Experiment – 5

Architecture and Programming of Timer/Counter (TC) Module of ATmega328 Microcontroller using Arduino UNO Platform

Perform every step of this experiment. When a step is done, check it (√) with a wood pencil. If problem, ask Lab Teacher for help.

5.1 Introduction

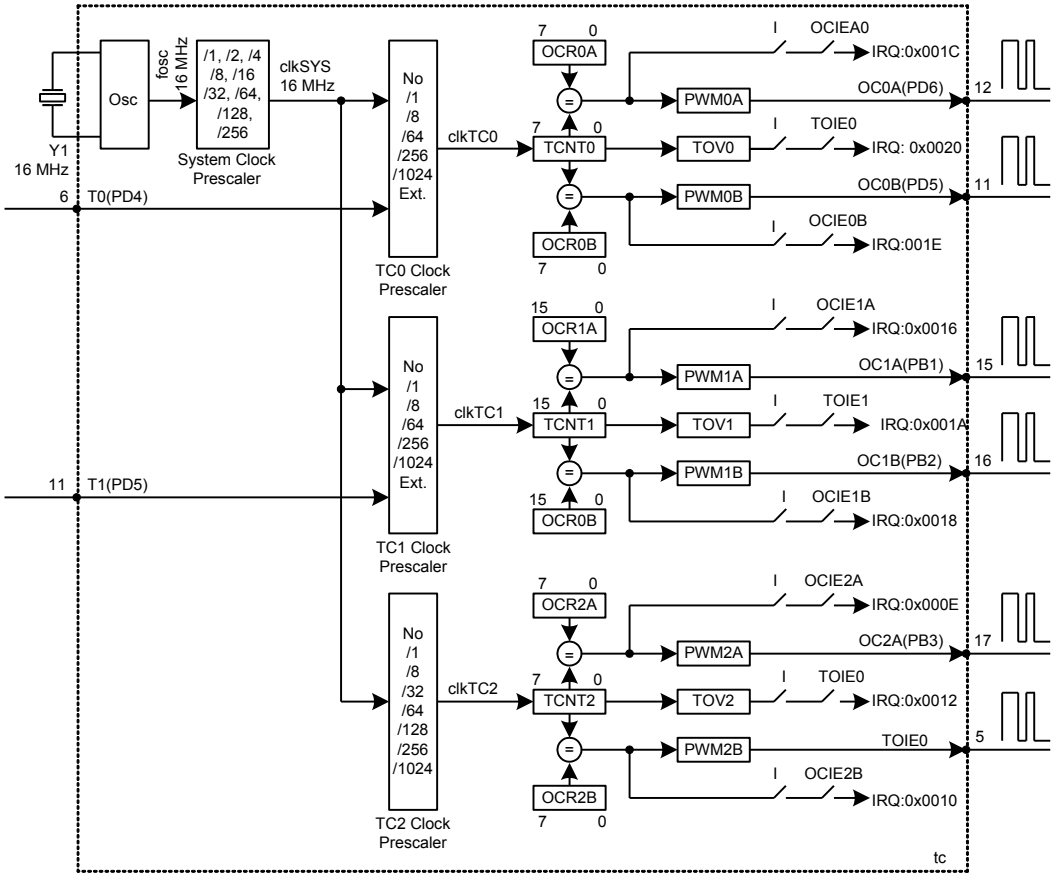

Figure-5.1: *Simplified block diagram for the internal resources of Timer/Counter (TC) module of ATmega328*

(1) There are there Timer/Counter Modules inside the ATmega328 Microcontroller. They are known as: TC0, TC1, and TC2.
(2) TC) stands for Timer-1/Counter-1 (T0/C0 or TC0); similarly, we have TC1 and TC2.
(3) A TC Module (say, TC0) is said to be working as Timer Module (T0) when it receives the clocking pulses (clkTC0) from the internal oscillator via System Clock Prescaler and TC0 Clock Prescaler.
(4) A TC Module (say, TC0) is said to be working as Counter Module (C0) when it receives the clocking pulses (clkTC0) directly from external source via Pin-6.
(5) The TC2 has no capability to receive clocking pulse from external source.
(6) The TC0 is an 8-bit Register, TC1 is a 16-bit Register, and TC2 is an 8-bit Register.

(6) Functional behavior of TC0 Module

(a) TC0 collectively refers to all these resources: TC0 Clock Prescaler, TCNT0 Register, OCR0A Compare Register, OCR0B, PWM0A Pulse Width Modulated Wave Generator, PWM0B, TOV0 Timer/Counter-0 Overflow Flag, and Interrupt Related Software Switches.

(b) The TC0 can be configured to work as a 'Normal Pre-settable Binary Up-counter'. In this mode, the TCNT0 (pronounced as: tee-che-en-tee-zer0=Timer/Counter-0 Register) keeps up-counting from a pre-settable value. After sometimes, it reaches at the full count (0xFF) and when the next clocking pulse arrives, the TCNT0 rolls-over from all 1s to all 0s. This event of roll-over in known as overflow condition; s a result, the TOV0 flag assumes LH state.

The user program may continuously monitor the value (known as polling) TOV0 bit in order to catch the overflow event and to take action as per schedule prepared during initialization.

If interrupt logic are made active (I and TOV0 switches) during initialization. The TOV0 flag can interrupt the MCU to notify the event of the occurrence of roll-over event.

(c) The TC0 module can also be configure to generate two pulse width modulated (PWM) signals.
(d) The TC0 can also be programmed to generate normal square wave signals.
(e) The TC0 can also be programmed to generate many other timing functions which are described in the Atmel data sheets.

(7) The functional behavior of TC1 and TC2 are similar to that of TC0 Module.

5.2 Program TC1 as Timer-1 (T1) to Generate 5-sec Time Delay from 1-sec Time Tick

In this task, we will program and operate TC1 to create '5-sec Time Delay' function using suitable frequency from the internal oscillator. The function will be checked by blinking the built-in LED (L) of the Arduino UNO Kit.

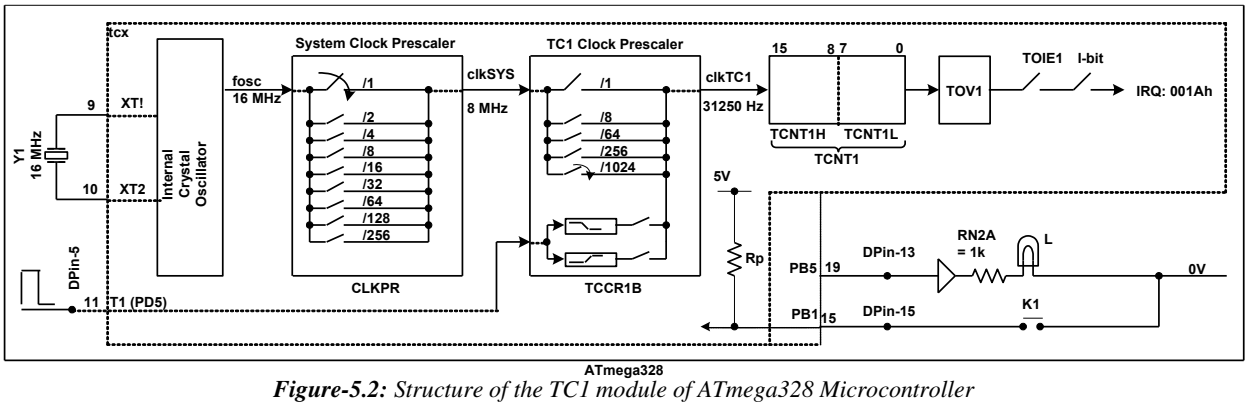

Figure-5.2: *Structure of the TC1 module of ATmega328 Microcontroller*

(1) Connect K1 with the Arduino UNO Kit. The LED (L) circuit is already there in the Arduino board.
(2) **P522.ino** Convert the following pseudo code into Arduino code. Save the program as P522.ino under the working directory.

```
Void setup()
{
        //------Initialize as everything needed ---
        Set direction of PB5-line as output
        Initialize secCounter at 0x00;                    // to count 5 sec for 1-sec Time Tick

        //----Initialize TC1---------------------------------
        Set Normal Up-counting Mode of TC1 by sending     : 0x00 → TCCR1A register
        Keep TC1 OFF by sending                           : 0x00 → TCCR1B

        //--compute preset value for TCNT1 assuming clkTC1 = clkSYS / 1024 = 15625 Hz----------
        //-- Preset value for 1-sec Time Tick (TT) : 1000h – 15625 = 10000h – 3D09h = C2F7h-----
        Load preset value into TCNT1 by sending           : 0xC2F7 → TCN1
        Run TC1 at clkTC1 = 15625 Hz by sending           : 0xC2F7 → TCC1B
}
```

```
Void loop()
{
        //Check if TOV1 had gone LH to detect roll-over meaning 1-sec has gone
L1:     If (TOV1 !=LH)
                Goto   L1
L2:     Clear TOV1 flag by writing LH at this Bit (see Register Layout in Section-5.5).
        Reload preset value into TC1
L3:     Increase secCounter
L4:     if (secCounter != 5)
                Goto L1
L5:     Clear secCounter
L6:     Toggle LED (L)
L7:     Goto L1
}
```

(3) Compile and upload P525.ino.
(4) Press Reset button on the Arduino.
(5) Check that LED (L) is toggling at 5-sec interval.
(6) **P526.ino** Write program to check at every 2-sec interval that the K1 is closed and then toggle LED (L). Save the new program as P553.ino.

5.3 Program TC1 as Counter-1 (C1) to Count External Pulses from a 555-based Oscillator

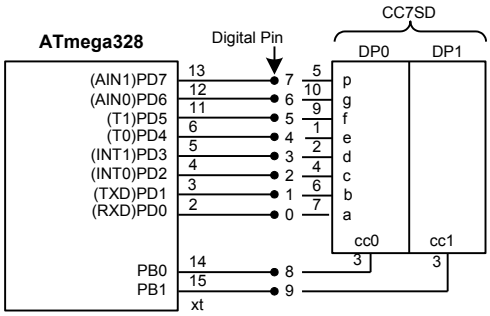

Figure-5.3: *Diagram for cc-type 2-digit multiplexed type display*

(1) Build the display circuit of Fig-5.3 on the breadboard and connect it with the Arduino Board.
(2) Build the following 555-based low frequency oscillator circuit on the breadboard. Connect the output of the oscillator at T1-pin (Fig-5.2) of the Microcontroller.

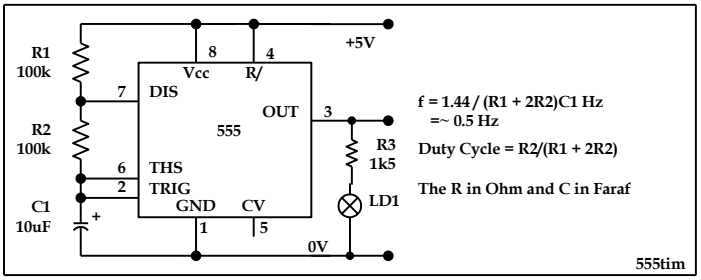

Figure-5.4: *555-based low frequency oscillator*

(3) **P553.ino** Write program to accomplish the following tasks:
- **(a)** Configure TC1 as Counter-1 (C1).
- **(b)** Receive every that is coming from the 555-based oscillator and show it on the cc-type multiplexed display.
- **(c)** Configure T0 of TC0 (Fig-5.1) to generate 2-ms 'Time Delay' function. This delay will be inserted between successive displaying of the digits of the multiplexed display.
- **(d)** There will be no option to call the Arduino's *delay();* function.

(e) Pseudo Code of the Program

```
void setup()
{
        L1: Initialize directions of the port lines of Port-B and Port-D as needed
        L2: Initialize C1 to receive rising edge pulses from external source.
        L3: Initialize T0 to generate 2-ms Time Delay function from the internal oscillator.
        L4: Connect C1 with external oscillator.
        L5: Connect T0 with internal oscillator.
}

void loop()
{
        L6: If (TOV1 !=LH)
                {
                        refreshDisplay();
                        Goto L6
                }
        L7: Clear TOV1
            Reload Preset value into TCNT1
        L8: Update Display
            refreshDisplay();
        L9: Goto L6
}

Void refreshDisplay()
{


}
```

5.4 Program TC1 as Timer-1 (T1) to Count the Execution Times of the Arduino Instructions

In this task, we will experimentally determine the execution times of a few Arduino high level instructions. The procedure involves the following tricks: start T1 at 16 MHz clock before the execution of the 'target' instruction; execute the target instruction; stop the T1 after the execution of the target instruction; show the clock accumulated by TCN1 on LCD.

(1) Build the following LCD circuit on the breadboard, and connect it with the Arduino Kit.

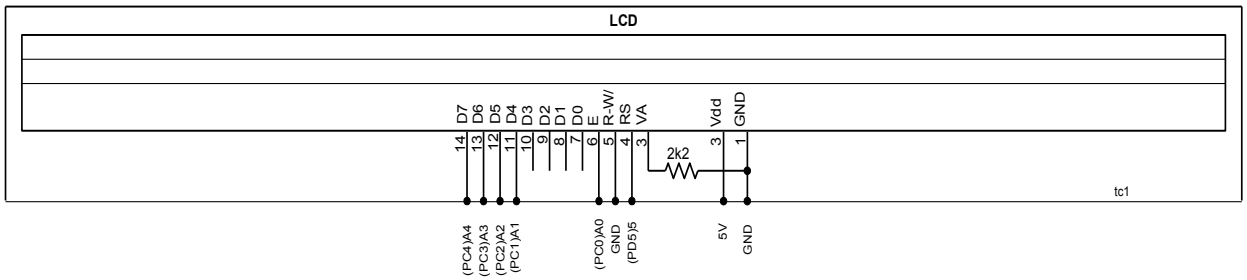

Figure-5.5: *LCD module for interfacing with Arduino UNO*

(2) **P542.ino** Write program to determine the execution times of various Arduino instructions as shown in the following Table.

: Arduino UNO R3 : Timer-1 Clock = 16 MHz :				
Sn	Instruction	What does the function do?	Execution Time	
			Cycles	µs
1	pinMode(2, INPUT_PULLUP)	Port Line 2 of Port-D is input with internal pull-up	78	4.875
2	pinMode(4, OUTPUT);	Port Line 4 of Port-D is output	70	4.375
3	DDRD = 0b00001000;		4	0.250
4	digitalWrite(4, HIGH);	Asserts LH at port line 4 of PORTD	59	3.687
5	bitSet(PORTD, 4);	Same as above	4	0.250
6	bitWrite(PORTD, 4, HIGH)	Same as above	4	0.250
7	digitalWrite(4, LOW);	Asserts LL at port line 4 of PORTD	61	3.812
8	bitClear(PORTD, 4);	Same as above	4	0.250
9	bitWrite(PORTD, 4, LOW);	Same as above	4	0.250
10	bitWrite(PORTD, 4, !bitRead(PORTD, 4));	Toggles the logic value of port line 4 of PORTD	8	0.500
11	bitRead(PORTD, 4);	Reads (from internal latch) the value previously written into port line 4 of PORTD	3	0.187
12	digitalRead(2);	Reads logic value from physical pin of port line 2 of PIND.	36	2.250
13	bitRead(PIND, 2);	Same as above	3	0.187
14	int x = analogRead(A0);	Reads ADC value of Ch-0 and assigns to x.	3328	208.000
15	analogReference(DEFAULT);	Sets the Reference value (5V) for ADC	4	0.250
16	analogWrite(3, 0x23);	Direct value for pulse width of PWM	94	5.875
	analogWrite(3, OCR2B);	Same as above, but value comes from a register	153	9.562
17	analogWrite(3, analogRead(A0));	Same as above, but value comes from ADC	3477	217.312

5.9 Register Description

(1) TCC0A TC0 Control Register A

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 7:6 – COM0[A1:A0]: Compare Output Mode for Channel A
= 0, 0 Normal up-counter operation of TC0
No PWM signal on PD6-pin. PD6 line is disconnected from PWM0A nodule.
PD6-pin will work as normal digital IO line.

Bits 5:4–COM0[B1:B0] Compare Output Mode for Channel B
= 0, 0 Normal up-counter operation of TC0
No PWM signal on PD5-pin. PD5 line is disconnected from PWM0B nodule.
PD5-pin will work as normal digital IO line.

(2) TCC0B TC0 Control Register B

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bits 2:0 –CS0[2:0]: Clock Select 0
= 000 No clock is selected. Stop condition of TC0
= 001 /1 Clock Prescaler
= 010 /8 Clock Prescaler
= 011 /64 Clock Prescaler
= 100 /256 Clock Prescaler
= 101 /1024 Clock Prescaler
= 110 External Clock, falling edge
= 111 External Clock, Rising Edge

(3) TIMSK0 TC0 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0
						OCIEB	OCIEA	TOIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 0 – TOIE: 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 sub routine ISR (TIMER0_OVF_vect) is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in TIFRO register.