Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of "Quality Parts,Customers Priority,Honest Operation,and Considerate Service",our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Atmel®

ATA6286C

Embedded AVR Microcontroller with LF Receiver and UHF Transmitter

DATASHEET

Features

- 8-bit Atmel® AVR® RISC low-power microcontroller
- Embedded ultra-low-power flash 8-bit AVR (Atmel ATA6289) with on-chip sensitive LF receiver, temperature sensor and integrated UHF transmitter IC
- 8KB of in-system self-programmable Flash memory and 320 Bytes of EEPROM
- 8-bit AVR RISC low-power microcontroller requires typically < 0.5µA sleep current with active interval timer
- 8mA active current requested at 6dBm output power in transmission mode within 432MHz to 448MHz (Atmel ATA5757) frequency range
- ASK and FSK modulation with up to 20Kbaud data rate in Manchester mode
- Programmable 125kHz wake-up receiver channel with typically 2.3µA current consumption in listening mode
- Typically 25mbar ADC resolution (measured with a typical pressure sensor)
- Low-power measurement mode for directly connected capacitive sensors with typically 350µA in 30ms
- Three interfaces for simple capacitive sensors (3pF to 16pF)
- One interface can be configured for motion wake-up (1pF to 4pF)
- Operation voltage 2V to 3.6V for single Li-Cell power supply
- Operating temperature –40°C to +85°C and storage temperature –40°C to +85°C
- Less than 10 external passive components
- QFN 32 (5 x 5) package

Applications

- Active RFID
- Access control

1. Description

The Atmel® ATA6286C is a embedded ultra-low-power AVR 8-bit microcontroller ICs with integrated RF transmission and LF receiving functionality for wake-up purposes in a small QFN32 package.

The RF transmission is based on well-known Atmel IPs for 432MHz to 448MHz (Atmel ATA5757) frequency range with a typical output power up to 6dBm. They are suited for ASK and FSK modulation with up to 20Kbaud data rate in Manchester mode.

The integrated programmable 125kHz LF receiver channel has extremely low current consumption in active listening mode. As a result, the LF receiver is particularly well suited for wake-up purposes.

The programmable AVR 8-bit Flash microcontroller includes 8KB of in-system self-programmable Flash memory and 320 bytes of EEPROM thus allowing the system integrator to install field programmable firmware to meet flexible system requirements on different platforms. The Atmel ATA6286C is configurable to meet extremely low-power requests in sleep mode, measurement mode and transmission mode.

Furthermore, a low-power interval timer and brown-out detector is integrated.

The Atmel ATA6286Cis suited for powering single cell battery applications. Therefore, a single LiMnO2 battery coin cell can supply a whole system.

The AVR 8-bit Flash microcontroller also delivers a dedicated integrated simple capacitive sensor interface, as well as an on-chip calibratable temperature sensor.

Three sensor interfaces are available for capacitive sensing in a range of 3pF to 16pF. In addition, one channel can be configured for motion sensing in a range of 1pF to 4pF.

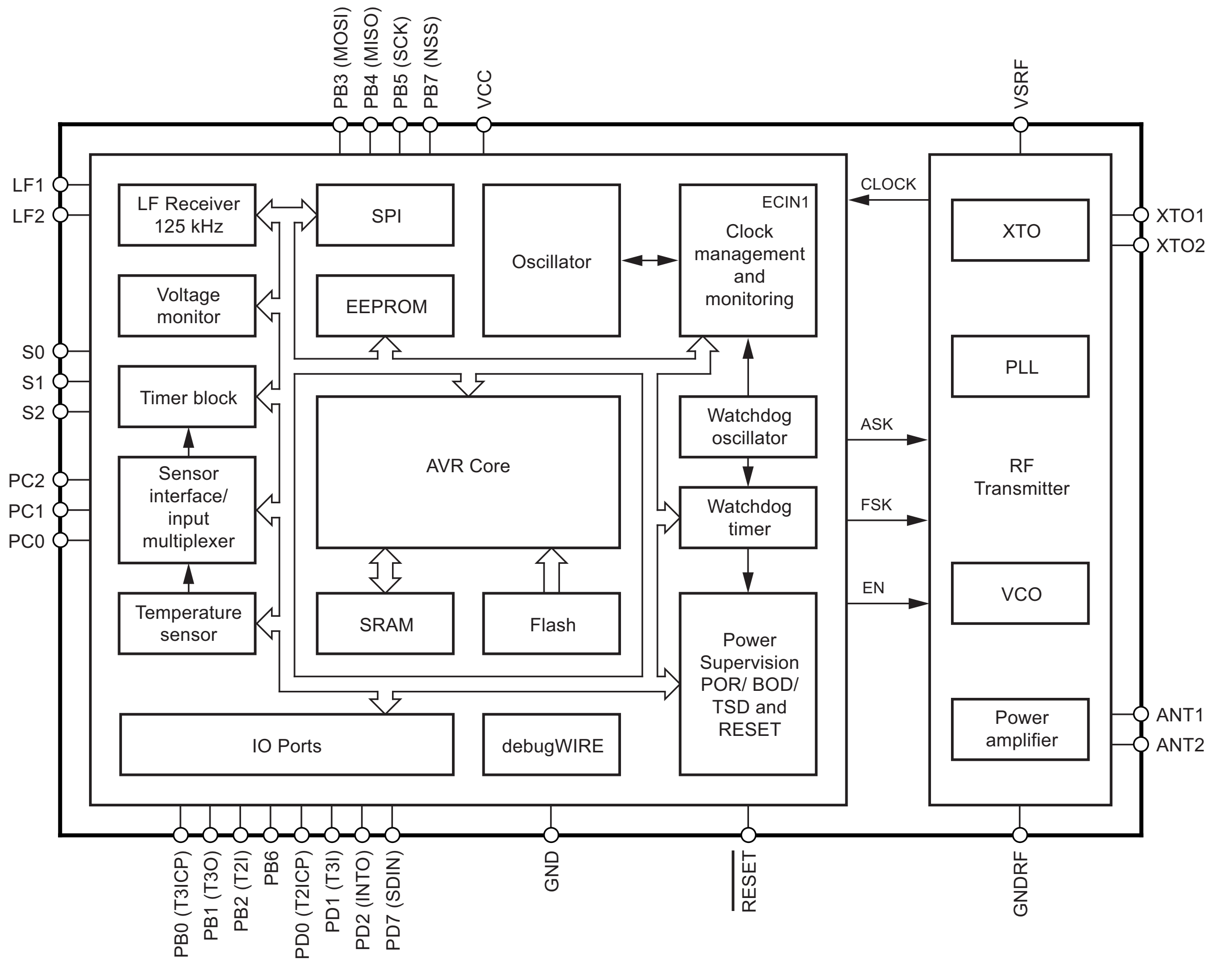
The Atmel ATA6286C is thus proposed for applications requiring very extreme low-power consumption such as active RFID tags with an extended service life.

Owing to the integrated capacitive sensor interface, these ICs may also be used in pressure sensor applications.

2. Overview

2.1 Block Diagram

Figure 2-1. Atmel ATA6286C Block Diagram (MCP)



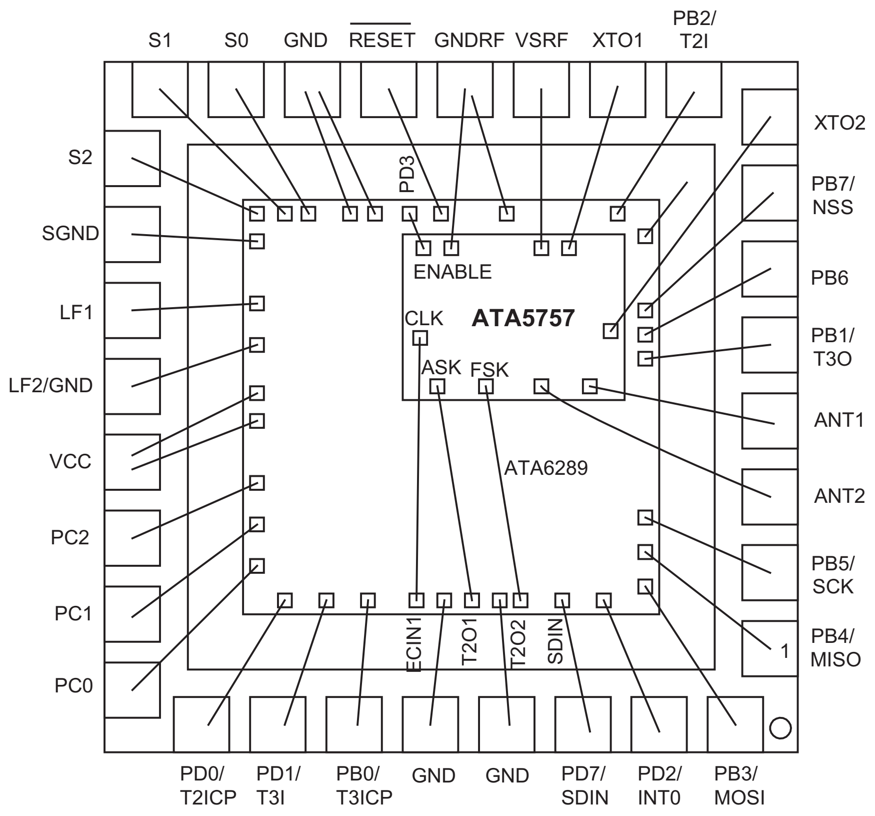
2.2 Bonding Diagram

The Atmel® ATA6286C is a smart RF microtransmitter-based multichip package (MCP). Figure 2-2 on page 4 shows the internal assembly of the MCP. This assembly has two internal dies with four inter-die connections, as shown in Table 2-1.

Table 2-1. Inter-Die Connection Description of the MCP

Atmel ATA6289 Pin	Atmel ATA5757 Pin
PD3 (INT1) – External Interrupt Input 1	EN – Enable Input
PD4 (ECIN1) – External Clock Input 1	CLK – Clock Output Signal
PD5 (T2O1) – Timer2 Modulator Output 1	ASK – Input Signal
PD6 (T2O2) – Timer2 Modulator Output 2	FSK – Input Signal

Figure 2-2. Multichip Package (MCP)



2.3 Pin Configurations

Figure 2-3. Pinout for QFN32 5mm × 5mm Package

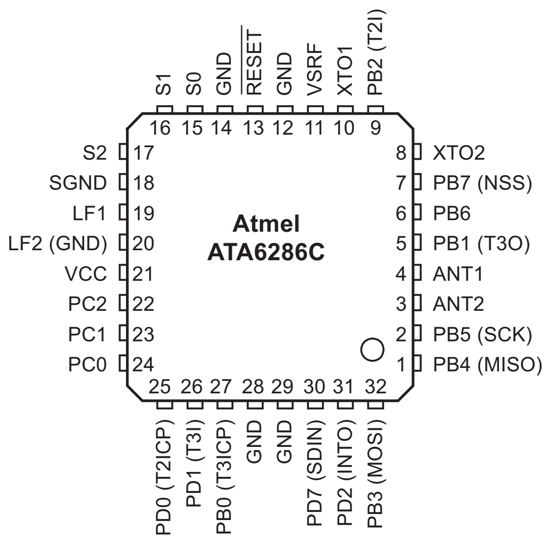


Table 2-2. Pin Description

Pin	Symbol	Alternate Function 1	Alternate Function 2	Function	Comment
1	PB4	MISO	PCINT4	SPI	Port B4
2	PB5	SCK	PCINT5	SPI	Port B5
3	ANT2	-	-	RF antenna 2, emitter of antenna output stage	RF pin
4	ANT1	-	-	RF antenna 1, open collector antenna output	RF pin
5	PB1	T3O	PCINT1	Timer3 output	Port B1
6	PB6	-	PCINT6		Port B6
7	PB7	SS	PCINT7	SPI	Port B7
8	XT02	-	-	Switch for FSK modulation	RF pin
9	PB2	T2I	PCINT2	Timer1, timer2, timer3 external input clock	Port B2
10	XT01	-	-	Connection for crystal	RF pin
11	VSRF	-	-	Power supply voltage for RF	RF pin
12	GNDRF	-	-	Power supply ground for RF	RF pin
13	RESET	debugWIRE	-	Reset input / debugWIRE interface	
14	GND	-	-	Power supply ground	
15	S0	-	-	Sensor input 0 – pressure sensor (cap.)	
16	S1	-	-	Sensor input 1 – sensor (cap.)	
17	S2	-	-	Sensor input 2 – motion sensor (cap.) wake-up	
18	SGND	-	-	Sensor ground	
19	LF1	-	-	LF receiver input 1	
20	LF2 / GND	-	-	LF receiver input 2 internally to GND	
21	V_{CC}	-	-	Power supply voltage (analog + digital)	
22	PC2	-	PCINT10	-	Port C2
23	PC1	CLKO	PCINT9	System clock output	Port C1
24	PC0	ECIN0	PCINT8	External clock input 0	Port C0
25	PD0	T2ICP	PCINT16	Timer2 external input capture	Port D0
26	PD1	T3I	PCINT17	Timer1, timer2, timer3 external input clock	Port D1
27	PB0	T3ICP	PCINT0	Timer3 external input capture	Port B0
28	GND	-	-	Power supply ground	
29	GND	-	-	Power supply ground	
Inter-die(1)	PD3	INT1	PCINT19	External interrupt 1 $\longrightarrow$ inter-die connection	Port D2
Inter-die(1)	PD4	ECIN1	PCINT20	External clock input 1 $\longrightarrow$ inter-die connection	Port D4
Inter-die(1)	PD5	T2O1	PCINT21	Timer2 modulator output 1 $\longrightarrow$ inter-die connection	Port D5
Inter-die(1)	PD6	T2O2	PCINT22	Timer2 modulator output 2 $\longrightarrow$ inter-die connection	Port D6
30	PD7	SDIN	PCINT23	SSI – serial data input	Port D7
31	PD2	INT0	PCINT18	External interrupt input 0	Port D2
32	PB3	MOSI	PCINT3	SPI	Port B3

Note: 1. Internal inter-die connection of the MCP

2.4 Pin Names

2.4.1 V_{CC}

Supply voltage

2.4.2 GND

Ground

2.4.3 Port B (PB7..0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port B output buffers have symmetrical drive characteristics with both high-sink and source-current capability. As inputs, port B pins that are pulled low externally will source current if the pull-up resistors are activated. The port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special ATA6289 features as listed in Section 3.12.3.1 “Alternate Functions of Port B” on page 51.

2.4.4 Port C (PC2..0)

Port C is a 3-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port C output buffers have symmetrical drive characteristics with both high-sink and source-current capability. As inputs, port C pins that are pulled low externally will source current if the pull-up resistors are activated. The port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port C also serves the functions of various special ATA6289 features as listed in Section 3.12.3.3 “Alternate Functions of Port C” on page 53.

2.4.5 Port D (PD7..0)

Port D is a 8(4)-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PD(6..3) pins are used as internal inter-die connections I/O ports. The port D output buffers have symmetrical drive characteristics with both high-sink and source-current capability. As inputs, port D pins that are pulled low externally will source current if the pull-up resistors are activated. The port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various ATA6289 features as listed in Section 3.12.3.4 “Alternate Functions of Port D” on page 54.

2.4.6 RESET

Reset input. A low level on this pin for longer than the minimum pulse length generates a reset, even if the clock is not running. The minimum pulse length is given in Table 3-13 on page 35. Shorter pulses do not ensure that a reset is generated.

2.4.7 LF (2..1)

Input coil pins for the LF receiver.

2.4.8 S (2..0)

Measuring input pins for external capacitance sensor elements.

2.4.9 ANT(2, 1)

RF antenna pins.

2.4.10 XTO(0, 1)

External crystal pins for the internal RF transmitter IC.

2.5 Disclaimer

Typical values contained in this datasheet are based on simulations and the characterization of other AVR microcontrollers manufactured based on the same process technology. Minimum and maximum values become available after device characterization.

3. AVR Microcontroller ATA6289

3.1 Features

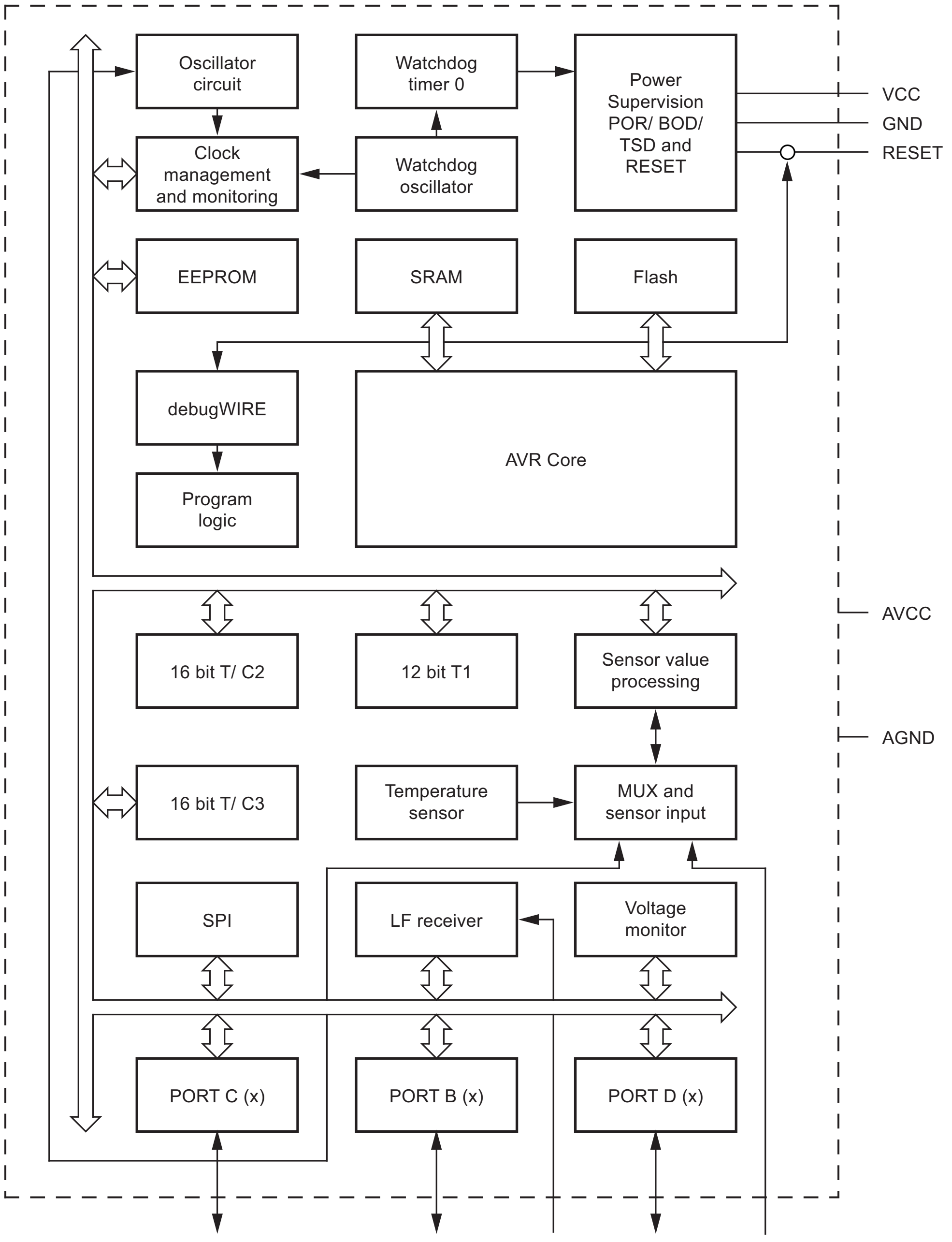
- High performance, extremely low-power Atmel AVR 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions
  - 32 × 8 general purpose working registers
  - Fully static operation
  - On-chip 2-cycle multiplier
- Non-volatile program and data memories
  - 8KB of in-system self-programmable Flash
  - Optional boot code section with independent lock bits
  - 320 (256 + 64) bytes of EEPROM
  - 512-byte internal SRAM
  - Programming lock for software security
- Peripheral features
  - Programmable watchdog/interval timer with separate internal calibrated extremely low-power oscillator
  - Two 16-bit timer/counter with compare mode, capture mode, and on-chip digital data modulator circuitry
  - Integrated (not calibrated) on-chip temperature sensor with thermal shutdown function
  - Sensor interface for external pressure sensor and motion sensor with wake-up function
  - Highly sensitive 1D LF receiver
  - Programmable voltage monitor
  - System clock management and clock monitoring
  - Master/Slave SPI serial interface
  - Integrated debug-wire-interface
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated RC oscillator
  - External and internal interrupt sources
  - Three sleep modes: idle, sensor noise reduction, and power-down
- I/O and package
  - 15 (19) programmable I/O lines
  - QFN32 package, 5mm × 5mm
- Operating voltage
  - 1.9V to 3.6V for ADC and LF receiver
  - 1.8V to 3.6V all other components
- Speed
  - 0 to 2MHz (system clock CLK)
  - 0 to 4MHz (timer clock CLT)
- Temperature range
  - –40°C to +85°C

3.2 Overview

The Atmel® ATA6289 is a CMOS 8-bit microcontroller with extremely low-power consumption based on the Atmel AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATA6289 achieves throughputs approaching 1MIPS per MHz allowing the designer to optimize power consumption versus processing speed.

3.3 Block Diagram

Figure 3-1. Block Diagram of Atmel ATA6289



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All 32 registers are directly connected to the Arithmetic Logic Unit (ALU) allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code-efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The Atmel ATA6289 provides the following features: 8KB of in-system programmable Flash with read-while-write capabilities, 320 (256+64) bytes of EEPROM, 512 bytes of SRAM, 15 (19) general purpose I/O lines, 32 general purpose working registers, on-chip debugging support and programming, three flexible timers/counters, two of them with compare modes, internal and external interrupts, a sensor interface for the external pressure sensor and an acceleration/motion sensor, a programmable watchdog timer with internally calibrated oscillator, an SPI serial port, and three software-selectable power-saving modes.

The device is manufactured using Atmel high-density non-volatile memory technology. On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional non-volatile memory programmer, or by an on-chip boot program running on the Atmel AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the boot Flash section continues to run while the application Flash section is updated, providing true read-while-write operation. By combining an 8-bit RISC CPU with in-system self-programmable Flash on a monolithic chip, the Atmel ATA6289 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The Atmel ATA6289 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

3.4 About Code Examples

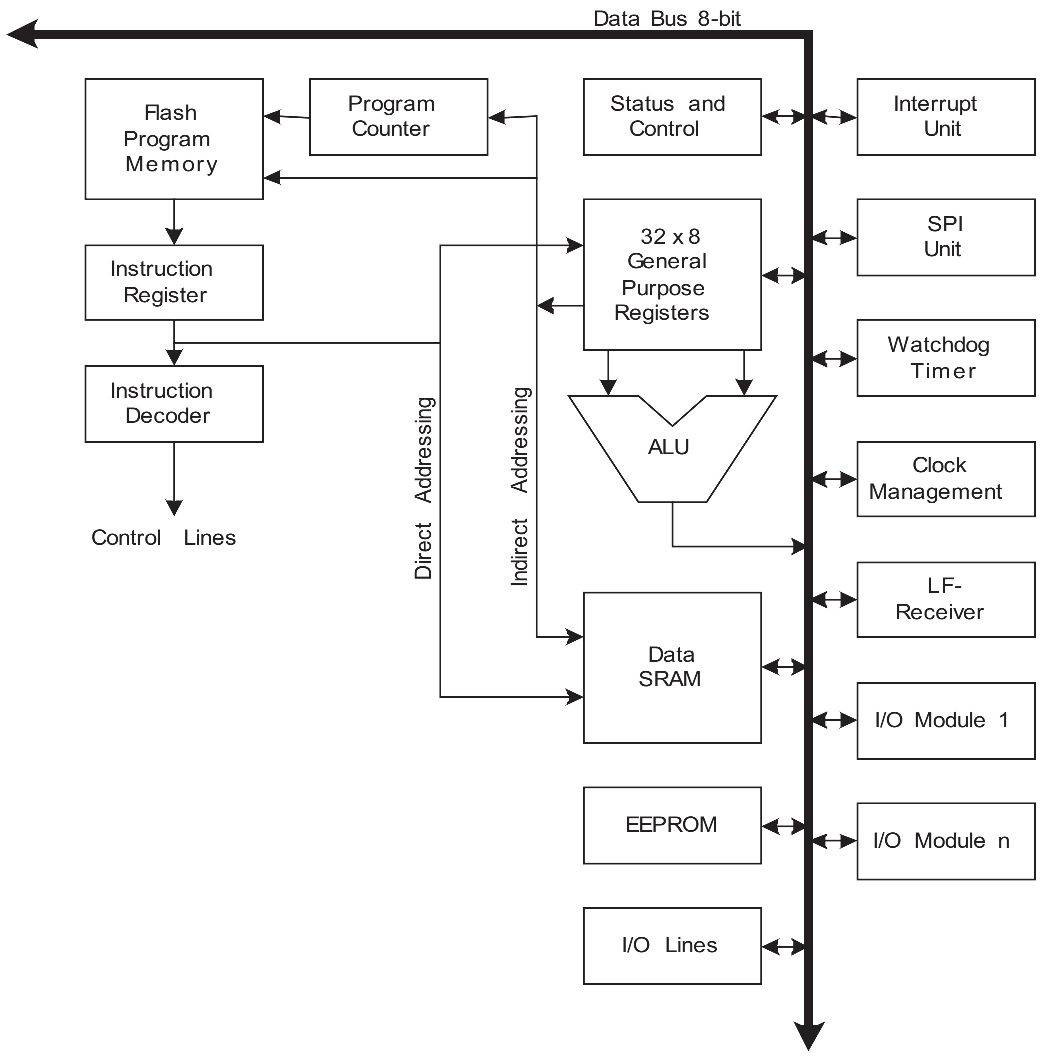
This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part-specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler-dependent. Please refer to the C compiler documentation for more details.

The Atmel AVR Studio® can be used for code development. Please select the Atmel AVR device “ATA6289.”

3.5 Atmel AVR CPU Core

3.5.1 Architectural Overview

Figure 3-2. Block Diagram of the Atmel AVR Architecture



In order to maximize performance and parallelism, the Atmel® AVR® uses Harvard architecture with separate memories and buses for program and data. Instructions in the program memory are executed with 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 program memory is in-system reprogrammable Flash memory.

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 ALU operation. In a typical ALU operation, two operands are output from the register file, the operation is executed and the result is stored back in the register file—all 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 these address pointers can also be used as an address pointer to look up tables in the Flash program memory. These added function registers are the 16-bit X, Y and Z registers 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 operation outcome.

Program flow is enabled by conditional and unconditional jump and call instructions, allowing the entire address space to be addressed directly. Most Atmel® AVR® instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided into two sections, the boot program section and the application program section. Both sections have dedicated lock bits for write and read/write protection. The Store Program Memory (SPM) instruction that writes into the application Flash memory section must reside in the boot program section.

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 Stack Pointer (SP) in the reset routine (before subroutines or interrupts are executed). The SP is read/write-accessible in the I/O space. The data SRAM can be accessed easily through the five different addressing modes supported in the Atmel AVR architecture.

The memory spaces in the Atmel AVR 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 as the data space locations following those of the register file, 0x20 - 0x5F. In addition, the ATA6289 has extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

3.5.2 ALU – Arithmetic Logic Unit

The high-performance Atmel AVR ALU operates in direct connection with all 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories—arithmetic, logic and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. For a detailed description, see Section 3.22 "Instruction Set Summary" on page 156.

3.5.3 Status Register

The status register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering the 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. In many cases—this eliminates the need to use 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.

3.5.3.1 The Atmel AVR Status Register (SREG):

Bit	7	6	5	4	3	2	1	0	
	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	

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 independently 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 the 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 by the BLD instruction into a bit in a register in the register file.

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 Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

Bit 4 – S: Sign Bit, S = N $\oplus$ V
The S bit is always exclusive or located between the negative flag N and the two's complement overflow flag V. See Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

Bit 3 – V: Two's Complement Overflow Flag
The two's complement overflow flag V supports two's complement arithmetic. See Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

Bit 2 – N: Negative Flag
The negative flag N indicates a negative result in an arithmetic or logic operation. See Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

Bit 1 – Z: Zero Flag
The zero flag Z indicates a zero result in an arithmetic or logic operation. See Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

Bit 0 – C: Carry Flag
The carry flag C indicates a carry in an arithmetic or logic operation. See Section 3.22 "Instruction Set Summary" on page 156 for detailed information.

3.5.4 General Purpose Register File

The register file is optimized for the Atmel® AVR® 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 3-3 shows how the 32 general purpose working registers in the CPU are structured.

Figure 3-3. Atmel AVR CPU General Purpose Working Registers

Bit:	7 0	Addr.	
	R0	0x00	
	R1	0x01	
	R2	0x02	
	…		
	R13	0x0D	
General	R14	0x0E	
Purpose	R15	0x0F	
Working	R16	0x10	
Registers	R17	0x11	
	…		
	R26	0x1A	X register low byte
	R27	0x1B	X register high byte
	R28	0x1C	Y register low byte
	R29	0x1D	Y register high byte
	R30	0x1E	Z register low byte
	R31	0x1F	Z register high byte

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 3-3, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not physically implemented as SRAM locations, this memory organization provides considerable flexibility in accessing the registers because the X-, Y- and Z-pointer registers can be set to index any register in the file.

3.5.4.1 The X, Y and Z Registers

The registers R26..R31 have some added functions to their general purpose usage. These registers are 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 3-4.

Figure 3-4. The X, Y and Z Registers

	15	XH	XL	0
X Register	**7**	**0**	**7**	**0**
	R27(0x1B)		R26 (0x1A)	

	15	YH	YL	0
Y Register	**7**	**0**	**7**	**0**
	R29 (0x1D)		R28 (0x1C)	

	15	ZH	ZL	0
Z Register	**7**	**0**	**7**	**0**
	R31 (0x1F)		R30 (0x1E)	

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

3.5.5 The Stack Pointer

The stack is primarily 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 0x100, preferably RAMEND. 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 a 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 the return address is popped from the stack with return from subroutine RET or return from interrupt RETI.

The Atmel® AVR® Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation-dependent. Note that the data space in some implementations of the Atmel AVR architecture is so small that only SPL is needed. In this case, the SPH register is not present.

In ATA6289 only SP9 and Sp8 of high byte (SPH) are used.

Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
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	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

3.5.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The Atmel® AVR® 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 3-5 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 1MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks and functions per power unit.

Figure 3-5. The Parallel Instruction Fetches and Instruction Executions

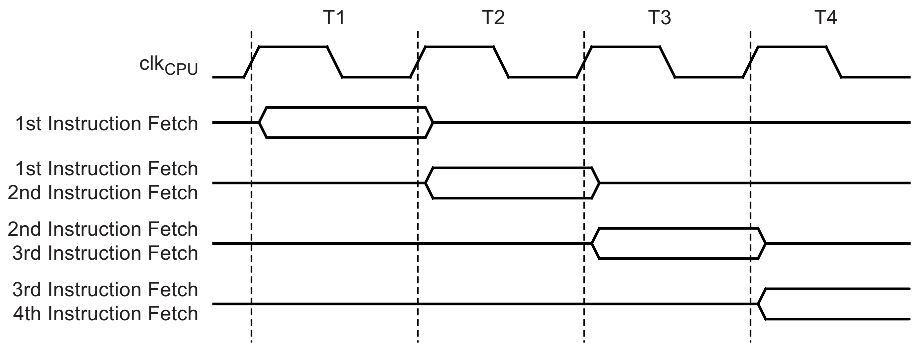
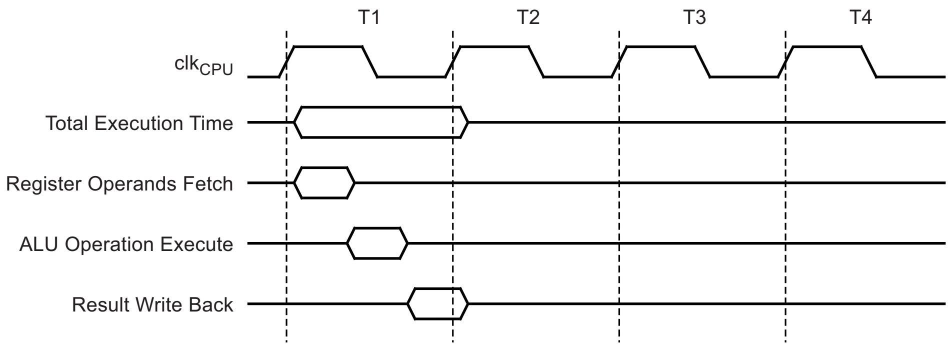


Figure 3-6 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed with the result stored to the destination register.

Figure 3-6. Single-Cycle ALU Operation



3.5.7 Reset and Interrupt Handling

The Atmel AVR 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 unique 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. Depending on the program counter value, interrupts may be automatically disabled when boot lock bits BLB02 or BLB12 are programmed. This feature improves software security. See Section 3.19.5.1 “Store Program Memory Control and Status Register – SPMCSR” on page 136 for more details.

By default the lowest addresses in the program memory space are defined as the reset and interrupt vectors. The complete list of vectors is found in Section 3.10 “Interrupts” on page 39. This list also determines the priority levels of the different interrupts. The lower the address the higher the priority level. RESET has the highest priority, followed by INT0—the external interrupt request 0. The interrupt vectors can be moved to the start of the boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). For more information, see Section 3.10 “Interrupts” on page 39. The reset vector can also be moved to the start of the boot Flash section by programming the BOOTRST fuse. See Section 3.19 “Boot Loader Support – Read-While-Write Self-Programming” on page 126 for more details.

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 is 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) is set and remembered until the global interrupt enable bit is set, and is then executed by order of priority.

The second type of interrupts triggers 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 is not triggered.

When the Atmel AVR exits from an interrupt, it always returns to the main program and executes one more instruction before any pending interrupt is served.

Note that the status register is not automatically stored when entering an interrupt routine, and not restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, interrupts are immediately disabled. No interrupt is executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
        inr16, SREG; store SREG value
        cli ; disable interrupts during timed sequence
        sbiEECR, EEMWE; start EEPROM write
        sbiEECR, EEWE
        outSREG, r16; restore SREG value (I-bit)
```

C Code Example

```
        char cSREG;
        cSREG = SREG;/* store SREG value */
        /* disable interrupts during timed sequence */
        _CLI();
        EECR |= (1<<EEMWE); /* start EEPROM write */
        EECR |= (1<<EEWE);
        SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI is 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

```
        _SEI(); /* set Global Interrupt Enable */
        _SLEEP(); /* enter sleep, waiting for interrupt */
        /* note: will enter sleep before any pending interrupt(s) */
```

3.5.7.1 Interrupt Response Time

The interrupt execution response for all the enabled Atmel® AVR® interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During these four 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 multicycle 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 four clock cycles. This increase comes in addition to the startup time from the selected sleep mode.

A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the program counter (two bytes) is popped back from the stack, the stack pointer is incremented by two, and the I bit in SREG is set.

3.6 Atmel AVR ATA6289 Memories

This section describes the different memories in the ATA6289. The Atmel® AVR® architecture has two main memory spaces, the data memory and the program memory space. In addition, the ATA6289 features an EEPROM memory for data storage. All three memory spaces are linear and regular.

3.6.1 In-System Reprogrammable Flash Program Memory

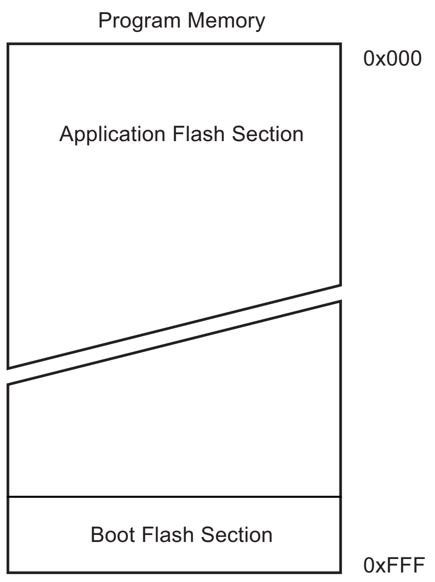
The ATA6289 contains 8KB on-chip in-system reprogrammable Flash memory for program storage. Because all Atmel AVR instructions are 16 bits or 32 bits wide, the Flash is organized as 4K × 16. For software security, the Flash program memory space is divided into two sections, the boot program section and the application program section.

The Flash memory has a service life expectancy of at least 200 write/erase cycles (according to the AECQ100 standard: cycles at room temperature followed by 1,000 hours of High Temperature Operation Lifetime (HTOL) while constantly reading the Flash memory. The ATA6289 Program Counter (PC) is 12 bits wide and thus addresses the 4K program memory locations. The operation of the boot program section and associated boot lock bits for software protection are described in detail in Section 3.19 "Boot Loader Support – Read-While-Write Self-Programming" on page 126. Section 3.21.5 "Serial Downloading" on page 149 contains a detailed description about Flash data serial downloading using the SPI pins.

Constant tables can be allocated within the entire program memory address space (see the Load Program Memory (LPM) instruction description (Table 3-83 on page 151).)

Timing diagrams for instruction fetch and execution are presented in Section 3.5.6 "Instruction Execution Timing" on page 14.

Figure 3-7. Program Memory Map

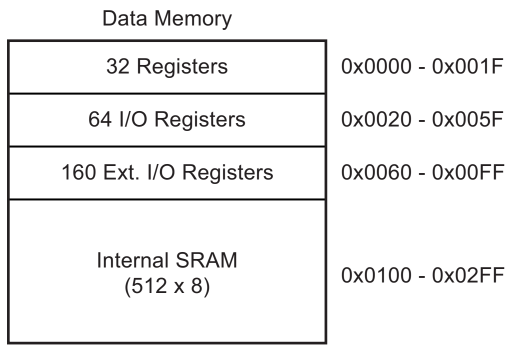


3.6.2 SRAM Data Memory

Figure 3-8 shows how the ATA6289 SRAM memory is organized. The ATA6289 is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the opcode for 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. The lower 768 data memory locations address not only the register file, the I/O memory and extended I/O memory, but also the internal data SRAM. The first 32 locations address the register file, the next 64 location the standard I/O memory. This is followed by 160 locations of extended I/O memory, with the next 512 locations addressing 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 512 bytes of internal SRAM in the ATA6289 are all accessible through all these addressing modes. The register file is described in Section 3.5.4 "General Purpose Register File" on page 12.

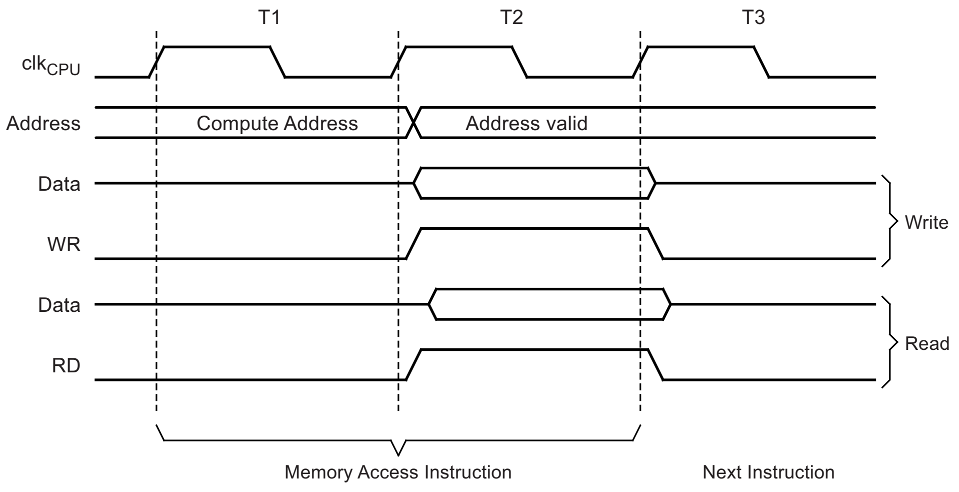
Figure 3-8. Data Memory Map



3.6.2.1 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 3-9 on page 17.

Figure 3-9. On-Chip Data SRAM Access Cycles



3.6.3 I/O Memory

The I/O space definition of the ATA6289 is shown in Section 3.21.6 "Register Summary" on page 153. All ATA6289 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 using the SBIS and SBIC instructions. Refer to Section 3.22 "Instruction Set Summary" on page 156 for more details. When using the I/O specific commands IN and OUT, the 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 ATA6289 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in opcode for 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 logic one to them. Note that, unlike most other Atmel® AVR®s, the CBI and SBI instructions 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.

3.6.4 General Purpose I/O Registers

The Atmel ATA6289 contains 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.

3.6.4.1 General Purpose I/O Register 2 – GPIOR2

Bit	7	6	5	4	3	2	1	0	
	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	

3.6.4.2 General Purpose I/O Register 1 – GPIOR1

Bit	7	6	5	4	3	2	1	0	
	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	

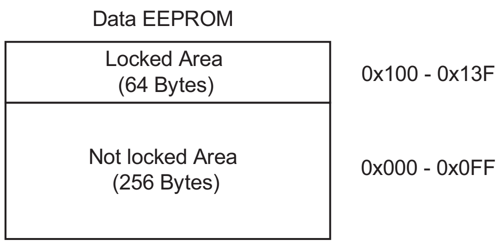
3.6.4.3 General Purpose I/O Register 0 – GPIOR0

Bit	7	6	5	4	3	2	1	0	
	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	

3.6.5 EEPROM Data Memory

The Atmel® ATA6289 contains 320 (256 + 64) bytes of data EEPROM memory. It is organized as a separate data space in which single bytes can be read and written. As shown in Figure 3-10, the EEPROM contains a locked area of 64 bytes. The EEPROM addresses above 0x0FF are only readable if the fuse bit EELOCK is programmed. The EELOCK bit is located in the fuse high byte. For more information, see Table 3-71 on page 138. If the EELOCK bit is unprogrammed (default), the EEPROM area above 0x0FF is also writable. The EEPROM has a service life expectancy of at least 2,000 write/erase cycles (according to the AECQ100 standard: cycles at room temperature followed by 1,000 hours of High Temperature Operation Lifetime (HTOL) while constantly reading the Flash memory). The access between the EEPROM and the CPU is described in the following, with information about the EEPROM address registers, the EEPROM data register and the EEPROM control register.

Figure 3-10. EEPROM Map



3.6.5.1 EEPROM Read/Write Access

The EEPROM address registers are accessible in the I/O space. The write access time for the EEPROM is given in Table 3-1 on page 20. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on power-up/down. For a period of time this causes the device to run at a voltage lower than specified as minimum for the clock frequency used. See Section 3.6.5.5 "Preventing EEPROM Corruption" on page 23 for more information on how to avoid problems when encountering these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. For more information, see Section 3.6.5.4 "The EEPROM Control Register – EECR" on page 20.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

3.6.5.2 The EEPROM Address Register – EEARH, EEARL

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	-	**EEAR8**	**EEARH**
	EEAR[7..0]								**EEARL**
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	X	X	X	X	X	X	X	X	

Bits 15..9 – Res: Reserved Bits
These bits are reserved bits on the ATA6289 and are always read as zero.

Bits 8..0 – EEAR7..0: EEPROM Address
The EEPROM address register (EEAR) specifies the EEPROM address in the 320(256)-byte EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 319(255). The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

3.6.5.3 The EEPROM Data Register – EEDR

Bit	7	6	5	4	3	2	1	0	
	EEDR[7..0]								EEDR
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	

Bits 7..0 – EEDR7.0: EEPROM Data
For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

3.6.5.4 The EEPROM Control Register – EECR

Bit	7	6	5	4	3	2	1	0	
	-	-	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	X	X	0	0	X	0	

Bits 7..6 – Res: Reserved Bits
These bits are reserved bits on the ATA6289 and are always read as zero.

Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits
The EEPROM programming mode bits setting defines which programming action is triggered when writing EEWE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the erase and write operations into two different operations. The programming times for the different modes are shown in Table 3-1. While EEWE is set, any write command to EEPMn is ignored. During reset, the EEPMn bits are reset to 0b00 unless the EEPROM is busy programming.

Table 3-1. EEPROM Mode Bits

EEPM1	EEPM0	Programming Time	Operation
0	0	3.4ms	Erase and write in one operation (atomic operation)
0	1	1.8ms	Erase only
1	0	1.8ms	Write only
1	1	-	Reserved for future use

Bit 3 – EERIE: EEPROM Ready Interrupt Enable
Writing EERIE to one enables the EEPROM ready interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM ready interrupt generates a constant interrupt when EEWE is cleared.

Bit 2 – EEMWE: EEPROM Master Write Enable
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set, setting EEWE within four clock cycles writes data to the EEPROM at the address selected. If EEMWE is zero, setting EEWE has no effect. When EEMWE has been written to one by the software, the hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for an EEPROM write procedure.

Bit 1 – EEWE: EEPROM Write Enable
The EEPROM Write Enable (EEWE) signal is the write strobe to the EEPROM. When the address and data are correctly set up, the EEWE bit must be written to one to write the value into the EEPROM. The EEMWE bit must be written to one before a logic one is written to EEWE; otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (steps 3 and 4 can be done in any order):

- Wait until EEWE becomes zero.
- Wait until SELFPRGEN in the SPMCSR register becomes zero.
- Write the new EEPROM address to the EEAR register (optional).
- Write the new EEPROM data to the EEDR register (optional).
- Write a logic one to the EEMWE bit while writing a zero to the EEWE bit in the EECR register.
- Within four clock cycles after setting EEMWE, write a logic one to EEWE.

The EEPROM cannot be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a boot loader allowing the CPU to program the Flash. If the Flash is never updated by the CPU, step 2 can be omitted. See Section 3.19 "Boot Loader Support – Read-While-Write Self-Programming" on page 126 for more information about boot programming.

Caution: An interrupt between step 5 and step 6 makes the write cycle fail because the EEPROM master write enable times out. If an interrupt routine accessing the EEPROM interrupts another EEPROM access, the EEAR or EEDR register is modified, causing the interrupted EEPROM access to fail. It is recommended to have the global interrupt flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEWE bit is cleared by the hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 – EERE: EEPROM Read Enable
The EEPROM Read Enable (EERE) signal is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be written to a logic one to trigger the EEPROM read. The EEPROM read access takes one instruction with the requested data becoming available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed. The user should poll the EEWE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM nor change the EEAR register.

The calibrated oscillator is used for timing the EEPROM accesses. Table 3-2 lists the typical programming time for EEPROM access from the CPU.

Table 3-2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ. Programming Time
EEPROM write (from CPU)	67 584	8.5ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts occur during execution of these functions. The examples also assume that no Flash boot loader is present in the software. If there is such code present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example

```
EEPROM_write:
; Wait for completion of previous write
sbic EECR,EEWE
rjmp EEPROM_write
; Set up address (r18:r17) in address register
out EEARH, r18
out EEARL, r17
; Write data (r16) to Data Register
out EEDR,r16
; Write logical one to EEMWE
sbi EECR,EEMWE
; Start eeprom write by setting EEWE
sbi EECR,EEWE
ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;
/* Set up address and Data Registers */
EEAR = uiAddress;
EEDR = ucData;
/* Write logical one to EEMWE */
EECR |= (1<<EEMWE);
/* Start eeprom write by setting EEWE */
EECR |= (1<<EEWE);
}
```

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts occur while these functions are executed.

Assembly Code Example

```
EEPROM_read:
; Wait for completion of previous write
sbic EECR,EEWE
rjmp EEPROM_read
; Set up address (r18:r17) in address register
out EEARH, r18
out EEARL, r17
; Start eeprom read by writing EERE
sbi EECR,EERE
; Read data from Data Register
in r16,EEDR
ret
```

C Code Example

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;
/* Set up address register */
EEAR = uiAddress;
/* Start eeprom read by writing EERE */
EECR |= (1<<EERE);
/* Return data from Data Register */
return EEDR;
}
```

3.6.5.5 Preventing EEPROM Corruption

During periods of low V_{CC}, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board-level systems using EEPROM, and the same design solutions should be applied.

EEPROM data corruption can be caused by two situations if there is insufficient voltage. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the Atmel® AVR® $\overline{\text{RESET}}$ active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-Out Detector (BOD). If the detection level of the internal BOD does not match the detection level needed, an external low V_{CC} reset protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation is completed, provided the power supply voltage is sufficient.

3.7 Clock Generation

3.7.1 Clock Module

The ATA6289 contains a clock module with two internal oscillator types:

FRC: Fast running, programmable and calibrated RC oscillator (1MHz/4MHz ±10%)

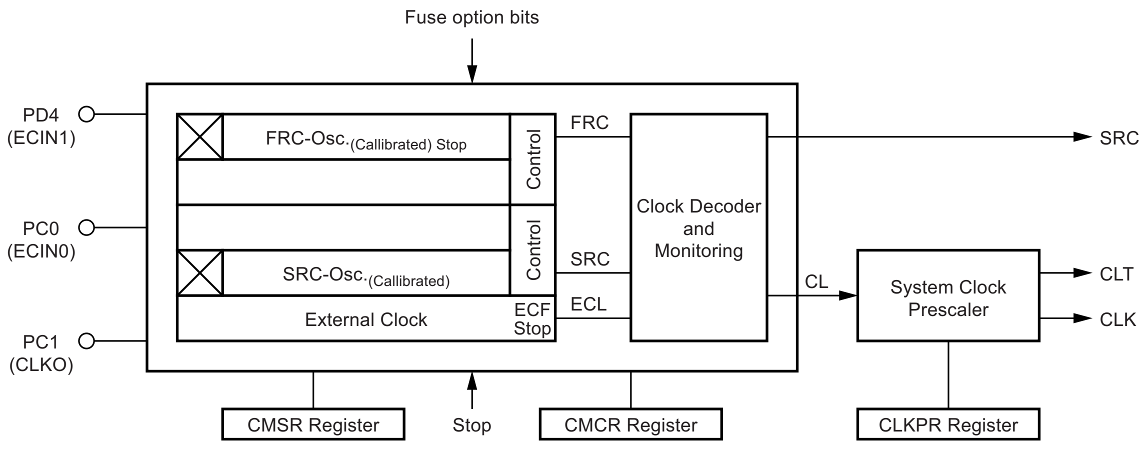
SRC: Slow running and calibrated RC oscillator (90kHz ±10%)

The PC1/ECIN0 pin and PD4/ECIN1 pin can be used as input for two different external clocks and the PC1/CLKO pin as output for the divided system clock. All of these oscillator types and external input clocks can be selected to generate the system Clock (CLK). A special feature of the clock management is the capability to switch between these different clock sources at run time. This new feature offers the advantage that the controller can now start operation after the wake-up signal with the calibrated internal RC oscillator and can switch to an external clock as its system clock. A synchronization stage avoids clock periods of insufficient length if the clock source or the clock speed is changed. If an external input clock is selected, a supervisor circuit monitors the external input and automatically switches to an internal RC oscillator clock if the external clock source fails. The ECF bit indicates the condition of the external input clock monitoring circuit in the CMSR register. If the accessory interrupt enable is set, the corresponding monitoring interrupt is executed.

In applications that do not require exact timing, it is possible to use the fully integrated RC oscillators. The center frequency tolerance of both RC oscillators can be calibrated by VCC = 3V/25°C within ±1% accuracy. The SRC and the timer0 can work together as an ultra-low-power watchdog/interval timer stage.

The clock module is programmable via software with the Clock Management Control Register (CMCR) and the Clock Prescaler Register (CLKPR). The required oscillator configuration can be selected with the CMM[1..0] bits in the CMCR. A system clock prescaler contains a programmable 7-bit divider stage. This stage subdivides the system clock by setting the CLKPR and allows the adjustment of the system Clock speed (CLK) and also the Timer Clock speed (CLT). This can be used with all clock source options and affects the clock frequency of the CPU, with all synchronous peripherals. $CLK_{I/O}$, CLK_{CPU} and CLK_{Flash} divided by a factor shown in Table 3-10 on page 31.

Figure 3-11. Clock Module Unit



3.7.1.1 External Clock Monitor

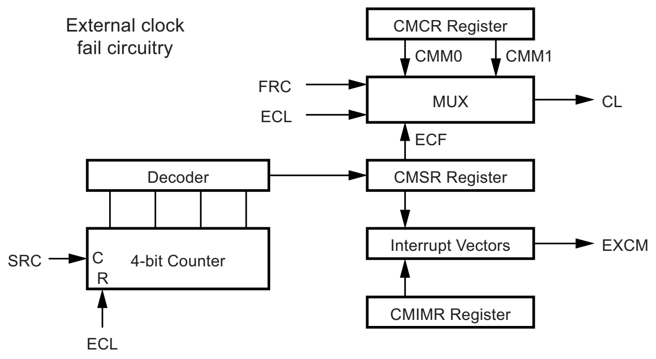
If an external clock is used as the system clock, internal clock monitor circuitry is activated. If the external clock fails for a given time, the ECF bit is set in the Clock Management Status Register (CMSR). After an external clock fail is detected, the system uses the internal RC Oscillator (FRC) as the system clock by switching the CCS bit in the CMCR to zero.

The external clock monitor circuit uses the internal SRC oscillator (90kHz) as the clock source for a 4-bit counter. If the external clock does not reset the internal 4-bit counter periodically, a counter value is reached which triggers the external clock fail bit ECF. For more information, see Figure 3-12.

A typical time value for the external clock fail detection is 100µs. Therefore the minimum external clock frequency is limited to typically 10kHz.

An external frequency < 10kHz forces a clock fail reset.

Figure 3-12. External Clock Fail Circuitry



3.7.1.2 Clock Management Control Register – CMCR

Bit	7	6	5	4	3	2	1	0	
	CMCCE	**-**	**ECINS**	**CCS**	**CMONEN**	**SRCD**	**CMM1**	**CMM0**	**CMCR**
Read/Write	RW	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – CMCCE: Clock Management Control Change Enable Bit
The CMCCE bit must be written to logic one to enable change of the CMMn bits, SRCD bit, CMONEN, CCS bit, and ECINS bit. The CMCCE bit is only updated when the other bits in CMCR are simultaneously written to zero. CMCCE is cleared by the hardware four cycles after it is written or when CMMn, CCS and ECINS bits are written. Rewriting the CMCCE bit within this time-out period neither extends the time-out period nor clears the CMCCE bit.

Bit 6 – Res: Reserved Bit
This bit is a reserved bit on the ATA6289 and is always read as zero.

Bit 5 – ECINS: External Clock Input Select Bit
This bit selects one of the two external clock input PC0(ECIN0) or PD4(ECIN1). The ECINS bit must be written to logic one to enable the ECIN1 clock input, and if the ECINS bit is written to logic zero, then ECIN0 clock input is enabled. The ECINS bit should be only changed if the CCS bit has been cleared. If the CCS bit is set to zero, the internal FRC is activated during bit change for synchronization reasons.

Table 3-3. External Clock Input Select Bit Description

ECINS	Description
0	PC0 —–> External input clock 0 (ECIN0)
1	PD4 —–> External input clock 1 (ECIN1)

Bit 4 – CCS: Core Clock Select Bit
This bit selects from the FRC oscillator clock and all other clock sources. The CCS bit must be written to logic one to enable the mode selected with the CMM[1..0] bits. If the CCS bit is written to logic zero, the FRC oscillator clock is enabled. When the CCS bit is logic one, the CMM[1..0] bits in CMCR must not be changed. After external clock fail detection, the CCS bit is written to zero.

Table 3-4. Core Clock Select Bit Description

CCS	Description
0	The FRC oscillator generates CL
1	The SRC oscillator or an External Clock source (ECL) generates depending on the setting of the CMM[1..0] bits.

Bit 3 – CMONEN: Clock Monitoring Enable
This bit controls clock monitoring. The CMONEN bit must be written to logic one to enable clock monitoring. If the CMONEN bit is written to logic zero, clock monitoring is always disabled.

Bit 2 – SRCD: Slow RC oscillator (SRC) Disable Bit
This bit controls the SRC oscillator used as the clock source for the watchdog (also called WDRC). The SRCD bit must be written to logic one to disable (stop) the SRC, and if the SRCD bit is written to logic zero, the SRC is always enabled (running). The SRC oscillator cannot be disabled if the fuse bit WDRCON is programmed.

Bits 1..0 – CMM1..0: Clock Management Mode Bits 1 - 0
These bits select the input clock source (CL) of the system clock prescaler. Bits CMM1 and CMM0 are not affected by an external clock fail. Before changing the CMM1..0 bits, CCS must first be cleared. After the CMM1..0 bits have been changed, CCS can be set to logic one.

Table 3-5. Clock Source of the System Clock Prescaler Select Bit Description

Mode	Clock Source for System Clock Prescaler (CL)			
	CMM1	CMM0	CCS = 0	CCS = 1
0	0	0	FRC	Reserved
1	0	1	FRC	Reserved
2	1	0	FRC	SRC
3	1	1	FRC	ECL

3.7.1.3 Clock Management Status Register – CMSR

Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	ECF	CMSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 7 to 1 – Res: Reserved Bits
These bits are reserved bits on the ATA6289 and are always read as zero.

Bit 0 – ECF: External Clock Input Flag Bit
This bit is set if the clock monitoring circuit detects a breakdown of the selected external input clock (ECIN0 or ECIN1). ECF is automatically cleared when the clock monitoring interrupt vector is executed. Alternatively, ECF can be cleared by writing a logic one to its bit location.