無線リモートコントローラ送信部 (品番RF-0299L-TX)説明書

概要

本キットは、マイクロチップ社ナノワット超低電力(XLP)テクノロジー、8ビットマイクロコントローラ、ミッドレンジ、エンハンスシリーズPIC 16LF1826を使用した無線リモートコントローラ(ラジコン)送信部(品番RF-0299L-TX)(完成品)です。

無線リモートコントローラ(ラジコン)受信部(品番RF-0299-RX)と組み合わせて、315メガ帯での無線によるリモコン制御ができます。

キーパッド(品番ACC-0295-KEY4x4)は別売です。

特長

1. 消費電力が極めて小さい。約28μA@3V(スリープ時)
   スイッチが押されたとき約1-2mA@3V
2. 小型でローコスト
3. 16個のキースイッチを使って最大26種類のキーコード(5ビット)を送信。
4. 16個のキースイッチ中、二度押しスイッチが4個、四度押しスイッチが2個、単発スイッチが10個ある。
5. デバイスコードは、4ビットDIPスイッチを使って14個のデバイスを設定できる。
6. 通信可能距離は直線で約50m(注1)
7. 動作電源電圧は2.2V-3.3V(単三電池2個。5V電源不可。)

準備するもの

1. 無線リモートコントローラ(ラジコン)送信部(品番RF-0299L-TX)
   完成品、ICはプログラム済み、動作確認済み
2. 無線リモートコントローラ(ラジコン)受信部(品番RF-0299-RX)
3. 単三電池2個　2P電池ケース1個
4. 3ピンのピンソケット(メス)1個
5. キーパッド(品番ACC-0295-KEY4x4)1個
6. 10cm程の線材(アンテナとして使用する)。

組み立て

1. キーパッドのピンヘッダCN1,CN2と本体のピンソケットCN1,CN2のコネクターを差し込む。
2. 2P電池ケースのプラス側を本体(CN4)3V、マイナス側をグランドGNDにピンソケットを使ってCN4に繋ぐ。
   極性に十分注意すること。
3. 本機に電源が入れば直ちにラジコン送信機として動作する。

デバイスコードの変更

デバイスコードは4PDIPスイッチSW1を使って設定する。1番がLSB(LEAST-SIGNIFICANT-BIT)、ONにするとTRUE(負論理で'0')となる。4ビット使用で"0001"から"1111"まで14種類の設定が可能。

"0001" "0010" "0011" "0100" "0101" "0110" "0111" "1001" "1010" "1011" "1100" "1101" "1110"(デフォルト) "1111"

但し、"0000","1000"(DIPスイッチ1,2,3番ON)は使用禁止。

例えば"1110"の場合DIPスイッチの1をON、2,3,4をOFFとする。

デバイスコードは受信側と必ず一致させる。

通信形式について

本機はEUSERT(Enhanced Universal Synchronous Receiver Transmitter)周辺回路モジュールを使用し、無線でシリアル送信を行う。

一個のデータは8ビットで、LSBからMSBの順にボーレートジェネレターのクロック信号に同期して自動的に送信されなる。

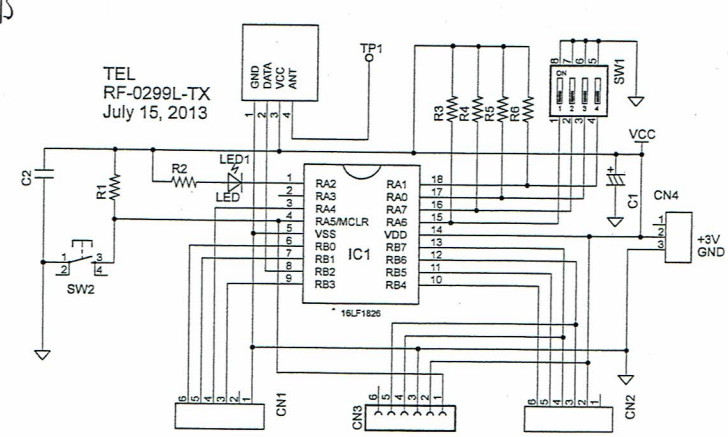
先ず、8個のデータを一回分として、最初に文字データ'T' ,'E', 'L' を送信し、続いて、デバイスコード、データコード、反転デバイスコード、反転データコード、最後に文字データ'E'を送信し一回分を終了する。

次に同じ8個のデータを三回、1mSの間隔で連続送信し、一回の通信を完了する。

プログラムについて

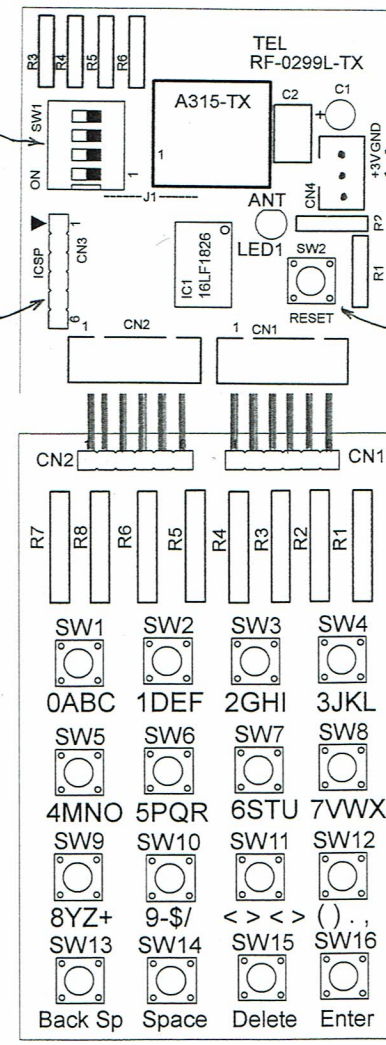
MPLAB XC8 C コンパイラー ライトモードV1.20を使用。
ROM(プログラムメモリ) 2048ワード中1172使用 約57%
RAM(データーメモリ) 884バイト中40バイト使用 約5%
参考資料としてCソースファイルのリスト(Listing)を添付する。
(注意)CONFIGワードのコードプロテクションビット はOFFになっており、ICプログラムメモリの上書き読込みが自由にできますが、ICSPを使用しプログラムを変更した場合には、保証できなくなりますから注意してください。また、プログラムに関するご質問はお受けできませんのでご了承ください。

注1:通信可能距離は直線は障害物、アンテナの長さ、形状などによって大きく変わってくる。受信側のアンテナについては制限がないが、送信側のアンテナを余り大きくすると電波法に抵触する可能性があるので注意すること。



無線リモートコントローラ送信部
(品番RF-0299L-TX)回路図

無線リモートコントローラ送信部
(品番RF-0299L-TX)本体



DIPスイッチ
3ビットデバイス
コードを設定
例えば"1110"は1番
をON,2,3,4番をOFF
とする。受信側も
"1110"と設定する
こと。

ICSP(In-Circuit
SerialProgramming)
用コネクター。
使用しないこと。

電源グランドGND
3V電源プラス側
(単三電池2個直列)

RESET(SW2)
リセットすると
二度押しキー、
四度押しキー
がデフォルト値
に戻る。
デフォルト値
(初期値)は
SW1-0,SW2-0,
SW3-0,SW4-0,
SW5-0,SW6-0,
SW7(単発SW)

ANT
10cm程の線材を
アンテナとして
ここに接続する。

LED1
スイッチを押すと
約32mS点灯する。

キーパット(別売)
(品番ACC-0295-KEY4X4)

TEL エレクトロニクス・キット (有)谷岡電子
〒164-0003 東京都中野区東中野1-51-13
大島ビル第一別館402 ☎ (03)3366-4552

無線リモートコントローラ送信部
(品番RF-0299L-TX)
Cソースファイルリスト(Listing)

```
/*****************************************************************
******* RF Wireless Remote Controller Transmitter *******
* Project name: RF0299TX
* File name: RF0299TX.c
* Features:
*  *nano watt extreme low power(XLP), 1.8μwatt typical,
*  *Micro controller PIC enhanced mid-range 16LF1826.
*  *3 bits device code
*  *Wide operating voltage, 2.0V-3.3V.
* Pin assignments:
*  *Device code input <RA6:RA7>,<RA1:RA0>
*  *Data Output  <RB2>
*  *the Keypad Input <RB7:RB4>
*  *the keypad Output <RB3,RA4> <RB1:RB0>
*  *LED Output   <RA2>
* Compiler: xc8 v1.12
* AUG 10, 2013. Coded by yuji Tanioka(TEL company)
*****************************************************************/
#include <xc.h>

#pragma config MCLRE = ON, CP = OFF, BOREN = OFF, WDTE = OFF,
#pragma config PWRTE = ON, CPD = OFF, CLKOUTEN = OFF, FCMEN = OFF
#pragma config WRT = OFF, STVREN = OFF, BORV = LO, LVP = OFF,
#pragma config FOSC = INTOSC, PLLEN = OFF

#define  _XTAL_FREQ     8000000  //8MHZ

typedef volatile bit Bool;                      //flags

#defineTRUE1
#defineFALSE0
#define ERROR0xFF
#define         REPEAT_TIME             2000    //1mS

//funcfion prototype declaration
void reg_init(void);                                    //F13
void rf_send(unsigned char num);                        //F7
void interrupt isr(void);                               //F17
unsigned char toggle_key_1(void);                       //F8-1-1
unsigned char toggle_key_2(void);                       //F8-1-2
unsigned char toggle_key_3(void);                       //F8-1-3
unsigned char toggle_key_4(void);                       //F8-1-4
unsigned char four_push_key_1(void);                    //F8-2-1
unsigned char four_push_key_2(void);                    //F8-2-2
unsigned char find_key(void);                           //F16
unsigned char rf_key_update(unsigned char num);         //F8
void putch(unsigned char ch);                           //F10
void led_on(void);                                      //F18

Bool    sw1 = FALSE, sw2 = FALSE, sw3 = FALSE, sw4 = FALSE,
        sw5 = FALSE, sw6 = FALSE, sw7 = FALSE, sw8 = FALSE,
        sw9 = FALSE, sw10 = FALSE, sw11 = FALSE, sw12 = FALSE,
        sw13 = FALSE, sw14 = FALSE, sw15 = FALSE, sw16 = FALSE;

Bool    key_flag = FALSE;

    /*Main function*/
void main(vpid)
{
    unsigned char  n, key_num, sIOCBF;

    reg_init();                                 //F13

    for(;;)
        {
            key_num = find_key();               //F16
            if(key_num != ERROR)
            {
```



```
                led_on();                       //LED ON
                n = rf_key_update(key_num);
                rf_send(n);                     //F7
            }
            LATB0 = 0; LATB1 = 0; LATA4 = 0; LATB3 = 0;
            if(key_flag == FALSE)
            {

                 IOCIE = 1;
                 SLEEP();                       //before sleep
                 sIOCBF = IOCBF;                //CLEAR ALL FLAG
                 sIOCBF ^= 0XFF;
                 IOCBF &= sIOCBF;
            }
        }
}

/*function definition F13, register initialization*/
void reg_init(void)
{
    OSCCON = 0b01110000;            //8Mhz,HF:IRCF=01110
    OSCTUNE = 0X00;
    TRISA = 0b11100011;             //<RA2:RA4>OUTPUT
    TRISB = 0b11110000;             //<RB7:RB4> INPUT
    ANSELA = 0X00; ANSELB = 0X00;   //DIGITAL I/O
    WPUA5 = 0;                      //WEAK PULL-UP RA5 DISABLED
    WPUB = 0XF0;                    //<RB7:RB4> WEAK PULL-UP
    PORTB = 0;
    OPTION_REG = 0b00000011;        //0x03
                //0-------          PULL UP ENABLED
                //--0-----          TIMER0 CLOCK SELECT, Fosc/4
                //----0---          PRESCALER IS ASSIGNED TO TIMER0
                //-----011          PRESCALER RATE 1:16
                //          TIMER0 debounce time 0.5μSx16x256=2.048mS
    TMR0 = 0;

    T2CON = 0b00000011;             //0x0B
          //-0011---                1:4 POSTSCALER TOUTPS<3:0>
          //------0--               TIMER2 OFF
          //-------11               PRESCALER IS 64 T2CKPS<1:0>]
    TMR2 = 0x00;                    //TIMER 2 CLEARE
    PR2 = 0xFF;                     //TOTAL LED ON TIME:
                                    //0.5μSx64x256x4=32.768mS
    TMR2IE = 1;                     //TIMER 2 ENABLED

    LATA = 0X04;                    //LATB = 0X00;
    PORTA = 0X04;                   //PORTB = 0X00;
    IOCIE = 1;                      //IOC WAKE-UP ENABLE
    IOCBN = 0XF0;                   //NEGATIVE GOING EDGE IOCB ENABLE
    IOCBF = 0X00;                   //CLEAR IOCB FLAG
    TXSTA = 0b00000000;             //0x24
          //-0------                TX9=0
          //--0-----                TXEN=0
          //---0----                SYNC=0
          //-----0--                BRGH=0

    BAUDCON = 0b00000000;           //0x08
            //---0----              //CLOCK POLARITY SELECT SCKP=0
            //----0---              //BRG16=0

    //SPBRGL = 0X40;                //BAUD RATE=2400,BRGH=1,BRG16=1
    //SPBRGH = 0X03;                //832(0X0340), ERROR 0.04%
    //SPBRGL = 0X0A;                //BAUD RATE=300,BRGH=1,BRG16=1
    //SPBRGH = 0X1A;                //6666(0X1A0A), ERROR 0%
    SPBRGL = 103;                   //BAUD RATE=1200,BRGH=0,BRG16=0
    SPBRGH = 0;                     //103 ERROR 0.16%

    RCSTA = 0b10000000;             //0x80
```



```
            //1-------              //Serial Port enable(SPEN)
            //-0------              //use ninth bit(RX9)
            //---0----              //CREN=0
            //----0---              //ADDEN=0

    TXIE = 0;                       //TX1 INTERRUPT DISABLED

    PEIE = 1;
    ei();
}

void rf_send(unsigned char num)     //F7
{
    unsigned char pin_code;

    if(num != ERROR)
    {

        pin_code = PORTA;
        pin_code &= 0xC3;           //CLEAR <RA5:RA2>
        pin_code = pin_code << 2 | pin_code >> 6; //rotate 2 bits

        IOCIE =0;
        TXEN = 1;

        putch('T');
        putch('E');
        putch('L');
        putch(pin_code);
        putch(num);
        putch(~pin_code);
        putch(~num);
        putch('E');

        _delay(REPEAT_TIME);                //1mS

        putch('T');
        putch('E');
        putch('L');
        putch(pin_code);
        putch(num);
        putch(~pin_code);
        putch(~num);
        putch('E');

        _delay(REPEAT_TIME);                //1mS

        putch('T');
        putch('E');
        putch('L');
        putch(pin_code);
        putch(num);
        putch(~pin_code);
        putch(~num);
        putch('E');

        TXEN = 0;
        IOCIE = 1;
    }
}

void putch(unsigned char ch)                            //F10
{
    while(!TXIF)
        continue;
    TXREG = ch;
}

```



```
void interrupt isr(void)                                //F17
{

    if(IOCIE && IOCIF)          //INTERRUPT ON CHANGE ON PORT B
    {
        IOCIE = 0;
        IOCIF = 0;
        key_flag = TRUE;
        TMR0 = 0;
        TMR0IE = 1;
    }

    if(TMR2IE && TMR2IF)
{
TMR2IF = 0;
TMR2ON = 0;
                LATA2 = 1;              //LED OFF
}

    if(TMR0IE && TMR0IF)                //TIMER0 INTERRUPT
    {
         unsigned char sPORTB;          //shadow register

         static unsigned char sw1_counter = 0, sw2_counter = 0,
            sw3_counter = 0, sw4_counter = 0, sw5_counter = 0,
            sw6_counter = 0, sw7_counter = 0, sw8_counter = 0,
            sw9_counter = 0, sw10_counter = 0, sw11_counter = 0,
            sw12_counter = 0, sw13_counter = 0, sw14_counter = 0,
            sw15_counter = 0, sw16_counter = 0;

        TMR0IF = 0;
        TMR0IE = 0;

        LATB0 = 0; LATB1 = 1; LATA4 = 1; LATB3 = 1;
        NOP();
        sPORTB = PORTB;
        sPORTB = ~sPORTB;
        sPORTB &= 0xF0;
        sPORTB >>= 4;

        if(sPORTB == 1)                 //2.048mSx8=16.38ms
            sw1_counter++;              //total deounce time
        else sw1_counter = 0;
        if(sw1_counter >= 9)    sw1_counter = 9;
        if(sw1_counter == 8)    sw1 = TRUE;     //PRODUCE A SINGLE PULSE
                                                //PER PUSH
        if(sPORTB == 2)
            sw2_counter++;
        else sw2_counter = 0;
        if(sw2_counter >= 9) sw2_counter = 9;
        if(sw2_counter == 8) sw2 = TRUE;

        if(sPORTB == 4)
            sw3_counter++;
        else sw3_counter = 0;
        if(sw3_counter >= 9) sw3_counter = 9;
        if(sw3_counter == 8) sw3 = TRUE;

        if(sPORTB == 8)
            sw4_counter++;
        else sw4_counter = 0;
        if(sw4_counter >= 9) sw4_counter = 9;
        if(sw4_counter == 8) sw4 = TRUE;

        LATB0 = 1; LATB1 = 0; LATA4 = 1; LATB3 = 1;

        NOP();
        sPORTB = PORTB;
```

```
        sPORTB = ~sPORTB;
        sPORTB &= 0xF0;
        sPORTB >>= 4;

        if(sPORTB == 1)
            sw5_counter++;
        else sw5_counter = 0;
        if(sw5_counter >= 9) sw5_counter = 9;
        if(sw5_counter == 8) sw5 = TRUE;

        if(sPORTB == 2)
            sw6_counter++;
        else sw6_counter = 0;
        if(sw6_counter >= 9) sw6_counter = 9;
        if(sw6_counter == 8) sw6 = TRUE;

        if(sPORTB == 4)
            sw7_counter++;
        else sw7_counter = 0;
        if(sw7_counter >= 9) sw7_counter = 9;
        if(sw7_counter == 8) sw7 = TRUE;

        if(sPORTB == 8)
            sw8_counter++;
        else sw8_counter = 0;
        if(sw8_counter >= 9) sw8_counter = 9;
        if(sw8_counter == 8) sw8 = TRUE;

        LATB0 = 1; LATB1 = 1; LATA4 = 0; LATB3 = 1;

        NOP();
        sPORTB = PORTB;
        sPORTB = ~sPORTB;
        sPORTB &= 0xF0;
        sPORTB >>= 4;

        if(sPORTB == 1)
            sw9_counter++;
        else sw9_counter = 0;
        if(sw9_counter >= 9) sw9_counter = 9;
        if(sw9_counter == 8) sw9 = TRUE;

        if(sPORTB == 2)
            sw10_counter++;
        else sw10_counter = 0;
        if(sw10_counter >= 9) sw10_counter = 9;
        if(sw10_counter == 8) sw10 = TRUE;

        if(sPORTB == 4)
            sw11_counter++;
        else sw11_counter = 0;
        if(sw11_counter >= 9) sw11_counter = 9;
        if(sw11_counter == 8) sw11 = TRUE;

        if(sPORTB == 8)
            sw12_counter++;
        else sw12_counter = 0;
        if(sw12_counter >= 9) sw12_counter = 9;
        if(sw12_counter == 8) sw12 = TRUE;

        LATB0 = 1; LATB1 = 1; LATA4 = 1; LATB3 = 0;

        NOP();
        sPORTB = PORTB;
        sPORTB = ~sPORTB;
        sPORTB &= 0xF0;
        sPORTB >>= 4;
```



```
        if(sPORTB == 1)
            sw13_counter++;
        else sw13_counter = 0;
        if(sw13_counter >= 9) sw13_counter = 9;
        if(sw13_counter == 8) sw13 = TRUE;

        if(sPORTB == 2)
            sw14_counter++;
        else sw14_counter = 0;
        if(sw14_counter >= 9) sw14_counter = 9;
        if(sw14_counter == 8) sw14 = TRUE;

        if(sPORTB == 4)
            sw15_counter++;
        else sw15_counter = 0;
        if(sw15_counter >= 9) sw15_counter = 9;
        if(sw15_counter == 8) sw15 = TRUE;

        if(sPORTB == 8)
            sw16_counter++;
        else sw16_counter = 0;
        if(sw16_counter >= 9) sw16_counter = 9;
        if(sw16_counter == 8) sw16 = TRUE;

        LATB0 = 0; LATB1 = 0; LATA4 = 0; LATB3 = 0;
        key_flag = FALSE;
    }
}                                           //INTERRUPT ISR ENDS HERE

/*function definition F8-1-1,-2,-3,-4, toggle key value */
unsigned char toggle_key_1(void)                //F8-1-1
{
    static unsigned char state_variable  = 0;
    unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 0;
        break;

            default: break;
    }
    return reval;
}

unsigned char toggle_key_2(void)                //F8-1-2
{
    static unsigned char state_variable  = 0;
    unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 0;
        break;
```



```

        default: break;
    }
    return reval;
}

unsigned char toggle_key_3(void)            //F8-1-3
{
    static unsigned char state_variable  = 0;
    unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 0;
        break;

        default: break;
    }
    return reval;
}

unsigned char toggle_key_4(void)                //F8-1-4
{
    static unsigned char state_variable  = 0;
    unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 0;
        break;

        default: break;
    }
    return reval;
}

/*function definition F8-2-1,-2, four(4) sequence key values */
unsigned char four_push_key_1(void)             //F8-2-1
{
    static unsigned char state_variable = 0;
    static unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 2;
        break;
```



```
        case 2:
        reval = 2;
        state_variable = 3;
        break;

        case 3:
        reval = 3;
        state_variable = 0;
        break;
        default: break;
    }
    return reval;
}

unsigned char four_push_key_2(void)             //F8-2-2
{
    static unsigned char state_variable = 0;
    static unsigned char reval;

    switch(state_variable)
    {
        case 0:
        reval = 0;
        state_variable = 1;
        break;

        case 1:
        reval = 1;
        state_variable = 2;
        break;

        case 2:
        reval = 2;
        state_variable = 3;
        break;

        case 3:
        reval = 3;
        state_variable = 0;
        break;
        default: break;
    }
    return reval;
}

/*function F8, select a key */
unsigned char rf_key_update(unsigned char num)              //F8
{
    unsigned char j = 0;
    unsigned char reval;

    switch(num)
    {
        case 1:
        j = toggle_key_1();
        if(j == 0)
            reval = 0;

        if(j == 1)
            reval = 1;
        break;

        case 2:
        j = toggle_key_2();
        if(j == 0)
            reval = 2;
        if(j == 1)
```

```
            reval = 3;
        break;

        case 3:
        j = toggle_key_3();
        if(j == 0)
            reval = 4;
        if(j == 1)
            reval = 5;
        break;

        case 4:
        j = toggle_key_4();
        if(j == 0)
            reval = 6;
        if(j == 1)
            reval = 7;
        break;

        case 5:
        j = four_push_key_1();
        if(j == 0)
            reval = 8;
        if(j == 1)
            reval = 9;
        if(j == 2)
            reval = 10;
        if(j == 3)
            reval = 11;
        break;

        case 6:
        j = four_push_key_2();
        if(j == 0)
            reval = 12;
        if(j == 1)
            reval = 13;
        if(j == 2)
            reval = 14;
        if(j == 3)
            reval = 15;
        break;

        case 7:
            reval = 16;
        break;

        case 8:
            reval = 17;
        break;

        case 9:
            reval = 18;
        break;

        case 10:
            reval = 19;
        break;

        case 11:
            reval = 20;
        break;

        case 12:
            reval = 21;
        break;

        case 13:
```



```
            reval = 22;
        break;

        case 14:
            reval = 23;
        break;

        case 15:
            reval = 24;
        break;

        case 16:
            reval = 25;
        break;
        default: reval = ERROR; break;
    }
    return reval;
}
/*function definition F16, find push button switch number  */
unsigned char find_key(void)                //F16
{
    unsigned char reval;
    if(sw1 == TRUE){
        sw1 = FALSE;
        reval = 1;
    }
    else if(sw2 == TRUE){
        sw2 = FALSE;
        reval = 2;
    }
    else if(sw3 == TRUE){
        sw3 = FALSE;
        reval = 3;
    }
    else if(sw4 == TRUE){
        sw4 = FALSE;
        reval = 4;
    }
    else if(sw5 == TRUE){
        sw5 = FALSE;
        reval = 5;
    }
    else if(sw6 == TRUE){
        sw6 = FALSE;
        reval = 6;
    }
    else if(sw7 == TRUE){
        sw7 = FALSE;
        reval = 7;
    }
    else if(sw8 == TRUE){
        sw8 = FALSE;
        reval = 8;
    }
    else if(sw9 == TRUE){
        sw9 = FALSE;
        reval = 9;
    }
    else if(sw10 == TRUE){
        sw10 = FALSE;
    reval = 10;
    }
    else if(sw11 == TRUE){
        sw11 = FALSE;
        reval = 11;
    }
    else if(sw12 == TRUE){
        sw12 = FALSE;
```



```
        reval = 12;
    }
    else if(sw13 == TRUE){
        sw13 = FALSE;
        reval = 13;
    }
    else if(sw14 == TRUE){
        sw14 = FALSE;
        reval = 14;
    }
    else if(sw15 == TRUE){
        sw15 = FALSE;
            reval = 15;
    }
    else if(sw16 == TRUE){
        sw16 = FALSE;
        reval = 16;
    }
    else reval = ERROR;
    return reval;
}

/*function definition F18, led on time*/
void led_on(void)                                   //F18
{
        PR2 = 0xFF;
        TMR2IE = 1;
TMR2ON = 1;
        LATA2 = 0;                                  //LED ON
}
```

表1 キーパッド各スイッチと出力(5ビットキーコード)の関係

コネクター番号	CN2-3	CN2-4	CN2-5	CN2-6	CN1-3	CN1-4	CN1-5	CN1-6
ポート	RB6	RB7	RB5	RB4	RB3	RB2	RA4	RB0
	3ビットデバイスコード(負論理) デイフォルト値='110'				5ビットキーコード(正論理)			
SW1-0	1	1	0	0	0	0	0	0
SW1-1	1	1	0	0	0	0	0	1
SW2-0	1	1	0	0	0	0	1	0
SW2-1	1	1	0	0	0	0	1	1
SW3-0	1	1	0	0	0	1	0	0
SW3-1	1	1	0	0	0	1	0	1
SW4-0	1	1	0	0	0	1	1	0
SW4-1	1	1	0	0	0	1	1	1
SW5-0	1	1	0	0	1	0	0	0
SW5-1	1	1	0	0	1	0	0	1
SW5-2	1	1	0	0	1	0	1	0
SW5-3	1	1	0	0	1	0	1	1
SW6-0	1	1	0	0	1	1	0	0
SW6-1	1	1	0	0	1	1	0	1
SW6-2	1	1	0	0	1	1	1	0
SW6-3	1	1	0	0	1	1	1	1
SW7	1	1	0	1	0	0	0	0
SW8	1	1	0	1	0	0	0	1
SW9	1	1	0	1	0	0	1	0
SW10	1	1	0	1	0	0	1	1
SW11	1	1	0	1	0	1	0	0
SW12	1	1	0	1	0	1	0	1
SW13	1	1	0	1	0	1	1	0
SW14	1	1	0	1	0	1	1	1
SW15	1	1	0	1	1	0	0	0
SW16	1	1	0	1	1	0	0	1