1. **Get 0 points, if any ways of cheating**
2. **NO communication devices or applications allowed**
3. **Examination time: 1200-1600 and the duration can be extended per request**

Hands-on Exam

Breathing LEDs with background music

Score: 100%

Oscillator frequency must be 4Mhz.

1. [15%] Let pic18f4520 and your computer comunicate with each other. Input the mode as below on Command Line Interface(or terminal).
   - Mode1 > Type “blink 2 1”, and then do the thing for the 2nd question.
   - Mode2 > Type “breath”, and then do the thing for the 3rd question.
   - Mode3 > Type “music 1”, and then do the thing for the 5th question.

   Typing ‘**q**’ on modeX will come back mode-selected function, typing ‘**s**’ will stop music, and typing ‘**p**’ will play music.
2. [15%] Adjust the count of LED and time on blinking LEDs, and use 4 LEDs at most. The time unit is 250ms. e.g, “blink 2 1”, The blink time interval is 1 unit, and use 2 LEDs.
3. [15%] Design the breathing LED at CCP1 (Refer to Breathing LED).
4. [15%] Control the frequency (duty cycle) of breathing LED by Potentiometer (Variable Resistor).
5. [15%] Play the song you select (Refer to The song format. Maybe you are good at music and can build a better song yourself, and you are just care about the frequency of notes). E.g, “music 1” will play the first song. You can choose songs to play.
6. [15%] Following the 5th question, you can do something and play the song on background **simultanoutly**.
7. [10%] Design 4 breathing LEDs. (Don’t care about the pattern, but the only thing that is forbidden at the same status on 4 LEDs.)

Description

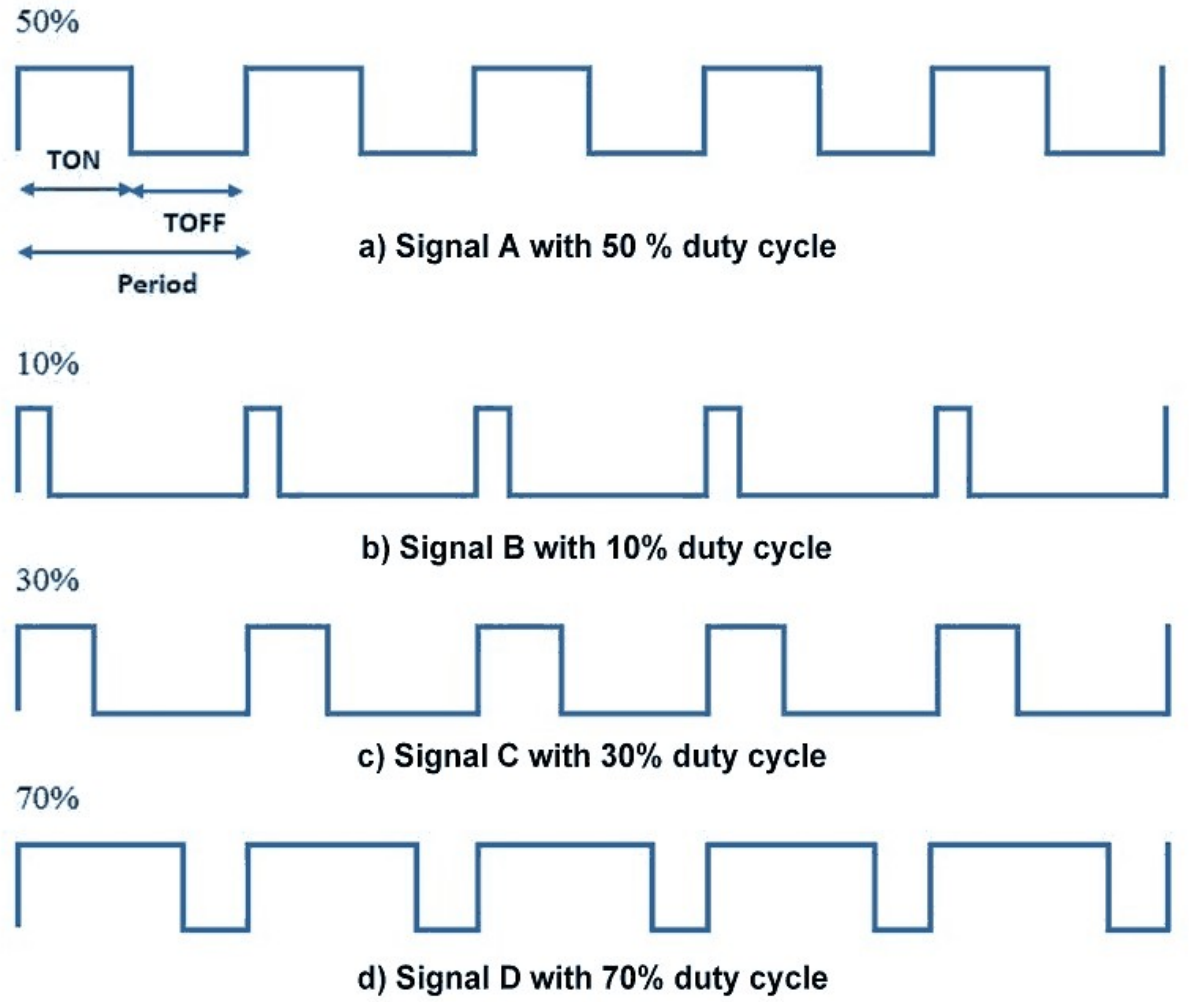
Potentiometer

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Breathing LED

Through PWM technique, we can control the power delivered to the load by using ON-OFF signal. The PWM signals can be used to control the speed of DC motors and to change the intensity of the LED. Moreover, it can also be used to generate sine signals.

Pulse Width Modulated signals with different duty cycle are shown below



The song format

- The note frequency definition:

```
#define C4 262
#define F4 349
#define G4 392
#define A4 440
#define B4 494
#define C5 523
#define D5 587
#define F5 698 ...
```

OCTAVE NUMBER									
	0	1	2	3	4	5	6	7	8
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01
C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64
D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65
F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93
G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00
A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13

- The frequency of song “Lightly Row” partly:

```
// Note of the song, 0 is a rest/pulse.
int per_arrp[30] = { G5, E5, E5, 0, F5, D5, D5, 0, C5, D5, E5, F5, G5, G5, G5, 0,
        G5, E5, E5, 0, F5, D5, D5, 0, C5, E5, G5, G5, E5, 0
}
```

- The duration of song “Lightly Row” partly:

```
// 1 unit for 250ms
int wait[2] = {1}
```

- The frequency of song “He’s a pirate” partly:

```
// Note of the song, 0 is a rest/pulse.
int per_arrp[19] = { E4, G4, A4, A4, 0, A4, B4, C5, C5, 0, C5, D5, B4, B4, 0, A4, G4, A4, 0
}
```

- The duration of song “He’s a pirate” partly:

```
// 1 unit for 250ms
int wait[19] = {1,1,2,1,1,1,1,2,1,1,1,1,2,1,1,1,1,3,1}
```

Written Exam

Score: 40%

1. [8%][Timer] According to the figure of block diagram, please describe how do the Timer3 work by controlling each bit field of the registers.

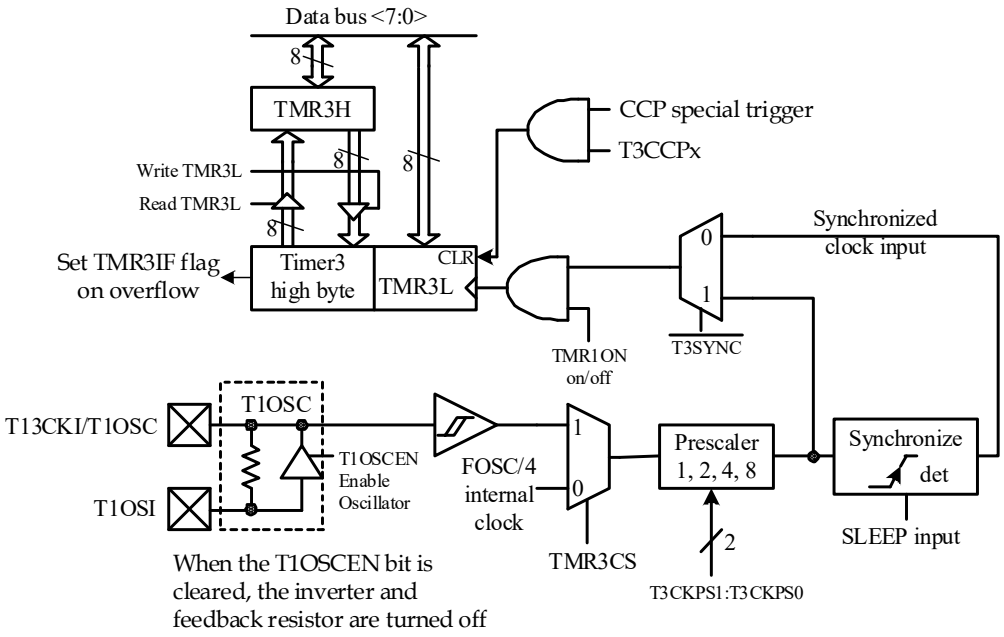


Figure 8.7 Timer3 block diagram: 16-bit mode (redraw with permission of Microchip)

2. [8%][USART] Compute the actual baud rate and the value to be written into the SPBRG register to generate 19200 baud for asynchronous mode high-speed transmission assuming the frequency of the crystal oscillator is 40 MHz and BRGH=0.

3. [8%][ADC] Please describe how you can achieve the better similarity and resolution between an analog signal and a digital signal sequence during the data acquisition.

4. [8%][Interrupt] An ADC ISR collects the sampled data every 1 ms. If we would like to do a complicated computation, which takes about 4 ms, for every 16 collected samples, please using the FLOW CHART to explain how you will do this job.

5. [8%][PWM] Please explain how you can generate a digital waveform with 40% duty cycle and 10 KHz frequency using Timer only, NOT CCP.