```
/
*************************************************************************
**/
/*プログラム名　著作権表示　プログラム開発環境                          */
/*  Pressure Vibration Telegraph Paddle (感圧振動パドル)                 */
/*  Author JH1IFZ K.A.Yoshida PressElecKeyProgを元に変更開始　2017.04.29－　*/
/*  mikroC Ver.  6.0.0                                                  */
/*  wrighter picKit2 App Ver.2.61 Device File Ver.1.62.14  OS Firm Ver.2.32  */
/*                                                                      */
/
*************************************************************************
**/
/*ＭＰＵ名　ＰＩＣ　１６Ｆ１８２３　ハードウェア設定                    */
/*                                                                      */
/*ピン番号　名称　　設定　　用途                                        */
/*　１　Vdd　　電源＋３.７Ｖ　電源＋３.７Ｖ                             */
/*　２　RA5/AN0　Digital output　振動出力（TR介しアクチュエータ駆動）   */
/*　３　RA4/AN3　Digital output　電鍵出力（半導体リレー）               */
/*　４　RA3/MCLR　Digital input　ストレートキー入力（タクトSW）         */
/*　５　RC5/　　Digital output　モニターLED青                           */
/*　６　RC4/　　Digital output　未使用　　PBSW入力（定型句１）          */
/*　７　RC3/AN7　Digital output　未使用　　PBSW入力（定型句２）         */
/*　８　RC2/AN6　Digital output　未使用                                  */
/*　９　RC1/AN5　Digital output　未使用                                  */
/*　10　RC0/AN4　Digital output　未使用                                  */
/*　11　RA2/AN2　Analog　input　送出スピード調整用VR電圧                */
/*　12　RA1/AN1　Analog　input　短点感圧素子電圧入力                    */
/*　13　RA0/AN0　Analog　input　長点感圧素子電圧入力                    */
/*　14　Vss　　電源GND　　電源GND                                       */
/*                                                                      */
/*                                                                      */
/*　ポート設定                                                          */
/*　　B8 7 6 5 4 3 2 1 0　　1:入力　　0:出力                           */
/*　TRISA　－－－００１１１１                                           */
/*　TRISC　－－－００００００                                           */
/*　　1:アナログ　0:デジタル                                            */
/*　anselA　－－－０００１１１                                          */
/*　anselC　－－－００００００                                          */
/*                                                                      */
/*　WPUA　－－０００１０１１　　弱プルアップ　１：有効                 */
/*                                                                      */
/*                                                                      */
/*                                                                      */
/
*************************************************************************
**/
/*ＭＰＵ名　ＰＩＣ　１２Ｆ６７５　ハードウェア設定                      */
/*                                                                      */
/*ピン番号　名称　　設定　　用途                                        */
/*　１　Vdd　　電源＋３.７Ｖ                                            */
/*　２　GP5/AN0　Digital output　振動出力（TR介しアクチュエータ駆動）   */
```

```
/*    3   GP4/AN3    Digital output    電鍵出力（半導体リレー）          */
/*    4   GP3/MCLR   Digital input     モード切替PBSW入力（予備）        */
/*    5   GP2/AN2    Analog  input     送出スピード設定用VR電圧          */
/*    6   GP1/AN1    Analog  input     短点感圧素子電圧入力              */
/*    7   GP0/AN0    Analog  input     長点感圧素子電圧入力              */
/*    8   Vss        電源GND                                           */
/*                                                                     */
/
**********************************************************************
**/
/*        &GPIO3   portA                    portc.b2          */
/*        GPIO.F5        portA.F5                                  */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/*                                                                 */
/
**********************************************************************
**/
/* mikroC 設定                                                     */
/*                                                                 */
/* 【Project Settings】                                            */
/*                                                                 */
/*    Device Name  P12F675                                         */
/*    Clock Freq.  8MHz                                            */
/*    Build Debug  Release , Software                              */
/*                                                                 */
/*  【Library Manager】     ADC,Button,Conversion,C_String,EEProm   */
/*                                                                 */
/*    ADC,Button,Conversion,C_String,EEProm,Sound,Time             */
/*                                                                 */
/*  【Edit Project】                                               */
/*                                                                 */
/*     Oscillator Selection : INTOSC :I/O function on GP4/OSC2/    */
/*     WatchdogTimer : Disabled                                    */
/*     Power-Up Timer : Disabled                                   */
/*     GP3/MCLR pin function select : Disabled                     */
/*     Brown-out Detect : Enabled                                  */
/*     Code Protection : ProgramMemory code protection is disabled */
/*     Data Cpde protection : Data mamory code protection is disabled */
/*     Heap Size 2000 (automatically setting)                      */
/*                                                                 */
/
**********************************************************************
**/
/* 開発記録                                                        */
/*                                                                 */
/* 2015.06.04 開始                                                */
/* 2015.06. 一応動作は可能になったが、、、                         */
/* 2015.08.10   割込み動作に変更                                   */
/*      22:12  SpeedVR   LED点滅 modeSW 動作 OK                    */
/* 2015.08.11   一応完成 Ver.0.1                                   */
/* 2015.08.15 dot通過しdash圧力移行の場合をcase2に追加記述         */
```

```
/*          これに伴い、KeyPressH の値を300から400に増加した              */
/*          さらにDash直後のdotを拾いやするするため一時的にKeyPressLを変更 */
/
*=============================================================================
=*/
/* 2017.04.29  PressElecKeyProgを元にをベースに新プロジェクトに移行          */
/*     04.30  PIC16F1823に決定　ピン割り当て検討      12F675コードは正常    */
/*     05.01  プロジェクトを12F675用と16F1823用に2分岐した  これは16F1823   */
/*                                                                          */
/
*****************************************************************************
**/


unsigned int adc_rd;              // AD変換用一時変数

unsigned v_KeyDownP;               // Key押下圧力 格納変数
unsigned  v_Speed;                // 送出スピードVR電 圧格納変数
unsigned int speed;               // 送出スピードCNT対応変数

char Kdown;                       // ストレートキー押下       UP=0,DOWN=1
char mode;                        // エレキーモード（１）     UP=0,DOWN=1  <NotUse>

//   char KeyPos;                   // キー位置　UP=0,DOWN=1
char KeyR;                        // パドル右　OFF=0,ON=1
char KeyL;                        // パドル左　OFF=0,ON=1

char sendf = 0;                   // Dot / Space 送信後だけ１          <NotUse>
char dash  = 0;                   // dash送信後のみ１                  <NotUse>

char Beep = 1;                    //                                   <NotUse>


int KeyPressL = 800;              // keyL ON 判定閾値（大きいと感度高）
int KeyPressR = 800;              // keyR ON 判定閾値  ex800


int Silent =0;                   //                                   <NotUse>

//変数定義　状態遷移                // State=0 KEYdown 待ち受け
char State;                       // State=1 短点送出
                                  // State=2 （短間送出）
                                  // State=3 補長点送出中
                                  // State=4 短点延長出力中
                                  // State=5 予備


//変数定義　割り込み処理
unsigned int cnt;                 // 変数定義：外部割込カウンタ
```

```
int  DotLength   = 5;                // 外部割込カウンタcntを単位とする
int  DashLength  =200;



/
* ------------------------------------------------------------------------
*/
// interrupt()  タイマー割込処理    5msecインターバルタイマ
/
* ------------------------------------------------------------------------
*/
void interrupt() {

   if (INTCON.T0IF) {              // Timer0割込みの場合
      INTCON.T0IF = 0;             // 割込みフラグクリア
      TMR0 = -32;                  // TMR0 値再設定 160回(for 5mS Up Count)

      cnt--;                       // 外部割込回カウンタ カウントダウン


/*    if(Button(&GPIO,2,1,0)){       // 電鍵読み込み   KeyDownに格納   1で押下

     gpio.f4 = 1;  // for Debug
     KeyDown = 1;
    }
   else {
     gpio.f4 = 0;  // for Debug
     KeyDown = 0;
    }

*/
   }

}




/
* ------------------------------------------------------------------------
*/
// ReadKeyDownP()   KEY圧力からのの電圧を読み、変数   v_Key に格納する
/
* ------------------------------------------------------------------------
*/
void ReadKeyDownP()
{

                              // パドル右   チェック
   adc_rd  = ADC_Read(0);          // アナログポートAN0 よりAD変換
   v_KeyDownP = adc_rd;            // long に変換し格納


   if (v_KeyDownP <= KeyPressR) KeyR=1;      // KeyUp   なら 0 (NA)
```

```
    else KeyR=0;                        // KeyDown なら 1 (Dot)

                                // パドル左　チェック
   adc_rd  = ADC_Read(1);           // アナログポートAN1 よりAD変換
   v_KeyDownP = adc_rd;             // long に変換し格納


   if (v_KeyDownP <= KeyPressL) KeyL=1;      // KeyUp なら 0 (NA)
    else KeyL=0;                          // KeyDown　なら 1 (Dash)

}

/
* ------------------------------------------------------------------------
*/
// ReadSpeedVR()　スピード調整用VRを読み、変数v_Speed に格納する
/
* ------------------------------------------------------------------------
*/
void ReadSpeedVR()
{
   adc_rd  = ADC_Read(2);             // アナログポートAN0 よりAD変換
   v_Speed = adc_rd;                 // long に変換し格納

   speed = (v_Speed /5 )+1;           //  元　 (v_Speed / 5)+ 5;

}



/
* ------------------------------------------------------------------------
*/
// ReadKeySW()　ストレートキースイッチを読み変数　Kdown に格納する
/
* ------------------------------------------------------------------------
*/
void ReadKeySW()
{
   if(Button(&portA,3,1,0)){        // RA0 スイッチHiでKdown 0 LoでKdown 1
     Kdown = 1;
     }
   else {
     Kdown = 0;
     }

}


/
* ------------------------------------------------------------------------
*/
// BeepShort()　短点ビープ
/
* ------------------------------------------------------------------------
*/
```

```
void BeepShort()
{
int i;

    if(Beep==1){

    for (i=0;i<50;i++){
       portA.F5 = 1;
       delay_us( 400 );
       portA.F5 = 0;
       delay_us( 400 );
      }
    }

}

/
* ------------------------------------------------------------------------
*/
// BeepLong()   長点ビープ
/
* ------------------------------------------------------------------------
*/
void BeepLong()
{
int i;

    if(Beep==1){

    for (i=0;i<150;i++){       //
       portA.F5 = 1;            //
       delay_us( 400 );
       portA.F5 = 0;
       delay_us( 400 );
      }

     }

}

/
* ------------------------------------------------------------------------
*/
// BeepSemiLong()   長点ビープ
/
* ------------------------------------------------------------------------
*/
void BeepSemiLong()
{
int i;

    if(Beep==1){

    for (i=0;i<100;i++){            // アナログポートAN2 よりAD変換
       portA.F5 = 1;            // long に変換し格納
       delay_us( 400 );
       portA.F5 = 0;
       delay_us( 400 );
      }
```

```
    }
}



/
* ------------------------------------------------------------------------
*/
// main()  メインルーチン
/
* ------------------------------------------------------------------------
*/

unsigned temp;


void main() {

//    【重要Project設定】メニューバーEdit Project にて設定する必要あり
//    Oscilator Selection           INTOSC oscilator  内部クロックに設定
//    MCU and Oscllator Frequency    500kHz使用に設定（ OSCCON = 0b01110000; ）
//    Power-up Timer Enable          Enable 使用するに設定
//    MCLR Pin Function Select       Disabled   リセットではなく、入力ピンとして使用
する
//    Clock Out Enabele              Disabled
//    Internal/External Switchover  Disabled
//    Fail-Safe Clock Monitor        Disabled
//    Flash Memory Self-Write Protection    Disabeled
//  PLL enable                     Disabled
//  Stack overflow/Underflow Reset Enable       Enabled
//    Brown-out Reset Enabel         Enabled 使用するに設定
//  In-Circuit Debugger Mode       Disabled
//  Low-Voltage Programming Enable         Disabled
//
//  configure VDD as Vref, リセット後は自動でVddになる



int i;

/
*-------------------------------------------------------------------------
-------*/
// 初期化                         // 初期化開始
/
*-------------------------------------------------------------------------
-------*/

  intcon=0;                      // 割り込み禁止

    PORTA = 0b00000000;             //PortA初期状態設定
    PORTC = 0b00000000;             //PORTC初期状態設定
```

```
    TRISA = 0b00001111;             //PORT A 0,1,2,3(MCLR)を1:入力に設定 4,5は0:出力
    TRISC = 0b00000000;             //PORT C 0,1,2,3,4,5 を0:全Cポートは出力に設定

// A/D利用PORTの設定 ANALOG=1, DIGITAL=0      ※必須※

    ANSELA = 0b00000111;            //AN0（長点感圧）,AN1（短点感圧),AN2（送出速
度)
    ANSELC = 0b00000000;            //RC0～RC5 は全デジタルで使用  RA3,RA4,RA5も
デジタル

  CM1CON0 = 7;                    // コンパレータ機能OFF

  ADCON0 =0b1100000;              // ADコン有効、電源電圧を基準電圧とする


//  ADCON0.ADFM = 1;

    OPTION_REG = 0x84; //プリスケーラ値設定0x84(=32回)
    TMR0 = -160; //TMR0カウント値設定 160回(アップカウンタ)

    INTCON.T0IE = 1; //タイマ割込み許可
    INTCON.GIE = 1; //全体割込み許可

    ADC_Init();                             // ADコン初期化

   cnt = DotLength;                             //外部割込カウンタ値設定
                                              // 1 秒(=5msec * 200回)


                                // 初期化終了


  do {                          // プログラムループ開始点

        ReadKeyDownP();

//        ReadSpeedVR();


        if(cnt==0) {                        // 割込み処理
//                  portC.F5 = ~portC.F5;
//                  portA.F5 = ~portA.F5;       // バイブレーション
                  cnt=1 ;                   //  cnt=speed;


          if(portA.F3==0) {
                  portC.F5 = 1;             // モニターLED点灯
                  portA.F5 = ~portA.F5;       // バイブレーション
           }
```

```
        else if(KeyR==1) {
                portC.F5 = 1;               // モニターLED点灯
                portA.F5 = ~portA.F5;       // バイブレーション
         }
        else if(KeyL==1) {

//                portC.F5 = 1;               // モニターLED点灯
//                portA.F5 = ~portA.F5;       // バイブレーション

                portC.F5 = 1;               // モニターLED点灯
                portA.F5 = 1;               // クリックバイブ

//                delay_ms(10);

                portA.F5 = 0;
                portC.F5 = 0;               // モニターLED消灯

         }
        else    {
                portC.F5 = 0;               // モニターLED消灯
                portA.F5 = 0;       // バイブレーション OFF
               }




//          if(KeyL==1) portC.F5 = 1;               // モニターLED消灯
//            else      portC.F5 = 0;               // モニターLED消灯



/*
        if(sendf==1) {                  // 直前がdotかdashの場合

            cnt=  speed;
            portA.F5 = 0;               // 短スペースを挿入する
            sendf =0;
            dash =0;
          }
        else {


         switch(KeyPos){

           case 0:{
             cnt=  speed;
             portA.F5 = 0;

             dash =0;

             break;

            }
```

```
                case 1:{

                  cnt=  speed;
                  portA.F5 = 1;
                  sendf = 1;

                  dash =0;

                  BeepShort();
                  break;

                 }

                case 2:{

                 if(sendf ==1){                 // dot通過しdash圧力移行の場合
                  if(portA.F5 ==1 ){
                   cnt= (speed*2);
                   portA.F5 = 1;
                   sendf = 1;
                   dash =1;
                   BeepSemiLong();


                   break;

                  }

                 }

                  cnt= (speed*3);
                  portA.F5 = 1;
                  sendf = 1;
                  dash =1;
                  BeepLong();
                  break;

                 }

              }
           }
*/

         }


/*       ReadModeSW();

       if(mode ==1) Beep=1;
       else Beep=0;
*/



   } while (1);              // endless loop (as this condition is always satisfied)
}
```