Chapter 2 AVR Microcontrollers

2.1 Introduction to AVR microcontroller: Overview of AVR family, ATmega32 pin configuration & function of each pin
2.2 AVR Microcontroller architecture: Internal Memory, Registers, Digital I/O ports, Serial I/O
2.3 AVR Microcontroller architecture: Timers/Counters, Analog Comparator, ADC,, Interrupts.
2.4 Features of Arduino specific AVR microcontroller ATmega 168/328

2.1 Introduction to AVR microcontroller: Overview of AVR family, ATmega32 pin configuration & function of each pin

Overview of AVR family

- AVR was developed in the year 1996 by Atmel Corporation.
- The architecture of AVR was developed by Alf-Egil Bogen and Vegard Wollan.
- AVR derives its name from its developers and stands for Alf-Egil Bogen Vegard Wollan RISC microcontroller, also known as Advanced Virtual RISC.
- The AT90S8515 was the first microcontroller which was based on AVR architecture
- However the first microcontroller to hit the commercial market was AT90S1200 in the year 1997.

Features

- RISC architecture
- 131 instructions
- 32 8-bit general-purpose registers
- Up to 20 MHz clock rate (20 MIPS operation)
- On-board memory
  —Flash program memory (up to 256K)
  —EEPROM (up to 4K)
  —Internal SRAM (up to 32K)
- Operating voltage
  —VCC = 1.8 to 5.5V DC
- Inbuilt ADC
- Inbuilt Timer
- Parallel Ports, PortA, PortB, PortC and PortD for ATmega
- Serial ports: USART, I2C and SPI

Classification of AVR family

- Classic
- Tiny
- Mega
- Special Purpose

1. **Classic**
   This is the first series produced of AVR in 1997 which has been replaced by newer AVR chips.
   e.g. AT90S2313, AT90S4433

Part Num.	Code ROM	Data RAM	Data EEPROM	I/O pins	ADC	Timers	Pin numbers & Package
AT90S2313	2K	128	128	15	0	2	SOIC20, PDIP20
AT90S2323	2K	128	128	3	0	1	SOIC8, PDIP8
AT90S4433	4K	128	256	20	6	2	TQFP32, PDIP28

Notes:
1. All ROM, RAM, and EEPROM memories are in bytes.
2. Data RAM (general-purpose RAM) is the amount of RAM available for data manipulation (scratch pad) in addition to the register space.

2. **Mega AVR**
   (**ATmegaxxx**), e.g. ATmega32, **ATmega328p**.
   A powerful microcontroller which can be used in different designs.
   4–256 kB program memory.
   28–100-pin package.
   Extended (Rich) instruction set.
   Extensive peripheral set.

Table 1-4: Some Members of the Mega Family

Part Num	Code ROM	Data RAM	Data EEPROM	I/O pins pins	ADC	Timers	Pin numbers & Package
ATmega8	8K	1K	0.5K	23	8	3	TQFP32,PDIP28
ATmega16	16K	1K	0.5K	32	8	3	TQFP44,PDIP40
ATmega32	32K	2K	1K	32	8	3	TQFP44,PDIP40
ATmega64	64K	4K	2K	54	8	4	TQFP64,MLF64
ATmega1280	128K	8K	4K	86	16	6	TQFP100,CBGA

3. **Tiny AVR (ATtinyXXXX)**
   This group has less instructions and **smaller packages** in comparison to mega families.
   • You can design systems with **low costs and power consumptions** using the Tiny AVRs .
   •Program memory: IK to 8K bytes
   •Package 8 to 28 pins
   •Limited peripheral set
   •Limited instruction set: The instruction sets are limited. For example, some of them do not have the multiply instruction.
   •e.g. ATtiny13, ATtiny25

Table 1-5: Some Members of the Tiny Family							
Part Num	Code ROM	Data RAM	Data EEPROM	I/O pins pins	ADC	Timers	Pin numbers & Package
ATtiny13	1K	64	64	6	4	1	SOIC8,PDIP8
ATtiny25	2K	128	128	6	4	2	SOIC8,PDIP8
ATtiny44	4K	256	256	12	8	2	SOIC14,PDIP14
ATtiny84	8K	512	512	12	8	2	SOIC14,PDIP14

4. **Application Specific**

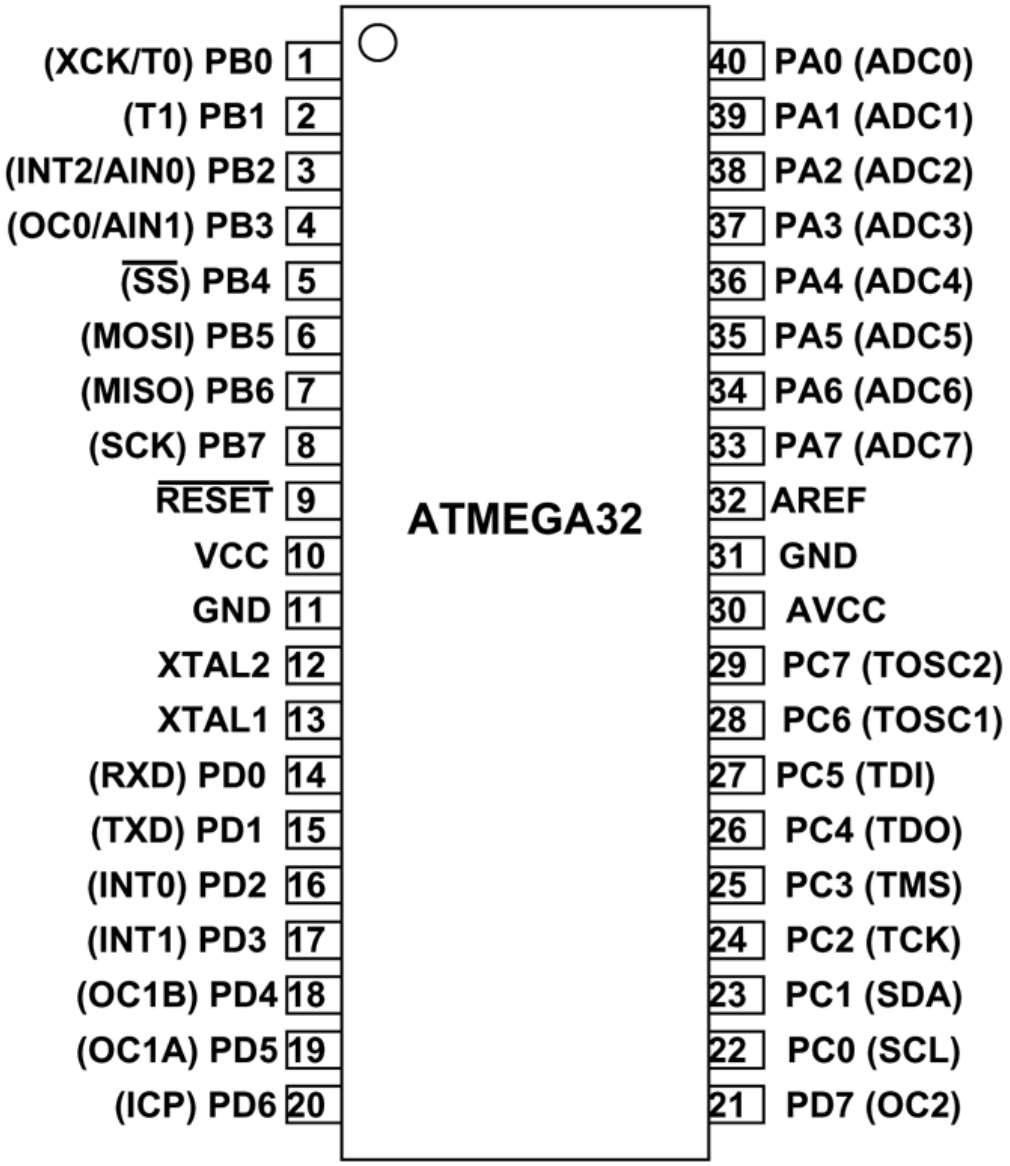
   The ICs of this group can be considered as a subset of other groups, but their special capabilities are made for designing specific applications.

   MegaAVRs with special features not found on the other members of the AVR family, such as LCD controller, USB controller, advanced PWM, CAN, etc

   In addition to all these types, another advanced type is 32 bit AVR

Table 1-6: Some Members of the Special purpose Family							
Part Num	Code ROM	Data RAM	Data EEPROM	Max I/O pins	Special Capabilities	Timers	Pin numbers & Package
AT90CAN128	128K	4K	4K	53	CAN	4	LQFP64
AT90USB1287	128K	8K	4K	48	USB Host	4	TQFP64
AT90PWM216	16K	1K	0.5K	19	Advanced PWM	2	SOIC24
ATmega169	16K	1K	0.5K	54	LCD	3	TQFP64,MLF64

2.1 ATmega32 pin configuration & function of each pin



Atmega32 has got 40 pins. Two for Power (pin no.10: +5v, pin no. 11: ground), two for oscillator (pin 12, 13), one for reset (pin 9), three for providing necessary power and reference voltage to its internal ADC, and 32 (4×8) I/O pins.

VCC: Digital supply voltage.

GND: Ground.

Port A pins (PA7-PA0): Port A is an 8-bit bi-directional I/O port with individually selectable internal pull-up resistors. The PORT A output buffers can sink 20mA and thus drive LED displays directly. It should be noted that the supply voltage for Port A comes from the Avcc pin.
Port A also serves as the analog inputs to the A/D Converter. Each pin of PortA to be used as input for the ADC, ie ADC0 to ADC7

Port B pins (PB7-PB0): Port B is an 8-bit bi-directional I/O port with individually selectable internal pull-up resistors. The PORT B output buffers can sink 20mA and thus drive LED

displays directly.

Port B has alternate function as shown below:

Port Pin	Alternate Functions
PB0	T0 (Timer/Counter 0 External Counter Input)
PB1	T1 (Timer/Counter 1 External Counter Input)
PB2	AIN0 (Analog Comparator Positive Input)
PB3	AIN1 (Analog Comparator Negative Input)
PB4	SS (SPI Slave Select Input)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

- Bit 7 - SCK:
  Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the settings of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.

- Bit 6 - MISO:
  Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit.

- Bit 5 - MOSI:
  SPI master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit.

- Bit 4 - SS:
  Slave port select input. When the SPI is enabled as a slave, this pin is configured as

an input regardless of the setting of DDB4. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 bit.

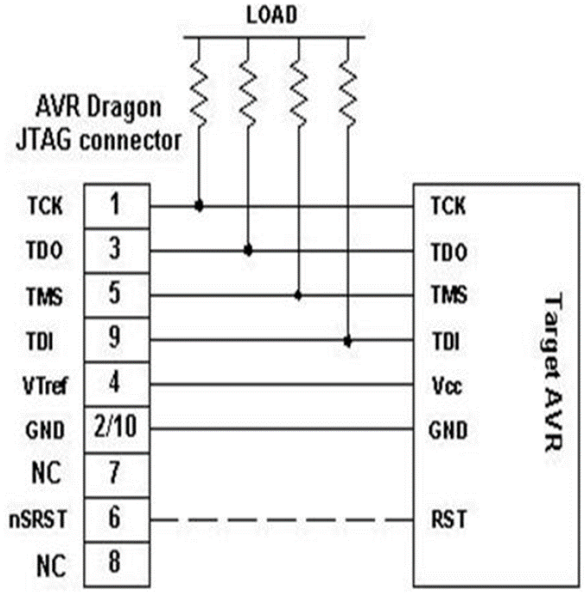
- Bit 3 - AIN1:
  Analog Comparator Negative Input. When configured as an input and with the internal MOS pull up resistor switched off, this pin also serves as the negative input of the On-chip analog comparator.

- Bit 2 - AIN0:
  Analog Comparator Positive Input. When configured as an input and with the internal MOS pull up resistor switched off, this pin also serves as the positive input of the On-chip analog comparator.

- Bit 1 - T1:
  Timer/Counter1 counter source.

- Bit 0 - T0:
  Timer/Counter0 counter source.

Port C (PC7-PC0): Port C is an 8-bit bi-directional I/O port with individually selectable internal pull-up resistors. The PORT C output buffers can sink 20mA and thus drive LED displays directly.
If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. The TD0 pin is tri-stated unless TAP states that shift out data are entered. Port C also serves the functions of the JTAG interface as shown in figure below..



Port C has alternate function as shown below:

Port Pin	Alternate Function

PC0	SCL (2-wire Serial Bus Clock Line)
PC1	SDA (2-wire Serial Bus Data Input/Output Line)
PC2	TCK (JTAG Test Clock)
PC3	TMS (JTAG Test Mode Select)
PC4	TDO (JTAG Test Data Out)
PC5	TDI (JTAG Test Data In)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC7	TOSC2 (Timer Oscillator Pin 2)

- Bit 7 - TOSC2

  Timer Oscillator pin 2: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter2, pin PC7 is disconnected from the port, and becomes the inverting output of the oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- Bit 6 - TOSC1

  Timer Oscillator pin 1: When the AS2 bit in ASSR is set (one) to enable asynchronous clocking of Timer/Counter1, pin PC6 is disconnected from the port, and becomes the input of the inverting oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and the pin can not be used as an I/O pin.

- Bit 1 - SDA

  When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, pin PC1 is disconnected from the port and becomes the Serial Data I/O pin for the 2-wire Serial Interface. In this mode, there is a spike filter on the pin to capture spikes shorter than 50ns on the input signal, and the pin is driven by an open collector driver with slew rate limitation.

- Bit 0 - SCL

  When the TWEN bit in TWCR is set (one) to enable the 2-wire Serial Interface, PC0 is disconnected from the port and becomes the Serial Clock I/O pin for the 2-wire Serial Interface. In this mode, there is a spike filter on the pin to capture spikes shorter than 50ns on the input signal.

Port D (PD7-PD0): Port D is an 8-bit bi-directional I/O port with individually selectable internal pull-up resistors. The PORT D output buffers can sink 20mA and thus drive LED displays directly.

Port D has alternate function as shown below:

Port Pin	Alternate Functions
PD0	RXD (UART Input Pin)
PD1	TXD (UART Output Pin)
PD2	INT0 (External Interrupt 0 Input)
PD3	INT1 (External Interrupt 1 Input)
PD4	OC1B (Timer/Counter1 Output CompareB Match Output)
PD5	OC1A (Timer/Counter1 Output CompareA Match Output)
PD6	ICP (Timer/Counter1 Input Capture Pin)
PD7	OC2 (Timer/Counter2 Output Compare Match Output)

- Bit 7 - OC2:
  The PD7 pin can serve as an external output for the Timer/Counter2 output compare. The pin has to be configured as an output to serve this function. This pin is also the output pin for the PWM mode timer function.

- Bit 6 - ICP:
  The PD6 pin can act as an input capture pin for Timer/Counter1. The pin has to be configured as an input to serve this function.

- Bit 5 - OC1A:
  The PD5 pin can serve as an external output for the Timer/Counter1 output compareA. The pin has to be configured as an output to serve this function.

- Bit 4 - OC1B:
  The PD4 pin can serve as an external output for the Timer/Counter1 output compareB. The pin has to be configured as an output to serve this function.

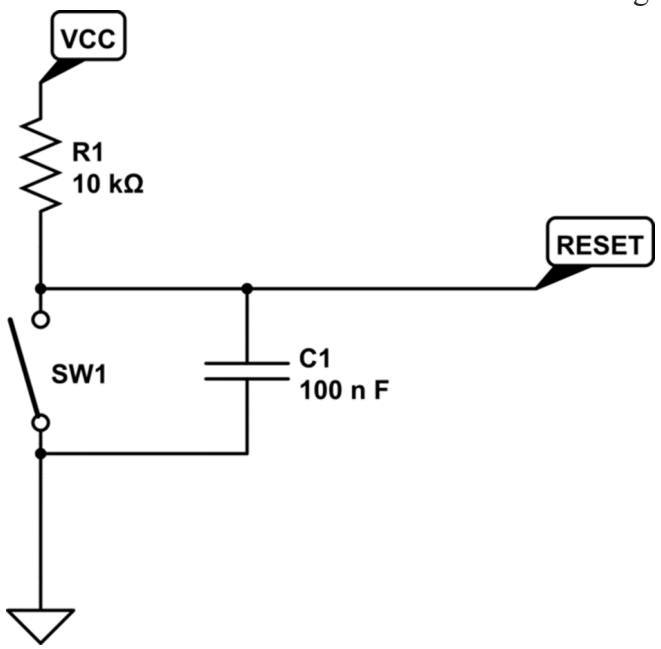
- Bit 3 - INT1:
  The PD3 pin can serve as an external interrupt source to the MCU.

- Bit 2 - INT0:
  The PD2 pin can serve as an external interrupt source to the MCU.

- Bit 1 - TXD:
  When the UART transmitter is enabled, this pin is configured as the output data pin for the UART regardless of the value of DDRD1.

- Bit 0 - RXD:
  When the UART receiver is enabled, this pin is configured as the input data pin for the UART regardless of the value of DDRD0.

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. Shorter pulses are not guaranteed to generate a reset.
The Power ON and Manual reset circuit of ATmega32 is given below



Power On reset: As Vcc is switched on, the signal at reset pin will be logic 0, as the capacitor takes some time to get charged. As soon as the capacitor is fully charged, it blocks dc current, hence the Reset pin will be at logic 1. Hence the reset signal is activated (logic 0) for some time, till the capacitor is charged, which resets the microcontroller.
Manual Reset : When the switch SW1 is pressed, Reset pin gets ground path and will be at logic 0, hence microcontroller will get reset.

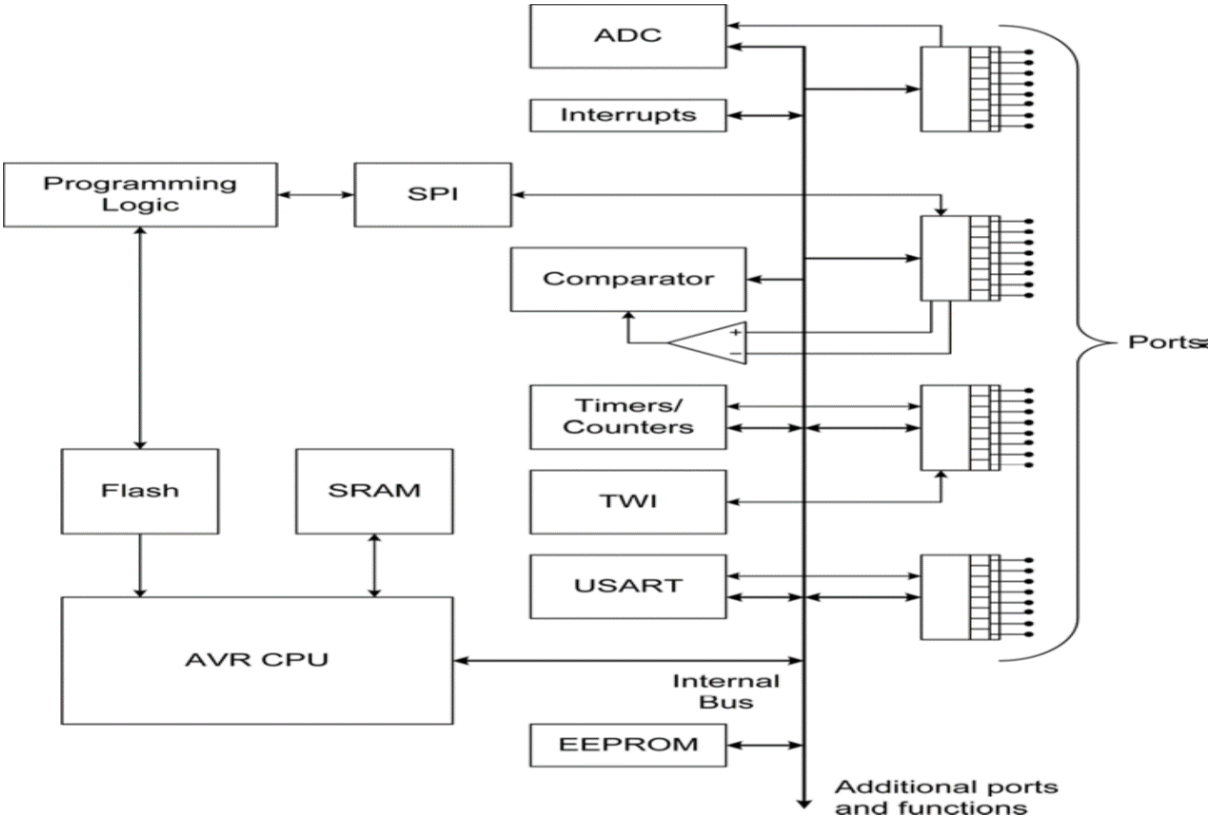
XTAL1: Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2: Output from the inverting Oscillator amplifier.

AVCC: AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to VCC, even if the ADC is not used. If the ADC is used, it should be connected to VCC through a low-pass filter.

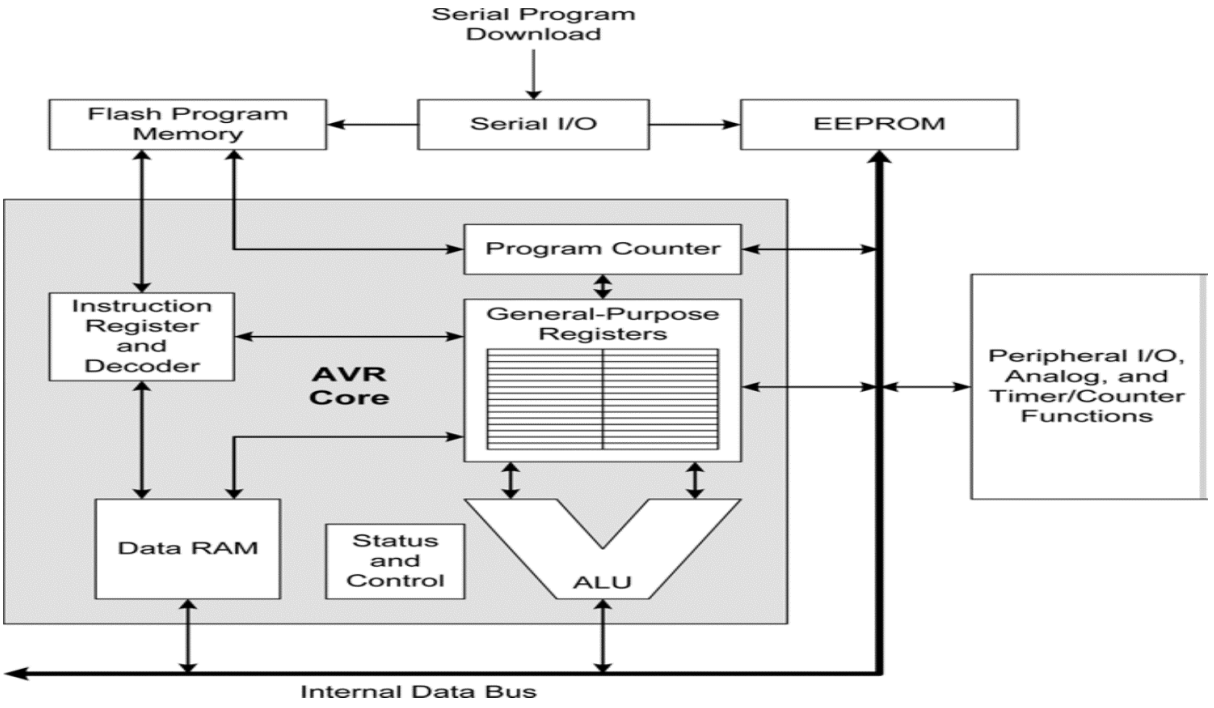
AREF: AREF is the analog reference pin for the A/D Converter.

2.2 AVR Microcontroller architecture: Internal Memory, Registers, Digital I/O ports, Serial I/O

The generalized architecture of AVR is given below



The architecture of the main block, ie AVR CPU is as follows



Architecture of AVR can be divided to various blocks such as

1. Internal Memory and Registers
2. Digital I/O ports
3. Serial I/O
4. Timer /counters
5. ADC
6. Analog Comparator
7. Interrupts etc.

These blocks are covered below individually

1. Internal Memory and registers

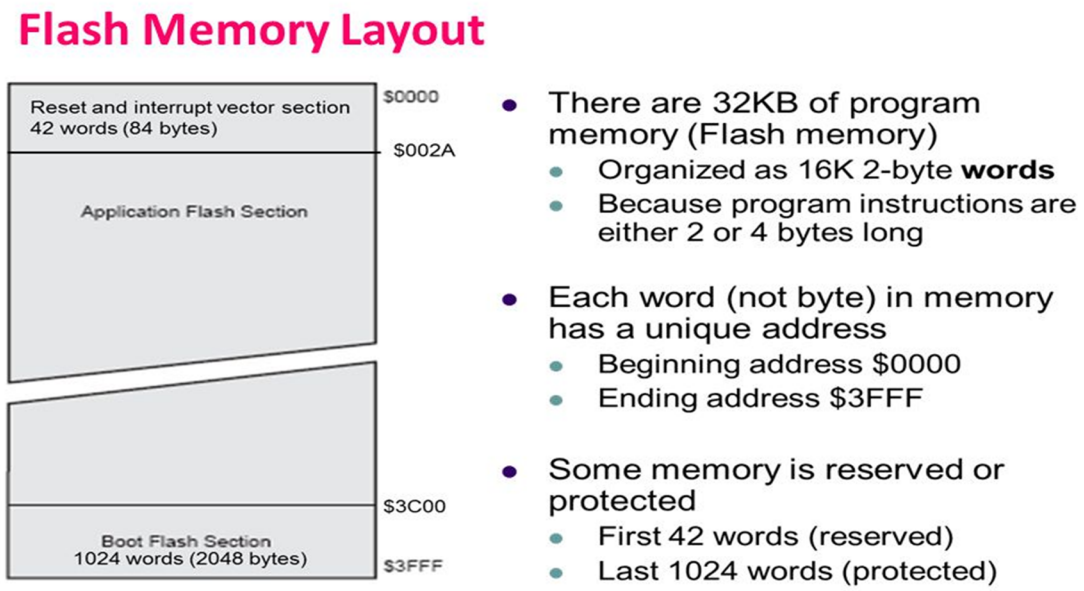
Different AVR microcontrollers have different capacities of memory. The internal memory organization of ATmega32 is discussed below.
There are three types of memory in the AVR architecture:

- Program Memory
- Data Memory
- EEPROM

Program Memory

Program Memory, also referred to as *flash*, is where program code is stored. Depending on settings in the fuse bits, the entire flash memory can be used as application storage or a portion can be code-blocked off for a bootloader program.

Flash memory is *non-volatile* meaning its data does not go away when the microcontroller loses power. Flash cannot be accessed as quickly as Data Memory and individual bytes cannot be modified at a time so it is not a good place to store variables which constantly change. However, in addition to storing your application code, it is a good place to put constants like lookup tables or strings. The flash memory layout of ATmega32 is given below



Data Memory

Data Memory is composed of four parts:

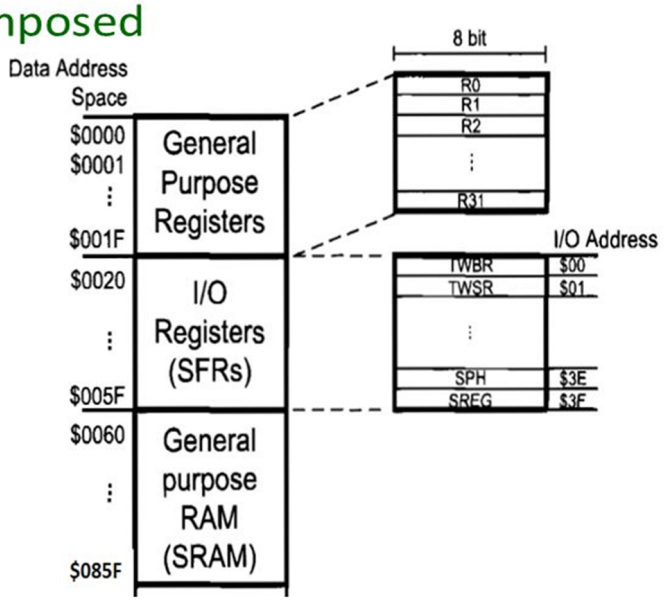
- Register File
- I/O Registers
- Extended I/O Registers (not present in all ICs)
- Internal SRAM

Unlike flash, Data Memory is not composed of a single memory type. Rather, it is a contiguous mapping of addresses across multiple memory types.

The breakdown of data memory is shown below. The location and size of the general purpose working registers, I/O registers, and extended I/O registers are the same across all chips (even if some of the extended I/O features are not implemented). However, the size of internal SRAM varies between different models.
ATmega32 data memory organization is given below. Note that it doesn't have extended I/O section. Hence there are only three divisions.

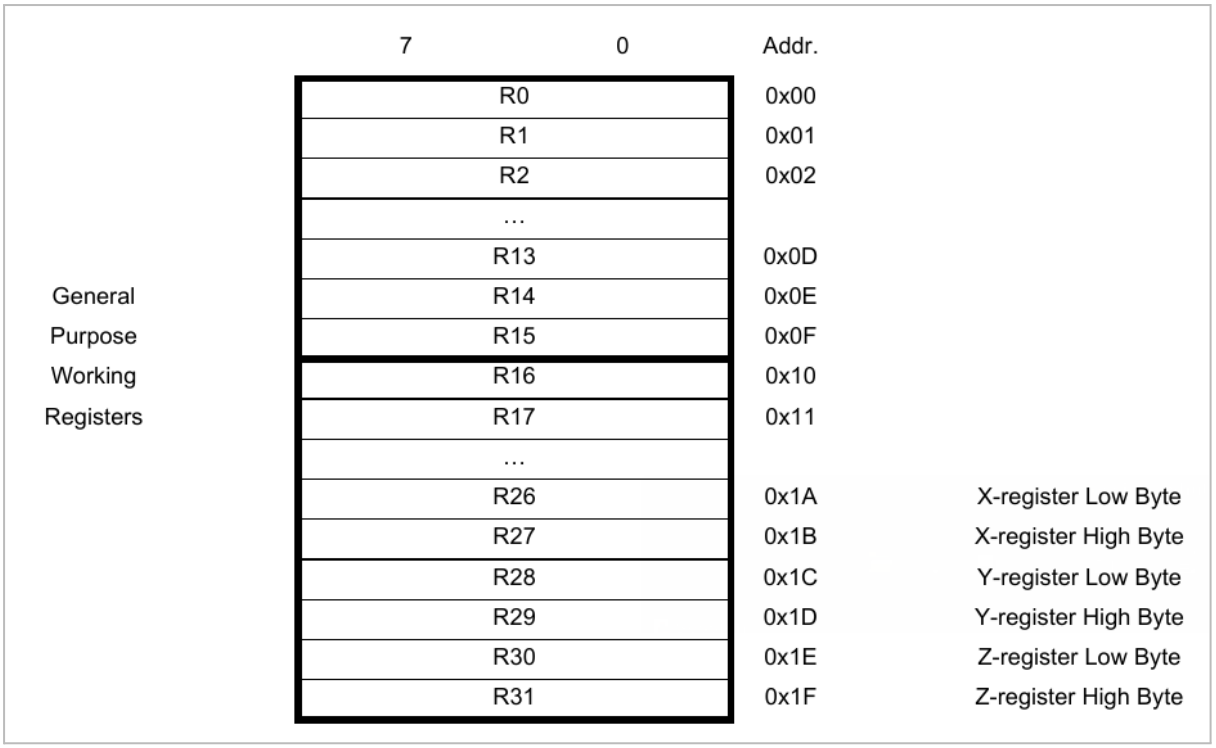
The data memory is composed of three parts:

- **GPRs (general purpose registers),**
- **Special Function Registers (SFRs), and**
- **Internal data SRAM.**

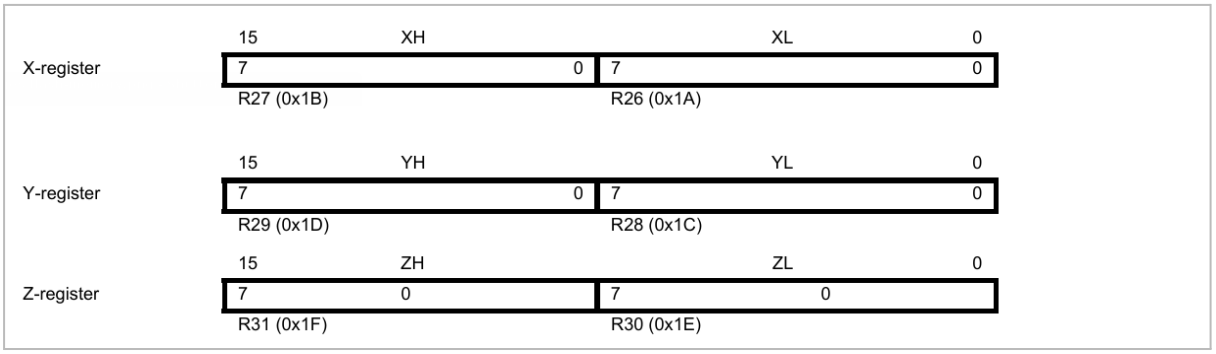


Register File or general purpose registers

The register file begins at the start of Data Memory and contains 32 general purpose working registers that are directly connected to the arithmetic logic unit (ALU). The general purpose working registers are used as operands for most instructions. The general purpose working registers are extremely important and useful. Any data manipulation in the AVR requires using one of these registers.



Registers R26:R27, R28:R29, and R30:R31 are given the names X, Y, and Z respectively. They are often referred to as the X, Y and Z *pointers* as they can be used to store or retrieve data from memory by placing an address in them.



I/O Registers

The I/O Registers are locations in Data Memory which control certain microcontroller functions. All of the data direction registers and ports (e.g. DDRB and PORTB) are located in the I/O Register portion of Data Memory. Because they are needed so often, there are special instructions which allow quick access and modification of the I/O Registers.

The I/O Registers are non-volatile. They will initialize to default values that may or may not need to be changed by your program.

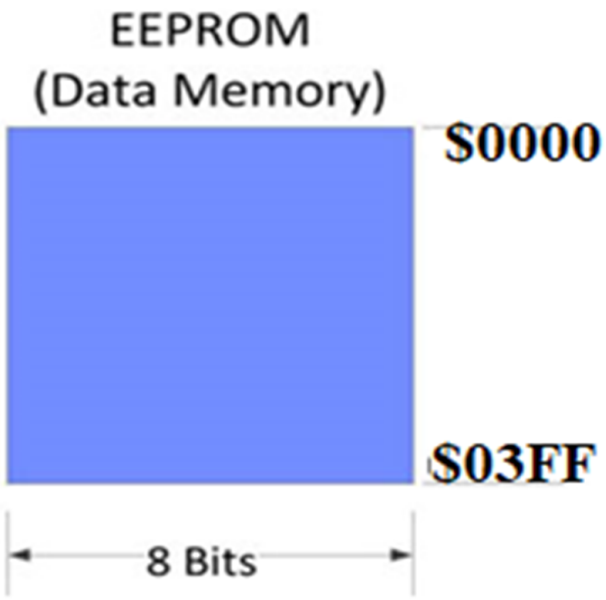
Internal SRAM

Static Random Access Memory - referred to as SRAM - is the last part of Data Memory. SRAM is volatile so its contents cannot be guaranteed at startup. However, unlike flash, single bytes can be written to SRAM and its access time is quicker, making it much better for storing temporary values.

SRAM is a very convenient runtime memory type. It is great when you have more variables than can fit in the general purpose working registers and need a place to put them while they are not being used.

EEPROM

EEPROM is another non-volatile storage location in the AVR architecture and can be called as permanent data memory. Unlike flash, individual bytes may be written to it, although its access time is much slower. EEPROM is best served for configuration settings, i.e. things that need to be available at startup but may be changed at runtime and used the next time the chip starts up.



2. Digital I/O ports

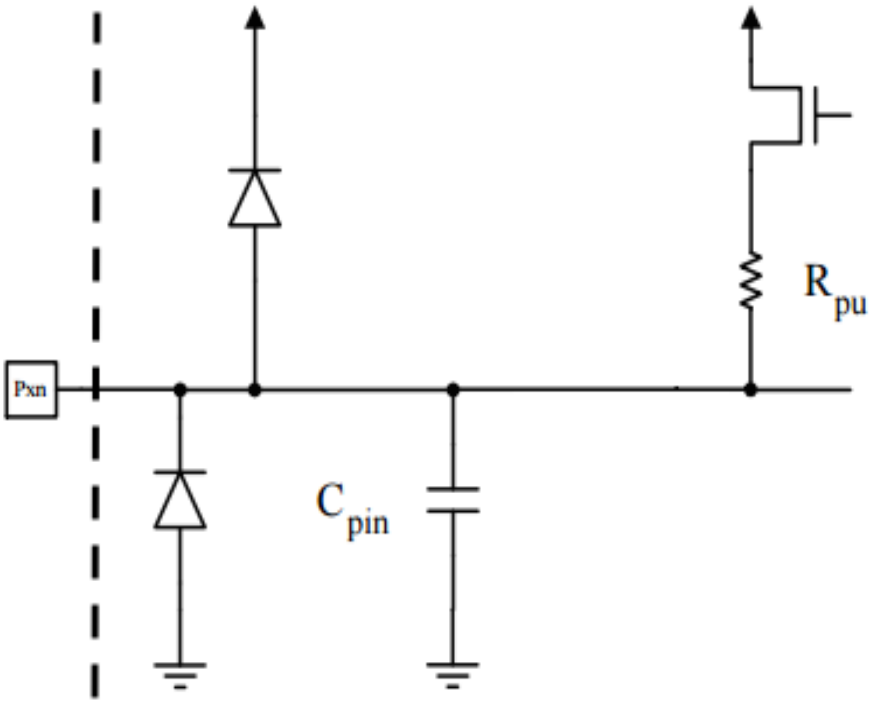
In the AVR microcontroller family, there are many ports available for I/O operations, depending on which family microcontroller you choose as per the table given below. For the ATmega32 40-pin chip 32 Pins are available for I/O operation. The four ports PORTA, PORTB, PORTC, and PORTD are programmed for performing desired operation.

Pins	8-pin	28-pin	40-pin	64-pin	100-pin
Chip	ATtiny25/45/85	ATmega8/48/88	ATmega32/16	ATmega64/128	ATmega1280
Port A			X	X	X
Port B	6 bits	X	X	X	X
Port C		7 bits	X	X	X
Port D		X	X	X	X
Port E				X	X
Port F				X	X
Port G				5 bits	6 bits
Port H					X
Port J					X
Port K					X
Port L					X

Here the digital I/O ports of ATmega32 are discussed

- AVR ATmega16 has 32 pins constituting four ports. The ports are listed below :
  1. PORT A
  2. PORT B
  3. PORT C
  4. PORT D.
- Each port has 8 pins.
- The pins of these four ports can be used as general-purpose inputs/outputs.
- These pins can be configured as input or output using the three I/O registers for each port. These registers are listed below :
  1. **DDRxregisters:** These are used to decide the direction of the pins, i.e. whether the pins will act as input pins or as output pins
  2. **PINxregisters:** These are used to read the logic level on the port pins.
  3. **PORTxregisters:** These are used to set the logic on the pins HIGH or LOW.
  (Where x can be A, B, C, or D depending on which port registers are being addressed).

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other. The pin driver is robust enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both VCC and Ground as indicated in the figure.



I/O Port Registers

There are three I/O registers associated with each port. These are as follows:

- Data register PORTx
- Data direction register DDRx

- Input pins address register PINx

Where x can be A, B, C, or D depending on which port is being addressed.

DDRx: Data Direction Registers

- These are 8-bit registers.
- These are used to configure the pins of the ports as input or output.
- Writing a one to the bits in this register sets those specific pins as output pins.
- Writing a zero to the bits in this register sets those specific pins as input pins.
- All bits in these registers can be read as well as written to.
- The initial value of these bits is zero.

Example :

1. Setting Port D as an output port :
   DDRD = 0xFF;
   Writing 1 to the bits of this register configures the pins corresponding to those pins as output pins.
   Here we have configured all 8 pins of Port D as output pins by writing 1 to each of them. (0xFF = 0b11111111).
2. Setting Port D as an input port :
   DDRD = 0x00;
   Writing 0 to the bits of this register configures the pins corresponding to those pins as output pins.
   Here we have configured all 8 pins of Port D as input pins by writing 0 to each of them. (0x00 = 0b00000000).

PORTx: Data Registers

- These are 8-bit registers.
- These are used to put the pins of the ports in a logic HIGH or logic LOW state.
- Writing a one to the bits in this register puts a HIGH logic (5V) on those pins.
- Whereas writing a zero to the bits in this register puts a LOW logic (0V) on those pins.
- All bits in these registers can be read as well as written to.
- The initial value of these bits is zero.

Example:

We will use the PORTx register of Port D to write a value 0x55 to Port D.

PORTD = 0x55;

PINx: Input Pins Address Registers

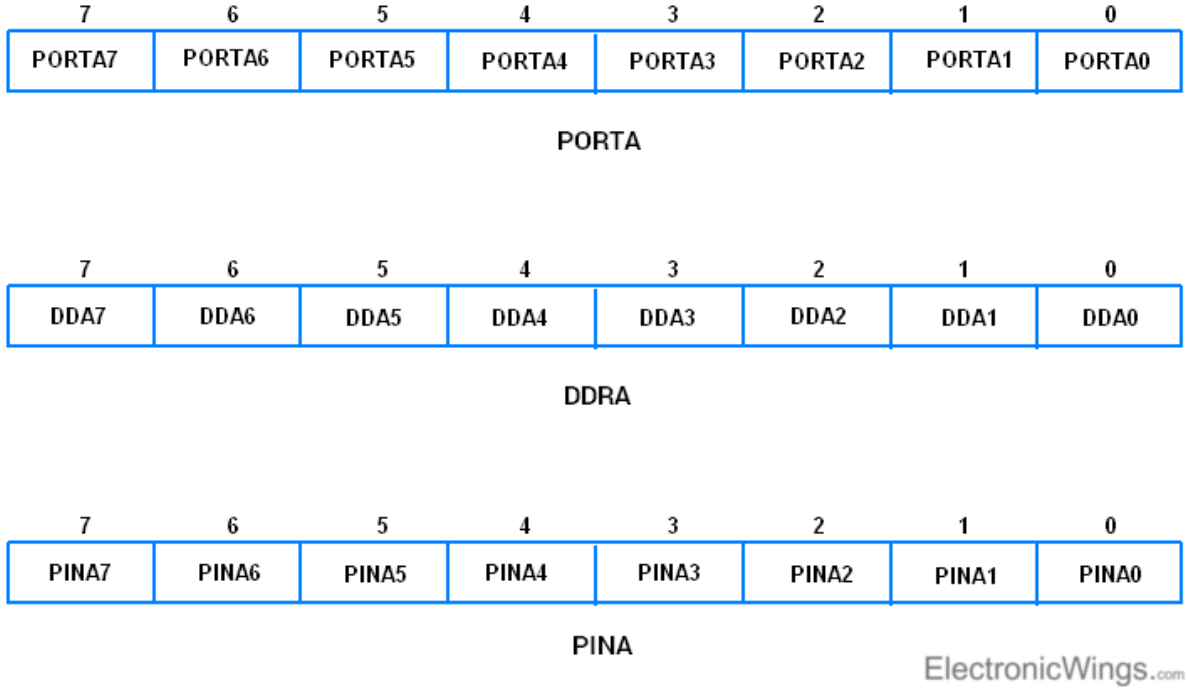
- These are 8-bit registers.
- These are used to read the values on the specific pins of the port.
- These bits are read-only bits and cannot be written to.

Example:

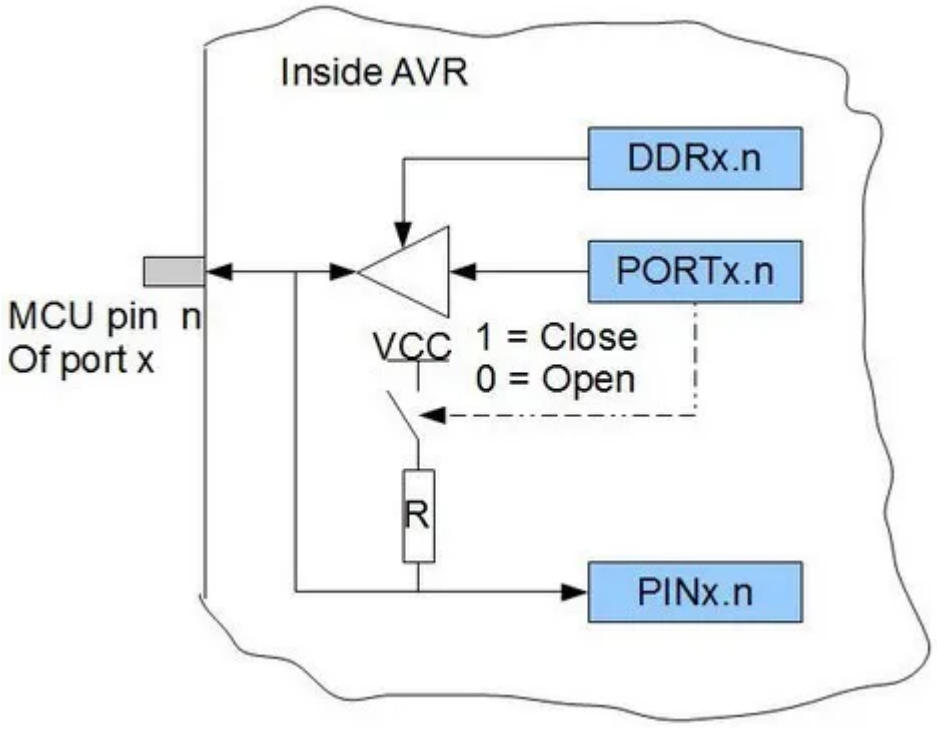
We will read the value on Port D in an 8-bit variable named ‘port_value’.

port_value = PIND;

In the figure shown below, the above mentioned three registers for PortA are shown. These are 8-bit registers.



Inside AVR Port (PORT STructure)



To enable output to the pin, we need to write logic ‘1’ to DDx.n pin. This will enable buffer

to let bit through from the PORTx register. If PORTx.n bit is '1' then it can source a pin with VCC voltage and up to 20mA of the source; in another hand, if logic '0' is written then the pin can source the target circuit.

If DDRx.n has logic '0' value, then the buffer enters the high impedance tri-state (Hi-Z). In this case, PORTx.n is disconnected from the pin, but there appears an ability to read pin value directly. As PORTx.n bit is free we can have some use of it – write '1' to it to enable internal pull-up resistor. Otherwise, the pin will be left in the Hi-Z state, and you will need to connect external pull-ups to read pins correctly.

3. Serial I/O

Most of the AVR microcontrollers support three types of serial COmmunication

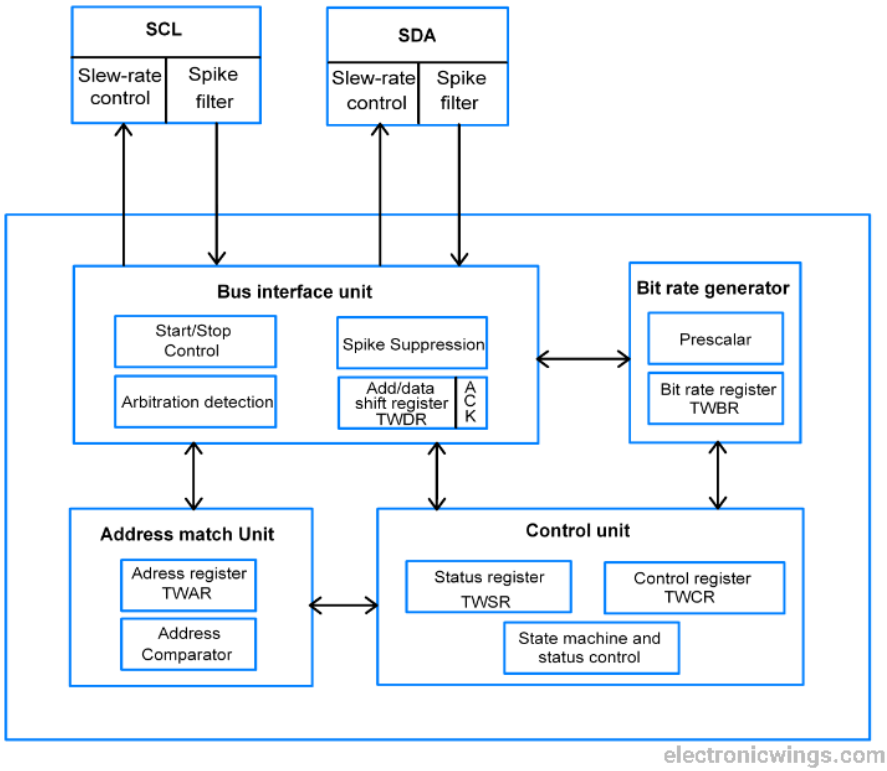
1. Inter IC bus or I2C
2. Serial Peripheral Interface or SPI
3. Universal Synchronous/Asynchronous receiver Transmitter or USART

I2C in AVR ATmega16/ATmega32

I2C (Inter-Integrated Circuit) is a serial bus interface connection protocol. It is also called TWI (two-wire interface) since it uses only two wires for communication, that two wires called SDA (serial data) and SCL (serial clock). AVR-based ATmega16/ATmega32 has a TWI module made up of several submodules as shown in the figure.

I2Cworks in two modes namely,

- Master mode
- Slave mode



ATmega16/32 I2C Module

SDA & SCL pins

- These pins are used to interface the TWI based external peripherals and microcontroller.
- The output drivers contain a slew-rate limiter. The input stages contain a spike suppression unit which removes spikes shorter than 50 ns.

Bus interface unit

- The bus interface unit contains Start/Stop control which is responsible for the generation and detection of START and STOP conditions.
- TWDR add/data shift register contains data to be transmitted and received.
- ACK bit receives ack/nack in transmitter mode and it is generated through software in receiving mode.
- The Spike suppression unit takes care of spikes whereas arbitration detection continuously monitors bus status and informs the control unit about it.

Address match unit

In slave mode, the Address match unit receives an incoming 7-bit address and compares with the address in TWAR (Two Wire Address Register) register to check whether it matches or not and upon match occur it intimate to control unit to take necessary action. It also considers the general call address if the TWGCE bit in TWAR is enabled.

Bit rate generator unit

The bit rate generator unit controls the SCL period in the master mode to generate SCL frequency. It is calculated by,

SCL frequency = (CPU CLK frequency)/(16+2(TWBR)*4^TWPS)

Where TWPS is a value of a pre-scaler bit in TWSR.

Control unit

- The Control unit contains TWSR (TWI status register), TWCR (TWI control register).
- It controls the overall process of attention for necessary events, identifying events when occur, TWINT interrupts assertion, and update TWSR.
- As long as the TWINT flag set SCL held low. TWINT set whenever TWI completes the current task.

Let see registers in the ATmega16/32 I2C module

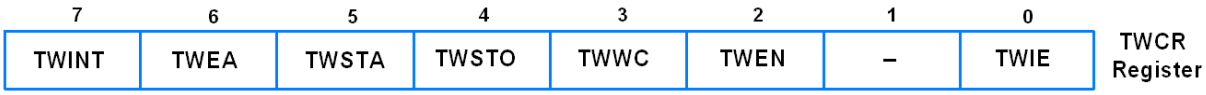
TWBR: TWI Bit Rate Register

TWI bit rate register used in generating SCL frequency while operating in master mode

7	6	5	4	3	2	1	0	
TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR Register

TWCR: TWI Control Register

TWI control resistor used to control events of all I2C communication.



Bit 7 – TWINT: TWI interrupt

- This bit gets set whenever TWI completes its current event (like start, stop, transmit, receive, etc).
- While the TWIE bit in TWCR is enabled then the TWI interrupt vector is called whenever TWI interrupt occurs.
- **TWI interrupt flag must be cleared by software by writing a logical one to it.** This bit is not automatically cleared by hardware.

Bit 6 – TWEA: TWI enable acknowledgment bit

- This is TWI acknowledgment enable bit, it is set in receiver mode to generate acknowledgment and cleared in transmit mode.

Bit 5 – TWSTA: TWI START condition bit

- The master device set this bit to generate START conditions by monitoring free bus status to take control over the TWI bus.

Bit 4 – TWSTO: TWI STOP condition bit

- The master device set this bit to generate STOP condition to leave control over the TWI bus.

Bit 3 – TWWC: TWI write collision

- This bit is set when writing to the TWDR register before the current transmission not complete i.e. TWINT is low.

Bit 2 – TWEN: TWI enable bit

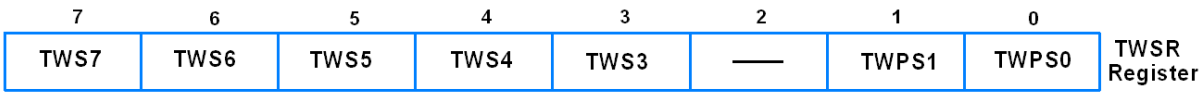
- This bit is set to enable the TWI interface in the device and takes control over the I/O pins.

Bit 1 - Reserved

Bit 0 – TWIE: TWI interrupt enable

- This bit is used to enable TWI to interrupt routine while the I-bit of SREG is set as long as the TWINT flag is high.

TWSR: TWI Status Register



Bit 7:Bit 3 - TWS7: TWS3: TWI status bits

- TWI status bits shows the status of TWI control and bus

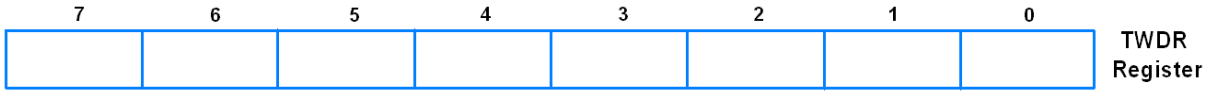
Bit 1:0 - TWPS1:TWPS0: TWI pre-scaler bits

- TWI pre-scaler bits used in bit rate formula to calculate SCL frequency

TWPS1	TWPS0	Exponent	Pre-scaler value
0	0	0	1
0	1	1	4
1	0	2	16
1	1	3	64

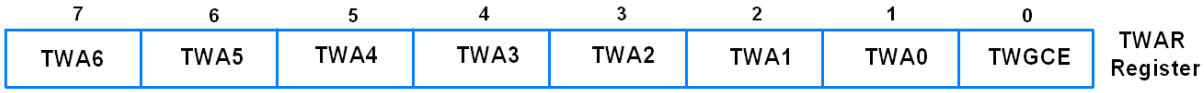
TWDR: TWI Data Register

- TWDR contains data to be transmitted or received.
- It's not writable while TWI is in process of shifting a byte.
- The data remains stable as long as TWINT is set.



TWAR: TWI Address Register

- TWAR register contains the address of the TWI unit in slave mode.
- It is mostly used in the multi-master system.



Bit 7:1 - TWA6: TWA0: TWI address bits

- TWI address bits contain TWI 7-bit address with which it can called by other masters in slave mode.

Bit 0 – TWGCE: TWI general call enable bit

- TWI general call enable bit when set it enables recognition of general call over the TWI bus. (The general call address comprises of a slave address equal to 0000 000 followed by R/W = 0 and is reserved to implement special operations of devices. The general call address is for addressing every device connected to the I2C-bus at the same time)

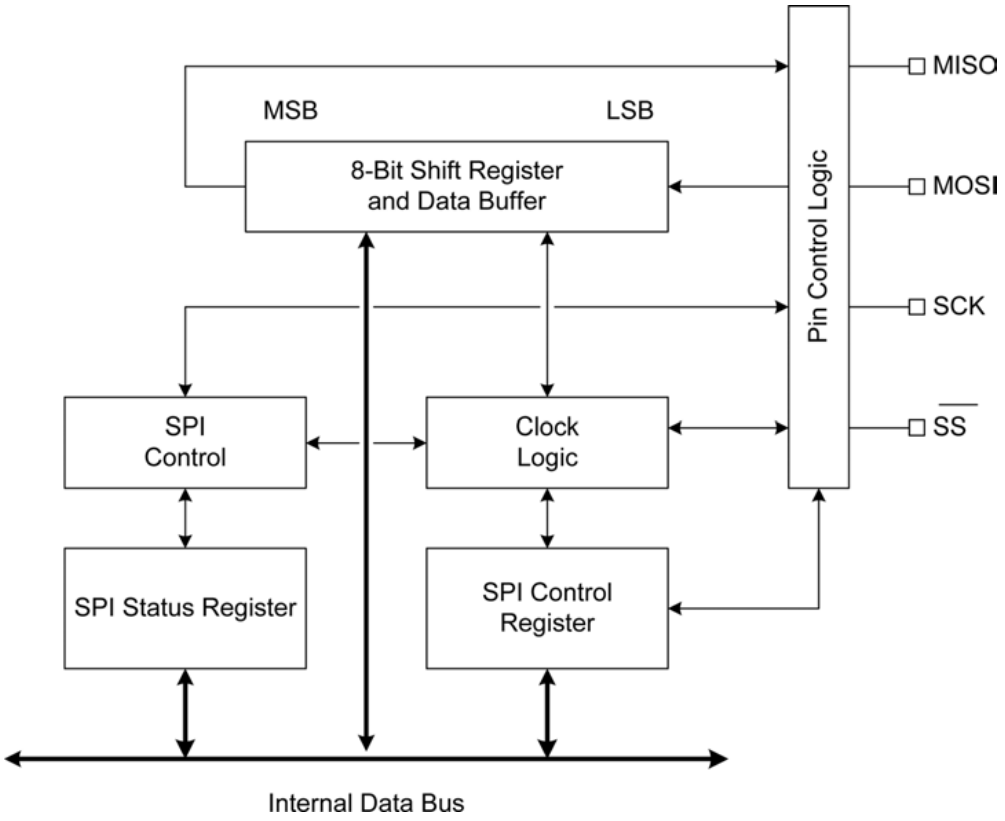
SPI in AVR ATmega16/ATmega32

The Serial Peripheral Interface (SPI) is a bus interface connection protocol originally started by Motorola Corp. It uses four pins for communication.

- **SDI** (Serial Data Input)
- **SDO** (Serial Data Output)
- **SCLK** (Serial Clock)

- **CS** (Chip Select)

It has two pins for data transfer called SDI (Serial Data Input) and SDO (Serial Data Output). SCLK (Serial Clock) pin is used to synchronize data transfer and Master provides this clock. CS (Chip Select) pin is used by the master to select the slave device.



SPI devices have 8-bit shift registers to send and receive data. Whenever a master needs to send data, it places data on the shift register and generates a required clock. Whenever a master wants to read data, the slave places the data on the shift register and the master generates a required clock. Note that SPI is a full-duplex communication protocol i.e. data on master and slave shift registers get interchanged at the same time.

ATmega16/32 SPI Communication

ATmega16 has an inbuilt SPI module. It can act as a master and slave SPI device. SPI communication pins in AVR ATmega are:

- **MISO** (Master In Slave Out)

  The Master receives data and the slave transmits data through this pin.

- **MOSI** (Master Out Slave In)

  The Master transmits data and the slave receives data through this pin.

- **SCK** (Shift Clock)

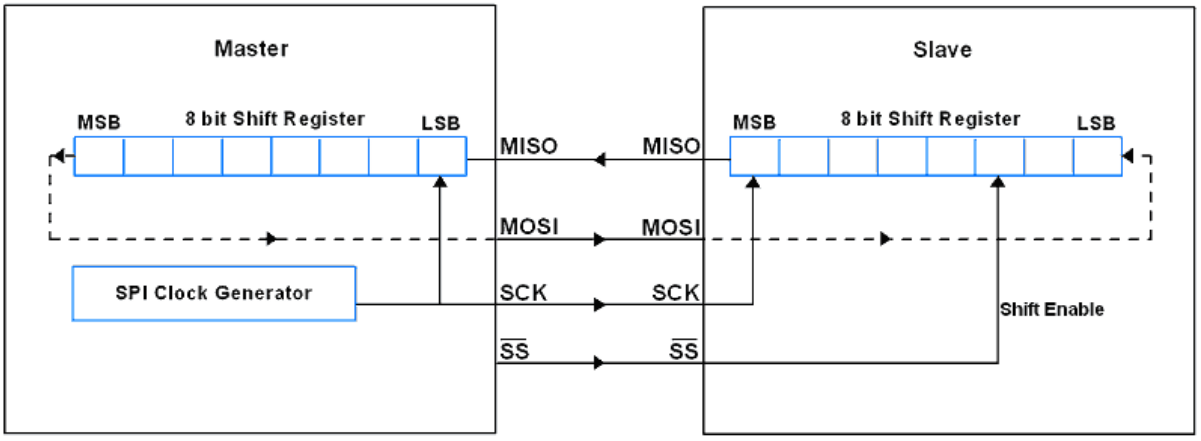
  The Master generates this clock for the communication, which is used by the slave.

  The only master can initiate a serial clock.

- **SS** (Slave Select)

  Master can select slaves through this pin.

The interconnection between master and slave using SPI is shown in the below figure.



SPI Master Slave Interconnection

ATmega16/32 SPI Communication Pins

SPI Pins of ATmega 16/32

Pins Configurations

SPI Pins	Pin No. on ATmega16 chip	Pin Direction in master mode	Pin Direction in slave mode
MISO	7	Input	Output
MOSI	6	Output	Input
SCK	8	Output	Input
SS	5	Output	Input

AVR ATmega16 uses three registers to configure SPI communication that are SPI Control Register, SPI Status Register and SPI Data Register.

Let's see these Registers.

SPCR: SPI Control Register

7	6	5	4	3	2	1	0	
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR

Bit 7 – SPIE: SPI Interrupt Enable bit

1 = Enable SPI interrupt.

0 = Disable SPI interrupt.

Bit 6 – SPE: SPI Enable bit

1 = Enable SPI.

0 = Disable SPI.

Bit 5 – DORD: Data Order bit

1 = LSB transmitted first.

0 = MSB transmitted first.

Bit 4 – MSTR: Master/Slave Select bit

1 = Master mode.

0 = Slave Mode.

Bit 3 – CPOL: Clock Polarity Select bit

1 = Clock start from logical one.

0 = Clock start from logical zero.

Bit 2 – CPHA: Clock Phase Select bit

1 = Data sample on trailing clock edge.

0 = Data sample on the leading clock edge.

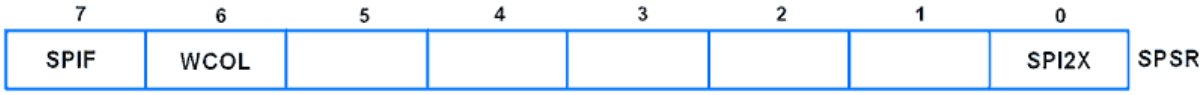
Bit 1:0 – SPR1: SPR0 SPI Clock Rate Select bits

The below table shows the SCK clock frequency select bit setting.

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	Fosc/4
0	0	1	Fosc/16
0	1	0	Fosc/64
0	1	1	Fosc/128
1	0	0	Fosc/2
1	0	1	Fosc/8
1	1	0	Fosc/32

1	1	1	Fosc/64

SPSR: SPI Status Register



Bit 7 – SPIF: SPI interrupt flag bit

- This flag gets set when the serial transfer is complete.
- Also gets set when the SS pin is driven low in master mode.
- It can generate an interrupt when SPIE bit in SPCR and a global interrupt is enabled.

Bit 6 – WCOL: Write Collision Flag bit

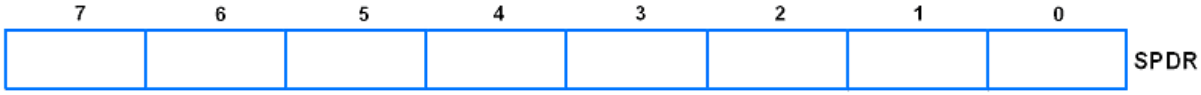
- This bit gets set when SPI data register writes occurs during previous data transfer.

Bit 5:1 – Reserved Bits

Bit 0 – SPI2X: Double SPI Speed bit

- When set, SPI speed (SCK Frequency) gets doubled.

SPDR: SPI Data Register



- SPI Data register used to transfer data between the Register file and SPI Shift Register.
- Writing to the SPDR initiates data transmission.

When the device is in master mode

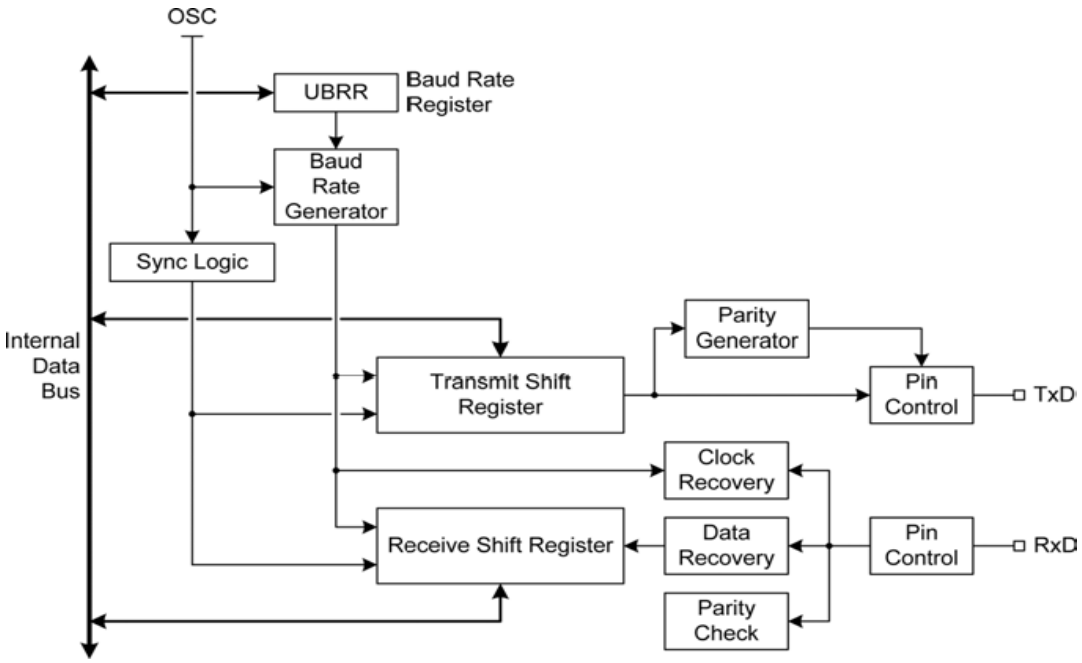
- Master writes data byte in SPDR. Writing to SPDR starts data transmission.
- 8-bit data starts shifting out towards slave and after the complete byte is shifted, SPI clock generator stops, and SPIF bit gets set.

When the device is in slave mode

- THE Slave SPI interface remains in sleep as long as the SS pin is held high by the master.
- It activates only when the SS pin is driven low. Data is shifted out with incoming SCK clock from master during a write operation.
- SPIF is set after the complete shifting of a byte.

USART in AVR ATmega16/ATmega32

AVR ATmega has flexible USART, which can be used for serial communication with other devices like computers, serial GSM, GPS modules, etc.



To program, first, we need to understand the basic registers used for USART

1. **UDR: USART Data Register**

It has basically two registers, one is Tx. Byte and the other is Rx Byte. Both share the same UDR register. Do remember that, when we write to the UDR reg. Tx buffer will get written and when we read from this register, Rx Buffer will get read. Buffer uses the FIFO shift register to transmit the data.

2. **UCSRA**: USART Control and Status Register A. As the name suggests, is used for control and status flags. In a similar fashion, there are two more USART control and status registers, namely UCSRB and UCSRC.

3. **UBRR**: USART Baud Rate Register, this is a 16-bit register used for the setting baud rate.

We will see this register in detail:

UCSRA: USART Control and Status Register A

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

- **Bit 7 – RXC:** USART Receive Complete

This flag bit is set when there is unread data in UDR. The RXC Flag can be used to generate a Receive Complete interrupt.

- **Bit 6 – TXC:** USART Transmit Complete

This flag bit is set when the entire frame from Tx Buffer is shifted out and there is no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically

cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt.

- **Bit 5 – UDRE:** USART Data Register Empty

If UDRE is one, the buffer is empty which indicates the transmit buffer (UDR) is ready to receive new data. The UDRE Flag can generate a Data Register Empty Interrupt. UDRE is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FE:** Frame Error
- **Bit 3 – DOR:** Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters) and a new character is waiting in the receive Shift Register.

- **Bit 2 – PE:** Parity Error
- **Bit 1 – U2X:** Double the USART Transmission Speed
- **Bit 0 – MPCM:** Multi-processor Communication Mode

UCSRB: USART Control and Status Register B

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

- **Bit 7 – RXCIE:** RX Complete Interrupt Enable

  Writing one to this bit enables interrupt on the RXC Flag.

- **Bit 6 – TXCIE:** TX Complete Interrupt Enable

  Writing one to this bit enables interrupt on the TXC Flag.

- **Bit 5 – UDRIE:** USART Data Register Empty Interrupt Enable

  Writing one to this bit enables interrupt on the UDRE Flag.

- **Bit 4 – RXEN:** Receiver Enable

  Writing one to this bit enables the USART Receiver.

- **Bit 3 – TXEN:** Transmitter Enable

  Writing one to this bit enables the USART Transmitter.

- **Bit 2 – UCSZ2:** Character Size

The UCSZ2 bits combined with the UCSZ1:0 bit in UCSRC sets the number of data bits (Character Size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8:** Receive Data Bit 8
- **Bit 0 – TXB8:** Transmit Data Bit 8

UCSRC: USART Control and Status Register C

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	USCZ1	USCZ0	UCPOL

- **Bit 7 – URSEL:** Register Select

This bit selects between accessing the **UCSRC** or the **UBRRH** Register, as both register shares the same address. The URSEL must be one when writing the UCSRC or else data will be written in the UBRRH register.

- **Bit 6 – UMSEL:** USART Mode Select

  This bit selects between the Asynchronous and Synchronous mode of operation.

  0 = Asynchronous Operation

  1 = Synchronous Operation

- **Bit 5:4 – UPM1:0**: Parity Mode

These bits enable and set the type of parity generation and check. If parity a mismatch is detected, the PE Flag in UCSRA will be set.

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBS**: Stop Bit Select

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

0 = 1-bit

1 = 2-bit

- **Bit 2:1 – UCSZ1:0**: Character Size

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

UCSZ2	UCSZ1	UCSZ0	Character Size

0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	**1**	**1**	**8-bit**
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOL:** Clock Polarity

This bit is used for synchronous mode only. Write this bit to zero when the asynchronous mode is used.

UBRRL and UBRRH: USART Baud Rate Registers

- **Bit 15 – URSEL:** Register Select

This bit selects between accessing the **UCSRC** or the **UBRRH** Register, as both register shares the same address. The URSEL must be one when writing the UCSRC or else data will be written in the UBRRH register.

- **Bit 11:0 – UBRR11:0:** USART Baud Rate Register.

  Used to define the baud rate

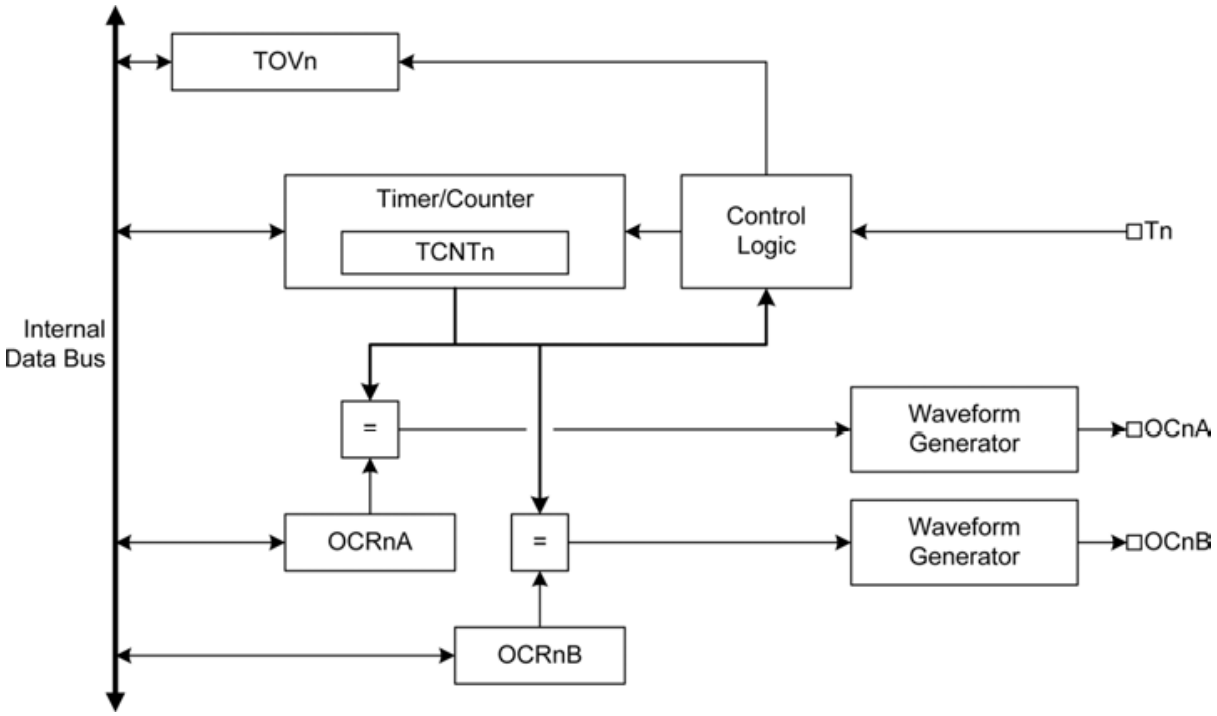
$$\mathbf{UBBR} = \frac{\mathbf{Fosc}}{\mathbf{16 * BaudRate}} - \mathbf{1} \qquad \mathbf{BaudRate} = \frac{\mathbf{Fosc}}{\mathbf{16 * (UBBR + 1)}}$$

Example: suppose Fosc=8 MHz and required baud rate= 9600 bps.

Then the value of **UBRR= 51.088 i.e. 51.**

2.3 AVR Microcontroller architecture: Timers/Counters, Analog Comparator, ADC Interrupts.

4. Timer Counters in AVR



Generally, we use a timer/counter to generate time delays, waveforms, or to count events. Also, the timer is used for PWM generation, capturing events, etc.

In AVR ATmega16 / ATmega32, there are three timers:

- Timer0: 8-bit timer
- Timer1: 16-bit timer
- Timer2: 8-bit timer

Basic registers and flags of the Timers

TCNTn: Timer / Counter Register

Every timer has a timer/counter register. It is zero upon reset. We can access value or write a value to this register. It counts up with each clock pulse.

TOVn: Timer Overflow Flag

Each timer has a Timer Overflow flag. When the timer overflows, this flag will get set.

TCCRn: Timer Counter Control Register

This register is used for setting the modes of timer/counter.

OCRn: Output Compare Register

The value in this register is compared with the content of the TCNTn register. When

they are equal, the OCFn flag will get set.

Let us see Timer0 to understand the timers in ATmega16 / ATmega32

Timer0

First, we need to understand the basic registers of the Timer0

1. TCNT0: Timer / Counter Register 0

   It is an 8-bit register. It counts up with each pulse.

2. TCCR0: Timer / Counter Control register 0

   This is an 8-bit register used for the operation mode and the clock source selection.

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 7- FOC0: Force compare match

Write only a bit, which can be used while generating a wave. Writing 1 to this bit causes the wave generator to act as if a compare match has occurred.

Bit 6, 3 - WGM00, WGM01: Waveform Generation Mode

WGM00	WGM01	Timer0 mode selection bit
0	0	Normal
0	1	CTC (Clear timer on Compare Match)
1	0	PWM, Phase correct
1	1	Fast PWM

Bit 5:4 - COM01:00: Compare Output Mode

These bits control the waveform generator. We will see this in the compare mode of the timer.

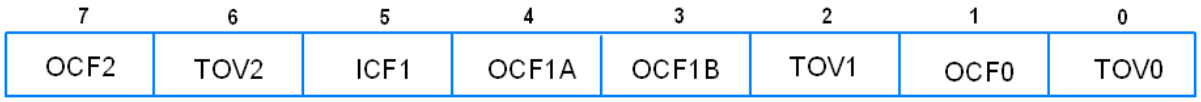
Bit 2:0 - CS02:CS00: Clock Source Select

These bits are used to select a clock source. When CS02: CS00 = 000, then timer is stopped. As it gets a value between 001 to 101, it gets a clock source and starts as the timer.

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer / Counter stopped)

0	0	1	clk (no pre-scaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge.

3. TIFR: Timer Counter Interrupt Flag register



Bit 0 - TOV0: Timer0 Overflow flag

0 = Timer0 did not overflow

1 = Timer0 has overflown (going from 0xFF to 0x00)

Bit 1 - OCF0: Timer0 Output Compare flag

0 = Compare match did not occur

1 = Compare match occurred

Bit 2 - TOV1: Timer1 Overflow flag

Bit 3 - OCF1B: Timer1 Output Compare B match flag

Bit 4 - OCF1A: Timer1 Output Compare A match flag

Bit 5 - ICF1: Input Capture flag

Bit 6 - TOV2: Timer2 Overflow flag

Bit 7 - OCF2: Timer2 Output Compare match flag

Timer0 Overflow

Normal mode: When the counter overflows i.e. goes from 0xFF to 0x00, the TOV0 flag is set.

Timer Input Capture Mode in AVR ATmega16/ATmega32

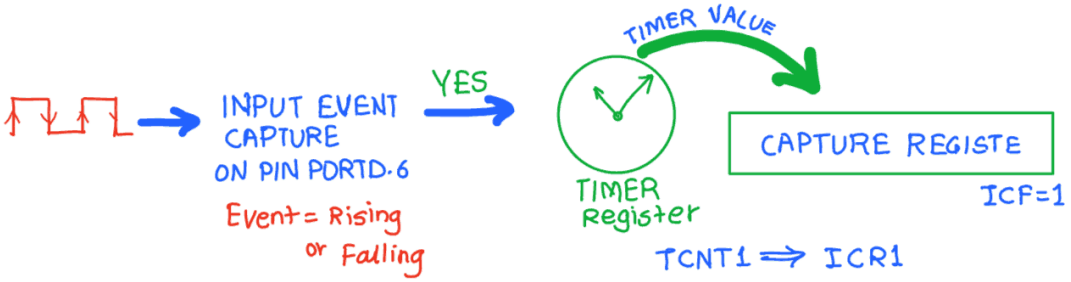
The input capture function is used in many applications such as:

- Pulse width measurement

- Period measurement
- Capturing the time of an event

In AVR ATmega32, Timer1 can be used as an input capture to detect and measure events happening outside the microcontroller.

Upon detection of a defined event i.e. rising edge or falling edge on ICP pin (PORTD.6), the TCNT1(Timer / Counter register) value is loaded into the ICR1 (input capture) register and the ICF1 flag will get set.



Programming

To program, first, let us see TCCR1B (Timer Counter Control Register B)

TCCR1B: Timer Counter Control Register B

7	6	5	4	3	2	1	0
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10

Bit 7 - ICNC1: Input Capture Noise canceller

Setting this bit activates the noise canceller. It causes a delay of 4 clock cycles as it considers a change only if it persists for at least 4 successive system clocks.

Bit 6 - ICES1: Input Capture Edge select

Select edge detection for input capture function.

0 = Capture on the falling edge

1 = Capture on rising edge

Bit 4: 3 - WGM13 : WGM12: Timer1 Mode select

These bits are used for mode selection like Normal mode, PWM mode, CTC mode, etc. here we will select normal mode, so set these bits to zero.

Bit 2: 0 - CS12: CS10: Timer1 Clock Select

CS02	CS01	CS00	Description
0	**0**	**0**	No clock source (Timer / Counter stopped)
0	**0**	**1**	clk (no pre-scaling)
0	**1**	**0**	clk / 8
0	**1**	**1**	clk / 64
1	**0**	**0**	clk / 256
1	**0**	**1**	clk / 1024
1	**1**	**0**	External clock source on T0 pin. Clock on falling edge
1	**1**	**1**	External clock source on T0 pin. Clock on rising edge.

16-Bit Timer/Counters

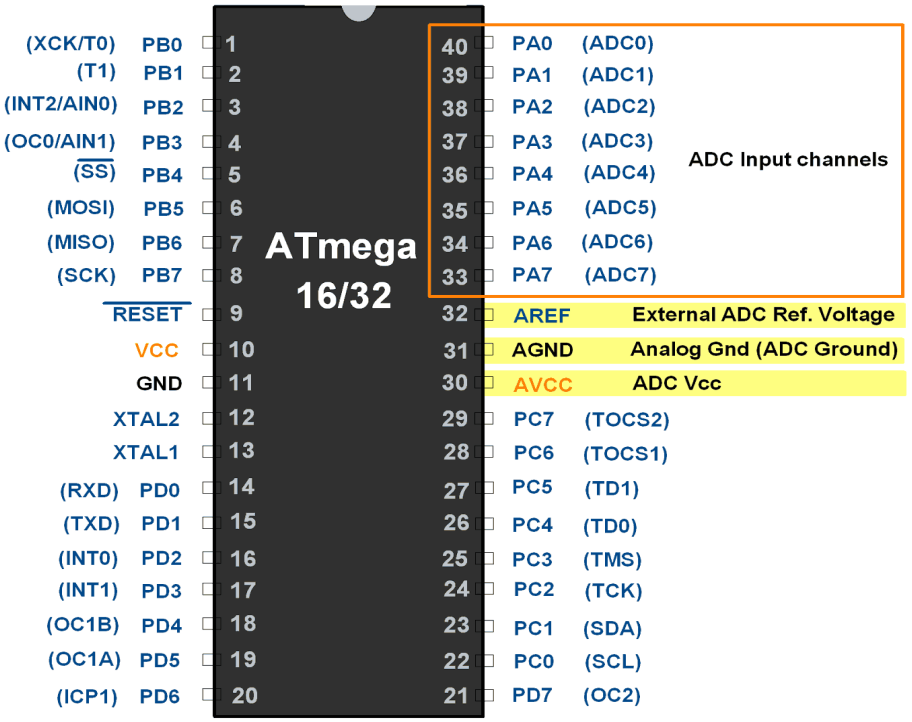
The 16-bit timer/counter is similar to the 8-bit version, but with an extended count range. It is true 16-bit logic, which allows for 16-bit variable period PWM generation. The module also features two independent output comparison circuits, double buffered output comparison registers, and an input capture circuit with noise canceling. In addition to PWM generation the 16-bit timer/counter can be used for high resolution external event capture, frequency generation, and signal timing measurement. It has the ability to generate four different interrupts (TOV1, OCF1A, OCF1B, and ICF1).

5. ADC in AVR

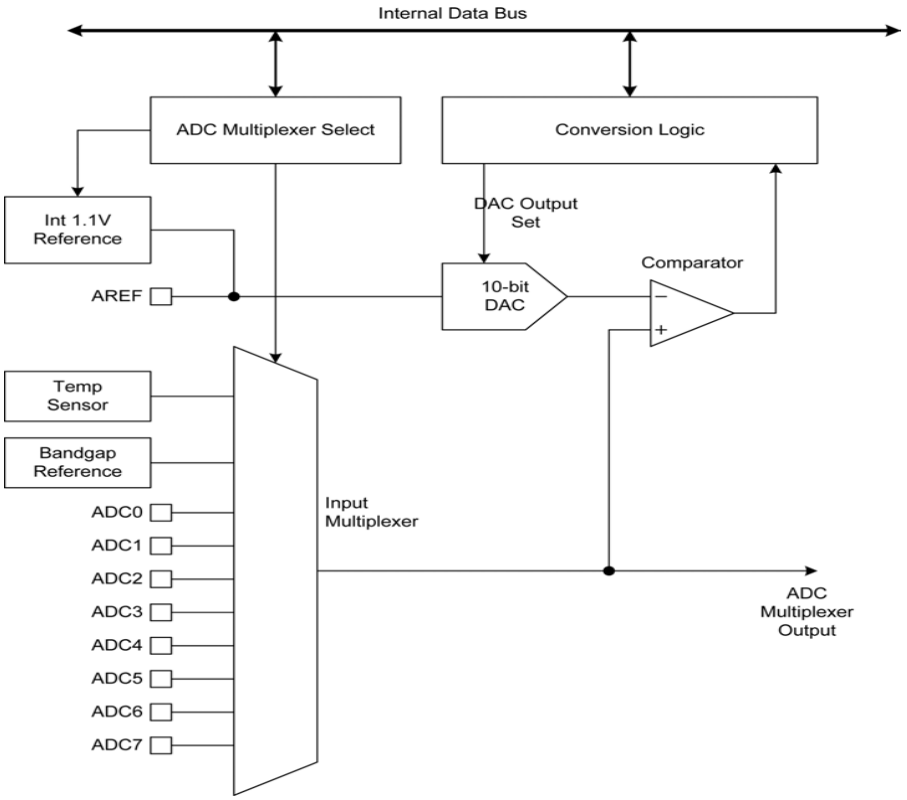
ADC (Analog to Digital converter) is the most widely used device in embedded systems which is designed especially for data acquisition. In the AVR ATmega series normally 10-bit ADC is inbuilt in the controller.

Let us see how to use the ADC of AVR ATmega16 / ATmega32.

ATmega16/32 supports eight ADC channels, which means we can connect eight analog inputs at a time. ADC channel 0 to channel 7 are present on PORTA. i.e. Pin no.33 to 40.



ADC Pins of ATmega 16/32



The controller has a 10 bit ADC, which means we will get digital output 0 to 1023.

i.e. When the input is 0V, the digital output will be 0V & when input is 5V (and Vref=5V), we will get the highest digital output corresponding to 1023 steps, which is 5V.

So controller ADC has 1023 steps and

- Step size with Vref=5V : 5/1023 = 4.88 mV.
- Step size with Vref=2.56 : 2.56/1023 = 2.5 mV.

So Digital data output will be D*out* = Vin / step size.

ATmega16/32 ADC

- It is 10-bit ADC, which means we will get digital output 0 to 1023.
- 8 analog input channels
- Converted output binary data is held in two special functions: 8-bit register ADCL (result Low) and ADCH (result in High).
- ADC gives 10-bit output, so (ADCH: ADCL) only 10-bits are useful out of 16-bits.
- We have options to use this 10-bits as upper bits or lower bits.
- We also have three options for Vref. 1. AVcc (analog Vcc), 2. Internal 2.56 v3. External A*ref*. Pin.
- The total conversion time depends on crystal frequency and ADPS0: 2 (frequency divisor)
- If you decide to use an AVcc or Vref pin as ADC voltage reference, you can make it more stable and increase the precision of ADC by connecting a capacitor between that pin and GND.

ADC Registers

In AVR ADC, we need to understand four main register -

1. ADCH: Holds digital converted data higher byte
2. ADCL: Holds digital converted data lower byte
3. ADMUX: ADC Multiplexer selection register
4. ADCSRA: ADC Control and status register

ADCH: ADCL register

First, two-register holds the digital converted data, which is 10-bit.

ADMUX Register

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Bit 7: 6 – REFS1 : 0: Reference Selection Bits

Reference voltage selection for ADC

REFS1	REFS0	Vref to ADC
0	0	AREF pin
0	1	AVCC pin i.e. Vcc 5 V
1	0	Reserved
1	1	Internal 2

Bit 5 – ADLAR: ADC Left Adjust Result

Use 10-bits output as upper bits or lower bits in ADCH & ADCL.

Bits 4 : 0 – MUX4 : 0: Analog Channel and Gain Selection Bits

We can select input channel ADC0 to ADC7 by using these bits. These bits are also used to select comparator (inbuilt in AVR) inputs with various gain. We will cover these comparator operations in another part.

Selecting a channel is very easy, just put the channel number in MUX4 : 0.

Suppose you are connecting the input to ADC channel 2 then put 00010 in MUX4 : 0.

Suppose you are connecting the input to ADC channel 5 then put 00101 in MUX4 : 0.

ADCSRA Register:

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

- Bit 7 – ADEN: ADC Enable

Writing one to this bit enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- Bit 6 – ADSC: ADC Start Conversion

Writing one to this bit starts the conversion.

- Bit 5 – ADATE: ADC Auto Trigger Enable

Writing one to this bit, results in Auto Triggering of the ADC is enabled.

- Bit 4 – ADIF: ADC Interrupt Flag

This bit is set when an ADC conversion completes and the Data Registers are updated.

- Bit 3 – ADIE: ADC Interrupt Enable

Writing one to this bit, the ADC Conversion Complete Interrupt is activated.

- Bits 2 : 0 – ADPS2 : 0: ADC Prescaler Select Bits

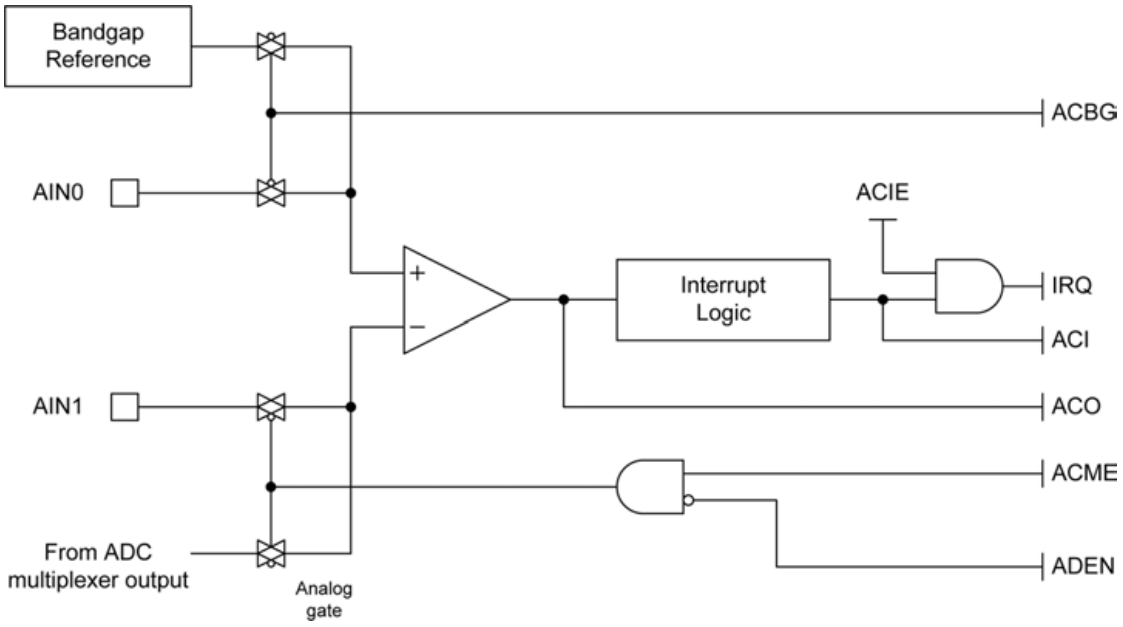
These bits determine the division factor between the XTAL frequency and the input clock to the ADC

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

We can select any divisor and set frequency Fosc/2, Fosc/4, etc. for ADC, But in AVR, ADC requires an input clock frequency less than 200KHz for max. accuracy. So we have to always take care of not exceeding ADC frequency more than 200KHz.

Suppose your clock frequency of AVR is 8MHz, then we must have to use divisor 64 or 128. Because it gives 8MHz/64 = 125KHz, which is lesser than 200KHz

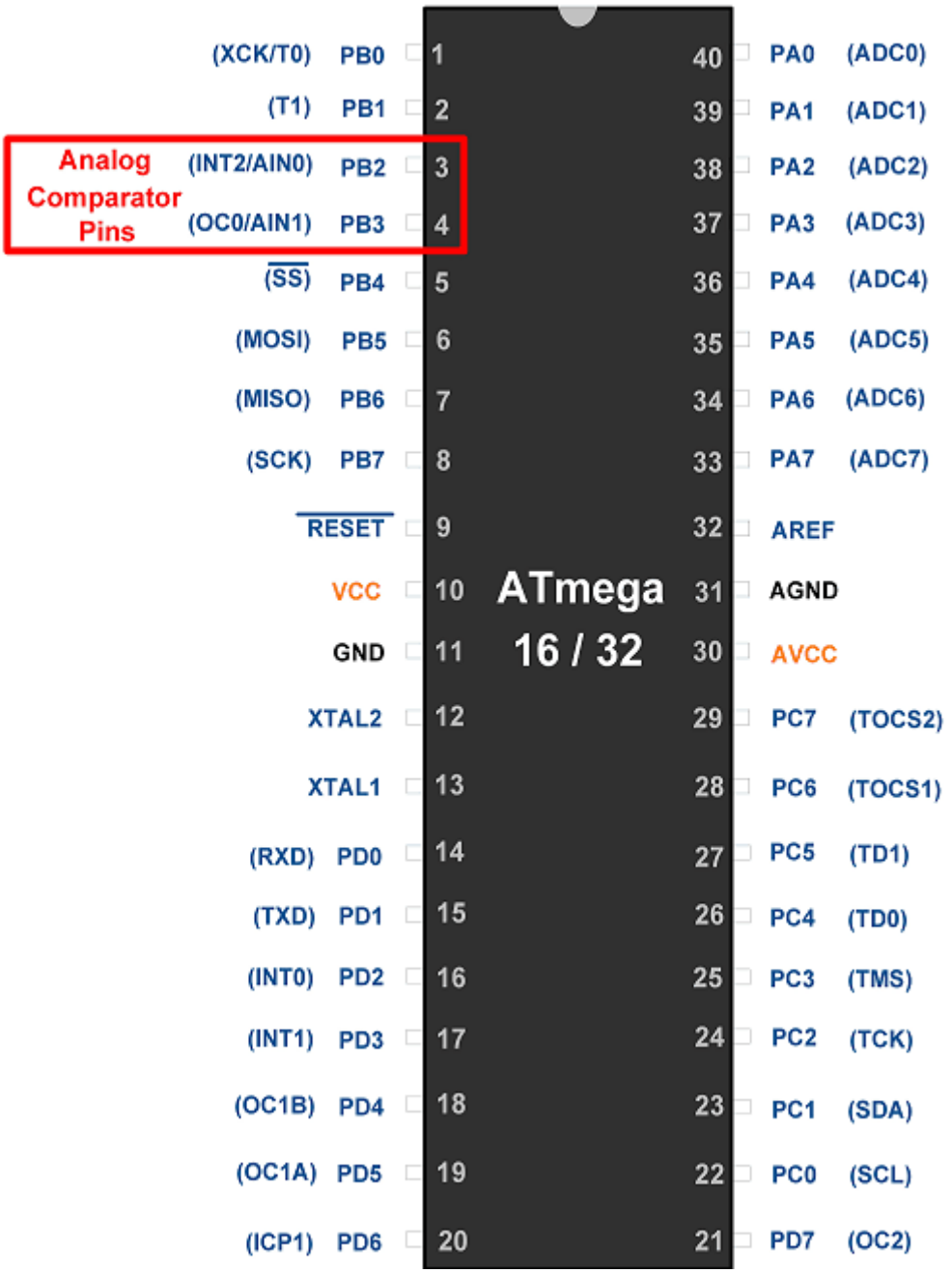
6. Analog Comparator



- ATmega16/ATmega32 based on AVR has two pins for analog voltage compare i.e. AIN0 and AIN1. AIN0 is the positive terminal whereas AIN1 is the negative terminal.
- When the voltage on positive pin AIN0 is higher than negative pin AIN1, the ACO bit of ACSR register is set.
- It is possible to use ADC channels (PA0 to PA7) as a negative terminal (AIN1) of a comparator. In this condition, AIN1 will not be considered as a negative input to the comparator. ADC multiplexer is used to select the ADC channel to be connected. ADC must be switched off to use this feature.
- It means it is possible to compare nine analog signals with one positive analog signal,

sequentially but not simultaneously.

Pins of Analog Comparator



Analog Comparator Pins of ATmega16 / 32

The register configurations for analog comparator are explained below.

SFIOR: Special Function IO Register

ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10

Bit 3 – ACME: Analog Comparator Multiplexer Enable

When the ACME bit is one and the ADC is switched off, the ADC multiplexer selects the input connected at ADC channels as a negative input to the Analog Comparator and AIN1 is not considered for negative input to the comparator. When the ACME bit is zero, AIN1 acts as the negative terminal of the Analog Comparator.

ACSR: Analog Comparator Control and Status Register

ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0

Bit 7 – ACD: Analog Comparator Disable

When ACD is one, then the power to the Analog Comparator is switched Off. That means analog comparator is disabled.

Bit 6 – ACBG: Analog Comparator Bandgap Select

When ACBG is one, then the bandgap reference voltage replaces the positive input to the analog comparator.

When ACBG is zero, then AIN0 is applied to the positive of the analog comparator.

Bit 5 – ACO: Analog Comparator Output

This bit indicates the output of the comparator, when AIN0 voltage is higher than the negative pin voltage then the ACO bit is set, otherwise it is clear.

Bit 4 – ACI: Analog Comparator Interrupt Flag

ACI bit is set by hardware. When a comparator output event triggers, the interrupt mode is defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 – ACIE: Analog Comparator Interrupt Enable

When the ACIE bit set to one and the I-bit in the Status Register is set, the Analog Comparator Interrupt is activated. When set to zero, the interrupt is disabled.

Bit 2 – ACIC: Analog Comparator Input Capture Enable

When ACIC set to one, this bit enables the Input Capture function in Timer/Counter1 to be triggered by the Analog Comparator. When Set to zero, the connection between Analog Comparator and Input Capture function does not exist.

Bit 1:0 - ACIS1:ACIS0: Analog Comparator Interrupt Mode Select

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle

0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

ADMUX: ADC Multiplexer Selection Register

REFS1	REFS0	ADLRA	MUX4	MUX3	MUX2	MUX1	MUX0

Bit 2:0 – MUX2: MUX0 – To select the ADC input (from ADC0 to ADC7) for the negative input of the comparator.

ACME	ADEN	MUX2.0	Analog Comparator Negative Input
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

7. Interrupts

AVR ATmega16/ATmega32 has three external hardware interrupts on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively. Upon activation of these interrupts, the ATmega controller gets interrupted in whatever task it is doing and jumps to perform the interrupt service routine.

External interrupts can be level-triggered or edge-triggered. We can program this triggering. INT0 and INT1 can be level-triggered and edge-triggered whereas INT2 can be only edge-triggered.

Programming External Interrupt

We can enable/disable external interrupts by the **GICR register**.

7	6	5	4	3	2	1	0
INT1	INT0	INT2	–	–	–	IVSEL	IVCE

- Bit 7 – INT1: External Interrupt Request 1 Enable

  0: Disable external interrupt

  1: Enable external interrupt
- Bit 6 – INT0: External Interrupt Request 0 Enable

  0: Disable external interrupt

  1: Enable external interrupt
- Bit 5 – INT2: External Interrupt Request 2 Enable

  0: Disable external interrupt

  1: Enable external interrupt

MCU Control Register (MCUCR)

To define a level trigger or edge trigger on external INT0 and INT1 pins MCUCR register is used.

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC01, ISC00 (Interrupt Sense Control bits)

These bits define the level or edge that triggers the INT0 pin.

ISC01	ISC00		Description
0	0		The low level on the INT0 pin generates an interrupt request.
0	1		Any logical change on the INT0 pin generates an interrupt request.
1	0		The falling edge on the INT0 pin generates an interrupt request.
1	1		The rising edge on the INT0 pin generates an interrupt request.

ISC11, ISC10 (Interrupt Sense Control bits)

These bits define the level or edge that triggers the INT1 pin.

ISC01	ISC00		Description
0	0		The low level on the INT1 pin generates an interrupt request.
0	1		Any logical change on the INT1 pin generates an interrupt request.
1	0		The falling edge on the INT1 pin generates an interrupt request.
1	1		The rising edge on the INT1 pin generates an interrupt request.

MCU Control and Status Register (MCUCSR)

To define the INT2 interrupt activity, bit 6 of MCUCSR is used.

7	6	5	4	3	2	1	0
JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF

ISC2 bit defines the INT2 interrupt triggering.

ISC2		Description
0		The falling edge on the INT2 pin generates an interrupt request.
1		The rising edge on the INT2 pin generates an interrupt request.

2.4 Features of Arduino specific AVR microcontroller ATmega 168/328

ATmega168 Features

Following table shows the main features of this module.

ATmega168 Features	
No. of Pins	28
CPU	RISC 8-Bit CMOS
Operating Voltage	1.8 to 5.5 V
Program Memory	16K
Program Memory Type	Flash
RAM	1K
EEPROM	512 Bytes
ADC Number of ADC Channels	10-Bit 8 channel
Comparator	1
In-circuit serial programming	Yes
Oscillator	up to 20 MHz
Timer (3)	16-Bit Timer (1) 8-Bit Timer (2)
Capture/Compare/PWM	1/1/6
Power Up Timer	Yes
I/O Pins	23 (PortB-8, PortC-7, PortD-8)
USART	Yes
SPI	2

I2C	Yes
Watchdog Timer	Yes
Brown out Detection (BOD)	Yes
Power on Reset	Yes
Data Retention	20 Years
Minimum Operating Temperature	-40 ºC
Maximum Operating Temperature	85 ºC

ATmega 328 Features

CPU	8-bit AVR
Number of Pins	28
Operating Voltage (V)	+1.8 V TO +5.5V
Number of programmable I/O lines	23

Communication Interface	Master/Slave SPI Serial Interface(17,18,19 PINS) [Can be used for programming this controller] Programmable Serial USART(2,3 PINS) [Can be used for programming this controller] Two-wire Serial Interface(27,28 PINS)[Can be used to connect peripheral devices like Servos, sensors and memory devices]
JTAG Interface	Not available
ADC Module	6channels, 10-bit resolution ADC
Timer Module	Two 8-bit counters with Separate Prescaler and compare mode, One 16-bit counter with Separate Prescaler,compare mode and capture mode.
Analog Comparators	1(12,13 PINS)
DAC Module	Nil
PWM channels	6
External Oscillator	0-4MHz @ 1.8V to 5.5V 0-10MHz @ 2.7V to 5.5V

	0-20MHz @ 4.5V to 5.5V
Internal Oscillator	8MHz Calibrated Internal Oscillator
Program Memory Type	Flash
Program Memory or Flash memory	32Kbytes[10000 write/erase cycles]
CPU Speed	1MIPS for 1MHz
RAM	2Kbytes Internal SRAM
EEPROM	1Kbytes EEPROM
Watchdog Timer	Programmable Watchdog Timer with Separate On-chipOscillator
Power Save Modes	Six Modes[Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby]
Operating Temperature	-40°C to +105°C(+105 being absolute maximum, -40 being absolute minimum)