Mechatronics Engineering

Lab 3

Timers in ATmega328P

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

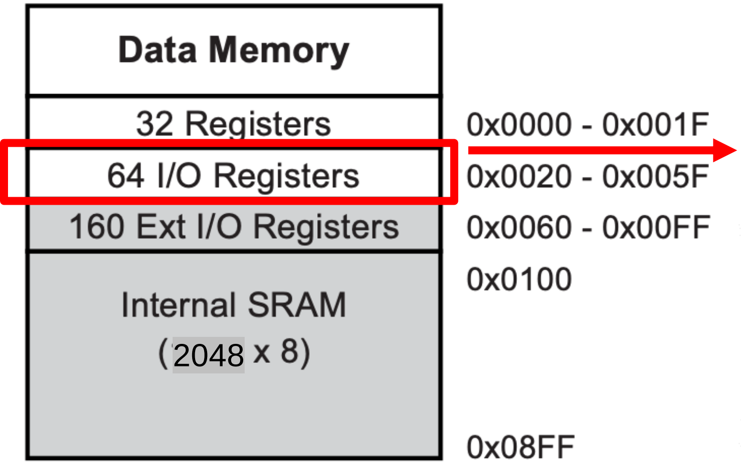
- Recap
  - Input-Output Configuration Registers in ATmega328P
- Timers in ATmega328P
- Lab 3 Validation

Tutorial Contents

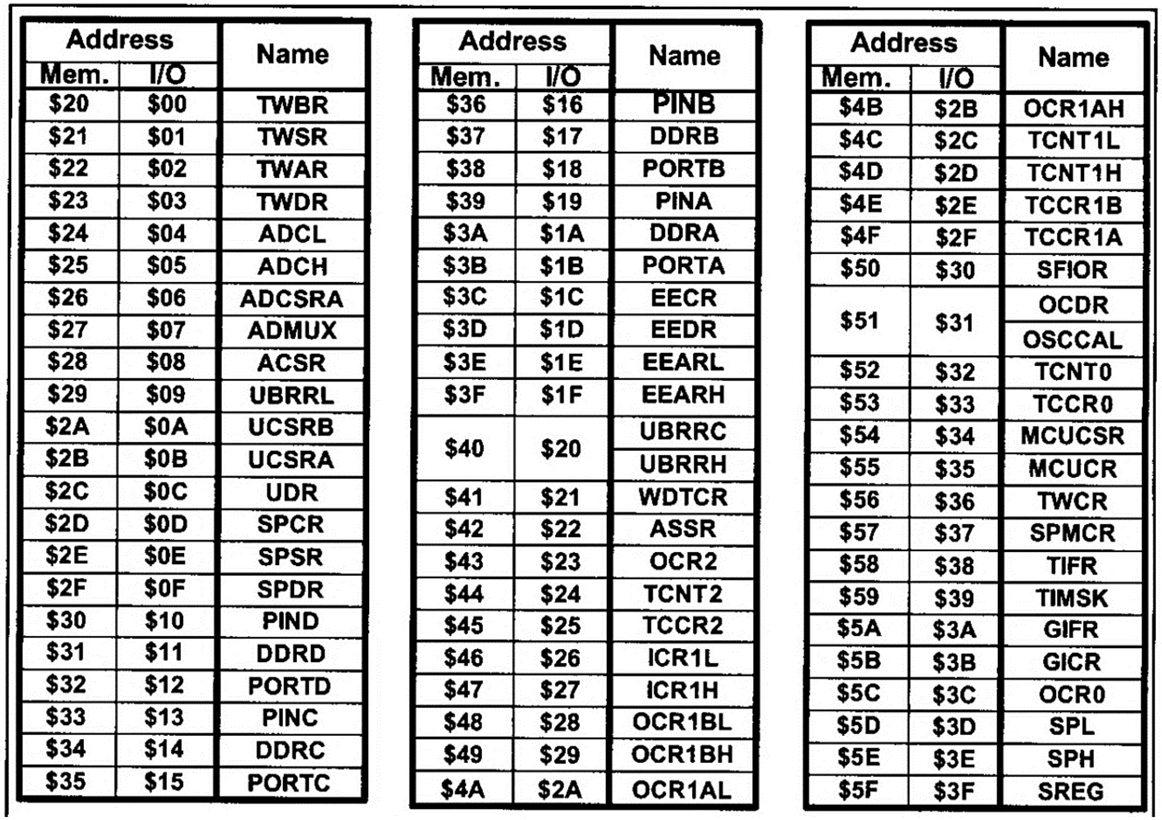
- **Recap**
  - **Input-Output Configuration Registers in ATmega328P**
- Timers in ATmega328P
- Lab 3 Validation

ATmega328P Memory

Data Memory:



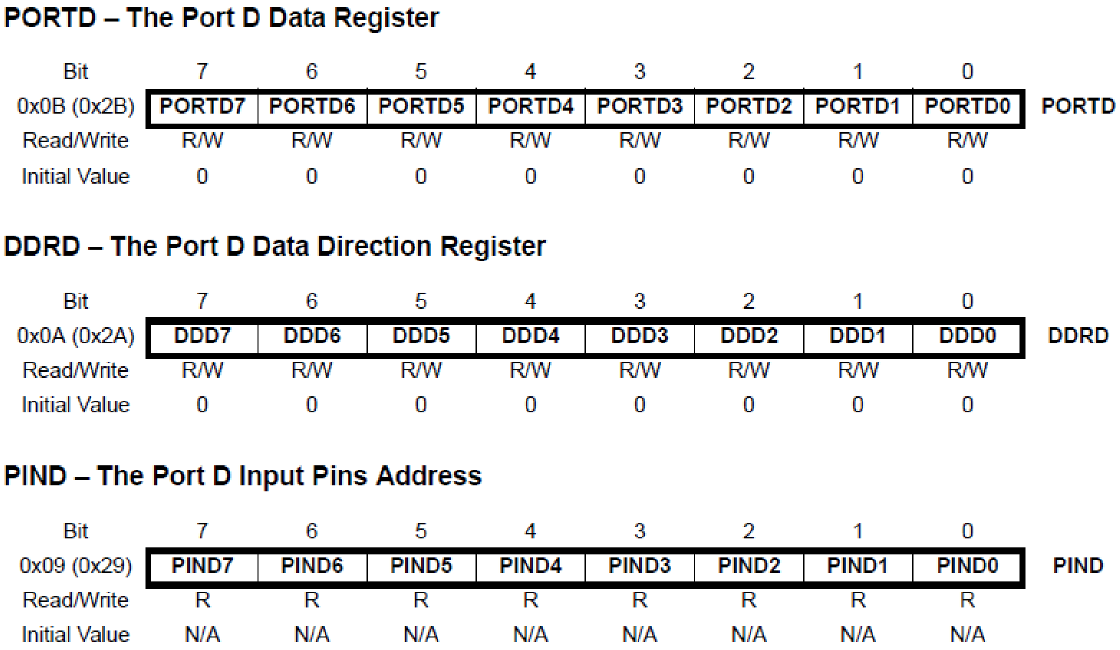
Data Memory	
32 Registers	0x0000 - 0x001F
64 I/O Registers	0x0020 - 0x005F
160 Ext I/O Registers	0x0060 - 0x00FF
Internal SRAM (2048 x 8)	0x0100 0x08FF



Address		Name
Mem.	I/O	
$20	$00	TWBR
$21	$01	TWSR
$22	$02	TWAR
$23	$03	TWDR
$24	$04	ADCL
$25	$05	ADCH
$26	$06	ADCSRA
$27	$07	ADMUX
$28	$08	ACSR
$29	$09	UBRRL
$2A	$0A	UCSRB
$2B	$0B	UCSRA
$2C	$0C	UDR
$2D	$0D	SPCR
$2E	$0E	SPSR
$2F	$0F	SPDR
$30	$10	PIND
$31	$11	DDRD
$32	$12	PORTD
$33	$13	PINC
$34	$14	DDRC
$35	$15	PORTC
$36	$16	PINB
$37	$17	DDRB
$38	$18	PORTB
$39	$19	PINA
$3A	$1A	DDRA
$3B	$1B	PORTA
$3C	$1C	EECR
$3D	$1D	EEDR
$3E	$1E	EEARL
$3F	$1F	EEARH
$40	$20	UBRRC UBRRH
$41	$21	WDTCR
$42	$22	ASSR
$43	$23	OCR2
$44	$24	TCNT2
$45	$25	TCCR2
$46	$26	ICR1L
$47	$27	ICR1H
$48	$28	OCR1BL
$49	$29	OCR1BH
$4A	$2A	OCR1AL
$4B	$2B	OCR1AH
$4C	$2C	TCNT1L
$4D	$2D	TCNT1H
$4E	$2E	TCCR1B
$4F	$2F	TCCR1A
$50	$30	SFIOR
$51	$31	OCDR OSCCAL
$52	$32	TCNT0
$53	$33	TCCR0
$54	$34	MCUCSR
$55	$35	MCUCR
$56	$36	TWCR
$57	$37	SPMCR
$58	$38	TIFR
$59	$39	TIMSK
$5A	$3A	GIFR
$5B	$3B	GICR
$5C	$3C	OCR0
$5D	$3D	SPL
$5E	$3E	SPH
$5F	$3F	SREG

Input-Output Configuration Registers in ATmega328P

- For example, for PORTD:



PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

When specifiying a pin as input, we check pins in PINx register.

DDRDx	PORTDx	Configuration
0	0	Input
0	1	Input with pull-up resistor enabled
1	0	Output (LOW)
1	1	Output (HIGH)

Tutorial Contents

- Recap
  - Input-Output Configuration Registers in ATmega328P
- **Timers in ATmega328P**
- Lab 3 Validation

Timers in ATmega328P

- The ATmega328P has 3 timers referred **Timer 0**,**Timer 1** and **Timer 2**. Timer 0 and Timer 2 are **8-bit timers** while Timer 1 is **16-bit**. These Timers can be used as counters or timers.

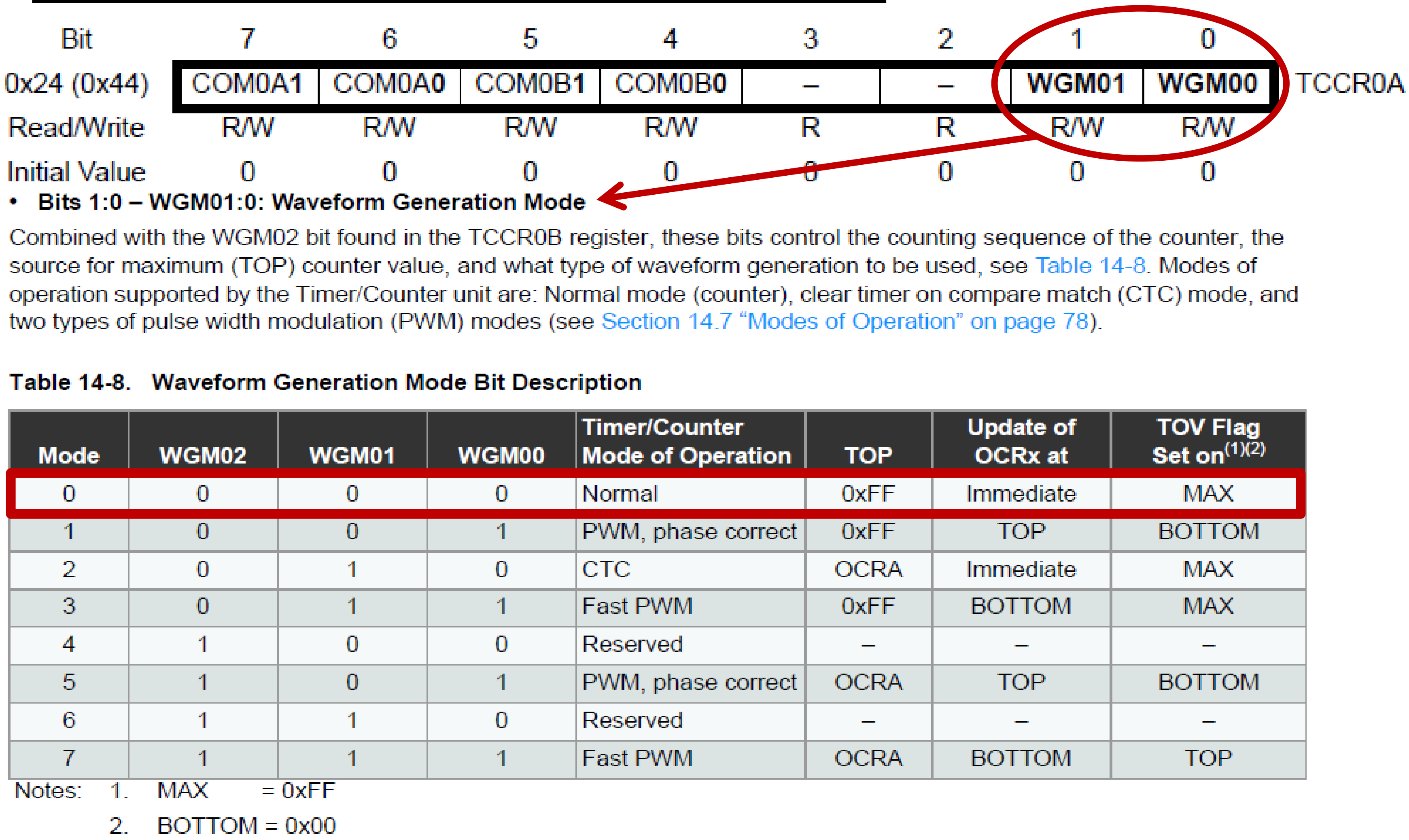
- We can use Timer 0 as **counter** that counts using input signal from pin **T0**.

- Timer 0 can also be used as a **timer** and is incremented on every timer **clock cycle.** It counts from 0 to 255. The TCNT0 register holds the timer count. If the timer is turned on it ticks from 0 to 255 and overflows. When it overflows, a Timer Overflow Flag(TOV) is set. You can as well load a count value in TCNT0 and start the timer from a specific count.

- Registers Associated with Timer 0:
  - TCNT0: Timer/Counter 0 Register
  - TCCR0A: Timer/Counter 0 Control Register A
  - TCCR0B: Timer/Counter 0 Control Register B
  - TIMSK0: Timer/Counter 0 Interrupt Mask Register
  - TIFR0: Timer/Counter 0 Interrupt Flag Register

Timers in ATmega328P

- **TCCR0A (Timer/Counter 0 Control Register A):**



Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

Combined with the WGM02 bit found in the TCCR0B register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used, see Table 14-8. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), clear timer on compare match (CTC) mode, and two types of pulse width modulation (PWM) modes (see Section 14.7 "Modes of Operation" on page 78).

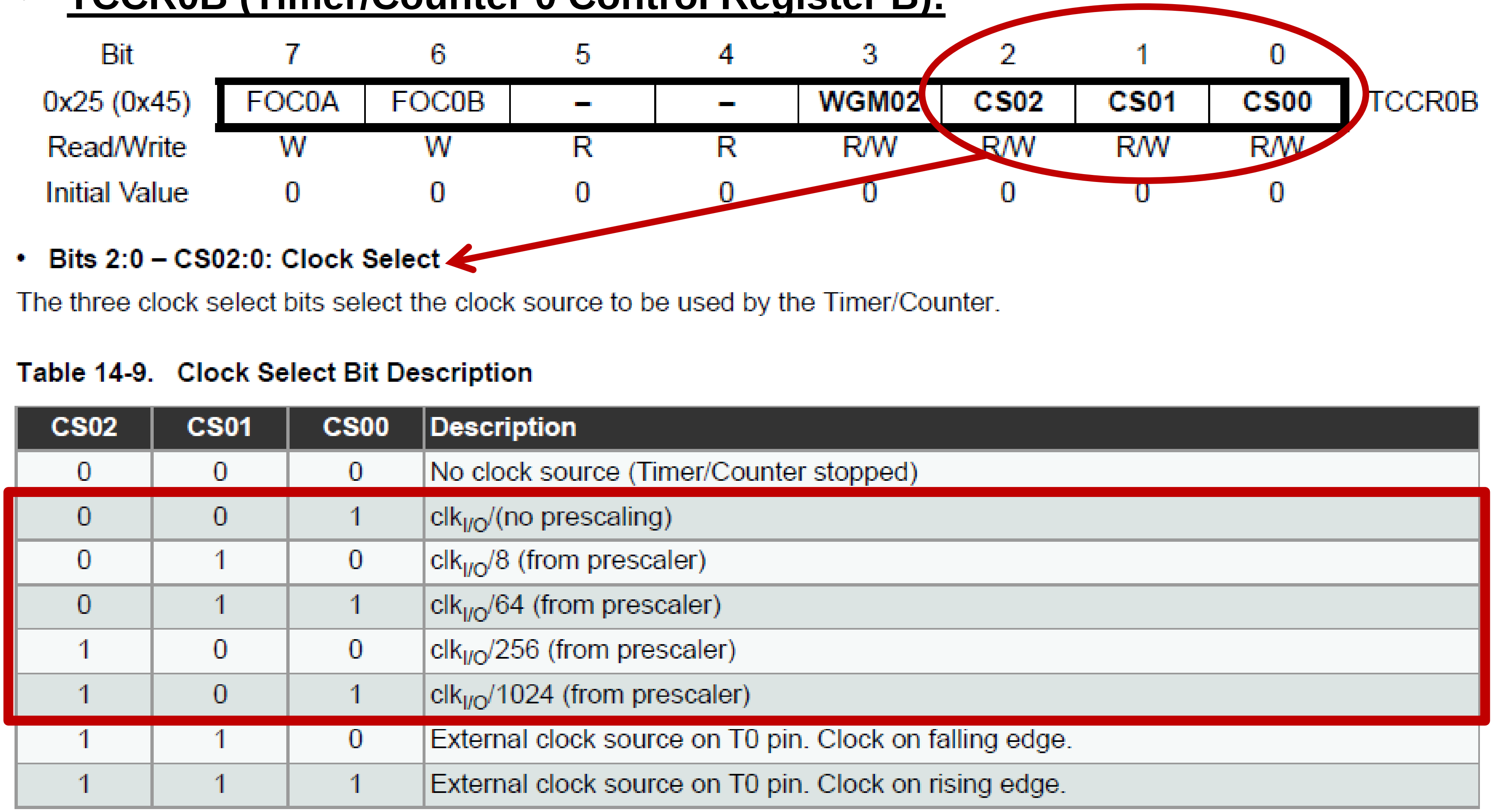
Table 14-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, phase correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, phase correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00

Timers in ATmega328P

- **TCCR0B (Timer/Counter 0 Control Register B):**



Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 2:0 – CS02:0: Clock Select**

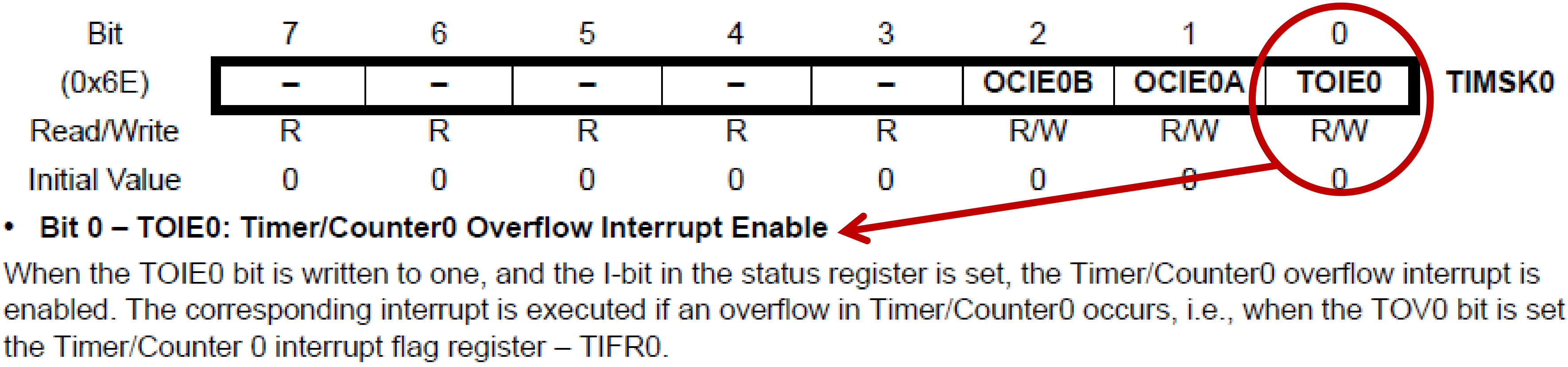
The three clock select bits select the clock source to be used by the Timer/Counter.

Table 14-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(no prescaling)
0	1	0	$clk_{I/O}$/8 (from prescaler)
0	1	1	$clk_{I/O}$/64 (from prescaler)
1	0	0	$clk_{I/O}$/256 (from prescaler)
1	0	1	$clk_{I/O}$/1024 (from prescaler)
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.

Timers in ATmega328P

- **TIMSK0 (Timer/Counter 0 Interrupt Mask Register):**

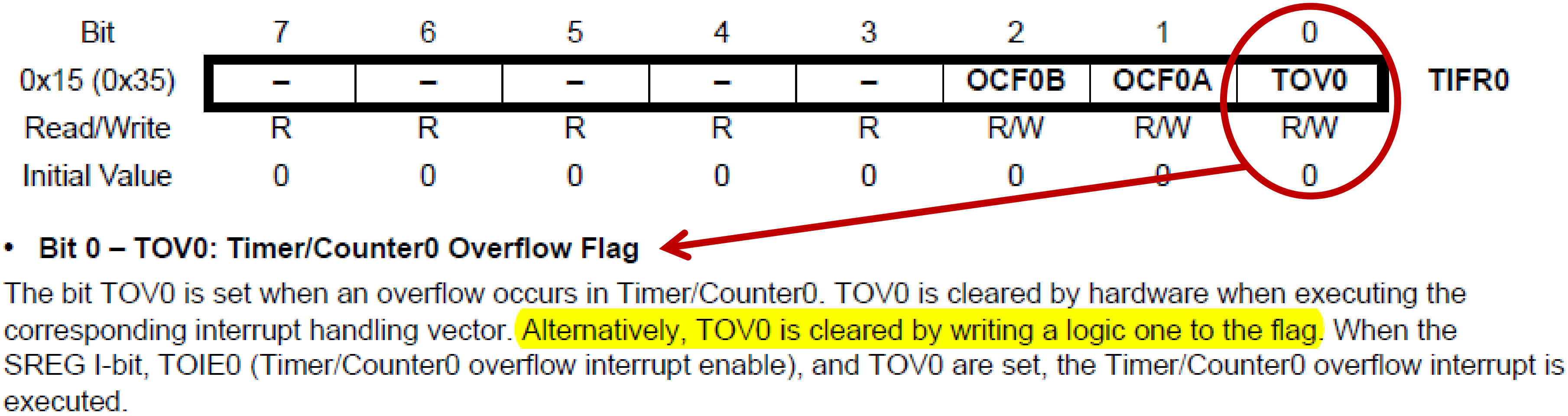


Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the status register is set, the Timer/Counter0 overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 interrupt flag register – TIFR0.

- **TIFR0 (Timer/Counter 0 Interrupt Flag Register):**



Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – TOV0: Timer/Counter0 Overflow Flag**

The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 overflow interrupt enable), and TOV0 are set, the Timer/Counter0 overflow interrupt is executed.

Timers in ATmega328P

- Thus, the overflow time can be expressed as:

$$Overflow\ Time = \frac{1}{f_{osc}} \times Prescaler \times (256 - T_0)$$

where T_0 is the initial value of Timer 0.

- The ATMega328P is connected to a 16 MHz oscillator.

- The maximum overflow time you can get from Timer 0 is when the $T_0 = 0$ and $Prescaler = 1024$.

$$Overflow\ Time = \frac{1}{16 \times 10^6} \times 1024 \times (256 - 0) = 16.38\ ms$$

- **Example**: If you want to get a 5 ms delay from Timer 0, how can you set it?
  - ➢ Choose a pre-scaler of $Prescaler = 1024$.
  - ➢ Find the initial value of Timer 0 we should start counting from.

$$5 \times 10^{-3} = \frac{1}{16 \times 10^6} \times 1024 \times (256 - T_0)$$

$$\rightarrow T_0 = 177.875 \approx 177$$

Timers in ATmega328P

In-Lab Project 1:

It is required to build a circuit that produces a 500KHz square wave signal.

Circuit Operation:

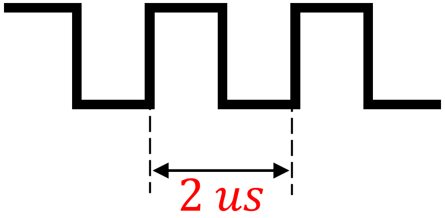
- The circuit should generate a 500 KHz square wave signal on Pin PD2

Design Requirements:

- Use the AVR embedded board.
- Use Embedded C for programming.
- The output indicator is connected to PD2 (PORTD pin 2).
- Do not use pre-defined delay functions.
- Use Timer 0 in your implementation.
- Include a reset push button in your hardware implementation.

Timers in ATmega328P

In-Lab Project 1: Solution:



$$T = \frac{1}{500*10^3\ Hz} = 2*10^{-6}s$$

1. Choose the settings of Timer 0.
   - We will use Timer 0 in the Normal mode.
   - Full period is 2 us.
   - So, we need Timer 0 to overflow after half the period (1us).

     $$Overflow\ time\ with\ no\ prescaling = \frac{1}{16*10^6}*(256-0) = 16\ us$$

   - So, we can make Timer 0 to start counting from another value x.

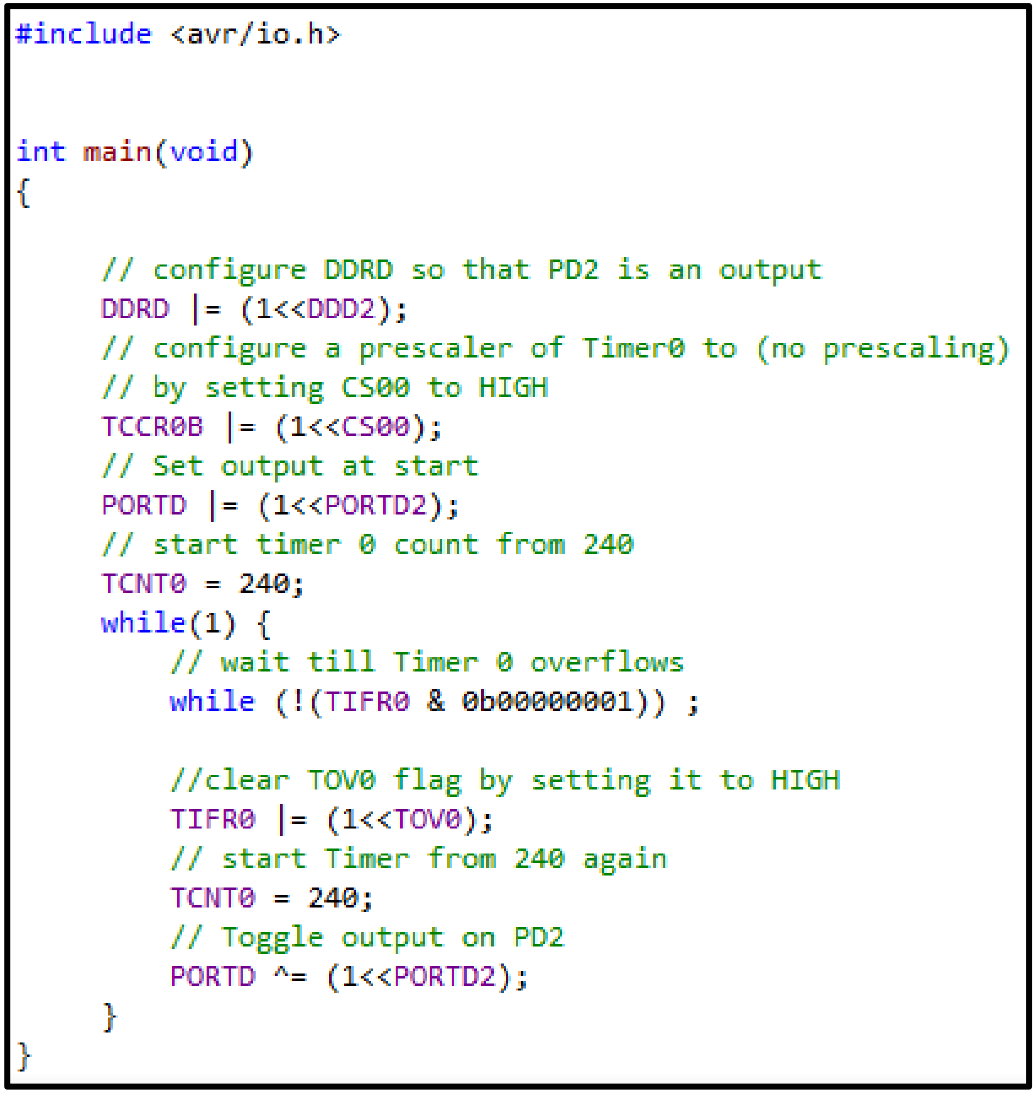
     $$Overflow\ time = 1\ us = \frac{1}{16*10^6}*(256-x)$$

   - Solving for $x = 240$
   - So we need to put the value of x in TCNT0 and wait till TCNT0 overflows.

Timers in ATmega328P

In-Lab Project 1: Solution:

- **Bitwise Logical XOR (^):**
  - Example:
    00110101 ^ 00001111 = 00111010
  - Example: byte ^ 00000000 = byte
  - Example: byte ^ 111111111 = !byte
- **Toggling bit 2 in PORTD:**
  - PORTD ^= (1<<2)



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


int main(void)
{

    // configure DDRD so that PD2 is an output
    DDRD |= (1<<DDD2);
    // configure a prescaler of Timer0 to (no prescaling)
    // by setting CS00 to HIGH
    TCCR0B |= (1<<CS00);
    // Set output at start
    PORTD |= (1<<PORTD2);
    // start timer 0 count from 240
    TCNT0 = 240;
    while(1) {
        // wait till Timer 0 overflows
        while (!(TIFR0 & 0b00000001)) ;

        //clear TOV0 flag by setting it to HIGH
        TIFR0 |= (1<<TOV0);
        // start Timer from 240 again
        TCNT0 = 240;
        // Toggle output on PD2
        PORTD ^= (1<<PORTD2);
    }
}
```

Timers in ATmega328P

In-Lab Project 2:

It is required to build a circuit with a LED indicator.

Circuit Operation:

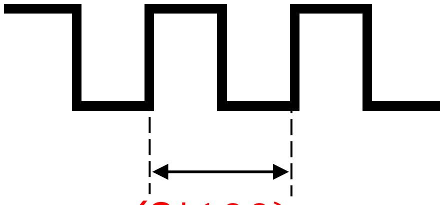
- The LED indicator should blink every 100 milliseconds (i.e the LED indicator toggles every 100 milliseconds).

Design Requirements:

- Use the AVR embedded board.
- Use Embedded C for programming.
- The LED indicator is connected to PD2 (PORTD pin 2).
- Do not use pre-defined delay functions.
- Use Timer 0 in your implementation.
- Include a reset push button in your hardware implementation.

Timers in ATmega328P

In-Lab Project 2: Solution:



$$\mathrm{f} = \frac{1}{2*100*10^{-3}\ s} = 5\ \mathrm{Hz}$$

1. Choose the settings of Timer 0.
   - We will use Timer 0 in the Normal mode.

$$Overflow\ Time = \frac{1}{16\times10^{6}} \times 1024 \times (256 - 0) = 16.38\ ms$$

   - Delay time required is 100ms which is greater than 16.38 ms - the maximum overflow time we can get from Timer 0.

   - So, we can let Timer 0 overflow several times till we get the required delay time of 100 ms.

$$Number\ of\ Overflows = \frac{100}{16.38} \approx 6\ times$$

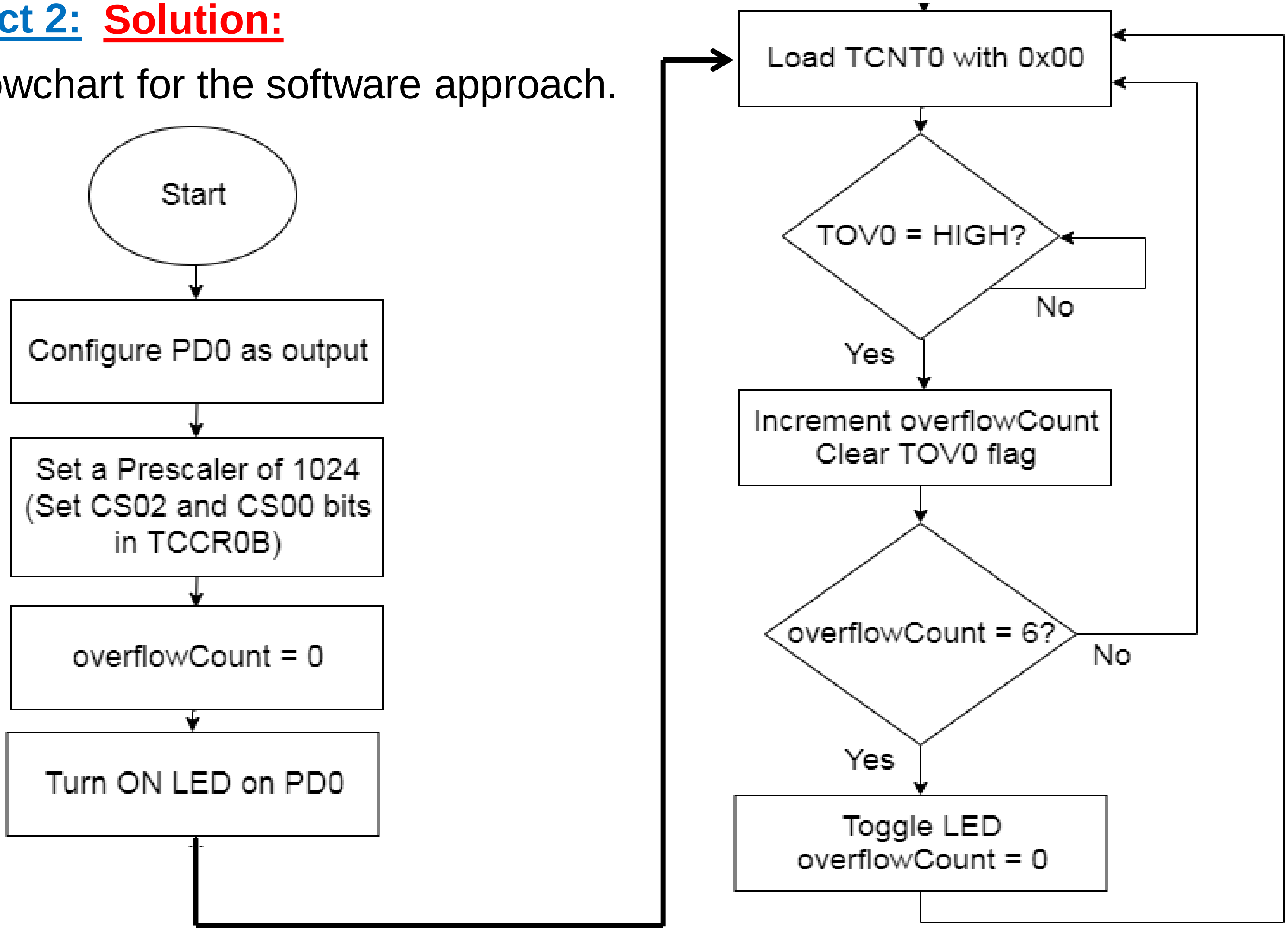
   - So, we can make Timer 0 count from 0 till 255 and overflows 6 times.

   - This is with a pre-scaler of 1024. So, we need to set bits CS02 and CS01 in TCCR0B register.

Timers in ATmega328P

In-Lab Project 2: Solution:

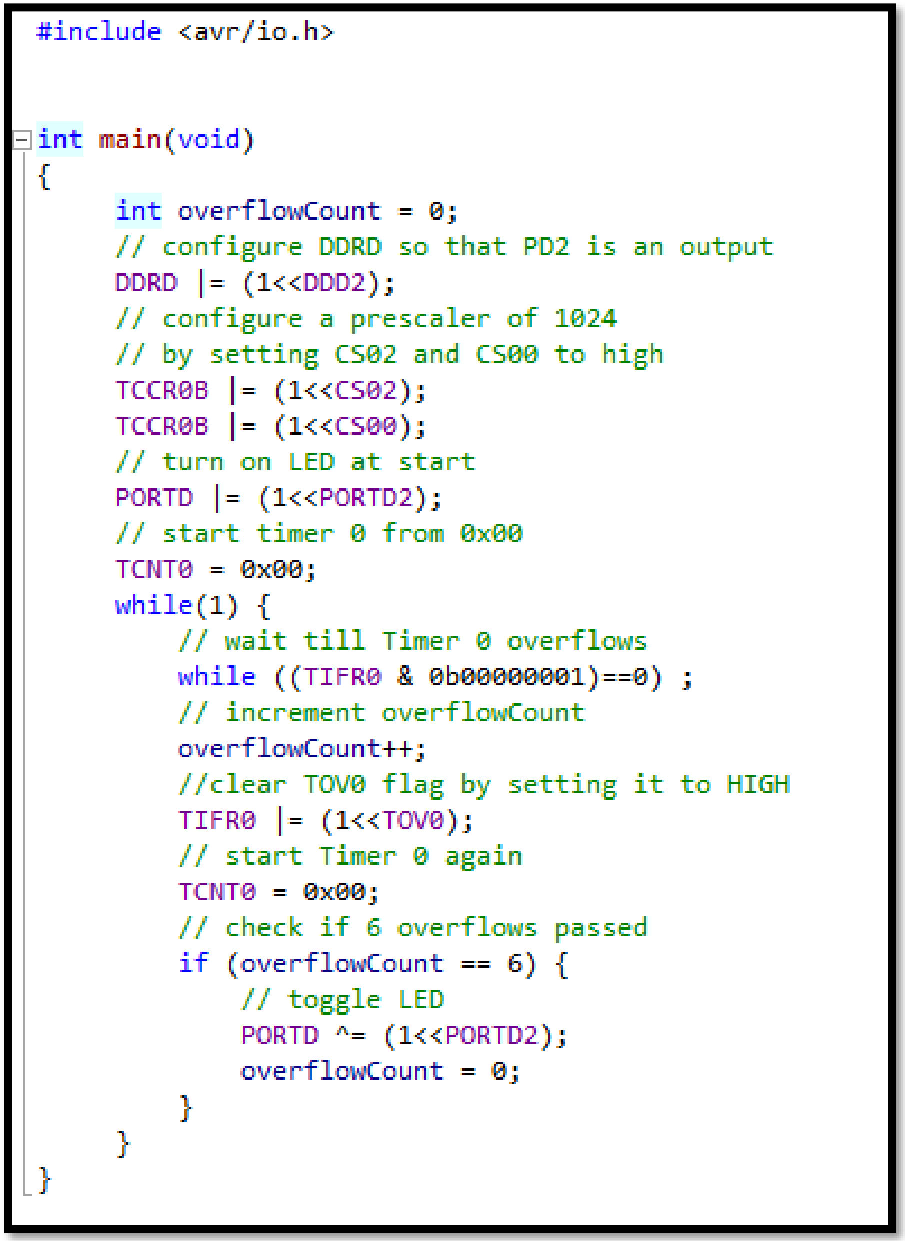
2. Make a flowchart for the software approach.



Timers in ATmega328P

In-Lab Project 2: Solution:

- **Bitwise Logical XOR (^):**
  - Example:
    00110101 ^ 00001111 = 00111010
  - Example: byte ^ 00000000 = byte
  - Example: byte ^ 111111111 = !byte
- **Toggling bit 2 in PORTD:**
  - PORTD ^= (1<<2)

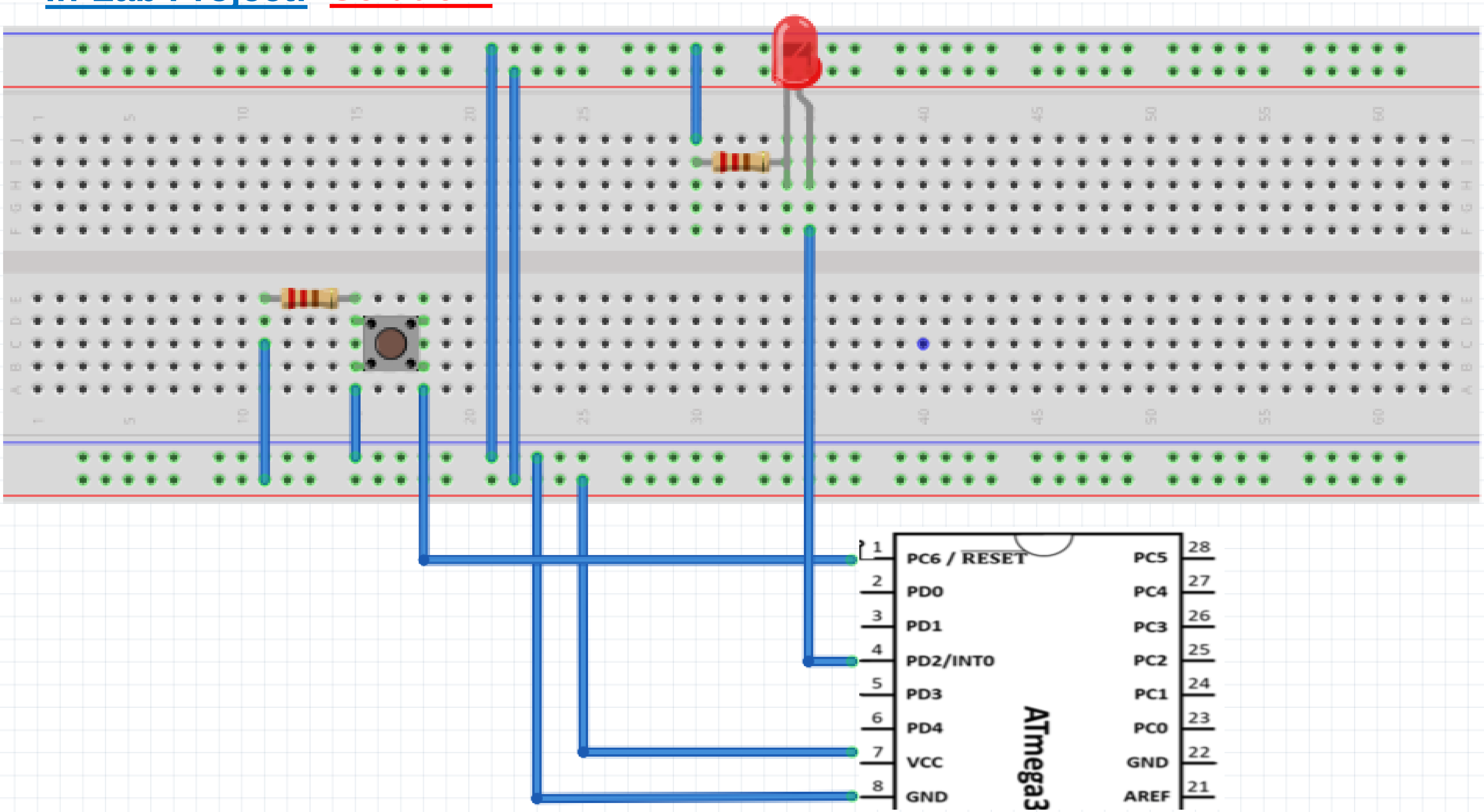


```c
#include <avr/io.h>


int main(void)
{
	int overflowCount = 0;
	// configure DDRD so that PD2 is an output
	DDRD |= (1<<DDD2);
	// configure a prescaler of 1024
	// by setting CS02 and CS00 to high
	TCCR0B |= (1<<CS02);
	TCCR0B |= (1<<CS00);
	// turn on LED at start
	PORTD |= (1<<PORTD2);
	// start timer 0 from 0x00
	TCNT0 = 0x00;
	while(1) {
		// wait till Timer 0 overflows
		while ((TIFR0 & 0b00000001)==0) ;
		// increment overflowCount
		overflowCount++;
		//clear TOV0 flag by setting it to HIGH
		TIFR0 |= (1<<TOV0);
		// start Timer 0 again
		TCNT0 = 0x00;
		// check if 6 overflows passed
		if (overflowCount == 6) {
			// toggle LED
			PORTD ^= (1<<PORTD2);
			overflowCount = 0;
		}
	}
}
```

Timers in ATmega328P

In-Lab Project: Solution:



Tutorial Contents

- Recap
    - Input-Output Configuration Registers in ATmega328P
- Timers in ATmega328P
- **Lab 3 Validation**

Timers in ATmega328P

Lab 2 Validation: (to be submitted in your Lab next week)

It is required to build a circuit with two push buttons and one LED indicator.

Circuit Operation:

- When the user presses the first push button, the LED indicator blinks (i.e. toggles) every 100 milliseconds.
- When the user presses the second push button, the LED indicator blinks (i.e. toggles) every 500 milliseconds.
- This never changes during operation. The user has to press the reset button to change the LED blinking frequency.

Design Requirements:

- Use the AVR embedded board.
- Use Embedded C for programming.
- The two push buttons are connected to PD2 and PD3.
- The LED indicator is connected to PB0.
- Do not use pre-defined delay functions.
- Use Timer 0 in your implementation.
- Include a reset push button in your hardware implementation.

Recommended: Use Timer 1 instead of Timer 0.