RHR.

TWI continues receiving data until a STOP condition or a REPEATED_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See Figure 23-18 on page 217.

23.11.4.3 Clock Synchronization Sequence

In the case where TWI_THR or TWI_RHR is not written/read in time, TWI performs a clock synchronization.

Clock stretching information is given by the SCLWS (Clock Wait state) bit.

See Figure 23-20 on page 219 and Figure 23-21 on page 220.

23.11.4.4 General Call

In the case where a GENERAL CALL is performed, GACC (General Call ACCess) flag is set.

After GACC is set, it is up to the programmer to interpret the meaning of the GENERAL CALL and to decode the new address programming sequence.

See Figure 23-19 on page 218.

23.11.4.5 PDC

As it is impossible to know the exact number of data to receive/send, the use of PDC is NOT recommended in SLAVE mode.

23.11.5 Data Transfer

23.11.5.1 Read Operation

The read mode is defined as a data requirement from the master.

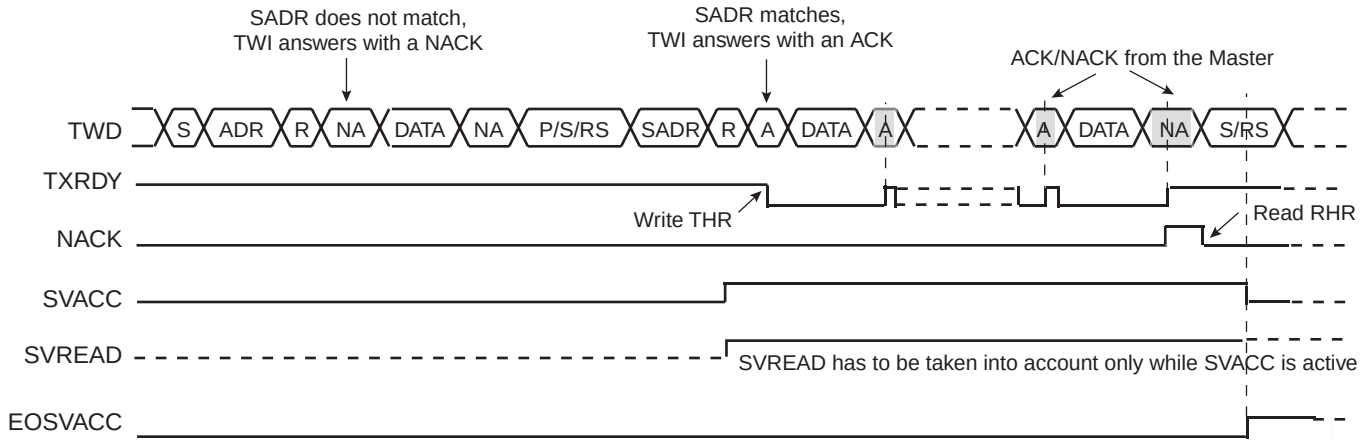
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI_THR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 23-17 on page 217 describes the write operation.

Figure 23-17. Read Access Ordered by a MASTER



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. TXRDY is reset when a data has been transmitted from TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.

23.11.5.2 Write Operation

The write mode is defined as a data transmission from the master.

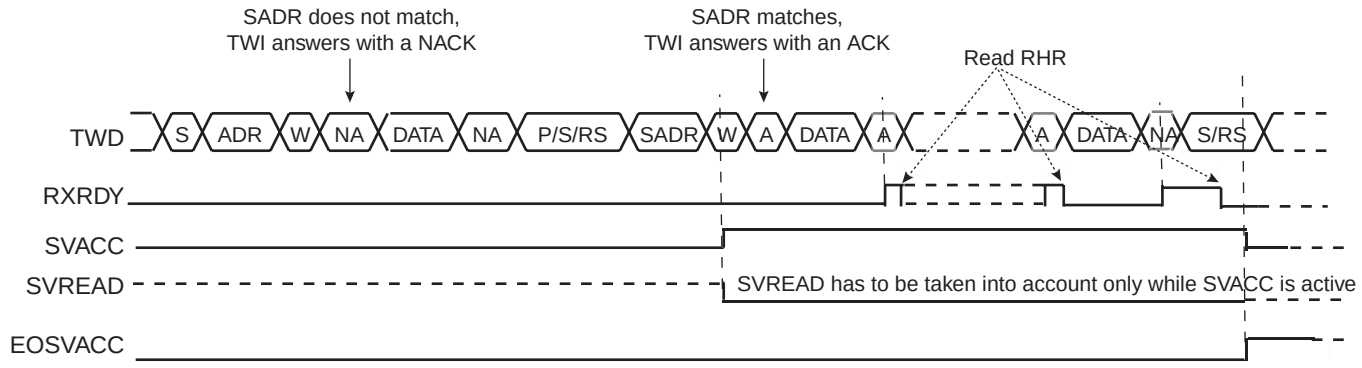
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI_RHR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 23-18 on page 217 describes the Write operation.

Figure 23-18. Write Access Ordered by a Master



Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
2. RXRDY is set when data has been transmitted from the shift register to the TWI_RHR and reset when this data is read.

23.11.5.3 General Call

The general call is performed in order to change the address of the slave.

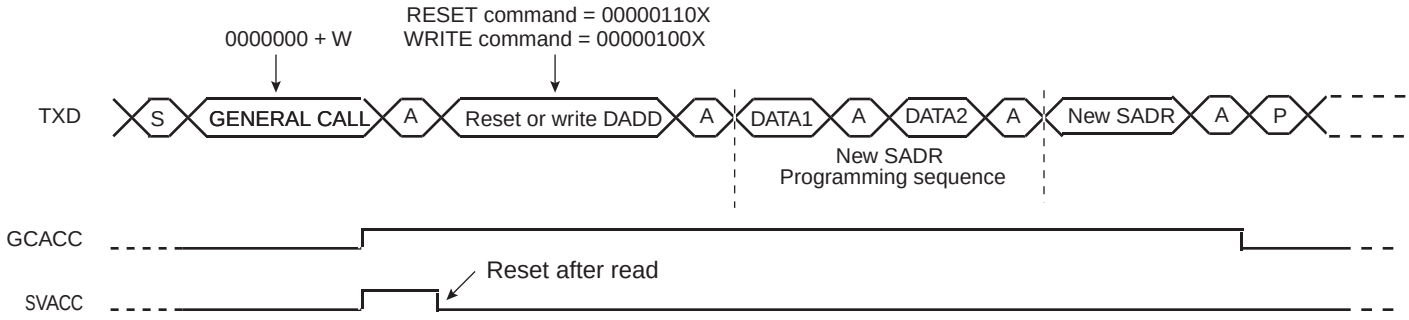
If a GENERAL CALL is detected, GACC is set.

After the detection of General Call, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 23-19 on page 218 describes the General Call access.

Figure 23-19. Master Performs a General Call



Note: This method allows the user to create an own programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

23.11.5.4 Clock Synchronization

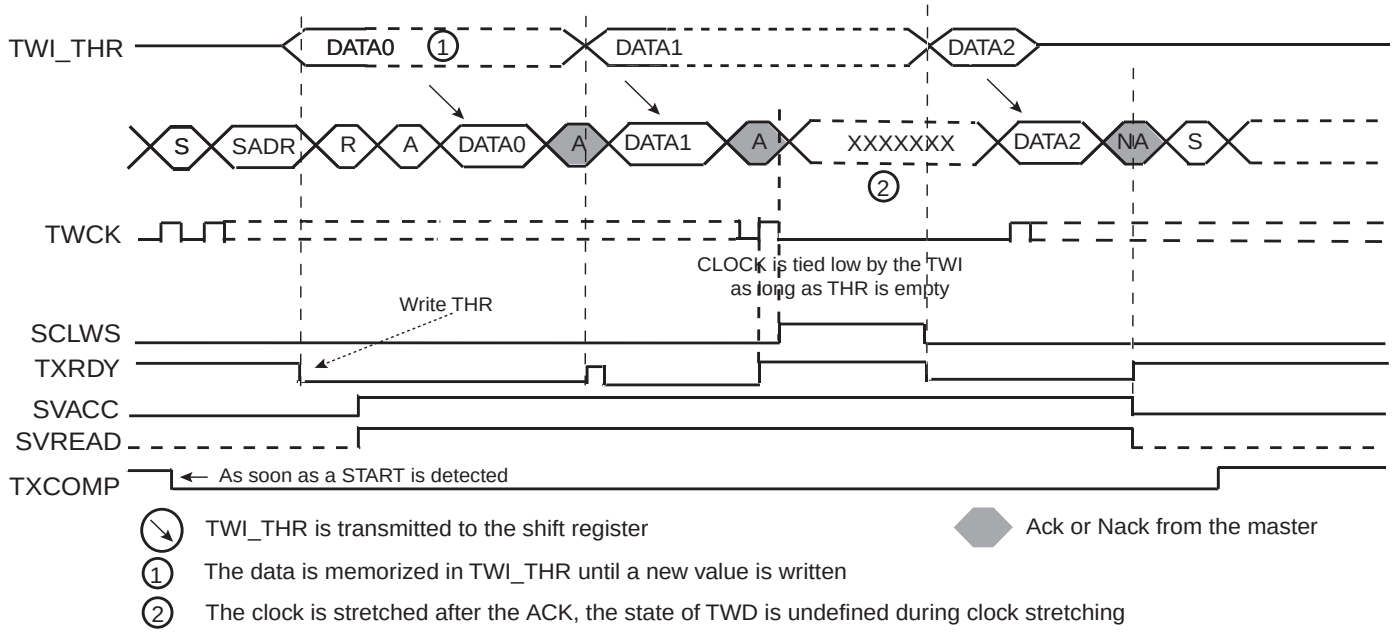
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

23.11.5.5 Clock Synchronization in Read Mode

The clock is tied low if the shift register is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the shift register is loaded.

Figure 23-20 on page 219 describes the clock synchronization in Read mode.

Figure 23-20. Clock Synchronization in Read Mode



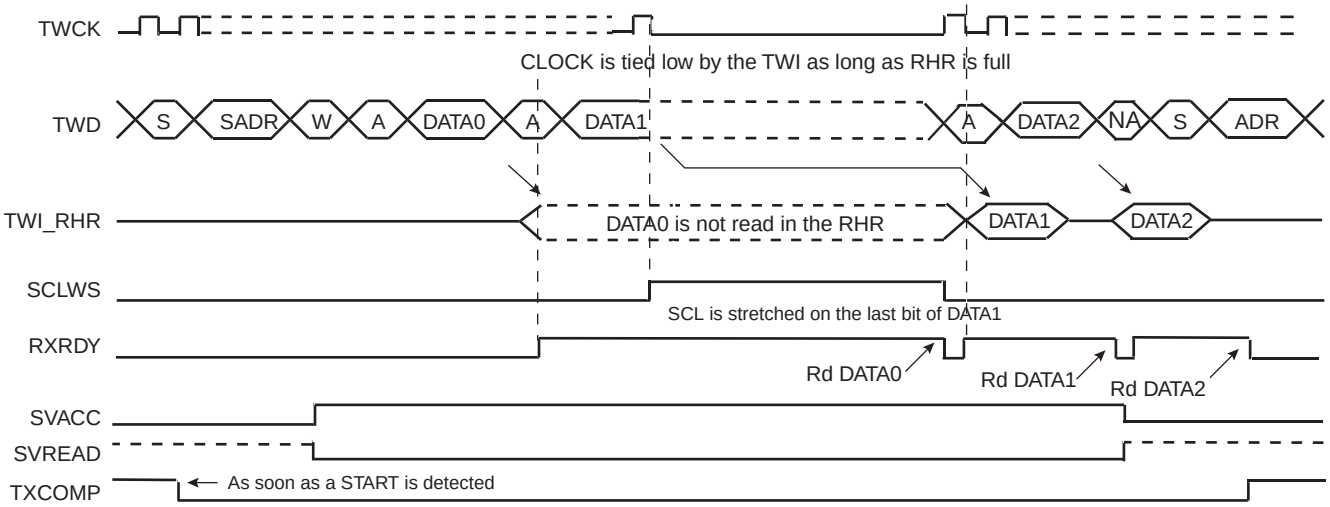
Notes:
1. TXRDY is reset when data has been written in theTWI_TH to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

23.11.5.6 Clock Synchronization in Write Mode

The clock is tied low if the shift register and the TWI_RHR is full. If a STOP or REPEATED_START condition was not detected, it is tied low until TWI_RHR is read.

Figure 23-21 on page 220 describes the clock synchronization in Read mode.

Figure 23-21. Clock Synchronization in Write Mode



Notes: 1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

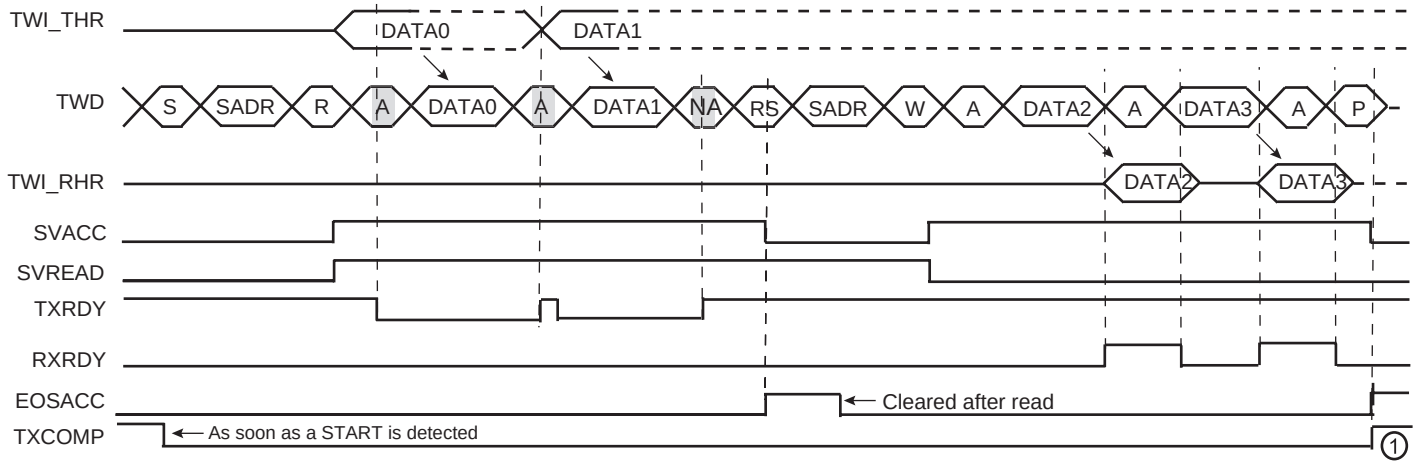
23.11.5.7 Reversal after a Repeated Start

23.11.5.8 Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 23-22 on page 220 describes the repeated start + reversal from Read to Write mode.

Figure 23-22. Repeated Start + Reversal from Read to Write Mode



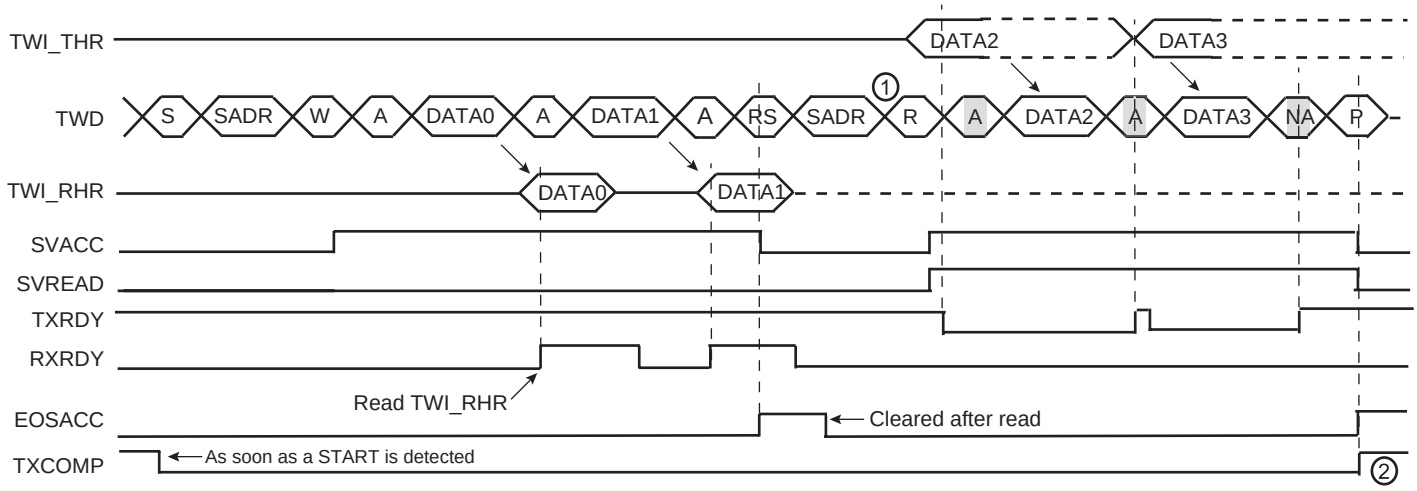
Note: 1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

23.11.5.9 Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.

Figure 23-23 on page 221 describes the repeated start + reversal from Write to Read mode.

Figure 23-23. Repeated Start + Reversal from Write to Read Mode

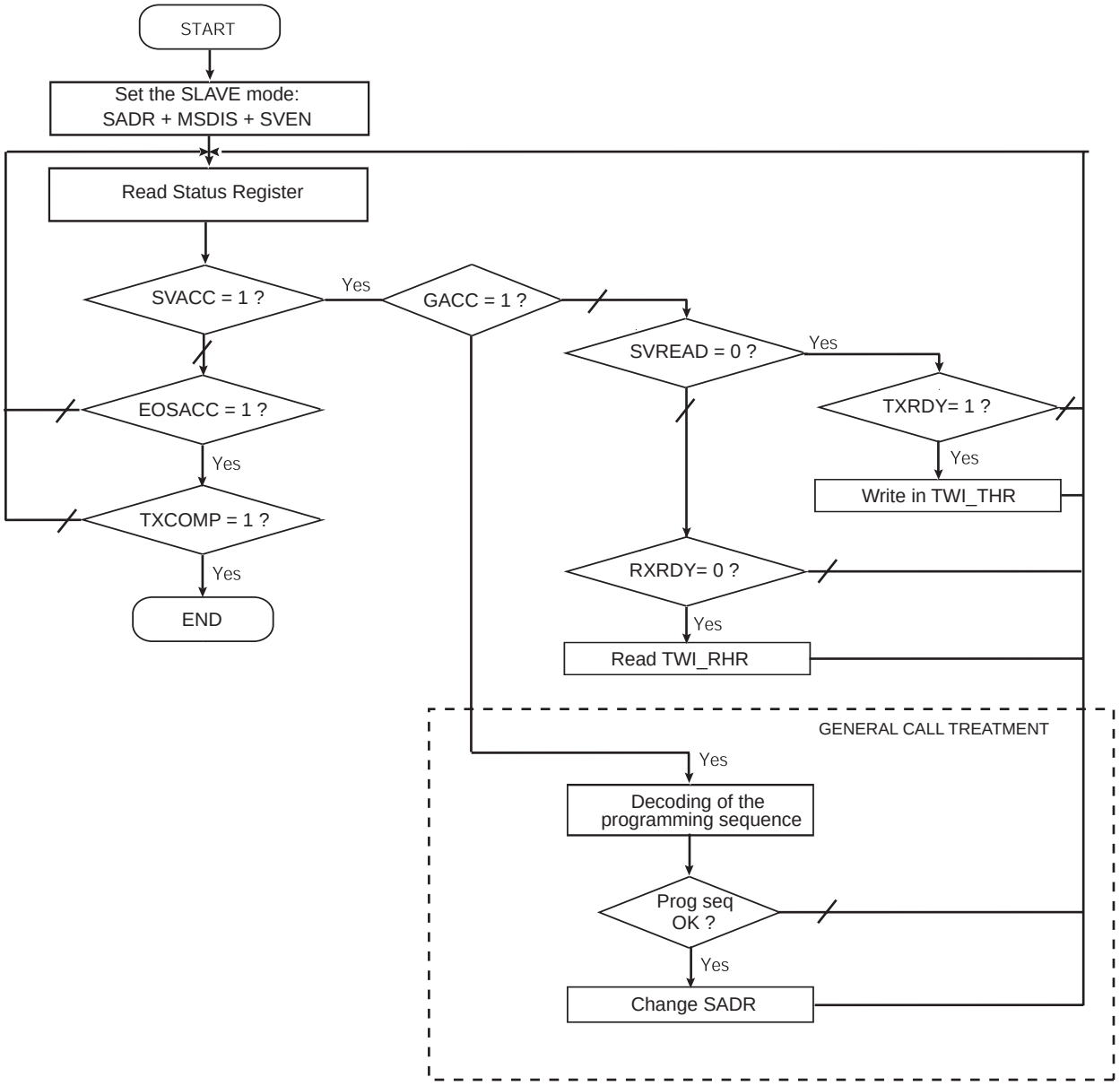


Notes:
1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

23.11.6 Read Write Flowcharts

The flowchart shown in Figure 23-24 on page 222 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 23-24. Read Write Flowchart in Slave Mode



23.12 Two-wire Interface (TWI) User Interface

Table 23-3. Two-wire Interface (TWI) User Interface

Offset	Register	Name	Access	Reset State
0x00	Control Register	TWI_CR	Write-only	N / A
0x04	Master Mode Register	TWI_MMR	Read/Write	0x00000000
0x08	Slave Mode Register	TWI_SMR	Read/Write	0x00000000
0x0C	Internal Address Register	TWI_IADR	Read/Write	0x00000000
0x10	Clock Waveform Generator Register	TWI_CWGR	Read/Write	0x00000000
0x20	Status Register	TWI_SR	Read-only	0x0000F009
0x24	Interrupt Enable Register	TWI_IER	Write-only	N / A
0x28	Interrupt Disable Register	TWI_IDR	Write-only	N / A
0x2C	Interrupt Mask Register	TWI_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWI_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWI_THR	Write-only	0x00000000
0x38 - 0xFC	Reserved	–	–	–

23.12.1 TWI Control Register

Name: TWI_CR

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	SVDIS	SVEN	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent with the mode register as soon as the user writes a character in the holding register.

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read or write mode.

In single data byte master read or write, the START and STOP must both be set.

In multiple data byte master read, the STOP must be set before ACK/NACK bit transmission to be taken into account at the end of the current byte.

In master read mode, if a NACK bit is received, the STOP is automatically performed.

In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

23.12.2 TWI Master Mode Register

Name: TWI_MMR

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	DADR						

15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: Internal Device Address Size**

IADRSZ[9:8]		
0	0	No internal device address
0	1	One-byte internal device address
1	0	Two-byte internal device address
1	1	Three-byte internal device address

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.

23.12.3 TWI Slave Mode Register

Name: TWI_SMR

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	SADR						
15	14	13	12	11	10	9	8
–	–	–	–	–	–		
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

23.12.4 TWI Internal Address Register

Name: TWI_IADR

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

23.12.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
					CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

TWI_CWGR is only used in Master mode.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$$T_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

23.12.6 TWI Status Register

Name: TWI_SR

Access: Read-only

Reset Value: 0x0000F009

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCLWS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed (automatically set / reset)**

TXCOMP used in Master mode:

0 = During the length of the current frame.

1 = When both holding and shifter registers are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 23-7 on page 208 and in Figure 23-9 on page 208.

TXCOMP used in Slave mode:

0 = As soon as a Start is detected.

1 = After a Stop or a Repeated Start + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 23-20 on page 219, Figure 23-21 on page 220, Figure 23-22 on page 220 and Figure 23-23 on page 221.

- **RXRDY: Receive Holding Register Ready (automatically set / reset)**

0 = No character has been received since the last TWI_RHR read operation.

1 = A byte has been received in the TWI_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 23-9 on page 208.

RXRDY behavior in Slave mode can be seen in Figure 23-18 on page 217, Figure 23-21 on page 220, Figure 23-22 on page 220 and Figure 23-23 on page 221.

- **TXRDY: Transmit Holding Register Ready (automatically set / reset)**

TXRDY used in Master mode:

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI_THR register.

1 = As soon as data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

TXRDY behavior in Master mode can be seen in Figure 23-7 on page 208.

TXRDY used in Slave mode:

0 = As soon as a data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1 = It indicates that the TWI_THR is empty and that a data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI_THR to avoid losing it.

TXRDY behavior in Slave mode can be seen in Figure 23-17 on page 217, Figure 23-20 on page 219, Figure 23-22 on page 220 and Figure 23-23 on page 221.

- **SVREAD: Slave Read (automatically set / reset)**

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0 = Indicates that a write access is performed by a Master.

1 = Indicates that a read access is performed by a Master.

SVREAD behavior can be seen in Figure 23-17 on page 217, Figure 23-18 on page 217, Figure 23-22 on page 220 and Figure 23-23 on page 221.

- **SVACC: Slave Access (automatically set / reset)**

This bit is only used in Slave mode.

0 = TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1 = Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

SVACC behavior can be seen in Figure 23-17 on page 217, Figure 23-18 on page 217, Figure 23-22 on page 220 and Figure 23-23 on page 221.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, the programmer decoded the commands that follow and the programming sequence.

GACC behavior can be seen in Figure 23-19 on page 218.

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0 = TWI_RHR has not been loaded while RXRDY was set

1 = TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

NACK used in Master mode:

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0 = Each data byte has been correctly received by the Master.

1 = In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data are acknowledged by the TWI.

- **ARBLST: Arbitration Lost (clear on read)**

This bit is only used in Master mode.

0: Arbitration won.

1: Arbitration lost. Another master of the TWI bus has won the multi-master arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State (automatically set / reset)**

This bit is only used in Slave mode.

0 = The clock is not stretched.

1 = The clock is stretched. TWI_THR / TWI_RHR buffer is not filled / emptied before the emission / reception of a new character.

SCLWS behavior can be seen in Figure 23-20 on page 219 and Figure 23-21 on page 220.

- **EOSACC: End Of Slave Access (clear on read)**

This bit is only used in Slave mode.

0 = A slave access is being performing.

1 = The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

EOSACC behavior can be seen in Figure 23-22 on page 220 and Figure 23-23 on page 221

- **ENDRX: End of RX buffer**

This bit is only used in Master mode.

0 = The Receive Counter Register has not reached 0 since the last write in TWI_RCR or TWI_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in TWI_RCR or TWI_RNCR.

- **ENDTX: End of TX buffer**

This bit is only used in Master mode.

0 = The Transmit Counter Register has not reached 0 since the last write in TWI_TCR or TWI_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in TWI_TCR or TWI_TNCR.

- **RXBUFF: RX Buffer Full**

This bit is only used in Master mode.

0 = TWI_RCR or TWI_RNCR have a value other than 0.

1 = Both TWI_RCR and TWI_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

This bit is only used in Master mode.

0 = TWI_TCR or TWI_TNCR have a value other than 0.

1 = Both TWI_TCR and TWI_TNCR have a value of 0.

23.12.7 TWI Interrupt Enable Register

Name: TWI_IER

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ABRLST: Arbitration Lost Interrupt Enable**
- **SCL_WS: Clock Wait State Interrupt Enable**

0 = No effect.

1 = Disables the corresponding interrupt.

23.12.8 TWI Interrupt Disable Register

Name: TWI_IDR

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Disable**
- **RXRDY: Receive Holding Register Ready Interrupt Disable**
- **TXRDY: Transmit Holding Register Ready Interrupt Disable**
- **SVACC: Slave Access Interrupt Disable**
- **GACC: General Call Access Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **NACK: Not Acknowledge Interrupt Disable**
- **ABRLST: Arbitration Lost Interrupt Disable**
- **SCL_WS: Clock Wait State Interrupt Disable**
- **EOSACC: End Of Slave Access Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

23.12.9 TWI Interrupt Mask Register

Name: TWI_IMR

Access: Read-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ABRLST: Arbitration Lost Interrupt Mask**
- **SCL_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

23.12.10 TWI Receive Holding Register

Name: TWI_RHR

Access: Read-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: Master or Slave Receive Holding Data**

23.12.11 TWI Transmit Holding Register

Name: TWI_THR

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: Master or Slave Transmit Holding Data**

24. Synchronous Serial Controller (SSC)

Rev: 3.0.1.0

24.1 Features

- **Provides Serial Synchronous Communication Links Used in Audio and Telecom Applications**
- **Contains an Independent Receiver and Transmitter and a Common Clock Divider**
- **Interfaced with Two PDC Channels (DMA Access) to Reduce Processor Overhead**
- **Offers a Configurable Frame Sync and Data Length**
- **Receiver and Transmitter Can be Programmed to Start Automatically or on Detection of Different Events on the Frame Sync Signal**
- **Receiver and Transmitter Include a Data Signal, a Clock Signal and a Frame Synchronization Signal**

24.2 Description

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

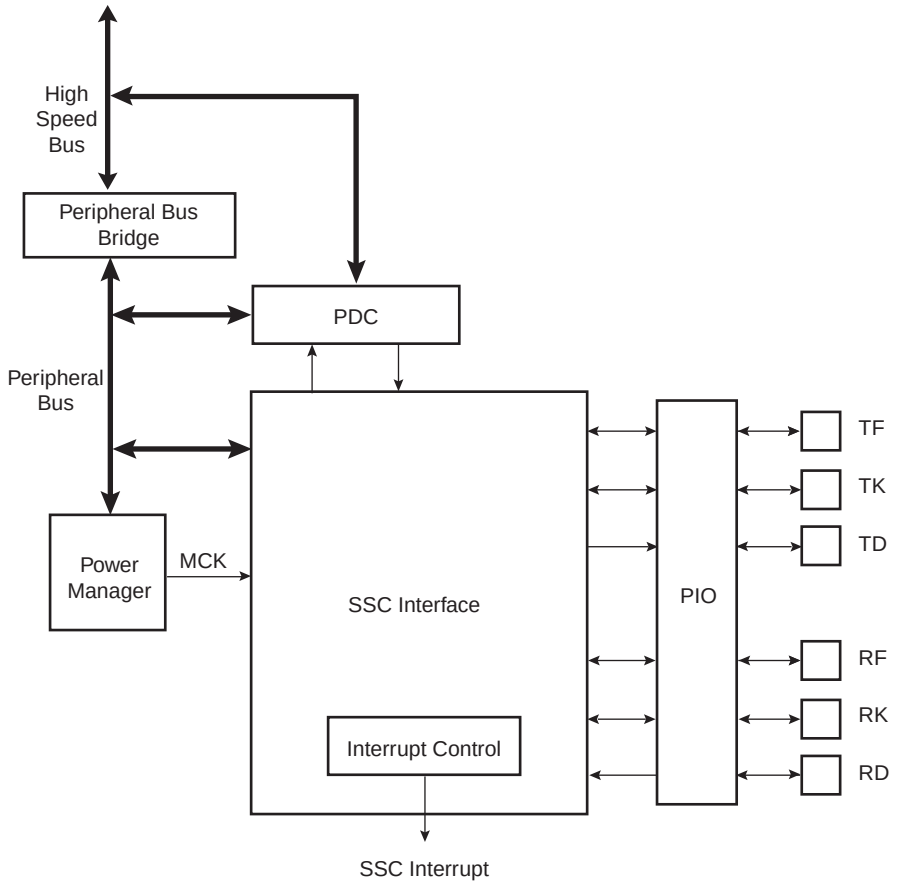
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

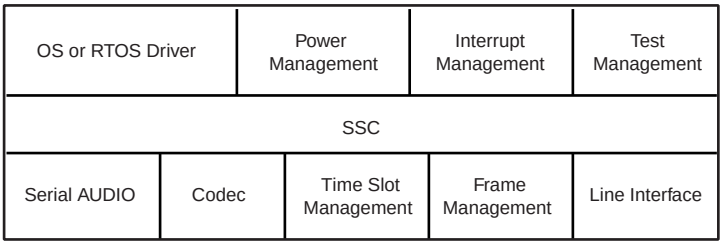
24.3 Block Diagram

Figure 24-1. Block Diagram



24.4 Application Block Diagram

Figure 24-2. Application Block Diagram



24.5 Pin Name List

Table 24-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

24.6 Product Dependencies

24.6.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

24.6.2 Power Management

The SSC clock is generated by the power manager. Before using the SSC, the programmer must ensure that the SSC clock is enabled in the power manager.

In the SSC description, Master Clock (MCK) is the bus clock of the peripheral bus to which the SSC is connected.

24.6.3 Interrupt

The SSC interface has an interrupt line connected to the interrupt controller. Handling interrupts requires programming the interrupt controller before configuring the SSC.

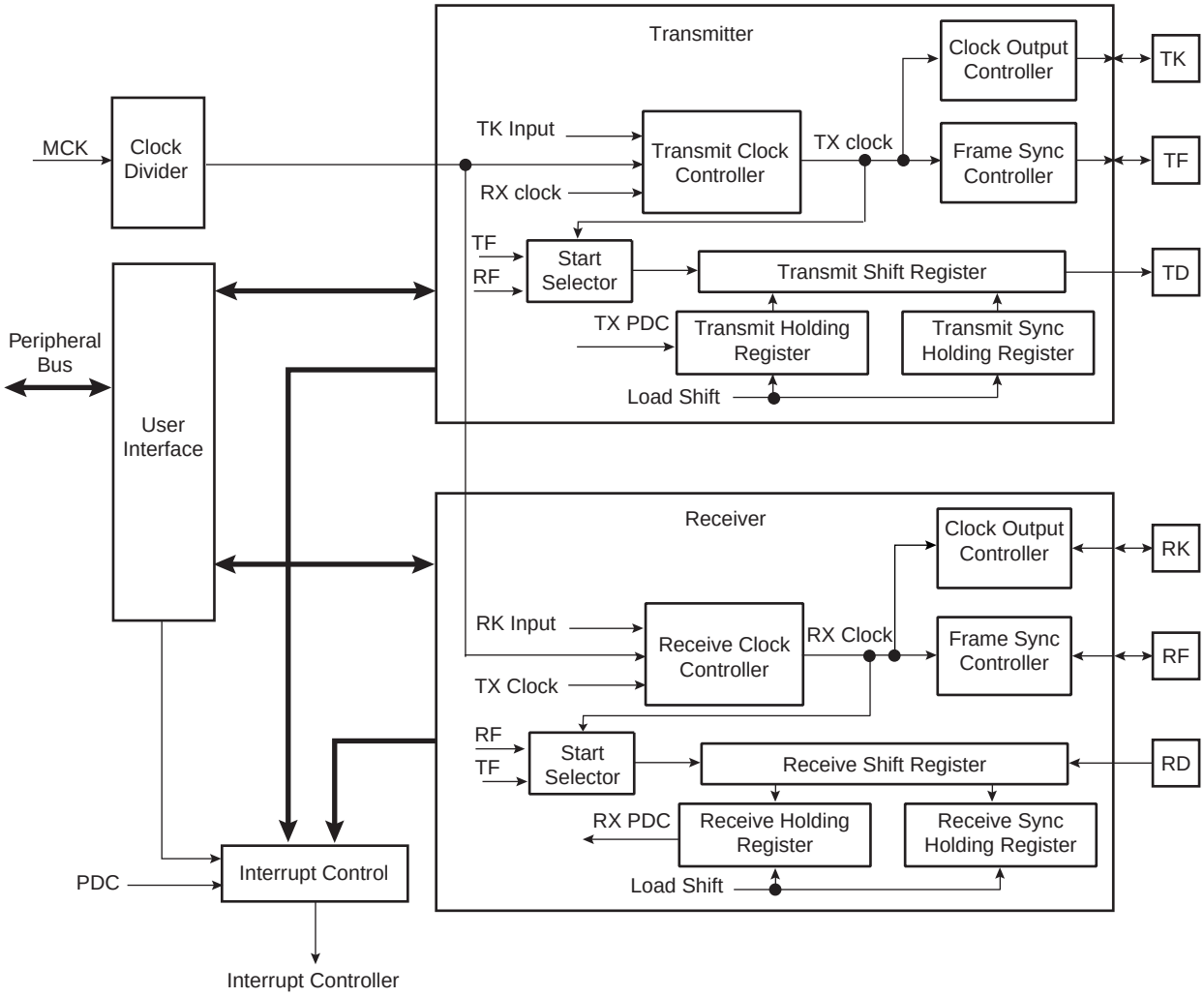
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

24.7 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

Figure 24-3. SSC Functional Block Diagram



24.7.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

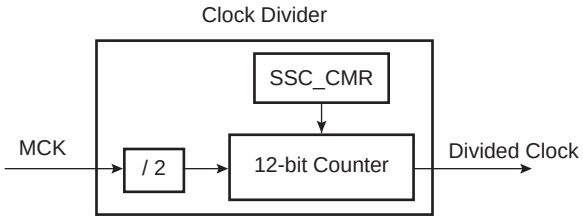
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

24.7.1.1 Clock Divider

Figure 24-4. Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register CMR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

Figure 24-5. Divided Clock Generation

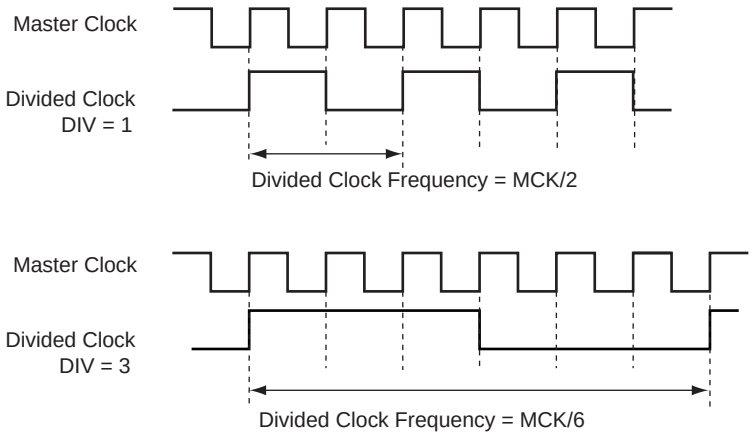


Table 24-2.

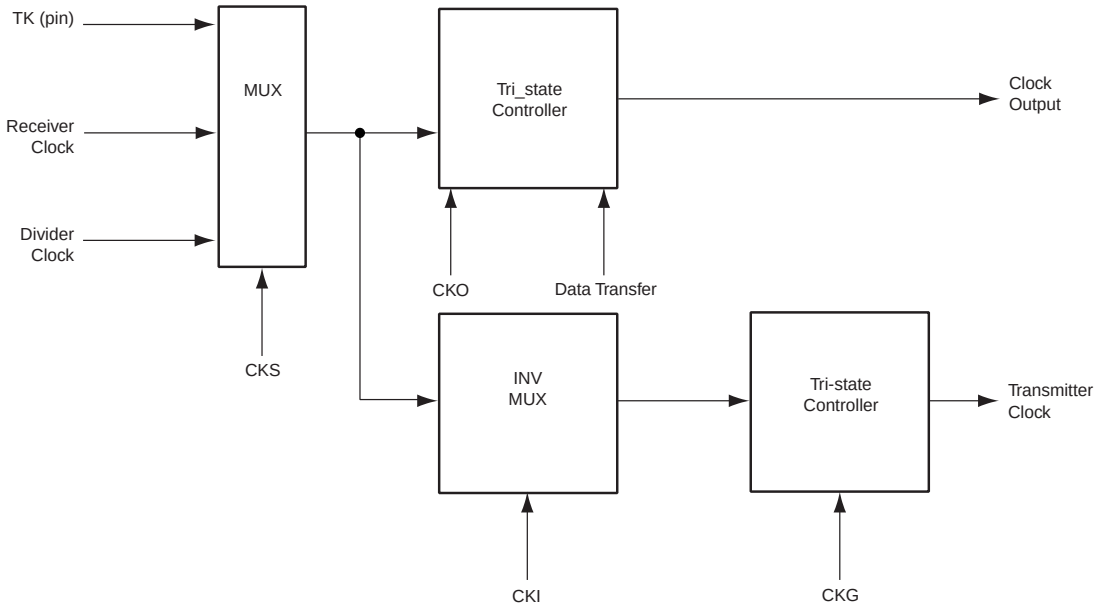
Maximum	Minimum
MCK / 2	MCK / 8190

24.7.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

Figure 24-6. Transmitter Clock Management

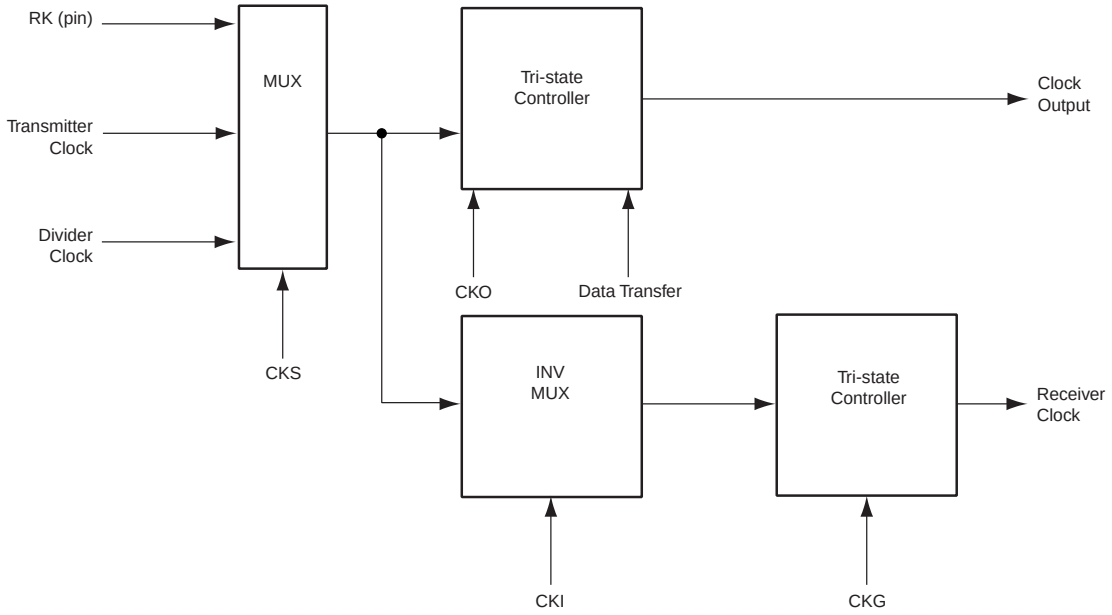


24.7.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 24-7. Receiver Clock Management



24.7.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Master Clock divided by 2 if Receiver Frame Synchro is input
- Master Clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Master Clock divided by 6 if Transmit Frame Synchro is input
- Master Clock divided by 2 if Transmit Frame Synchro is output

24.7.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

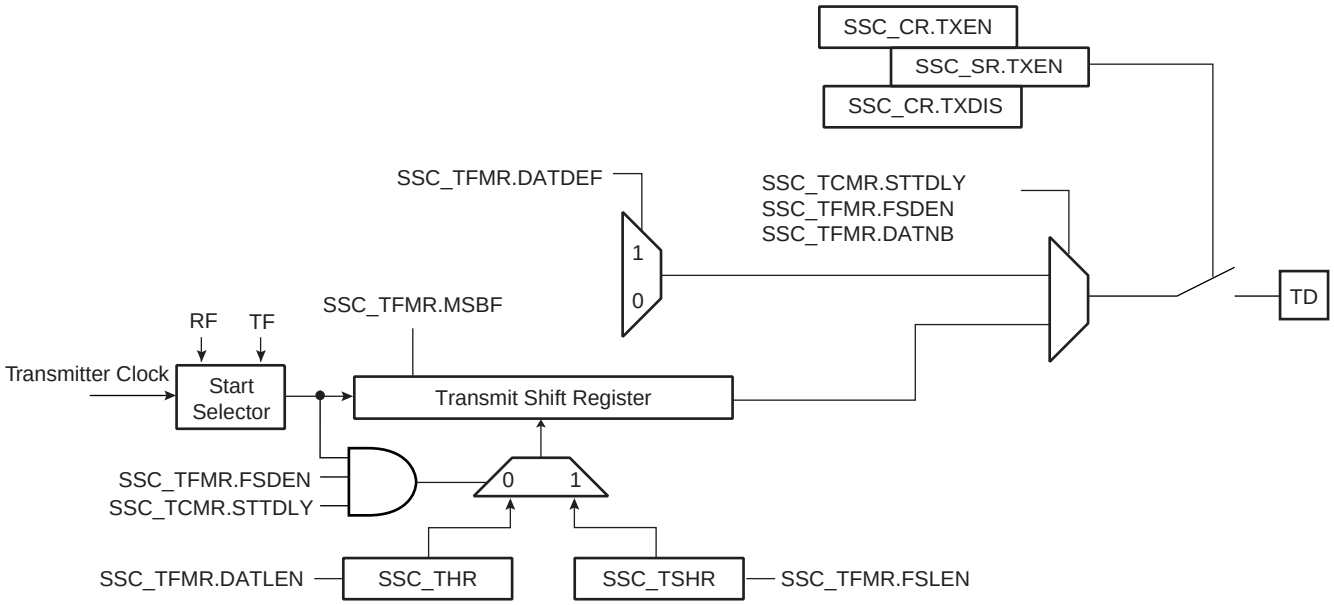
The start event is configured by setting the Transmit Clock Mode Register (TCMR). See Section “24.7.4” on page 246.

The frame synchronization is configured setting the Transmit Frame Mode Register (TFMR). See Section “24.7.5” on page 248.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the TCMR. Data is written by the application to the THR register then transferred to the shift register according to the data format selected.

When both the THR and the transmit shift register are empty, the status flag TXEMPTY is set in SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SR and additional data can be loaded in the holding register.

Figure 24-8. Transmitter Block Diagram



24.7.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

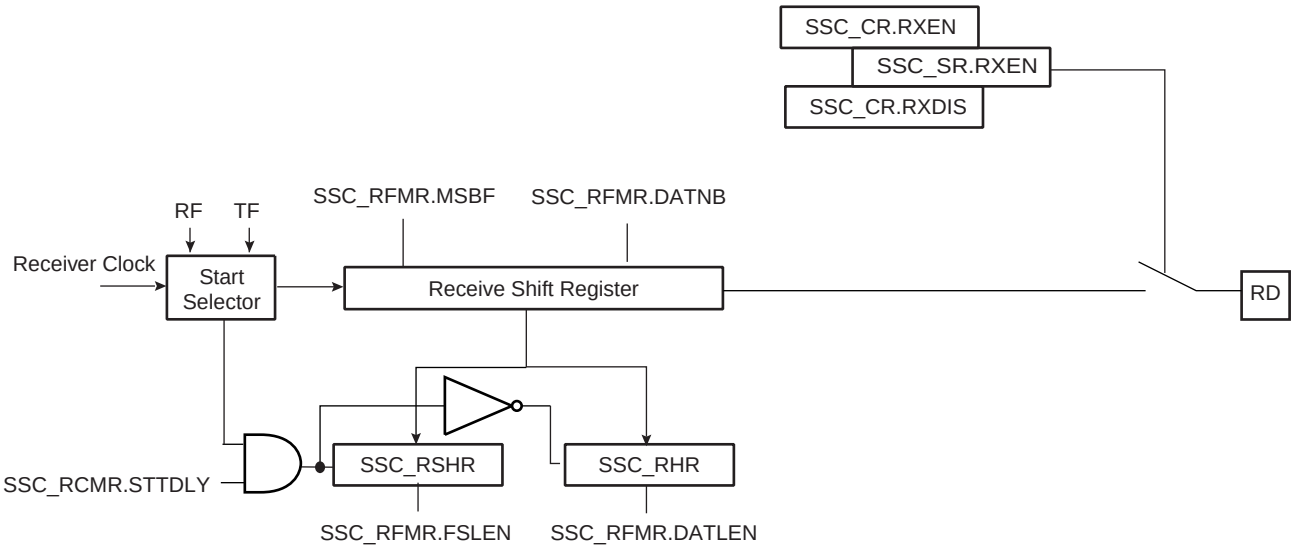
The start event is configured setting the Receive Clock Mode Register (RCMR). See Section “24.7.4” on page 246.

The frame synchronization is configured setting the Receive Frame Mode Register (RFMR). See Section “24.7.5” on page 248.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SR and the receiver shift register is transferred in the RHR register.

Figure 24-9. Receiver Block Diagram



24.7.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of TCMR and in the Receive Start Selection (START) field of RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

Figure 24-10. Transmit Start Mode

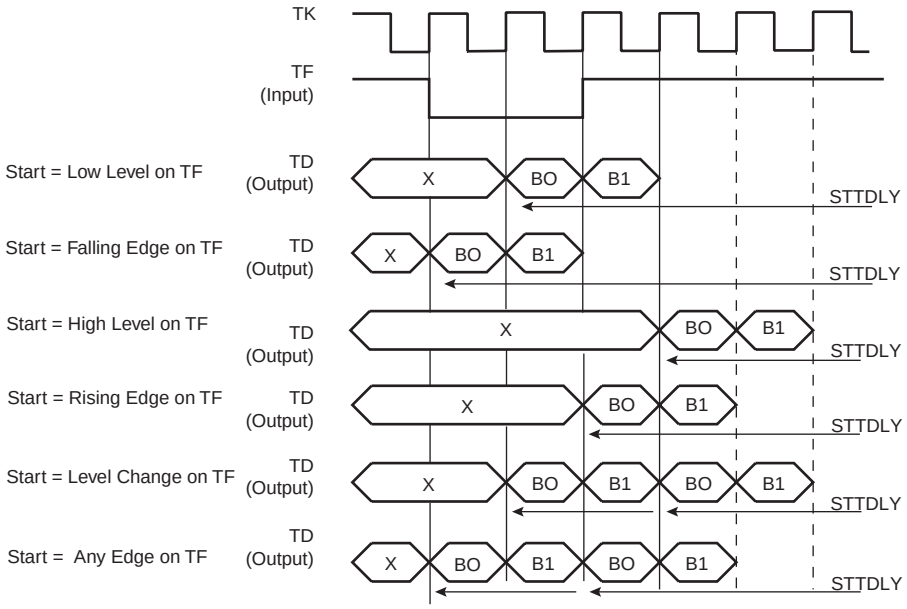
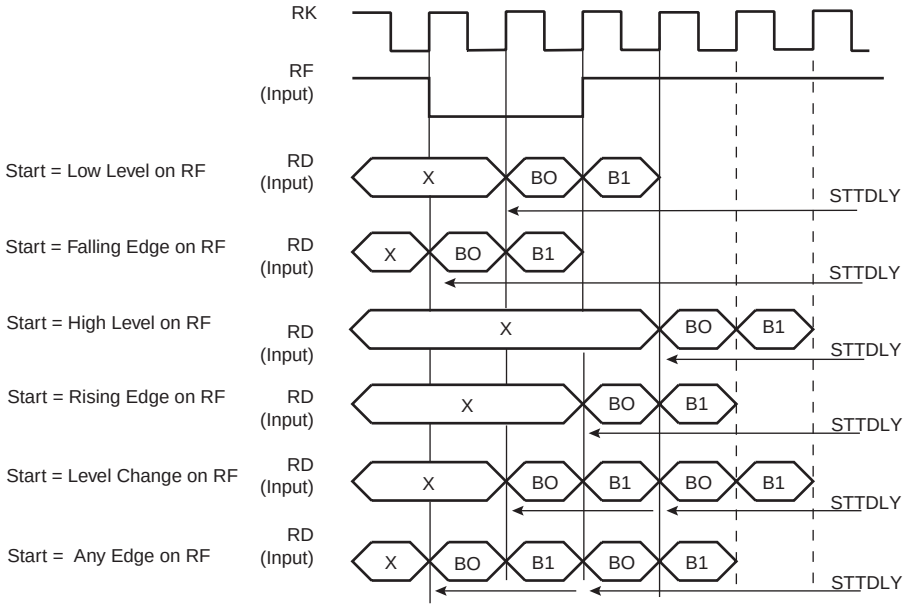


Figure 24-11. Receive Pulse/Edge Start Modes



24.7.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (RFMR) and in the Transmit Frame Mode Register (TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in RFMR and TFMR programs the length of the pulse, from 1 bit time up to 16 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in RCMR and TCMR.

24.7.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in RFMR/TFMR.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

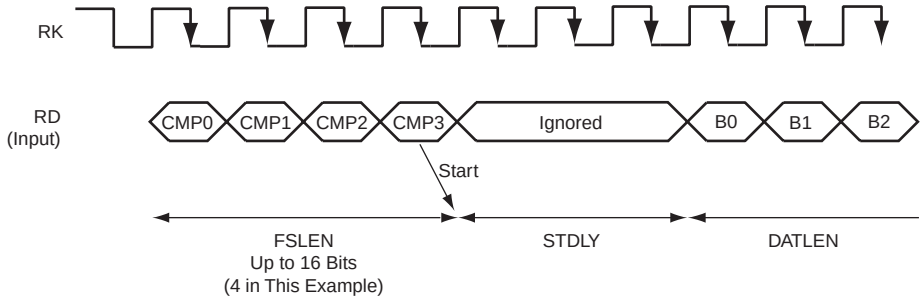
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

24.7.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in RFMR/TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SR) on frame synchro edge detection (signals RF/TF).

24.7.6 Receive Compare Modes

Figure 24-12. Receive Compare Modes



24.7.6.1 Compare Functions

Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last FSLEN bits received at the FSLEN lower bit of the data contained in the Compare 0 Register (RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in RCMR.

24.7.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (TFMR) and the Receiver Frame Mode Register (RFMR). In either case, the user can independently select:

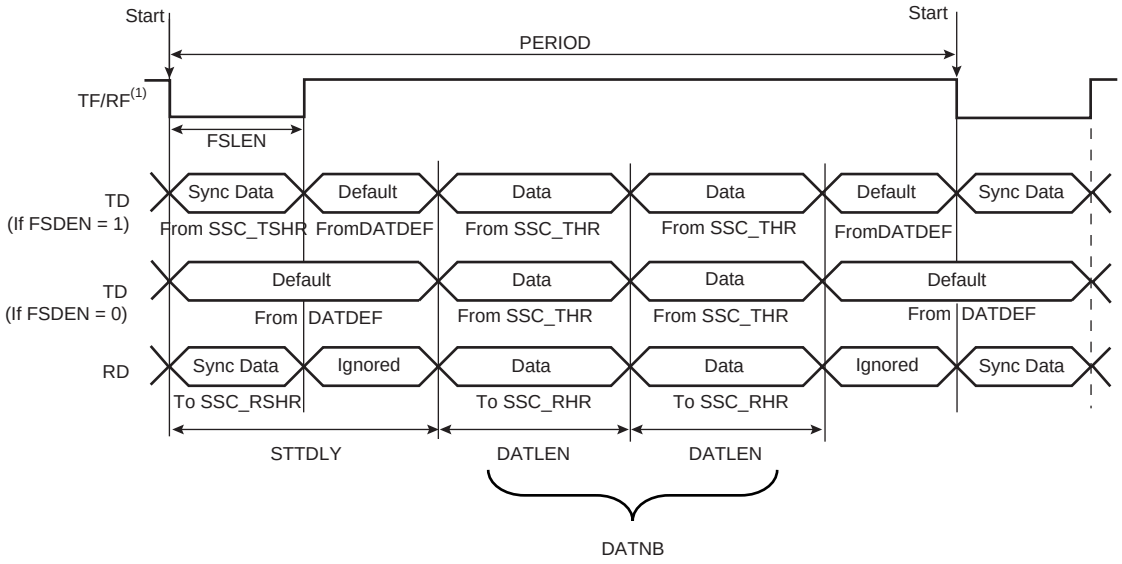
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF).

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in TFMR.

Table 24-3. Data Frame Registers

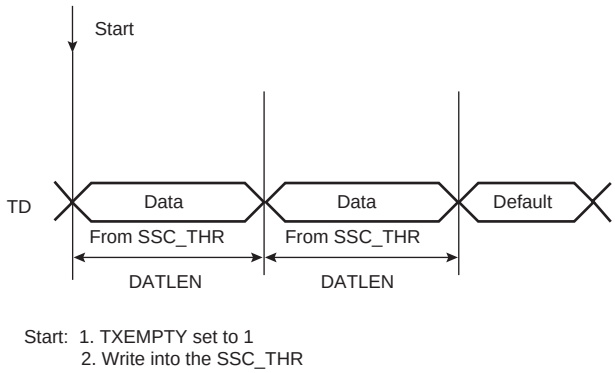
Transmitter	Receiver	Field	Length	Comment
TFMR	RFMR	DATLEN	Up to 32	Size of word
TFMR	RFMR	DATNB	Up to 16	Number of words transmitted in frame
TFMR	RFMR	MSBF		Most significant bit first
TFMR	RFMR	FSLEN	Up to 16	Size of Synchro data register
TFMR		DATDEF	0 or 1	Data default value ended
TFMR		FSDEN		Enable send TSHR
TCMR	RCMR	PERIOD	Up to 512	Frame size
TCMR	RCMR	STTDLY	Up to 255	Size of transmit start delay

Figure 24-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



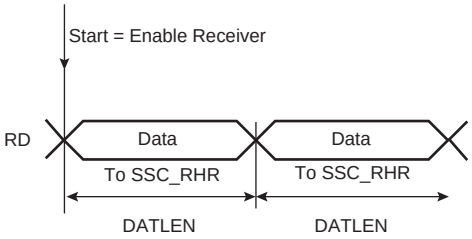
Note: 1. Example of input on falling edge of TF/RF.

Figure 24-14. Transmit Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0. In this example, THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 24-15. Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

24.7.8 Loop Mode

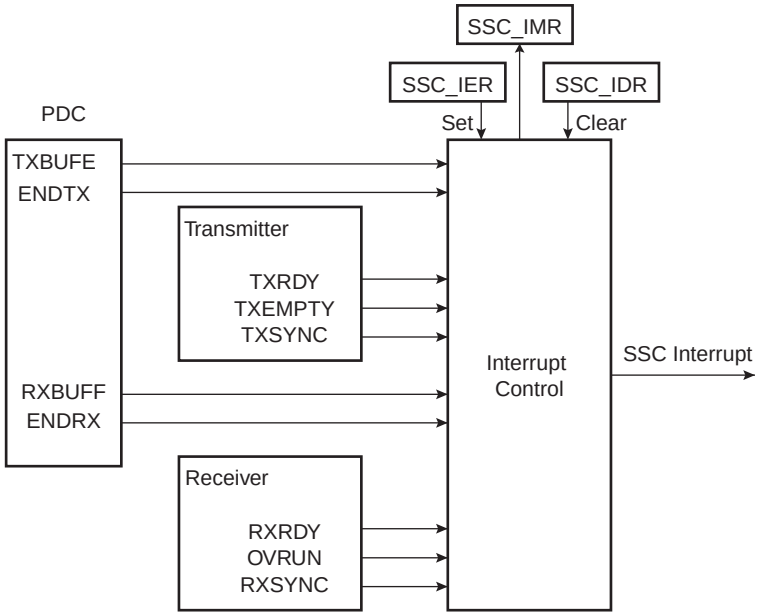
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

24.7.9 Interrupt

Most bits in SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing IER (Interrupt Enable Register) and IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the interrupt controller.

Figure 24-16. Interrupt Block Diagram



24.8 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 24-17. Audio Application Block Diagram

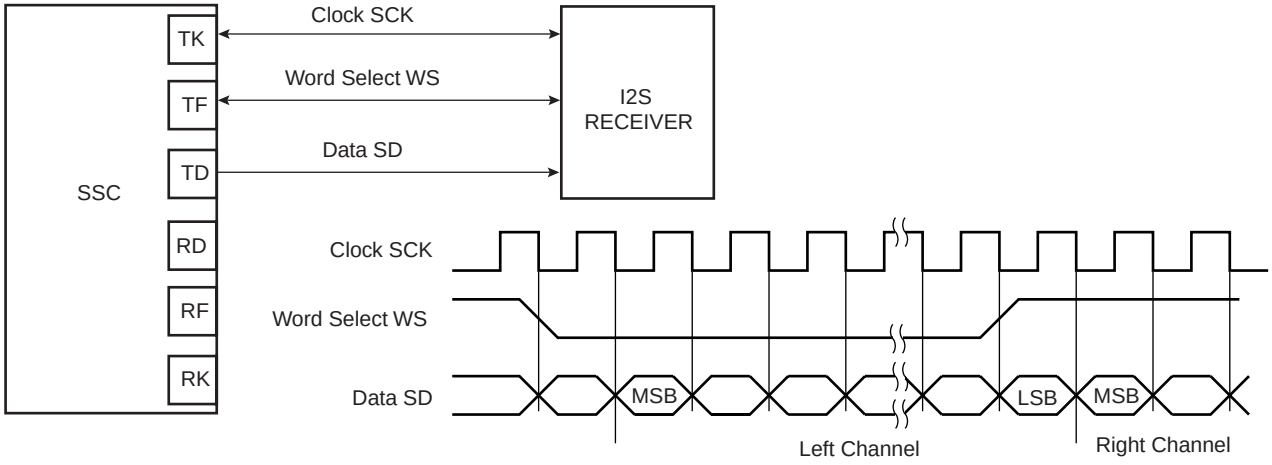


Figure 24-18. Codec Application Block Diagram

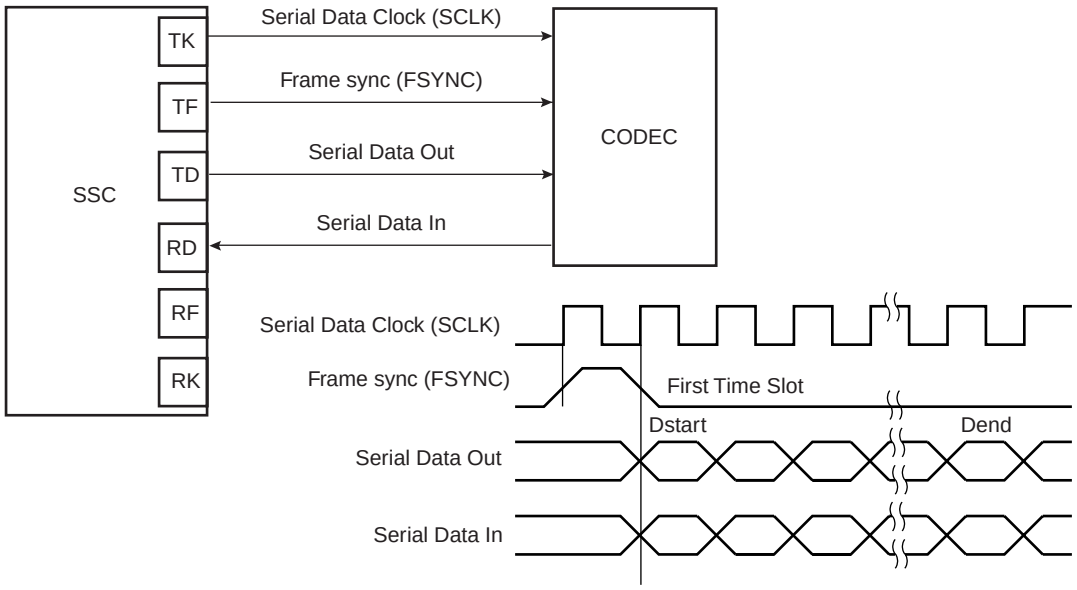
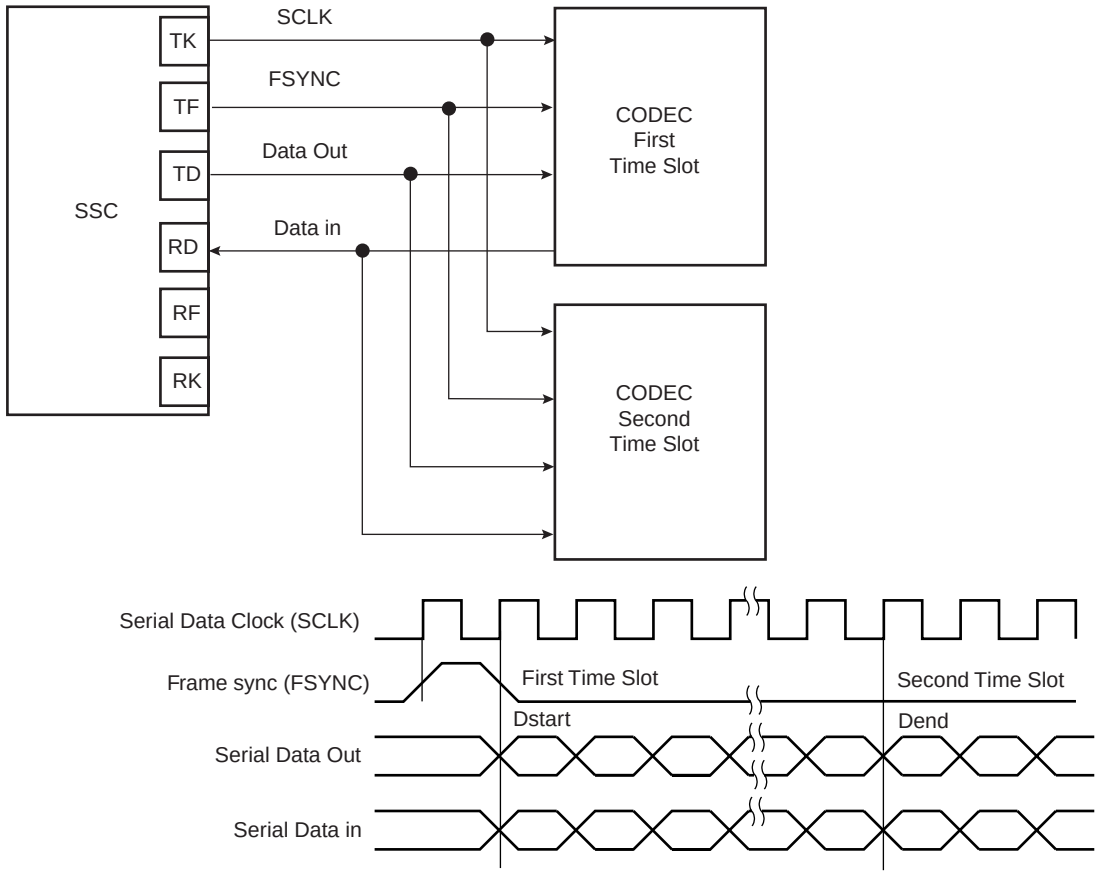


Figure 24-19. Time Slot Application Block Diagram



24.9 Synchronous Serial Controller (SSC) User Interface

Table 24-4. Register Mapping

Offset	Register	Register Name	Access	Reset
0x0	Control Register	CR	Write	–
0x4	Clock Mode Register	CMR	Read/Write	0x0
0x8	Reserved	–	–	–
0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	TFMR	Read/Write	0x0
0x20	Receive Holding Register	RHR	Read	0x0
0x24	Transmit Holding Register	THR	Write	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	RSHR	Read	0x0
0x34	Transmit Sync. Holding Register	TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	RC1R	Read/Write	0x0
0x40	Status Register	SR	Read	0x000000CC
0x44	Interrupt Enable Register	IER	Write	–
0x48	Interrupt Disable Register	IDR	Write	–
0x4C	Interrupt Mask Register	IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–

24.9.1 SSC Control Register

Name: CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SWRST	–	–	–	–	–	TXDIS	TXEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXDIS	RXEN

- **RXEN: Receive Enable**

0: No effect.

1: Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in CR.

24.9.2 SSC Clock Mode Register

Name: CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

- **DIV: Clock Divider**

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is MCK/2 x 4095 = MCK/8190.

24.9.3 SSC Receive Clock Mode Register

Name: RCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **CKS: Receive Clock Selection**

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

- **CKO: Receive Clock Output Mode Selection**

CKO	Receive Clock Output Mode	RK pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Receive Clock Inversion**

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

- **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

- **START: Receive Start Selection**

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9-0xF	Reserved

- **STOP: Receive Stop Selection**

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.

24.9.4 SSC Receive Frame Mode Register

Name: RFMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
FSLENHI				–	–	–	FSEDGE

23	22	21	20	19	18	17	16
–	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).
Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the length of the Receive Frame Sync Signal and the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register. Note: The four most significant bits fo this bitfield are in the FSLENHI bitfield.

Pulse length is equal to ({FSLENHI,FSLEN} + 1) Receive Clock periods. Thus, if {FSLENHI,FSLEN} is 0, the Receive Frame Sync signal is generated during one Receive Clock period.

- **FSOS: Receive Frame Sync Output Selection**

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

- **FSLENHI: Receive Frame Sync Length High part**

The four MSB of the FSLEN bitfield.

24.9.5 SSC Transmit Clock Mode Register

Name: TCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	–	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **CKS: Transmit Clock Selection**

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

- **CKO: Transmit Clock Output Mode Selection**

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Transmit Clock Inversion**

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

- **START: Transmit Start Selection**

START	Transmit Start
0x0	Continuous, as soon as a word is written in the THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8 - 0xF	Reserved

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 x (PERIOD+1) Transmit Clock.

24.9.6 SSC Transmit Frame Mode Register

Name: TFMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
FSLENHI				–	–	–	FSEDGE

23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			

15	14	13	12	11	10	9	8
–	–	–	–	DATNB			

7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Transmit. If DATLEN is lower or equal to 7, data transfers are bytes, if DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB +1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1. Note: The four most significant bits fo this bitfield are in the FSLENHI bitfield.

Pulse length is equal to ({FSLENHI,FSLEN} + 1) Transmit Clock periods, i.e., the pulse length can range from 1 to 16 Transmit Clock periods. If {FSLENHI,FSLEN} is 0, the Transmit Frame Sync signal is generated during one Transmit Clock period.

- **FSOS: Transmit Frame Sync Output Selection**

FSOS	Selected Transmit Frame Sync Signal	TF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

- **FSLENHI: Transmit Frame Sync Length High part**

The four MSB of the FSLEN bitfield.

24.9.7 SSC Receive Holding Register

Name: RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
RDAT							
23	22	21	20	19	18	17	16
RDAT							
15	14	13	12	11	10	9	8
RDAT							
7	6	5	4	3	2	1	0
RDAT							

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in RFMR.

24.9.8 SSC Transmit Holding Register

Name: THR

Access Type: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in TFMR.

24.9.9 SSC Receive Synchronization Holding Register

Name: RSHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

24.9.10 SSC Transmit Synchronization Holding Register

Name: TSHR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

24.9.11 SSC Receive Compare 0 Register

Name: RC0R

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

- **CP0: Receive Compare Data 0**

24.9.12 SSC Receive Compare 1 Register

Name: RC1R

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CP1							

7	6	5	4	3	2	1	0
CP1							

- **CP1: Receive Compare Data 1**

24.9.13 SSC Status Register

Name: SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0: Data has been loaded in THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1: THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in THR or is currently transmitted from TSR.

1: Last data written in THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **ENDTX: End of Transmission**

0: The register TCR has not reached 0 since the last write in TCR or TNCR.

1: The register TCR has reached 0 since the last write in TCR or TNCR.

- **TXBUFE: Transmit Buffer Empty**

0: TCR or TNCR have a value other than 0.

1: Both TCR and TNCR have a value of 0.

- **RXRDY: Receive Ready**

0: RHR is empty.

1: Data has been received and loaded in RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0: RCR or RNCR have a value other than 0.

1: Both RCR and RNCR have a value of 0.

- **CP0: Compare 0**

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit is disabled.

1: Transmit is enabled.

- **RXEN: Receive Enable**

0: Receive is disabled.

1: Receive is enabled.

24.9.14 SSC Interrupt Enable Register

Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0: No effect.

1: Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Enable**

0: No effect.

1: Enables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Buffer Empty Interrupt

- **RXRDY: Receive Ready Interrupt Enable**

0: No effect.

1: Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0: No effect.

1: Enables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Enable**

0: No effect.

1: Enables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0: No effect.

1: Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.

24.9.15 SSC Interrupt Disable Register

Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0: No effect.

1: Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Disable**

0: No effect.

1: Disables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Buffer Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0: No effect.

1: Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0: No effect.

1: Disables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Disable**

0: No effect.

1: Disables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0: No effect.

1: Disables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0: No effect.

1: Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0: No effect.

1: Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Disables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Disables the Rx Sync Interrupt.

24.9.16 SSC Interrupt Mask Register

Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

- **ENDTX: End of Transmission Interrupt Mask**

0: The End of Transmission Interrupt is disabled.

1: The End of Transmission Interrupt is enabled.

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0: The Transmit Buffer Empty Interrupt is disabled.

1: The Transmit Buffer Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

- **ENDRX: End of Reception Interrupt Mask**

0: The End of Reception Interrupt is disabled.

1: The End of Reception Interrupt is enabled.

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0: The Receive Buffer Full Interrupt is disabled.

1: The Receive Buffer Full Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

- **TXSYN: Tx Sync Interrupt Mask**

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

- **RXSYN: Rx Sync Interrupt Mask**

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

25. Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

Rev: 3.1.9.0

25.1 Features

- **Programmable Baud Rate Generator**
- **5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications**
  - **1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode**
  - **Parity Generation and Error Detection**
  - **Framing Error Detection, Overrun Error Detection**
  - **MSB- or LSB-first**
  - **Optional Break Generation and Detection**
  - **By 8 or by 16 Over-sampling Receiver Frequency**
  - **Optional Hardware Handshaking RTS-CTS**
  - **Optional Modem Signal Management DTR-DSR-DCD-RI**
  - **Receiver Time-out and Transmitter Timeguard**
  - **Optional Multidrop Mode with Address Generation and Detection**
- **RS485 with Driver Control Signal**
- **ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards**
  - **NACK Handling, Error Counter with Repetition and Iteration Limit**
- **IrDA Modulation and Demodulation**
  - **Communication at up to 115.2 Kbps**
- **Test Modes**
  - **Remote Loopback, Local Loopback, Automatic Echo**
- **Supports Connection of Two Peripheral DMA Controller Channels (PDC)**
  - **Offers Buffer Transfer without Processor Intervention**

25.2 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

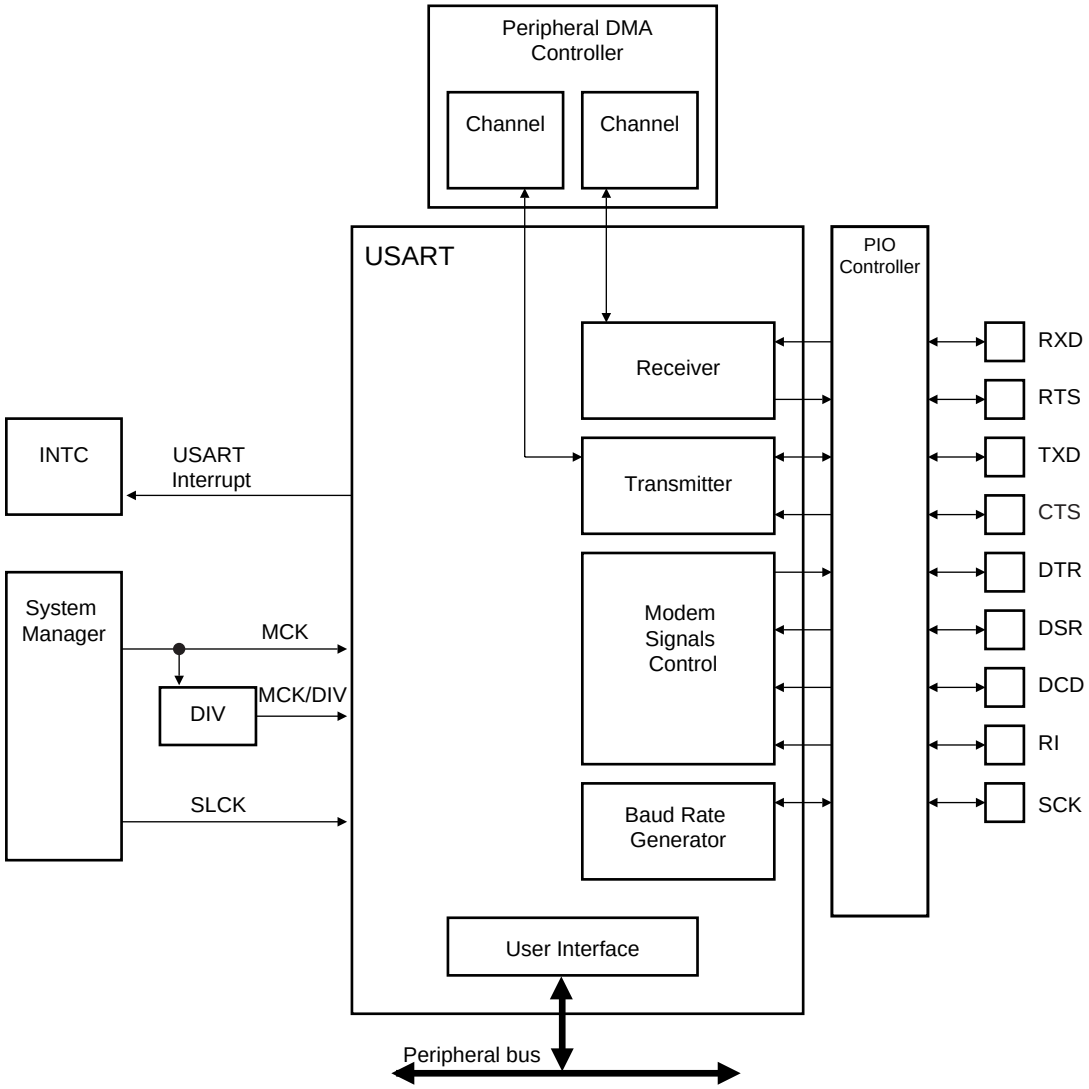
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

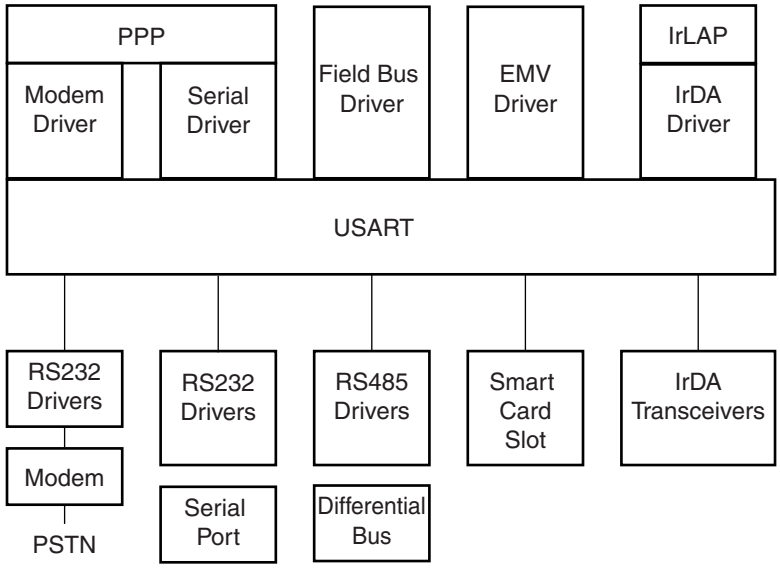
25.3 Block Diagram

Figure 25-1. USART Block Diagram



25.4 Application Block Diagram

Figure 25-2. Application Block Diagram



25.5 I/O Lines Description

Table 25-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data	I/O	
RXD	Receive Serial Data	Input	
RI	Ring Indicator	Input	Low
DSR	Data Set Ready	Input	Low
DCD	Data Carrier Detect	Input	Low
DTR	Data Terminal Ready	Output	Low
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

25.6 Product Dependencies

25.6.1 I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory.

All the pins of the modems may or may not be implemented on the USART within a product. Frequently, only the USART1 is fully equipped with all the modem signals. For the other USARTs of the product not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

25.6.2 Power Management

The USART is not continuously clocked. The programmer must ensure that the USART clock is enabled in the Power Manager (PM) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off. Master Clock (MCK) in the USART description is the clock for the peripheral bus to which the USART is connected.

25.6.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Interrupt Controller. Using the USART interrupt requires the interrupt controller to be programmed first.

25.7 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - Remote loopback, local loopback, automatic echo

25.7.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

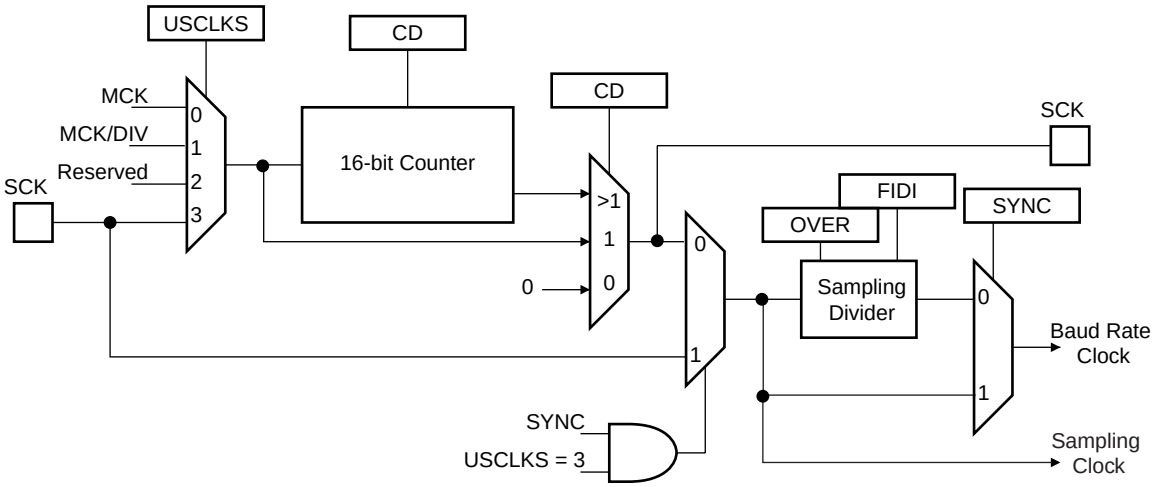
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

Figure 25-3. Baud Rate Generator



25.7.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

25.7.1.2 *Baud Rate Calculation Example*

Table 25-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 25-2. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%

Table 25-2. Baud Rate Example (OVER = 0) (Continued)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

The baud rate is calculated with the following formula:

$$BaudRate = MCK/CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

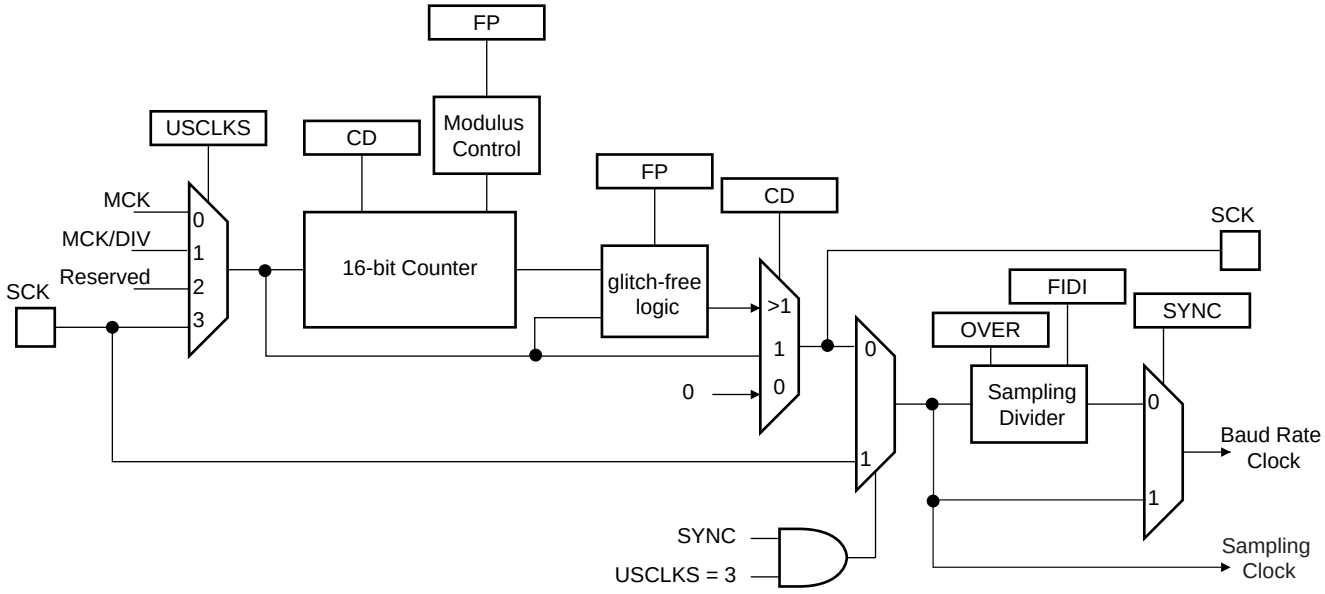
25.7.1.3 Fractional Baud Rate in Asynchronous Mode

The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register (BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2 - Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

Figure 25-4. Fractional Baud Rate Generator



25.7.1.4 *Baud Rate in Synchronous Mode*

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

25.7.1.5 *Baud Rate in ISO 7816 Mode*

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 25-3.

Table 25-3. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 25-4.

Table 25-4. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 25-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 25-5. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

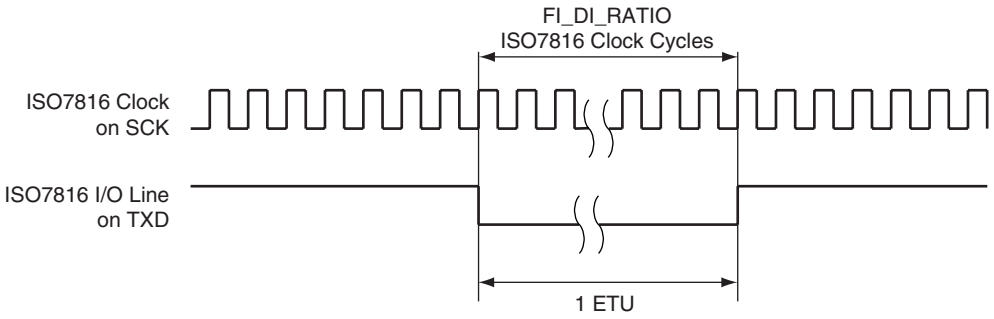
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 25-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 25-5. Elementary Time Unit (ETU)



25.7.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (CR). The reset commands have the same effect as a hardware reset on the corresponding logic. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (THR). If a time-guard is programmed, it is handled normally.

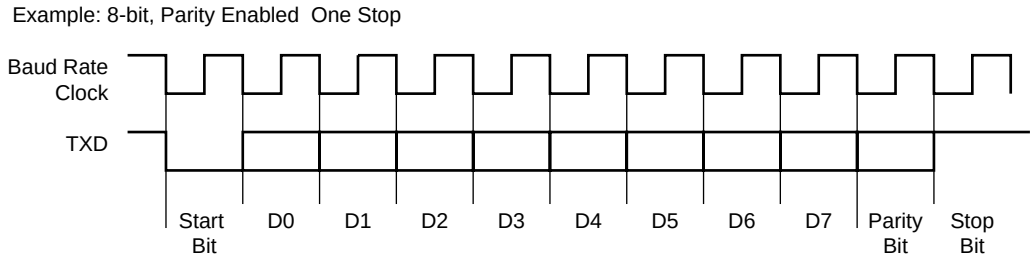
25.7.3 Synchronous and Asynchronous Modes

25.7.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in MR. The 1.5 stop bit is supported in asynchronous mode only.

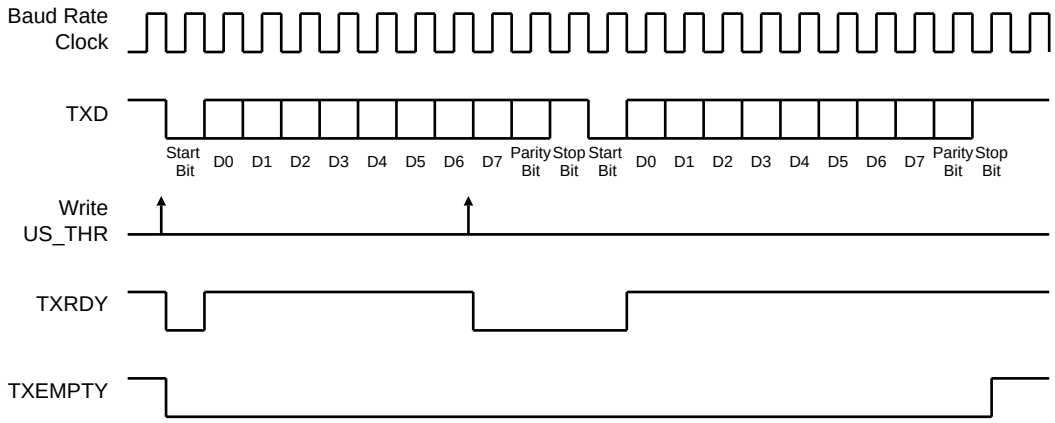
Figure 25-6. Character Transmit



The characters are sent by writing in the Transmit Holding Register (THR). The transmitter reports two status bits in the Channel Status Register (CSR): TXRDY (Transmitter Ready), which indicates that THR is empty and TXEMPTY, which indicates that all the characters written in THR have been processed. When the current character processing is completed, the last character written in THR is transferred into the Shift Register of the transmitter and THR becomes empty, thus TXRDY raises.

Both TXRDY and TXEMPTY bits are low since the transmitter is disabled. Writing a character in THR while TXRDY is active has no effect and the written character is lost.

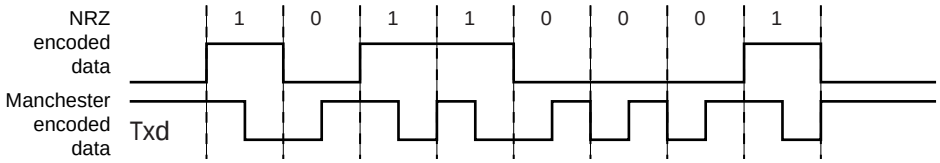
Figure 25-7. Transmitter Status



25.7.3.2 Manchester Encoder

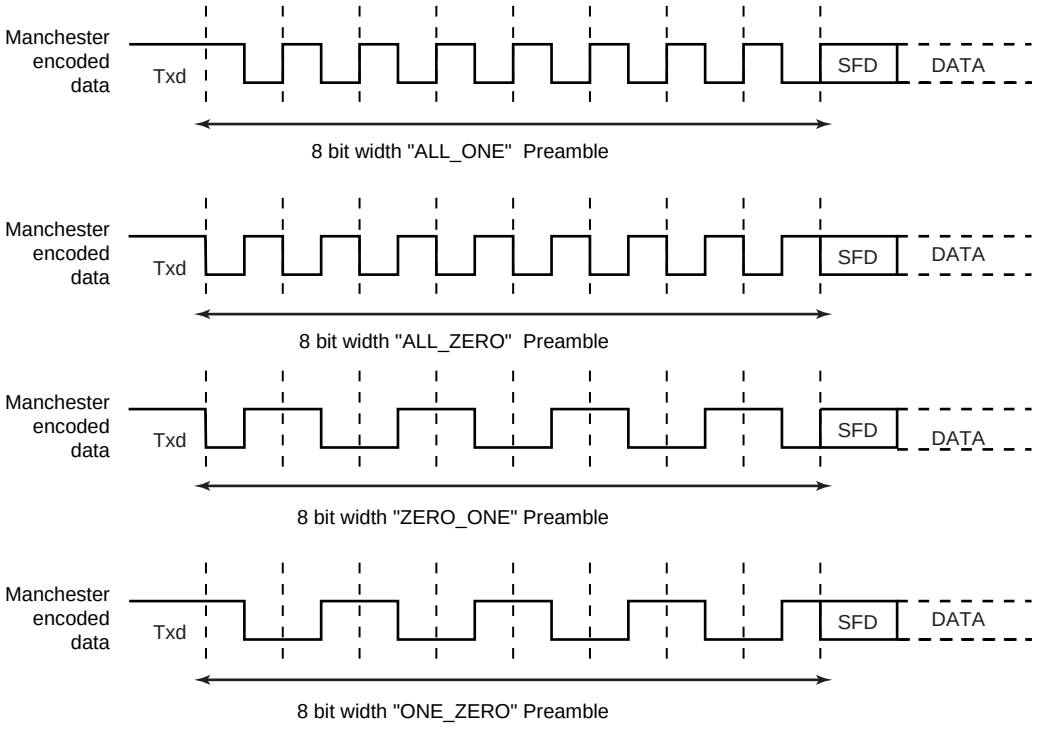
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 25-8 illustrates this coding scheme.

Figure 25-8. NRZ to Manchester Encoding



The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the MAN register, the field TX_PL is used to configure the preamble length. Figure 25-9 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

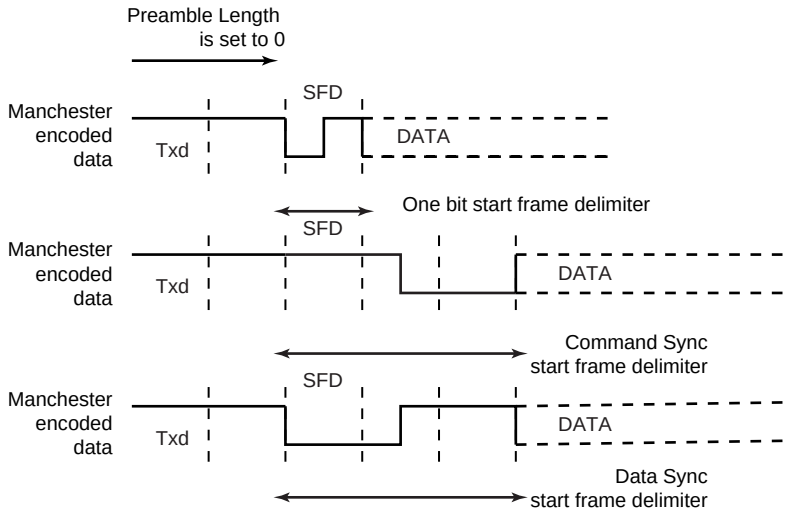
Figure 25-9. Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT field in the MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 25-10 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new character.

The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in MR register must be set to 1. In this case, the MODSYNC field in MR is bypassed and the sync configuration is held in the TXSYNH in the THR register. The USART character format is modified and includes sync information.

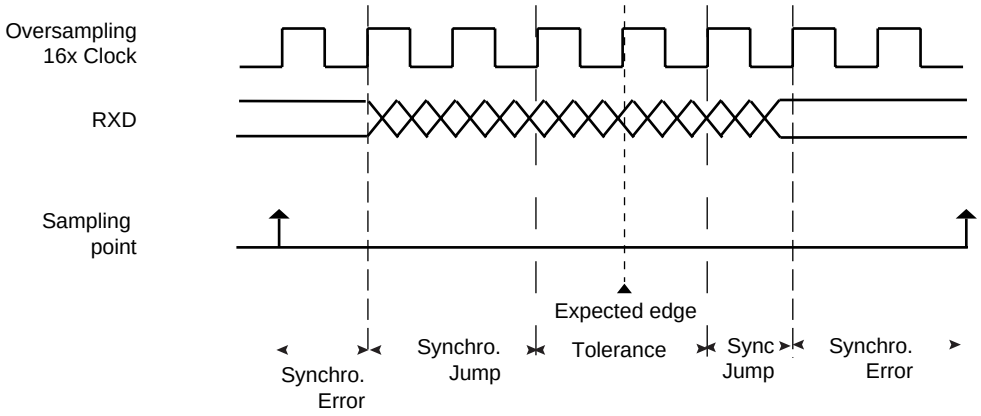
Figure 25-10. Start Frame Delimiter



25.7.3.3 Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 25-11. Bit Resynchronization



25.7.3.4 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. The number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 25-12 and Figure 25-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 25-12. Asynchronous Start Detection

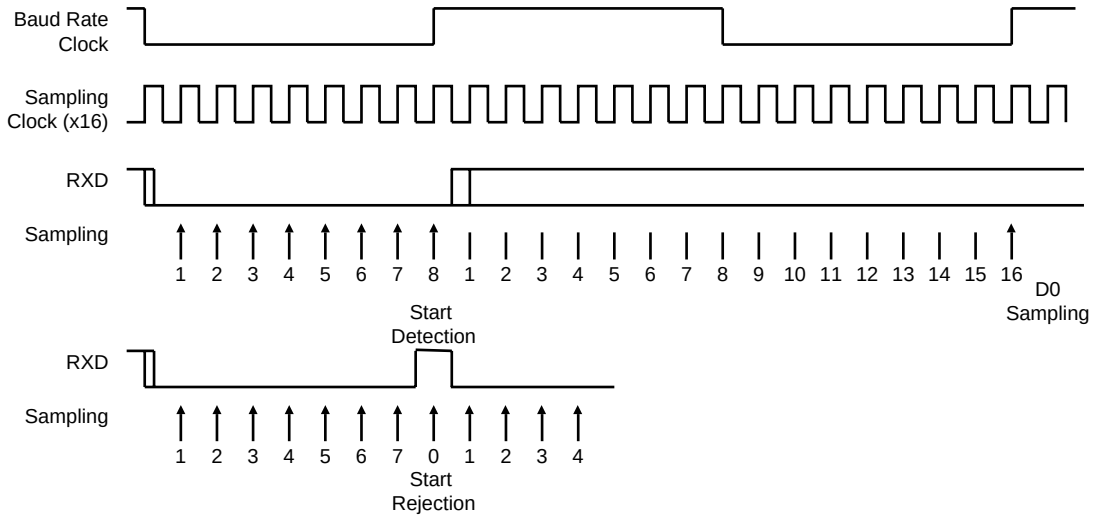
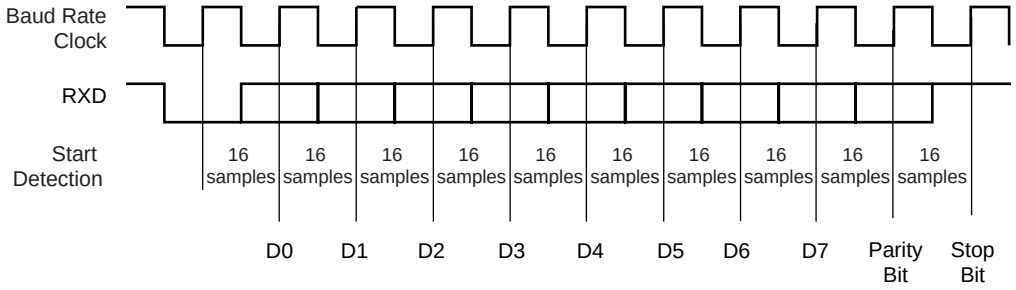


Figure 25-13. Asynchronous Character Reception

Example: 8-bit, Parity Enabled



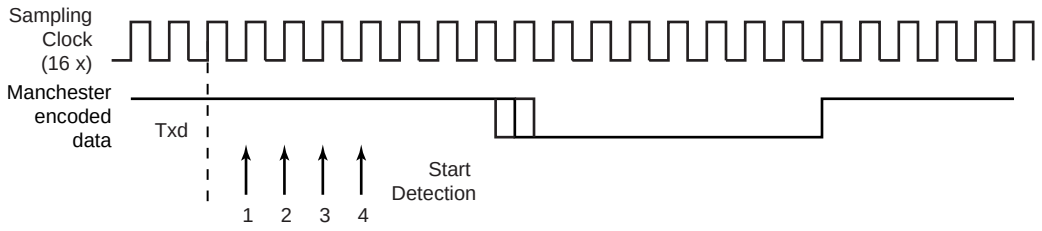
25.7.3.5 Manchester Decoder

When the MAN field in MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in MAN. See Figure 25-9 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 25-14. The sample pulse rejection mechanism applies.

Figure 25-14. Asynchronous Start Bit Detection



The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 25-15 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in CSR register is raised. It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1. See Figure 25-16 for an example of Manchester error detection during data phase.

Figure 25-15. Preamble Pattern Mismatch

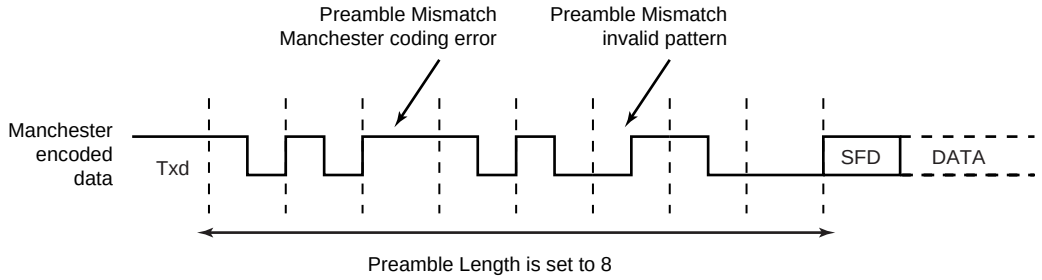
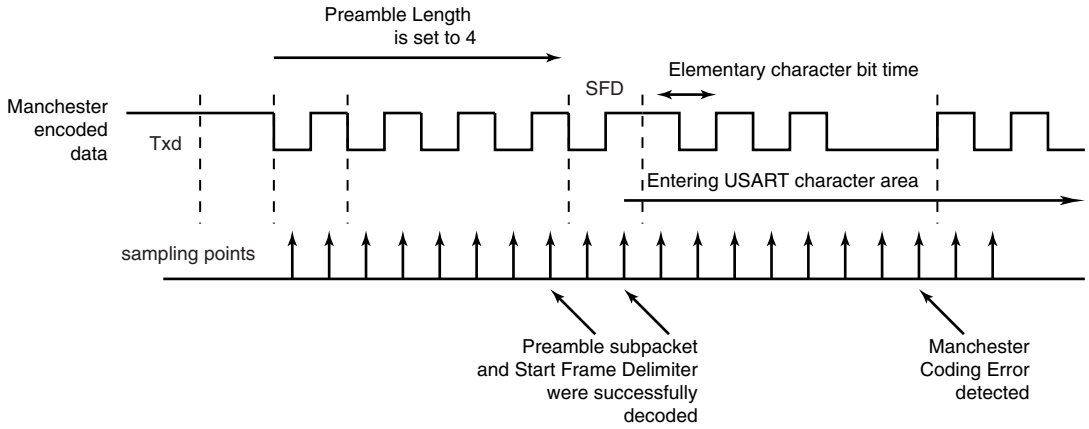


Figure 25-16. Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR

field in the RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

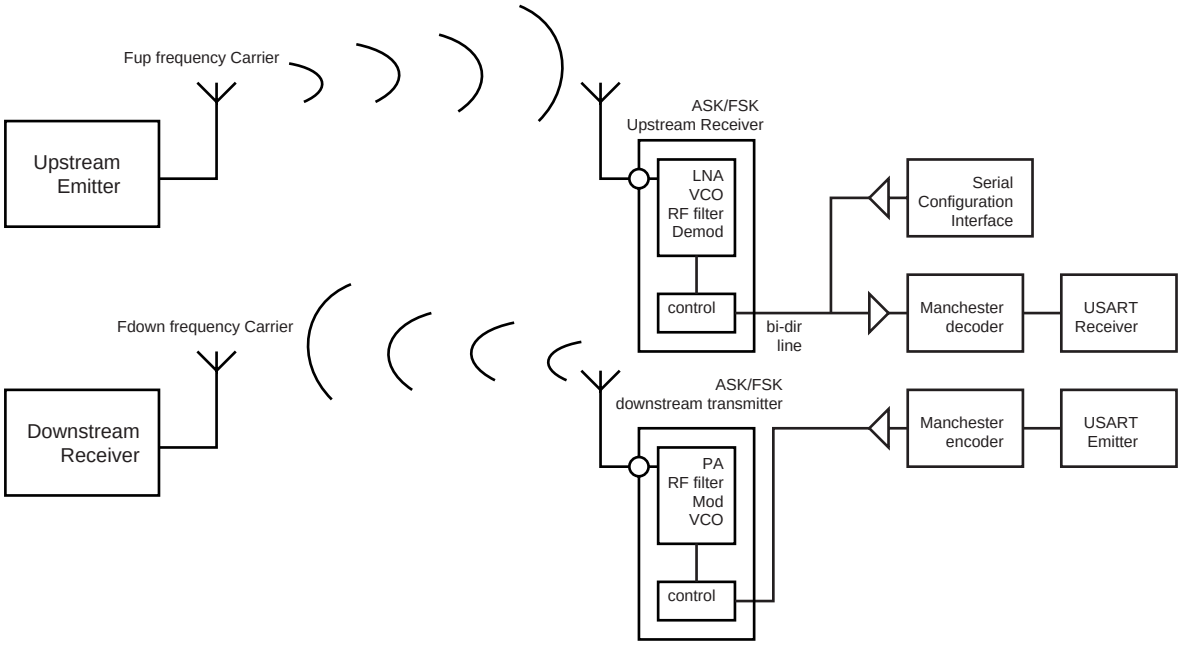
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

25.7.3.6 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 25-17.

Figure 25-17. Manchester Encoded Characters RF Transmission



The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 25-18 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 25-19.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a

user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 25-18. ASK Modulator Output

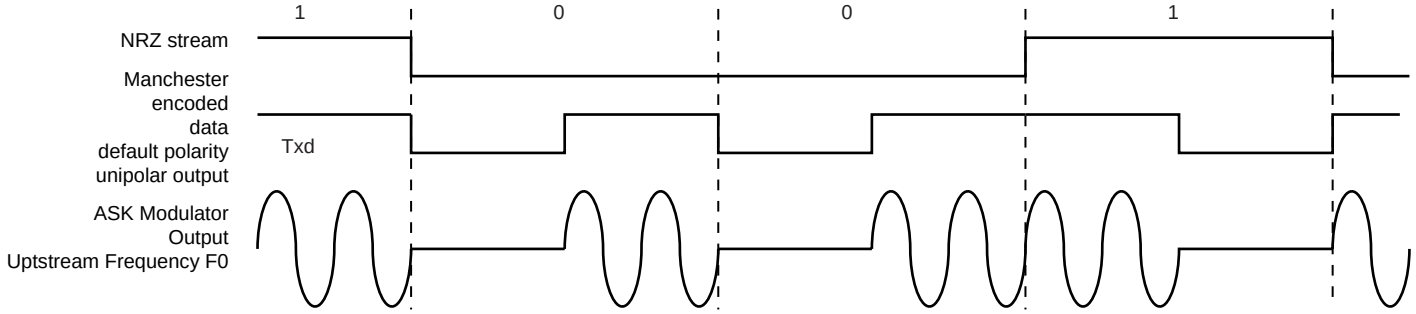
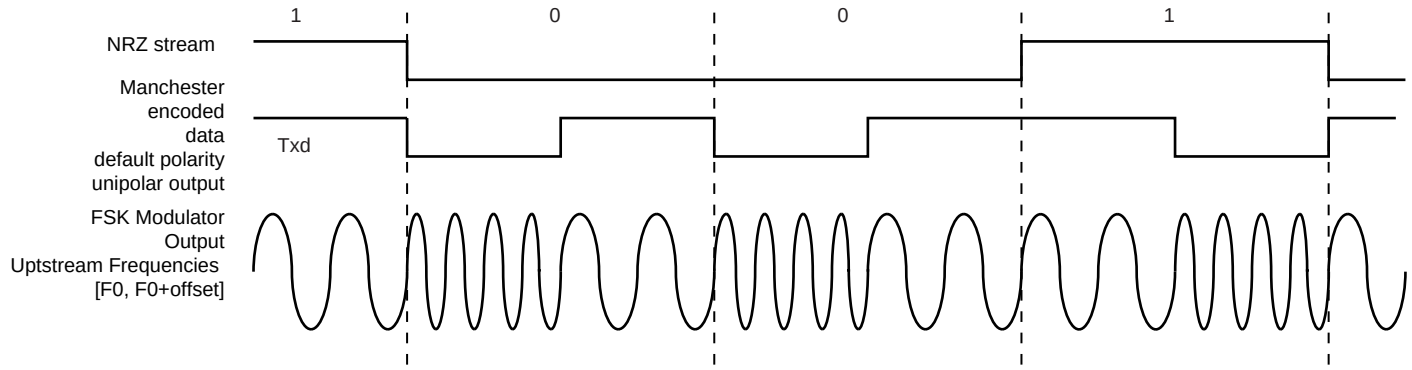


Figure 25-19. FSK Modulator Output



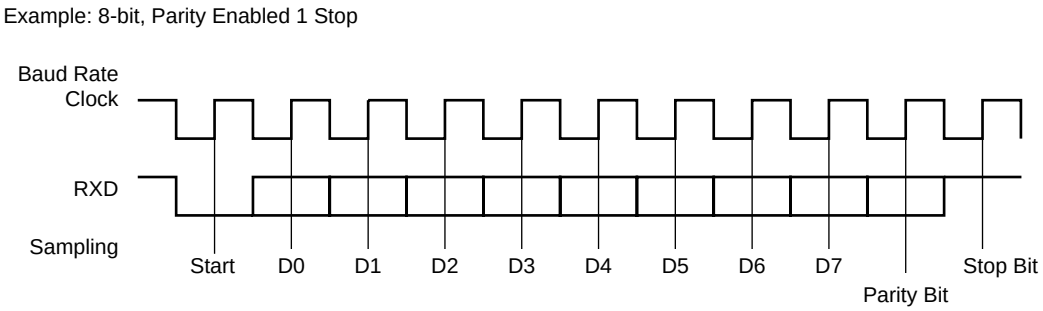
25.7.3.7 Synchronous Receiver

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 25-20 illustrates a character reception in synchronous mode.

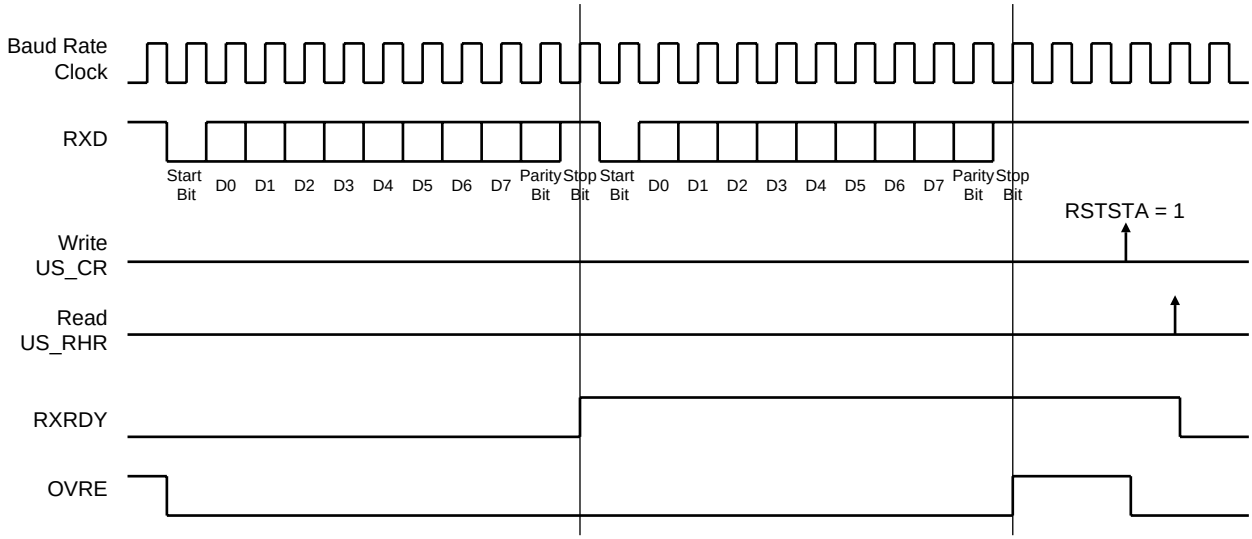
Figure 25-20. Synchronous Mode Character Reception



25.7.3.8 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (RHR) and the RXRDY bit in the Status Register (CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (CR) with the RSTSTA (Reset Status) bit at 1.

Figure 25-21. Receiver Status



25.7.3.9 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (MR). The PAR field also enables the Multidrop mode, see ”Multidrop Mode” on page 299. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

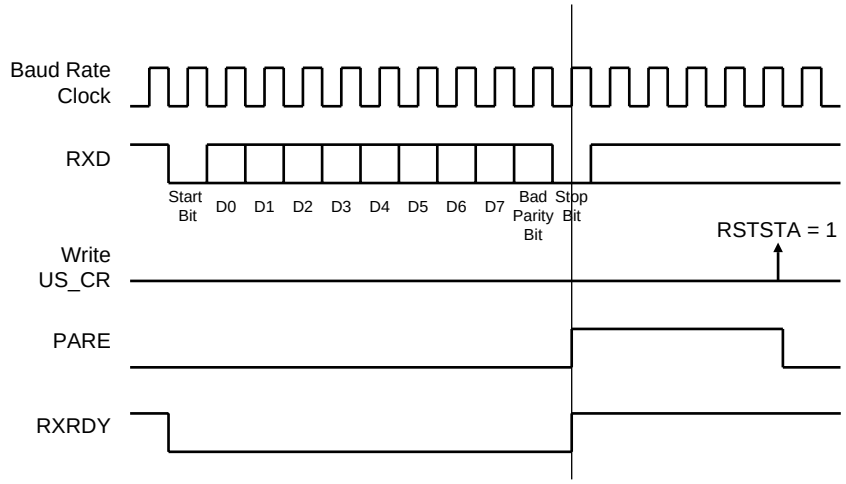
Table 25-6 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 25-6. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (CSR). The PARE bit can be cleared by writing the Control Register (CR) with the RSTSTA bit at 1. Figure 25-22 illustrates the parity bit status setting and clearing.

Figure 25-22. Parity Error



25.7.3.10 Multidrop Mode

If the PAR field in the Mode Register (MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to CR. In this case, the next byte written to THR is transmitted as an address. Any character written in THR without having written the command SENDA is transmitted normally with the parity at 0.

25.7.3.11 Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 25-23, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 25-23. Timeguard Operations

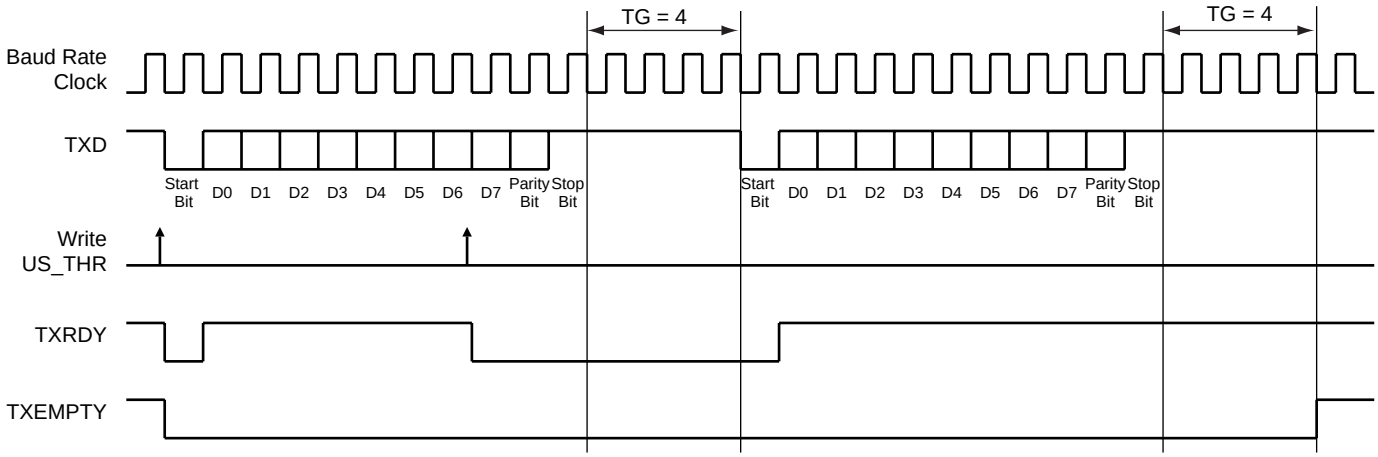


Table 25-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 25-7. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

25.7.3.12 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises.

The user can either:

- Obtain an interrupt when a time-out is detected after having received at least one character. This is performed by writing the Control Register (CR) with the STTTO (Start Time-out) bit at 1.
- Obtain a periodic interrupt while no character is received. This is performed by writing CR with the RETTO (Reload and Start Time-out) bit at 1.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 25-24 shows the block diagram of the Receiver Time-out feature.

Figure 25-24. Receiver Time-out Block Diagram

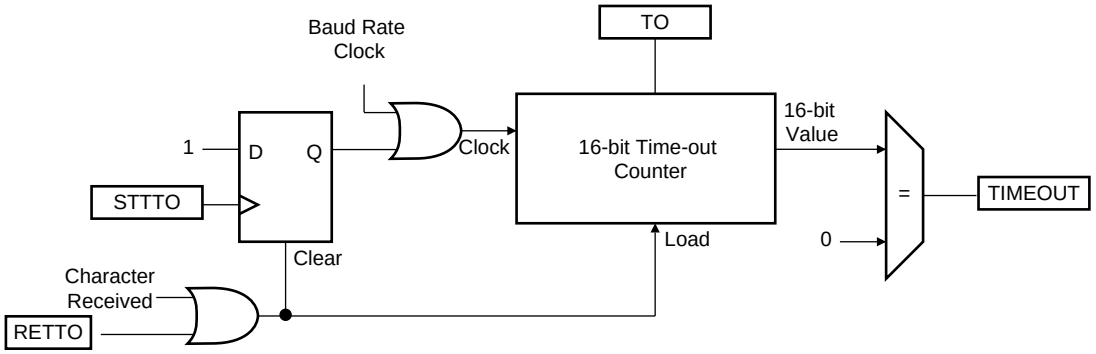


Table 25-8 gives the maximum time-out period for some standard baud rates.

Table 25-8. Maximum Time-out Period

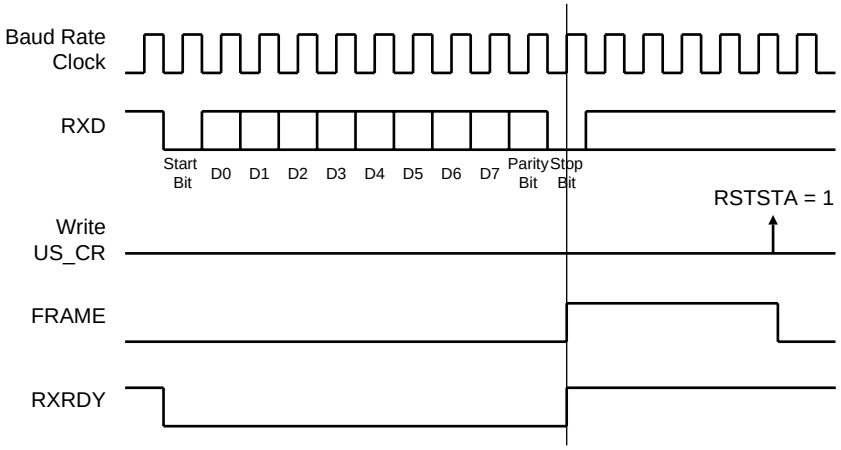
Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962
56000	18	1 170
57600	17	1 138
200000	5	328

25.7.3.13 Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (CR) with the RSTSTA bit at 1.

Figure 25-25. Framing Error Status



25.7.3.14 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

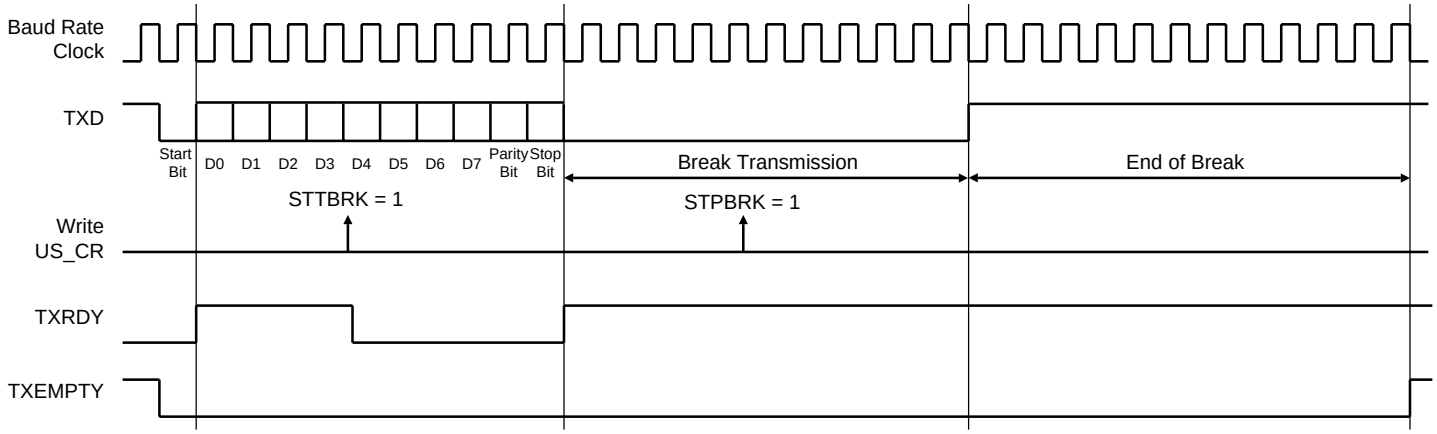
Writing CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 25-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 25-26. Break Transmission



25.7.3.15 *Receive Break*

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

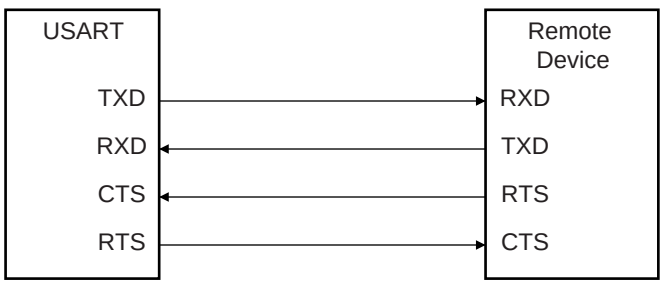
When the low stop bit is detected, the receiver asserts the RXBRK bit in CSR. This bit may be cleared by writing the Control Register (CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

25.7.3.16 *Hardware Handshaking*

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 25-27.

Figure 25-27. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the MODE field in the Mode Register (MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 25-28 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 25-28. Receiver Behavior when Operating with Hardware Handshaking

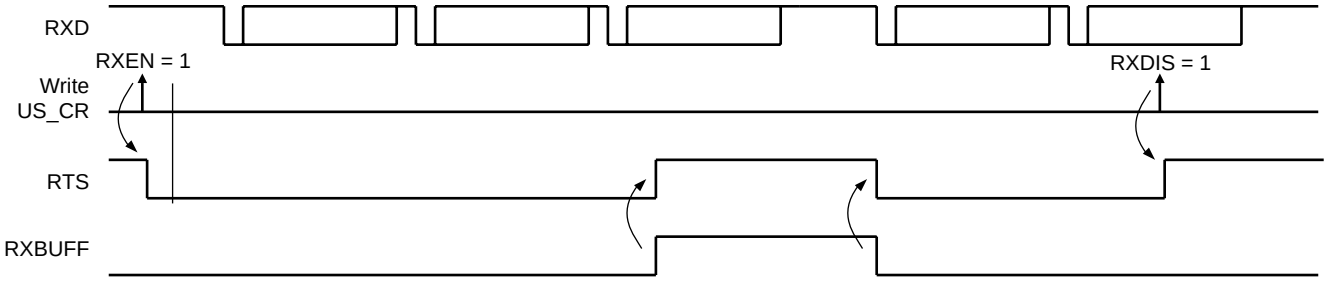
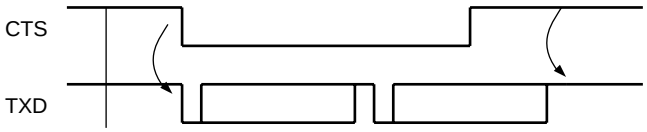


Figure 25-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 25-29. Transmitter Behavior when Operating with Hardware Handshaking



25.7.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

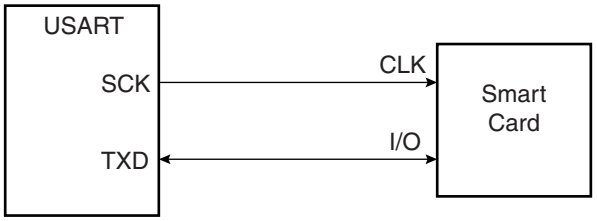
Setting the USART in ISO7816 mode is performed by writing the MODE field in the Mode Register (MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

25.7.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 283).

The USART connects to a smart card as shown in Figure 25-30. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 25-30. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "USART Mode Register" on page 317 and "PAR: Parity Type" on page 318.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (THR) or after reading it in the Receive Holding Register (RHR).

25.7.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 25-31.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 25-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (RHR). It appropriately sets the PARE bit in the Status Register (SR) so that the software can handle the error.

Figure 25-31. T = 0 Protocol without Parity Error

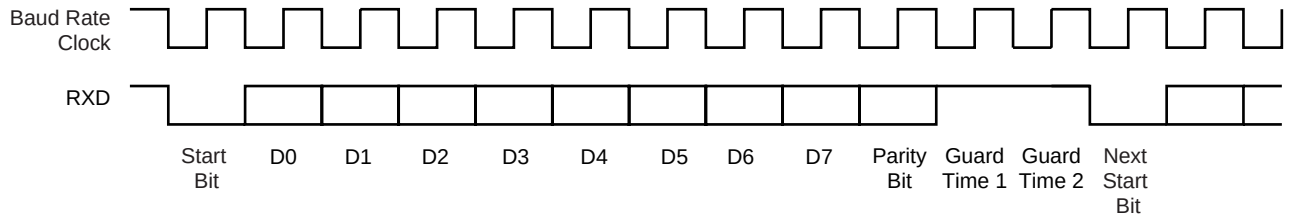
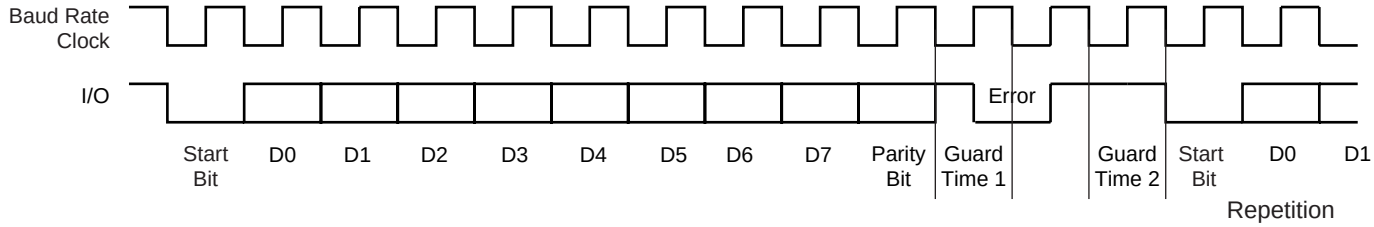


Figure 25-32. T = 0 Protocol with Parity Error



25.7.4.3 Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (NER) register. The NB_ERRORS field can record up to 255 errors. Reading NER automatically clears the NB_ERRORS field.

25.7.4.4 Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (SR). The INACK bit can be cleared by writing the Control Register (CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

25.7.4.5 Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in CSR can be cleared by writing the Control Register with the RSIT bit at 1.

25.7.4.6 Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as

MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

25.7.4.7 *Protocol T = 1*

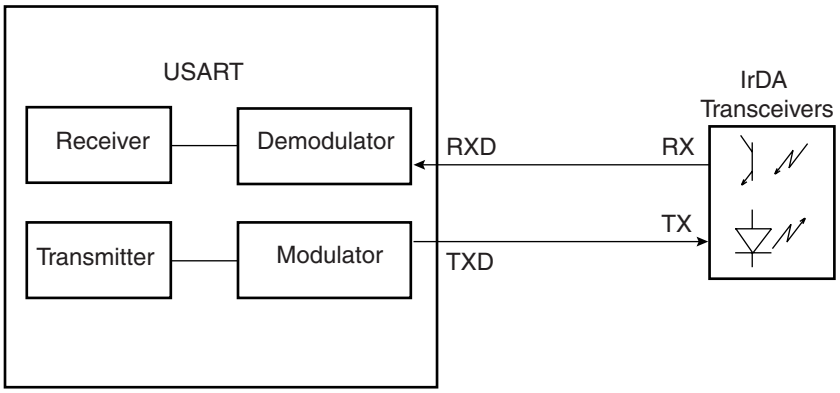
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (CSR).

25.7.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 25-33. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the MODE field in the Mode Register (MR) to the value 0x8. The IrDA Filter Register (IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 25-33. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

25.7.5.1 *IrDA Modulation*

For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 25-9.

Table 25-9. IrDA Pulse Duration

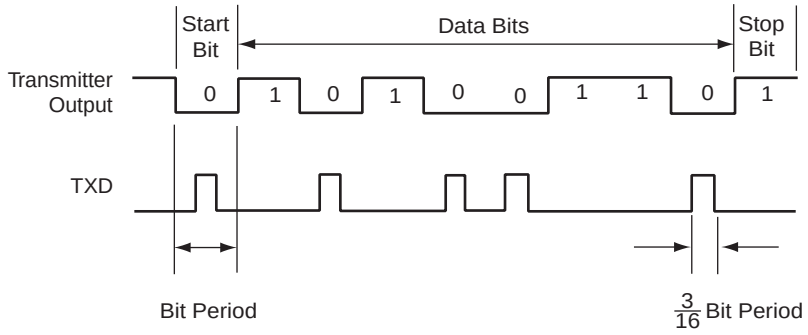
Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs

Table 25-9. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 25-34 shows an example of character transmission.

Figure 25-34. IrDA Modulation



25.7.5.2 IrDA Baud Rate

Table 25-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 25-10. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77

Table 25-10. IrDA Baud Rate Error (Continued)

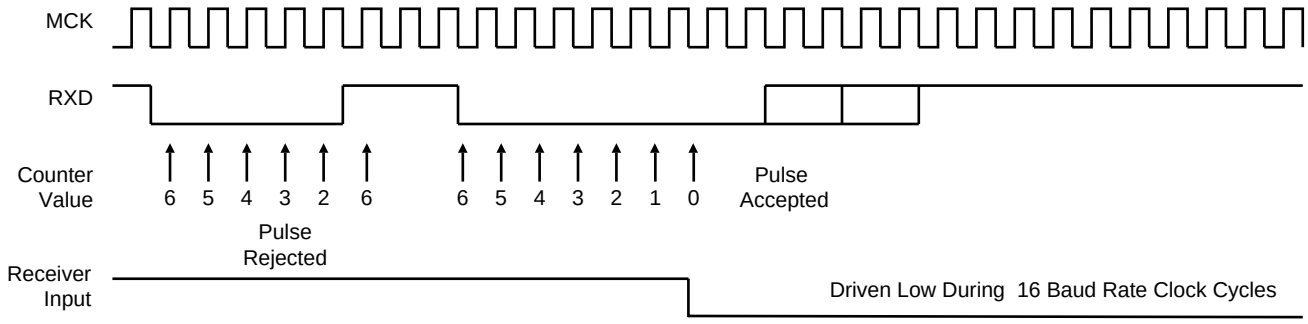
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

25.7.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 25-35 illustrates the operations of the IrDA demodulator.

Figure 25-35. IrDA Demodulator Operations

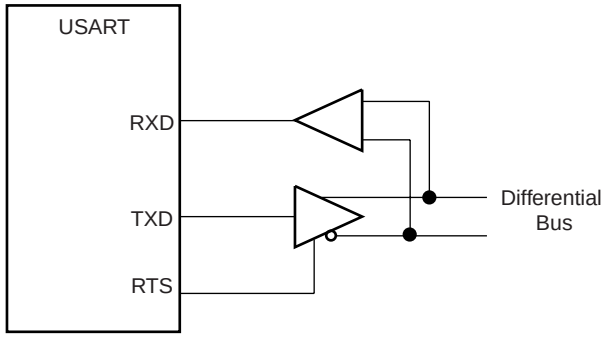


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

25.7.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 25-36.

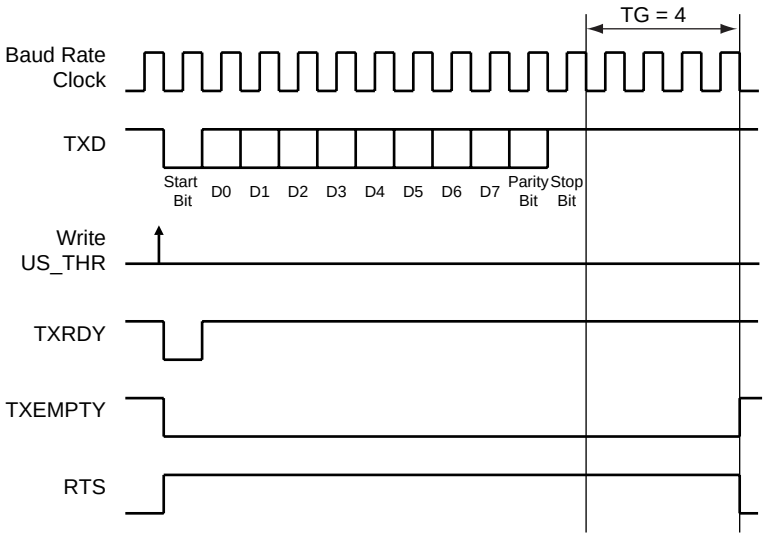
Figure 25-36. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the MODE field in the Mode Register (MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 25-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 25-37. Example of RTS Drive with Timeguard



25.7.7 Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the MODE field in the Mode Register (MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 25-11 gives the correspondence of the USART signals with modem connection standards.

Table 25-11. Circuit References

USART Pin	V24	CCITT	Direction
TXD	2	103	From terminal to modem
RTS	4	105	From terminal to modem
DTR	20	108.2	From terminal to modem
RXD	3	104	From modem to terminal
CTS	5	106	From terminal to modem
DSR	6	107	From terminal to modem
DCD	8	109	From terminal to modem
RI	22	125	From terminal to modem

The control of the DTR output pin is performed by writing the Control Register (US_CR) with the DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable command forces the corresponding pin to its active level, i.e. low. RTS output pin is automatically controlled in this mode

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (US_CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when US_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

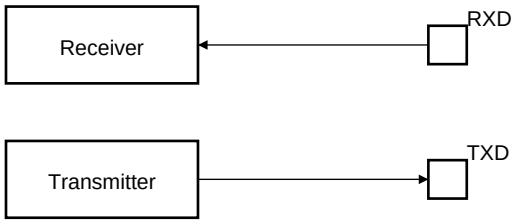
25.7.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

25.7.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

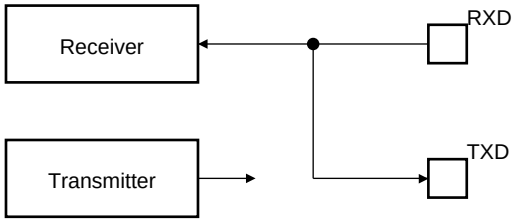
Figure 25-38. Normal Mode Configuration



25.7.8.2 Automatic Echo Mode

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 25-39. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

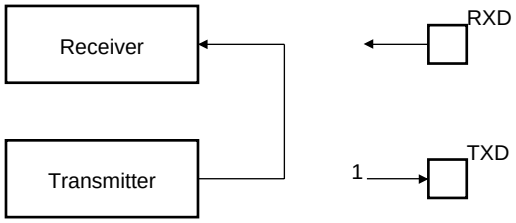
Figure 25-39. Automatic Echo Mode Configuration



25.7.8.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 25-40. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

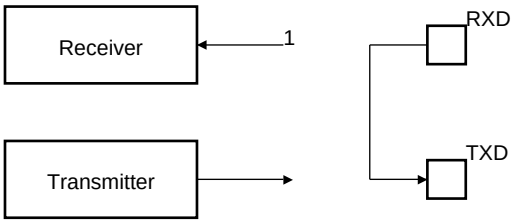
Figure 25-40. Local Loopback Mode Configuration



25.7.8.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 25-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 25-41. Remote Loopback Mode Configuration



25.8 USART User Interface

Table 25-12. USART Memory Map

Offset	Register	Name	Access	Reset State
0x0000	Control Register	CR	Write-only	–
0x0004	Mode Register	MR	Read/Write	–
0x0008	Interrupt Enable Register	IER	Write-only	–
0x000C	Interrupt Disable Register	IDR	Write-only	–
0x0010	Interrupt Mask Register	IMR	Read-only	0x0
0x0014	Channel Status Register	CSR	Read-only	–
0x0018	Receiver Holding Register	RHR	Read-only	0x0
0x001C	Transmitter Holding Register	THR	Write-only	–
0x0020	Baud Rate Generator Register	BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	TTGR	Read/Write	0x0
0x2C - 0x3C	Reserved	–	–	–
0x0040	FI DI Ratio Register	FIDI	Read/Write	0x174
0x0044	Number of Errors Register	NER	Read-only	–
0x0048	Reserved	-	–	–
0x004C	IrDA Filter Register	IF	Read/Write	0x0
0x0050	Manchester Encoder Decoder Register	MAN	Read/Write	0x30011004
0x5C - 0xFC	Reserved	–	–	–
0x100 - 0x128	Reserved for PDC Registers	–	–	–

25.8.1 USART Control Register

Name: CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

1: Resets the status bits PARE, FRAME, OVRE, MANERR and RXBRK in CSR.

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect

1: Starts waiting for a character before clocking the time-out counter.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR at 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

25.8.2 USART Mode Register

Name: MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC–	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		MODE			

- **MODE**

MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Modem
0	1	0	0	IS07816 Protocol: T = 0
0	1	0	1	Reserved
0	1	1	0	IS07816 Protocol: T = 1
0	1	1	1	Reserved
1	0	0	0	IrDA
1	1	x	x	Reserved

- **USCLKS: Clock Selection**

USCLKS		Selected Clock
0	0	MCK
0	1	MCK / DIV
1	0	Reserved
1	1	SCK

- **CHRL: Character Length.**

CHRL		Character Length
0	0	5 bits

0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **NBSTOP: Number of Stop Bits**

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **CKLO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16x Oversampling.

1: 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **VAR_SYNC: Variable synchronization of command/data sync Start Frame Delimiter**

0: User defined configuration of command or data sync field depending on SYNC value.

1: The sync field is updated when a character is written into THR register.

- **MAX_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T= 0.

- **FILTER: Infrared Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

- **MAN: Manchester Encoder/Decoder Enable**

0: Manchester Encoder/Decoder are disabled.

1: Manchester Encoder/Decoder are enabled.

- **MODSYNC: Manchester Synchronization mode**

0:The Manchester Start bit is a 0 to 1 transition

1: The Manchester Start bit is a 1 to 0 transition.

- **ONEBIT: Start Frame Delimiter selector**

0: Start Frame delimiter is COMMAND or DATA SYNC.

1: Start Frame delimiter is One Bit.

25.8.3 USART Interrupt Enable Register

Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Transfer Interrupt Enable**
- **ENDTX: End of Transmit Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITERATION: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **RIIC: Ring Indicator Input Change Enable**
- **DSRIC: Data Set Ready Input Change Enable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

25.8.4 USART Interrupt Disable Register

Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITERATION: Iteration Interrupt Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **RIIC: Ring Indicator Input Change Disable**
- **DSRIC: Data Set Ready Input Change Disable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**
- **MANE: Manchester Error Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

25.8.5 USART Interrupt Mask Register

Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Transfer Interrupt Mask**
- **ENDTX: End of Transmit Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITERATION: Iteration Interrupt Mask**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**
- **DSRIC: Data Set Ready Input Change Mask**
- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**
- **MANE: Manchester Error Interrupt Mask**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

25.8.6 USART Channel Status Register

Name: CSR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	MANERR
23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC
15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

1: At least one complete character has been received and RHR has not yet been read.

- **TXRDY: Transmitter Ready**

0: A character is in the THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the THR.

- **RXBRK: Break Received/End of Break**

0: No Break received or End of Break detected since the last RSTSTA.

1: Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0: The End of Transfer signal from the Receive PDC channel is inactive.

1: The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The End of Transfer signal from the Transmit PDC channel is inactive.

1: The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

0: There has not been a time-out since the last Start Time-out command or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command.

- **TXEMPTY: Transmitter Empty**

0: There are characters in either THR or the Transmit Shift Register, or the transmitter is disabled.

1: There is at least one character in either THR or the Transmit Shift Register.

- **ITERATION: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSIT.

1: Maximum number of repetitions has been reached since the last RSIT.

- **TXBUFE: Transmission Buffer Empty**

0: The signal Buffer Empty from the Transmit PDC channel is inactive.

1: The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0: The signal Buffer Full from the Receive PDC channel is inactive.

1: The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0: No Non Acknowledge has not been detected since the last RSTNACK.

1: At least one Non Acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag**

0: No input change has been detected on the RI pin since the last read of CSR.

1: At least one input change has been detected on the RI pin since the last read of CSR.

- **DSRIC: Data Set Ready Input Change Flag**

0: No input change has been detected on the DSR pin since the last read of CSR.

1: At least one input change has been detected on the DSR pin since the last read of CSR.

- **DCDIC: Data Carrier Detect Input Change Flag**

0: No input change has been detected on the DCD pin since the last read of CSR.

1: At least one input change has been detected on the DCD pin since the last read of CSR.

- **CTSIC: Clear to Send Input Change Flag**

0: No input change has been detected on the CTS pin since the last read of CSR.

1: At least one input change has been detected on the CTS pin since the last read of CSR.

- **RI: Image of RI Input**

0: RI is at 0.

1: RI is at 1.

- **DSR: Image of DSR Input**

0: DSR is at 0

1: DSR is at 1.

- **DCD: Image of DCD Input**

0: DCD is at 0.

1: DCD is at 1.

- **CTS: Image of CTS Input**

0: CTS is at 0.

1: CTS is at 1.

- **MANERR: Manchester Error**

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

25.8.7 USART Receive Holding Register

Name: RHR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
RXSYNH	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

25.8.8 USART Transmit Holding Register

Name: THR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TXSYNH	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

25.8.9 USART Baud Rate Generator Register

Name: BRGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	FP		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divider**

CD	**MODE ≠ ISO7816**			MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	**OVER = 1**		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

25.8.10 USART Receiver Time-out Register

Name: RTOR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

25.8.11 USART Transmitter Timeguard Register

Name: TTGR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TG							

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

25.8.12 USART FI DI RATIO Register

Name: FIDI

Access Type: Read/Write

Reset Value : 0x174

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.

25.8.13 USART Number of Errors Register

Name: NER

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

25.8.14 USART Manchester Configuration Register

Name: MAN

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	–	RX_MPOL	–	–	RX_PP	

23	22	21	20	19	18	17	16
–	–	–	–	RX_PL			

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

- **TX_PL: Transmitter Preamble Length**

0: The Transmitter Preamble pattern generation is disabled

1 - 15: The Preamble Length is TX_PL x Bit Period

- **TX_PP: Transmitter Preamble Pattern**

TX_PP		Preamble Pattern default polarity assumed (TX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1 - 15: The detected preamble length is RX_PL x Bit Period

- **RX_PP: Receiver Preamble Pattern detected**

RX_PP		Preamble Pattern default polarity assumed (RX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **RX_MPOL: Receiver Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **DRIFT: Drift compensation**

0: The USART can not recover from an important clock drift

1: The USART can recover from clock drift. The 16X clock mode must be enabled.

25.8.15 USART IrDA FILTER Register

Name: IF

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IRDA_FILTER							

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

26. USB On-The-Go Interface (USBB)

Rev: 2.6.0.0

26.1 Features

- **USB 2.0 Compliant, Full-/Low-Speed (FS/LS) and On-The-Go (OTG), 12 Mbit/s**
- **7 Pipes/Endpoints**
- **960 bytes of Embedded Dual-Port RAM (DPRAM) for Pipes/Endpoints**
- **Up to 2 Memory Banks per Pipe/Endpoint (Not for Control Pipe/Endpoint)**
- **Flexible Pipe/Endpoint Configuration and Management with Dedicated DMA Channels**
- **On-Chip Transceivers Including Pull-Ups**

26.2 Description

The Universal Serial Bus (USB) MCU device complies with the Universal Serial Bus (USB) 2.0 specification, but it does NOT feature high-speed USB (480 Mbit/s).

Each pipe/endpoint can be configured in one of several transfer types. It can be associated with one or more banks of a dual-port RAM used to store the current data payload. If several banks are used (“ping-pong” mode), then one DPRAM bank is read or written by the CPU or the DMA while the other is read or written by the USB macro core. This feature is mandatory for isochronous pipes/endpoints.

Table 26-1 describes the hardware configuration of the USB MCU device.

Table 26-1. Description of USB Pipes/Endpoints

Pipe/Endpoint	Mnemonic	Max. Size	Max. Nb. Banks	DMA	Type
0	PEP0	64 bytes	1	N	Control
1	PEP1	64 bytes	2	Y	Isochronous/Bulk/Interrupt
2	PEP2	64 bytes	2	Y	Isochronous/Bulk/Interrupt
3	PEP3	64 bytes	2	Y	Isochronous/Bulk/Interrupt
4	PEP4	64 bytes	2	Y	Isochronous/Bulk/Interrupt
5	PEP5	256 bytes	2	Y	Isochronous/Bulk/Interrupt
6	PEP6	256 bytes	2	Y	Isochronous/Bulk/Interrupt

The theoretical maximal pipe/endpoint configuration (1600 bytes) exceeds the real DPRAM size (960 bytes). The user needs to be aware of this when configuring pipes/endpoints. To fully use

the 960 bytes of DPRAM, the user could for example use the configuration described in Table 26-2.

Table 26-2. Example of Configuration of Pipes/Endpoints Using the Whole DPRAM

Pipe/Endpoint	Mnemonic	Size	Nb. Banks
0	PEP0	64 bytes	1
1	PEP1	64 bytes	2
2	PEP2	64 bytes	2
3	PEP3	64 bytes	1
4	PEP4	64 bytes	1
5	PEP5	256 bytes	1
6	PEP6	256 bytes	1

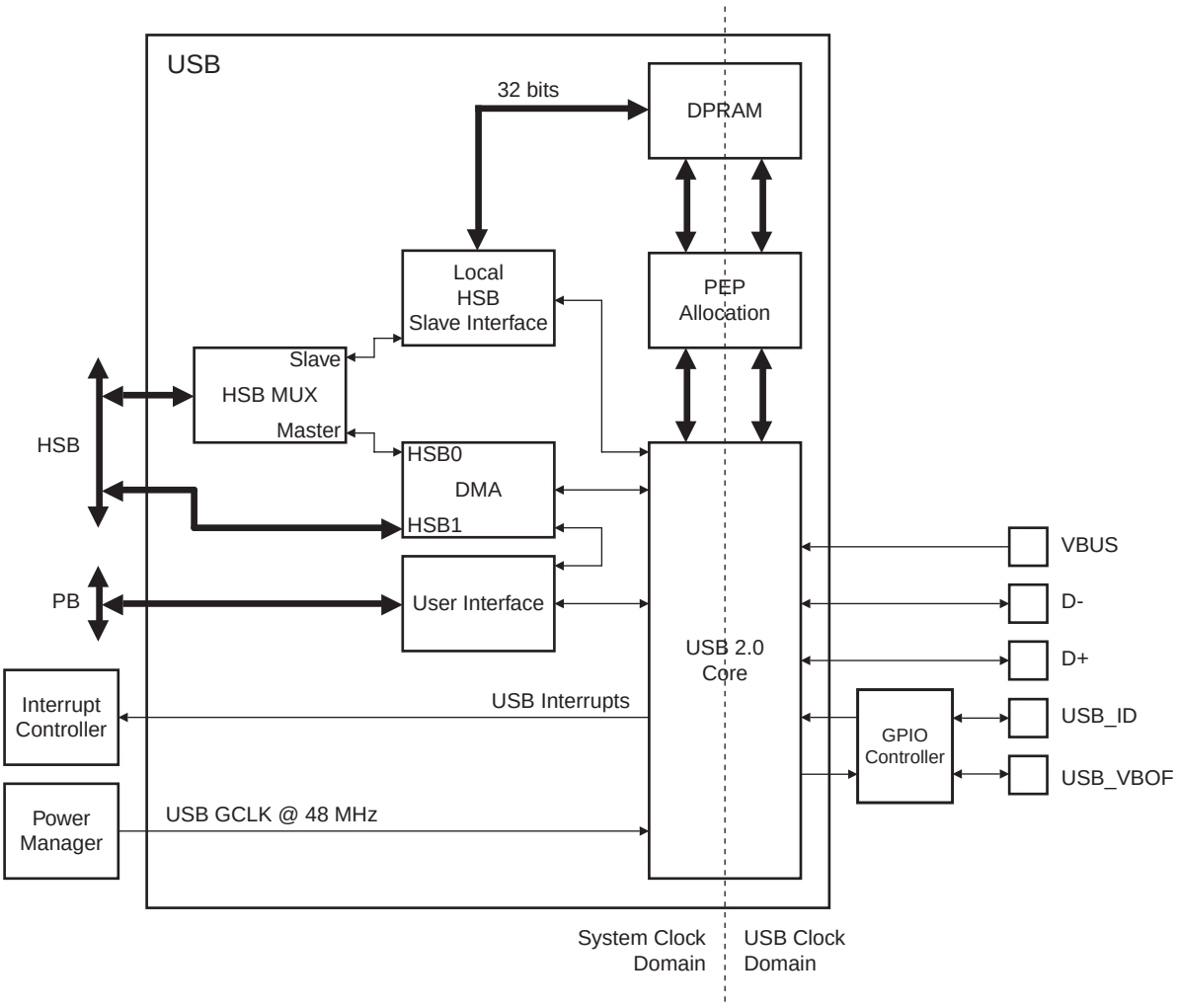
26.3 Block Diagram

The USB controller provides a hardware device to interface a USB link to a data flow stored in a dual-port RAM (DPRAM).

The USB controller requires a 48 MHz ± 0.25% reference clock, which is the USB generic clock generated from one of the power manager oscillators, optionally through one of the power manager PLLs.

The 48 MHz clock is used to generate a 12 MHz full-speed (or 1.5 MHz low-speed) bit clock from the received USB differential data and to transmit data according to full- or low-speed USB device tolerance. Clock recovery is achieved by a digital phase-locked loop (a DPLL, not represented), which complies with the USB jitter specifications.

Figure 26-1. Block Diagram



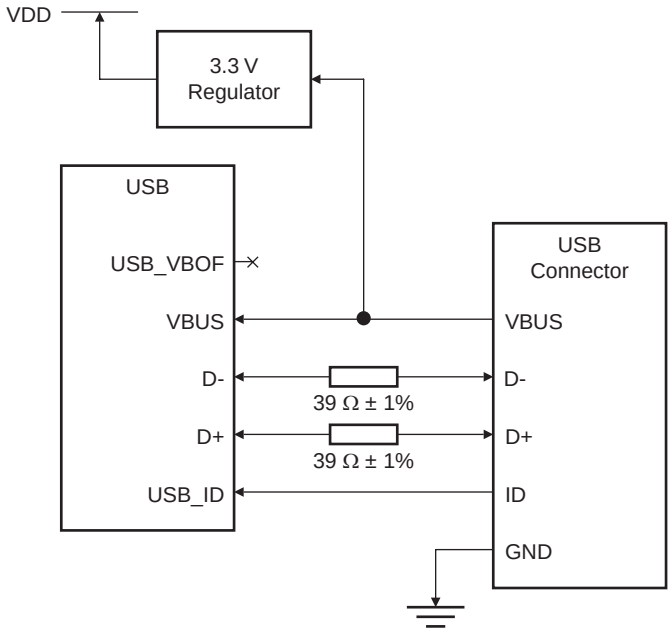
26.4 Application Block Diagram

Depending on the USB operating mode (device-only, reduced-host or OTG mode) and the power source (bus-powered or self-powered), the uC301x requires different typical hardware implementations.

26.4.1 Device Mode

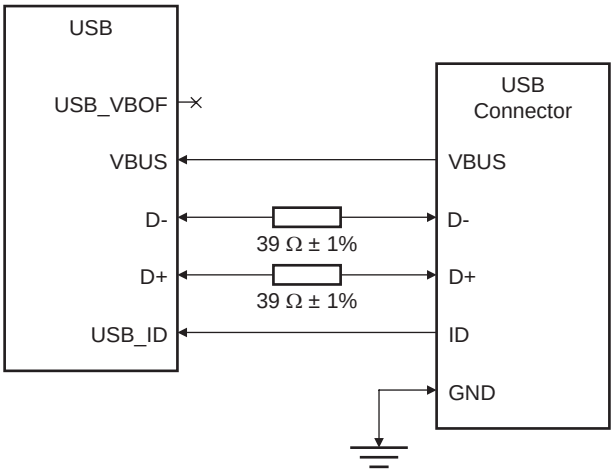
26.4.1.1 Bus-Powered Device

Figure 26-2. Bus-Powered Device Application Block Diagram



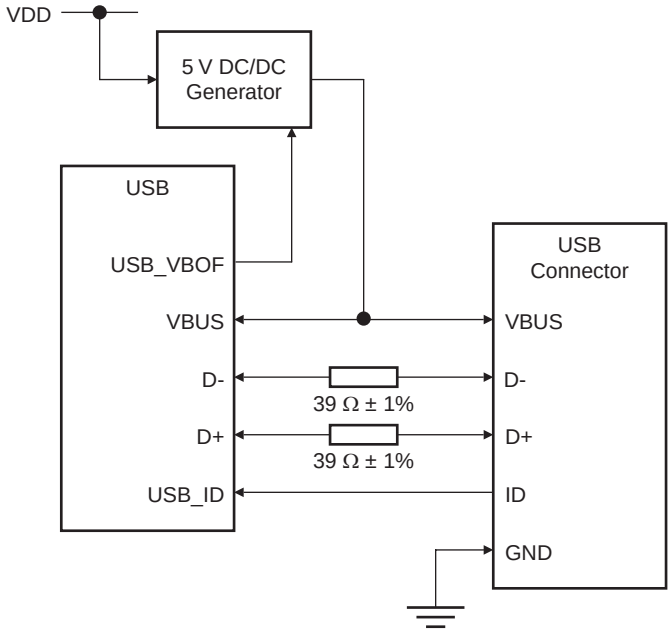
26.4.1.2 Self-Powered Device

Figure 26-3. Self-Powered Device Application Block Diagram



26.4.2 Host and OTG Modes

Figure 26-4. Host and OTG Application Block Diagram



26.5 I/O Lines Description

Table 26-3. I/O Lines Description

Name	Description	Type	Active Level
USB_VBOF	USB VBus On/Off: Bus Power Control Port	Output	$\overline{\text{VBUSPO}}$
VBUS	VBus: Bus Power Measurement Port	Input	High
D-	Data -: Differential Data Line - Port	Input/Output	N/A
RI		Input	Low
DSR		Input	Low
DCD		Input	Low
DTR		Output	Low
D+	Data +: Differential Data Line + Port	Input/Output	N/A
USB_ID	USB Identification: Mini Connector Identification Port	Input	Low: Mini-A plug High Z: Mini-B plug

26.6 Product Dependencies

26.6.1 I/O Lines

The USB_VBOF and USB_ID pins are multiplexed with GPIO lines and may also be multiplexed with lines of other peripherals. In order to use them with the USB, the programmer must first program the GPIO controller to assign them to their USB peripheral functions. Moreover, if USB_ID is used, the GPIO controller must be configured to enable the internal pull-up resistor of its pin.

If USB_VBOF or USB_ID is not used by the application, the corresponding pin can be used for other purposes by the GPIO controller or by other peripherals.

26.6.2 Power Management

The 48 MHz USB clock is generated by a dedicated generic clock from the power manager. Before using the USB, the programmer must ensure that the USB generic clock (USB GCLK) is enabled at 48 MHz in the power manager.

26.6.3 Interrupts

The USB interface has an interrupt line connected to the interrupt controller. In order to handle USB interrupts, the interrupt controller must be programmed first.

26.7 Functional Description

26.7.1 USB General Operation

26.7.1.1 Introduction

After a hardware reset, the USB controller is disabled. When enabled, the USB controller runs either in device mode or in host mode according to the ID detection.

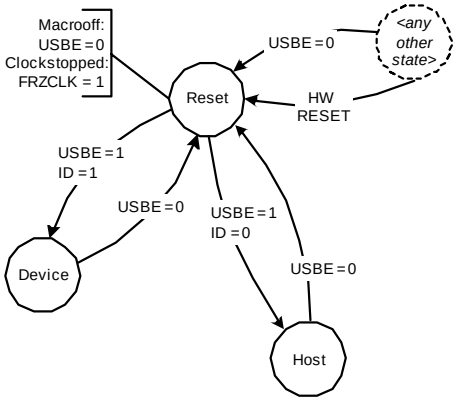
If the USB_ID pin is not connected to ground, the ID bit is set by hardware (the internal pull-up resistor of the USB_ID pin must be enabled by the GPIO controller) and device mode is engaged.

The ID bit is cleared by hardware when a low level has been detected on the USB_ID pin. Host mode is then engaged.

26.7.1.2 Power-On and Reset

Figure 26-5 describes the USB controller main states.

Figure 26-5. General States



After a hardware reset, the USB controller is in the Reset state. In this state:

- the macro is disabled (USBE = 0);
- the macro clock is stopped in order to minimize power consumption (FRZCLK = 1);
- the pad is in suspend mode;
- the internal states and registers of the device and host modes are reset;
- the DPRAM is not cleared and is accessible;
- the ID and VBUS read-only bits reflect the states of the USB_ID and VBUS input pins;
- the OTGPADE, VBUSPO, FRZCLK, USBE, UIDE, UIMOD and LS bits can be written by software, so that the user can program pads and speed before enabling the macro, but their value is only taken into account once the macro is enabled and unfrozen.

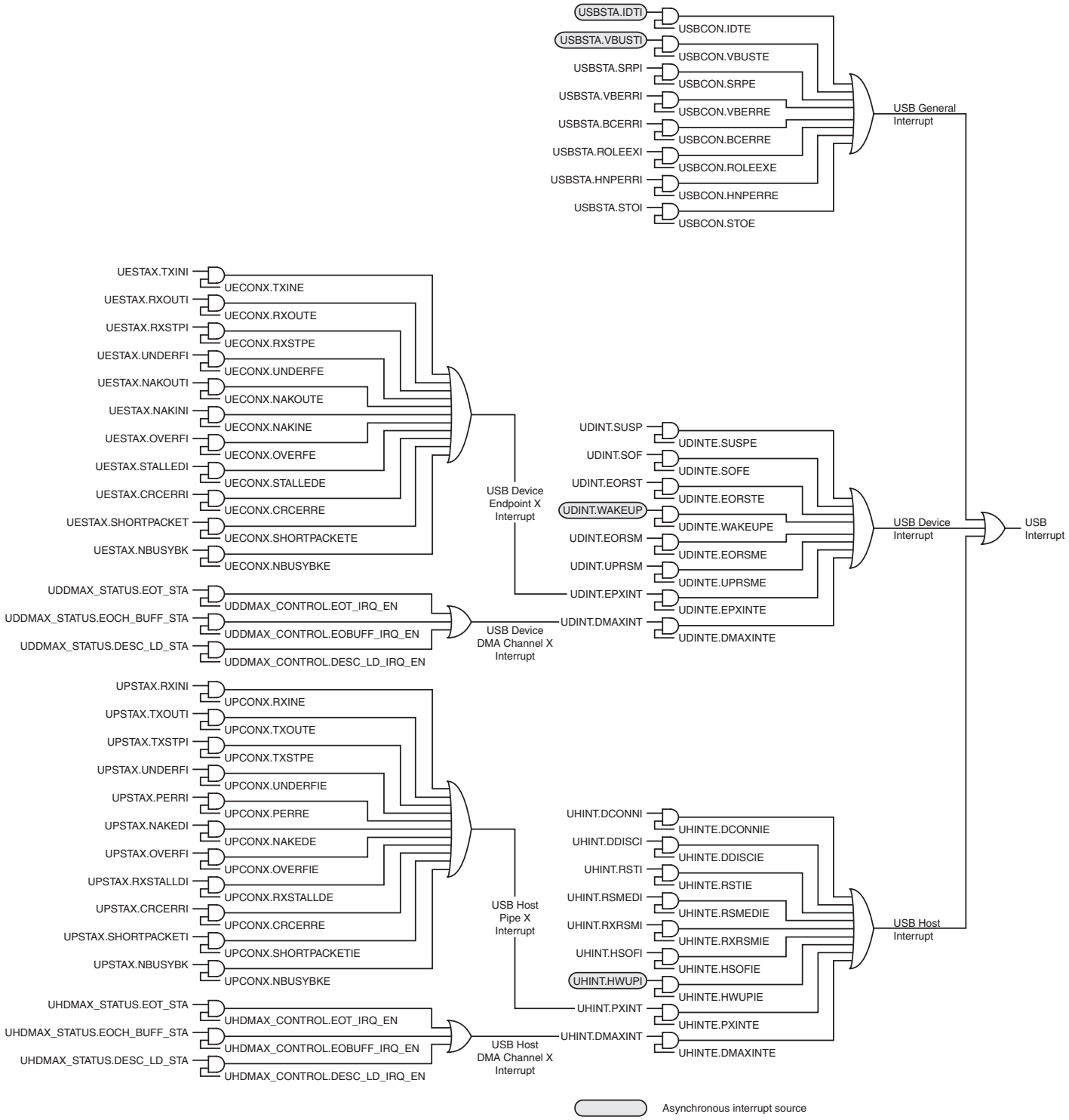
After setting USBE, the USB controller enters the Device or the Host mode (according to the ID detection) in idle state.

The USB controller can be disabled at any time by clearing USBE. In fact, clearing USBE acts as a hardware reset, except that the OTGPADE, VBUSPO, FRZCLK, UIDE, UIMOD and LS bits are not reset.

26.7.1.3 Interrupts

One interrupt vector is assigned to the USB interface. Figure 26-6 shows the structure of the USB interrupt system.

Figure 26-6. Interrupt System



See Section 26.7.2.17 on page 359 and Section 26.7.3.13 on page 367 for further details about device and host interrupts.

There are two kinds of general interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

The processing general interrupts are:

- the ID Transition interrupt (IDTI);
- the VBus Transition interrupt (VBUSTI);
- the SRP interrupt (SRPI);
- the Role Exchange interrupt (ROLEEXI).

The exception general interrupts are:

- the VBus Error interrupt (VBERRI);
- the B-Connection Error interrupt (BCERRI);
- the HNP Error interrupt (HNPERRI);
- the Suspend Time-Out interrupt (STOI).

26.7.1.4 MCU Power Modes

26.7.1.4.1 Run Mode

In this mode, all MCU clocks can run, including the USB clock.

26.7.1.4.2 Idle Mode

In this mode, the CPU is halted, i.e. the CPU clock is stopped. The Idle mode is entered whatever the state of the USB macro. The MCU wakes up on any USB interrupt.

26.7.1.4.3 Frozen Mode

Same as the Idle mode, except that the HSB module is stopped, so the USB DMA, which is an HSB master, can not be used. Moreover, the USB DMA must be stopped before entering this sleep mode in order to avoid erratic behavior. The MCU wakes up on any USB interrupt.

26.7.1.4.4 Standby, Stop and Static Modes

Same as the Frozen mode, except that the USB generic clock and other clocks are stopped, so the USB macro is frozen. In the current version of the MCU, no USB interrupt can wake up the MCU in these modes, even the asynchronous interrupt sources.

26.7.1.4.5 USB Clock Frozen

In the Run, Idle and Frozen MCU modes, the USB macro can be frozen by setting the FRZCLK bit, what reduces power consumption.

In this case, it is still possible to access the following elements, but only in Run mode:

- the OTGPADE, VBUSPO, FRZCLK, USBE, UIDE, UIMOD and LS bits;
- the DPRAM (through the USB_FIFOX_DATA registers, but not through USB bus transfers which are frozen).

Moreover, when FRZCLK is set, only the asynchronous interrupt sources may trigger the USB interrupt:

- the ID Transition interrupt (IDTI);

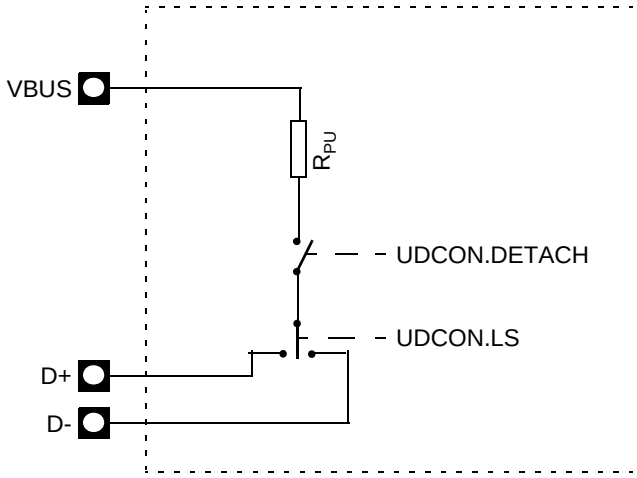
- the VBus Transition interrupt (VBUSTI);
- the Wake-Up interrupt (WAKEUP);
- the Host Wake-Up interrupt (HWUPI).

26.7.1.5 Speed Control

26.7.1.5.1 Device Mode

When the USB interface is in device mode, the speed selection (full-/low-speed) depends on which of D+ and D- is pulled up. The LS bit allows to connect an internal pull-up resistor either on D+ (full-speed mode) or on D- (low-speed mode). The LS bit should be configured before attaching the device, what can be done by clearing the DETACH bit.

Figure 26-7. Speed Selection in Device Mode



26.7.1.5.2 Host Mode

When the USB interface is in host mode, internal pull-down resistors are connected on both D+ and D- and the interface detects the speed of the connected device, which is reflected by the SPEED bit-field.

26.7.1.6 DPRAM Management

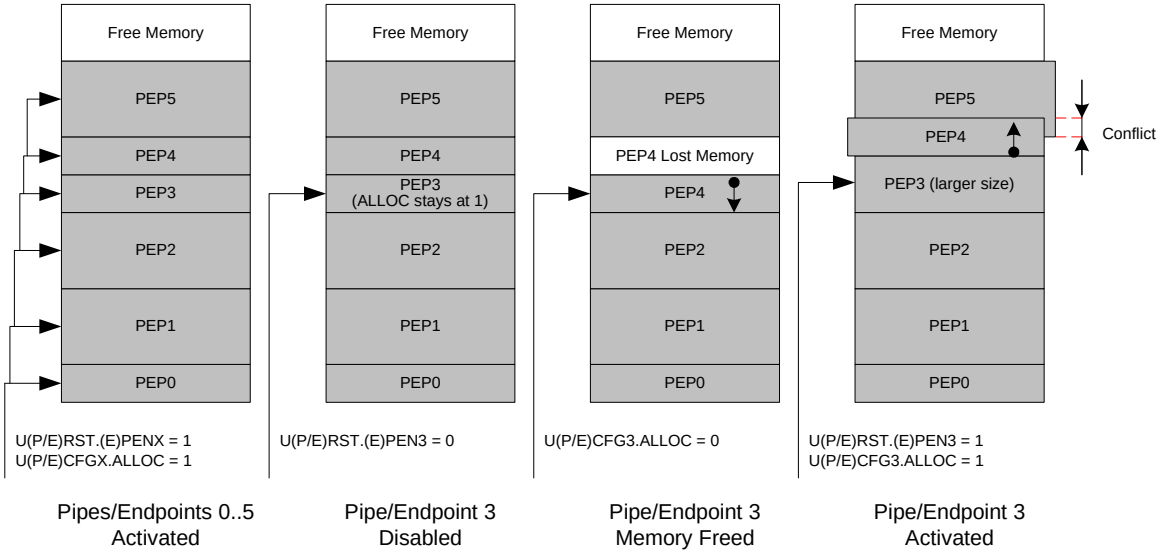
Pipes and endpoints can only be allocated in ascending order (from the pipe/endpoint 0 to the last pipe/endpoint to be allocated). The firmware shall therefore configure them in the same order.

The allocation of a pipe/endpoint k_i starts when its ALLOC bit is set. Then, the hardware allocates a memory area in the DPRAM and inserts it between the k_{i-1} and k_{i+1} pipes/endpoints. The k_{i+1} pipe/endpoint memory window slides up and its data is lost. Note that the following pipe/endpoint memory windows (from k_{i+2}) do not slide.

Disabling a pipe (PENX = 0) or an endpoint (EPENX = 0) resets neither its ALLOC bit nor its configuration (PBK/EPBK, PSIZE/EPSIZE, PTOKEN/EPDIR, PTYPE/EPTYPE, PEPNUM, INTFRQ). To free its memory, the firmware should clear its ALLOC bit. The k_{i+1} pipe/endpoint memory window then slides down and its data is lost. Note that the following pipe/endpoint memory windows (from k_{i+2}) do not slide.

Figure 26-8 illustrates the allocation and reorganization of the DPRAM in a typical example.

Figure 26-8. Allocation and Reorganization of the DPRAM



- First, the pipes/endpoints 0 to 5 are enabled, configured and allocated in ascending order. Each pipe/endpoint then owns a memory area in the DPRAM.
- Then, the pipe/endpoint 3 is disabled, but its memory is kept allocated by the controller.
- In order to free its memory, its ALLOC bit is then cleared by the firmware. The pipe/endpoint 4 memory window slides down, but the pipe/endpoint 5 does not move.
- Finally, if the firmware chooses to reconfigure the pipe/endpoint 3 with a larger size, the controller allocates a memory area after the pipe/endpoint 2 memory area and automatically slides up the pipe/endpoint 4 memory window. The pipe/endpoint 5 does not move and a memory conflict appears as the memory windows of the pipes/endpoints 4 and 5 overlap. The data of these pipes/endpoints is potentially lost.

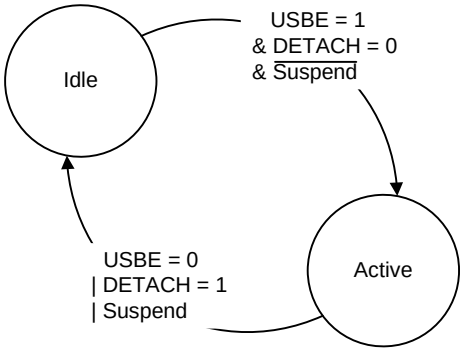
Note that:

- there is no way the data of the pipe/endpoint 0 can be lost (except if it is de-allocated) as memory allocation and de-allocation may affect only higher pipes/endpoints;
- deactivating then reactivating a same pipe/endpoint with the same configuration only modifies temporarily the controller DPRAM pointer and size for this pipe/endpoint, but nothing changes in the DPRAM, so higher endpoints seem to not have been moved and their data is preserved as far as nothing has been written or received into them while changing the allocation state of the first pipe/endpoint;
- when the firmware sets the ALLOC bit, the CFGOK bit is set by hardware only if the configured size and number of banks are correct compared to their maximal allowed values for the endpoint and to the maximal FIFO size (i.e. the DPRAM size), so the value of CFGOK does not consider memory allocation conflicts.

26.7.1.7 Pad Suspend

Figure 26-9 shows the pad behavior.

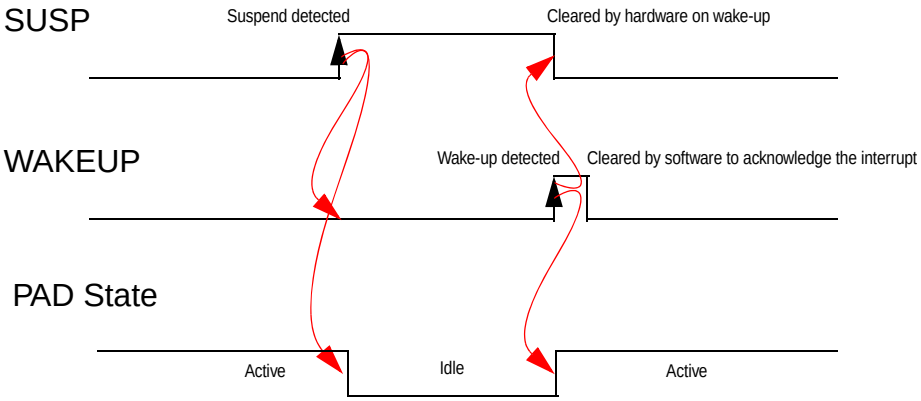
Figure 26-9. Pad Behavior



- In the Idle state, the pad is put in low power consumption mode.
- In the Active state, the pad is working.

Figure 26-10 illustrates the pad events leading to a PAD state change.

Figure 26-10. Pad Events



The Suspend interrupt flag (SUSP) is set and the Wake-Up interrupt flag (WAKEUP) is cleared when a USB “Suspend” state has been detected on the USB bus. This event automatically puts the USB pad in the Idle state. The detection of a non-idle event sets WAKEUP, clears SUSP and wakes up the USB pad.

Moreover, the pad goes to the Idle state if the macro is disabled or if the DETACH bit is set. It returns to the Active state when USBE = 1 and DETACH = 0.

26.7.1.8 Customizing of OTG Timers

It is possible to refine some OTG timers thanks to the TIMPAGE and TIMVALUE bit-fields, as shown by Figure 26-4.

Table 26-4. Customizing of OTG Timers

		TIMPAGE			
		00b: AWaitVrise Time-Out ([OTG] Chapter 6.6.5.1)	**01b: VbBusPulsing Time-Out ([OTG] Chapter 5.3.4)**	**10b: PdTmOutCnt Time-Out ([OTG] Chapter 5.3.2)**	**11b: SRPDetTmOut Time-Out ([OTG] Chapter 5.3.3)**
TIMVALUE	**00b**	20 ms	15 ms	93 ms	10 µs
	01b	50 ms	23 ms	105 ms	100 µs
	10b	70 ms	31 ms	118 ms	1 ms
	11b	100 ms	40 ms	131 ms	11 ms

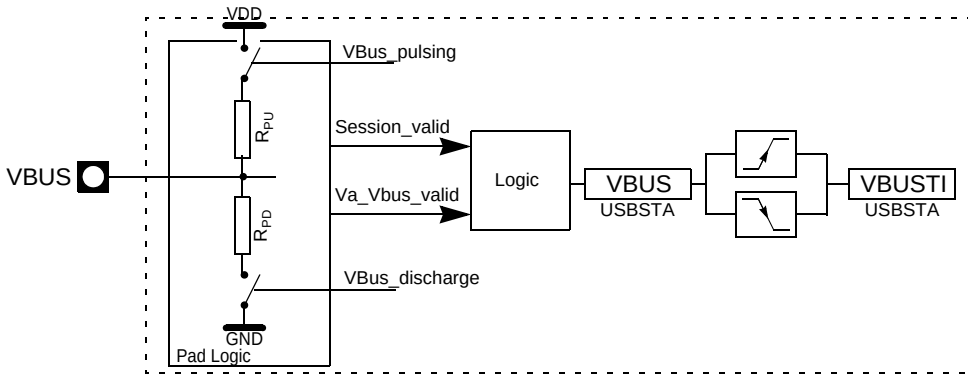
TIMPAGE is used to select the OTG timer to access while TIMVALUE indicates the time-out value of the selected timer.

TIMPAGE and TIMVALUE can be read or written. Before writing them, the firmware should unlock write accesses by setting the UNLOCK bit. This is not required for read accesses, except before accessing TIMPAGE if it has to be written in order to read the TIMVALUE bit-field of another OTG timer.

26.7.1.9 Plug-In Detection

The USB connection is detected from the VBUS pad. Figure 26-11 shows the architecture of the plug-in detector.

Figure 26-11. Plug-In Detection Input Block Diagram



The control logic of the VBUS pad outputs two signals:

- the Session_valid signal is high when the voltage on the VBUS pad is higher than or equal to 1.4 V;
- the Va_Vbus_valid signal is high when the voltage on the VBUS pad is higher than or equal to 4.4 V.

In device mode, the VBUS bit follows the Session_valid comparator output:

- it is set when the voltage on the VBUS pad is higher than or equal to 1.4 V;

- it is cleared when the voltage on the VBUS pad is lower than 1.4 V.

In host mode, the VBUS bit follows an hysteresis based on Session_valid and Va_Vbus_valid:

- it is set when the voltage on the VBUS pad is higher than or equal to 4.4 V;
- it is cleared when the voltage on the VBUS pad is lower than 1.4 V.

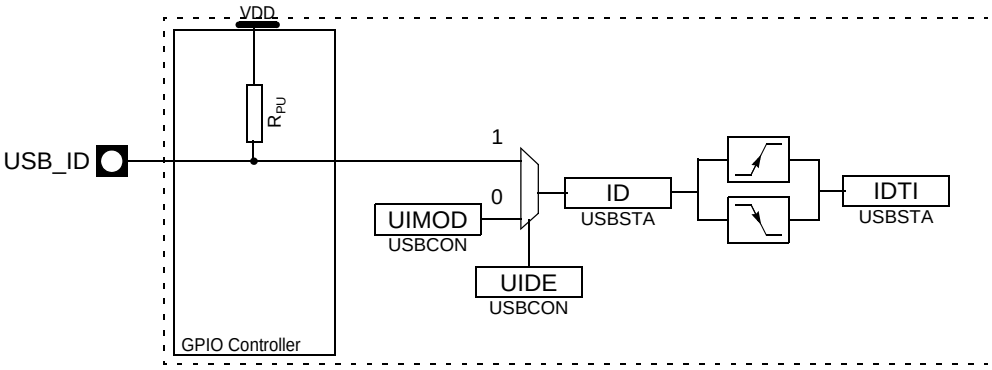
The VBus Transition interrupt (VBUSTI) is raised on each transition of the VBUS bit.

The VBUS bit is effective whether the USB macro is enabled or not.

26.7.1.10 ID Detection

Figure 26-12 shows how the ID transitions are detected.

Figure 26-12. ID Detection Input Block Diagram



The USB mode (device or host) can be either detected from the USB_ID pin or software selected from the UIMOD bit, according to the UIDE bit. This allows the USB_ID pin to be used as a general purpose I/O pin even when the USB interface is enabled.

By default, the USB_ID pin is selected (UIDE = 1) and the USB macro is in device mode (ID = 1), what corresponds to the case where no Mini-A plug is connected, i.e. no plug or a Mini-B plug is connected and the USB_ID pin is kept high by the internal pull-up resistor from the GPIO controller (which must be enabled if USB_ID is used).

The ID Transition interrupt (IDTI) is raised on each transition of the ID bit, i.e. when a Mini-A plug (host mode) is connected or disconnected. This does not occur when a Mini-B plug (device mode) is connected or disconnected.

The ID bit is effective whether the USB macro is enabled or not.

26.7.2 USB Device Operation

26.7.2.1 Introduction

In device mode, the USB controller supports full- and low-speed data transfers.

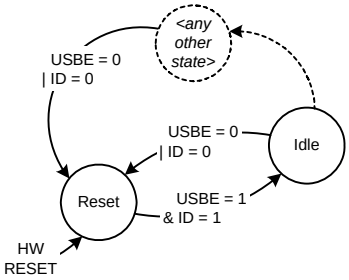
In addition to the default control endpoint, six endpoints are provided, which can be configured with the types isochronous, bulk or interrupt, as described in Table 26-1 on page 336.

The device mode starts in the Idle state, so the pad consumption is reduced to the minimum.

26.7.2.2 Power-On and Reset

Figure 26-13 describes the USB controller device mode main states.

Figure 26-13. Device Mode States



After a hardware reset, the USB controller device mode is in the Reset state. In this state:

- the macro clock is stopped in order to minimize power consumption (FRZCLK = 1);
- the internal registers of the device mode are reset;
- the endpoint banks are de-allocated;
- neither D+ nor D- is pulled up (DETACH = 1).

D+ or D- will be pulled up according to the selected speed as soon as the DETACH bit is cleared and VBus is present. See Section 26.7.1.5.1 on page 345 for further details.

When the USB macro is enabled (USBE = 1) in device mode (ID = 1), its device mode state goes to the Idle state with minimal power consumption. This does not require the USB clock to be activated.

The USB controller device mode can be disabled and reset at any time by disabling the USB macro (USBE = 0) or when host mode is engaged (ID = 0).

26.7.2.3 USB Reset

The USB bus reset is managed by hardware. It is initiated by a connected host.

When a USB reset is detected on the USB line, the following operations are performed by the controller:

- all the endpoints are disabled, except the default control endpoint;
- the default control endpoint is reset (see Section 26.7.2.4 on page 351 for more details);
- the data toggle sequence of the default control endpoint is cleared;
- at the end of the reset process, the End of Reset interrupt (EORST) is raised.

26.7.2.4 Endpoint Reset

An endpoint can be reset at any time by setting its EPRSTX bit in the UERST register. This is recommended before using an endpoint upon hardware reset or when a USB bus reset has been received. This resets:

- the internal state machine of this endpoint;
- the receive and transmit bank FIFO counters;
- all the registers of this endpoint (UECFGX, UESTAX, UECONX), except its configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE) and its Data Toggle Sequence bit-field (DTSEQ).

The endpoint configuration remains active and the endpoint is still enabled.

The endpoint reset may be associated with a clear of the data toggle sequence as an answer to the CLEAR_FEATURE USB request. This can be achieved by setting the RSTDT bit (by setting the RSTDTS bit).

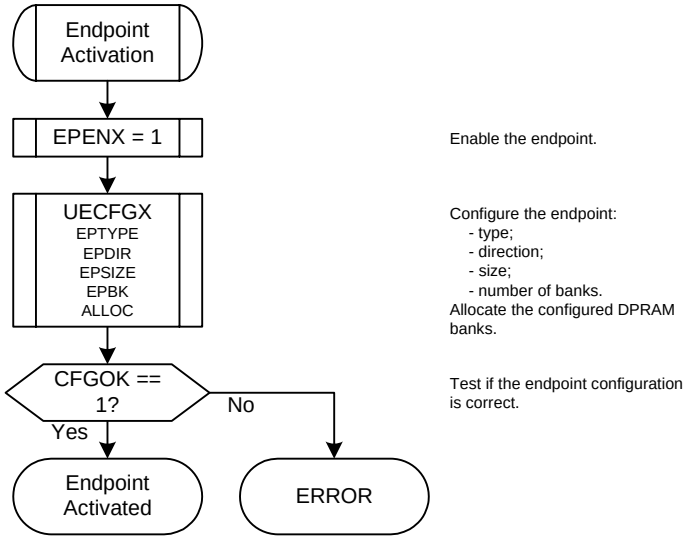
In the end, the firmware has to clear the EPRSTX bit to complete the reset operation and to start using the FIFO.

26.7.2.5 Endpoint Activation

The endpoint is maintained inactive and reset (see Section 26.7.2.4 on page 351 for more details) as long as it is disabled (EPENX = 0). The Data Toggle Sequence bit-field (DTSEQ) is also reset.

The algorithm represented on Figure 26-14 must be followed in order to activate an endpoint.

Figure 26-14. Endpoint Activation Algorithm



As long as the endpoint is not correctly configured (CFGOK = 0), the controller does not acknowledge the packets sent by the host to this endpoint.

The CFGOK bit is set by hardware only if the configured size and number of banks are correct compared to their maximal allowed values for the endpoint (see Table 26-1 on page 336) and to the maximal FIFO size (i.e. the DPRAM size).

See Section 26.7.1.6 on page 345 for more details about DPRAM management.

26.7.2.6 Address Setup

The USB device address is set up according to the USB protocol:

- after all kinds of resets, the USB device address is 0;
- the host starts a SETUP transaction with a SET_ADDRESS(addr) request;
- the firmware records this address into the UADD bit-field, leaving the ADDEN bit cleared, so the actual address is still 0;
- the firmware sends a zero-length IN packet from the control endpoint;
- the firmware enables the recorded USB device address by setting ADDEN.

Once the USB device address is configured, the controller filters the packets to only accept those targeting the address stored in UADD.

UADD and ADDEN shall not be written all at once.

UADD and ADDEN are cleared by hardware:

- on a hardware reset;
- when the USB macro is disabled (USBE = 0);
- when a USB reset is detected.

When UADD or ADDEN is cleared, the default device address 0 is used.

26.7.2.7 Suspend and Wake-Up

When an idle USB bus state has been detected for 3 ms, the controller raises the Suspend interrupt (SUSP). The firmware may then set the FRZCLK bit to reduce power consumption. The MCU can also enter the Idle or Frozen sleep mode to lower again power consumption.

To recover from the Suspend mode, the firmware should wait for the Wake-Up interrupt (WAKEUP), which is raised when a non-idle event is detected, then clear FRZCLK.

As the WAKEUP interrupt is raised when a non-idle event is detected, it can occur whether the controller is in the Suspend mode or not. The SUSP and WAKEUP interrupts are thus independent of each other except that one's flag is cleared by hardware when the other is raised.

26.7.2.8 Detach

The reset value of the DETACH bit is 1.

It is possible to initiate a device re-enumeration simply by setting then clearing DETACH.

DETACH acts on the pull-up connections of the D+ and D- pads. See Section 26.7.1.5.1 on page 345 for further details.

26.7.2.9 Remote Wake-Up

The Remote Wake-Up request (also known as Upstream Resume) is the only one the device may send on its own initiative, but the device should have beforehand been allowed to by a DEVICE_REMOTE_WAKEUP request from the host.

- First, the USB controller must have detected a "Suspend" state on the bus, i.e. the Remote Wake-Up request can only be sent after a SUSP interrupt has been raised.
- The firmware may then set the RMWKUP bit to send an upstream resume to the host for a remote wake-up. This will automatically be done by the controller after 5 ms of inactivity on the USB bus.

- When the controller sends the upstream resume, the Upstream Resume interrupt (UPRSM) is raised and SUSP is cleared by hardware.
- RMWKUP is cleared by hardware at the end of the upstream resume.
- If the controller detects a valid “End of Resume” signal from the host, the End of Resume interrupt (EORSM) is raised.

26.7.2.10 STALL Request

For each endpoint, the STALL management is performed using:

- the STALL Request bit (STALLRQ) to initiate a STALL request;
- the STALLed interrupt (STALLEDI) raised when a STALL handshake has been sent.

To answer the next request with a STALL handshake, STALLRQ has to be set by setting the STALLRQS bit. All following requests will be discarded (RXOUTI, etc. will not be set) and handshaked with a STALL until the STALLRQ bit is cleared, what is done by hardware when a new SETUP packet is received (for control endpoints) or when the STALLRQC bit is set.

Each time a STALL handshake is sent, the STALLEDI flag is set by the USB controller and the EPXINT interrupt is raised.

26.7.2.10.1 Special Considerations for Control Endpoints

If a SETUP packet is received into a control endpoint for which a STALL is requested, the Received SETUP interrupt (RXSTPI) is raised and STALLRQ and STALLEDI are cleared by hardware. The SETUP has to be ACKed.

This management simplifies the enumeration process management. If a command is not supported or contains an error, the firmware requests a STALL and can return to the main task, waiting for the next SETUP request.

26.7.2.10.2 STALL Handshake and Retry Mechanism

The retry mechanism has priority over the STALL handshake. A STALL handshake is sent if the STALLRQ bit is set and if there is no retry required.

26.7.2.11 Management of Control Endpoints

26.7.2.11.1 Overview

A SETUP request is always ACKed. When a new SETUP packet is received, the Received SETUP interrupt (RXSTPI) is raised, but not the Received OUT Data interrupt (RXOUTI).

The FIFOCON and RWALL bits are irrelevant for control endpoints. The firmware shall therefore never use them on these endpoints. When read, their value is always 0.

Control endpoints are managed using:

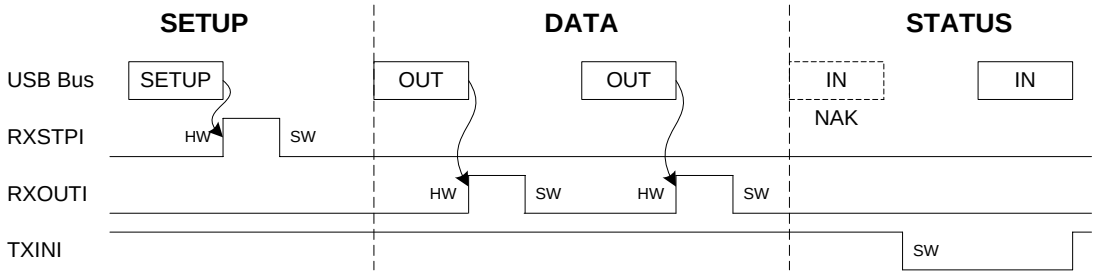
- the Received SETUP interrupt (RXSTPI) which is raised when a new SETUP packet is received and which shall be cleared by firmware to acknowledge the packet and to free the bank;
- the Received OUT Data interrupt (RXOUTI) which is raised when a new OUT packet is received and which shall be cleared by firmware to acknowledge the packet and to free the bank;
- the Transmitted IN Data interrupt (TXINI) which is raised when the current bank is ready to accept a new IN packet and which shall be cleared by firmware to send the packet.

26.7.2.11.2 Control Write

Figure 26-15 shows a control write transaction. During the status stage, the controller will not necessarily send a NAK on the first IN token:

- if the firmware knows the exact number of descriptor bytes that must be read, it can then anticipate the status stage and send a zero-length packet after the next IN token;
- or it can read the bytes and wait for the NAKed IN interrupt (NAKINI) which tells that all the bytes have been sent by the host and that the transaction is now in the status stage.

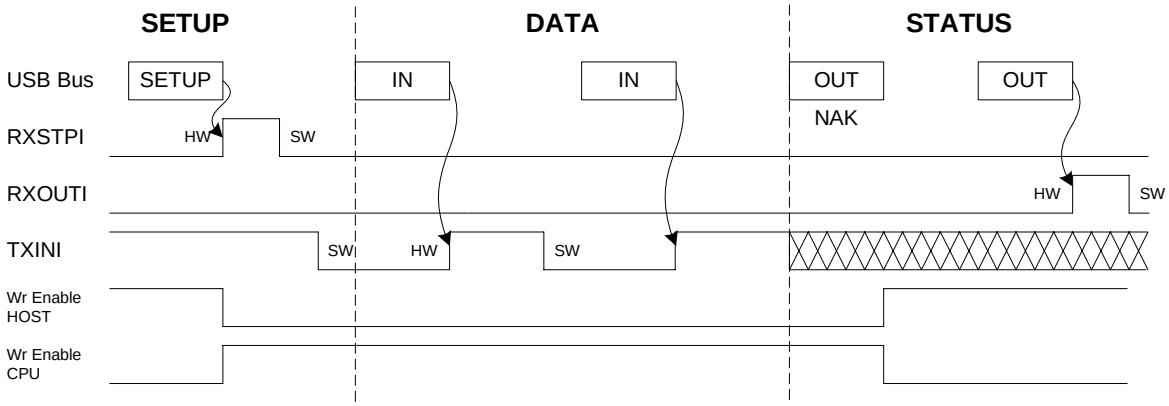
Figure 26-15. Control Write



26.7.2.11.3 Control Read

Figure 26-16 shows a control read transaction. The USB controller has to manage the simultaneous write requests from the CPU and the USB host.

Figure 26-16. Control Read



A NAK handshake is always generated on the first status stage command.

When the controller detects the status stage, all the data written by the CPU is lost and clearing TXINI has no effect.

The firmware checks if the transmission or the reception is complete.

The OUT retry is always ACKed. This reception sets RXOUTI and TXINI. Handle this with the following software algorithm:

```
set TXINI
wait for RXOUTI OR TXINI
if RXOUTI, then clear flag and return
```

```
if TXINI, then continue
```

Once the OUT status stage has been received, the USB controller waits for a SETUP request. The SETUP request has priority over any other request and has to be ACKed. This means that any other flag should be cleared and the FIFO reset when a SETUP is received.

The firmware has to take care of the fact that the byte counter is reset when a zero-length OUT packet is received.

26.7.2.12 Management of IN Endpoints

26.7.2.12.1 Overview

IN packets are sent by the USB device controller upon IN requests from the host. All the data can be written by the firmware which acknowledges or not the bank when it is full.

The endpoint must be configured first.

The TXINI bit is set by hardware at the same time as FIFOCON when the current bank is free. This triggers an EPXINT interrupt if TXINE = 1.

TXINI shall be cleared by software (by setting the TXINIC bit) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The firmware then writes into the FIFO and clears the FIFOCON bit to allow the USB controller to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The TXINI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

TXINI shall always be cleared before clearing FIFOCON.

The RWALL bit is set by hardware when the current bank is not full, i.e. the software can write further data into the FIFO.

Figure 26-17. Example of an IN Endpoint with 1 Data Bank

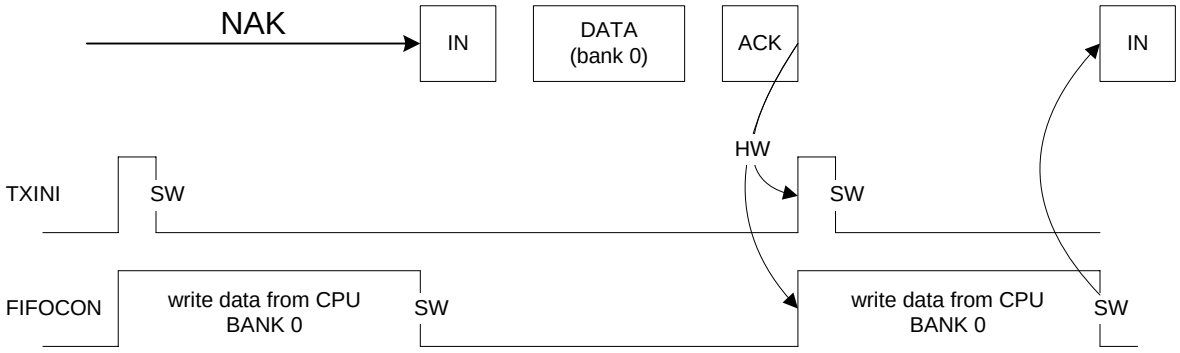
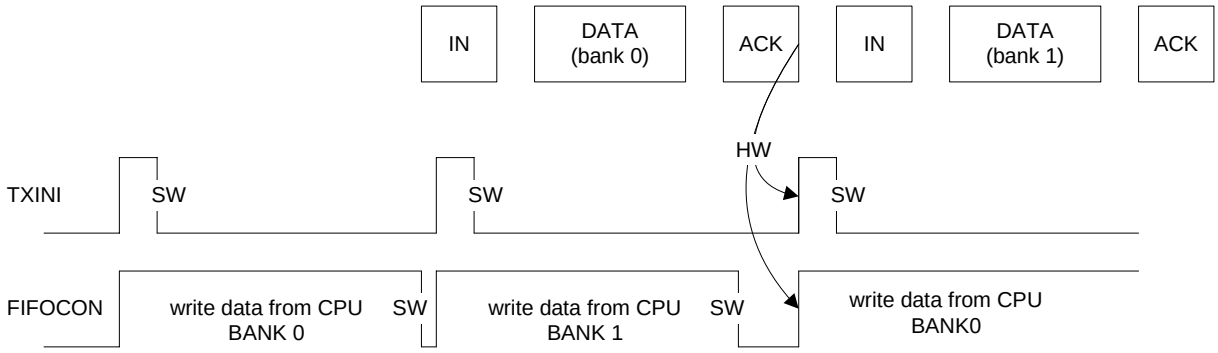


Figure 26-18. Example of an IN Endpoint with 2 Data Banks



26.7.2.12.2 Detailed Description

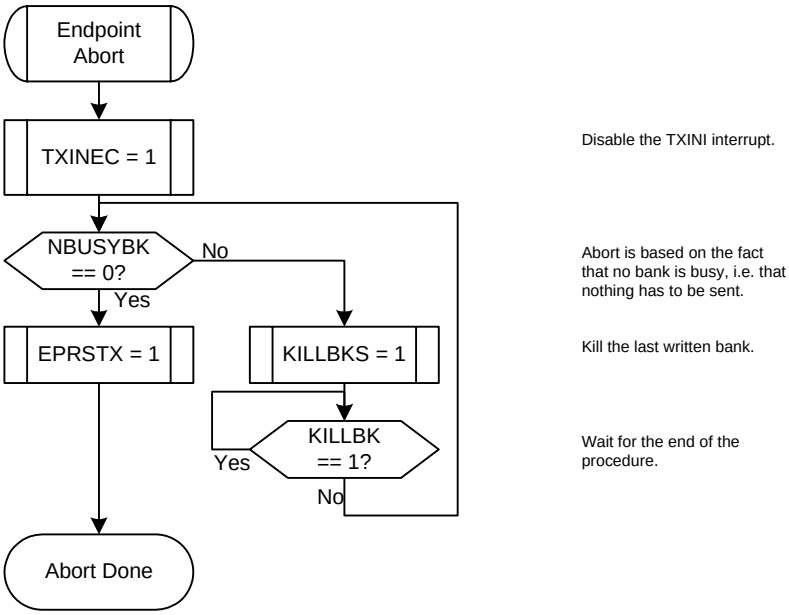
The data is written by the firmware, following the next flow:

- when the bank is empty, TXINI and FIFOCON are set, what triggers an EPXINT interrupt if TXINE = 1;
- the firmware acknowledges the interrupt by clearing TXINI;
- the firmware writes the data into the current bank by using the USB Pipe/Endpoint X FIFO Data register (USB_FIFOX_DATA), until all the data frame is written or the bank is full (in which case RWALL is cleared by hardware and BYCT reaches the endpoint size);
- the firmware allows the controller to send the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be written by the firmware while the previous one is being read by the host. Then, when the firmware clears FIFOCON, the following bank may already be free and TXINI is set immediately.

An “Abort” stage can be produced when a zero-length OUT packet is received during an IN stage of a control or isochronous IN transaction. The KILLBK bit is used to kill the last written bank. The best way to manage this abort is to apply the algorithm represented on Figure 26-19.

Figure 26-19. Abort Algorithm



26.7.2.13 Management of OUT Endpoints

26.7.2.13.1 Overview

OUT packets are sent by the host. All the data can be read by the firmware which acknowledges or not the bank when it is empty.

The endpoint must be configured first.

The RXOUTI bit is set by hardware at the same time as FIFOCON when the current bank is full. This triggers an EPXINT interrupt if RXOUTE = 1.

RXOUTI shall be cleared by software (by setting the RXOUTIC bit) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The firmware then reads from the FIFO and clears the FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The RXOUTI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

RXOUTI shall always be cleared before clearing FIFOCON.

The RWALL bit is set by hardware when the current bank is not empty, i.e. the software can read further data from the FIFO.

Figure 26-20. Example of an OUT Endpoint with 1 Data Bank

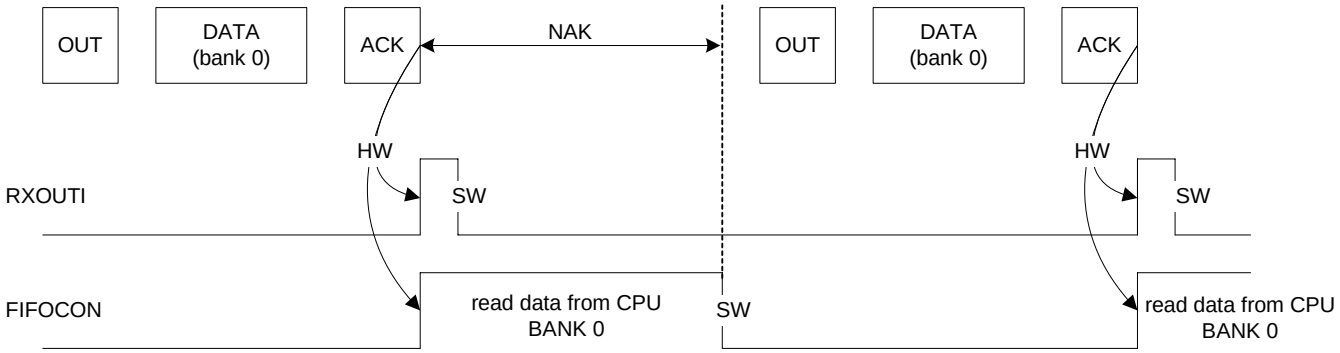
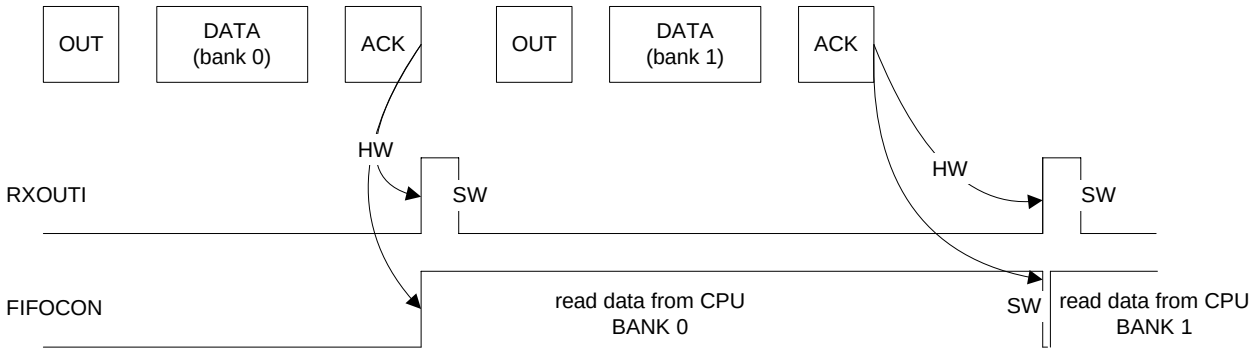


Figure 26-21. Example of an OUT Endpoint with 2 Data Banks



26.7.2.13.2 Detailed Description

The data is read by the firmware, following the next flow:

- when the bank is full, RXOUTI and FIFOCON are set, what triggers an EPXINT interrupt if RXOUTE = 1;
- the firmware acknowledges the interrupt by clearing RXOUTI;
- the firmware can read the byte count of the current bank from BYCT to know how many bytes to read, rather than polling RWALL;
- the firmware reads the data from the current bank by using the USB Pipe/Endpoint X FIFO Data register (USB_FIFOX_DATA), until all the expected data frame is read or the bank is empty (in which case RWALL is cleared by hardware and BYCT reaches 0);
- the firmware frees the bank and switches to the next bank (if any) by clearing FIFOCON.

If the endpoint uses several banks, the current one can be read by the firmware while the following one is being written by the host. Then, when the firmware clears FIFOCON, the following bank may already be ready and RXOUTI is set immediately.

26.7.2.14 Underflow

This error exists only for isochronous IN/OUT endpoints. It raises the Underflow interrupt (UNDERFI), what triggers an EPXINT interrupt if UNDERFE = 1.

An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USB controller.

An underflow can not occur during OUT stage on a CPU action, since the firmware may read only if the bank is not empty (RXOUTI = 1 or RWALL = 1).

An underflow can also occur during OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.

An underflow can not occur during IN stage on a CPU action, since the firmware may write only if the bank is not full (TXINI = 1 or RWALL = 1).

26.7.2.15 *Overflow*

This error exists for all endpoint types. It raises the Overflow interrupt (OVERFI), what triggers an EPXINT interrupt if OVERFE = 1.

An overflow can occur during OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the Received OUT Data interrupt (RXOUTI) is raised as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.

An underflow can not occur during IN stage on a CPU action, since the firmware may write only if the bank is not full (TXINI = 1 or RWALL = 1).

26.7.2.16 *CRC Error*

This error exists only for isochronous OUT endpoints. It raises the CRC Error interrupt (CRCERRI), what triggers an EPXINT interrupt if CRCERRE = 1.

A CRC error can occur during OUT stage if the USB controller detects a corrupted received packet. The OUT packet is stored in the bank as if no CRC error had occurred (RXOUTI is raised).

26.7.2.17 *Interrupts*

See the structure of the USB device interrupt system on Figure 26-6 on page 343.

There are two kinds of device interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

26.7.2.17.1 *Global Interrupts*

The processing device global interrupts are:

- the Suspend interrupt (SUSP);
- the Start of Frame interrupt (SOF) with no frame number CRC error (FNCERR = 0);
- the End of Reset interrupt (EORST);
- the Wake-Up interrupt (WAKEUP);
- the End of Resume interrupt (EORSM);
- the Upstream Resume interrupt (UPRSM);
- the Endpoint X interrupt (EPXINT);
- the DMA Channel X interrupt (DMAXINT).

The exception device global interrupts are:

- the Start of Frame interrupt (SOF) with a frame number CRC error (FNCERR = 1).

26.7.2.17.2 *Endpoint Interrupts*

The processing device endpoint interrupts are:

- the Transmitted IN Data interrupt (TXINI);
- the Received OUT Data interrupt (RXOUTI);
- the Received SETUP interrupt (RXSTPI);
- the Short Packet interrupt (SHORTPACKET);
- the Number of Busy Banks interrupt (NBUSYBK).

The exception device endpoint interrupts are:

- the Underflow interrupt (UNDERFI);
- the NAKed OUT interrupt (NAKOUTI);
- the NAKed IN interrupt (NAKINI);
- the Overflow interrupt (OVERFI);
- the STALLed interrupt (STALLEDI);
- the CRC Error interrupt (CRCERRI).

26.7.2.17.3 DMA Interrupts

The processing device DMA interrupts are:

- the End of USB Transfer Status interrupt (EOT_STA);
- the End of Channel Buffer Status interrupt (EOCH_BUFF_STA);
- the Descriptor Loaded Status interrupt (DESC_LD_STA).

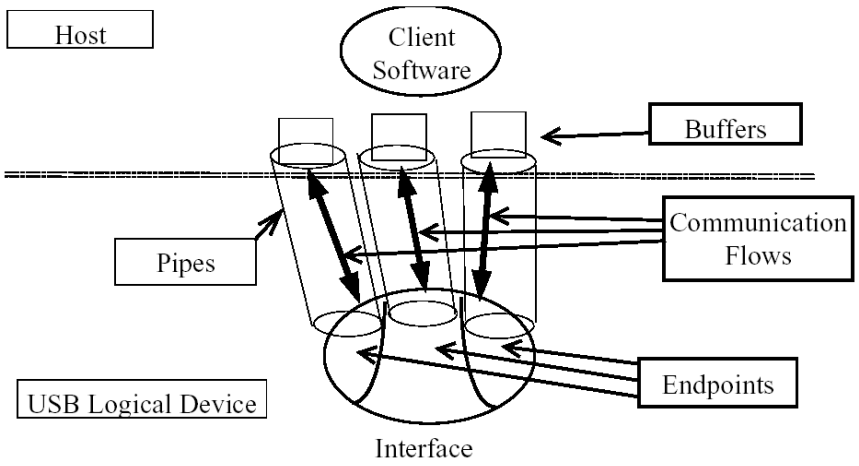
There is no exception device DMA interrupt.

26.7.3 USB Host Operation

26.7.3.1 Description of Pipes

For the USB controller in host mode, the term “pipe” is used instead of “endpoint” (used in device mode). A host pipe corresponds to a device endpoint, as described by the Figure 26-22 from the USB specification.

Figure 26-22. USB Communication Flow

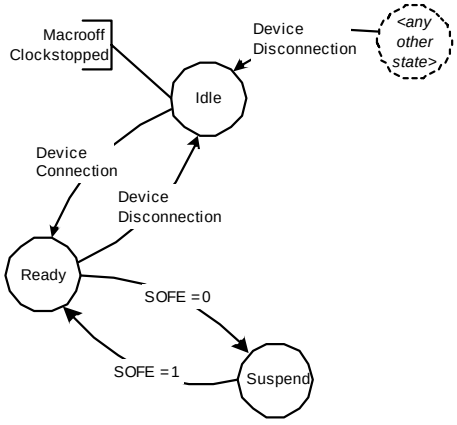


In host mode, the USB controller associates a pipe to a device endpoint, considering the device configuration descriptors.

26.7.3.2 Power-On and Reset

Figure 26-23 describes the USB controller host mode main states.

Figure 26-23. Host Mode States



After a hardware reset, the USB controller host mode is in the Reset state.

When the USB macro is enabled (USBE = 1) in host mode (ID = 0), its host mode state goes to the Idle state. In this state, the controller waits for device connection with minimal power consumption. The USB pad should be in the Idle state. Once a device is connected, the macro enters the Ready state, what does not require the USB clock to be activated.

The controller enters the Suspend state when the USB bus is in a “Suspend” state, i.e. when the host mode does not generate the “Start of Frame”. In this state, the USB consumption is minimal. The host mode exits the Suspend state when starting to generate the SOF over the USB line.

26.7.3.3 Device Detection

A device is detected by the USB controller host mode when D+ or D- is no longer tied low, i.e. when the device D+ or D- pull-up resistor is connected. To enable this detection, the host controller has to provide the VBus power supply to the device by setting the VBUSRQ bit (by setting the VBUSRQS bit).

The device disconnection is detected by the host controller when both D+ and D- are pulled down.

26.7.3.4 USB Reset

The USB controller sends a USB bus reset when the firmware sets the RESET bit. The USB Reset Sent interrupt (RSTI) is raised when the USB reset has been sent. In this case, all the pipes are disabled and de-allocated.

If the bus was previously in a “Suspend” state (SOFE = 0), the USB controller automatically switches it to the “Resume” state, the Host Wake-Up interrupt (HWUPI) is raised and the SOFE bit is set by hardware in order to generate SOFs immediately after the USB reset.

26.7.3.5 Pipe Reset

A pipe can be reset at any time by setting its PRSTX bit in the UPRST register. This is recommended before using a pipe upon hardware reset or when a USB bus reset has been sent. This resets:

- the internal state machine of this pipe;
- the receive and transmit bank FIFO counters;
- all the registers of this pipe (UPCFGX, UPSTAX, UPCONX), except its configuration (ALLOC, PBK, PSIZE, PTOKEN, PTYPE, PEPNUM, INTFRQ) and its Data Toggle Sequence bit-field (DTSEQ).

The pipe configuration remains active and the pipe is still enabled.

The pipe reset may be associated with a clear of the data toggle sequence. This can be achieved by setting the RSTDT bit (by setting the RSTDTS bit).

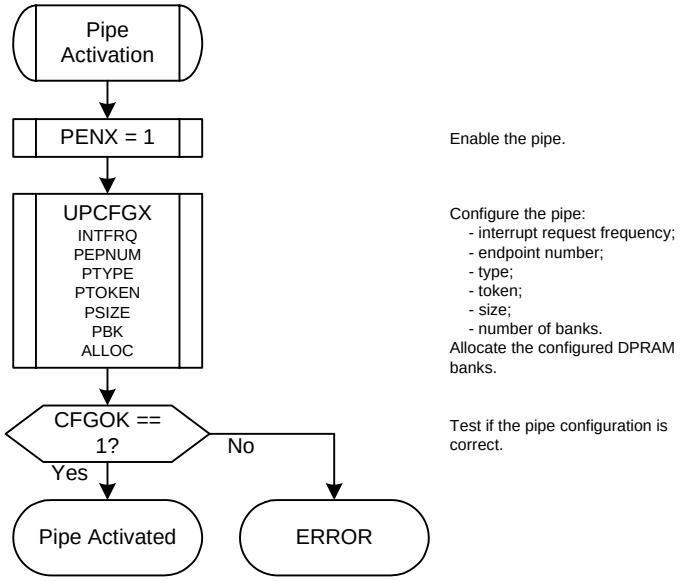
In the end, the firmware has to clear the PRSTX bit to complete the reset operation and to start using the FIFO.

26.7.3.6 Pipe Activation

The pipe is maintained inactive and reset (see Section 26.7.3.5 on page 362 for more details) as long as it is disabled (PENX = 0). The Data Toggle Sequence bit-field (DTSEQ) is also reset.

The algorithm represented on Figure 26-24 must be followed in order to activate a pipe.

Figure 26-24. Pipe Activation Algorithm



As long as the pipe is not correctly configured (CFGOK = 0), the controller can not send packets to the device through this pipe.

The CFGOK bit is set by hardware only if the configured size and number of banks are correct compared to their maximal allowed values for the pipe (see Table 26-1 on page 336) and to the maximal FIFO size (i.e. the DPRAM size).

See Section 26.7.1.6 on page 345 for more details about DPRAM management.

Once the pipe is correctly configured (CFGOK = 1), only the PTOKEN and INTFRQ bit-fields can be modified by software. INTFRQ is meaningless for non-interrupt pipes.

When starting an enumeration, the firmware gets the device descriptor by sending a GET_DESCRIPTOR USB request. This descriptor contains the maximal packet size of the device default control endpoint (bMaxPacketSize0) and the firmware re-configures the size of the default control pipe with this size parameter.

26.7.3.7 Address Setup

Once the device has answered the first host requests with the default device address 0, the host assigns a new address to the device. The host controller has to send a USB reset to the device and to send a SET_ADDRESS(addr) SETUP request with the new address to be used by the device. Once this SETUP transaction is over, the firmware writes the new address into the HADDR bit-field. All following requests, on all pipes, will be performed using this new address.

When the host controller sends a USB reset, the HADDR bit-field is reset by hardware and the following host requests will be performed using the default device address 0.

26.7.3.8 Remote Wake-Up

The controller host mode enters the Suspend state when the SOFE bit is cleared. No more “Start of Frame” is sent on the USB bus and the USB device enters the Suspend state 3 ms later.

The device awakes the host by sending an Upstream Resume (Remote Wake-Up feature). When the host controller detects a non-idle state on the USB bus, it raises the Host Wake-Up

interrupt (HWUPI). If the non-idle bus state corresponds to an Upstream Resume (K state), the Upstream Resume Received interrupt (RXRSMI) is raised. The firmware has to generate a Downstream Resume within 1 ms and for at least 20 ms by setting the RESUME bit. It is mandatory to set SOFE before setting RESUME to enter the Ready state, else RESUME will have no effect.

26.7.3.9 Management of Control Pipes

A control transaction is composed of three stages:

- SETUP;
- Data (IN or OUT);
- Status (OUT or IN).

The firmware has to change the pipe token according to each stage.

For the control pipe, and only for it, each token is assigned a specific initial data toggle sequence:

- SETUP: Data0;
- IN: Data1;
- OUT: Data1.

26.7.3.10 Management of IN Pipes

IN packets are sent by the USB device controller upon IN requests from the host. All the data can be read by the firmware which acknowledges or not the bank when it is empty.

The pipe must be configured first.

When the host requires data from the device, the firmware has to select beforehand the IN request mode with the INMODE bit:

- when INMODE is cleared, the USB controller will perform (INRQ + 1) IN requests before freezing the pipe;
- when INMODE is set, the USB controller will perform IN requests endlessly when the pipe is not frozen by the firmware.

The generation of IN requests starts when the pipe is unfrozen (PFREEZE = 0).

The RXINI bit is set by hardware at the same time as FIFOCON when the current bank is full. This triggers a PXINT interrupt if RXINE = 1.

RXINI shall be cleared by software (by setting the RXINIC bit) to acknowledge the interrupt, what has no effect on the pipe FIFO.

The firmware then reads from the FIFO and clears the FIFOCON bit to free the bank. If the IN pipe is composed of multiple banks, this also switches to the next bank. The RXINI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

RXINI shall always be cleared before clearing FIFOCON.

The RWALL bit is set by hardware when the current bank is not empty, i.e. the software can read further data from the FIFO.

Figure 26-25. Example of an IN Pipe with 1 Data Bank

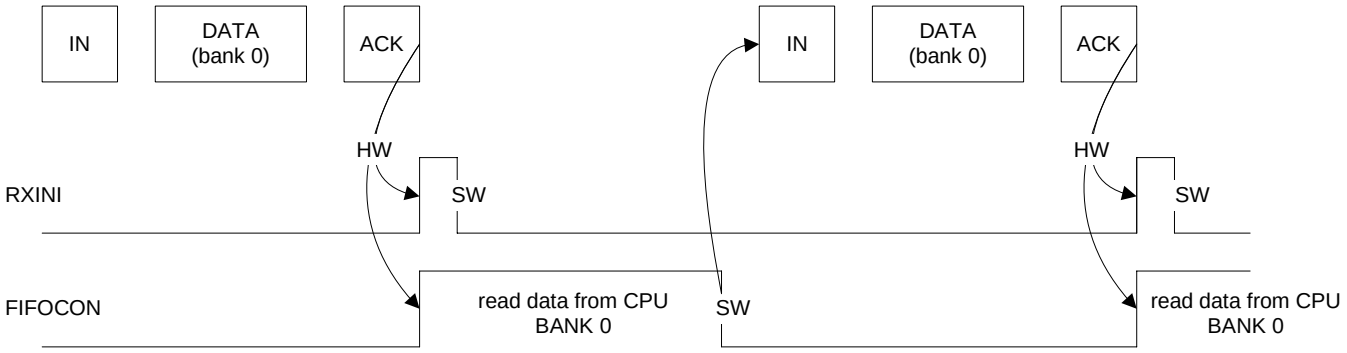
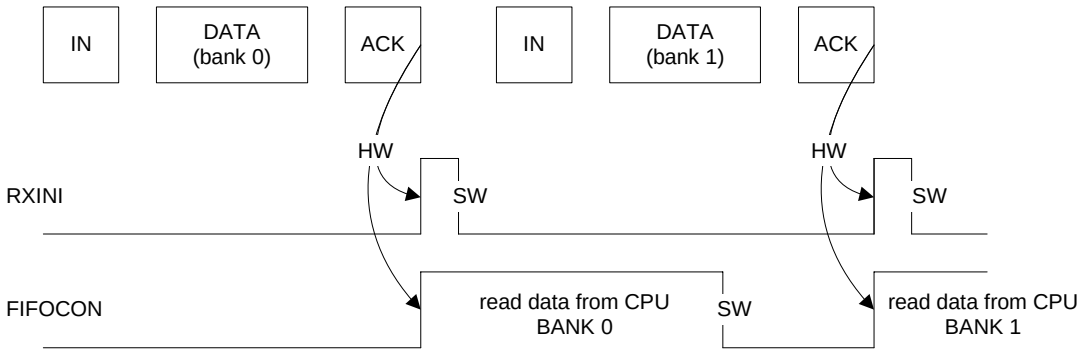


Figure 26-26. Example of an IN Pipe with 2 Data Banks



26.7.3.11 Management of OUT Pipes

OUT packets are sent by the host. All the data can be written by the firmware which acknowledges or not the bank when it is full.

The pipe must be configured and unfrozen first.

The TXOUTI bit is set by hardware at the same time as FIFOCON when the current bank is free. This triggers a PXINT interrupt if TXOUTE = 1.

TXOUTI shall be cleared by software (by setting the TXOUTIC bit) to acknowledge the interrupt, what has no effect on the pipe FIFO.

The firmware then writes into the FIFO and clears the FIFOCON bit to allow the USB controller to send the data. If the OUT pipe is composed of multiple banks, this also switches to the next bank. The TXOUTI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

TXOUTI shall always be cleared before clearing FIFOCON.

The RWALL bit is set by hardware when the current bank is not full, i.e. the software can write further data into the FIFO.

Note that if the firmware decides to switch to the Suspend state (by clearing the SOFE bit) while a bank is ready to be sent, the USB controller automatically exits this state and the bank is sent.

Figure 26-27. Example of an OUT Pipe with 1 Data Bank

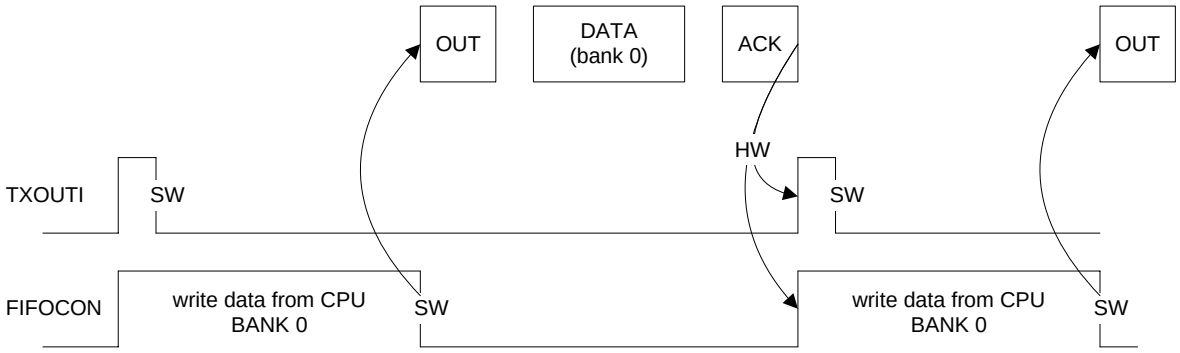


Figure 26-28. Example of an OUT Pipe with 2 Data Banks and no Bank Switching Delay

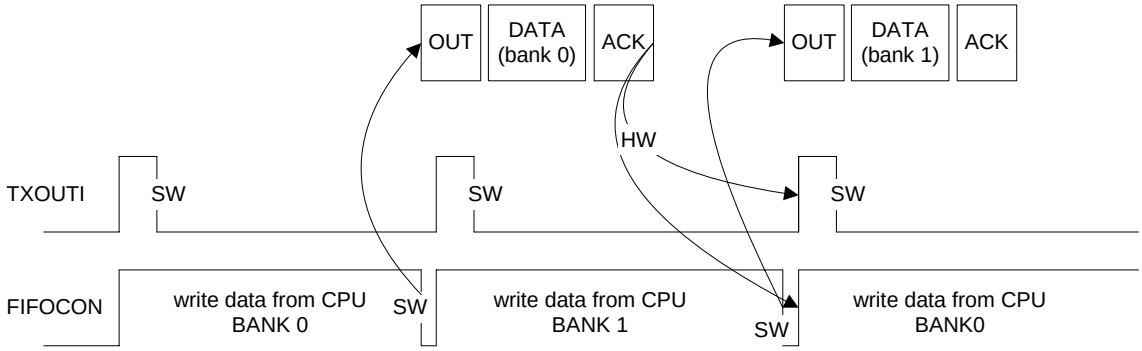
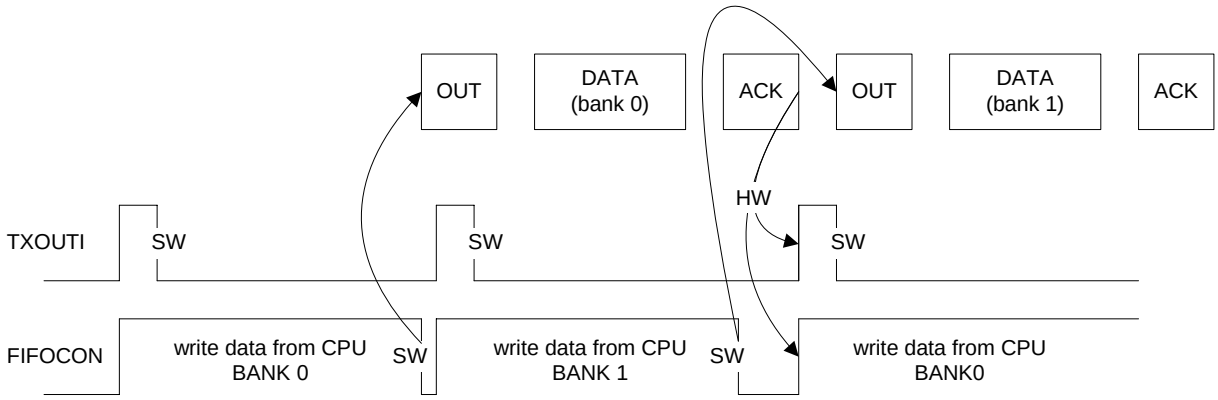


Figure 26-29. Example of an OUT Pipe with 2 Data Banks and a Bank Switching Delay



26.7.3.12 CRC Error

This error exists only for isochronous IN pipes. It raises the CRC Error interrupt (CRCERRI), what triggers a PXINT interrupt if CRCERRE = 1.

A CRC error can occur during IN stage if the USB controller detects a corrupted received packet. The IN packet is stored in the bank as if no CRC error had occurred (RXINI is raised).

26.7.3.13 Interrupts

See the structure of the USB host interrupt system on Figure 26-6 on page 343.

There are two kinds of host interrupts: processing, i.e. their generation is part of the normal processing, and exception, i.e. errors (not related to CPU exceptions).

26.7.3.13.1 Global Interrupts

The processing host global interrupts are:

- the Device Connection interrupt (DCONNI);
- the Device Disconnection interrupt (DDISCI);
- the USB Reset Sent interrupt (RSTI);
- the Downstream Resume Sent interrupt (RSMEDI);
- the Upstream Resume Received interrupt (RXRSMI);
- the Host Start of Frame interrupt (HSOFI);
- the Host Wake-Up interrupt (HWUPI);
- the Pipe X interrupt (PXINT);
- the DMA Channel X interrupt (DMAXINT).

There is no exception host global interrupt.

26.7.3.13.2 Pipe Interrupts

The processing host pipe interrupts are:

- the Received IN Data interrupt (RXINI);
- the Transmitted OUT Data interrupt (TXOUTI);
- the Transmitted SETUP interrupt (TXSTPI);
- the Short Packet interrupt (SHORTPACKETI);
- the Number of Busy Banks interrupt (NBUSYBK).

The exception host pipe interrupts are:

- the Underflow interrupt (UNDERFI);
- the Pipe Error interrupt (PERRI);
- the NAKed interrupt (NAKEDI);
- the Overflow interrupt (OVERFI);
- the Received STALLed interrupt (RXSTALLDI);
- the CRC Error interrupt (CRCERRI).

26.7.3.13.3 DMA Interrupts

The processing host DMA interrupts are:

- the End of USB Transfer Status interrupt (EOT_STA);
- the End of Channel Buffer Status interrupt (EOCH_BUFF_STA);
- the Descriptor Loaded Status interrupt (DESC_LD_STA).

There is no exception host DMA interrupt.

26.7.4 USB DMA Operation

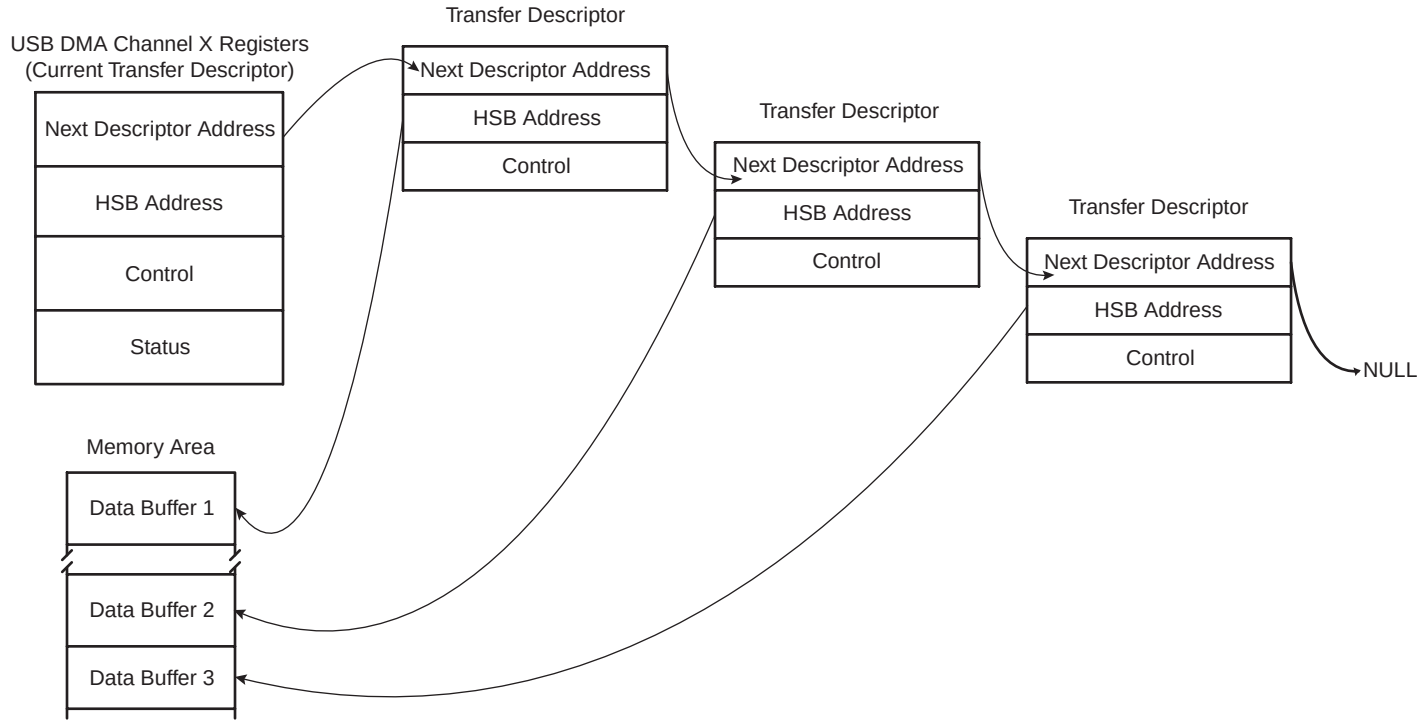
USB packets of any length may be transferred when required by the USB controller. These transfers always feature sequential addressing. These two characteristics mean that in case of high USB controller throughput, both HSB ports will benefit from “incrementing burst of unspecified length” since the average access latency of HSB slaves can then be reduced.

The DMA uses word “incrementing burst of unspecified length” of up to 256 beats for both data transfers and channel descriptor loading. A burst may last on the HSB busses for the duration of a whole USB packet transfer, unless otherwise broken by the HSB arbitration or the HSB 1 kbyte boundary crossing.

Packet data HSB bursts may be locked on a DMA buffer basis for drastic overall HSB bus bandwidth performance boost with paged memories. This is because these memories row (or bank) changes, which are very clock-cycle consuming, will then likely not occur or occur once instead of dozens of times during a single big USB packet DMA transfer in case other HSB masters address the memory. This means up to 128 words single cycle unbroken HSB bursts for bulk pipes/endpoints and 256 words single cycle unbroken bursts for isochronous pipes/endpoints. This maximal burst length is then controlled by the lowest programmed USB pipe/endpoint size (PSIZE/EPSIZE) and DMA channel byte length (CH_BYTE_LENGTH).

The USB controller average throughput may be up to nearly 1.5 Mbyte/s. Its average access latency decreases as burst length increases due to the 0 wait-state side effect of unchanged pipe/endpoint. Word access allows reducing the HSB bandwidth required for the USB by 4 compared to native byte access. If at least 0 wait-state word burst capability is also provided by the other DMA HSB bus slaves, each of both DMA HSB busses need less than 1.1% bandwidth allocation for full USB bandwidth usage at 33 MHz, and less than 0.6% at 66 MHz.

Figure 26-30. Example of DMA Chained List



26.8 USB User Interface

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Device General Control Register	UDCON	Read/Write	0x00000100
0x0004	Device Global Interrupt Register	UDINT	Read-Only	0x00000000
0x0008	Device Global Interrupt Clear Register	UDINTCLR	Write-Only	0x00000000
0x000C	Device Global Interrupt Set Register	UDINTSET	Write-Only	0x00000000
0x0010	Device Global Interrupt Enable Register	UDINTE	Read-Only	0x00000000
0x0014	Device Global Interrupt Enable Clear Register	UDINTECLR	Write-Only	0x00000000
0x0018	Device Global Interrupt Enable Set Register	UDINTESET	Write-Only	0x00000000
0x001C	Endpoint Enable/Reset Register	UERST	Read/Write	0x00000000
0x0020	Device Frame Number Register	UDFNUM	Read-Only	0x00000000
0x0024 - 0x00FC	Reserved	–	–	–
0x0100	Endpoint 0 Configuration Register	UECFG0	Read/Write	0x00000000
0x0104	Endpoint 1 Configuration Register	UECFG1	Read/Write	0x00000000
0x0108	Endpoint 2 Configuration Register	UECFG2	Read/Write	0x00000000
0x010C	Endpoint 3 Configuration Register	UECFG3	Read/Write	0x00000000
0x0110	Endpoint 4 Configuration Register	UECFG4	Read/Write	0x00000000
0x0114	Endpoint 5 Configuration Register	UECFG5	Read/Write	0x00000000
0x0118	Endpoint 6 Configuration Register	UECFG6	Read/Write	0x00000000
0x011C - 0x012C	Reserved	–	–	–
0x0130	Endpoint 0 Status Register	UESTA0	Read-Only	0x00000100
0x0134	Endpoint 1 Status Register	UESTA1	Read-Only	0x00000100
0x0138	Endpoint 2 Status Register	UESTA2	Read-Only	0x00000100
0x013C	Endpoint 3 Status Register	UESTA3	Read-Only	0x00000100
0x0140	Endpoint 4 Status Register	UESTA4	Read-Only	0x00000100
0x0144	Endpoint 5 Status Register	UESTA5	Read-Only	0x00000100
0x0148	Endpoint 6 Status Register	UESTA6	Read-Only	0x00000100
0x014C - 0x015C	Reserved	–	–	–
0x0160	Endpoint 0 Status Clear Register	UESTA0CLR	Write-Only	0x00000000
0x0164	Endpoint 1 Status Clear Register	UESTA1CLR	Write-Only	0x00000000
0x0168	Endpoint 2 Status Clear Register	UESTA2CLR	Write-Only	0x00000000
0x016C	Endpoint 3 Status Clear Register	UESTA3CLR	Write-Only	0x00000000
0x0170	Endpoint 4 Status Clear Register	UESTA4CLR	Write-Only	0x00000000
0x0174	Endpoint 5 Status Clear Register	UESTA5CLR	Write-Only	0x00000000
0x0178	Endpoint 6 Status Clear Register	UESTA6CLR	Write-Only	0x00000000
0x017C - 0x018C	Reserved	–	–	–
0x0190	Endpoint 0 Status Set Register	UESTA0SET	Write-Only	0x00000000

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0194	Endpoint 1 Status Set Register	UESTA1SET	Write-Only	0x00000000
0x0198	Endpoint 2 Status Set Register	UESTA2SET	Write-Only	0x00000000
0x019C	Endpoint 3 Status Set Register	UESTA3SET	Write-Only	0x00000000
0x01A0	Endpoint 4 Status Set Register	UESTA4SET	Write-Only	0x00000000
0x01A4	Endpoint 5 Status Set Register	UESTA5SET	Write-Only	0x00000000
0x01A8	Endpoint 6 Status Set Register	UESTA6SET	Write-Only	0x00000000
0x01AC - 0x01BC	Reserved	–	–	–
0x01C0	Endpoint 0 Control Register	UECON0	Read-Only	0x00000000
0x01C4	Endpoint 1 Control Register	UECON1	Read-Only	0x00000000
0x01C8	Endpoint 2 Control Register	UECON2	Read-Only	0x00000000
0x01CC	Endpoint 3 Control Register	UECON3	Read-Only	0x00000000
0x01D0	Endpoint 4 Control Register	UECON4	Read-Only	0x00000000
0x01D4	Endpoint 5 Control Register	UECON5	Read-Only	0x00000000
0x01D8	Endpoint 6 Control Register	UECON6	Read-Only	0x00000000
0x01DC - 0x01EC	Reserved	–	–	–
0x01F0	Endpoint 0 Control Set Register	UECON0SET	Write-Only	0x00000000
0x01F4	Endpoint 1 Control Set Register	UECON1SET	Write-Only	0x00000000
0x01F8	Endpoint 2 Control Set Register	UECON2SET	Write-Only	0x00000000
0x01FC	Endpoint 3 Control Set Register	UECON3SET	Write-Only	0x00000000
0x0200	Endpoint 4 Control Set Register	UECON4SET	Write-Only	0x00000000
0x0204	Endpoint 5 Control Set Register	UECON5SET	Write-Only	0x00000000
0x0208	Endpoint 6 Control Set Register	UECON6SET	Write-Only	0x00000000
0x020C - 0x021C	Reserved	–	–	–
0x0220	Endpoint 0 Control Clear Register	UECON0CLR	Write-Only	0x00000000
0x0224	Endpoint 1 Control Clear Register	UECON1CLR	Write-Only	0x00000000
0x0228	Endpoint 2 Control Clear Register	UECON2CLR	Write-Only	0x00000000
0x022C	Endpoint 3 Control Clear Register	UECON3CLR	Write-Only	0x00000000
0x0230	Endpoint 4 Control Clear Register	UECON4CLR	Write-Only	0x00000000
0x0234	Endpoint 5 Control Clear Register	UECON5CLR	Write-Only	0x00000000
0x0238	Endpoint 6 Control Clear Register	UECON6CLR	Write-Only	0x00000000
0x023C - 0x030C	Reserved	–	–	–
0x0310	Device DMA Channel 1 Next Descriptor Address Register	UDDMA1_ NEXTDESC	Read/Write	0x00000000
0x0314	Device DMA Channel 1 HSB Address Register	UDDMA1_ ADDR	Read/Write	0x00000000
0x0318	Device DMA Channel 1 Control Register	UDDMA1_ CONTROL	Read/Write	0x00000000

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x031C	Device DMA Channel 1 Status Register	UDDMA1_ STATUS	Read/Write	0x00000000
0x0320	Device DMA Channel 2 Next Descriptor Address Register	UDDMA2_ NEXTDESC	Read/Write	0x00000000
0x0324	Device DMA Channel 2 HSB Address Register	UDDMA2_ ADDR	Read/Write	0x00000000
0x0328	Device DMA Channel 2 Control Register	UDDMA2_ CONTROL	Read/Write	0x00000000
0x032C	Device DMA Channel 2 Status Register	UDDMA2_ STATUS	Read/Write	0x00000000
0x0330	Device DMA Channel 3 Next Descriptor Address Register	UDDMA3_ NEXTDESC	Read/Write	0x00000000
0x0334	Device DMA Channel 3 HSB Address Register	UDDMA3_ ADDR	Read/Write	0x00000000
0x0338	Device DMA Channel 3 Control Register	UDDMA3_ CONTROL	Read/Write	0x00000000
0x033C	Device DMA Channel 3 Status Register	UDDMA3_ STATUS	Read/Write	0x00000000
0x0340	Device DMA Channel 4 Next Descriptor Address Register	UDDMA4_ NEXTDESC	Read/Write	0x00000000
0x0344	Device DMA Channel 4 HSB Address Register	UDDMA4_ ADDR	Read/Write	0x00000000
0x0348	Device DMA Channel 4 Control Register	UDDMA4_ CONTROL	Read/Write	0x00000000
0x034C	Device DMA Channel 4 Status Register	UDDMA4_ STATUS	Read/Write	0x00000000
0x0350	Device DMA Channel 5 Next Descriptor Address Register	UDDMA5_ NEXTDESC	Read/Write	0x00000000
0x0354	Device DMA Channel 5 HSB Address Register	UDDMA5_ ADDR	Read/Write	0x00000000
0x0358	Device DMA Channel 5 Control Register	UDDMA5_ CONTROL	Read/Write	0x00000000
0x035C	Device DMA Channel 5 Status Register	UDDMA5_ STATUS	Read/Write	0x00000000
0x0360	Device DMA Channel 6 Next Descriptor Address Register	UDDMA6_ NEXTDESC	Read/Write	0x00000000
0x0364	Device DMA Channel 6 HSB Address Register	UDDMA6_ ADDR	Read/Write	0x00000000
0x0368	Device DMA Channel 6 Control Register	UDDMA6_ CONTROL	Read/Write	0x00000000
0x036C	Device DMA Channel 6 Status Register	UDDMA6_ STATUS	Read/Write	0x00000000
0x0370 - 0x03FC	Reserved	–	–	–

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0400	Host General Control Register	UHCON	Read/Write	0x00000000
0x0404	Host Global Interrupt Register	UHINT	Read-Only	0x00000000
0x0408	Host Global Interrupt Clear Register	UHINTCLR	Write-Only	0x00000000
0x040C	Host Global Interrupt Set Register	UHINTSET	Write-Only	0x00000000
0x0410	Host Global Interrupt Enable Register	UHINTE	Read-Only	0x00000000
0x0414	Host Global Interrupt Enable Clear Register	UHINTECLR	Write-Only	0x00000000
0x0418	Host Global Interrupt Enable Set Register	UHINTESET	Write-Only	0x00000000
0x0041C	Pipe Enable/Reset Register	UPRST	Read/Write	0x00000000
0x0420	Host Frame Number Register	UHFNUM	Read/Write	0x00000000
0x0424 - 0x04FC	Reserved	–	–	–
0x0500	Pipe 0 Configuration Register	UPCFG0	Read/Write	0x00000000
0x0504	Pipe 1 Configuration Register	UPCFG1	Read/Write	0x00000000
0x0508	Pipe 2 Configuration Register	UPCFG2	Read/Write	0x00000000
0x050C	Pipe 3 Configuration Register	UPCFG3	Read/Write	0x00000000
0x0510	Pipe 4 Configuration Register	UPCFG4	Read/Write	0x00000000
0x0514	Pipe 5 Configuration Register	UPCFG5	Read/Write	0x00000000
0x0518	Pipe 6 Configuration Register	UPCFG6	Read/Write	0x00000000
0x051C - 0x052C	Reserved	–	–	–
0x0530	Pipe 0 Status Register	UPSTA0	Read-Only	0x00000000
0x0534	Pipe 1 Status Register	UPSTA1	Read-Only	0x00000000
0x0538	Pipe 2 Status Register	UPSTA2	Read-Only	0x00000000
0x053C	Pipe 3 Status Register	UPSTA3	Read-Only	0x00000000
0x0540	Pipe 4 Status Register	UPSTA4	Read-Only	0x00000000
0x0544	Pipe 5 Status Register	UPSTA5	Read-Only	0x00000000
0x0548	Pipe 6 Status Register	UPSTA6	Read-Only	0x00000000
0x054C - 0x055C	Reserved	–	–	–
0x0560	Pipe 0 Status Clear Register	UPSTA0CLR	Write-Only	0x00000000
0x0564	Pipe 1 Status Clear Register	UPSTA1CLR	Write-Only	0x00000000
0x0568	Pipe 2 Status Clear Register	UPSTA2CLR	Write-Only	0x00000000
0x056C	Pipe 3 Status Clear Register	UPSTA3CLR	Write-Only	0x00000000
0x0570	Pipe 4 Status Clear Register	UPSTA4CLR	Write-Only	0x00000000
0x0574	Pipe 5 Status Clear Register	UPSTA5CLR	Write-Only	0x00000000
0x0578	Pipe 6 Status Clear Register	UPSTA6CLR	Write-Only	0x00000000
0x057C - 0x058C	Reserved	–	–	–
0x0590	Pipe 0 Status Set Register	UPSTA0SET	Write-Only	0x00000000
0x0594	Pipe 1 Status Set Register	UPSTA1SET	Write-Only	0x00000000

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0598	Pipe 2 Status Set Register	UPSTA2SET	Write-Only	0x00000000
0x059C	Pipe 3 Status Set Register	UPSTA3SET	Write-Only	0x00000000
0x05A0	Pipe 4 Status Set Register	UPSTA4SET	Write-Only	0x00000000
0x05A4	Pipe 5 Status Set Register	UPSTA5SET	Write-Only	0x00000000
0x05A8	Pipe 6 Status Set Register	UPSTA6SET	Write-Only	0x00000000
0x05AC - 0x05BC	Reserved	–	–	–
0x05C0	Pipe 0 Control Register	UPCON0	Read-Only	0x00000000
0x05C4	Pipe 1 Control Register	UPCON1	Read-Only	0x00000000
0x05C8	Pipe 2 Control Register	UPCON2	Read-Only	0x00000000
0x05CC	Pipe 3 Control Register	UPCON3	Read-Only	0x00000000
0x05D0	Pipe 4 Control Register	UPCON4	Read-Only	0x00000000
0x05D4	Pipe 5 Control Register	UPCON5	Read-Only	0x00000000
0x05D8	Pipe 6 Control Register	UPCON6	Read-Only	0x00000000
0x05DC - 0x05EC	Reserved	–	–	–
0x05F0	Pipe 0 Control Set Register	UPCON0SET	Write-Only	0x00000000
0x05F4	Pipe 1 Control Set Register	UPCON1SET	Write-Only	0x00000000
0x05F8	Pipe 2 Control Set Register	UPCON2SET	Write-Only	0x00000000
0x05FC	Pipe 3 Control Set Register	UPCON3SET	Write-Only	0x00000000
0x0600	Pipe 4 Control Set Register	UPCON4SET	Write-Only	0x00000000
0x0604	Pipe 5 Control Set Register	UPCON5SET	Write-Only	0x00000000
0x0608	Pipe 6 Control Set Register	UPCON6SET	Write-Only	0x00000000
0x060C - 0x061C	Reserved	–	–	–
0x0620	Pipe 0 Control Clear Register	UPCON0CLR	Write-Only	0x00000000
0x0624	Pipe 1 Control Clear Register	UPCON1CLR	Write-Only	0x00000000
0x0628	Pipe 2 Control Clear Register	UPCON2CLR	Write-Only	0x00000000
0x062C	Pipe 3 Control Clear Register	UPCON3CLR	Write-Only	0x00000000
0x0630	Pipe 4 Control Clear Register	UPCON4CLR	Write-Only	0x00000000
0x0634	Pipe 5 Control Clear Register	UPCON5CLR	Write-Only	0x00000000
0x0638	Pipe 6 Control Clear Register	UPCON6CLR	Write-Only	0x00000000
0x063C - 0x064C	Reserved	–	–	–
0x0650	Pipe 0 IN Request Register	UPINRQ0	Read/Write	0x00000000
0x0654	Pipe 1 IN Request Register	UPINRQ1	Read/Write	0x00000000
0x0658	Pipe 2 IN Request Register	UPINRQ2	Read/Write	0x00000000
0x065C	Pipe 3 IN Request Register	UPINRQ3	Read/Write	0x00000000
0x0660	Pipe 4 IN Request Register	UPINRQ4	Read/Write	0x00000000
0x0664	Pipe 5 IN Request Register	UPINRQ5	Read/Write	0x00000000

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0668	Pipe 6 IN Request Register	UPINRQ6	Read/Write	0x00000000
0x066C - 0x067C	Reserved	–	–	–
0x0680	Pipe 0 Error Register	UPERR0	Read/Write	0x00000000
0x0684	Pipe 1 Error Register	UPERR1	Read/Write	0x00000000
0x0688	Pipe 2 Error Register	UPERR2	Read/Write	0x00000000
0x068C	Pipe 3 Error Register	UPERR3	Read/Write	0x00000000
0x0690	Pipe 4 Error Register	UPERR4	Read/Write	0x00000000
0x0694	Pipe 5 Error Register	UPERR5	Read/Write	0x00000000
0x0698	Pipe 6 Error Register	UPERR6	Read/Write	0x00000000
0x069C - 0x070C	Reserved	–	–	–
0x0710	Host DMA Channel 1 Next Descriptor Address Register	UHDMA1_ NEXTDESC	Read/Write	0x00000000
0x0714	Host DMA Channel 1 HSB Address Register	UHDMA1_ ADDR	Read/Write	0x00000000
0x0718	Host DMA Channel 1 Control Register	UHDMA1_ CONTROL	Read/Write	0x00000000
0x071C	Host DMA Channel 1 Status Register	UHDMA1_ STATUS	Read/Write	0x00000000
0x0720	Host DMA Channel 2 Next Descriptor Address Register	UHDMA2_ NEXTDESC	Read/Write	0x00000000
0x0724	Host DMA Channel 2 HSB Address Register	UHDMA2_ ADDR	Read/Write	0x00000000
0x0728	Host DMA Channel 2 Control Register	UHDMA2_ CONTROL	Read/Write	0x00000000
0x072C	Host DMA Channel 2 Status Register	UHDMA2_ STATUS	Read/Write	0x00000000
0x0730	Host DMA Channel 3 Next Descriptor Address Register	UHDMA3_ NEXTDESC	Read/Write	0x00000000
0x0734	Host DMA Channel 3 HSB Address Register	UHDMA3_ ADDR	Read/Write	0x00000000
0x0738	Host DMA Channel 3 Control Register	UHDMA3_ CONTROL	Read/Write	0x00000000
0x073C	Host DMA Channel 3Status Register	UHDMA3_ STATUS	Read/Write	0x00000000
0x0740	Host DMA Channel 4 Next Descriptor Address Register	UHDMA4_ NEXTDESC	Read/Write	0x00000000
0x0744	Host DMA Channel 4 HSB Address Register	UHDMA4_ ADDR	Read/Write	0x00000000
0x0748	Host DMA Channel 4 Control Register	UHDMA4_ CONTROL	Read/Write	0x00000000
0x074C	Host DMA Channel 4 Status Register	UHDMA4_ STATUS	Read/Write	0x00000000

Table 26-5. USB PB Memory Map

Offset	Register	Name	Access	Reset Value
0x0750	Host DMA Channel 5 Next Descriptor Address Register	UHDMA5_ NEXTDESC	Read/Write	0x00000000
0x0754	Host DMA Channel 5 HSB Address Register	UHDMA5_ ADDR	Read/Write	0x00000000
0x0758	Host DMA Channel 5 Control Register	UHDMA5_ CONTROL	Read/Write	0x00000000
0x075C	Host DMA Channel 5 Status Register	UHDMA5_ STATUS	Read/Write	0x00000000
0x0760	Host DMA Channel 6 Next Descriptor Address Register	UHDMA6_ NEXTDESC	Read/Write	0x00000000
0x0764	Host DMA Channel 6 HSB Address Register	UHDMA6_ ADDR	Read/Write	0x00000000
0x0768	Host DMA Channel 6 Control Register	UHDMA6_ CONTROL	Read/Write	0x00000000
0x076C	Host DMA Channel 6 Status Register	UHDMA6_ STATUS	Read/Write	0x00000000
0x0770 - 0x07FC	Reserved	–	–	–
0x0800	General Control Register	USBCON	Read/Write	0x03004000
0x0804	General Status Register	USBSTA	Read-Only	0x00000400
0x0808	General Status Clear Register	USBSTACLR	Write-Only	0x00000000
0x080C	General Status Set Register	USBSTASET	Write-Only	0x00000000
0x0818	IP Version Register	UVERS	Read-Only	0x00000260
0x081C	IP Features Register	UFEATURES	Read-Only	0x00012467
0x0820	IP PB Address Size Register	UADDRSIZE	Read-Only	0x00001000
0x0824	IP Name Register 1	UNAME1	Read-Only	0x48555342 (“HUSB”)
0x0828	IP Name Register 2	UNAME2	Read-Only	0x004F5447 (“\0OTG“)
0x082C - 0x0BFC	Reserved	–	–	–

Table 26-6. USB HSB Memory Map

Offset	Register	Name	Access	Reset Value
0x00000 - 0x0FFFC	Pipe/Endpoint 0 FIFO Data Register	USB_ FIFO0_DATA	Read/Write	Undefined
0x10000 - 0x1FFFC	Pipe/Endpoint 1 FIFO Data Register	USB_ FIFO1_DATA	Read/Write	Undefined
0x20000 - 0x2FFFC	Pipe/Endpoint 2 FIFO Data Register	USB_ FIFO2_DATA	Read/Write	Undefined
0x30000 - 0x3FFFC	Pipe/Endpoint 3 FIFO Data Register	USB_ FIFO3_DATA	Read/Write	Undefined
0x40000 - 0x4FFFC	Pipe/Endpoint 4 FIFO Data Register	USB_ FIFO4_DATA	Read/Write	Undefined

Table 26-6. USB HSB Memory Map

Offset	Register	Name	Access	Reset Value
0x50000 - 0x5FFFC	Pipe/Endpoint 5 FIFO Data Register	USB_ FIFO5_DATA	Read/Write	Undefined
0x60000 - 0x6FFFC	Pipe/Endpoint 6 FIFO Data Register	USB_ FIFO6_DATA	Read/Write	Undefined
0x70000 - 0xFFFFC	Reserved	–	–	–

In the following subsections, the bit and bit-field access types use the following flags:

- “r”: readable;
- “w”: writable;
- “u”: may be updated by hardware.

26.8.1 USB General Registers

26.8.1.1 USB General Control Register (USBCON)

Offset: 0x0800

Register Name: USBCON

Access Type: Read/Write

Reset Value: 0x03004000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	UIMOD	UIDE
						rw	rw
						1	1
23	22	21	20	19	18	17	16
–	UNLOCK	TIMPAGE		–	–	TIMVALUE	
	rw	rw				rw	
	0	0	0			0	0
15	14	13	12	11	10	9	8
USBE	FRZCLK	VBUSPO	OTGPADE	HNPREQ	SRPREQ	SRPSEL	VBUSHWC
rw	rw	rw	rw	rwu	rwu	rw	rw
0	1	0	0	0	0	0	0
7	6	5	4	3	2	1	0
STOE	HNPERRE	ROLEEXE	BCERRE	VBERRE	SRPE	VBUSTE	IDTE
rw	rw	rw	rw	rw	rw	rw	rw
0	0	0	0	0	0	0	0

- **IDTE: ID Transition Interrupt Enable**

Set to enable the ID Transition interrupt (IDTI).

Clear to disable the ID Transition interrupt (IDTI).

- **VBUSTE: VBus Transition Interrupt Enable**

Set to enable the VBus Transition interrupt (VBUSTI).

Clear to disable the VBus Transition interrupt (VBUSTI).

- **SRPE: SRP Interrupt Enable**

Set to enable the SRP interrupt (SRPI).

Clear to disable the SRP interrupt (SRPI).

- **VBERRE: VBus Error Interrupt Enable**

Set to enable the VBus Error interrupt (VBERRI).

Clear to disable the VBus Error interrupt (VBERRI).

- **BCERRE: B-Connection Error Interrupt Enable**

Set to enable the B-Connection Error interrupt (BCERRI).

Clear to disable the B-Connection Error interrupt (BCERRI).

- **ROLEEXE: Role Exchange Interrupt Enable**

Set to enable the Role Exchange interrupt (ROLEEXI).

Clear to disable the Role Exchange interrupt (ROLEEXI).

- **HNPERRE: HNP Error Interrupt Enable**

Set to enable the HNP Error interrupt (HNPERRI).

Clear to disable the HNP Error interrupt (HNPERRI).

- **STOE: Suspend Time-Out Interrupt Enable**

Set to enable the Suspend Time-Out interrupt (STOI).

Clear to disable the Suspend Time-Out interrupt (STOI).

- **VBUSHWC: VBus Hardware Control**

Set to disable the hardware control over the USB_VBOF output pin.

Clear to enable the hardware control over the USB_VBOF output pin.

If cleared, then the USB macro considers VBus problems and resets the USB_VBOF output pin in that event.

- **SRPSEL: SRP Selection**

Set to choose VBus pulsing as SRP method.

Clear to choose data line pulsing as SRP method.

- **SRPREQ: SRP Request**

Set to initiate an SRP when the controller is in device mode.

Cleared by hardware when the controller is initiating an SRP.

- **HNPREQ: HNP Request**

When the controller is in device mode:

Set to initiate an HNP.

Cleared by hardware when the controller is initiating an HNP.

When the controller is in host mode:

Set to accept an HNP.

Clear otherwise.

- **OTGPADE: OTG Pad Enable**

Set to enable the OTG pad.

Clear to disable the OTG pad.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit.

- **VBUSPO: VBus Polarity**

When 0, the USB_VBOF output signal is in its default mode (active high).

When 1, the USB_VBOF output signal is inverted (active low).

To be generic. May be useful to control an external VBus power module.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit.

- **FRZCLK: Freeze USB Clock**

Set to disable the clock inputs (the resume detection is still active). This reduces power consumption. Unless explicitly stated, all registers then become read-only.

Clear to enable the clock inputs.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit, but this freezes the clock inputs whatever its value.

- **USBE: USB Macro Enable**

Set to enable the USB controller.

Clear to disable and reset the USB controller, to disable the USB transceiver and to disable the USB controller clock inputs. Unless explicitly stated, all registers then become read-only and are reset.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1.

- **TIMVALUE: Timer Value**

Set to initialize the new value of the special timer register selected by TIMPAGE.

- **TIMPAGE: Timer Page**

Write the page value to access a special timer register.

- **UNLOCK: Timer Access Unlock**

Set to unlock the TIMPAGE and TIMVALUE fields before writing them.

Reset to lock the TIMPAGE and TIMVALUE fields.

Note that the TIMPAGE and TIMVALUE fields can always be read, whatever the value of UNLOCK.

- **UIDE: USB_ID Pin Enable**

Set to select the USB mode (device/host) from the USB_ID input pin.

Clear to select the USB mode (device/host) with the UIMOD bit.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit.

- **UIMOD: USB Macro Mode**

This bit has no effect when UIDE = 1 (USB_ID input pin activated).

Set to select the USB device mode.

Clear to select the USB host mode.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit.

26.8.1.2 USB General Status Register (USBSTA)

Offset: 0x0804

Register Name: USBSTA

Access Type: Read-Only

Reset Value: 0x00000400

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8
–	–	SPEED		VBUS	ID	VBUSRQ	–
		ru		ru	ru	ru	
		0	0	0	1	0	

7	6	5	4	3	2	1	0
STOI	HNPERRI	ROLEEXI	BCERRI	VBERRI	SRPI	VBUSTI	IDTI
ru	ru	ru	ru	ru	ru	ru	ru
0	0	0	0	0	0	0	0

- **IDTI: ID Transition Interrupt Flag**

Asynchronous interrupt.

Set by hardware when a transition (high to low, low to high) has been detected on the USB_ID input pin. This triggers a USB interrupt if IDTE = 1.

Shall be cleared by software (by setting the IDTIC bit) to acknowledge the interrupt (USB clock inputs must be enabled before).

Note that this interrupt is generated even if the clock is frozen by the FRZCLK bit.

- **VBUSTI: VBus Transition Interrupt Flag**

Asynchronous interrupt.

Set by hardware when a transition (high to low, low to high) has been detected on the VBUS pad. This triggers a USB interrupt if VBUSTE = 1.

Shall be cleared by software (by setting the VBUSTIC bit) to acknowledge the interrupt (USB clock inputs must be enabled before).

Note that this interrupt is generated even if the clock is frozen by the FRZCLK bit.

- **SRPI: SRP Interrupt Flag**

Shall only be used in host mode.

Set by hardware when an SRP has been detected. This triggers a USB interrupt if SRPE = 1.

Shall be cleared by software (by setting the SRPIC bit) to acknowledge the interrupt.

- **VBERRI: VBus Error Interrupt Flag**

In host mode, set by hardware when a VBus drop has been detected. This triggers a USB interrupt if VBERRE = 1.

Shall be cleared by software (by setting the VBERRIC bit) to acknowledge the interrupt.

Note that if a VBus problem occurs, then the VBERRI interrupt is generated even if the USB macro does not go to an error state because of VBUSHWC = 1.

- **BCERRI: B-Connection Error Interrupt Flag**

In host mode, set by hardware when an error occurs during the B-connection. This triggers a USB interrupt if BCERRE = 1.

Shall be cleared by software (by setting the BCERRIC bit) to acknowledge the interrupt.

- **ROLEEXI: Role Exchange Interrupt Flag**

Set by hardware when the USB controller has successfully switched its mode because of an HNP negotiation (host to device or device to host). This triggers a USB interrupt if ROLEEXE = 1.

Shall be cleared by software (by setting the ROLEEXIC bit) to acknowledge the interrupt.

- **HNPERRI: HNP Error Interrupt Flag**

In device mode, set by hardware when an error has been detected during an HNP negotiation. This triggers a USB interrupt if HNPERRE = 1.

Shall be cleared by software (by setting the HNPERRIC bit) to acknowledge the interrupt.

- **STOI: Suspend Time-Out Interrupt Flag**

In host mode, set by hardware when a time-out error (more than 200ms) has been detected after a suspend. This triggers a USB interrupt if STOE = 1.

Shall be cleared by software (by setting the STOIC bit) to acknowledge the interrupt.

- **VBUSRQ: VBus Request**

In host mode, set by software (by setting the VBUSRQS bit) to assert the USB_VBOF output pin in order to enable the VBus power supply generation.

Cleared by software by setting the VBUSRQC bit.

Cleared by hardware when a VBus error occurs when VBUSHWC = 0.

- **ID: USB_ID Pin State**

Set/cleared by hardware and reflects the state of the USB_ID input pin, even if USBE = 0.

- **VBUS: VBus Level**

Set/cleared by hardware and reflects the level of the VBus line, even if USBE = 0.

This bit can be used in device mode to monitor the USB bus connection state of the application.

- **SPEED: Speed Status**

Set by hardware according to the controller speed mode:

SPEED		Speed Status
0	0	FULL-SPEED mode
1	0	LOW-SPEED mode
X	1	Reserved

Shall only be used in host mode.

26.8.1.3 USB General Status Clear Register (USBSTACLR)

Offset: 0x0808

Register Name: USBSTACLR

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	VBUSRQC	–
						w	
						0	
7	6	5	4	3	2	1	0
STOIC	HNPERRIC	ROLEEXIC	BCERRIC	VBERRIC	SRPIC	VBUSTIC	IDTIC
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **IDTIC: ID Transition Interrupt Flag Clear**

Set to clear IDTI.

Clearing has no effect.

Always read as 0.

- **VBUSTIC: VBus Transition Interrupt Flag Clear**

Set to clear VBUSTI.

Clearing has no effect.

Always read as 0.

- **SRPIC: SRP Interrupt Flag Clear**

Set to clear SRPI.

Clearing has no effect.

Always read as 0.

- **VBERRIC: VBus Error Interrupt Flag Clear**

Set to clear VBERRI.

Clearing has no effect.

Always read as 0.

- **BCERRIC: B-Connection Error Interrupt Flag Clear**

Set to clear BCERRI.

Clearing has no effect.

Always read as 0.

- **ROLEEXIC: Role Exchange Interrupt Flag Clear**

Set to clear ROLEEXI.

Clearing has no effect.

Always read as 0.

- **HNPERRIC: HNP Error Interrupt Flag Clear**

Set to clear HNPERRI.

Clearing has no effect.

Always read as 0.

- **STOIC: Suspend Time-Out Interrupt Flag Clear**

Set to clear STOI.

Clearing has no effect.

Always read as 0.

- **VBUSRQC: VBus Request Clear**

Set to clear VBUSRQ.

Clearing has no effect.

Always read as 0.

26.8.1.4 USB General Status Set Register (USBSTASET)

Offset: 0x080C

Register Name: USBSTASET

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	VBUSRQS	–
						w	
						0	
7	6	5	4	3	2	1	0
STOIS	HNPERRIS	ROLEEXIS	BCERRIS	VBERRIS	SRPIS	VBUSTIS	IDTIS
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **IDTIS: ID Transition Interrupt Flag Set**

Set to set IDTI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **VBUSTIS: VBus Transition Interrupt Flag Set**

Set to set VBUSTI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **SRPIS: SRP Interrupt Flag Set**

Set to set SRPI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **VBERRIS: VBus Error Interrupt Flag Set**

Set to set VBERRI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **BCERRIS: B-Connection Error Interrupt Flag Set**

Set to set BCERRI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **ROLEEXIS: Role Exchange Interrupt Flag Set**

Set to set ROLEEXI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **HNPERRIS: HNP Error Interrupt Flag Set**

Set to set HNPERRI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **STOIS: Suspend Time-Out Interrupt Flag Set**

Set to set STOI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **VBUSRQS: VBus Request Set**

Set to set VBUSRQ.

Clearing has no effect.

Always read as 0.

26.8.1.5 USB IP Version Register (UVERS)

Offset: 0x0818

Register Name: UVERS

Access Type: Read-Only

Read Value: 0x00000260

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	METAL_FIX_NUM		
					r		
					0	0	0

15	14	13	12	11	10	9	8
VERSION_NUM							
r							
0					2		

7	6	5	4	3	2	1	0
VERSION_NUM							
r							
6					0		

- **VERSION_NUM: IP Version Number**

This field indicates the version number of the USB macro IP, encoded with 1 version digit per nibble, e.g. 0x0260 for version 2.6.0.

- **METAL_FIX_NUM: Number of Metal Fixes**

This field indicates the number of metal fixes of the USB macro IP.

26.8.1.6 USB IP Features Register (UFEATURES)

Offset: 0x081C

Register Name: UFEATURES

Access Type: Read-Only

Read Value: 0x00012467

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
BYTE_WRITE_DPRAM	FIFO_MAX_SIZE			DMA_FIFO_WORD_DEPTH			
r		r			r		
0	0	1	0	0	1	0	0

7	6	5	4	3	2	1	0
DMA_BUFFER_SIZE	DMA_CHANNEL_NBR			EPT_NBR_MAX			
r		r			r		
0	1	1	0	0	1	1	1

- **EPT_NBR_MAX: Maximal Number of Pipes/Endpoints**

This field indicates the number of hardware-implemented pipes/endpoints:

EPT_NBR_MAX				Maximal Number of Pipes/Endpoints
0	0	0	0	16
0	0	0	1	1
0	0	1	0	2
				...
1	1	1	1	15

- **DMA_CHANNEL_NBR: Number of DMA Channels**

This field indicates the number of hardware-implemented DMA channels:

DMA_CHANNEL_NBR			Number of DMA Channels
0	0	0	Reserved
0	0	1	1
0	1	0	2
			...
1	1	1	7

- **DMA_BUFFER_SIZE: DMA Buffer Size**

This field indicates the size of the DMA buffer:

DMA_BUFFER_SIZE	DMA Buffer Size
0	16 bits
1	24 bits

- **DMA_FIFO_WORD_DEPTH: DMA FIFO Depth in Words**

This field indicates the DMA FIFO depth controller in words:

DMA_FIFO_WORD_DEPTH				DMA FIFO Depth in Words
0	0	0	0	16
0	0	0	1	1
0	0	1	0	2
				...
1	1	1	1	15

- **FIFO_MAX_SIZE: Maximal FIFO Size**

This field indicates the maximal FIFO size, i.e. the DPRAM size:

FIFO_MAX_SIZE			Maximal FIFO Size
0	0	0	>= 128 bytes
0	0	1	>= 256 bytes
0	1	0	>= 512 bytes
0	1	1	>= 1024 bytes
1	0	0	>= 2048 bytes
1	0	1	>= 4096 bytes
1	1	0	>= 8192 bytes
1	1	1	>= 16384 bytes

- **BYTE_WRITE_DPRAM: DPRAM Byte-Write Capability**

This field indicates whether the DPRAM is byte-write capable:

BYTE_WRITE_DPRAM	DPRAM Byte-Write Capability
0	DPRAM byte write lanes have shadow logic implemented in the USB macro IP interface.
1	DPRAM is natively byte-write capable.

26.8.1.7 USB IP PB Address Size Register (UADDRSIZE)

Offset: 0x0820

Register Name: UADDRSIZE

Access Type: Read-Only

Read Value: 0x00001000

31	30	29	28	27	26	25	24
UADDRSIZE							
r							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
UADDRSIZE							
r							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
UADDRSIZE							
r							
0	0	0	1	0	0	0	0
7	6	5	4	3	2	1	0
UADDRSIZE							
r							
0	0	0	0	0	0	0	0

- **UADDRSIZE: IP PB Address Size**

This field indicates the size of the PB address space reserved for the USB macro IP interface (2 at the power of the number of bits reserved to encode the PB addresses of the USB macro IP interface relatively to its base address).

26.8.1.8 USB IP Name Register 1 (UNAME1)

Offset: 0x0824

Register Name: UNAME1

Access Type: Read-Only

Read Value: 0x48555342 (“HUSB”)

31	30	29	28	27	26	25	24
UNAME1							
r							
“H”							
23	22	21	20	19	18	17	16
UNAME1							
r							
“U”							
15	14	13	12	11	10	9	8
UNAME1							
r							
“S”							
7	6	5	4	3	2	1	0
UNAME1							
r							
“B”							

- **UNAME1: IP Name Part 1**

This field indicates the 1st part of the ASCII-encoded name of the USB macro IP.

26.8.1.9 USB IP Name Register 2 (UNAME2)

Offset: 0x0828

Register Name: UNAME2

Access Type: Read-Only

Read Value: 0x004F5447 (“\0OTG“)

31	30	29	28	27	26	25	24
UNAME2							
r							
“\0“							
23	22	21	20	19	18	17	16
UNAME2							
r							
“O“							
15	14	13	12	11	10	9	8
UNAME2							
r							
“T“							
7	6	5	4	3	2	1	0
UNAME2							
r							
“G“							

- **UNAME2: IP Name Part 2**

This field indicates the 2nd part of the ASCII-encoded name of the USB macro IP.

26.8.2 USB Device Registers

26.8.2.1 USB Device General Control Register (UDCON)

Offset: 0x0000

Register Name: UDCON

Access Type: Read/Write

Reset Value: 0x00000100

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8
–	–	–	LS	–	–	RMWKUP	DETACH
			rw			rwu	rw
			0			0	1

7	6	5	4	3	2	1	0
ADDEN	UADD						
rwu				rwu			
0	0	0	0	0	0	0	0

- **UADD: USB Address**

Set to configure the device address.

Cleared by hardware upon receiving a USB reset.

- **ADDEN: Address Enable**

Set to activate the UADD field (USB address).

Cleared by hardware upon receiving a USB reset.

Clearing by software has no effect.

- **DETACH: Detach**

Set to physically detach the device (disconnect internal pull-up resistor from D+ and D-).

Clear to reconnect the device.

- **RMWKUP: Remote Wake-Up**

Set to send an upstream resume to the host for a remote wake-up.

Cleared by hardware upon receiving a USB reset or once the upstream resume has been sent.

Clearing by software has no effect.

- **LS: Low-Speed Mode Force**

Set to force the low-speed mode.

Clear to unforce the low-speed mode. Then, the full-speed mode is active.

Note that this bit can be set/cleared even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset this bit.

26.8.2.2 USB Device Global Interrupt Register (UDINT)

Offset: 0x0004

Register Name: UDINT

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INT	DMA5INT	DMA4INT	DMA3INT	DMA2INT	DMA1INT	–
	ru	ru	ru	ru	ru	ru	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	EP6INT	EP5INT	EP4INT
					ru	ru	ru
					0	0	0
15	14	13	12	11	10	9	8
EP3INT	EP2INT	EP1INT	EP0INT	–	–	–	–
ru	ru	ru	ru				
0	0	0	0				
7	6	5	4	3	2	1	0
–	UPRSM	EORSM	WAKEUP	EORST	SOF	–	SUSP
	ru	ru	ru	ru	ru		ru
	0	0	0	0	0		0

- **SUSP: Suspend Interrupt Flag**

Set by hardware when a USB “Suspend” idle bus state has been detected for 3 frame periods (J state for 3 ms). This triggers a USB interrupt if SUSPE = 1.

Shall be cleared by software (by setting the SUSPC bit) to acknowledge the interrupt.

Cleared by hardware when a Wake-Up interrupt (WAKEUP) is raised.

- **SOF: Start of Frame Interrupt Flag**

Set by hardware when a USB “Start of Frame” PID (SOF) has been detected (every 1 ms). This triggers a USB interrupt if SOFE = 1. The FNUM field is updated.

Shall be cleared by software (by setting the SOFC bit) to acknowledge the interrupt.

- **EORST: End of Reset Interrupt Flag**

Set by hardware when a USB “End of Reset” has been detected. This triggers a USB interrupt if EORSTE = 1.

Shall be cleared by software (by setting the EORSTC bit) to acknowledge the interrupt.

- **WAKEUP: Wake-Up Interrupt Flag**

Asynchronous interrupt.

Set by hardware when the USB controller is reactivated by a filtered non-idle signal from the lines (not by an upstream resume). This triggers an interrupt if WAKEUPE = 1.

Shall be cleared by software (by setting the WAKEUPC bit) to acknowledge the interrupt (USB clock inputs must be enabled before).

Cleared by hardware when a Suspend interrupt (SUSP) is raised.

Note that this interrupt is generated even if the clock is frozen by the FRZCLK bit.

- **EORSM: End of Resume Interrupt Flag**

Set by hardware when the USB controller detects a valid “End of Resume” signal initiated by the host. This triggers a USB interrupt if EORSME = 1.

Shall be cleared by software (by setting the EORSMC bit) to acknowledge the interrupt.

- **UPRSM: Upstream Resume Interrupt Flag**

Set by hardware when the USB controller sends a resume signal called “Upstream Resume”. This triggers a USB interrupt if UPRSME = 1.

Shall be cleared by software (by setting the UPRSMC bit) to acknowledge the interrupt (USB clock inputs must be enabled before).

- **EPXINT, X in [0..6]: Endpoint X Interrupt Flag**

Set by hardware when an interrupt is triggered by the endpoint X (UESTAX, UECONX). This triggers a USB interrupt if EPXINTE = 1.

Cleared by hardware when the interrupt source is serviced.

- **DMAXINT, X in [1..6]: DMA Channel X Interrupt Flag**

Set by hardware when an interrupt is triggered by the DMA channel X. This triggers a USB interrupt if DMAXINTE = 1.

Cleared by hardware when the UDDMAX_STATUS interrupt source is cleared.

26.8.2.3 USB Device Global Interrupt Clear Register (UDINTCLR)

Offset: 0x0008

Register Name: UDINTCLR

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	UPRSMC	EORSMC	WAKEUPC	EORSTC	SOFC	–	SUSPC
	w	w	w	w	w		w
	0	0	0	0	0		0

- **SUSPC: Suspend Interrupt Flag Clear**

Set to clear SUSP.

Clearing has no effect.

Always read as 0.

- **SOFC: Start of Frame Interrupt Flag Clear**

Set to clear SOF.

Clearing has no effect.

Always read as 0.

- **EORSTC: End of Reset Interrupt Flag Clear**

Set to clear EORST.

Clearing has no effect.

Always read as 0.

- **WAKEUPC: Wake-Up Interrupt Flag Clear**

Set to clear WAKEUP.

Clearing has no effect.

Always read as 0.

- **EORSMC: End of Resume Interrupt Flag Clear**

Set to clear EORSM.

Clearing has no effect.

Always read as 0.

- **UPRSMC: Upstream Resume Interrupt Flag Clear**

Set to clear UPRSM.

Clearing has no effect.

Always read as 0.

26.8.2.4 USB Device Global Interrupt Set Register (UDINTSET)

Offset: 0x000C

Register Name: UDINTSET

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTS	DMA5INTS	DMA4INTS	DMA3INTS	DMA2INTS	DMA1INTS	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	UPRSMS	EORSMS	WAKEUPS	EORSTS	SOFS	–	SUSPS
	w	w	w	w	w		w
	0	0	0	0	0		0

- **SUSPS: Suspend Interrupt Flag Set**

Set to set SUSP, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **SOFS: Start of Frame Interrupt Flag Set**

Set to set SOF, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **EORSTS: End of Reset Interrupt Flag Set**

Set to set EORST, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **WAKEUPS: Wake-Up Interrupt Flag Set**

Set to set WAKEUP, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **EORSMS: End of Resume Interrupt Flag Set**

Set to set EORSM, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **UPRSMS: Upstream Resume Interrupt Flag Set**

Set to set UPRSM, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **DMAXINTS, X in [1..6]: DMA Channel X Interrupt Flag Set**

Set to set DMAXINT, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

26.8.2.5 USB Device Global Interrupt Enable Register (UDINTE)

Offset: 0x0010

Register Name: UDINTE

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTE	DMA5INTE	DMA4INTE	DMA3INTE	DMA2INTE	DMA1INTE	–
	r	r	r	r	r	r	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	EP6INTE	EP5INTE	EP4INTE
					r	r	r
					0	0	0
15	14	13	12	11	10	9	8
EP3INTE	EP2INTE	EP1INTE	EP0INTE	–	–	–	–
r	r	r	r				
0	0	0	0				
7	6	5	4	3	2	1	0
–	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	–	SUSPE
	r	r	r	r	r		r
	0	0	0	0	0		0

- **SUSPE: Suspend Interrupt Enable**

Set by software (by setting the SUSPES bit) to enable the Suspend interrupt (SUSP).

Clear by software (by setting the SUSPEC bit) to disable the Suspend interrupt (SUSP).

- **SOFE: Start of Frame Interrupt Enable**

Set by software (by setting the SOFES bit) to enable the Start of Frame interrupt (SOF).

Clear by software (by setting the SOFEC bit) to disable the Start of Frame interrupt (SOF).

- **EORSTE: End of Reset Interrupt Enable**

Set by software (by setting the EORSTES bit) to enable the End of Reset interrupt (EORST).

Clear by software (by setting the EORSTEC bit) to disable the End of Reset interrupt (EORST).

- **WAKEUPE: Wake-Up Interrupt Enable**

Set by software (by setting the WAKEUPES bit) to enable the Wake-Up interrupt (WAKEUP).

Clear by software (by setting the WAKEUPEC bit) to disable the Wake-Up interrupt (WAKEUP).

- **EORSME: End of Resume Interrupt Enable**

Set by software (by setting the EORSMES bit) to enable the End of Resume interrupt (EORSM).

Clear by software (by setting the EORSMEC bit) to disable the End of Resume interrupt (EORSM).

- **UPRSME: Upstream Resume Interrupt Enable**

Set by software (by setting the UPRSMES bit) to enable the Upstream Resume interrupt (UPRSM).

Clear by software (by setting the UPRSMEC bit) to disable the Upstream Resume interrupt (UPRSM).

- **EPXINTE, X in [0..6]: Endpoint X Interrupt Enable**

Set by software (by setting the EPXINTES bit) to enable the Endpoint X interrupt (EPXINT).

Clear by software (by setting the EPXINTEC bit) to disable the Endpoint X interrupt (EPXINT).

- **DMAXINTE, X in [1..6]: DMA Channel X Interrupt Enable**

Set by software (by setting the DMAXINTES bit) to enable the DMA Channel X interrupt (DMAXINT).

Clear by software (by setting the DMAXINTEC bit) to disable the DMA Channel X interrupt (DMAXINT).

26.8.2.6 USB Device Global Interrupt Enable Clear Register (UDINTECLR)

Offset: 0x0014

Register Name: UDINTECLR

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTEC	DMA5INTEC	DMA4INTEC	DMA3INTEC	DMA2INTEC	DMA1INTEC	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	EP6INTEC	EP5INTEC	EP4INTEC
					w	w	w
					0	0	0
15	14	13	12	11	10	9	8
EP3INTEC	EP2INTEC	EP1INTEC	EP0INTEC	–	–	–	–
w	w	w	w				
0	0	0	0				
7	6	5	4	3	2	1	0
–	UPRSMEC	EORSMEC	WAKEUPEC	EORSTEC	SOFEC	–	SUSPEC
	w	w	w	w	w		w
	0	0	0	0	0		0

- **SUSPEC: Suspend Interrupt Enable Clear**

Set to clear SUSPE.

Clearing has no effect.

Always read as 0.

- **SOFEC: Start of Frame Interrupt Enable Clear**

Set to clear SOFE.

Clearing has no effect.

Always read as 0.

- **EORSTEC: End of Reset Interrupt Enable Clear**

Set to clear EORSTE.

Clearing has no effect.

Always read as 0.

- **WAKEUPEC: Wake-Up Interrupt Enable Clear**

Set to clear WAKEUPE.

Clearing has no effect.

Always read as 0.

- **EORSMEC: End of Resume Interrupt Enable Clear**

Set to clear EORSME.

Clearing has no effect.

Always read as 0.

- **UPRSMEC: Upstream Resume Interrupt Enable Clear**

Set to clear UPRSME.

Clearing has no effect.

Always read as 0.

- **EPXINTEC, X in [0..6]: Endpoint X Interrupt Enable Clear**

Set to clear EPXINTE.

Clearing has no effect.

Always read as 0.

- **DMAXINTEC, X in [1..6]: DMA Channel X Interrupt Enable Clear**

Set to clear DMAXINTE.

Clearing has no effect.

Always read as 0.

26.8.2.7 USB Device Global Interrupt Enable Set Register (UDINTESET)

Offset: 0x0018

Register Name: UDINTESET

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTES	DMA5INTES	DMA4INTES	DMA3INTES	DMA2INTES	DMA1INTES	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	EP6INTES	EP5INTES	EP4INTES
					w	w	w
					0	0	0
15	14	13	12	11	10	9	8
EP3INTES	EP2INTES	EP1INTES	EP0INTES	–	–	–	–
w	w	w	w				
0	0	0	0				
7	6	5	4	3	2	1	0
–	UPRSMES	EORSMES	WAKEUPES	EORSTES	SOFES	–	SUSPES
	w	w	w	w	w		w
	0	0	0	0	0		0

- **SUSPES: Suspend Interrupt Enable Set**

Set to set SUSPE.

Clearing has no effect.

Always read as 0.

- **SOFES: Start of Frame Interrupt Enable Set**

Set to set SOFE.

Clearing has no effect.

Always read as 0.

- **EORSTES: End of Reset Interrupt Enable Set**

Set to set EORSTE.

Clearing has no effect.

Always read as 0.

- **WAKEUPES: Wake-Up Interrupt Enable Set**

Set to set WAKEUPE.

Clearing has no effect.

Always read as 0.

- **EORSMES: End of Resume Interrupt Enable Set**

Set to set EORSME.

Clearing has no effect.

Always read as 0.

- **UPRSMES: Upstream Resume Interrupt Enable Set**

Set to set UPRSME.

Clearing has no effect.

Always read as 0.

- **EPXINTES, X in [0..6]: Endpoint X Interrupt Enable Set**

Set to set EPXINTE.

Clearing has no effect.

Always read as 0.

- **DMAXINTES, X in [1..6]: DMA Channel X Interrupt Enable Set**

Set to set DMAXINTE.

Clearing has no effect.

Always read as 0.

26.8.2.8 USB Device Frame Number Register (UDFNUM)

Offset: 0x0020

Register Name: UDFNUM

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
FNCERR	–	FNUM					
ru		ru					
0		0	0	0	0	0	0

7	6	5	4	3	2	1	0
FNUM					–	–	–
		ru					
0	0	0	0	0			

- **FNUM: Frame Number**

Set by hardware. These bits are the 11-bit frame number information. They are provided in the last received SOF packet.

Cleared by hardware upon receiving a USB reset.

Note that FNUM is updated even if a corrupted SOF is received.

- **FNCERR: Frame Number CRC Error**

Set by hardware when a corrupted frame number is received. This bit and the SOF interrupt flag are updated at the same time.

Cleared by hardware upon receiving a USB reset.

26.8.2.9 USB Endpoint Enable/Reset Register (UERST)

Offset: 0x001C

Register Name: UERST

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	EPRST6	EPRST5	EPRST4	EPRST3	EPRST2	EPRST1	EPRST0
	rwu	rwu	rwu	rwu	rwu	rwu	rwu
	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	EPEN6	EPEN5	EPEN4	EPEN3	EPEN2	EPEN1	EPEN0
	rw	rw	rw	rw	rw	rw	rw
	0	0	0	0	0	0	0

- **EPENX, X in [0..6]: Endpoint X Enable**

Set to enable the endpoint X.

Clear to disable the endpoint X, what forces the endpoint X state to inactive (no answer to USB requests) and resets the endpoint X registers (UECFGX, UESTAX, UECONX) but not the endpoint configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE).

- **EPRSTX, X in [0..6]: Endpoint X Reset**

Set by software to reset the endpoint X FIFO prior to any other operation, upon hardware reset or when a USB bus reset has been received. This resets the endpoint X registers (UECFGX, UESTAX, UECONX) but not the endpoint configuration (ALLOC, EPBK, EPSIZE, EPDIR, EPTYPE).

All the endpoint mechanism (FIFO counter, reception, transmission, etc.) is reset apart from the Data Toggle Sequence field (DTSEQ) which can be cleared by setting the RSTDT bit (by setting the RSTDTS bit).

The endpoint configuration remains active and the endpoint is still enabled.

Then, clear by software to complete the reset operation and to start using the FIFO.

Cleared by hardware upon receiving a USB reset.

26.8.2.10 USB Endpoint X Configuration Register (UECFGX)

Offset: 0x0100 + X . 0x04

Register Name: UECFGX, X in [0..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	EPTYPE		–	AUTOSW	EPDIR
			rwu			rwu	rwu
			0	0		0	0
7	6	5	4	3	2	1	0
–	EPSIZE			EPBK		ALLOC	–
		rwu		rwu		rwu	
	0	0	0	0	0	0	

- **ALLOC: Endpoint Memory Allocate**

Set to allocate the endpoint memory.

Clear to free the endpoint memory.

Cleared by hardware upon receiving a USB reset (except for the endpoint 0).

Note that after setting this bit, the user should check the CFGOK bit to know whether the allocation of this endpoint is correct.

- **EPBK: Endpoint Banks**

Set to select the number of banks for the endpoint:

EPBK		Endpoint Banks
0	0	1 (single-bank endpoint)
0	1	2 (double-bank endpoint)
1	0	3 (triple-bank endpoint)
1	1	Reserved

For control endpoints, a single-bank endpoint (00b) should be selected.

Cleared by hardware upon receiving a USB reset (except for the endpoint 0).

- **EPSIZE: Endpoint Size**

Set to select the size of each endpoint bank:

EPSIZE			Endpoint Size
0	0	0	8 bytes
0	0	1	16 bytes
0	1	0	32 bytes
0	1	1	64 bytes
1	0	0	128 bytes
1	0	1	256 bytes
1	1	0	512 bytes
1	1	1	1024 bytes

Cleared by hardware upon receiving a USB reset (except for the endpoint 0).

- **EPDIR: Endpoint Direction**

Set to select the endpoint direction:

EPDIR	Endpoint Direction
0	OUT
1	IN (not for control endpoints)

Cleared by hardware upon receiving a USB reset.

- **AUTOSW: Automatic Switch**

Set to automatically switch bank when it is ready.

Clear to disable the automatic bank switching.

Cleared by hardware upon receiving a USB reset.

- **EPTYPE: Endpoint Type**

Set to select the endpoint type:

EPTYPE		Endpoint Type
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

Cleared by hardware upon receiving a USB reset.

26.8.2.11 USB Endpoint X Status Register (UESTAX)

Offset: 0x0130 + X . 0x04

Register Name: UESTAX, X in [0..6]

Access Type: Read-Only

Reset Value: 0x00000100

31	30	29	28	27	26	25	24
–	BYCT						
				ru			
	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16
BYCT				–	CFGOK	CTRLDIR	RWALL
	ru				ru	ru	ru
0	0	0	0		0	0	0

15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
	ru		ru				ru
0	0	0	0			0	1

7	6	5	4	3	2	1	0
SHORT PACKET	STALLEDI/ CRCERRI	OVERFI	NAKINI	NAKOUTI	RXSTPI/ UNDERFI	RXOUTI	TXINI
ru	ru	ru	ru	ru	ru	ru	ru
0	0	0	0	0	0	0	0

- **TXINI: Transmitted IN Data Interrupt Flag**

For control endpoints:

Set by hardware when the current bank is ready to accept a new IN packet. This triggers an EPXINT interrupt if TXINE = 1.

Shall be cleared by software (by setting the TXINIC bit) to acknowledge the interrupt and to send the packet.

For isochronous, bulk and interrupt IN endpoints:

Set by hardware at the same time as FIFOCON when the current bank is free. This triggers an EPXINT interrupt if TXINE = 1.

Shall be cleared by software (by setting the TXINIC bit) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The software then writes into the FIFO and clears the FIFOCON bit to allow the USB controller to send the data. If the IN endpoint is composed of multiple banks, this also switches to the next bank. The TXINI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

TXINI shall always be cleared before clearing FIFOCON.

This bit is inactive (cleared) for isochronous, bulk and interrupt OUT endpoints.

- **RXOUTI: Received OUT Data Interrupt Flag**

For control endpoints:

Set by hardware when the current bank contains a bulk OUT packet (data or status stage). This triggers an EPXINT interrupt if RXOUTE = 1.

Shall be cleared by software (by setting the RXOUTIC bit) to acknowledge the interrupt and to free the bank.

For isochronous, bulk and interrupt OUT endpoints:

Set by hardware at the same time as FIFOCON when the current bank is full. This triggers an EPXINT interrupt if RXOUTE = 1.

Shall be cleared by software (by setting the RXOUTIC bit) to acknowledge the interrupt, what has no effect on the endpoint FIFO.

The software then reads from the FIFO and clears the FIFOCON bit to free the bank. If the OUT endpoint is composed of multiple banks, this also switches to the next bank. The RXOUTI and FIFOCON bits are updated by hardware in accordance with the status of the next bank.

RXOUTI shall always be cleared before clearing FIFOCON.

This bit is inactive (cleared) for isochronous, bulk and interrupt IN endpoints.

- **RXSTPI: Received SETUP Interrupt Flag**

For control endpoints, set by hardware to signal that the current bank contains a new valid SETUP packet. This triggers an EPXINT interrupt if RXSTPE = 1.

Shall be cleared by software (by setting the RXSTPIC bit) to acknowledge the interrupt and to free the bank.

This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means UNDERFI for isochronous IN/OUT endpoints.

- **UNDERFI: Underflow Interrupt Flag**

For isochronous IN/OUT endpoints, set by hardware when an underflow error occurs. This triggers an EPXINT interrupt if UNDERFE = 1.

An underflow can occur during IN stage if the host attempts to read from an empty bank. A zero-length packet is then automatically sent by the USB controller.

An underflow can also occur during OUT stage if the host sends a packet while the bank is already full. Typically, the CPU is not fast enough. The packet is lost.

Shall be cleared by software (by setting the UNDERFIC bit) to acknowledge the interrupt.

This bit is inactive (cleared) for bulk and interrupt IN/OUT endpoints and it means RXSTPI for control endpoints.

- **NAKOUTI: NAKed OUT Interrupt Flag**

Set by hardware when a NAK handshake has been sent in response to an OUT request from the host. This triggers an EPXINT interrupt if NAKOUTE = 1.

Shall be cleared by software (by setting the NAKOUTIC bit) to acknowledge the interrupt.

- **NAKINI: NAKed IN Interrupt Flag**

Set by hardware when a NAK handshake has been sent in response to an IN request from the host. This triggers an EPXINT interrupt if NAKINE = 1.

Shall be cleared by software (by setting the NAKINIC bit) to acknowledge the interrupt.

- **OVERFI: Overflow Interrupt Flag**

Set by hardware when an overflow error occurs. This triggers an EPXINT interrupt if OVERFE = 1.

For all endpoint types, an overflow can occur during OUT stage if the host attempts to write into a bank that is too small for the packet. The packet is acknowledged and the Received OUT Data interrupt (RXOUTI) is raised as if no overflow had occurred. The bank is filled with all the first bytes of the packet that fit in.

Shall be cleared by software (by setting the OVERFIC bit) to acknowledge the interrupt.

- **STALLEDI: STALLed Interrupt Flag**

Set by hardware to signal that a STALL handshake has been sent. To do that, the software has to set the STALLRQ bit (by setting the STALLRQS bit). This triggers an EPXINT interrupt if STALLEDE = 1.

Shall be cleared by software (by setting the STALLEDIC bit) to acknowledge the interrupt.

- **CRCERRI: CRC Error Interrupt Flag**

Set by hardware to signal that a CRC error has been detected in an isochronous OUT endpoint. The OUT packet is stored in the bank as if no CRC error had occurred. This triggers an EPXINT interrupt if CRCERRE = 1.

Shall be cleared by software (by setting the CRCERRIC bit) to acknowledge the interrupt.

- **SHORTPACKET: Short Packet Interrupt Flag**

For non-control OUT endpoints, set by hardware when a short packet has been received.

For non-control IN endpoints, set by hardware when a short packet is transmitted upon ending a DMA transfer, thus signaling an end of isochronous frame or a bulk or interrupt end of transfer, this only if the End of DMA Buffer Output Enable bit (DMAEND_EN) and the Automatic Switch bit (AUTOSW) are set.

This triggers an EPXINT interrupt if SHORTPACKETE = 1.

Shall be cleared by software (by setting the SHORTPACKETC bit) to acknowledge the interrupt.

- **DTSEQ: Data Toggle Sequence**

Set by hardware to indicate the PID of the current bank:

DTSEQ		Data Toggle Sequence
0	0	Data0
0	1	Data1
1	X	Reserved

For IN transfers, it indicates the data toggle sequence that will be used for the next packet to be sent. This is not relative to the current bank.

For OUT transfers, this value indicates the last data toggle sequence received on the current bank.

Note that by default DTSEQ = 01b, as if the last data toggle sequence was Data1, so the next sent or expected data toggle sequence should be Data0.

- **NBUSYBK: Number of Busy Banks**

Set by hardware to indicate the number of busy banks:

NBUSYBK		Number of Busy Banks
0	0	0 (all banks free)
0	1	1
1	0	2
1	1	3

For IN endpoints, it indicates the number of banks filled by the user and ready for IN transfer. When all banks are free, this triggers an EPXINT interrupt if NBUSYBKE = 1.

For OUT endpoints, it indicates the number of banks filled by OUT transactions from the host. When all banks are busy, this triggers an EPXINT interrupt if NBUSYBKE = 1.

Note that when the FIFOCON bit is cleared (by setting the FIFOCONC bit) to validate a new bank, this field is updated 2 or 3 clock cycles later to calculate the address of the next bank.

- **CURRBK: Current Bank**

For non-control endpoints, set by hardware to indicate the current bank:

CURRBK		Current Bank
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Reserved

Note that this field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt flag.

- **RWALL: Read/Write Allowed**

For IN endpoints, set by hardware when the current bank is not full, i.e. the software can write further data into the FIFO.

For OUT endpoints, set by hardware when the current bank is not empty, i.e. the software can read further data from the FIFO.

Never set if STALLRQ = 1 or in case of error.

Cleared by hardware otherwise.

This bit shall not be used for control endpoints.

- **CTRLDIR: Control Direction**

Set by hardware after a SETUP packet to indicate the direction of the following packet:

CTRLDIR	Control Direction
0	OUT
1	IN

Can not be set or cleared by software.

- **CFGOK: Configuration OK Status**

This bit is updated when the ALLOC bit is set.

Set by hardware if the endpoint X number of banks (EPBK) and size (EPSIZE) are correct compared to the maximal allowed number of banks and size for this endpoint and to the maximal FIFO size (i.e. the DPRAM size).

If this bit is cleared by hardware, the user should reprogram the UECFGX register with correct EPBK and EPSIZE values.

- **BYCT: Byte Count**

Set by the hardware to indicate the byte count of the FIFO.

For IN endpoints, incremented after each byte written by the software into the endpoint and decremented after each byte sent to the host.

For OUT endpoints, incremented after each byte received from the host and decremented after each byte read by the software from the endpoint.

Note that this field may be updated 1 clock cycle after the RWALL bit changes, so the user should not poll this field as an interrupt flag.

26.8.2.12 USB Endpoint X Status Clear Register (UESTAXCLR)

Offset: 0x0160 + X . 0x04

Register Name: UESTAXCLR, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SHORT PACKETC	STALLEDIC/ CRCERRIC	OVERFIC	NAKINIC	NAKOUTIC	RXSTPIC/ UNDERFIC	RXOUTIC	TXINIC
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **TXINIC: Transmitted IN Data Interrupt Flag Clear**

Set to clear TXINI.

Clearing has no effect.

Always read as 0.

- **RXOUTIC: Received OUT Data Interrupt Flag Clear**

Set to clear RXOUTI.

Clearing has no effect.

Always read as 0.

- **RXSTPIC: Received SETUP Interrupt Flag Clear**

Set to clear RXSTPI.

Clearing has no effect.

Always read as 0.

- **UNDERFIC: Underflow Interrupt Flag Clear**

Set to clear UNDERFI.

Clearing has no effect.

Always read as 0.

- **NAKOUTIC: NAKed OUT Interrupt Flag Clear**

Set to clear NAKOUTI.

Clearing has no effect.

Always read as 0.

- **NAKINIC: NAKed IN Interrupt Flag Clear**

Set to clear NAKINI.

Clearing has no effect.

Always read as 0.

- **OVERFIC: Overflow Interrupt Flag Clear**

Set to clear OVERFI.

Clearing has no effect.

Always read as 0.

- **STALLEDIC: STALLed Interrupt Flag Clear**

Set to clear STALLEDI.

Clearing has no effect.

Always read as 0.

- **CRCERRIC: CRC Error Interrupt Flag Clear**

Set to clear CRCERRI.

Clearing has no effect.

Always read as 0.

- **SHORTPACKETC: Short Packet Interrupt Flag Clear**

Set to clear SHORTPACKET.

Clearing has no effect.

Always read as 0.

26.8.2.13 USB Endpoint X Status Set Register (UESTAXSET)

Offset: 0x0190 + X . 0x04

Register Name: UESTAXSET, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–		–
			w				
			0				
7	6	5	4	3	2	1	0
SHORT PACKETS	STALLEDIS/ CRCERRIS	OVERFIS	NAKINIS	NAKOUTIS	RXSTPIS/ UNDERFIS	RXOUTIS	TXINIS
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **TXINIS: Transmitted IN Data Interrupt Flag Set**

Set to set TXINI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **RXOUTIS: Received OUT Data Interrupt Flag Set**

Set to set RXOUTI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **RXSTPIS: Received SETUP Interrupt Flag Set**

Set to set RXSTPI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **UNDERFIS: Underflow Interrupt Flag Set**

Set to set UNDERFI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **NAKOUTIS: NAKed OUT Interrupt Flag Set**

Set to set NAKOUTI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **NAKINIS: NAKed IN Interrupt Flag Set**

Set to set NAKINI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **OVERFIS: Overflow Interrupt Flag Set**

Set to set OVERFI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **STALLEDIS: STALLed Interrupt Flag Set**

Set to set STALLEDI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **CRCERRIS: CRC Error Interrupt Flag Set**

Set to set CRCERRI, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **SHORTPACKETS: Short Packet Interrupt Flag Set**

Set to set SHORTPACKET, what may be useful for test or debug purposes.

Clearing has no effect.

Always read as 0.

- **NBUSYBKS: Number of Busy Banks Interrupt Flag Set**

Set to force the Number of Busy Banks interrupt flag (NBUSYBK), what may be useful for test or debug purposes.

Set again to unforce the Number of Busy Banks interrupt flag (NBUSYBK).

Clearing has no effect.

Always read as 0.

26.8.2.14 USB Endpoint X Control Register (UECONX)

Offset: 0x01C0 + X . 0x04

Register Name: UECONX, X in [0..6]

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	STALLRQ	RSTDT	–	EPDISHDMA
				ru	ru		r
				0	0		0
15	14	13	12	11	10	9	8
–	FIFOCON	KILLBK	NBUSYBKE	–	–	–	–
	ru	ru	r				
	0	0	0				
7	6	5	4	3	2	1	0
SHORT PACKETE	STALLEDE/ CRCERRE	OVERFE	NAKINE	NAKOUTE	RXSTPE/ UNDERFE	RXOUTE	TXINE
r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0

- **TXINE: Transmitted IN Data Interrupt Enable**

Set by software (by setting the TXINES bit) to enable the Transmitted IN Data interrupt (TXINI).

Clear by software (by setting the TXINEC bit) to disable the Transmitted IN Data interrupt (TXINI).

- **RXOUTE: Received OUT Data Interrupt Enable**

Set by software (by setting the RXOUTES bit) to enable the Received OUT Data interrupt (RXOUT).

Clear by software (by setting the RXOUTEC bit) to disable the Received OUT Data interrupt (RXOUT).

- **RXSTPE: Received SETUP Interrupt Enable**

Set by software (by setting the RXSTPES bit) to enable the Received SETUP interrupt (RXSTPI).

Clear by software (by setting the RXSTPEC bit) to disable the Received SETUP interrupt (RXSTPI).

- **UNDERFE: Underflow Interrupt Enable**

Set by software (by setting the UNDERFES bit) to enable the Underflow interrupt (UNDERFI).

Clear by software (by setting the UNDERFEC bit) to disable the Underflow interrupt (UNDERFI).

- **NAKOUTE: NAKed OUT Interrupt Enable**

Set by software (by setting the NAKOUTES bit) to enable the NAKed OUT interrupt (NAKOUTI).

Clear by software (by setting the NAKOUTEC bit) to disable the NAKed OUT interrupt (NAKOUTI).

- **NAKINE: NAKed IN Interrupt Enable**

Set by software (by setting the NAKINES bit) to enable the NAKed IN interrupt (NAKINI).

Clear by software (by setting the NAKINEC bit) to disable the NAKed IN interrupt (NAKINI).

- **OVERFE: Overflow Interrupt Enable**

Set by software (by setting the OVERFES bit) to enable the Overflow interrupt (OVERFI).

Clear by software (by setting the OVERFEC bit) to disable the Overflow interrupt (OVERFI).

- **STALLEDE: STALLed Interrupt Enable**

Set by software (by setting the STALLEDES bit) to enable the STALLed interrupt (STALLEDI).

Clear by software (by setting the STALLEDEC bit) to disable the STALLed interrupt (STALLEDI).

- **CRCERRE: CRC Error Interrupt Enable**

Set by software (by setting the CRCERRES bit) to enable the CRC Error interrupt (CRCERRI).

Clear by software (by setting the CRCERREC bit) to disable the CRC Error interrupt (CRCERRI).

- **SHORTPACKETE: Short Packet Interrupt Enable**

Set by software (by setting the SHORTPACKETES bit) to enable the Short Packet interrupt (SHORTPACKET).

Clear by software (by setting the SHORTPACKETEC bit) to disable the Short Packet interrupt (SHORTPACKET).

- **NBUSYBKE: Number of Busy Banks Interrupt Enable**

Set by software (by setting the NBUSYBKES bit) to enable the Number of Busy Banks interrupt (NBUSYBK).

Clear by software (by setting the NBUSYBKEC bit) to disable the Number of Busy Banks interrupt (NBUSYBK).

- **KILLBK: Kill IN Bank**

Set by software (by setting the KILLBKS bit) to kill the last written bank.

Cleared by hardware when the bank is killed.

Caution: The bank is really cleared when the “kill packet” procedure is accepted by the USB macro core. This bit is automatically cleared after the end of the procedure:

  – The bank is really cleared or the bank is sent (IN transfer): NBUSYBK is decremented.
  – The bank is not cleared but sent (IN transfer): NBUSYBK is decremented.
  – The bank is not cleared because it was empty.

The software shall wait for this bit to be cleared before trying to kill another packet.

Note that this kill request is refused if at the same time an IN token is coming and the last bank is the current one being sent on the USB line. If at least 2 banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. Indeed, in this case, the current bank is sent (IN transfer) while the last bank is killed.

- **FIFOCON: FIFO Control**

For control endpoints:

The FIFOCON and RWALL bits are irrelevant. The software shall therefore never use them on these endpoints. When read, their value is always 0.

For IN endpoints:

Set by hardware when the current bank is free, at the same time as TXINI.

Clear by software (by setting the FIFOCONC bit) to send the FIFO data and to switch to the next bank.

For OUT endpoints:

Set by hardware when the current bank is full, at the same time as RXOUTI.

Clear by software (by setting the FIFOCONC bit) to free the current bank and to switch to the next bank.

- **EPDISHDMA: Endpoint Interrupts Disable HDMA Request Enable**

Set by software (by setting the EPDISHDMAS bit) to pause the on-going DMA channel X transfer on any Endpoint X interrupt (EPXINT), whatever the state of the Endpoint X Interrupt Enable bit (EPXINTE).

The software then has to acknowledge or to disable the interrupt source (e.g. RXOUTI) or to clear the EPDISHDMA bit (by setting the EPDISHDMAC bit) in order to complete the DMA transfer.

In ping-pong mode, if the interrupt is associated to a new system-bank packet (e.g. Bank1) and the current DMA transfer is running on the previous packet (Bank0), then the previous-packet DMA transfer completes normally, but the new-packet DMA transfer will not start (not requested).

If the interrupt is not associated to a new system-bank packet (NAKINI, NAKOUTI, etc.), then the request cancellation may occur at any time and may immediately pause the current DMA transfer.

This may be used for example to identify erroneous packets, to prevent them from being transferred into a buffer, to complete a DMA transfer by software after reception of a short packet, etc.

- **RSTDT: Reset Data Toggle**

Set by software (by setting the RSTDTS bit) to clear the data toggle sequence, i.e. to set to Data0 the data toggle sequence of the next sent (IN endpoints) or received (OUT endpoints) packet.

Cleared by hardware instantaneously.

The software does not have to wait for this bit to be cleared.

- **STALLRQ: STALL Request**

Set by software (by setting the STALLRQS bit) to request to send a STALL handshake to the host.

Cleared by hardware when a new SETUP packet is received.

Can also be cleared by software by setting the STALLRQC bit.

26.8.2.15 USB Endpoint X Control Clear Register (UECONXCLR)

Offset: 0x0220 + X . 0x04

Register Name: UECONXCLR, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	STALLRQC	–	–	EPDISHDMAC
				w			w
				0			0

15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
	w		w				
	0		0				

7	6	5	4	3	2	1	0
SHORT PACKETEC	STALLEDEC/ CRCERREC	OVERFEC	NAKINEC	NAKOUTEC	RXSTPEC/ UNDERFEC	RXOUTEC	TXINEC
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **TXINEC: Transmitted IN Data Interrupt Enable Clear**

Set to clear TXINE.

Clearing has no effect.

Always read as 0.

- **RXOUTEC: Received OUT Data Interrupt Enable Clear**

Set to clear RXOUTE.

Clearing has no effect.

Always read as 0.

- **RXSTPEC: Received SETUP Interrupt Enable Clear**

Set to clear RXSTPE.

Clearing has no effect.

Always read as 0.

- **UNDERFEC: Underflow Interrupt Enable Clear**

Set to clear UNDERFE.

Clearing has no effect.

Always read as 0.

- **NAKOUTEC: NAKed OUT Interrupt Enable Clear**

Set to clear NAKOUTE.

Clearing has no effect.

Always read as 0.

- **NAKINEC: NAKed IN Interrupt Enable Clear**

Set to clear NAKINE.

Clearing has no effect.

Always read as 0.

- **OVERFEC: Overflow Interrupt Enable Clear**

Set to clear OVERFE.

Clearing has no effect.

Always read as 0.

- **STALLEDEC: STALLed Interrupt Enable Clear**

Set to clear STALLEDE.

Clearing has no effect.

Always read as 0.

- **CRCERREC: CRC Error Interrupt Enable Clear**

Set to clear CRCERRE.

Clearing has no effect.

Always read as 0.

- **SHORTPACKETEC: Short Packet Interrupt Enable Clear**

Set to clear SHORTPACKETE.

Clearing has no effect.

Always read as 0.

- **NBUSYBKEC: Number of Busy Banks Interrupt Enable Clear**

Set to clear NBUSYBKE.

Clearing has no effect.

Always read as 0.

- **FIFOCONC: FIFO Control Clear**

Set to clear FIFOCON.

Clearing has no effect.

Always read as 0.

- **EPDISHDMAC: Endpoint Interrupts Disable HDMA Request Enable Clear**

Set to clear EPDISHDMA.

Clearing has no effect.

Always read as 0.

- **STALLRQC: STALL Request Clear**

Set to clear STALLRQ.

Clearing has no effect.

Always read as 0.

26.8.2.16 USB Endpoint X Control Set Register (UECONXSET)

Offset: 0x01F0 + X . 0x04

Register Name: UECONXSET, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	STALLRQS	RSTDTS	–	EPDISHDMAS
				w	w		w
				0	0		0

15	14	13	12	11	10	9	8
–	–	KILLBKS	NBUSYBKES	–	–	–	–
		w	w				
		0	0				

7	6	5	4	3	2	1	0
SHORT PACKETES	STALLEDES/ CRCERRES	OVERFES	NAKINES	NAKOUTES	RXSTPES/ UNDERFES	RXOUTES	TXINES
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **TXINES: Transmitted IN Data Interrupt Enable Set**

Set to set TXINE.

Clearing has no effect.

Always read as 0.

- **RXOUTES: Received OUT Data Interrupt Enable Set**

Set to set RXOUTE.

Clearing has no effect.

Always read as 0.

- **RXSTPES: Received SETUP Interrupt Enable Set**

Set to set RXSTPE.

Clearing has no effect.

Always read as 0.

- **UNDERFES: Underflow Interrupt Enable Set**

Set to set UNDERFE.

Clearing has no effect.

Always read as 0.

- **NAKOUTES: NAKed OUT Interrupt Enable Set**

Set to set NAKOUTE.

Clearing has no effect.

Always read as 0.

- **NAKINES: NAKed IN Interrupt Enable Set**

Set to set NAKINE.

Clearing has no effect.

Always read as 0.

- **OVERFES: Overflow Interrupt Enable Set**

Set to set OVERFE.

Clearing has no effect.

Always read as 0.

- **STALLEDES: STALLed Interrupt Enable Set**

Set to set STALLEDE.

Clearing has no effect.

Always read as 0.

- **CRCERRES: CRC Error Interrupt Enable Set**

Set to set CRCERRE.

Clearing has no effect.

Always read as 0.

- **SHORTPACKETES: Short Packet Interrupt Enable Set**

Set to set SHORTPACKETE.

Clearing has no effect.

Always read as 0.

- **NBUSYBKES: Number of Busy Banks Interrupt Enable Set**

Set to set NBUSYBKE.

Clearing has no effect.

Always read as 0.

- **KILLBKS: Kill IN Bank Set**

Set to set KILLBK.

Clearing has no effect.

Always read as 0.

- **EPDISHDMAS: Endpoint Interrupts Disable HDMA Request Enable Set**

Set to set EPDISHDMA.

Clearing has no effect.

Always read as 0.

- **RSTDTS: Reset Data Toggle Set**

Set to set RSTDT.

Clearing has no effect.

Always read as 0.

- **STALLRQS: STALL Request Set**

Set to set STALLRQ.

Clearing has no effect.

Always read as 0.

26.8.2.17 USB Device DMA Channel X Next Descriptor Address Register (UDDMAX_NEXTDESC)

Offset: 0x0310 + (X - 1) . 0x10

Register Name: *UDDMAX_NEXTDESC*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
NXT_DESC_ADDR				–	–	–	–
rwu							
0	0	0	0				

- **NXT_DESC_ADDR: Next Descriptor Address**

This field contains the bits 31:4 of the 16-byte aligned address of the next channel descriptor to be processed.

Note that this field is written either by software or by descriptor loading.

26.8.2.18 USB Device DMA Channel X HSB Address Register (UDDMAX_ADDR)

Offset: 0x0314 + (X - 1) . 0x10

Register Name: *UDDMAX_ADDR*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0

- **HSB_ADDR: HSB Address**

This field determines the HSB bus current address of a channel transfer.

The address set on the HSB address bus is HSB_ADDR rounded down to the nearest word-aligned address, i.e. HSB_ADDR[1:0] is considered as 00b since only word accesses are performed.

Channel HSB start and end addresses may be aligned on any byte boundary.

The software may write this field only when the Channel Enabled bit (CH_EN) of the UDDMAX_STATUS register is clear.

This field is updated at the end of the address phase of the current access to the HSB bus. It is incremented of the HSB access byte-width.

The HSB access width is 4 bytes, or less at packet start or end if the start or end address is not aligned on a word boundary.

The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer.

The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

The channel start address is written by software or loaded from the descriptor, whereas the channel end address is either determined by the end of buffer or the end of USB transfer if the Buffer Close Input Enable bit (BUFF_CLOSE_IN_EN) is set.

26.8.2.19 USB Device DMA Channel X Control Register (UDDMAX_CONTROL)

Offset: 0x0318 + (X - 1) . 0x10

Register Name: *UDDMAX_CONTROL*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH_BYTE_LENGTH							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
CH_BYTE_LENGTH							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BURST_LOCK_EN	DESC_LD_IRQ_EN	EOBUFF_IRQ_EN	EOT_IRQ_EN	DMAEND_EN	BUFF_CLOSE_IN_EN	LD_NXT_CH_DESC_EN	CH_EN
rwu	rwu	rwu	rwu	rwu	rwu	rwu	rwu
0	0	0	0	0	0	0	0

- **CH_EN: Channel Enable**
- **LD_NXT_CH_DESC_EN: Load Next Channel Descriptor Enable**
- **BUFF_CLOSE_IN_EN: Buffer Close Input Enable**
- **DMAEND_EN: End of DMA Buffer Output Enable**
- **EOT_IRQ_EN: End of USB Transfer Interrupt Enable**
- **EOBUFF_IRQ_EN: End of Buffer Interrupt Enable**
- **DESC_LD_IRQ_EN: Descriptor Loaded Interrupt Enable**
- **BURST_LOCK_EN: Burst Lock Enable**
- **CH_BYTE_LENGTH: Channel Byte Length**

26.8.2.20 *USB Device DMA Channel X Status Register (UDDMAX_STATUS)*

Offset: 0x031C + (X - 1) . 0x10

Register Name: *UDDMAX_STATUS*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH_BYTE_CNT							
ru							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
CH_BYTE_CNT							
ru							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	DESC_LD_STA	EOCH_BUFF_STA	EOT_STA	–	–	CH_ACTIVE	CH_EN
	ru	ru	ru			rwu	rwu
	0	0	0			0	0

- **CH_EN: Channel Enabled**
- **CH_ACTIVE: Channel Active**
- **EOT_STA: End of USB Transfer Status**
- **EOCH_BUFF_STA: End of Channel Buffer Status**
- **DESC_LD_STA: Descriptor Loaded Status**
- **CH_BYTE_CNT: Channel Byte Count**

26.8.3 USB Host Registers

26.8.3.1 USB Host General Control Register (UHCON)

Offset: 0x0400

Register Name: UHCON

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	RESUME	RESET	SOFE
					rwu	rwu	rwu
					0	0	0
7	6	5	4	3	2	1	0
–	HADDR						
				rwu			
	0	0	0	0	0	0	0

- **HADDR: USB Host Address**
- **SOFE: Start of Frame Generation Enable**
- **RESET: Send USB Reset**
- **RESUME: Send USB Resume**

26.8.3.2 USB Host Global Interrupt Register (UHINT)

Offset: 0x0404

Register Name: UHINT

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INT	DMA5INT	DMA4INT	DMA3INT	DMA2INT	DMA1INT	–
	r	r	r	r	r	r	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	P6INT	P5INT	P4INT	P3INT	P2INT	P1INT	P0INT
	r	r	r	r	r	r	r
	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
–	HWUPI	HSOFI	RXRSMI	RSMEDI	RSTI	DDISCI	DCONNI
	r	r	r	r	r	r	r
	0	0	0	0	0	0	0

- **DCONNI: Device Connection Interrupt Flag**

- **DDISCI: Device Disconnection Interrupt Flag**

- **RSTI: USB Reset Sent Interrupt Flag**

- **RSMEDI: Downstream Resume Sent Interrupt Flag**

- **RXRSMI: Upstream Resume Received Interrupt Flag**

- **HSOFI: Host Start of Frame Interrupt Flag**

- **HWUPI: Host Wake-Up Interrupt Flag**

Asynchronous interrupt.

Note that this interrupt is generated even if the clock is frozen by the FRZCLK bit.

- **PXINT, X in [0..6]: Pipe X Interrupt Flag**

- **DMAXINT, X in [1..6]: DMA Channel X Interrupt Flag**

26.8.3.3 USB Host Global Interrupt Clear Register (UHINTCLR)

Offset: 0x0408

Register Name: UHINTCLR

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	HWUPIC	HSOFIC	RXRSMIC	RSMEDIC	RSTIC	DDISCIC	DCONNIC
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0

- **DCONNIC: Device Connection Interrupt Flag Clear**
- **DDISCIC: Device Disconnection Interrupt Flag Clear**
- **RSTIC: USB Reset Sent Interrupt Flag Clear**
- **RSMEDIC: Downstream Resume Sent Interrupt Flag Clear**
- **RXRSMIC: Upstream Resume Received Interrupt Flag Clear**
- **HSOFIC: Host Start of Frame Interrupt Flag Clear**
- **HWUPIC: Host Wake-Up Interrupt Flag Clear**

26.8.3.4 USB Host Global Interrupt Set Register (UHINTSET)

Offset: 0x040C

Register Name: UHINTSET

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTS	DMA5INTS	DMA4INTS	DMA3INTS	DMA2INTS	DMA1INTS	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	HWUPIS	HSOFIS	RXRSMIS	RSMEDIS	RSTIS	DDISCIS	DCONNIS
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0

- **DCONNIS: Device Connection Interrupt Flag Set**
- **DDISCIS: Device Disconnection Interrupt Flag Set**
- **RSTIS: USB Reset Sent Interrupt Flag Set**
- **RSMEDIS: Downstream Resume Sent Interrupt Flag Set**
- **RXRSMIS: Upstream Resume Received Interrupt Flag Set**
- **HSOFIS: Host Start of Frame Interrupt Flag Set**
- **HWUPIS: Host Wake-Up Interrupt Flag Set**
- **DMAXINTS, X in [1..6]: DMA Channel X Interrupt Flag Set**

26.8.3.5 *USB Host Global Interrupt Enable Register (UHINTE)*

Offset: 0x0410

Register Name: UHINTE

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTE	DMA5INTE	DMA4INTE	DMA3INTE	DMA2INTE	DMA1INTE	–
	r	r	r	r	r	r	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	P6INTE	P5INTE	P4INTE	P3INTE	P2INTE	P1INTE	P0INTE
	r	r	r	r	r	r	r
	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
–	HWUPIE	HSOFIE	RXRSMIE	RSMEDIE	RSTIE	DDISCIE	DCONNIE
	r	r	r	r	r	r	r
	0	0	0	0	0	0	0

- **DCONNIE: Device Connection Interrupt Enable**
- **DDISCIE: Device Disconnection Interrupt Enable**
- **RSTIE: USB Reset Sent Interrupt Enable**
- **RSMEDIE: Downstream Resume Sent Interrupt Enable**
- **RXRSMIE: Upstream Resume Received Interrupt Enable**
- **HSOFIE: Host Start of Frame Interrupt Enable**
- **HWUPIE: Host Wake-Up Interrupt Enable**
- **PXINTE, X in [0..6]: Pipe X Interrupt Enable**
- **DMAXINTE, X in [1..6]: DMA Channel X Interrupt Enable**

26.8.3.6 *USB Host Global Interrupt Enable Clear Register (UHINTECLR)*

Offset: 0x0414

Register Name: UHINTECLR

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTEC	DMA5INTEC	DMA4INTEC	DMA3INTEC	DMA2INTEC	DMA1INTEC	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	P6INTEC	P5INTEC	P4INTEC	P3INTEC	P2INTEC	P1INTEC	P0INTEC
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
–	HWUPIEC	HSOFIEC	RXRSMIEC	RSMEDIEC	RSTIEC	DDISCIEC	DCONNIEC
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0

- **DCONNIEC: Device Connection Interrupt Enable Clear**
- **DDISCIEC: Device Disconnection Interrupt Enable Clear**
- **RSTIEC: USB Reset Sent Interrupt Enable Clear**
- **RSMEDIEC: Downstream Resume Sent Interrupt Enable Clear**
- **RXRSMIEC: Upstream Resume Received Interrupt Enable Clear**
- **HSOFIEC: Host Start of Frame Interrupt Enable Clear**
- **HWUPIEC: Host Wake-Up Interrupt Enable Clear**
- **PXINTEC, X in [0..6]: Pipe X Interrupt Enable Clear**
- **DMAXINTEC, X in [1..6]: DMA Channel X Interrupt Enable Clear**

26.8.3.7 USB Host Global Interrupt Enable Set Register (UHINTESET)

Offset: 0x0418

Register Name: UHINTESET

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	DMA6INTES	DMA5INTES	DMA4INTES	DMA3INTES	DMA2INTES	DMA1INTES	–
	w	w	w	w	w	w	
	0	0	0	0	0	0	
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	P6INTES	P5INTES	P4INTES	P3INTES	P2INTES	P1INTES	P0INTES
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
–	HWUPIES	HSOFIES	RXRSMIES	RSMEDIES	RSTIES	DDISCIES	DCONNIES
	w	w	w	w	w	w	w
	0	0	0	0	0	0	0

- **DCONNIES: Device Connection Interrupt Enable Set**
- **DDISCIES: Device Disconnection Interrupt Enable Set**
- **RSTIES: USB Reset Sent Interrupt Enable Set**
- **RSMEDIES: Downstream Resume Sent Interrupt Enable Set**
- **RXRSMIES: Upstream Resume Received Interrupt Enable Set**
- **HSOFIES: Host Start of Frame Interrupt Enable Set**
- **HWUPIES: Host Wake-Up Interrupt Enable Set**
- **PXINTES, X in [0..6]: Pipe X Interrupt Enable Set**
- **DMAXINTES, X in [1..6]: DMA Channel X Interrupt Enable Set**

26.8.3.8 USB Host Frame Number Register (UHFNUM)

Offset: 0x0420

Register Name: UHFNUM

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16
FLENHIGH							
ru							
0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8
–	–	FNUM					
		rwu					
		0	0	0	0	0	0

7	6	5	4	3	2	1	0
FNUM					–	–	–
rwu							
0	0	0	0	0			

- **FNUM: Frame Number**

- **FLENHIGH: Frame Length**

26.8.3.9 USB Pipe Enable/Reset Register (UPRST)

Offset: 0x0041C

Register Name: UPRST

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	PRST6	PRST5	PRST4	PRST3	PRST2	PRST1	PRST0
	rwu	rwu	rwu	rwu	rwu	rwu	rwu
	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0
	rw	rw	rw	rw	rw	rw	rw
	0	0	0	0	0	0	0

- **PENX, X in [0..6]: Pipe X Enable**

- **PRSTX, X in [0..6]: Pipe X Reset**

26.8.3.10 USB Pipe X Configuration Register (UPCFGX)

Offset: 0x0500 + X . 0x04

Register Name: UPCFGX, X in [0..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
INTFRQ							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
–	–	–	–	PEPNUM			
				rwu			
				0	0	0	0
15	14	13	12	11	10	9	8
–	–	PTYPE		–	AUTOSW	PTOKEN	
		rwu			rwu	rwu	
		0	0		0	0	0
7	6	5	4	3	2	1	0
–	PSIZE			PBK		ALLOC	–
	rwu			rwu		rwu	
	0	0	0	0	0	0	

- **ALLOC: Pipe Memory Allocate**
- **PBK: Pipe Banks**
- **PSIZE: Pipe Size**
- **PTOKEN: Pipe Token**
- **AUTOSW: Automatic Switch**
- **PTYPE: Pipe Type**
- **PEPNUM: Pipe Endpoint Number**
- **INTFRQ: Pipe Interrupt Request Frequency**

26.8.3.11 USB Pipe X Status Register (UPSTAX)

Offset: 0x0530 + X . 0x04

Register Name: UPSTAX, X in [0..6]

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	PBYCT						
				r			
	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
PBYCT				–	CFGOK	–	RWALL
	r				r		r
0	0	0	0		0		0
15	14	13	12	11	10	9	8
CURRBK		NBUSYBK		–	–	DTSEQ	
r		r				r	
0	0	0	0			0	0
7	6	5	4	3	2	1	0
SHORT PACKETI	RXSTALLDI/ CRCERRI	OVERFI	NAKEDI	PERRI	TXSTPI/ UNDERFI	TXOUTI	RXINI
r	r	r	r	r	r	r	r
0	0	0	0	0	0	0	0

- **RXINI: Received IN Data Interrupt Flag**
- **TXOUTI: Transmitted OUT Data Interrupt Flag**
- **TXSTPI: Transmitted SETUP Interrupt Flag**
- **UNDERFI: Underflow Interrupt Flag**
- **PERRI: Pipe Error Interrupt Flag**
- **NAKEDI: NAKed Interrupt Flag**
- **OVERFI: Overflow Interrupt Flag**
- **RXSTALLDI: Received STALLed Interrupt Flag**
- **CRCERRI: CRC Error Interrupt Flag**

- **SHORTPACKETI: Short Packet Interrupt Flag**
- **DTSEQ: Data Toggle Sequence**
- **NBUSYBK: Number of Busy Banks**
- **CURRBK: Current Bank**
- **RWALL: Read/Write Allowed**
- **CFGOK: Configuration OK Status**
- **PBYCT: Pipe Byte Count**

26.8.3.12 USB Pipe X Status Clear Register (UPSTAXCLR)

Offset: 0x0560 + X . 0x04

Register Name: UPSTAXCLR, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SHORT PACKETIC	RXSTALLDIC/ CRCERRIC	OVERFIC	NAKEDIC	–	TXSTPIC/ UNDERFIC	TXOUTIC	RXINIC
w	w	w	w		w	w	w
0	0	0	0		0	0	0

- **RXINIC: Received IN Data Interrupt Flag Clear**
- **TXOUTIC: Transmitted OUT Data Interrupt Flag Clear**
- **TXSTPIC: Transmitted SETUP Interrupt Flag Clear**
- **UNDERFIC: Underflow Interrupt Flag Clear**
- **NAKEDIC: NAKed Interrupt Flag Clear**
- **OVERFIC: Overflow Interrupt Flag Clear**
- **RXSTALLDIC: Received STALLed Interrupt Flag Clear**
- **CRCERRIC: CRC Error Interrupt Flag Clear**
- **SHORTPACKETIC: Short Packet Interrupt Flag Clear**

26.8.3.13 USB Pipe X Status Set Register (UPSTAXSET)

Offset: 0x0590 + X . 0x04

Register Name: UPSTAXSET, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKS	–	–	–	–
			w				
			0				
7	6	5	4	3	2	1	0
SHORT PACKETIS	RXSTALLDIS/ CRCERRIS	OVERFIS	NAKEDIS	PERRIS	TXSTPIS/ UNDERFIS	TXOUTIS	RXINIS
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **RXINIS: Received IN Data Interrupt Flag Set**
- **TXOUTIS: Transmitted OUT Data Interrupt Flag Set**
- **TXSTPIS: Transmitted SETUP Interrupt Flag Set**
- **UNDERFIS: Underflow Interrupt Flag Set**
- **PERRIS: Pipe Error Interrupt Flag Set**
- **NAKEDIS: NAKed Interrupt Flag Set**
- **OVERFIS: Overflow Interrupt Flag Set**
- **RXSTALLDIS: Received STALLed Interrupt Flag Set**
- **CRCERRIS: CRC Error Interrupt Flag Set**

- **SHORTPACKETIS: Short Packet Interrupt Flag Set**
- **NBUSYBKS: Number of Busy Banks Interrupt Flag Set**

26.8.3.14 USB Pipe X Control Register (UPCONX)

Offset: 0x05C0 + X . 0x04

Register Name: UPCONX, X in [0..6]

Access Type: Read-Only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDT	PFREEZE	PDISHDMA
					ru	ru	ru
					0	0	0
15	14	13	12	11	10	9	8
–	FIFOCON	–	NBUSYBKE	–	–	–	–
	ru		ru				
	0		0				
7	6	5	4	3	2	1	0
SHORT PACKETIE	RXSTALLDE/ CRCERRE	OVERFIE	NAKEDE	PERRE	TXSTPE/ UNDERFIE	TXOUTE	RXINE
ru	ru	ru	ru	ru	ru	ru	ru
0	0	0	0	0	0	0	0

- **RXINE: Received IN Data Interrupt Enable**
- **TXOUTE: Transmitted OUT Data Interrupt Enable**
- **TXSTPE: Transmitted SETUP Interrupt Enable**
- **UNDERFIE: Underflow Interrupt Enable**
- **PERRE: Pipe Error Interrupt Enable**
- **NAKEDE: NAKed Interrupt Enable**
- **OVERFIE: Overflow Interrupt Enable**
- **RXSTALLDE: Received STALLed Interrupt Enable**
- **CRCERRE: CRC Error Interrupt Enable**

- **SHORTPACKETIE: Short Packet Interrupt Enable**
- **NBUSYBKE: Number of Busy Banks Interrupt Enable**
- **FIFOCON: FIFO Control**
- **PDISHDMA: Pipe Interrupts Disable HDMA Request Enable**
- **PFREEZE: Pipe Freeze**
- **RSTDT: Reset Data Toggle**

26.8.3.15 USB Pipe X Control Clear Register (UPCONXCLR)

Offset: 0x0620 + X . 0x04

Register Name: UPCONXCLR, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	PFREEZEC	PDISHDMAC
						w	w
						0	0
15	14	13	12	11	10	9	8
–	FIFOCONC	–	NBUSYBKEC	–	–	–	–
	w		w				
	0		0				
7	6	5	4	3	2	1	0
SHORT PACKETIEC	RXSTALLDEC/ CRCERREC	OVERFIEC	NAKEDEC	PERREC	TXSTPEC/ UNDERFIEC	TXOUTEC	RXINEC
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **RXINEC: Received IN Data Interrupt Enable Clear**
- **TXOUTEC: Transmitted OUT Data Interrupt Enable Clear**
- **TXSTPEC: Transmitted SETUP Interrupt Enable Clear**
- **UNDERFIEC: Underflow Interrupt Enable Clear**
- **PERREC: Pipe Error Interrupt Enable Clear**
- **NAKEDEC: NAKed Interrupt Enable Clear**
- **OVERFIEC: Overflow Interrupt Enable Clear**
- **RXSTALLDEC: Received STALLed Interrupt Enable Clear**
- **CRCERREC: CRC Error Interrupt Enable Clear**

- **SHORTPACKETIEC: Short Packet Interrupt Enable Clear**
- **NBUSYBKEC: Number of Busy Banks Interrupt Enable Clear**
- **FIFOCONC: FIFO Control Clear**
- **PDISHDMAC: Pipe Interrupts Disable HDMA Request Enable Clear**
- **PFREEZEC: Pipe Freeze Clear**

26.8.3.16 USB Pipe X Control Set Register (UPCONXSET)

Offset: 0x05F0 + X . 0x04

Register Name: UPCONXSET, X in [0..6]

Access Type: Write-Only

Read Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RSTDTS	PFREEZES	PDISHDMAS
					w	w	w
					0	0	0
15	14	13	12	11	10	9	8
–	–	–	NBUSYBKES	–	–	–	–
			w				
			0				
7	6	5	4	3	2	1	0
SHORT PACKETIES	RXSTALLDES/ CRCERRES	OVERFIES	NAKEDES	PERRES	TXSTPES/ UNDERFIES	TXOUTES	RXINES
w	w	w	w	w	w	w	w
0	0	0	0	0	0	0	0

- **RXINES: Received IN Data Interrupt Enable Set**
- **TXOUTES: Transmitted OUT Data Interrupt Enable Set**
- **TXSTPES: Transmitted SETUP Interrupt Enable Set**
- **UNDERFIES: Underflow Interrupt Enable Set**
- **PERRES: Pipe Error Interrupt Enable Set**
- **NAKEDES: NAKed Interrupt Enable Set**
- **OVERFIES: Overflow Interrupt Enable Set**
- **RXSTALLDES: Received STALLed Interrupt Enable Set**
- **CRCERRES: CRC Error Interrupt Enable Set**

- **SHORTPACKETIES: Short Packet Interrupt Enable Set**
- **NBUSYBKES: Number of Busy Banks Interrupt Enable Set**
- **PDISHDMAS: Pipe Interrupts Disable HDMA Request Enable Set**
- **PFREEZES: Pipe Freeze Set**
- **RSTDTS: Reset Data Toggle Set**

26.8.3.17 USB Pipe X IN Request Register (UPINRQX)

Offset: 0x0650 + X . 0x04

Register Name: UPINRQX, X in [0..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	INMODE
0	0	0	0	0	0	0	rw
							0

7	6	5	4	3	2	1	0
INRQ							
rwu							
0	0	0	0	0	0	0	0

- **INRQ: IN Request Number before Freeze**

- **INMODE: IN Request Mode**

26.8.3.18 USB Pipe X Error Register (UPERRX)

Offset: 0x0680 + X . 0x04

Register Name: UPERRX, X in [0..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	COUNTER		CRC16	TIMEOUT	PID	DATAPID	DATATGL
	rwu		rwu	rwu	rwu	rwu	rwu
	0	0	0	0	0	0	0

- **DATATGL: Data Toggle Error**
- **DATAPID: Data PID Error**
- **PID: PID Error**
- **TIMEOUT: Time-Out Error**
- **CRC16: CRC16 Error**
- **COUNTER: Error Counter**

26.8.3.19 USB Host DMA Channel X Next Descriptor Address Register (UHDMAX_NEXTDESC)

Offset: 0x0710 + (X - 1) . 0x10

Register Name: *UHDMAX_NEXTDESC*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
NXT_DESC_ADDR							
rwu							
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
NXT_DESC_ADDR				–	–	–	–
rwu							
0	0	0	0				

Same as ”USB Device DMA Channel X Next Descriptor Address Register (UDDMAX_NEXTDESC)” on page 428.

26.8.3.20 USB Host DMA Channel X HSB Address Register (UHDMAX_ADDR)

Offset: 0x0714 + (X - 1) . 0x10

Register Name: *UHDMAX_ADDR*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
HSB_ADDR							
rwu							
0	0	0	0	0	0	0	0

Same as "USB Device DMA Channel X HSB Address Register (UDDMAX_ADDR)" on page 429.

26.8.3.21 USB Host DMA Channel X Control Register (UHDMAX_CONTROL)

Offset: 0x0718 + (X - 1) . 0x10

Register Name: *UHDMAX_CONTROL*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH_BYTE_LENGTH							
rwu							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
CH_BYTE_LENGTH							
rwu							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
BURST_LOCK_EN	DESC_LD_IRQ_EN	EOBUFF_IRQ_EN	EOT_IRQ_EN	DMAEND_EN	BUFF_CLOSE_IN_EN	LD_NXT_CH_DESC_EN	CH_EN
rwu	rwu	rwu	rwu	rwu	rwu	rwu	rwu
0	0	0	0	0	0	0	0

Same as "USB Device DMA Channel X Control Register (UDDMAX_CONTROL)" on page 430.

26.8.3.22 *USB Host DMA Channel X Status Register (UHDMAX_STATUS)*

Offset: 0x071C + (X - 1) . 0x10

Register Name: *UHDMAX_STATUS*, X in [1..6]

Access Type: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
CH_BYTE_CNT							
ru							
0	0	0	0	0	0	0	0
23	22	21	20	19	18	17	16
CH_BYTE_CNT							
ru							
0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	DESC_LD_STA	EOCH_BUFF_STA	EOT_STA	–	–	CH_ACTIVE	CH_EN
	ru	ru	ru			rwu	rwu
	0	0	0			0	0

Same as "USB Device DMA Channel X Status Register (UDDMAX_STATUS)" on page 431.

26.8.4 USB Pipe/Endpoint X FIFO Data Register (USB_FIFOX_DATA)

Note that this register can be accessed even if USBE = 0 or FRZCLK = 1. Disabling the USB controller (by clearing the USBE bit) does not reset the DPRAM.

27. Timer/Counter (TC)

Rev: 2.2.2.0

27.1 Features

- **Three 16-bit Timer Counter Channels**
- **A Wide Range of Functions Including:**
  - **Frequency Measurement**
  - **Event Counting**
  - **Interval Measurement**
  - **Pulse Generation**
  - **Delay Timing**
  - **Pulse Width Modulation**
  - **Up/down Capabilities**
- **Each Channel is User-configurable and Contains:**
  - **Three External Clock Inputs**
  - **Five Internal Clock Inputs**
  - **Two Multi-purpose Input/Output Signals**
- **Internal Interrupt Signal**
- **Two Global Registers that Act on All Three TC Channels**

27.2 Description

The Timer Counter (TC) includes three identical 16-bit Timer Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

Each channel has three external clock inputs, five internal clock inputs and two multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts.

The Timer Counter block has two global registers which act upon all three TC channels.

The Block Control Register allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

27.3 Block Diagram

Figure 27-1. Timer Counter Block Diagram

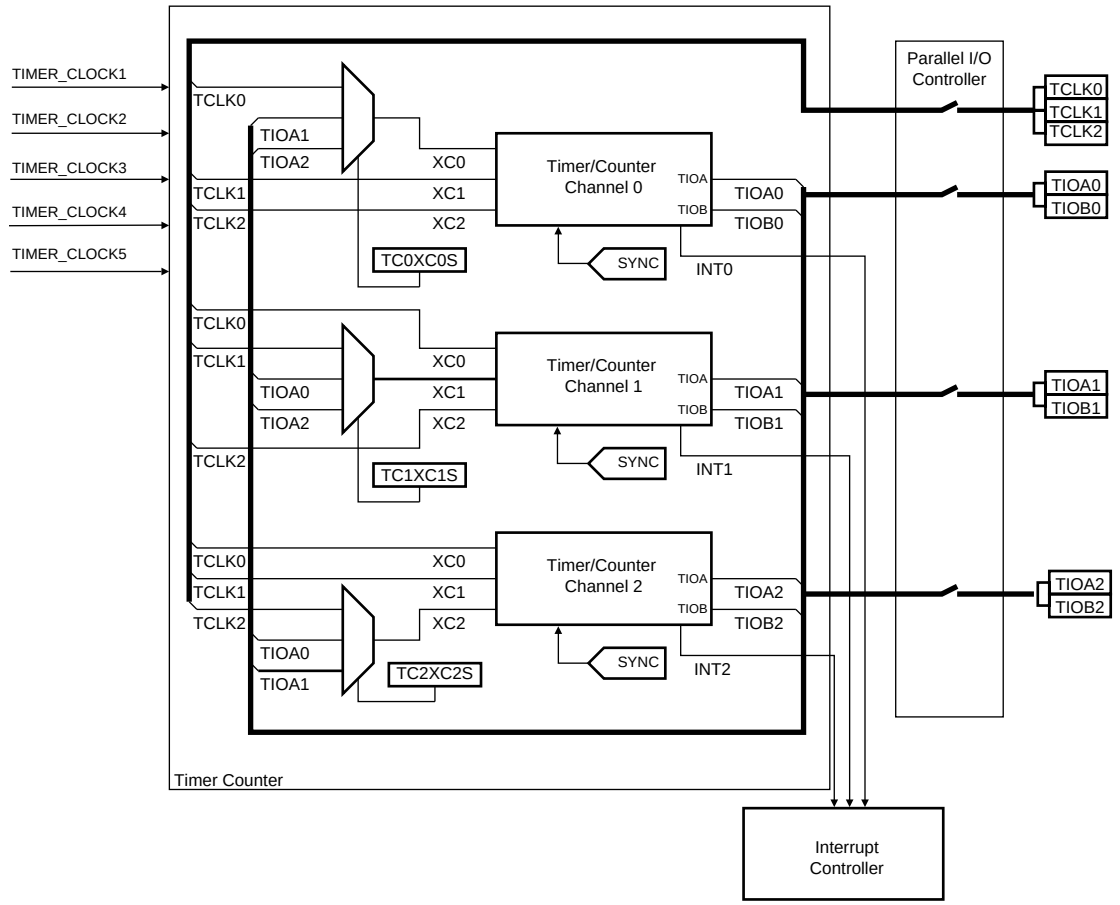


Table 27-1. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

27.4 Pin Name List

Table 27-2. TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

27.5 Product Dependencies

27.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

27.5.2 Power Management

The Timer Counter clock is generated by the power manager. Before using the TC, the programmer must ensure that the TC clock is enabled in the power manager.

27.5.3 Interrupt

The TC has an interrupt line connected to the interrupt controller. Handling the TC interrupt requires programming the interrupt controller before configuring the TC.

27.6 Functional Description

27.6.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 27-4 on page 476.

27.6.1.1 16-bit Counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in SR (Status Register) is set.

The current value of the counter is accessible in real time by reading the Counter Value Register, CV. The counter can be reset by a trigger. In this case, the counter value passes to 0x0000 on the next valid edge of the selected clock.

27.6.1.2 Clock Selection

At block level, input clock signals of each channel can either be connected to the external inputs TCLK0, TCLK1 or TCLK2, or be connected to the configurable I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the BMR (Block Mode). See Figure 27-2.

Each channel can independently select an internal or external clock source for its counter:

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5. The Peripherals Chapter details the connection of these clock sources.
- External clock signals: XC0, XC1 or XC2. The Peripherals Chapter details the connection of these clock sources.

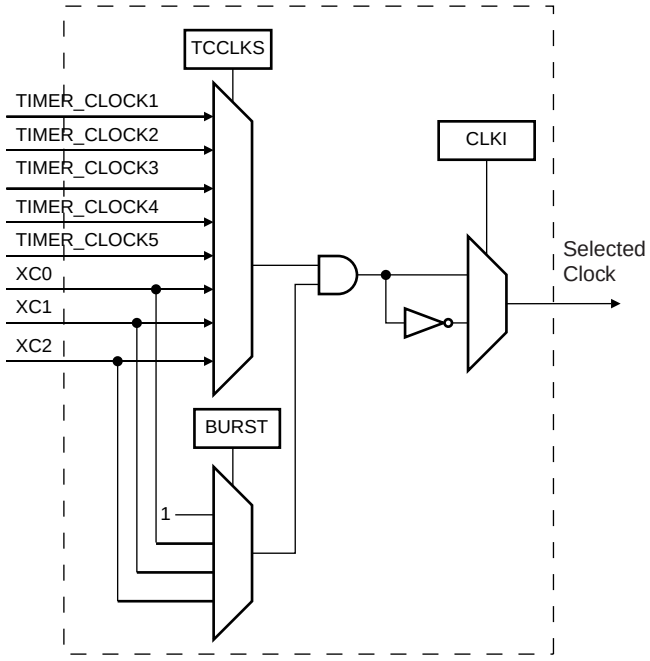
This selection is made by the TCCLKS bits in the TC Channel Mode Register .

The selected clock can be inverted with the CLKI bit in CMR. This allows counting on the opposite edges of the clock.

The burst function allows the clock to be validated when an external signal is high. The BURST parameter in the Mode Register defines this signal (none, XC0, XC1, XC2).

Note: In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock

Figure 27-2. Clock Selection



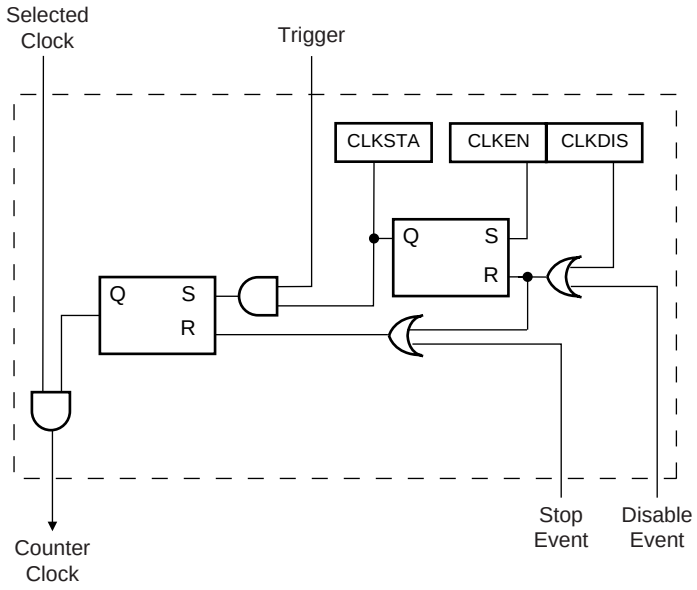
27.6.1.3 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 27-3.

- The clock can be enabled or disabled by the user with the CLKEN and the CLKDIS commands in the Control Register. In Capture Mode it can be disabled by an RB load event if LDBDIS is set to 1 in CMR. In Waveform Mode, it can be disabled by an RC Compare event if CPCDIS is set to 1 in CMR. When disabled, the start or the stop actions have no effect: only a CLKEN command in the Control Register can re-enable the clock. When the clock is enabled, the CLKSTA bit is set in the Status Register.
- The clock can also be started or stopped: a trigger (software, synchro, external or compare) always starts the clock. The clock can be stopped by an RB load event in Capture Mode

(LDBSTOP = 1 in CMR) or a RC compare event in Waveform Mode (CPCSTOP = 1 in CMR). The start and the stop commands have effect only if the clock is enabled.

Figure 27-3. Clock Control



27.6.1.4 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

27.6.1.5 Trigger

A trigger resets the counter and starts the counter clock. Three types of triggers are common to both modes, and a fourth external trigger is available to each mode.

The following triggers are common to both modes:

- Software Trigger: Each channel has a software trigger, available by setting SWTRG in CCR.
- SYNC: Each channel has a synchronization signal SYNC. When asserted, this signal has the same effect as a software trigger. The SYNC signals of all channels are asserted simultaneously by writing BCR (Block Control) with SYNC set.
- Compare RC Trigger: RC is implemented in each channel and can provide a trigger when the counter value matches the RC value if CPCTRG is set in CMR.

The channel can also be configured to have an external trigger. In Capture Mode, the external trigger signal can be selected between TIOA and TIOB. In Waveform Mode, an external event can be programmed on one of the following signals: TIOB, XC0, XC1 or XC2. This external event can then be programmed to perform a trigger by setting ENETRG in CMR.

If an external trigger is used, the duration of the pulses must be longer than the master clock period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

27.6.2 Capture Operating Mode

This mode is entered by clearing the WAVE parameter in CMR (Channel Mode Register).

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 27-4 shows the configuration of the TC channel when programmed in Capture Mode.

27.6.2.1 Capture Registers A and B

Registers A and B (RA and RB) are used as capture registers. This means that they can be loaded with the counter value when a programmable event occurs on the signal TIOA.

The LDRA parameter in CMR defines the TIOA edge for the loading of register A, and the LDRB parameter defines the TIOA edge for the loading of Register B.

RA is loaded only if it has not been loaded since the last trigger or if RB has been loaded since the last loading of RA.

RB is loaded only if RA has been loaded since the last trigger or the last loading of RB.

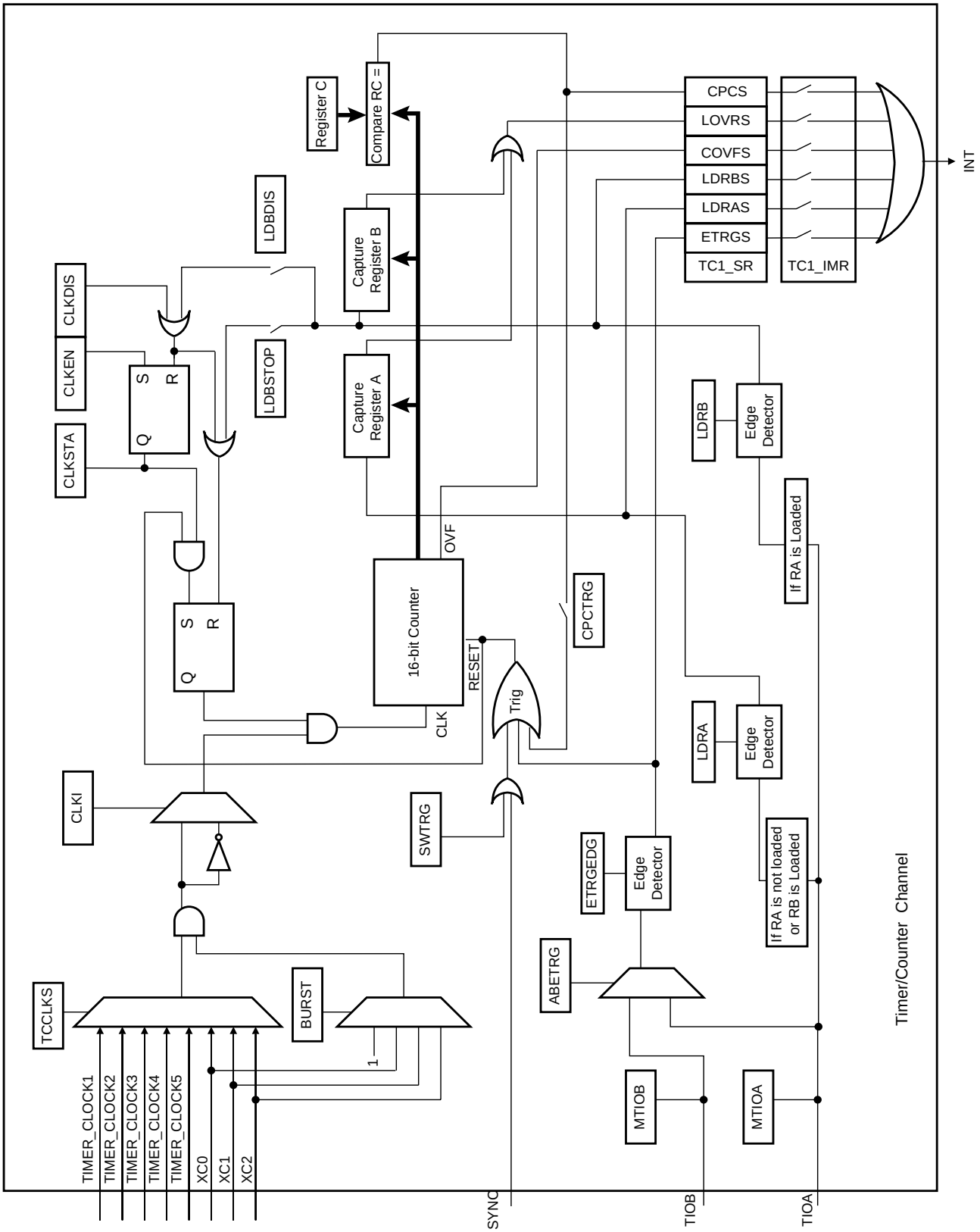
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in SR (Status Register). In this case, the old value is overwritten.

27.6.2.2 Trigger Conditions

In addition to the SYNC signal, the software trigger and the RC compare trigger, an external trigger can be defined.

The ABETRG bit in CMR selects TIOA or TIOB input signal as an external trigger. The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 27-4. Capture Mode



27.6.3 Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in CMR (Channel Mode Register).

In Waveform Operating Mode the TC channel generates 1 or 2 PWM signals with the same frequency and independently programmable duty cycles, or generates different types of one-shot or repetitive pulses.

In this mode, TIOA is configured as an output and TIOB is defined as an output if it is not used as an external event (EEVT parameter in CMR).

Figure 27-5 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

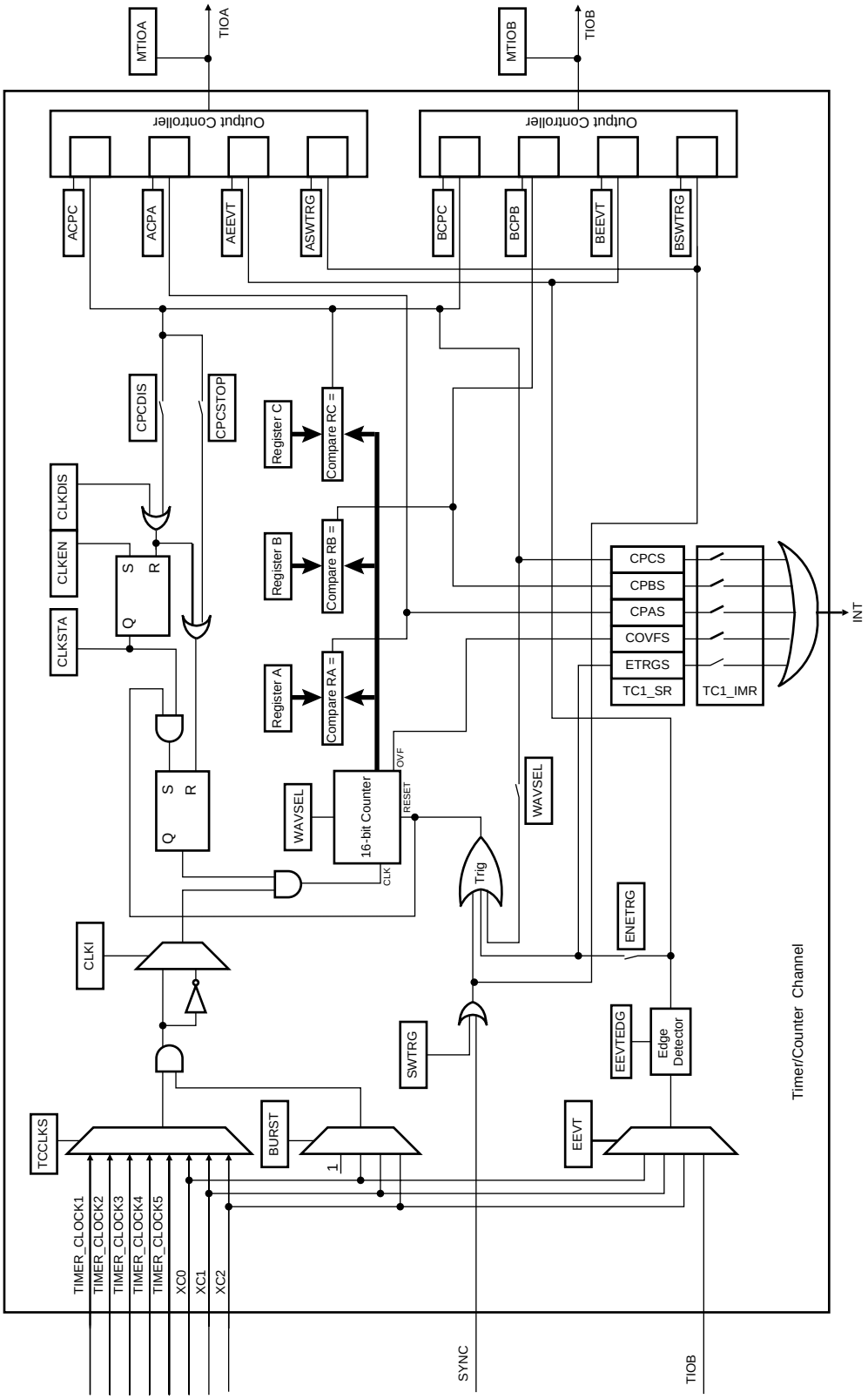
27.6.3.1 Waveform Selection

Depending on the WAVSEL parameter in CMR (Channel Mode Register), the behavior of CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 27-5. Waveform Mode



27.6.3.2 WAVSEL = 00

When WAVSEL = 00, the value of CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of CV is reset. Incrementation of CV starts again and the cycle continues. See Figure 27-6.

An external event trigger or a software trigger can reset the value of CV. It is important to note that the trigger may occur at any time. See Figure 27-7.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in CMR) and/or disable the counter clock (CPCDIS = 1 in CMR).

Figure 27-6. WAVSEL= 00 without trigger

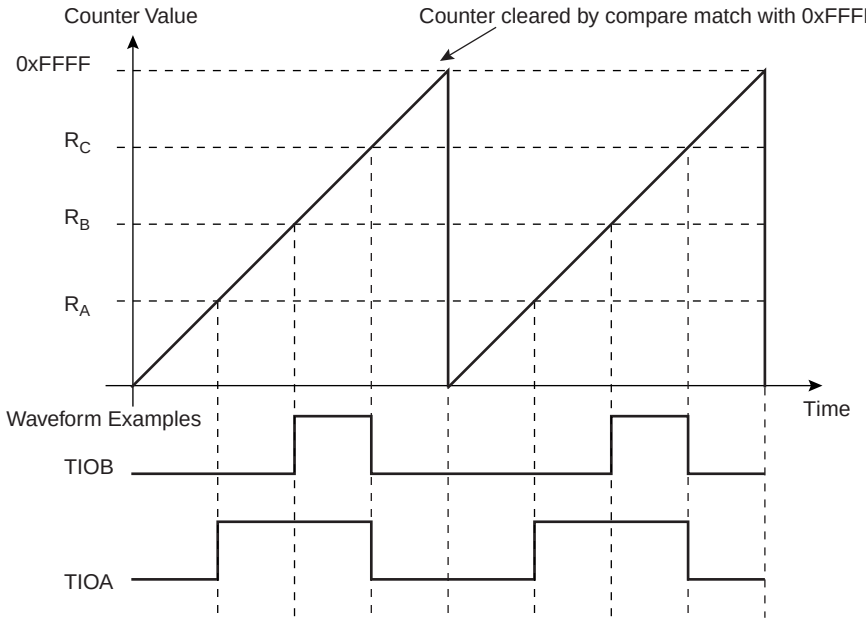
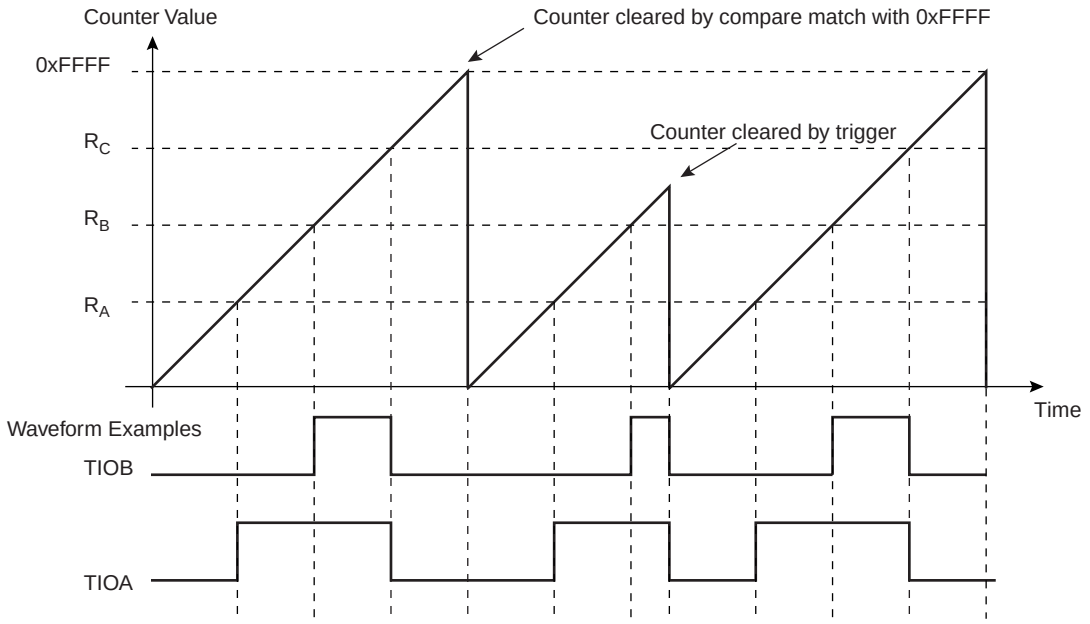


Figure 27-7. WAVSEL= 00 with trigger



27.6.3.3 *WAVSEL = 10*

When WAVSEL = 10, the value of CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of CV has been reset, it is then incremented and so on. See Figure 27-8.

It is important to note that CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 27-9.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in CMR) and/or disable the counter clock (CPCDIS = 1 in CMR).

Figure 27-8. WAVSEL = 10 Without Trigger

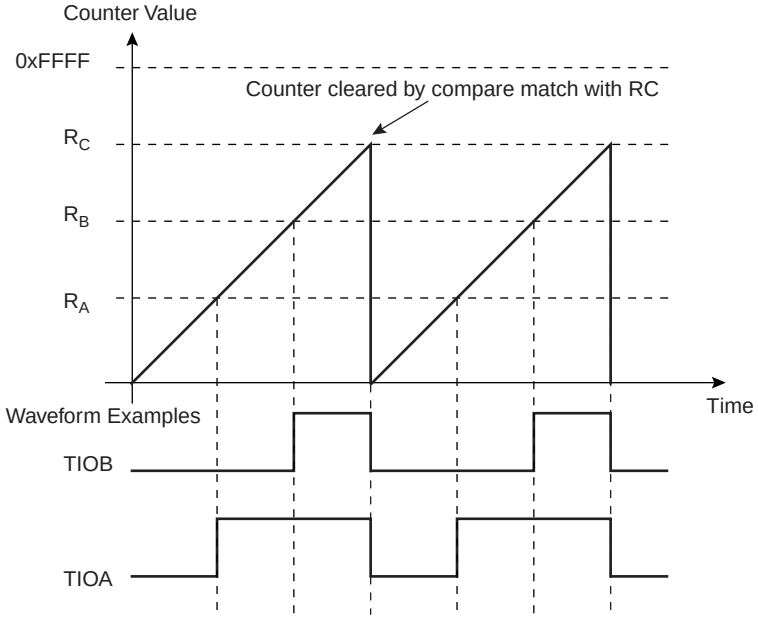
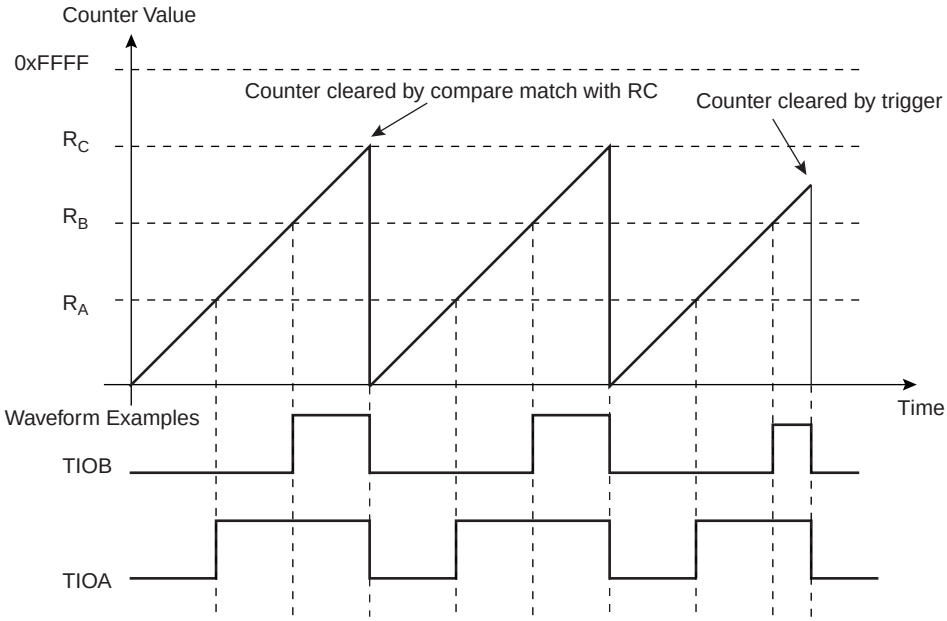


Figure 27-9. WAVSEL = 10 With Trigger



27.6.3.4 *WAVSEL = 01*

When WAVSEL = 01, the value of CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 27-10.

A trigger such as an external event or a software trigger can modify CV at any time. If a trigger occurs while CV is incrementing, CV then decrements. If a trigger is received while CV is decrementing, CV then increments. See Figure 27-11.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 27-10. WAVSEL = 01 Without Trigger

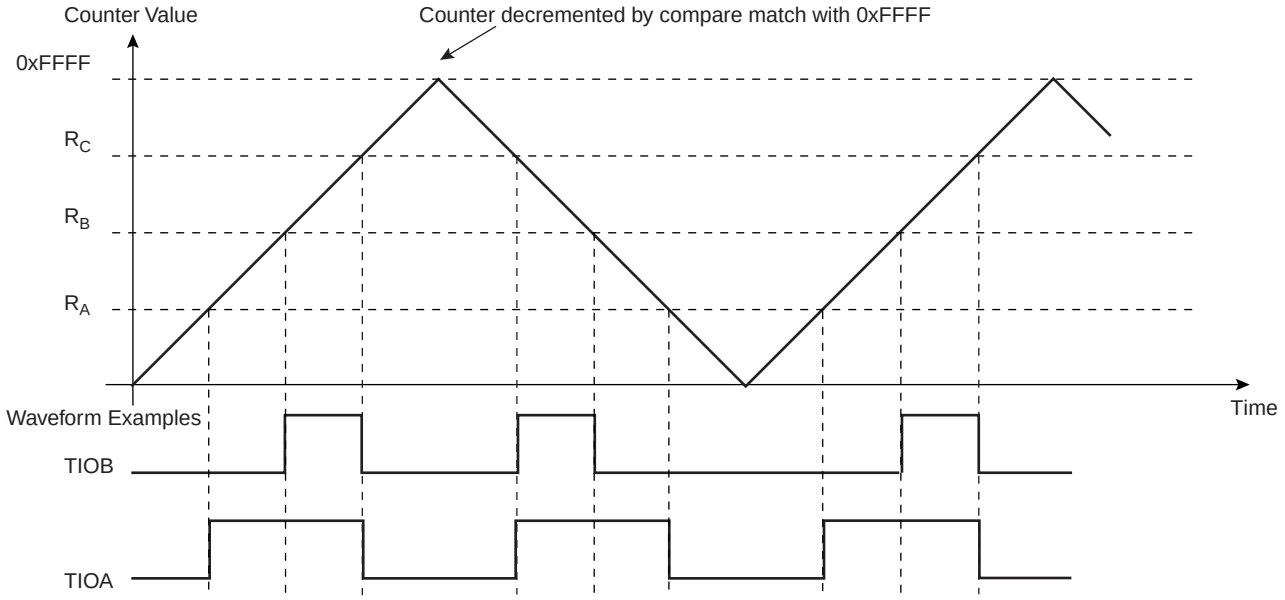
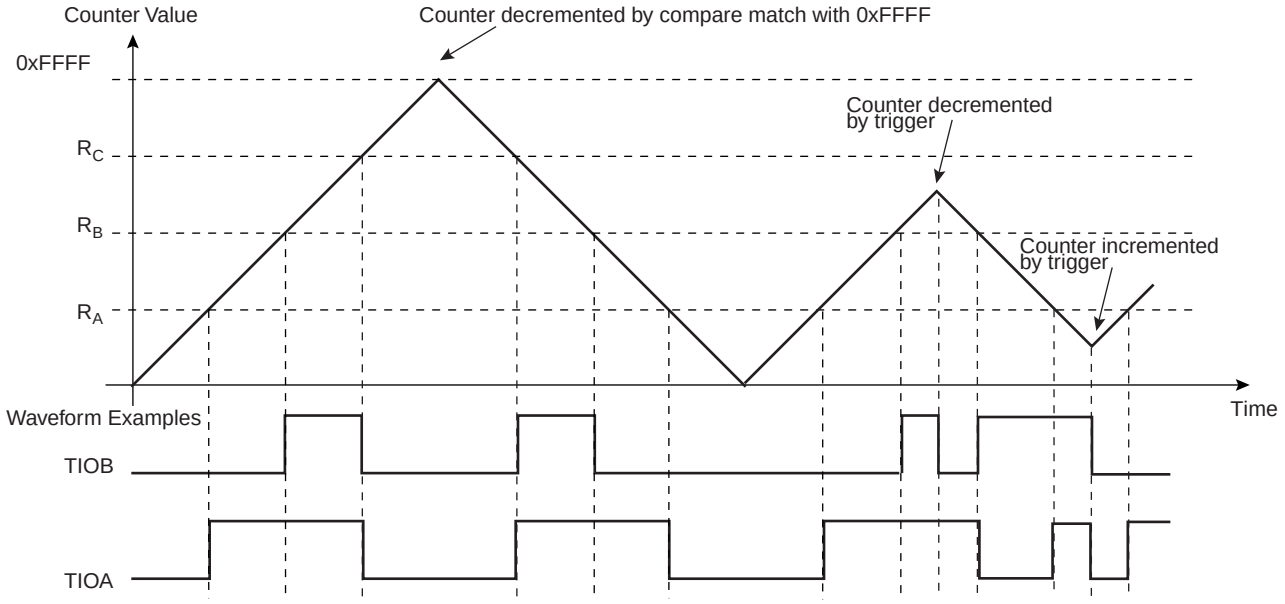


Figure 27-11. WAVSEL = 01 With Trigger



27.6.3.5 WAVSEL = 11

When WAVSEL = 11, the value of CV is incremented from 0 to RC. Once RC is reached, the value of CV is decremented to 0, then re-incremented to RC and so on. See Figure 27-12.

A trigger such as an external event or a software trigger can modify CV at any time. If a trigger occurs while CV is incrementing, CV then decrements. If a trigger is received while CV is decrementing, CV then increments. See Figure 27-13.

RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 27-12. WAVSEL = 11 Without Trigger

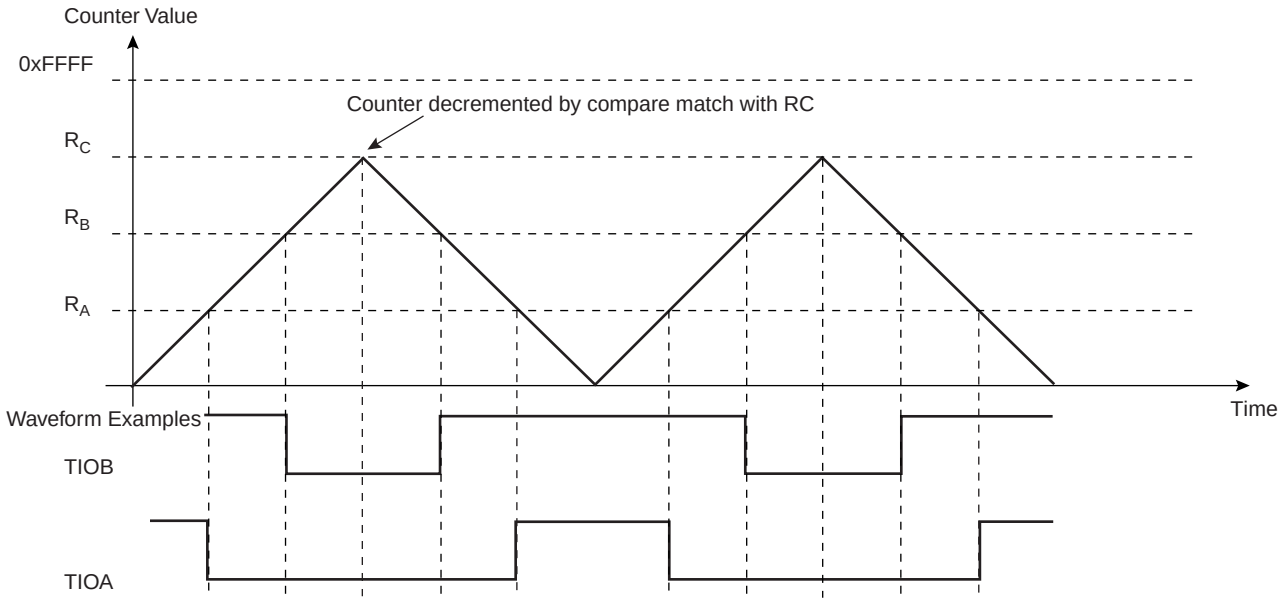
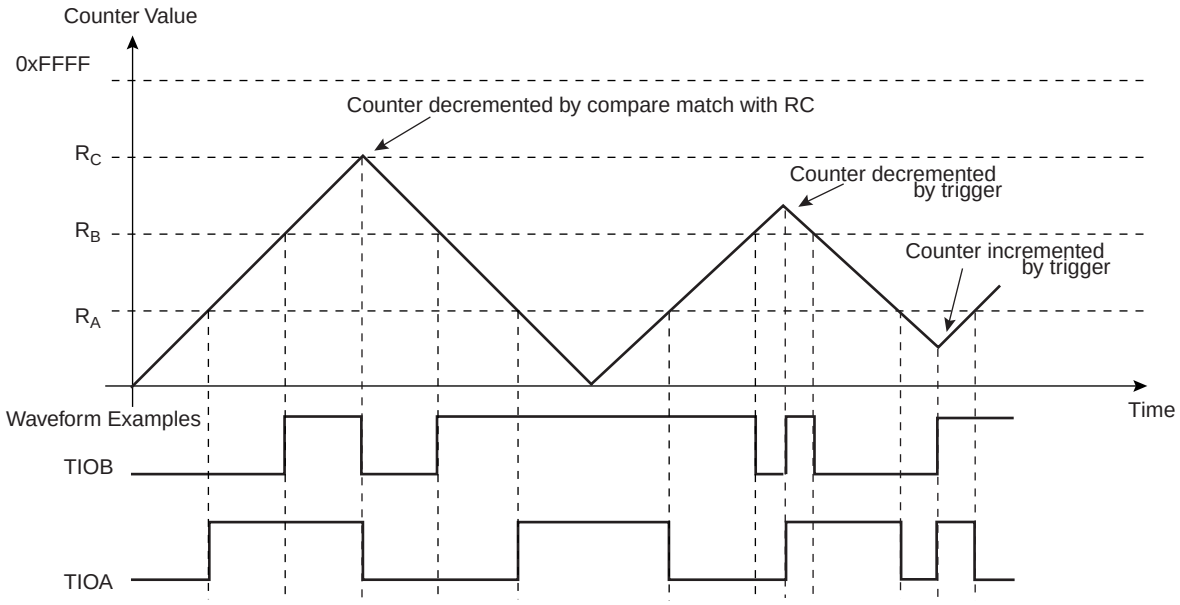


Figure 27-13. WAVSEL = 11 With Trigger



27.6.3.6 External Event/Trigger Conditions

An external event can be programmed to be detected on one of the clock sources (XC0, XC1, XC2) or TIOB. The external event selected can then be used as a trigger.

The EEVT parameter in CMR selects the external trigger. The EEVTEDG parameter defines the trigger edge for each of the possible external triggers (rising, falling or both). If EEVTEDG is cleared (none), no external event is defined.

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

When an external event is defined, it can be used as a trigger by setting bit ENETRG in CMR.

As in Capture Mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

27.6.3.7 Output Controller

The output controller defines the output level changes on TIOA and TIOB following an event. TIOB control is used only if TIOB is defined as output (not as an external event).

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in CMR.

27.7 Timer Counter (TC) User Interface

Table 27-3. TC Global Memory Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0	See Table 27-4		
0x40	TC Channel 1	See Table 27-4		
0x80	TC Channel 2	See Table 27-4		
0xC0	TC Block Control Register	BCR	Write-only	–
0xC4	TC Block Mode Register	BMR	Read/Write	0

BCR (Block Control Register) and BMR (Block Mode Register) control the whole TC block. TC channels are controlled by the registers listed in Table 27-4. The offset of each of the channel registers in Table 27-4 is in relation to the offset of the corresponding channel as mentioned in Table 27-4.

Table 27-4. TC Channel Memory Map

Offset	Register	Name	Access	Reset Value
0x00	Channel Control Register	CCR	Write-only	–
0x04	Channel Mode Register	CMR	Read/Write	0
0x08	Reserved			–
0x0C	Reserved			–
0x10	Counter Value	CV	Read-only	0
0x14	Register A	RA	Read/Write(1)	0
0x18	Register B	RB	Read/Write(1)	0
0x1C	Register C	RC	Read/Write	0
0x20	Status Register	SR	Read-only	0
0x24	Interrupt Enable Register	IER	Write-only	–
0x28	Interrupt Disable Register	IDR	Write-only	–
0x2C	Interrupt Mask Register	IMR	Read-only	0

Notes: 1. Read-only if WAVE = 0

27.7.1 TC Block Control Register

Register Name: BCR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

27.7.2 TC Block Mode Register

Register Name: BMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

- **TC0XC0S: External Clock Signal 0 Selection**

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

27.7.3 TC Channel Control Register

Register Name: CCR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

27.7.4 TC Channel Mode Register: Capture Mode

Register Name: CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	
15	14	13	12	11	10	9	8
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **LDBSTOP: Counter Clock Stopped with RB Loading**

0 = Counter clock is not stopped when RB loading occurs.

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

0 = Counter clock is not disabled when RB loading occurs.

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

LDRA		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

- **LDRB: RB Loading Selection**

LDRB		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

27.7.5 TC Channel Mode Register: Waveform Mode

Register Name: CMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	
23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	
15	14	13	12	11	10	9	8
WAVE = 1	WAVSEL		ENETRG	EEVT		EEVTEDG	
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **CPCSTOP: Counter Clock Stopped with RC Compare**

0 = Counter clock is not stopped when counter reaches RC.

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

EEVTEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **EEVT: External Event Selection**

EEVT		Signal selected as external event	TIOB Direction
0	0	TIOB	input[1]
0	1	XC0	output
1	0	XC1	output
1	1	XC2	output

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms and subsequently no IRQs.*

- **ENETRG: External Event Trigger Enable**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE = 1**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

ACPA		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ACPC: RC Compare Effect on TIOA**

ACPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **AEEVT: External Event Effect on TIOA**

AEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ASWTRG: Software Trigger Effect on TIOA**

ASWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB		Effect
0	0	none

BCPB		Effect
0	1	set
1	0	clear
1	1	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BEEVT: External Event Effect on TIOB**

BEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BSWTRG: Software Trigger Effect on TIOB**

BSWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

27.7.6 TC Counter Value Register

Register Name: CV

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CV							
7	6	5	4	3	2	1	0
CV							

- **CV: Counter Value**

CV contains the counter value in real time.

27.7.7 TC Register A

Register Name: RA

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RA							
7	6	5	4	3	2	1	0
RA							

- **RA: Register A**

RA contains the Register A value in real time.

27.7.8 TC Register B

Register Name: RB

Access Type: Read-only if WAVE = 0, Read/Write if WAVE = 1

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RB							
7	6	5	4	3	2	1	0
RB							

- **RB: Register B**

RB contains the Register B value in real time.

27.7.9 TC Register C

Register Name: RC

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RC							
7	6	5	4	3	2	1	0
RC							

- **RC: Register C**

RC contains the Register C value in real time.

27.7.10 TC Status Register

Register Name: SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow Status**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

27.7.11 TC Interrupt Enable Register

Register Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.

27.7.12 TC Interrupt Disable Register

Register Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

27.7.13 TC Interrupt Mask Register

Register Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.

28. Pulse Width Modulation Controller (PWM)

Rev: 1.2.0.1

28.1 Features

- **4 Channels**
- **One 20-bit Counter Per Channel**
- **Common Clock Generator Providing Thirteen Different Clocks**
  - **A Modulo n Counter Providing Eleven Clocks**
  - **Two Independent Linear Dividers Working on Modulo n Counter Outputs**
- **Independent Channels**
  - **Independent Enable Disable Command for Each Channel**
  - **Independent Clock Selection for Each Channel**
  - **Independent Period and Duty Cycle for Each Channel**
  - **Double Buffering of Period or Duty Cycle for Each Channel**
  - **Programmable Selection of The Output Waveform Polarity for Each Channel**
  - **Programmable Center or Left Aligned Output Waveform for Each Channel**

28.2 Description

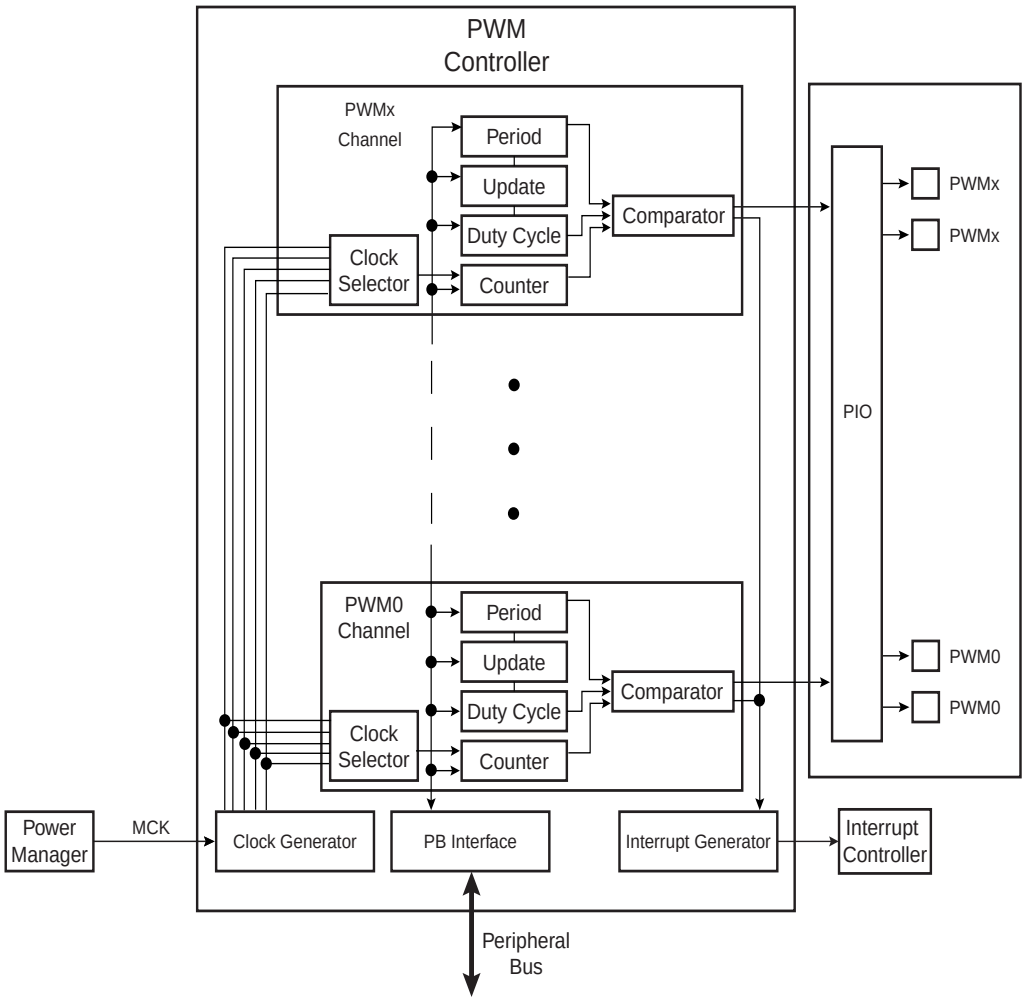
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through registers mapped on the peripheral bus.

Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

28.3 Block Diagram

Figure 28-1. Pulse Width Modulation Controller Block Diagram



28.4 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 28-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

28.5 Product Dependencies

28.5.1 I/O Lines

The pins used for interfacing the PWM may be multiplexed with PIO lines. The programmer must first program the PIO controller to assign the desired PWM pins to their peripheral function. If I/O lines of the PWM are not used by the application, they can be used for other purposes by the PIO controller.

Not all PWM outputs may be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

28.5.2 Power Management

The PWM clock is generated by the Power Manager. Before using the PWM, the programmer must ensure that the PWM clock is enabled in the Power Manager. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

In the PWM description, Master Clock (MCK) is the clock of the peripheral bus to which the PWM is connected.

28.5.3 Interrupt Sources

The PWM interrupt line is connected to the interrupt controller. Using the PWM interrupt requires the interrupt controller to be programmed first.

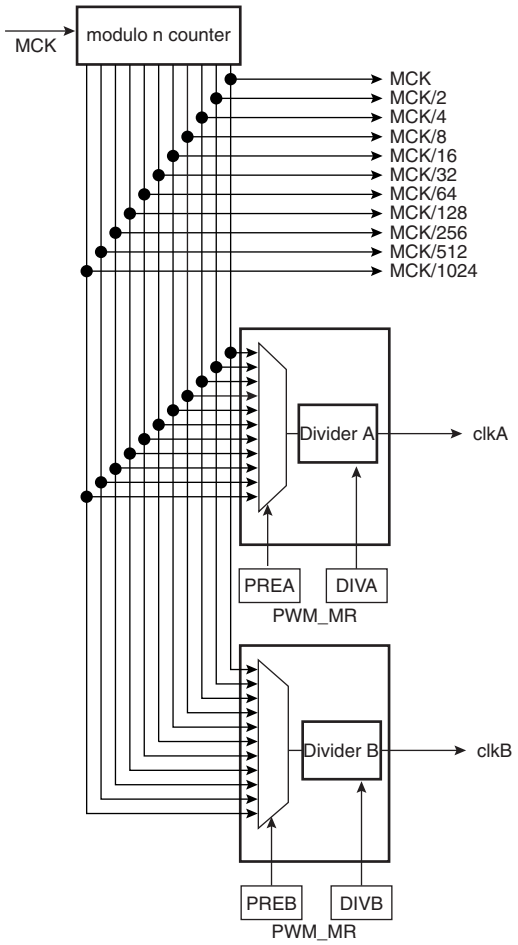
28.6 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the system clock, MCK, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

28.6.1 PWM Clock Generator

Figure 28-2. Functional View of the Clock Generator Block Diagram



Caution: Before using the PWM macrocell, the programmer must ensure that the PWM clock in the Power Manager is enabled.

The PWM macrocell master clock, MCK, is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The clock generator is divided in three blocks:

– a modulo n counter which provides 11 clocks: F_{MCK}, $F_{MCK}/2$, $F_{MCK}/4$, $F_{MCK}/8$, $F_{MCK}/16$, $F_{MCK}/32$, $F_{MCK}/64$, $F_{MCK}/128$, $F_{MCK}/256$, $F_{MCK}/512$, $F_{MCK}/1024$
– two linear dividers (1, 1/2, 1/3, ... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Mode register (MR). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value in the PWM Mode register (MR).

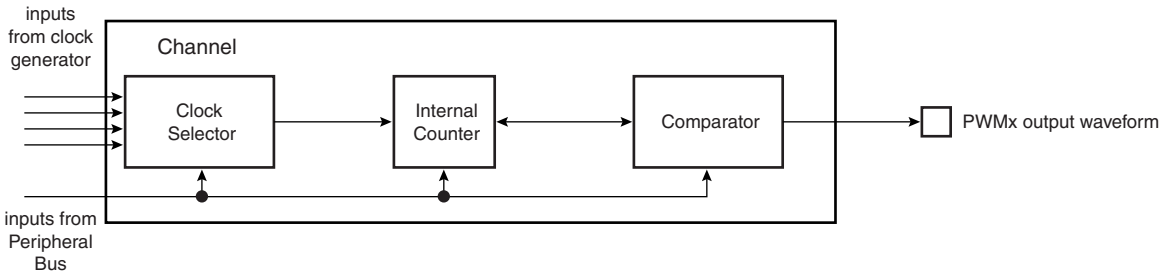
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except clock “clk”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

28.6.2 PWM Channel

28.6.2.1 Block Diagram

Figure 28-3. Functional View of the Channel Block Diagram



Each of the 4 channels is composed of three blocks:

- A clock selector which selects one of the clocks provided by the clock generator described in Section 28.6.1 ”PWM Clock Generator” on page 498.
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 20 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

28.6.2.2 Waveform Properties

The different properties of output waveforms are:

- the ***internal clock selection***. The internal channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the CMRx register. This field is reset at 0.
- the ***waveform period***. This channel parameter is defined in the CPRD field of the CPRDx register.

- If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:
By using the Master Clock (MCK) divided by an X given prescaler value
(with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:
By using the Master Clock (MCK) divided by an X given prescaler value
(with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- the ***waveform duty cycle***. This channel parameter is defined in the CDTY field of the CDTYx register.
If the waveform is left aligned then:

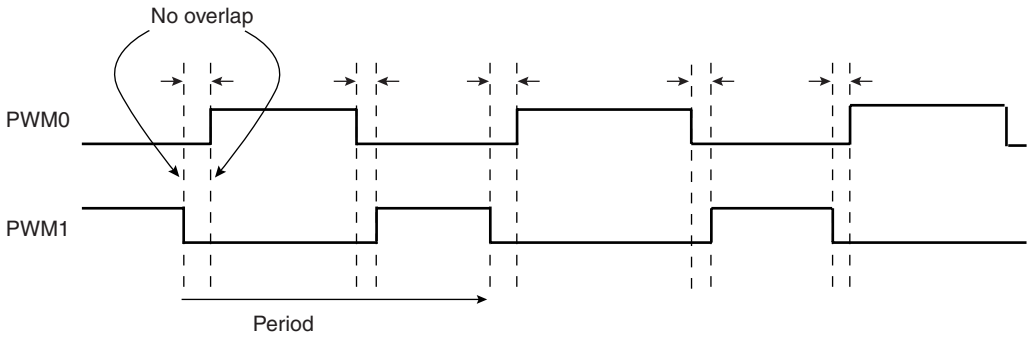
$$\text{duty cycle} = \frac{(period - 1/\text{fchannel\_x\_clock} \times CDTY)}{period}$$

If the waveform is center aligned, then:

$$\text{duty cycle} = \frac{((period/2) - 1/\text{fchannel\_x\_clock} \times CDTY))}{(period/2)}$$

- the ***waveform polarity.*** At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the CMRx register. By default the signal starts by a low level.
- the ***waveform alignment***. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the CMRx register. The default mode is left aligned.

Figure 28-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 28-5 on page 502 for a detailed description of center aligned waveforms.

When center aligned, the internal channel counter increases up to CPRD and.decreases down to 0. This ends the period.

When left aligned, the internal channel counter increases up to CPRD and is reset. This ends the period.

Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

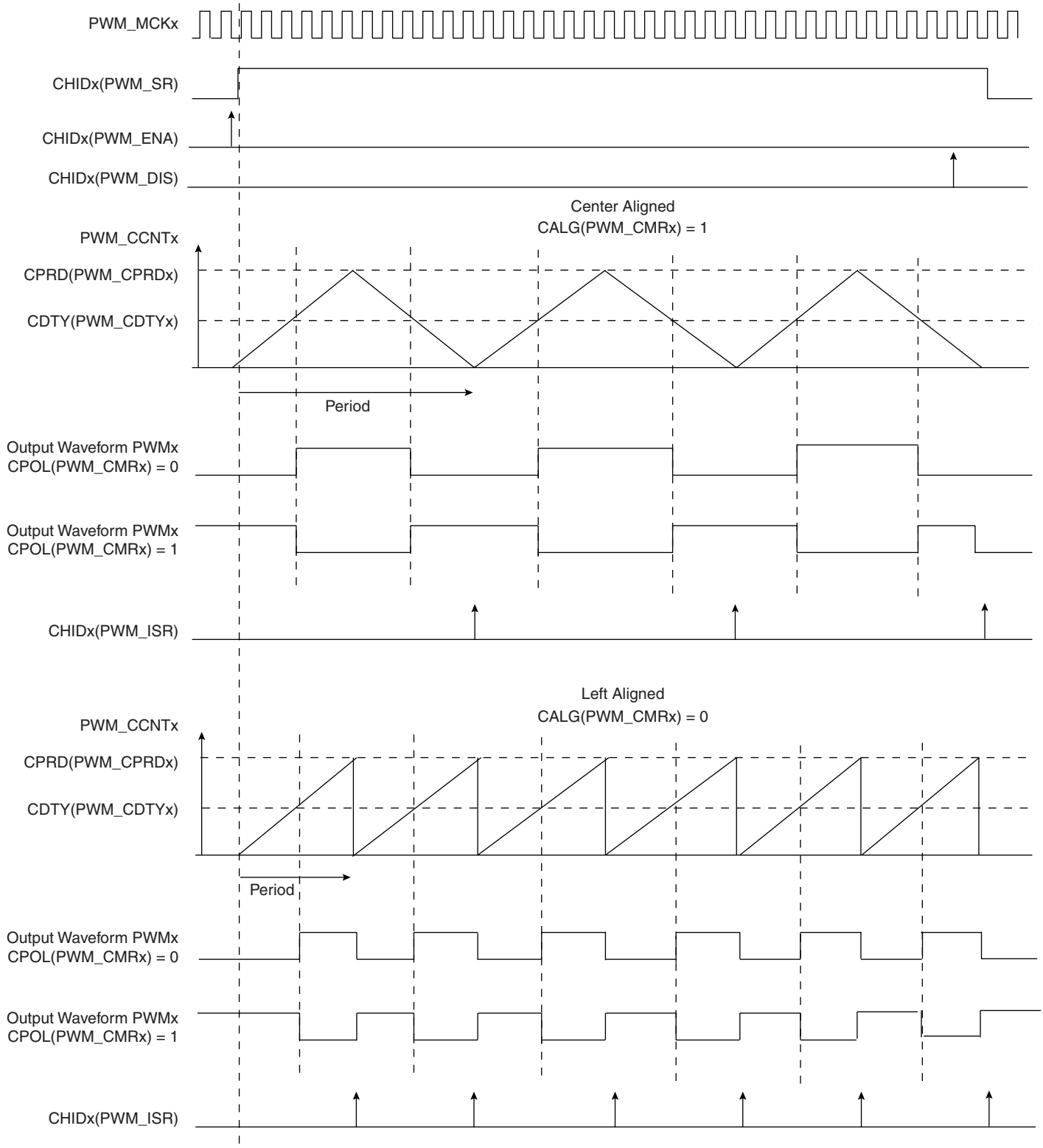
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

Figure 28-5. Waveform Properties



28.6.3 PWM Controller Operations

28.6.3.1 Initialization

Before enabling the output channel, this channel must have been configured by the software application:

- Configuration of the clock generator if DIVA and DIVB are required
- Selection of the clock for each channel (CPRE field in the CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the CMRx register)
- Configuration of the period for each channel (CPRD in the CPRDx register). Writing in CPRDx Register is possible while the channel is disabled. After validation of the channel, the user must use CUPDx Register to update CPRDx as explained below.
- Configuration of the duty cycle for each channel (CDTY in the CDTYx register). Writing in CDTYx Register is possible while the channel is disabled. After validation of the channel, the user must use CUPDx Register to update CDTYx as explained below.
- Configuration of the output waveform polarity for each channel (CPOL in the CMRx register)
- Enable Interrupts (Writing CHIDx in the IER register)
- Enable the PWM channel (Writing CHIDx in the ENA register)

It is possible to synchronize different channels by enabling them at the same time by means of writing simultaneously several CHIDx bits in the ENA register.

In such a situation, all channels may have the same clock selector configuration and the same period specified.

28.6.3.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the Period Register (CPRDx) and the Duty Cycle Register (CDTYx) can help the user in choosing. The event number written in the Period Register gives the PWM accuracy. The Duty Cycle quantum cannot be lower than *1/CPRDx* value. The higher the value of CPRDx, the greater the PWM accuracy.

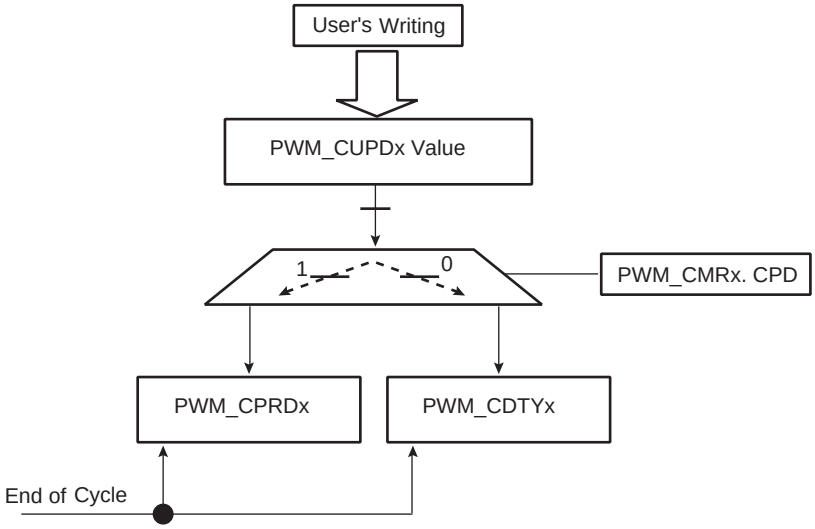
For example, if the user sets 15 (in decimal) in CPRDx, the user is able to set a value between 1 up to 14 in CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

28.6.3.3 Changing the Duty Cycle or the Period

It is possible to modulate the output waveform duty cycle or period.

To prevent unexpected output waveform, the user must use the update register (PWM_CUPDx) to change waveform parameters while the channel is still enabled. The user can write a new period value or duty cycle value in the update register (CUPDx). This register holds the new value until the end of the current cycle and updates the value for the next cycle. Depending on the CPD field in the CMRx register, CUPDx either updates CPRDx or CDTYx. Note that even if the update register is used, the period must not be smaller than the duty cycle.

Figure 28-6. Synchronized Period or Duty Cycle Update



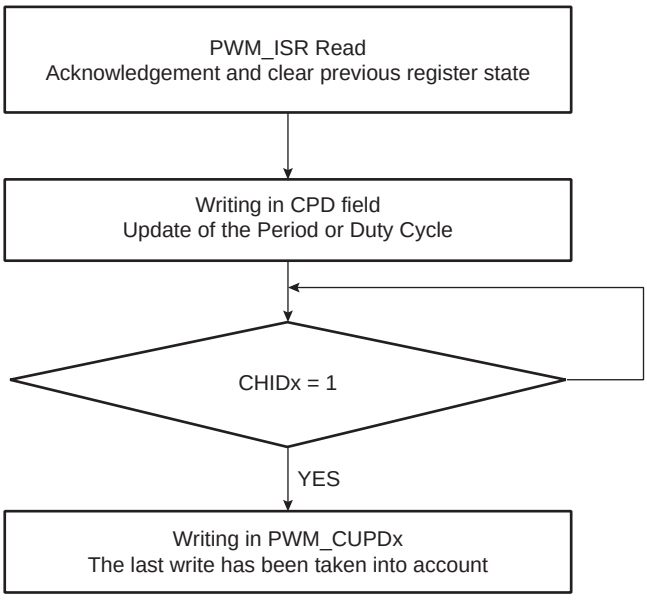
To prevent overwriting the CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in ISR Register according to the enabled channel(s). See Figure 28-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the ISR register automatically clears CHIDx flags.

Figure 28-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

28.6.3.4 Interrupts

Depending on the interrupt mask in the IMR register, an interrupt is generated at the end of the corresponding channel period. The interrupt remains active until a read operation in the ISR register occurs.

A channel interrupt is enabled by setting the corresponding bit in the IER register. A channel interrupt is disabled by setting the corresponding bit in the IDR register.

28.7 Pulse Width Modulation (PWM) Controller User Interface

28.7.1 Register Mapping

Table 28-2. PWM Controller Registers

Offset	Register	Name	Access	Peripheral Reset Value
0x00	PWM Mode Register	MR	Read/Write	0
0x04	PWM Enable Register	ENA	Write-only	-
0x08	PWM Disable Register	DIS	Write-only	-
0x0C	PWM Status Register	SR	Read-only	0
0x10	PWM Interrupt Enable Register	IER	Write-only	-
0x14	PWM Interrupt Disable Register	IDR	Write-only	-
0x18	PWM Interrupt Mask Register	IMR	Read-only	0
0x1C	PWM Interrupt Status Register	ISR	Read-only	0
0x4C - 0xF8	Reserved	–	–	–
0x4C - 0xFC	Reserved	–	–	–
0x100 - 0x1FC	Reserved			
0x200	Channel 0 Mode Register	CMR0	Read/Write	0x0
0x204	Channel 0 Duty Cycle Register	CDTY0	Read/Write	0x0
0x208	Channel 0 Period Register	CPRD0	Read/Write	0x0
0x20C	Channel 0 Counter Register	CCNT0	Read-only	0x0
0x210	Channel 0 Update Register	CUPD0	Write-only	-
...	Reserved			
0x220	Channel 1 Mode Register	CMR1	Read/Write	0x0
0x224	Channel 1 Duty Cycle Register	CDTY1	Read/Write	0x0
0x228	Channel 1 Period Register	CPRD1	Read/Write	0x0
0x22C	Channel 1 Counter Register	CCNT1	Read-only	0x0
0x230	Channel 1 Update Register	CUPD1	Write-only	-
...	...	...	...	...

28.7.2 PWM Mode Register

Register Name: MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	PREB			

23	22	21	20	19	18	17	16
DIVB							

15	14	13	12	11	10	9	8
–	–	–	–	PREA			

7	6	5	4	3	2	1	0
DIVA							

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

DIVA, DIVB	CLKA, CLKB
0	CLKA, CLKB clock is turned off
1	CLKA, CLKB clock is clock selected by PREA, PREB
2-255	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

- **PREA, PREB**

PREA, PREB				Divider Input Clock
0	0	0	0	MCK.
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
Other				Reserved

28.7.3 PWM Enable Register

Register Name: ENA

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

28.7.4 PWM Disable Register

Register Name: DIS

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.

28.7.5 PWM Status Register

Register Name: SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

28.7.6 PWM Interrupt Enable Register

Register Name: IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

28.7.7 PWM Interrupt Disable Register

Register Name: IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

28.7.8 PWM Interrupt Mask Register

Register Name: IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

28.7.9 PWM Interrupt Status Register

Register Name: ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	CHID6	CHID5	CHID4	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No new channel period since the last read of the ISR register.

1 = At least one new channel period since the last read of the ISR register.

Note: Reading ISR automatically clears CHIDx flags.

28.7.10 PWM Channel Mode Register

Register Name: CMRx

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	CPD	CPOL	CALG
7	6	5	4	3	2	1	0
–	–	–	–	CPRE			

- **CPRE: Channel Pre-scaler**

CPRE				Channel Pre-scaler
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
1	0	1	1	CLKA
1	1	0	0	CLKB
Other				Reserved

- **CALG: Channel Alignment**

0 = The period is left aligned.

1 = The period is center aligned.

- **CPOL: Channel Polarity**

0 = The output waveform starts at a low level.

1 = The output waveform starts at a high level.

- **CPD: Channel Update Period**

0 = Writing to the CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the CUPDx will modify the period at the next period start event.

28.7.11 PWM Channel Duty Cycle Register

Register Name: CDTYx

Access Type: Read/Write

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 20 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (CPRx).

28.7.12 PWM Channel Period Register

Register Name: CPRDx

Access Type: Read/Write

31	30	29	28	27	26	25	24
CPRD							
23	22	21	20	19	18	17	16
CPRD							
15	14	13	12	11	10	9	8
CPRD							
7	6	5	4	3	2	1	0
CPRD							

Only the first 20 bits (internal channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

28.7.13 PWM Channel Counter Register

Register Name: CCNTx

Access Type: Read-only

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

- **CNT: Channel Counter Register**

Internal counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the ENA register).
- the counter reaches CPRD value defined in the CPRDx register if the waveform is left aligned.

28.7.14 PWM Channel Update Register

Register Name: CUPDx

Access Type: Write-only

31	30	29	28	27	26	25	24
CUPD							

23	22	21	20	19	18	17	16
CUPD							

15	14	13	12	11	10	9	8
CUPD							

7	6	5	4	3	2	1	0
CUPD							

This register acts as a double buffer for the period or the duty cycle. This prevents an unexpected waveform when modifying the waveform period or duty-cycle.

Only the first 20 bits (internal channel counter size) are significant.

CPD (CMRx Register)	
0	The duty-cycle (CDTY in the CDTYx register) is updated with the CUPD value at the beginning of the next period.
1	The period (CPRD in the CPRDx register) is updated with the CUPD value at the beginning of the next period.

29. Analog-to-Digital Converter (ADC)

Rev: 1.0.2.1

29.1 Features

- **Integrated Multiplexer Offering Up to Eight Independent Analog Inputs**
- **Individual Enable and Disable of Each Channel**
- **Hardware or Software Trigger**
  - **External Trigger Pin**
  - **Timer Counter Outputs (Corresponding TIOA Trigger)**
- **PDC Support**
- **Possibility of ADC Timings Configuration**
- **Sleep Mode and Conversion Sequencer**
  - **Automatic Wakeup on Trigger and Back to Sleep Mode after Conversions of all Enabled Channels**

29.2 Description

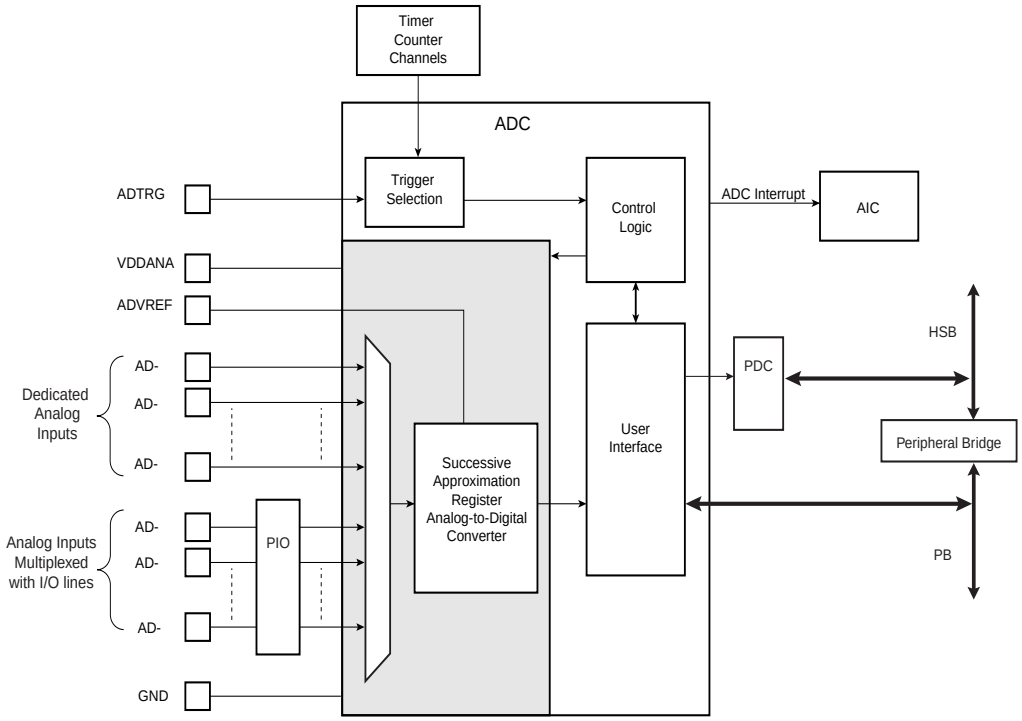
The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an 8-to-1 analog multiplexer, making possible the analog-to-digital conversions of 8 analog lines.

The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

29.3 Block Diagram



29.4 Signal Description

Table 29-1. ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
ADVREF	Reference voltage
AD0 - AD7	Analog input channels
ADTRG	External trigger

29.5 Product Dependencies

29.5.1 Power Management

The ADC is automatically clocked after the first conversion in Normal Mode. In Sleep Mode, the ADC clock is automatically stopped after each conversion. As the logic is small and the ADC cell can be put into Sleep Mode, the Power Management Controller has no effect on the ADC behavior.

29.5.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the ADC interrupt requires the AIC to be programmed first.

29.5.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

29.5.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

29.5.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be non-connected.

29.5.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

29.6 Functional Description

29.6.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Sample and Hold Clock cycles as defined in the field SHTIM of the "ADC Mode Register" on page 532 and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR).

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/128, if PRESCAL is set to 63 (0x3F). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the Product definition section.

29.6.2 Conversion Reference

The conversion is performed on a full range between 0V and
Analog inputs between these voltages convert to values based on a linear conversion.

29.6.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the bit LOWRES, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

Moreover, when a PDC channel is connected to the ADC, 10-bit resolution sets the transfer request sizes to 16-bit. Setting the bit LOWRES automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

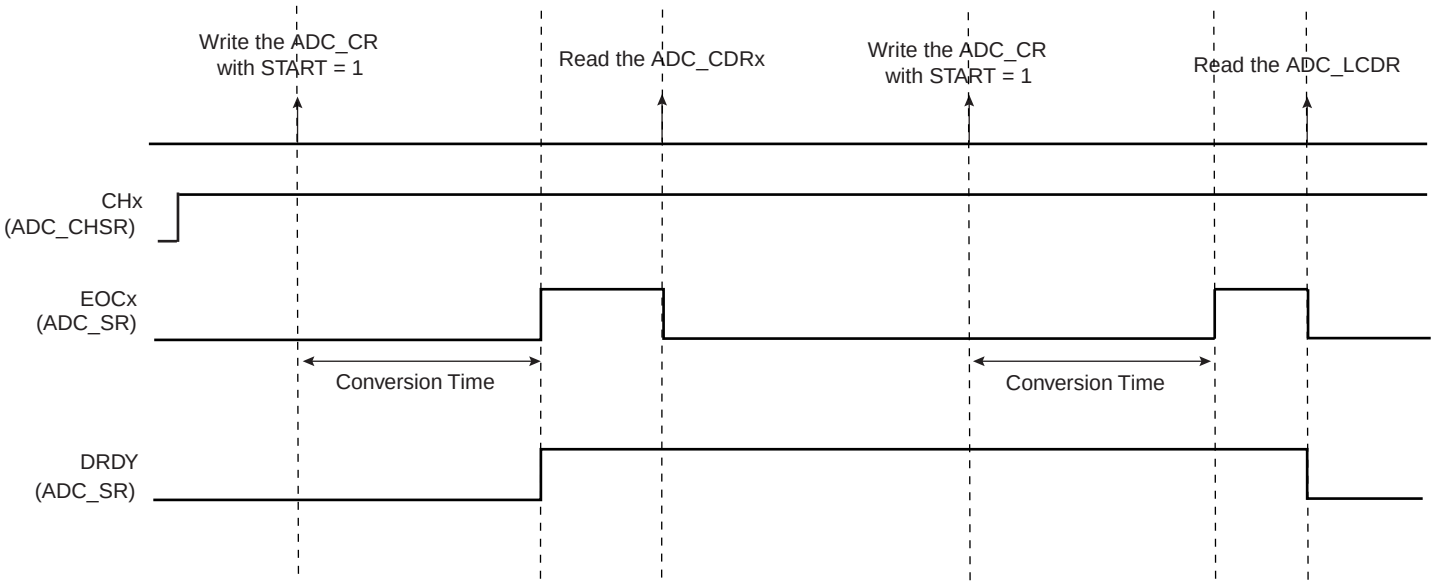
29.6.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDR) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR).

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected PDC channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

Figure 29-2. EOCx and DRDY Flag Behavior

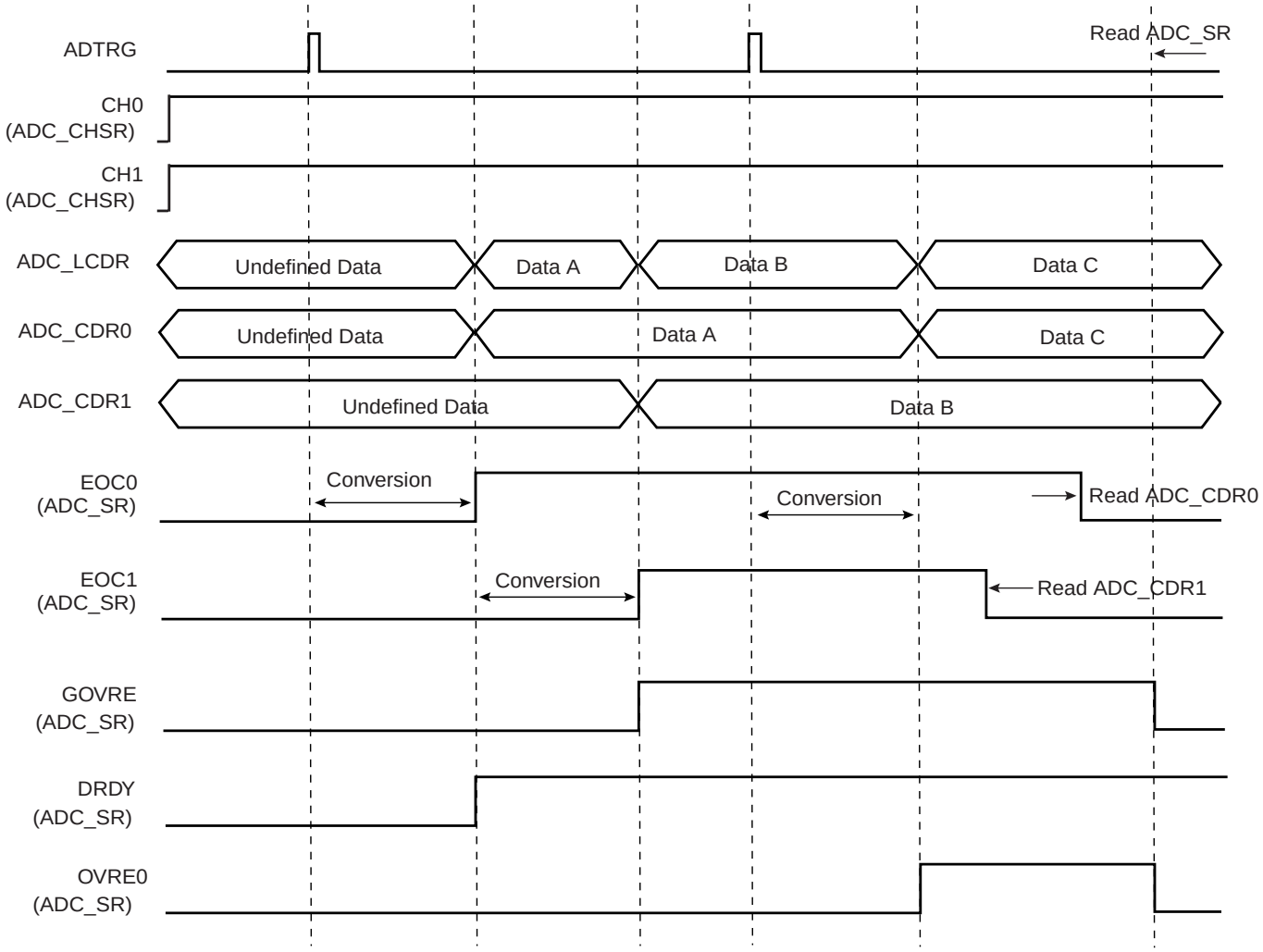


If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in ADC_SR.

The OVRE and GOVRE flags are automatically cleared when ADC_SR is read.

Figure 29-3. GOVRE and OVREx Flag Behavior



Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

29.6.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the bit START at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the field TRGSEL in the Mode Register (ADC_MR). The selected hardware trigger is enabled with the bit TRGEN in the Mode Register (ADC_MR).

If a hardware trigger is selected, the start of a conversion is detected at each rising edge of the selected signal. If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a PDC, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

Warning: Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can be initiated either by the hardware or the software trigger.

29.6.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the bit SLEEP in the Mode Register ADC_MR.

The SLEEP mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using a Timer/Counter output. The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the PDC.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

29.6.7 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register ADC_MR.

In the same way, a minimal Sample and Hold Time is necessary for the ADC to guarantee the best converted final value between two channels selection. This time has to be programmed through the bitfield SHTIM in the Mode Register ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the SHTIM field. See the section, ADC Characteristics in the product datasheet.

29.7 Analog-to-digital Converter (ADC) User Interface

Table 29-2. ADC Register Mapping

Offset	Register	Name	Access	Reset State
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read/Write	0x00000000
0x08	Reserved	–	–	–
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Status Register	ADC_SR	Read-only	0x000C0000
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x34	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x4C	Channel Data Register 7	ADC_CDR7	Read-only	0x00000000
0x50 - 0xFC	Reserved	–	–	–

29.7.1 ADC Control Register

Register Name: ADC_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

29.7.2 ADC Mode Register

Register Name: ADC_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	SHTIM			
23	22	21	20	19	18	17	16
–	–	–	STARTUP				
15	14	13	12	11	10	9	8
–	–	PRESCAL					
7	6	5	4	3	2	1	0
–	–	SLEEP	LOWRES	TRGSEL			TRGEN

- **TRGEN: Trigger Enable**

TRGEN	Selected TRGEN
0	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

TRGSEL			Selected TRGSEL
0	0	0	TIOA Ouput of the Timer Counter Channel 0
0	0	1	TIOA Ouput of the Timer Counter Channel 1
0	1	0	TIOA Ouput of the Timer Counter Channel 2
0	1	1	TIOA Ouput of the Timer Counter Channel 3
1	0	0	TIOA Ouput of the Timer Counter Channel 4
1	0	1	TIOA Ouput of the Timer Counter Channel 5
1	1	0	External trigger
1	1	1	Reserved

- **LOWRES: Resolution**

LOWRES	Selected Resolution
0	10-bit resolution
1	8-bit resolution

- **SLEEP: Sleep Mode**

SLEEP	Selected Mode
0	Normal Mode
1	Sleep Mode

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ((PRESCAL+1) * 2)
- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock
- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = (SHTIM+1) / ADCClock

29.7.3 ADC Channel Enable Register

Register Name: ADC_CHER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

29.7.4 ADC Channel Disable Register

Register Name: ADC_CHDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

29.7.5 ADC Channel Status Register

Register Name: ADC_CHSR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

29.7.6 ADC Status Register

Register Name: ADC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished.

1 = Corresponding analog channel is enabled and conversion is complete.

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_SR.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_SR.

1 = At least one General Overrun Error has occurred since the last read of ADC_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.

29.7.7 ADC Last Converted Data Register

Register Name: ADC_LCDR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	LDATA	
7	6	5	4	3	2	1	0
LDATA							

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

29.7.8 ADC Interrupt Enable Register

Register Name: ADC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Enable x**
- **OVREx: Overrun Error Interrupt Enable x**
- **DRDY: Data Ready Interrupt Enable**
- **GOVRE: General Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

29.7.9 ADC Interrupt Disable Register

Register Name: ADC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**
- **OVREx: Overrun Error Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

29.7.10 ADC Interrupt Mask Register

Register Name: ADC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**
- **OVREx: Overrun Error Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

29.7.11 ADC Channel Data Register

Register Name: ADC_CDRx

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	DATA	
7	6	5	4	3	2	1	0
DATA							

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

30. Debug and Test

Rev: 1.0.0.2

30.1 Features

- **IEEE1149.1 compliant JTAG and boundary scan**
- **Direct memory access and programming capabilities through JTAG interface**
- **Extensive On-Chip Debug features in compliance with IEEE-ISTO 5001-2003 (Nexus 2.0) Class 2+**
- **Auxiliary port for high-speed trace information**
- **Hardware support for 6 Program and 2 data breakpoints**
- **Unlimited number of software breakpoints supported**
- **Advanced Program, Data, Ownership, and Watchpoint trace supported**

30.2 JTAG Interface

Access to debug and test features is provided through a IEEE1149.1 compliant JTAG interface, using the pins shown in Table 30-1. The Test and Programming Technical Reference Manual details operation of the JTAG and the various commands, and only a brief overview follows.

The JTAG is a synchronous, serial protocol, which allows several devices on a circuit board to be accessed through a common JTAG port. The clock signal TCK and control signal TMS is common to all devices, while the data output TDO is chained to the data input TDI of the next device in the JTAG chain, effectively forming one long scan chain. Devices are addressed through their position in the JTAG chain.

Each JTAG device contains a TAP controller, which can be navigated through the TMS pin. The state of the TAP controller determines whether the serial data on TDI is a JTAG instruction or JTAG data. A number of serial data registers can be selected according to which JTAG instruction is loaded.

Table 30-1. JTAG pins

Pin	Direction	Description
TCK	Input	Test Clock. Data is driven on falling edge, sampled on rising edge.
TMS	Input	Test Mode Select
TDI	Input	Test Data In
TDO	Output	Test Data Out

The AVR32 JTAG has a 5-bit instruction register, which selects data registers of varying length. The implemented set of JTAG instructions provides the following capabilities:

- IEEE1149.1 compliant boundary-scan for testing interconnections between devices on a PCB.
- Internal and external memory programming
- Access to on-chip debug mechanisms
- Access to production test mechanisms

Production test specific features are described only in the Test and Programming Technical Reference Manual. Other features are additionally introduced below.

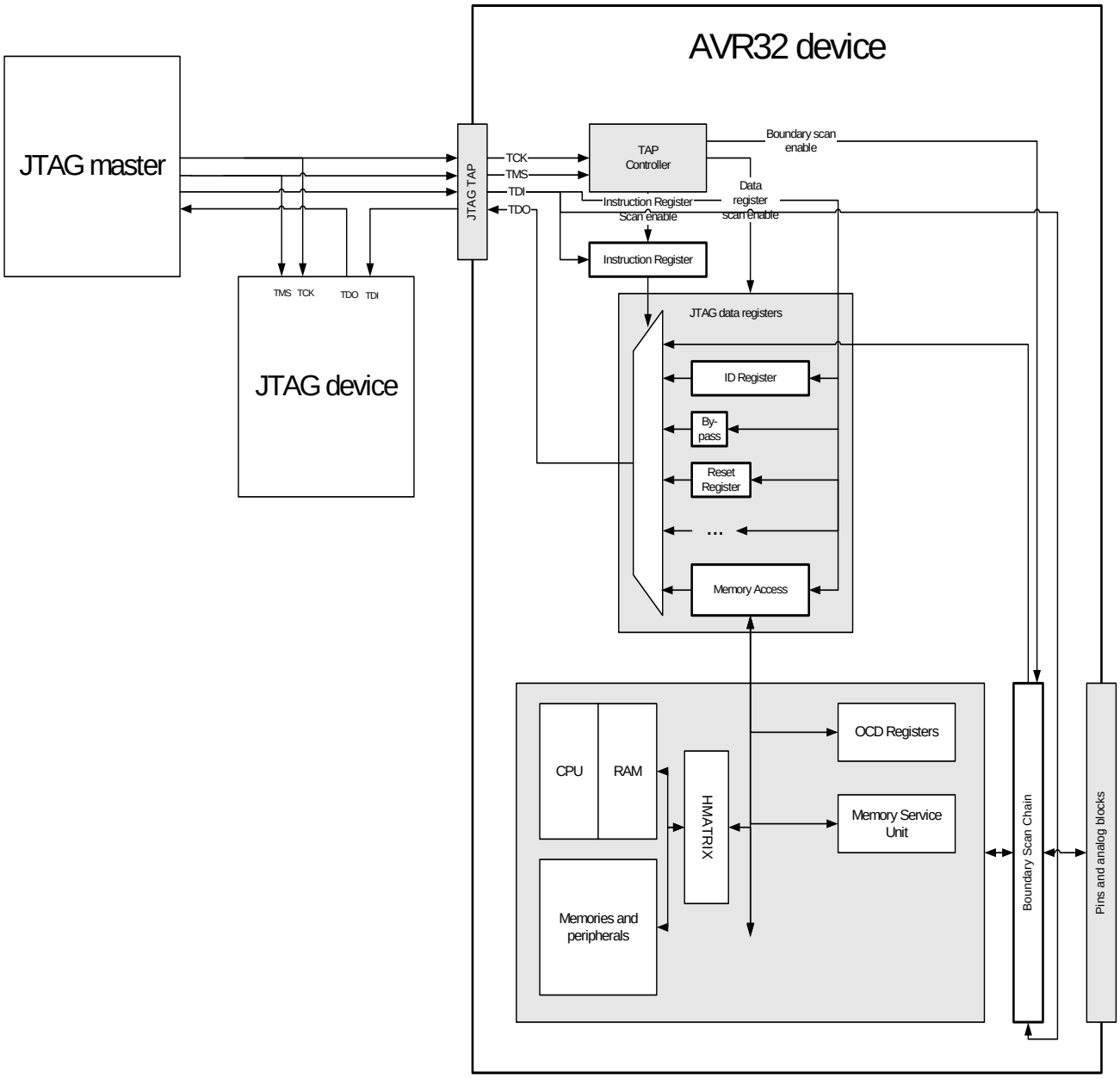


Figure 30-1. AVR32 JTAG connections

30.3 Public JTAG instructions

The following public (standard-defined) JTAG instructions are provided:

- IDCODE
- SAMPLE_PRELOAD
- EXTEST
- INTEST

- CLAMP
- BYPASS

The IDCODE is the default instruction loaded into the JTAG on power-up or after a JTAG reset. This selects the 32-bit JTAG IDCODE register, unique to each JTAG device. See “JTAG ID Codes” for actual values for this device.

BYPASS selects the 1-bit bypass register as data register. Devices in a JTAG chain should normally be placed in BYPASS when not being addressed by the JTAG master.

The other instructions are used by boundary-scan, which allows testing PCB interconnections by scanning known data to the device pins, or sampling data driven from other circuits on the PCB.

30.4 Memory programming

The MEMORY_WORD_ACCESS JTAG Instruction gives the user access to the HSB bus through the JTAG interface. The physical address and the direction bit (read/write) is scanned into the JTAG, followed by scanning out the read data or scanning in the write data, depending on the direction of the transfer. Any physical memory address can be read or written, allowing internal memories to be written in the same way as when accessed by the CPU. Similarly, external memories can be accessed through the External Bus Interface (EBI). A polling mechanism provides the JTAG master with status information, indicating when the memory operation is complete.

The MEMORY_BLOCK_ACCESS command allows for a burst mode memory access through JTAG. This command automatically repeats a previous MEMORY_WORD_ACCESS command, with automatic incrementation of the address. Thus only data needs to be scanned in or out, giving negligible protocol overhead on the JTAG interface.

The JTAG master can use the CRC check feature described in Section 30.5.2.1 on page 546 to automatically calculate the CRC-32 value of the programmed memory contents, to ensure that no transmission or programming errors have occurred.

When loading a new program to memory, the CPU should be kept halted by the HALT JTAG command. Otherwise, the CPU could execute a partially loaded program, leading to unpredictable result and possible program corruption.

30.5 Debugging

Debugging on the AT32UC3B is facilitated by a powerful On-Chip Debug (OCD) system. The user accesses this through an external debug tool which connects to the JTAG port and the Auxiliary (AUX) port. The AUX port is primarily used for trace functions, and a JTAG-based debugger is sufficient for basic debugging.

The debug system is based on the Nexus 2.0 standard, class 2+, which includes:

- Basic run-time control
- Program breakpoints
- Data breakpoints
- Program trace
- Ownership trace
- Data trace

In addition to the mandatory Nexus debug features, the AT32UC3B implements several useful OCD features, such as:

- Debug Communication Channel between CPU and JTAG
- Run-time PC monitoring
- CRC checking
- NanoTrace
- Software Quality Assurance (SQA) support

The OCD features are controlled by OCD registers, which can be accessed by JTAG when the NEXUS_ACCESS JTAG instruction is loaded. The CPU can also access OCD registers directly using mtdr/mfdr instructions in any privileged mode. The OCD registers are implemented based on the recommendations in the Nexus 2.0 standard, and are detailed in the AVR32UC Technical Reference Manual.

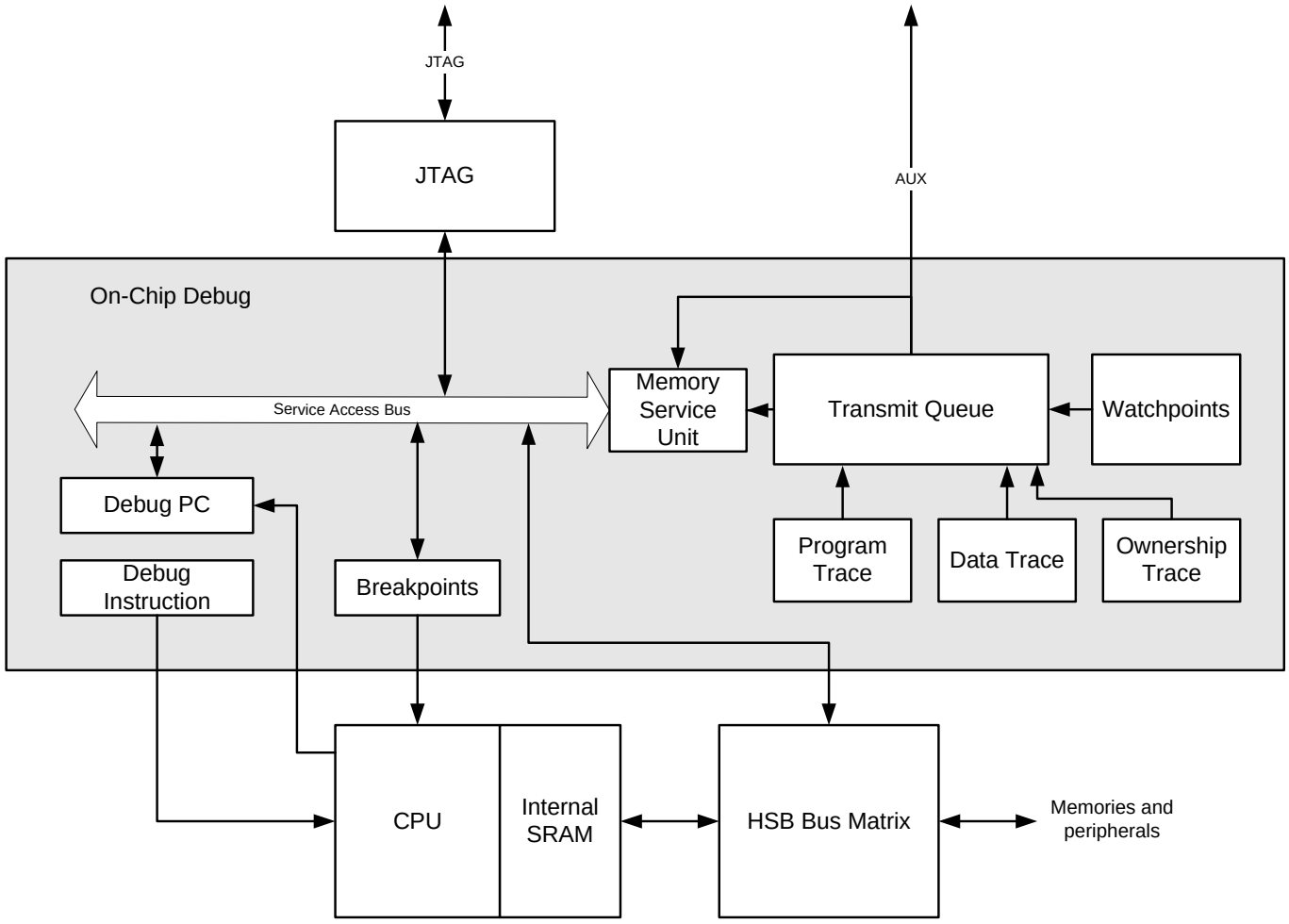


Figure 30-2. On-Chip Debug block diagram

30.5.1 JTAG-based debug features

A debugger can control all OCD features by writing OCD registers over the JTAG interface. Many of these do not depend on output on the AUX port, allowing a JTAG-based debugger to be used.

A JTAG-based debugger should connect to the device through a standard 10-pin IDC connector as described in the AVR32UC Technical Reference Manual.

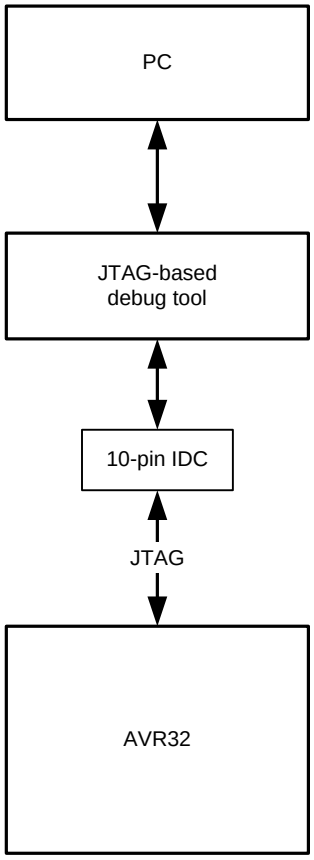


Figure 30-3. JTAG-based debugger

30.5.1.1 Debug Communication Channel

The Debug Communication Channel (DCC) consists of a pair OCD registers with associated handshake logic, accessible to both CPU and JTAG. The registers can be used to exchange data between the CPU and the JTAG master, both runtime as well as in debug mode.

30.5.1.2 Breakpoints

One of the most fundamental debug features is the ability to halt the CPU, to examine registers and the state of the system. This is accomplished by breakpoints, of which many types are available:

- Unconditional breakpoints are set by writing OCD registers by JTAG, halting the CPU immediately.
- Program breakpoints halt the CPU when a specific address in the program is executed.
- Data breakpoints halt the CPU when a specific memory address is read or written, allowing variables to be watched.
- Software breakpoints halt the CPU when the breakpoint instruction is executed.

When a breakpoint triggers, the CPU enters debug mode, and the D bit in the status register is set. This is a privileged mode with dedicated return address and return status registers. All privileged instructions are permitted. Debug mode can be entered as either OCD Mode, running instructions from JTAG, or Monitor Mode, running instructions from program memory.

30.5.1.3 OCD Mode

When a breakpoint triggers, the CPU enters OCD mode, and instructions are fetched from the Debug Instruction OCD register. Each time this register is written by JTAG, the instruction is executed, allowing the JTAG to execute CPU instructions directly. The JTAG master can e.g. read out the register file by issuing mtdr instructions to the CPU, writing each register to the Debug Communication Channel OCD registers.

30.5.1.4 Monitor Mode

Since the OCD registers are directly accessible by the CPU, it is possible to build a software-based debugger that runs on the CPU itself. Setting the Monitor Mode bit in the Development Control register causes the CPU to enter Monitor Mode instead of OCD mode when a breakpoint triggers. Monitor Mode is similar to OCD mode, except that instructions are fetched from the debug exception vector in regular program memory, instead of issued by JTAG.

30.5.1.5 Program Counter monitoring

Normally, the CPU would need to be halted for a JTAG-based debugger to examine the current PC value. However, the AT32UC3B also proves a Debug Program Counter OCD register, where the debugger can continuously read the current PC without affecting the CPU. This allows the debugger to generate a simple statistic of the time spent in various areas of the code, easing code optimization.

30.5.2 Memory Service Unit

The Memory Service Unit is a block dedicated to test and debug functionality. It is controlled through a dedicated set of registers addressed through the MEMORY_SERVICE JTAG command.

30.5.2.1 Cyclic Redundancy Check (CRC)

The MSU can be used to automatically calculate the CRC of a block of data in memory. The OCD will then read out each word in the specified memory block and report the CRC32-value in an OCD register.

30.5.2.2 NanoTrace

The MSU additionally supports NanoTrace. This is an AVR32-specific feature, in which trace data is output to memory instead of the AUX port. This allows the trace data to be extracted by JTAG MEMORY_ACCESS, enabling trace features for JTAG-based debuggers. The user must write MSU registers to configure the address and size of the memory block to be used for NanoTrace. The NanoTrace buffer can be anywhere in the physical address range, including internal and external RAM, through an EBI, if present. This area may not be used by the application running on the CPU.

30.5.3 AUX-based debug features

Utilizing the Auxiliary (AUX) port gives access to a wide range of advanced debug features. Of prime importance are the trace features, which allow an external debugger to receive continuous

information on the program execution in the CPU. Additionally, Event In and Event Out pins allow external events to be correlated with the program flow.

The AUX port contains a number of pins, as shown in Table 30-2. These are multiplexed with PIO lines, and must explicitly be enabled by writing OCD registers before the debug session starts. The AUX port is mapped to two different locations, selectable by OCD Registers, minimizing the chance that the AUX port will need to be shared with an application.

Debug tools utilizing the AUX port should connect to the device through a Nexus-compliant Mictor-38 connector, as described in the AVR32UC Technical Reference manual. This connector includes the JTAG signals and the RESET_N pin, giving full access to the programming and debug features in the device.

Table 30-2. Auxiliary port signals

Signal	Direction	Description
MCKO	Output	Trace data output clock
MDO[5:0]	Output	Trace data output
MSEO[1:0]	Output	Trace frame control
EVTI_N	Input	Event In
EVTO_N	Output	Event Out

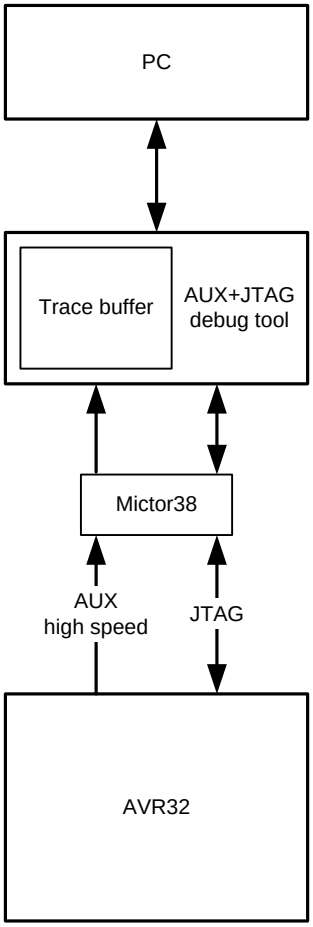


Figure 30-4. AUX+JTAG based debugger

30.5.3.1 Trace operation

Trace features are enabled by writing OCD registers by JTAG. The OCD extracts the trace information from the CPU, compresses this information and formats it into variable-length messages according to the Nexus standard. The messages are buffered in a 16-frame transmit queue, and are output on the AUX port one frame at a time.

The trace features can be configured to be very selective, to reduce the bandwidth on the AUX port. In case the transmit queue overflows, error messages are produced to indicate loss of data. The transmit queue module can optionally be configured to halt the CPU when an overflow occurs, to prevent the loss of messages, at the expense of longer run-time for the program.

30.5.3.2 Program Trace

Program trace allows the debugger to continuously monitor the program execution in the CPU. Program trace messages are generated for every branch in the program, and contains compressed information, which allows the debugger to correlate the message with the source code to identify the branch instruction and target address.

30.5.3.3 Data Trace

Data trace outputs a message every time a specific location is read or written. The message contains information about the type (read/write) and size of the access, as well as the address and data of the accessed location. The AT32UC3B contains two data trace channels, each of which are controlled by a pair of OCD registers which determine the range of addresses (or single address) which should produce data trace messages.

30.5.3.4 Ownership Trace

Program and data trace operate on virtual addresses. In cases where an operating system runs several processes in overlapping virtual memory segments, the Ownership Trace feature can be used to identify the process switch. When the O/S activates a process, it will write the process ID number to an OCD register, which produces an Ownership trace message, allowing the debugger to switch context for the subsequent program and data trace messages. As the use of this feature depends on the software running on the CPU, it can also be used to extract other types of information from the system.

30.5.3.5 Watchpoint messages

The breakpoint modules normally used to generate program and data breakpoints can also be used to generate Watchpoint messages, allowing a debugger to monitor program and data events without halting the CPU. Watchpoints can be enabled independently of breakpoints, so a breakpoint module can optionally halt the CPU when the trigger condition occurs. Data trace modules can also be configured to produce watchpoint messages instead of regular data trace messages.

30.5.3.6 Event In and Event Out pins

The AUX port also contains an Event In pin (EVTI_N) and an Event Out pin (EVTO_N). EVTI_N can be used to trigger a breakpoint when an external event occurs. It can also be used to trigger specific program and data trace synchronization messages, allowing an external event to be correlated to the program flow.

When the CPU enters debug mode, a Debug Status message is transmitted on the trace port. All trace messages can be timestamped when they are received by the debug tool. However, due to the latency of the transmit queue buffering, the timestamp will not be 100% accurate. To improve this, EVTO_N can toggle every time a message is inserted into the transmit queue, allowing trace messages to be timestamped precisely. EVTO_N can also toggle when a breakpoint module triggers, or when the CPU enters debug mode, for any reason. This can be used to measure precisely when the respective internal event occurs.

30.5.3.7 Software Quality Analysis (SQA)

Software Quality Analysis (SQA) deals with two important issues regarding embedded software development. *Code coverage* involves identifying untested parts of the embedded code, to improve test procedures and thus the quality of the released software. *Performance analysis* allows the developer to precisely quantify the time spent in various parts of the code, allowing bottlenecks to be identified and optimized.

Program trace must be used to accomplish these tasks without instrumenting (altering) the code to be examined. However, traditional program trace cannot reconstruct the current PC value without correlating the trace information with the source code, which cannot be done on-the-fly.

This limits program trace to a relatively short time segment, determined by the size of the trace buffer in the debug tool.

The OCD system in AT32UC3B extends program trace with SQA capabilities, allowing the debug tool to reconstruct the PC value on-the-fly. Code coverage and performance analysis can thus be reported for an unlimited execution sequence.

31. Boot Sequence

This chapter summarizes the boot sequence of the AT32UC3B. The behaviour after power-up is controlled by the Power Manager. For specific details, refer to Section 13. ”Power Manager (PM)” on page 44.

31.1 Starting of clocks

After power-up, the device will be held in a reset state by the Power-On Reset circuitry, until the power has stabilized throughout the device. Once the power has stabilized, the device will use the internal RC Oscillator as clock source.

On system start-up, the PLLs are disabled. All clocks to all modules are running. No clocks have a divided frequency, all parts of the system recieves a clock with the same frequency as the internal RC Oscillator.

31.2 Fetching of initial instructions

After reset has been released, the AVR32 UC CPU starts fetching instructions from the reset address, which is 0x8000_0000. This address points to the first address in the internal Flash.

The code read from the internal Flash is free to configure the system to use for example the PLLs, to divide the frequency of the clock routed to some of the peripherals, and to gate the clocks to unused peripherals.

32. Electrical Characteristics

32.1 Absolute Maximum Ratings*

Operating Temperature ..... -40°C to +85°C

Storage Temperature ....-60°C to +150°C

Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground .....-TBDV to V_{CC}+TBDV

Voltage on $\overline{\text{RESET}}$ with respect to Ground....-TBDV to +TBDV

Maximum Operating Voltage (VDDCORE, VDDSYS) 1.95V

Maximum Operating Voltage (VDDIO)..... 3.6V

DC Current per I/O Pin ..... TBD mA

DC Current V_{CC} and GND Pins..... TBD mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

32.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$V_{VDDCORE}$	DC Supply Core			1.65		1.95
V_{VDDBU}	DC Supply Backup			1.65		1.95
V_{VDDOSC}	DC Supply Oscillator			1.65		1.95
V_{VDDPLL}	DC Supply PLL			1.65		1.95
V_{VDDUSB}	DC Supply USB			1.65		1.95
V_{VDDIO}	DC Supply Peripheral I/Os			3.0		3.6
V_{REF}	Analog reference voltage			2.6		3.6
V_{IL}	Input Low-level Voltage			-0.3		+0.8
V_{IH}	Input High-level Voltage			2.0		V_{VDDIO}+0.3
V_{OL}	Output Low-level Voltage					0.4
V_{OH}	Output High-level Voltage	V_{VDDIO}= V_{VDDIOM} or V_{VDDIOP}		V_{VDDIO}-0.4		
I_{LEAK}	Input Leakage Current	Pullup resistors disabled				TBD
C_{IN}	Input Capacitance					TBD
R_{PULLUP}	Pull-up Resistance				TBD	
I_O	Output Current					TBD
I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.8V, CPU = 0 Hz	T_A =25°C		TBD	
		All inputs driven; RESET_N=1	T_A =85°C			TBD

32.3 Power Consumption

The values in Table 32-1 and Table 32-2 on page 556 are measured values of power consumption with operating conditions as follows:

- V_{DDIO} = 3.3V
- V_{DDCORE} = V_{DDSYS} = 1.8V
- T_A = 25°C
- I/Os are inactive

These figures represent the power consumption measured on the power supplies.

Table 32-1. Power Consumption for Different Modes[1]

Mode	Conditions	Consumption	Unit
Active	Core/HSB clock is 60 MHz. PBA clock is 30 MHz. PBB clock is 60 MHz. All peripheral clocks activated. Measured while the processor is executing a recursive Fibonacci algorithm.	TBD	mA

Table 32-2. Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
GPIO	TBD	mA
USART	TBD	
USB	TBD	
ADC	TBD	
TWI	TBD	
PWM	TBD	
SPI	TBD	
SSC	TBD	
Timer Counter Channels	TBD	

32.4 Clock Characteristics

These parameters are given in the following conditions:

- V_{DDCORE} = 1.8V
- Ambient Temperature = 25°C

32.4.1 CPU/HSB Clock Characteristics

Table 32-3. Core Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPCPU})$	CPU Clock Frequency			60	MHz
t_{CPCPU}	CPU Clock Period		15		ns

32.4.2 PBA Clock Characteristics

Table 32-4. PBA Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPBA})$	PBA Clock Frequency			30	MHz
t_{CPPBA}	PBA Clock Period		30		ns

32.4.3 PBB Clock Characteristics

Table 32-5. PBB Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPBB})$	PBB Clock Frequency			60	MHz
t_{CPPBB}	PBB Clock Period		15		ns

32.4.4 XIN Clock Characteristics

Table 32-6. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency		3	24	MHz
t_{CPXIN}	XIN Clock Period		20.0		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN Clock Low Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN Input Capacitance	(1)		TBD	pF
R_{IN}	XIN Pulldown Resistor	(1)		TBD	kΩ

Note: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR register.)

32.5 Crystal Oscillator Characteristis

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

32.5.1 32 KHz Oscillator Characteristics

Table 32-7. 32 KHz Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			32 768		Hz
	Duty Cycle		TBD		TBD	%
t_{ST}	Startup Time	R_S = TBD kΩ, C_L = TBD pF(1)			TBD	ms

Note: 1. R_S is the equivalent series resistance, C_L is the equivalent load capacitance.

32.5.2 Main Oscillators Characteristics

Table 32-8. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		3		16	MHz
C_{L1}, C_{L2}	Internal Load Capacitance (C_{L1} = C_{L2})			TBD		pF
C_L	Equivalent Load Capacitance			TBD		pF
	Duty Cycle		TBD	TBD	TBD	%
t_{ST}	Startup Time				TBD	ms
I_{OSC}	Current Consumption	Active mode @TBD MHz			TBD	µA
		Standby mode @TBD V			TBD	µA

Notes: 1. C_S is the shunt capacitance

32.5.3 PLL Characteristics

Table 32-9. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{OUT}	Output Frequency		48		150	MHz
F_{IN}	Input Frequency		TBD		TBD	MHz
I_{PLL}	Current Consumption	active mode			TBD	mA
		standby mode			TBD	µA

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

32.6 USB Transceiver Characteristics

32.6.1 Electrical Characteristics

Table 32-10. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level				TBD	V
V_{IH}	High Level		TBD			V
V_{DI}	Differential Input Sensivity	|(D+) - (D-)|	TBD			V
V_{CM}	Differential Input Common Mode Range		TBD		TBD	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			TBD	pF
I	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	TBD		TBD	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		TBD		Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	TBD		TBD	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	TBD		TBD	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 32-1	TBD		TBD	V

32.6.2 Switching Characteristics

Table 32-11. In Low Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 400 pF	TBD		TBD	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 400 pF	TBD		TBD	ns
t_{FRFM}	Rise/Fall time Matching	C_{LOAD} = 400 pF	TBD		TBD	%

Table 32-12. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	TBD		TBD	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	TBD		TBD	ns
t_{FRFM}	Rise/Fall time Matching		TBD		TBD	%

Figure 32-1. USB Data Signal Rise and Fall Times

32.7 AC Characteristics - TBD

33. Mechanical Characteristics

33.1 Thermal Considerations

33.1.1 Thermal Data

Table 33-1 summarizes the thermal resistance data depending on the package.

Table 33-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	TQFP100	TBD	°C/W
θ_{JC}	Junction-to-case thermal resistance		TQFP100	TBD	
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	LQFP144	TBD	°C/W
θ_{JC}	Junction-to-case thermal resistance		LQFP144	TBD	

33.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

1. $T_J = T_A + (P_D \times \theta_{JA})$
2. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

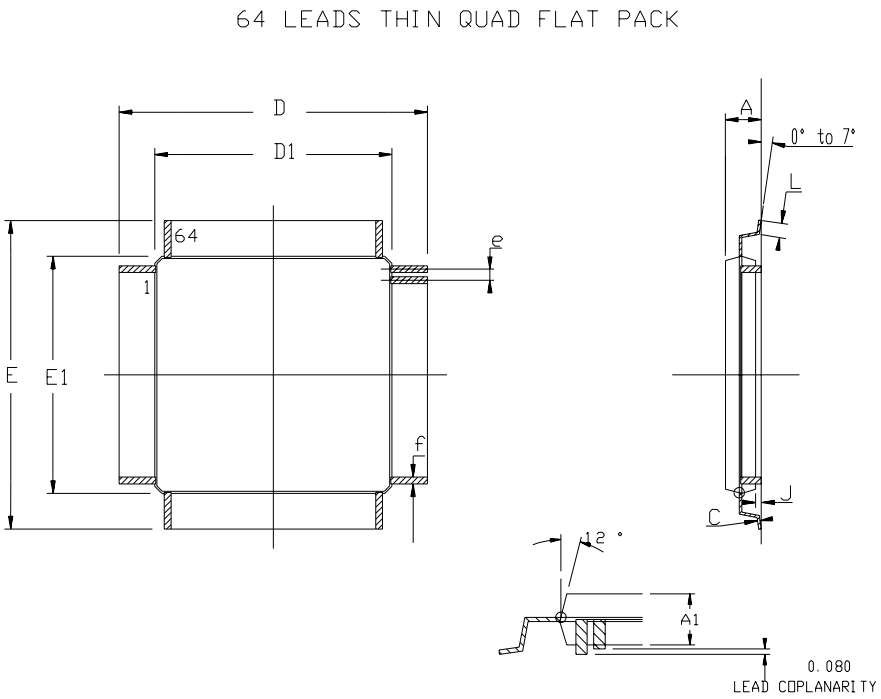
where:

- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 33-1 on page 562.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 33-1 on page 562.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section "Power Consumption" on page 555.
- T_A = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

33.2 Package Drawings

Figure 33-1. TQFP-64 package drawing



	MM		INCH	
	Min	Max	Min	Max
A	----	1.20	----	.047
A1	0.95	1.05	.037	.041
C	0.09	0.20	.004	.008
D	12.00 BSC		.472 BSC	
D1	10.00 BSC		.394 BSC	
E	12.00 BSC		.472 BSC	
E1	10.00 BSC		.394 BSC	
J	0.05	0.15	.002	.006
L	0.45	0.75	.018	.030
e	0.50 BSC		.0197 BSC	
f	0.17	0.27	.007	.011

Table 33-2. Device and Package Maximum Weight

TBD	mg

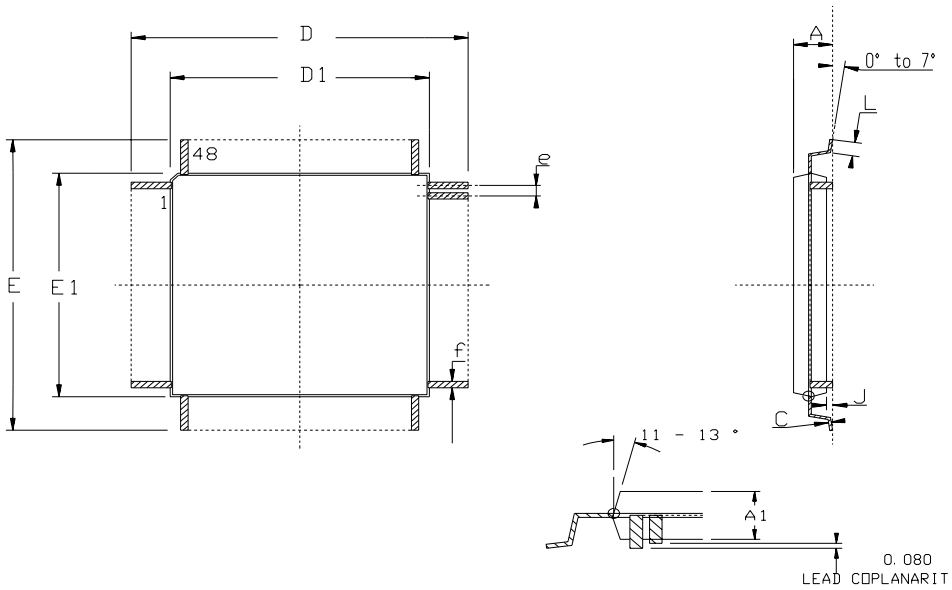
Table 33-3. Package Characteristics

Moisture Sensitivity Level	TBD

Table 33-4. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 33-2. TQFP-48 package drawing



	MM		INCH	
	Min	Max	Min	Max
A	----	1.20	----	.047
A1	0.95	1.05	.037	.041
C	0.09	0.20	.004	.008
D	9.00 BSC		.354 BSC	
D1	7.00 BSC		.276 BSC	
E	9.00 BSC		.354 BSC	
E1	7.00 BSC		.276 BSC	
J	0.05	0.15	.002	.006
L	0.45	0.75	.018	.030
e	0.50 BSC		.0197 BSC	
f	0.17	0.27	.0067	.0106

Table 33-5. Device and Package Maximum Weight

TBD	mg

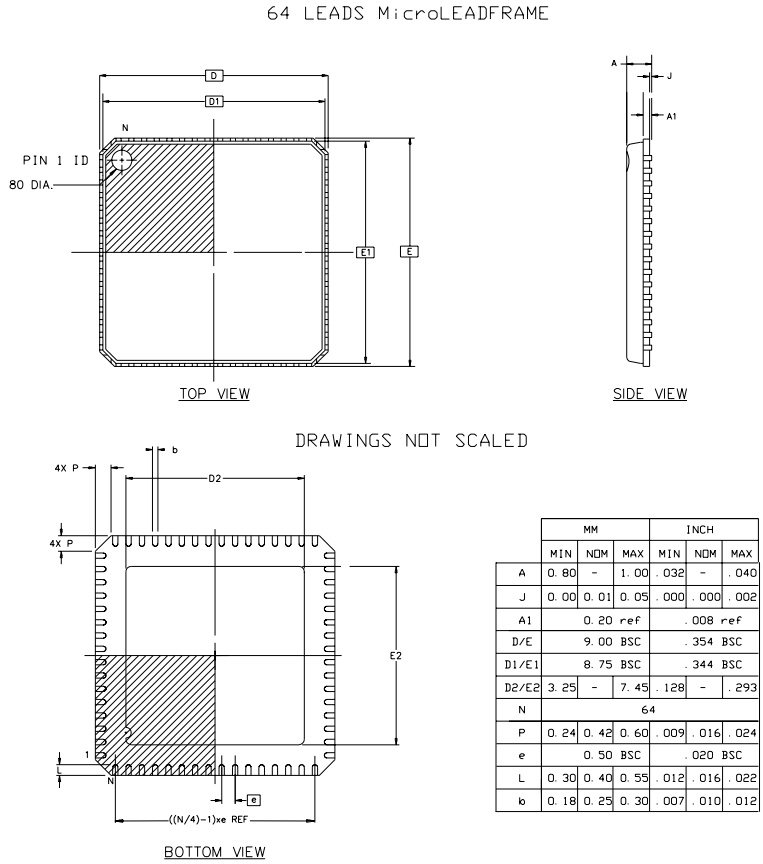
Table 33-6. Package Characteristics

Moisture Sensitivity Level	TBD

Table 33-7. Package Reference

JEDEC Drawing Reference	MS-026
JESD97 Classification	E3

Figure 33-3. MLF-64 package drawing



	MM			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.80	-	1.00	.032	-	.040
J	0.00	0.01	0.05	.000	.000	.002
A1		0.20 ref			.008 ref	
D/E		9.00 BSC			.354 BSC	
D1/E1		8.75 BSC			.344 BSC	
D2/E2	3.25	-	7.45	.128	-	.293
N		64				
P	0.24	0.42	0.60	.009	.016	.024
e		0.50 BSC			.020 BSC	
L	0.30	0.40	0.55	.012	.016	.022
b	0.18	0.25	0.30	.007	.010	.012

Table 33-8. Device and Package Maximum Weight

TBD	mg

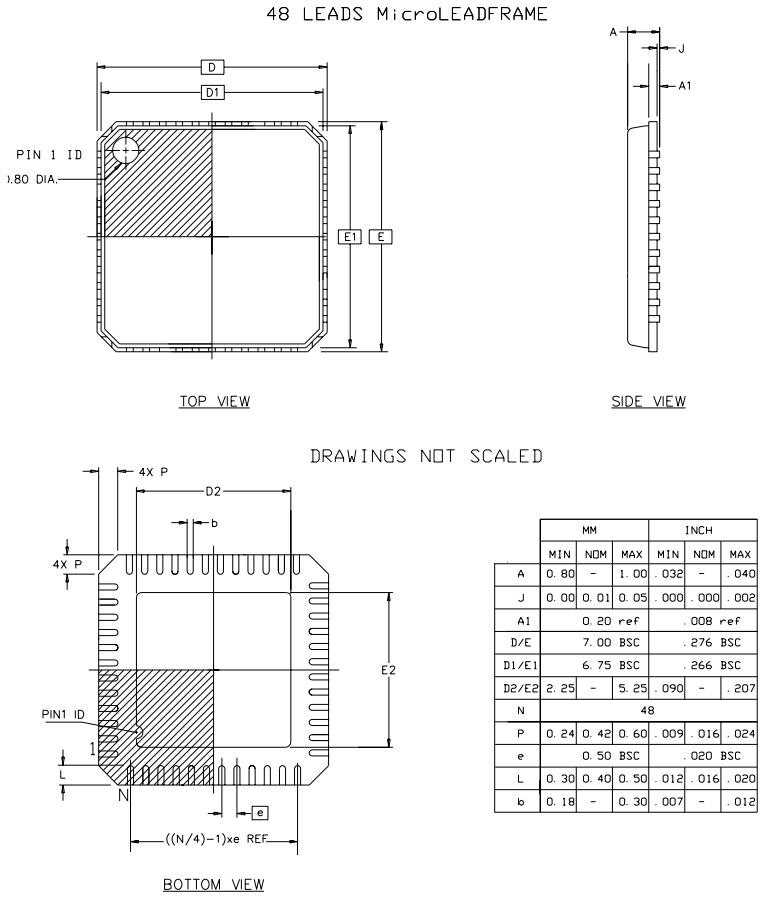
Table 33-9. Package Characteristics

Moisture Sensitivity Level	TBD

Table 33-10. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	E3

Figure 33-4. MLF-48 package drawing



	MM			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.80	-	1.00	.032	-	.040
J	0.00	0.01	0.05	.000	.000	.002
A1		0.20 ref			.008 ref	
D/E		7.00 BSC			.276 BSC	
D1/E1		6.75 BSC			.266 BSC	
D2/E2	2.25	-	5.25	.090	-	.207
N		48				
P	0.24	0.42	0.60	.009	.016	.024
e		0.50 BSC			.020 BSC	
L	0.30	0.40	0.50	.012	.016	.020
b	0.18	-	0.30	.007	-	.012

Table 33-11. Device and Package Maximum Weight

TBD	mg

Table 33-12. Package Characteristics

Moisture Sensitivity Level	TBD

Table 33-13. Package Reference

JEDEC Drawing Reference	M0-220
JESD97 Classification	E3

33.3 Soldering Profile

Table 33-14 gives the recommended soldering profile from J-STD-20.

Table 33-14. Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	TBD
Preheat Temperature 175°C ±25°C	TBD
Temperature Maintained Above 217°C	TBD
Time within 5°C of Actual Peak Temperature	TBD
Peak Temperature Range	TBD
Ramp-down Rate	TBD
Time 25°C to Peak Temperature	TBD

Note: It is recommended to apply a soldering temperature higher than 250°C.

A maximum of three reflow passes is allowed per component.

34. Ordering Information

Device	Ordering Code	Package	Conditioning	Temperature Operating Range
AT32UC3B0256	AT32UC3B0256-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0256-Z2UT	MLF 64	Tray	Industrial (-40°C to 85°C)
AT32UC3B0128	AT32UC3B0128-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B0128-Z2UT	MLF 64	Tray	Industrial (-40°C to 85°C)
AT32UC3B064	AT32UC3B064-A2UT	TQFP 64	Tray	Industrial (-40°C to 85°C)
	AT32UC3B064-Z2UT	MLF 64	Tray	Industrial (-40°C to 85°C)
AT32UC3B1256	AT32UC3B1256-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1256-Z1UT	MLF 48	Tray	Industrial (-40°C to 85°C)
AT32UC3B1128	AT32UC3B1128-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B1128-Z1UT	MLF 48	Tray	Industrial (-40°C to 85°C)
AT32UC3B164	AT32UC3B164-AUT	TQFP 48	Tray	Industrial (-40°C to 85°C)
	AT32UC3B164-Z1UT	MLF 48	Tray	Industrial (-40°C to 85°C)

35. Errata

35.1 Rev. B

1. **SPI FDIV option does not work**
   Selecting clock signal using FDIV = 1 does not work as specified.

   Fix/Workaround
   Do not set FDIV = 1.

2. **PWM counter restarts at 0x0001**
   The PWM counter restarts at 0x0001 and not 0x0000 as specified. Because of this the first PWM period has one more clock cycle.

   Fix/Workaround
   - The first period is 0x0000, 0x0001, ..., period

   - Consecutive periods are 0x0001, 0x0002, ..., period

3. **PWM channel interrupt enabling triggers an interrupt**
   When enabling a PWM channel that is configured with center aligned period (CALG=1), an interrupt is signalled.

   Fix/Workaround
   When using center aligned mode, enable the channel and read the status before channel interrupt is enabled.

4. **PWM update period to a 0 value does not work**
   It is impossible to update a period equal to 0 by the using the PWM update register (PWM_CUPD).

Fix/Workaround

Do not update the PWM_CUPD register with a value equal to 0.

5. **PWM channel status may be wrong if disabled before a period has elapsed**

Before a PWM period has elapsed, the read channel status may be wrong. The CHIDx-bit for a PWM channel in the PWM Enable Register will read '1' for one full PWM period even if the channel was disabled before the period elapsed. It will then read '0' as expected.

Fix/Workaround

Reading the PWM channel status of a disabled channel is only correct after a PWM period has elapsed.

6. **SSC does not trigger RF when data is low**

The SSC cannot transmit or receive data when CKS = CKDIV and CKO = none, in TCMR or RCMR respectively.

Fix/Workaround

Set CKO to a value that is not "none" and bypass the output of the TK/RK pin with the PIO.

7. **SSC Data is not sent unless clock is set as output**

The SSC cannot transmit or receive data when CKS = CKDIV and CKO = none, in TCMR or RCMR respectively.

Fix/Workaround

Set CKO to a value that is not "none" and bypass the output of the TK/RK pin with the PIO.

8. **USB No end of host reset signaled upon disconnection**

In host mode, in case of an unexpected device disconnection whereas a usb reset is being sent by the usb controller, the UHCON.RESET bit may not been cleared by the hardware at the end of the reset.

Fix/Workaround

A software workaround consists in testing (by polling or interrupt) the disconnection (UHINT.DDISCI == 1) while waiting for the end of reset (UHCON.RESET == 0) to avoid being stuck.

9. **Incorrect Processor ID**

The processor ID reads 0x01 and not 0x02 as it should.

Fix/Workaround

None.

10. **Bus error should be masked in Debug mode**

If a bus error occurs during debug mode, the processor will not respond to debug commands through the DINST register.

Fix/Workaround

A reset of the device will make the CPU respond to debug commands again.

11. **Read Modify Write (RMW) instructions on data outside the internal RAM does not work.**

Read Modify Write (RMW) instructions on data outside the internal RAM does not work.

Fix/Workaround

Do not perform RMW instructions on data outside the internal RAM.

12. USART Manchester Encoder Not Working

Manchester encoding/decoding is not working.

Fix/Workaround

Do not use manchester encoding.

13. USART RXBREAK problem when no timeguard

In asynchronous mode the RXBREAK flag is not correctly handled when the timeguard is 0 and the break character is located just after the stop bit.

Fix/Workaround

If the NBSTOP is 1, timeguard should be different from 0.

14. USART Handshaking: 2 characters sent / CTS rises when TX

If CTS switches from 0 to 1 during the TX of a character, if the Holding register is not empty, the TXHOLDING is also transmitted.

Fix/Workaround

None.

15. USART PDC and TIMEGUARD not supported in MANCHESTER

Manchester encoding/decoding is not working.

Fix/Workaround

Do not use manchester encoding.

16. ADC possible miss on DRDY when disabling a channel

The ADC does not work properly when more than one channel is enabled.

Fix/Workaround

Do not use the ADC with more than one channel enabled at a time.

17. ADC OVRE flag sometimes not reset on Status Register read

The OVRE flag does not clear properly if read simultaneously to an end of conversion.

Fix/Workaround

None.

18. Need two NOPs instruction after instructions masking interrupts

The instructions following in the pipeline the instruction masking the interrupt through SR may behave abnormally.

Fix/Workaround

Place two NOPs instructions after each SSRF or MTSR instruction setting IxM or GM in SR

36. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

36.1 Rev. B 07/07

1. Updated registered trademarks.
2. Updated address page.

36.2 Rev. A 05/07

1. Initial revision.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature