Introduction to Embedded Systems

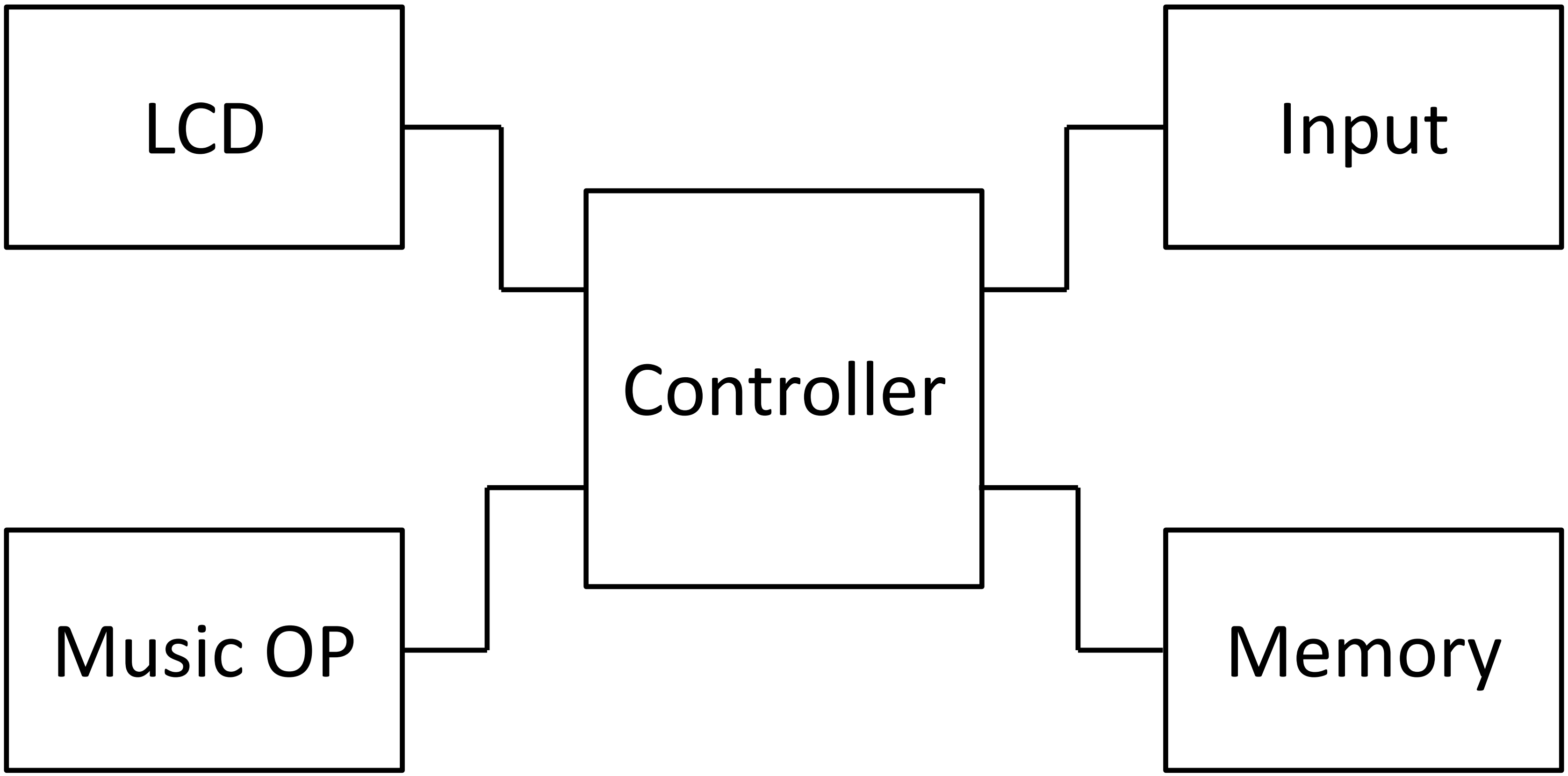
Chirag Sangani

Electronics Club
IIT Kanpur

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 MicroController (μC)

- 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.

The Computer – µC analogy

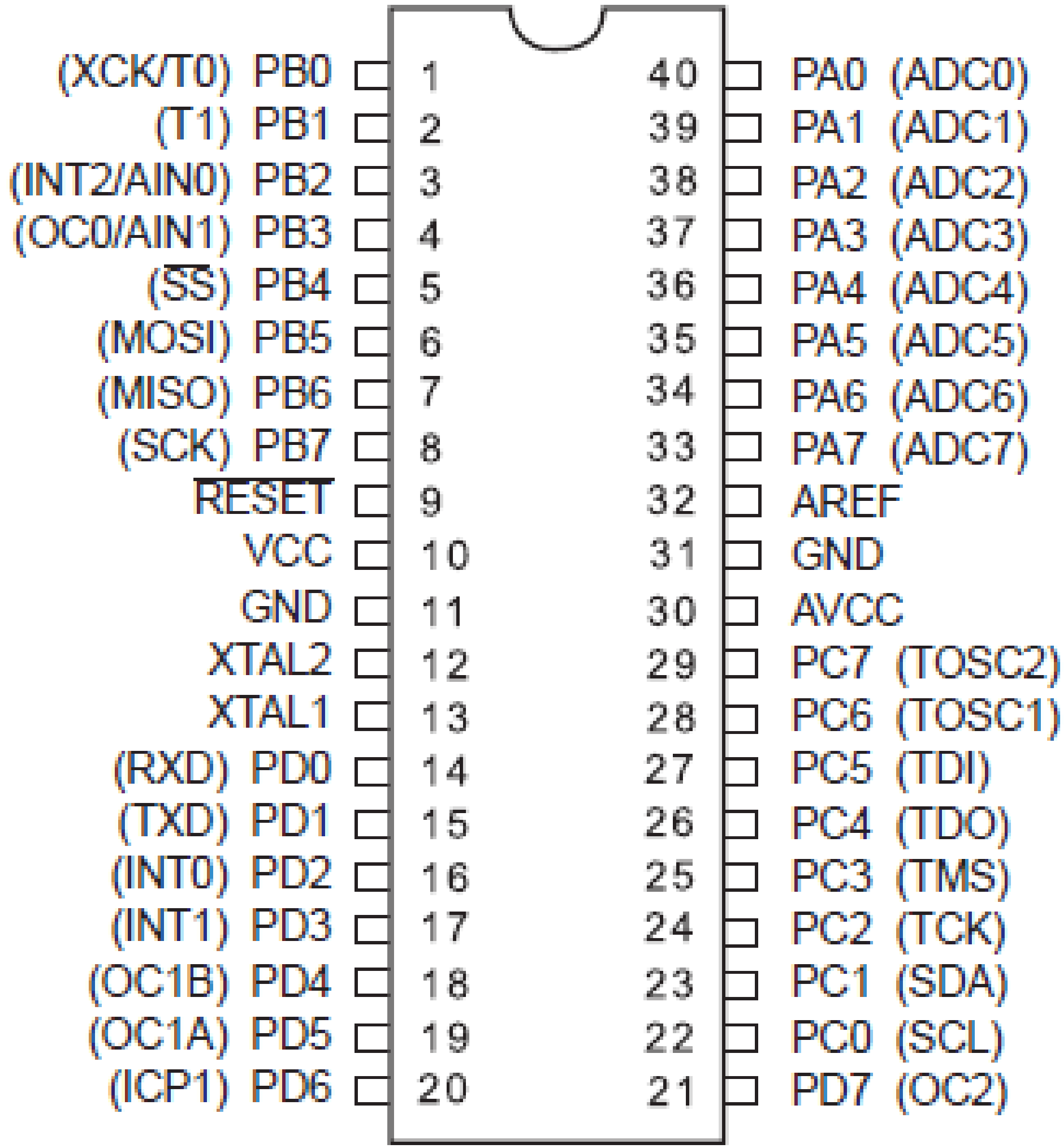
- Inside the CPU, the main components are the processor, RAM, hard disk and I/O.
- The microcontrollers has all the analogous components inside a single IC: the processor core, RAM, EEPROM (hard disk).
- I/O is present in the form of the pins of a microcontroller.

Microcontroller(s)

- 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.

The ATmega16

- 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.



Electronics Club IIT Kanpur

How does a microcontroller work?

- Just like a computer, a microcontroller executes a program.
- After the program is finished, nothing happens.
- The program for a microcontroller is written in C language (although other languages are possible).

Some C operators

- | 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.

Sample C program for a µC

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

I/O

- 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.

Example Register Manipulation

```
#include <…>

int main(){
     int i;
     for(i=0;i<10;i++){
          REG1 = i;
     }
     return 0;
}
```

I/O Registers

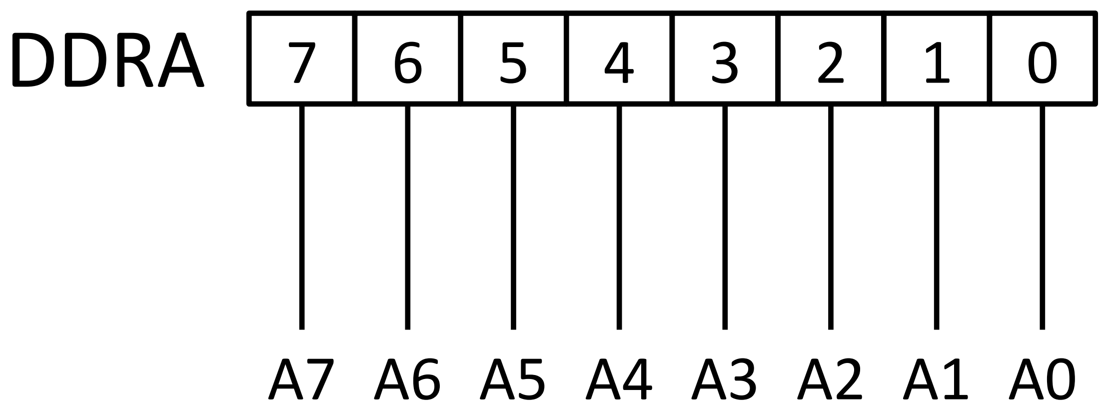
- There are 3 registers that control the I/O pins: **DDR**, **PORT** and **PIN**.
- Each port has it's own registers. Hence, port A has registers **DDRA**, **PORTA**, **PINA**; port B has registers **DDRB**, **PORTB**, **PINB**; and so on.
- **DDR**, **PORT** and **PIN** serve different functions.

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.



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

Electronics Club IIT Kanpur

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 0.

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**

Example Program 1

```
#include <avr/io.h>

int main(){
     DDRA = 0b11111111; // or 255 or 0xFF
     while(1){
          PORTA = PINC;
     }
     return 0;
}
```

Example Program 2

```
#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;
}
```

Thank you