

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

- 消費電力が極めて小さい。約28 μ A@3V(スリープ時)
スイッチが押されたとき約1-2mA@3V
- 小型でローコスト
- 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。
- 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。
- 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。
- 通信可能距離は直線約50m(注1)
- 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

- 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
- 無線リモートコントローラ(ラジコン)受信部(品番RF-0299-RX)
- 単三電池2個 2P電池ケース1個
- 3ピンのピンソケット(メス)1個
- キーパッド(品番ACC-0295-KEY4x4)1個
- 10cm程の線材(アンテナとして使用する)。

組み立て

- 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。
- 2P電池ケースのプラス側を本体(CN4)3V、マイナス側をグラウンド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。
- 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。

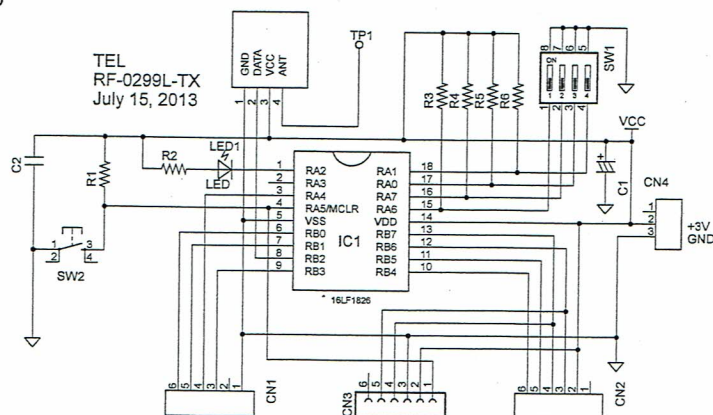
通信形式について

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ーターのクロック信号に同期して自動的に送信される。
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み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ため注意すること。

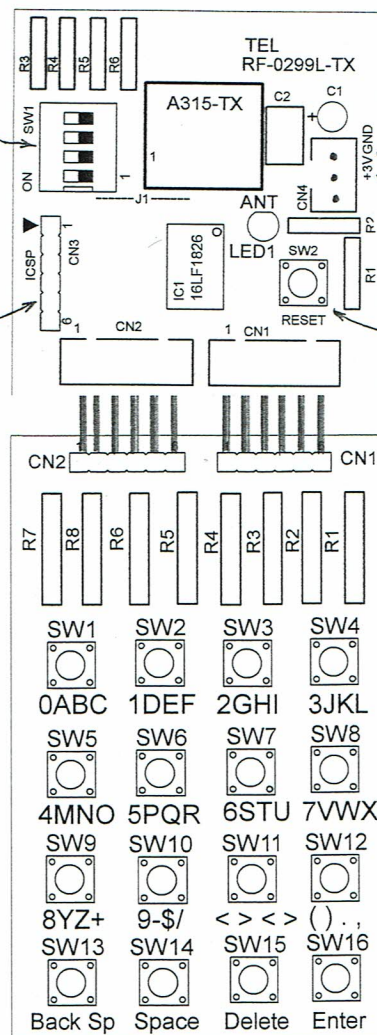


無線リモートコントローラ送信部 (品番RF-0299L-TX)回路図

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
Serial Programming)
用コネクター。
使用しないこと。



電源グラウンド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。

キーパッド(別売)
(品番ACC-0295-KEY4X4)

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)

```
1 /***** RF Wireless Remote Controller Transmitter *****/
2 * Project name: RF0299TX
3 * File name: RF0299TX.c
4 * Features:
5 * *nano watt extreme low power(XLP), 1.8uWatt typical,
6 * *Micro controller PIC enhanced mid-range 16F1826.
7 * *3 bits device code
8 * *Wide operating voltage, 2.0V-3.3V.
9 * Pin assignments:
10 * *Device code input <RA6:RA7>, <RA1:RA0>
11 * *Data Output <RB2>
12 * *the Keypad Input <RB7:RB4>
13 * *the Keypad Output <RB3,RA4> <RB1:RB0>
14 * *LED Output <RA2>
15 * Compiler: xc8 v1.12
16 * AUG 10, 2013. Coded by yuji Tanioka (TEL company)
17
18 #include <xc.h>
19
20 #pragma config MCLRE = ON, CP = OFF, BOREN = OFF, WDT = OFF,
21 #pragma config FWRT = ON, CPD = OFF, CLKOUTEN = OFF, FCMEN = OFF
22 #pragma config WRT = OFF, STVREN = OFF, BORV = LO, LVP = OFF,
23 #pragma config FOSC = INTOSC, FLEN = OFF
24
25 #define _XTAL_FREQ 8000000 //8MHz
26
27 typedef volatile bit Bool;
28
29 #define TRUE 1
30 #define FALSE 0
31 #define ERROR 0xFF
32 #define REPEAT_TIME 2000 //1ms
33
34 //function prototype declaration
35 void reg_init(void); //F13
36 void rf_send(unsigned char num); //F7
37 void interrupt isr(void); //F17
38 unsigned char toggle_key_1(void); //F8-1-1
39 unsigned char toggle_key_2(void); //F8-1-2
40 unsigned char toggle_key_3(void); //F8-1-3
41 unsigned char toggle_key_4(void); //F8-1-4
42 unsigned char four_push_key_1(void); //F8-2-1
43 unsigned char four_push_key_2(void); //F8-2-2
44 unsigned char find_key(void); //F16
45 unsigned char rf_key_update(unsigned char num); //F8
46 void putc(unsigned char ch); //F10
47 void led_on(void); //F18
48
49 Bool sw1 = FALSE, sw2 = FALSE, sw3 = FALSE, sw4 = FALSE,
50 sw5 = FALSE, sw6 = FALSE, sw7 = FALSE, sw8 = FALSE,
51 sw9 = FALSE, sw10 = FALSE, sw11 = FALSE, sw12 = FALSE,
52 sw13 = FALSE, sw14 = FALSE, sw15 = FALSE, sw16 = FALSE;
53
54 Bool key_flag = FALSE;
55
56 /*Main function*/
57 void main(void)
58 {
59     unsigned char n, key_num, sIOCBF;
60
61     reg_init(); //F13
62
63     for(;;)
64     {
65         key_num = find_key(); //F16
66         if(key_num != ERROR)
67         {
68             Page 1 OF 11 2013.08.10
69
70             led_on(); //LED ON
71             n = rf_key_update(key_num);
72             rf_send(n); //F7
73
74             LATB0 = 0; LATB1 = 0; LATA4 = 0; LATB3 = 0;
75             if(key_flag == FALSE)
76             {
77                 IOCBF = 1;
78                 SLEEP(); //before sleep
79                 sIOCBF = IOCBF; //CLEAR ALL FLAG
80                 sIOCBF ^= 0xFF;
81                 IOCBF ^= sIOCBF;
82             }
83         }
84     }
85
86 /*function definition F13, register initialization*/
87 void reg_init(void)
88 {
89     OSCCON = 0b01110000; //8MHz, HF:IRCF=01110
90     OSCUNE = 0x00;
91     TRISA = 0b11100011; //RA2:RA4 OUTPUT
92     TRISB = 0b11100000; //RB7:RB4 INPUT
93     ANSELA = 0x00; ANSELB = 0x00; //DIGITAL I/O
94     WPUEN = 0; //WEAK PULL-UP RAS DISABLED
95     WPUEN = 0x00; //WEAK PULL-UP
96     PORTB = 0;
97     OPTION_REG = 0b00000011; //0x03
98     //0-----
99     //0-----
100    //0-----
101    //0-----
102    //0-----
103    //0-----
104    TMR0 = 0;
105    T2CON = 0b00000011; //0x0B
106    //0011---
107    //0011---
108    //0011---
109    TMR2 = 0x00; //TIMER 2 CLEAR
110    PR2 = 0xFF; //TOTAL LED ON TIME:
111    //0.5uSx64x256x4=32.768ms
112    TMR2IE = 1; //TIMER 2 ENABLED
113
114    LATA = 0x04; //LATA = 0x00;
115    PORTA = 0x04; //PORTB = 0x00;
116    IOCBF = 1; //IOCBF = 0x00;
117    IOCBF = 0x00; //NEGATIVE GOING EDGE IOCB ENABLE
118    IOCBF = 0x00; //CLEAR IOCB FLAG
119    TXSTA = 0b00000000; //0x24
120    //0-----
121    //0-----
122    //0-----
123    //0-----
124
125    BAUDCON = 0b00000000; //0x08
126    //0-----
127    //0-----
128
129    //SPBRGL = 0x40; //BAUD RATE=2400, BRGH=1, BRG16=1
130    //SPBRGH = 0x03; //832(0x0340), ERROR 0.04%
131    //SPBRGL = 0x0A; //BAUD RATE=300, BRGH=1, BRG16=1
132    //SPBRGH = 0x1A; //6666(0x1A0A), ERROR 0%
133    SPBRGL = 103; //BAUD RATE=1200, BRGH=0, BRG16=0
134    SPBRGH = 0; //103 ERROR 0.16%
135
136    RCSTA = 0b10000000; //0x80
137    Page 2 OF 11 2013.08.10
```

```
137 //1----- //Serial Port enable(SPEN)
138 //0----- //use ninth bit(RX9)
139 //0----- //CREN=0
140 //0----- //ADDEN=0
141
142 TXIE = 0; //TX1 INTERRUPT DISABLED
143
144 PEIE = 1;
145 ei();
146
147 void rf_send(unsigned char num) //F7
148 {
149     unsigned char pin_code;
150
151     if(num != ERROR)
152     {
153         pin_code = PORTA;
154         pin_code ^= 0xC3; //CLEAR <RA5:RA2>
155         pin_code = pin_code << 2 | pin_code >> 6; //rotate 2 bits
156
157         IOCBF = 0;
158         TXEN = 1;
159
160         putc('T');
161         putc('E');
162         putc('L');
163         putc(pin_code);
164         putc(num);
165         putc(-pin_code);
166         putc(-num);
167         putc('E');
168
169         _delay(REPEAT_TIME); //1ms
170
171         putc('T');
172         putc('E');
173         putc('L');
174         putc(pin_code);
175         putc(num);
176         putc(-pin_code);
177         putc(-num);
178         putc('E');
179
180         _delay(REPEAT_TIME); //1ms
181
182         putc('T');
183         putc('E');
184         putc('L');
185         putc(pin_code);
186         putc(num);
187         putc(-pin_code);
188         putc(-num);
189         putc('E');
190
191         TXEN = 0;
192         IOCBF = 1;
193     }
194
195 void putc(unsigned char ch) //F10
196 {
197     while(!TXIF)
198     {
199         continue;
200     }
201     TXREG = ch;
202 }
203
204
205 void interrupt isr(void) //F17
206 {
207     if(IOCBF && IOCBF) //INTERRUPT ON CHANGE ON PORT B
208     {
209         IOCBF = 0;
210         IOCBF = 0;
211         key_flag = TRUE;
212         TMR0 = 0;
213         TMR0IE = 1;
214
215         if(TMR2IE && TMR2IF)
216         {
217             TMR2IF = 0;
218             TMR2CON = 0;
219             LATA2 = 1; //LED OFF
220         }
221
222         if(TMR0IE && TMR0IF) //TIMER0 INTERRUPT
223         {
224             unsigned char sPORTB; //shadow register
225
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             LATB0 = 0; LATB1 = 1; LATA4 = 1; LATB3 = 1;
237             NOP();
238             sPORTB = PORTB;
239             sPORTB = ~sPORTB;
240             sPORTB ^= 0xFF;
241             sPORTB >= 4;
242
243             if(sPORTB == 1) //2.048uSx8=16.38ms
244             {
245                 sw1_counter++; //total deunce time
246                 if(sw1_counter == 0) sw1_counter = 9;
247                 if(sw1_counter == 8) sw1 = TRUE; //PRODUCE A SINGLE PULSE
248                 //PER PUSH
249             }
250
251             if(sPORTB == 2)
252             {
253                 sw2_counter++;
254                 if(sw2_counter == 0) sw2_counter = 9;
255                 if(sw2_counter == 8) sw2 = TRUE;
256             }
257
258             if(sPORTB == 4)
259             {
260                 sw3_counter++;
261                 if(sw3_counter == 0) sw3_counter = 9;
262                 if(sw3_counter == 8) sw3 = TRUE;
263             }
264
265             if(sPORTB == 8)
266             {
267                 sw4_counter++;
268                 if(sw4_counter == 0) sw4_counter = 9;
269                 if(sw4_counter == 8) sw4 = TRUE;
270             }
271
272             LATB0 = 1; LATB1 = 0; LATA4 = 1; LATB3 = 1;
273             NOP();
274             sPORTB = PORTB;
275
276             Page 4 OF 11 2013.08.10
```

```

273     SPORTS = ~SPORTS;
274     SPORTS &= 0xF0;
275     SPORTS >>= 4;
276
277     if(sPORTS == 1)
278         sw5_counter++;
279     else sw5_counter = 0;
280     if(sw5_counter >= 9) sw5_counter = 9;
281     if(sw5_counter == 8) sw5 = TRUE;
282
283     if(sPORTS == 2)
284         sw6_counter++;
285     else sw6_counter = 0;
286     if(sw6_counter >= 9) sw6_counter = 9;
287     if(sw6_counter == 8) sw6 = TRUE;
288
289     if(sPORTS == 4)
290         sw7_counter++;
291     else sw7_counter = 0;
292     if(sw7_counter >= 9) sw7_counter = 9;
293     if(sw7_counter == 8) sw7 = TRUE;
294
295     if(sPORTS == 8)
296         sw8_counter++;
297     else sw8_counter = 0;
298     if(sw8_counter >= 9) sw8_counter = 9;
299     if(sw8_counter == 8) sw8 = TRUE;
300
301     LATB0 = 1; LATB1 = 1; LATA4 = 0; LATB3 = 1;
302
303     NOP();
304     SPORTS = PORTB;
305     SPORTS = ~SPORTS;
306     SPORTS &= 0xF0;
307     SPORTS >>= 4;
308
309     if(sPORTS == 1)
310         sw9_counter++;
311     else sw9_counter = 0;
312     if(sw9_counter >= 9) sw9_counter = 9;
313     if(sw9_counter == 8) sw9 = TRUE;
314
315     if(sPORTS == 2)
316         sw10_counter++;
317     else sw10_counter = 0;
318     if(sw10_counter >= 9) sw10_counter = 9;
319     if(sw10_counter == 8) sw10 = TRUE;
320
321     if(sPORTS == 4)
322         sw11_counter++;
323     else sw11_counter = 0;
324     if(sw11_counter >= 9) sw11_counter = 9;
325     if(sw11_counter == 8) sw11 = TRUE;
326
327     if(sPORTS == 8)
328         sw12_counter++;
329     else sw12_counter = 0;
330     if(sw12_counter >= 9) sw12_counter = 9;
331     if(sw12_counter == 8) sw12 = TRUE;
332
333     LATB0 = 1; LATB1 = 1; LATA4 = 1; LATB3 = 0;
334
335     NOP();
336     SPORTS = PORTB;
337     SPORTS = ~SPORTS;
338     SPORTS &= 0xF0;
339     SPORTS >>= 4;
340

```

Page 5 OF 11

2013.08.10

```

409         default: break;
410     }
411     return reval;
412 }
413
414
415 unsigned char toggle_key_3(void) //F8-1-3
416 {
417     static unsigned char state_variable = 0;
418     unsigned char reval;
419
420     switch(state_variable)
421     {
422         case 0:
423             reval = 0;
424             state_variable = 1;
425             break;
426
427         case 1:
428             reval = 1;
429             state_variable = 0;
430             break;
431
432         default: break;
433     }
434     return reval;
435 }
436
437 unsigned char toggle_key_4(void) //F8-1-4
438 {
439     static unsigned char state_variable = 0;
440     unsigned char reval;
441
442     switch(state_variable)
443     {
444         case 0:
445             reval = 0;
446             state_variable = 1;
447             break;
448
449         case 1:
450             reval = 1;
451             state_variable = 0;
452             break;
453
454         default: break;
455     }
456     return reval;
457 }
458
459 /*function definition F8-2-1,-2, four(4) sequence key values */
460 unsigned char four_push_key_1(void) //F8-2-1
461 {
462     static unsigned char state_variable = 0;
463     static unsigned char reval;
464
465     switch(state_variable)
466     {
467         case 0:
468             reval = 0;
469             state_variable = 1;
470             break;
471
472         case 1:
473             reval = 1;
474             state_variable = 2;
475             break;
476

```

Page 7 OF 11

2013.08.10

```

341     if(sPORTS == 1)
342         sw13_counter++;
343     else sw13_counter = 0;
344     if(sw13_counter >= 9) sw13_counter = 9;
345     if(sw13_counter == 8) sw13 = TRUE;
346
347     if(sPORTS == 2)
348         sw14_counter++;
349     else sw14_counter = 0;
350     if(sw14_counter >= 9) sw14_counter = 9;
351     if(sw14_counter == 8) sw14 = TRUE;
352
353     if(sPORTS == 4)
354         sw15_counter++;
355     else sw15_counter = 0;
356     if(sw15_counter >= 9) sw15_counter = 9;
357     if(sw15_counter == 8) sw15 = TRUE;
358
359     if(sPORTS == 8)
360         sw16_counter++;
361     else sw16_counter = 0;
362     if(sw16_counter >= 9) sw16_counter = 9;
363     if(sw16_counter == 8) sw16 = TRUE;
364
365     LATB0 = 0; LATB1 = 0; LATA4 = 0; LATB3 = 0;
366     key_flag = FALSE;
367 }
368 //INTERRUPT ISR ENDS HERE
369
370 /*function definition F8-1-1,-2,-3,-4, toggle key value */
371 unsigned char toggle_key_1(void) //F8-1-1
372 {
373     static unsigned char state_variable = 0;
374     unsigned char reval;
375
376     switch(state_variable)
377     {
378         case 0:
379             reval = 0;
380             state_variable = 1;
381             break;
382
383         case 1:
384             reval = 1;
385             state_variable = 0;
386             break;
387
388         default: break;
389     }
390     return reval;
391 }
392
393 unsigned char toggle_key_2(void) //F8-1-2
394 {
395     static unsigned char state_variable = 0;
396     unsigned char reval;
397
398     switch(state_variable)
399     {
400         case 0:
401             reval = 0;
402             state_variable = 1;
403             break;
404
405         case 1:
406             reval = 1;
407             state_variable = 0;
408             break;

```

Page 6 OF 11

2013.08.10

```

477     case 2:
478         reval = 2;
479         state_variable = 3;
480         break;
481
482     case 3:
483         reval = 3;
484         state_variable = 0;
485         break;
486     default: break;
487 }
488 return reval;
489 }
490
491 unsigned char four_push_key_2(void) //F8-2-2
492 {
493     static unsigned char state_variable = 0;
494     static unsigned char reval;
495
496     switch(state_variable)
497     {
498         case 0:
499             reval = 0;
500             state_variable = 1;
501             break;
502
503         case 1:
504             reval = 1;
505             state_variable = 2;
506             break;
507
508         case 2:
509             reval = 2;
510             state_variable = 3;
511             break;
512
513         case 3:
514             reval = 3;
515             state_variable = 0;
516             break;
517         default: break;
518     }
519     return reval;
520 }
521
522 /*function F8, select a key */
523 unsigned char rf_key_update(unsigned char num) //F8
524 {
525     unsigned char j = 0;
526     unsigned char reval;
527
528     switch(num)
529     {
530         case 1:
531             j = toggle_key_1();
532             if(j == 0)
533                 reval = 0;
534
535             if(j == 1)
536                 reval = 1;
537             break;
538
539         case 2:
540             j = toggle_key_2();
541             if(j == 0)
542                 reval = 2;
543             if(j == 1)

```

Page 8 OF 11

2013.08.10

```

545     reval = 3;
546     break;
547
548     case 3:
549     j = toggle_key_3();
550     if(j == 0)
551         reval = 4;
552     if(j == 1)
553         reval = 5;
554     break;
555
556     case 4:
557     j = toggle_key_4();
558     if(j == 0)
559         reval = 6;
560     if(j == 1)
561         reval = 7;
562     break;
563
564     case 5:
565     j = four_push_key_1();
566     if(j == 0)
567         reval = 8;
568     if(j == 1)
569         reval = 9;
570     if(j == 2)
571         reval = 10;
572     if(j == 3)
573         reval = 11;
574     break;
575
576     case 6:
577     j = four_push_key_2();
578     if(j == 0)
579         reval = 12;
580     if(j == 1)
581         reval = 13;
582     if(j == 2)
583         reval = 14;
584     if(j == 3)
585         reval = 15;
586     break;
587
588     case 7:
589     reval = 16;
590     break;
591
592     case 8:
593     reval = 17;
594     break;
595
596     case 9:
597     reval = 18;
598     break;
599
600     case 10:
601     reval = 19;
602     break;
603
604     case 11:
605     reval = 20;
606     break;
607
608     case 12:
609     reval = 21;
610     break;
611
612     case 13:

```

Page 9 OF 11

2013.08.10

```

681     reval = 12;
682 }
683 else if(sw13 == TRUE){
684     sw13 = FALSE;
685     reval = 13;
686 }
687 else if(sw14 == TRUE){
688     sw14 = FALSE;
689     reval = 14;
690 }
691 else if(sw15 == TRUE){
692     sw15 = FALSE;
693     reval = 15;
694 }
695 else if(sw16 == TRUE){
696     sw16 = FALSE;
697     reval = 16;
698 }
699 else reval = ERROR;
700 return reval;
701 }
702
703 /*function definition F18, led on time*/
704 void led_on(void) //F18
705 {
706     PR2 = 0xFF;
707     TMR2IE = 1;
708     TMR2ON = 1;
709     LATA2 = 0; //LED ON
710 }

```

Page 11 OF 11

2013.08.10

```

613     reval = 22;
614     break;
615
616     case 14:
617     reval = 23;
618     break;
619
620     case 15:
621     reval = 24;
622     break;
623
624     case 16:
625     reval = 25;
626     break;
627     default: reval = ERROR; break;
628 }
629 return reval;
630 }
631 /*function definition F16, find push button switch number */
632 unsigned char find_key(void) //F16
633 {
634     unsigned char reval;
635     if(sw1 == TRUE){
636         sw1 = FALSE;
637         reval = 1;
638     }
639     else if(sw2 == TRUE){
640         sw2 = FALSE;
641         reval = 2;
642     }
643     else if(sw3 == TRUE){
644         sw3 = FALSE;
645         reval = 3;
646     }
647     else if(sw4 == TRUE){
648         sw4 = FALSE;
649         reval = 4;
650     }
651     else if(sw5 == TRUE){
652         sw5 = FALSE;
653         reval = 5;
654     }
655     else if(sw6 == TRUE){
656         sw6 = FALSE;
657         reval = 6;
658     }
659     else if(sw7 == TRUE){
660         sw7 = FALSE;
661         reval = 7;
662     }
663     else if(sw8 == TRUE){
664         sw8 = FALSE;
665         reval = 8;
666     }
667     else if(sw9 == TRUE){
668         sw9 = FALSE;
669         reval = 9;
670     }
671     else if(sw10 == TRUE){
672         sw10 = FALSE;
673         reval = 10;
674     }
675     else if(sw11 == TRUE){
676         sw11 = FALSE;
677         reval = 11;
678     }
679     else if(sw12 == TRUE){
680         sw12 = FALSE;

```

Page 10 OF 11

2013.08.10

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

コネクタ番号 ポート	CN2-3 RB6	CN2-4 RB7	CN2-5 RB5	CN2-6 RB4	CN1-3 RB3	CN1-4 RB2	CN1-5 RA4	CN1-6 RB0
3ビットデバイスコード(負論理) デフォルト値='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