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

8-bit AVR Microcontroller

ATmega8A

DATASHEET COMPLETE

Introduction

The Atmel® ATmega8A is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8A achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Features

- High-performance, Low-power Atmel AVR 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions - Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 8KBytes of In-System Self-programmable Flash program memory
  - 512Bytes EEPROM
  - 1KByte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - Atmel QTouch and QMatrix acquisition
  - Up to 64 sense channels
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode

  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    - Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    - Six Channels 10-bit Accuracy
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V
- Speed Grades
  - 0 - 16MHz
- Power Consumption at 4MHz, 3V, 25°C
  - Active: 3.6mA
  - Idle Mode: 1.0mA
  - Power-down Mode: 0.5µA

Table of Contents

1. Description

The Atmel AVR core combines a rich instruction set with 32 general purpose working registers. All the 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 ATmega8A provides the following features: 8K bytes of In-System Programmable Flash with Read-While- Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, one byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, one SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

Atmel offers the QTouch library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Composer allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the 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 will continue 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 ATmega8A is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The device 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 kit.

2. Configuration Summary

Features	ATmega8A
Pin count	32
Flash (KB)	8
SRAM (KB)	1
EEPROM (Bytes)	512
General Purpose I/O pins	23
SPI	1
TWI (I^2C)	1
USART	1
ADC	10-bit 15ksps
ADC channels	6 (8 in TQFP and QFN/MLF packages)
AC propagation delay	Typ 400ns
8-bit Timer/Counters	2
16-bit Timer/Counters	1
PWM channels	3
RC Oscillator	+/-3%
Operating voltage	2.7 - 5.5V
Max operating frequency	16MHz
Temperature range	-40°C to +105°C

3. Ordering Information

Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
16	2.7 - 5.5V	ATmega8A-AU ATmega8A-AUR[3]	32A 32A	Industrial (-40°C to 85°C)
		ATmega8A-PU	28P3	
		ATmega8A-MU	32M1-A	
		ATmega8A-MUR[3]	32M1-A	
		ATmega8A-AN ATmega8A-ANR[3]	32A 32A	Extended (-40°C to 105°C)
		ATmega8A-MN	32M1-A	
		ATmega8A-MNR[3]	32M1-A	
		ATmega8A-PN	28P3	

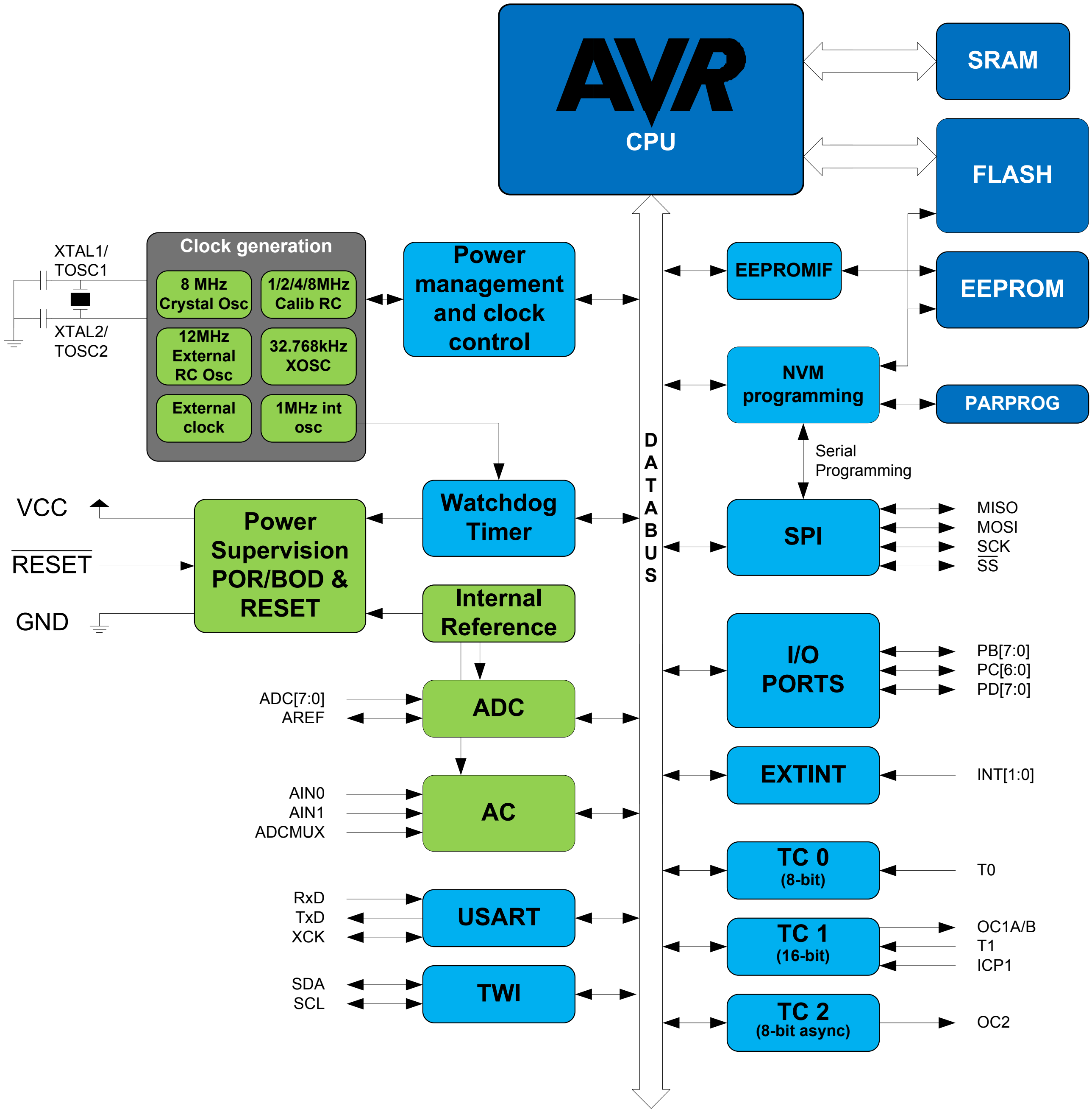
Note:

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Tape and Reel

Package Type	
32A	32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0mm body, lead pitch 0.50mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

4. Block Diagram

Figure 4-1 Block Diagram



5. Pin Configurations

Figure 5-1 PDIP

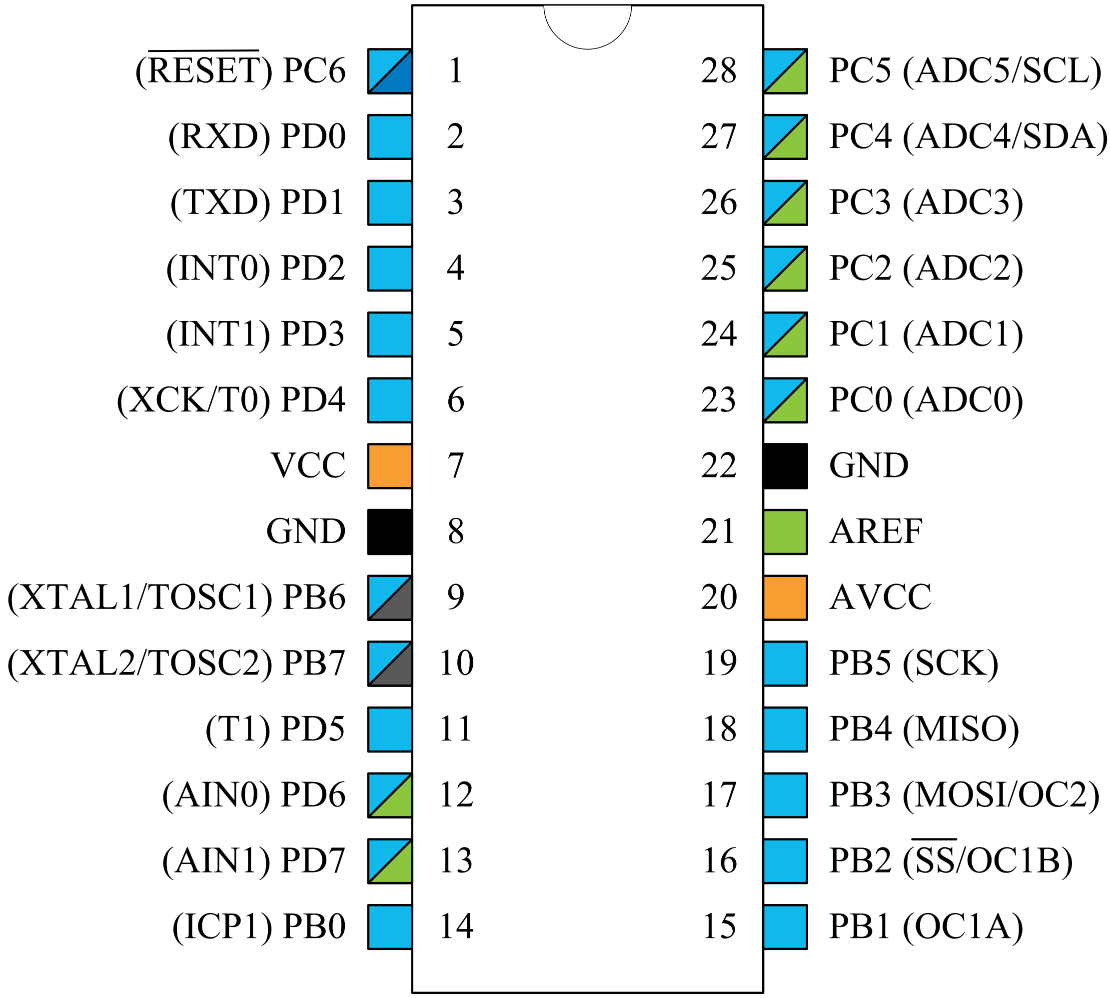


Figure 5-2 TQFP Top View

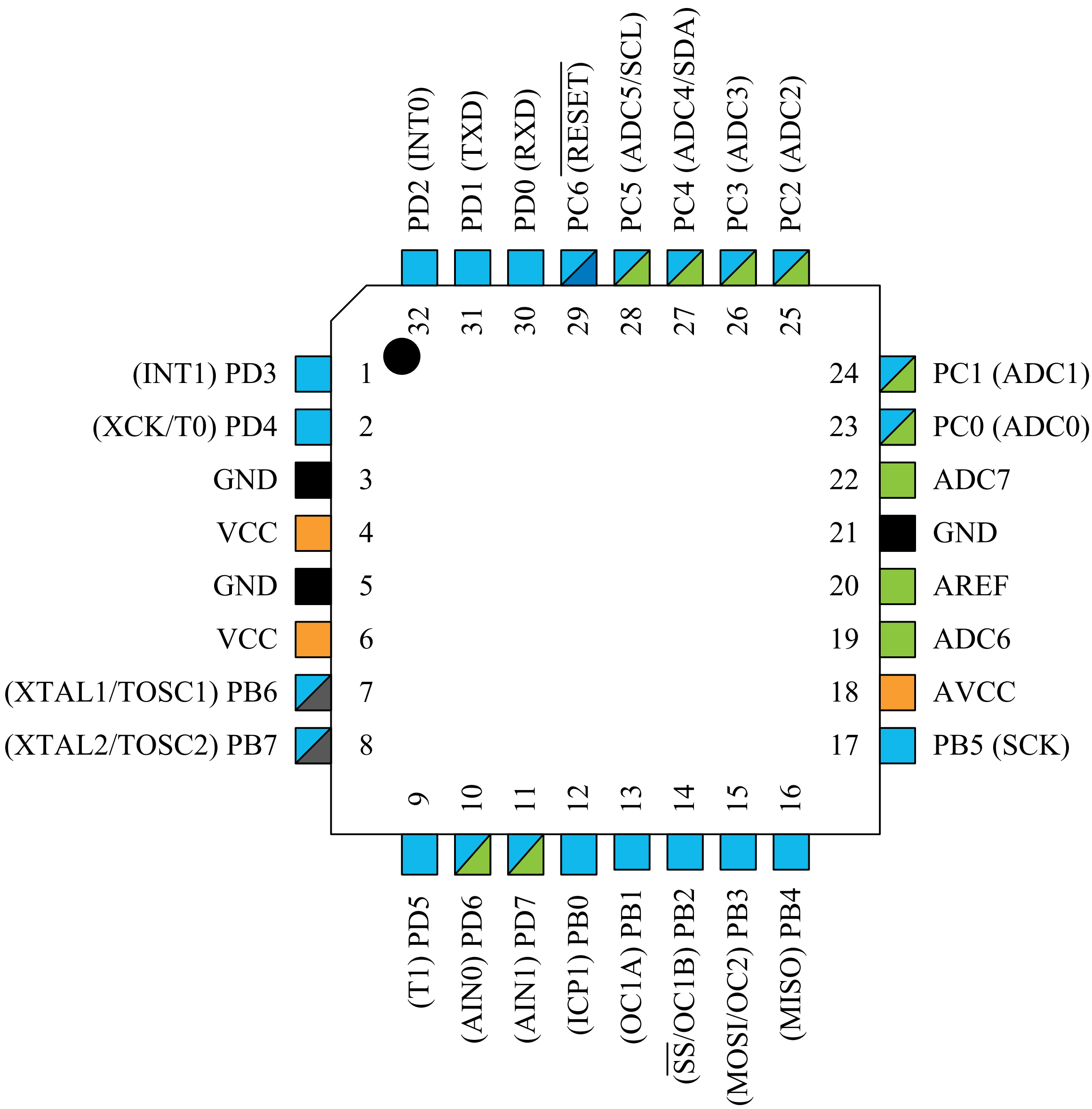
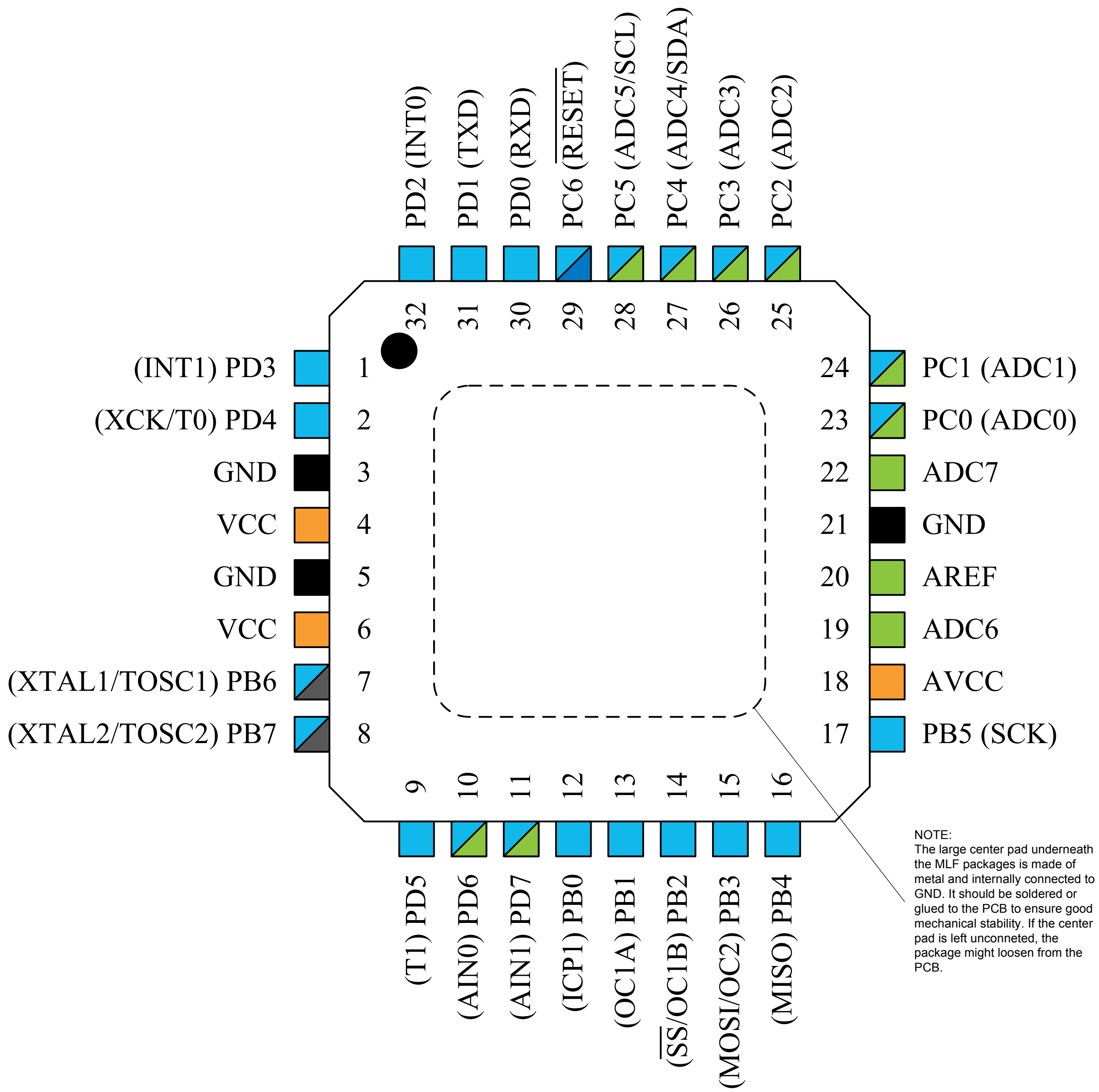


Figure 5-3 MLF Top View



5.1. Pin Descriptions

5.1.1. V_{CC}

Digital supply voltage.

5.1.2. GND

Ground.

5.1.3. Port B (PB7:PB0) – XTAL1/XTAL2/TOSC1/TOSC2

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 capability. As inputs, Port B pins that are externally pulled low 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.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7:6 is used as TOSC2:1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in Alternate Functions of Port B and System Clock and Clock Options.

5.1.4. Port C (PC5:PC0)

Port C is an 7-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 capability. As inputs, Port C pins that are externally pulled low 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.

5.1.5. PC6/$\overline{\text{RESET}}$

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 30-5. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in Alternate Functions of Port C.

5.1.6. Port D (PD7:PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low 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 special features of the ATmega8A as listed in Alternate Functions of Port D.

5.1.7. RESET

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 30-5. Shorter pulses are not guaranteed to generate a reset.

5.1.8. AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3:0), and ADC (7:6). It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that Port C (5:4) use digital supply voltage, V_{CC}.

5.1.9. AREF

AREF is the analog reference pin for the A/D Converter.

5.1.10. ADC7:6 (TQFP and QFN/MLF Package Only)

In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

5.2. Accessing 16-bit Registers

The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. A 16-bit register must be byte accessed using two read or write operations. The 16-bit timer has a single 8-bit register for temporary storing of the High byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within the 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/B 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/B and ICR1 Registers. Note that when using “C”, the compiler handles the 16-bit access.

Assembly Code Example(1)

```
   :.
; Set TCNT1 to 0x01FF
ldi    r17,0x01
ldi    r16,0xFF
out    TCNT1H,r17
out    TCNT1L,r16
; Read TCNT1 into r17:r16
in     r16,TCNT1L
in     r17,TCNT1H
   :.
```

C Code Example(1)

```
unsigned int i;
   :.
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
   :.
```

Note: 1. See *About Code Examples*.

The assembly code example returns the TCNT1 value in the r17:r16 Register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit Timer Registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.

The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Asesmbly Code Example[(1)]

```
TIM16_ReadTCNT1:
   ; Save global interrupt flag
   in    r18,SREG
   ; Disable interrupts
   cli
   ; Read TCNT1 into r17:r16
   in    r16,TCNT1L
   in    r17,TCNT1H
   ; Restore global interrupt flag
   out   SREG,r18
   ret
```

C Code Example[(1)]

```
unsigned int TIM16_ReadTCNT1( void )
{
   unsigned char sreg;
   unsigned int i;
   /* Save global interrupt flag */
   sreg = SREG;
   /* Disable interrupts */
   _CLI();
   /* Read TCNT1 into i */
   i = TCNT1;
   /* Restore global interrupt flag */
   SREG = sreg;
   return i;
}
```

Note: 1. See *About Code Examples*.

The assembly code example returns the TCNT1 value in the r17:r16 Register pair.

The following code examples show how to do an atomic write of the TCNT1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example[(1)]

```
TIM16_WriteTCNT1:
   ; Save global interrupt flag
   in     r18,SREG
   ; Disable interrupts
   cli
   ; Set TCNT1 to r17:r16
   out    TCNT1H,r17
   out    TCNT1L,r16
   ; Restore global interrupt flag
   out    SREG,r18
   ret
```

C Code Example[(1)]

```
void TIM16_WriteTCNT1( unsigned int i )
{
   unsigned char sreg;
   unsigned int i;
   /* Save global interrupt flag */
```

```
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Set TCNT1 to i */
    TCNT1 = i;
    /* Restore global interrupt flag */
    SREG = sreg;
}
```

Note: 1. See *About Code Examples*.

The assembly code example requires that the r17:r16 Register pair contains the value to be written to TCNT1.

Related Links

About Code Examples on page 23

6. I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions. This table describes the peripheral signals multiplexed to the PORT I/O pins.

Table 6-1 PORT Function Multiplexing

PAD	Pin #	EXTINT	PCINT	AC	Custom	OSC	TC1(16-bit)	TC2(8-bit)	USART	SPI	Misc
PD[4]	14		PCINT20	ACO	-	-	O1CA	-		-	
PB[6]	1		PCINT06	-	-	EXTCLK	-	-	-	-	
PD[5]	2		PCINT21	AINP1	-	-	CLK1		-	-	SII
PD[6]	3		PCINT22	AINP0	-	-	ICP1	-	-	-	SDO
PD[7]	4		PCINT23	AINN0	-	-	-	TC2-OCB	-	-	SDI
PB[2]	5		PCINT02	-	CLO0	CLKOUT	TC1-OCB	-	-	SS	
PB[3]	6		PCINT03	-	-	-		TC2-OCA	TXD	MOSI	
PB[4]	7		PCINT04	-	-	-	-	-	RXD	MISO	
PB[5]	8		PCINT05	-	CLO1	-	-	-	XCK	SCK	
PC[4]	9		PCINT12	AINN1	-	-	-	-	-	-	
PC[5]	10	INT0	PCINT13	AINN2	-	-	-	-	-	-	
PC[6]/ RESET	13		PCINT14	-		-	-	-	-	-	HVRST/dW
VCC	11										
GND	12										

7. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

8. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

9. About Code Examples

This datasheet 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 confirm with the C compiler documentation for more details.

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”.

10. Capacitive Touch Sensing

The Atmel QTouch Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR microcontrollers. The QTouch Library includes support for the QTouch and QMatrix® acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

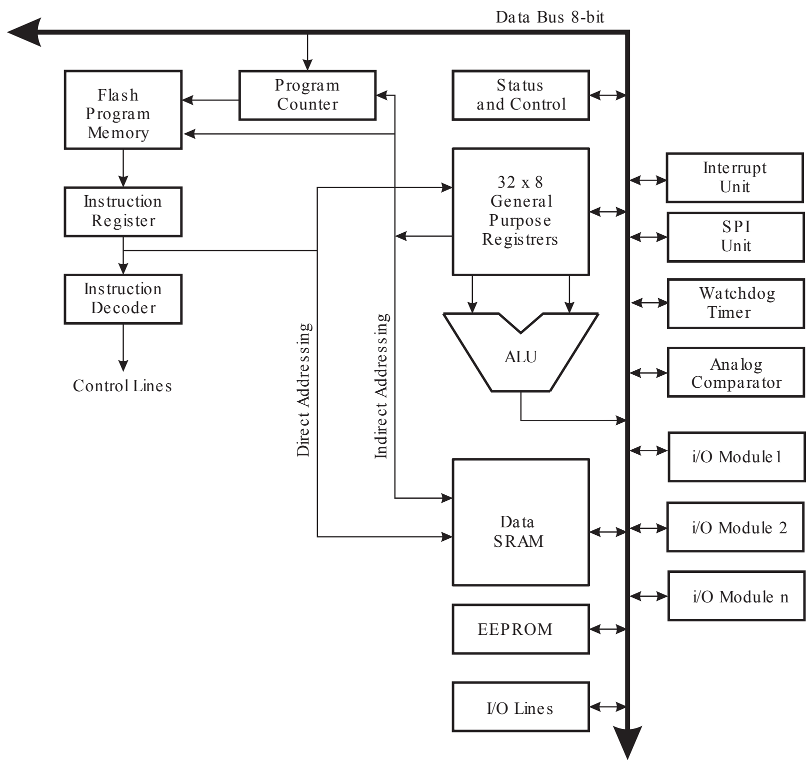
The QTouch Library is FREE and downloadable from the Atmel website at the following location: www.atmel.com/qtouchlibrary. For implementation details and other information, refer to the Atmel QTouch Library User Guide - also available for download from the Atmel website.

11. AVR CPU Core

11.1. Overview

This section discusses the Atmel AVR 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.

Figure 11-1 Block Diagram of the AVR MCU Architecture



In order to maximize performance and parallelism, the AVR 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 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 Arithmetic Logic Unit (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 – 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 Flash 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.