136kHz 帯 WSPR ビーコンの開発　　2011.6.14　JA5FP

——— PICWSPR のハードウェアとソースコード ———

1. 本機の用途と構成

137.5kHzWSPR ビーコンを長時間にわたって運用するための専用送信機の信号発生部が PICWSPR です。ハードウェアは PIC16F88 と GPS 受信機から成ります。

PICWSPR の機能ブロックを図 1 に示します。

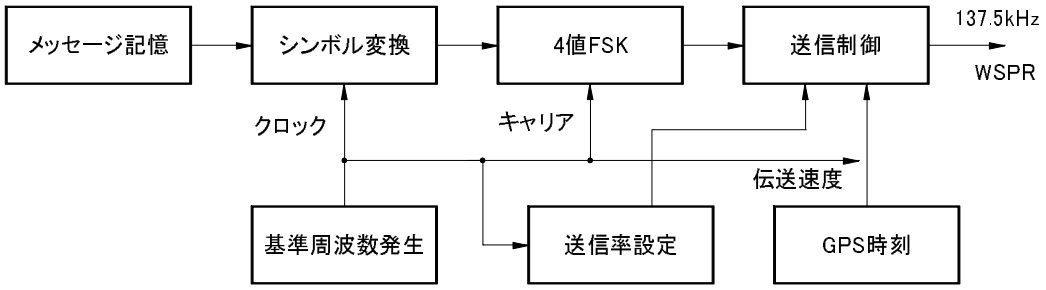

図 1: PC を使わない WSPR 発生器

2. 各端子の機能

PICWSPR の回路は図 4 のとおりです。PIC16F88 各端子を参照しながら、本機の機能を説明します。

- ・OSC1(pin 3)　内蔵発振器の入力端子です。11.000MHz 水晶発振子の負荷キャパシティによる FSK を行います。
- ・OSC2(pin 4)　内蔵発振器の出力端子です。
- ・RA0(pin 2)　FSK の最低周波数 (f_0) を制御します。
- ・RA1(pin 1)　FSK の低周波数 (f_1) を制御します。
- ・RA3(pin 17)　FSK の高周波数 (f_2) を制御します。最高周波数 (f_3) は固定です。
- ・RB6(pin 7)　モード切替を行います。H レベル (スイッチ開放) で通常運転、L レベル (スイッチ短絡) で FSK の周波数偏移調整用動作に入ります。調整モードでは RA0 および RA1 が共に H レベルとなりますが、個別に L レベルにするには TR のベース・エミッタ間を短絡します。
- ・RB7(pin 6)　送信レートを選択します。H レベル (スイッチ開放) で 20%、L レベル (スイッチ短絡) で 10%となります。
- ・RX(pin 11)　内蔵 ASUART の受信端子です。GPS ユニットのデータを取り込みます。
- ・TX(pin 8)　使用しません。
- ・RB3(pin 10)　内蔵 ASUART を GPS ユニットの通信速度に対応させます。H レベル (スイッチ開放) で 4,800bps、L レベル (スイッチ短絡) で 9,600bps となります。
- ・RB1(pin 12)　GPS の 1PPS 信号を取り込みます。
- ・RB4(pin 9)　PIC16F88 のプログラミング時に EEPROM に書き込んだメッセージ 2 種類から使用するメッセージを選択します。H レベル (スイッチ開放)

で、例えば”JG1JZL QM05 37” が使われて L レベル (スイッチ短絡) では ”JG1JZL QM05 50” が使われます。

・RA2(pin 18)　PTT 制御信号を出力します。
・CCP1(pin 13)　137.5kHz の WSPR 信号を出力します。
・RA4(pin 16)　シーケンス確認用の LED 駆動信号出力です。

なお、EIA-232E インターフェースは将来の拡張のためのオプションです。

3. PICWSPR の用法と動作

PIC16F88 のプログラミング・ソースコードは、本稿後段に記載します。その利用は自由ですが、著作権は筆者に所属します。

開発環境は MPLAB IDE ®v8.10 と HI-TECH C®コンパイラ、プログラミングは秋月電子の picpgm6 Ver. 6.72.17 で行いました。

操作手順と PIC 内部の動作シーケンスは次のとおりです。

(a) EEPROM へ書き込むデータの準備

送信者に固有のメッセージの内容は WSPR 規格で **Callsign GridLocater Power(dBm)** です。筆者の場合は例えば”JG1JZL QM05 37” となります。これを WSPR コードに変換するための DOS プログラムが K1JT から提供されていますので、これを使います。

(http://www.physics.princeton.edu/pulsar/K1JT/WSPRcode.exe)

WSPRcode.exe が吐き出す数字をメモしておきます。その内容は Data symbols に続いて Synchro symbols が現れる合計 162 個のバイナリーです。具体的には、例えば表 1 に示すように Channel symbols で言うと”3 1 2 2 ....” の 4 進符号です。その内容は、Data symbol を MSB、Synchro symbol を LSB とするものとなっています。このデータストリームは、最終的に WSPR 信号の 4 値 FSK の変調信号に相当します。

表 1: WSPR コードと EEPROM データの相対関係

操　作	形　態	最初のバイト				～	最後のバイト			
WSPRcode.exe	Channel symbols	3	1	2	2	略	0	0	-	-
	Data symbols	1	0	1	1	略	0	0	-	-
	Synchro symbols	1	1	0	0	略	0	0	-	-
手作業で換算	16 進数	D		A		略	0		F	
EEPROM 書込	データ	0xda				略	0x0f			
	アドレス	0x00				略	0x28			

これを PIC の EEPROM 領域に記憶するためには、PIC がバイト単位でデータのやりとりをする規格なので、若干の操作が必要です。まづ WSPRcode.exe で得られた 4 進符号を 8 ビット符号に変換します。この操作は単純に、手動で行うことにします (Appendix 1 を参照)。具体的に表 1 の例”1 1 0 1 1 0 1 0” は即ち”0xda” に他なりません。他のデータも同様に変換します。この手作業が間違いやすいので、G4JNT 提供の専用ソフト GENWSPR.EXE を使うのが良いでしょう。(http://www.g4jnt.com/JTModesBcns.htm)

1 メッセージに付き Channel データは 162 個ですから、EEPROM へのデータは 40.5 バイトとなる訳です。最後の 0.5 バイトは適当な値 (例えば 0xf) を挿入し全部で 41 バイトとしておきます。

(b) EEPROM へのデータ書き込みと書き換え

EEPROM 領域へのデータはソースコード内に記述する形でプログラミングしています。つまり、メッセージ内容がプログラミング時に固定されていることになります。ハードウェアでは PIC の RB4(pin 9) の H/L で 2 種類のメッセージから選択することができます。

当然ながら、このままでは特定の使用者の固有のメッセージしか使えませんが、EEPROM データを書き換えて対応する方法があります。一つは、最初からソースコードを書き直し、コンパイルおよびプログラミングをやり直すことです。もう一つは、picpgm6 上で EEPROM のデータを直接修正する方法です。さらに、ソースコードを拡充してハイパーターミナル等の通信ソフトで EEPROM 領域に対して書き込む方法があります。これについては本機では実装していませんので、別途の拡張が必要です。

(c) ランダム送信の意味

WSPR システムでは**偶数分の 02 秒から約 110.6 秒間の送信　送信率の設定　ランダム送信**が仕様です。従って、1 回の送信が約 2 分間弱ですから、送信率 100%とは 60 分当たり 30 回の送信ということになります。本機では送信率 20%または 10%をソースコード上で設定していますので、60 分に 6 回または 3 回の送信を行います。

この 6 回または 3 回を時間配置するのですが、60 分内ではあまり高度なランダム性はとれません。そもそもランダム送信をする必要性は、(1) 複数局の同時送信による第三者の受信者への混信防止の予防 (2) 同時送信する局を自局で受信不能となるまたはその逆の予防、という利点があるからでしょう。この内 (1) については、送信周波数が仮に 6Hz 以上離れておれば WSPR が十分デコードしますので、特別に不可欠の利点ではありません。(2) の理由は、WSPR を送受共に行う局では大事な要件です。

そこで、本機の使用形態である送信専用という条件では、ランダム性は絶対的に必要とは思えません。ソースコード Ver 1.6 では定期的な送信間隔をとっています。PICWSPR 仕様の複数局がオンエアしたとしても、それぞれのスイッチオンの時刻にはどの偶数分から送信開始するかでランダム性がありますから、同時送信となる確率は低いです。

(d) GPS による偶数分からの自動起動

内蔵 GPS 受信機からのデータは図 2 のような内容が毎秒得られます。

```
$GPGGA,002154,3540.07940,N,14010.31870,E,1,5,01,+0020,M,+039,M,00,0000*53
$GPGSA,A,3,05,26,02,10,15,00,00,00,00,00,00,00,03.0,01.6,02.4*06
$GPGSV,2,1,08,05,66,007,42,26,65,248,37,02,56,161,39,08,41,079,00*75
$GPGSV,2,2,08,10,34,080,37,15,28,240,44,07,24,049,00,04,18,146,00*71
$GPRMC,002154,A,3540.07940,N,14010.31870,E,000.0,169.,210411,,*27
```

図 2: GPS データの一例

PICWSPR ではこの内から時刻情報が得られる GGA センテンスまたは RMC センテンスに着目して、偶数分を取得します。この例をとると、「002154」が 00 時 21 分 54 秒、すなわち奇数分なので次の偶数分まで待機することになります。また、00 秒を待って送信準備をします。

これにより PC によってインターネット時刻を取得することなく、スタンドアローンで自動送信が行われます。

(e) GPS による標準時刻との同期

PICWSPR では、送信基準時刻を GPS の 1PPS 出力から取得します。

前項の偶数分 00 秒で予告された直後に来る正確な UTC 秒を送信開始のトリガーとします。これで WSPR システムにとって十分な DT 精度を常に確保します。

(f) クロック精度の考察

図 1 でご覧のとおり (1)137.5kHz キャリア発生用 (2)162 個のシンボル発生用 (3)GPS との通信用、という 3 つのクロックが必要です。PICWSPR ではこれらの周波数源を単一の 11.000MHz 発振器から周波数分周して供給します。

各クロックの分周比と周波数精度を表 2 にまとめました。

基準周波数を 11.000MHz に選びました。この他にも 5.5MHz、6.6MHz、7.7MHz、8.8MHz、9.9MHz などの適宜の周波数で構いませんが、AUSART のクロックに大きな周波数偏差が生じないように勘案しなければなりません。逆に AUSART に適当な水晶周波数を優先して選ぶと、137.5kHz に対しては分周比が整数にならないことがありますので、注意をします。

表 2: 単一基準周波数とクロックの精度の関係

基準周波数	クロックの用途	目的周波数	実際の周波数	
			分周数	偏差
11.000MHz	キャリア	137.5kHz	80	±0%
	シンボル長	1.464862Hz	7,509,333	ほぼ 0%
	AUSART	4,800Hz 9,600Hz	2,304 1,152	-0.53%

シンボル長を規定するクロックは極めて大きな分周数ですから、分周数を細かく設定すれば十分高い精度が得られます。ただし、シンボルが 162 個も続くので、最終シンボルではその送出タイミングが単位シンボルの偏差の 162 倍に達します。本来このクロックは 12kHz の倍数の周波数源から分周するのが良いのですが、PICWSPR では周波数源を単一にするために特に工夫しています。

4. 謝辞

PICWSPR の構想に積極的な賛同と適切な助言をいただいた 7L1RLL 若鳥さん JH1ARY 黒田さんおよび JR6RMZ 古堅さんの各氏に感謝申し上げます。

改訂履歴　2011.04.02
改訂履歴　2011.04.08
改訂履歴　2011.04.12
改訂履歴　2011.04.20
改訂履歴　2011.06.14

Appendix 1　4 進から 16 進への変換

表 3 の関係に従って、2 個の 4 進数を 1 個の 16 進数に変換します。

表 3: 4 進数と 16 進数の対比表

4 進	16 進	4 進	16 進
0 0	0	2 0	8
0 1	1	2 1	9
0 2	2	2 2	a
0 3	3	2 3	b
1 0	4	3 0	c
1 1	5	3 1	d
1 2	6	3 2	e
1 3	7	3 3	f

Appendix 2　PICWSPR の総合テスト結果

実際に PICWSPR でドライブした LF 帯電波を受信した様子を図 3 に示します。

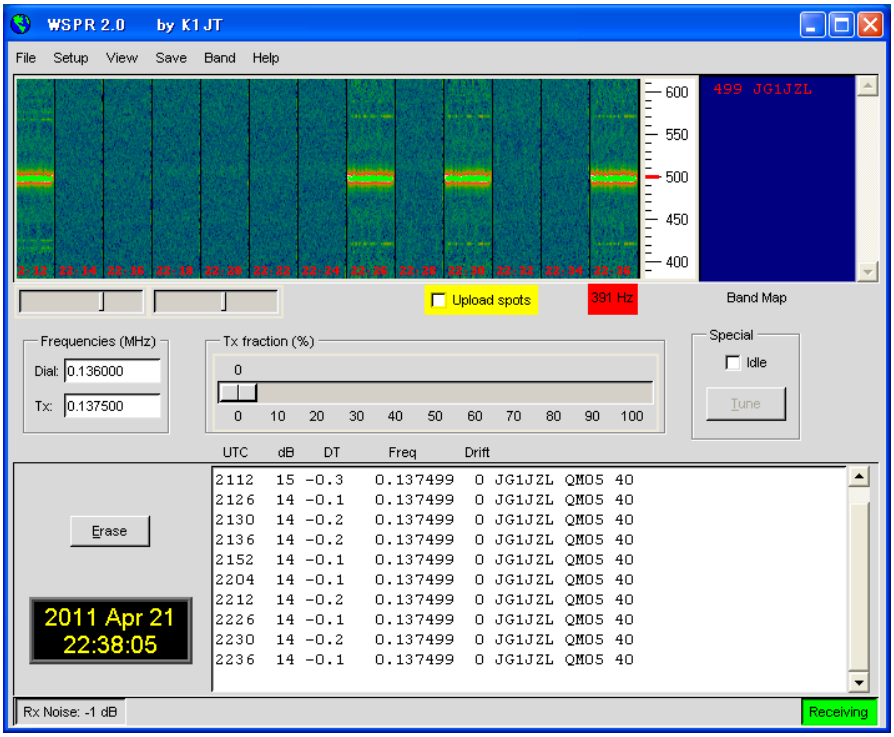

図 3: WSPR 画面の表示

PIC16F88 ソースコード　　©2011　JA5FP

```
/* picwspr.c for PICWSPR ver.1.6  (c) 2011.06.14 JA5FP
   This program may be compiled with HI-TECH C(r) Compiler
for PIC10/12/16 MCUs v.9.8.1 on MPLAB(r) IDE v8.63.
   Target hardware is the PICWSPR which produces continuous
WSPR signal on 137.5kHz as a beaconing signal.
   The functions are:
     (1)selectable one of pre-written messages (callsign,
        gridlocater, power).
     (2)selectable GPS port to 9,600bps or 4,800bps.
     (3)getable correct each even minutes start-up time from GPS.
     (4)selectable 20% or 10% taransmitting rate.
//     (5)randomized sequence.
*/

#include <htc.h>
#include <pic16f88.h>
//#define    ONE     0                                  // developer's
//#define    EVEN    1                                  // developer's
//#define    RATE    2                                  // developer's

__CONFIG(CP_OFF,CCPMX_RB0,DEBUG_OFF,FOSC0_HS,WDTE_OFF,
MCLRE_ON,BOREN_OFF,LVP_OFF,CPD_OFF,WRT_OFF);

__EEPROM_DATA(0xda,0x8a,0x4a,0x76,0x2c,0x99,0xdc,0x00); // "JG1JZL QM05 37"
__EEPROM_DATA(0xac,0xb1,0xa8,0x06,0x5a,0x79,0x2b,0x4e);
__EEPROM_DATA(0x2b,0xe6,0x44,0x49,0x86,0xf0,0x3c,0xc6);
__EEPROM_DATA(0x8c,0x0a,0x6b,0xa5,0x6f,0x8d,0x1a,0x37);
__EEPROM_DATA(0x82,0xb1,0x07,0x22,0x21,0x66,0x72,0x3e);
__EEPROM_DATA(0x0f,0xff,0xff,0xff,0xff,0xff,0xff,0xff);
__EEPROM_DATA(0xda,0x8a,0x4a,0x74,0x2e,0xbb,0xde,0x22); // "JG1JZL QM05 50"
__EEPROM_DATA(0xae,0xb1,0xaa,0x06,0x78,0x79,0x29,0x6c);
__EEPROM_DATA(0x29,0xc6,0x68,0x6b,0x84,0xd0,0x1c,0xe6);
__EEPROM_DATA(0x8e,0x2a,0x4b,0x87,0x4d,0x8d,0x1a,0x15);
__EEPROM_DATA(0x80,0xb1,0x25,0x02,0x21,0x66,0x72,0x3e);
__EEPROM_DATA(0x0f,0xff,0xff,0xff,0xff,0xff,0xff,0xff);

unsigned char    a_now, n4, n41, na41;         // use in assembler code

void    pic16f88(void){                        // hardware setting
            ANSEL=0;                           // digital port
```

```
            TRISA=0b00100000;                   // 1==input
            TRISB=0b11011110;                   // 0==output
            PORTA=0x00;                         //
            INTCON=0x00;                        // disable interrupt
            OPTION_REG=0x00;                    // pull-up all input
            T1CON=0x30;                         // TMR1 prescale 1:8, still stop
            CCP1CON=0x0c;                       // PWM mode
            PR2=0x13;                           // period=(19+1)/2.75MHz
            CCPR1=0x0a;                         // duty=50%
            TXSTA=0x00;                         // low baud rate
            RCSTA=0x90;                         // asynchronous
            if(RB3==1) SPBRG=0x23;              // select baud rate 4,800bps
            else    SPBRG=0x11;                 // or 9,600bps
            }

unsigned char    getch(void){                   // retrieve one byte
                   if(FERR)    RCREG;           // frame error
                   if(OERR){                    // overload error
                       CREN=0;                  // reset RCREG
                       CREN=1;                  // restart
                       }
                   while(!RCIF) continue;       // fill RCREG
                   return(RCREG);               // clear RCREG
                   }

void    train(void){                           // sequence of send a message
#asm
                BANKSEL(_PORTB)                ; bank0
            r00 BTFSS   _PORTB, 1              ; 00 1PPS rise
                goto    r00
            f00 BTFSC   _PORTB, 1              ; 00 1PPS fall
                goto    f00
                BSF     _PORTA, 2              ; PTT on
                CLRF    _na41
                CLRW                           ; default message
                BTFSC   _PORTB, 3              ; other message
                goto    fbyt
                ADDLW   0x30
            fbyt BANKSEL(_EEADR)               ; bank2
                MOVWF   _EEADR
                BANKSEL(_EECON1)               ; bank3
                BCF     _EECON1, 7             ; EEPGD
                BSF     _EECON1, 0             ; RD
```

```
    BANKSEL(_EEDATA)              ; bank2
    MOVF    _EEDATA, w
    MOVWF   _a_now                ;
    BANKSEL(_PORTA)               ; bank0
r01 BTFSS   _PORTB, 1             ; 01 1PPS rise
    goto r01
f01 BTFSC   _PORTB, 1             ; 01 1PPS fall
    goto    f01
r02 BTFSS   _PORTB, 1             ; 02 1PPS rise
    goto r02
f02 BTFSC   _PORTB, 1             ; 02 1PPS fall
    goto    f02
    BSF     _T2CON, 2             ; TMR2 start
    MOVLW   42                    ; 41 times
    MOVWF   _n41
bgn41 DECFSZ _n41, f
    goto    byt
    goto    end41
byt MOVLW   5                     ; 4 times
    MOVWF   _n4
bgn4 BCF    _PORTA, 0
     BCF    _PORTA, 1
     BCF    _PORTA, 3
    DECFSZ _n4, f
    goto    msb
    goto    end4
msb RLF     _a_now, f
    BTFSC   _STATUS, 0
    goto    lsb1
lsb0 RLF    _a_now, f
    BTFSC   _STATUS, 0
    goto    ra1
    goto    ra0
lsb1 RLF    _a_now, f
    BTFSC   _STATUS, 0
    goto    term
    goto    ra3
ra0 BSF     _PORTA, 0
    goto    term
ra1 BSF     _PORTA, 1
    goto    term
ra3 BSF     _PORTA, 3
term CLRF    _TMR1L
```

```
     CLRF    _TMR1H
     BCF     _PIR1, 0              ; TMR1IF clear
     BSF     _T1CON, 0             ; TMR1 start
 t_19 BTFSS  _PIR1, 0              ; step 0.19s
     goto    t_19                  ;
     BCF     _PIR1, 0              ; TMR1IF clear
 t_38 BTFSS  _PIR1, 0              ; step 0.38s
     goto    t_38                  ;
     BCF     _PIR1, 0              ; TMR1IF clear
 t_57 BTFSS  _PIR1, 0              ; step 0.57s
     goto    t_57                  ;
     BCF     _PIR1, 0              ; TMR1IF clear
     BCF     _T1CON, 0             ; TMR1 stop
     MOVLW   0x6b                  ; (65536-38058)*8
     MOVWF   _TMR1H                ;
     MOVLW   0x56                  ;
     MOVWF   _TMR1L                ;
     BSF     _T1CON, 0             ; TMR1 start
 t_64 BTFSS  _PIR1, 0              ; step 0.64s
     goto    t_64                  ;
     BCF     _PIR1, 0              ; TMR1IF clear
     BCF     _T1CON, 0             ; TMR1 stop
     goto    bgn4
 end4 INCF   _na41, f
     MOVF    _na41, w
     BTFSC   _PORTB, 3
     goto    nbyt
     ADDLW   0x30
 nbyt BANKSEL(_EEADR)              ; bank2
     MOVWF   _EEADR
     BANKSEL(_EECON1)              ; bank3
     BCF     _EECON1, 7            ; EEPGD
     BSF     _EECON1, 0            ; RD
     BANKSEL(_EEDATA)              ; bank2
     MOVF    _EEDATA, w
     BANKSEL(_PORTA)               ; bank0
     MOVWF   _a_now                ;
     goto    bgn41
end41 BTFSS  _PORTB, 1             ; 110 1PPS rise
     goto    end41
f110 BTFSC   _PORTB, 1             ; 110 1PPS fall
     goto    f110
     BCF     _T2CON, 2             ; TMR2 stop
```

```
           r111 BTFSS   _PORTB, 1              ; 111 1PPS rise
                goto    r111
           f111 BTFSC   _PORTB, 1              ; 111 1PPS fall
                goto    f111
                BCF     _PORTA, 2              ; PTT off
#endasm
               }// train()

void    t_even(void){                          // read NMEA message
            unsigned char   a4;                //
               do{
                  do{
                     do{                       // find "GGA"
                        while(getch()!='G') continue;
                        if(getch()=='G'&&getch()=='A') break;
                        } while(1);
                     for(a4=0; a4<=3; a4++) getch(); // overlook ",hhm"
                     a4=getch();               // get 'm' (a4 is other than above)
                     a4&=0x01;                 // LSB of 'm'
                     }while(a4);               // exit when even minute
                  if(getch()=='0'&&getch()=='0') return; // catch "00"
                  }while(1);
               }

void t_rate(void){
            unsigned char i, j, k;
            for(i=0; i<=5; i++){               // random and 20% duty ratio
               switch(i){
                  case 0:                      // rest a time
                     t_even();
                     t_even();
                     break;
                  case 1:                      // rest 4 times
                     for(j=0; j<=4; j++){
                        t_even();
                        }
                     break;
                  case 2:                      // rest 5 times
                     for(j=0; j<=5; j++){
                        t_even();
                        }
                     break;
                  case 3:                      // rest 4 times
```

```
                    for(j=0; j<=4; j++){
                       t_even();
                       }
                    break;
                  case 4:                       // rest 4 times
                    for(j=0; j<=4; j++){
                       t_even();
                       }
                    break;
                  case 5:                       // rest 5 times
                    for(j=0; j<=5; j++){
                       t_even();
                       }
                    break;
                  }// switch(i)
                train();
                if(!RB7){                       // 10% rate
                     for(k=0; k<=4; k++) t_even();
                     }// if(!RB7)
                }// for(i)
            }//

void t_const(void){
            unsigned char k;
            for(i=0; i<=4; i++){                // random and 20% duty ratio
                t_even();
                }
            train();
            if(!RB7){
                for(k=0; k<=4; k++)    t_even();
                }// if(!RB7)
            }

void main(void){
            unsigned int   i;
            pic16f88();                         // initialize
            while(1){
                if(RB6){                        // normal operation
//                    t_rate();                 // random
                      t_const();                // constant
                   }
                else{                           // frequency adjust
                   RA2=1;                       // PTT on
```

```
                for(i=0; i<=0xefff; i++) ;  // wait for relay make
                 RA0=1;                     //
                 RA1=1;                     //
                 RA3=1;
                 TMR2ON=1;                  // carrier on
                 }
            }// while(1)
        }// main()

/*      unsigned char    mode;              // develope mode
        mode=RATE;
        while(1){
            switch(mode){
                case(ONE):                  // send a unit
                    train();
                    break;
                case(EVEN):                 // send every even minute
                    t_even();
                    train();
                    break;
                case(RATE):                 // random send
                    t_rate();
                    break;
                default:
                    ;
                }// switch()
        }// while(1)
*/
```

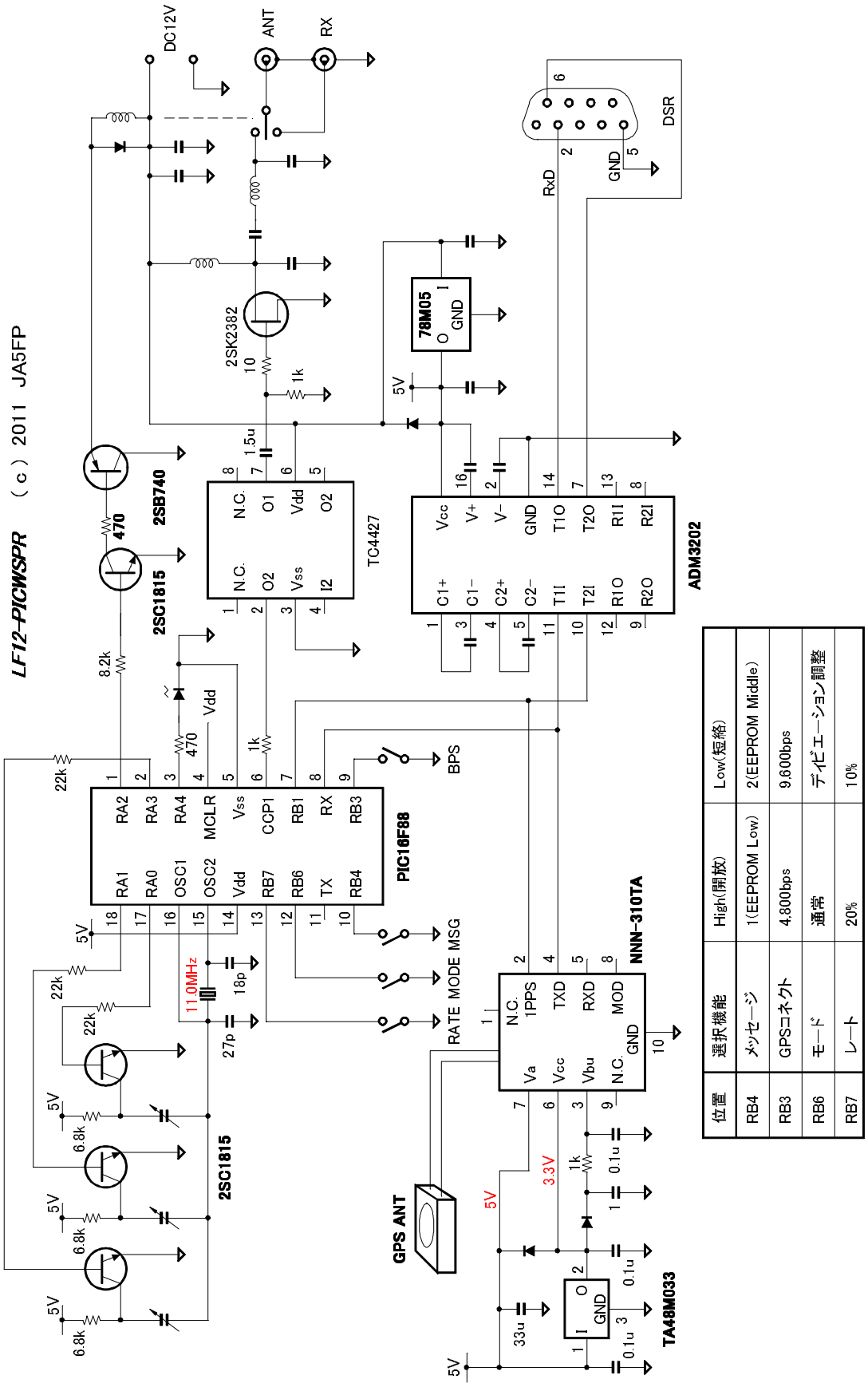

図 4: PICWSPR 回路図