14 ලිපිය - 8 කොටස

අතුරු බිඳුම්වල ප්‍රායෝගික භාවිත

මෙම ලිපි පෙළේ 6.4, 6.5 සහ 6.6 ලිපි තුන තුළින් අතුරු බිඳුම් හෙවත් Interrupts පිළිබඳ හැඳින්වීමක් සහ ඒවායේ ප්‍රායෝගික භාවිත කිහිපයක් විස්තර කෙරිණි. එහි දී අප ඉදිරිපත් කළ ක්‍රමලේඛ Assembly language භාවිතයෙන් ගොඩනඟන ලද ඒවා විය. එම ක්‍රමලේඛයන් ම C පරිගණක භාෂාව ඇසුරින් ගොඩනංවන ආකාරය විස්තර කර දීම මෙම ලිපියේ සහ මීළඟ ලිපියේ අරමුණ වේ. ඉහත සඳහන් ලිපි තුන හරහා අතුරු බිඳුම් පිළිබඳ මූලික කරුණු ඉදිරිපත් කර ඇති බැවින් එම කරුණු විස්තර කිරීම අනවශ්‍ය යයි හැඟේ. ඔබට අතුරු බිඳුම් පිළිබඳව මූලධර්ම හැදෑරීමට අවශ්‍ය නම් එම ලිපි පෙළ තුන කියවන මෙන් අපි උදක් ම ඉල්ලා සිටිමු.

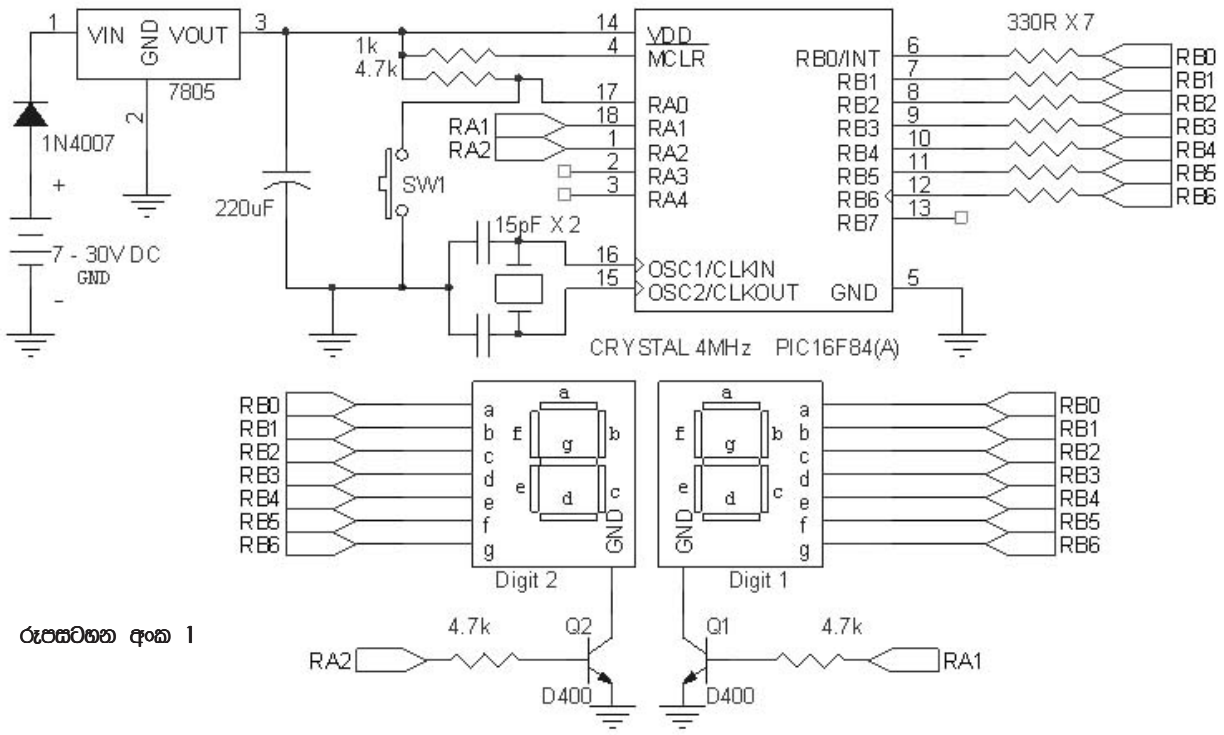
රූප සටහන අංක 1න් අදළ පරිපථ සටහන දැක්වේ. මෙය කලින් ලිපියෙහි සඳහන් පරිපථය ම වන අතර බොත්තමක් එබීම වෙනුවට තත්පරයකට වරක් සප්ත ඛණ්ඩ ප්‍රදර්ශකවල දිස් වන අගය එකකින් වැඩි වීම මෙහි දී සිදු වේ. තත්පරයක කාල පරාසය ම ලබාගැනීම සඳහා PIC 16F84(A) මයික්‍රොකොන්ට්‍රෝලරය තුළ තිබෙන Timer 0 නමැති කාල ගණකයේ සහාය ලබාගෙන ඇත. මෙම Timer 0 ඒකකය පිළිබඳ වැඩි විස්තර සඳහා 6.5 ලිපිය කියවන්න.

Timer 0 ඒකකයෙහි බිටු 8කින් සමන්විත රෙජිස්ටරයක් පවතී. එය TMR0 ලෙස නම් කර ඇත. එම රෙජිස්ටරයෙන් දැක්වෙන අගය එකින් එක වැඩි කරගෙන යාමේ දී උපරිම අගය වන 255ට පැමිණේ. (බිටු 8කින් 0 සිට 255 දක්වා වූ අගයන් නිරූපනය කළ හැකි ය.) එම අවස්ථාව Timer 0 Over flow ලෙස හැඳින්වෙයි. එවිට INTCOM නමැති රෙජිස්ටරයේ 22 වන බිටුව (TOIF - Timer 0 Interrupt flag) තාර්කික 1 බවට පත් වේ. එසේ වූ වහා ම අතුරුබිඳුමක් පතනය වේ. එවිට ප්‍රදර්ශනය වන අගය 1කින් වැඩි කර නැවත ප්‍රධාන ..... ගොනුවට පැමිණේ.

ඊට අදළ ක්‍රමලේඛය රූපසටහන අංක 2න් දැක්වේ. එම ක්‍රමලේඛයේ උපදෙස් ගොනු 3ක් පවතී. ඒවා Interrupt main සහ ssdecode ලෙස නම් කර ඇත. අතුරු බිඳුමක් ජනනය වූ වහාම interrupt නමැති උපදෙස් ගොනුවට පැමිණේ. එම උපදෙස් ගොනුවේ දී ජනනය වූ අතුරු බිඳුම Timer 0 කාල ගණකයෙහි අගය 255ට පැමිණීම හෙවත් Overflow වීම නිසා ඇති වූවක් දැයි පරීක්ෂා කර බලයි. ඒ සඳහා If(INTCON.TOIF) උපදෙස යොදාගෙන ඇත. අතුරු බිඳුම ජනනය වූයේ Timer 0 overflow නිසා නම් TOIF බිටුව තාර්කික 1 බවට පත් වේ. එවිට ඉහත සඳහන් පරීක්ෂා කිරීම සත්‍ය බවට පත් වී ඒ යටතේ එන උපදෙස් එකින් එක ක්‍රියාත්මක කිරීමට පටන් ගනී. මෙවැනි overflow බිටි 15ක් සිදු වීමට ගත වන කාලය තත්පරයකට ආසන්න වේ. එබැවින් එසේ වාර 15ක් ගිය පසු Digit 1 හෙවත් ප්‍රදර්ශනය කළ යුතු එකේ ඒවා ගණන ඉහළ දැමේ. Digit 1 අගය 9 පසු කළේ නම් Digit 2 හෙවත් දහයේ ඒවා ගණන එකකින් වැඩි කර එකේ ඒවා ගණන 0 බවට පත් කෙරේ.

අතුරු බිඳුමක දී ක්‍රියාත්මක කළ යුතු උපදෙස් දැක්වෙන ඉහත සඳහන් Interrupt නමැති උපදෙස් ගොනුව Interrupt Service Routine (ISR) ලෙස තාක්ෂණික ව්‍යවහාරයේ දී හැඳින්වේ.

main නමැති ප්‍රධාන උපදෙස් ගොනුවේ සුපුරුදු පරිදි A හා B තොටුපළවල් ප්‍රදාන හා ප්‍රතිදාන ලෙස සකසා ඇත. ඉන් පසුව OPTION_REG හෙවත් OPTION රෙජිස්ටරය 10000111 බිටු සැකැස්ම ලබා දී ඇත. ඒ අනුව මයික්‍රොකොන්ට්‍රෝලරයේ ප්‍රධාන ඝටිකාවේ ස්පන්ද 256ක් ගිය විට TMR0 රෙජිස්ටරයේ අගය 1කින් වැඩි වේ. එනම් ප්‍රධාන ඝටිකා ස්පන්ද 256×256 (65536)ක් ගිය විට Timer 0 overflow වීම සිදු වේ. INTCON රෙජිස්ටරයට 10100000 බිටු සැකැස්ම ලිවීමෙන් Timer 0 overflow නැමැති අතුරු බිඳුම් වර්ගය ක්‍රියාකාරී තත්ත්වයට ගෙන ඇත. මෙම රෙජිස්ටර් දෙක සහ ඒවායේ එක් එක් බිටුවලින් සිදු කෙරෙන කාර්යයන් පිළිබඳ වැඩි විස්තර දැනගැනීම සඳහා PIC 16F84(A) මයික්‍රොකොන්ට්‍රෝලරයේ දත්ත



රූපසටහන අංක 1

```
    // Introduce functions and variables
void ssdecode(int i);      // Function for 7 Segment decoding
int Digit1;                // Variable for digit1
int Digit2;                // Variable for digit2
int LoopCounter;           // Cycle counter

   // Interrupt Service Routine (ISR)
void interrupt()
{
 if (INTCON.T0IF)          // Check for Timer 0 interrupt
 {
  LoopCounter++;
  if(LoopCounter > 15) //Check for 15 cycles
  { Digit1 = Digit1 +1;    // Increase Digit 1
        if(Digit1>9)       // if it is 9 then
        {  Digit1=0;       // reset to 0 and
           Digit2++;       // increase Digit 2
             if(Digit2>9)// If Digit 2 is 9
                Digit2=0;// then reset it to 0
        }
   LoopCounter =0;         // Reset Loop Counter
  }

  INTCON.T0IF = 0;         // Clear Timer 0 overflow flag
 }


}
   //Main Function
void main()
{
  TRISA = 0b00011001; //RA0 input
  TRISB  = 0;   //PORTB output
  OPTION_REG = 0b10000111;          //Configure Timer 0
  INTCON = 0b10100000;       //Enable interrupts
  Digit1 = 0;                //Start from 0
  Digit2 = 0;                //Start from 0
  LoopCounter=0;             //Start from 0
while (1)                    // loop forever
{

      ssdecode(Digit1); //Display digit 1

      PORTA.F1 = 1;  //Enable Digit1
      //Delay_ms(20); //Small delay
      PORTA.F1 = 0;  //Disable Digit1

      ssdecode(Digit2); //Display digit 2

      PORTA.F2 = 1;  //Enable Digit2
      //Delay_ms(20); //Small delay
      PORTA.F2 = 0;  //Disable Digit2

      LoopCounter++;//Cycle counter

  }
}


    // Sewven Segment Decoder function
void ssdecode(int i)
{
    switch (i)
    {
    case 0: PORTB = 0b00111111; break;
    case 1: PORTB = 0b00000110; break;
    case 2: PORTB = 0b01011011; break;
    case 3: PORTB = 0b01001111; break;
    case 4: PORTB = 0b01100110; break;
    case 5: PORTB = 0b01101101; break;
    case 6: PORTB = 0b01111101; break;
    case 7: PORTB = 0b00000111; break;
    case 8: PORTB = 0b01111111; break;
    case 9: PORTB = 0b01101111; break;
    }
}
```

රූපසටහන අංක 2

පත්‍රිකාව පරිශීලනය කළ යුතු ය. ක්‍රමලේඛයේ ඉතිරි කොටස් පසුගිය ලිපියේ සඳහන් වූ ඒවාට බෙහෙවින් සමාන බැවින් ඒ පිළිබඳ නැවත නැවත විස්තර කිරීම අනවශ්‍ය යැයි හැඟේ. මීළඟ ලිපිය තුළින් RB0 බාහිර අතුරු බිඳුම ප්‍රායෝගිකව යොදාගන්නා ආකාරය ඉදිරිපත් කෙරේ.

මොරටුව විශ්ව විද්‍යාලයේ විද්‍යුත් හා විදුලි සංදේශ අංශයේ
ගාමිණී ජයසිංහ
කෝලිත ධර්මප්‍රිය