Technical Datasheet

TPR0521D

DATASHEET PRELIMINARY

SEAL SQ
semiconductors + quantum

Table of Content

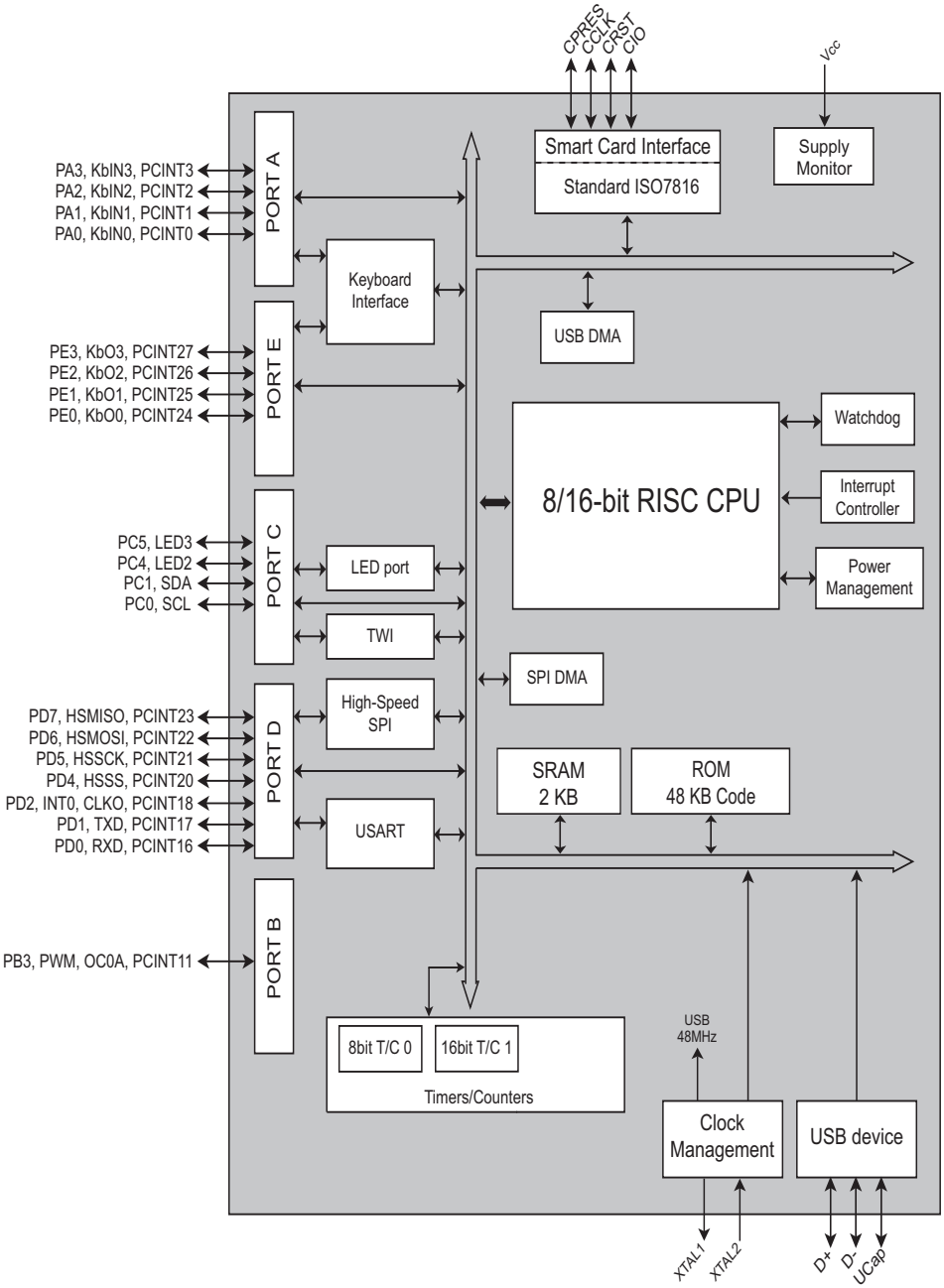
General Features

- High Performance, Low Power 8/16-bit RISC CPU Microcontroller
- Advanced RISC Architecture
  - 132 Powerful Instructions - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Up to 16MIPS Throughput at 16Mhz
  - On-chip 2-cycle Multiplier
- Memories
  - 48K Bytes of ROM program memory
  - 2K Bytes Internal SRAM
- ISO7816 UART Interface Fully compliant with EMV, GIE-CB and WHQL Standards
  - Programmable ISO clock from 1 Mhz to 4.8, 6, 8 or 12Mhz
  - Card insertion/removal detection with automatic deactivation sequence
  - Programmable Baud Rate Generator from 372 to 8 clock cycles
  - Synchronous/Asynchronous Protocols T=0 and T=1 with Direct or Inverse Convention
  - Automatic character repetition on parity errors
  - 32 Bit Waiting Time Counter
  - 16 Bit Guard Time Counter/Block Guard Time Counter
  - Internal Step Up/Down Converter with Programmable Voltage Output if DC/DC embedded:
  - Class A: 5V +/-8% at 60mA, Vcc>3.0V (55mA if Vcc >2.7)
  - Class B: 3V +/-8% at 60mA, Vcc>3.0V (55mA if Vcc >2.7)
  - Class C: 1.8V +/-8% at 35mA
  - Supports up to 55mA USB Smart Cards
  - Supports limited cable length to Smart Card Connector (~50cm)
  - 4 kV ESD (JEDEC JESD22-A114E) protection on whole Smart Card Interface
- USB 2.0 Full-speed Device Module
  - Complies fully with:
  - Universal Serial Bus Specification Rev 2.0
  - Supports data transfer rates up to 12 Mbit/s
  - Endpoint 0 for Control Transfers: up to 64-bytes
  - 8 Programmable Endpoints with IN or OUT Directions and with Bulk, Interrupt or Isochronous Transfers
  - 3 Programmable Endpoints with double buffering of 64x2 bytes
  - Suspend/Resume Interrupts, and Remote Wake-up Support
  - Power-on Reset and USB Bus Reset
  - 48 Mhz clock for Full-speed Bus Operation
  - USB Bus Disconnection on Microcontroller Request

- Peripheral Features
  - One 8-bit Timer/Counter with Separate Prescaler, Compare Mode and PWM channel
  - One 16-bit Timer/Counter with Separate Prescaler and Compare Mode
  - Hardware Watchdog
- Communication Peripherals
  - High Speed Master/Slave SPI Serial Interface (Up to 20Mhz)
  - 2-Wire Serial Interface
  - UART interface (up to 2Mbps)
- Special Microcontroller Feature
  - Power-on Reset and Brown-out Detection
  - Internal Callibrated Oscillator
  - External and Internal Interrupt Sources
  - Three Sleep Modes: Idle, Power-down and Standby.
- Keyboard Interface with up to 4x4 Matrix Management Capability with Interrupts and Wake-Up on Key Pressed Event
- Up to 18 x I/O Ports: Programmable I/O Port
- Up to 2 x LED Outputs with Programmable Current Sources: 2 or 4 mA
- Operating Temperature
  - Industrial (-40°C to +85°C)
- Core Operating Voltages
  - 2.7 - 5.5V
- DC/DC Operating Voltages (See “Smart Card Interface Characteristics” for details)
  - 2.7 - 5.5V
- Maximum Frequency
  - 8MHz Clock Input

1. Block Diagram

Figure 1-1.



Note

Except for the PORTC, all the other ports are connected to a Pin Change Interrupt Controller.

2. Pin List Configuration

- 3 package configurations to answer different needs
  - 32pins: small package size that embeds all required interfaces to be integrated in small embedded systems (AT90SCR075).
  - 40pins: Package which embeds more features than the 32pins, such as Keyboard feature (AT90SCR075).
  - 28pins:.Small package size that embeds all required interfaces to be integrated in small embedded systems (AT90SCR060)

Note

- USBReg refers to 3.3V USB specific regulator
- PCINTx refer to Pin Change Interrupts. See “External Interrupt Registers” on page 50.

Caution

Beware of the multiple functionality supported on each port. All functionality may not be active at the same time. The only way to disable a feature is to deactive it inside the corresponding peripheral block.

Table 2-1. Pin List Configuration

Portmap	ID	SCR075LQFN32	SCR075HQFN40	SCR060QFN28	Supply	Configuration, Role	
Unmapped, generic pins	Vcc	x	x	x	-	Vcc	Voltage Supply
	Vss1	e(1)	e(1)	e(1)		Vss1	Ground
	AVss	e(1)	e(1)	e(1)		AVss	PLL Ground
	RST	x	x	x	Vcc	RST	**Reset signal:** Drive low to reinitialize the chip
	Xtal1	x	x	x		XTAL1	**Clock Input:** Support up to 8 Mhz crystals
	Xtal2	x	x	x		XTAL2	
	DVcc	x	x	x		DVcc	**Digital Vcc**:Used for internal regulator decoupling
	Vdcdc	x	x	x		Vdcdc	Voltage Supply for DC/DC Converter.
	Vss2	e(1)	e(1)	e(1)		Vss2	Second Vss: To be tied to Vss
	Vss3	e(1)	e(1)	e(1)		Vss3	Third Vss: To be tied to Vss
	CGND	x	x	x		CGND	Core Ground
USB	D+	x	x	x	USB Reg	D+	USB Interface
	D-	x	x	x		D-	
	UCap	x	x	x		UCap	**USB Decoupling**: Used for specific USB regulator decoupling

Table 2-1. Pin List Configuration

Portmap	ID	SCR075LQFN32	SCR075HQFN40	SCR060QFN28	Supply	Configuration, Role				
PORT A	PA3	-	x	-	Vcc	KbIN3			PCINT3	**KbINx**: Input for "Keyboard Interface"
	PA2	-	x	-		KbIN2			PCINT2	
	PA1	-	x	-		KbIN1			PCINT1	
	PA0	-	x	-		KbIN0			PCINT0	
PORT B	PB3	x	x	x	Vcc	PWM	OC0A		PCINT11	**PWM:** Output from "8-bit Timer/Counter0 with PWM"
PORT C	PC5	x	x	x	Vcc	LED3				**SDA, SCL**: "2-wire Serial Interface _ TWI" signals **LEDx**: "LED" Outputs (IO driving current)
	PC4	x	x	x		LED2				
	PC1	x	x	x		SDA				
	PC0	x	x	x		SCL				
PORT D	PD7	x	x	-	Vcc	HSMISO			PCINT23	**HSxxxx**: "High-Speed SPI Controller" (MISO, MOSI, SCK, $\overline{SS}$) **INTx**: "External Interrupts" , default configuration **TXD, RXD**: "UART" signals **CLKO**: "Clock Output Buffer"
	PD6	x	x	-		HSMOSI			PCINT22	
	PD5	x	x	-		HSSCK			PCINT21	
	PD4	x	x	-		HSSS			PCINT20	
	PD2	x	x	x		INT0	CLKO		PCINT18	
	PD1	x	x	x		TXD			PCINT17	
	PD0	x	x	x		RXD			PCINT16	
PORT E	PE3	-	x	-	Vcc	KbO3			PCINT27	**KbOx**: Output for "Keyboard Interface"
	PE2	-	x	-		KbO2			PCINT26	
	PE1	-	x	-		KbO1			PCINT25	
	PE0	-	x	-		KbO0			PCINT24	
Smart Card PORT		x	x	x	Vcc	CPRES				**Cx**: "Smart Card Interface Block (SCIB)" : Standard ISO7816 port **Smart Card Interface**: "DC/DC Converter" Supply Signals
		x	x	x	CVcc	CVcc				
		x	x	x		CVSense				
		x	x	x		CVss				
		x	x	x		LI				
		x	x	x		LO				
		x	x	x	CVcc(2)	CVccIn				
		x	x	x	CVccIn	CCLK				
		x	x	x		CRST				
		x	x	x		CIO				

Notes: 1. Should be connected to e-pad underneath QFN package

2. Signal CVcc is filtered and connected to CVccIn

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.sealsq.com/eng/Products/Smart-Card-Readers/Contact.

4. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

The code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

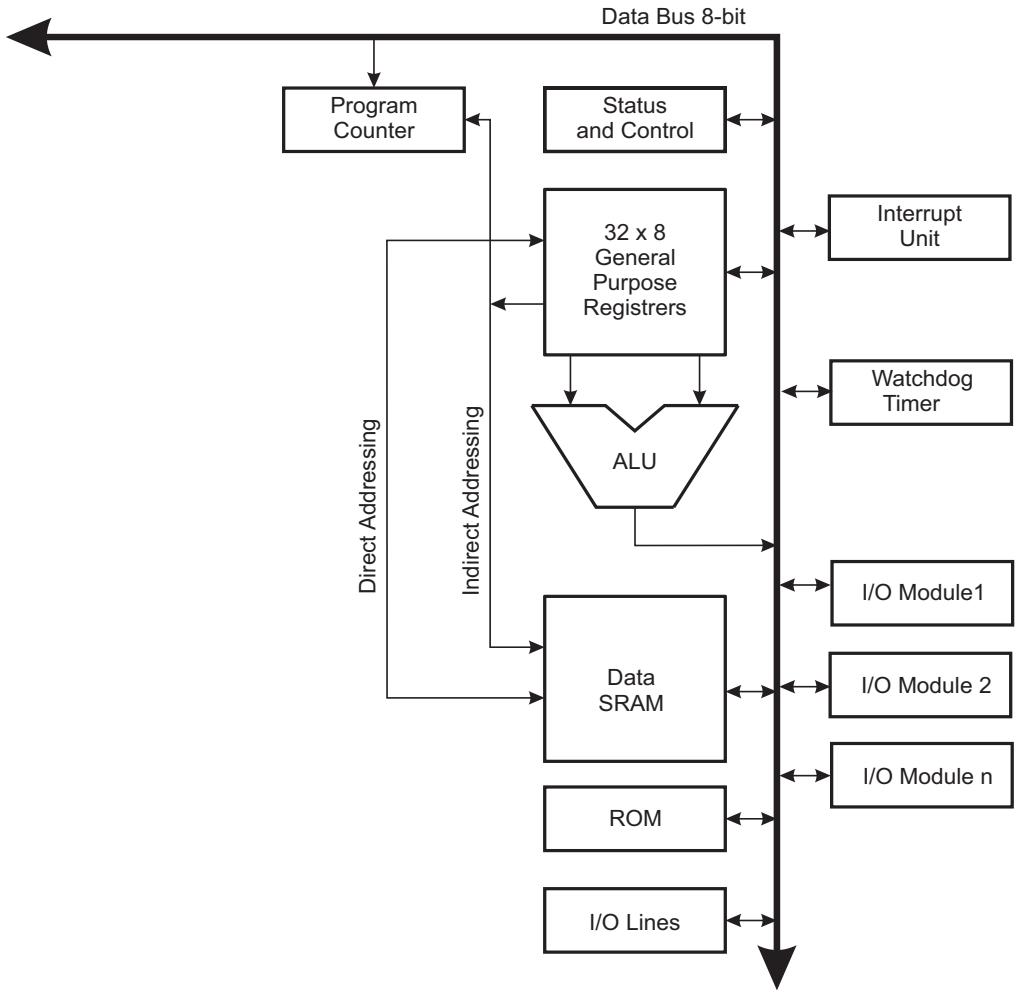
5. 8/16-bit RISC CPU Core

5.1 Introduction

This section discusses the 8/16-bit RISC CPU core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

5.2 Architectural Overview

Figure 5-1. Block Diagram of the 8/16-bit RISC CPU Architecture



In order to maximize performance and parallelism, the 8/16-bit RISC CPU uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typ-

ical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Rom program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most 8/16-bit RISC CPU instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the 8/16-bit RISC CPU architecture.

The memory spaces in the 8/16-bit RISC CPU architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or via the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the AT90SCR075_060 has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

5.3 ALU – Arithmetic Logic Unit

The high-performance 8/16-bit RISC CPU ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate operand are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. This implementation of the architecture also provides a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

5.4 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

5.4.1 SREG – Status Register

The 8/16-bit RISC CPU Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

- **Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic. See “Instruction Set Summary” on page 243 for detailed information.

- **Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive OR between the Negative Flag N and the Two’s Complement Overflow Flag V. See “Instruction Set Summary” on page 243 for detailed information.

- **Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetic. See “Instruction Set Summary” on page 243 for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See “Instruction Set Summary” on page 243 for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See “Instruction Set Summary” on page 243 for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See “Instruction Set Summary” on page 243 for detailed information.

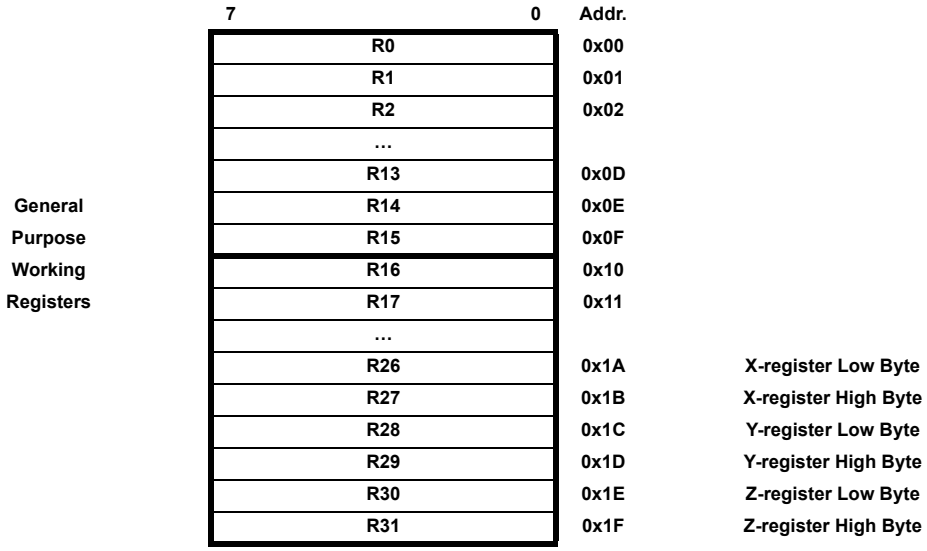
5.5 General Purpose Register File

The Register File is optimized for the 8/16-bit Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 5-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 5-2. 8/16-bit RISC CPU General Purpose Working Registers



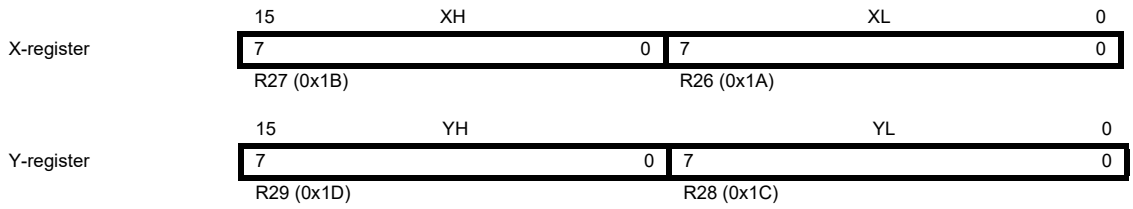
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

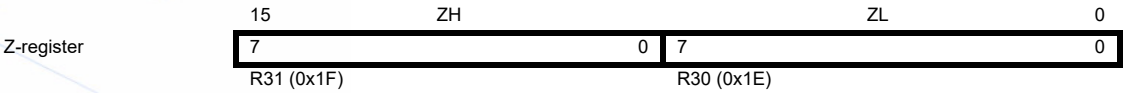
As shown in Figure 5-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

5.5.1 The X-register, Y-register, and Z-register

The registers R26..R31 can also be used as 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 5-3.

Figure 5-3. The X-, Y-, and Z-registers





In the different addressing modes these address registers have fixed displacement, automatic increment, and automatic decrement functionality (see the instruction set reference for details).

5.6 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x0100. The initial value of the stack pointer is the highest address of the internal SRAM. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The 8/16-bit RISC CPU Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent.

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	-	-	-	-	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
0x3D (0x5D)	**SP7**	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	1	0	0	0	0x08
	1	1	1	1	1	1	1	1	0xFF

5.7 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The 8/16-bit RISC CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 5-4 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 5-4. The Parallel Instruction Fetches and Instruction Executions

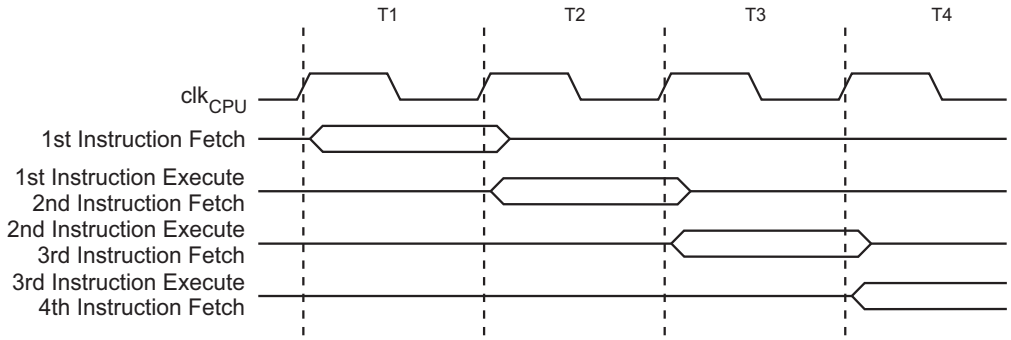
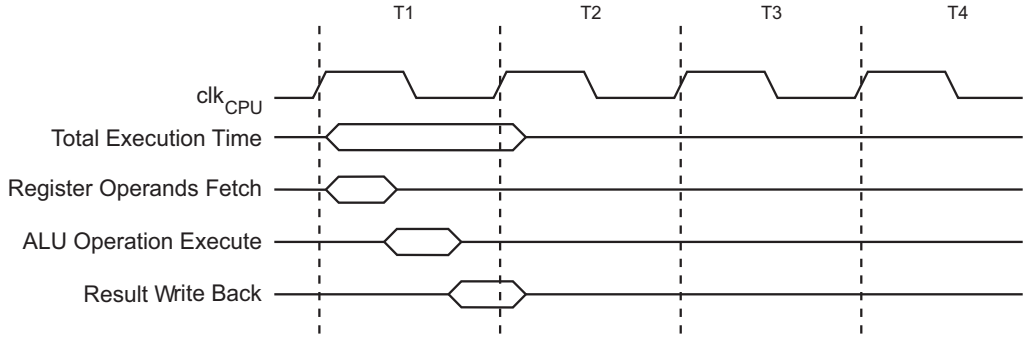


Figure 5-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 5-5. Single Cycle ALU Operation



5.8 Reset and Interrupt Handling

The 8/16-bit RISC CPU provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 47. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is

cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the 8/16-bit RISC CPU exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example
sei *; set Global Interrupt Enable* **sleep** *; enter sleep, waiting for interrupt* ; note: will enter sleep before any pending ; interrupt(s)
C Code Example
__enable_interrupt(); */* set Global Interrupt Enable */* __sleep(); */* enter sleep, waiting for interrupt */* /* note: will enter sleep before any pending interrupt(s) */

5.8.1 Interrupt Response Time

The interrupt execution response for all the enabled 8/16-bit RISC CPU interrupts is five clock cycles minimum. After five clock cycles the program vector address for the actual interrupt handling routine is executed. During these five clock cycle periods, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by five clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

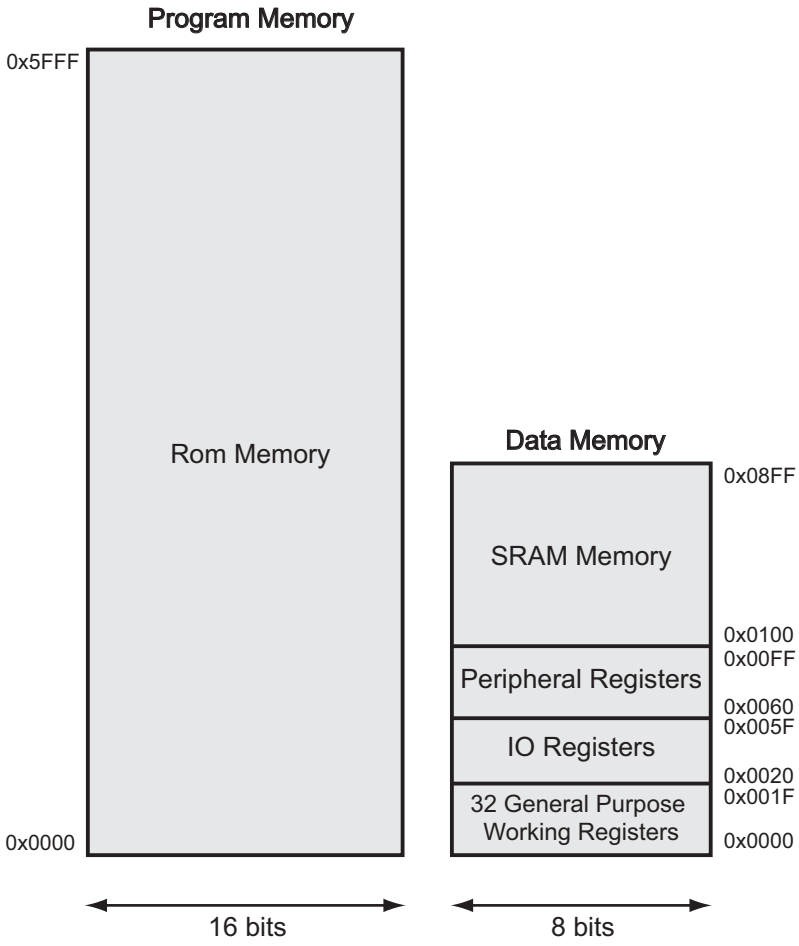
A return from an interrupt handling routine takes five clock cycles. During these five clock cycles, the Program Counter (three bytes) is popped back from the Stack, the Stack Pointer is incremented by three, and the I-bit in SREG is set.

6. AT90SCR075_060 Memories

This section describes the different memories in the AT90SCR075_060.The 8/16-bit RISC CPU architecture has two main memory spaces: the Data Memory and the Program Memory spaces. These memory spaces are linear and regular.

6.1 Memories Overview

Figure 6-1. Full Memory Map



6.2 SRAM Data Memory

Figure 6-1 shows how the SRAM Memory is organized.

The AT90SCR075_060 are complex microcontrollers with more peripheral units than can be supported within the 64 locations reserved in the Opcode of the IN and OUT instructions. For the Extended I/O space from 0x0060 - 0x00FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The address range $00 - $08FF Data Memory address the Register File, the I/O Memory, Extended I/O Memory, and the internal data SRAM. The first 32 locations address the Register file, the next 64 location the standard I/O Memory, then 160 locations of Extended I/O memory and the next 4,096 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

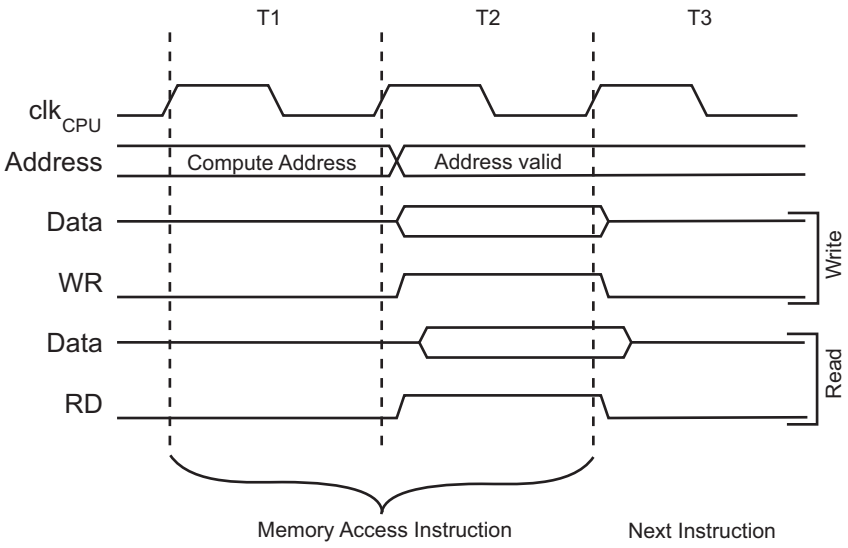
When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O registers, 160 Extended I/O Registers and the 4096 bytes of internal data SRAM are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 17.

6.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk_{CPU} cycles as described in Figure 6-2.

Figure 6-2. On-chip Data SRAM Access Cycles



6.3 I/O Memory

The I/O space definition of the AT90SCR075_060 is shown in “Register Summary” on page 239.

All AT90SCR075_060 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The AT90SCR075_060 are complex microcontrollers with more peripheral units than can be supported within the 64 location

reserved in the Opcode of the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other 8/16-bit RISC CPUs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

6.3.1 General Purpose I/O Registers

The AT90SCR075_060 contain three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

6.3.2 GPIOR2 – General Purpose I/O Register 2

Bit	7	6	5	4	3	2	1	0	
0x2B (0x4B)	GPIOR2 [7..0]								GPIOR2
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

6.3.3 GPIOR1 – General Purpose I/O Register 1

Bit	7	6	5	4	3	2	1	0	
0x2A (0x4A)	GPIOR1 [7..0]								GPIOR1
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

6.3.4 GPIOR0 – General Purpose I/O Register 0

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	GPIOR0 [7..0]								GPIOR0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

7. Clock System

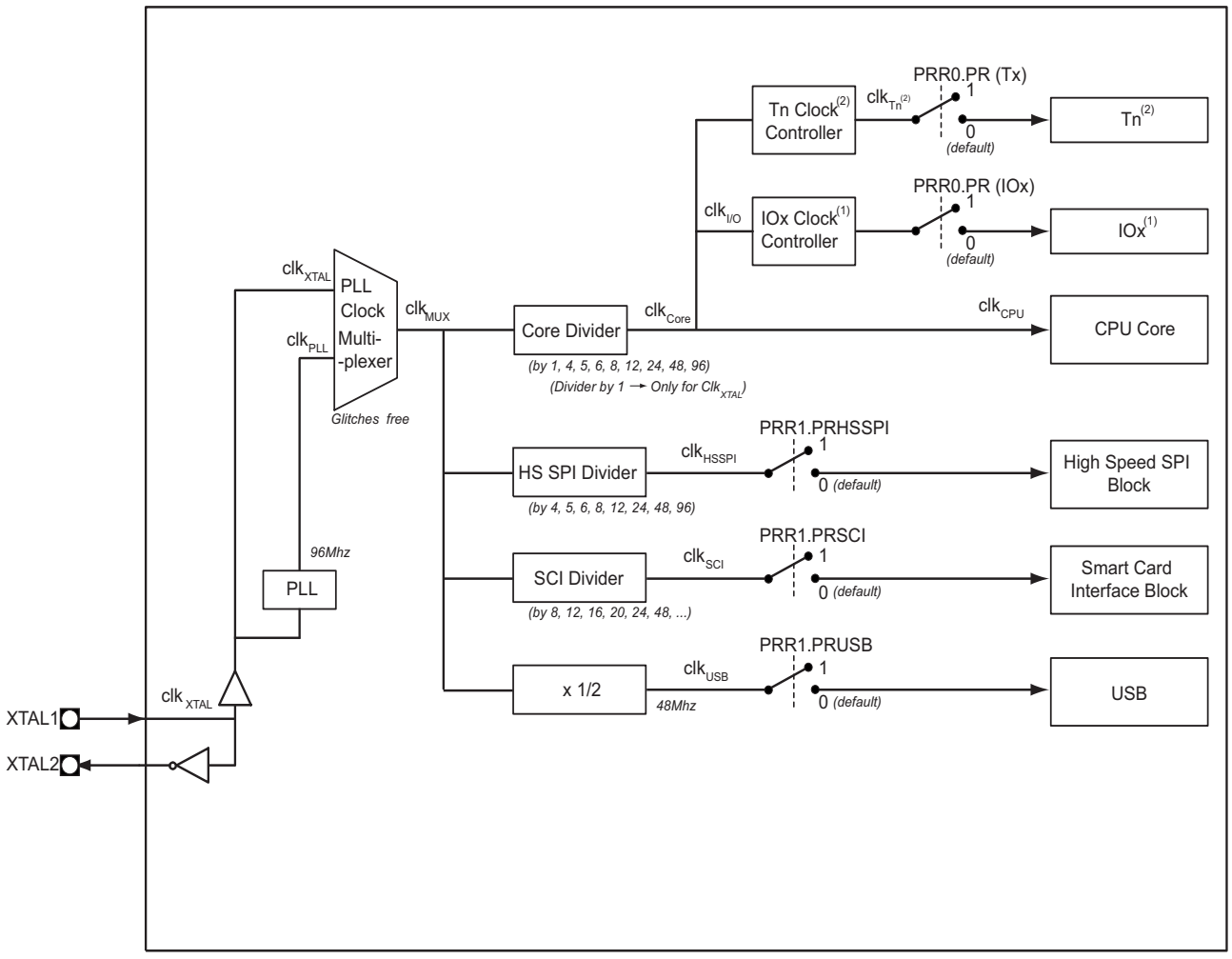
7.1 Overview

The Clock system of AT90SCR075_060 is based on a 8Mhz Oscillator which feeds a Phase Lock Loop (PLL) providing a 96Mhz clock frequency. Then, dividers permit to calibrate the frequencies available for the different Peripherals, Core and Memories.

All of the clocks do not need to be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be stopped by using different sleep modes, as described in “Power Management and Sleep Modes” on page 31. The clock systems are detailed below.

Figure 7-1 presents the principal clock systems in the AT90SCR075_060 and their distribution.

Figure 7-1. Clock Distribution



(1): IOx includes: USART, TWI, Keyboard peripherals
(2): Tn indicates: Timer0 and Timer1

7.1.1 8/16-bit RISC CPU Core Clock - clk_{Core}

The Core clock is providing a clock to all the systems linked with the 8/16-bit RISC CPU Core. The CPU and some peripherals, such as Timers, UART, TWI and Keyboard interface are directly connected to the clk_{Core}.

Clock divisions performed by the Core Divider will also affect $clk_{I/O}$ and clk_{Tn} (depending on the Timer Controller Clock selection).

Please see "CLKPR – Clock Prescale Register" on page 40 for Core Divider access descriptions.

7.1.2 CPU Clock - clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the 8/16-bit RISC CPU core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

7.1.3 I/O Clock - $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters and UART. The I/O clock is also used by the External Interrupt module, but note that some external interrupts are detected by asynchronous logic, allowing such interrupts to be detected even if the I/O clock is halted.

You can choose to disable a peripheral by cutting the clock supplying it. This will reduce the consumption of the AT90SCR075_060. Please refer to "PRR0 – Power Reduction Register 0" on page 33 and "PRR1 – Power Reduction Register 1" on page 34.

7.1.4 High-Speed SPI Clock - clk_{HSSPI}

The High-Speed SPI does not use the clk_{cpu}. Using the clk_{MUX}, through specific dividers, the High-Speed SPI can work to frequencies higher than CPU's one. Please see "HSSPIIER - HSSPI Interrupt Enable Register" on page 180, for details about divider description.

You can reduce AT90SCR075_060 consumption by disabling the clock input into HSSPI, if you don't need this peripheral. See "PRR1 – Power Reduction Register 1" on page 34 for details.

7.1.5 Smart Card Interface Clock - clk_{SCI}

The Smart Card Interface clock is generated from clk_{MUX} via a specific Divider described in the section "SCICLK - Smart Card Clock Register" on page 138. This means the SCIB is capable of operating at frequencies up to 12Mhz.

You can reduce AT90SCR075_060 consumption by disabling the clock input into SCIB, if you don't need this peripheral. See "PRR1 – Power Reduction Register 1" on page 34 for details.

7.1.6 USB Clock - clk_{USB}

The USB module can only work if $clk_{MUX} = clk_{PLL}$. Then, an automatic divider by 2 is applied to reach the 48 Mhz.

USB Device modules use clk_{USB}.

Note

You can reduce AT90SCR075_060 consumption by disabling the clock input into USB Device module if you don't need this peripheral. See "PRR1 – Power Reduction Register 1" on page 34 for details.

7.2 Clock Sources

At chip startup, $clk_{MUX} = clk_{XTAL}$. The external clock must be a 8Mhz clock, and can must be generated by a crystal oscillator.

7.2.1 Default Clock Source

The device is shipped in "Low Power Crystal Oscillator" mode. Its start-up time from Power-Down and Power Save mode is set to 32000 CK. Its start up time from Reset is set to 32000 CK + 14 CK.

Even if the AT90SCR075_060 core has switched to clk_{PLL}, it can switch to clk_{XTAL} again by clearing the PLLCR.PLLMUX bit. See "PLLCR – Phase Lock Loop (PLL) Control Register" on page 29.

7.2.2 Clock Startup Sequence

Any clock source needs a sufficient VCC to start oscillating and a minimum number of oscillating cycles before it can be considered stable.

The BOD circuit is used to ensure sufficient VCC has been achieved.

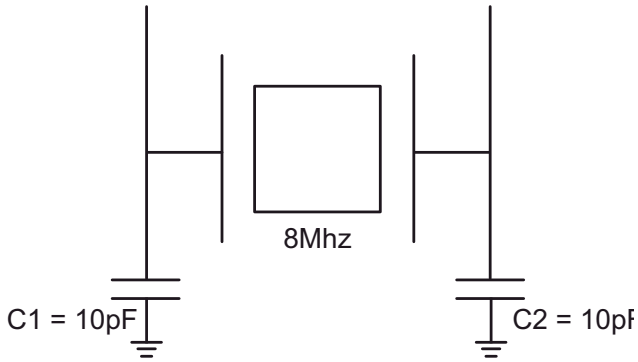
An internal ripple counter monitors the oscillator output clock, and keeps the internal reset active for 32K clock cycles. The reset is then released and the device will start to execute.

7.2.3 Clock Source Connections

The pins XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in Figure 7-2. Either a quartz crystal or a ceramic resonator may be used.

C1 and C2 should always be equal for both crystals and resonators. For ceramic resonators, the capacitor values given by the manufacturer should be used.

Figure 7-2. Crystal Oscillator Connections



7.2.3.1 Low Power Crystal Oscillator

This Crystal Oscillator is a low power oscillator, with reduced voltage swing on the XTAL2 output. It gives the lowest power consumption, but is not capable of driving other clock inputs.

The only crystal supported by the AT90SCR075_060 is an 8Mhz crystal.

7.2.4 PLL Clock

The AT90SCR075_060's PLL is used to generate internal high frequency clock synchronized by an external low-frequency clock of 8Mhz desribed in section "Clock Sources" on page 27.

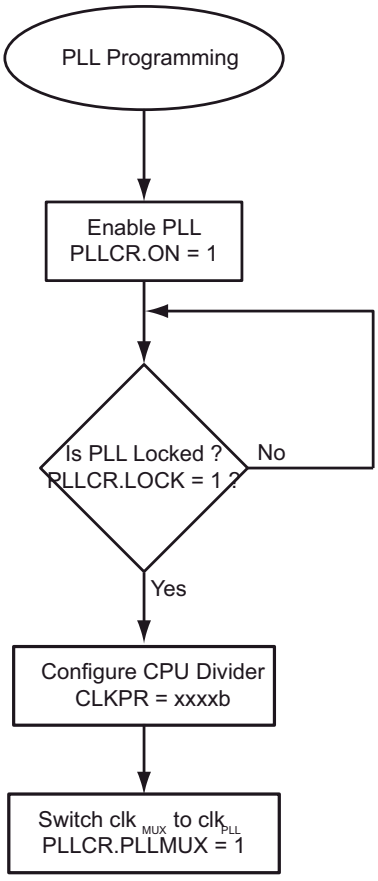
The PLL block combines Phase Frequency Comparator and Lock Detector. This block makes the comparison between a reference clock and a reverse clock and generates some pulses on the Up or Down signal depending on the edges of the reverse clock.

Enabling the PLL by setting PLLCR.ON bit starts the stabilization process. As soon as the PLL is locked, which means that the clock generated is stable, supporting a duty cycle of 50%, the PLLCR.LOCK bit is set.

When the PLL is locked, it is now possible to switch the PLL Clock Multiplexer to the PLL clock, by setting the PLLCR.PLLMUX bit. See "PLLCR – Phase Lock Loop (PLL) Control Register" on page 29.

It is highly recommended that the clk_{CPU} dividers are changed before switching to the clk_{PLL}.

Figure 7-3. Programming PLL



7.3 Clock Output Buffer

The device can output the system clock clk_{CORE} on the CLKO pin. To enable the output, the CKOUT option has to be programmed. This mode is suitable when the chip clock is used to drive

other circuits on the system. The clock will also be output during reset, and the normal operation of the I/O pin will be overriden when CKOUT option is programmed.

7.4 Clock System Registers

7.4.1 PLLCR – Phase Lock Loop (PLL) Control Register

Bit	7	6	5	4	3	2	1	0	
$000062	**PLLMUX**	-	-	-	-	-	**LOCK**	**ON**	**PLLCR**
Read/write	R/W	R	R	R	R	R	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – PLLMUX : PLL Clock Multiplexer Control bit**

Set this bit (1) to switch Clock Multiplexer (clk_{MUX}) to PLL clock (clk_{PLL}).

Clear this bit (0) to switch clk_{MUX} to external clock (clk_{XTAL}).

- **Bit 6..2 – Reserved Bits**

These bits are reserved for future use.

- **Bit 1 – LOCK : PLL Lock Bit Signal**

This bit is set by hardware as soon as the clock generated by the PLL (clk_{PLL}) is stable. It is forbidden to switch clk_{MUX} to clk_{PLL} if the LOCK bit is not set.

Wait for the PLLCR.LOCK bit to be set before switching to clk_{PLL} using the PLLMUX bit.

- **Bit 0 – ON : PLL Start Bit**

Setting this bit (1) will start the PLL. As soon as LOCK bit is set, you can switch on clk_{PLL} clock, not before.

Clearing this bit (0) will stop the PLL.

When the CPU runs on clk_{PLL}, and the PLL is stopped, the CPU will be clocked no longer. This will freeze the CPU and only a reset will be able to start the CPU again.

Before stopping the PLL, make sure that the CPU uses External Clock clk_{XTAL} by using the PLLMUX register.

7.4.2 CLKPR – Clock Prescale Register

Bit	7	6	5	4	3	2	1	0	
$000061	-	-	-	-	**CLKPS3**	**CLKPS2**	**CLKPS1**	**CLKPS0**	**CLKPR**
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..4 – Reserved Bits**

These bits are reserved for future use.

- **Bits 3..0 – CLKPS3..0 : Clock Prescaler Select Bits 3..0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written at run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 7-1.

You can change the CLKPR on the fly. The divider will automatically be active.

Default value of CLKPR is 0x00.

Table 7-1. Clock Prescaler Select (clk_{CPU})

CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR	Clock Division Factor	clk_{CPU} ($clk_{MUX} = clk_{PLL}$)(1)	clk_{CPU} ($clk_{MUX} = clk_{XTAL}$)(2)
0	0	0	0	0x00	1(3)	16Mhz(3)	8Mhz
0	0	0	1	0x01	4	24Mhz	2Mhz
0	0	1	0	0x02	5	19.2Mhz	1.60Mhz
0	0	1	1	0x03	6	16Mhz	1.33Mhz
0	1	0	0	0x04	8	12Mhz	1Mhz
0	1	0	1	0x05	12	8Mhz	0.67Mhz
0	1	1	0	0x06	24	4Mhz	0.33Mhz
0	1	1	1	0x07	48	2Mhz	0.17Mhz
1	0	0	0	0x08	96	1Mhz	0.08Mhz
1	0	0	1	0x09	Reserved		
1	0	1	0	0x0A			
1	0	1	1	0x0B			
1	1	0	0	0x0C			
1	1	0	1	0x0D			
1	1	1	0	0x0E			
1	1	1	1	0X0F			

Notes:
1. clk_{MUX}=96Mhz, clock generated by the PLL
2. clk_{MUX}=8Mhz, clock provided by external clock source XTAL.
3. It is impossible for the CPU core to support 96Mhz, thus, by default, the divide by one clock option generates a 16Mhz clock.

8. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The 8/16-bit RISC CPU provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

To enter any of the three sleep modes, the SE bit in the SMCR must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the SMCR Register select which sleep mode (Idle, Power-down or Standby) will be activated by the SLEEP instruction. See Table 8-2 for a summary. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

Figure 7-1 on page 25 presents the different clock systems in the AT90SCR075_060, and their distribution. The figure is helpful in selecting an appropriate sleep mode.

8.1 Power Modes Descriptions

Please refer to the section entitled "Important note about: Entering and Leaving low consumption modes" on page 35, to read important remarks on achieving minimum consumption.

When waking up from Power-down mode, there is a delay from when the wake-up condition occurs to when the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is to 32000 CK

8.1.1 Idle Mode

When the SM2:0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the peripherals and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and UART Transmit Complete interrupts, as shown in Table 8-1.

8.1.2 Power-down Mode

When the SM2:0 bits are written to 010, the SLEEP instruction makes the MCU enter Power-down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the 2-wire Serial Interface, and the Watchdog continue operating (if enabled). Only an External Reset, a coherent communication request on TWI or USB communication interface, a card insertion/removal, an external interrupt, a pin change interrupt or a keyboard pin pressed interrupt can wake up the MCU. This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

To obtain a minimum consumption level, don't forget to stop the DCDC and PLL, as remarked in section "Important note about: Entering and Leaving low consumption modes" on page 35.

Note that if a level triggered interrupt is used to wake-up from Power-down mode, the changed level must be held for some time to wake up the MCU. Refer to “External Interrupts” on page 50 for details.

When waking up from Power-down mode, there is a delay from when the wake-up condition occurs to when the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is to 32000 CK

8.1.3 Standby Mode

When the SM2:0 bits are 110, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

Table 8-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains				Oscillators		Wake-up Sources				
Sleep Mode	**clk_{CPU}**	**clk_{SCI}**	**$clk_{I/O}$**	**clk_{HSSPI}, clk_{USB}**	**Main Clock Source Enabled**	**Timer Osc Enabled**	**INT0 Pin Change**	**Coherent Communication request**	**Keyboard Pin pressed**	**WDT Interrupt**	**Other I/O**
Idle			X	X	X	X(1)	X	X	X	X	X
Power-down							X(1)	X	X	X	
Standby					X		X(1)	X	X	X	

Notes: 1. For INT0, only level interrupt.

8.2 Power Reduction Register

The Power Reduction Register, PRR, provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock.

8.2.1 SMCR – Sleep Mode Control Register

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	-	-	-	-	SM2	SM1	SM0	SE	SMCR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 3..1 – SM2..0 : Sleep Mode Select Bits 2..0**

These bits select between the five available sleep modes as shown in Table 8-2.

Table 8-2. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	Reserved
0	1	0	Power-down

Table 8-2. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Reserved

Note: 1. Standby modes is only recommended for use with external crystals or resonators.

- **Bit 0 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

8.2.2 PRR0 – Power Reduction Register 0

The Power Reduction Register allows the shut down of peripherals directly connected to CPU resources. These peripherals are activated by default and can be shut down for power consumption reasons if they are not used by an application.

Bit	7	6	5	4	3	2	1	0	
$000064	**PRTWI**	**-**	**PRTIM0**	**-**	**PRTIM1**	**-**	**PRUSART0**	**-**	**PRR0**
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - PRTWI: Power Reduction TWI**

Writing a logic one to this bit shuts down the TWI by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

- **Bit 6 - Res: Reserved bit**

Reserved for future use.

- **Bit 5 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue as before the shutdown.

- **Bit 4 - Res: Reserved bit**

Reserved for future use.

- **Bit 3 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2- Res: Reserved bit**

Reserved for future use.

- **Bit 1 - PRUSART0: Power Reduction UART0**

Writing a logic one to this bit shuts down the UART0 by stopping the clock to the module. When waking up the UART0 again, the UART0 should be reinitialized to ensure proper operation.

- **Bit 0 - Res: Reserved bit**

Reserved for future use.

8.2.3 PRR1 – Power Reduction Register 1

The Power Reduction Register allows the shut down of peripherals directly connected to CPU resources. These peripherals are activated by default and can be shut down for power consumption reasons if they are not used by an application.

Bit	7	6	5	4	3	2	1	0	
$000065	-	-	PRKB	-	PRSCI	PRHSSPI	PRUSB	-	PRR1
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - Res: Reserved bit**

Reserved for future use.

- **Bit 6 - Res: Reserved bit**

Reserved for future use.

- **Bit 5 - PRKB: Power Reduction Keyboard**

Writing a logic one to this bit shuts down the Keyboard module by stopping the clock to the module. When waking up the Keyboard again, the Keyboard should be re initialized to ensure proper operation.

- **Bit 4 - Res: Reserved bit**

Reserved for future use.

- **Bit 3 - PRSCI: Power Reduction Smart Card Interface**

Writing a logic one to this bit shuts down the Smart Card Interface module by stopping the clock to the module. When waking up the SCIB again, the SCIB should be re initialized to ensure proper operation.

- **Bit 2 - PRHSSPI: Power Reduction High Speed SPI**

Writing a logic one to this bit shuts down the High Speed Serial Peripheral Interface by stopping the clock to the module. When waking up the HSSPI again, the HSSPI should be re initialized to ensure proper operation.

- **Bit 1 - PRUSB: Power Reduction USB**

Writing a logic one to this bit shuts down the USB by stopping the clock to the module. When waking up the USB again, the USB should be reinitialized to ensure proper operation.

- **Bit 0 - Res: Reserved bit**

Reserved for future use.

8.3 Important note about: Entering and Leaving low consumption modes

8.3.1 Entering Power Down

It is very important to note that there is no automatic switch to a low consumption mode for the PLL Clock Multiplexer and for the DC/DC converter.

- To lower the powerdown consumption, the PLL Clock Multiplexer must be switched to External Clock and the PLL must be stopped. To do so:
  - Switch from PLL to External Clock by clearing PLLCR.PLLMUX.
  - Turn the PLL off by clearing PLLCR.ON.
  - Wait until PLLCR.LOCK bit is cleared.
- To lower the powerdown consumption, the DC/DC converter must also be switched off. To do so:
  - Clear DCCR.DCON bit. This will generate a deactivation sequence.
  - Wait until DCCR.DCRDY is cleared.

8.3.2 Waking up from Power Down

Do not forget to restore the PLL configuration and DC/DC converter when waking the chip up. Please refer to Figure 7-3 on page 28 and Figure 16-1 on page 142 to re-enable these peripherals.

8.4 Minimizing Power Consumption

There are several issues to consider when trying to minimize the power consumption in a 8/16-bit RISC CPU controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

8.4.1 Watchdog Timer

If the Watchdog Timer is not needed in the application, the module should be turned off. If the Watchdog Timer is enabled, it will be enabled in all sleep modes, and hence, always consume power. In the deeper sleep modes, this will contribute significantly to the total current consumption. Refer to "Interrupts" on page 47 for details on how to configure the Watchdog Timer.

8.4.2 Port Pins

When entering a sleep mode, all port pins should be configured to use minimum power. It is most important to ensure that no pins drive resistive loads. This ensures that no power is consumed by the input logic when not needed. In some cases, the input logic is needed to detect wake-up conditions, and it will then be enabled. Refer to the section "Digital Input Enable and Sleep Modes" on page 60 for details on which pins are enabled. If the input buffer is enabled and the input signal is left floating or has an analog signal level close to $V_{CC}/2$, the input buffer will use excessive power.

9. System Control and Reset

9.1 Resetting the 8/16-bit RISC CPU

During reset, all I/O Registers are set to their initial values, and the program starts execution from the Reset Vector. The instruction placed at the Reset Vector must be a JMP – Absolute Jump – instruction to the reset handling routine. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa. The circuit diagram in Figure 9-1 shows the reset logic. Table 9-1 defines the electrical parameters of the reset circuitry.

The I/O ports of the 8/16-bit RISC CPU are immediately reset to their initial state when a reset source goes active. This does not require any clock source to be running.

9.2 Reset Sources

The AT90SCR075_060 has four sources of reset:

- Power-on Reset. The MCU is reset when the supply voltage is below the Power-on Reset threshold (V_{POT}).
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for longer than the minimum pulse length.
- Watchdog Reset. The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
- Brown-out Reset. The MCU is reset when the supply voltage V_{CC} is below the Brown-out Reset threshold (V_{BOT}) and the Brown-out Detector is enabled.

Figure 9-1. Reset Logic

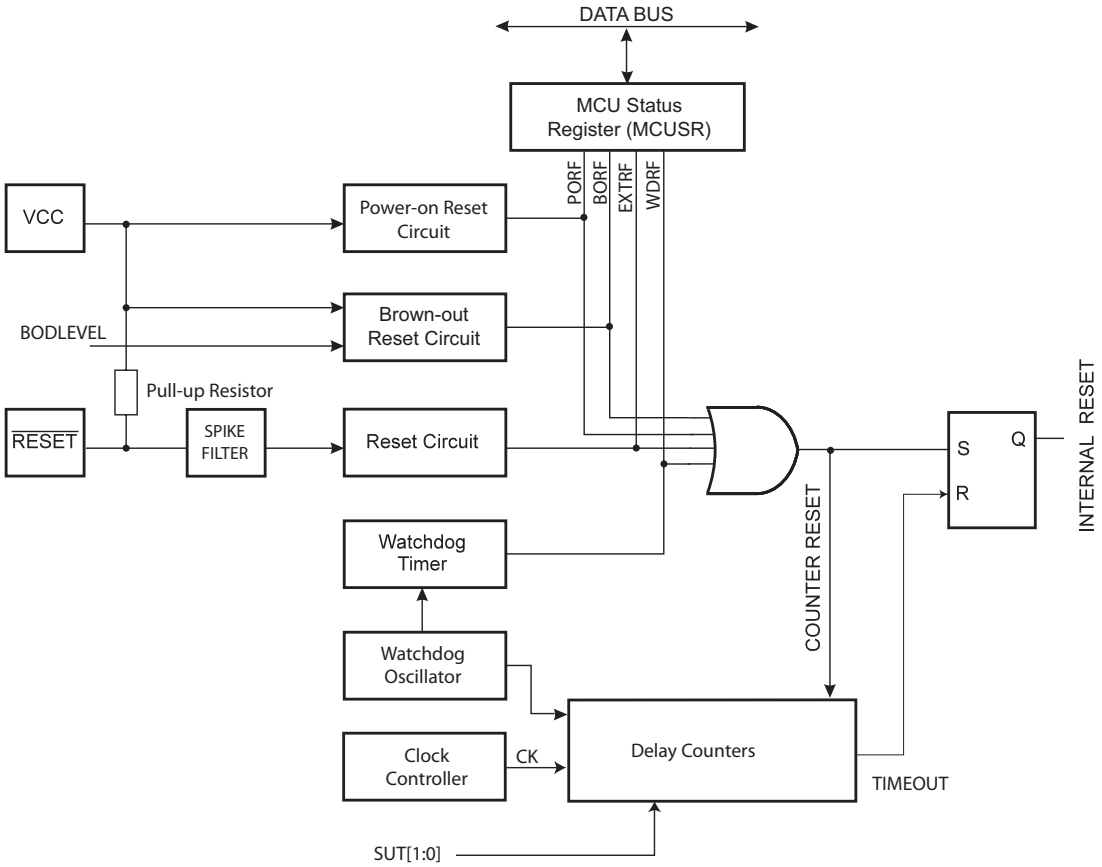


Table 9-1. Reset Characteristics[1]

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{POT}	Power-on Reset Threshold Voltage (rising)			1.4	1.8	V
	Power-on Reset Threshold Voltage (falling)[2]			1.3	1.8	V
V_{RST}	$\overline{\text{RESET}}$ Pin ViL				$0.2V_{CC}$	V
	$\overline{\text{RESET}}$ Pin ViH		$0.85V_{CC}$			

Notes: 1. Values are guidelines only.
2. The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling)

9.2.1 Power-on Reset

A Power-on Reset (POR) pulse is generated by an On-chip detection circuit. The detection level is defined in Table 9-1. The POR is activated whenever V_{CC} is below the detection level. The POR circuit can be used to trigger the start-up Reset, as well as to detect a failure in supply voltage.

A Power-on Reset (POR) circuit ensures that the device is reset from Power-on. Reaching the Power-on Reset threshold voltage invokes the delay counter, which determines how long the

device is kept in RESET after V_{CC} rise. The RESET signal is activated again, without any delay, when V_{CC} decreases below the detection level.

Figure 9-2. MCU Start-up, $\overline{\text{RESET}}$ Tied to VCC

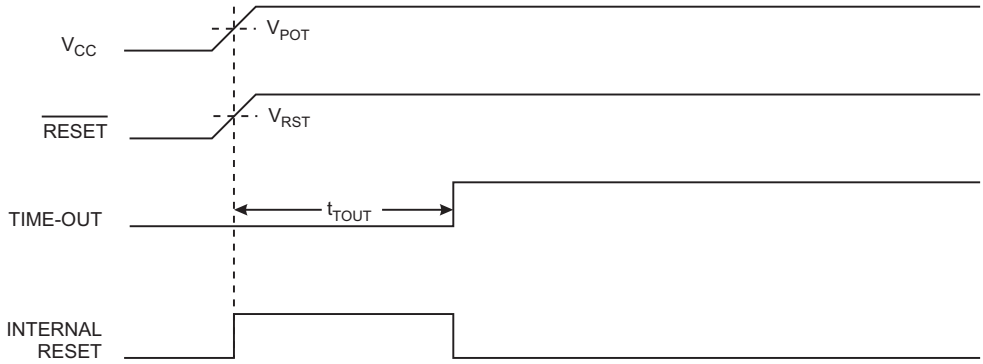
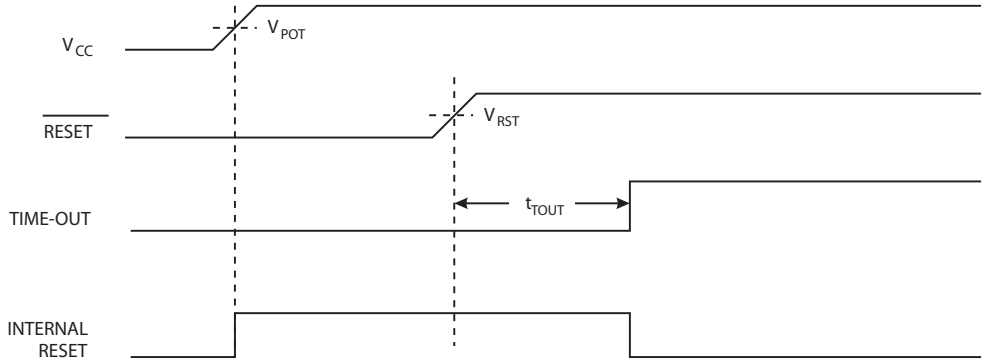


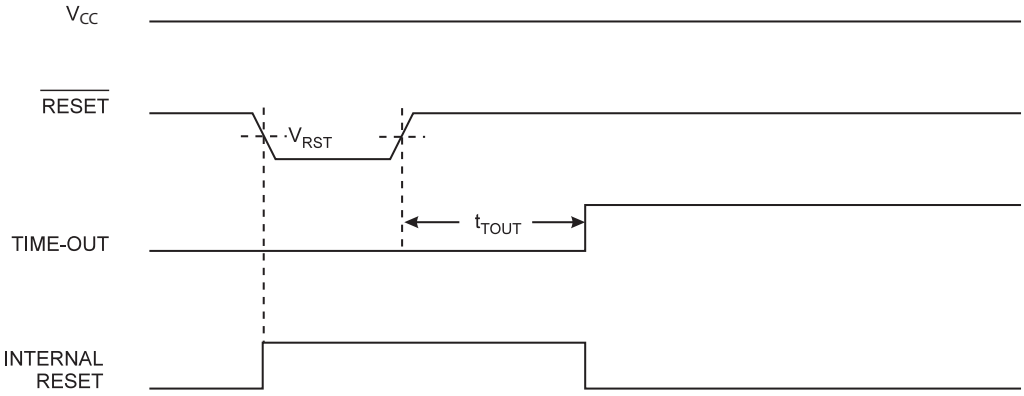
Figure 9-3. MCU Start-up, $\overline{\text{RESET}}$ Extended Externally



9.2.2 External Reset

An External Reset is generated by a low level on the $\overline{\text{RESET}}$ pin. Reset pulses longer than the minimum pulse width (see Table 9-1) will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset. When the applied signal reaches the Reset Threshold Voltage – V_{RST} – on its positive edge, the delay counter starts the MCU after the Time-out period – t_{TOUT} – has expired.

Figure 9-4. External Reset During Operation



9.2.3 Brown-out Detection

AT90SCR075_060 has an On-chip Brown-out Detection (BOD) circuit for monitoring the V_{CC} level during operation by comparing it to a fixed trigger level. The BOD is always activated. The trigger level has a hysteresis to ensure spike free Brown-out Detection. The hysteresis on the detection level should be interpreted as $V_{BOT+} = V_{BOT} + V_{HYST}/2$ and $V_{BOT-} = V_{BOT} - V_{HYST}/2$.

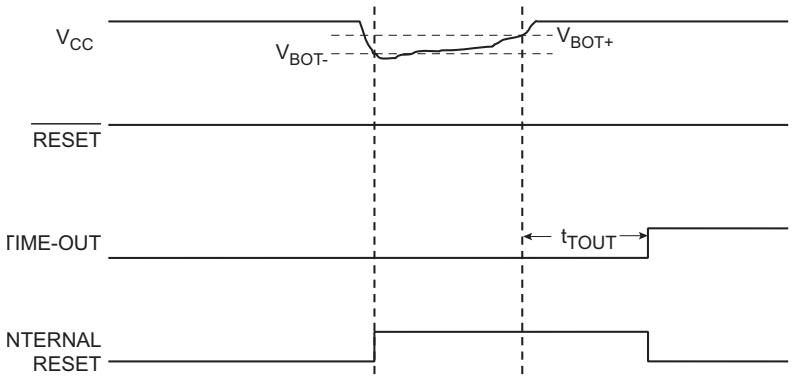
Table 9-2. Brown-out Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{BOT}	Brown-Out trigger level		2.3		V
V_{HYST}	Brown-out Detector Hysteresis		50		mV
t_{BOD}	Min Pulse Width on Brown-out Reset				ns

When V_{CC} decreases to a value below the trigger level (V_{BOT-} in Figure 9-5), the Brown-out Reset is immediately activated. When V_{CC} increases above the trigger level (V_{BOT+} in Figure 9-5), the delay counter starts the MCU after the Time-out period t_{TOUT} has expired.

The BOD circuit will only detect a drop in V_{CC} if the voltage stays below the trigger level for longer than t_{BOD} given in Table 9-1.

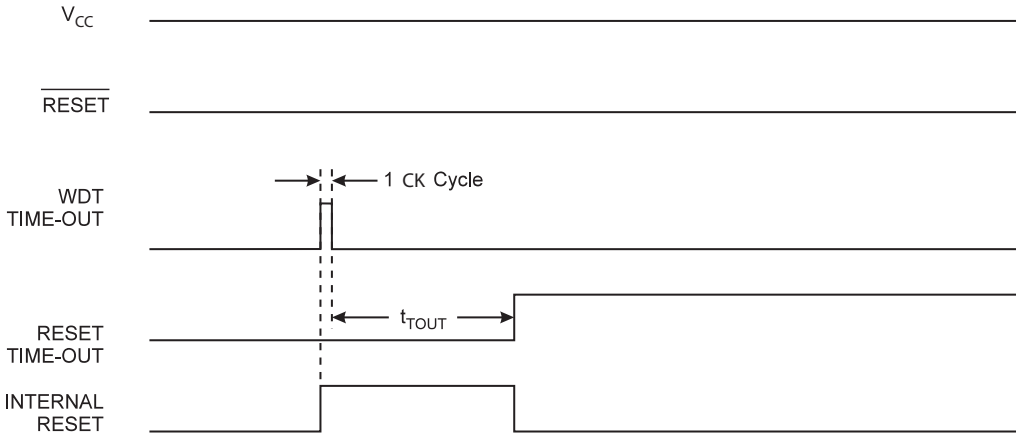
Figure 9-5. Brown-out Reset During Operation



9.2.4 Watchdog Reset

When the Watchdog times out, it will generate a short reset pulse of one CK cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to page 47 for details on operation of the Watchdog Timer.

Figure 9-6. Watchdog Reset During Operation

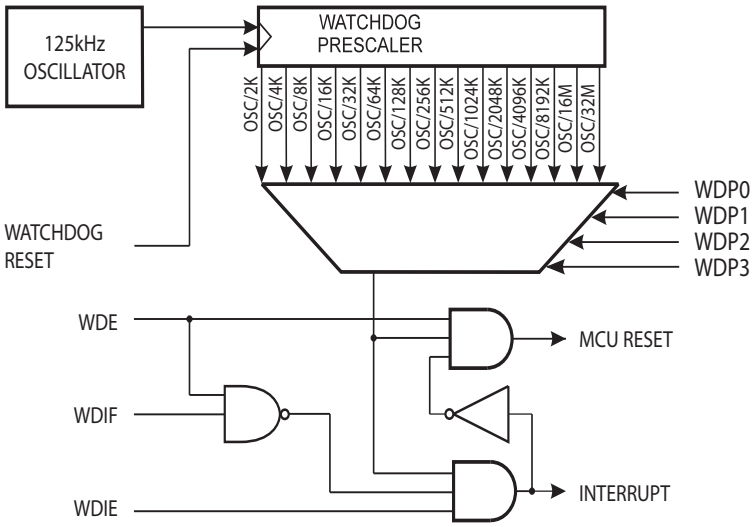


9.3 Watchdog Timer

AT90SCR075_060 has an Enhanced Watchdog Timer (WDT). The main features are:

- Clocked from separate On-chip Oscillator
- 3 Operating modes
  - Interrupt
  - System Reset
  - Interrupt and System Reset
- Selectable Time-out period from 16ms to 13 minutes
- Possible Hardware fuse Watchdog always on (WDTON) for fail-safe mode

Figure 9-7. Watchdog Timer



The Watchdog Timer (WDT) is a timer counting cycles of a separate on-chip 125 kHz oscillator. The WDT gives an interrupt or a system reset when the counter reaches a given time-out value. In normal operation mode, it is required that the system uses the WDR - Watchdog Timer Reset - instruction to restart the counter before the time-out value is reached. If the system doesn't restart the counter, an interrupt or system reset will be issued.

In Interrupt mode, the WDT gives an interrupt when the timer expires. This interrupt can be used to wake the device from sleep-modes, and also as a general system timer. One example is to limit the maximum time allowed for certain operations, giving an interrupt when the operation has run longer than expected. In System Reset mode, the WDT gives a reset when the timer expires. This is typically used to prevent system hang-up in case of runaway code. The third mode, Interrupt and System Reset mode, combines the other two modes by first giving an interrupt and then switch to System Reset mode. This mode will for instance allow a safe shutdown by saving critical parameters before a system reset.

The Watchdog always on (WDTON) fuse, if programmed, will force the Watchdog Timer to System Reset mode. With the fuse programmed the System Reset mode bit (WDE) and Interrupt mode bit (WDIE) are locked to 1 and 0 respectively. To further ensure program security, alterations to the Watchdog set-up must follow timed sequences. The sequence for clearing the WDE and changing the time-out configuration is as follows:

1. In the same operation, write a logic one to the Watchdog change enable bit (WDCE) and WDE. A logic one must be written to WDE regardless of the previous value of the WDE bit.
2. Within the next four clock cycles, write the WDE and Watchdog prescaler bits (WDP) as desired, but with the WDCE bit cleared. This must be done in one operation.

The following code example shows one assembly and one C function for turning off the Watchdog Timer. The example assumes that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during the execution of these functions.

Assembly Code Example

```
WDT_off:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Clear WDRF in MCUSR
  in    r16, MCUSR
  andi  r16, (0xff & (0<<WDRF))
  out   MCUSR, r16
  ; Write logical one to WDCE and WDE
  ; Keep old prescaler setting to prevent unintentional time-out
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; Turn off WDT
  ldi   r16, (0<<WDE)
  out   WDTCSR, r16
  ; Turn on global interrupt
  sei
  ret
```

C Code Example

```
void WDT_off(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Clear WDRF in MCUSR */
  MCUSR &= ~(1<<WDRF);
  /* Write logical one to WDCE and WDE */
  /* Keep old prescaler setting to prevent unintentional time-out */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Turn off WDT */
  WDTCSR = 0x00;
  __enable_interrupt();
}
```

If the Watchdog is accidentally enabled, for example by a runaway pointer or brown-out condition, the device will be reset and the Watchdog Timer will stay enabled. If the code is not set up to handle the Watchdog, this might lead to an eternal loop of time-out resets. To avoid this situation, the application software should always clear the Watchdog System Reset Flag (WDRF) and the WDE control bit in the initialisation routine, even if the Watchdog is not in use.

The following code example shows one assembly and one C function for changing the time-out value of the Watchdog Timer.

Assembly Code Example

```
WDT_Prescaler_Change:
  ; Turn off global interrupt
  cli
  ; Reset Watchdog Timer
  wdr
  ; Start timed sequence
  in    r16, WDTCSR
  ori   r16, (1<<WDCE) | (1<<WDE)
  out   WDTCSR, r16
  ; --  Got four cycles to set the new values from here -
  ; Set new prescaler(time-out) value = 64K cycles (~0.5 s)
  ldi   r16, (1<<WDE) | (1<<WDP2) | (1<<WDP0)
  out   WDTCSR, r16
  ; --  Finished setting new values, used 2 cycles -
  ; Turn on global interrupt
  sei
  ret
```

C Code Example

```
void WDT_Prescaler_Change(void)
{
  __disable_interrupt();
  __watchdog_reset();
  /* Start timed Sequence */
  WDTCSR |= (1<<WDCE) | (1<<WDE);
  /* Set new prescaler(time-out) value = 64K cycles (~0.5 s) */
  WDTCSR  = (1<<WDE) | (1<<WDP2) | (1<<WDP0);
  __enable_interrupt();
}
```

Note

The Watchdog Timer should be reset before any change of the WDP bits, since a change in the WDP bits can result in a time-out when switching to a shorter time-out period.

9.4 Register Description

9.4.1 MCUSR – MCU Status Register

The MCU Status Register provides information on which reset source caused an MCU reset.

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	-	-	-	-	WDRF	BORF	EXTRF	PORF	MCUSR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	See Bit Description								

- **Bit 3 – WDRF: Watchdog Reset Flag**

This bit is set if a Watchdog Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 2 – BORF: Brown-out Reset Flag**

This bit is set if a Brown-out Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 1 – EXTRF: External Reset Flag**

This bit is set if an External Reset occurs. The bit is reset by a Power-on Reset, or by writing a logic zero to the flag.

- **Bit 0 – PORF: Power-on Reset Flag**

This bit is set if a Power-on Reset occurs. The bit is reset only by writing a logic zero to the flag.

To make use of the Reset Flags to identify a reset condition, the user should read and then Reset the MCUSR as early as possible in the program. If the register is cleared before another reset occurs, the source of the reset can be found by examining the Reset Flags.

9.4.2 WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
$000060	**WDIF**	**WDIE**	**WDP3**	**WDCE**	**WDE**	**WDP2**	**WDP1**	**WDP0**	**WDTCSR**
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - WDIF: Watchdog Interrupt Flag**

This bit is set when a time-out occurs in the Watchdog Timer and the Watchdog Timer is configured for interrupt. WDIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, WDIF is cleared by writing a logic one to the flag. When the I-bit in SREG and WDIE are set, the Watchdog Time-out Interrupt is executed.

- **Bit 6 - WDIE: Watchdog Interrupt Enable**

When this bit is written to one and the I-bit in the Status Register is set, the Watchdog Interrupt is enabled. If WDE is cleared in combination with this setting, the Watchdog Timer is in Interrupt Mode, and the corresponding interrupt is executed if a time-out in the Watchdog Timer occurs.

If WDE is set, the Watchdog Timer is in Interrupt and System Reset Mode. The first time-out in the Watchdog Timer will set WDIF. Executing the corresponding interrupt vector will clear WDIE and WDIF automatically by hardware (the Watchdog goes to System Reset Mode). This is useful for keeping the Watchdog Timer security while using the interrupt. To stay in Interrupt and System Reset Mode, WDIE must be set after each interrupt. This should however not be done within the interrupt service routine itself, as this might compromise the safety-function of the

Watchdog System Reset mode. If the interrupt is not executed before the next time-out, a System Reset will be applied.

Table 9-3. Watchdog Timer Configuration

WDTON[1]	WDE	WDIE	Mode	Action on Time-out
0	0	0	Stopped	None
0	0	1	Interrupt Mode	Interrupt
0	1	0	System Reset Mode	Reset
0	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
1	x	x	System Reset Mode	Reset

Notes: 1. WDTON is a fuse bit (see “Fuse High Byte” on page 393).

- **Bit 4 - WDCE: Watchdog Change Enable**

This bit is used in timed sequences for changing WDE and prescaler bits. To clear the WDE bit, and/or change the prescaler bits, WDCE must be set.

Once written to one, hardware will clear WDCE after four clock cycles.

- **Bit 3 - WDE: Watchdog System Reset Enable**

WDE is overridden by WDRF in MCUSR. This means that WDE is always set when WDRF is set. To clear WDE, WDRF must be cleared first. This feature ensures multiple resets during conditions causing failure, and a safe start-up after the failure.

- **Bit 5, 2..0 - WDP3..0: Watchdog Timer Prescaler 3..0**

The WDP3:0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is running. The different prescaling values and their corresponding time-out periods are shown in Table 9-4 on page 45..

Table 9-4. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Max Time-out at V_{CC} = 5.0V[1]
0	0	0	0	2K (2048) cycles	24.1 ms
0	0	0	1	4K (4096) cycles	48.2 ms
0	0	1	0	8K (8192) cycles	96.4 ms
0	0	1	1	16K (16384) cycles	0.19 s
0	1	0	0	32K (32768) cycles	0.39 s
0	1	0	1	64K (65536) cycles	0.77 s
0	1	1	0	128K (131072) cycles	1.54 s
0	1	1	1	256K (262144) cycles	3.1 s
1	0	0	0	512K (524288) cycles	6.2 s
1	0	0	1	1024K (1048576) cycles	12.3 s
1	0	1	0	2048K (2097152) cycles	24.8s
1	0	1	1	4096K (4194304) cycles	49.6s

Table 9-4. Watchdog Timer Prescale Select

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Max Time-out at V_{CC} = 5.0V(1)
1	1	0	0	8192K (8388608) cycles	1m39.2s
1	1	0	1	16384K (16777216) cycles	3m18.4s
1	1	1	0	32768K (33554432) cycles	6m36.8s
1	1	1	1	65536K (67108864) cycles	13m13.6s

Note: 1. 85KHz Oscillator used for Max Time-Out

10. Interrupts

This section describes the specifics of the interrupt handling as performed in AT90SCR075_060. For a general explanation of the 8/16-bit RISC CPU interrupt handling, refer to “Reset and Interrupt Handling” on page 19.

10.1 Interrupt Vectors in AT90SCR075_060

Table 10-1. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset
2	$0002	INT0	External Interrupt Request 0
3	--	RFU	Reserved For Future use
4	--	RFU	Reserved For Future use
5	--	RFU	Reserved For Future use
6	$000A	PCINT0	Pin Change Interrupt Request 0
7	$000C	PCINT1	Pin Change Interrupt Request 1
8	$000E	PCINT2	Pin Change Interrupt Request 2
9	$0010	WDT	Watchdog Time-out Interrupt
10	--	RFU	Reserved For Future use
11	--	RFU	Reserved For Future use
12	--	RFU	Reserved For Future use
13	--	RFU	Reserved For Future use
14	$001A	TIMER1_COMPA	Timer/Counter1 Compare Match A
15	--	RFU	Reserved For Future use
16	$001E	TIMER1_OVF	Timer/Counter1 Overflow
17	$0020	TIMER0_COMPA	Timer/Counter0 Compare Match A
18	--	RFU	Reserved For Future use
19	$0024	TIMER0_OVF	Timer/Counter0 Overflow
20	--	RFU	Reserved For Future use
21	$0028	USART_RX	UART Rx Complete
22	$002A	USART_UDRE	UART Data Register Empty
23	$002C	USART_TX	UART Tx Complete
24	--	RFU	Reserved For Future use
25	--	RFU	Reserved For Future use
26	--	RFU	Reserved For Future use
27	$0034	TWI	2-wire Serial Interface
28	--	RFU	Reserved For Future use
29	$0038	KEYBOARD	Keyboard Input Changed

Table 10-1. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address	Source	Interrupt Definition
30	--	RFU	Reserved For Future use
31	$003C	HSSPI	High-Speed SPI Interface
32	$003E	USB Endpoint	USB Endpoint linked Interrupt
33	$0040	USB Protocol	USB Protocol Interrupt
34	$0042	SCIB	Smart Card Reader Interface
35	--	RFU	Reserved For Future use
36	--	RFU	Reserved For Future use
37	$0048	CPRES	Card Presence Detection
38	$004A	PCINT3	Pin Change Interrupt Request 3

The most typical and general program setup for the Reset and Interrupt Vector Addresses in AT90SCR075_060 is:

```
Address     Labels      Code                                Comments
0x                      jmp         RESET                   ; Reset Handler
0x02                    jmp         INT0                    ; IRQ0 Handler
0x04                    jmp         NO_VECT                 ; Vector Forbidden
0x06                    jmp         INT2                    ; IRQ2 Handler
0x08                    jmp         INT3                    ; IRQ3 Handler
0x0A                    jmp         PCINT0                  ; PCINT0 Handler
0x0C                    jmp         PCINT1                  ; PCINT1 Handler
0x0E                    jmp         PCINT2                  ; PCINT2 Handler
0x10                    jmp         WDT                     ; Watchdog Timeout Handler
0x12                    jmp         NO_VECT                 ; Vector Forbidden
0x14                    jmp         NO_VECT                 ; Vector Forbidden
0x16                    jmp         NO_VECT                 ; Vector Forbidden
0x18                    jmp         NO_VECT                 ; Vector Forbidden
0x1A                    jmp         TIM1_COMPA              ; Timer1 CompareA Handler
0x1C                    jmp         NO_VECT                 ; Vector Forbidden
0x1E                    jmp         TIM1_OVF                ; Timer1 Overflow Handler
0x20                    jmp         TIM0_COMPA              ; Timer0 CompareA Handler
0x22                    jmp         NO_VECT                 ; Vector Forbidden
0x24                    jmp         TIM0_OVF                ; Timer0 Overflow Handler
0x26                    jmp         NO_VECT                 ; Vector Forbidden
0x28                    jmp         USART0_RXC              ; UART0 RX Complete Handler
0x2A                    jmp         USART0_UDRE             ; UART0,UDR Empty Handler
0x2C                    jmp         USART0_TXC              ; UART0 TX Complete Handler
0x2E                    jmp         NO_VECT                 ; Vector Forbidden
0x30                    jmp         NO_VECT                 ; Vector Forbidden
0x32                    jmp         NO_VECT                 ; Vector Forbidden
0x34                    jmp         TWI                     ; 2-wire Serial Handler
0x36                    jmp         NO_VECT                 ; Vector Forbidden
```

```
0x38                jmp         KEYBOARD                ; Keyboard Handler
0x3A                jmp         NO_VECT                 ; Vector Forbidden
0x3C                jmp         HSSPI                   ; HighSPI IT Handler
0x3E                jmp         USB_Endpoint            ; USB Endpoint IT Handler
0x40                jmp         USB_Protocol            ; USB Protocol IT Handler
0x42                jmp         SCIB                    ; Smart Card Interface IT
0x44                jmp         NO_VECT                 ; Vector Forbidden
0x46                jmp         NO_VECT                 ; Vector Forbidden
0x48                jmp         CPRES                   ; Card inserted/removed
0x4A                jmp         PCINT3                  ; PCINT3 Handler
;
0x3E      RESET:    ldi         r16, high(RAMEND)       ; Main program start
0x3F                out         SPH,r16                 ; Set Stack Pointer to top
                                                        of RAM
0x40                ldi         r16, low(RAMEND)
0x41                out         SPL,r16
0x42                sei                                 ; Enable interrupts
0x43                <instr>     xxx
...       ...       ...         ...
```

11. External Interrupts

The External Interrupts are triggered by the INT0 pin or any of the PCINT27:0 pins. Observe that, if enabled, the interrupts will trigger even if the INT0 or PCINT27:0 pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin change interrupt PCI3 will trigger if any enabled PCINT27:24 pin toggles, Pin change interrupt PCI2 will trigger if any enabled PCINT23:16 pin toggles, Pin change interrupt PCI1 will trigger if enabled PCINT11 and Pin change interrupt PCI0 will trigger if any enabled PCINT3:0 toggles. PCMSK3, PCMSK2, PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT27:0 are detected asynchronously. This implies that these interrupts can also be used for waking the part from sleep modes other than Idle mode.

The External Interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Registers – EICRA (INT0). When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low. Low level interrupts and the edge interrupt on INT0 are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

If a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated..

11.1 External Interrupt Registers

11.1.1 EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
$000069	-	-	-	-	-	-	ISC01	ISC00	EICRA
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 – ISC00, ISC00 : External Interrupt 0 Sense Control Bits**

The External Interrupt 0 is activated by the external pins INT0 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in Table 11-1. Edges on INT0 is registered asynchronously. Pulses on INT0 pins wider than the minimum pulse width given in Table 11-2 will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If a low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low. When changing the ISC0n bit, an interrupt can occur. Therefore, it is recommended to first disable INT0 by clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISC0n bit can be changed. Finally, the INT0 interrupt flag should be cleared by writing

a logical one to its Interrupt Flag bit (INTFn) in the EIFR Register before the interrupt is re-enabled.

Table 11-1. Interrupt Sense Control[1]

ISCn1	ISCn0	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any edge of INT0 generates asynchronously an interrupt request.
1	0	The falling edge of INT0 generates asynchronously an interrupt request.
1	1	The rising edge of INT0 generates asynchronously an interrupt request.

Note: 1. When changing the ISC01/ISC00 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed

Table 11-2. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

11.1.2 EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	-	-	-	-	-	-	-	INT0	EIMSK
Read/write	R	R	R	R	R	R	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 3..0 – INT0 : External Interrupt Request 0 Enable**

When an INT0 bit is written to one and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt 0 is enabled. The Interrupt Sense Control bits in the External Interrupt Control Register, EICRA, defines whether the external interrupt is activated on rising or falling edge or level sensed. Activity on this pin will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

11.1.3 EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	-	-	-	-	-	-	-	INTF0	EIFR
Read/write	R	R	R	R	R	R	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 3..0 – INTF0 : External Interrupt Flags 0**

When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT0 in EIMSK, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is

executed. Alternatively, the flag can be cleared by writing a logical one to it. This flag is always cleared when INT0 isconfigured as level interrupt.

Note

When entering sleep mode with the INT0 interrupt disabled, the input buffers on this pin will be disabled. This may cause a logic change in internal signals which will set the INTF0 flag. See “Digital Input Enable and Sleep Modes” on page 60 for more information.

11.1.4 PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
0x68	-	-	-	-	PCIE3	PCIE2	PCIE1	PCIE0	PCICR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3 – PCIE3 : Pin Change Interrupt Enable 3**

When the PCIE3 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 3 is enabled. Any change on any enabled PCINT27:24 pin will cause an interrupt. The corresponding interrupt of the Pin Change Interrupt Request is executed from the PCI3 Interrupt Vector. PCINT27:24 pins are enabled individually by the PCMSK3 Register.

- **Bit 2 – PCIE2 : Pin Change Interrupt Enable 2**

When the PCIE2 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 2 is enabled. Any change on any enabled PCINT23:16 pin will cause an interrupt. The corresponding interrupt of the Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT23:16 pins are enabled individually by the PCMSK2 Register.

- **Bit 1 – PCIE1 : Pin Change Interrupt Enable 1**

When the PCIE1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 1 is enabled. Any change on enabled PCINT11 pin will cause an interrupt. The corresponding interrupt of the Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT11 pin is enabled by the PCMSK1 Register.

- **Bit 0 – PCIE0 : Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT3:0 pin will cause an interrupt. The corresponding interrupt of the Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT3:0 pins are enabled individually by the PCMSK0 Register.

11.1.5 PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	-	-	-	-	PCIF3	PCIF2	PCIF1	PCIF0	PCIFR
Read/write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3– PCIF3 : Pin Change Interrupt Flag 3**

When a logic change on any PCINT27:24 pin triggers an interrupt request, PCIF3 becomes set (one). If the I-bit in SREG and the PCIE3 bit in PCICR are set (one), the MCU will jump to the

corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 2 – PCIF2 : Pin Change Interrupt Flag 2**

When a logic change on any PCINT23:16 pin triggers an interrupt request, PCIF2 becomes set (one). If the I-bit in SREG and the PCIE2 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 1 – PCIF1 : Pin Change Interrupt Flag 1**

When a logic change on PCINT11 pin triggers an interrupt request, PCIF1 becomes set (one). If the I-bit in SREG and the PCIE1 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

- **Bit 0 – PCIF0 : Pin Change Interrupt Flag 0**

When a logic change on any PCINT3:0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in PCICR are set (one), the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

11.1.6 PCMSK3 – Pin Change Mask Register 3

Bit	7	6	5	4	3	2	1	0	
0x000073	-	-	-	-	PCINT27	PCINT26	PCINT25	PCINT24	PCMSK3
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3..0 – PCINT27..24 : Pin Change Enable Mask 27..24**

Each PCINT27:24 bit selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT27:24 is set and the PCIE3 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT27:24 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

11.1.7 PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
0x00006D	PCINT23	PCINT22	PCINT21	PCINT20	-	PCINT18	PCINT17	PCINT16	PCMSK2
Read/write	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..4 – PCINT23..20 : Pin Change Enable Mask 23..20**

Each PCINT23:20-bit selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT23:20 is set and the PCIE2 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT23:20 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

- **Bit 3 : Reserved Bit**

This bit is reserved for future use.

- **Bit 2..0 – PCINT18..16 : Pin Change Enable Mask 18..16**

Each PCINT18:16-bit selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT18:16 is set and the PCIE2 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT18:16 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

11.1.8 PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
0x00006C	-	-	-	-	**PCINT11**	-	-	-	**PCMSK1**
Read/write	R	R	R	R	R/W	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3 – PCINT11 : Pin Change Enable Mask 11**

PCINT11 bit selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT11 is set and the PCIE1 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT11 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

11.1.9 PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
0x00006B	-	-	-	-	**PCINT3**	**PCINT2**	**PCINT1**	**PCINT0**	**PCMSK0**
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..0 – PCINT3..0: Pin Change Enable Mask 3..0**

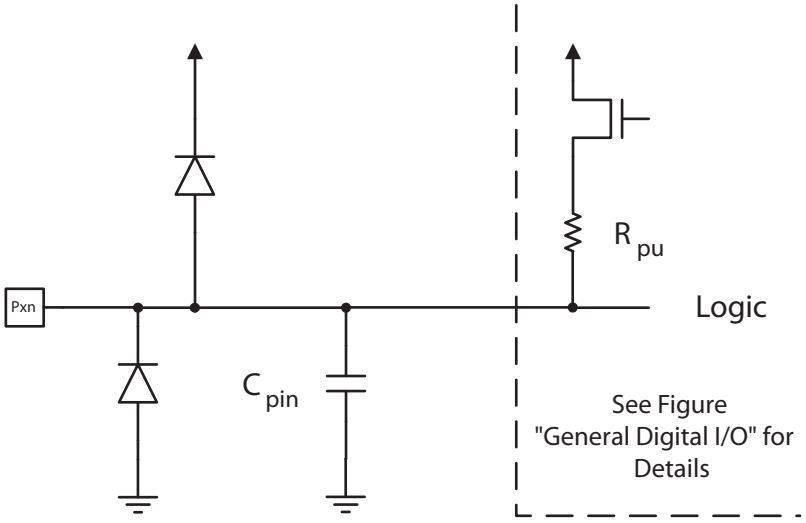
Each PCINT3:0 bit selects whether the pin change interrupt is enabled on the corresponding I/O pin. If PCINT3:0 is set and the PCIE0 bit in PCICR is set, the pin change interrupt is enabled on the corresponding I/O pin. If PCINT3:0 is cleared, the pin change interrupt on the corresponding I/O pin is disabled.

12. I/O Ports

12.1 Standard IO Ports

All 8/16-bit RISC CPU ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 12-1. Refer to “Electrical Characteristics” on page 232 for a complete list of parameters.

Figure 12-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the reference letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTD0 for bit no. 0 in Port D, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in “Register Description for I/O-Ports” on page 62.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

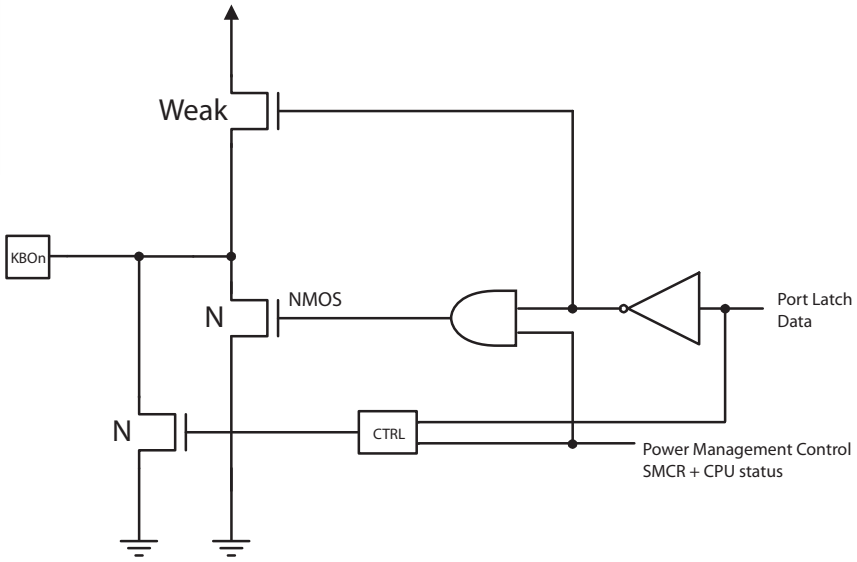
Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O” on page 57. Most port pins are multiplexed with alternate functions for the peripheral features on the device. To get a full description of the different pin configuration, refer to “Pin List Configuration” on page 9.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

12.2 Specific Low Speed Keyboard Output

To avoid abusive EMC generation, AT90SCR075_060 embeds specific Low Speed Output Pads.

Figure 12-2. Low-speed Output Schematic



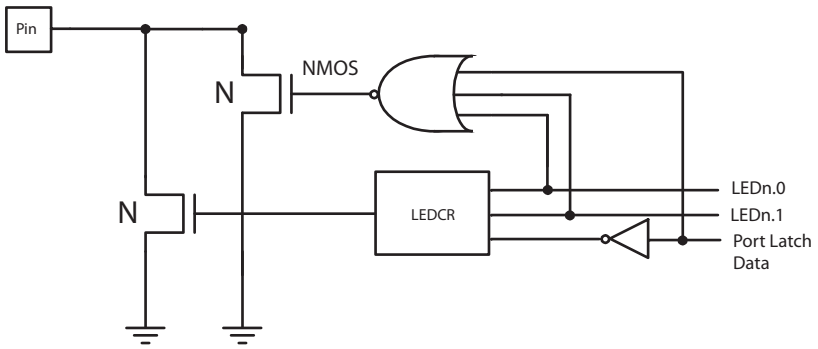
The current limitation of the CTRL block requires a polarisation current of about 250µA. This block is automatically disabled in power-down, power-save, standby and extended standby modes.

These pads only concern KBOn pins located on port PE [0..7]. See “Pin List Configuration” on page 9.

12.3 LED

AT90SCR075_060 supports specific ports driving current to allow easy control of LED displays.

Figure 12-3. LED Source Current



Note

1. When switching a low level, LEDCR device has a permanent current of about N mA/15 (N is 2 or 4)
2. The port must be configured and driven as standard IO port.

Table 12-1. LED Configuration

LEDx.1	LEDx.0	Port Latch Data	NMOS	PIN	Comments
0	0	0	1	0	LED control disabled(1)
0	0	1	0	1	
0	1	0	0	0	LED mode 2 mA
0	1	1	0	1	
1	0	0	0	0	LED mode 4 mA
1	0	1	0	1	

Notes: 1. When LED control disabled, a current of ~8mA is provided on the port.

12.3.1 LEDCR - LED Control Register

Bit	7	6	5	4	3	2	1	0	
$000075	LED3 [1..0]		LED2 [1..0]		-		-		LEDCR
Read/write	R/W	R/W	R/W	R/W	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3..2, 1..0 - LEDn [1..0]: LED Port configuration bits**

Table 15-1 .LED Port Selection

LEDn1	LEDn0	Description
0	0	LED control disabled
0	1	2 mA Current Source
1	0	4 mA Current Source
1	1	Reserved Configuration

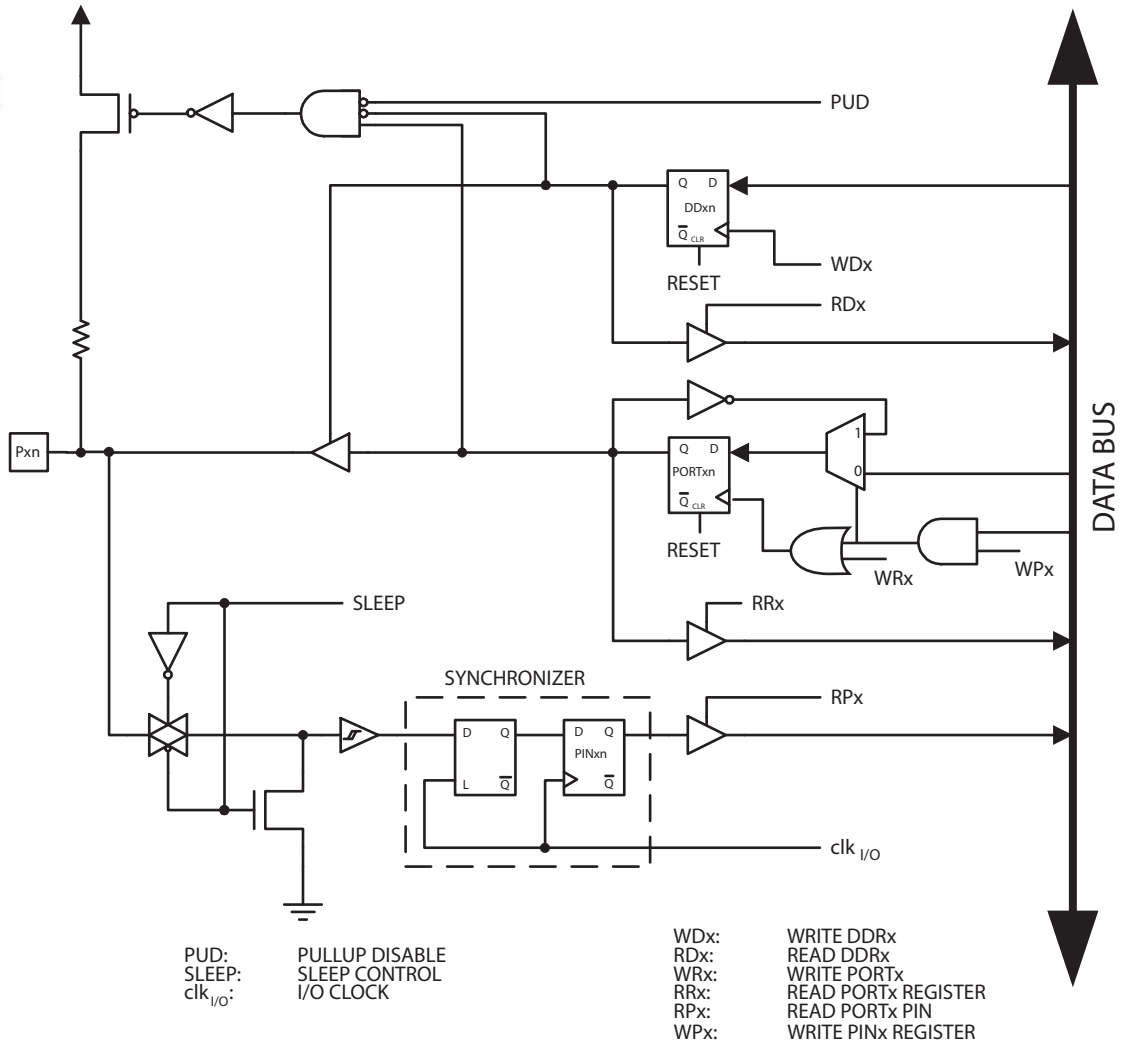
Note

For implementation example, please see section “Typical Application” on page 224.

12.4 Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 12-4 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 12-4. General Digital I/O



Note

WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

12.4.1 Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in “Register Description for I/O-Ports” on page 62, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when the reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

12.4.2 Toggling the Pin

Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DDRxn. Note that the SBI instruction can be used to toggle one single bit in a port.

12.4.3 Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

Table 12-2 summarizes the control signals for the pin value.

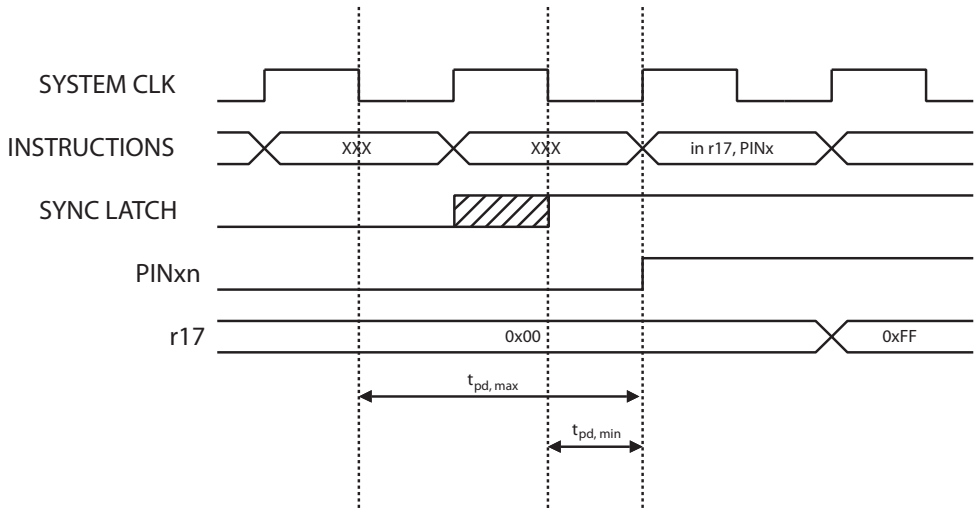
Table 12-2. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low.
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

12.4.4 Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Figure 12-4, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 12-5 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

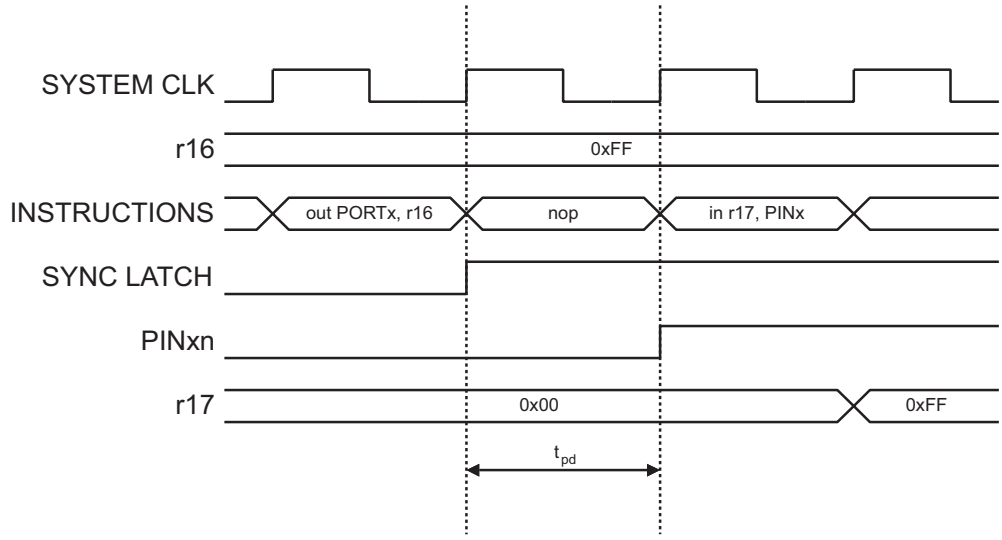
Figure 12-5. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in Figure 12-6. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 12-6. Synchronization when Reading a Software Assigned Pin Value



12.4.5 Digital Input Enable and Sleep Modes

As shown in Figure 12-4, the digital input signal can be clamped to ground at the input of the schmitt-trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in

Power-down mode and Standby to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins.

If a logic high level (“one”) is present on an asynchronous external interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is *not* enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

12.4.6 Unconnected Pins

If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

12.4.7 MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	-	-	-	**PUD**	-	-	-	-	**MCUCR**
Read/write	R/W	R	R	R/W	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See “Configuring the Pin” on page 58 for more details about this feature.

12.5 Register Description for I/O-Ports

12.5.1 PORTA – Port A Data Register

Bit	7	6	5	4	3	2	1	0	
0x02 (0x22)	PORTA [7..0]								PORTA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.2 DDRA – Port A Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x01 (0x21)	DDA [7..0]								DDRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.3 PINA – Port A Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x00 (0x20)	PINA [7..0]								PINA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

12.5.4 PORTB – Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB [7..0]								PORTB
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.5 DDRB – Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB [7..0]								DDRB
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.6 PINB – Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB [7..0]								PINB
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

12.5.7 PORTC – Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	-	-	PORTC [5..0]						PORTC
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.8 DDRC – Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	-	-	DDC [5..0]						DDRC
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.9 PINC – Port C Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	-	-	PINC [5..0]						PINC
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

12.5.10 PORTD – Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD [7..0]								PORTD
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.11 DDRD – Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD [7..0]								DDRD
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.12 PIND – Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND [7..0]								PIND
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

12.5.13 PORTE – Port E Data Register

Bit	7	6	5	4	3	2	1	0	
0x0E (0x2E)	PORTE [7..0]								PORTE
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.14 DDRE – Port E Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0D (0x2D)	DDE [7..0]								DDRE
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

12.5.15 PINE – Port E Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x0C (0x2C)	PINE [7..0]								PINE
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

•

13. Timers

This section contains the detailed description of all 2timers embedded in the AT90SCR075_060. Finally, a global chapter describes the mechanism for prescaler reset used on all 2 registers.

13.1 8-bit Timer/Counter0 with PWM

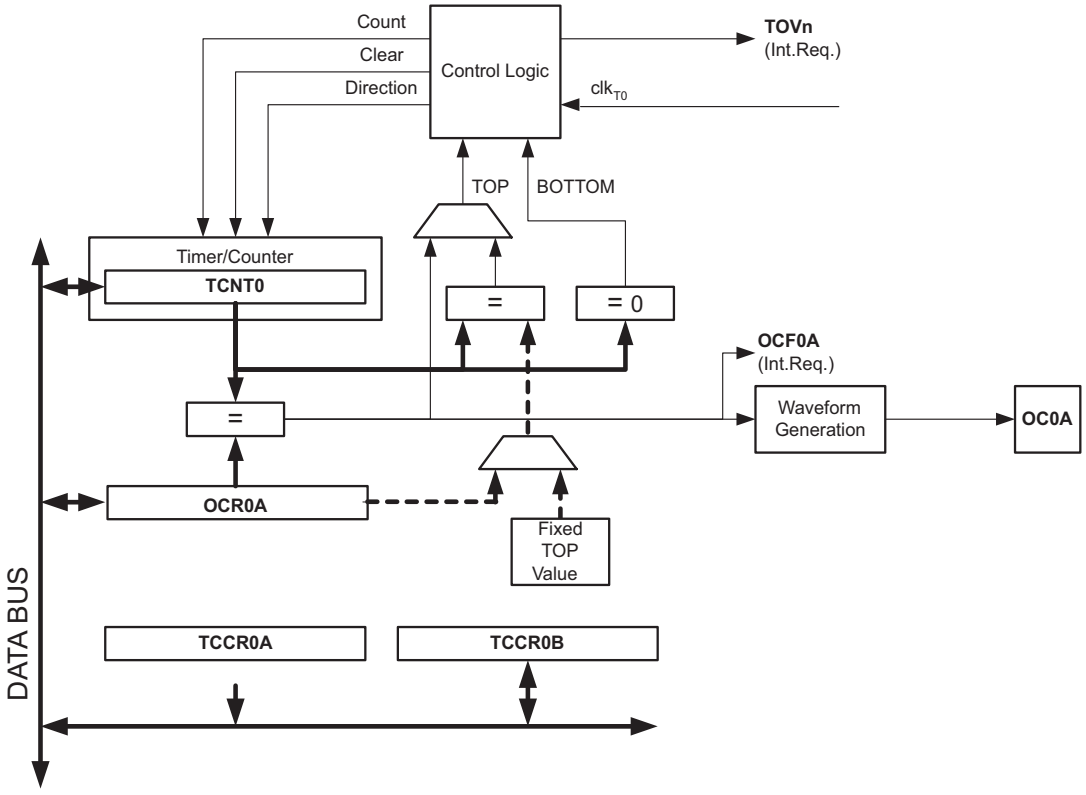
Timer/Counter0 is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and with PWM support. It allows accurate program execution timing (event management) and wave generation. The main features are:

- **Simple Buffered Output Compare Registers**
- **Clear Timer on Compare Match (Auto Reload)**
- **Two Independent Interrupt Sources (TOV0 and OCF0A)**
- **Glitch Free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**

13.1.1 Overview

A simplified block diagram of the 8-bit Timer/Counter0 is shown in Figure 13-1. For the actual placement of I/O pins, refer to “Pin List Configuration” on page 9. The device-specific I/O Register and bit locations are listed in the “8-bit Timer/Counter0 Register Description” on page 76”.

Figure 13-1. 8-bit Timer/Counter0 Block Diagram



13.1.1.1 Registers

The Timer/Counter (TCNT0) and Output Compare Register (OCR0A) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The Timer/Counter is clocked internally, via the prescaler. The Clock Select logic block controls which edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when internal clock source is not selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The Output Compare Register (OCR0A) is compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC0A and OC0B). See “Output Compare Unit” on page 67. for details. The Compare Match event will also set the Compare Flag (OCF0A) which can be used to generate an Output Compare interrupt request.

13.1.1.2 Definitions

Many register and bit references in this section are written in general form. A lower case “n” replaces the Timer/Counter number, in this case 0. However, when using the register or bit defines in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value and so on.

The definitions in Table 13-1 are also used extensively throughout the document.

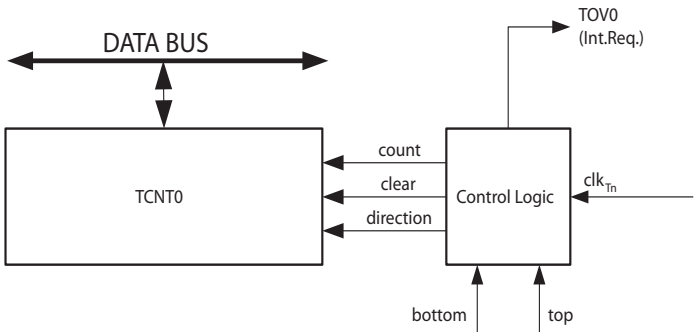
Table 13-1. Definitions

BOTTOM	The counter reaches the BOTTOM when it becomes 0x00.
MAX	The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

13.1.2 Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Figure 13-2 shows a block diagram of the counter and its surroundings.

Figure 13-2. Counter Unit Block Diagram



Signal description (internal signals):

count Increment or decrement TCNT0 by 1.

direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signifies that TCNT0 has reached maximum value.
bottom	Signifies that TCNT0 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). When no clock source is selected (CS02:0 = 0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

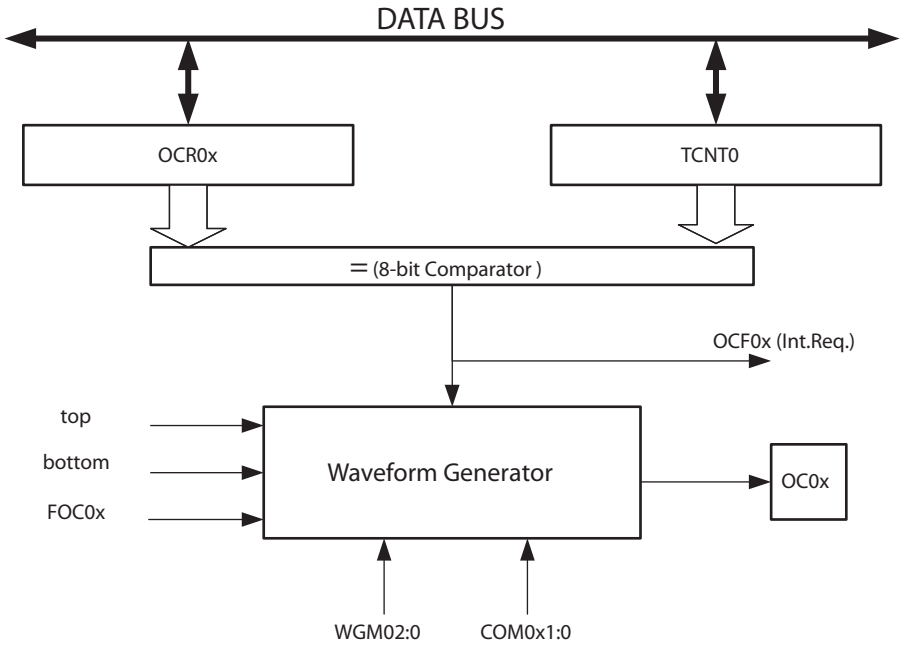
The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A). For more details about advanced counting sequences and waveform generation, see “Compare Match Output Unit” on page 69.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM01:0 bits. TOV0 can be used for generating a CPU interrupt.

13.1.3 Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A). Whenever TCNT0 equals OCR0A , the comparator signals a match. A match will set the Output Compare Flag (OCF0A) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM02:0 bits and Compare Output mode (COM0A1:0) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases that occur in some modes of operation when maximum and minimum values are reached (“Compare Match Output Unit” on page 69).Figure 13-3 shows a block diagram of the Output Compare unit.

Figure 13-3. Output Compare Unit, Block Diagram



The OCR0A Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR0A Compare Registers to either the top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR0A Register access may seem complex, but this is not case. When the double buffering is enabled, the CPU has access to the OCR0A Buffer Register, and if double buffering is disabled the CPU will access the OCR0A directly.

13.1.3.1 Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC0A) bit. Forcing Compare Match will not set the OCF0A Flag or reload/clear the timer, but the OC0A pin will be updated as if a real Compare Match had occurred (the COM0A1:0 bits settings define whether the OC0A pin is set, cleared or toggled).

13.1.3.2 Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any Compare Match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0A to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

13.1.3.3 Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all Compare Matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the Compare Match will be missed. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down-counting.

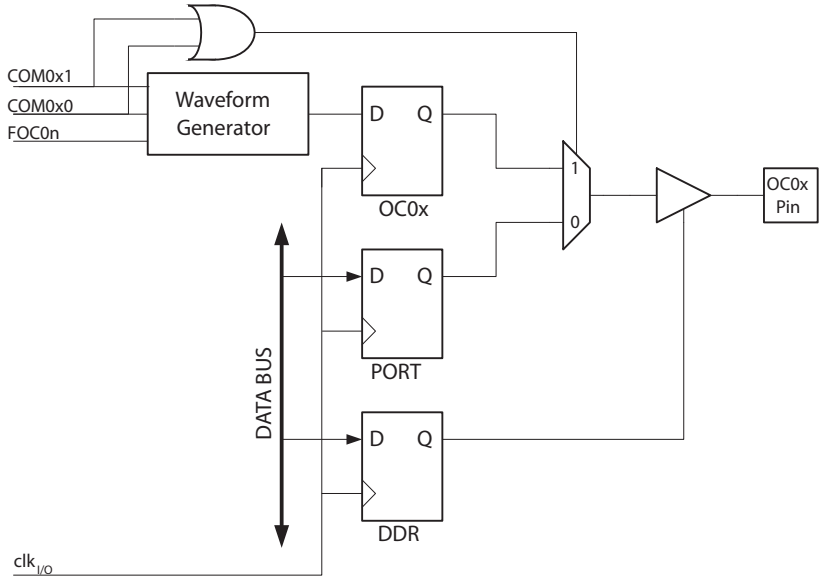
The setup of the OC0A should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0A value is to use the Force Output Compare (FOC0A) strobe bits in Normal mode. The OC0A Registers keep their values even when changing between Waveform Generation modes.

Changing the COM0A1:0 bits will take effect immediately.

13.1.4 Compare Match Output Unit

The Compare Output mode (COM0A1:0) bits have two functions. The Waveform Generator uses the COM0A1:0 bits for defining the Output Compare (OC0A) state at the next Compare Match. Also, the COM0A1:0 bits control the OC0A pin output source. Figure 13-4 shows a simplified schematic of the logic affected by the COM0A1:0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control Registers (DDR and PORT) that are affected by the COM0A1:0 bits are shown. When referring to the OC0A state, the reference is for the internal OC0A Register, not the OC0A pin. If a system reset occurs, the OC0A Register is reset to “0”.

Figure 13-4. Compare Match Output Unit, Schematic



The general I/O port function is overridden by the Output Compare (OC0A) from the Waveform Generator if either of the COM0A1:0 bits are set. However, the OC0A pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC0A pin (DDR_OC0A) must be set as an output before the OC0A value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows the initialization of the OC0A state before the output is enabled. Note that some COM0A1:0 bit settings are reserved for certain modes of operation. See “8-bit Timer/Counter0 Register Description” on page 76.

13.1.5 Compare Output Mode and Waveform Generation

The Waveform Generator uses the COM0A1:0 bits differently in Normal, CTC, and PWM modes. For all modes, setting the COM0A1:0 = 0 instructs the Waveform Generator that no action on the OC0A Register is to be performed on the next Compare Match. For compare output actions in the non-PWM modes refer to Table 13-2 on page 76. For fast PWM mode, refer to Table 13-3 on page 76, and for phase correct PWM refer to Table 13-4 on page 77.

A change of the COM0A1:0 bits state will have an effect at the first Compare Match after the bits are written. For non-PWM modes, the action can be forced to have an immediate effect by using the FOC0A strobe bits.

13.1.6 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM02:0) and the Compare Output mode (COM0A1:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0A1:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0A1:0 bits control whether the output should be set, cleared, or toggled at a Compare Match (See “Compare Match Output Unit” on page 69.).

For detailed timing information see “Fast PWM Mode” on page 71.

13.1.6.1 Normal Mode

The simplest mode of operation is the Normal mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

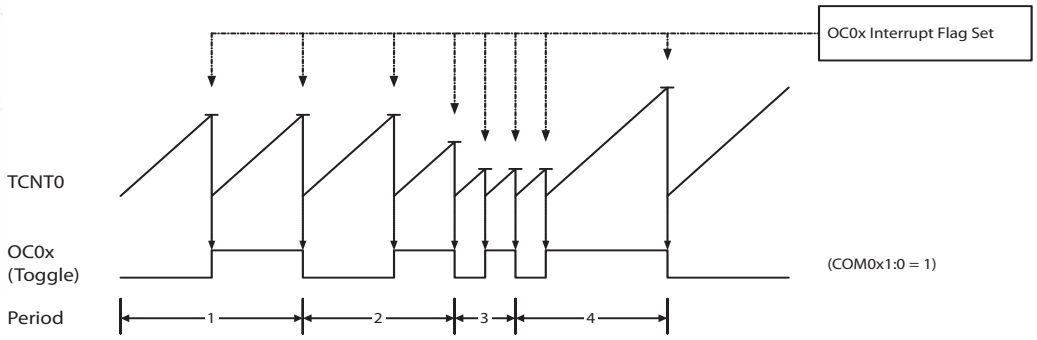
The Output Compare Unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

13.1.6.2 Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM01:0 = 2), the OCR0A Register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the Compare Match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 13-5. The counter value (TCNT0) increases until a Compare Match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 13-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by using the OCF0A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the Compare Match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at 0x00 before the Compare Match can occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each Compare Match by setting the Compare Output mode bits to toggle mode (COM0A1:0 = 1). The OC0A value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of f_{OC0} = $f_{clk\_I/O}/2$ when OCR0A is set to zero (0x00). The waveform frequency is defined by the following equation:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnx)}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the TOV0 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x00.

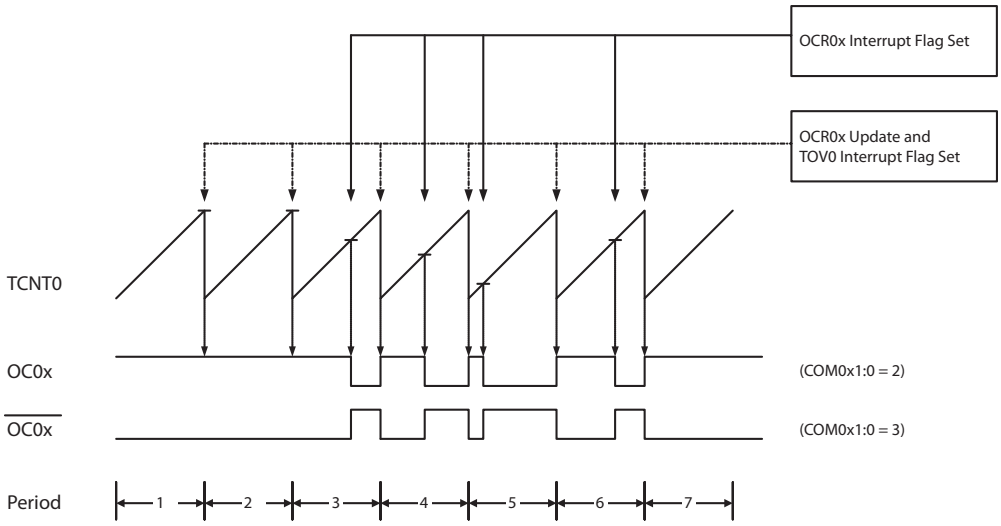
13.1.6.3 Fast PWM Mode

The fast Pulse Width Modulation or fast PWM mode (WGM02:0 = 3 or 7) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. TOP is defined as 0xFF when WGM2:0 = 3, and OCR0A when WGM2:0 = 7. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the Compare Match between TCNT0 and OCR0A, and set at BOTTOM. In inverting Compare Output mode, the output is set on Compare Match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 13-6. The TCNT0 value is in the timing diagram shown as a his-

togram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0A and TCNT0.

Figure 13-6. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pins. Setting the COM0A1:0 bits to two will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM0A1:0 to three: Setting the COM0A1:0 bits to one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. This option is not available for the OC0B pin (See Table 13-3 on page 76). The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0A Register at the Compare Match between OCR0A and TCNT0, and clearing (or setting) the OC0A Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

The *N* variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR0A is set equal to BOTTOM, the output will be a narrow spike for each MAX+1 timer clock cycle. Setting the OCR0A equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A1:0 bits.)

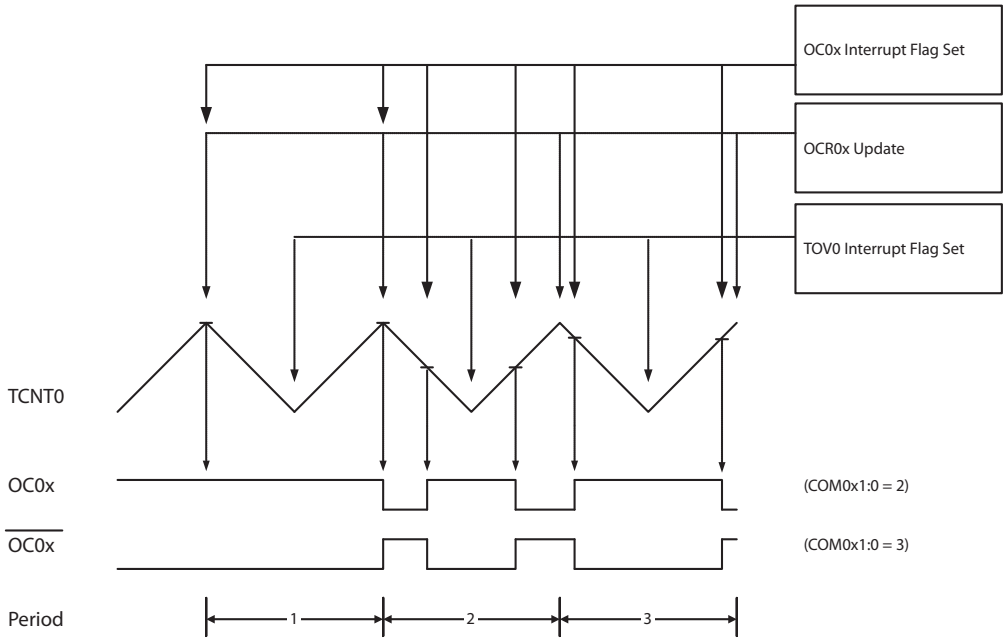
A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC0A to toggle its logical level on each Compare Match (COM0A1:0 = 1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is set to zero. This feature is similar to the OC0A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

13.1.6.4 *Phase Correct PWM Mode*

The phase correct PWM mode (WGM02:0 = 1 or 5) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-slope operation. The counter counts repeatedly from BOTTOM to TOP and then from TOP to BOTTOM. TOP is defined as 0xFF when WGM2:0 = 1, and OCR0A when WGM2:0 = 5. In non-inverting Compare Output mode, the Output Compare (OC0A) is cleared on the Compare Match between TCNT0 and OCR0A while upcounting, and set on the Compare Match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has lower maximum operation frequency than single slope operation. However, due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In phase correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 13-7. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent Compare Matches between OCR0x and TCNT0.

Figure 13-7. Phase Correct PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0A pin. Setting the COM0A1:0 bits to two will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM0A1:0 to three: Setting the COM0A0 bits to one allows the OC0A pin to toggle on Compare Matches if the WGM02 bit is set. The actual OC0A value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0A Register at the

Compare Match between OCR0A and TCNT0 when the counter increments, and setting (or clearing) the OC0A Register at Compare Match between OCR0A and TCNT0 when the counter decrements. The PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR0A is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in Figure 13-7 OC0A has a transition from high to low even though there is no Compare Match. The point of this transition is to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match.

- OCR0A changes its value from MAX, like in Figure 13-7. When the OCR0A value is MAX the OC0 pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OC0 value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts counting from a value higher than the one in OCR0A, and for that reason misses the Compare Match and hence the OC0 change that would have happened on the way up.

13.1.7 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set. Figure 13-8 contains timing data for basic Timer/Counter operation. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.

Figure 13-8. Timer/Counter Timing Diagram, no Prescaling

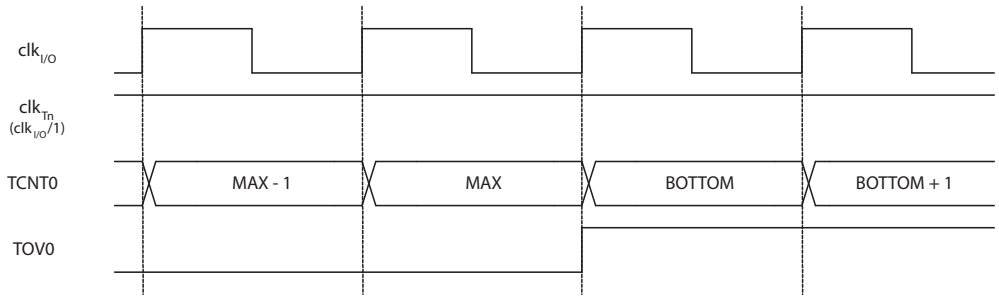


Figure 13-9 shows the same timing data, but with the prescaler enabled.

Figure 13-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\ I/O}/8$)

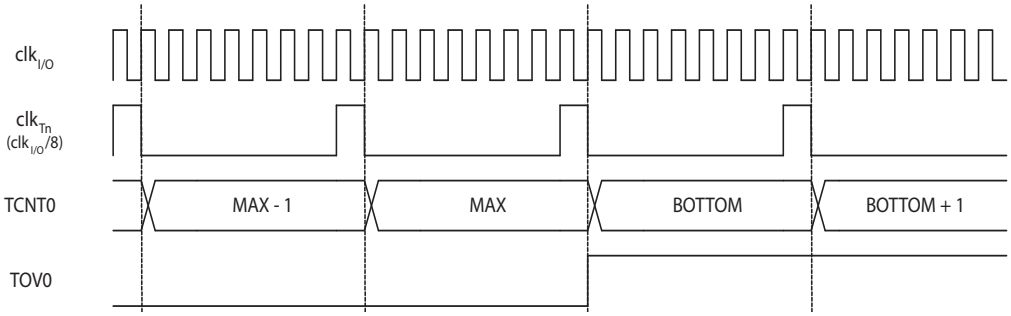


Figure 13-10 shows the setting of OCF0A in all modes except CTC mode and PWM mode, where OCR0A is TOP.

Figure 13-10. Timer/Counter Timing Diagram, Setting of OCF0A, with Prescaler ($f_{clk\ I/O}/8$)

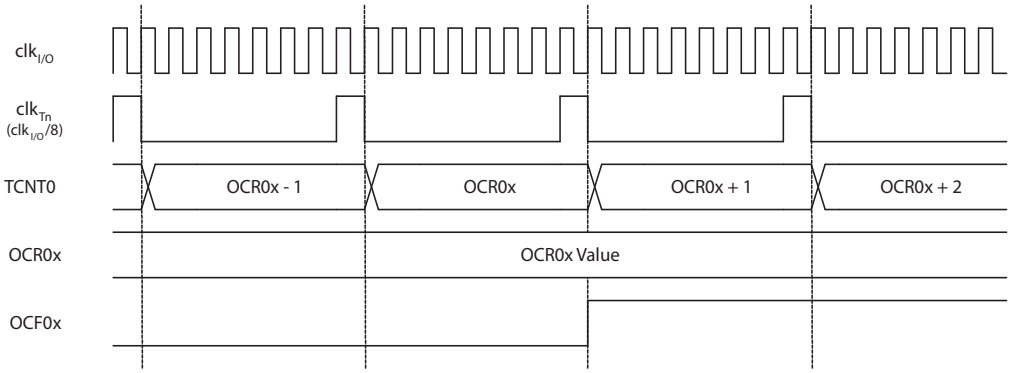
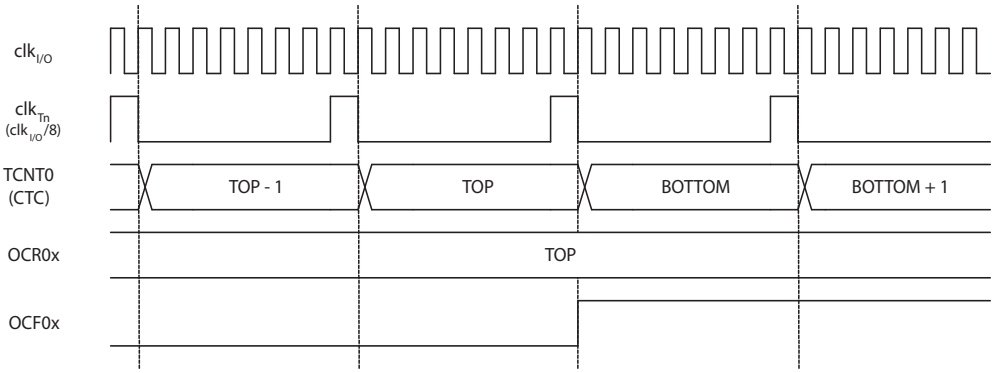


Figure 13-11 shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 13-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\ I/O}/8$)



13.1.8 8-bit Timer/Counter0 Register Description

13.1.8.1 TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	**COM0A1**	**COM0A0**	-	-	-	-	**WGM01**	**WGM00**	**TCCR0A**
Read/write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..6 – COM0A1..0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected. However, note that the Data Direction Register (DDR) bit corresponding to which the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A1:0 bits depends on the WGM02:0 bit setting. Table 13-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 13-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Table 13-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 13-3. Compare Output Mode, Fast PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at BOTTOM. See “Fast PWM Mode” on page 71 for more details.

Table 13-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 13-4. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note: 1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 73 for more details.

- **Bits 7..2 – Res: Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and will always read as zero.

- **Bits 1..0 – WGM01..0: Waveform Generation Mode**

When combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 13-5. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see "Compare Match Output Unit" on page 69).

Table 13-5. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	TOP	TOP

Note: 1. MAX = 0xFF, BOTTOM = 0x00

13.1.8.2 TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	**FOC0A**	**-**	**-**	**-**	**WGM02**	**CS02**	**CS01**	**CSO0**	**TCCR0B**
Read/write	W	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – FOC0A: Force Output Compare A**

The FOC0A bit is only active when the WGM bits specify a non-PWM mode.

However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

- **Bits 7..4 – Res: Reserved Bits**

These bits are reserved bits and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the “TCCR0A – Timer/Counter Control Register A” on page 76.

- **Bits 2..0 – CS02..0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 13-6. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	Reserved
1	1	1	Reserved

13.1.8.3 *TCNT0 – Timer/Counter Register*

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0 [7..0]								TCNT0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

13.1.8.4 *OCR0A – Output Compare Register A*

Bit	7	6	5	4	3	2	1	0	
0x27 (0x47)	OCR0A [7..0]								OCR0A
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

13.1.8.5 *TIMSK**0** – Timer/Counter Interrupt Mask Register*

Bit	7	6	5	4	3	2	1	0	
0x6E	-	-	-	-	-	-	OCIE0A	TOIE0	TIMSK0
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..2 – Res: Reserved Bits**

These bits are reserved bits and will always read as zero.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

13.1.8.6 *TIFR0 – Timer/Counter 0 Interrupt Flag Register*

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	-	-	-	-	-	-	**OCF0A**	**TOV0**	**TIFR0**
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..2 – Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and will always read as zero.

- **Bit 1 – OCF0A: Timer/Counter 0 Output Compare A Match Flag**

The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM01:0 bit setting. Refer to Table 13-5, “Waveform Generation Mode Bit Description” on page 77.

13.2 16-bit Timer/Counter1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), wave generation, and signal timing measurement. The main features are:

- **True 16-bit Design**
- **Clear Timer on Compare Match (Auto Reload)**
- **Two independent interrupt Sources (TOV1, OCF1A1)**

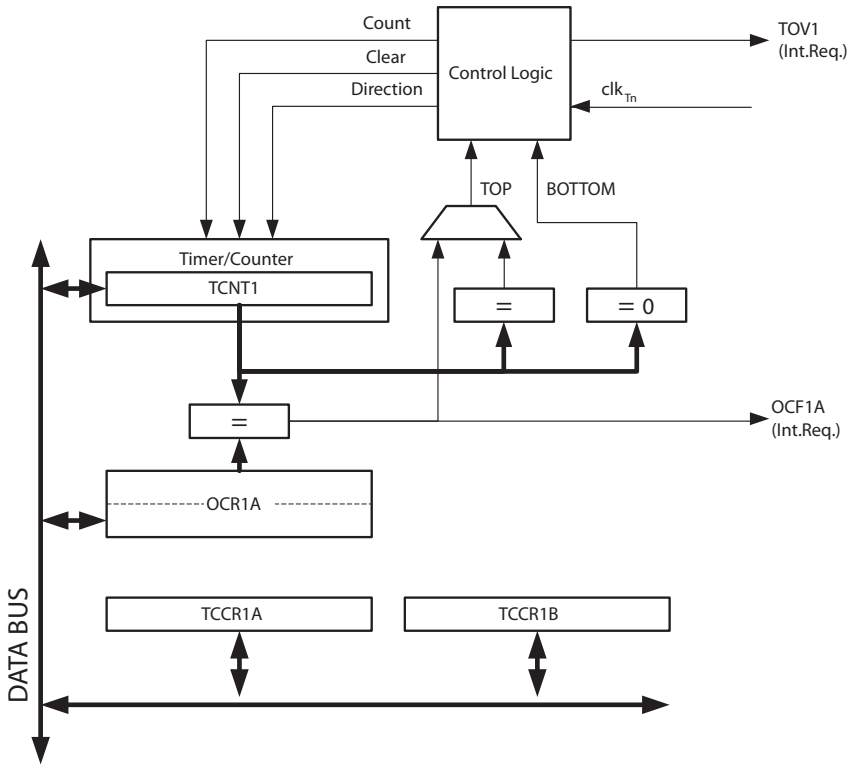
13.2.1 Overview

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 13-12. For the actual placement of I/O pins, refer to “Pin List Configuration” on page 9. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the “16-bit Timer/Counter Register Description” on page 87.

The PRTIM1 bit in “PRR0 – Power Reduction Register 0” on page 33 must be written to zero to enable Timer/Counter1 module.

Refer to “Pin List Configuration” on page 9 for Timer/Counter1 pin placement and description.

Figure 13-12. 16-bit Timer/Counter Block Diagram



13.2.1.1 *Registers*

The *Timer/Counter1* (TCNT1) and *Output Compare Registers* (OCR1A) are 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section “Accessing 16-bit Registers” on page 82. The *Timer/Counter Control Registers* (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the figure) signals are all visible in the *Timer Interrupt Flag*

Register (TIFR1). All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

The Timer/Counter1 is clocked internally, via the prescaler. The Clock Select logic block controls which edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter1 is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{Tn}).

The Output Compare Registers (OCR1A) is compared with the Timer/Counter value at all time. The compare match event will set the Compare Match Flag (OCF1A) which can be used to generate an Output Compare interrupt request.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register or by a set of fixed values.

13.2.1.2 Definitions

The following definitions are used extensively throughout the section:

Table 13-7. Definitions

BOTTOM	The counter reaches the *BOTTOM* when it becomes 0x0000.
MAX	The counter reaches its *MAX*imum when it becomes 0xFFFF (decimal 65535).
TOP	The counter reaches the *TOP* when it becomes equal to the highest value in the count sequence. The assignment is dependent of the mode of operation.

13.2.2 Accessing 16-bit Registers

The TCNT1 and OCR1A are 16-bit registers that can be accessed by the 8/16-bit RISC CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storage of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the low byte triggers the 16-bit read or write operation. When the low byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses uses the temporary register for the high byte. Reading the OCR1A 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit Timer Registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A Register. Note that when using “C”, the compiler handles the 16-bit access.

13.2.2.1 *Reusing the Temporary High Byte Register*

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

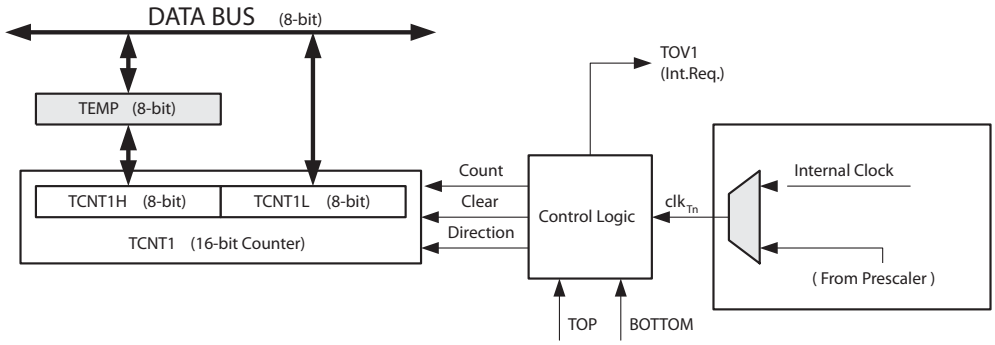
13.2.3 Timer/Counter Clock Sources

The Timer/Counter is clocked by an internal clock source. The clock prescaler selected by the Clock Select logic which is controlled by the *Clock Select* (CS12:0) bits located in the *Timer/Counter control Register B* (TCCR1B). For details on clock sources and prescaler, see "GTCCR – General Timer/Counter Control Register" on page 91.

13.2.4 Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 13-13 shows a block diagram of the counter and its surroundings.

Figure 13-13. Counter Unit Block Diagram



Signal description (internal signals):

Count	Increment or decrement TCNT1 by 1.
Direction	Select between increment and decrement.
Clear	Clear TCNT1 (set all bits to zero).
clk_{Tn}	Timer/Counter clock.
TOP	Signifies that TCNT1 has reached maximum value.
BOTTOM	Signifies that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: *Counter High* (TCNT1H) containing the upper eight bits of the counter, and *Counter Low* (TCNT1L) containing the lower eight bits. The TCNT1H Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1H I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1H value when the TCNT1L is read, and TCNT1H is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the 8-bit data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each *timer clock* (clk_{Tn}). The clk_{Tn} is generated from an external clock source, selected by the *Clock Select* bits (CS12:0). When no clock source is selected (CS12:0 = 0) the timer is stopped. However, the TCNT1 value can be accessed by the CPU, independent of whether clk_{Tn} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

The counting sequence is determined by the setting of the *Waveform Generation mode* bits (WGM13:0) located in the *Timer/Counter Control Registers* A and B (TCCR1A and TCCR1B). For more details about advanced counting sequences and waveform generation, see “Modes of Operation” on page 85.

The Timer/Counter Overflow Flag (TOV1) is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.

13.2.5 Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the *Output Compare Register* (OCR1A). If TCNT equals OCR1A the comparator signals a match. A match will set the *Output Compare Flag* (OCF1A) at the next timer clock cycle. If enabled (OCIE1A = 1), the Output Compare Flag generates an Output Compare interrupt. The OCF1A Flag is automatically cleared when the interrupt is executed. Alternatively the OCF1A Flag can be cleared by software by writing a logical one to its I/O bit location.

The content of the OCR1A (Buffer or Compare) Register is only changed by a write operation (the Timer/Counter does not update this register automatically as happens with the TCNT1 and ICR1 Register). It is a good practice to read the low byte first as when accessing other 16-bit registers.

Writing the OCR1A Registers must be done via a temporary Register since the compare of all 16 bits is done continuously. The high byte (OCR1AH) has to be written first. When the high byte I/O location is written by the CPU, the temporary Register will be updated by the value written. Then when the low byte (OCR1AL) is written to the lower eight bits, the high byte will be copied into the upper 8-bits of either the OCR1A buffer or OCR1A Compare Register in the same system clock cycle.

For more information of how to access the 16-bit registers refer to “Accessing 16-bit Registers” on page 82.

13.2.5.1 Compare Match Blocking by TCNT1 Write

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR1A to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

13.2.5.2 Using the Output Compare Unit

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT1 when using any of the Output Compare channel, independent of whether the Timer/Counter is running or not. If the value written to TCNT1 equals the OCR1A value, the compare match will be missed. The compare match for the TOP will be ignored and the counter will continue to 0xFFFF. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is downcounting.

13.2.6 Modes of Operation

The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the *Waveform Generation mode* (WGM12:0). For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 86.

13.2.6.1 Normal Mode

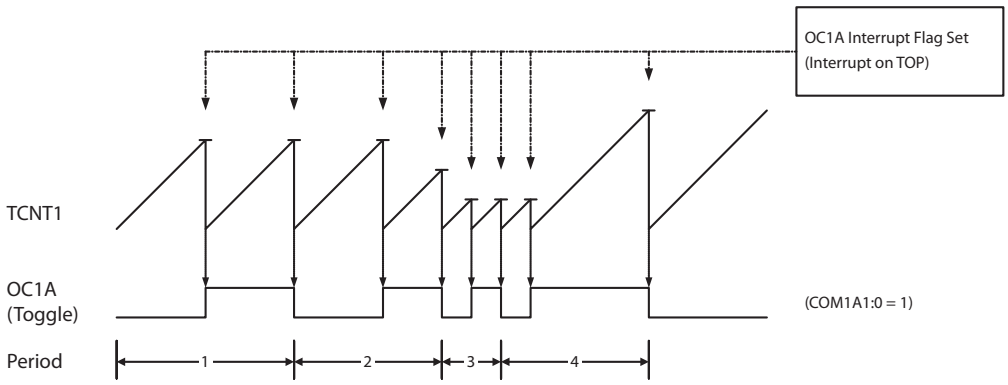
The simplest mode of operation is the *Normal mode* (WGM12:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the *Timer/Counter Overflow Flag* (TOV1) will be set in the same timer clock cycle as the TCNT1 becomes zero. The TOV1 Flag in this case behaves like a 17th bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

13.2.6.2 Clear Timer on Compare Match (CTC) Mode

In *Clear Timer on Compare* or CTC mode (WGM12:0 = 4), the OCR1A register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM12:0 = 4). The OCR1A or ICR1 define the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 13-14. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TCNT1) is cleared.

Figure 13-14. CTC Mode, Timing Diagram



An interrupt can be generated at each time the counter value reaches the TOP value by using the OCF1A Flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases this feature is not desirable.

As for the Normal mode of operation, the TOV1 Flag is set in the same timer clock cycle that the counter counts from MAX to 0x0000.

13.2.7 Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{Tn}) is therefore shown as a clock enable signal in the following figures. The figures include information on when Interrupt Flags are set, and when the OCR1A Register is updated with the OCR1A buffer value (only for modes utilizing double buffering). Figure 13-15 shows a timing diagram for the setting of OCF1A.

Figure 13-15. Timer/Counter Timing Diagram, Setting of OCF1A, no Prescaling

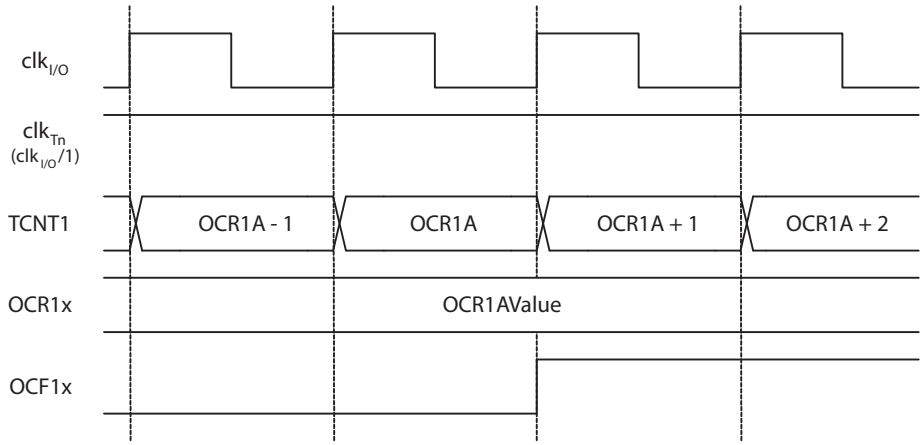


Figure 13-16 shows the same timing data, but with the prescaler enabled.

Figure 13-16. Timer/Counter Timing Diagram, Setting of OCF1A, with Prescaler ($f_{clk\ I/O}/8$)

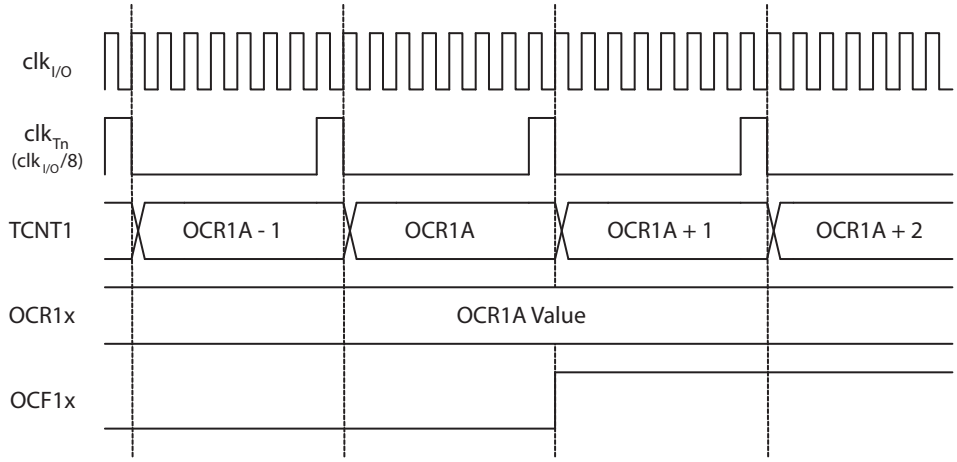


Figure 13-17 shows the count sequence close to TOP in various modes. When using phase and frequency correct PWM mode the OCR1A Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP-1 by BOTTOM+1 and so on. The same renaming applies for modes that set the TOV1 Flag at BOTTOM.

Figure 13-17. Timer/Counter Timing Diagram, no Prescaling

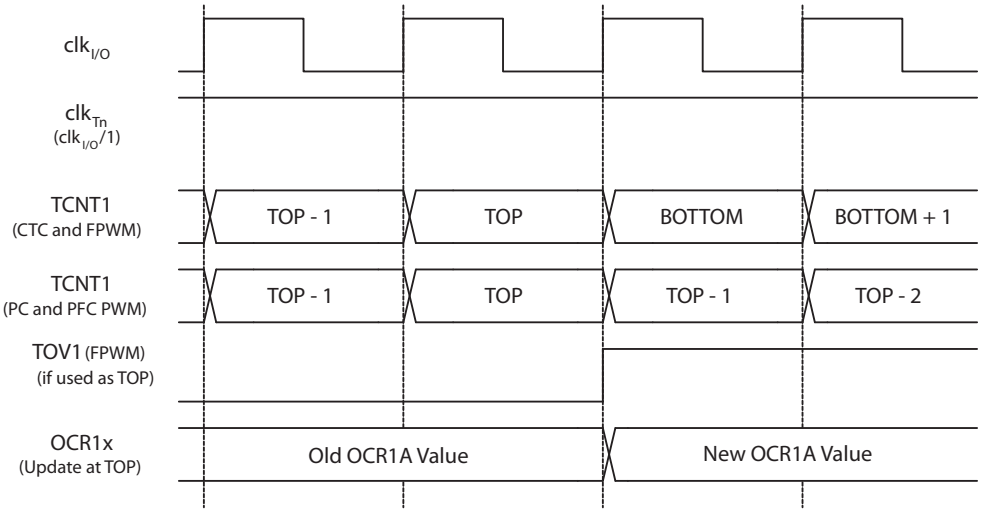
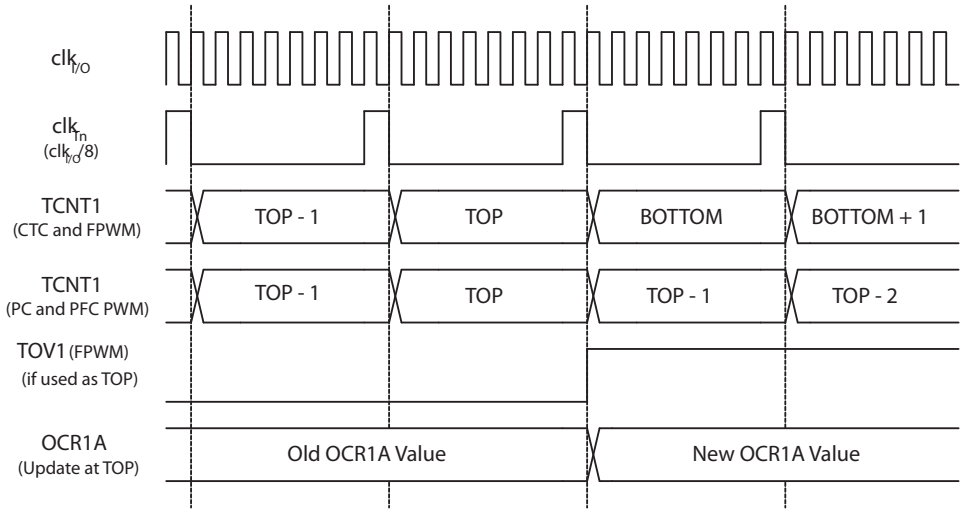


Figure 13-18 shows the same timing data, but with the prescaler enabled.

Figure 13-18. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\ I/O}/8$)



13.2.8 16-bit Timer/Counter Register Description

13.2.8.1 TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
$000080	-	-	-	-	-	-	**WGM11**	**WGM10**	**TCCR1A**
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 1..0 – WGM11..0: Waveform Generation Mode**

Combined with the WGM12:2 bits found in the TCCR1B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 13-8. Modes of operation supported by the Timer/Counter

unit are: Normal mode (counter), Clear Timer on Compare match (CTC) mode, and three types of Pulse Width Modulation (PWM) modes. (See “Modes of Operation” on page 85.).

Table 13-8. Waveform Generation Mode Bit Description

Mode	WGM12 (CTCn)	WGM11 (PWMn1)	WGM10 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	1	Reserved	-	-	-
2	0	1	0	Reserved	-	-	-
3	0	1	1	Reserved	-	-	-
4	1	0	0	CTC	OCR1A	Immediate	MAX
5	1	0	1	Reserved	-	-	-
6	1	1	0	Reserved	-	-	-
7	1	1	1	Reserved	-	-	-

13.2.8.2 TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
$000081	-	-	-	-	WGM12	CS12	CS11	CS10	TCCR1B
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..4 – Reserved Bit**

This bit is reserved for future use. For ensuring compatibility with future devices, this bit must be written to zero when TCCR1B is written.

- **Bit 3 – WGM12: Waveform Generation Mode**

See TCCR1A Register description.

- **Bit 2..0 – CS12..0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Figure 13-15 and Figure 13-16.

Table 13-9. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	Reserved
1	1	1	Reserved

13.2.8.3 TCNT1H and TCNT1L –Timer/Counter1

Bit	15	14	13	12	11	10	9	8	
$000085	TCNT[15..8]								TCNT1H
$000084	TCNT [7..0]								TCNT1L
Bit	7	6	5	4	3	2	1	0	
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00

The two *Timer/Counter* I/O locations (TCNT1H and TCNT1L, combined TCNT1) give direct access, both for read and for write operations, to the Timer/Counter unit 16-bit counter. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 82.

Modifying the counter (TCNT1) while the counter is running introduces a risk of missing a compare match between TCNT1 and one of the OCR1A Registers.

Writing to the TCNT1 Register blocks (removes) the compare match on the following timer clock for all compare units.

13.2.8.4 OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	15	14	13	12	11	10	9	8	
$000089	OCR1A[15..8]								OCR1AH
$000088	OCR1A [7..0]								OCR1AL
Bit	7	6	5	4	3	2	1	0	
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00

The Output Compare Registers contain a 16-bit value that is continuously compared with the counter value (TCNT1). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC1A pin.

The Output Compare Registers are 16-bit in size. To ensure that both the high and low bytes are written simultaneously when the CPU writes to these registers, the access is performed using an 8-bit temporary High Byte Register (TEMP). This temporary register is shared by all the other 16-bit registers. See “Accessing 16-bit Registers” on page 82.

13.2.8.5 *TIMSK1 – Timer/Counter1 Interrupt Mask Register*

Bit	7	6	5	4	3	2	1	0	
$00006F	-	-	-	-	-	-	**OCIE1A**	**TOIE1**	**TIMSK1**
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..2 – Res: Reserved Bits**

These bits are unused bits in the AT90SCR075_060, and will always read as zero.

- **Bit 1 – OCIE1A: Timer/Counter1, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 47.) is executed when the OCF1A Flag, located in TIFR1, is set.

- **Bit 0 – TOIE1: Timer/Counter1, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Counter1 Overflow interrupt is enabled. The corresponding Interrupt Vector (See “Watchdog Timer” on page 40.) is executed when the TOV1 Flag, located in TIFR1, is set.

13.2.8.6 *TIFR1 – Timer/Counter1 Interrupt Flag Register*

Bit	7	6	5	4	3	2	1	0	
$000016	-	-	-	-	-	-	**OCF1A**	**TOV1**	**TIFR1**
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..2 – Res: Reserved Bits**

These bits are unused bits in the AT90SCR075_060, and will always read as zero.

- **Bit 1 – OCF1A: Timer/Counter1, Output Compare A Match Flag**

This flag is set in the timer clock cycle after the counter (TCNT1) value matches the Output Compare Register A (OCR1A).

Note that a Forced Output Compare (FOC1A) strobe will not set the OCF1A Flag.

OCF1A is automatically cleared when the Output Compare Match A Interrupt Vector is executed. Alternatively, OCF1A can be cleared by writing a logic one to its bit location.

- **Bit 0 – TOV1: Timer/Counter1, Overflow Flag**

The setting of this flag is dependent of the WGM13:0 bits setting. In Normal and CTC modes, the TOV1 Flag is set when the timer overflows. Refer to Table 13-8 on page 88 for the TOV1 Flag behavior when using another WGM13:0 bit setting.

TOV1 is automatically cleared when the Timer/Counter1 Overflow Interrupt Vector is executed. Alternatively, TOV1 can be cleared by writing a logic one to its bit location.

13.2.9 GTCCR – General Timer/Counter Control Register

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	**TSM**	-	-	-	-	-	-	**PSRSYNC**	**GTCCR**
Read/write	R/W	R	R	R	R	R	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – TSM : Timer/Counter Synchronization Mode**

Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRSYNC bit is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRSYNC bit is cleared by hardware, and the Timer/Counters start counting simultaneously.

- **Bit 0 – PSRSYNC : Prescaler Reset Timer/Counter0, Timer/Counter1**

When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

14. USB Device Interface

This section contains an overview of the USB module and the description of the registers design to interface this USB module. It also details the USB interrupt, the Suspend and Resume modes, the double buffering, the mode detection and the attachment procedure. This section assumes the reader of this document is comfortable with the USB Specifications V2.0, available on the *www.usb.org* website. He may also refer to the USB register summary, located at the end of this document to have a complete overview of the registers available. See “Register Summary” on page 239.

14.1 Features

- **USB 2.0 FullSpeed compliant**
- **Data transfer rates up to 12Mbit/s**
- **8 Programmable Endpoints bi-directionnals**
  - **Endpoint 0 for control: 64bytes**
  - **Endpoints 1 to 3 support double buffers of 64bytes each**
  - **Endpoints 4 to 7: 8 bytes each**
- **Suspend/Resume Interrupts**
- **Resume Wake Up Capabilities**
- **Automatic NACK if USB not ready to transmit/receive**
- **Specific USBDMA connected for fast and easy copy from Endpoint to RAM**

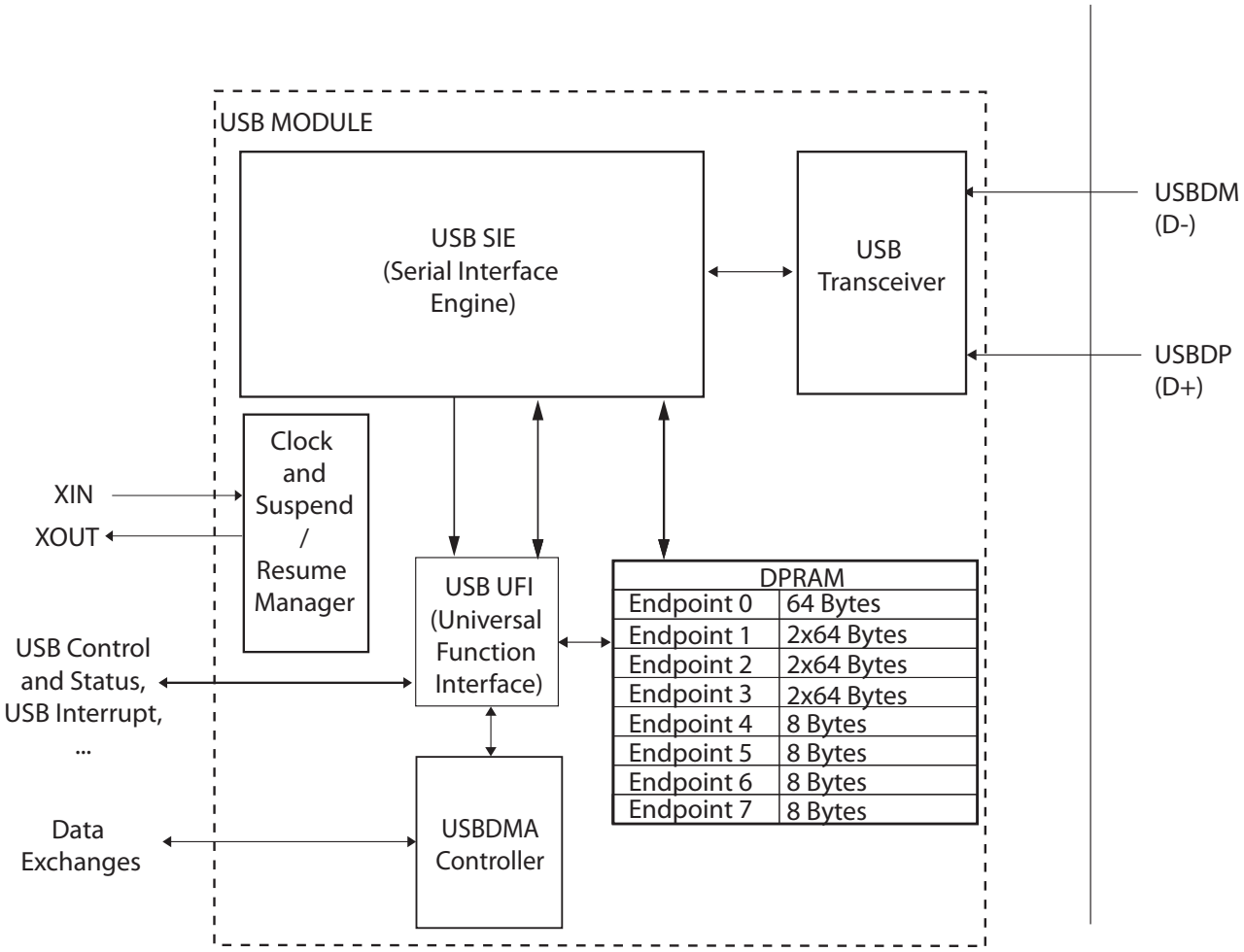
14.2 Overview

The following diagram represents the USB module that contains the necessary logic to communicate via a Full-Speed USB port.

In Register configuration, you may see ‘X’ suffix at the end of some register. This value is to be changed into 0, 1, 2, 3, 4, 5, 6 and 7 according to the endpoint targeted.

The USBENUM is designed to target choose the endpoint to target.

Figure 14-1. USB Module Diagram



The USB module of the AT90SCR075_060 is made up of the following elements:

- **The USB Transceiver**: it is the electrical and physical interface between the USB Bus and the internal logic of the AT90SCR075_060.
- **The USB Serial Interface Engine**: this logical part manages the NRZI coding/decoding, the bit stuffing/unstuffing, the CRC generation/checking and the serial-parallel data conversion as requested by the USB specifications.
- **The Universal Function Interface**: this block is the interface between the data flow and the internal Dual Port RAM of the USB module that contains the endpoints.
- **The DPRAM block**: the Dual Port RAM is an internal block of the USB module. It contains the seven available endpoints and can be accessed either by the SIE or the UFI. It is intended to be used as a buffer between the USB bus and the internal data bus of the AT90SCR075_060.
- **The USBDMA Controller**: the Direct Memory Access logical module allows fast data transfers from the DPRAM to the RAM of the AT90SCR075_060. See “USBDMA Controller” on page 108.

The USB interface of the AT90SCR075_060 is a module supporting one USB device address with eight configurable endpoints, managed by an embedded firmware running on the 8/16-bit RISC CPU. This firmware is responsible for handling the enumeration (particularly the SETUP packets management), executing the Suspend and Resume mode entries, filling and emptying the endpoints through the USBDMA Controller, sending STALL packets.

The device is characterized by a full-speed (12Mb/s) bus-powered interface supporting Suspend and Resume modes and fully compliant with the USB V2.0 specifications.

The endpoints 1, 2 and 3 have a double-buffering capability (double DPRAM size). This feature is particularly suitable for Bulk data transfers and is totally managed by the hardware. See “Double Buffering” on page 98.

In order to communicate via the USB interface, the chip operates with at 48MHz clock.

The chip internally de-activates and activates its oscillator when necessary in order to be compliant with the power consumption of the Suspend mode.

14.3 Endpoints Description

The USB module contains eight endpoints. They are fully configurable through their corresponding register USBFCEX (for further details, please refer to “USBFCEX - USB Function Control Registers for Endpoint X” on page 105).

The table below indicates the size (in Bytes) of each endpoint and also shows possible configurations for the USB device.

Table 14-1. Endpoints Description

Endpoint Number	Size in Bytes	Double- Buffering Capability	Recommended Data Transfer Type
EP0	64	NO	CONTROL
EP1	2x64	YES	BULK(1)
EP2	2x64	YES	BULK (1)
EP3	2x64	YES	BULK (1)
EP4	8	NO	CONTROL, BULK, INTERRUPT(2)
EP5	8	NO	CONTROL, BULK, INTERRUPT(2)
EP6	8	NO	CONTROL, BULK, INTERRUPT(2)
EP7	8	NO	CONTROL, BULK, INTERRUPT(2)

Notes: 1. Can also be INTERRUPT or ISOCHRONOUS, not CONTROL
2. Can also be ISOCHRONOUS

An endpoint featuring the double-buffering capability is allocated on two banks of DPRAM, each bank equal to the endpoint size. Endpoints 1, 2 and 3 support these double buffer of 64 Bytes. See “Double Buffering” on page 98.

The hardware is responsible for handling the internal data toggle bit for each endpoint. This mechanism guarantees data sequence synchronisation between data transmitter and receiver across multiple transactions. Synchronisation is achieved via use of the DATA0 and DATA1

PIDs and separate data toggle sequence bits for the data transmitter and receiver. Receiver sequence bits toggle only when the receiver is able to accept data and receives an error-free data packet with the correct data PID. Transmitter sequence bits toggle only when the data transmitter receives a valid ACK handshake. Data toggle synchronization is not supported for isochronous transfers.

14.4 Attachment Procedure

This procedure must be applied in order to connect the pull-up between the USB signal D+ and USBReg, thus identifying a Full-Speed USB device.

The USB hardware module integrates the attachment pull-up resistors connected between the USB D+ differential data line and the internal 3.3 V USB regulator (USBReg).

The serial resistors required by USB certification are not embedded in the USB hardware module. They must be added externally.

Even after reset, the pull-up is not connected. This operation must be controlled by the software in order to attach the device.

We can imagine two integration scenarios for AT90SCR075_060:

- If the AT90SCR075_060 is standalone USB device Smart Card reader, then, connecting the USB cable into a computer, for instance, will power up the AT90SCR075_060. After chip initialization, PLL running, the USB module can be enabled using USBCR.USBE bit, set attachement pull-up can also be set to make the host detect a full-speed peripheral. The host will then reset the communication, and the FEURI interruption will trigger. A enumeration procedure can finally go ahead naturally.
- If the AT90SCR075_060 can be controlled by different hosts, for example by UART and USB, and if the host detection must be done dynamically, then a specific detection must be initiated. After chip power up, communication modules initialisation and PLL activation, the interruption on UART and USB (in the example) must be activated. The attachement pull-up must be enabled. Then, the first interruption (USBPI.FEURI or UART reception) will indicate the host communication mode. Please note that in this case, the D- line may be in High-Z state, making the USB module consumpts until the USB module is disabled or communication is runnning. To prevent consumption issue, a resistor or 1MegaOhm should be place as pull-up on USB D- line.

14.5 USB Interrupts

The USB interrupt sources are split into two main families:

- The USB Protocol Interrupts (Falling Edge on USB Reset, Start Of Frame, Resume, Suspend). They are included into the USBPI register (see “USBPI - USB Protocol Interrupt register” on page 101 for further details).
- The USB Endpoint Interrupt (Endpoint0, 1, 2, 3, 4, 5, 6 and 7). They are included into the USBEI register (see “USBEI - USB Endpoint Interrupt Register” on page 102 for further details). This register only indicates which endpoint of the eight holds the source of the

interruption. The source of the interruption can then be retrieved by checking the register USBCSEX corresponding to the endpoint (see “USBCSEX - USB Control And Status Register for Endpoint X” on page 103 for further details). These sources are Stall sent, Setup packet received, Data received, Data sent events.

The AT90SCR075_060 has two interrupt vectors for both families. These vectors, called USB Endpoint and USB Protocol are only valid if the chip operates with the USB interface and are respectively located at addresses $003E and $0040.

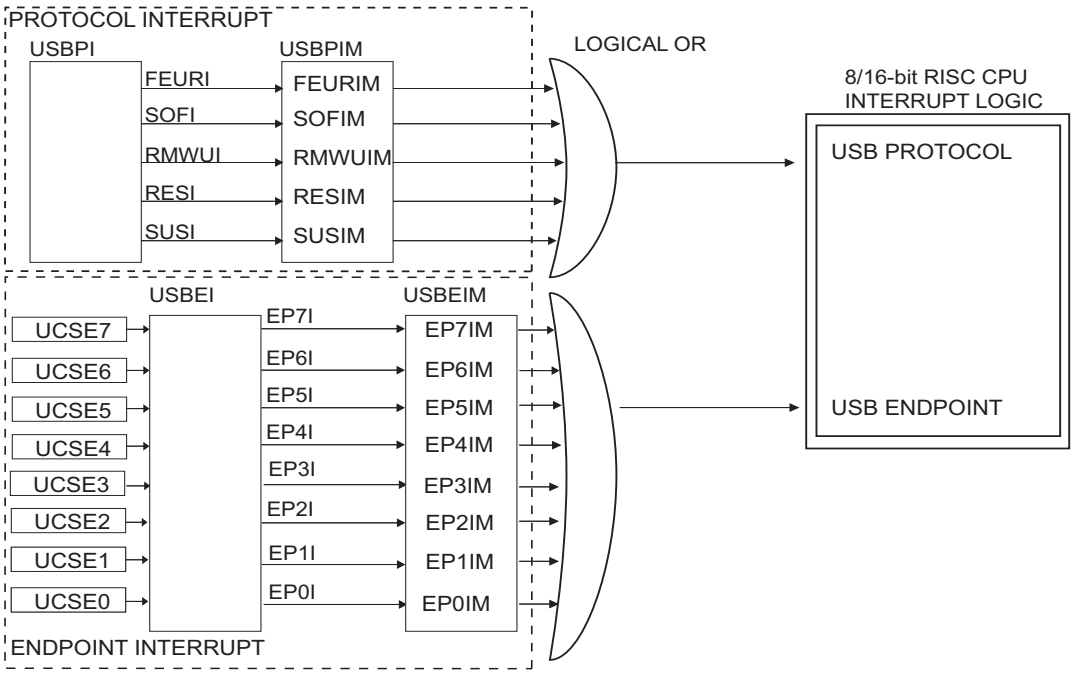
Every source can be enabled/disabled through bits of USBEIM or USBPIM registers.

Note

A USB interrupt is triggered assuming bits SREG.I is set (one) in order to enable the interrupt.

The figure below shows the implications between the registers, the USB interrupt sources and the 8/16-bit RISC CPU interrupt logic. For further details about the 8/16-bit RISC CPU Interrupt Logic please refer to Section 10. “Interrupts” on page 47.

Figure 14-2. USB Interrupt Hierarchy



14.6 Suspend and Resume Modes

In order to be compliant with the USB specifications V2.0, a device has to support the low-power consumption state called Suspend and its associated resumption activity called Resume.

The AT90SCR075_060 enters the Suspend mode only when requested by the host through bus inactivity for at least 3ms.

The USB module is able to detect this request and automatically sets the bit USBPI.SUSI. See “USBPI - USB Protocol Interrupt register” on page 101. This event generates a USB interruption if the Suspend Interrupt source is not masked and if the USB interrupt is enabled.

Note

In order to support the power consumption threshold fixed by the USB specifications, the oscillator is switched off by hardware. The POWER-DOWN mode must be entered by setting appropriate configuration in SMCR and followed by a SLEEP instruction.

Once Suspend mode has been entered, the USB module is able to detect a Resume request. This event immediately and automatically makes the chip come out the POWER-DOWN mode and sets (one) bit USBPI.RESI. The fact that bit USBPI.RESI is set (one) triggers a new USB interrupt if the Resume Interrupt source is not masked and if the USB interrupt is enabled. The application software shall then clear the interrupt source in its interrupt service routine.

Note

When a Resume signal has been detected the USB module does not automatically restarts the internal oscillator. See “Important note about: Entering and Leaving low consumption modes” on page 35.

14.6.1 Remote Wake-up

The USB Module is also able to generate a remote wake-up signal (K signal) to raise the communication with the host. To enable this feature, the USBGS.RMWUE bit must be set.

The wake-up signal start to be generated by setting the bit USBCR.URMWU bit.

As soon as this bit is set, a protocol interrupt is generated, if USBPIM.RMWUIM bit is set. through the raise of USBPI.RMWUI flag.

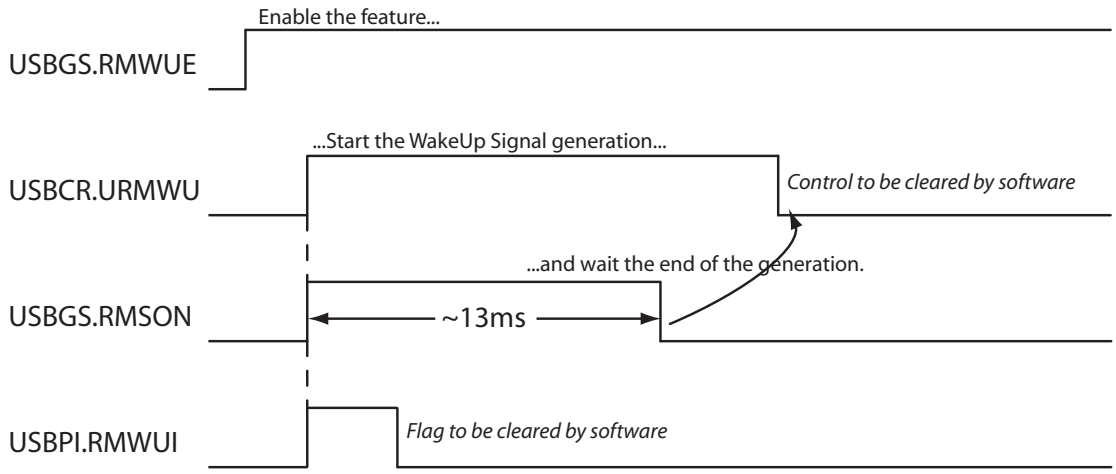
The remote wake-up signal will be automatically stopped after ~13ms generation. As soon as the signal stops, the USBGS.RSMON bit is cleared.

Note

When the USBGS.RSMON bit is cleared, do not forget to clear USBCR.URMWU bit, for next Remote Wake Up sequence.

Below is a scheme showing all the signal for Remote Wake Up management.

Figure 14-3. Remote Wake Up State Sequence Description



14.7 Double Buffering

This special feature available on the endpoints 1, 2 and 3 allows the user to save time during USB transfers.

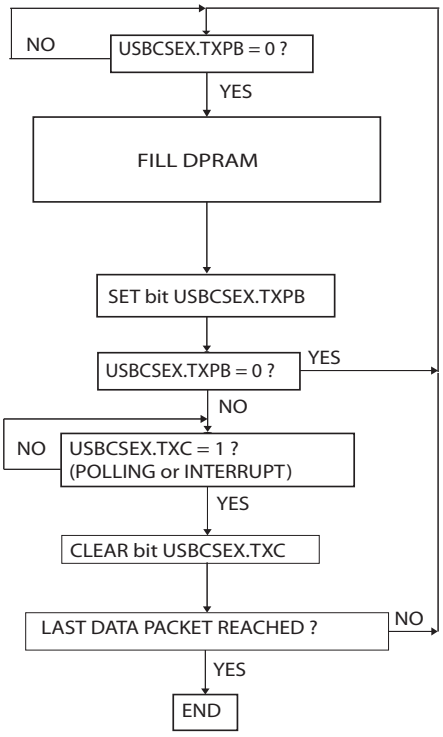
It is specially recommended for BULK data transfers but is also suitable for ISOCHRONOUS or INTERRUPT. CONTROL is not possible on these double buffered endpoints.

The double buffering uses two different data banks of the endpoint size. Its management remains transparent for the user. The advantages of the double buffering are the followings:

- When receiving data from the host, the software can process the data received in one of the two banks while the second is being filled. The endpoints that do not have the double buffering capability must wait for their DPRAM to be emptied by the software before being ready to receive new data.
- When sending data to the host, the software can fill the free bank while the USB module is sending data from the other one. The endpoints that do not have this feature must wait for the previous data to be sent before being allowed to fill again the single bank.

The two following figures provide code algorithm for sending data to the host and receiving data from the host (respectively IN and OUT tokens). They are also valid for endpoints operating normally (without the double buffering capability).

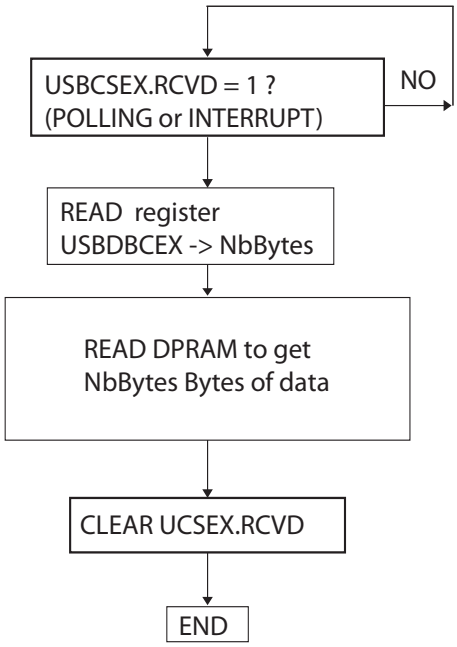
Figure 14-4. Sending Data Packets (Single Buffering)



Note

The figure is correct if several data packets have to be sent. If only one packet is to be sent the testing of the bit USBCSE.TXPB must be ignored.

Figure 14-5. Receiving Data Packets



14.8 USB Device Registers Description

14.8.1 USBCR - USB Control Register

Bit	7	6	5	4	3	2	1	0	
$0000E0	**URMWU**	**-**	**UPUC**	**-**	**-**	**-**	**USBE**	**-**	**USBCR**
Read/write	R/W	R	R/W	R	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - URMWU : USB Remote Wake up bit**

When in suspend mode, setting this bit to one generates a wake-up command to the host, according to USB2.0 specifications.

This bit must be cleared by software when USBGS.RSMON is cleared. See “Remote Wake-up” on page 97.

- **Bit 6 - Res : Reserved Bit**

This bit is reserved bits in the AT90SCR075_060 and is always read as zero.

- **Bit 5 - UPUC : USB Pull-Up Connection Bit**

This bit is set (one) and cleared (zero) by software.

It directly acts on the connection of the USB pull-up attachment resistors between USB bus signal D+ and VREG.

If set (one) the pull-up is connected. If cleared (zero) the pull-up is not connected.

- **Bit 4..2 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bit 1 - USBE : USB Enable Bit**

You must clear this bit before using it, even if the USB module has not been used previously.

Set this bit to enable the USB controller.

Clear this bit to disable and reset the USB controller, to disable the USB transceiver and to disable the USB controller clock inputs.

Clearing the USBE bit will freeze the USB macro the same way as PRR1.PRUSB. Anyway, these two bits are independant, and for the USB macro to run, USBE must be set and PRUSB must be cleared. See “PRR1 – Power Reduction Register 1” on page 34.

- **Bit 0 - Res : Reserved Bit**

This bit is reserved and will always be read as zero.

14.8.2 USBPI - USB Protocol Interrupt register

The following interrupt sources are all sources for the USB Protocol interruption

Bit	7	6	5	4	3	2	1	0	
$0000E1	-	-	-	FEURI	SOFI	RMWUI	RESI	SUSI	USBPI
Read/write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..5 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bit 4 - FEURI : Falling Edge on USB Reset Interrupt Bit**

Set (one) by hardware when a falling edge on the USB Reset has occurred. This bit indicates the end of the USB bus Reset signaling.

This interruption is not maskable at the USBPIM register level (see “USBPIM - USB Protocol Interrupt Mask Register” on page 101).

Cleared (zero) by software.

- **Bit 3 - SOFI : Start Of Frame Interrupt Bit**

Set (one) by hardware when a Start Of Frame PID has been detected on the USB bus.

Cleared (zero) by software.

- **Bit 2 - RMWUI :Remote WakeUp Interrupt Bit**

Set (one) byt hardware when the USBCR.URMWU bit is set.

Cleared (zero) by software.

- **Bit 1 - RESI : Resume Interrupt Bit**

Set (one) by hardware when a USB Resume signal has been detected on the USB bus.

Cleared (zero) by software.

- **Bit 0 - SUSI : Suspend Interrupt Bit**

Set (one) by hardware when a USB Suspend signal has been detected on the USB bus.

Cleared (zero) by software.

These bits can also be set (one) by software.

14.8.3 USBPIM - USB Protocol Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
$0000E2	-	-	-	-	SOFIM	RMWUIM	RESIM	SUSIM	USBPIM
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	1	0	0x02

See “USBPI - USB Protocol Interrupt register” on page 101.

- **Bits 7..4 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bit 3 - SOFIM : Start Of Frame Interrupt Mask Bit**

When SOFIM is set (one) the Start Of Frame interrupt is enabled.

When SOFIM is cleared (zero) the Start Of Frame interrupt is masked.

- **Bit 2 - RMWUIM : Remote Wake-Up Interrupt Mask Bit**

When RMWUIM is set (one), the Remote WakeUp interrupt is enabled.

When RMWUIM is cleared (zero), the Remote WakeUp interrupt is disabled.

- **Bit 1 - RESIM : Resume Interrupt Mask Bit**

When RESIM is set (one) the Resume interrupt is enabled.

When RESIM is cleared (zero) the Resume interrupt is masked.

- **Bit 0 - SUSIM : Suspend Interrupt Mask Bit**

When SUSIM is set (one) the Suspend interrupt is enabled. When SUSIM is cleared (zero) the Suspend interrupt is masked.

14.8.4 USBEI - USB Endpoint Interrupt Register

The following interrupt sources are all sources for the USB Endpoint interruption.

Bit	7	6	5	4	3	2	1	0	
$0000E3	**EP7I**	**EP6I**	**EP5I**	**EP4I**	**EP3I**	**EP2I**	**EP1I**	**EP0I**	**USBEI**
Read/write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit n - EPnI : Endpoint n Interrupt Bit (n=0..7)**

This bit is set (one) by hardware when an endpoint interrupt condition has occurred on the endpoint n.

This bit is cleared (zero) by hardware when the endpoint interrupt source has been cleared (zero) by the software (see “USBCSEX - USB Control And Status Register for Endpoint X” on page 103).

The endpoint interrupt conditions are listed below and further detailed in “USBCSEX - USB Control And Status Register for Endpoint X” on page 103. They are the same for the seven endpoints ($0 \leq n \leq 7$):

- USBCSEn.TXC bit is set in Bulk In, Interrupt In or Control mode.
- USBCSEn.RCVD bit is set in Bulk Out, Interrupt Out, Isochronous Out or Control mode.
- USBCSEn.RXSETUP is set in Control mode.

USBCSEn.STSENT bit is set in Bulk, Interrupt or Control mode.

14.8.5 USBEIM - USB Endpoint Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
$0000E4	EP7IM	EP6IM	EP5IM	EP4IM	EP3IM	EP2IM	EP1IM	EP0IM	USBEIM
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

See “USBEI - USB Endpoint Interrupt Register” on page 102.

- **Bit n - EPnIM : Endpoint n Interrupt Mask Bit (n=0..7)**

When EPnIM is set (one) the Endpoint n interrupt is enabled.

When EPnIM is cleared (zero) the Endpoint n interrupt is masked.

14.8.6 USBENUM - USB Endpoint Number Register

Bit	7	6	5	4	3	2	1	0	
$0000CA	-	-	-	-	-	ENUM [2..0]			USBENUM
Read/write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..3 - Res : Reserved Bits**

These bits are reserved and are always read as zero.

- **Bit 2..0 - ENUM2..0 : Endpoint Number**

Use this register to select an endpoint. The USB Device registers ended by a X correspond then to this number.

14.8.7 USBCSEX - USB Control And Status Register for Endpoint X

Set USBENUM register to point to the relevant Endpoint before using the USBCEX register.

Bit	7	6	5	4	3	2	1	0	
$0000CB	-	IERR	FSTALL	TXPB	STSENT	RXSETUP	RCVD	TXC	USBCSEX
Read/write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - Res : Reserved Bit**

This bit is a reserved bit in the AT90SCR075_060 and is always read as zero.

- **Bit 6 - IERR : Isochronous Error Bit**

This bit is a status bit.

It is set (one) by hardware when the CRC of a packet received in an Isochronous transfer is incorrect or corrupted.

If the CRC is correct the bit is cleared (zero) by hardware.

This bit is not a source for the Endpoint interrupt.

- **Bit 5 - FSTALL : Force Stall Bit**

This bit is a control bit.

It is set (one) and cleared (zero) by software.

If it is set (one) the corresponding endpoint is stalled and the host immediately receives a STALL as response to the IN or OUT tokens sent to this endpoint (see procedure on next page).

If it is cleared (zero) the corresponding endpoint is not stalled.

This bit is not a source for the Endpoint interrupt.

- **Bit 4 - TXPB : Tx Packet Busy Bit**

This bit is a control bit.

It is set (one) by software and cleared (zero) by hardware.

If it is cleared (zero), the DPRAM bank corresponding to the endpoint is empty. It can then be filled with data to be sent.

If it is set (one), the data present in the DPRAM bank is sent.

This bit is not a source for the Endpoint interrupt.

- **Bit 3 - STSENT : Stall Sent Bit**

This bit is a status bit.

It is set (one) by hardware and cleared (zero) by software.

If it is set (one), a STALL has been sent from the corresponding endpoint to the host through the USB bus.

This bit is a source for the Endpoint interrupt and must be cleared to acknowledge the interruption.

- **Bit 2 - RXSETUP : Rx Setup Bit**

This bit is a status bit.

It is set (one) by hardware and cleared (zero) by software.

If it is set (one) and if the corresponding endpoint operates with Control mode transfers, a valid Setup packet has been received from the host through the USB bus.

This bit is a source for the Endpoint interrupt and must be cleared to acknowledge the interruption.

- **Bit 1 - RCVD : Received Data Bit**

This bit is a status bit.

It is set (one) by hardware and cleared (zero) by software.

If it is set (one), the DPRAM of the corresponding endpoint is filled with data coming from the host through the USB bus.

This bit is a source for the Endpoint interrupt and must be cleared to acknowledge the interruption.

- **Bit 0 - TXC : Tx Complete Bit**

This bit is a status bit.

It is set (one) by hardware and cleared (zero) by software.

If it is set (one), an ACK handshake from the host has been received on the corresponding end-point through the USB bus.

This bit is a source for the Endpoint interrupt and must be cleared to acknowledge the interruption.

14.8.8 USBDBCEX - USB Data Byte Count Registers for Endpoint X

Set USBENUM register to point to the relevant Endpoint before using the USBDBCEX register.

If a packet of data has been received from the host in the corresponding endpoint through the USB bus, these registers indicate the amount of data bytes available in the DPRAM.

The value is considered valid if the bit USBCSEX.RCVD has been previously set by hardware.

Bit	7	6	5	4	3	2	1	0	
$0000CC	**BCT7**	**BCT6**	**BCT5**	**BCT4**	**BCT3**	**BCT2**	**BCT1**	**BCT0**	**USBDBCEX**
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 - BCT7..0 : Bytes Count Bits**

These bits are set (one) and cleared (zero) by hardware.

14.8.9 USBFCEX - USB Function Control Registers for Endpoint X

Set USBENUM register to point to the relevant Endpoint before using the USBFCEX register.

Bit	7	6	5	4	3	2	1	0	
$0000CD	**EPE**	**-**	**-**	**-**	**-**	**EPDIR**	**EPTYP1**	**EPTYP0**	**USBFCEX**
Read/write	R/W	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7 - EPE : Endpoint Enable Bit**

This bit is set (one) and cleared (zero) by software.

When it is set (one) the corresponding endpoint is enabled.

When it is cleared (zero) the corresponding endpoint is disabled.

A disabled endpoint does not respond when addressed (read or written) by the host.

At USB reset, EPE for endpoint 0 (USBFCEX.EPE for USBENUM=0) is automatically set by hardware.

- **Bits 6..3 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bit 2- EPDIR : Endpoint Direction Bit**

This bit is set (one) and cleared (zero) by software.

This bit indicates the direction of the endpoint and is not valid for endpoints operating in Control transfer mode as this type of transfer occurs in both direction.

If it is set (one), the endpoint direction is IN.

If it is cleared (zero), the endpoint direction is OUT.

- **Bits 1..0 - EPTYP0 EPTYP1 : Endpoint Type Bits**

These bits are set (one) and cleared (zero) by software.

These bit indicate the type of USB data transfer of the corresponding endpoint. See table below for the values available:

Table 17-1 .Endpoint Types Selection

EPTYP1	EPTYP0	Transfer
0	0	Control
0	1	Isochronous
1	0	Bulk
1	1	Interrupt

14.8.10 USBRSTE - USB Reset Endpoint Register

Bit	7	6	5	4	3	2	1	0	
$0000E5	**RSTE7**	**RSTE6**	**RSTE5**	**RSTE4**	**RSTE3**	**RSTE2**	**RSTE1**	**RSTE0**	**USBRSTE**
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 - RSTE7..0 : Reset Endpoint 7..0 Bits**

These bits are set (one) by software and cleared (zero) by hardware.

Each endpoint has its corresponding bit (e.g RSTE3 corresponds to the endpoint 3) which is used to reset the endpoint.

When resetting an endpoint, the following actions are performed:

- Reset the DPRAM address pointers.
- Set the internal data toggle bit to zero.

To reset endpoint n (n=0..7), which is necessary when changing the device configuration and recommended when receiving a USB bus Reset signaling (before starting the enumeration operations), the following procedure shall be applied:

1. Clear (zero) USBRSTE.RSTEn.
2. Set (one) bit USBRSTE.RSTEn.
3. Wait for USBRSE.RSTEn to be cleared (zero) by hardware (polling).

14.8.11 USBGS - USB Global State Register

Bit	7	6	5	4	3	2	1	0	
$0000E6	**-**	**-**	**-**	**-**	**RSMON**	**RMWUE**	**FCF**	**FAF**	**USBGS**
Read/write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..4 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bits 3 - RSMON : Resume Signal ON**

This bit is set and cleared by hardware.

Set by hardware when a resume is sent to the USB host during remote wake-up sequence.

Automatically cleared when the resume signal emission is halted (last ~13ms).

- **Bits 2 - RMWUE : Remote Wake-Up Enable**

This bit is set and cleared by software.

Set this bit to enable the remote wake-up feature.

- **Bit 1 - FCF : Function Configured Flag Bit**

This bit is cleared (zero) by the hardware when a USB Reset signaling is received.

The software must set (one) this bit after receiving a valid SET_CONFIGURATION request from the host unless it is equal to zero.

The software must clear (zero) this bit after receiving a valid SET_CONFIGURATION request from the host with a zero value.

- **Bit 0 - FAF : Function Addressed Flag Bit**

This bit is cleared (zero) by the hardware when a USB Reset signaling is received.

The software must set (one) this bit after receiving a valid SET_ADDRESS request from the host.

14.8.12 USBFA - USB Function Address Register

Bit	7	6	5	4	3	2	1	0	
$0000E7	-	FADD6	FADD5	FADD4	FADD3	FADD2	FADD1	FADD0	USBFA
Read/write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - Res : Reserved Bit :**

This bit is reserved bit in the AT90SCR075_060 and is always read as zero.

- **Bits 6..0 - FADD6..0 : Function Address Register 6..0 Bits**

These bits are cleared (zero) by the hardware when a USB Reset is received.

The software must update these bits with the address value received during a valid SET_ADDRESS request.

It must then set (one) the bit USBGS.FAF (see “USBGS - USB Global State Register” on page 106).

14.8.13 USBFN - USB Frame Number Registers

Bit	15	14	13	12	11	10	9	8	
$0000E9	-	-	-	FNEND	FNERR	FN10	FN9	FN8	USBFNH
$0000E8	FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0	USBFNL
Bit	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00

- **Bits 15..13 - Res: Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bit 12 - FNEND : Frame Number End Bit**

This bit is set (one) and cleared (zero) by hardware.

This bit is set (one) when an End Of Packet Start Of Frame (SOF) transaction has occurred.

This bit is cleared (zero) by the next detected SOF.

- **Bit 11 - FNERR : Frame Number Error Bit**

This bit is set (one) and cleared (zero) by hardware.

This bit is set (one) if the last Frame Number Field received is corrupted. Otherwise it is cleared (zero).

- **Bits 10..0 - FN10..0 : Frame Number Bits**

These bits are set (one) and cleared (zero) by hardware.

These bits represent the Frame Number value.

The Frame Number value represented by the eleven bits USBFNH.FN2..0 and USBFNL.FN7..0 should only be read when bit USBFNH.FNEND is set (one).

14.9 USBDMA Controller

The USBDMA controller, implemented on the AT90SCR075_060, is intended to be used for executing fast transfers between the RAM memory and the DPRAM (Dual Port RAM) which is dedicated to the USB endpoints. This feature allows the application software of the AT90SCR075_060 to manage the exchanges imposed by the USB protocol.

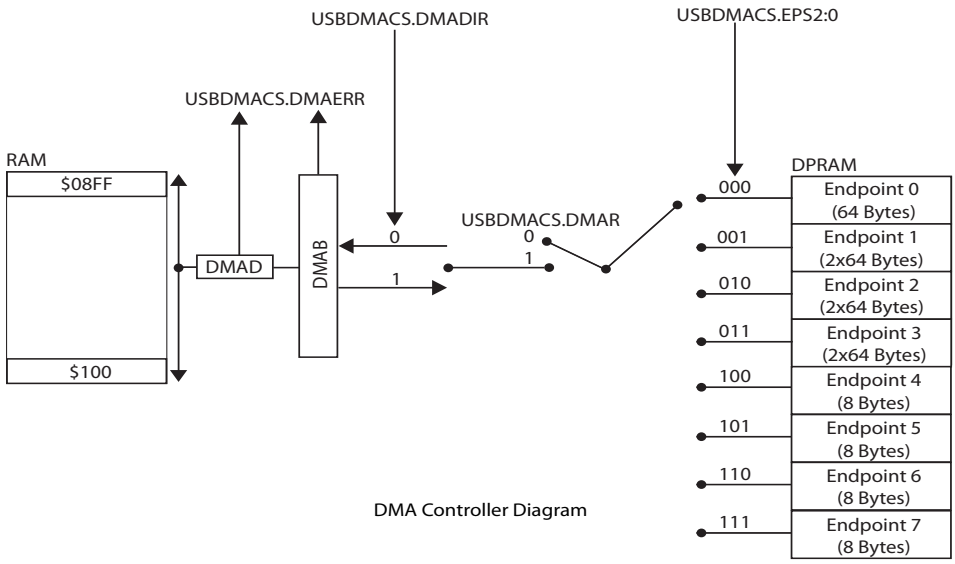
All the USB registers described in this section cannot be accessed if the USB module is not enabled.

The USBDMA controller basically requires four I/O registers to run and can be configured either to send or to receive data through the USB bus. Actually, its main purpose is to transfer data between the RAM of the AT90SCR075_060 and the DPRAM of the USB module. Very easy to use, it just requires the application software to select an endpoint (which can be the source or

the destination), to set a valid base address in RAM (which can be the source or the destination), to fix the amount of data to be exchanged and the direction of the transfer.

One USBDMA operation can transfer up to N bytes in (N+1) 8/16-bit RISC CPU cycles.

Figure 14-6. USBDMA Controller Diagram



Note

Even if not represented above, the exchanges between the RAM and the DPRAM are controlled by the USB USBFI (see “USB Module Diagram” on page 34).

When a USBDMA operation is started, the 8/16-bit RISC CPU is automatically stopped. At the end of the USBDMA operation, the application software automatically restarts where it left (actually with the instruction following the launching of the USBDMA operation). Thus the application software does not need to wait for an interruption or to poll the end of the USBDMA operation.

14.9.1 USBDMACS - USBDMA Control and Status Register

This is the control and status register of the USBDMA controller.

Bit	7	6	5	4	3	2	1	0	
$0000EA	-	EPS2	EPS1	EPS0	-	USBDMAERR	USBDMAIR	USBDMAR	USBDMACS
Read/write	R	R/W	R/W	R/W	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - Res : Reserved Bit**

This bit is reserved in the AT90SCR075_060 and is always read as zero.

- **Bit 6..4 - EPS2..0 : Endpoint Selection Bits**

These bits are set (one) and cleared (zero) by software.

They are used to select the source or destination endpoint for the next USBDMA operation.

A violation is triggered if a USBDMA operation is launched with 0 byte to transfer or when USBMACS.EPS2:0 value is more than 4.

Table 14-2. Endpoint Selection Bits

EPS2	EPS1	EPS0	Endpoint Selected	Endpoint Size (Bytes)
0	0	0	Endpoint 0	64
0	0	1	Endpoint 1	2x64
0	1	0	Endpoint 2	2x64
0	1	1	Endpoint 3	2x64
1	0	0	Endpoint 4	8
1	0	1	Endpoint 5	8
1	1	0	Endpoint 6	8
1	1	1	Endpoint 7	8

- **Bit 3 - Res : Reserved Bit**

This bit is reserved in the AT90SCR075_060 and is always read as zero.

- **Bit 2 - USBDMAERR : USB DMA Error Bit**

When launching the USBDMA controller, this bit is cleared (zero) by the hardware if the values into USBDMADH, USBDMADL and USBDMAB registers are suitable for the USBDMA operation requested.

This bit can also be cleared(zero) by software.

This bit is set (one) by hardware when starting a USBDMA operation and whenever one of these following cases occurs:

- The base address contained in the registers USBDMADH and USBDMADL is incorrect (out of the allowed range).
- According to the values of the registers USBDMADH, USBDMADL and USBDMAB and even if the base address is correct, an address out of the allowed range is going to be reached.
- The value in the register USBDMAB is greater than the size of the selected endpoint for the USBDMA operation (see bits USBDMACS.EPS2:0 below).

When this bit is set, and if the USB unterruptions are enabled, a USB interruption is generated.

Don’t forget to clear the USBDMACS.DMAERR bit before leaving the interruption routine to avoid repetitive and endless interruptions.

- **Bit 1 - USBDMADIR : USB DMA Direction Bit**

This bit is set (one) and cleared (zero) by software.

It indicates the direction of the next USBDMA operation transfer between the RAM memory and the selected endpoint (represented by the bits USBDMACS.EPS2:0):

- If the bit is cleared (zero), the transfer will be from the selected endpoint to the RAM memory (receiving mode).
- If the bit is set (one), the transfer will be from the RAM memory to the selected endpoint (emission mode).

Note: It's not possible to read data back previously stored in DPRAM.

- **Bit 0 - USBDMAR : USB DMA Run Bit**

This bit is set (one) by software and cleared (zero) by hardware.

This bit controls the USBDMA operation launching.

It is set (one) by software when a USBDMA operation is to be performed.

It is cleared (zero) by hardware at the end of the operation.

Note: The software does not need to poll this bit in order to detect the end of the USB-DMA operation. Indeed, when the USBDMACS.DMAR bit is set by the software, the 8/16-bit RISC CPU is automatically stopped. When the end of the USBDMA operation is reached, the 8/16-bit RISC CPU then automatically executes the instructions following the setting of the bit USBDMACS.DMAR.

Note: A USBDMA operation can not be interrupted because the CPU is not available during this time.

14.9.2 USBDMAD - USBDMA ADdress Registers

Bit	15	14	13	12	11	10	9	8	
$0000EC	-	-	USBDMAD [11..8]						USBDMADH
$0000EB	USBDMAD [7..0]								USBDMADL
Bit	7	6	5	4	3	2	1	0	
Read/write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	1	0x01
	0	0	0	0	0	0	0	0	0x00

- **Bits 15..12 - Res : Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bits 11..0 - USBDMAD13..0 : DMA Address**

These bits represents the 12-bit USBDMA Address.

These two registers set the base address in RAM. This address represents the source of the data to be sent if the USBDMA controller is configured in the emission mode. It represents the destination to store the data if the USBDMA controller is configured in the receiving mode.

The initial value corresponds to RAM address $000100.

You can address the whole RAM with this parameter. Values in RAM that must not be dumped, shall be stored out of the USBDMA RAM accessible range.

When starting a USBDMA operation, the hardware will check if the values of USBDMADH, USBDMADL and USBDMAB registers does not exceed the specific RAM area ($000100 to $0008FF). If an error is detected, USBDMACS.DMAERR bit is automatically set (one). A Supervisor RAM Illegal Access Attempt Violation security interrupt (if not masked) is so triggered. USBDMADH, USBDMADL and USBDMAB registers keep their previous value.

After a USBDMA operation, USBDMADH and USBDMADL are set to the last value reached in RAM and incremented by one. For instance, after a 64-byte transfer started from base address $000100, USBDMAD equals to $000140 (USBDMADH = $01 and USBDMADL = $40). This feature allows to simplify registers and bits handlings when several USBDMA operations are to be successively performed, which can be the case when getting or sending several packets.

14.9.3 USBDMAB - USBDMA Amount of Bytes Register

This register is dedicated to the amount of bytes to be transferred during the next USBDMA operation setting.

Bit	7	6	5	4	3	2	1	0	
$0000ED	-	USBDMAB [6..0]							USBDMAB
Read/write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - Res : Reserved Bit**

This bit is reserved bit in the AT90SCR075_060 and is always read as zero.

- **Bits 6..0 - USBDMAB6..0 : DMA Amount of Bytes Bits**

These bits are the (6..0) bits of the 7-bit USBDMA Amount of Bytes value.

When starting a USBDMA operation, the hardware will check if the values of USBDMADH, USBDMADL and USBDMAB registers does not exceed the specific RAM area ($000100 to $0008FF). If an error is detected, USBDMACS.DMAERR bit is automatically set (one). A Supervisor RAM Illegal Access Attempt Violation security interrupt (if not masked) is so triggered. USBDMADH, USBDMADL and USBDMAB registers keep their previous value.

After a USBDMA operation completion, the value of this register is not reset.

The maximum value allowed for USBDMAB depends on the endpoint selected.

15. Smart Card Interface Block (SCIB)

The SCIB provides all signals to interface directly with a smart card. The compliance with the ISO7816, EMV'2000 and GSM standards has been certified. Also, the WHQL standard can be achieved with an appropriate software.

15.1 Features

- **Support of ISO/IEC 7816**
- **Performances: Up to 3 cycles per etu, and ISO clock up to 4.8Mhz**
- **Frequency up to 12Mhz**
- **Character mode**
- **One transmit/receive buffer**
- **11 bits ETU counter**
- **9 bits guard time counter**
- **32 bits waiting time counter**
- **Auto character repetition on error signal detection in transmit mode**
- **Auto error signal generation on parity error detection in receive mode**
- **Power off sequence generation**
- **Manual mode to drive directly the card I/O**

15.2 Overview

All synchronous (e.g. memory card), asynchronous and USB smart cards (e.g. microprocessor card) are supported. The component supplies the different voltages requested by the smart card. The power_on and power_off sequence is directly managed by the SCIB.

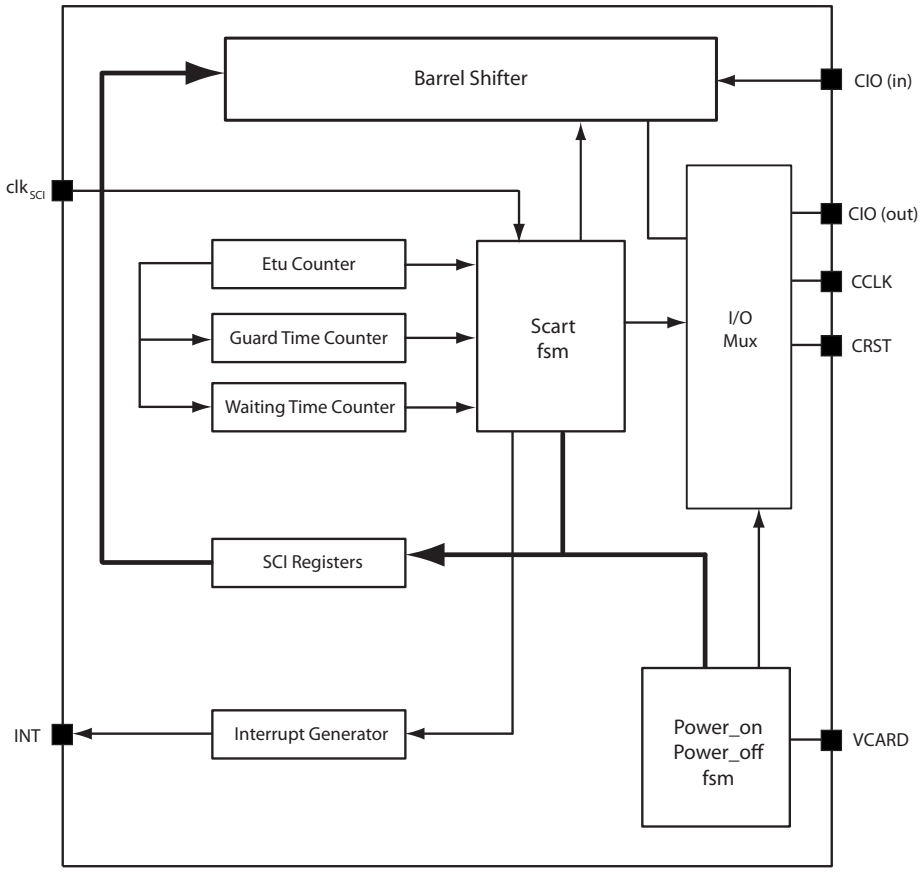
The card presence switch of the smart card connector is used to detect card insertion or card removal. In the case of card removal, the SCIB will automatically initiate a smart card deactivation sequence. An interrupt can be generated when a card is inserted or removed.

Any malfunction is reported to the microcontroller (interrupt + control register).

15.3 Block Diagram

The Smart Card Interface Block diagram is shown Figure 15-1:

Figure 15-1. SCIB Block Diagram



15.4 Definitions

This paragraph introduces some of the terms used in ISO 7816-3 and EMV standards. Please refer to the full standards for a complete list of terms.

Terminal and ICC

Terminal is the reader, ICC is the Integrated Circuit Card

ETU

Elementary Timing Unit (Bit time)

T=0

Character oriented half duplex protocol T=0

T=1

Block oriented half duplex protocol T=1

Activation: Cold Reset

Reset initiated by the Terminal with Vcc power-up. The card will answer with ATR (see below)

Activation: Warm Reset

Reset initiated by the Terminal with Vcc already powered-up, and after a prior ATR or Warm Reset

De-Activation

Deactivation by the Terminal as a result of : unresponsive ICC or ICC removal.

ATR

Answer To Reset. Response from the ICC to a Reset initiated by the Terminal

F and D

F = Clock Rate Conversion Factor, D = Bit rate adjustment factor. ETU is defined as: ETU = F/(D*f) with f= Card Clock frequency. If f is in Hertz, ETU is in second. F and D are available in the ATR (byte TA1 character). The default values are F=372 and D=1.

Guard Time

The time between 2 leading edges of the start bit of 2 consecutive characters is comprised of the character duration (10 ETUs) plus the guard time. Be aware that the Guard Time counter and the Guard Time registers in the AT90SCR075_060 consider the time between 2 consecutive characters. So the equation is Guard Time Counter = Guard Time + 10. In other words, the Guard Time is the number of Stop Bits between 2 characters sent in the same direction.

Extra Guard Time

ISO IEC 7816-3 and EMV introduce the Extra Guard time to be added to the minimum Guard Time. Extra Guard Time only apply to consecutive characters sent by the terminal to the ICC. The TC1 byte in the ATR define the number N. For N=0 the character to character duration is 12 ETUs. For N=254 the character to character duration is 266. For N=255 (special case) The minimum character to character duration is to be used : 12 for T=0 protocol and 11 for T=1 protocol.

Block Guard Time

The time between the leading edges of 2 consecutive characters sent in opposit direction. ISO/IEC 7816-3 and EMV recommend a fixed Block Guard Time of 22 ETUs.

Work Waiting Time (WWT)

In T=0 protocol WWT is the interval between the leading edge of any character sent by the ICC, and the leading edge of the previous character sent either by the ICC or the Terminal. If no character is received by the terminal after WWTmax time, the Terminal initiates a deactivation sequence.

Character Waiting Time (CWT)

In T=1 protocol CWT is the interval between the leading edge of 2 consecutive characters sent by the ICC. If the next character is not received by the Terminal after CWTmax time, the Terminal initiates a deactivation sequence.

Block Waiting Time (BWT)

In T=1 protocol BWT is the interval between the leading edge of the start bit of the last character sent by the Terminal that gives the right to sent to the ICC, and the leading edge of the start bit of the first character sent by the ICC. If the first character from the ICC is not received by the Terminal after BWTmax time, the Terminal initiates a deactivation sequence.

Waiting Time Extention (WTX)

In T=1 protocol the ICC can request a Waiting Time Extension with a S(WTX request) request. The Terminal should acknowlege it. The Waiting time between the leading edge of the start bit of

the last character sent by the Terminal that gives the right to sent to the ICC, and the leading edge of the start bit of the first character sent by the ICC will be BWT*WTX ETUs.

Parity error in T=0 protocol

In T=0 protocol, a Terminal (respectively an ICC) detecting a parity error while receiving a character shall force the Card IO line at 0 starting at 10.5 ETUs, thus reducing the first Guard bit by half the time. The Terminal (respectively an ICC) shall maintain a 0 for 1 ETU min and 2 ETUs max (according to ISO IEC) or to 2 ETUs (according to EMV). The ICC (respectively a Terminal) shall monitor the Card IO to detect this error signal then attempt to repeat the character. According to EMV, following a parity error the character can be repeated one time, if parity error is detected again this procedure can be repeated 3 more times. The same character can be transmitted 5 times in total. ISO IEC7816-3 says this procedure is mandatory in ATR for card supporting T=0 while EMV says this procedure is mandatory for T=0 but does not apply for ATR.

15.5 Functional Description

The architecture of the Smart Card Interface Block can be detailed as follows:

15.5.1 Barrel Shifter

The Barrel Shifter performs the translation between 1 bit serial data and 8-bit parallel data.

The barrel function is useful for character repetition as the character is still present in the shifter at the end of the character transmission.

This shifter is able to shift the data in both directions and to invert the input or output value in order to manage both direct and inverse ISO7816-3 convention.

Coupled with the barrel shifter is a parity checker and generator.

There are 2 registers connected to this barrel shifter, one for the transmission and one for the reception. They act as buffers to relieve the CPU of timing constraints.

15.5.2 SCART FSM

SCART FSM (Smart Card Asynchronous Receiver Transmitter Finite State Machine) is the core of the block. Its purpose is to control the barrel shifter. To sequence correctly the barrel shifter for a reception or a transmission, it uses the signals issued by the different counters. One of the most important counters is the guard time counter that gives time slots corresponding to the character frame.

The SCART FSM is enabled only in UART mode. See “SCICR - Smart Card Interface Control Register” on page 129.

The transition from reception mode to the transmission mode is done automatically. Priority is given to the transmission. Transmission refers to Terminal transmission to the ICC. Reception refers to reception by the Terminal from the ICC.

15.5.3 ETU Counter

The ETU (Elementary Timing Unit) counter controls the operating frequency of the barrel shifter. In fact, it generates the enable signal of the barrel shifter. It receives the Card Clock, and generates the ETU clock. The Card Clock frequency is called “f” below. The ETU counter is 11 bit wide.

A special compensation mode can be activated. It accomodates situations where the ETU is not an integer number of Card Clock (clk_{SCI}). The compensation mode is controlled by the COMP bit in SCETUH register bit position 7. With COMP=1 the ETU of every character even bits is

reduced by 1 Card Clock period. As a result, the average ETU is : ETU_average = (ETU - 0.5). One should bear in mind that the ETU counter should be programmed to deliver a faster ETU which will be reduced by the COMP mechanism, not the other way around. This allows to reach the required precision of the character duration specified by the ISO7816-3 standard.

Example1: F=372, D=32 => ETU= F/D = 11.625 clock cycles.

We select ETU[10-0]= 12 and COMP=1. ETUaverage= 12 - (0.5*COMP) = 11.5

The result will be a full character duration (10 bit) = (10 - 0.107)*ETU. The EMV specification is (10 +/- 0.2)*ETU

15.5.4 Guard Time Counter

The minimum time between the leading edge of the start bits of 2 consecutive characters transmitted by the Terminal is controlled by the Guard Time counter, as described in Figure 15-4.

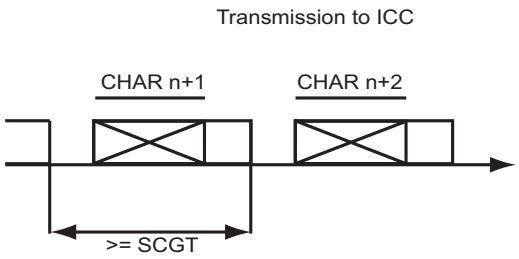
The Guard Time counter is an 9 bit counter. It is initialized to 001h at the start of a transmission by the Terminal. It then increments itself at each ETU until it reaches the 9 bit value loaded into the SCGT register. At this time a new Terminal transmission is enabled and the Guard Time Counter stops incrementing. As soon as a new transmission starts, the Guard Time Counter is re-initialized to 1.

Note

The value of the Guard Time Counter cannot be read. Reading SCGT only gives the minimum time between 2 characters that the Guard Time Counter allows.

Care must be taken with the Guard Time Counter which counts the duration between the leading edges of 2 consecutive characters. This corresponds to the character duration (10 ETUs) plus the Guard Time as defined by the ISO and EMV recommendations. To program Guard Time = 2 : 2 stop bits between 2 characters which is equivalent to the minimum delay of 12 ETUs between the leading edges of 2 consecutive characters, SCGT should be loaded with the value 12 decimal. See Figure 15-2.

Figure 15-2. Guard Time



15.5.5 Block Guard Time Counter

The Block Guard Time counter provides a way to program a minimum time between the leading edge of the start bit of a character transmitted by the ICC and the leading edge of the start bit of a character sent by the TERMINAL. ISO/IEC 7816-3 and EMV recommend a fixed Block Guard Time of 22 ETUs. The AT90SCR075_060 offers the possibility to extend this delay up to 512 ETUs.

The Block Guard Time is a 9 bit counter. When the Block Guard Time mode is enabled (BGTEN=1 in SCSR register) the Block Guard Time counter is initialized to 000h at the start of a

character receptions by the ICC. It then increments at each ETU until it reaches the 9 bit value loaded into shadow SCGT registers, or until it is re-initialized by the start of an new transmission by the ICC. If the Block Guard Time counter reaches the 9 bit value loaded into shadow SCGT registers, a transmission by the TERMINAL is enabled, and the Block Guard Time counter stops incrementing. The Block Guard Time counter is re-initialized at the start of each TERMINAL transmission.

The SCGT shadow registers are loaded with the content of GT[8-0] contained in the registers SCGT with the rising edge of the bit BGTEN in the SCSR register. See Figure 15-4.

Figure 15-3. Block Guard Time

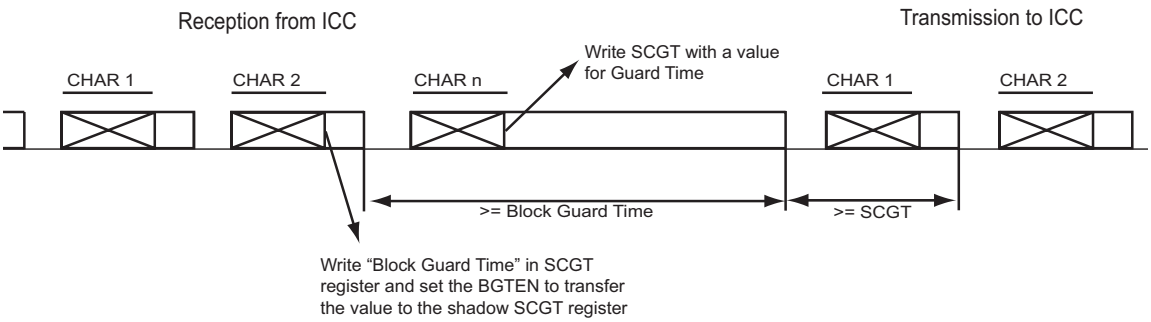
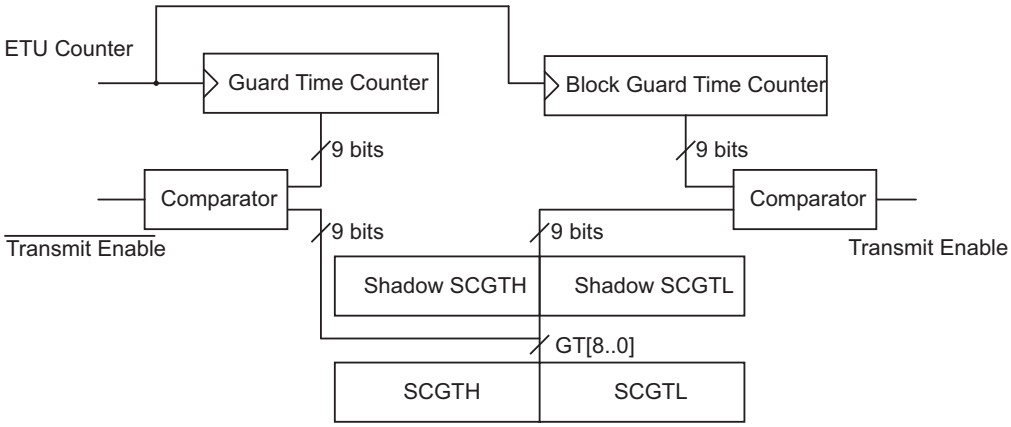


Figure 15-4. Guard Time and Block Guard Time counters



To illustrate the use of Guard Time and Block Guard Time, consider the ISO/IEC 7816-3 recommendation: Guard Time = 2 (minimum delay between 2 consecutive characters sent by the Terminal = 12 ETUs) and Block Guard Time = 22 ETUs.

After a smart Card Reset

- First write ‘00’ decimal in SCGTH, then write ‘22’ decimal in SCGTL
- Set BGTEN in SCSR (BGTEN was 0 before as a result of the smart card reset)
- Write ‘12’ decimal in SCGTL

Now the Guard Time and Block Guard Time are properly initialized. The Terminal will insure a minimun 12 ETUs between 2 leading edges of 2 consecutive characters transmitted. The Terminal will also insure a minimum of 22 ETUs between the leading edge of a character sent by the

ICC, and the leading edge of a character sent by the Terminal. There is no need to write SCGT again and again.

15.5.5.1 *Changing BGT and ETU values*

According to the Answer To Reset or the Protocol and Parameters Selection (PPS, protocol T=0) received, the ETU value should be changed. This ETU duration modification is done after the last byte received from the card. However the ETU change will directly impact the Block Guard Time that is based on ETU duration.

Hence, a special care must be taken to keep the original ETU length until the first block is transmitted by the Terminal in spite of the ETU value change.

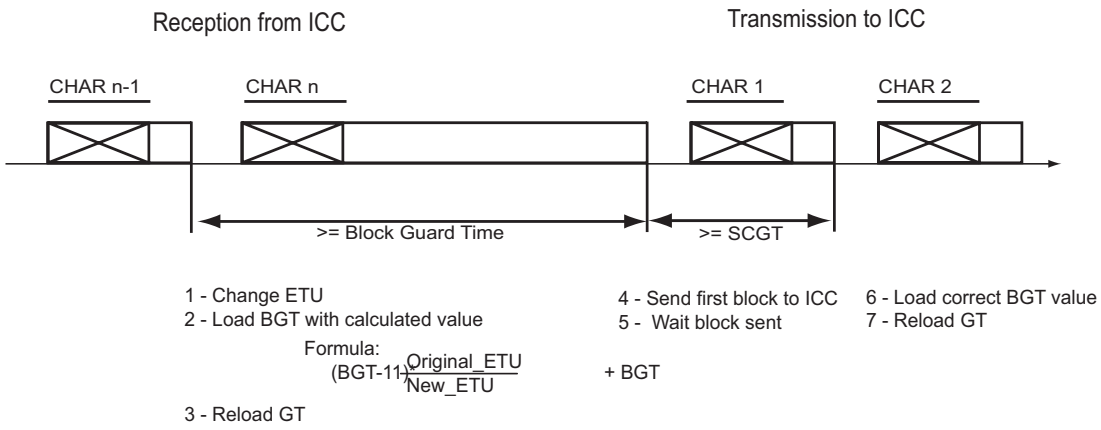
The following procedure describes the way for changing the ETU and BGT values:

1. When changing the ETU, BGT Counter must be reloaded with a value equal to::

$$(BGT + 11)\frac{OriginalETU}{NewETU} + BGT$$

2. Then following transmission of the first block from Terminal to the card, the BGT Counter should be reloaded with the correct BGT value. See Figure 15-5 on page 119 for details.

Figure 15-5. ETU change procedure

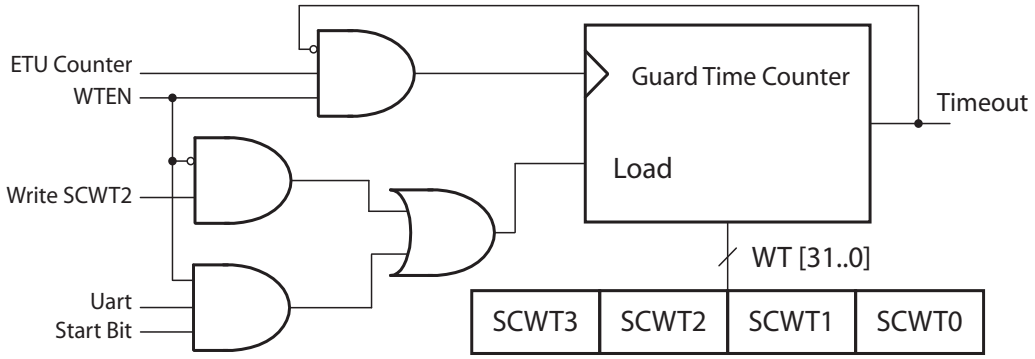


15.5.6 Waiting Time (WT) Counter

The WT counter is a 32 bits down counter which can be loaded with the value contained in the SCWT3, SCWT2, SCWT1, SCWT0 registers. Its main purpose is timeout signal generation. It is 32 bits wide and is decremented at the ETU rate. see Figure 15-6.

When the WT counter times out, an interrupt is generated and the SCIB function is locked: reception and emission are disabled. It can be enabled by resetting the macro or reloading the counter.

Figure 15-6. Waiting Time Counter



15.5.6.1 *Manual or automatic mode*

Two modes are available to load the WT counter. According to the SCICR.UART and SCICR.WTEN bits, the WT counter will be automatically or manually loaded

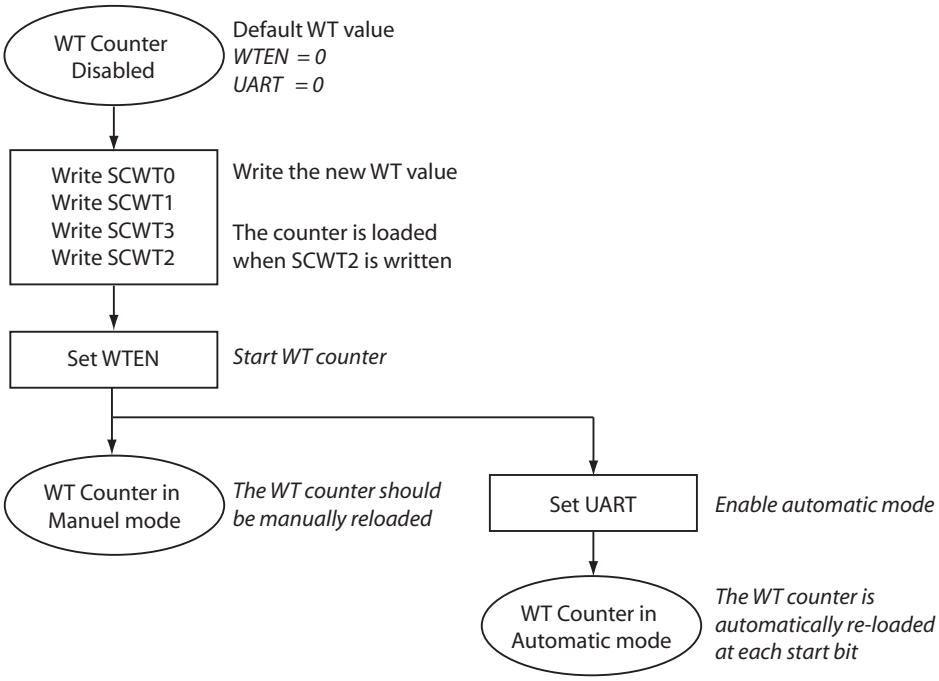
- Manual Mode

When SCICR.UART bit is cleared, the counter is in manual mode and so the counter should be manually loaded.

The WTEN signal controls the start (rising edge) and the stop of the counter (falling edge). After a timeout of the counter, WTEN should be cleared, SCWT2 reloaded and and then WTEN set to start again the counter and to release the SCIB macro.

SCWT3, SCWT2, SCWT1 and SCWT0 registers (SCWT0 contains WT[7-0] bytes) are used to load the Waiting Time counter hold registers with a 32 bit word. In manual mode, the counter hold registers are loaded with SCWT0, SCWT1, SCWT2, SCWT3 values when SCWT2 is written. Please refer to Figure 15-7 for more details.

Figure 15-7. WT Counter Configuration



To reload the WT counter, SCICR.WTEN bit should be cleared before writing a new value in SCWT registers. Then the counter will be enabled by setting SCICR.WTEN bit.

Note

A timeout can occur before the expected time due to resynchronization. To avoid this constraint, a short time has to be added:

$$WTcounter \times ETU + \Delta$$

$$\Delta > 2$$

- Automatic mode

If both SCICR.UART and SCICR.WTEN bits are set, the WT counter is in automatic mode. The WT counter is automatically re-loaded at each start bit detection.

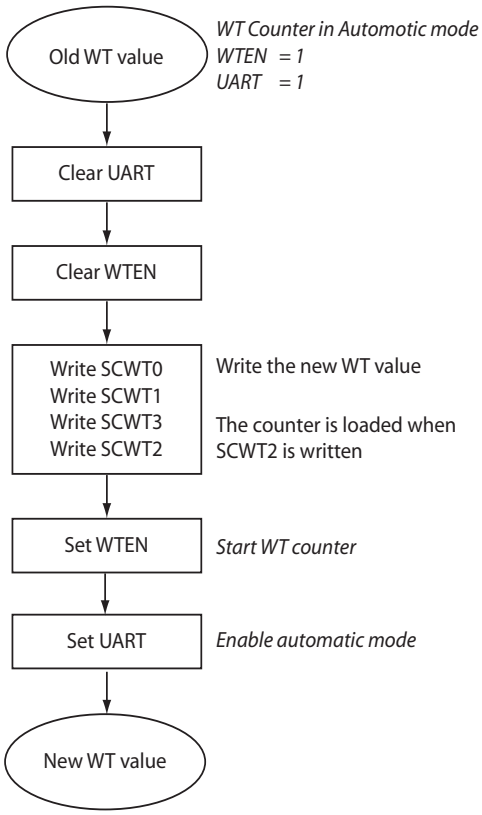
This automatic load is very useful for changing on-the-fly the timeout value since there is a register to hold the load value. This is the case for T=1 protocol. .

Note

In automatic mode, the counter is automatically reloaded if a character is received before or equal to *(WDT+1)*ETU*. Below, SCIB function will be locked.

The Figure 15-8 on page 121 shows the sequence to follow in order to modify the WT in automatic mode.

Figure 15-8. WT change sequence

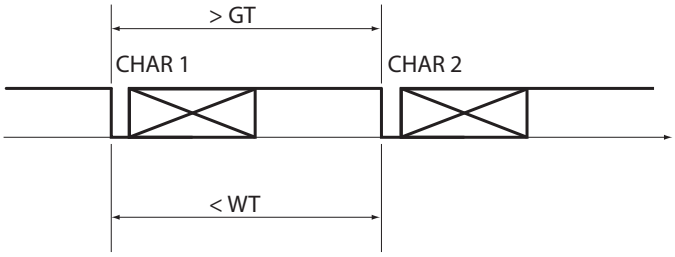


15.5.6.2 *Waiting Time Counter use case*

The Waiting Time Counter can be used in T=0 protocol for the Work Waiting Time. It can be used in T=1 protocol for the Character Waiting Time and for the Block Waiting Time.

In T=0 protocol the maximun interval between the start leading edge of any character sent by the ICC and the start of the previous character sent by either the ICC or the Terminal is the maximum Work Waiting Time. The Work Waiting Time shall not exceed 960*D*WI ETUs with D and WI parameters are returned by the field TA1 and TC2 respectively in the Answer To Reset (ATR). This is the value the user shall write in the SCWT0,1,2,3 register. This value will be reloaded in the Waiting Time counter every start bit.

Figure 15-9. T=0 mode



In T=1 protocol : The maximum interval between the leading edge of the start bit of 2 consecutive characters sent by the ICC is called maximum Character Waiting Time (CWT). The Character Waiting Time shall not exceed (2^{CWI} + 11) ETUs with 0 =< CWI =< 5 (Character Waiting time Integer, CWI). Consequently 12 ETUs =< CWT =< 43 ETUs.

T=1 protocol also specify the maximum Block Waiting Time. This is the time between the leading edge of the last character sent by the Terminal giving the right to send to the ICC, and the leading edge of the start bit of the first character sent by the ICC. The Block Waiting Time shall not exceed {(2^{BWI} * 960) + 11} ETUs with 0 =< BWI =< 4 (Block Waiting time Integer, BWI). Consequently 971 ETUs =< BWT =< 15371 ETUs.

In T=1 protocol, it is possible to extend the Block Waiting Time with the Waiting Time Extension (WTX). When selected the waiting time becomes BWT*WTX ETUs. The Waiting Time counter is 32 bit wide to accomodate this feature.

It is possible to take advantage of the automatic reload of the Waiting Time counter with a start bit in UART mode (T=1 protocol use UART mode) . If the Terminal sends a block of N characters, and the ICC is supposed to answer immediately after, then the following sequence can be used.

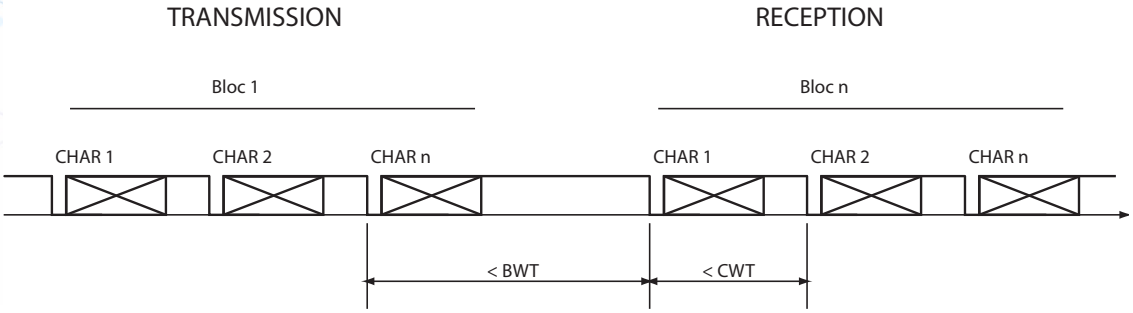
While sending the (N-1)th character of the block, the Terminal can write the SCWT0,1,2,3 with BWImax.

At the start bit of the Nth character, the BWImax is loaded in the Waiting Time counter

During the transmission of the Nth character, the Terminal can write SCWT0,1,2,3 with the CWImax.

At the start bit of the first character sent by the ICC, the CWImax will be loaded in the Waiting Time counter.

Figure 15-10. T=1 Mode



15.5.7 Power-on and Power-off FSM

The Power-on Power-off Finite State Machine (FSM) applies the signals on the smart card in accordance with ISO7816-3 standard. It drives the Activation (Cold Reset and Warm Reset as well as De-Activation) it also manages the exception conditions such as overcurrent (see DC/DC Converter).

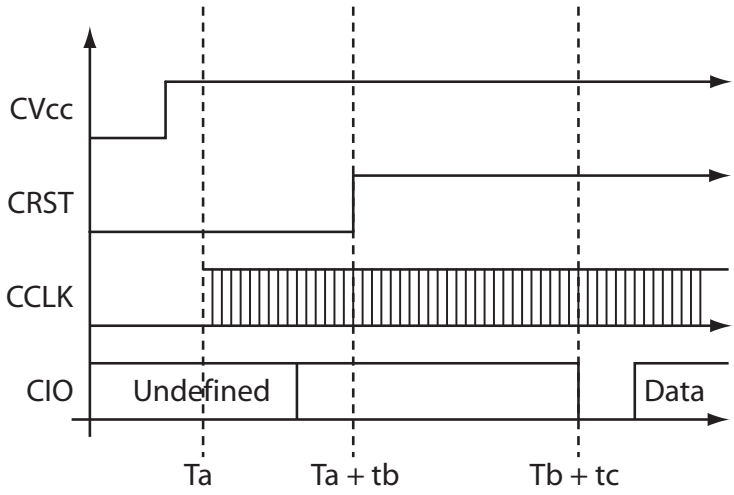
The activation sequence (cold reset and warm reset) and deactivation sequence are managed by software. However, in certain specific cases (e.g. lost of power supply or card extraction), the deactivation sequence is automatically managed by hardware.

To be able to power on the SCIB, the card must be present. After the detection of a card presence, the Terminal initiate a Cold Reset Activation.

The Cold Reset Activation Terminal procedure is as follow and the Figure 15-11. Timing indications are given according to ISO IEC 7816:

- RESET= Low , I/O in the receive state
- Power Vcc (see DC/DC Converter)
- Once Vcc is established, apply Clock at time Ta
- Maintain Reset Low until time Ta+tb (tb< 400 clocks)
- Monitor The I/O line for the Answer To Reset (ATR) between 400 and 40000 clock cycles after Tb (400 clocks < tc < 40000 clocks).

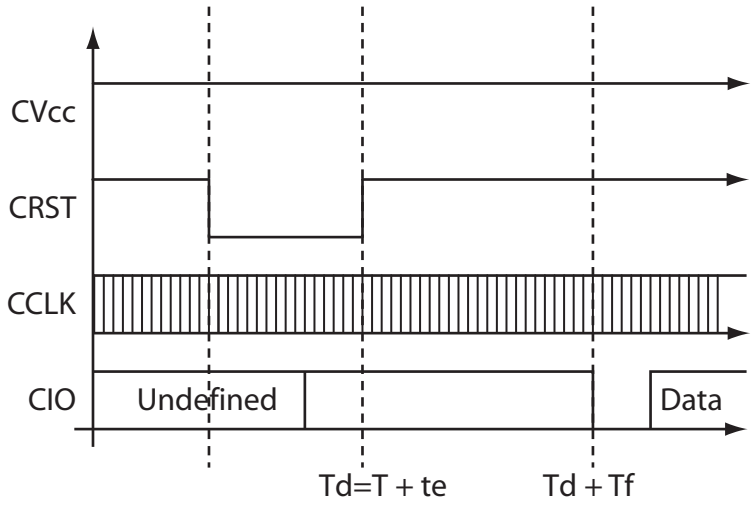
Figure 15-11. SCIB Activation Cold Reset Sequence after a Card Insertion



The Warm Reset Activation Terminal procedure is as follow and the Figure 15-12

- Vcc active, Reset = High, CLK active
- Terminal drives Reset low at time T to initiate the warm Reset. Reset=0 maintained for at least 400 clocks until time Td = T+te (400 clocks < te)
- Terminal keeps the IO line in receive state
- Terminal drives Reset high after at least 400 clocks at time Td
- ICC shall respond with an ATR within 40000 clocks (tf<40000 clocks)

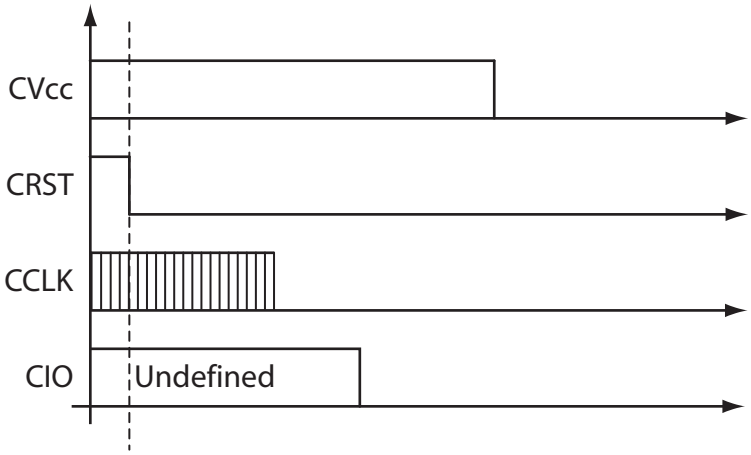
Figure 15-12. SCIB Activation Warm Reset Sequence



The removal of the smart card will automatically start the power off sequence as described in Figure 15-13.

The SCIB deactivation sequence after a lost of power supply is ISO7816-3 compliant. The switch order of the signals is the same as in Figure 15-13 but the delay between signals is analog and not clock dependant.

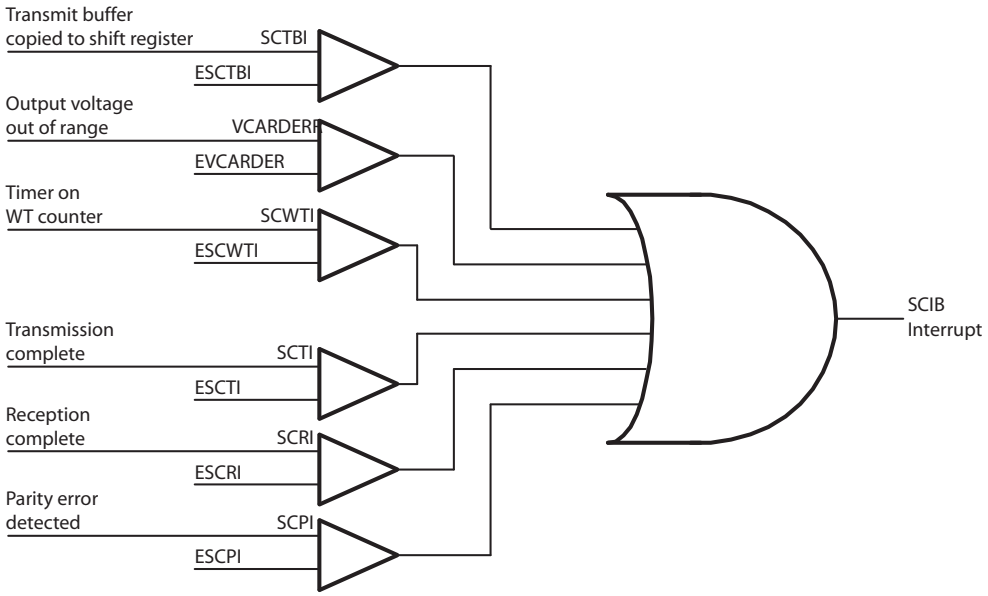
Figure 15-13. SCIB Deactivation Sequence after a Card Extraction



15.5.8 Interrupt Generator

There are several sources of interruption issued by the SCIB. All these interrupts generate the signal: SCIB interruption. See “Interrupts” on page 47.

Figure 15-14. SCIB Interrupt Sources



This signal is high level active. Each of the sources is able to activate the SCIB interruption which is cleared by software by clearing the corresponding bits in the Smart Card Interrupt register.

If another interrupt occurs during the read of the Smart Card Interrupt register, the activation of the corresponding bit in the Smart Card Interrupt register and the new SCIB interruption is delayed until the interrupt register is read by the microcontroller.

Caution

Each bit of the SCIIR register is irrelevant while the corresponding interruption is disabled in SCIER register. When the interruption mode is not used, the bits of the SCISR register must be used instead of the bits of the SCIIR register.

15.6 Additional Features

15.6.1 Clock

The clk_{SCI} input must be in the range 1 - 5 MHz according to ISO 7816.

The clk_{SCI} can be programmed up to 12 MHz. In this case, the timing specification of the output buffer will not be ISO 7816 compliant.

Please refer to section “Clock System” on page 25 for the description of the input clock.

Rising time and Falling time can be changed, please refer to See “SCISRCR - Smart Card Slew Rate Control Register” on page 139.

The dividers values are designed to access most common frequencies of ISO7816 norm. See section “SCICLK - Smart Card Clock Register” on page 138, for the clock frequency available in output of the Smart Card interface.

15.6.2 Card Presence Input

The CPRES input can generate an interrupt on card insertion or on card removal. To do so, global interrupt must be enabled and SCIER.CARDINE must be set. The CPRES interrupt is generated by an event on CPRES (i.e. a high or low edge depending on the setting of SCICR.CARDDET).

CPRES interrupt is triggered either on card insertion or on card removal, so the card presence can be checked thanks to the SCISR.CARDIN bit.

As soon as the program executes the CPRES interrupt routine, the CPRES interrupt is automatically cleared. See “Interrupts” on page 47 for CPRES interrupt vector address.

If Card Presence interruption is enabled, the AT90SCR075_060 wakes up from low power mode as soon as an event on CPRES is detected.

An internal pull-up on Card Presence input can be disconnected in order to reduce the consumption (SCSR.CPRESRES). In this case, an external resistor (typically 1 MΩ) must be externally tied to Vcc.

15.6.3 Transmit / Receive Buffer

The contents of the SCIBUF Transmit / Receive Buffer is transferred or received into / from the Shift Register. The Shift Register is not accessible by the microcontroller. Its role is to prepare the byte to be copied on the I/O pin for a transmission or in the SCIBUF buffer after a reception.

During a character transmission process, as soon as the contents of the SCIBUF buffer is transferred to the shift register, the SCTBE bit is set in SCISR register to indicate that the SCIBUF buffer is empty and ready to accept a new byte. This mechanism avoids to wait for the complete transmission of the previous byte before writing a new byte in the buffer and enables to speed up the transmission.

- If the Character repetition mode is not selected (bit CREP=0 in SCICR), as soon as the contents of the Shift Register is transferred to I/O pin, the SCTC bit is set in SCISR register to indicate that the byte has been transmitted.
- If the Character repetition mode is selected (bit CREP=1 in SCICR) The terminal will be able to repeat characters as requested by the ICC (See the Parity Error in T=0 protocol description in the definition paragraph above). The SCTC bit in SCISR register will be set after a successful transmission (no retry or no further retry requested by the ICC). If the number of retries is exhausted (up to 4 retries depending on CREPSEL bit in SCSR) and the last attempt is still unsuccessful, the SCTC bit in SCISR will not be set and the SCPE bit in SCISR register will be set instead.

During a character reception process, the contents of the Shift Register is transferred in the SCIBUF buffer.

- If the Character repetition mode is not selected (bit CREP=0 in SCICR), as soon as the contents of the Shift Register is transferred to the SCIBUF the SCRC bit is set in SCISR register to indicate that the byte has been received, and the SCIBUF contains a valid character ready to be read by the microcontroller.

- If the Character repetition mode is selected (bit CREP=1 in SCICR) The terminal will be able to request repetition if the received character exhibit a parity error. Up to 4 retries can be requested depending on CREPSEL bit in SCSR. The SCRC bit will be set in SCISR register after a successful reception, first reception or after retry(ies). If the number of retries is exhausted (up to 4 retries depending on CREPSEL bit in SCSR) and the last retry is still unsuccessful, the SCRC bit and the SCPE bit in SCISR register will be set. It will be possible to read the erroneous character.

Caution

The SCTBI, SCTI, SCRI and SCPI bits have the same functions as SCTBE, SCTC, SCRC and SCPE bits. The first ones are able to generate interruptions if the interruptions are enabled in SCIER register while the second ones are only status bits to be used in polling mode. If the interruption mode is not used, the status bits must be used. The SCTBI, SCTI and SCRI bits do not contain valid information while their respective interrupt enable bits ESCTBI, EXCTI, ESCRI are cleared.

Figure 15-15. Character Transmission Diagram

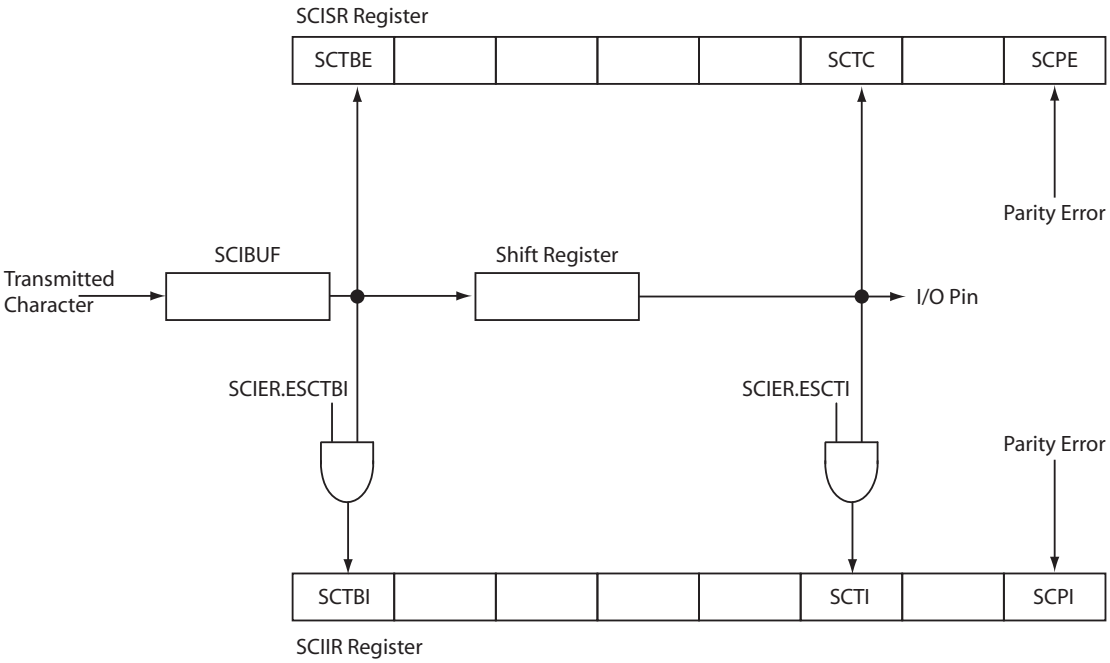
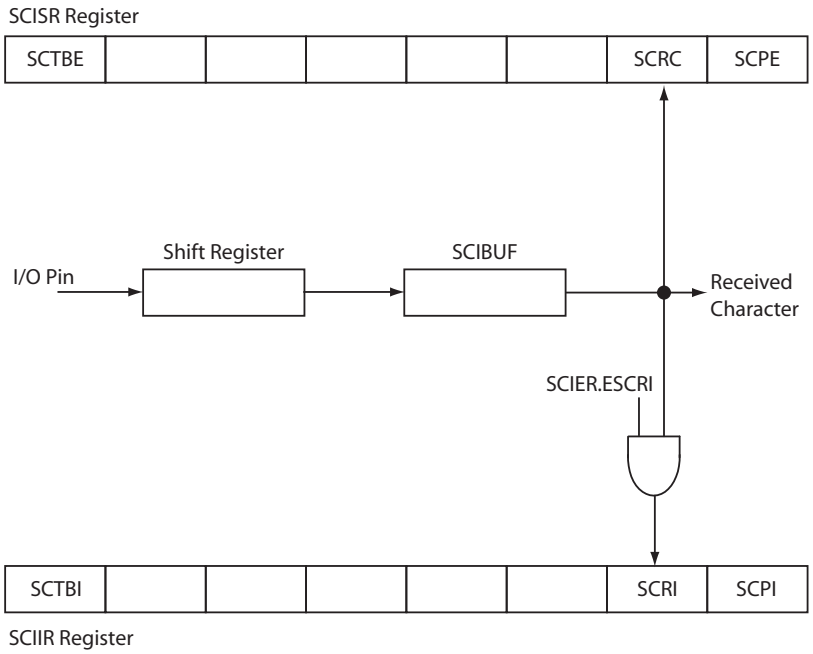


Figure 15-16. Character Reception Diagram



15.6.4 SCIB Reset

The SCICR register contains a reset bit. If set, this bit generates a reset of the SCIB and its registers. All default values will be set into the SCIB registers.

15.7 SCI registers access

Due to synchronisation between the core clock (clk_{Core}) and the Smart Card Interface clock (clk_{SCI}), some precautions have to be taken for software access.

Indeed a delay can occur between the CPU writing and when the data is present in the smart card interface registers. Hence if the CPU writes a new data in a SCI register, during a delay:

- The value read may be equal to the old one.
- Using a mask to set/clear some bits may be ignored.
- Reading some bits that control the output of SCI pads (e.g. CARD CLK, RST...) may give the value seen on the pad.

After writting a data, it is recommended to check the register content until the new value is well updated. All registers that require this check will be noted in the following paragraph, “Smart Card Interface Block Registers” on page 129.

15.8 Smart Card Interface Block Registers

15.8.1 SCICR - Smart Card Interface Control Register

Bit	7	6	5	4	3	2	1	0	
$0000FF	SCIRESET	CARDDET	VCARD [1..0]		UART	WTEN	CREP	CONV	SCICR[1]
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

Note: 1. See “SCI registers access” on page 128.

- **Bit 7 – SCIRESET: Smart Card Interface Reset Bit**

Set this bit to reset the Smart Card Interface. This bit acts as an active high software reset.

There is no auto-deactivation of the SCIB when resetting the block via SCIRESET. The software must ensure the interface is powered down prior to executing the reset sequence. This sequence will ensure ISO compliance.

Clear this bit to activate the Smart Card interface. The read back value becomes 0 only when the card interface is completely activated and ready.

When clearing this bit by software, it is required to wait till this bit is cleared before using the interface.

- **Bit 6 – CARDDET: Card Presence Detector Sense**

Clear this bit to indicate the card presence detector is open when no card is inserted (CPRES is high and will go low when a card is inserted).

Set this bit to indicate the card presence detector is closed when no card is inserted (CPRES is low and will go high when a card is inserted).

Switching the value of CARDDET parameter can generate interruption if SCIER.CARDINIE is set.

SCIER.CARDINIE must be set only when CARDDET bit is correctly configured.

- **Bit 5..4 – VCARD[1..0]: Card Voltage Selection**

Table 15-1. Card Voltage Selection

VCARD1	VCARD0	CVcc
0	0	1.8V
0	1	
1	0	3.0V
1	1	5.0V

Note

The DCDC peripheral must be ON and the SCCON.CARDVCC bit must be high to generate the programmed card voltage.

Caution

Changing VCARD while DC/DC is workding and CVcc is already ON could generate unexpected behavior. Please power_off the CVcc, and take care DC/DC is not busy before changing these bits and reapplying new voltage.

See Figure 16-2 on page 143.

- **Bit 3 – UART: Card UART Selection**

Clear this bit to use the CARDIO bit to drive the Card I/O (CIO) pin.

Set this bit to use the Smart Card UART to drive the Card I/O (CIO) pin.

Note

It is recommended to set Card I/O pin (SCCON.CARDIO) before setting the UART bit to avoid a glitch on the line.

Controls also the Waiting Time Counter as described in “Waiting Time (WT) Counter” on page 119

- **Bit 2 – WTEN: Waiting Time Counter Enable**

Clear this bit to stop the counter and enable the load of the Waiting Time counter hold registers.

The hold registers are loaded with SCWT0, SCWT1, SCWT2, SCWT3 values when SCWT2 is written.

Set this bit to start the Waiting Time Counter. The counters stop when it reaches the timeout value.

If the UART bit is set, the Waiting Time Counter automatically reloads with the hold registers whenever a start bit is sent or received. .

Caution

WTEN must be set only when UART is cleared. Else the counter will not work correctly.

- **Bit 1 – CREP: Character Repetition**

Clear this bit to disable parity error detection and indication on the Card I/O pin in receive mode and to disable character repetition in transmit mode.

Set this bit to enable parity error indication on the Card I/O pin in receive mode and to set automatic character repetition when a parity error is indicated in transmit mode.

Depending upon CREPSET bit is SCSR register, the receiver can indicate parity error up to 4times (3 repetitions) or up to 5times (4 repetitions) after which it will raise the parity error bit

SCPE bit in the SCISR register. If parity interrupt is enabled, the SCPI bit in SCIIR register will be set too.

Alternately, the transmitter will detect ICC character repetition request. After 3 or 4 unsuccessful repetitions (depending on CREPSEL bit in SCSR register), the transmitter will raise the parity error bit SCPE bit in the SCISR register. If parity interrupt is enabled, the SCPI bit in SCIIR register will be set too.

Character repetition mode is specified for T=0 protocol only and should not be used in T=1 protocol (block oriented protocol)

- **Bit 0 – CONV: ISO Convention**

Clear this bit to use the direct convention: b0 bit (LSB) is sent first, the parity bit is added after b7 bit and a low level on the Card I/O pin represents a'0'.

Set this bit to use the inverse convention: b7 bit (LSB) is sent first, the parity bit is added after b0 bit and a low level on the Card I/O pin represents a'1'.

15.8.2 SCCON - Smart Card Contacts Register

Bit	7	6	5	4	3	2	1	0	
$0000FE	CLK	-	-	-	CARDIO	CARDCLK	CARDRST	CARDVCC	SCCON(1)
Read/write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

Note: 1. See "SCI registers access" on page 128.

- **Bit 7 – CLK: Card Clock Selection**

Clear this bit to use the SCCON.CARDCLK bit to drive Card CLK pin.

Set this bit to use clk_{SCI} signal to drive the Card CLK pin

Internal synchronization avoids glitches on the CLK pin when switching this bit.

- **Bit 6..4 – Reserved Bit**

This bit is reserved for future use.

- **Bit 3 – CARDIO: Card I/O**

If UART bit is cleared in SCICR register, this bit enables the use of the Card IO pin (CIO pin) as a standard bi-directional port :

- To read from CIO port pin : set CARDIO bit then read CARDIO bit to have the CIO port value
- To write in CIO port pin : set CARDIO bit to write a 1 in CIO port pin , clear CARDIO bit to write a 0 in CIO port pin.

VCARDOK=1 (SCISR.4 bit) condition must be true to change the state of CIO pin.

Caution It is mandatory to set CIO before setting UART bit.

- **Bit 2 – CARDCLK: Card CLK**

When the CLK bit is cleared in SCCON Register, the value of this bit is driven to the Card CLK pin.

VCARDOK=1 (SCISR.4 bit) condition must be true to change the state of Card CLK pin.

- **Bit 1 – CARDRST: Card RST**

Clear this bit to drive a low level on the Card RST pin.

Set this bit to set a high level on the Card RST pin.

VCARDOK=1 (SCISR.4 bit) condition must be true to change the state of Card RST pin.

- **Bit 0 – CARDVCC: Card Vcc Control**

Clear this bit to deactivate the Card interface and set its power-off. The other bits of SCCON register have no effect while this bit is cleared.

Set this bit to power-on the Card interface. The activation sequence should be handled by software.

15.8.3 SCISR - Smart Card UART Interface Status Register

Bit	7	6	5	4	3	2	1	0	
$0000FD	SCTBE	CARDIN	-	VCARDOK	SCWTO	SCTC	SCRC	SCPE	SCISR[1]
Read/write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	1	0	0	0	0	0	0	0	0x80

Note: 1. See “SCI registers access” on page 128.

- **Bit 7 – SCTBE: UART Transmit Buffer Empty Status**

This bit is set by hardware when the Transmit Buffer is copied to the transmit shift register of the Smart Card UART.

It is cleared by hardware when SCIBUF register is written.

Caution This bit can only be used if the Guardtime parameter is lesser or equal to 11etu. The meaning of this bit if guardtime is higher than 11etu is not reliable; In this case, using SCTC is mandatory.

- **Bit 6 – CARDIN: Card Presence Status**

This bit is set by hardware if there is a card presence (debouncing filter has to be done by software).

This bit is cleared by hardware if there is no card presence.

If global interrupts are activated and SCIER.CARDINE is set (one), and Card Presence goes high, an interrupt is generated. See “Card Presence Input” on page 126.

- **Bit 5 – Res: Reserved Bit**

This bit is reserved and will always be read as ‘0’.

- **Bit 4 – VCARDOK: Card Voltage Correct Status**

This bit is set when the output voltage is within the voltage range specified by VCARD[1:0] in SCICR register.

It is cleared otherwise.

To modify the bits CARDIO, CARDCLK and CARDRST in register SCCON, the bit VCAROK must be equal to 1.

- **Bit 3 – SCWTO: Waiting Time Counter Timeout Status**

This bit is set by hardware when the Waiting Time Counter has expired.

It is cleared by reloading the counter or by reseting the SCIB.

- **Bit 2 – SCTC: UART Transmitted Character Status**

This bit is set by hardware when the Smart Card UART has transmitted a character.

If character repetition mode is selected, this bit will be set only after a successful transmission. If the last allowed repetition is not successful, this bit will not be set.

This bit must be cleared by software.

- **Bit 1 – SCRC: UART Received Character Status**

This bit is set by hardware when the Smart Card UART has received a character.

This bit must be cleared by software.

If character repetition mode is selected, this bit will be set only after a successful reception. If the last allowed repetition is still unsuccessful, this bit will be set to let the user read the erroneous value if necessary.

- **Bit 0 – SCPE: Character Reception Parity Error Status**

This bit is set when a parity error is detected on the received character.

This bit must be cleared by software.

If character repetition mode is selected, this bit will be set only if the ICC report an error on the last allowed repetition of a TERMINAL transmission, or if a reception parity error is found on the last allowed ICC character repetition.

15.8.4 SCIIR - Smart Card UART Interrupt Identification Register

Bit	7	6	5	4	3	2	1	0	
$0000FC	**SCTBI**	**-**	**-**	**VCARDERR**	**SCWTI**	**SCTI**	**SCRI**	**SCPI**	**SCIIR**
Read/write	R	R	R/W	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – SCTBI: UART Transmit Buffer Empty Interrupt**

This bit is set by hardware when the Transmit Buffer is copied into the transmit shift register of the Smart Card UART. It generates an interrupt if ESCTBI bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

- **Bit 5..6 – Res: Reserved Bits**

Those bits are reserved.

- **Bit 4 – VCARDERR: Card Voltage Error Interrupt**

This bit is set when the output voltage goes out of the voltage range specified by VCARD field. It generates an interrupt if EVCARDER bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

- **Bit 3 – SCWTI: Waiting Time Counter Timeout Interrupt**

This bit is set by hardware when the Waiting Time Counter has expired. It generates an interrupt if ESCWTI bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

- **Bit 2 – SCTI: UART Transmitted Character Interrupt**

This bit is set by hardware when the Smart Card UART has completed the character transmission. It generates an interrupt if ESCTI bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

- **Bit 1 – SCRI: UART Received Character Interrupt**

This bit is set by hardware when the Smart Card UART has completed the character reception. It generates an interrupt if ESCRI bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

- **Bit 0 – SCPI: Character Reception Parity Error Interrupt**

This bit is set at the same time as SCTI or SCRI if a parity error is detected on the received character. It generates an interrupt if ESCPI bit is set in SCIER register otherwise this bit is irrelevant.

This bit must be cleared by software.

15.8.5 SCIER - Smart Card UART Interrupt Enable Register

Bit	7	6	5	4	3	2	1	0	
$0000FB	**ESCTBI**	**CARDINE**	**-**	**EVCARDER**	**ESCWTI**	**ESCTI**	**ESCRI**	**ESCPI**	**SCIER**(1)
Read/write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

Note: 1. See “SCI registers access” on page 128.

- **Bit 7 – ESCTBI: UART Transmit Buffer Empty Interrupt Enable**

Clear this bit to disable the Smart Card UART Transmit Buffer Empty interrupt.

Set this bit to enable the Smart Card UART Transmit Buffer Empty interrupt.

- **Bit 6 – CARDINE: Card In Interrupt Enable**

Clear this bit to disable te Smart Card Card Presence Detection interrupt generation.

Set this bit to enable te Smart Card Card Presence Detection interrupt generation.

- **Bit 5 – Res: Reserved Bits**

Those bits are reserved and will always be read as ‘0’.

- **Bit 4 – EVCARDER: Card Voltage Error Interrupt Enable**

Clear this bit to disable the Card Voltage Error interrupt.

Set this bit to enable the Card Voltage Error interrupt.

- **Bit 3 – ESCWTI: Waiting Time Counter Timeout Interrupt Enable**

Clear this bit to disable the Waiting Time Counter timeout interrupt.

Set this bit to enable the Waiting Time Counter timeout interrupt.

- **Bit 2 – ESCTI: UART Transmitted Character Interrupt Enable**

Clear this bit to disable the Smart Card UART Transmitted Character interrupt.

Set this bit to enable the Smart Card UART Transmitted Character interrupt.

- **Bit 1 – ESCRI: UART Received Character Interrupt Enable**

Clear this bit to disable the Smart Card UART Received Character interrupt.

Set this bit to enable the Smart Card UART Received Character interrupt.

- **Bit 0 – ESCPI: Character Reception Parity Error Interrupt Enable**

Clear this bit to disable the Smart Card Character Reception Parity Error interrupt.

Set this bit to enable the Smart Card Character Reception Parity Error interrupt.

15.8.6 SCSR - Smart Card Selection Register

Bit	7	6	5	4	3	2	1	0	
$0000FA	-	BGTEN	-	CREPSEL	CPRESRES	-	-	-	SCSR[1]
Read/write	R	R/W	R	R/W	R/W	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

Note: 1. See “SCI registers access” on page 128.

- **Bit 7 – Reserved Bit**

This bit is reserved for future use.

- **Bit 6 – BGTEN: Block Guard Time Enable**

Set this bit to select the minimum interval between the leading edge of the start bits of the last character received from the ICC and the first character sent by the Terminal. The transfer of GT[8-0] value to the BGT counter is done on the rising edge of the BGTEN.

Clear this bit to suppress the minimum time between reception and transmission..

The SCSR.BGTEN must not be cleared and set quickly because the macro may ignore it and the BGT counter won't be reloaded.

- **Bit 5 – Reserved Bit**

This bit is reserved for future use.

- **Bit 4 – CREPSEL: Character Repetition Selection**

Clear this bit to select 5 times transmission (1 original + 4 repetitions) before parity error indication (conform to EMV).

Set this bit to select 4 times transmission (1 original + 3 repetitions) before parity error indication.

- **Bit 3 – CPRESRES: Card Presence Pull-up Resistor**

Clear this bit to connect the internal 100K Pull-up on CPRES pin.

Set this bit to disconnect the internal pull-up from this pin.

- **Bit 2..0 – Reserved Bits**

These bits are reserved for future use. Writing these reserved bits can have side effects. Take care of not writing them.

15.8.7 SCIBUF - Smart Card Transmit/Receive Buffer

Bit	7	6	5	4	3	2	1	0	
$0000F9	SCIBUFD [7..0]								SCIBUF
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..0 – SCIBUFD7..0: Smart Card Transmit/Receive Buffer**

A new byte can be written in the buffer to be transmitted on the I/O pin when SCTBE bit is set. The bits are sorted and copied on the I/O pin according to the active convention. The register can not be read anymore as soon as SCTBE is set.

A new byte received from I/O pin is ready to be read when SCRI bit is set. The bits are sorted according to the active convention.

15.8.8 SCETU - Smart Card ETU Register

Bit	15	14	13	12	11	10	9	8	
$0000F8	**COMP**	**-**	**-**	**-**	**-**	**ETU [10..8]**			**SCETUH**(1)
$0000F7	**ETU [7..0]**								**SCETUL**(1)
Bit	7	6	5	4	3	2	1	0	
Read/write	R/W	R	R	R	R	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	1	0x01
	0	1	1	1	0	1	0	0	0x74

Note: 1. See “SCI registers access” on page 128.

- **Bit 15 – COMP: Compensation**

Clear this bit when no time compensation is needed (i.e. when the ETU to Card CLK period ratio is close to an integer with an error less than 1/4 of Card CLK period).

Set this bit otherwise and reduce the ETU period by 1 Card CLK cycle for even bits.

- **Bit 14..11 – Reserved Bits**

These bits are reserved for future use.

- **Bit 10..0 – ETU: ETU Value**

The Elementary Time Unit is (ETU[10:0] - 0.5*COMP)/f, where f is the Card CLK frequency.

According to ISO 7816, ETU[10:0] can be set between 11 and 2048.

The default reset value of ETU[10:0] is 372 (F=372, D=1).

The ETU counter is reloaded at each register’s write operation.

Do not change this register during character reception or transmission or while Guard Time or Waiting Time Counters are running.

15.8.9 SCGT - Smart Card Guard Time Register

Bit	15	14	13	12	11	10	9	8	
$0000F6	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**GT8**	**SCGTH**(1)
$0000F5	**GT [7..0]**								**SCGTL**(1)
Bit	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	1	1	0	0	0x0C

Note: 1. See “SCI registers access” on page 128.

- **Bit 15..9 – Reserved Bits**

These bits are reserved for future use.

- **Bit 8..0 – GT8..0: Transmit Guard Time**

The minimum time between two consecutive start bits in transmit mode is GT[8:0] x ETU. This is equal to ISO IEC Guard Time +10. See “Block Guard Time Counter” on page 117.

According to ISO IEC 7816, the time between 2 consecutive leading edge start bits can be set between 11 and 266 (11 to 254+12 ETUs).

15.8.10 SCWT - Smart Card Character/Block Waiting Time Register

Bit	7	6	5	4	3	2	1	0	
$0000F4	WT [31..24]								SCWT3(1)
$0000F3	WT (23..16]								SCWT2(1)
$0000F2	WT [15..8]								SCWT1(1)
$0000F1	WT [7..0]								SCWT0(1)
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00
	0	0	1	0	0	1	0	1	0x25
	1	0	0	0	0	0	0	0	0x80

Note: 1. See “SCI registers access” on page 128.

- **Bit 31..0 – WT: Waiting Time Byte**

WT[31:0] is the reload value of the Waiting Time Counter (WTC).

The WTC is a general-purpose timer. It uses the ETU clock and is controlled by the WTEN bit (See “SCICR - Smart Card Interface Control Register” on page 129. and See “Waiting Time (WT) Counter” on page 119.).

When UART bit of “Smart Card Interface Block Registers” is set, the WTC is automatically reloaded at each start bit of the UART. It is used to check the maximum time between to consecutive start bits.

15.8.11 SCICLK - Smart Card Clock Register

Bit	7	6	5	4	3	2	1	0	
$0000F0	-	-	SCICLK [5..0]						SCICLK
Read/write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	1	0	0	0x04

- **Bit 7..6 – Reserved Bits**

These bits are reserved for future use.

- **Bit 5..0 – SCICLK5..0: Clock Bits**

Combination of SCICLK bits provide clock divider for Smart Card Interface output as follow:

Table 15-2. Clock Dividers For Smart Card Interface

Clk_{PLL}	SCICLK [5-0]	SCIB Frequency (Mhz)	Divider
96	0x00	12	8
96	0x01	8	12
96	0x02	6	16
96	0x03	4.8	20
96	0x04	4	24
96	0x05	2	48
96	0x06	1.5	64
96	0x07	1.2	80
96	0x08	1	96
96	0x09	0.75	128
96	0x0A	0.6	160
96	0x0B	0.5	192
96	0x0C	Reserved	Reserved
96	0x0D	Reserved	Reserved
96	0x0E	Reserved	Reserved

15.8.12 SCISRCR - Smart Card Slew Rate Control Register

Bit	7	6	5	4	3	2	1	0	
$0000EE	-	-	-	-	SRC3	SRC2	SRC1	SRC0	SCISRCR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..4 – Reserved Bits**

These bits are reserved for future use.

- **Bit 3..0 – SRC3..0: Slew Rate Control Value**

The SRC3:0 bits determine the Slew Rate Control value when SCI Clock is active. The different values are shown in

Table 15-3. Slew Rate Control Values

SRC [3-0]	Slope Value for Rising and Falling time
0x00	Default Value, T0
0x01	T0-1ns
0x03	T0-2ns
0x07	T0-3ns
0x0F	T0-4ns

SRC [3-0]	Slope Value for Rising and Falling time
0x05	2
0x06	1.5
0x07	1.2
0x08	1
0x09	0.75
0x0A	0.6
0x0B	0.5
0x0C	Reserved
0x0D	Reserved
0x0E	Reserved

16. DC/DC Converter

16.1 Overview

The AT90SCR075_060 embeds a DC/DC converter to generate voltage to supply any kind of ISO7816 Smart Card, and to generate the VBUS voltage for USB Smart Cards.

This peripheral is also controlled by “Smart Card Interface Block (SCIB)” .

16.2 Features

- **Operating Voltages : 2.7 - 5.5 V**
- **Programmable Ouput Voltages : 1.8V, 3V or 5V**
- **Automatic deactivation sequence when switched offf**

To prevent any perturbation avoiding the DC/DC Converter to work correctly, an RSense Resistance is connected between Power Supply and CVSense Pin. The value of this resistance depends of the resistance of the Bonding Wires “Smart Card Interface Characteristics” on page 237 and the resistance of the wires.

The total value of these resistances should be around 500m Ohms and the RSense value is chosen in consequence, See “Application Information” on page 223.

16.3 Description

The Smart Card voltage (CVcc) is supplied by the integrated DC/DC converter which is controlled by several registers:

- The DCCR register controls the start-up, shutdown of DC/DC peripheral. See “DCCR - DC/DC Converter Register” on page 143.
- The SCICR register controls the CVcc level by means of VCARD[1..0] bits. See “SCICR - Smart Card Interface Control Register” on page 129.
- The SCCON register controls the generation of by means of CARDVCC bit. See “SCCON - Smart Card Contacts Register” on page 131.

The CVcc cannot be generated while the CPRES pin remains inactive. If CPRES pin becomes inactive while the DC/DC converter is operating, an automatic power_off sequence of the DC/DC converter is initiated by the internal logic.

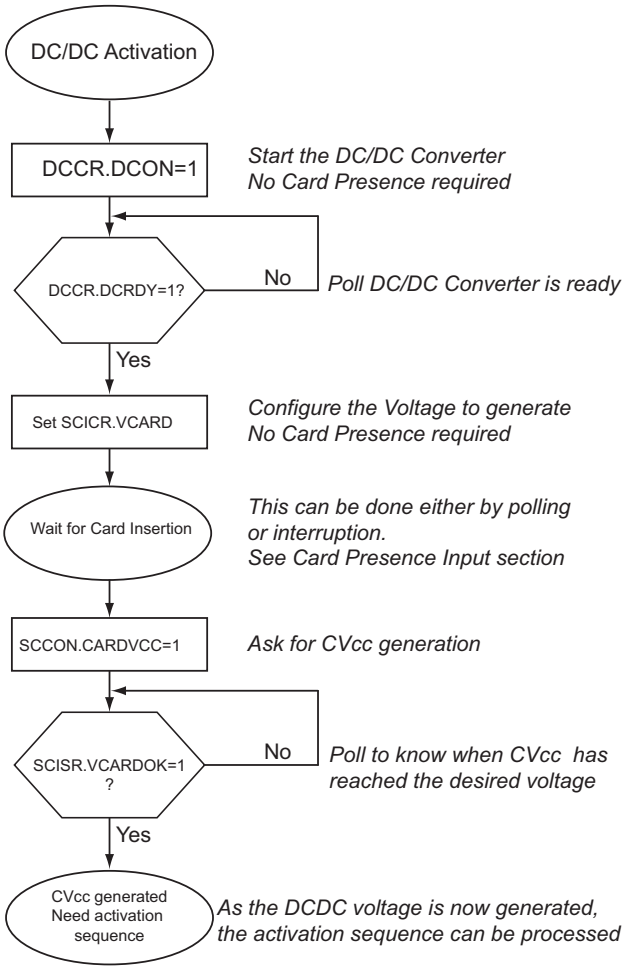
It is mandatory to switch off the DC/DC Converter before entering in Power-down mode, by clearing DCCR.DCON.

16.3.1 Initialization Procedure

To generate a specific voltage on CVcc, please follow at the following diagram:

To generate a VBUS voltage to supply a USB Smart Card, the below process is mandatory, with SCICR.VCARD = b11, to generate 5V.

Figure 16-1. DC/DC initialization procedure:



Caution

This procedure does not take into account any error signal. This should be added for full Smart Card Interface management.

16.3.2 Changing VCard Level Parameter

It is forbidden to modify the voltage delivered by CVcc while DC/DC is loaded.

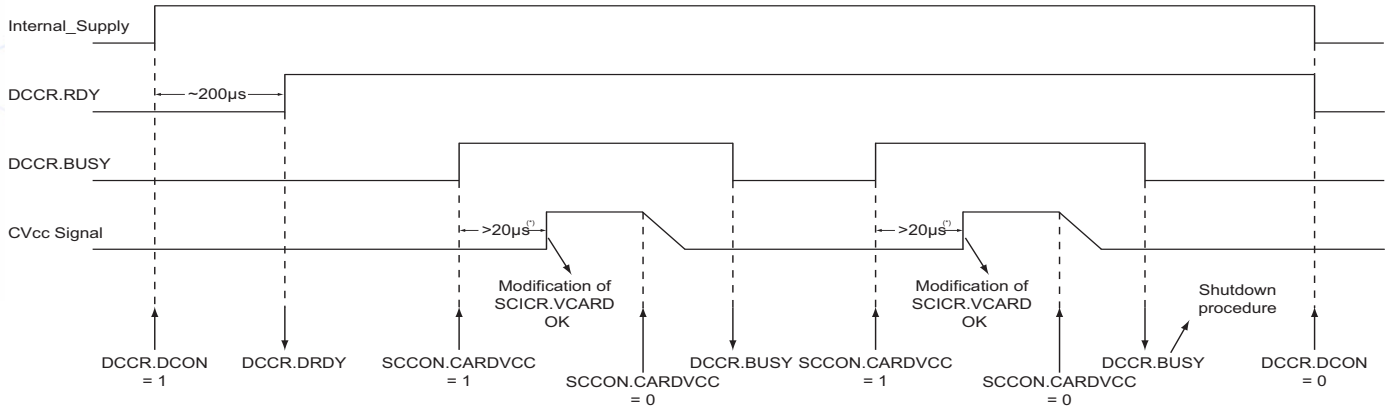
The DCCR.DCBUSY bit permits to check the state of DC/DC load. While it is set, it is advised to not:

- Shut the DC/DC Off (using DCCR.DCON bit)
- Change SCICR.VCARD[0..1] parameter

16.4 Summary: State Machine

The graph below permits to see how work the different signals of the DC/DC, and understand what is accepted, or not, regarding the moment of DC/DC use:

Figure 16-2. DC/DC State Machine



(*): This timing depends on SCICR.VCARD configuration but is, at least, 20µs

16.5 DC/DC Registers

16.5.1 DCCR - DC/DC Converter Register

Bit	7	6	5	4	3	2	1	0	
$0000EF	**DCON**	**DCRDY**	**DCBUSY**	**-**	**-**	**-**	**-**	**-**	**DCCR**
Read/write	R/W	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – DCON : DC/DC ON bit**

Set (one) this bit to start the DC/DC up. The DC/DC must be started before trying to generate a Card Vcc using Smart Card Interface.

DCCR.DCRDY bit will inform you when the DCDC is ready to use.

Clear (zero) this bit to shut the DC/DC down.

- **Bit 6 – DCRDY : DC/DC Ready bit**

This bit is cleared and set by hardware.

After having started the DC/DC using DCCR.DCON, this bit indicates when the DC/DC is ready to be used.

This bit is cleared when the DC/DC is off.

- **Bit 5 – DCBUSY : DC/DC Busy bit**

This bit is cleared and set by hardware.

This bit is set when the DC/DC is loaded and running. This means that it is advised to wait for its reset before changing parameters of DC/DC, shutting DC/DC down etc... See Figure 16-2 on page 143.

17. UART

The Universal Asynchronous serial Receiver and Transmitter (UART) is a highly flexible serial communication device.

17.1 Features

- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **High Resolution Baud Rate Generator**
- **Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits**
- **Odd or Even Parity Generation and Parity Check Supported by Hardware**
- **Data OverRun Detection**
- **Framing Error Detection**
- **Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter**
- **Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete**
- **Multi-processor Communication Mode**
- **Double Speed Asynchronous Communication Mode**

17.2 UART0

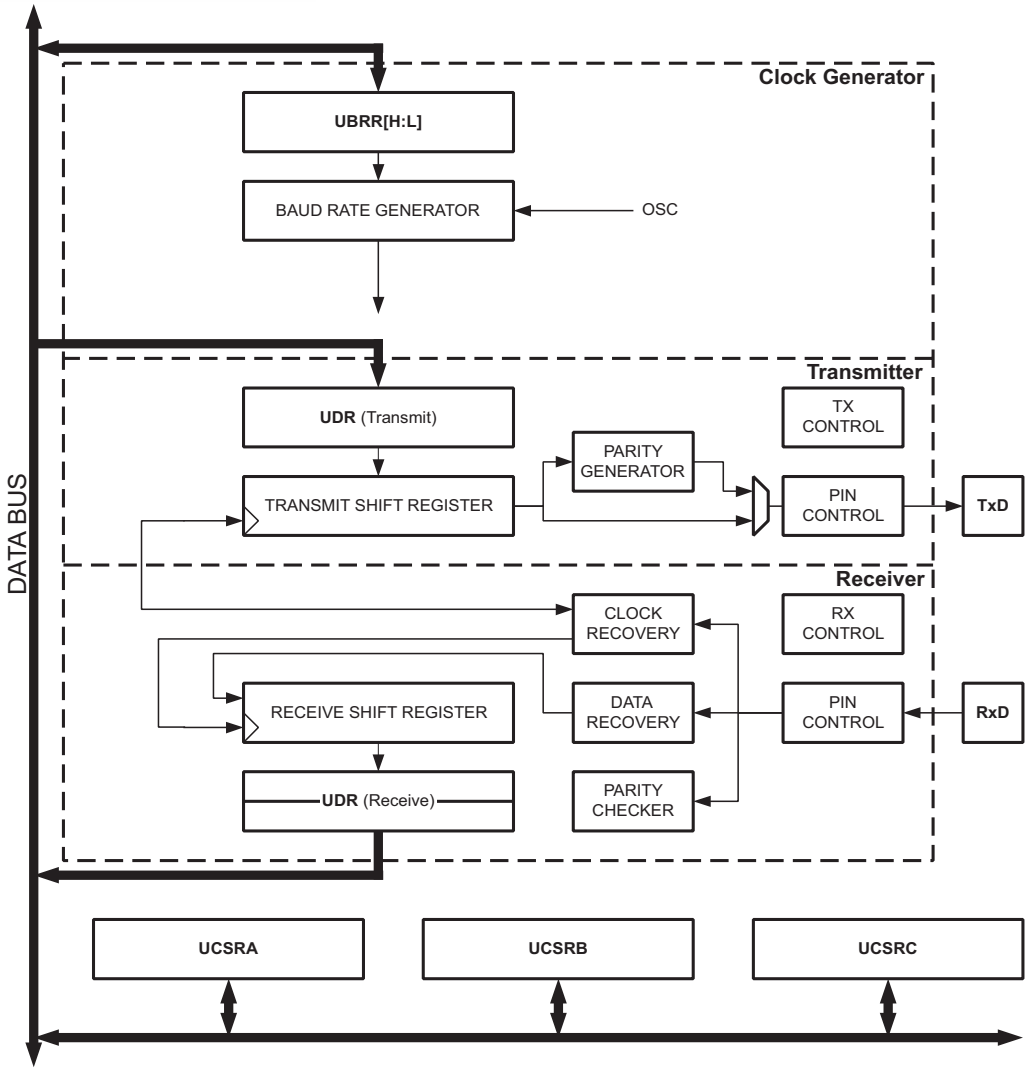
The AT90SCR075_060 has one UART, UART0.

17.3 Overview

A simplified block diagram of the UART Transmitter is shown in Figure 17-1 on page 145. CPU accessible I/O Registers and I/O pins are shown in bold.

The Power Reducion in UART0 must be disabled by writing a logical zero to PRUSART0 bit in PRR0 register.

Figure 17-1. UART Block Diagram

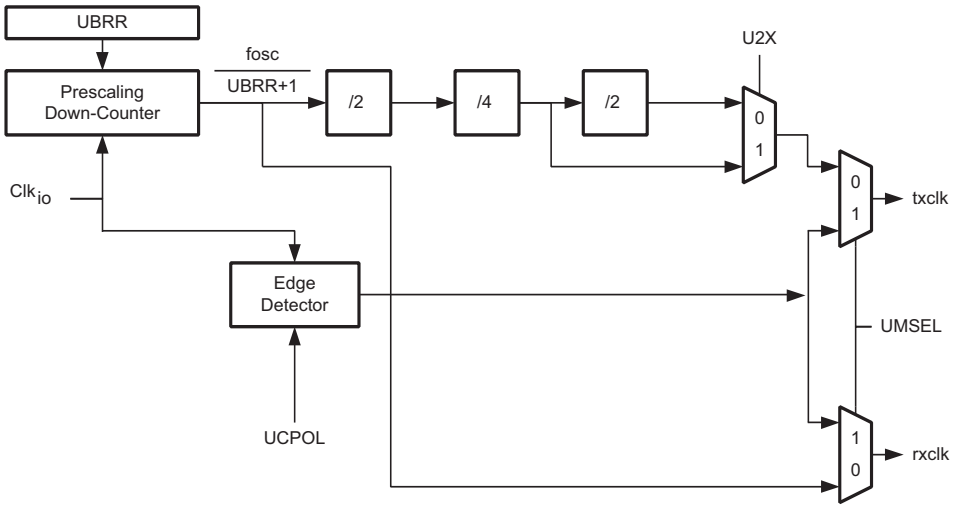


The dashed boxes in the block diagram separate the three main parts of the UART (listed from the top): Clock Generator, Transmitter and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of the baud rate generator. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the UART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register and a two level receive buffer (UDR0). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun and Parity Errors.

17.4 Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The UART0 supports two modes of clock operation: Normal asynchronous and Double Speed asynchronous mode. Double Speed is controlled by the U2X0 found in the UCSR0A Register.

Figure 17-2. Clock Generation Logic, Block Diagram



Signal description:

txclk Transmitter clock (Internal Signal).

rxclk Receiver base clock (Internal Signal).

f_{osc} UART Block clock: clk_{IO}

17.4.1 Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous master modes of operation. The description in this section refers to Figure 17-2.

The UART Baud Rate Register (UBRR0) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRR0 value each time the counter has counted down to zero or when the UBRRL0 Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRR0+1)). The Transmitter divides the baud rate generator clock output by 2, 8 or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8 or 16 states depending on mode set by the state of the U2X0 bit.

Table 17-1 contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR0 value for each mode of operation using an internally generated clock source.

Table 17-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate (1)	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X0 = 0)	$BAUD = \frac{f_{OSC}}{16(\boldsymbol{UBRR}0 + 1)}$	$\boldsymbol{UBRR}0 = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2X0 = 1)	$BAUD = \frac{f_{OSC}}{8(\boldsymbol{UBRR}0 + 1)}$	$\boldsymbol{UBRR}0 = \frac{f_{OSC}}{8BAUD} - 1$

1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System Oscillator clock frequency injected in UART block: clk_{IO}

UBRRn Contents of the UBRRH0 and UBRRL0 Registers, (0-4095)

Some examples of UBRR0 values for some system clock frequencies are found in Table 17-7 on page 166.

17.4.2 Double Speed Operation (U2X0)

The transfer rate can be doubled by setting the U2X0 bit in UCSR0A. Setting this bit only has effect for the asynchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. Note however that the Receiver will in this case only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used. For the Transmitter, there are no downsides.

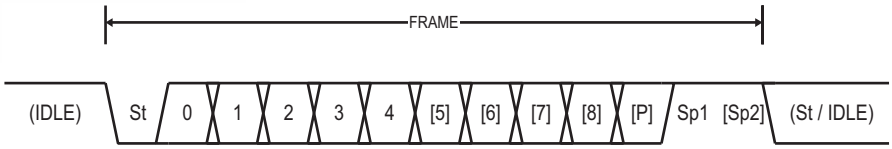
17.5 Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The UART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit followed by the least significant data bit. Then the next data bits, up to a total of nine, are succeeding, ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. Figure 17-3 illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 17-3. Frame Formats



St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxD0 or TxD0). An IDLE line must be high.

The frame format used by the UART is set by the UCSZ02:0, UPM01:0 and USBS0 bits in UCSR0B and UCSR0C. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

The UART Character SiZe (UCSZ02:0) bits select the number of data bits in the frame. The UART Parity mode (UPM01:0) bits enable and set the type of parity bit. The selection between one or two stop bits is done by the UART Stop Bit Select (USBS0) bit. The Receiver ignores the second stop bit. An FE (Frame Error) will therefore only be detected in the cases where the first stop bit is zero.

17.5.1 Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows::

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} Parity bit using even parity

P^{odd} Parity bit using odd parity

d_n Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

17.6 UART Initialization

The UART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven by UART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC0 Flag can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer.

The TXC0 Flag must be cleared before each transmission (before UDR0 is written) if it is used for this purpose.

The following simple UART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume asynchronous operation using polling (no interrupts enabled) and a fixed frame format. The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 Registers.

Assembly Code Example[1]

```
UART_Init:
  ; Set baud rate
  out  UBRRH0, r17
  out  UBRRLn0 r16
  ; Enable receiver and transmitter
  ldi  r16, (1<<RXEN0)|(1<<TXEN0)
  out  UCSR0B,r16
  ; Set frame format: 8data, 2stop bit
  ldi  r16, (1<<USBS0)|(3<<UCSZ00)
  out  UCSR0C,r16
  ret
```

C Code Example [1]

```
void UART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH0 = (unsigned char)(baud>>8);
  UBRRL0 = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);
  /* Set frame format: 8data, 2stop bit */
  UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

Note: 1. See “About Code Examples” on page 13.

More advanced initialization routines can be made that include frame format as parameters, disable interrupts and so on. However, many applications use a fixed setting of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

17.7 Data Transmission – The UART Transmitter

The UART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSR0B Register. When the Transmitter is enabled, the normal port operation of the TxD0 pin is overridden by the UART and given the function as the Transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions.

17.7.1 Sending Frames with 5 to 8 Data Bit

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDR0 I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register or U2X0 bit depending on mode of operation.

The following code examples show a simple UART transmit function based on polling of the *Data Register Empty* (UDRE0) Flag. When using frames with less than eight bits, the most significant bits written to the UDR0 are ignored. The UART has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R16

Assembly Code Example[1]

```
UART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSR0A,UDRE0
  rjmp UART_Transmit
  ; Put data (r16) into buffer, sends the data
  out  UDR0,r16
  ret
```

C Code Example [1]

```
void UART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR0A & (1<<UDRE0)) )
       ;
  /* Put data into buffer, sends the data */
  UDR0 = data;
}
```

Note: 1. See "About Code Examples" on page 13.

The function simply waits for the transmit buffer to be empty by checking the UDRE0 Flag, before loading it with new data to be transmitted. If the Data Register Empty interrupt is utilized, the interrupt routine writes the data into the buffer.

17.7.2 Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZ0 = 7), the ninth bit must be written to the TXB8 bit in UCSR0B before the low byte of the character is written to UDR0. The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example[1]

```
UART_Transmit:
  ; Wait for empty transmit buffer
  sbis UCSR0A,UDRE0
  rjmp UART_Transmit
  ; Copy 9th bit from r17 to TXB8
  cbi  UCSR0B,TXB8
  sbrc r17,0
  sbi  UCSR0B,TXB8
  ; Put LSB data (r16) into buffer, sends the data
  out  UDR0,r16
  ret
```

C Code Example [1]

```
void UART_Transmit( unsigned int data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR0A & (1<<UDRE0)) )
       ;
  /* Copy 9th bit to TXB8 */
  UCSR0B &= ~(1<<TXB8);
  if ( data & 0x0100 )
    UCSR0B |= (1<<TXB8);
  /* Put data into buffer, sends the data */
  UDR0 = data;
}
```

Note: 1. These transmit functions are written to be general functions. They can be optimized if the contents of the UCSR0B is static. For example, only the TXB8 bit of the UCSR0B Register is used after initialization. See “About Code Examples” on page 13.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

17.7.3 Transmitter Flags and Interrupts

The UART Transmitter has two flags that indicate its state: UART Data Register Empty (UDRE0) and Transmit Complete (TXC0). Both flags can be used to generate interrupts.

The Data Register Empty (UDRE0) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not already been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSR0A Register.

When the Data Register Empty Interrupt Enable (UDRIE0) bit in UCSR0B is written to one, the UART Data Register Empty Interrupt will be executed as long as UDRE0 is set (assuming that global interrupts are enabled). UDRE0 is cleared by writing UDR0. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDR0 in order to clear UDRE0 or disable the Data Register Empty interrupt, otherwise a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC0) Flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer. The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE0) bit in UCSR0B is set, the UART Transmit Complete Interrupt will be executed when the TXC0 Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC0 Flag, this is done automatically when the interrupt is executed.

17.7.4 Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UPM01 = 1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

17.7.5 Disabling the Transmitter

The disabling of the Transmitter (setting the TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 pin.

17.8 Data Reception – The UART Receiver

The UART Receiver is enabled by writing the Receive Enable (RXEN0) bit in the UCSR0B Register to one. When the Receiver is enabled, the normal pin operation of the RxD0 pin is overridden by the UART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done.

17.8.1 Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCK clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received, i.e., a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDR0 I/O location.

The following code example shows a simple UART receive function based on polling of the Receive Complete (RXC0) Flag. When using frames with less than eight bits the most significant

bits of the data read from the UDR0 will be masked to zero. The UART has to be initialized before the function can be used.

Assembly Code Example[1]

```
UART_Receive:
  ; Wait for data to be received
  sbis UCSR0A, RXC0
  rjmp UART_Receive
  ; Get and return received data from buffer
  in   r16, UDR0
  ret
```

C Code Example[1]

```
unsigned char UART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSR0A & (1<<RXC0)) )
       ;
  /* Get and return received data from buffer */
  return UDR0;
}
```

Note: 1. See “About Code Examples” on page 13.

The function simply waits for data to be present in the receive buffer by checking the RXC0 Flag, before reading the buffer and returning the value.

17.8.2 Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZ0=7) the ninth bit must be read from the RXB80 bit in UCSR0B **before** reading the low bits from the UDRn. This rule applies to the FE0, DOR0 and UPE0 Status Flags as well. Read status from UCSR0A, then data from UDR0. Reading the UDR0 I/O location will change the state of the receive buffer FIFO and consequently the TXB80, FE0, DOR0 and UPE0 bits, which all are stored in the FIFO, will change.

The following code example shows a simple UART receive function that handles both nine bit characters and the status bits.

Assembly Code Example[1]

```
UART_Receive:
  ; Wait for data to be received
  sbis UCSR0A, RXC0
  rjmp UART_Receive
  ; Get status and 9th bit, then data from buffer
  in   r18, UCSR0A
  in   r17, UCSR0B
  in   r16, UDR0
  ; If error, return -1
  andi r18,(1<<FE0)|(1<<DOR0)|(1<<UPE0)
  breq UART_ReceiveNoError
  ldi  r17, HIGH(-1)
  ldi  r16, LOW(-1)
UART_ReceiveNoError:
  ; Filter the 9th bit, then return
  lsr  r17
  andi r17, 0x01
  ret
```

C Code Example[1]

```
unsigned int UART_Receive( void )
{
  unsigned char status, resh, resl;
  /* Wait for data to be received */
  while ( !(UCSR0A & (1<<RXC0)) )
       ;
  /* Get status and 9th bit, then data */
  /* from buffer */
  status = UCR0A;
  resh = UCSR0B;
  resl = UDR0;
  /* If error, return -1 */
  if ( status & (1<<FE0)|(1<<DOR0)|(1<<UPE0) )
    return -1;
  /* Filter the 9th bit, then return */
  resh = (resh >> 1) & 0x01;
  return ((resh << 8) | resl);
}
```

Note: 1. See “About Code Examples” on page 13.

The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

17.8.3 Receive Compete Flag and Interrupt

The UART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXC0) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXEN0 = 0), the receive buffer will be flushed and consequently the RXC0 bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE0) in UCSR0B is set, the UART Receive Complete interrupt will be executed as long as the RXC0 Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR0 in order to clear the RXC0 Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

17.8.4 Receiver Error Flags

The UART Receiver has three Error Flags: Frame Error (FE0), Data OverRun (DOR0) and Parity Error (UPE0). All can be accessed by reading UCSR0A. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSR0A must be read before the receive buffer (UDR0), since reading the UDR0 I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSR0A is written for upward compatibility of future UART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FE0) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE0 Flag is zero when the stop bit was correctly read (as one), and the FE0 Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE0 Flag is not affected by the setting of the USBS0 bit in UCSR0C since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSR0A.

The Data OverRun (DOR0) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. If the DOR0 Flag is set there was one or more serial frame lost between the frame last read from UDR0, and the next frame read from UDR0. For compatibility with future devices, always write this bit to zero when writing to UCSR0A. The DOR0 Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPE0) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE0 bit will always be read zero. For compatibility with future devices, always set this bit to zero when writing to UCSR0A. For more details see “Parity Bit Calculation” on page 148 and “Parity Checker” on page 155.

17.8.5 Parity Checker

The Parity Checker is active when the high UART Parity mode (UPM01) bit is set. Type of Parity Check to be performed (odd or even) is selected by the UPM00 bit. When enabled, the Parity Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together

with the received data and stop bits. The Parity Error (UPE0) Flag can then be read by software to check if the frame had a Parity Error.

The UPE0 bit is set if the next character that can be read from the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read.

17.8.6 Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., the RXEN0 is set to zero) the Receiver will no longer override the normal function of the RxD0 port pin. The Receiver buffer FIFO will be flushed when the Receiver is disabled. Remaining data in the buffer will be lost

17.8.7 Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDR0 I/O location until the RXC0 Flag is cleared. The following code example shows how to flush the receive buffer.

Assembly Code Example(1)

```
UART_Flush:
  sbis UCSR0A, RXC0
  ret
  in   r16, UDR0
  rjmp UART_Flush
```

C Code Example(1)

```
void UART_Flush( void )
{
  unsigned char dummy;
  while ( UCSR0A & (1<<RXC0) ) dummy = UDR0;
}
```

Note: 1. See “About Code Examples” on page 13.

17.9 Asynchronous Data Reception

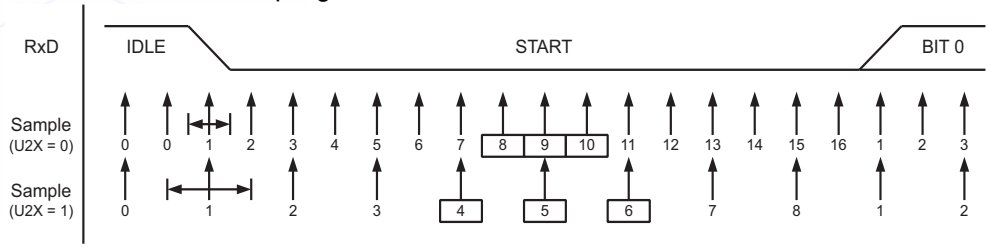
The UART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD0 pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

17.9.1 Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. Figure 17-4 illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16 times the baud rate for Normal mode, and eight times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation is obtained when using the Double Speed mode (U2X0 = 1) of operation.

Samples denoted zero are samples done when the RxD0 line is idle (i.e., no communication activity).

Figure 17-4. Start Bit Sampling

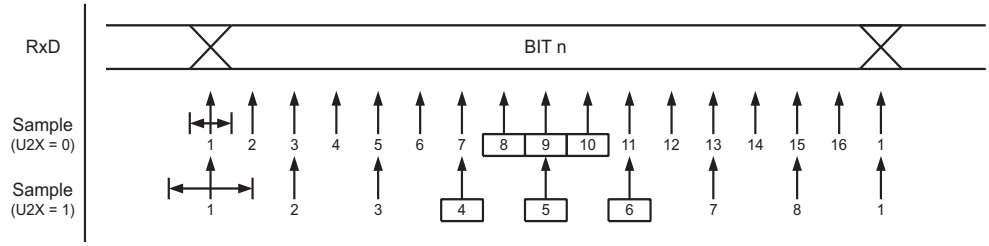


When the clock recovery logic detects a high (idle) to low (start) transition on the RxD0 line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

17.9.2 Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. Figure 17-5 shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

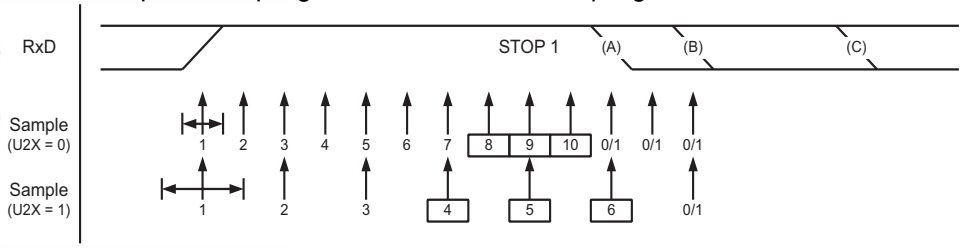
Figure 17-5. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit. The center samples are emphasized on the figure by having the sample number inside boxes. The majority voting process is done as follows: If two or all three samples have high levels, the received bit is registered to be a logic 1. If two or all three samples have low levels, the received bit is registered to be a logic 0. This majority voting process acts as a low pass filter for the incoming signal on the RxD0 pin. The recovery process is then repeated until a complete frame is received. Including the first stop bit. Note that the Receiver only uses the first stop bit of a frame.

Figure 17-6 shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 17-6. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic 0 value, the Frame Error (FE0) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be at point marked (A) in Figure 17-6. For Double Speed mode the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

17.9.3 Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar (see Table 17-2) base frequency, the Receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

Table 20-1 .

$$R_{slow} = \frac{(D+1)S}{S-1+D\cdot S+S_F} \qquad R_{fast} = \frac{(D+2)S}{(D+1)S+S_M}$$

D	Sum of character size and parity size (D = 5 to 10 bit)
S	Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
S_F	First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for Double Speed mode.
S_M	Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for Double Speed mode.
R_{slow}	is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

Table 17-2 and Table 17-3 list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 17-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2Xn = 0)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

Table 17-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2Xn = 1)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the Receiver and Transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The Receiver's system clock (XTAL) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator the system clock may differ more than 2% depending of the resonators tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRR value that gives an acceptable low error can be used if possible.

17.10 Multi-processor Communication Mode

Setting the Multi-processor Communication mode (MPCM0) bit in UCSR0A enables a filtering function of incoming frames received by the UART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCM0 setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the Receiver is set up for frames with

nine data bits, then the ninth bit (RXB8n) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is one, the frame contains an address. When the frame type bit is zero the frame is a data frame.

The Multi-processor Communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

17.10.1 Using MPCM0

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZ0 = 7). The ninth bit (TXB80) must be set when an address frame (TXB80 = 1) or cleared when a data frame (TXB = 0) is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-processor Communication mode:

1. All Slave MCUs are in Multi-processor Communication mode (MPCM0 in UCSR0A is set).
2. The Master MCU sends an address frame, and all slaves receive and read this frame. In the Slave MCUs, the RXC0 Flag in UCSR0A will be set as normal.
3. Each Slave MCU reads the UDR0 Register and determines if it has been selected. If so, it clears the MPCM0 bit in UCSR0A, otherwise it waits for the next address byte and keeps the MPCM0 setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other Slave MCUs, which still have the MPCM0 bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM0 bit and waits for a new address frame from master. The process then repeats from 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the Transmitter and Receiver uses the same character size setting. If 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit (USBS0 = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM0 bit. The MPCM0 bit shares the same I/O location as the TXC0 Flag and this might accidentally be cleared when using SBI or CBI instructions.

17.11 UART Register Description

17.11.1 UDR0 – UART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	UDR0
$0000C6	RXB[7..0]								(Read)
$0000C6	TXB[7..0]								(Write)
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

The UART Transmit Data Buffer Register and UART Receive Data Buffer Registers share the same I/O address referred to as UART Data Register or UDR0. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR0 Register location. Reading the UDR0 Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE0 Flag in the UCSR0A Register is set. Data written to UDR0 when the UDRE0 Flag is not set, will be ignored by the UART Transmitter. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxD0 pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

17.11.2 UCSR0A – UART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
$0000C0	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/write	R	R/W	R	R	R	R	R/W	R/W	
Initial value	0	0	1	0	0	0	0	0	0x20

- **Bit 7 – RXC0: UART Receive Complete**

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

- **Bit 6 – TXC0: UART Transmit Complete**

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

- **Bit 5 – UDRE0: UART Data Register Empty**

The UDRE0 flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit).

UDRE0 is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FE0: Frame Error**

This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FE0 bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

- **Bit 3 – DOR0: Data OverRun**

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 2 – UPE0: UART Parity Error**

This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

- **Bit 1 – U2X0: Double the UART Transmission Speed**

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCM0: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCM0 bit is written to one, all the incoming frames received by the UART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM0 setting. For more detailed information see “Multi-processor Communication Mode” on page 159.

17.11.3 UCSR0B – UART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
$0000C1	**RXCIE0**	**TXCIE0**	**UDRIE0**	**RXEN0**	**TXEN0**	**UCSZ02**	**RXB80**	**TXB80**	**UCSR0B**
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 – RXCIE0: RX Complete Interrupt Enable 0**

Writing this bit to one enables interrupt on the RXC0 Flag. A UART Receive Complete interrupt will be generated only if the RXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC0 bit in UCSR0A is set.

- **Bit 6 – TXCIE0: TX Complete Interrupt Enable 0**

Writing this bit to one enables interrupt on the TXC0 Flag. A UART Transmit Complete interrupt will be generated only if the TXCIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC0 bit in UCSR0A is set.

- **Bit 5 – UDRIE0: UART Data Register Empty Interrupt Enable 0**

Writing this bit to one enables interrupt on the UDRE0 Flag. A Data Register Empty interrupt will be generated only if the UDRIE0 bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE0 bit in UCSR0A is set.

- **Bit 4 – RXEN0: Receiver Enable 0**

Writing this bit to one enables the UART Receiver. The Receiver will override normal port operation for the RxD0 pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE0, DOR0, and UPE0 Flags.

- **Bit 3 – TXEN0: Transmitter Enable 0**

Writing this bit to one enables the UART Transmitter. The Transmitter will override normal port operation for the TxD0 pin when enabled. The disabling of the Transmitter (writing TXENn0to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxD0 port.

- **Bit 2 – UCSZ02: Character Size 0**

The UCSZ02 bits combined with the UCSZ01:0 bit in UCSR0C sets the number of data bits (Character size) in a frame the Receiver and Transmitter use.

- **Bit 1 – RXB80: Receive Data Bit 8 0**

RXB80 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDR0.

- **Bit 0 – TXB80: Transmit Data Bit 8 0**

TXB80 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDR0.

17.11.4 UCSR0C – UART Control and Status Register C

Bit	7	6	5	4	3	2	1	0	
$0000C2	**-**	**-**	**UPM01**	**UPM00**	**USBS0**	**UCSZ01**	**UCSZ00**	**-**	**UCSR0C**
Read/write	R	R	R/W	R/W	R/W	R/W	R/W	R	
Initial value	0	0	0	0	0	1	1	0	0x06

- **Bits 7..6 – Reserved Bits**

These bits are reserved for future use.

- **Bits 5..4 – UPM01..0: Parity Mode**

These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM0 setting. If a mismatch is detected, the UPE0 Flag in UCSR0A will be set.

Table 17-4. UPM0 Bits Settings

UPM01	UPM00	Parity Mode
0	0	Disabled

Table 17-4. UPM0 Bits Settings

UPM01	UPM00	Parity Mode
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBS0: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 17-5. USBS Bit Settings

USBS0	Stop Bit(s)
0	1-bit
1	2-bit

- **Bit 2:1 – UCSZ01:0: Character Size**

The UCSZ01:0 bits combined with the UCSZ02 bit in UCSR0B sets the number of data bits (Character size) in a frame the Receiver and Transmitter use.

Table 17-6. UCSZ0 Bits Settings

UCSZ02	UCSZ01	UCSZ00	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 : Reserved Bit**

This bit is reserved for future use.

17.11.5 UBRR0L and UBRR0H – UART Baud Rate Registers

Bit	7	6	5	4	3	2	1	0	
$0000C5	-	-	-	-	**UBRR[11..8]**				**UBRRH0**
$0000C4	**UBRR[7..0]**								**UBRRL0**
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00

- **Bit 15..12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11..0 – UBRR11..0: UART Baud Rate Register**

This is a 12-bit register which contains the UART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the UART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

17.12 Examples of Baud Rate Setting

For AT90SCR075_060 frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings in Table 17-7 and Table 17-8. UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see “Asynchronous Operational Range” on page 158). The error values are calculated using the following equation:

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Table 17-7. Examples of UBRR0 Settings accessible Oscillator Frequencies

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 16 MHz				f_{osc} = 19.2 MHz			
	U2X0 = 0		U2X0 = 1		U2X0 = 0		U2X0 = 1		U2X0 = 0		U2X0 = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	416	-0.1%	832	0%	499	0%	999	0%
4800	103	0.2%	207	0.2%	207	0.2%	416	-0.1%	249	0%	499	0%
9600	51	0.2%	103	0.2%	103	0.2%	207	0.2%	124	0%	249	0%
14.4k	34	-0.8%	68	0.6%	68	0.6%	138	-0.1%	82	0.4%	166	-0.2%
19.2k	25	0.2%	51	0.2%	51	0.2%	103	0.2%	62	-0.8%	124	0%
28.8k	16	2.1%	34	-0.8%	34	-0.8%	68	0.6%	41	-0.8%	82	0.4%
38.4k	12	0.2%	25	0.2%	25	0.2%	51	0.2%	30	0.8%	62	-0.8%
57.6k	8	-3.5%	16	2.1%	16	2.1%	34	-0.8%	20	-0.8%	41	-0.8%
76.8k	6	-7%	12	0.2%	12	0.2%	25	0.2%	15	2.3%	30	0.8%
115.2k	3	8.5%	8	-3.5%	8	-3.5%	16	2.1%	9	4.2%	20	-0.8%
230.4k	1	8.5%	3	8.5%	3	8.5%	8	-3.5%	4	4.2%	9	4.2%
250k	1	0%	3	0%	3	0%	7	0%	4	-4%	9	-4%
500k	0	0%	1	0%	1	0%	3	0%	1	20%	4	-4%
1M	-	-	0	0%	0	0%	1	0%	0	20%	1	20%
1.5M	-	-	-	-	-	-	0	33.3%	-	-%	1	-20%
2M	-	-	-	-	-	-	0	0%	-	-%	0	20%
Max.(1)	500kbps		1Mbps		1Mbps		2Mbps		1.2Mbps		2.4Mbps	

Note: 1. UBRR = 0, Error = 0.0%

Table 17-8. Examples of UBRR0 Settings accessible Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 20 MHz			
	U2X0 = 0		U2X0 = 1	
	UBRR	Error	UBRR	Error
2400	520	0%	1041	0%
4800	259	0.2%	520	0%
9600	129	0.2%	259	0.2%
14.4k	86	-0.2%	173	-0.2%
19.2k	64	0.2%	129	0.2%
28.8k	42	0.9%	86	-0.2%
38.4k	32	-1.4%	64	0.2%
57.6k	21	-1.4%	42	0.9%
76.8k	15	1.7%	32	-1.4%
115.2k	10	-1.4%	21	-1.4%
230.4k	4	8.5%	10	-1.4%
250k	4	0%	9	0%
500k	2	-16.7%	4	0%
1M	0	25%	2	-16.7%
1.5M			1	-16.7%
2M			0	25%
Max.(1)	1.25Mbps		2.5Mbps	

Note: 1. UBRR = 0, Error = 0.0%

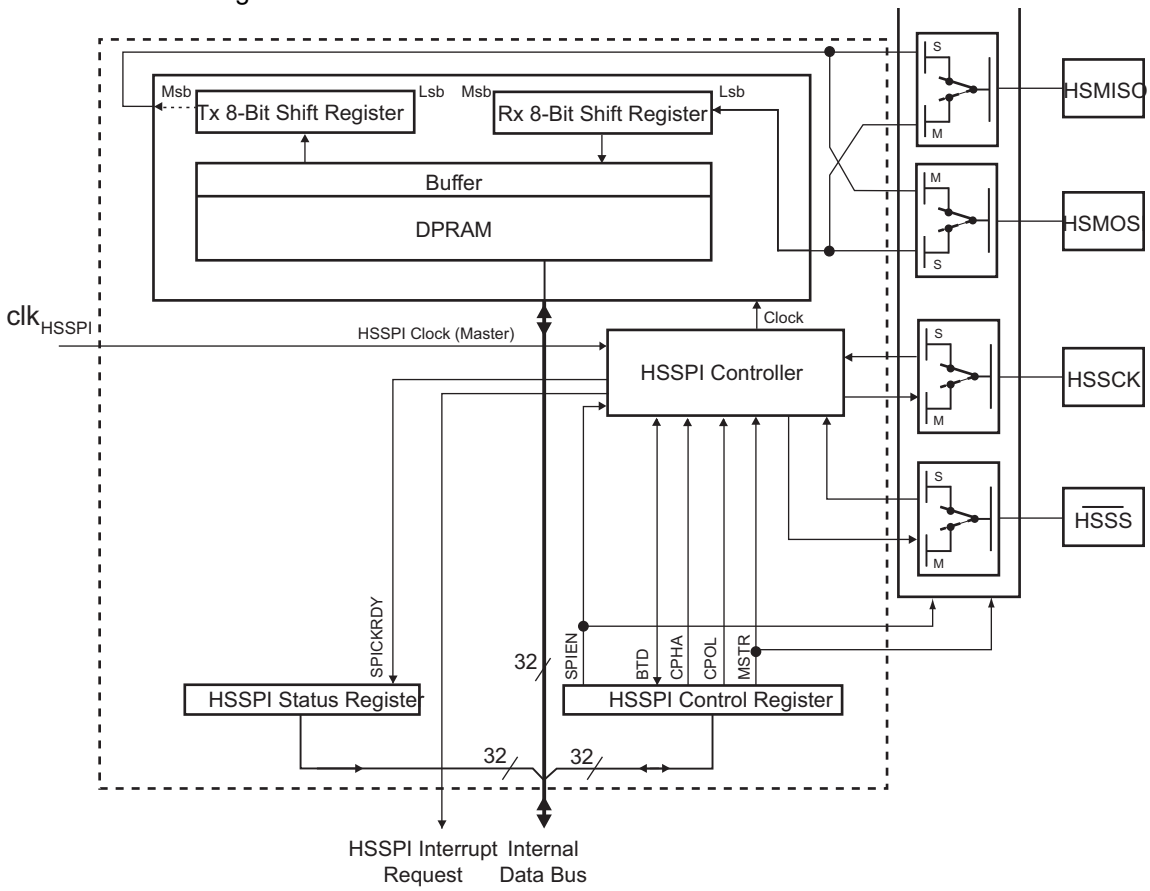
18. High-Speed SPI Controller

This High Speed SPI (HSSPI) Interface comes with 4 dedicated pads (HSMISO, HSMOSI, HSSCK, $\overline{\text{HSSS}}$).

18.1 Features

- **Support clock up to 20Mhz in Master and Slave Mode**
- **Full-duplex, 4-wire Synchronous Data Transfer**
- **Master or Slave Operation**
- **Transmission / Reception**
- **Three sources of Interrupt: Byte Transmitted, Time-out and Reception Overflow**
- **Specific DMA for fast copy from internal DPRAM to RAM**
- **4 DPRAM buffers of 16 bytes each: 2 for Reception and 2 for Transmission**
- **Internal Double Buffering for high performance**
- **Programmable clock and inter-bytes (guardtime) delays**

Figure 18-1. HSSPI Block Diagram



18.2 Description

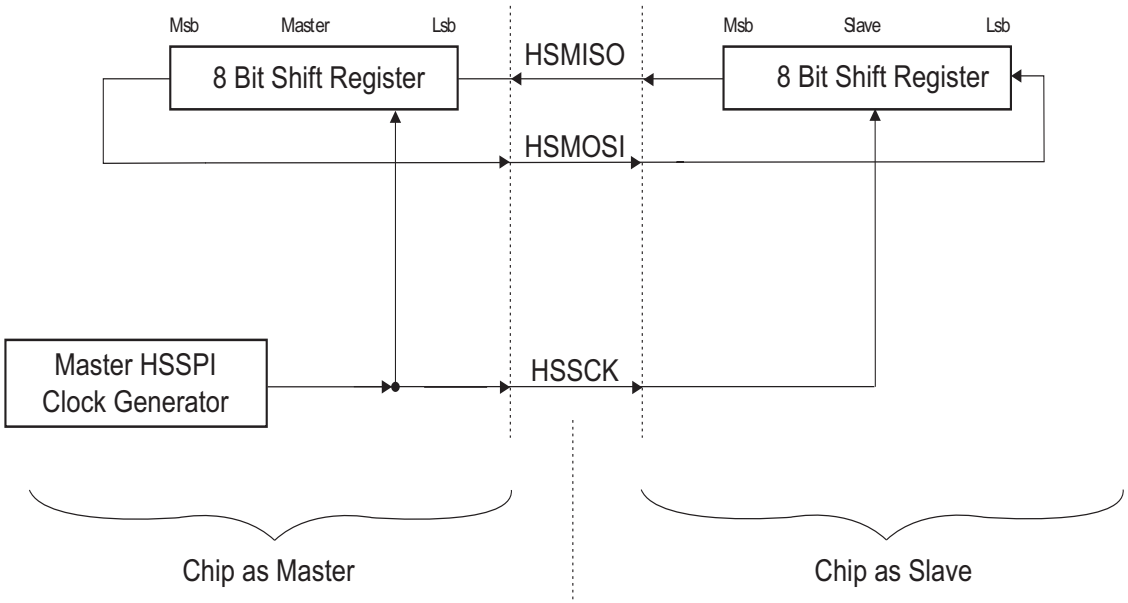
The interconnection between Master and Slave CPUs with HSSPI is shown in Figure 18-2.

The HSSPI Controller is enabled by setting the HSSPICFG.HSSPIEN bit (by default, the HSSPI Controller is in Slave mode) and fully operational when the HSSPISR.SPICKRDY bit is set. To select the frequency of the HSSPI clock, the HSSPICFG.SPICKDIV bits (described in page 178)

must be configured before enabling the HSSPI Controller. Any attempt at modifying the HSSPI Clock frequency once the HSSPI Controller is enabled will not affect it. The HSSPICFG.HSMSTR bit selects either Master or Slave operations. The HSSPI Master initiates the communication cycle after pulling low the $\overline{\text{HSSS}}$ line of the desired Slave.

The Master and Slave prepare the data to be sent using their respective shift registers or using the internal DPRAM (according to the HSSPICFG.DPRAM bit), and the Master generates the required clock pulses on the HSSCK line to exchange data. Data is always shifted from Master to Slave on the Master Out - Slave In, HSMOSI line, and from Slave to Master on the Master In - Slave Out, HSMISO line, simultaneously.

Figure 18-2. HSSPI Master - Slave Interconnection



The DPRAM stores two 16 Bytes buffers for transmission and two 16 Bytes buffers for reception.

18.2.1 HSSPI Controller Configured as a Master

When configured as a Master, the HSSPI interface has no automatic control of the $\overline{\text{HSSS}}$ line. This must be handled by software before starting the communication.

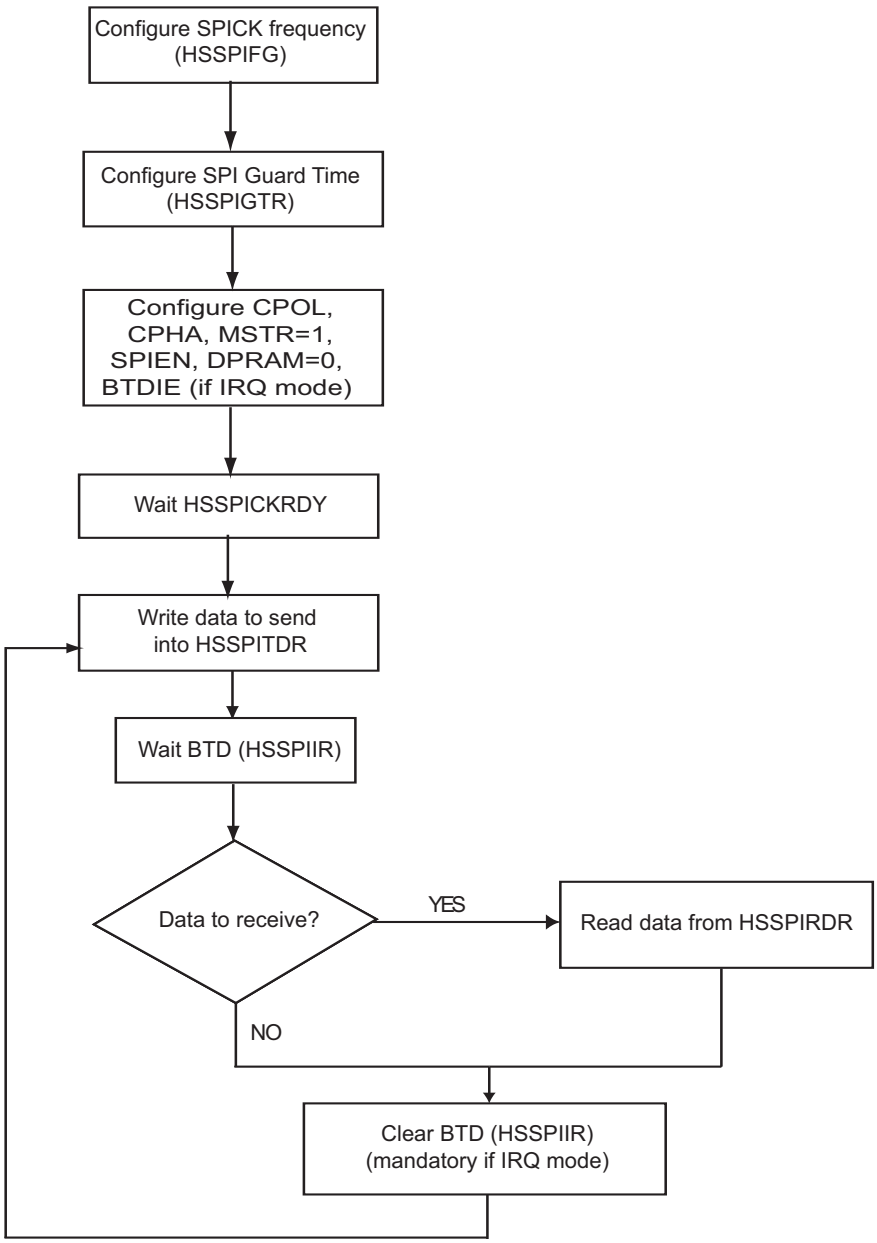
18.2.1.1 One byte sending using shift register (without DPRAM)

When the DPRAM is not activated (HSSPICFG.DPRAM bit cleared) and a byte is written to the HSSPI Transmit Data Register (HSSPITDR register), the hardware shifts the 8 bits (MSB first) into the Slave and receives a byte from the Slave simultaneously. After shifting one byte, the Byte Transfer Done flag (HSSPIIR.BTD) is set. If global interrupt and the HSSPIIER.BTDIE bit have been previously set, an interrupt is generated.

At this stage, the Master can read the HSSPIRDR register to retrieve the byte sent by the other device. The HSSPIIR.BTD flag must then be cleared by software.

The figure below illustrates the way the HSSPI works when configured as a Master without DPRAM mode.

Figure 18-3. HSSPI - Master without DPRAM operation



18.2.1.2 Buffer sending with DPRAM and HSSPIDMA (DPRAM mode)

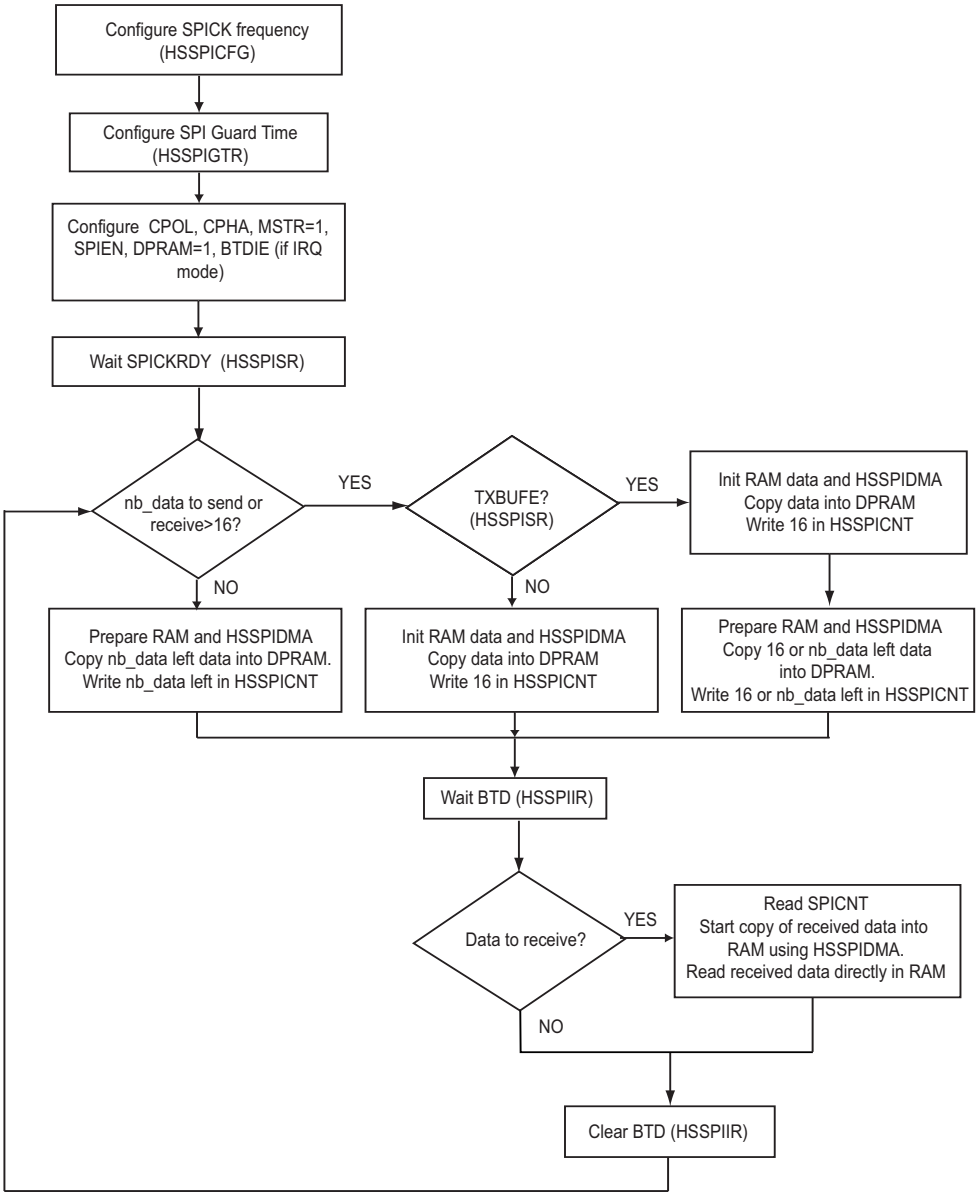
When the DPRAM is activated (HSSPICFG.DPRAM bit set), the data to be sent must be written to one of the internal DPRAM buffers using the HSSPIDMA. To do so, use the HSSPIDMA mechanism to copy HSSPIDMAB bytes from RAM to DPRAM. It is not possible to copy more than 16 bytes per copy. The DPRAM targeted by the copy is totally transparent to the user. Once this is done, the number of bytes to be transmitted is written to the HSSPICNT register. The HSSPI interface will start to shift this number of bytes from the master to the slave and from the slave to the master simultaneously. If the HSSPISR.TXBUFE = 1, new data can be copied in the second buffer using HSSPIDMA again. After shifting all the bytes of one buffer, the Byte Transfer Done flag (HSSPIIR.BTD) is set. If the HSSPI interrupt enable bit and the HSSPI-IER.BTDIE bit have been previously set, an interrupt is generated.

At this point, the Master can copy the received data from reception DPRAM into RAM, using HSSPIDMA. The HSSPIIR.BTD flag must then be cleared by software.

If the HSSPIIR.BTD is set and the next buffer becomes full, the HSSPIIR.RCVOF is set. If the HSSPI interrupt enable bit and the HSSPIIER.RCVOFIE bit have been previously set, an interrupt is generated.

The figure below illustrates the way the HSSPI works when configured as a Master with DPRAM/HSSPIDMA.

Figure 18-4. SPI - Master with DPRAM/HSSPIDMA activated



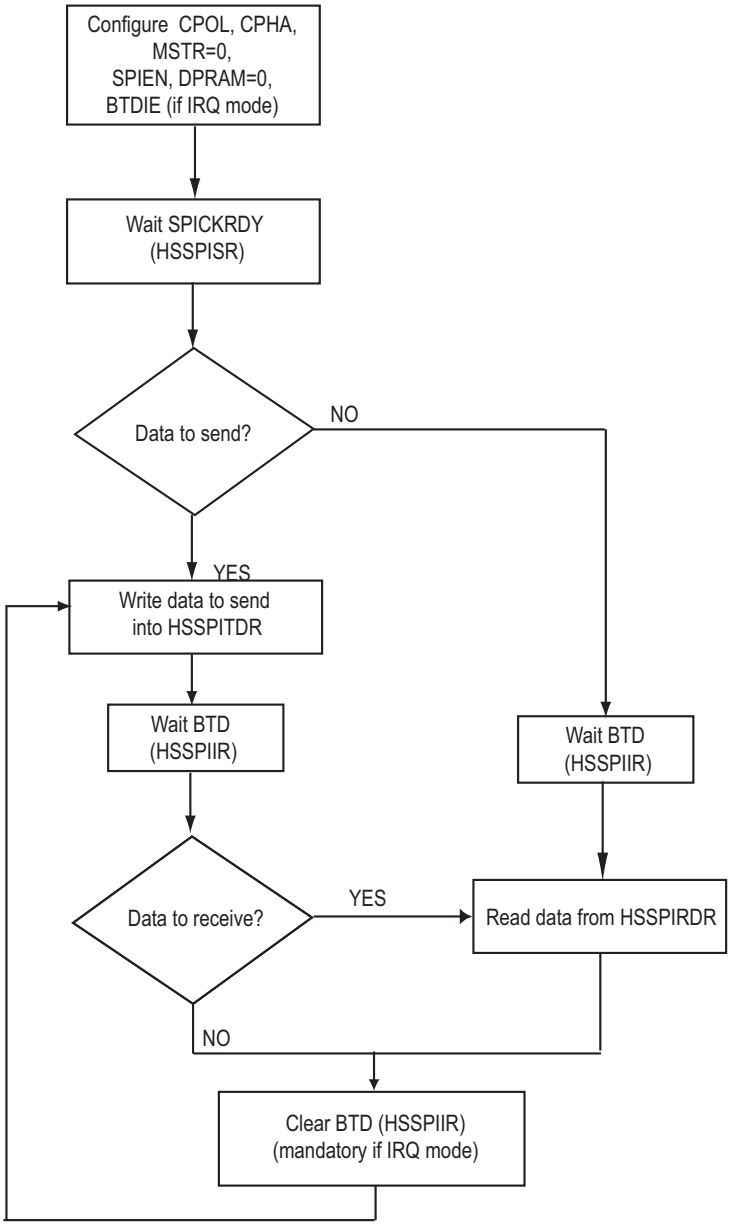
18.2.2 SPI Controller Configured as a Slave

18.2.2.1 One byte sending using shift register (without DPRAM)

If the DPRAM is not activated (HSSPICFG.DPRAM bit cleared), the software may update the contents of the HSSPI Transmit Data Register, SPITDR, that will be shifted out by incoming clock pulses on the clock pin. As soon as the byte has been completely shifted, the Byte transfer Done flag (HSSPIIR.BTD) is set. If global interrupt and the HSSPIIER.BTDIE bit have been previously set, an interrupt is generated.

At this stage, the Slave can read the HSSPIRDR register to retrieve the byte sent by the other device. The HSSPICFG.BTD flag must then be cleared by software.

Figure 18-5. HSSPI - Slave without DPRAM activated



18.2.2.2 *Buffer sending with DPRAM and HSSPIDMA (DPRAM mode)*

When the DPRAM is activated (HSSPICFG.DPRAM bit set), the data to be sent must be written to one of the internal DPRAM buffers using HSSPIDMA. Once this is done, the number of bytes to be transmitted is written to the HSSPICNT register. When the master applies the clock, this number of bytes is shifted from the master to the slave and from the slave to the master simultaneously. At this point, new data can be stored in the second buffer if the HSSPISR.TXBUFE = 1. Each time all the data in one of the buffers is sent, the Byte Transfer Done flag (HSSPIIR.BTD) is set. If global interrupt and the HSSPIIER.BTDIE bit have been previously set, an interrupt is generated.

At this stage, the incoming data can be copied from internal DPRAM to RAM using HSSPIDMA. The HSSPIIR.BTD flag must then be cleared by software.

If the HSSPIIR.BTD is set and the next buffer becomes full, the HSSPIIR.RCVOF is set. If global interrupt and the HSSPIIER.RCVOFIE bit have been previously set, an interrupt is generated.

The figures below (Figure 18-6, Figure 18-7 & Figure 18-8) illustrate the way the SPI works when configured as a Slave with DPRAM mode.

Figure 18-6. SPI - Slave with DPRAM operation - First part

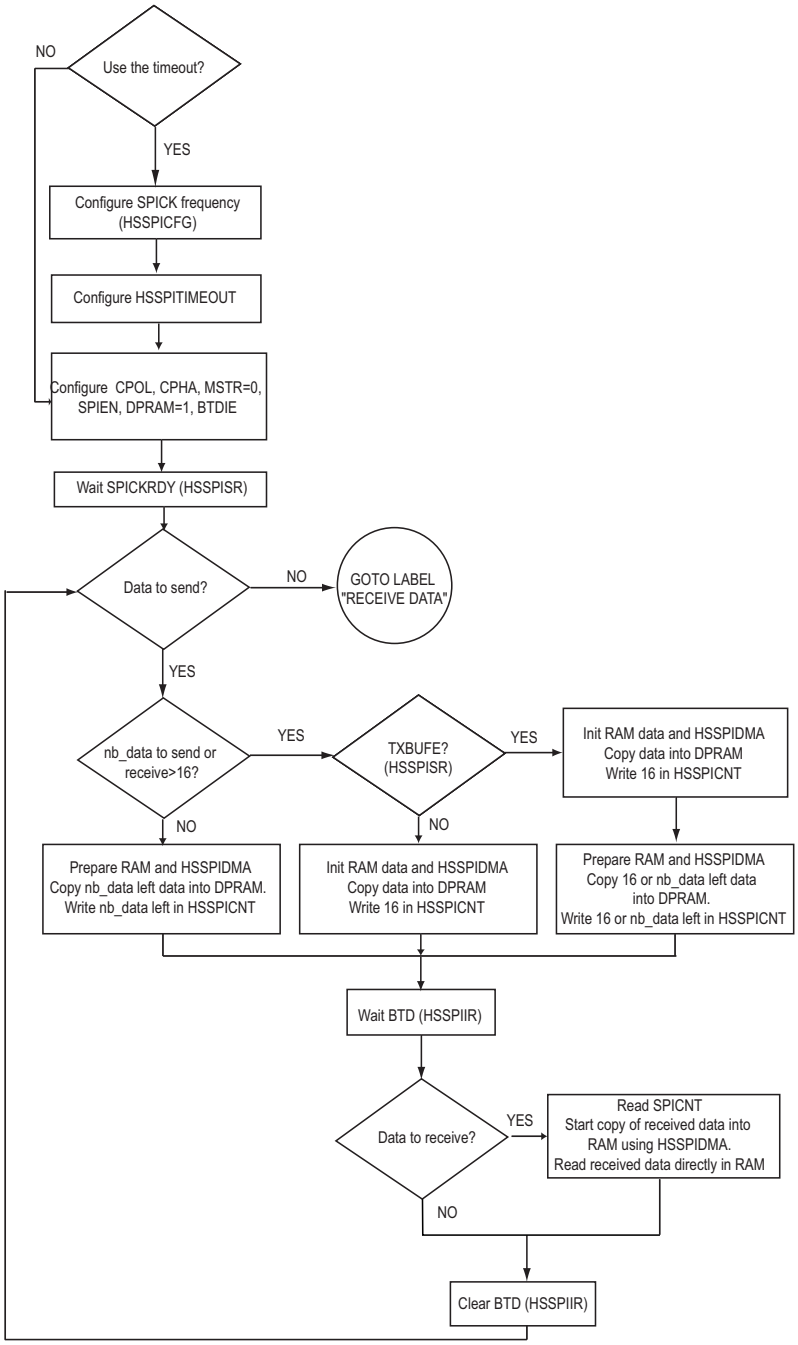
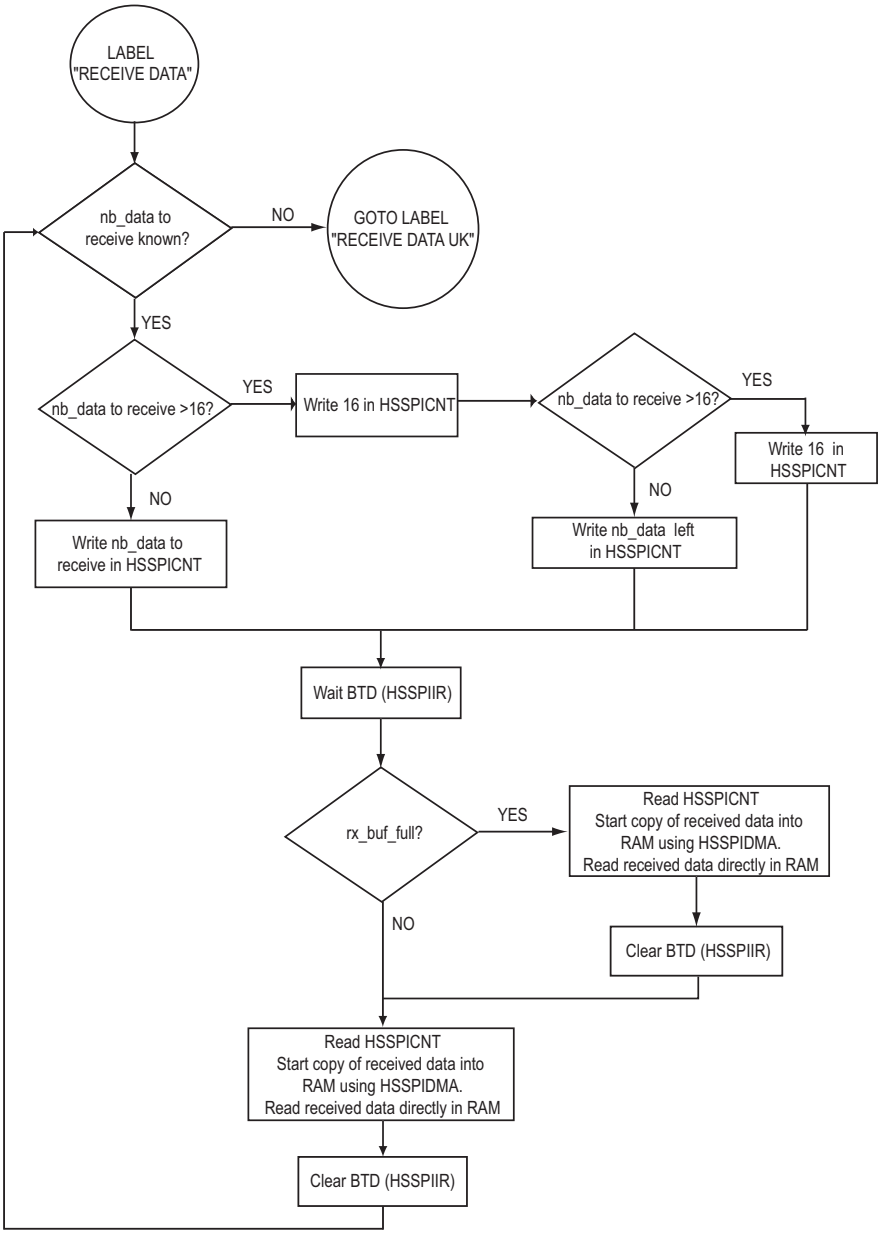
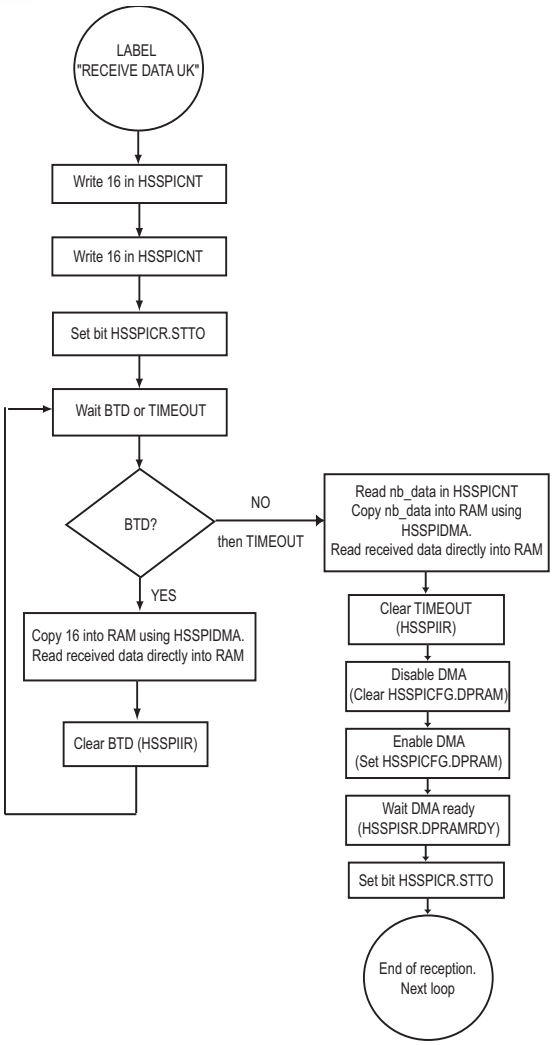


Figure 18-7. SPI - Slave with DPRAM operation - Second part



If the number of bytes to be received or sent is not known, a timeout can be used. It can be activated by entering a time bit value in the HSSPITIMEOUT and setting the HSSPICR.STTTO bit. As the amount of data is unknown, the two registers (HSSPICNT) have to be set to the maximum value, 16. When the timeout occurs, the HSSPIIR.TIMEOUT bit is set. If the HSSPI interrupt enable bit and the HSSPIIER.TIMEOUTIE bit have been previously set, an interrupt is generated. The number of byte received is available in the HSSPICNT register. Clearing the HSSPIIR.TIMEOUT bit by soft or by setting the HSSPICR.STTTO bit will update the internal DPRAM

Figure 18-8. SPI - Slave with DPRAM operation - Third part



In SPI slave mode, the control logic will sample the incoming signal of the HSSCK pin. To ensure correct sampling of the clock signal, the frequency of the HSSPI clock should never exceed f_{SCKmax} (f_{SCKmax}=24 Mbits/s). When the HSSPI is enabled, the data direction of the HSMOSI, HSMISO, HSSCK, and $\overline{HSSS}$ pins is overridden according to Table 18-1

.

Table 18-1. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	Output	Input
MISO	Input	Output
SCK	Output	Input
$\overline{NSS}$	Output	Input

18.3 $\overline{\text{HSSS}}$ line Functionality

18.3.1 Slave Mode

When the HSSPI is configured as a Slave (HSSPICFG.HSMSTR bit cleared), the Slave Select ($\overline{\text{HSSS}}$) line is driven by the master device. The $\overline{\text{HSSS}}$ has no effect on the HSSPI logic. It is the software's responsibility to manage the $\overline{\text{HSSS}}$ level.

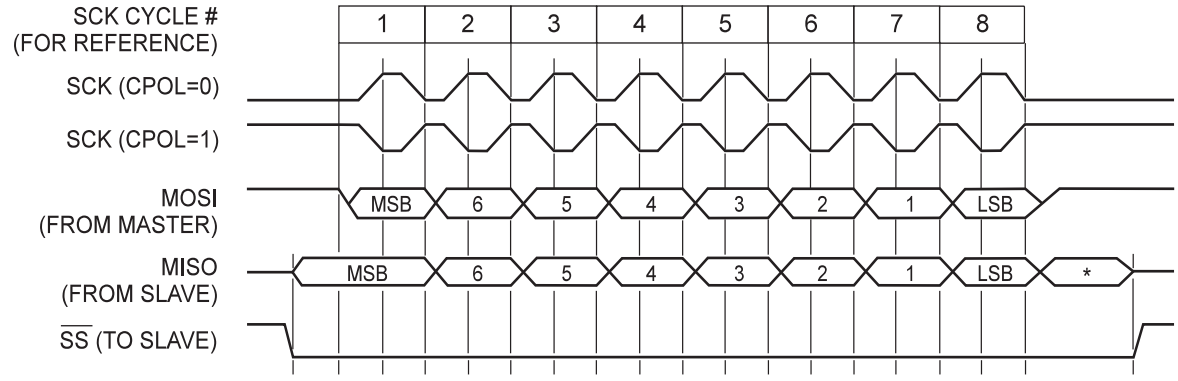
18.3.2 Master Mode

When the HSSPI is configured as a Master (HSSPICFG.HSMSTR bit set), the $\overline{\text{HSSS}}$ line is used to select the slave.

18.4 Data Modes

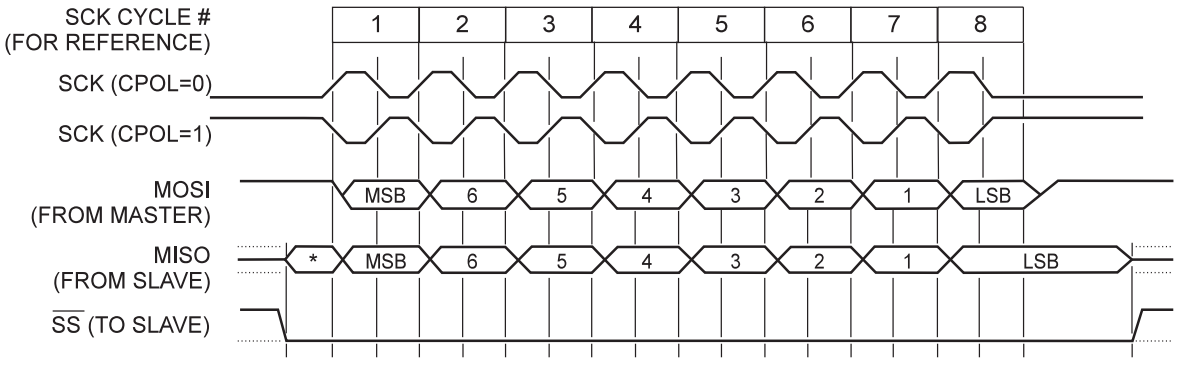
There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 18-9 and Figure 18-10.

Figure 18-9. SPI Transfer Format with CPHA = 0



* Not defined but normally MSB of character just received

Figure 18-10. SPI Transfer Format with CPHA = 1



* Not defined but normally LSB of previously transmitted character

18.5 HSSPI Interface Registers

18.5.1 HSSPICFG - HSSPI Config Register

Bit	7	6	5	4	3	2	1	0	
$0000D9	SPICKDIV [2..0]			DPRAM	HSCPHA	HSCPOL	HSMSTR	HSSPIEN	HSSPICFG
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	0	0	0	0	0	0	0	0x80

- **Bits 7..5 - SPICKDIV2..0: SPI Clock Divider Ratio Bits**

Defines the HSSPI Clock divider ratio.

The following table gives, for each combination of SPICKDIV2, SPICKDIV1 and SPICKDIV0, the division applied to the clock. Figure 7-1 on page 25 gives information about the clock tree and base frequency for HSSPI.

Table 18-2. HSSPI dividers(1)

SPICKDIV2	SCKCKDIV1	SCKCKDIV0	Divider	External clock divider ratio(1)
0	0	0	4	24 Mhz
0	0	1	5	19.2 Mhz
0	1	0	6	16 Mhz
0	1	1	8	12 Mhz
1	0	0	12	8 Mhz
1	0	1	24	4 Mhz
1	1	0	48	2 Mhz
1	1	1	96	1 Mhz

Note: 1. Base frequency is clk_{IO} = 96Mhz

Note

Any attempt at modifying the HSSPICFG.SPICKDIV value once the HSSPI Controller is enabled will not affect the HSSPI Clock frequency.

- **Bit 4 - DPRAM: DPRAM Bit**

When set (one), this bit enables the DPRAM/HSSPIDMA systems.

When cleared (zero), this bit disables the DPRAM/HSSPIDMA systems.

Note

After a DPRAM is disabled, the DPRAM is reset. This reset is active for two clock cycles of the DPRAM clock. It is recommended to do nothing on the DPRAM during these two clock cycles.

- **Bit 3 - HSCPHA: High Speed Clock PHAse Bit**

The settings of the clock phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of HSSCK. Refer to Figure 18-9 and Figure 18-10.

The CPHA functionality is summarized below:

Table 18-3. CPHA Configuration

CPHA	Leading edge	Trailing edge
0	Sample	Setup
1	Setup	Sample

- **Bit 2 - HSCPOL: High Speed Clock POLarity Bit**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 18-9 and Figure 18-10.

The CPOL functionality is summarized below:

Table 18-4. CPOL Configuration

CPOL	Leading edge	Trailing edge
0	Rising	Falling
1	Falling	Rising

- **Bit 1 - HSMSTR: High Speed Master/Slave Select Bit**

This bit selects the Master HSSPI mode when set (one), and Slave HSSPI mode when cleared (zero).

This bit is cleared by software or by hardware when the HSSPICFG.HSSPIEN gets cleared (disabling the HSSPI).

- **Bit 0 - HSSPIEN: High Speed SPI Enable Bit**

When this bit is set (one), the HSSPI Controller is enabled with the predefined configuration.

When this bit is cleared (zero), the HSSPI Controller is disabled and HSSPICFG.HSMSTR is driven low.

Configuring and activating the HSSPI peripheral with the same instruction is a valid action.

18.5.2 HSSPIIR - HSSPI Interrupt Register

Bit	7	6	5	4	3	2	1	0	
$0000DA	**TIMEOUT**	**BTD**	**RCVOF**	**NSSRE**	**NSSFE**	**-**	**-**	**-**	**HSSPIIR**
Read/write	R/W	R/W	R/W	R/W	R/W	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - TIMEOUT: Timeout Interrupt Bit**

This bit is set (one) when a Timeout occurs. An interrupt is generated if HSSPIIER.TIMEOUTIE is set and global interrupt is enabled.

This bit is cleared by software or by hardware when the HSSPICFG.HSSPIEN is toggled (enabling or disabling the HSSPI).

- **Bit 6 - BTD: Byte Transfer Done Bit**

This bit is set (one) when a serial transfer of the number of bytes specified in HSSPICNT is completed. An interrupt is generated if HSSPIIER.BTDIE is set and global interrupt is enabled.

This bit is cleared by software or by hardware when the HSSPICFG.HSSPIEN is toggled (enabling or disabling the HSSPI).

- **Bit 5 - RCVOF: Receive Overflow Bit**

This bit is set by hardware, when the DPRAM mode is set and the two 16 bytes receive buffers are full. An interrupt is generated if HSSPIIER.RCVOFIE is set and global interrupt is enabled.

- **Bit 4 - NSSRE: NSS Rising Edge**

Set (one) by hardware on a rising edge on $\overline{\text{HSSS}}$

Cleared (zero) by software

- **Bit 3 - NSSFE: NSS Falling edge Bit**

Set (one) by hardware on a falling edge on $\overline{\text{HSSS}}$

Cleared (zero) by software

18.5.3 HSSPIIER - HSSPI Interrupt Enable Register

Bit	7	6	5	4	3	2	1	0	
$0000D4	**TIMEOUTIE**	**BTDIE**	**RCVOFIE**	**NSSIE**	**-**	**-**	**-**	**-**	**HSSPIIER**
Read/write	R/W	R/W	R/W	R/W	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7 - TIMEOUTIE: Timeout Interrupt Enable Bit**

When set (one), this bit enables the Timeout event as the source of the HSSPI interrupt.

When cleared (zero), this bit disables the Timeout source of the HSSPI interrupt.

- **Bit 6 - BTDIE: Byte Transfer Done Interrupt Enable Bit**

When set (one), this bit enables the Byte Transfer Done event as the source of the HSSPI interrupt.

When cleared (zero), this bit disables the Byte Transfer Done source of the HSSPI interrupt.

- **Bit 5 - RCVOFIE: Receive Overflow Interrupt Enable Bit**

When set (one), this bit enables the Receive Overflow event as the source of the HSSPI interrupt.

When cleared (zero), this bit disables the Receive Overflow source of the HSSPI interrupt.

- **Bit 4 - NSSIE: NSS Interrupt Enable**

When set (one), the $\overline{\text{HSSS}}$ interrupt is enabled on falling and rising edges

When cleared (zero), the $\overline{\text{HSSS}}$ interrrupt is disabled

18.5.4 HSSPICR - HSSPI Control Register

Bit	7	6	5	4	3	2	1	0	
$0000DB	-	-	-	-	-	STTTO	RETTO	CS	HSSPICR
Read/write	R	R	R	R	R	W	W	W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 2 - STTTO: Start Time-Out**

When set (one), this will rearm the time-out timer and start it once the first byte is received

When cleared (zero), no action is performed.

- **Bit 1 - RETTO: Rearm Timeout Interrupt Enable Bit**

When set (one), this will rearm the time-out timer, giving more time to receive the next byte.

When cleared (zero), no action is performed.

- **Bit 0 - CS: SPI CS Bit**

When set (one), the HSSPI $\overline{\text{HSSS}}$ pin is set (one).

When cleared (zero), the SPI $\overline{\text{HSSS}}$ pin is cleared (zero).

18.5.5 HSSPISR - HSSPI Status Register

Bit	7	6	5	4	3	2	1	0	
$0000D8	-	RXBUFRDY	TXBUFFREE	DPRAMRDY	NSS	RXBUFF	TXBUFE	SPICKRDY	HSSPISR
Read only	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 6 - RXBUFRDY: Reception Buffer Ready Bit**

When this bit is set (one), at least one reception buffer is full.

- **Bit 5 - TXBUFFREE: Transmission Buffer Free Bit**

When this bit is set (one), at least one transmission buffer is empty.

- **Bit 4 - DPRAMRDY: DPRAM Ready Bit**

When this bit is set (one), the DPRAM is running and operational.

When this bit is cleared (zero), the DPRAM is not available.

- **Bit 3 - NSS: NSS Bit**

Status of the $\overline{\text{HSSS}}$ pin

- **Bit 2 - RXBUFF: Reception Buffer Full Bit**

When this bit is set (one), all reception buffers are full.

- **Bit 1 - TXBUFE: Transmission Buffer Empty Bit**

When this bit is set (one), all transmission buffers are empty.

- **Bit 0 - SPICKRDY: SPI Clock Ready Bit**

When this bit is set (one), the HSSPI Controller is running and operational.

When this bit is cleared (zero), the HSSPI Controller is not available.

18.5.6 HSSPITDR - HSSPI Transmit Data Register

Bit	7	6	5	4	3	2	1	0	
$0000D7	HSSPITDD [7..0]								HSSPITDR
Read/write	W	W	W	W	W	W	W	W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 - HSSPITDD7..0: Transmit Data value**

This register is used only when the DPRAM mode is disabled.

The HSSPI Transmit Data register is a write register used for data transmission from the register file to the HSSPI Shift register. Writing to the register initiates data transmission.

18.5.7 HSSPIRDR - HSSPI Received Data Register

Bit	7	6	5	4	3	2	1	0	
$0000D6	HSSPIRDD [7..0]								HSSPIRDR
Read/write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 - HSSPIRDD7..0: Receive Data value**

This register is used only when the DPRAM mode is disabled.

The HSSPI Receive Data register is a read register used for data reception to the register file from the HSSPI Shift register. Reading this register causes the Shift Register holding the first received data to be read.

18.5.8 HSSPIGTR - HSSPI Guard Time Register

Bit	7	6	5	4	3	2	1	0	
$0000D5	HSSPIGTD [7..0]								HSSPIGTR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 - HSSPIGTD7..0: Guard Time value**

The HSSPI Guard Time register value is the number of HSSPI clock cycles to be inserted between characters, in master mode only.

18.5.9 HSSPICNT - HSSPI Byte Count Register

Bit	7	6	5	4	3	2	1	0	
$0000D3	-	-	-	HSSPICNTD [4..0]					HSSPICNT
Read/write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 4..0 - HSSPICNTD4..0 : Number of byte received or to send**

This register is used only when DPRAM mode is activated.

Writing this register specifies the number of bytes to be sent and received for the current received and transmit buffers.

Reading this register gives the amount of data received for the current received buffer (In this case, the current received buffer means that this buffer received the expected number of data or had a timeout).

Note

Writing HSSPICNT register will internally switch the DPRAM buffers.

Thus, the DMA controller will point to the other DPRAM buffer.

18.5.10 HSSPITIMEOUT - HSSPI Timeout Register

Bit	15	14	13	12	11	10	9	8	
$0000D2	**SPITIMEOUT[15..8]**								**HSSPITOH**
$0000D1	**SPITIMEOUT [7..0]**								**HSSPITOL**
Bit	7	6	5	4	3	2	1	0	
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00
	0	0	0	0	0	0	0	0	0x00

- **Bits 15..0 - SPITIMEOUT15..0 : Communication Timeout**

This register is used only when DPRAM mode is activated. It specifies the number of clock cycles before a timeout occurs (bit HSSPIIR.TIMEOUT is set). See “HSSPIIR - HSSPI Interrupt Register” on page 179. The clock is defined by HSSPICFG.SPICKDIV bits. See “HSSPICFG - HSSPI Config Register” on page 178.

18.6 HSSPIDMA Controller

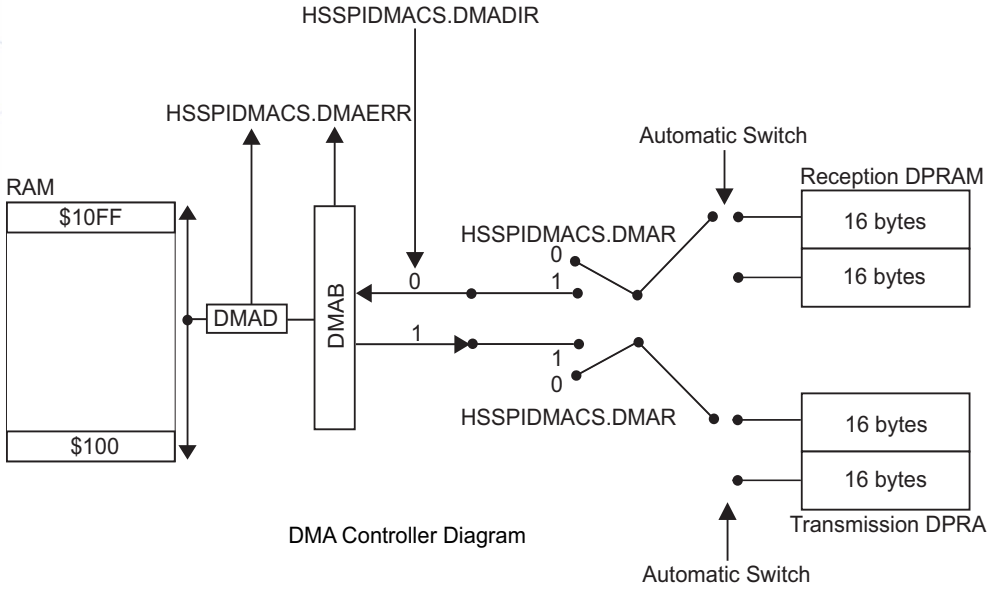
The HSSPIDMA controller, implemented on the AT90SCR075_060, is intended to execute fast transfers between the RAM memory and the internal HSSPI DPRAM. This feature allows the application software of the AT90SCR075_060 to manage the exchanges imposed by the SPI communication at high frequency.

Caution

All the registers described in this section cannot be accessed if the HSSPI module is not enabled.

The HSSPIDMA is enabled as soon as DPRAM mode is activated (by setting bit HSSPICFG.DPRAM). One HSSPIDMA operation can transfer up to N bytes in (N+1) 8/16-bit RISC CPU cycles.

Figure 18-11. HSSPIDMA Controller Diagram



Note

When a HSSPIDMA operation is started, the 8/16-bit RISC CPU is automatically stopped. At the end of the HSSPIDMA operation, the application software automatically restarts where it left (actually with the instruction following the launching of the HSSPIDMA operation). Thus the application software does not need to wait for an interrupt or to need poll the end of the HSSPIDMA operation.

18.6.1 HSSPIDMACS - HSSPIDMA Control and Status register

This is the control and status register of the HSSPIDMA controller.

Bit	7	6	5	4	3	2	1	0	
$0000DC	-	-	-	-	-	**HSSPIDMAERR**	**HSSPIDMADIR**	**HSSPIDMAR**	**HSSPIDMACS**
Read/write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 2 - HSSPIDMAERR : DMA Error Bit**

When starting the HSSPIDMA controller, this bit is cleared (zero) by the hardware if the values in HSSPIDMADH, HSSPIDMADL and HSSPIDMAB registers are suitable for the HSSPIDMA operation requested.

This bit can also be cleared(zero) by software.

This bit is set (one) by hardware when starting a HSSPIDMA operation and whenever one of these following cases occurs:

- The base address contained in the registers HSSPIDMADH and HSSPIDMADL is incorrect (out of the allowed range).
- According to the values of the registers HSSPIDMADH, HSSPIDMADL and HSSPIDMAB and even if the base address is correct, an address out of the allowed range is going to be reached.

- The value in the register HSSPIDMAB is greater than the size of the DPRAM (16bytes) for the HSSPIDMA operation.

When this bit is set, and if the HSSPI interruptions are enabled, an HSSPI interruption is generated.

Don't forget to clear the HSSPIDMACS.DMAERR bit before leaving the interruption routine to avoid repetitive and endless interruptions.

- **Bit 1- HSSPIDMADIR: DMA Direction Bit:**

This bit is set (one) and cleared (zero) by software.

It indicates the direction of the next HSSPIDMA operation transfer between the RAM memory and the internal DPRAM of HSSPI block.

- If the bit is set (one), the transfer will be from the RAM memory to the current transmission DPRAM (emission mode). The empty transmission DPRAM will be automatically selected.
- If the bit is cleared (zero), the transfer will be from the current reception DPRAM to the RAM memory (receiving mode). The firstly filled DPRAM will be selected before the other (more recently modified).

- **Bit 0 - HSSPIDMAR: DMA Run Bit:**

This bit is set (one) by software and cleared (zero) by hardware.

This bit controls the HSSPIDMA operation launching.

It is set (one) by software when a HSSPIDMA operation is to be performed.

It is cleared (zero) by hardware at the end of the operation.

The software does not need to poll this bit in order to detect the end of the HSSPIDMA operation. Indeed, when the HSSPIDMACS.DMAR bit is set by the software, the 8/16-bit RISC CPU is automatically stopped. When the end of the HSSSPIDMA operation is reached, the 8/16-bit RISC CPU then automatically executes the instructions following the setting of the bit HSSPIDMACS.DMAR.

A HSSPIDMA operation can not be interrupted because the CPU is not available during this time.

18.6.2 HSSPIDMAD - HSSPIDMA ADdress registers

Bit	15	14	13	12	11	10	9	8	
$0000DE	-	-			HSSPIDMAD [11..8]				HSSPIDMADH
$0000DD	HSSPIDMAD [8..0]								HSSPIDMADL
Bit	7	6	5	4	3	2	1	0	
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	1	0x01
	0	0	0	0	0	0	0	0	0x00

- **Bits 7..6 - Res: Reserved Bits**

These bits are reserved bits in the AT90SCR075_060 and are always read as zero.

- **Bits 13..0 - HSSPIDMAD11..0 : HS SPI DMA Address**

These bits represents the 12-bit HSSPIDMA Address.

These two registers set the base address in RAM. This address represents the source of the data to be sent if the HSSPIDMA controller is configured in the emission mode. It represents the destination to store the data if the HSSPIDMA controller is configured in the receiving mode.

The initial value corresponds to RAM address $000100.

You can address the whole RAM with this parameter. Values in RAM that must not be dumped, shall be stored out of the HSSPIDMA RAM accessible range.

When starting a HSSPIDMA operation, the hardware will check if the values of HSSPIDMADH, HSSPIDMADL and HSSPIDMAB registers does not exceed the specific RAM area ($000100 to $0008FF). If an error is detected, HSSPIDMACS.DMAERR bit is automatically set (one). A Supervisor RAM Illegal Access Attempt Violation security interrupt (if not masked) is so triggered. HSSPIDMADH, HSSPIDMADL and HSSPIDMAB registers keep their previous value.

After a HSSPIDMA operation, HSSPIDMADH and HSSPIDMADL are set to the last value reached in RAM and incremented by one. For instance, after a 64-byte transfer started from base address $000100, HSSPIDMAD equals to $000140 (HSSPIDMADH = $01 and HSSPIDMADL = $40). This feature allows to simplify registers and bits handlings when several HSSPIDMA operations are to be successively performed, which can be the case when getting or sending several packets.

18.6.3 HSSPIDMAB - DMA Amount of Bytes Register

This register is dedicated to the amount of bytes to be transferred during the next HSSPIDMA operation setting.

Bit	7	6	5	4	3	2	1	0	
$00DF	-	-	-	HSSPIDMAB [4..0]					HSSPIDMAB
Read/write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 7..5 - Res : Reserved Bits**

This bit is reserved bit in the AT90SCR075_060 and is always read as zero.

- **Bits 4..0 - HSSPIDMAB4..0 : HS SPI DMA Amount of Bytes Bits**

These bits are the (4..0) bits of the 7-bit HSSPIDMA Amount of Bytes value.

When starting a HSSPIDMA operation, the hardware will check if the values of HSSPIDMADH, HSSPIDMADL and HSSPIDMAB registers does not exceed the specific RAM area ($000100 to $0008FF). If an error is detected, HSSPIDMACS.DMAERR bit is automatically set (one). A Supervisor RAM Illegal Access Attempt Violation security interrupt (if not masked) is so triggered. HSSPIDMADH, HSSPIDMADL and HSSPIDMAB registers keep their previous value.

After a HSSPIDMA operation completion, the value of this register is not reset.

The maximum value allowed for HSSPIDMAB is 16.

19. 2-wire Serial Interface _ TWI

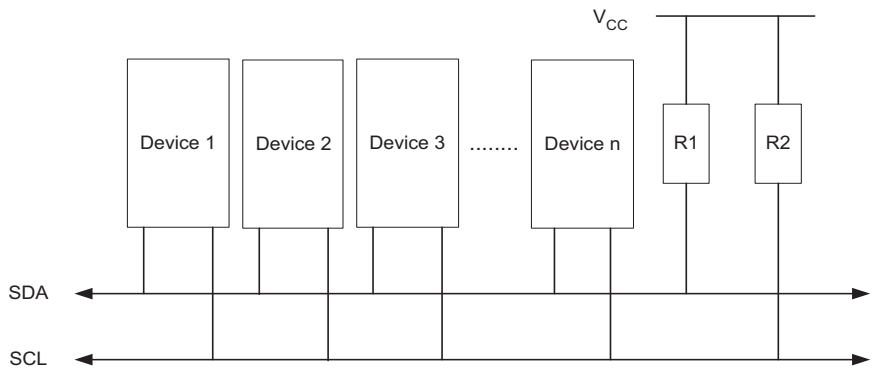
19.1 Features

- **Simple Yet Powerful and Flexible Communication Interface, only two Bus Lines Needed**
- **Both Master and Slave Operation Supported**
- **Device can Operate as Transmitter or Receiver**
- **7-bit Address Space Allows up to 128 Different Slave Addresses**
- **Multi-master Arbitration Support**
- **Up to 400 kHz Data Transfer Speed**
- **Slew-rate Limited Output Drivers**
- **Noise Suppression Circuitry Rejects Spikes on Bus Lines**
- **Fully Programmable Slave Address with General Call Support**
- **Address Recognition Causes Wake-up When 8/16-bit RISC CPU is in Sleep Mode**

19.2 TWI Serial Interface Bus Definition

The 2-wire Serial Interface (TWI) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol.

Figure 19-1. TWI Bus Interconnection



19.2.1 TWI Terminology

The following definitions are frequently encountered in this section.

Table 19-1. TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock.

Table 19-1. TWI Terminology

Term	Description
Slave	The device addressed by a Master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

The Power Reduction TWI bit, PRTWI bit in PRR0 must be written to zero to enable the TWI Serial Interface.

19.2.2 Electrical Interconnection

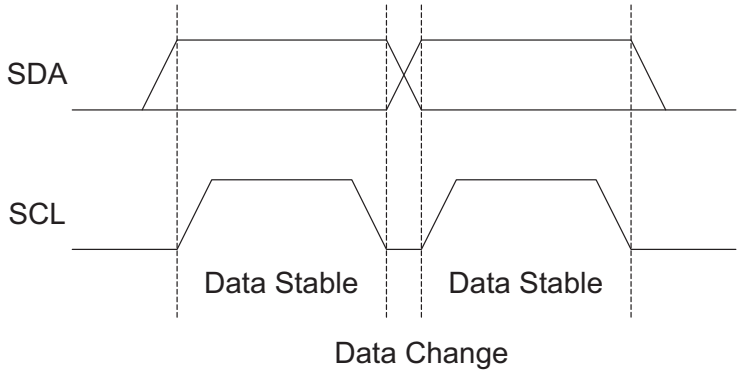
The number of devices that can be connected to the bus is only limited by the bus capacitance-limit of 400 pF and the 7-bit slave address space. A detailed specification of the electrical characteristics of the TWI is given in section “2-wire Serial Interface Characteristics” on page 234. Two different sets of specifications are presented there, one relevant for bus speeds below 100 kHz, and one valid for bus speeds up to 400 kHz.

19.3 Data Transfer and Frame Format

19.3.1 Transferring Bits

Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

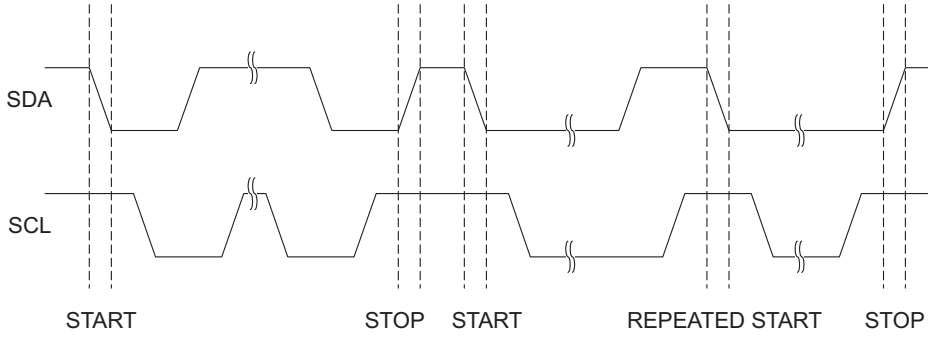
Figure 19-2. Data Validity



19.3.2 START and STOP Conditions

The Master initiates and terminates a data transmission. The transmission is initiated when the Master issues a START condition on the bus, and it is terminated when the Master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the Master wishes to initiate a new transfer without relinquishing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signalled by changing the level of the SDA line when the SCL line is high.

Figure 19-3. START, REPEATED START and STOP conditions



19.3.3 Address Packet Format

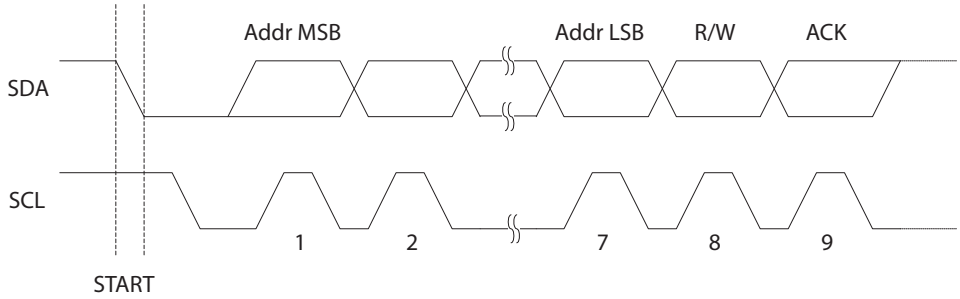
All address packets transmitted on the TWI bus are 9 bits long, consisting of 7 address bits, one READ/WRITE control bit and an acknowledge bit. If the READ/WRITE bit is set, a read operation is to be performed, otherwise a write operation should be performed. When a Slave recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle. If the addressed Slave is busy, or for some other reason can not service the Master's request, the SDA line should be left high in the ACK clock cycle. The Master can then transmit a STOP condition, or a REPEATED START condition to initiate a new transmission. An address packet consisting of a slave address and a READ or a WRITE bit is called SLA+R or SLA+W, respectively.

The MSB of the address byte is transmitted first. Slave addresses can freely be allocated by the designer, but the address 0000 000 is reserved for a general call.

When a general call is issued, all slaves should respond by pulling the SDA line low in the ACK cycle. A general call is used when a Master wishes to transmit the same message to several slaves in the system. When the general call address followed by a Write bit is transmitted on the bus, all slaves set up to acknowledge the general call will pull the SDA line low in the ack cycle. The following data packets will then be received by all the slaves that acknowledged the general call. Note that transmitting the general call address followed by a Read bit is meaningless, as this would cause contention if several slaves started transmitting different data.

All addresses of the format 1111 xxx should be reserved for future purposes.

Figure 19-4. Address Packet Format

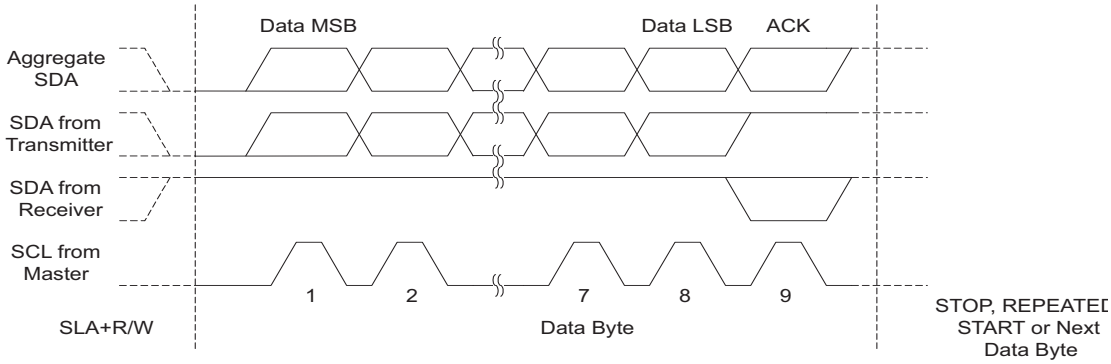


19.3.4 Data Packet Format

All data packets transmitted on the TWI bus are nine bits long, consisting of one data byte and an acknowledge bit. During a data transfer, the Master generates the clock and the START and

STOP conditions, while the Receiver is responsible for acknowledging the reception. An Acknowledge (ACK) is signalled by the Receiver pulling the SDA line low during the ninth SCL cycle. If the Receiver leaves the SDA line high, a NACK is signalled. When the Receiver has received the last byte, or for some reason cannot receive any more bytes, it should inform the Transmitter by sending a NACK after the final byte. The MSB of the data byte is transmitted first.

Figure 19-5. Data Packet Format

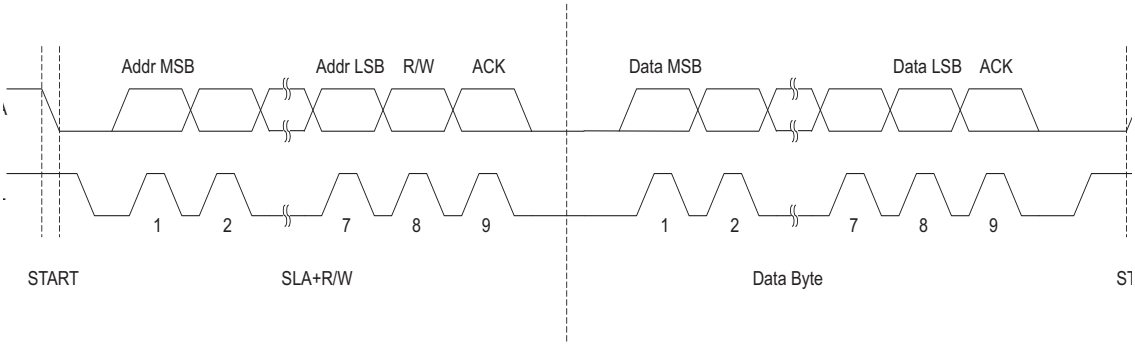


19.3.5 Combining Address and Data Packets into a Transmission

A transmission basically consists of a START condition, a SLA+R/W, one or more data packets and a STOP condition. An empty message, consisting of a START followed by a STOP condition, is illegal. Note that the Wired-ANDing of the SCL line can be used to implement handshaking between the Master and the Slave. The Slave can extend the SCL low period by pulling the SCL line low. This is useful if the clock speed set up by the Master is too fast for the Slave, or the Slave needs extra time for processing between the data transmissions. The Slave extending the SCL low period will not affect the SCL high period, which is determined by the Master. As a consequence, the Slave can reduce the TWI data transfer speed by prolonging the SCL duty cycle.

Figure 19-6 shows a typical data transmission. Note that several data bytes can be transmitted between the SLA+R/W and the STOP condition, depending on the software protocol implemented by the application software.

Figure 19-6. Typical Data Transmission



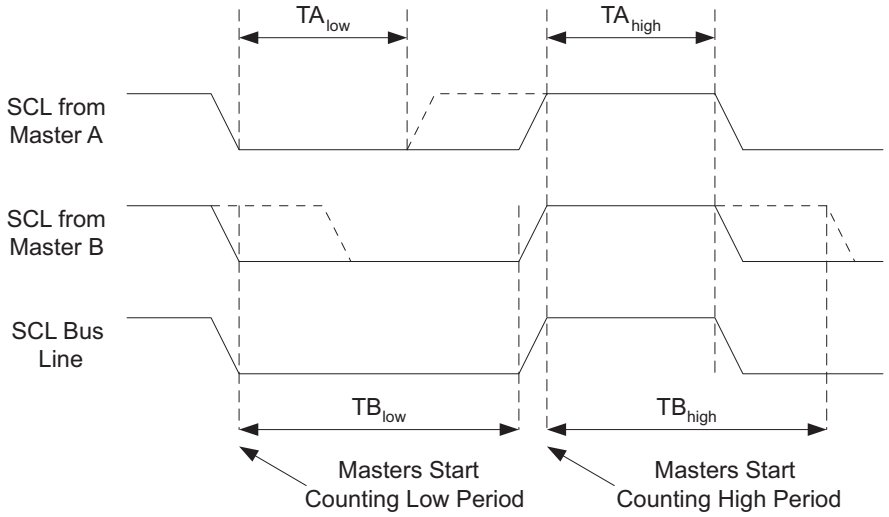
19.4 Multi-master Bus Systems, Arbitration and Synchronization

The TWI protocol allows bus systems with several masters. Special concerns have been taken in order to ensure that transmissions will proceed as normal, even if two or more masters initiate a transmission at the same time. Two problems arise in multi-master systems:

- An algorithm must be implemented allowing only one of the masters to complete the transmission. All other masters should cease transmission when they discover that they have lost the selection process. This selection process is called arbitration. When a contending master discovers that it has lost the arbitration process, it should immediately switch to Slave mode to check whether it is being addressed by the winning master. The fact that multiple masters have started transmission at the same time should not be detectable to the slaves, i.e. the data being transferred on the bus must not be corrupted.
- Different masters may use different SCL frequencies. A scheme must be devised to synchronize the serial clocks from all masters, in order to let the transmission proceed in a lockstep fashion. This will facilitate the arbitration process.

The wired-ANDing of the bus lines is used to solve both these problems. The serial clocks from all masters will be wired-ANDed, yielding a combined clock with a high period equal to the one from the Master with the shortest high period. The low period of the combined clock is equal to the low period of the Master with the longest low period. Note that all masters listen to the SCL line, effectively starting to count their SCL high and low time-out periods when the combined SCL line goes high or low, respectively.

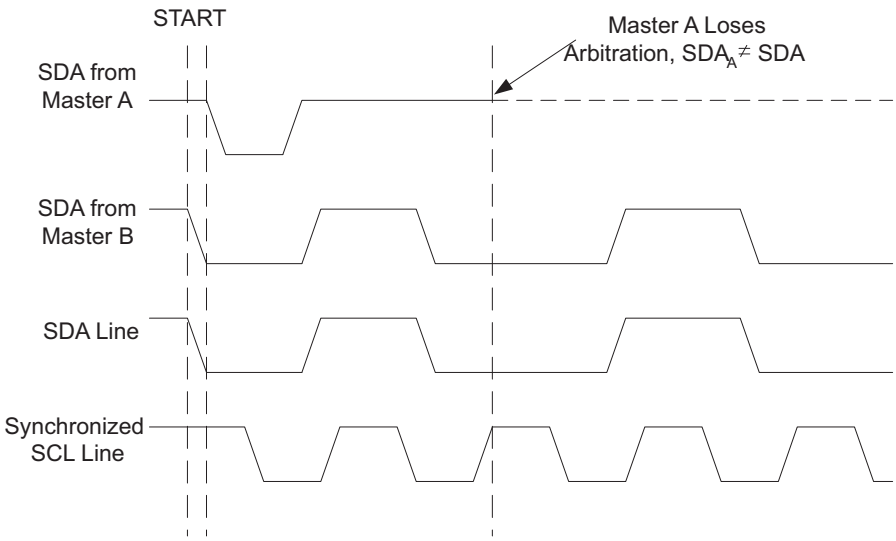
Figure 19-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many

bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

Figure 19-8. Arbitration Between Two Masters



Note that arbitration is not allowed between:

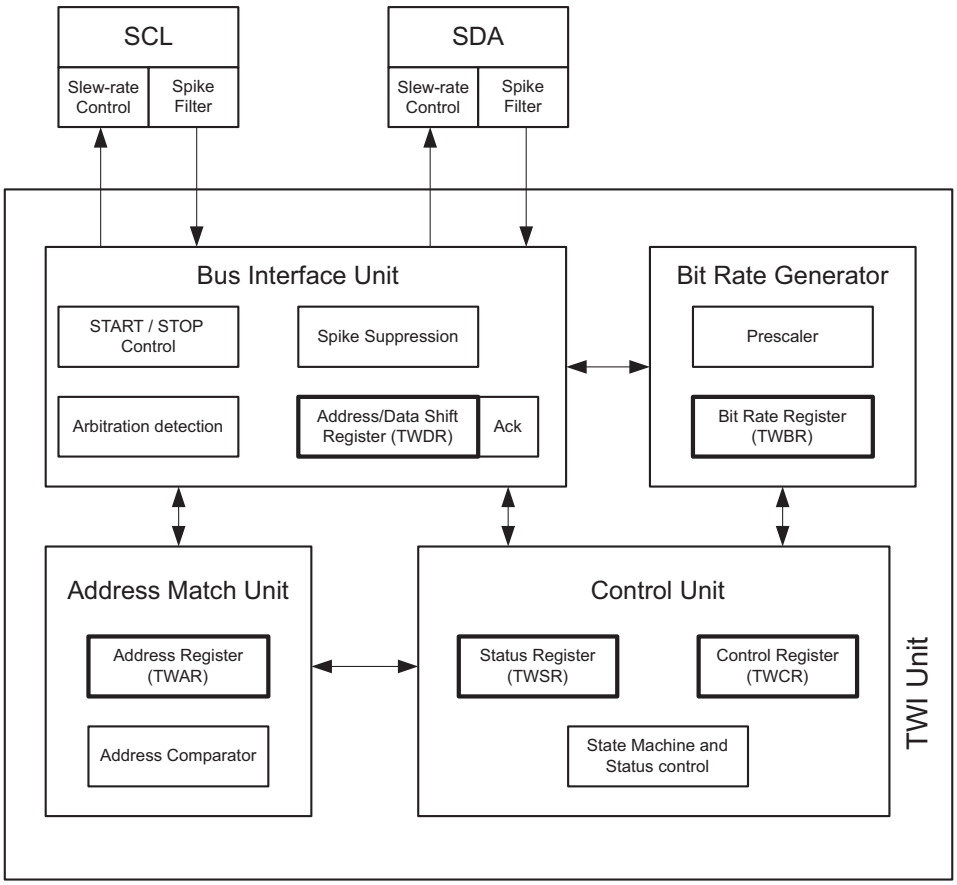
- A REPEATED START condition and a data bit.
- A STOP condition and a data bit.
- A REPEATED START and a STOP condition.

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

19.5 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 19-9. All registers drawn in a thick line are accessible through the 8/16-bit RISC CPU data bus.

Figure 19-9. Overview of the TWI Module



19.5.1 SCL and SDA Pins

These pins interface the 8/16-bit RISC CPU TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns. Note that SDA and SCL pads supports internal pull-ups. The internal pull-ups can in some systems eliminate the need for external ones.

19.5.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR) and the Prescaler bits in the TWI Status Register (TWSR). Slave operation does not depend on Bit Rate or Prescaler settings, but the CPU clock frequency in the Slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the average TWI bus clock period. The SCL frequency is generated according to the following equation:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register.

- TWPS = Value of the prescaler bits in the TWI Status Register

Note

Pull-up resistor values should be selected according to the SCL frequency and the capacitive bus line load.

19.5.3 Bus Interface Unit

This unit contains the Data and Address Shift Register (TWDR), a START/STOP Controller and Arbitration detection hardware. The TWDR contains the address or data bytes to be transmitted, or the address or data bytes received. In addition to the 8-bit TWDR, the Bus Interface Unit also contains a register containing the (N)ACK bit to be transmitted or received. This (N)ACK Register is not directly accessible by the application software. However, when receiving, it can be set or cleared by manipulating the TWI Control Register (TWCR). When in Transmitter mode, the value of the received (N)ACK bit can be determined by the value in the TWSR.

The START/STOP Controller is responsible for generation and detection of START, REPEATED START, and STOP conditions. The START/STOP controller is able to detect START and STOP conditions even when the 8/16-bit RISC CPU MCU is in one of the sleep modes, enabling the MCU to wake up if addressed by a Master.

If the TWI has initiated a transmission as Master, the Arbitration Detection hardware continuously monitors the transmission trying to determine if arbitration is in process. If the TWI has lost an arbitration, the Control Unit is informed. Correct action can then be taken and appropriate status codes generated.

19.5.4 Address Match Unit

The Address Match unit checks if received address bytes match the seven-bit address in the TWI Address Register (TWAR). If the TWI General Call Recognition Enable (TWGCE) bit in the TWAR is written to one, all incoming address bits will also be compared against the General Call address. Upon an address match, the Control Unit is informed, allowing correct action to be taken. The TWI may or may not acknowledge its address, depending on settings in the TWCR. The Address Match unit is able to compare addresses even when the 8/16-bit RISC CPU MCU is in sleep mode, enabling the MCU to wake up if addressed by a Master. If another interrupt (e.g., INT0) occurs during TWI Power-down address match and wakes up the CPU, the TWI aborts operation and return to its idle state. If this causes any problems, ensure that TWI Address Match is the only enabled interrupt when entering Power-down.

19.5.5 Control Unit

The Control unit monitors the TWI bus and generates responses corresponding to settings in the TWI Control Register (TWCR). When an event requiring the attention of the application occurs on the TWI bus, the TWI Interrupt Flag (TWINT) is asserted. In the next clock cycle, the TWI Status Register (TWSR) is updated with a status code identifying the event. The TWSR only contains relevant status information when the TWI Interrupt Flag is asserted. At all other times, the TWSR contains a special status code indicating that no relevant status information is available. As long as the TWINT Flag is set, the SCL line is held low. This allows the application software to complete its tasks before allowing the TWI transmission to continue.

The TWINT Flag is set in the following situations:

- After the TWI has transmitted a START/REPEATED START condition.
- After the TWI has transmitted SLA+R/W.

- After the TWI has transmitted an address byte.
- After the TWI has lost arbitration.
- After the TWI has been addressed by own slave address or general call.
- After the TWI has received a data byte.
- After a STOP or REPEATED START has been received while still addressed as a Slave.
- When a bus error has occurred due to an illegal START or STOP condition.

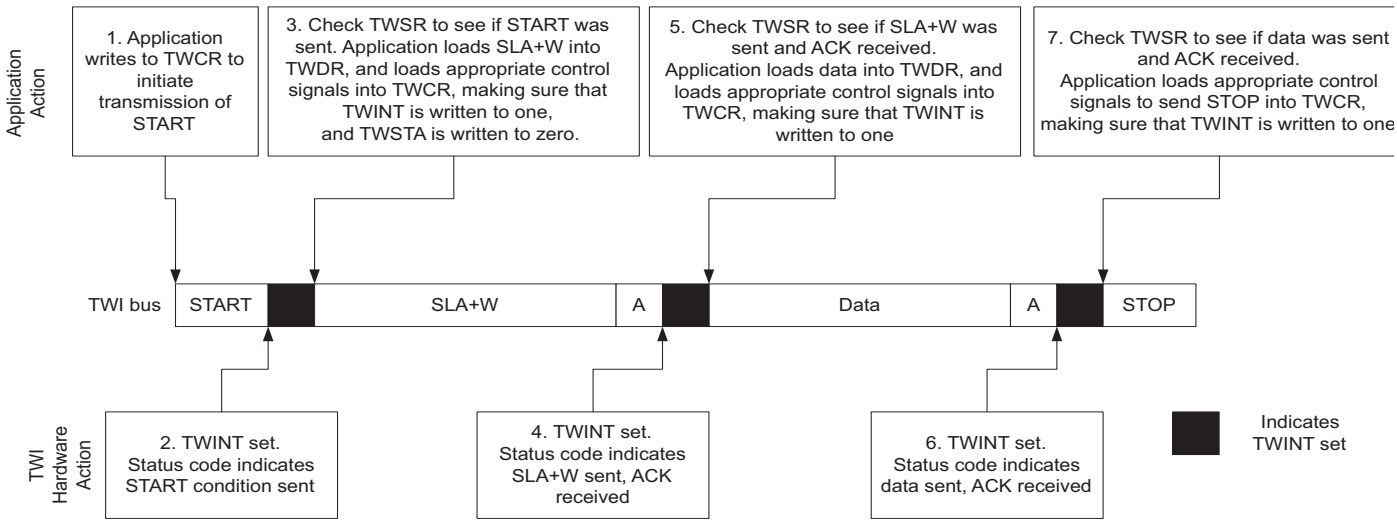
19.6 Using the TWI

The 8/16-bit RISC CPU TWI is byte-oriented and interrupt based. Interrupts are issued after all bus events, like reception of a byte or transmission of a START condition. Because the TWI is interrupt-based, the application software is free to carry on other operations during a TWI byte transfer. Note that the TWI Interrupt Enable (TWIE) bit in TWCR together with the Global Interrupt Enable bit in SREG allow the application to decide whether or not assertion of the TWINT Flag should generate an interrupt request. If the TWIE bit is cleared, the application must poll the TWINT Flag in order to detect actions on the TWI bus.

When the TWINT Flag is asserted, the TWI has finished an operation and awaits application response. In this case, the TWI Status Register (TWSR) contains a value indicating the current state of the TWI bus. The application software can then decide how the TWI should behave in the next TWI bus cycle by manipulating the TWCR and TWDR Registers.

Figure 19-10 is a simple example of how the application can interface to the TWI hardware. In this example, a Master wishes to transmit a single data byte to a Slave. This description is quite abstract, a more detailed explanation follows later in this section. A simple code example implementing the desired behavior is also presented.

Figure 19-10. Interfacing the Application to the TWI in a Typical Transmission



1. The first step in a TWI transmission is to transmit a START condition. This is done by writing a specific value into TWCR, instructing the TWI hardware to transmit a START condition. The value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after

the application has cleared TWINT, the TWI will initiate transmission of the START condition.

2. When the START condition has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the START condition has successfully been sent.
3. The application software should now examine the value of TWSR, to make sure that the START condition was successfully transmitted. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load SLA+W into TWDR. Remember that TWDR is used both for address and data. After TWDR has been loaded with the desired SLA+W, a specific value must be written to TWCR, instructing the TWI hardware to transmit the SLA+W present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the address packet.
4. When the address packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the address packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
5. The application software should now examine the value of TWSR, to make sure that the address packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must load a data packet into TWDR. Subsequently, a specific value must be written to TWCR, instructing the TWI hardware to transmit the data packet present in TWDR. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the data packet.
6. When the data packet has been transmitted, the TWINT Flag in TWCR is set, and TWSR is updated with a status code indicating that the data packet has successfully been sent. The status code will also reflect whether a Slave acknowledged the packet or not.
7. The application software should now examine the value of TWSR, to make sure that the data packet was successfully transmitted, and that the value of the ACK bit was as expected. If TWSR indicates otherwise, the application software might take some special action, like calling an error routine. Assuming that the status code is as expected, the application must write a specific value to TWCR, instructing the TWI hardware to transmit a STOP condition. Which value to write is described later on. However, it is important that the TWINT bit is set in the value written. Writing a one to TWINT clears the flag. The TWI will not start any operation as long as the TWINT bit in TWCR is set. Immediately after the application has cleared TWINT, the TWI will initiate transmission of the STOP condition. Note that TWINT is NOT set after a STOP condition has been sent.

Even though this example is simple, it shows the principles involved in all TWI transmissions. These can be summarized as follows:

- When the TWI has finished an operation and expects application response, the TWINT Flag is set. The SCL line is pulled low until TWINT is cleared.

- When the TWINT Flag is set, the user must update all TWI Registers with the value relevant for the next TWI bus cycle. As an example, TWDR must be loaded with the value to be transmitted in the next bus cycle.
- After all TWI Register updates and other pending application software tasks have been completed, TWCR is written. When writing TWCR, the TWINT bit should be set. Writing a one to TWINT clears the flag. The TWI will then start executing whatever operation was specified by the TWCR setting.

In the following an assembly and C implementation of the example is given. Note that the code below assumes that several definitions have been made, for example by using include-files.

	Assembly Code Example	C Example	Comments
1	`ldi r16, (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)`	Send START condition

2	`wait1:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait1`
		`while (!(TWCR & (1<<TWINT)))`
		`;`
			Wait for TWINT Flag set. This indicates that the START condition has been transmitted

3	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, START`
	`brne ERROR`
		`if ((TWSR & 0xF8) != START)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR

	`ldi r16, SLA_W`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT) | (1<<TWEN)`
	`out TWCR, r16`
		`TWDR = SLA_W;`
		`TWCR = (1<<TWINT) | (1<<TWEN);`
			Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address

4	`wait2:`
	`in r16,TWCR`
	`sbrs r16,TWINT`
	`rjmp wait2`
		`while (!(TWCR & (1<<TWINT)))`
		`;`
			Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

5	`in r16,TWSR`
	`andi r16, 0xF8`
	`cpi r16, MT_SLA_ACK`
	`brne ERROR`
		`if ((TWSR & 0xF8) != MT_SLA_ACK)`
		`ERROR();`
			Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR

	`ldi r16, DATA`
	`out TWDR, r16`
	`ldi r16, (1<<TWINT) | (1<<TWEN)`
	`out TWCR, r16`
		`TWDR = DATA;`
		`TWCR = (1<<TWINT) | (1<<TWEN);`
			Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data

	Assembly Code Example	C Example	Comments
6	wait3: in r16,TWCR sbrs r16,TWINT rjmp wait3	while (!(TWCR & (1<<TWINT))) ;	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.
7	in r16,TWSR andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR	if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR
	ldi r16, (1<<TWINT)|(1<<TWEN)| (1<<TWSTO) out TWCR, r16	TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);	Transmit STOP condition

19.7 Transmission Modes

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM. If other masters are present in the system, some of these might transmit data to the TWI, and then SR mode would be used. It is the application software that decides which modes are legal.

The following sections describe each of these modes. Possible status codes are described along with figures detailing data transmission in each of the modes. These figures contain the following abbreviations:

S: START condition

Rs: REPEATED START condition

R: Read bit (high level at SDA)

W: Write bit (low level at SDA)

A: Acknowledge bit (low level at SDA)

$\overline{A}$: Not acknowledge bit (high level at SDA)

Data: 8-bit data byte

P: STOP condition

SLA: Slave Address

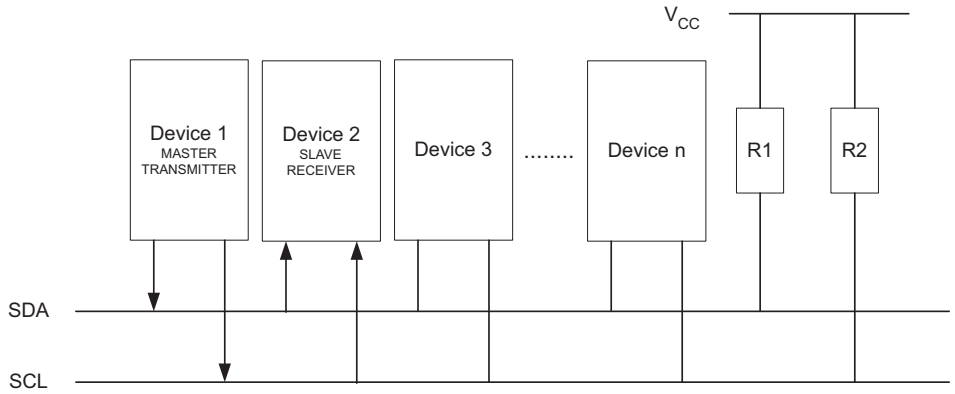
In Figure 19-12 to Figure 19-18, circles are used to indicate that the TWINT Flag is set. The numbers in the circles show the status code held in TWSR, with the prescaler bits masked to zero. At these points, actions must be taken by the application to continue or complete the TWI transfer. The TWI transfer is suspended until the TWINT Flag is cleared by software.

When the TWINT Flag is set, the status code in TWSR is used to determine the appropriate software action. For each status code, the required software action and details of the following serial transfer are given in Table 19-2 to Table 19-4. Note that the prescaler bits are masked to zero in these tables.

19.7.1 Master Transmitter Mode

In the Master Transmitter mode, a number of data bytes are transmitted to a Slave Receiver (see Figure 19-11). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 19-11. Data Transfer in Master Transmitter Mode



Note

Please note that R1 and R2 resistors may not be required. The internal pull-up resistors on SDA and SCL pads of 8/16-bit RISC CPU may be sufficient to setup a communication. However, if the edges are not strong enough to get a clean signals, R1 and R2 can be added.

A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN must be set to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be written to one to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (see Table 19-2). In order to enter MT mode, SLA+W must be transmitted. This is done by writing SLA+W to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

When SLA+W have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x18, 0x20, or 0x38. The appropriate action to be taken for each of these status codes is detailed in Table 19-2.

When SLA+W has been successfully transmitted, a data packet should be transmitted. This is done by writing the data byte to TWDR. TWDR must only be written when TWINT is high. If not,

the access will be discarded, and the Write Collision bit (TWWC) will be set in the TWCR Register. After updating TWDR, the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

This scheme is repeated until the last byte has been sent and the transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control of the bus.

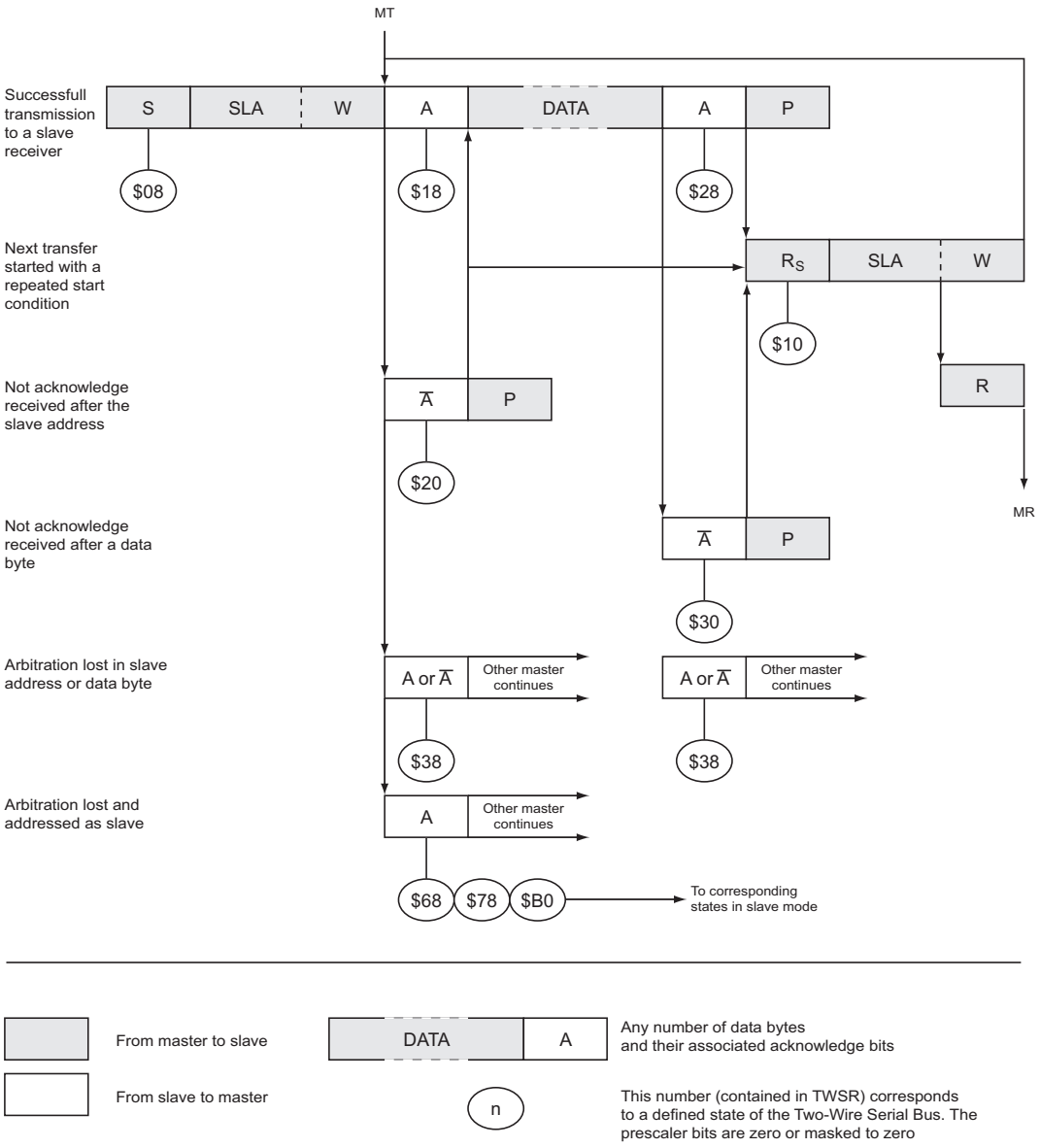
Table 19-2. Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

Table 19-2. Status codes for Master Transmitter Mode

0x28	Data byte has been transmitted; ACK has been received	Load data byte or No TWDR action or No TWDR action or No TWDR action	0 1 0 1	0 0 1 1	1 1 1 1	X X X X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or No TWDR action or No TWDR action or No TWDR action	0 1 0 1	0 0 1 1	1 1 1 1	X X X X	Data byte will be transmitted and ACK or NOT ACK will be received Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or No TWDR action	0 1	0 0	1 1	X X	2-wire Serial Bus will be released and not addressed Slave mode entered A START condition will be transmitted when the bus becomes free

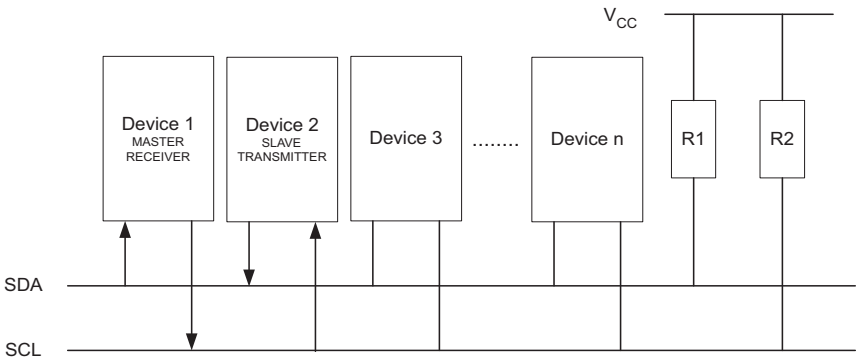
Figure 19-12. Formats and States in the Master Transmitter Mode



19.7.2 Master Receiver Mode

In the Master Receiver mode, a number of data bytes are received from a Slave Transmitter (Slave see Figure 19-13). In order to enter a Master mode, a START condition must be transmitted. The format of the following address packet determines whether Master Transmitter or Master Receiver mode is to be entered. If SLA+W is transmitted, MT mode is entered, if SLA+R is transmitted, MR mode is entered. All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 19-13. Data Transfer in Master Receiver Mode



Note

Please note that R1 and R2 resistors may be useless. Indeed, internal pull-up resistors on SDA and SCL pads of 8/16-bit RISC CPU may be sufficient to setup a communication. Anyway, R1 and R2 are welcomed if the edges are not strong enough to get a clean signals.

A START condition is sent by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	1	0	X	1	0	X

TWEN must be written to one to enable the 2-wire Serial Interface, TWSTA must be written to one to transmit a START condition and TWINT must be set to clear the TWINT Flag. The TWI will then test the 2-wire Serial Bus and generate a START condition as soon as the bus becomes free. After a START condition has been transmitted, the TWINT Flag is set by hardware, and the status code in TWSR will be 0x08 (See Table 19-2). In order to enter MR mode, SLA+R must be transmitted. This is done by writing SLA+R to TWDR. Thereafter the TWINT bit should be cleared (by writing it to one) to continue the transfer. This is accomplished by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	0	X	1	0	X

When SLA+R have been transmitted and an acknowledgement bit has been received, TWINT is set again and a number of status codes in TWSR are possible. Possible status codes in Master mode are 0x38, 0x40, or 0x48. The appropriate action to be taken for each of these status codes is detailed in Table 19-3. Received data can be read from the TWDR Register when the TWINT Flag is set high by hardware. This scheme is repeated until the last byte has been received. After the last byte has been received, the MR should inform the ST by sending a NACK after the last received data byte. The transfer is ended by generating a STOP condition or a repeated START condition. A STOP condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	1	X	0	1	X	1	0	X

A REPEATED START condition is generated by writing the following value to TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE

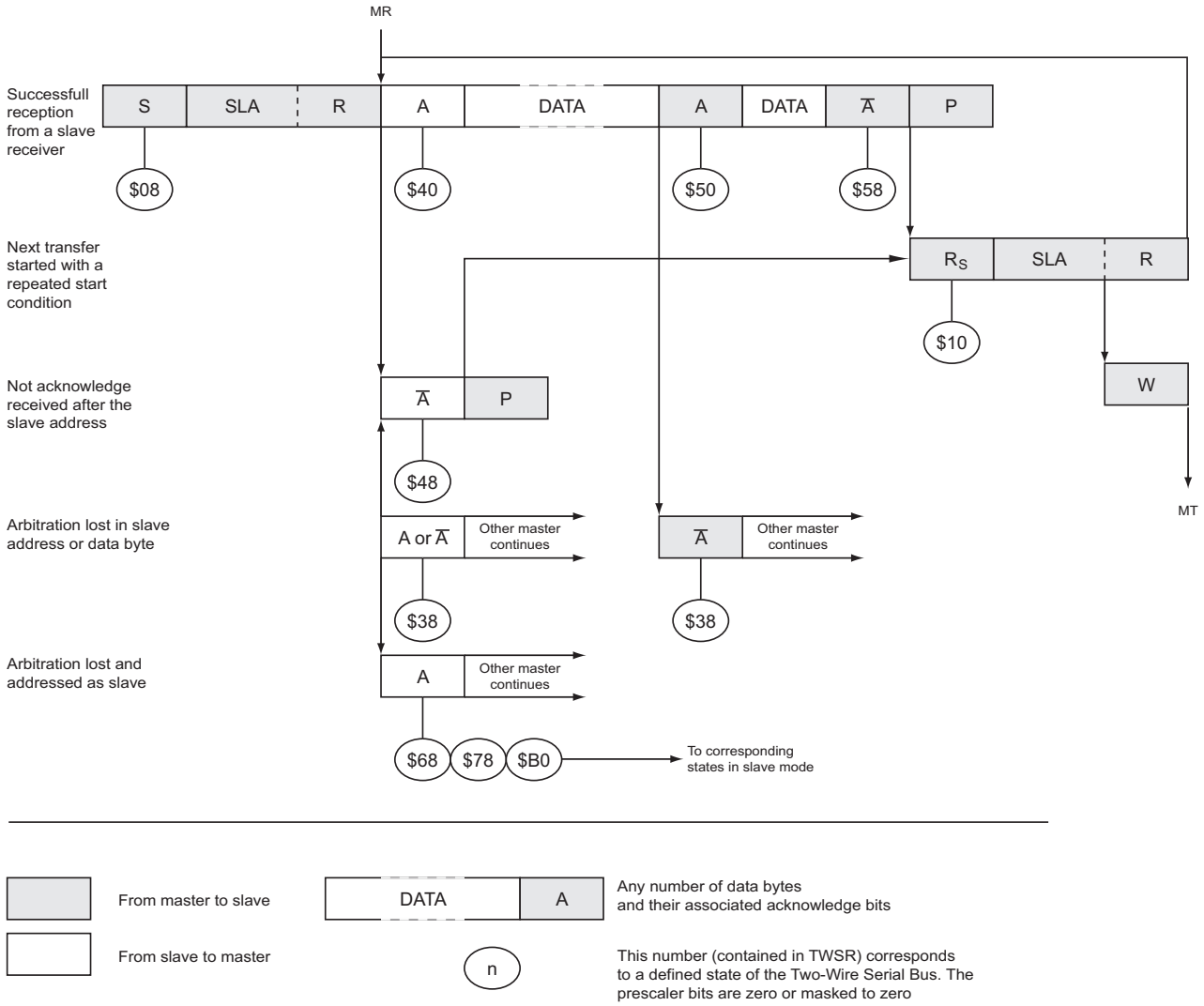
value	1	X	1	0	X	1	0	X

After a repeated START condition (state 0x10) the 2-wire Serial Interface can access the same Slave again, or a new Slave without transmitting a STOP condition. Repeated START enables the Master to switch between Slaves, Master Transmitter mode and Master Receiver mode without losing control over the bus.

Table 19-3. Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
		Load SLA+W	0	0	1	X	SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode will be entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	0	0	1	1	Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or	0	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	0	0	1	1	Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or	1	0	1	X	Repeated START will be transmitted
		Read data byte or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		Read data byte	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

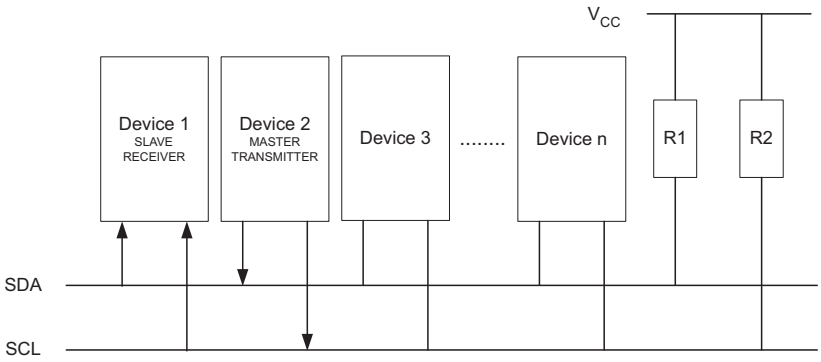
Figure 19-14. Formats and States in the Master Receiver Mode



19.7.3 Slave Receiver Mode

In the Slave Receiver mode, a number of data bytes are received from a Master Transmitter (see Figure 19-15). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 19-15. Data transfer in Slave Receiver mode



Note

Please note that R1 and R2 resistors may be useless. Indeed, internal pull-up resistors on SDA and SCL pads of 8/16-bit RISC CPU may be sufficient to setup a communication. Anyway, R1 and R2 are welcomed if the edges are not strong enough to get a clean signals.

To initiate the Slave Receiver mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

The upper 7 bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "0" (write), the TWI will operate in SR mode, otherwise ST mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate soft-ware action. The appropriate action to be taken for each status code is detailed in Table 19-4. The Slave Receiver mode may also be entered if arbitration is lost while the TWI is in the Master mode (see states 0x68 and 0x78).

If the TWEA bit is reset during a transfer, the TWI will return a "Not Acknowledge" ("1") to SDA after the next received data byte. This can be used to indicate that the Slave is not able to receive any more bytes. While TWEA is zero, the TWI does not acknowledge its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and

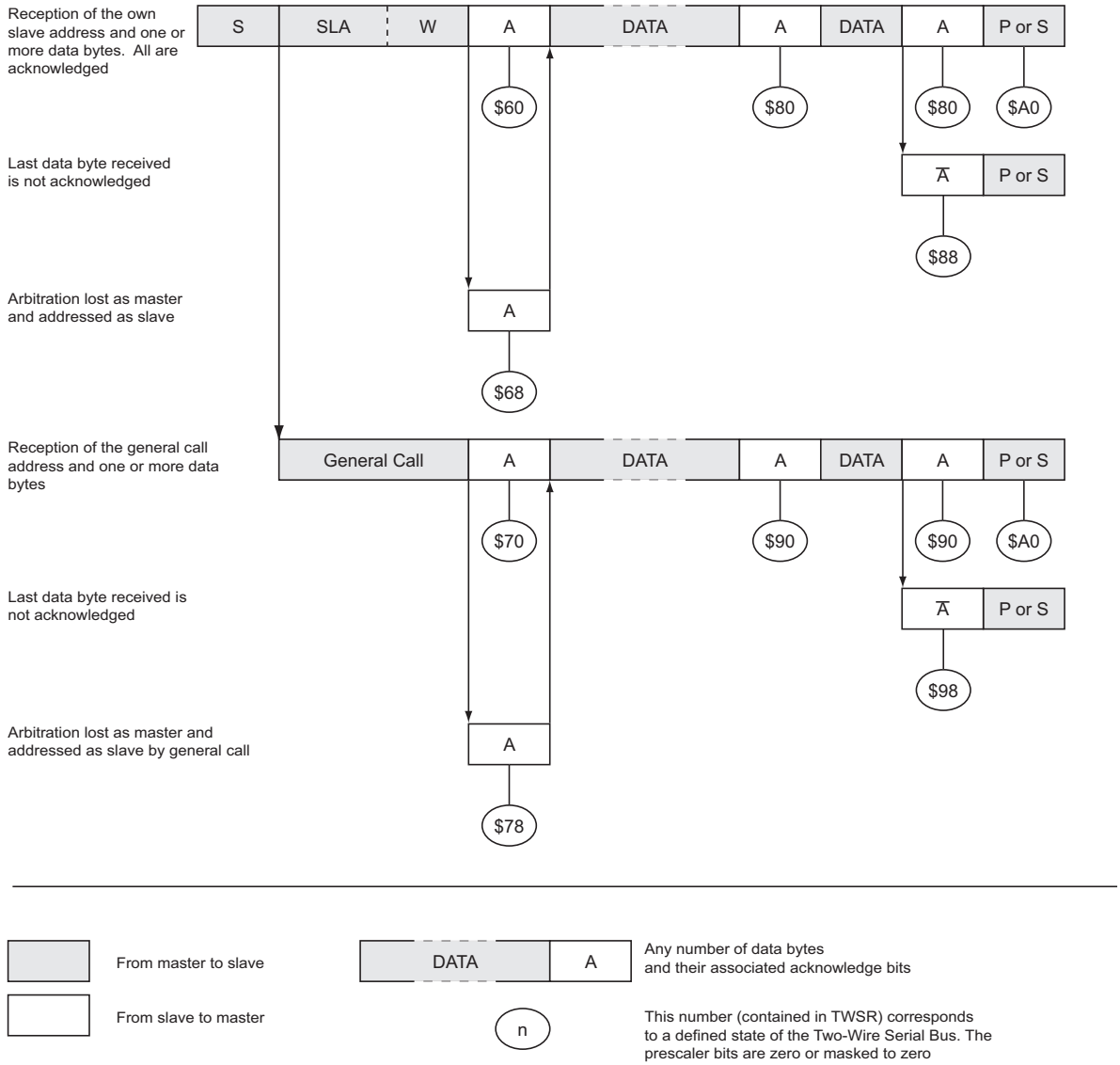
the TWI will hold the SCL clock low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data reception will be carried out as normal, with the 8/16-bit RISC CPU clocks running as normal. Observe that if the 8/16-bit RISC CPU is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

theTWI Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these Sleep modes.

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

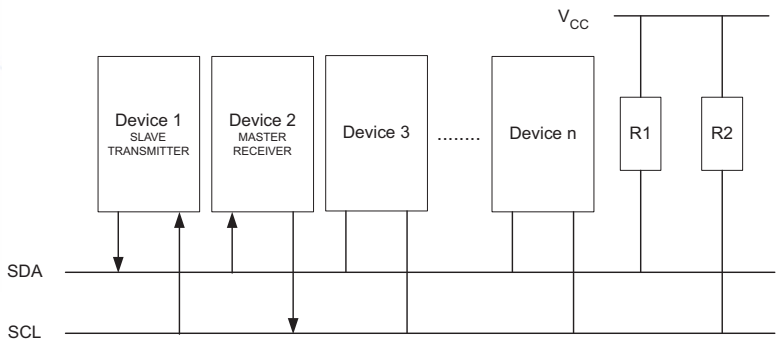
Figure 19-16. Formats and States in the Slave Receiver Mode



19.7.4 Slave Transmitter Mode

In the Slave Transmitter mode, a number of data bytes are transmitted to a Master Receiver (see Figure 19-17). All the status codes mentioned in this section assume that the prescaler bits are zero or are masked to zero.

Figure 19-17. Data Transfer in Slave Transmitter Mode



To initiate the Slave Transmitter mode, TWAR and TWCR must be initialized as follows:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
value	Device's Own Slave Address							

The upper seven bits are the address to which the 2-wire Serial Interface will respond when addressed by a Master. If the LSB is set, the TWI will respond to the general call address (0x00), otherwise it will ignore the general call address.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
value	0	1	0	0	0	1	0	X

TWEN must be written to one to enable the TWI. The TWEA bit must be written to one to enable the acknowledgement of the device's own slave address or the general call address. TWSTA and TWSTO must be written to zero.

When TWAR and TWCR have been initialized, the TWI waits until it is addressed by its own slave address (or the general call address if enabled) followed by the data direction bit. If the direction bit is "1" (read), the TWI will operate in ST mode, otherwise SR mode is entered. After its own slave address and the write bit have been received, the TWINT Flag is set and a valid status code can be read from TWSR. The status code is used to determine the appropriate software action. The appropriate action to be taken for each status code is detailed in Table 19-4. The Slave Transmitter mode may also be entered if arbitration is lost while the TWI is in the Master mode (see state 0xB0).

If the TWEA bit is written to zero during a transfer, the TWI will transmit the last byte of the transfer. State 0xC0 or state 0xC8 will be entered, depending on whether the Master Receiver transmits a NACK or ACK after the final byte. The TWI is switched to the not addressed Slave mode, and will ignore the Master if it continues the transfer. Thus the Master Receiver receives all "1" as serial data. State 0xC8 is entered if the Master demands additional data bytes (by transmitting ACK), even though the Slave has transmitted the last byte (TWEA zero and expecting NACK from the Master).

While TWEA is zero, the TWI does not respond to its own slave address. However, the 2-wire Serial Bus is still monitored and address recognition may resume at any time by setting TWEA. This implies that the TWEA bit may be used to temporarily isolate the TWI from the 2-wire Serial Bus.

In all sleep modes other than Idle mode, the clock system to the TWI is turned off. If the TWEA bit is set, the interface can still acknowledge its own slave address or the general call address by

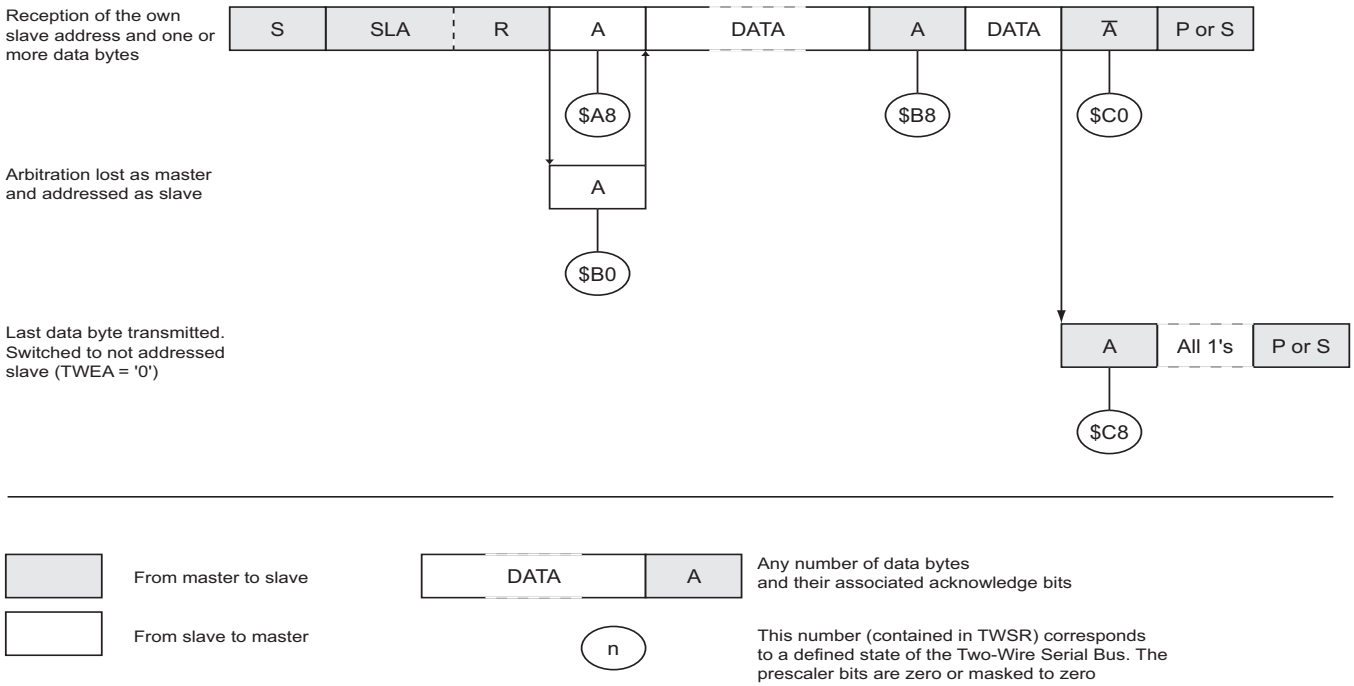
using the 2-wire Serial Bus clock as a clock source. The part will then wake up from sleep and the TWI will hold the SCL clock will low during the wake up and until the TWINT Flag is cleared (by writing it to one). Further data transmission will be carried out as normal, with the 8/16-bit RISC CPU clocks running as normal. Observe that if the 8/16-bit RISC CPU is set up with a long start-up time, the SCL line may be held low for a long time, blocking other data transmissions.

Note that the 2-wire Serial Interface Data Register – TWDR does not reflect the last byte present on the bus when waking up from these sleep modes.

Table 19-4. Status Codes for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free
0xC8	Last data byte in TWDR has been transmitted (TWEA = “0”); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = “1”; a START condition will be transmitted when the bus becomes free

Figure 19-18. Formats and States in the Slave Transmitter Mode



19.7.5 Miscellaneous States

There are two status codes that do not correspond to a defined TWI state, see Table 19-5.

Status 0xF8 indicates that no relevant information is available because the TWINT Flag is not set. This occurs between other states, and when the TWI is not involved in a serial transfer.

Status 0x00 indicates that a bus error has occurred during a 2-wire Serial Bus transfer. A bus error occurs when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. When a bus error occurs, TWINT is set. To recover from a bus error, the TWSTO Flag must set and TWINT must be cleared by writing a logic one to it. This causes the TWI to enter the not addressed Slave mode and to clear the TWSTO Flag (no other bits in TWCR are affected). The SDA and SCL lines are released, and no STOP condition is transmitted.

Table 19-5. Miscellaneous States

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xF8	No relevant state information available; TWINT = “0”	No TWDR action	No TWCR action				Wait or proceed current transfer
0x00	Bus error due to an illegal START or STOP condition	No TWDR action	0	1	1	X	Only the internal hardware is affected, no STOP condition is sent on the bus. In all cases, the bus is released and TWSTO is cleared.

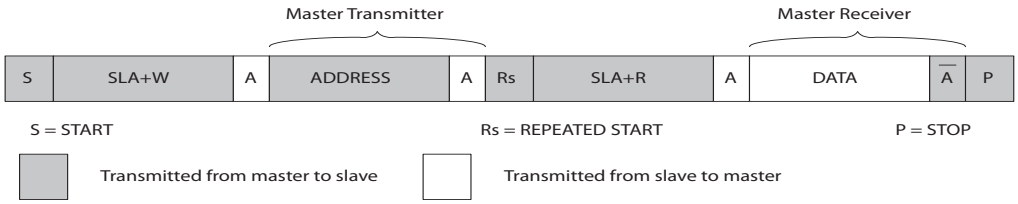
19.7.6 Combining Several TWI Modes

In some cases, several TWI modes must be combined in order to complete the desired action. Consider for example reading data from a serial EEPROM. Typically, such a transfer involves the following steps:

1. The transfer must be initiated.
2. The EEPROM must be instructed what location should be read.
3. The reading must be performed.
4. The transfer must be finished.

Note that data is transmitted both from Master to Slave and vice versa. The Master must instruct the Slave what location it wants to read, requiring the use of the MT mode. Subsequently, data must be read from the Slave, implying the use of the MR mode. Thus, the transfer direction must be changed. The Master must keep control of the bus during all these steps, and the steps should be carried out as an atomical operation. If this principle is violated in a multimaster system, another Master can alter the data pointer in the EEPROM between steps 2 and 3, and the Master will read the wrong data location. Such a change in transfer direction is accomplished by transmitting a REPEATED START between the transmission of the address byte and reception of the data. After a REPEATED START, the Master keeps ownership of the bus. The following figure shows the flow in this transfer.

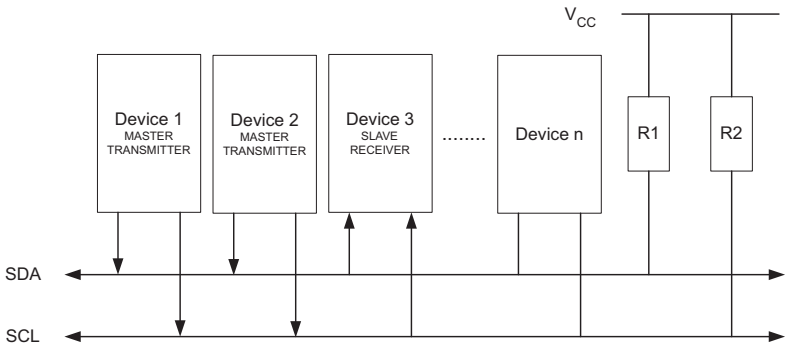
Figure 19-19. Combining Several TWI Modes to Access a Serial EEPROM



19.8 Multi-master Systems and Arbitration

If multiple masters are connected to the same bus, transmissions may be initiated simultaneously by one or more of them. The TWI standard ensures that such situations are handled in such a way that one of the masters will be allowed to proceed with the transfer, and that no data will be lost in the process. An example of an arbitration situation is depicted below, where two masters are trying to transmit data to a Slave Receiver.

Figure 19-20. An Arbitration Example

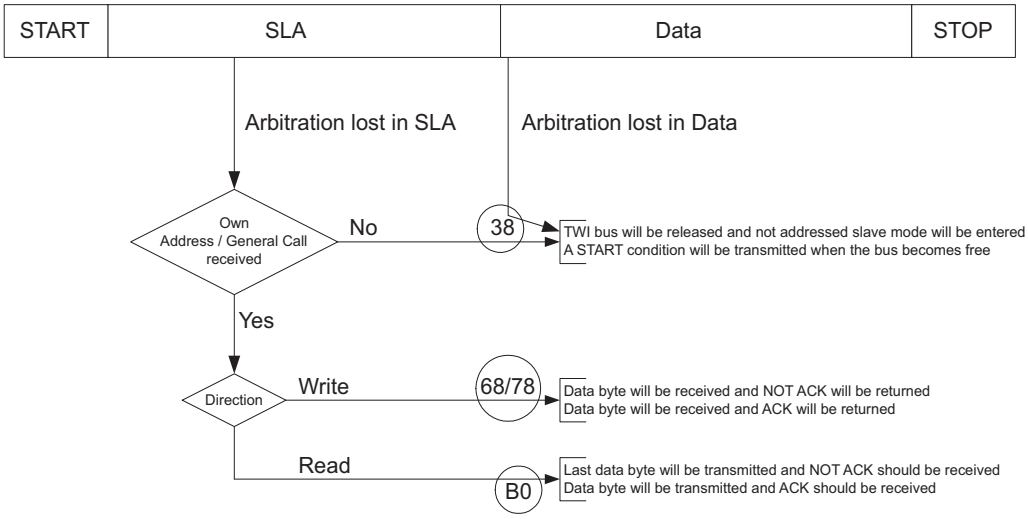


Several different scenarios may arise during arbitration, as described below:

- Two or more masters are performing identical communication with the same Slave. In this case, neither the Slave nor any of the masters will know about the bus contention.
- Two or more masters are accessing the same Slave with different data or direction bit. In this case, arbitration will occur, either in the READ/WRITE bit or in the data bits. The masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Losing masters will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.
- Two or more masters are accessing different slaves. In this case, arbitration will occur in the SLA bits. Masters trying to output a one on SDA while another Master outputs a zero will lose the arbitration. Masters losing arbitration in SLA will switch to Slave mode to check if they are being addressed by the winning Master. If addressed, they will switch to SR or ST mode, depending on the value of the READ/WRITE bit. If they are not being addressed, they will switch to not addressed Slave mode or wait until the bus is free and transmit a new START condition, depending on application software action.

This is summarized in Figure 19-21. Possible status values are given in circles.

Figure 19-21. Possible Status Codes Caused by Arbitration



19.9 TWI Register Description

19.9.1 TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
$0000B8	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..0 – TWI Bit Rate Register**

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See “Bit Rate Generator Unit” on page 194 for calculating bit rates.

19.9.2 TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
$0000BC	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

- **Bit 7 – TWINT: TWI Interrupt Flag**

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 6 – TWEA: TWI Enable Acknowledge Bit**

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

1. The device's own slave address has been received.
2. A general call has been received, while the TWGCE bit in the TWAR is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

- **Bit 5 – TWSTA: TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a Master on the 2-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit and will always read as zero.

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

19.9.3 TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
$0000B9	**TWS7**	**TWS6**	**TWS5**	**TWS4**	**TWS3**	**–**	**TWPS1**	**TWPS0**	**TWSR**
Read/write	R	R	R	R	R	R	R/W	R/W	
Initial value	1	1	1	1	1	0	0	0	0x00

- **Bits 7..3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes are described earlier in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

Table 19-6. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see “Bit Rate Generator Unit” on page 194. The value of TWPS1..0 is used in the equation.

19.9.4 TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
$0000BB	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	1	1	1	1	1	1	1	0xFF

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

- **Bits 7..0 – TWD: TWI Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire Serial Bus.

19.9.5 TWAR – TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
$0000BA	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	1	1	1	1	1	1	1	0	0xFE

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. In multimaster systems, TWAR must be set in masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bits 7..1 – TWA: TWI (Slave) Address Register**

These seven bits constitute the slave address of the TWI unit.

- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the 2-wire Serial Bus.

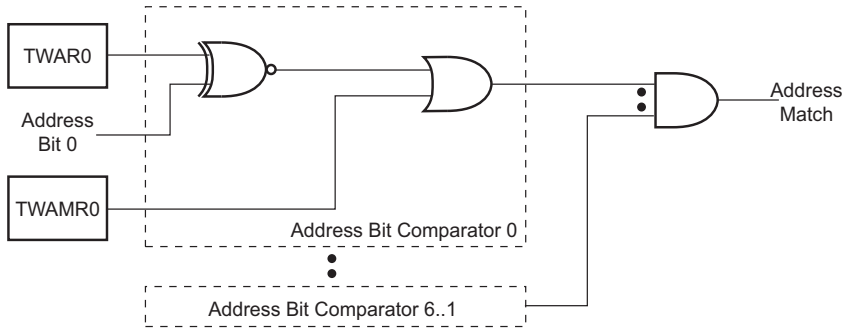
19.9.6 TWAMR – TWI (Slave) Address Mask Register

Bit	7	6	5	4	3	2	1	0	
$0000BD	TWAM [6:0]							-	TWAMR
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bits 7..1 – TWAM: TWI Address Mask**

The TWAMR can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWI Address Register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR. Figure 19-22 shows the address match logic in detail.

Figure 19-22. TWI Address Match Logic, Block Diagram



- **Bit 0 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

20. Keyboard Interface

20.1 Features

- **Allows connection of a 4x4 matrix keyboard**
- **Based on 4 inputs pins and 4 outputs specific pins**
- **Supports slow edge pads to avoid abusive EMC generation (on outputs pads only)**
- **Specific Keyboard pins only available in Full Pin Count package.** See “Pin List Configuration” on page 9.
- **Allows chip wake-up on key pressed event**

20.2 General Description

20.2.1 Overview

The keyboard interfaces with the 8/16-bit RISC CPU core through 3 registers: KBLS, the Keyboard Level Selection register (page 222), KBE, The Keyboard interrupt Enable register (page 222), and KBF, the Keyboard Flag register (page 221).

Note

You can choose to create a Keyboard interface only by using Pin Change Management.This module’s purpose is to ease the creation of little Pinpads.

20.2.2 Interrupt

The keyboard inputs are considered as 4 independant interrupt sources sharing the same interrupt vector. An interrupt enable bit allows global enable or disable of the keyboard interrupt (see Figure 20-1). As detailed in Figure 20-2 each keyboard input can detect a programmable level according to KBLS.x bit value. Level detection is then reported in interrupt flags KBFR.x that can be masked by software using KBER.x bits.

The KBFR.x flags are set by hardware when an active level is on input PAx.

The KBFR register is reset by writing any data inside the KBFR.

Figure 20-1. Keyboard Interface Block Diagram

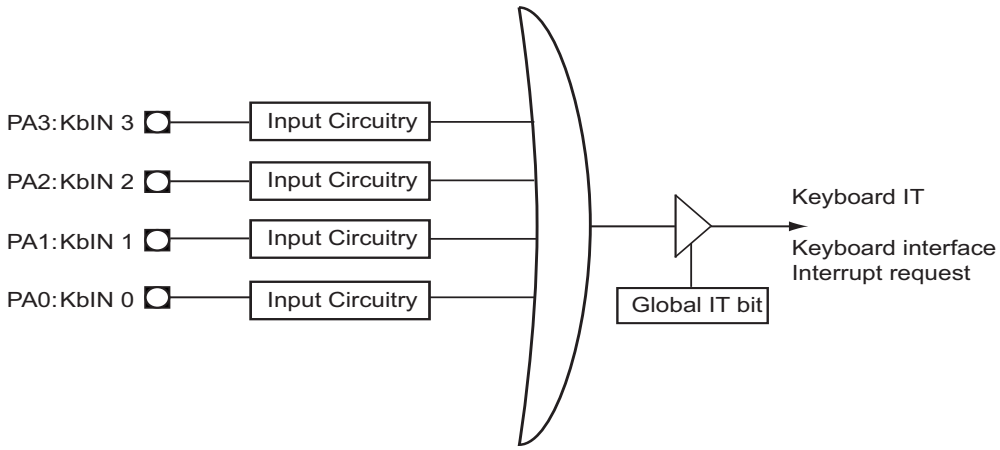
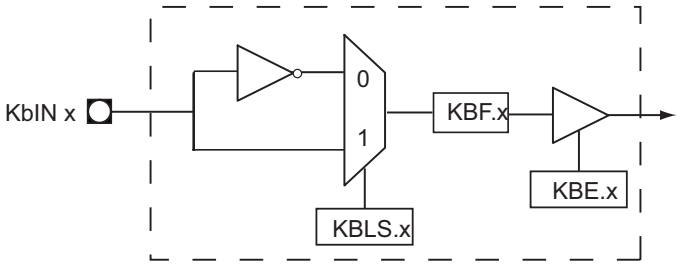


Figure 20-2. Keyboard Input Circuitry



20.2.3 Power Reduction Mode

Keyboard inputs allow wake up from power-down, power-save, Standby, Extended Standbay modes “Active Clock Domains and Wake-up Sources in the Different Sleep Modes.” on page 32.

20.3 Keyboard Register Description

20.3.1 KBFR - KeyBoard Flag Register

Bit	7	6	5	4	3	2	1	0	
$008F	-	-	-	-	KBF3	KBF2	KBF1	KBF0	KBFR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3 - KBF3 : Keyboard Flag for KbIN3 pin**

Set by hardware when the pin function KbIN3 (PA3) detects a programmed level. It generates a Keyboard interrupt request if the KBER.KBE3 bit is set.

Writing a ‘0’ or a ‘1’ into this bit will erase all the bits (including this one) of the KBFR register

- **Bit 2 - KBF2 : Keyboard Flag for KbIN2 pin**

Set by hardware when the pin function KbIN2 (PA2) detects a programmed level. It generates a Keyboard interrupt request if the KBER.KBE2 bit is set.

Writing a ‘0’ or a ‘1’ into this bit will erase all the bits (including this one) of the KBFR register

- **Bit 1 - KBF1: Keyboard Flag for KbIN1 pin**

Set by hardware when the pin function KbIN1 (PA1) detects a programmed level. It generates a Keyboard interrupt request if the KBER.KBE1 bit is set.

Writing a ‘0’ or a ‘1’ into this bit will erase all the bits (including this one) of the KBFR register

- **Bit 0 - KBF0 : Keyboard Flag for KbIN0 pin**

Set by hardware when the pin function KbIN0 (PA0) detects a programmed level. It generates a Keyboard interrupt request if the KBER.KBE0 bit is set.

Writing a ‘0’ or a ‘1’ into this bit will erase all the bits (including this one) of the KBFR register

20.3.2 KBER - KeyBoard Enable Register

Bit	7	6	5	4	3	2	1	0	
$008E	-	-	-	-	KBE3	KBE2	KBE1	KBE0	KBER
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3 - KBE3 : Keyboard KbIN3 Enable bit**

Cleared to enable standard I/O pin.

Set to enable KBFR.KBF3 to generate an interrupt request.

- **Bit 2 - KBE2 : Keyboard KbIN2 Enable bit**

Cleared to enable standard I/O pin.

Set to enable KBFR.KBF2 to generate an interrupt request.

- **Bit 1 - KBE1 : Keyboard KbIN1 Enable bit**

Cleared to enable standard I/O pin.

Set to enable KBFR.KBF1 to generate an interrupt request.

- **Bit 0 - KBE0 : Keyboard KbIN0 Enable bit**

Cleared to enable standard I/O pin.

Set to enable KBFR.KBF0 to generate an interrupt request.

20.3.3 KBLSR - KeyBoard Level Selection Register

Bit	7	6	5	4	3	2	1	0	
$008D	-	-	-	-	KBLS3	KBLS2	KBLS1	KBLS0	KBLSR
Read/write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	0x00

- **Bit 3 - KBLS3 : Keyboard KbIN3 Level Selection bit**

Cleared to enable low level detection on KbIN3.

Set to enable a high level detection on KbIN3.

- **Bit 2 - KBLS2 : Keyboard KbIN2 Level Selection bit**

Cleared to enable low level detection on KbIN2.

Set to enable a high level detection on KbIN2.

- **Bit 1 - KBLS4 : Keyboard KbIN1 Level Selection bit**

Cleared to enable low level detection on KbIN1.

Set to enable a high level detection on KbIN1.

- **Bit 0 - KBLS0 : Keyboard KbIN0 Level Selection bit**

Cleared to enable low level detection on KbIN0.

Set to enable a high level detection on KbIN0.

21. Application Information

21.1 Ordering Information

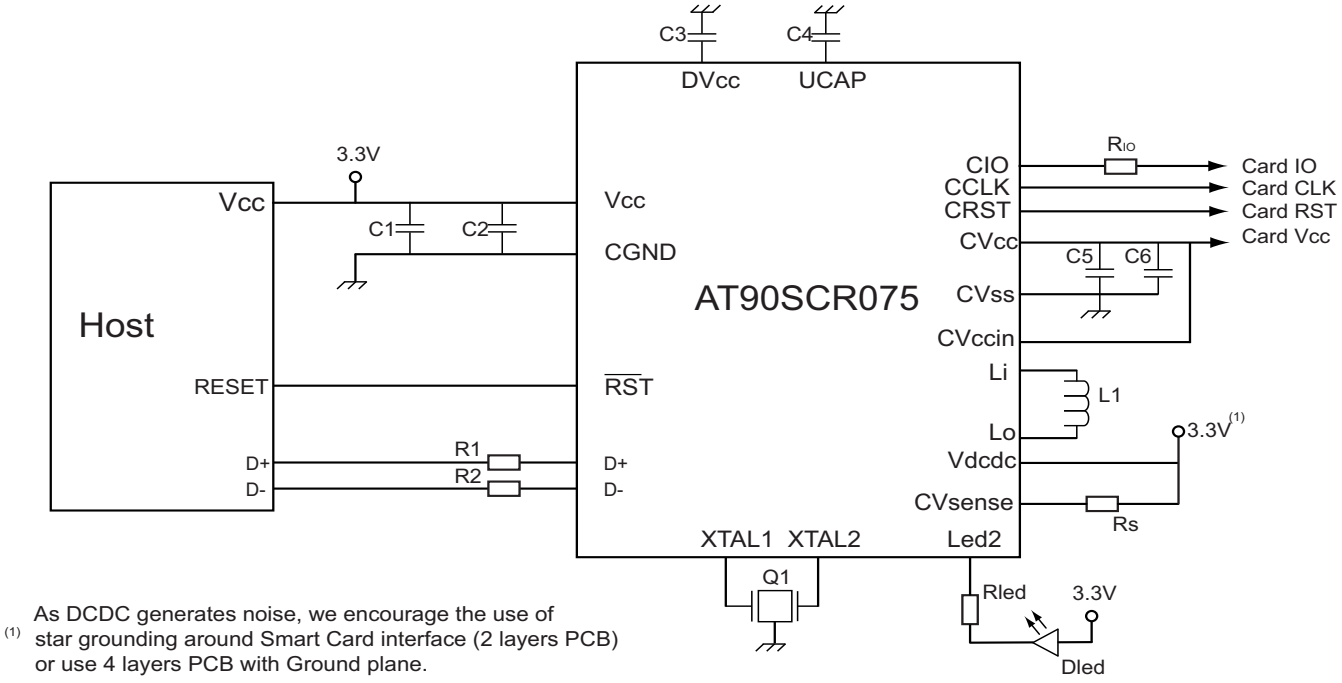
Table 21-1. Ordering Information [1]

Part Number	Voltage Range (V)	Temperature Range	Package	Packing
AT90SCR075Lxxx[2]-Z1T	2.7-5.5	Industrial & Green	QFN32	Tray
AT90SCR075Lxxx-Z1R	2.7-5.5	Industrial & Green	QFN32	Reel
AT90SCR075Hxxx-Z1T	2.7-5.5	Industrial & Green	QFN40	Tray
AT90SCR075Hxxx-Z1R	2.7-5.5	Industrial & Green	QFN40	Reel
AT90SCR060xxx-Z1T	2.7-5.5	Industrial & Green	QFN28	Tray
AT90SCR060xxx-Z1R	2.7-5.5	Industrial & Green	QFN28	Reel

Note:
1. For differences between different configurations please See “Pin List Configuration” on page 9.
2. xxx refers to ROM mask codification..

On all packages, the e-pad must be connected to ground.

21.2 Typical Application



(1) As DCDC generates noise, we encourage the use of star grounding around Smart Card interface (2 layers PCB) or use 4 layers PCB with Ground plane.

Table 21-2. External Components, Bill Of Materials

Reference	Description	Value	Comment
R1, R2	USB Pad Serial Resistor	22Ω +/-10%	-
R_{IO}	Impedance matching Resistor	470Ω	
Rs	DCDC Sense Resistor	(1) 200Ω	Current Sensing: Overcurrent detection
Rled/Dled	LED mechanism	--	Depends on the configuration of the Led Controller
C1	Power Supply Decoupling capacitor	4.7µF +/-10%	Maximum application capacitance allowed by USB standard is 10µF
C2, C6	Power Supply Filter capacitor	100nF	-
C3	Internal Core Regulator Decoupling capacitor	2.2µF +/-10%	Used for internal regulator stability
C4	Internal USB Regulator Decoupling capacitor	2.2µF +/-10%	Used for internal regulator stability
C5	DCDC Decoupling Capacitor	10µF +/-10% esr=100mΩ	Tantalum capacitor Recommended AVX TPSE106-035-200
L1	DCDC inductance	6.8µH esr=20.2mΩ	Recommended Murata LQH43CN6R8M03L
Q1	Crystal	8.0 Mhz	Murata CSTCE8M00GH5L99-R0

Note: 1. Rs value

Note

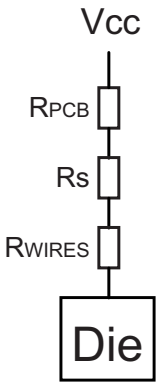
To prevent any perturbation avoiding the DC/DC Converter to work correctly, an RSense Resistance is connected between Power Supply and CVSense Pin. The value of this resistance depends of the resistance of the Bonding Wires and the PCB wires.

The total value of these resistances should be around 500mΩ and the RSense value is chosen in consequence:

$Rs = 500m\Omega - R_{PCB} - R_{Wires}$

R_{PCB} depends of the wire's section and length. Please note that this wire must be designed to support 200mA and be the thinest possible.

RWire is given in "Smart Card Interface Characteristics" on page 237.



21.2.1 Recommendations

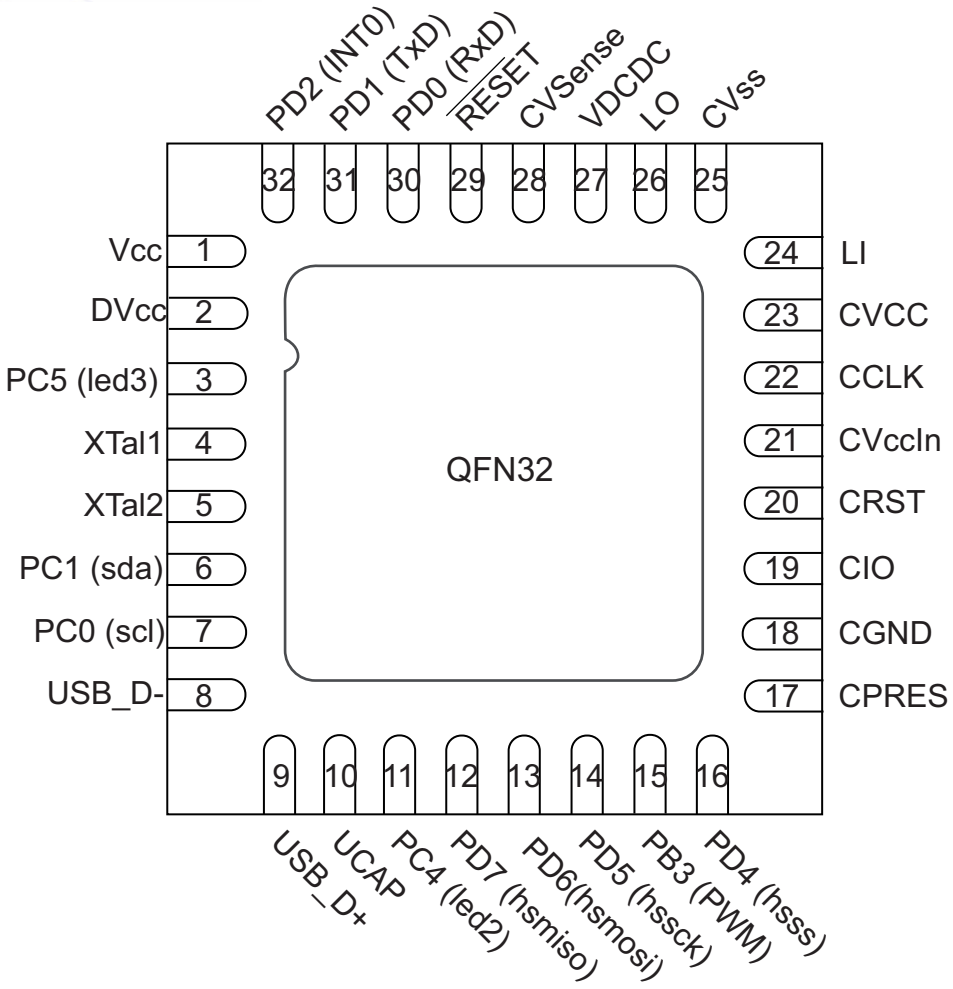
To reach EMV certification, all recommendations are in DC/DC Guideline EMV [R1].

21.2.2 Clock Output

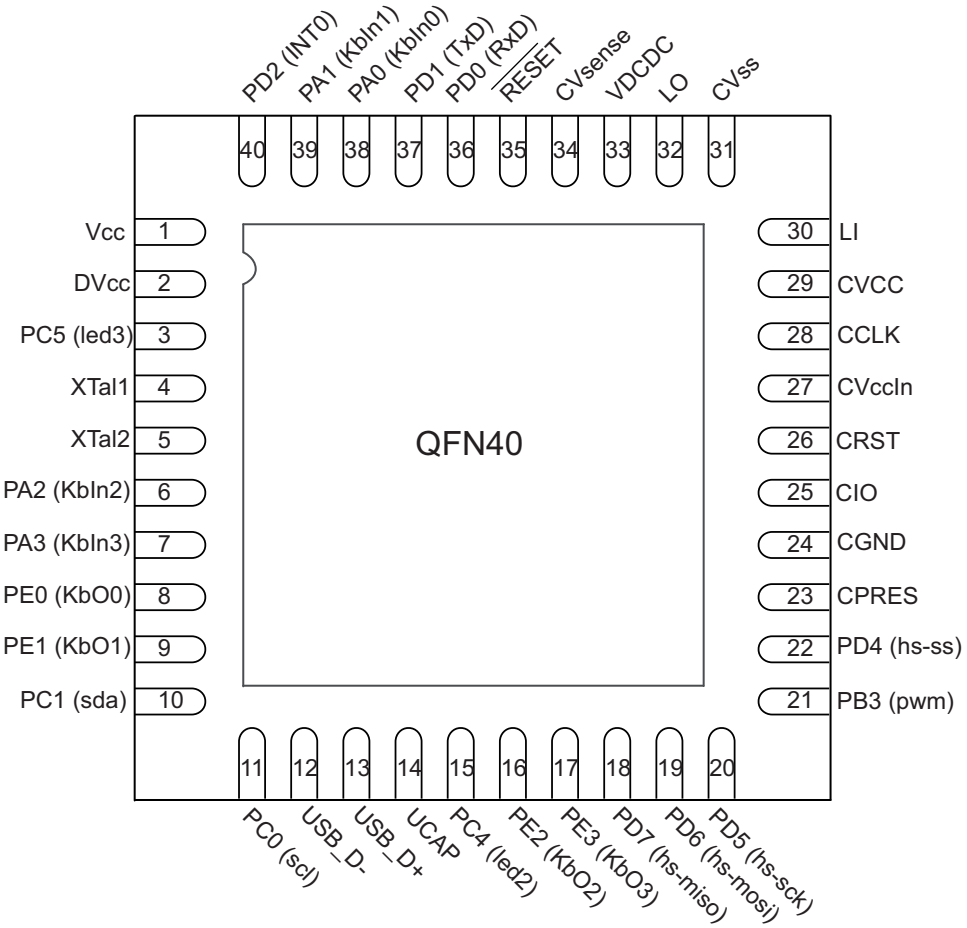
CKOUT option is activated while submitting ROM code, See "Clock Output Buffer" on page 28.. Please use our ISET tool to select this option.

21.3 Pinout

21.3.1 AT90SCR075L - QFN32

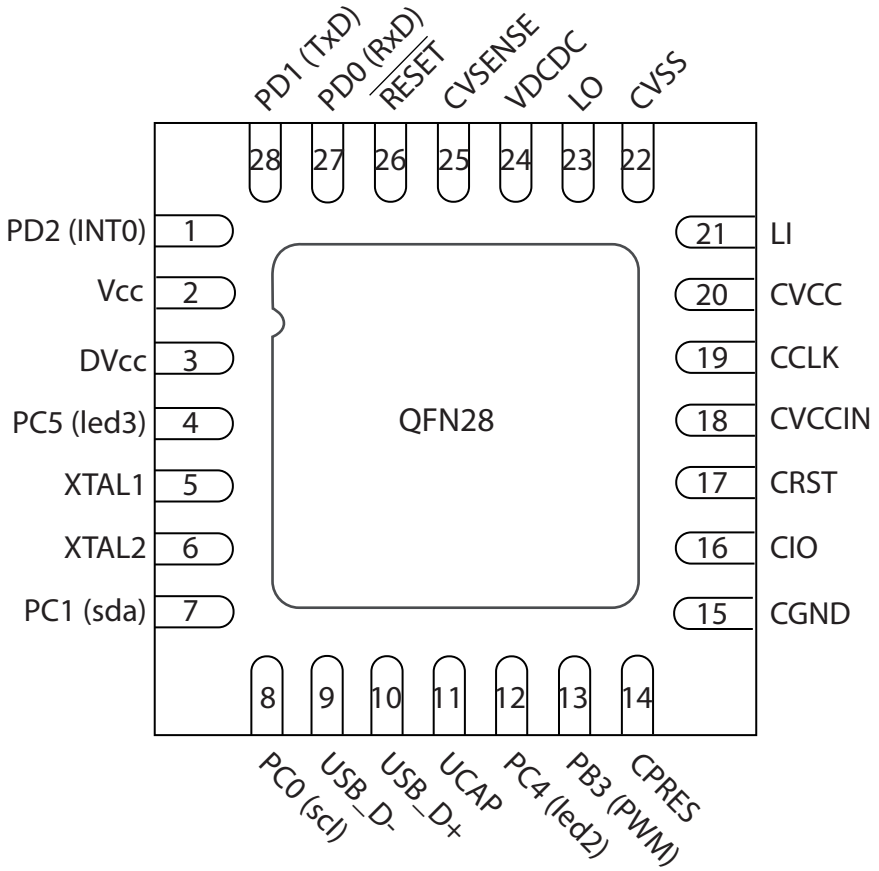


21.3.2 AT90SCR075H - QFN40



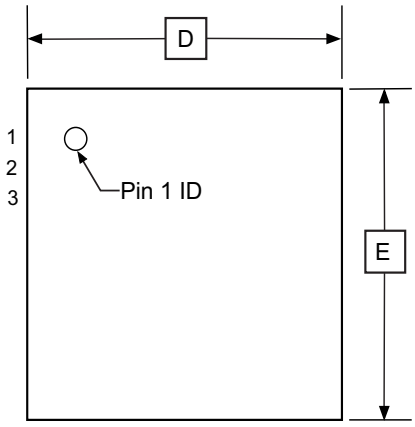
e-pad, underneath package must be connected to ground
NC: Not Connected (pin16, 32)

21.3.3 AT90SCR060 - QFN28

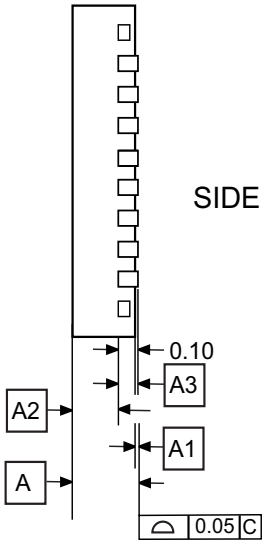


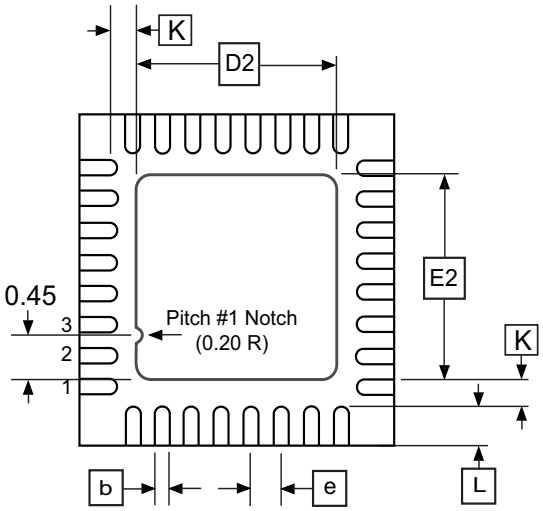
21.4 Mechanical Information

21.4.1 QFN32



TOP VIEW





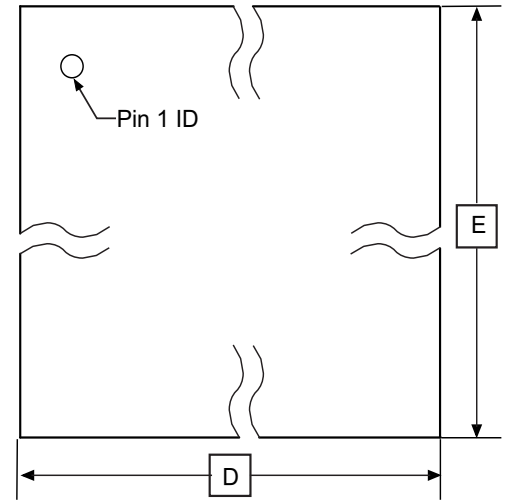
BOTTOM

COMMON DIMENSIONS
(Unit of Measure = mm)

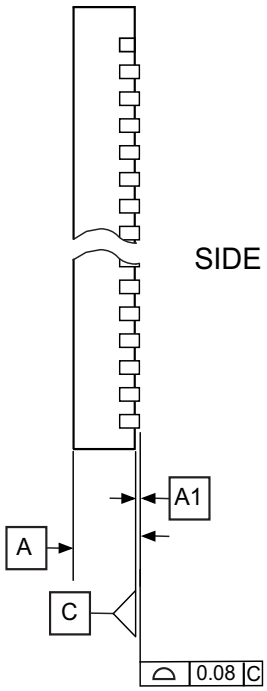
SYMBOL	MIN	NOM	MAX
A	–	–	0.90
A1	–	–	0.05
A2	–	0.65	0.70
A3		0.20 REF	
b	0.18	0.25	0.30
D		5.00 BSC	
D2	3.40	3.50	3.80
E		5.00 BSC	
E2	3.40	3.50	3.60
e		0.50 BSC	
L	0.35	0.40	0.45
K	–	0.35	–

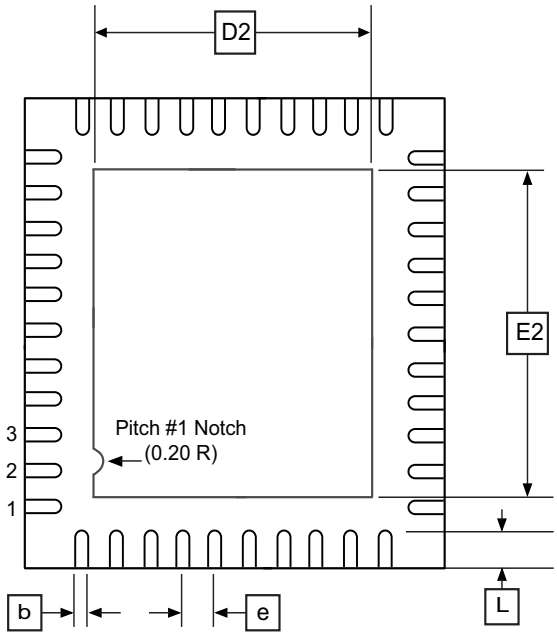
Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2
1. All dimensions are in millimeters.
2. Die thickness allowable is 0.305mm maximum.
3. Dimensioning & Tolerances conform to ASME Y14.5M. 1994.
4. The Pin #1 identifier must be placed on the top surface of the package by using identification mark or other feature of package body.
5. Exact shape and size of this feature is optimal.
6. Package warpage max 0.08mm.
7. Applied for exposed pad and terminals. Exclude embedding part of exposed pard from measuring
8. Applied only to terminals

21.4.2 QFN 40



TOP VIEW





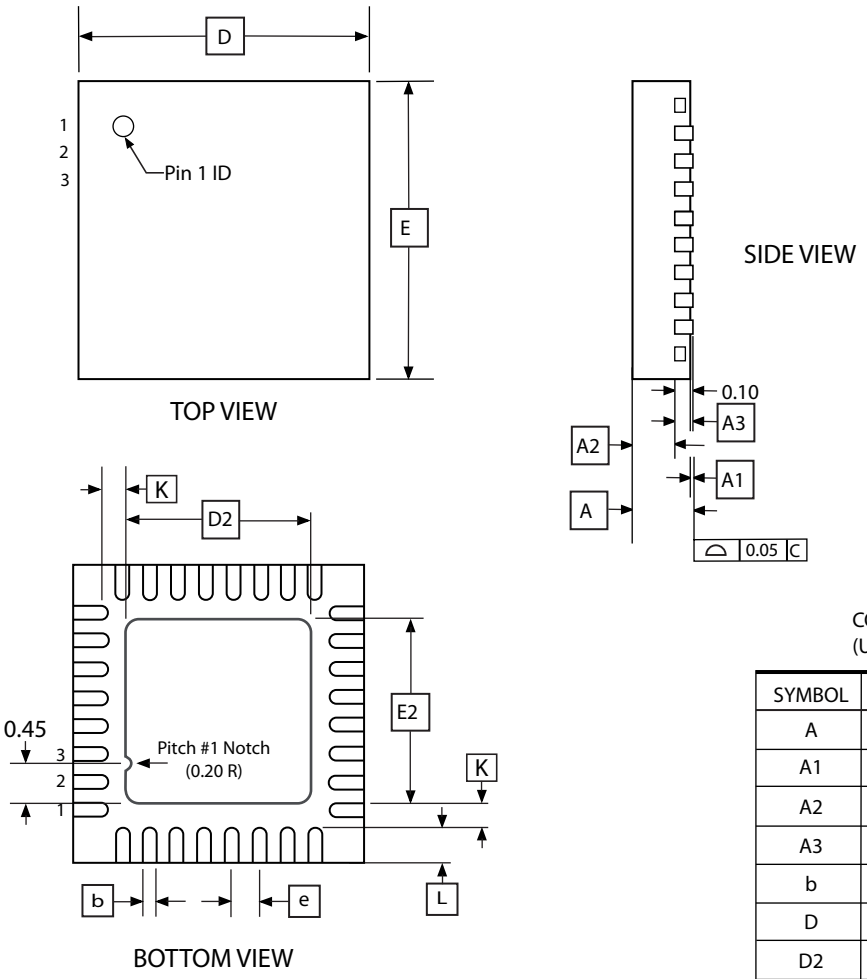
BOTTOM

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX
A	0.80	0.85	0.90
A1	0	–	0.05
D/E		6.00 BSC	
D2/E2	4.00	4.10	4.20
e		0.50 BSC	
L	0.50	0.40	0.30
b	0.23	0.25	0.28

Note: Compliant JEDEC Standard MO-220 variation VKKD-1

21.4.3 QFN28



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX
A	–	–	0.90
A1	–	–	0.05
A2	–	0.65	0.70
A3		0.20 REF	
b	0.18	0.25	0.30
D		5.00 BSC	
D2	3.40	3.50	3.80
E		5.00 BSC	
E2	3.40	3.50	3.60
e		0.50 BSC	
L	0.35	0.40	0.45
K	–	0.35	–

Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2
1. All dimensions are in millimeters. Angles in degree.
2. Coplanarity applied to the exposed as well as the terminals.
Coplanarity shall not exceed 0.08mm
3. Warpage should not exceed 0.10mm.
4. Package length / Package width are considered special characteristic (E).
5. Refer JDEC MO-220.

22. Electrical Characteristics

Absolute Maximum Ratings*

Operating Temperature..... -40°C to +85°C Storage Temperature ..... -65°C to +150°C Voltage on any Pin with respect to Ground..-0.5V to V_{CC}+0.5V Maximum Operating Voltage 6.0V	*NOTICE: Stresses beyond those listed under "Absoute 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.

22.1 Scope

The AT90SCR075_060 application can make the use of the internal DC/DC converter or not. However, it is not possible to supply Smart Card Interface with an external power supply.

Please pay attention to the configuration of the AT90SCR075_060 before reading the table, to match your particular needs.

AT90SCR075_060 without active DC/DC: Vcc range: 2.7V - 5.5V

AT90SCR075_060 with active DC/DC: Vcc range: 2.7V - 5.5V

22.2 DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.(1)	Typ.	Max.(1)	Units
V_{IL}(2)	Input Low Voltage	V_{CC} = 2.7V - 5.5V			0.2 x Vcc(3)	V
V_{IH}(2)	Input High Voltage	V_{CC} = 2.7V - 5.5V	0.7 x Vcc(4)			V
V_{OL}(2)	Output Low Voltage,	V_{CC} = 2.7V - 5.5V			0.4	V
V_{OH}(2)	Output High Voltage,	V_{CC} = 2.7V - 5.5V	0.8V x Vcc			V
I_{IL}(2)	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin low (absolute value)			1	µA
I_{IH}(2)	Input Leakage Current I/O Pin	V_{CC} = 5.5V, pin high (absolute value)			1	µA
R_{RST}	Reset Pull-up Resistor		75		135	kΩ
R_{PU}	I/O Pin Pull-up Resistor		8		25	kΩ
I_{CC}	Power Supply Current(5)(8)	Active 8MHz, V_{CC} = 5V		6.0(8)		mA
		Idle 8MHz, V_{CC} = 5V		450(8)		µA
	Power-down mode(6) / USB Suspend(7)	V_{CC} = 5V		300		µA

Notes: 1. All DC Characteristics contained in this datasheet are based on characterization of other 8/16-bit RISC CPU microcontrollers and peripherals. These values are preliminary values representing design targets, and will be updated after characterization of silicon.

2. These parameters are only for standard I/Os. These only include: Keyboard Inputs, External Interrupts, Standard Serial Peripheral Interface (SPI), High-speed Serial Peripheral Interface (HSSPI), JTAG/LED ports (driving only 2 or 4mA), 2-Wire Interface, Timers I/Os, USART.
3. "Max" means the highest value where the pin is guaranteed to be read as low
4. "Min" means the lowest value where the pin is guaranteed to be read as high
5. Values with "PRR0 – Power Reduction Register 0" and "PRR1 – Power Reduction Register 1" disabled (0x00).
6. Power-down values includes Input in tri-state mode
7. For USB Suspend power-down current measure, you should add 200µA generated by the different resistors on USB D+/D- lines.
8. Measures done with CLKPR = 0x01 (24MHz)

22.3 PORTS

Symbol	Parameter	Condition	Min.(1)	Typ.	Max.(1)	Units
V_{OL}	Output Low Voltage	V_{CC} = 2.7V - 5.5V I_{0L}(2) = 8 mA I_{0L}(3) = 4 mA I_{0L}(4) = 2 mA			0,4	V

Notes:
1. All DC Characteristics contained in this datasheet are based on characterization of other 8/16-bit RISC CPU microcontrollers and peripherals. These values are preliminary values representing design targets, and will be updated after characterization of silicon.
2. Standard Ports
3. LED Ports driving 4 mA
4. LED Ports driving 2 mA and keyboard output

22.4 Clocks

22.4.1 XTAL1 : External Clock In

This clock relates to Core and System clock. See "Clock Sources" on page 27.

Table 22-1. XTAL1 = XIN Clock Electrical Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Units
$1/t_{CPXIN}$	XIN Clock Frequency		8		MHz
XIN_{DC}	XIN Clock Duty Cycle	40%	50%	60%	MHz
C_{IN}	XIN Input Capacitance			10	pF

22.4.2 Calibrated Internal RC Oscillator Accuracy

This clock concerns the Flash/Eeprom specific clock. See "Internal RC Oscillator" on page 47.

Table 22-2. Calibration accuracy of Internal RC Oscillator

	Frequency	V_{cc}	Temperature	Calibration Accuracy
Factory Calibration	10.0 MHz	3V	25°C	±5%
User Calibration	9.5 - 10.5 MHz	2.7V - 5.5V	-40°C - 85°C	±1%

22.5 Communication Interfaces

22.5.1 2-wire Serial Interface Characteristics

Table 22-3 describes the requirements for devices connected to the 2-wire Serial Bus. The AT90SCR075_060 2-wire Serial Interface meets or exceeds these requirements under the noted conditions.

Timing symbols refer to Figure 22-1.

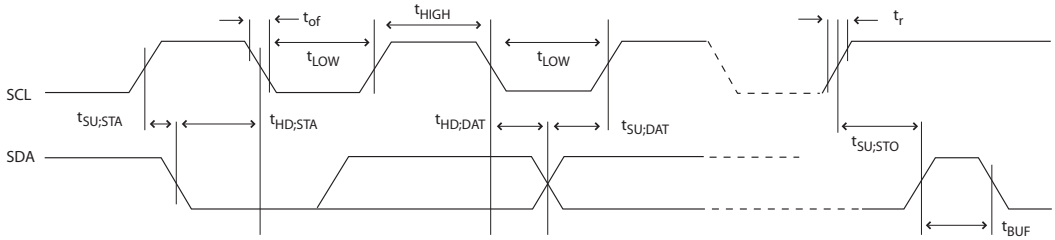
Table 22-3. 2-wire Serial Bus Requirements

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage		-0.5	0.3 V_{CC}	V
V_{IH}	Input High-voltage		0.6 V_{CC}	V_{CC} + 0.5	V
V_{OL} (1)	Output Low-voltage	3 mA sink current	0	0.4	V
t_r (1)(8)	Rise Time for both SDA and SCL		20 + 0.1C_b (2)(3)	300	ns
t_{of} (1)(8)	Output Fall Time from V_{IHmin} to V_{ILmax}	10 pF < C_b < 400 pF (3)	20 + 0.1C_b (2)(3)	250	ns
t_{SP} (1)(8)	Spikes Suppressed by Input Filter		0	50 (2)	ns
C_i (1)(8)	Capacitance for each I/O Pin		–	10	pF
f_{SCL}	SCL Clock Frequency	f_{CK} (4)> max(16f_{SCL}, 250kHz) (5)	0	400	kHz
$t_{HD;STA}$	Hold Time (repeated) START Condition	$f_{SCL} \le$ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
t_{LOW}	Low Period of the SCL Clock	$f_{SCL} \le$ 100 kHz (6)	4.7	–	µs
		f_{SCL} > 100 kHz (7)	1.3	–	µs
t_{HIGH}	High period of the SCL clock	$f_{SCL} \le$ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
$t_{SU;STA}$	Set-up time for a repeated START condition	$f_{SCL} \le$ 100 kHz	4.7	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
$t_{HD;DAT}$	Data hold time	$f_{SCL} \le$ 100 kHz	0	3.45	µs
		f_{SCL} > 100 kHz	0	0.9	µs
$t_{SU;DAT}$	Data setup time	$f_{SCL} \le$ 100 kHz	250	–	ns
		f_{SCL} > 100 kHz	100	–	ns
$t_{SU;STO}$	Setup time for STOP condition	$f_{SCL} \le$ 100 kHz	4.0	–	µs
		f_{SCL} > 100 kHz	0.6	–	µs
t_{BUF}	Bus free time between a STOP and START condition	$f_{SCL} \le$ 100 kHz	4.7	–	µs
		f_{SCL} > 100 kHz	1.3	–	µs

Notes:
1. In AT90SCR075_060, this parameter is characterized and not 100% tested.
2. Required only for f_{SCL} > 100 kHz.
3. Cb = capacitance of one bus line in pF.
4. f_{CK} = CPU clock frequency

5. This requirement applies to all AT90SCR075_060 Two-wire Serial Interface operation. Other devices connected to the Two-wire Serial Bus need only obey the general f_{SCL} requirement.
6. The actual low period generated by the AT90SCR075_060 Two-wire Serial Interface is ($1/f_{SCL}$ - $2/f_{CK}$), thus f_{CK} must be greater than 6 MHz for the low time requirement to be strictly met at f_{SCL} = 100 kHz.
7. The actual low period generated by the AT90SCR075_060 Two-wire Serial Interface is ($1/f_{SCL}$ - $2/f_{CK}$), thus the low time requirement will not be strictly met for f_{SCL} > 308 kHz when f_{CK} = 8 MHz. Still, AT90SCR075_060 devices connected to the bus may communicate at full speed (400 kHz) with other AT90SCR075_060 devices, as well as any other device with a proper t_{LOW} acceptance margin.
8. This parameter is theorical and is not part of the production test flow. It represents a some idea of the size of the parameter, but the exact value is not guaranteed.

Figure 22-1. 2-wire Serial Bus Timing



22.5.2 SPI Timing Characteristics

See Figure 22-2 and Figure 22-3 for details.

Table 22-4. SPI Timing Parameters(1)(1)

	Description	Mode	Min	Typ	Max	
1	SCK period	Master		See Table 20-4		
2	SCK high/low	Master		50% duty cycle		
3	Rise/Fall time	Master		3.6		
4	Setup	Master		10		
5	Hold	Master		10		
6	Out to SCK	Master		15		
7	SCK to out	Master		15		
8	SCK to out high	Master		15		
9	$\overline{SS}$ low to out	Slave		15		
10	SCK_{HSSPI} period	Slave	40			ns
11	SCK_{SPI} period	Slave	170			
12	SCK high/low	Slave	20			
13	Rise/Fall time	Slave			1600	
14	Setup	Slave		10		
15	Hold	Slave		10		
16	SCK to out	Slave		15		
17	SCK to $\overline{SS}$ high	Slave	20			
18	$\overline{SS}$ high to tri-state	Slave		15		
19	$\overline{SS}$ low to SCK	Slave	25			

Notes: 1. Test conditions:
clk_{HSSPI}:24MHz, see “High-Speed SPI Clock - clkHSSPI” on page 26
clk_{IO}: 6MHz, see “I/O Clock - clkI/O” on page 26

2. These values are for information only. They are not tested to production test flow. They are deduced from characterization session.

Figure 22-2. SPI Interface Timing Requirements (Master Mode)

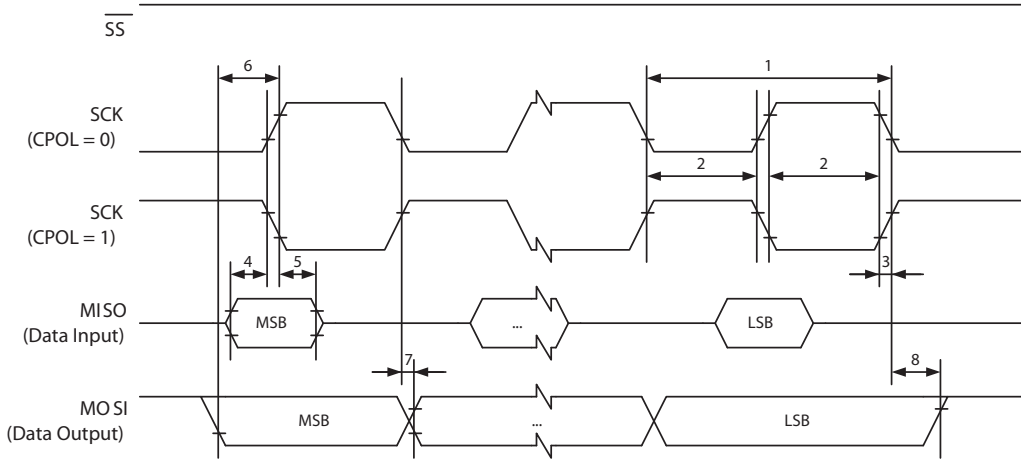
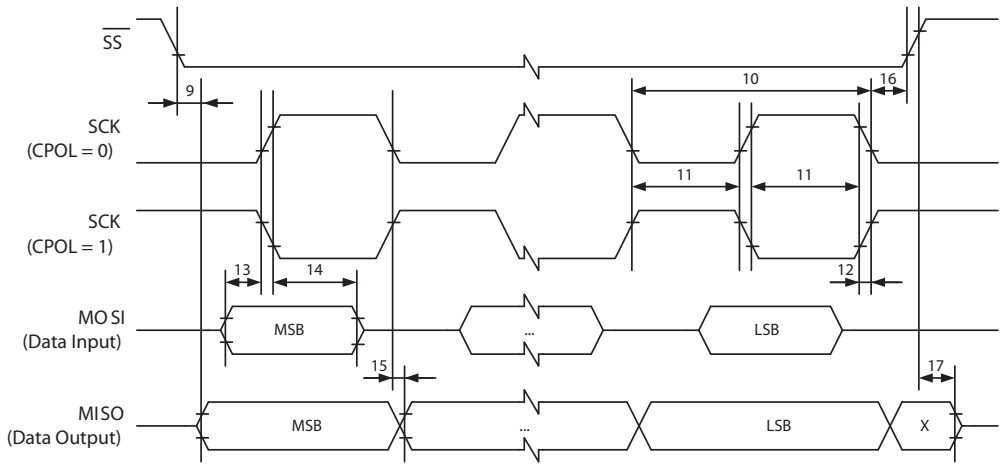


Figure 22-3. SPI Interface Timing Requirements (Slave Mode)



22.6 USB Interface Characteristics

Table 22-5. USB Electrical Parameters

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{IH}	Input Level High (driven)		2.0			V
V_{IHZ}	Input Level High (floating)		2.7			V
V_{IL}	Input Level Low				0.8	V
V_{DI}	Differential Input Sensitivity		0.2			V
V_{CM}	Differential Common Mode Range		0.8		2.5	V
V_{OL1}	Static Output Low				0.3	V
V_{OH1}	Static Output High		2.8		3.6	V
C_{IN}	Input Capacitance				20	pF
t_r	Rise Time	Cout = 50 pF	4		20	ns
t_f	Fall Time	Cout = 50 pF	4		20	ns
t_{rfm}	Rise / Fall Time matching (t_r/t_r)		90		110	%
Z_{drv}	Driver Output Resistance	Steady State Drive	28		44	ohm
t_{drate}	Full Speed Data Rate	Average Bit Rate	11.97		12.03	Mb/s
t_{frame}	Frame Interval		0.9995		1.0005	ms

Vcc must be higher than 3.2V to supply power for USB interface and use correctly this macro.

22.7 Smart Card Interface Characteristics

Table 22-6. Smart Card Class A, 5V (CVcc)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
CVCC	Smart card voltage					
	with Vcc ≥ 2.7V to 3.0V	4.6	5	5.4	V	Load = 55mA
	with Vcc ≥ 3.0V	4.6	5	5.4		Load = 60mA
Vcardok	Vcardok level threshold	4.2		4.6	V	
T_{VHL}	CVCC valid to 0.4V			1	ms	
T_{VLH}	CVCC 0 to valid		1		ms	

Table 22-7. Smart Card Class B, 3V (CVcc)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
CVCC	Smart card voltage with Vcc ≥ 2.7V to 3.0V with Vcc ≥ 3.0V	 2.76 2.76	 3 3	 3.24 3.24	V	 Load = 55mA Load = 60mA
Vcardok	Vcardok level threshold	2.5		2.76	V	
T_{VHL}	CVCC valid to 0.4V			1	ms	
T_{VLH}	CVCC 0 to valid		500		µs	

Table 22-8. Smart Card C, 1.8V (CVcc)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
CVCC	Smart card voltage	1.66	1.8	1.944	V	Load = 35mA
Vcardok	Vcardok level threshold	1.5		1.66	V	
T_{VHL}	CVCC valid to 0.4V			1	ms	
T_{VLH}	CVCC 0 to valid		300		µs	

Table 22-9. Resistance of the Bonding Wires, R_{BW}

Value	Unit	Package
220	mOhms	QFN40 / QFN32
50	mOhms	QFN28

Table 22-10. Smart Card Card Presence (CPRES)

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{OL1}	CPRES weak pull-up output current Vcc = 3V	20		35	µA	Short to VSS PULLUP = 1: Internal pull-up active

23. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	SCICR	RESET	CARDDET	VCARD1	VCARD0	UART	WTEN	CREP	CONV	129
(0xFE)	SCCON	CLK	-	-	-	CARDIO	CARDCLK	CARDRST	CARDVCC	131
(0xFD)	SCISR	SCTBE	CARDIN	-	VCARDOK	SCWTO	SCTC	SCRC	SCPE	132
(0xFC)	SCIIR	SCTBI	-	-	VCARDERR	SCWTI	SCTI	SCRI	SCPI	133
(0xFB)	SCIER	ESCTBI	CARDINE	-	IVCARDER	ESCWTI	ESCTI	ESCRI	ESCPI	134
(0xFA)	SCSR	-	BGTEN	-	CREPSEL	CPRESRES	-	-	-	135
(0xF9)	SCIBUF	SCIBUFD [7..0]								136
(0xF8)	SCETUH	COMP	-	-	-	-	ETU [10:8]			137
(0xF7)	SCETUL	ETU [7:0]								137
(0xF6)	SCGTH	-	-	-	-	-	-	-	GT8	137
(0xF5)	SCGTL	GT [7:0]								137
(0xF4)	SCWT3	WT31	WT30	WT29	WT28	WT27	WT26	WT25	WT24	138
(0xF3)	SCWT2	WT23	WT22	WT21	WT20	WT19	WT18	WT17	WT16	138
(0xF2)	SCWT1	WT15	WT14	WT13	WT12	WT11	WT10	WT9	WT8	138
(0xF1)	SCWT0	WT7	WT6	WT5	WT4	WT3	WT2	WT1	WT0	138
(0xF0)	SCICLK	-	-	SCICLK5	SCICLK4	SCICLK3	SCICLK2	SCICLK1	SCICLK0	138
(0xEF)	DCCR	DCON	DCRDY	DCBUSY	-	-	-	-	-	143
(0xEE)	SCISRCR	-	-	-	-	SRC3	SRC2	SRC1	SRC0	139
(0xED)	USBDMAB	-	USBDMAB [6..0]							112
(0xEC)	USBDMADH	-	-			USBDMAD [11..8]				111
(0xEB)	USBDMADL	USBDMAD [7..0]								111
(0xEA)	USBDMACS	-	EPS2	EPS1	EPS0	-	DMAERR	DMAIR	DMAR	109
(0xE9)	USBFNH	-	-	-	FNEND	FNERR	FN10	FN9	FN8	108
(0xE8)	USBFNL	-	FN6	FN5	FN4	FN3	FN2	FN1	FN0	108
(0xE7)	USBFA	-	FADD6	FADD5	FADD4	FADD3	FADD2	FADD1	FADD0	107
(0xE6)	USBGS	-	-	-	-	RSMON	RMWUE	FCF	FAF	106
(0xE5)	USBRSTE	RST	RSTE6	RSTE5	RSTE4	RSTE3	RSTE2	RSTE1	RSTE0	106
(0xE4)	USBEIM	EP7IM	EP7IM	EP5IM	EP4IM	EP3IM	EP2IM	EP1IM	EP0IM	103
(0xE3)	USBEI	-	EP6I	EP5I	EP4I	EP4I	EP2I	EP1I	EP0I	102
(0xE2)	USBPIM	-	-	-	-	SOFIM	RMWUIM	RESIM	SUSIM	101
(0xE1)	USBPI	-	-	-	FEURI	SOFI	RMWUI	RESI	SUSI	101
(0xE0)	USBCR	URMWU	-	UPUC	-	-	-	USBE	-	100
(0xDF)	HSSPIDMAB	-	-	-	HSSPIDMAB [4..0]					186
(0xDE)	HSSPIDMADH	-	-	HSSPIDMAD [13..8]						186
(0xDD)	HSSPIDMADL	HSSPIDMAD [7..0]								186
(0xDC)	HSSPIDMACS	-	-	-	-	-	DMAERR	DMADIR	DMAR	184
(0xDB)	HSSPICR	-	-	-	-	-	STTTO	RETTO	CS	181
(0xDA)	HSSPIIR	TIMEOUT	BTD	RCVOF	NSSRE	NSSFE	-	-	-	179
(0xD9)	HSSPICFG	SPICKDIV2	SPICKDIV1	SPICKDIV0	DPRAM	CPHA	CPOL	MSTR	SPIEN	178
(0xD8)	HSSPISR	-	RXBUFRDY	TXBUFFREE	DPRAMRDY	NSS	RXBUFF	TXBUFE	SPICKRDY	181
(0xD7)	HSSPITDR	HSSPITDD [7..0]								182
(0xD6)	HSSPIRDR	HSSPIRDD [7..0]								182
(0xD5)	HSSPIGTR	HSSPIGTD [7..0]								182
(0xD4)	HSSPIIER	TIMEOUTIE	BTDIE	RCVOFIE	NSSIE	-	-	-	-	180
(0xD3)	HSSPICNT	-	-	-	HSSPICNT [4..0]					182
(0xD2)	HSSPITOH	SPITIMEOUT[15:8]								183
(0xD1)	HSSPITOL	SPITIMEOUT[7:0]								183
(0xD0)	Reserved	-	-	-	-	-	-	-	-	
(0xCF)	Reserved	-	-	-	-	-	-	-	-	
(0xCE)	Reserved	-	-	-	-	-	-	-	-	
(0xCD)	USBFCEX	EPE	-	-	-	-	EPDIR	EPTYP1	EPTYP0	105
(0xCC)	USBDBCEX	-	BCT6	BCT5	BCT4	BCT3	BCT2	BCT1	BCT0	105
(0xCB)	USBCSEX	-	IERR	FSTALL	TXPB	STSENT	RXSETUP	RCVD	TXC	103
(0xCA)	USBENUM	-	-	-	-	-	ENUM2	ENUM1	ENUM0	103
(0xC9)	Reserved	-	-	-	-	-	-	-	-	
(0xC8)	Reserved	-	-	-	-	-	-	-	-	
(0xC7)	Reserved	-	-	-	-	-	-	-	-	
(0xC6)	UDR0	USART0 I/O Data Register								161
(0xC5)	UBRR0H	-	-	-	-	UART0 Baud Rate Register High Byte				164
(0xC4)	UBRR0L	UART0 Baud Rate Register Low Byte								164
(0xC3)	Reserved	-	-	-	-	-	-	-	-	
(0xC2)	UCSR0C	-	-	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	-	163
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	162
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	161
(0xBF)	Reserved	-	-	-	-	-	-	-	-	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBE)	Reserved	-	-	-	-	-	-	-	-	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	218
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	216
(0xBB)	TWDR	2-wire Serial Interface Data Register								218
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	218
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	217
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								215
(0xB7)	Reserved	-	-	-	-	-	-	-	-	
(0xB6)	Reserved	-	-	-	-	-	-	-	-	
(0xB5)	Reserved	-	-	-	-	-	-	-	-	
(0xB4)	Reserved	-								
(0xB3)	Reserved	-								
(0xB2)	Reserved	-								
(0xB1)	Reserved	-	-	-	-	-	-	-	-	
(0xB0)	Reserved	-	-	-	-	-	-	-	-	
(0xAF)	Reserved	-								
(0xAE)	Reserved	-	-	-	-	-	-	-	-	
(0xAD)	Reserved	-								
(0xAC)	Reserved	-	-	-	-	-		-	-	
(0xAB)	Reserved	-		-		-		-	-	
(0xAA)	Reserved	-		-		-				
(0xA9)	Reserved	-	-	-	-	-	-	-	-	
(0xA8)	Reserved	-	-	-	-	-	-	-	-	
(0xA7)	Reserved	-	-	-	-	-	-	-	-	
(0xA6)	Reserved	-	-	-	-	-	-	-	-	
(0xA5)	Reserved	-								
(0xA4)	Reserved	-								
(0xA3)	Reserved	-	-	-	-	-	-			
(0xA2)	Reserved	-								
(0xA1)	Reserved	-				-				
(0xA0)	Reserved	-	-	-	-	-	-	-	-	
(0x9F)	Reserved	-	-	-	-	-	-	-	-	
(0x9E)	Reserved	-	-		-	-	-	-	-	
(0x9D)	Reserved	-	-		-	-	-	-	-	
(0x9C)	Reserved	-	-	-	-	-	-			
(0x9B)	Reserved	-								
(0x9A)	Reserved	-	-	-	-	-	-	-	-	
(0x99)	Reserved	-	-	-	-	-	-	-	-	
(0x98)	Reserved	-	-	-	-	-	-	-	-	
(0x97)	Reserved	-	-	-	-	-	-	-	-	
(0x96)	Reserved	-	-	-	-	-	-	-	-	
(0x95)	Reserved	-	-	-	-	-	-	-	-	
(0x94)	Reserved	-								
(0x93)	Reserved	-								
(0x92)	Reserved	-								
(0x91)	Reserved	-								
(0x90)	Reserved	-								
(0x8F)	KBFR	-	-	-	-	KBF3	KBF2	KBF1	KBF0	221
(0x8E)	KBER	-	-	-	-	KBE3	KBE2	KBE1	KBE0	222
(0x8D)	KBLSR	-	-	-	-	KBLS3	KBLS2	KBLS1	KBLS0	222
(0x8C)	Reserved	-	-	-	-	-	-	-	-	
(0x8B)	Reserved	-								
(0x8A)	Reserved	-								
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								89
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								89
(0x87)	Reserved	-								
(0x86)	Reserved	-								
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								89
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								89
(0x83)	Reserved	-	-	-	-	-	-	-	-	
(0x82)	Reserved	-	-	-	-	-	-	-	-	
(0x81)	TCCR1B	-	-	-	-	WGM12	CS12	CS11	CS10	88
(0x80)	TCCR1A	-	-	-	-	-	-	WGM11	WGM10	87
(0x7F)	Reserved	-	-	-	-	-	-	-	-	
(0x7E)	Reserved	-	-	-	-	-	-	-	-	
(0x7D)	Reserved	-	-	-	-	-	-	-	-	
(0x7C)	Reserved	-	-	-	-	-	-	-	-	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7B)	Reserved	-								
(0x7A)	Reserved	-								
(0x79)	Reserved	-	-	-	-	-	-	-	-	
(0x78)	Reserved	-	-	-	-	-	-	-	-	
(0x77)	Reserved	-	-	-	-	-	-	-	-	
(0x76)	Reserved	-	-	-	-	-	-	-	-	
(0x75)	LEDCR	-LED3		LED2		-		-		57
(0x74)	Reserved	-	-	-	-	-	-	-	-	
(0x73)	PCMSK3	-	-	-	-	PCINT27	PCINT26	PCINT25	PCINT24	53
(0x72)	Reserved	-	-	-	-	-	-	-	-	
(0x71)	Reserved	-	-	-	-	-	-	-	-	
(0x70)	Reserved	-	-	-	-	-	-	-	-	
(0x6F)	TIMSK1	-	-	-	-	-	-	OCIE1A	TOIE1	90
(0x6E)	TIMSK0	-	-	-	-	-	-	OCIE0A	TOIE0	79
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	-	PCINT18	PCINT17	PCINT16	53
(0x6C)	PCMSK1	-	-	-	-	PCINT11	-	-	-	54
(0x6B)	PCMSK0	-	-	-	-	PCINT3	PCINT2	PCINT1	PCINT0	54
(0x6A)	Reserved	-	-	-	-	-	-	-	-	
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	50
(0x68)	PCICR	-	-	-	-	PCIE3	PCIE2	PCIE1	PCIE0	52
(0x67)	Reserved	-	-	-	-	-	-	-	-	
(0x66)	Reserved	-	-	-	-	-	-	-	-	
(0x65)	PRR1	-	-	PRKB	-	PRSCI	PRHSSPI	PRUSB	-	34
(0x64)	PRR0	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	-	PRUSART0	-	33
(0x63)	Reserved	-	-	-	-	-	-	-	-	
(0x62)	PLLCR	PLLMUX	-	-	-	-	-	LOCK	ON/OFF	29
(0x61)	CLKPR	-	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	29
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	44
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	16
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	18
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	18
0x3C (0x5C)	Reserved	-	-	-	-	-	-	-	-	
0x3B (0x5B)	Reserved	-	-	-	-	-	-	-	-	
0x3A (0x5A)	Reserved	-	-	-	-	-	-	-	-	
0x39 (0x59)	Reserved	-	-	-	-	-	-	-	-	
0x38 (0x58)	Reserved	-	-	-	-	-	-	-	-	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	381
0x36 (0x56)	Reserved	-	-	-	-	-	-	-	-	
0x35 (0x55)	Reserved	-	-	-	PUD	-	-	-	-	61,
0x34 (0x54)	MCUSR	-	-	-	JTRF	WDRF	BORF	EXTRF	PORF	43
0x33 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	32
0x32 (0x52)	Reserved	-	-	-	-	-	-	-	-	
0x31 (0x51)	OCDR	D7/IDRD	D6	D5	D4	D3	D2	D1	D0	366
0x30 (0x50)	Reserved	-	-	-	-	-	-	-	-	
0x2F (0x4F)	Reserved	-	-	-	-	-	-	-	-	
0x2E (0x4E)	Reserved	-								
0x2D (0x4D)	Reserved	-	-	-	-	-	-	-	-	
0x2C (0x4C)	Reserved	-	-	-	-	-	-	-	-	
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								23
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								23
0x29 (0x49)	Reserved	-	-	-	-	-	-	-	-	
0x28 (0x48)	Reserved	-								
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								79
0x26 (0x46)	TCNT0	Timer/Counter0 (8 Bit)								79
0x25 (0x45)	TCCR0B	FOC0A	-	-	-	WGM02	CS02	CS01	CS00	78
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	-	-	-	-	WGM01	WGM00	76
0x23 (0x43)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC	91
0x22 (0x42)	Reserved	-	-	-	-	-				
0x21 (0x41)	Reserved	-								
0x20 (0x40)	Reserved	-								
0x1F (0x3F)	Reserved	-	-	-	-	-	-	-	-	
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								24
0x1D (0x3D)	EIMSK	-	-	-	-	INT3	INT2	INT1	INT0	51
0x1C (0x3C)	EIFR	-	-	-	-	INTF3	INTF2	INTF1	INTF0	51
0x1B (0x3B)	PCIFR	-	-	-	-	PCIF3	PCIF2	PCIF1	PCIF0	52
0x1A (0x3A)	Reserved	-	-	-	-	-	-	-	-	
0x19 (0x39)	Reserved	-	-	-	-	-	-	-	-	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x18 (0x38)	Reserved	-	-	-	-	-	-	-	-	
0x17 (0x37)	Reserved	-	-	-	-	-	-	-	-	
0x16 (0x36)	TIFR1	-	-	-	-	-	-	OCF1A	TOV1	90
0x15 (0x35)	TIFR0	-	-	-	-	-	-	OCF0A	TOV0	80
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-	
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-	
0x11 (0x31)	Reserved	-	-	-	-	-	-	-	-	
0x10 (0x30)	Reserved	-	-	-	-	-	-	-	-	
0x0F (0x2F)	Reserved	-	-	-	-	-	-	-	-	
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	63
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	64
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	64
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	63
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	63
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	63
0x08 (0x28)	PORTC	-	-	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	62
0x07 (0x27)	DDRC	-	-	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	63
0x06 (0x26)	PINC	-	-	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	63
0x05 (0x25)	Reserved	-	-	-	-	-	-	-	-	
0x04 (0x24)	Reserved	-	-	-	-	-	-	-	-	
0x03 (0x23)	Reserved	-	-	-	-	-	-	-	-	
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	62
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	62
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	62

1. For compatiliblity with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00-$1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, this clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses $00-$3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The AT90SCR075_060 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from $60-$FF, only the ST/STS/STD and LD/LDS/LDD instructions ca be used.

All ‘Reserved’ registers must not be read or written.
Writing or Reading these registers may generate unhandled state.

24. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	4
ICALL		Indirect Call to (Z)	PC ← Z	None	4
CALL	k	Direct Subroutine Call	PC ← k	None	5
RET		Subroutine Return	PC ← STACK	None	5
RETI		Interrupt Return	PC ← STACK	I	5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
		BIT AND BIT-TEST INSTRUCTIONS			
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
		DATA TRANSFER INSTRUCTIONS			
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
ELPM	Rd, Z	Extended Load Program Memory	Rd ← (Z)	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Reference List

Check with your local Seal SQ sales office that you have the latest revision of the following documents.

[R1]	DC/DC Guideline EMV	TPR0535

Revision History

Document Details

Title: Datasheet Preliminary

Literature Number: TPR0521D

Date: 16Jan23

- **Revision D:**
  - New Seal SQ template
- **Revision C:**
  - New WISeKey template
- **Revision A:**
  - First Release

Headquarters

SealSQ
Arteparc de Bachasson - Bat A
Rue de la Carrière de Bachasson
CS 70025
13590 Meyreuil - France
Tel: +33 (0)4-42-370-370
Fax: +33 (0)4-42-370-024

Product Contact

Web Site
www.sealsq.com

Technical Support
scr@wisekey.com

Sales Contact
sales@wisekey.com