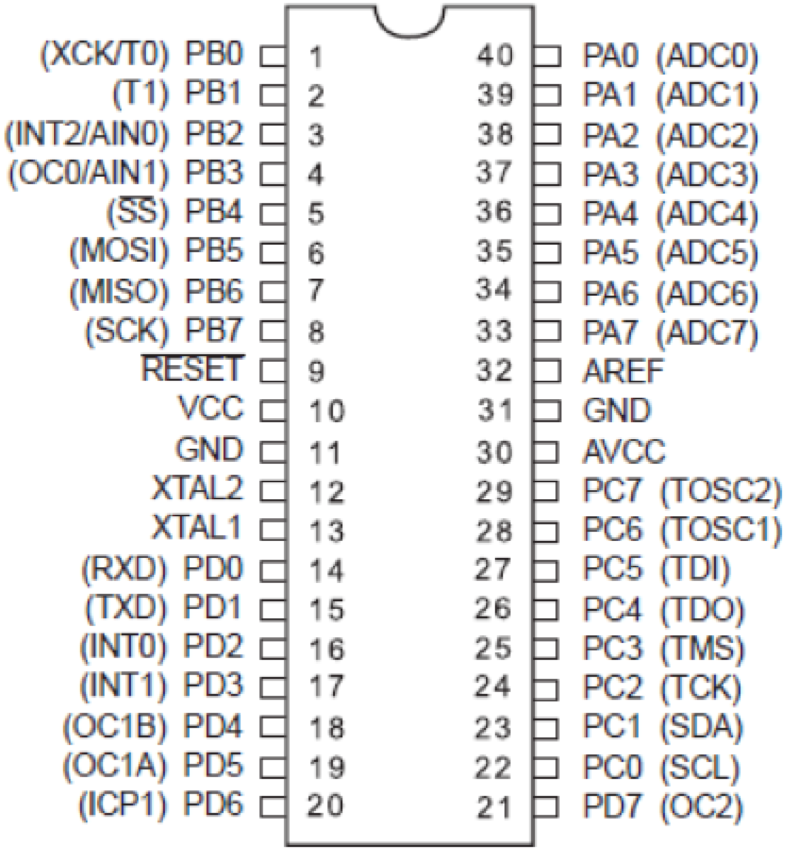
Timers and Interrupts

Anurag Dwivedi

Let Us Revise

Micro-Controllers

- A small computer integrated in a single IC
- Has I/O pins, RAM and Memory
- We Use Atmega 16



Software Used

- CvAvr : Editor and Compiler
- Avr Studio : Used to program the code into the micro-controller

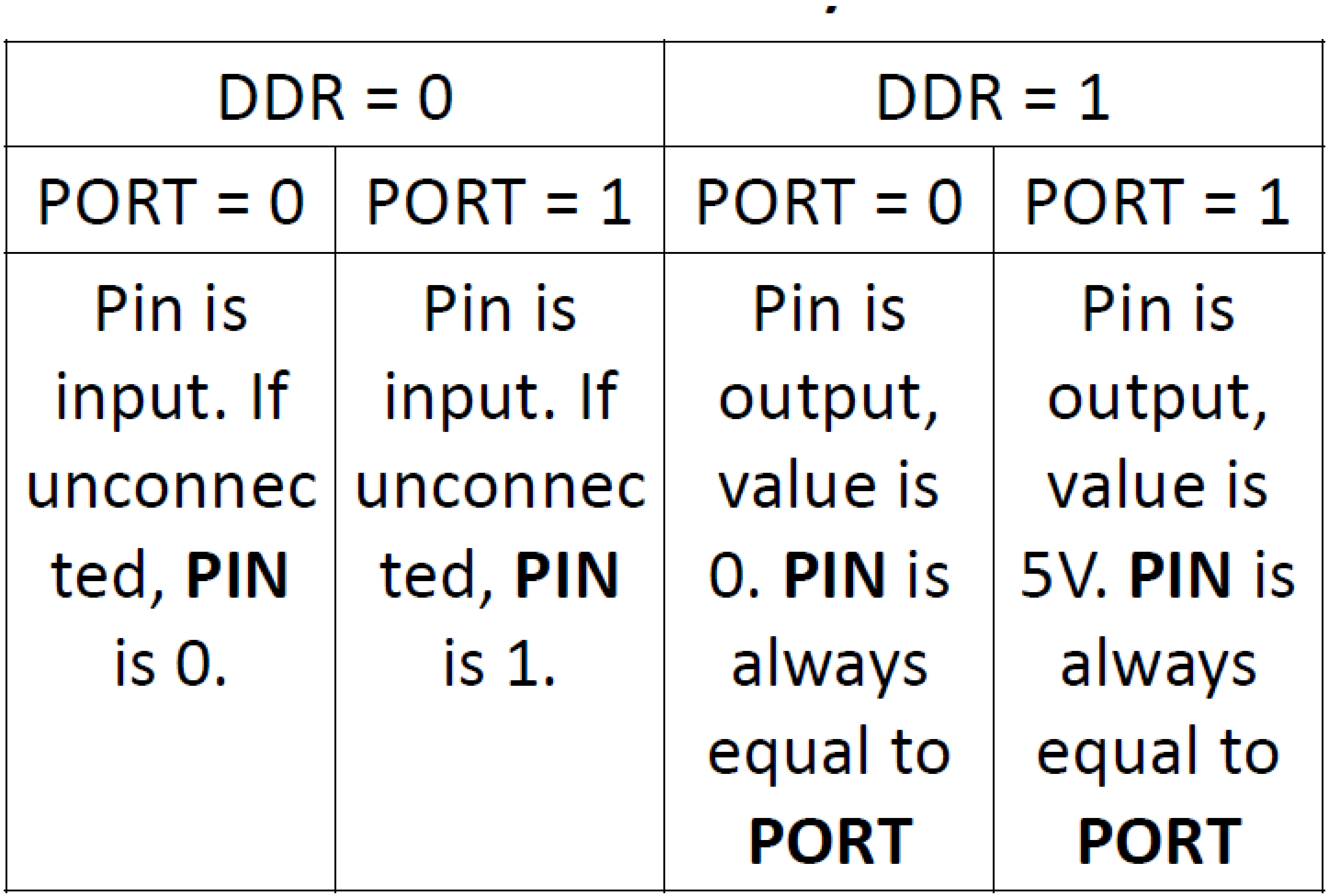
Registers

- Registers are actual hardware memory locations inside the µC.
- What do we mean by this??
- Consider a 8-bit long register. Each bit of the register can be realized as a flip-flop.
- Ex. PORTA is a register.
- When you set the value of PORTA = 0X01, you physically set the corresponding flip-flop a value of +5 Volts.

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 Register : DDR decides whether the pins of a port are input pins or output pins.
- PORT Register : PORT is used to set the output value.
- PIN Register : PIN gives the value of the input.

Summary of Last Lecture



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**

An example program

```
#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;
}
```

Timers

- A Timer is usually a 8-bit register.
- Values starts from 0 and goes up to 255.
- Timer value increases by 1,after each period.
- When the timer reaches its maximum value, in the next cycle, its value becomes 0 again and the process repeats itself.
- The timer frequency can be factors of the base frequency of the MCU.
- This process is independent of the CPU.

Simple Statistics

- Maximum value of timer is n andclock period is t, then:
- 1. Timer period = t
- 2. Timer cycle period = $(n+1)\times t$
- 3. Frequency of timer (f) = $1/t$
- 4. Frequency of timer cycle = $1/(n+1)\times t$

Interrupts

- Interrupts means causing a break in a continuing process.

Why Interrupts ??

- Suppose you need to check for a condition A while running another condition B

- Simple Solution..
- while(1){
- ---- -> if (Event A == true)
- ---- -> // print event A has occurred
- ----
- ----
- ---- -> Event B
- ---- -> Suppose Event A happens here
- ----
- }

- Do you see the problem in this approach??

A Better Solution

- .
- .
- while(1){
- ---
- ---
- EVENT B
- ---
- ---
- }
- .

- handleA(){
- .
- //print A
- }

We execute the event B in a normal way, in a while(1) loop.

We consider the occurrence of event A as an interrupt.

It means that whenever an interrupt (event A) occurs the code stops and a function is called …..

We execute the required code in the Handler of A

More on Interrupts

- Interrupts are special events that can “interrupt” the normal flow of a program.
- Whenever an Interrupt is called, the processor stops the normal program, handles the interrupt, and then resumes its normal work.
- There are two types of interrupts: External and Internal

External Interrupts

- The controller monitors the input at the special pins INT0 and INT1, whenever external interrupt is set on.
- We can configure the program to call an external interrupt whenever any of the following conditions are met.
- Rising Edge
- Falling Edge
- Any change

Internal Interrupts

- The internal interrupts are called when different specific conditions are met by the timer value.
- This brings us to the next topic..

Timers and Interrupts

- Timers can generate certain interrupts: two, to be precise.
- These are called OVERFLOW interrupt and COMPARE MATCH interrupt.

Overflow Interrupts

- An overflow interrupt is generated when the timer exceeds its maximum value and resets to 0
- The interrupt may or may not have a handler.
- In either case, the timer continues to run; remember: timers are independent of the CPU.

Overflow Statistics

- Suppose a timer of maximum value n has a time period t (also called as clock period).

Then :

- 1. Timer cycle frequency = $1/(n+1)\times t$
- 2. OVERFLOW interrupt frequency = $1/(n+1)\times t$

- If OVERFLOW interrupt is enabled, then aninterrupt is generated in every cycle.

Compare Match Interrupts

- A compare match interrupt is called when the value of the timer equals a specific value, set by the user.
- This value is set by setting the value of OCR register.
- Before incrementing, the value of the timer is compared to OCR. If the two are equal, a COMPARE MATCH interrupt is generated

Compare Match Statistics

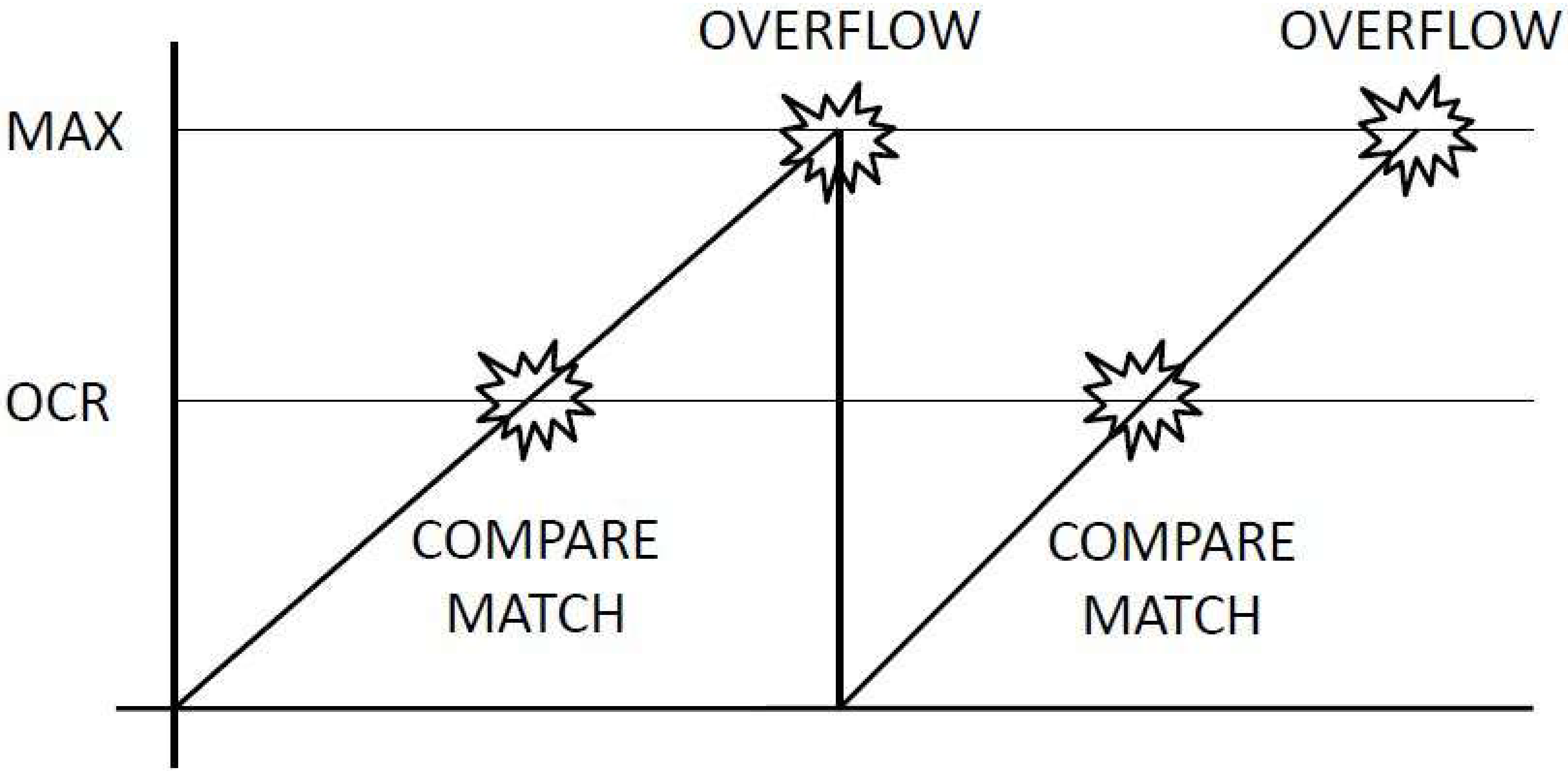
- Suppose a timer of maximum value n has a time period t (also called as clock period).

Then :

- 1. Timer cycle frequency = $1/(n+1) \times t$
- 2. COMPARE MATCH interrupt frequency = $1/(n+1) \times t$

- If COMPARE MATCH interrupt is enabled, then an interrupt is generated in every cycle.

Interrupts – Overflow and Compare Match



Timer Modes

- A timer works in three modes: Normal, CTC and PWM.
- All three modes differ in the response of the controller to the interrupts generated.

Normal Mode

- Standard mode: Timer starts at 0, goes to maximum value and then resets itself.
- OVERFLOW and COMPARE MATCH interrupts generated as normal.
- The timer mode used so far in this presentation is normal mode.

CTC Mode

- Known as Clear Timer on Compare.
- As evident by the name, the timer starts at 0 as usual, but instead of resetting after maximum value, it resets after reaching value specified in OCR register.
- Compare match interrupt if enabled will be generated but not overflow interrupt (Why?)

CTC Mode Statistics

- If clock time period is t:
- 1. Timer cycle time period = $(OCR+1)\times t$
- 2. Frequency = 1/$(OCR+1)\times t$

- With the use of CTC Mode we can theoretically generate any frequency up to 8 MHz.

- Example of 1 Hz generation.

Questions …