

**MICROCHIP****ATmega48A/PA/88A/PA/168A/PA/328/P**

megaAVR[®] Data Sheet

Introduction

The ATmega48A/PA/88A/PA/168A/PA/328/P is a low power, CMOS 8-bit microcontrollers based on the AVR[®] enhanced RISC architecture. By executing instructions in a single clock cycle, the devices achieve CPU throughput approaching one million instructions per second (MIPS) per megahertz, allowing the system designer to optimize power consumption versus processing speed.

Features

- High Performance, Low Power AVR[®] 8-Bit Microcontroller Family
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 20 MIPS Throughput at 20MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
 - 256/512/512/1KBytes EEPROM
 - 512/1K/1K/2KBytes 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
- QTouch[®] library support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix[™] acquisition
 - Up to 64 sense channels
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

- Real Time Counter with Separate Oscillator
- Six PWM Channels
- 8-channel 10-bit ADC in TQFP and QFN/MLF package
 - Temperature Measurement
- 6-channel 10-bit ADC in PDIP Package
 - Temperature Measurement
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 23 Programmable I/O Lines
 - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
 - 1.8 - 5.5V
- Temperature Range:
 - -40°C to 85°C
- Speed Grade:
 - 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
 - Active Mode: 0.2mA
 - Power-down Mode: 0.1µA
 - Power-save Mode: 0.75µA (Including 32kHz RTC)

Table of Contents

1	<i>Pin Configurations</i>	12
1.1	Pin Descriptions.....	13
2	<i>Overview</i>	15
2.1	Block Diagram	15
2.2	Comparison Between Processors	16
3	<i>Resources</i>	17
4	<i>Data Retention</i>	17
5	<i>About Code Examples</i>	17
6	<i>Capacitive Touch Sensing</i>	17
7	<i>AVR CPU Core</i>	18
7.1	Overview.....	18
7.2	ALU – Arithmetic Logic Unit.....	19
7.3	Status Register	19
7.4	General Purpose Register File	20
7.5	Stack Pointer	22
7.6	Instruction Execution Timing	23
7.7	Reset and Interrupt Handling.....	23
8	<i>AVR Memories</i>	26
8.1	Overview.....	26
8.2	In-System Reprogrammable Flash Program Memory	26
8.3	SRAM Data Memory.....	28
8.4	EEPROM Data Memory	29
8.5	I/O Memory	30
8.6	Register Description	31
9	<i>System Clock and Clock Options</i>	36
9.1	Clock Systems and their Distribution	36
9.2	Clock Sources	37
9.3	Low Power Crystal Oscillator.....	38
9.4	Full Swing Crystal Oscillator	39
9.5	Low Frequency Crystal Oscillator.....	42
9.6	Calibrated Internal RC Oscillator	43
9.7	128kHz Internal Oscillator	43

9.8	External Clock	44
9.9	Clock Output Buffer	45
9.10	Timer/Counter Oscillator.....	45
9.11	System Clock Prescaler	45
9.12	Register Description	46
10	Power Management and Sleep Modes	48
10.1	Sleep Modes.....	48
10.2	BOD Disable ⁽¹⁾	49
10.3	Idle Mode.....	49
10.4	ADC Noise Reduction Mode.....	49
10.5	Power-down Mode.....	49
10.6	Power-save Mode.....	50
10.7	Standby Mode	50
10.8	Extended Standby Mode	51
10.9	Power Reduction Register.....	51
10.10	Minimizing Power Consumption	51
10.11	Register Description	53
11	System Control and Reset	56
11.1	Resetting the AVR	56
11.2	Reset Sources	56
11.3	Power-on Reset.....	57
11.4	External Reset	58
11.5	Brown-out Detection	58
11.6	Watchdog System Reset.....	59
11.7	Internal Voltage Reference.....	59
11.8	Watchdog Timer	60
11.9	Register Description	63
12	Interrupts	66
12.1	Interrupt Vectors in ATmega48A and ATmega48PA.....	66
12.2	Interrupt Vectors in ATmega88A and ATmega88PA.....	68
12.3	Interrupt Vectors in ATmega168A and ATmega168PA.....	71
12.4	Interrupt Vectors in ATmega328 and ATmega328P.....	74
12.5	Register Description	77
13	External Interrupts	79
13.1	Pin Change Interrupt Timing.....	79

13.2	Register Description	80
14	<i>I/O-Ports</i>	84
14.1	Overview.....	84
14.2	Ports as General Digital I/O	85
14.3	Alternate Port Functions	89
14.4	Register Description	100
15	<i>8-bit Timer/Counter0 with PWM</i>	102
15.1	Features	102
15.2	Overview.....	102
15.3	Timer/Counter Clock Sources	103
15.4	Counter Unit	103
15.5	Output Compare Unit.....	104
15.6	Compare Match Output Unit.....	106
15.7	Modes of Operation	107
15.8	Timer/Counter Timing Diagrams	111
15.9	Register Description	113
16	<i>16-bit Timer/Counter1 with PWM</i>	120
16.1	Features	120
16.2	Overview.....	120
16.3	Accessing 16-bit Registers	122
16.4	Timer/Counter Clock Sources	125
16.5	Counter Unit	125
16.6	Input Capture Unit	126
16.7	Output Compare Units.....	128
16.8	Compare Match Output Unit.....	130
16.9	Modes of Operation	131
16.10	Timer/Counter Timing Diagrams	138
16.11	Register Description	140
17	<i>Timer/Counter0 and Timer/Counter1 Prescalers</i>	147
17.1	Internal Clock Source	147
17.2	Prescaler Reset	147
17.3	External Clock Source	147
17.4	Register Description	149
18	<i>8-bit Timer/Counter2 with PWM and Asynchronous Operation</i>	150
18.1	Features	150

18.2	Overview.....	150
18.3	Timer/Counter Clock Sources	151
18.4	Counter Unit	151
18.5	Output Compare Unit.....	152
18.6	Compare Match Output Unit.....	154
18.7	Modes of Operation	155
18.8	Timer/Counter Timing Diagrams	159
18.9	Asynchronous Operation of Timer/Counter2	160
18.10	Timer/Counter Prescaler	161
18.11	Register Description	162
19	<i>SPI – Serial Peripheral Interface</i>	169
19.1	Features	169
19.2	Overview.....	169
19.3	$\overline{SS}$ Pin Functionality	174
19.4	Data Modes	174
19.5	Register Description	176
20	<i>USART0</i>	179
20.1	Features	179
20.2	Overview.....	179
20.3	Clock Generation	180
20.4	Frame Formats	183
20.5	USART Initialization.....	184
20.6	Data Transmission – The USART Transmitter	185
20.7	Data Reception – The USART Receiver	188
20.8	Asynchronous Data Reception	192
20.9	Multi-processor Communication Mode	195
20.10	Examples of Baud Rate Setting.....	196
20.11	Register Description	200
21	<i>USART in SPI Mode</i>	205
21.1	Features	205
21.2	Overview.....	205
21.3	Clock Generation	205
21.4	SPI Data Modes and Timing.....	206
21.5	Frame Formats	206
21.6	Data Transfer.....	209

21.7	AVR USART MSPIM vs. AVR SPI	211
21.8	Register Description	212
22	2-wire Serial Interface	215
22.1	Features	215
22.2	2-wire Serial Interface Bus Definition	215
22.3	Data Transfer and Frame Format.....	216
22.4	Multi-master Bus Systems, Arbitration and Synchronization.....	219
22.5	Overview of the TWI Module	221
22.6	Using the TWI.....	223
22.7	Transmission Modes	225
22.8	Multi-master Systems and Arbitration.....	238
22.9	Register Description	239
23	Analog Comparator	243
23.1	Overview.....	243
23.2	Analog Comparator Multiplexed Input	243
23.3	Register Description	244
24	Analog-to-Digital Converter	246
24.1	Features	246
24.2	Overview.....	246
24.3	Starting a Conversion	248
24.4	Prescaling and Conversion Timing.....	249
24.5	Changing Channel or Reference Selection	251
24.6	ADC Noise Canceler	252
24.7	ADC Conversion Result.....	256
24.8	Temperature Measurement	256
24.9	Register Description	257
25	debugWIRE On-chip Debug System	262
25.1	Features	262
25.2	Overview.....	262
25.3	Physical Interface	262
25.4	Software Break Points	263
25.5	Limitations of debugWIRE	263
25.6	Register Description	263
26	Self-Programming the Flash, ATmega 48A/48PA	264
26.1	Overview.....	264

26.2	Addressing the Flash During Self-Programming	265
26.3	Register Description	270
27	Boot Loader Support – Read-While-Write Self-Programming	272
27.1	Features	272
27.2	Overview	272
27.3	Application and Boot Loader Flash Sections	272
27.4	Read-While-Write and No Read-While-Write Flash Sections	273
27.5	Boot Loader Lock Bits	275
27.6	Entering the Boot Loader Program	276
27.7	Addressing the Flash During Self-Programming	277
27.8	Self-Programming the Flash	278
27.9	Register Description	287
28	Memory Programming	289
28.1	Program And Data Memory Lock Bits	289
28.2	Fuse Bits	290
28.3	Signature Bytes	293
28.4	Calibration Byte	293
28.5	Page Size	294
28.6	Parallel Programming Parameters, Pin Mapping, and Commands	294
28.7	Parallel Programming	296
28.8	Serial Downloading	303
29	Electrical Characteristics – (TA = -40°C to 85°C)	308
29.1	Absolute Maximum Ratings*	308
29.2	DC Characteristics	308
29.3	Speed Grades	312
29.4	Clock Characteristics	313
29.5	System and Reset Characteristics	314
29.6	SPI Timing Characteristics	315
29.7	Two-wire Serial Interface Characteristics	317
29.8	ADC Characteristics	319
29.9	Parallel Programming Characteristics	320
30	Electrical Characteristics (TA = -40°C to 105°C)	322
30.1	Absolute Maximum Ratings*	322
30.2	DC Characteristics	322
31	Typical Characteristics – (TA = -40°C to 85°C)	326

31.1	ATmega48A Typical Characteristics	327
31.2	ATmega48PA Typical Characteristics	352
31.3	ATmega88A Typical Characteristics	377
31.4	ATmega88PA Typical Characteristics	401
31.5	ATmega168A Typical Characteristics	427
31.6	ATmega168PA Typical Characteristics	451
31.7	ATmega328 Typical Characteristics	477
31.8	ATmega328P Typical Characteristics	501
32	ATmega48PA Typical Characteristics – (TA = -40°C to 105°C)	526
32.1	Active Supply Current.....	526
32.2	Idle Supply Current.....	529
32.3	Power-down Supply Current.....	531
32.4	Standby Supply Current	532
32.5	Pin Pull-Up.....	533
32.6	Pin Driver Strength	536
32.7	Pin Threshold and Hysteresis.....	538
32.8	BOD Threshold.....	541
32.9	Internal Oscillator Speed	542
32.10	Current Consumption of Peripheral Units	545
32.11	Current Consumption in Reset and Reset Pulsewidth	547
33	ATmega88PA Typical Characteristics – (TA = -40°C to 105°C)	549
33.1	Active Supply Current.....	549
33.2	Idle Supply Current.....	552
33.3	Power-down Supply Current.....	554
33.4	Power-save Supply Current.....	555
33.5	Pin Pull-Up.....	556
33.6	Pin Driver Strength	559
33.7	Pin Threshold and Hysteresis.....	561
33.8	BOD Threshold.....	564
33.9	Internal Oscillator Speed	565
33.10	Current Consumption of Peripheral Units	568
33.11	Current Consumption in Reset and Reset Pulsewidth	570
34	ATmega168PA Typical Characteristics – (TA = -40°C to 105°C)	572
34.1	Active Supply Current.....	572
34.2	Idle Supply Current.....	575

34.3	Power-down Supply Current.....	577
34.4	Power-save Supply Current.....	578
34.5	Standby Supply Current	579
34.6	Pin Pull-Up.....	579
34.7	Pin Driver Strength	582
34.8	Pin Threshold and Hysteresis.....	584
34.9	BOD Threshold	587
34.10	Internal Oscillator Speed	590
34.11	Current Consumption of Peripheral Units	592
34.12	Current Consumption in Reset and Reset Pulsewidth	595
35	ATmega328P Typical Characteristics – (TA = -40°C to 105°C)	597
35.1	ATmega328P Active Supply Current.....	597
35.2	Idle Supply Current.....	600
35.3	Power-down Supply Current.....	602
35.4	Power-save Supply Current.....	603
35.5	Standby Supply Current	604
35.6	Pin Pull-Up.....	604
35.7	Pin Driver Strength	607
35.8	Pin Threshold and Hysteresis.....	609
35.9	BOD Threshold	612
35.10	Internal Oscillator Speed	614
35.11	Current Consumption of Peripheral Units	617
35.12	Current Consumption in Reset and Reset Pulsewidth	619
36	Register Summary	621
37	Instruction Set Summary	625
38	Ordering Information	628
38.1	ATmega48A	628
38.2	ATmega48PA	629
38.3	ATmega88A.....	630
38.4	ATmega88PA	631
38.5	ATmega168A.....	632
38.6	ATmega168PA	633
38.7	ATmega328	634
38.8	ATmega328P	635
39	Packaging Information	636

39.1	32A	636
39.2	32CC1	637
39.3	28M1.....	638
39.4	32M1-A	639
39.5	28P3	640
40	Errata	641
40.1	Errata ATmega48A.....	641
40.2	Errata ATmega48PA	642
40.3	Errata ATmega88A.....	644
40.4	Errata ATmega88PA	645
40.5	Errata ATmega168A.....	647
40.6	Errata ATmega168PA	649
40.7	Errata ATmega328	651
40.8	Errata ATmega328P.....	654
41	Datasheet Revision History	657
41.1	Rev. A – 10/2018.....	657
41.2	Rev. 8271J – 11/2015	657
41.3	Rev. 8271I – 10/2014	657
41.4	Rev. 8271H – 08/2014.....	657
41.5	Rev. 8271G – 02/2013	657
41.6	Rev. 8271F – 08/2012	658
41.7	Rev. 8271E – 07/2012.....	658
41.8	Rev. 8271D – 05/11.....	658
41.9	Rev. 8271C – 08/10.....	659
41.10	Rev. 8271B – 04/10.....	659
41.11	Rev. 8271A – 12/09.....	659

1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port B (PB7:0) 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" on page 91](#) and ["System Clock and Clock Options" on page 36](#).

1.1.4 Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5...0 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.

1.1.5 PC6/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 29-11 on page 314](#). Shorter pulses are not ensured to generate a Reset.

The various special features of Port C are elaborated in ["Alternate Functions of Port C" on page 94](#).|

1.1.6 Port D (PD7:0)

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.

The various special features of Port D are elaborated in ["Alternate Functions of Port D" on page 97](#).

1.1.7 **AV_{CC}**

AV_{CC} is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. 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 PC6...4 use digital supply voltage, V_{CC}.

1.1.8 **AREF**

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

1.1.9 **ADC7:6 (TQFP and QFN/MLF Package Only)**

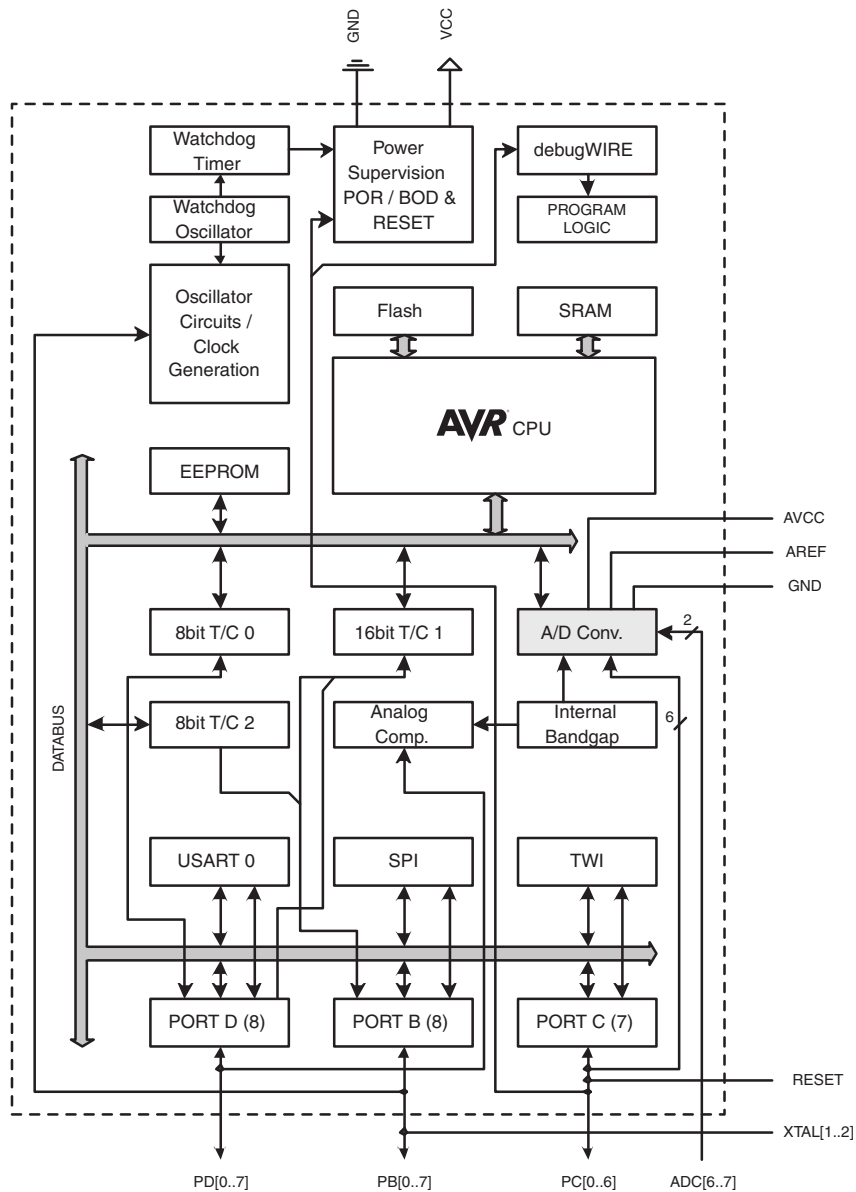
In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

2. Overview

The ATmega48A/PA/88A/PA/168A/PA/328/P 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 ATmega48A/PA/88A/PA/168A/PA/328/P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The 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 ATmega48A/PA/88A/PA/168A/PA/328/P provides the following features: 4K/8Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 256/512/512/1Kbytes EEPROM, 512/1K/1K/2Kbytes 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, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, 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.

Microchip 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 Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Microchip'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 non-volatile 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 ATmega48A/PA/88A/PA/168A/PA/328/P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega48A/PA/88A/PA/168A/PA/328/P 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.

2.2 Comparison Between Processors

The ATmega48A/PA/88A/PA/168A/PA/328/P differ only in memory sizes, boot loader support, and interrupt vector sizes. [Table 2-1](#) summarizes the different memory and interrupt vector sizes for the devices.

Table 2-1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega48PA	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega88A	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega88PA	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega168A	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega168PA	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega328	32KBytes	1KBytes	2KBytes	2 instruction words/vector
ATmega328P	32KBytes	1KBytes	2KBytes	2 instruction words/vector

ATmega48A/PA/88A/PA/168A/PA/328/P support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega 48A/48PA there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash

3. Resources

A comprehensive set of development tools, application notes and data sheets are available for download on www.microchip.com

4. 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.

5. 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. 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 “SBR”, “SBRC”, “SBR”, and “CBR”.

6. Capacitive Touch Sensing

The QTouch Library provides a simple to use solution to realize touch sensitive interfaces on most 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 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 APIs to retrieve the channel information and determine the touch sensor states.

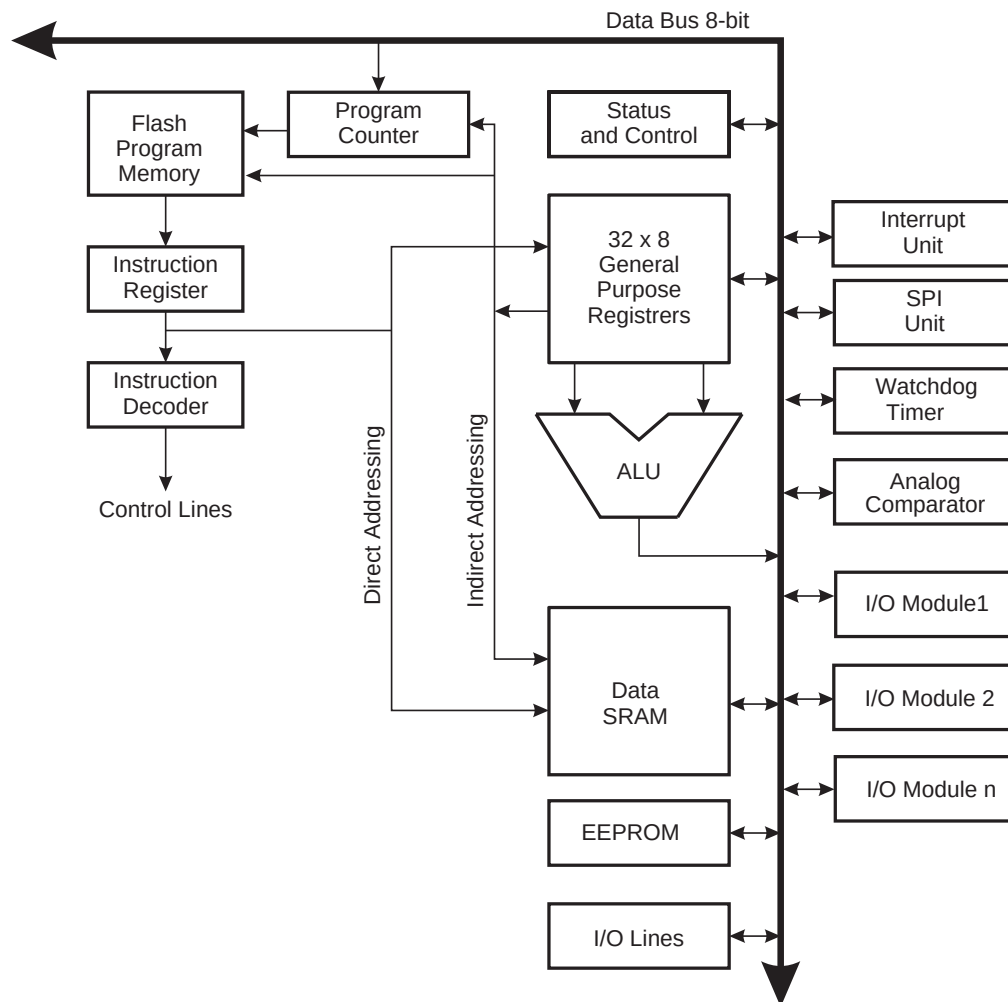
The QTouch Library is FREE and downloadable from the Microchip website at the following location <http://www.microchip.com>. For implementation details and other information, refer to the QTouch Library User Guide - also available for download from the Microchip website.

7. AVR CPU Core

7.1 Overview

This section discusses the 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 7-1. Block Diagram of the AVR 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 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.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most 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 in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The 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 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 AVR architecture.

The memory spaces in the 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 ATmega48A/PA/88A/PA/168A/PA/328/P has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

7.2 ALU – Arithmetic Logic Unit

The high-performance AVR 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 are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

7.3 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering 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.

7.3.1 SREG – AVR Status Register

The AVR 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	

- **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 the “Instruction Set Description” for detailed information.

- **Bit 4 – S: Sign Bit, $S = N \oplus V$**

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

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

The Two’s Complement Overflow Flag V supports two’s complement arithmetic. See the “Instruction Set Description” for detailed information.

- **Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

- **Bit 0 – C: Carry Flag**

The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

7.4 General Purpose Register File

The Register File is optimized for the 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 7-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 7-2. AVR CPU General Purpose Working Registers

	7	0	Addr.	
	R0		0x00	
	R1		0x01	
	R2		0x02	
	...			
	R13		0x0D	
	R14		0x0E	
	R15		0x0F	
General	R16		0x10	
Purpose	R17		0x11	
Working	...			
Registers	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 7-2, 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.

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

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 7-3.

Figure 7-3. The X-, Y-, and Z-registers



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

7.5 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. Note that the Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack. The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer.

The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM, see [Table 8-3 on page 28](#).

See [Table 7-1](#) for Stack Pointer details.

Table 7-1. Stack Pointer instructions

Instruction	Stack pointer	Description
PUSH	Decrement by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decrement by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Increment by 1	Data is popped from the stack
RET RETI	Increment by 2	Return address is popped from the stack with return from subroutine or return from interrupt

The 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 AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

7.5.1 SPH and SPL – Stack Pointer High and Stack Pointer Low Register

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	SP15	SP14	SP13	SP12	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/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	

7.6 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The 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 7-4 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 7-4. The Parallel Instruction Fetches and Instruction Executions

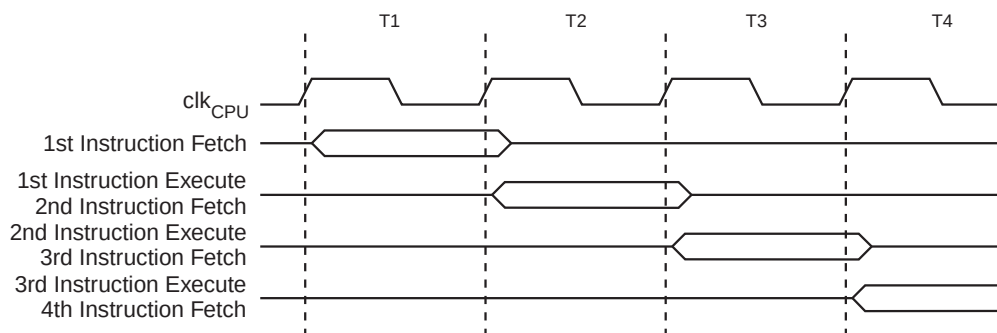
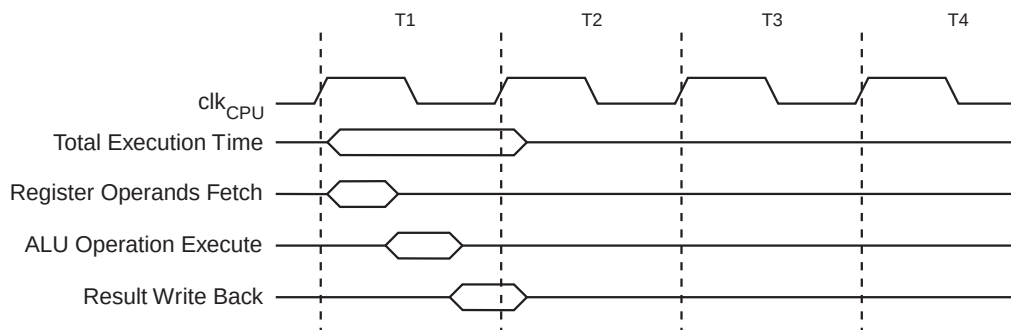


Figure 7-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 7-5. Single Cycle ALU Operation



7.7 Reset and Interrupt Handling

The 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 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. 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 the section ["Memory Programming" on page 289](#) for details.

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 66](#). 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. 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). Refer to ["Interrupts" on page 66](#) for more information. The Reset Vector can also be moved to the start of the Boot Flash section by

programming the BOOTRST Fuse, see ["Boot Loader Support – Read-While-Write Self-Programming"](#) on page 272.

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 AVR 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 restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be 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	
in	r16, SREG ; store SREG value
cli	; disable interrupts during timed sequence
sbi	EECR, EEMPE ; start EEPROM write
sbi	EECR, EEPE
out	SREG, r16 ; restore SREG value (I-bit)
C Code Example	
<pre> char cSREG; cSREG = SREG; /* store SREG value */ /* disable interrupts during timed sequence */ _cli(); EECR = (1<<EEMPE); /* start EEPROM write */ EECR = (1<<EEPE); SREG = cSREG; /* restore SREG value (I-bit) */ </pre>	

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) */
```

7.7.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, 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 four 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 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.