赤外線リモートコントローラ送信部
(品番IR-0298L-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を使用した赤外線リモートコントローラ送信部(品番IR-0298L-TX)(完成品)です。

LCD表示付リモートコントローラ受信部(品番IR-0294-RX)又は、表示なし赤外線リモートコントローラ受信部(品番IR-0298-RX)と組み合わせて、赤外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1mA@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、3ビットDIPスイッチを使って設定する。
6. LCD表示付リモートコントローラ受信部(品番IR-0294-RX)と組み合わせて使えばキースイッチ名を表示できる。
7. 通信可能距離は直線で約5m(直射日光は避ける)
8. 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. 表示なし赤外線リモートコントローラ送信部本体(品番IR-0298L-TX)完成品、ICはプログラム済み、動作確認済み
2. LCD表示付リモートコントローラ受信部(品番IR-0294-RX)又は、表示なし赤外線リモートコントローラ受信部(品番IR-0298-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

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リモコン送信機として動作する。

キーパッド各スイッチのデフォルト表示

キーパッドの各スイッチの機能と受信側に表示されるテキストを下記する。ただし、テキスト表示は表示つき受信部を使った時。表示なし受信部を使用した場合は、十進変換アダプターを接続すればキーパッドの各スイッチに対応した発光ダイオード(LED)が点灯する。

SW1(二度押しキー)　"DUOSW1-0"　"DUOSW1-1"
SW2(二度押しキー)　"DUOSW2-0"　"DUOSW2-1"
SW3(二度押しキー)　"DUOSW3-0"　"DUOSW3-1"
SW4(二度押しキー)　"DUOSW4-0"　"DUOSW4-1"
SW5(四度押)"QRTSW5-0"　"QRTSW5-1"　"QRTSW5-2"　"QRTSW5-3"
SW6(四度押)"QRTSW6-0""QRTSW6-1"　"QRTSW6-2"　"QRTSW6-3"
SW7(単発キー)　"UNOSW 7"　SW8(単発キー)　"UNOSW 8"
SW9(単発キー)　"UNOSW 9"　SW10(単発キー)　"UNOSW 10"
SW11(単発キー)　"UNOSW 11"　SW12(単発キー)　"UNOSW 12"
SW13(単発キー)　"UNOSW 13"　SW14(単発キー)　"UNOSW 14"
SW15(単発キー)　"UNOSW 15"　SW16(単発キー)　"UNOSW 16"

キーパッド各スイッチの使い方

1、二度押しスイッチはトグルスイッチとして使用できる。
   例えば、パワースイッチなどON/OFFスイッチ。
2、四度押しスイッチは4個の設定が必要な制御に使用できる。
   例えば、音量調整(切る、小、中、大)、風速調整(切る、弱、中、強)。
3、単発キーは十連のロータリースイッチとして使える。
   また、スライドスイッチ(2個以上のスイッチを使う)としても使用できる。
   初期値(デフォルト値)は10進変換アダプターで設定しSW7とした。
4、.リセット(RESET SW1)、電源投入時の二度押し四度押しキーのデフォルト値は、SW1-0,SW2-0,SW3-0,SW4-0,SW5-0,SW6-0である。

デバイスコードの変更

デバイスコードは4PDIPスイッチSW2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。3ビット使用で"001"から"111"まで7種類の設定が可能。4番ピン無使用。
"001"　"010"　"011"　"100"　"101"　"110"(デフォルト)　"111"
但し、"000"(DIPスイッチ,1,2,3番ON)は使用禁止。
例えば"110"の場合DIPスイッチの1番をON、2と3番をOFFとする。
デバイスコードは受信側と必ず一致させる。

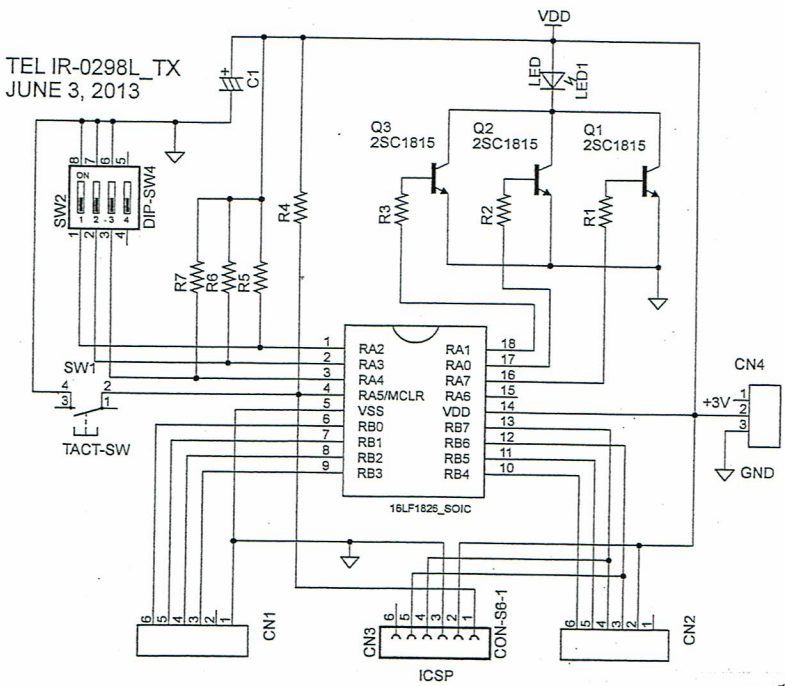
通信形式について

本機の赤外線リモートコントローラ通信形式次の通りである。
NRZ(NON- RETURN-TO-ZERO)方式なので変調信号ある間はHi(正論理)のPULSEが保持される。信号がない区間はLo、変調信号がある区間はHiパルスとなる。受信の方は信号がない区間がHi、変調信号がある 区間がLo(負論理)と認識される。
スタートから始まる1フレームは16ビットとなる。二度の送信照合は通信速度が遅くなるのでしていない。
送信順序はLSBからMSBの順になる。
変調周波数;約38Khz　1ビットパルス幅;600μS
<LSB>リーダーパルス;8ビットLo,Lo,Lo,Lo,LoLo,Lo,Lo(固定)
スタートパルス;1ビットHi(固定)　デバイスコード;3ビット
セパレータ;2ビットLo,Hi(固定)　キーコード;5ビット
ストップパルス;5ビットHi,Lo,Lo,Lo,Lo(固定)<MSB>

プログラムについて

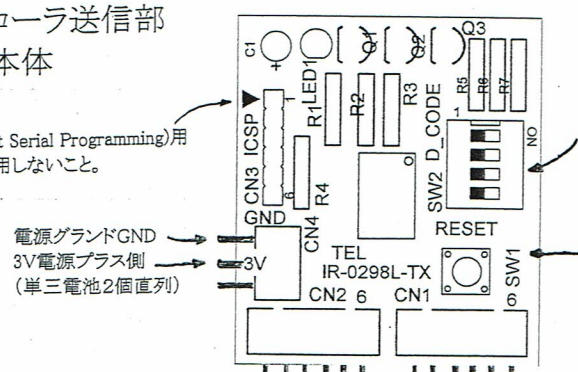
ハイテック C コンパイラー ライトモードV9.83を使用。
ROM(プログラムメモリ) 2048ワード中1577使用 約81.8%
RAM(データーメモリ) 256バイト中49バイト使用 約18.4%
参考資料としてCソースファイルのリスト(Listing)を添付する。
(注意)CONFIGワードのコードプロテクションビット はOFFになっており、ICプログラムメモリの上書き読込みが自由にできますが、ICSPを使用しプログラムを変更した場合には、保証できなくなりますから注意してください。また、プログラムに関するご質問はお受けできませんのでご了承ください。

TEL 赤外線リモートコントローラ送信部
(品番IR-0298L-TX)回路図



T.E.L. キット製造販売
(有)谷岡電子
〒164　東京都中野区東中野1-51-13
-0003　大島ビル第一別館402
☎03-3366-4552

赤外線リモートコントローラ送信部
(品番IR-0298L-TX)本体



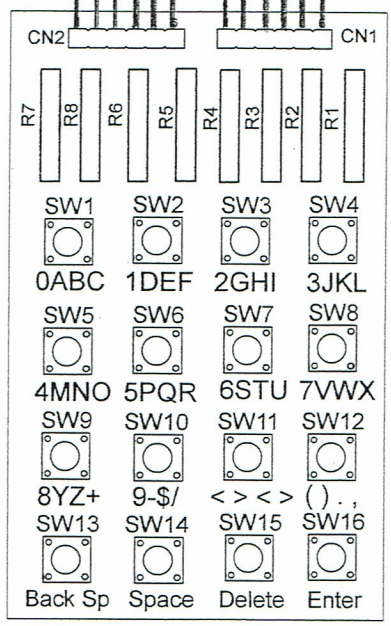
ICSP(In-Circuit Serial Programming)用
コネクター。使用しないこと。

電源グランドGND
3V電源プラス側
(単三電池2個直列)

DIPスイッチ
3ビットデバイスコードを設定
4番は不使用
例えば"001"は1番をOFF,2,3番をONとする。
受信側も"001"と設定すること。

RESET(SW1)
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

赤外線リモートコントローラ送信部
(品番IR-0298L-TX)組立図



キーパッド(別売)
(品番ACC-0295-KEY4X4)

赤外線リモートコントローラ送信部
(品番IR-0298L-TX)
Cソースファイルリスト(Listing)

C:\Documents and Settings\User\My Documents\TEL_kits\PIC_IR\ir_02981f1826.c

```
/*****************************************************************
******* Infrared Remote Controller Transmitter *******
* Project name: IR-0298L-TX
* File name: ir02981f1826_tx.c  v1.0
* Features:
*  *nano watt extreme low power(XLP), 1.8μwatt typical,
*  *Micro controller PIC enhanced mid-range 16LF1826.
*  *16 push button switch(4x4 key pad) and 3 bits device code
*  *Wide operating voltage, 2.0-3.3V.
* Pin assignments:
*  *Device code input <RA4:RA2>
*  *Output for the irLED <RA7> <RA1:RA0>
*  *Input for the key pad <RB7:RB4>
*  *Output for the key pad <RB3:RB0>
* Compiler: hi-tech c lite mode v9.83
* JUNE 3, 2013. Coded by yuji Tanioka(TEL company)
*****************************************************************/
#include <htc.h>

__CONFIG( MCLRE_ON & CP_OFF & BOREN_OFF & WDTE_OFF & FOSC_INTOSC &
          PWRTE_ON & CPD_OFF & CLKOUTEN_OFF & FCMEN_OFF );
__CONFIG( WRT_OFF & STVREN_OFF & BORV_LO & LVP_OFF & PLLEN_OFF );

#define  _XTAL_FREQ     8000000         //8MHZ

typedef volatile bit Bool;                          //flags

#define     TRUE            1
#define     FALSE           0
#define     ERROR           0xFF
#define     CYCLE_TIME      26

//funcfion prototype declaration
void reg_init(void);                                //F13
void ir_send(unsigned char num);                    //F7
void pin_send(unsigned char pin);                   //F7-3
void ir_leader(void);                               //F7-1
void ir_stop(void);                                 //F7-5
void interrupt isr(void);                           //F17
void hi_out(void);                                  //F7-7
void lo_out(void);                                  //F7-8
void key_send(unsigned char val);                   //F7-4
void ir_start(void);                                //F7-2
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 ir_key_update(unsigned char num); //F8

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

    reg_init();                                 //F13
    //Main loop
    for(;;)
    {
        key_num = find_key();                   //F16
        if(key_num != ERROR)
        {
            n = ir_key_update(key_num);
            ir_send(n);                         //F7
        }
        LATB = 0x00;
        if(key_flag == FALSE)
```


C:\Documents and Settings\User\My Documents\TEL_kits\PIC_IR\ir_02981f1826.c

```
        {
            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 = 0b01111100;                         //<RA0:RA1,RA7>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
    OPTION_REG = 0b000000011;                   //0x03
                //0-------             PULL UP ENABLED
                //--0-----             TIMER0 CLOCK SELECT, Fosc/4
                //----0---             PRESCALER IS ASSIGNED TO TIMER0
                //-----011             PRESCALER RATE 1:16
                //      TIMER0 debunce time 0.5μSx16x256=2.048mS
    TMR0 = 0;
    LATA = 0X00; LATB = 0X00;
    PORTA = 0X00; PORTB = 0X00;
    IOCIE = 1;                                  //IOC WAKE-UP ENABLE
    IOCBN = 0XF0;               //NEGATIVE GOING EDGE IOCB ENABLE
    IOCBF = 0X00;                               //CLEAR IOCB FLAG
    PEIE = 1;
    ei();
}

/*function definition F7, send out infrared signal */
void ir_send(unsigned char num)
{
unsigned char  pin_code, sPORTA;

    if(num != ERROR)
    {
        IOCIE = 0;
        sPORTA = PORTA;
        sPORTA &= 0x1C;
        sPORTA >>= 2;
        pin_code = sPORTA;
        ir_leader();                                //F7-1
        ir_start();                                 //F7-2
        pin_send(pin_code);                         //F7-3
        key_send(num);                              //F7-4
        ir_stop();                                  //F7-5
        IOCIE = 1;
    }
}

/*function definition F7-3, send ir dvice code */
void pin_send(unsigned char pin)                    //F7-3
{
    unsigned char i;
    pin &= 0x07;
    for(i = 3; i > 0; i--)
    {
        if(pin & 1 << (i-1))                        //testing i bit
            hi_out();                               //F7-7
        else
            lo_out();                               //F7-8
    }
    lo_out();                                       //separation
    hi_out();                                       //separation
}

void ir_leader(void)                                //F7-1
{
```

```
153     unsigned char i;
154     for(i = 0; i < 8; i++)
155         lo_out();
156 }
157
158
159 void ir_stop(void)                              //F7-5
160 {
161     unsigned char i;
162     hi_out();
163     for(i = 0; i < 4; i++)
164         lo_out();
165 }
166
167 void hi_out(void)                               //F7-7
168 {
169     unsigned char i;
170
171     for(i = 0; i < 23; i++)
172     {
173         LATA = 0x83;
174         _delay(CYCLE_TIME);
175         LATA = 0;
176         _delay(CYCLE_TIME);
177     }
178 }
179
180 void lo_out(void)                               //F7-8
181 {
182     unsigned char i;
183
184     for(i = 0; i < 23; i++)
185     {
186         LATA = 0;
187         _delay(CYCLE_TIME);
188         LATA = 0;
189         _delay(CYCLE_TIME);
190     }
191 }
192
193 void key_send(unsigned char val)                //F7-4
194 {
195     unsigned char i;
196     val &= 0x1f;
197     for(i = 5; i > 0; i--)
198     {
199         if(val & 1 << (i-1))                    //testing i bit
200             hi_out();                           //F7-7
201         else
202             lo_out();                           //F7-8
203     }
204 }
205
206 void ir_start(void)                             //F7-2
207 {hi_out();}
208
209
210
211 void interrupt isr(void)                        //F17
212 {
213
214     if(IOCIE && IOCIF)              //INTERRUPT ON CHANGE ON PORT B
215     {
216         IOCIE = 0;
217         IOCIF = 0;
218         key_flag = TRUE;
219         TMR0 = 0;
220         TMR0IE = 1;
221     }
222
223     if(TMR0IE && TMR0IF)            //TIMER0 INTERRUPT
224     {
225         unsigned char sPORTB;                   //shadow register
226         static unsigned char    sw1_counter = 0, sw2_counter = 0,
227             sw3_counter = 0, sw4_counter = 0, sw5_counter = 0,
228             sw6_counter = 0, sw7_counter = 0, sw8_counter = 0,
```



```
229             sw9_counter = 0, sw10_counter = 0, sw11_counter = 0,
230             sw12_counter = 0, sw13_counter = 0, sw14_counter = 0,
231             sw15_counter = 0, sw16_counter = 0;
232
233         TMR0IF = 0;
234         TMR0IE = 0;
235
236         LATB = 0x0E;
237         NOP();
238         sPORTB = PORTB;
239         sPORTB = ~sPORTB;
240         sPORTB &= 0xF0;
241         sPORTB >>= 4;
242
243         if(sPORTB == 1)                     //2.048mSx8=16.38ms
244             sw1_counter++;                  //total deounce time
245         else sw1_counter = 0;
246         if(sw1_counter >= 9) sw1_counter = 9;
247         if(sw1_counter == 8) sw1 = TRUE;    //PRODUCE A SINGLE PULSE
248                                             //PER PUSH
249         if(sPORTB == 2)
250             sw2_counter++;
251         else sw2_counter = 0;
252         if(sw2_counter >= 9) sw2_counter = 9;
253         if(sw2_counter == 8) sw2 = TRUE;
254
255         if(sPORTB == 4)
256             sw3_counter++;
257         else sw3_counter = 0;
258         if(sw3_counter >= 9) sw3_counter = 9;
259         if(sw3_counter == 8) sw3 = TRUE;
260
261     if(sPORTB == 8)
262             sw4_counter++;
263         else sw4_counter = 0;
264         if(sw4_counter >= 9) sw4_counter = 9;
265         if(sw4_counter == 8) sw4 = TRUE;
266
267         LATB = 0x0D;
268         NOP();
269         sPORTB = PORTB;
270         sPORTB = ~sPORTB;
271         sPORTB &= 0xF0;
272         sPORTB >>= 4;
273
274         if(sPORTB == 1)
275             sw5_counter++;
276         else sw5_counter = 0;
277         if(sw5_counter >= 9) sw5_counter = 9;
278         if(sw5_counter == 8) sw5 = TRUE;
279
280         if(sPORTB == 2)
281             sw6_counter++;
282         else sw6_counter = 0;
283         if(sw6_counter >= 9) sw6_counter = 9;
284         if(sw6_counter == 8) sw6 = TRUE;
285
286         if(sPORTB == 4)
287             sw7_counter++;
288         else sw7_counter = 0;
289         if(sw7_counter >= 9) sw7_counter = 9;
290         if(sw7_counter == 8) sw7 = TRUE;
291
292         if(sPORTB == 8)
293             sw8_counter++;
294         else sw8_counter = 0;
295         if(sw8_counter >= 9) sw8_counter = 9;
296         if(sw8_counter == 8) sw8 = TRUE;
297
298         LATB = 0x0B;
299         NOP();
300         sPORTB = PORTB;
301         sPORTB = ~sPORTB;
302         sPORTB &= 0xF0;
303         sPORTB >>= 4;
304
```



```
305         if(sPORTB == 1)
306             sw9_counter++;
307         else sw9_counter = 0;
308         if(sw9_counter >= 9) sw9_counter = 9;
309         if(sw9_counter == 8) sw9 = TRUE;
310
311         if(sPORTB == 2)
312             sw10_counter++;
313         else sw10_counter = 0;
314         if(sw10_counter >= 9) sw10_counter = 9;
315         if(sw10_counter == 8) sw10 = TRUE;
316
317         if(sPORTB == 4)
318             sw11_counter++;
319         else sw11_counter = 0;
320         if(sw11_counter >= 9) sw11_counter = 9;
321         if(sw11_counter == 8) sw11 = TRUE;
322
323         if(sPORTB == 8)
324             sw12_counter++;
325         else sw12_counter = 0;
326         if(sw12_counter >= 9) sw12_counter = 9;
327         if(sw12_counter == 8) sw12 = TRUE;
328
329         LATB = 0x07;
330         NOP();
331         sPORTB = PORTB;
332         sPORTB = ~sPORTB;
333         sPORTB &= 0xF0;
334         sPORTB >>= 4;
335
336         if(sPORTB == 1)
337             sw13_counter++;
338         else sw13_counter = 0;
339         if(sw13_counter >= 9) sw13_counter = 9;
340         if(sw13_counter == 8) sw13 = TRUE;
341
342         if(sPORTB == 2)
343             sw14_counter++;
344         else sw14_counter = 0;
345         if(sw14_counter >= 9) sw14_counter = 9;
346         if(sw14_counter == 8) sw14 = TRUE;
347
348         if(sPORTB == 4)
349             sw15_counter++;
350         else sw15_counter = 0;
351         if(sw15_counter >= 9) sw15_counter = 9;
352         if(sw15_counter == 8) sw15 = TRUE;
353
354         if(sPORTB == 8)
355             sw16_counter++;
356         else sw16_counter = 0;
357         if(sw16_counter >= 9) sw16_counter = 9;
358         if(sw16_counter == 8) sw16 = TRUE;
359
360         LATB = 0x00;
361         key_flag = FALSE;
362     }
363 }                                       //INTERRUPT ISR ENDS HERE
364
365 /*function definition F8-1-1,-2,-3,-4, toggle key value */
366 unsigned char toggle_key_1(void)                //F8-1-1
367 {
368     static unsigned char state_variable  = 0;
369     unsigned char reval;
370
371     switch(state_variable)
372     {
373         case 0:
374         reval = 0;
375         state_variable = 1;
376         break;
377
378         case 1:
379         reval = 1;
380         state_variable = 0;
```



```
381         break;
382
383         default: break;
384     }
385     return reval;
386 }
387
388 unsigned char toggle_key_2(void)                //F8-1-2
389 {
390     static unsigned char state_variable  = 0;
391     unsigned char reval;
392
393     switch(state_variable)
394     {
395         case 0:
396         reval = 0;
397         state_variable = 1;
398         break;
399
400         case 1:
401         reval = 1;
402         state_variable = 0;
403         break;
404
405         default: break;
406     }
407     return reval;
408 }
409
410 unsigned char toggle_key_3(void)                //F8-1-3
411 {
412     static unsigned char state_variable  = 0;
413     unsigned char reval;
414
415     switch(state_variable)
416     {
417         case 0:
418         reval = 0;
419         state_variable = 1;
420         break;
421
422         case 1:
423         reval = 1;
424         state_variable = 0;
425         break;
426
427         default: break;
428     }
429     return reval;
430 }
431
432 unsigned char toggle_key_4(void)                //F8-1-4
433 {
434     static unsigned char state_variable  = 0;
435     unsigned char reval;
436
437     switch(state_variable)
438     {
439         case 0:
440         reval = 0;
441         state_variable = 1;
442         break;
443
444         case 1:
445         reval = 1;
446         state_variable = 0;
447         break;
448
449         default: break;
450     }
451     return reval;
452 }
453
454 /*function definition F8-2-1,-2, four(4) sequence key values */
455 unsigned char four_push_key_1(void)             //F8-2-1
456 {
```

```
457     static unsigned char state_variable = 0;
458     static unsigned char reval;
459
460     switch(state_variable)
461     {
462         case 0:
463         reval = 0;
464         state_variable = 1;
465         break;
466
467         case 1:
468         reval = 1;
469         state_variable = 2;
470         break;
471
472         case 2:
473         reval = 2;
474         state_variable = 3;
475         break;
476
477         case 3:
478         reval = 3;
479         state_variable = 0;
480         break;
481         default: break;
482     }
483     return reval;
484 }
485
486 unsigned char four_push_key_2(void)                 //F8-2-2
487 {
488     static unsigned char state_variable = 0;
489     static unsigned char reval;
490
491     switch(state_variable)
492     {
493         case 0:
494         reval = 0;
495         state_variable = 1;
496         break;
497
498         case 1:
499         reval = 1;
500         state_variable = 2;
501         break;
502
503         case 2:
504         reval = 2;
505         state_variable = 3;
506         break;
507
508         case 3:
509         reval = 3;
510         state_variable = 0;
511         break;
512         default: break;
513     }
514     return reval;
515 }
516
517
518 /*function F8, select a key */
519 unsigned char ir_key_update(unsigned char num)          //F8
520 {
521     static unsigned char j = 0;
522     unsigned char reval;
523
524     switch(num)
525     {
526         case 1:
527         j = toggle_key_1();
528         if(j == 0)
529             reval = 0;
530
531         if(j == 1)
532             reval = 1;
```



```
533         break;
534
535         case 2:
536         j = toggle_key_2();
537         if(j == 0)
538             reval = 2;
539         if(j == 1)
540             reval = 3;
541         break;
542
543         case 3:
544         j = toggle_key_3();
545         if(j == 0)
546             reval = 4;
547         if(j == 1)
548             reval = 5;
549         break;
550
551         case 4:
552         j = toggle_key_4();
553         if(j == 0)
554             reval = 6;
555         if(j == 1)
556             reval = 7;
557         break;
558
559         case 5:
560         j = four_push_key_1();
561         if(j == 0)
562             reval = 8;
563         if(j == 1)
564             reval = 9;
565         if(j == 2)
566             reval = 10;
567         if(j == 3)
568             reval = 11;
569         break;
570
571         case 6:
572         j = four_push_key_2();
573         if(j == 0)
574             reval = 12;
575         if(j == 1)
576             reval = 13;
577         if(j == 2)
578             reval = 14;
579         if(j == 3)
580             reval = 15;
581         break;
582
583         case 7:
584             reval = 16;
585         break;
586
587         case 8:
588             reval = 17;
589         break;
590
591         case 9:
592             reval = 18;
593         break;
594
595         case 10:
596             reval = 19;
597         break;
598
599         case 11:
600             reval = 20;
601         break;
602
603         case 12:
604             reval = 21;
605         break;
606
607         case 13:
608             reval = 22;
```



```
609         break;
610
611         case 14:
612             reval = 23;
613         break;
614
615         case 15:
616         reval = 24;
617         break;
618
619         case 16:
620             reval = 25;
621         break;
622         default: reval = ERROR; break;
623     }
624     return reval;
625 }
626 /*function definition F16, find push button switch number  */
627 unsigned char find_key(void)                    //F16
628 {
629     unsigned char reval;
630     if(sw1 == TRUE){
631         sw1 = FALSE;
632         reval = 1;
633     }
634     else if(sw2 == TRUE){
635         sw2 = FALSE;
636         reval = 2;
637     }
638     else if(sw3 == TRUE){
639         sw3 = FALSE;
640         reval = 3;
641     }
642     else if(sw4 == TRUE){
643         sw4 = FALSE;
644         reval = 4;
645     }
646     else if(sw5 == TRUE){
647         sw5 = FALSE;
648         reval = 5;
649     }
650     else if(sw6 == TRUE){
651         sw6 = FALSE;
652         reval = 6;
653     }
654     else if(sw7 == TRUE){
655         sw7 = FALSE;
656         reval = 7;
657     }
658     else if(sw8 == TRUE){
659         sw8 = FALSE;
660         reval = 8;
661     }
662     else if(sw9 == TRUE){
663         sw9 = FALSE;
664         reval = 9;
665     }
666     else if(sw10 == TRUE){
667         sw10 = FALSE;
668         reval = 10;
669     }
670     else if(sw11 == TRUE){
671         sw11 = FALSE;
672         reval = 11;
673     }
674     else if(sw12 == TRUE){
675         sw12 = FALSE;
676         reval = 12;
677     }
678     else if(sw13 == TRUE){
679         sw13 = FALSE;
680         reval = 13;
681     }
682     else if(sw14 == TRUE){
683         sw14 = FALSE;
684         reval = 14;
```



```
685     }
686     else if(sw15 == TRUE){
687         sw15 = FALSE;
688         reval = 15;
689     }
690     else if(sw16 == TRUE){
691         sw16 = FALSE;
692         reval = 16;
693     }
694     else reval = ERROR;
695     return reval;
696 }
697
```

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	RB1	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