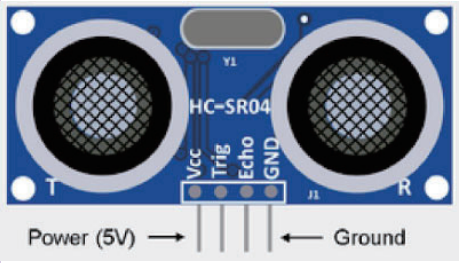
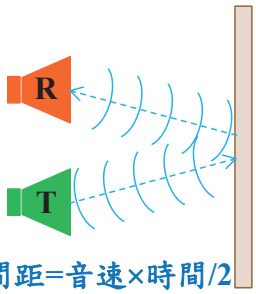
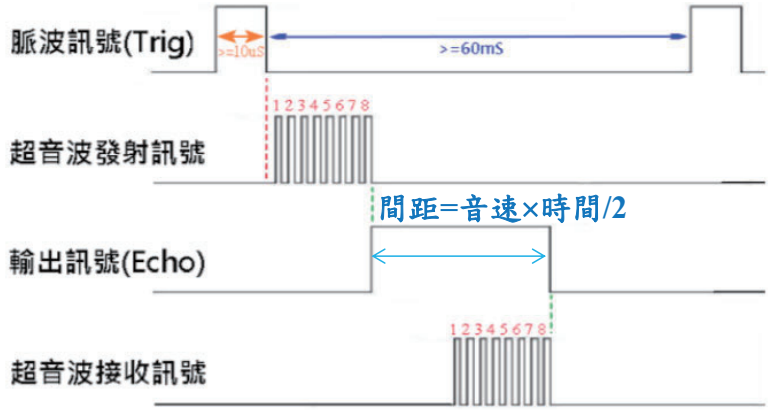
Holtek – Lab04 補充/ 超音波模組 (HC-SR04) Timer/Counter Mode







http://maker.tn.edu.tw/modules/tad_book3/page.php?tbdsn=201
https://sites.google.com/view/rayarduino/%E8%B6%85%E9%9F%B3%E6%B3%A2%E8%B7%9D%E9%9B%A2%E6%84%9F%E6%B8%AC%E5%99%A8

- 右邊的那顆是**發射器（Transmitter）**標示為**T**，會發出 40 kHz 的聲波，由於這個聲波的頻率超過人類可聽見的 20 kHz，因此被稱為超音波。左邊的那顆的是**接收器（Receiver）**標示為**R**，可以接收超音波。它可以**感測的距離為 2cm到 400cm，感應角度為 15 度**。**HC-SR04** 腳位由右至左分別是 **Vcc、Trig、Echo 與 Gnd**。
- **Trig 腳**送進至少維持 **10 微秒以上的高位準訊號**，便能觸發模組中的超音波**發射器送出 8 個連續的 40KHz 超音波脈衝**，**接收器收到反射波後便會在 Echo 腳輸出一個與量測距離成正比的高位準脈衝**，此高位準脈衝上緣可以看成超音波開始發射時間；而下緣則是接收到反射波的時間，所以整個**高位準脈衝的寬度**就是**超音波往返的總時間**，要特別注意的地方是**被測物體最好大於 0.5 平方公尺**，而 **Trig 時間間隔最好大於 60ms**，避免 Trig 與 Echo 互相干擾。
- 超音波在空氣中傳播的速度為**Vs =331.5+ 0.607 t** (t 為 當時的溫度)，傳播速度會受到當時溫度的影響，而且溫度愈高，傳播速度就愈快。 **假設當時的溫度是 28℃**，Vs ≈ 348.5 m/s，超音波行走 1cm 反射回來的時間 T，表示**超音波行走 1cm只需57.4微秒**。單程時間約為28.7**微秒**。
- 使用TM2的**Timer/Counter Mode**或**Capture Input Mode**(此模式可能需要用到Timer2的中斷功能!)，在送出TRIG信號後，啟動Timer2(**_t2on=1**)，**然後依序取得TP2_0(_pc3)在正緣與負緣準位轉換的時間，求得時間差(以微秒為單位)，除以二再除以28.7就可得到物件與超音波模組的距離**。

Holtek – Lab04 補充/ 超音波模組 (HC-SR04) Timer/Counter Mode

```
// Interconnect lines on the Practice Board :
//   (1) LEDs --> MCU : PC6,PC7-->PC6,PC7
//   (2) Ultrasonic --> MCU : TRIG --> PF0, ECHO --> PC3
//   (3) 7-segment-LEDs 4 digit display --> MCU : PD-->PD, PE-->PE
//
#include "HT66F50.h"

#define Led7_com _pe // pd 埠
#define Led7_seg _pd // pe 埠

const unsigned char led7com[4] = { 0x08, 0x04, 0x02, 0x01 } ;
const unsigned char led7seg[11]= { 0x3f, 0x06, 0x5b, 0x4f, 0x66,
                                   0x6d, 0x7d, 0x07, 0x7f, 0x67, 0x00 } ;

unsigned char dig_bcd[4];
unsigned int  echo_pos, echo_neg, duration;
void delay (unsigned int n) //(2+(4+1+2)*4+1+1+2)*n+1=34n+1
{
    unsigned int idy1, idy2 ;

    for(idy1 = 0; idy1 < n ; idy1++)   //
    {
       //GCC_NOP();
       //GCC_NOP();
       for(idy2=0; idy2<4 ; idy2++)
       {
          GCC_NOP();
          GCC_NOP();
          GCC_NOP();
          GCC_NOP();  //
       }
    }
}
void delay_us (unsigned int n) //2+(1+1+2)*n+1=4n+3
{
    unsigned int idy1;

    for(idy1 = 0; idy1 < n ; idy1++)   //
    {
       GCC_NOP();
    }
}

// transform an unsigned integer into 4 bytes (one byte for one digit)
void distance_to_4byte (unsigned int dist)
{
    unsigned int val1,val2;
    dig_bcd[3]=dist/1000;
    val1=(dist%1000);
    dig_bcd[2]=val1/100;
    val2=(val1%100);
    dig_bcd[1]=val2/10;
    dig_bcd[0]=val2%10;
    if (dist<1000){
       dig_bcd[3]=10;
       if (dist<100){
          dig_bcd[2]=10;
          if (dist<10){
             dig_bcd[1]=10;
          }
       }
    }
}
```

Holtek – Lab04 補充/ 超音波模組 (HC-SR04) Timer/Counter Mode

```
void main(void) // 主函式
{
    unsigned char t2d_hi_pos,t2d_lo_pos,t2d_hi_neg,t2d_lo_neg;
    unsigned int distance;
    unsigned char dig_pos;
    unsigned char meas_delay;
// [7:5]=100->fH/16,101->fH/8,110-->fH/4, 000,001->fL; [4]:fsten; [3]:LTO;
// [2]:HTO; [1]:IDLEN, 0:SLEEP; [0]=1-->fH, 0-->fH/? or fL
    smod=0b10000001;   使用高頻時脈信號作為MPU的系統時脈。在Configuration
//  smod=0b00000000;   Options視窗設定HIRC@8MHz為高頻時脈源。
//  wdtc=0b01111010; //[3:0]=1010:default-->disable WDT timer
    _pdc = 0 ;  // 設定 PD 埠為輸出; seven-segment LEDs
    _pec = 0 ;  // 設定 PE 埠為輸出; PE[3:0]=com port
    Led7_com = led7com[0] ;                  // initial common port
    Led7_seg = led7seg[0];    // initial LED port
//Time2:____                ____
//  PC3/PINT/TP2_0/C1- ; PC4/[INT0]/[PINT]/TCK3/TP2_1
    _c1sel=0;  //set PC3/C1- as I/O after assigning _c1sel=0
    _pcc3=1;   // 設定 PC3 埠為輸入, Work as ECHO signal of Ultrasonic Module
    _pcc4=0;   // 設定 PC4 埠為輸出
//fclk_tm2=ftbc (LXT:32.768kHz, LIRC:32kHz)
//ultrasonic speed = 348 m/sec
//Measured Distance = 348*(echo count value)/32000
//Measured Distance = 348*(echo count value)/32000
//  _tm2c0=0b01000000; //[6:4]=000:fsys/4; 001:fSYS; 010:fH/16; 011:fH/64; 100:ftbc,
//[7]=T2PAU; [6:4]=TnCK2~TnCK0; [3]=T2ON; [2:0] unused
//  _tm2c1 register setting
//[7:6]=T2M1,T2M0; [5:4]=T2IO1,T2IO0; [3]=T2OC; [2]=T2POL; [1]=T2DPX; [0]=T2CCLR
//[7:6]=11:Timer/Counter, [5:4]=unused, [3:0] not used
//In the Timer/Counter Mode, the TM output pin control must be disabled.
//  TM output pin control resgister: T2CP1(PC4),T2CP0(PC3)
//fclk_tm2=8Mhz/16=0.5MHz --> 2us; counting value = 20 --> 40us
    _tm2c0=0b00100000; //[6:4]=000:fsys/4; 001:fSYS; 010:fH/16; 011:fH/64; 100:ftbc
    _tm2c1=0b11110001;      //[7:6]=11:Timer/Counter,
    _t2cp0=0;  //t2cp0:pc3, t2cp1:pc4
    _t2cp1=0;  //t2cp0:pc3, t2cp1:pc4
    _t2on=0;
    _pfc0 = 0;  //Trigger Ultrasonic, 1:輸入, 0:輸出
    _pcc7 = 0;
    _pcc6 = 0;
//TRIG-->PF0, ECHO-->PC3
    while(1)
    {
      //Trigger Ultrasonic 0->1->0
      _pf0=0;              送出至少10us脈波寬的TRIG信號
      delay_us(5);         給超音波感測模組
      _pf0=1;
      delay_us(20);
      _pf0=0;
      _t2on=1;             啟動Timer2的計時/計數功能
      //wait for the positive edge of the echoed pulse
      while(_pc3==0)       等待從超音波感測模組的ECHO
      {                    回傳的高準位脈波之正緣
        _pc7=1;
      }
      t2d_hi_pos=_tm2dh;       讀取ECHO正脈波正緣時的
      t2d_lo_pos=_tm2dl;       TM2計數值
      //echo_pos=t2d_hi_pos*256+t2d_lo_pos;
      //wait for the negative edge of the echoed pulse
      while(_pc3==1)       等待從超音波感測模組的ECHO
      {                    回傳的高準位脈波之負緣
        _pc6=1;
      }
      t2d_hi_neg=_tm2dh;       讀取ECHO正脈波負緣時的
      t2d_lo_neg=_tm2dl;       TM2計數值
      echo_pos=t2d_hi_pos*256+t2d_lo_pos;
      echo_neg=t2d_hi_neg*256+t2d_lo_neg;
      duration=echo_neg-echo_pos;  計算ECHO脈波寬時間
```

設定PC3為輸入接腳(要確保PC3的其他優先接腳功能已經被關閉)，超音波感測模組的ECHO信號會從PC3傳回MPU

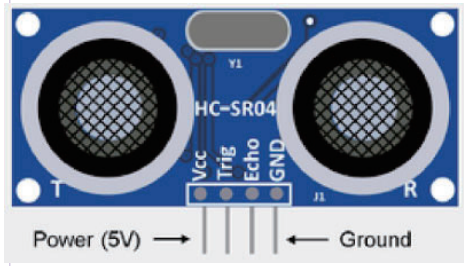
Holtek – Lab04 補充/ 超音波模組 (HC-SR04) Timer/Counter Mode

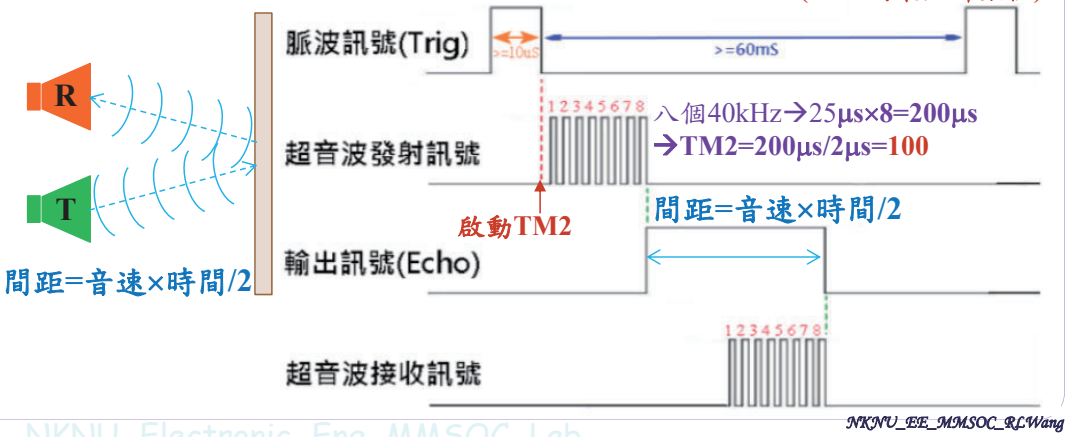
```
      //wait for the response of the ultrasonic module
      //delay(300);
      delay(10);
      //
      _t2on=0;
      distance=duration/29; //0.0348*echo=echo/29-->0~2281cm
      distance_to_4byte(distance);

      //Sweeping 7-seg 4 digit display within (1/16) second to satisfy the phenomenon of the persistence of vision
      for(meas_delay=0; meas_delay<50; meas_delay++)
      {
        for( dig_pos = 0 ; dig_pos < 4 ; dig_pos++ )
        {
          Led7_com = 0x00 ;                  // 8x8 LED X-axis (PD) OFF
          //  delay(1);
          Led7_seg = led7seg[dig_bcd[dig_pos]];          // 8x8 LED Y-axis (PE) ON
          Led7_com = led7com[dig_pos] ;                  // 8x8 LED X-axis (PD) OFF
          if (dig_pos<3)
              delay(20);       //adjust the brightness of LEDs of the 7-segment display
        }
      }
    }
}
```

meas_delay for-loop是讓七段顯示器的掃描總時間與整個while(1){ }迴圈的總時間相比，具有夠大的百分比，以確保七段顯示器的亮度夠亮，並確保有滿足視覺暫留的條件。

fsys=fH=8MHz，TM2的時脈源為fH/16=0.5MHz。
TM2每計數增1，代表時間增加1/0.5MHz=2μs。
間距(cm)=音速×(ECHO脈波寬的TM2計數差值×2μs)/2
=音速×(ECHO脈波寬的TM2計數差值)
=0.0348cm/μs×(ECHO脈波寬的TM2計數差值)
≈(ECHO脈波寬的TM2計數差值)/29

音速=331.5+0.607×t (m/s)
t=28°C,
音速=331.5+0.607×28≈348 (m/s)
=348×100/10⁶=0.0348 (cm/μs)
規格書標示的可測最大距離=400cm
脈波寬的TM2計數之最大差值≈400×29=11600
11600+100=11700<<2^{16}-1=65535(TM2的最大計數值)





Holtek – Calculation of Delay Time

對CPU系統內建的高頻時脈源HIRC(12MHz)除頻，產生音樂的音頻，驅動蜂鳴器以發出音樂。 TM2(16位元)利用compare match的toggle out模式，每次計數達到(match)條件就切換準位，也就是，從零開始計數達到條件就是半周期，最低的輸出頻率為12MHz/65536/2=**91.553Hz。tone frequency=12MHz/(tm2ah, tm2al)**/2

頻率，單位為**赫茲 Hz**（括號內為半音距離，"（0）"為中央C）										
八度→ 音名↓	0	1	2	3	4	5	6	7	8	9
C	16.352（−48）	32.703（−36）	65.406（−24）	130.81（−12）	**261.63 (0)**	523.25（+12）	1046.5（+24）	2093.0（+36）	4186.0（+48）	8372.0（+60）
C♯/D♭	17.324（−47）	34.648（−35）	69.296（−23）	138.59（−11）	**277.18（+1）**	554.37（+13）	1108.7（+25）	2217.5（+37）	4434.9（+49）	8869.8（+61）
D	18.354（−46）	36.708（−34）	73.416（−22）	146.83（−10）	**293.66（+2）**	587.33（+14）	1174.7（+26）	2349.3（+38）	4698.6（+50）	9397.3（+62）
D♯/E♭	19.445（−45）	38.891（−33）	77.782（−21）	155.56（−9）	**311.13（+3）**	622.25（+15）	1244.5（+27）	2489.0（+39）	4978.0（+51）	9956.1（+63）
E	20.602（−44）	41.203（−32）	82.407（−20）	164.81（−8）	**329.63（+4）**	659.26（+16）	1318.5（+28）	2637.0（+40）	5274.0（+52）	10548（+64）
F	21.827（−43）	43.654（−31）	87.307（−19）	174.61（−7）	**349.23（+5）**	698.46（+17）	1396.9（+29）	2793.8（+41）	5587.7（+53）	11175（+65）
F♯/G♭	23.125（−42）	46.249（−30）	92.499（−18）	185.00（−6）	**369.99（+6）**	739.99（+18）	1480.0（+30）	2960.0（+42）	5919.9（+54）	11840（+66）
G	24.500（−41）	48.999（−29）	97.999（−17）	196.00（−5）	**392.00（+7）**	783.99（+19）	1568.0（+31）	3136.0（+43）	6271.9（+55）	12544（+67）
G♯/A♭	25.957（−40）	51.913（−28）	103.83（−16）	207.65（−4）	**415.30（+8）**	830.61（+20）	1661.2（+32）	3322.4（+44）	6644.9（+56）	13290（+68）
A	27.500（−39）	55.000（−27）	110.00（−15）	220.00（−3）	**440.00（+9）**	880.00（+21）	1760.0（+33）	3520.0（+45）	7040.0（+57）	14080（+69）
A♯/B♭	29.135（−38）	58.270（−26）	116.54（−14）	233.08（−2）	**466.16（+10）**	932.33（+22）	1864.7（+34）	3729.3（+46）	7458.6（+58）	14917（+70）
B	30.868（−37）	61.735（−25）	123.47（−13）	246.94（−1）	**493.88（+11）**	987.77（+23）	1975.5（+35）	3951.1（+47）	7902.1（+59）	15804（+71）

http://zh.wikipedia.org/zh-tw/%E9%9F%B3%E9%AB%98

Holtek – Calculation of Delay Time

八度→	Tone (Hz)	value to count	High_byte	Low_byte	Tone (Hz)	value to count	High_byte	Low_byte	Tone (Hz)	value to count	High_byte	Low_byte
音名↓	**3**	**12MHz/tone/2**	value/256	mod(,256)	**4**	**12MHz/tone/2**	value/256	mod(,256)	**5**	**12MHz/tone/2**	value/256	mod(,256)
C	130.81	45868	179	44	**261.63**	**22933**	**89**	**149**	**523.25**	11467	44	203
C♯/D♭	138.59	43293	169	29	277.18	21647	84	143	554.37	10823	42	71
D	146.83	40864	159	160	**293.66**	20432	79	208	587.33	10216	39	232
D♯/E♭	155.5	38585	150	185	311.13	19285	75	85	622.25	9642	37	170
E	164.81	36406	142	54	**329.63**	18202	71	26	659.26	9101	35	141
F	174.61	34362	134	58	**349.23**	17181	67	29	698.46	8590	33	142
F♯/G♭	185	32432	126	176	369.99	16217	63	89	739.99	8108	31	172
G	196	30612	119	148	**392**	15306	59	202	783.99	7653	29	229
G♯/A♭	207.65	28895	112	223	415.3	14447	56	111	830.61	7224	28	56
A	220	27273	106	137	**440**	13636	53	68	880	6818	26	162
A♯/B♭	233.08	25742	100	142	466.16	12871	50	71	932.33	6435	25	35
B	246.94	24297	94	233	**493.88**	12149	47	117	987.77	6074	23	186

```
//                                  0  1(C) 2(D) 3(E) 4(F) 5(G) 6(A) 7(B) 1(C)
const unsigned char tone_HB[9] = {0,  89,  79,  71,  67,  59,  53,  47,  44};
const unsigned char tone_LB[9] = {0, 149, 208,  26,  29, 202,  68, 117, 203};
// music_tone[ ]:陣列內是簡譜, music_beat[ ]是對應簡譜每個音的節拍長度(如4:全音符, 3:3/4音符, 2:1/2音符)
const unsigned char music_tone[24] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 0, 0, 5, 3, 3, 4, 2, 2, 1, 2, 3, 4, 5, 5, 5} ;
const unsigned char music_beat[24] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4} ;
for (i=0; i<=music_length-1;  i++)   // music_length=24
        _tm2al=tone_LB[music_tone[i]];          //tone frequency=12MHz/(tm2ah, tm2al)/2
        _tm2ah=tone_HB[music_tone[i]];          //(tm2ah, tm2al) =12MHz/(tone frequency)/2
        _t2on=1;
        delay_m(music_beat[i]*4);       //(music_beat[i]*4+1)*0.0123s; music_beat[i]=4-->0.2091s
        //delay_m(music_beat[i]*20);    //(music_beat[i]*20+1)*0.0123s; music_beat[i]=4-->1s

        t2on=0;
        delay_m(1);
}
```

Holtek – C to Assembly Code

```
#include "HT66F50.h"
#define LedPortx _pd // pd 埠
#define LedPorty _pe // pe 埠
void delay (unsigned char n)
{
    unsigned char idy1, idy2, idy3;
    for(idy1 = 0; idy1 < n ; idy1++)   //
    {
            GCC_NOP();
            for(idy2=0; idy2<200 ; idy2++)
            {
               GCC_NOP();
               for(idy3=0; idy3<200 ; idy3++)
               {
                        GCC_NOP();
               }
            }
    }
}
void main(void)
{
    _smod=0b11000000;       // [7:5]=100->fH/16,110-->fH/4
    _pdc = 0 ;              // 設定 PD 埠為輸出
    LedPortx = 0x00 ;       // 8x8 LED X-axis (PD) OFF
    _pec=0;                 // 設定 PE 埠為輸出
    LedPorty = 0x00;        // 8x8 LED Y-axis (PE) ON
    while(1)
    {
            LedPortx = 0x01 ;
            LedPorty = 0xff;            // 8x8 LED Y-axis (PE) ON
            delay(10);
            LedPortx = 0x02 ;
            LedPorty = 0x7e;            // 8x8 LED Y-axis (PE) ON
            delay(10);
            LedPortx = 0x04 ;
```

Assembly code

MOV A, [m]	Move Data Memory to ACC
MOV [m], A	Move ACC to Data Memory
MOV A, x	Move immediate data to ACC
SUB A, [m]	Subtract Data Memory from ACC
SDZ [m]	Skip if decrement Data Memory is zero
SNZ [m].i	Skip if bit i of Data Memory is not 0

```
0000  2801  JMP  1H
0001  2816  JMP  16H
0002  4083  MOV  [83H],A
0003  5F00  CLR  [80H]
0004  2811  JMP  11H
0005  0000  NOP                    n次
0006  0FC8  MOV  A,0c8H
0007  4081  MOV  [81H],A
0008  0000  NOP
0009  0FC8  MOV  A,0c8H
000A  4082  MOV  [82H],A
000B  0000  NOP                    200次
000C  5782  SDZ  [82H]   200次
000D  280B  JMP  0bH
000E  5781  SDZ  [81H]
000F  2808  JMP  8H
0010  5480  INC  [80H]             result≠0 →Z=0
0011  4700  MOV  A,[80H]           result=0 →Z=1
0012  4203  SUB  A,[83H]
0013  390A  SNZ  Z   <----(Z flag status)
0014  2805  JMP  5H
0015  0003  RET
0016  0FC0  MOV  A,0c0H
0017  008B  MOV  SMOD,A
0018  1F24  CLR  PDC
0019  1F23  CLR  PD
001A  1F27  CLR  PEC
001B  1F26  CLR  PE
001C  0F01  MOV  A,1H
001D  00A3  MOV  PD,A
001E  1FA6  SET  PE
001F  0F0A  MOV  A,0aH
0020  2002  CALL 2H
0021  0F02  MOV  A,2H
0022  00A3  MOV  PD,A
```

Increase idy1 and check whether idy1 equals to n

Holtek – Calculation of Delay Time

```
// machine periosd = 4/12E6; jump: 2 machine period
// 3 GCC_NOP() in the 2nd for-loop:
//   {2_move+[2_move+(3_nop+1_minus1+1_jump)*30+1_minus1+1_jump]*n}*4/12E6
//   ={2+[2+(3+1+2)*30+1+2]*n}*4/12E6=(2+185n)*4/12E6=(740n+8)/12E6
// 4 GCC_NOP() in the 2nd for-loop:
//   {2_move+[2_move+(4_nop+1_minus1+1_jump)*30+1_minus1+1_jump]*n}*4/12E6
//   ={2+[2+(4+1+2)*30+1+2]*n}*4/12E6=(2+215n)*4/12E6=(860n+8)/12E6
// 5 GCC_NOP() in the 2nd for-loop:
//   {2_move+[2_move+(5_nop+1_minus1+1_jump)*30+1_minus1+1_jump]*n}*4/12E6
//   ={2+[2+(5+1+2)*30+1+2]*n}*4/12E6=(2+245n)*4/12E6=(980n+8)/12E6
void delay (unsigned int n)
{
        unsigned int idy1, idy2 ;
        for(idy1 = 0; idy1 < n ; idy1++)   //
                {
                        //GCC_NOP();
                        //GCC_NOP();
                        for(idy2=0; idy2<30 ; idy2++)
                        {
                                GCC_NOP();
                                GCC_NOP();
                                GCC_NOP();
                                //GCC_NOP(); //
                                //GCC_NOP(); //
                        }
                }
}
// machine periosd = 4/12E6
//  [2_move+(2_move+185*250+1_minus1+1_jump)*n+1_minus1+1_jump]*4/12E6
//      =[2+(2+185*250+1+2)*n]*4/12E6=[2+46255n]*4/12E6-->0.0154*n s
//  [2+(2+185*200+1+2)*n]*4/12E6=[37005n+5]*4/12E6-->0.0123*n s
//  [2+(2+245*200+1+2)*n]*4/12E6=[42804n+5]*4/12E6-->0.0163*n s
//  [2+(2+245*250+1+2)*n]*4/12E6=[53504n+4]*4/12E6-->0.0204*n s
```

```
void delay_m (unsigned int n)
{
        unsigned int i ;
        for (i = 0; i < n; i++)
        {
                delay(200);
        }
}
```

[2_move+(2_move+185*200+1_minus1+1_jump)*n]*4/12E6

4_inc_mov_sub_snz

{2_move+[2_move+(3_nop+1_minus1+1_jump)*30+1_minus1+1_jump]*n}*4/12E6

If 1 beat = quarter not (1/4音符), 1 whole note = 4 beats

Time that 1 whole note lasts for is as follows

delay_m(music_beat[i]*4);

(4*4+1)*0.0588s=1s;

(4*4+1)*0.0154s=0.2618s;

(4*4+1)*0.0123s=0.2091s;

(4*4+1)*0.0163=0.2771s

Adjust time for a beat

delay_m(music_beat[i]*20);

(4*20+1)*0.0123s=1s;

(4*20+1)*0.0154s=1.2474s;

(4*20+1)*0.0123s=1s;

(4*20+1)*0.0163=1.3203s

Holtek – Instruction Set Summary

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	Z, C, AC, OV
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV
ADCM A,[m]	Add ACC to Data memory with Carry	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	1Note	Z, C, AC, OV
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C

Note:

1. **For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.**
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged

Holtek – Instruction Set Summary

Mnemonic	Description	Cycles	Flag Affected
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1Note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRD [m]	Read table to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note:

1. **For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.**
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the "CLR WDT1" and "CLR WDT2" instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both "CLR WDT1" and "CLR WDT2" instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged

Holtek – Exercise Lab05

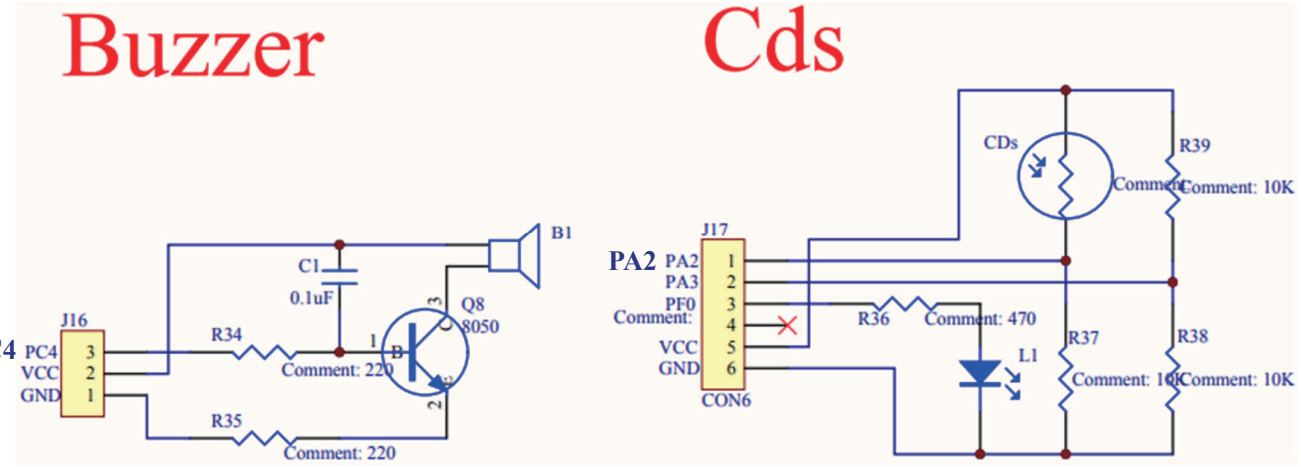
▸▸此次的”音樂”範例，是使用Timer Module 2(16 bit)的compare match output功能來對系統內建的高頻時脈源HIRC(12MHz)除頻，產生音樂的音頻，透過變數陣列之查表方式，建立各音頻的除頻所需之計數值。另外使用delay_m(n)函數，來建立音拍的時間，這也是透過變數陣列之查表方式，建立各音拍延遲時間之引數n的值。Time module與CPU程式運作是兩個不同硬體，Time module的功能由CPU內的特殊功能暫存器設定。在Time module持續以compare match output模式產生所要頻率的時脈，驅動蜂鳴器發出聲音時，CPU程式正在執行delay_m()函式，兩者同時運作來發出所要的音頻，並維持所需音拍的時間。透過for-loop依序發出各音拍時間的音頻，以驅動蜂鳴器來播放出所要的音樂。

▸▸以光敏電阻來切換兩首音樂的播放，強光與弱光時，分別播放不同首音樂。目前程式的功能是光強度明顯變化成另一種光強度(暗或亮)時，要等原來正在播放的音樂播放完畢，才會切換至對應光強度的音樂。

作業要求的規格:

1.請自行將其中一首音樂換成你自己選擇的音樂，

2.並修改程式，讓光強度明顯變化時， 不用等到整首播放完畢才能切換， 而是對應光強度，立即切換音樂。



Holtek – Exercise Lab05 / (HC-SR04) Timer/Counter Mode

▸▸此次的”超音波測距”範例，是使用Timer Module 2(16 bit)的Timer/Couter功能來偵測HC-SR04超音波模組的ECHO接腳回傳的脈波寬度，採用晶片內建的HIRC高頻時脈產生電路輸出的8MHz時脈，設定為除4的頻率做為TM2的計數時脈(也就是，計數值每增加1，表示時間經過了2us)。在由MPU透過PF0接腳送出脈波寬度大於10us的觸發信號後，啟動TM2開始計數，然後分別在ECHO正緣信號變化時及ECHO負緣信號變化時，取得TM2的計數值，在取得ECHO負緣信號變化時的計數值之後，將兩計數值相減，將差值除以2再乘以2us，就可得到物件與超音波模組間單趟的傳輸時間，將傳輸時間乘上音波速度，即可得到物件與超音波模組間的距離。將TM2關閉以便將計數器歸零。最後將此距離數值用四位元七段顯示器顯示出來。依此方式透過while(1)的無窮迴圈，持續偵測物件與超音波模組的間距。

作業要求的規格:

1.加入三色LED模組，以PA5、PA6、PA7分別控制紅、藍、綠的LED接腳，(1)當間距≤10cm，紅燈亮，(2)當間距>10cm且≤25cm，藍燈亮，(3)當間距>25cm，綠燈亮