Introduction to Micro-controllers

Anurag Dwivedi

Lecture Structure

- Things to be covered today..

- What is a micro-controller?
- What are the basic features of a micro-controller?
- How to input and output from a micro-controller?
- How to program a micro-controller?
- Demonstration : Led Blinking .
- Interfacing Character LCD with Micro-controller.
- Demonstration : CLCD Interfacing.

Embedded Systems

- Layman definition: Gadgets and devices
- Technical definition: Self-controlled devices
- Usually, such systems consist of I/O (input/output) devices such as LCDs, keypads, etc. and other devices like EEPROM (for storing data) connected to a central controlling device.

Example : MP3 Player
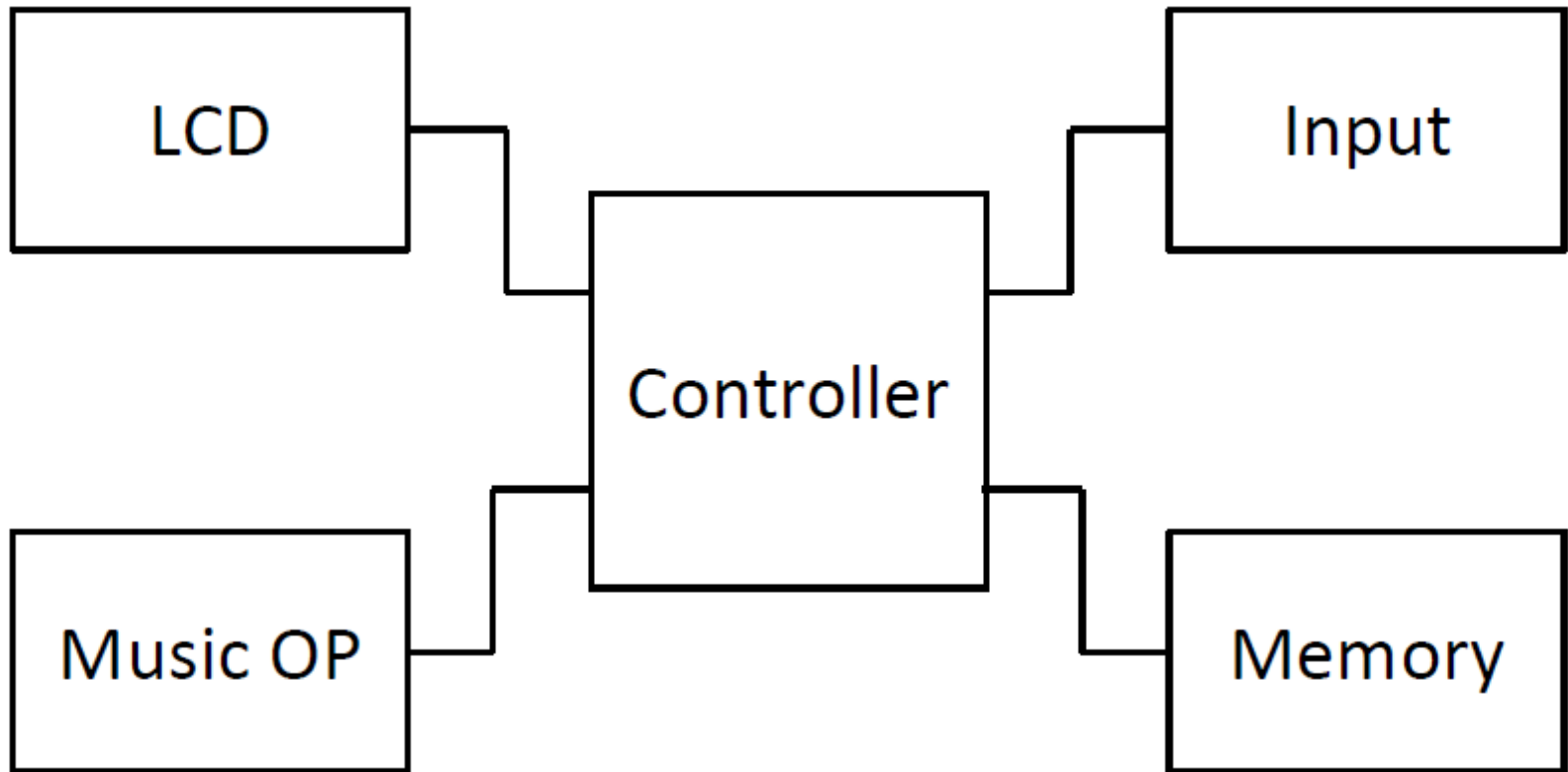
The Micro-controller

- Why "micro"?
- Larger controllers are available too: processors that run computers are an example.
- A microcontroller is essentially a mini-computer inside a single IC.

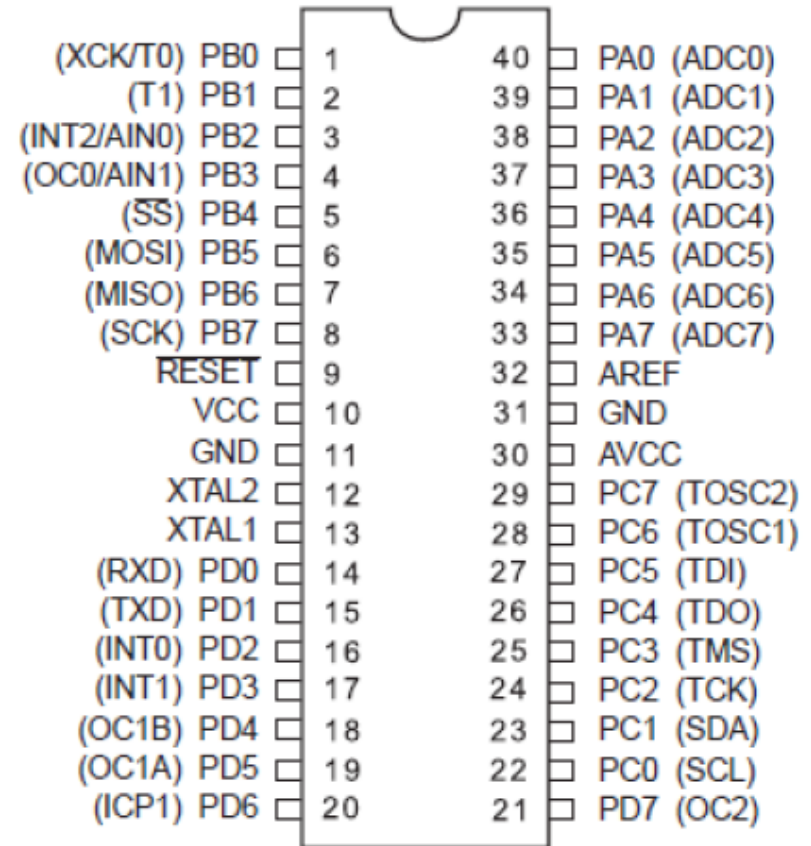# Micro-controllers

- Multiple microcontrollers available in the market.
- Vendors include Atmel, Intel, ARM, Cypress, etc.
- We will use Atmel ATmega microcontrollers because they are cheap, easy to use and powerful.

Atmega 16

- 40 pin IC.
- 32 pins for I/O.
- 8 pins reserved.
- I/O pins divided into 4 groups of 8 pins, called ports.
- Ports labeled as A, B, C and D.

(XCK/T0) PB0	1		40	PA0 (ADC0)
(T1) PB1	2		39	PA1 (ADC1)
(INT2/AIN0) PB2	3		38	PA2 (ADC2)
(OC0/AIN1) PB3	4		37	PA3 (ADC3)
(SS) PB4	5		36	PA4 (ADC4)
(MOSI) PB5	6		35	PA5 (ADC5)
(MISO) PB6	7		34	PA6 (ADC6)
(SCK) PB7	8		33	PA7 (ADC7)
RESET	9		32	AREF
VCC	10		31	GND
GND	11		30	AVCC
XTAL2	12		29	PC7 (TOSC2)
XTAL1	13		28	PC6 (TOSC1)
(RXD) PD0	14		27	PC5 (TDI)
(TXD) PD1	15		26	PC4 (TDO)
(INT0) PD2	16		25	PC3 (TMS)
(INT1) PD3	17		24	PC2 (TCK)
(OC1B) PD4	18		23	PC1 (SDA)
(OC1A) PD5	19		22	PC0 (SCL)
(ICP1) PD6	20		21	PD7 (OC2)

How does a micro-controller work

- Just like a computer, a microcontroller executes a program.
- The task to be performed is written in a while loop, and the micro-controller performs is endlessly.
- The program for a microcontroller is written in C language (although other languages are possible).

Input/Output (I/0)

▸ Input / Output is via special variables called "registers".

▸ Registers are actual hardware memory locations inside the μC. Their names and sizes are predefined.

▸ When we assign a value to these registers in the program, the actual value in the hardware changes.

▸ These values can be changed multiple times at any point in the program.

I/O Registers

▸ There are 3 registers that control the I/O pins: DDR, PORT and PIN.

▸ Each port has it's own registers. Hence, port A has registers DDRA, PORTA, PINA; port B has registers DDRB, PORTB, PINB; and so on.

▸ DDR, PORT and PIN serve different functions.

DDR (Data Direction Register)

▸ DDR decides whether the pins of a port are input pins or output pins.

▸ If the pin is input, then the voltage at that pin is undecided until an external voltage is applied.

▸ If the pin is output, then the voltage at that pin is fixed to a particular value (5V or 0).

Setting Register Values

▸ DDR is an 8 bit register. Each bit corresponds to a particular pin on the associated port.

▸ For example, the MSB on DDRA corresponds to the pin A7.

DDRA	7	6	5	4	3	2	1	0
	A7	A6	A5	A4	A3	A2	A1	A0
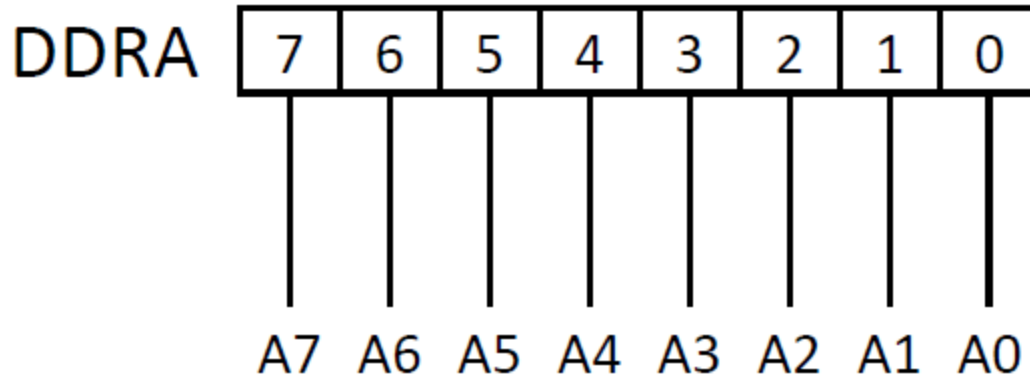
Interpretation of DDR Values

▸ If a bit on the DDR register is 0, then the corresponding pin on the associated port is set as input.

▸ Similarly, if the bit is 1, then the pin is set as output.

▸ Example: if DDRA = 0b10010110, then:

DDRA | OP | IN | IN | OP | IN | OP | OP | IN |

MSB   LSB
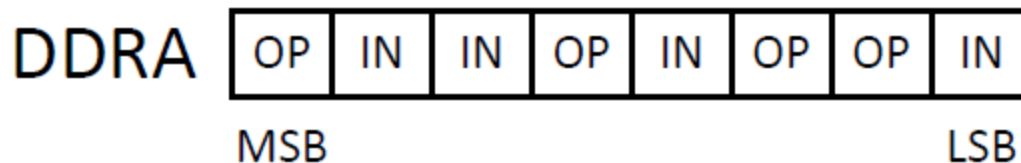
PORT Register

▸ PORT is also an 8 bit register. The bits on the PORT register correspond to the pins of the associated port in the same manner as in the case of the DDR register.

▸ PORT is used to set the output value.

▸ If the pin is set as output, then a PORT value of 1 will set voltage at that pin to 5V. If PORT value is 0, then voltage is set to 0V.

Pull Up / Pull Down

- What if we try to set the PORT value of a pin that is configured as input?

- A separate purpose is served: that of pull up or pull down.

- When an input pin is connected by a wire to some specific voltage, it's voltage also becomes that same value.

Pull Up/ Pull Down

- But, when the input pin is left free, it's voltage value is undecided. This is bad.
- To prevent this, a "default" value is assigned. This value can be either 5V or 0, and is of consequence only when the pin is unconnected.
- The PORT value becomes this "default" value.
- If "default" value is 0, then pin is pulled down. If it is 5V, then it is pulled up.

Pin Register

- PIN is a register whose value can be read, but cannot be changed inside the program.
- It gives the value of the actual voltage at a particular pin. 5V corresponds to 1, and 0 corresponds to 0.

Summary

DDR = 0		DDR = 1	
PORT = 0	PORT = 1	PORT = 0	PORT = 1
Pin is input. If unconnected, **PIN** is 0.	Pin is input. If unconnected, **PIN** is 1.	Pin is output, value is 0. **PIN** is always equal to **PORT**	Pin is output, value is 5V. **PIN** is always equal to **PORT**

Some C concepts

- | is bitwise OR. Eg. 10100111 | 11000101 = 11100111
- & is bitwise AND. Eg. 10100111 & 11000101 = 10000101
- ~ is bitwise NOT. Eg. ~10100110 = 01011001
- << is shift left. >> is shift right.

Simplest C program for a micro-controller

```
int main(){
return 0;
}
```

Example Program 1

```c
#include <avr/io.h>
int main(){
DDRA = 0b11111111; // or 255 or 0xFF
while(1){
PORTA = PINC;
}
return 0;
}
```

Example Program 2

```c
#include <avr/io.h>
#include <util/delay.h>
int main(){
DDRA = 0xFF;
while(1){
PORTA = 0xAA;
_delay_ms(1000);
PORTA = 0x55;
_delay_ms(1000);
}
return 0;
}
```