

KeyPad(4x4)ユニット(I2C対応)

概要

前回製作したKeyPad(4x4)を、I2C対応に改良し、汎用的に使えるようにしました。

動作原理

I2Cのスレーブ機能としての、基本的な構造は、前回製作したLCDモニタ(I2C対応)と同じです。KeyPadの制御方式(キーマトリクススキャン)は、前回製作したKeyPad(4x4)と同じです。

<メモリ構造> KeyPadの押下されたキーのデータを設定するために、1バイトのメモリを使用します。0x00は押下キーデータ

- 押下時には"1234567890ABCD*#"の何れかがセットされます。
- 非押下時には0x00がセットされます。
 - 本ユニットが、一定時間(200msec)経過後に自動的に0x00をセットする方法(SW_MODE(0))
 - マスターからの操作によって0x00をセットする方法(SW_MODE(1))

<処理の流れ>

1. KeyPadをキーマトリクススキャンする。(何れかのキーが押下されるまで繰り返す)
2. 押下されたキーの内容を、メモリ0x00にセットする。
3. SW_MODE(1)の場合には、1.へ戻る。
4. SW_MODE(0)の場合には200msec後に、メモリ0x00に0x00をセットする。

<マスター側から、本ユニットの押下キーデータを、取得する方法>

1. 0x00は押下キーデータの内容を読み込む。

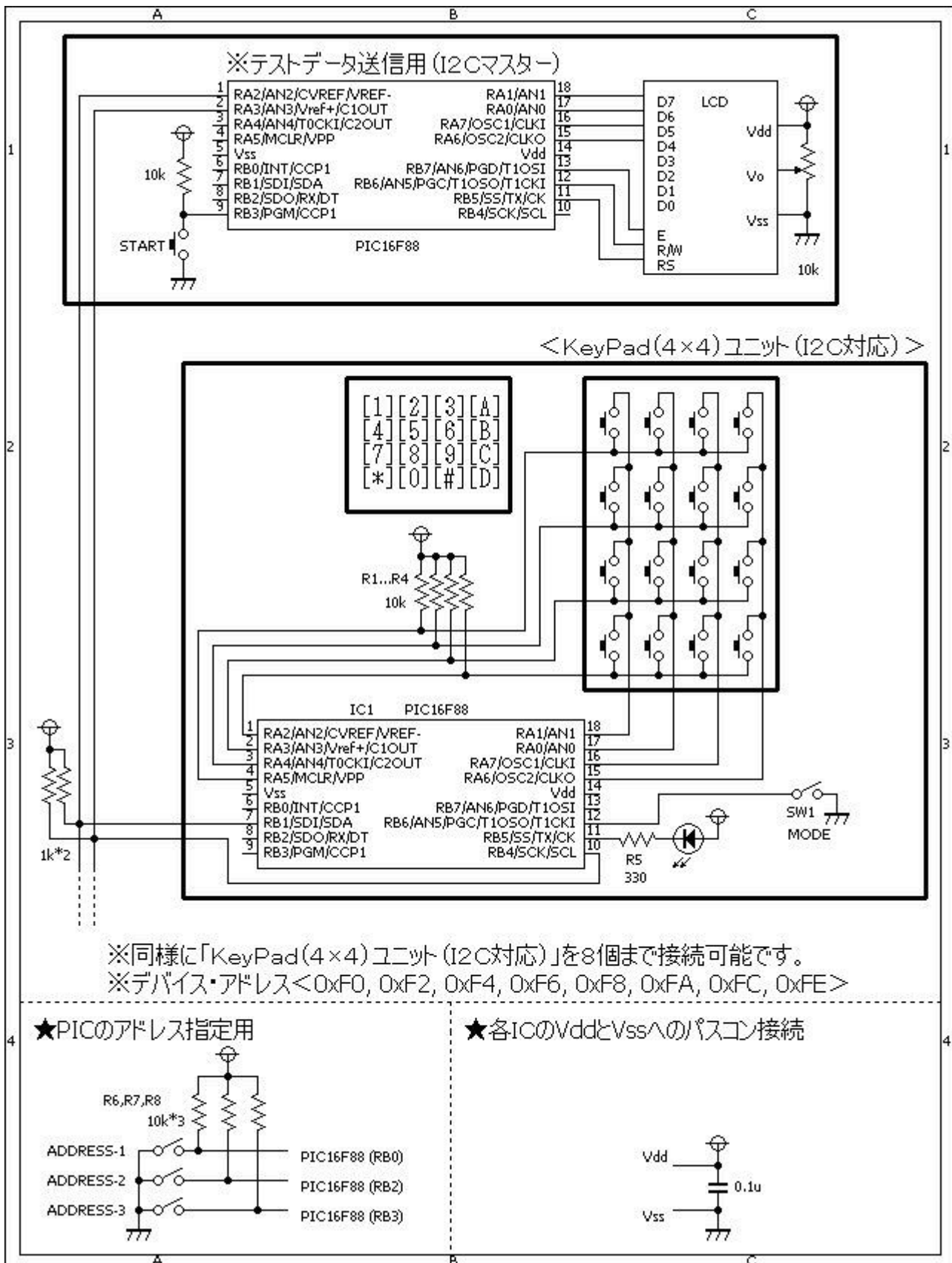
```
Soft_I2C_Start();
Soft_I2C_Write(0xF0);
Soft_I2C_Write(0x00);
Soft_I2C_Start();
Soft_I2C_Write(0xF1);
kp = Soft_I2C_Read(NO_ACK); //押下キーデータを読み込んでkpにセットする。
Soft_I2C_Stop();
```

2. キーが押されているかを確認する。
 - kpが、0x00であれば、1.へ戻る。
 - kpが、0x00以外であれば、キーが押されているので、3.へ。
3. 0x00は押下キーデータの内容をクリアする。

```
Soft_I2C_Start();
Soft_I2C_Write(0xF0);
Soft_I2C_Write(0x00);
Soft_I2C_Write(0x00); //押下キーデータを、0x00でクリアする。
```

Soft_I2C_Stop();

回路図



※同様に「KeyPad(4x4)ユニット (I2C対応)」を8個まで接続可能です。
 ※デバイス・アドレス<0xF0, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, 0xFE>

ソースコード

KeyPad_i2c.c

```
//*****
*
/*
   □□□□□□□□×□□□□□□対応) >
*/
//*****
*

#define      LED          PORTB.F5

#define      SW_ADDR1    PORTB.F0
#define      SW_ADDR2    PORTB.F2
#define      SW_ADDR3    PORTB.F3

#define      SW_MODE      PORTB.F6

#define      ADDR_BASE    0xF0

#define      ON           0
#define      OFF          1

#define      DATA_SIZE   1

//*****
*

void  i2c_Write(unsigned short dat)
{
    while (SSPSTAT.BF == 1)
        ;
    while (1) {
        SSPCON.WCOL = 0;
        SSPBUF = dat;
        if (SSPCON.WCOL == 1)
            continue;
        //
        SSPCON.CKP = 1;
        return;
    }
}

//*****
*

static unsigned  short  i2c_memory[DATA_SIZE], i2c_pnt, i2c_flg,
i2c_tmp;
```

```
//*****  
*  
void i2c_Handler()  
{  
    if (SSPSTAT.P == 1) //ストップビットを検出しても何もしない。  
        asm nop;  
    if (SSPSTAT.S == 1) //スタートビットを検出しても何もしない。  
        asm nop;  
    //  
    i2c_tmp = SSPSTAT & 0b00101101;  
    //  
    if (i2c_tmp == 0b00001001) { //書き込みモード、デバイスアドレス  
        i2c_tmp = SSPBUF;  
        i2c_flg = 0;  
        i2c_pnt = 0;  
        return;  
    }  
    if (i2c_tmp == 0b00101001) { //書き込みモード、データ  
        if (i2c_flg == 0) {  
            i2c_pnt = SSPBUF;  
            i2c_flg = 1;  
            return;  
        }  
        if (i2c_flg == 1) {  
            i2c_memory[i2c_pnt] = SSPBUF;  
            i2c_pnt++;  
            return;  
        }  
    }  
    if (i2c_tmp == 0b00001100) { //読み込みモード、デバイスアドレス  
        i2c_Write(i2c_memory[i2c_pnt]);  
        i2c_pnt++;  
        return;  
    }  
    if (i2c_tmp == 0b00101100) { //読み込みモード、データ□□□□□  
        i2c_Write(i2c_memory[i2c_pnt]);  
        i2c_pnt++;  
        return;  
    }  
    if (i2c_tmp == 0b00101000) { //読み込みモード、データ□□□□□□□□  
        i2c_tmp = SSPBUF;  
        SSPCON = 0b00111110;  
        return;  
    }  
}  
  
//*****  
*
```

```
void interrupt()
{
    if (PIR1.SSPIF == 1) {
        PIR1.SSPIF = 0;
        //
        LED = ON;
        i2c_Handler();
        LED = OFF;
    }
}

//*****
*

unsigned short keyPad_keySearch()
{
    unsigned short kp;
    //
    kp = 0x00;
    //
    PORTA.F6 = 0;
    switch (PORTA & 0b00111100) {
    case 0b00111000:
        kp = 'D';
        break;
    case 0b00110100:
        kp = 'C';
        break;
    case 0b00101100:
        kp = 'B';
        break;
    case 0b00011100:
        kp = 'A';
        break;
    }
    PORTA.F6 = 1;
    //
    PORTA.F7 = 0;
    switch (PORTA & 0b00111100) {
    case 0b00111000:
        kp = '#';
        break;
    case 0b00110100:
        kp = '9';
        break;
    case 0b00101100:
        kp = '6';
        break;
    case 0b00011100:
        kp = '3';
        break;
    }
```

```
    }
    PORTA.F7 = 1;
    //
    PORTA.F0 = 0;
    switch (PORTA & 0b00111100) {
    case 0b00111000:
        kp = '0';
        break;
    case 0b00110100:
        kp = '8';
        break;
    case 0b00101100:
        kp = '5';
        break;
    case 0b00011100:
        kp = '2';
        break;
    }
    PORTA.F0 = 1;
    //
    PORTA.F1 = 0;
    switch (PORTA & 0b00111100) {
    case 0b00111000:
        kp = '*';
        break;
    case 0b00110100:
        kp = '7';
        break;
    case 0b00101100:
        kp = '4';
        break;
    case 0b00011100:
        kp = '1';
        break;
    }
    PORTA.F1 = 1;
    //
    return (kp);
}

//*****
*

unsigned short keyPad_GetKey()
{
    unsigned short newkp, oldkp, cnt;
    //
    oldkp = 0x00;
    //
    for (cnt = 0; cnt < 10; cnt++) {
        newkp = keyPad_keySearch();
    }
}
```

```

    if ((newkp == 0x00) || (newkp != oldkp))
        cnt = 0;
    oldkp = newkp;
    Delay_ms(5);
}
//
for (cnt = 0; cnt < 10; cnt++) {
    if (keyPad_keySearch() != 0x00)
        cnt = 0;
    Delay_ms(5);
}
//
return (newkp);
}

//*****
*

static unsigned short Luminance[10] = {255, 128, 64, 32, 16, 8,
4, 2, 1, 0};

void main()
{
    unsigned short cnt;
    //
    CMCON = 0b00000111; //コンパレータは使用しない。
    ANSEL = 0b00000000; //A/Dコンバータは使用しない。
    OSCCON = 0b01110000; //クロックは内臓8MHzを使用する。
    TRISA = 0b00111100; //PORTAを設定する。
    TRISB = 0b01011111; //PORTBを設定する。
    OPTION_REG.NOT_RBPU = 0; //PORTBをプルアップする。
    //□□□を設定する。
    SSPSTAT.SMP = 1;
    SSPSTAT.CKE = 1;
    SSPCON.WCOL = 0;
    SSPCON.SSPOV = 0;
    SSPCON.SSPEN = 1;
    SSPCON.CKP = 1;
    SSPCON.SSPM0 = 0;
    SSPCON.SSPM1 = 1;
    SSPCON.SSPM2 = 1;
    SSPCON.SSPM3 = 1;
    SSPADD = ADDR_BASE + ((SW_ADDR3 == 1) ? 8 : 0) + ((SW_ADDR2 == 1) ?
4 : 0) + ((SW_ADDR1 == 1) ? 2 : 0);
    PIE1.SSPIE = 1;
    PIR1.SSPIF = 0;
    //
    //
    PORTA.F6 = 1;
    PORTA.F7 = 1;
    PORTA.F0 = 1;

```

```

PORTA.F1 = 1;
//
LED = OFF;
i2c_pnt = 0;
i2c_flg = 0;
i2c_memory[0] = 0x00;
//
for (cnt = 0; cnt < 10; cnt++) {
    LED = ON;
    Delay_ms(50);
    LED = OFF;
    Delay_ms(50);
}
// 割り込み(全体)の設定
INTCON.PEIE = 1;
INTCON.GIE = 1;
//
while (1) {
    i2c_memory[0] = keyPad_GetKey();
    //
    if (SW_MODE == 0) {
        Delay_ms(200);
        i2c_memory[0] = 0x00;
    }
}
}

//*****
*
```

<参考> テストデータ送信用(I2Cマスター/PIC16F88)のプログラムです。

keyPad_i2c_master.c

```

//*****
*
/*
   □□□□□□□□×□□のテストデータ送信用(マスター) >
*/
//*****
*

#define      ACK          1
#define      NO_ACK      0

//*****
*

void  SwitchONcheck()
{
```

```
while (Button(&PORTB, 3, 1, 0) == 0)
;
while (Button(&PORTB, 3, 1, 1) == 0)
;
}

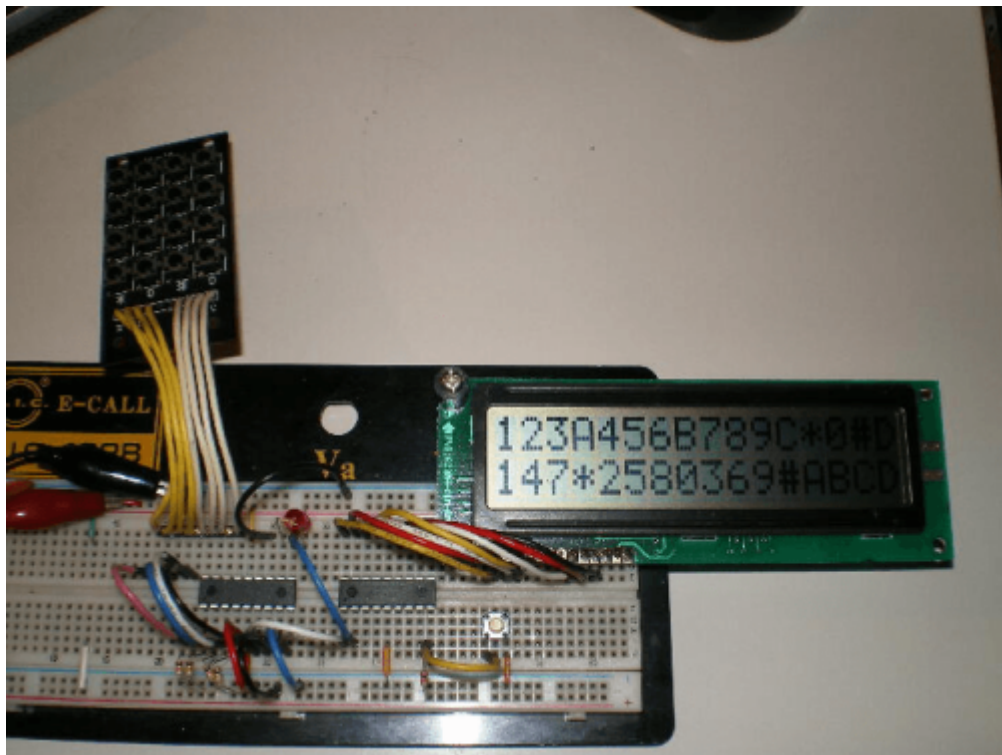
//*****
*

void main()
{
    unsigned    short    kp, l_cnt, c_cnt, cnt;
    //
    OSCCON = 0b01110000;    // クロックを8Mhzに設定する。
    ANSEL = 0b00000000;    // □□□変換は使用しない。
    TRISA = 0b00110000;
    TRISB = 0b00001111;
    //
    Lcd_Custom_Config(&PORTA, 1, 0, 7, 6, &PORTB, 5, 6, 7);
    Lcd_Custom_Cmd(LCD_CURSOR_OFF);
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(1, cnt + 1, 0xFF);
        Delay_ms(50);
    }
    for (cnt = 0; cnt < 16; cnt++) {
        Lcd_Custom_Chr(2, cnt + 1, 0xFF);
        Delay_ms(50);
    }
    Lcd_Custom_Cmd(LCD_CLEAR);
    //
    l_cnt = 1;
    c_cnt = 1;
    //
    Soft_I2C_Config(&PORTA, 2, 3);    // SDA, SCL
    //
    while (1) {
        //
        Soft_I2C_Start();
        Soft_I2C_Write(0xF0);
        Soft_I2C_Write(0x00);
        Soft_I2C_Start();
        Soft_I2C_Write(0xF1);
        kp = Soft_I2C_Read(NO_ACK);
        Soft_I2C_Stop();
        //
        if (kp == 0x00) {
            Delay_ms(100);
            continue;
        }
    }
}
```

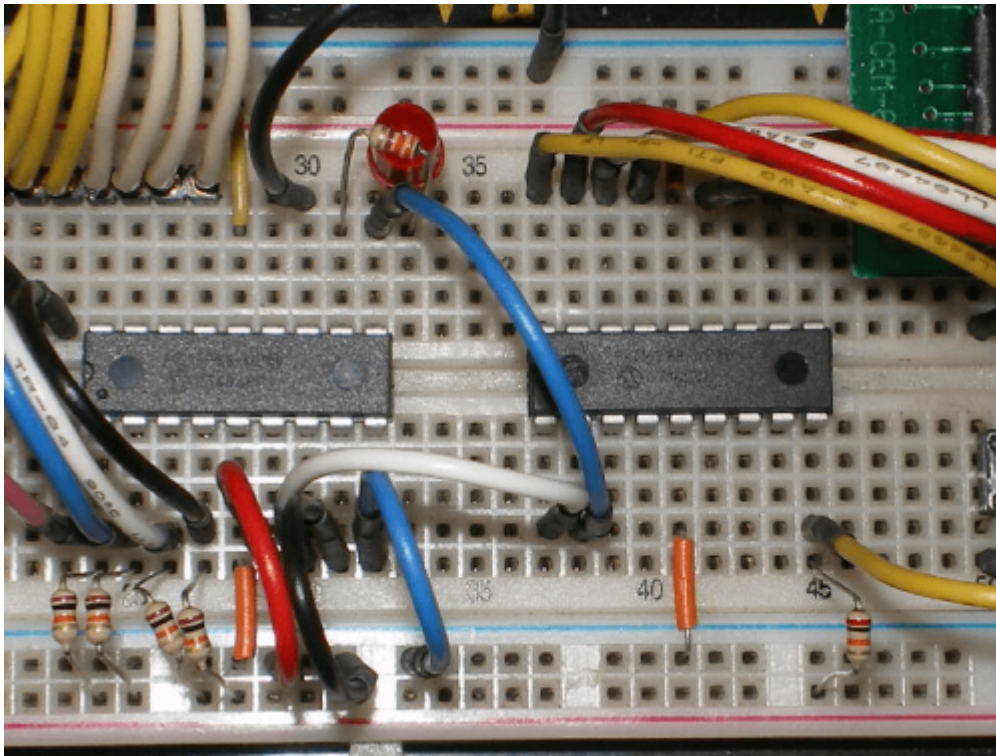
```
//
Soft_I2C_Start();
Soft_I2C_Write(0xF0);
Soft_I2C_Write(0x00);
Soft_I2C_Write(0x00);
Soft_I2C_Stop();
//
Lcd_Custom_Chr(l_cnt, c_cnt, kp);
c_cnt++;
if (c_cnt == 17) {
    c_cnt = 1;
    l_cnt++;
    if (l_cnt == 3)
        l_cnt = 1;
}
Delay_ms(100);
}
}

//*****
*
```

動作確認



左側が、スレーブ(本ユニット)、右側が、マスター側です。



KeyPadのスイッチを、横順、縦順で押していくと、その結果が、LCDに順次表示されていきます。



著作権表示 **copyright notice**

このページは稲崎様の閉鎖したHPのコピーで、著作権は稲崎様にあります。[詳細](#) This page is a copy of Mr. Inasaki's closed website, and the copyright is held by him.[Details](#)

From:

<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:100>

Last update: **2025/10/17 14:29**

