Introduction to AVR ATmega32

By: Yehia M. Abu Eita

Outlines

- Features
- Pin configurations
- Block diagram
- AVR CPU core
- ATmega32 Memories
- System clock
- Accessing I/O registers

Features

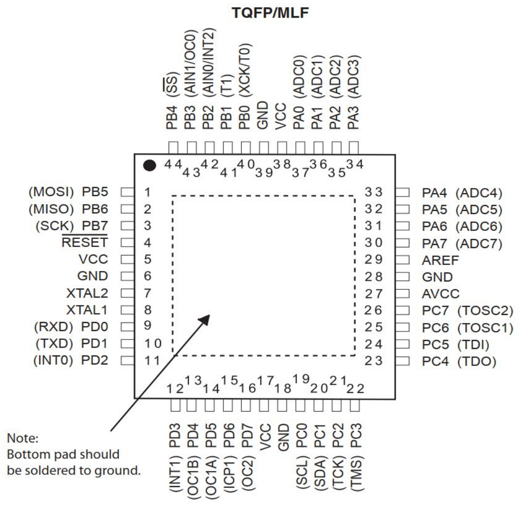
- **Advanced RISC Architecture:**
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - **32 × 8 General Purpose** Working Registers
  - Fully Static Operation
  - **Up to 16 MIPS** Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- **High Endurance Non-volatile Memory segments:**
  - **32K bytes** of In-System Self-programmable **Flash** program memory
  - **1024 Bytes EEPROM**
  - **2K bytes** 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
  - Programming Lock for Software Security

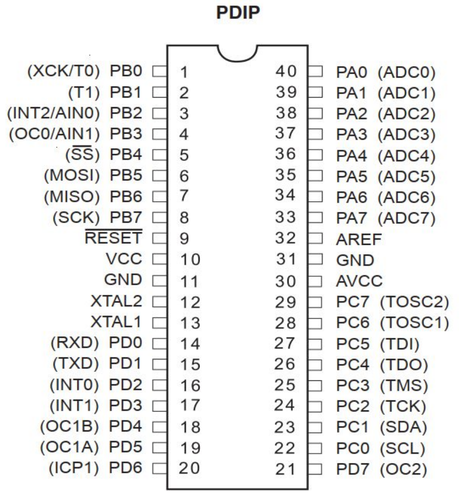
Features

- **Peripheral Features:**
  - **Two 8-bit Timer/Counters** with Separate Prescalers and Compare Modes
  - **One 16-bit Timer/Counter** with Separate Prescaler, Compare Mode, and Capture Mode
  - **Four PWM** Channels
  - **8-channel, 10-bit ADC**
    - 8 Single-ended Channels
    - 7 Differential Channels in TQFP Package Only
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented **Two-wire Serial Interface**
  - Programmable **Serial USART**
  - Master/Slave **SPI Serial Interface**
  - Programmable **Watchdog Timer** with Separate On-chip Oscillator
  - On-chip **Analog Comparator**

Pin configurations

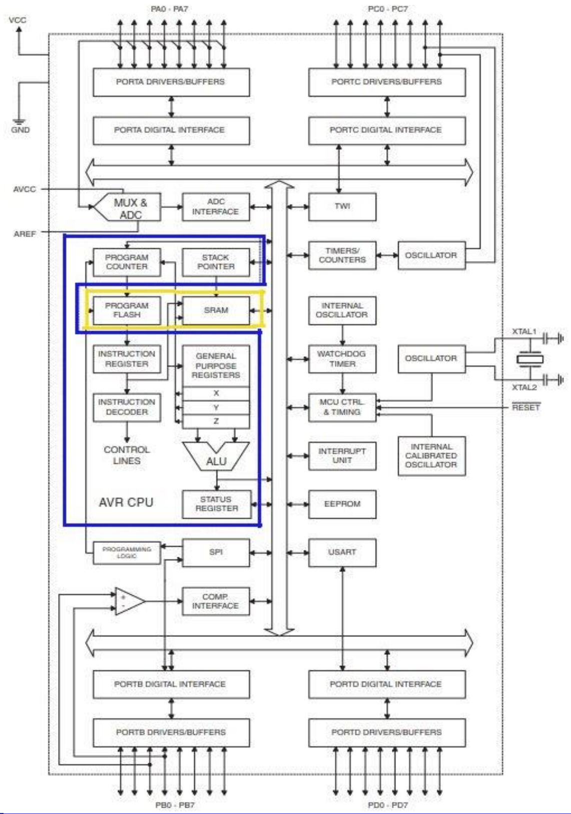
- 32 Programmable I/O Lines
- **Packaging**:
  - 40-pin PDIP (Plastic Dual Inline Package).
  - 44-lead TQFP(Thin Quad Flat Pack)
  - 44-pad QFN/MLF (Quad Flat No-lead package)/(Micro Lead Frame)





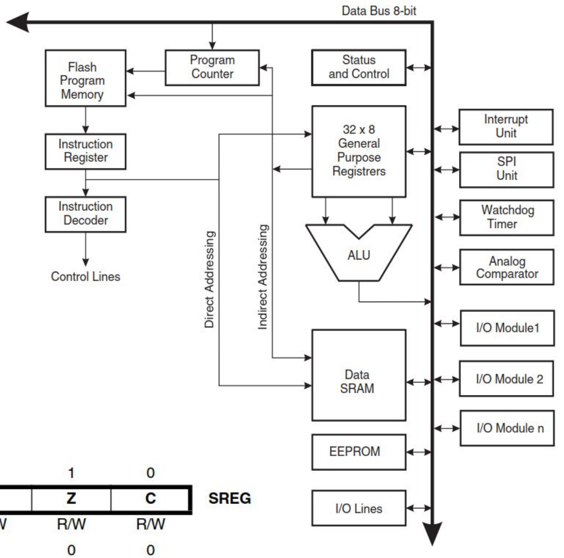
Block diagram

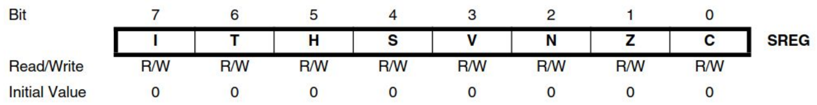
- 32K bytes of Flash program memory.
- 2K bytes Internal SRAM.
- 32 GPIO pins.
- Three timers.
- Four PWM Channels.
- Byte-oriented Two-wire Serial Interface.
- Programmable Serial USART.
- Master/Slave SPI Serial Interface.
- Programmable Watchdog Timer.
- 8-channel, 10-bit ADC.



AVR CPU core

- Harvard architecture.
- Instruction are executed with a **single level pipelining**.
- **Single-cycle ALU** operation
- **The Status Register (SREG)**:
  - Contains information about the result of the most recently executed arithmetic instruction.

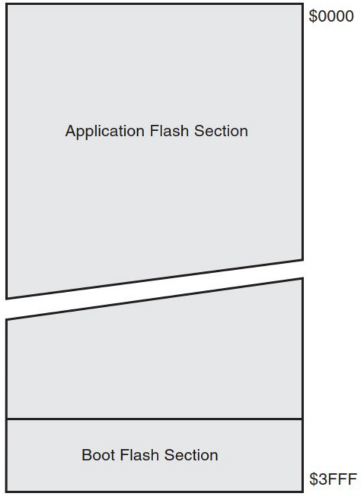




Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

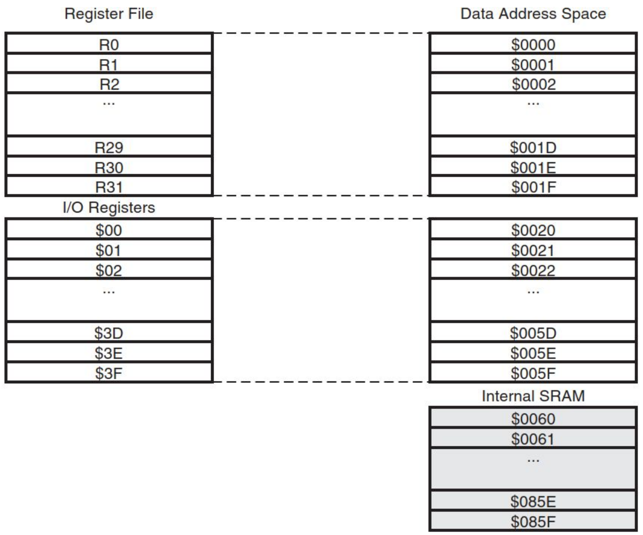
ATmega32 Memories

- **In-System Programmable Flash Program Memory:**
  - **Size**: **32K bytes**
  - AVR **instructions are 16 or 32 bits** wide
  - Flash is organized as **16K × 16 bits**
  - Flash Program memory space is divided into two sections
  - Program Counter **(PC) is 14 bits** wide



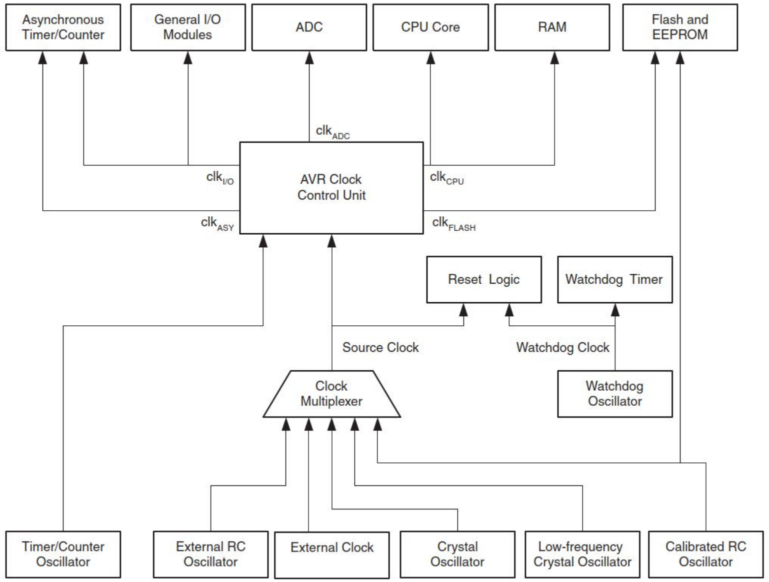
ATmega32 Memories

- **SRAM Data Memory:**
  - **Size**: **2K bytes**
  - **Sections**:
    - **GPR register file, 32 Register**
    - **I/O registers, 64 Register**
    - **Internal SRAM, 2048 location**
  - **Addressing modes**:
    - **Direct**
    - **Indirect with Displacement**
    - **Indirect**
    - **Indirect with Pre-decrement**
    - **Indirect with Post-increment**



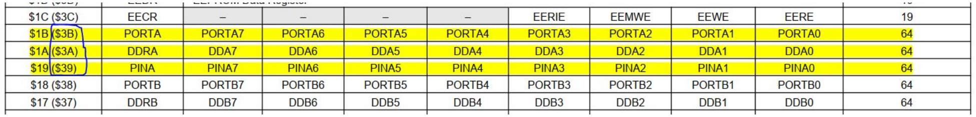
System clock

- **Device Clocking Option**:
  - External Crystal/Ceramic Resonator
  - External Low-frequency Crystal
  - External RC Oscillator
  - Calibrated Internal RC Oscillator
  - External Clock
  - The default clock source is 1MHz Internal RC Oscillator



Accessing I/O registers

- **Navigate** to the **register summary** section in datasheet.



$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	19
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	64

- Suppose we need to access **PORTA register:**
  - **Get its address**, 0x3B.
  - Declare a **pointer to point** to that **address**
    - `volatile uint8_t *PORTA = (volatile uint8_t*)(0x3B);`
  - The **0x3B is an integer,** you **must convert it to an address**, `(volatile uint8_t*)(0x3B)`
  - You can use PORTA now to write data into the register with address 0x3B, `*PORTA = 0x10;`
  - Or you can use, `*((volatile uint8_t*)(0x3B)) = 0x10;`, which will save memory.
  - Then you can define it as a macro, `#define PORTA (*((volatile uint8_t*)(0x3B))`

```
#define PA_DATA (*((volatile uint8_t*)(0x3B)))
#define PA_CTRL (*((volatile uint8_t*)(0x3A)))
#define PA_STAT (*((volatile uint8_t*)(0x39)))
```

```
int main()
{
        PA_CTRL = 0x55;
        while(1)
        {
                PA_DATA = 0x10;
        }
}
```

Summary

- You are now familiar with AVR ATmega32
- You have learned its architecture, memory organization, and system clock.
- The most important thing to take care of is hardware I/O registers accessing .