



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



---

## UHF ASK/FSK Transmitter with the Atmel AVR Microcontroller

---

### DATASHEET

---

### General Features

---

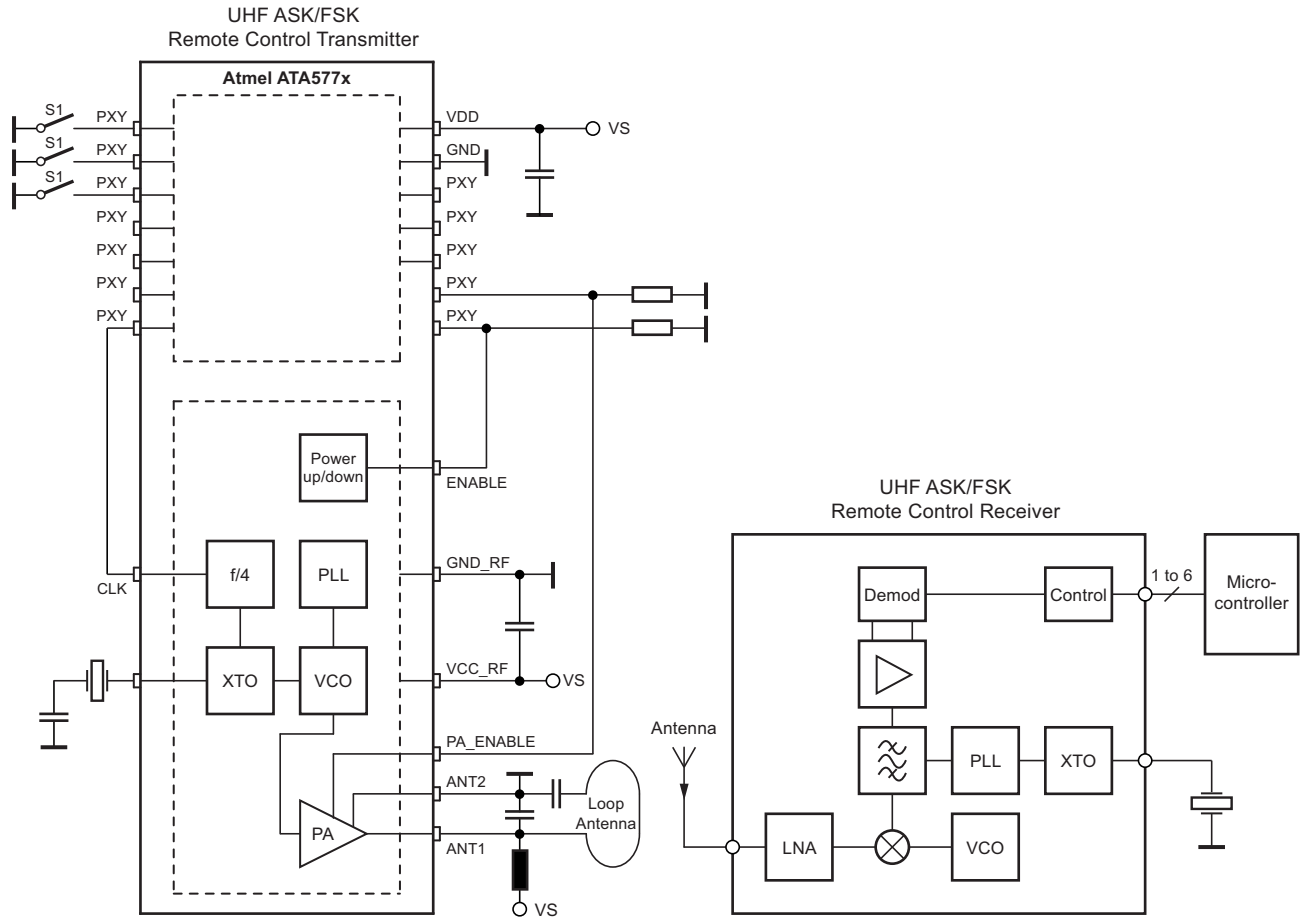
- Atmel® AVR® microcontroller and RF transmitter PLL in a single QFN24 5mm × 5mm package (pitch 0.65mm)
  - Operating frequency ranges 310MHz to 350MHz, 429MHz to 439MHz and 868MHz to 928MHz
- Temperature range –40°C to +85°C
- Supply voltage 2.0V to 3.6V allowing usage of single Li-cell power supply
- Low power consumption
  - Active mode: typical 9.8mA at 3.0V and 4MHz microcontroller-clock
  - Power-down mode: Typical 200nA at 3.0V
- Modulation scheme ASK/FSK
- Integrated PLL loop filter
- Output power of 8dBm at 315MHz / 7.5dBm at 433.92MHz / 5.5dBm at 868.3MHz
- Easy to design-in due to excellent isolation of the PLL from the PA and power supply
- Single-ended antenna output with high efficient power amplifier
- Very robust ESD protection: HBM 2500V, MM100V, CDM 1000V
- High performance, low power AVR 8-bit microcontroller
- Advanced RISC architecture
- Non-volatile program and data memories
  - 4Kbytes of in-system programmable program memory flash
  - 256Bytes in-system programmable EEPROM
  - 256Bytes internal SRAM
- Programming lock for self-programming flash program and EEPROM data security
- Peripheral features
  - Two timer/counter, 8- and 16-bit counters with two PWM channels on both
  - 10-bit ADC
  - On-chip analog comparator
  - Programmable watchdog timer with separate on-chip oscillator
  - Universal serial interface (USI)

- Special microcontroller features
  - debugWIRE on-chip debug system
  - In-system programmable via SPI port
  - External and internal interrupt sources
  - Pin change interrupt on 12 pins
  - Enhanced power-on reset circuit
  - Programmable brown-out detection circuit
  - Internal calibrated oscillator
  - On-chip temperature sensor
- 12 programmable I/O lines

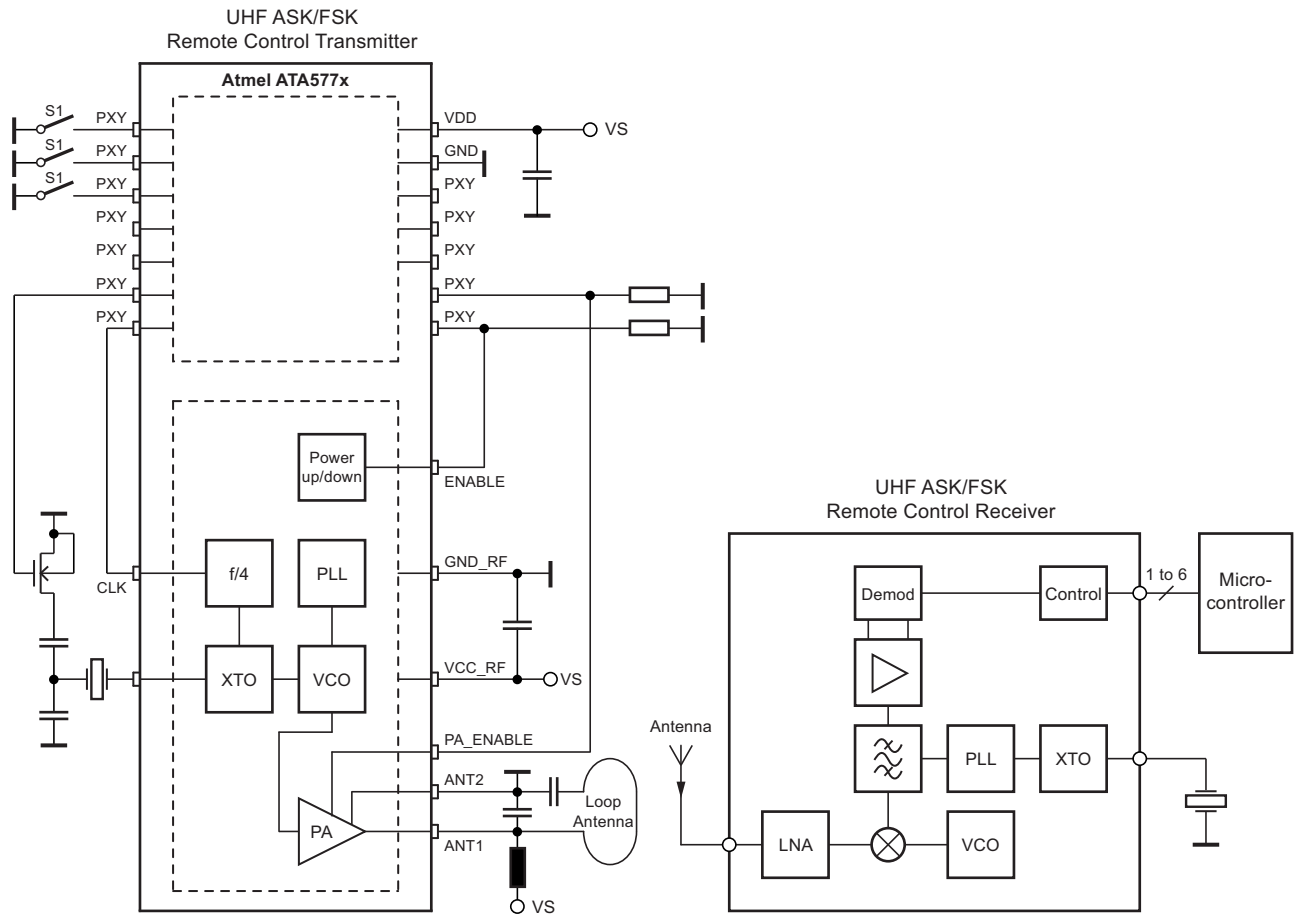
# 1. General Description

The Atmel® ATA5771C/73C/74C is a highly flexible programmable transmitter containing the Atmel AVR® microcontroller Atmel ATtiny44V and the UHF PLL transmitters in a small QFN24 5mm × 5mm package. This device is a member of a transmitter family covering several operating frequency ranges, which has been specifically developed for the demands of RF low-cost data transmission systems with data rates up to 32kBit/s using ASK or FSK modulation. Its primary applications are in the application of Remote Keyless-Entry (RKE), Passive Entry Go (PEG) System and Remote Start. The ATA5771 is designed for 868MHz application, whereas ATA5773 for 315MHz application and ATA5774 for 434MHz application.

**Figure 1-1. ASK System Block Diagram**



**Figure 1-2. FSK System Block Diagram**





## 2. Pin Configuration

Figure 2-1. Pinning QFN24 5mm × 5mm

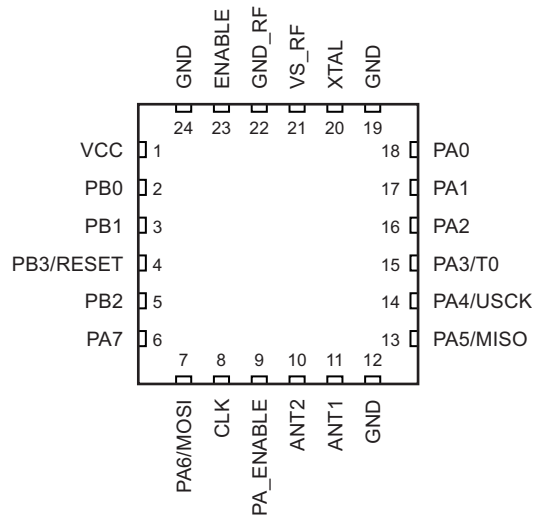


Table 2-1. Pin Description

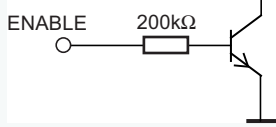
Pin	Symbol	Function
1	VCC	Microcontroller supply voltage
2	PB0	Port B is a 4-bit bi-directional I/O port with internal pull-up resistor
3	PB1	Port B is a 4-bit bi-directional I/O port with internal pull-up resistor
4	PB3/RESET	Port B is a 4-bit bi-directional I/O port with internal pull-up resistor/reset input
5	PB2	Port B is a 4-bit bi-directional I/O port with internal pull-up resistor
6	PA7	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
7	PA6 / MOSI	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
8	CLK	Clock output signal for microcontroller. The clock output frequency is set by the crystal to $f_{XTAL}/4$
9	PA_ENABLE	Switches on power amplifier. Used for ASK modulation
10	ANT2	Emitter of antenna output stage
11	ANT1	Open collector antenna output
12	GND	Ground
13	PA5/MISO	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
14	PA4/SCK	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
15	PA3/T0	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
16	PA2	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
17	PA1	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
18	PA0	Port A is an 8-bit bi-directional I/O port with internal pull-up resistor
19	GND	Microcontroller ground
20	XTAL	Connection for crystal
21	VS_RF	Transmitter supply voltage
22	GND_RF	Transmitter ground
23	ENABLE	Enable input
24	GND	Ground
	GND	Ground/backplane (exposed die pad)

## 2.1 Pin Configuration of RF Pins

Table 2-2. Pin Description

Pin	Symbol	Function	Configuration
8	CLK	Clock output signal for microcontroller. The clock output frequency is set by the crystal to $f_{XTAL}/4$ .	
9	PA_ENABLE	Switches on power amplifier. Used for ASK modulation.	
10	ANT2	Emitter of antenna output stage.	
11	ANT1	Open collector antenna output.	
20	XTAL	Connection for crystal.	

**Table 2-2. Pin Description (Continued)**

Pin	Symbol	Function	Configuration
21	VS	Supply voltage	See ESD protection circuitry (see <a href="#">Figure 5-1 on page 155</a> ).
22	GND	Ground	See ESD protection circuitry (see <a href="#">Figure 5-1 on page 155</a> ).
23	ENABLE	Enable input	



## 3. Functional Description

Figure 1-1 on page 3 and Figure 1-2 on page 4 show the interconnections between the microcontroller and the RF part for a typical application. In the recommended application circuits the clock output of the RF transmitter is connected to the microcontroller in order to be able to generate data rate with tolerance lower than 3%. The transmitter's crystal oscillator (XTO), phase locked loop (PLL) and clock generation are started using pin ENABLE. The power amplifier (PA) is activated using the connection to the pin PA\_ENABLE. The FSK modulation is performed due to pulling of the crystal load capacitance for this purpose the microcontroller output port together with an external switch applies this modulation technique. For the ASK modulation the power amplifier will be switched on and off by modulating the PA\_ENABLE pin due to the data.

To wake up the system from standby mode at least one event is required, which will be performed by pushing tone button. After this event the microcontroller starts up with the internal RC oscillator. For the TX operation the user software must additionally control just 2 pins, the pin ENABLE and pin PA\_ENABLE. In case of the FSK modulation one additional connection from microcontroller is necessary to perform the pulling of the crystal load capacitance.

If ENABLE and PA\_ENABLE are set to LOW the transmitter is in standby mode with the suitable mode setting of the microcontroller (MCU) the power consumption will be reduced.

If ENABLE is set to HIGH and PA\_ENABLE to LOW, the XTO, PLL, and the Clock driver of the RF transmitter are activated and the VCO frequency is 32 times the XTO frequency. The Atmel ATA5771 and Atmel ATA5774 require typically shorter than 1 ms until the PLL is locked and the transmitter's clock output is stable, while the Atmel ATA5773 requires time shorter than 3 ms for this progress.

If both ENABLE and PA\_ENABLE are set to HIGH the whole RF transmitter (XTO, PLL, Clock driver and power Amplifier) is activated. The ASK modulation is achieved by switching on and off the power amplifier via pin PA\_ENABLE. The FSK modulation is performed by pulling the crystal load capacitor which will change the reference frequency of the PLL due to the data. The microcontroller modulates the load capacitance of the crystal using an external switch. A MOS transistor with a low parasitic capacitance is recommended to be used for this purpose. During the FSK modulation is the PA\_ENABLE pin set to HIGH.

To generate the data for the telegram the internal RC oscillator of the microcontroller is not accurate enough because this will be affected by ambient temperature and operating voltage. To reduce the variation of the data rate lower than 3% the clock frequency generated by the RF transmitter should be used as a reference. The MCU has to wait at least longer than 3ms for ATA5773 after setting ENABLE to HIGH, before the clock output from the RF transmitter can be used. For ATA5771 and ATA5774 the MCU must wait longer than 1 ms until the clock output is stable. The clock output with the crystal tolerance is connected to the timer0 of the MCU. This timer clocks the USI to generate the data rate. In the Two serial synchronous data transfer modes will be provided by USI. This will be pass out with different physical I/O ports, two wire mode is used for ASK and the three wire mode for FSK.

### 3.1 Frequency Generation

In Atmel ATA5773 and Atmel ATA5774 the VCO is locked to 32 times crystal frequency hence the following crystal is needed

- 9.8438MHz for 315MHz application
- 13.56MHz for 433.92MHz application

The VCO of ATA5771 is locked to 64 times crystal frequency therefore the necessary crystal frequency is

- 13.5672MHz for 868.3MHz application
- 14.2969MHz for 915MHz application

Due to the high integration the PLL and VCO peripheral elements are integrated.

The XTO is a series resonance oscillator that only one capacitor together with a crystal connected in series to GND are needed as external elements. Until the PLL and clock output is stable the following time can be expected

- 3ms for ATA5773
- 1ms for ATA5771 and ATA5774

Therefore, a time delay of  $\geq 3$  ms for ATA5773 and  $\geq 1$  ms for ATA5771/74 between activation of pin ENABLE and switching on the pin PA\_ENABLE must be implemented in the software.

## 3.2 ASK Transmission

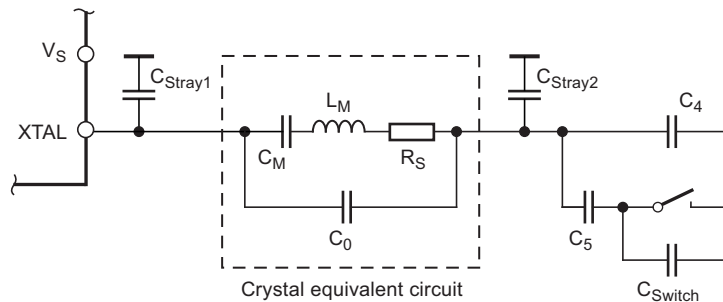
The ASK modulation will be performed by switching the power amplifier on and off due to the data to be transmitted. The transmitter's XTO and PLL are activated by setting the pin ENABLE to HIGH. Between the activation of the pin ENABLE and the pin PA\_ENABLE minimum 3ms time delay must be taken into account for the application with ATA5773, whereas a minimum 1ms time delay for an application using ATA5771 or ATA5774. After the mentioned time delay the generated clock frequency by the RF transmitter can be used as reference for the data generation of the microcontroller block.

## 3.3 FSK Transmission

The transmitter's XTO and PLL are activated by setting the pin ENABLE to HIGH. Like the ASK transmission a defined time delay must be taken into account between the activation of the pin ENABLE and the pin PA\_ENABLE. After this time delay the clock frequency can be used as reference for the data rate generation and the data transmission using FSK modulation is ready. For this purpose an additional capacitor to the crystal's load capacitor will be switched between the high impedance and ground due to the data rate. Thus the reference frequency, which is crystal frequency, of the RF transmitter will be modulated. This results also in the transmitted spectrum. It is important that the switching element must have a defined low parasitic capacitance.

The accuracy of the frequency deviation with XTAL pulling method is about  $\pm 25\%$  when the following tolerances are considered.

**Figure 3-1. Tolerances of Frequency Modulation**



Using  $C_4 = 8.2\text{pF} \pm 5\%$ ,  $C_5 = 10\text{pF} \pm 5\%$ , a switch port with  $C_{\text{Switch}} = 3\text{pF} \pm 10\%$ , stray capacitances on each side of the crystal of  $C_{\text{Stray1}} = C_{\text{Stray2}} = 1\text{pF} \pm 10\%$ , a parallel capacitance of the crystal of  $C_0 = 3.2\text{pF} \pm 10\%$  and a crystal with  $C_M = 13\text{fF} \pm 10\%$ , results in a typical FSK deviation of  $\pm 21.5\text{kHz}$  with worst case tolerances of  $\pm 16.25\text{kHz}$  to  $\pm 28.01\text{kHz}$ .

## 3.4 CLK Output

RF transmitter generated clock signal based on the divided crystal frequency. This will be available for the microcontroller as reference. The delivered signal is CMOS compatible if the load capacitance is lower than 10pF.

### 3.4.1 Clock Pulse Take-over

The clock of the crystal oscillator can be used for clocking the microcontroller, which starts with an integrated RC-oscillator. After the generated clock signal of the RF transmitter is stable, the microcontroller will take over the clock signal and use it as reference generating the data rate, so that the message can be transmitted with crystal accuracy.

### 3.4.2 Output Matching and Power Setting

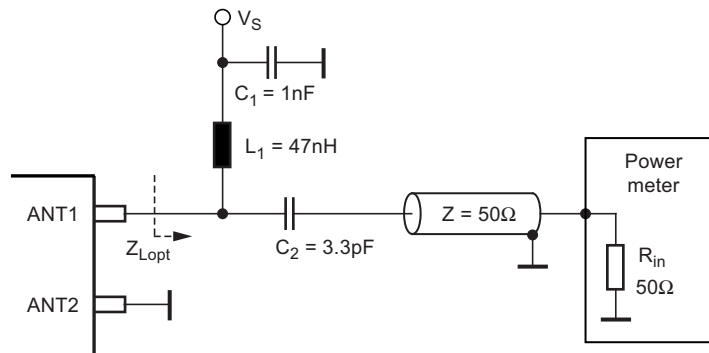
The power amplifier is an open-collector output delivering a current pulse, which is nearly independent from the load impedance. Thus the delivered output power can be tuned via the load impedance of the antenna and the matching elements. This output configuration enables simple matching to any kind of antenna or to 50Ω which results in a high power efficiency  $\{\eta = P_{out}/(I_{S,PA} V_S)\}$ . The maximum output power can be achieved at 3V supply voltage when the load impedance is optimized to

- $Z_{Load} = (255 + j192)\Omega$  for the Atmel ATA5773 with the power efficiency of 40%
  - **Background:** The current pulse of the power amplifier is 9mA and the maximum output power is delivered to a resistive load of 400Ω if the 1.0pF output capacitance of the power amplifier is compensated by the load impedance. And thus the load impedance of  $Z_{Load} = 400\Omega \parallel j/(2 \times \pi \times f \times 1.0\text{pF}) = (255 + j192)\Omega$  is achieved for the maximum output power of 8dBm.
- $Z_{Load} = (166 + j223)\Omega$  for the Atmel ATA5774 with the power efficiency of 36%
  - **Background:** The current pulse of the power amplifier is 9mA and the maximum output power is delivered to a resistive load of 465Ω if the 1.0pF output capacitance of the power amplifier is compensated by the load impedance. And thus the load impedance of  $Z_{Load} = 465\Omega \parallel j/(2 \times \pi \times f \times 1.0\text{pF}) = (166 + j223)\Omega$  is achieved for the maximum output power of 7.5dBm.
- $Z_{Load} = (166 + j226)\Omega$  for the Atmel ATA5771 with the power efficiency of 24%
  - **Background:** The current pulse of the power amplifier is 7.7mA and the maximum output power is delivered to a resistive load of 475Ω if the 0.53pF output capacitance of the power amplifier is compensated by the load impedance. And thus the load impedance of  $Z_{Load} = 475\Omega \parallel j/(2 \times \pi \times f \times 0.53\text{pF}) = (166 + j226)\Omega$  is achieved for the maximum output power of 5.5dBm.

The load impedance is defined as the impedance seen from the power amplifier (pin ANT1 and pin ANT2) into the matching network. This large signal load impedance should not be mixed up with the small signal input impedance delivered as input characteristic of RF amplifiers and measured from the application into the IC, instead of from the IC into the application. Please take note that there must be a low resistive path between the  $V_S$  and the collector output of the PA to deliver the DC current. Reduced output power will be achieved by lowering the real parallel part of the load impedance where the parallel imaginary part should be kept constant.

Output power measurement can be performed using the circuit shown in Figure 3-2. Note that the component values must be changed to compensate for the individual board parasitics until the RF power amplifier has the right load impedance. In addition, the damping of the cable used to measure the output power must be calibrated out.

**Figure 3-2. Output Power Measurement Atmel ATA5771C/73C/74C**



## 4. Microcontroller Block

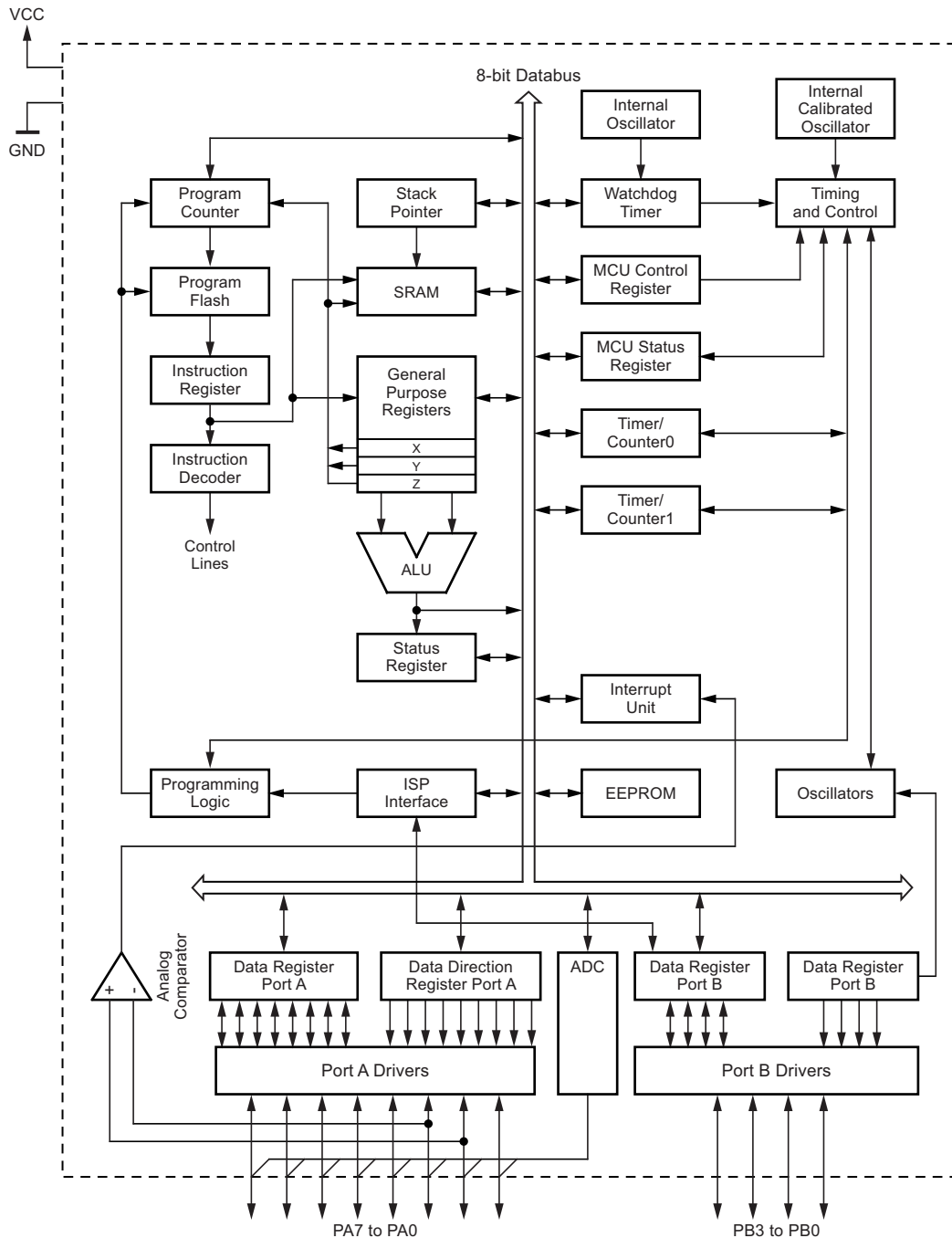
These data are referred to the data base of microcontroller Atmel ATtiny44V.

### 4.1 Overview

The ATtiny44V is a low-power CMOS 8-bit microcontroller based on the Atmel AVR<sup>®</sup> enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATtiny44V achieves throughputs approaching 1MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

### 4.2 Block Diagram

Figure 4-1. Block Diagram



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 Atmel ATtiny44V provides the following features: 4K byte of In-System Programmable Flash, 256 bytes EEPROM, 256 bytes SRAM, 12 general purpose I/O lines, 32 general purpose working registers, a 8-bit Timer/Counter with two PWM channels, a 16-bit timer/counter with two PWM channels, Internal and External Interrupts, a 8-channel 10-bit ADC, programmable gain stage (1x, 20x) for 12 differential ADC channel pairs, a programmable watchdog timer with internal oscillator, internal calibrated oscillator, and three software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counter, ADC, analog comparator, and interrupt system to continue functioning. The power-down mode saves the register contents, disabling all chip functions until the next interrupt or hardware reset. The ADC noise reduction mode stops the CPU and all I/O modules except 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.

The device is manufactured using the Atmel high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be re-programmed in-system through an SPI serial interface, by a conventional non-volatile memory programmer or by an on-chip boot code running on the AVR core.

The ATtiny44V 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.

### 4.3 Automotive Quality Grade

The ATtiny44V have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949 grade 1. This datasheet contains limit values extracted from the results of extensive characterization (temperature and voltage). The quality and reliability of the ATtiny44V have been verified during regular product qualification as per AEC-Q100.

As indicated in the ordering information paragraph, the product is available in only one temperature grade.

**Table 4-1. Temperature Grade Identification for Automotive Products**

Temperature	Temperature Identifier	Comments
-40°C; +125°C	Z	Full automotive temperature range

## 4.4 Pin Descriptions

### 4.4.1 VCC

Supply voltage.

### 4.4.2 GND

Ground.

### 4.4.3 Port B (PB3...PB0)

Port B is a 4-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 except PB3 which has the RESET capability. To use pin PB3 as an I/O pin, instead of RESET pin, program ('0') RSTDISBL fuse. 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.

Port B also serves the functions of various special features of the Atmel ATtiny44V as listed on [Section 4.14.3 "Alternate Port Functions" on page 57](#).

### 4.4.4 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 [Figure 4-13 on page 39](#). Shorter pulses are not guaranteed to generate a reset.

### 4.4.5 Port A (PA7...PA0)

Port A is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A has an alternate functions as analog inputs for the ADC, analog comparator, timer/counter, SPI and pin change interrupt as described in [Section 4.14.3 "Alternate Port Functions" on page 57](#).

## 4.5 Resources

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

## 4.6 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 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".

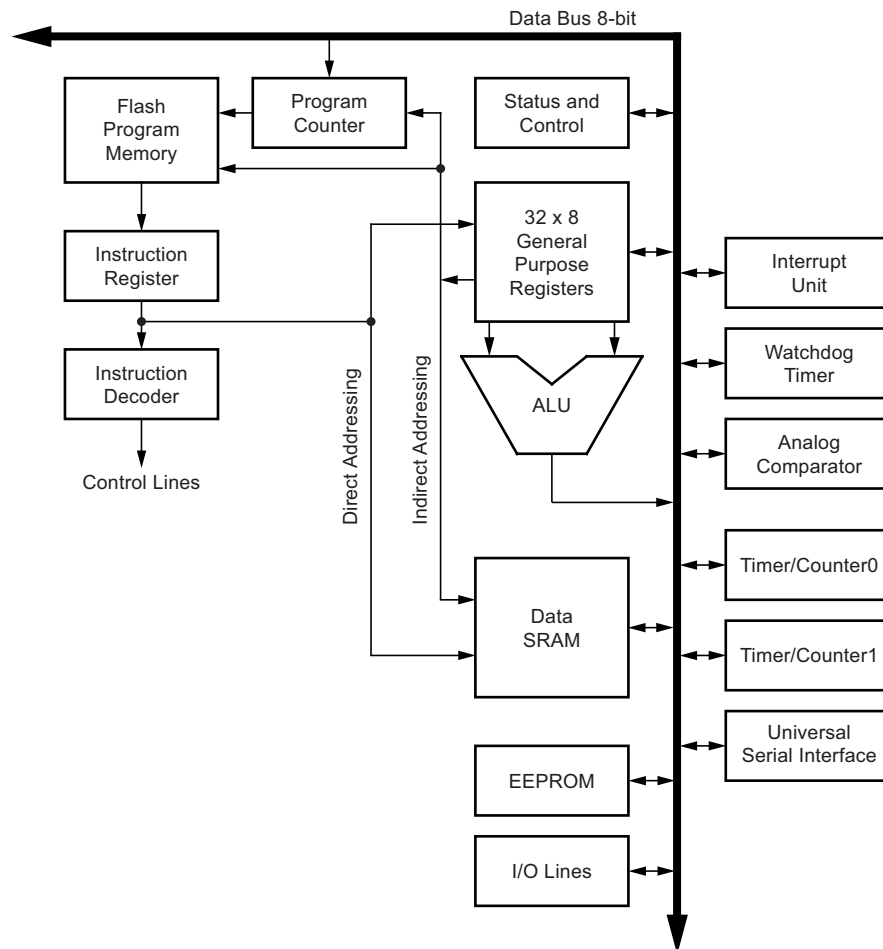
## 4.7 CPU Core

### 4.7.1 Overview

This section discusses the Atmel AVR<sup>®</sup> 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.

### 4.7.2 Architectural Overview

Figure 4-2. Block Diagram of the Atmel 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 Atmel AVR® instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The stack pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the 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.

### 4.7.3 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.

### 4.7.4 Status Register

The status register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the status register is updated after all ALU operations, as specified in the instruction set reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code. The status register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

#### 4.7.4.1 SREG – AVR Status Register

Bit	7	6	5	4	3	2	1	0	
0x3F (0xSF)	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 arithmetics. 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.

## 4.7.5 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 4-3 shows the structure of the 32 general purpose working registers in the CPU.

**Figure 4-3. Atmel AVR CPU General Purpose Working Registers**

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

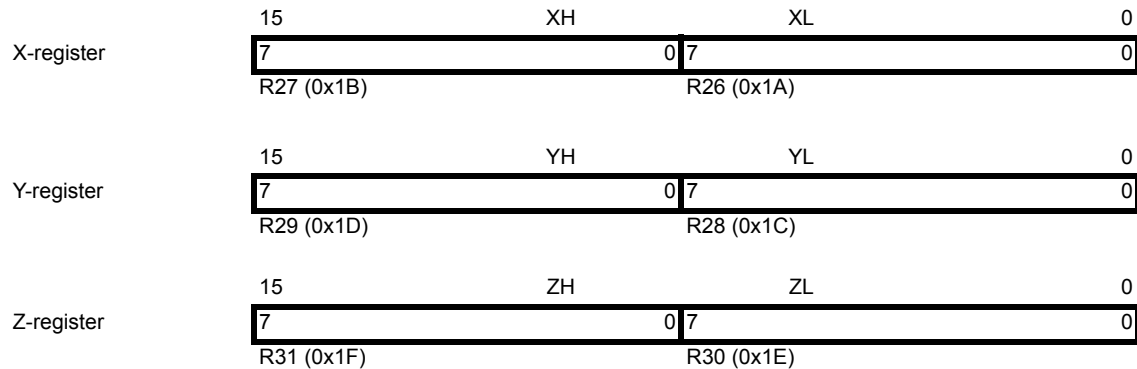
Most of the instructions operating on the register file have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4-3, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user data space. Although not 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.

### 4.7.5.1 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 4-4 on page 18](#).

**Figure 4-4. 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).

### 4.7.6 Stack Pointer

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

The stack pointer points to the data SRAM stack area where the subroutine and interrupt stacks are located. This stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The stack pointer must be set to point above 0x60. The stack pointer is decremented by one when data is pushed onto the stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the stack with subroutine call or interrupt. The stack pointer is incremented by one when data is popped from the stack with the POP instruction, and it is incremented by two when data is popped from the stack with return from subroutine RET or return from interrupt RETI.

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

#### 4.7.6.1 SPH and SPL – Stack Pointer High and Low

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	<b>SP15</b>	<b>SP14</b>	<b>SP13</b>	<b>SP12</b>	<b>SP11</b>	<b>SP10</b>	<b>SP9</b>	<b>SP8</b>	<b>SPH</b>
0x3D (0x5D)	<b>SP7</b>	<b>SP6</b>	<b>SP5</b>	<b>SP4</b>	<b>SP3</b>	<b>SP2</b>	<b>SP1</b>	<b>SP0</b>	<b>SPL</b>
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

## 4.7.7 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 4-5 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 4-5. The Parallel Instruction Fetches and Instruction Executions

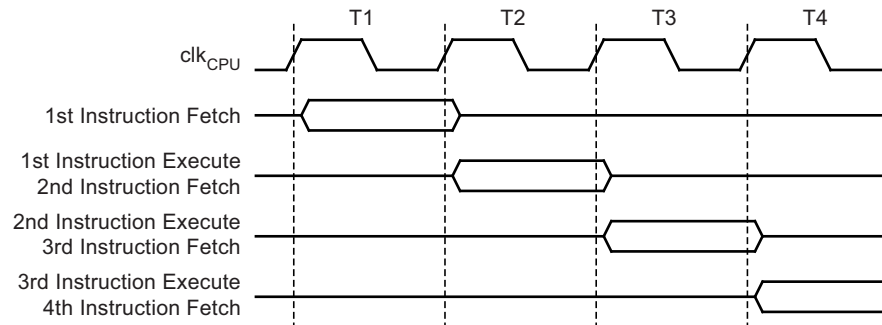
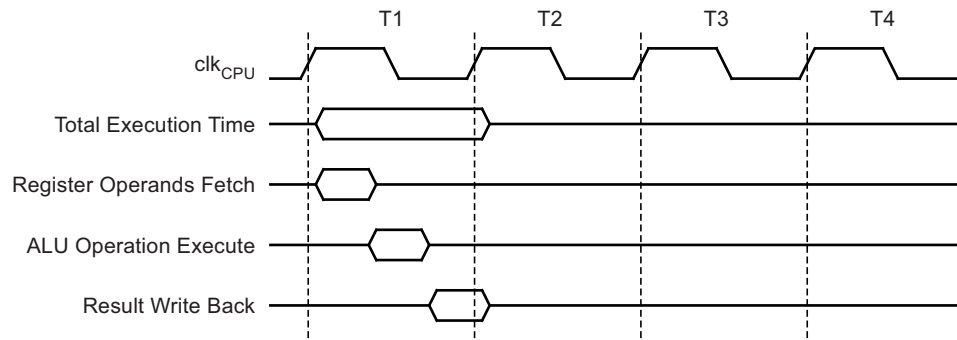


Figure 4-6 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 4-6. Single Cycle ALU Operation



## 4.7.8 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.

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 Section 4.12 “Interrupts” on page 47. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INTO – the external interrupt request 0.

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

There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt flag. For these interrupts, the program counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, and hardware clears the corresponding interrupt flag. Interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the interrupt flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the global interrupt enable bit is cleared, the corresponding interrupt flag(s) will be set and remembered until the global interrupt enable bit is set, and will then be executed by order of priority.

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

When the Atmel AVR<sup>®</sup> 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	
<b>in</b>	r16, SREG ; store SREG value
<b>cli</b>	; disable interrupts during timed sequence
<b>sbi</b>	EECR, EEMPE ; start EEPROM write
<b>sbi</b>	EECR, EEPE
<b>out</b>	SREG, r16 ; restore SREG value (I-bit)

C Code Example	
<b>char</b>	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) */

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	
<b>sei</b>	; set Global Interrupt Enable
<b>sleep</b>	; 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) */	

#### 4.7.8.1 Interrupt Response Time

The interrupt execution response for all the enabled Atmel<sup>®</sup> 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.

## 4.8 Memories

This section describes the different memories in the Atmel® ATtiny44V. The Atmel AVR® architecture has two main memory spaces, the Data memory and the Program memory space. In addition, the ATtiny44V features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

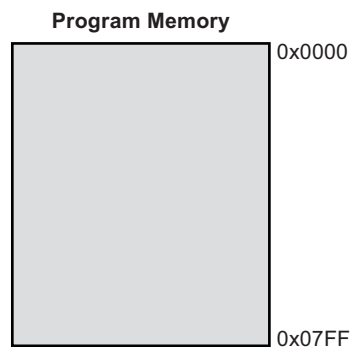
### 4.8.1 In-System Re-programmable Flash Program Memory

The ATtiny44V contains 4Kbyte on-chip in-system reprogrammable flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the flash is organized as 2048 x 16.

The flash memory has an endurance of at least 10,000 write/erase cycles. The ATtiny44V program counter (PC) is 11 bits wide, thus addressing the 2048 program memory locations. [Section 4.23 “Memory Programming” on page 142](#) contains a detailed description on flash data serial downloading using the SPI pins.

Constant tables can be allocated within the entire program memory address space (see the LPM – load program memory Instruction description). Timing diagrams for instruction fetch and execution are presented in [Figure 4-7](#).

**Figure 4-7. Program Memory Map**



### 4.8.2 SRAM Data Memory

[Figure 4-8 on page 21](#) shows how the ATtiny44V SRAM Memory is organized.

The lower 160 data memory locations address both the register File, the I/O memory and the internal data SRAM. The first 32 locations address the register File, the next 64 locations the standard I/O memory, and the last 256 locations address the internal data SRAM.

The five different addressing modes for the data memory cover: Direct, indirect with displacement, indirect, indirect with pre-decrement, and indirect with post-increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

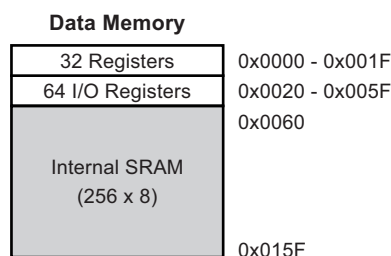
The direct addressing reaches the entire data space.

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

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

The 32 general purpose working registers, 64 I/O registers, and the 256 bytes of internal data SRAM in the ATtiny44V are all accessible through all these addressing modes. The register File is described in [Section 4.7.5 “General Purpose Register File” on page 17](#).

**Figure 4-8. Data Memory Map**

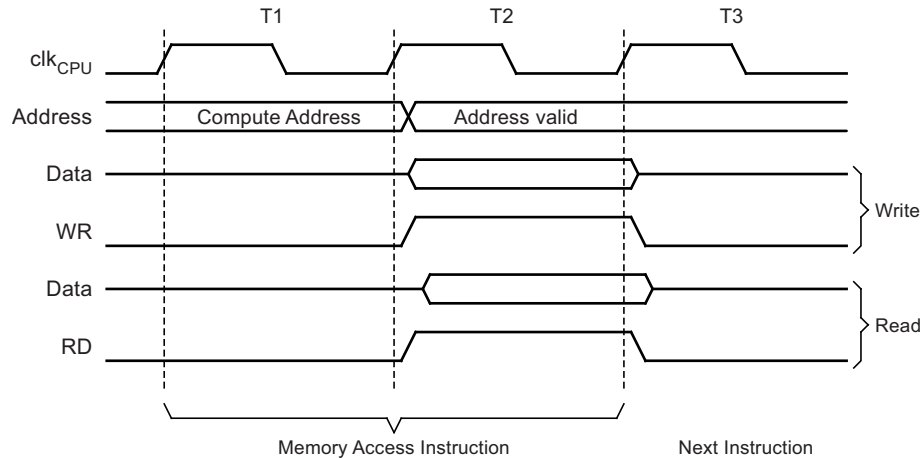




### 4.8.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two  $\text{clk}_{\text{CPU}}$  cycles as described in Figure 4-9.

Figure 4-9. On-chip Data SRAM Access Cycles



### 4.8.3 EEPROM Data Memory

The Atmel® ATtiny44V contains 256 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register. For a detailed description of Serial data downloading to the EEPROM, see Section 4.23.6 “Serial Downloading” on page 145.

#### 4.8.3.1 EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access times for the EEPROM are given in Table 4-2 on page 26. 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,  $V_{\text{CC}}$  is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See Section 4.8.3.6 “Preventing EEPROM Corruption” on page 24 for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. See Section 4.8.3.2 “Atomic Byte Programming” on page 22 and Section 4.8.3.3 “Split Byte Programming” on page 22 for details on this.

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.

#### 4.8.3.2 Atomic Byte Programming

Using atomic byte programming is the simplest mode. When writing a byte to the EEPROM, the user must write the address into the EEARL register and data into EEDR Register. If the EEPm bits are zero, writing EEPE (within four cycles after EEMPE is written) will trigger the erase/write operation. Both the erase and write cycle are done in one operation and the total programming time is given in Table 4-2 on page 26. The EEPE bit remains set until the erase and write operations are completed. While the device is busy with programming, it is not possible to do any other EEPROM operations.

#### 4.8.3.3 Split Byte Programming

It is possible to split the erase and write cycle in two different operations. This may be useful if the system requires short access time for some limited period of time (typically if the power supply voltage falls). In order to take advantage of this method, it is required that the locations to be written have been erased before the write operation. But since the erase and write operations are split, it is possible to do the erase operations when the system allows doing time-critical operations (typically after Power-up).

#### 4.8.3.4 Erase

To erase a byte, the address must be written to EEAR. If the EEP Mn bits are 0b01, writing the EEPE (within four cycles after EEMPE is written) will trigger the erase operation only (programming time is given in [Table 4-2 on page 26](#)). The EEPE bit remains set until the erase operation completes. While the device is busy programming, it is not possible to do any other EEPROM operations.

#### 4.8.3.5 Write

To write a location, the user must write the address into EEAR and the data into EEDR. If the EEP Mn bits are 0b10, writing the EEPE (within four cycles after EEMPE is written) will trigger the write operation only (programming time is given in [Table 4-2 on page 26](#)). The EEPE bit remains set until the write operation completes. If the location to be written has not been erased before write, the data that is stored must be considered as lost. While the device is busy with programming, it is not possible to do any other EEPROM operations.

The calibrated Oscillator is used to time the EEPROM accesses. Make sure the Oscillator frequency is within the requirements described in [Section 4.9.10.1 “Oscillator Calibration Register – OSCCAL” on page 33](#).

The following code examples show one assembly and one C function for erase, write, or atomic write of the EEPROM. The examples assume that interrupts are controlled (e.g., by disabling interrupts globally) so that no interrupts will occur during execution of these functions.

Assembly Code Example
<pre>EEPROM_write:     ; Wait for completion of previous write     sbic      EECR, EEPE     rjmp     EEPROM_write     ; Set Programming mode     ldi      r16, (0&lt;&lt;EEP M1) (0&lt;&lt;EEP M0)     out      EECR, r16     ; Set up address (r17) in address register     out      EEARL, r17     ; Write data (r16) to data register     out      EEDR, r16     ; Write logical one to EEMPE     sbi      EECR, EEMPE     ; Start eeprom write by setting EEPE     sbi      EECR, EEPE     ret</pre>
C Code Example
<pre>void EEPROM_write(unsigned char ucAddress, unsigned char ucData) {     /* Wait for completion of previous write */     while(EECR &amp; (1&lt;&lt;EEPE))         ;     /* Set Programming mode */     EECR = (0&lt;&lt;EEP M1) (0&gt;&gt;EEP M0)     /* Set up address and data registers */     EEARL = ucAddress;     EEDR = ucData;     /* Write logical one to EEMPE */     EECR  = (1&lt;&lt;EEMPE);     /* Start eeprom write by setting EEPE */     EECR  = (1&lt;&lt;EEPE); }</pre>

Note: The code examples are only valid for the Atmel® ATtiny44V, using 8-bit addressing mode.

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example
<pre> EEPROM_read:     ; Wait for completion of previous write     sbic      EECR,EEPE     rjmp     EEPROM_read     ; Set up address (r17) in address register     out      EEARL, r17     ; Start eeprom read by writing EERE     sbi      EECR,EERE     ; Read data from data register     in       r16,EEDR     ret         </pre>
C Code Example
<pre> unsigned char EEPROM_read(unsigned char ucAddress) {     /* Wait for completion of previous write */     while((EECR &amp; (1&lt;&lt;EEPE))         ;     /* Set up address register */     EEARL = ucAddress;     /* Start eeprom read by writing EERE */     EECR  = (1&lt;&lt;EERE);     /* Return data from data register */     return EEDR; }         </pre>

Note: 1. The code examples are only valid for the Atmel ATtiny44V, using 8-bit addressing mode.

#### 4.8.3.6 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.

An EEPROM data corruption can be caused by two situations when the voltage is too low. 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® 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 needed detection level, 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 will be completed provided that the power supply voltage is sufficient.

#### 4.8.4 I/O Memory

The I/O space definition of the Atmel ATtiny44V is shown in [Section 9.1 “Register Summary” on page 183](#).

All ATtiny44V I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. See the instruction set section 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.

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

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

The I/O and Peripherals Control Registers are explained in later sections.

#### 4.8.4.1 General Purpose I/O Registers

The ATtiny44V 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.

#### 4.8.5 Register Description

##### 4.8.5.1 EEARH – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1F (0x3F)	-	-	-	-	-	-	-	<b>EEAR8</b>	EEARH
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	X	

- **Bits 7..1 – Res: Reserved Bits**

These bits are reserved bits in the Atmel® ATtiny44V and will always read as zero.

- **Bit 0 – EEAR8: EEPROM Address**

This bit is reserved bit and will always read as zero. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

##### 4.8.5.2 EEARL – EEPROM Address Register

Bit	7	6	5	4	3	2	1	0	
0x1E (0x3E)	<b>EEAR7</b>	<b>EEAR6</b>	<b>EEAR5</b>	<b>EEAR4</b>	<b>EEAR3</b>	<b>EEAR2</b>	<b>EEAR1</b>	<b>EEAR0</b>	EEARL
Read/Write	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 7..0 – EEAR7..0: EEPROM Address**

The EEPROM address register – EEARL – specifies the EEPROM address. The EEPROM data bytes are addressed linearly between 0 and 256. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

##### 4.8.5.3 EEDR – EEPROM Data Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	<b>EEDR7</b>	<b>EEDR6</b>	<b>EEDR5</b>	<b>EEDR4</b>	<b>EEDR3</b>	<b>EEDR2</b>	<b>EEDR1</b>	<b>EEDR0</b>	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.

##### 4.8.5.4 EECR – EEPROM Control Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	-	-	<b>EEPROM1</b>	<b>EEPROM0</b>	<b>EERIE</b>	<b>EEMPE</b>	<b>EEPE</b>	<b>EERE</b>	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	

- **Bit 7 – Res: Reserved Bit**

This bit is reserved for future use and will always read as 0 in Atmel ATtiny44V. For compatibility with future Atmel AVR® devices, always write this bit to zero. After reading, mask out this bit.