```
// si5351a ARDUINO  setting
// Rotary switch:センシング、機能選択スイッチ：アナログポート
//  Si5351_ADDR = 0x60

//  20201025 Append LCD&KEY setting   next task:scan, search scan
//  To replace for debugging composit I decided compile switch as below
//  Compile switch ifdef LCD_KEY
//    LiquidCrystal lcd( 8, 9, 4, 5, 6, 7);  // for LCD&key

//  ifdef chibi
//    LiquidCrystal    lcd(5,6,9,10,11,12);
//    RS:D5,EN:D6,D9-11:LCD I/F

//  LCD display 0123456789abcdef    20201003 changed
//  1602        100.000MHz100kHz
//              M00S**********IF

// How to set fequency on Si5351a via i2c was based on TJ lab
// Thanks Dr.Uebo "https://tj-lab.org/2017/03/13/si5351/"

#include <Wire.h>
#include <LiquidCrystal.h>   // Sure 8x2 LCD works with this lib  ―→16ｘ2

// Thanks Kimura "http://zattouka.net/GarageHouse/micon/Arduino/LCD/I2CLCD.htm"
#include <skI2CLCDlib.h>

//lcd instance
  skI2CLCDlib LCD(0x3E, 16); // ＬＣＤのi2cアドレス、画面カラム数16文字

/* ChibiDuino2 LCD sample    ―――＞ LiquidCrystal lcd( 8, 9, 4, 5, 6, 7);
 * SETUP: (Hitachi HD44780 compatable LCD)
 * +5V LCD Vdd pin 2       >>>     Gnd LCD Vss pin 1, and R/W pin 5
 * LCD RS pin 4 to D5      >>>     LCD Enable pin 6 to D6
 * LCD D4 pin 11 to D9     >>>     LCD D5 pin 12 to D10
 * LCD D6 pin 13 to D11    >>>     LCD D7 pin 14 to D12
 * 10K pot: - ends to +5V and Gnd, wiper to LCD VO pin (pin 3)
 *   LiquidCrystal    lcd(5,6,9,10,11,12);
 */


#define LCD_Key       // or #define chibi

#ifdef LCD_Key
   LiquidCrystal lcd( 8, 9, 4, 5, 6, 7);  // for LCD&key
#endif

   #define SQUELTCH A3
   #define REA 15               // A1
   #define REB 16               // A2
```

```
#ifdef chibi
/*
 * instanciate the library and pass pins for (RS, Enable, D4, D5, D6, D7)
   LCD KeyPad Shield    A0のスイッチテスト
 #include <LiquidCrystal.h>
 LiquidCrystal lcd( 8, 9, 4, 5, 6, 7);  // for LCD&key
 void setup() {  lcd.begin(16, 2);
-----------------------------
  LiquidCrystal   lcd(5,6,9,10,11,12);  //  for chibiduino LCD
*/
// instanciate the library and pass pins for (RS, Enable, D4, D5, D6, D7)
//  for chibiduino LCD
  LiquidCrystal   lcd(5,6,9,10,11,12);
#endif

//
#include <EEPROM.h>

#define Si5351_ADDR 0x60
#define led_pin 13

#define NONE 0                // A1 port 1023 MAX
#define FREQ 1                // 7 modes defined
#define STEP 2
#define MEMORY 3
#define SCAN 4
#define FREQLONG 5            //  周波数書き込み
#define STEPPUT 6             //  STEPかきこみ
#define MEMORYPUT 7           //  メモリー長押し  周波数設定
#define SCANOFF 8             // SCAN長押し自動スキャン

#define FREQ_AD 0             // EEPROM address for FREQ data as long
#define FSTEP_AD 4            // EEPROM address for FSTEP data as long
#define MCHAN0 8              // MCHAN 0- 9  MCHAN0 + 4 * chan

/*
Rotary encoder handler for arduino. v1.1
Copyright 2011 Ben Buxton. Licenced under the GNU GPL Version 3. Contact:
bb@cactii.net
http://www.buxtronix.net/2011/10/rotary-encoders-done-properly.html
Rotary encoder test debug scketch
 PWM/3: Rotary A --> A1=15 -->D1
 PWM/2: Rotary B --> A2=16 -->D2
 GND: C
*/
#include <Rotary.h>
  Rotary r = Rotary(REA, REB);
```

```
// functions
void LCD_Disp ( char, char, char*);
void Dsp_Disp(void);
void Step_Disp(void);
void LongToStr(long , char*);
void Fdds_Space(char *);
void set_freq(unsigned long);
void si5351_init(void);
void cmd_si5351(char , char);
int function_key(void);
void monit(long,long);
unsigned char mode_define(unsigned char) ;
void sw_state_reset(void);
void s_meter_disp(void);

// variables
  volatile boolean if_mode;             //  1：局発モード10.7MHｚ＋
  volatile unsigned char mode = FREQ;   //  select mode as FREQ
  volatile unsigned char mode_last = FREQ;
  volatile char memo_ad = 0;            //  select  memory address
  unsigned char  set_sw = 1;            //  set freq on memory, or step to freq
  volatile unsigned long int freq1, fstep, freq_0;
                                        // freq1: current F, freq_0; last F
  volatile unsigned long int_time=0 ;
  volatile unsigned long time0=0;
  volatile unsigned long time1=0;
  volatile unsigned char sw_result=0, sw_result0=5; //
NONE:0,FREQ:1,STEP:2,MEMORY:3,SCAN:4

  volatile int ADCdata;
  volatile char scan_ad = 0;            //  scanning address counter
  volatile boolean k;                   // led ON/OFF status
  volatile unsigned char sw_pending = 0;
  volatile char i = 0;
  volatile unsigned char re_result = 0; // Rotary switch active result 0x10:right,
0x20:left
  volatile unsigned char result;
  char mdisp[4] = { 0x4D, 0x30, 0x20, 0};
  char stepdisp[4] = { 0x53, 0x54, 0x20,0 };
  char spdisp[4] = { 0x20, 0x20, 0x20, 0 };
  char scandisp[4] = { 0x53, 0x30, 0x20, 0 };
  char freqdisp[4] = { 0x46, 0x20, 0x20, 0 };
  volatile int agc_volt;

void setup() {
  Wire.begin();
  si5351_init();              // 100MHz set

  Serial.begin(9600);
```

```
 Serial.println("\nsi5351a oscillator V1.2");

 pinMode(led_pin, OUTPUT);



 pinMode(REA, INPUT_PULLUP);
 pinMode(REB, INPUT_PULLUP);

  sw_result0 = function_key();          //  受信機局発10.7MHｚモード開始判定
  if(sw_result0 == 0) if_mode=1;        //  電源開始時何もキー無いと10.7MHｚ＋局発で動く

// i2c ＬＣＤモジュールの初期化処理
// ICON OFF,コントラスト(0-63),VDD=5Vで使う
 LCD.Init(LCD_NOT_ICON,32,LCD_VDD5V) ; //5Vへ変更

// 並列接続LCDモジュールの初期化処理
  lcd.begin(16,2);                     // set lib for display size (8x2)
  lcd.clear();                         // clear the screen

  LCD_Disp ( 0, 0, "si5351a osc V1.2") ;
  delay(500);

  r.begin(true);

// #13 LED on_off
//   monit(500,200);              // RB6 LEDをON/OFF時間設定する関数
// pinMode (led_pin, OUTPUT), int mon = FREQ,

// EEPROM data recover for clock frequency data
// There are FREQ, FSTEP, and MCHAN0-9

    EEPROM.get( FREQ_AD, freq1);       // FREQ data  recover from EPROM
    EEPROM.get( FSTEP_AD, fstep);      // fstep data from EEPROM Read

    Dsp_Disp();                        // display freq
   Serial.println("\nFreq from EEPROM = ");
   Serial.println(freq1);
   freq_0=freq1;


   if(freq1<10000000 || freq1>200000000) {  //10-200MHz
       LCD_Disp ( 0, 0, "Freq Read Error  ") ;
       Serial.println("\nFreq Read Error");
       Serial.println(freq1);
       freq1 = 100000000;
       EEPROM.put( FREQ_AD, freq1);      // FREQ data  100MHz for EPROM
       freq_0 =  100000000;
       for ( memo_ad =0 ; memo_ad <50; memo_ad++ ) {
         EEPROM.put( FREQ_AD + 8 + memo_ad*4, freq_0);      // FREQ data  100MHz
```

```
for EPROM
        }
        delay(2000);
        LCD.PageClear() ;
        lcd.clear();                        // 1秒後いったんLCDをクリアする
        delay(500);
    }
   if( fstep==1000 || fstep==10000 || fstep==100000 || fstep==1000000 ||
fstep==10000000) {
    }
    else{
        LCD_Disp ( 0, 1, "Step Read Error") ;
        Serial.println("\nStep Read Error");
        Serial.println(fstep);
        fstep=100000;
        EEPROM.put( FSTEP_AD, fstep );   // fstep data 100kHz for EEPROM
        delay(1000);
        LCD.PageClear() ;
        lcd.clear();                        // 1秒後いったんLCDをクリアする
        delay(1000);
    }

    Dsp_Disp();                             // 現在周波数を表示する
    Step_Disp();                            // 現在STEP周波数を表示する
   freq_0=freq1;
   if (if_mode==1) freq_0=freq1+10700000 ; //  IFモード時は10.7MH上位を発振
    set_freq(freq_0 );                      // 現在周波数をSi5351aに設定する
   freq_0=freq1;
}

void loop() {
  unsigned char mode_last;

  sw_result0 = function_key();              //  A0ポートの電圧からキー判定
  mode_last = mode ;                        //  20201002
  mode = mode_define(mode_last);            //  キー入力受付と長押しキー判定

  switch ( mode ){
    case 1:{                                //  FREQ determining mode
      LCD_Disp ( 0, 1, freqdisp ) ;
      break;
    }
    case 2:{                                //  STEP select mode
      LCD_Disp ( 0, 1, stepdisp ) ;
      break;
    }
    case 3:{                                //  MEMORY ch select mode
      LCD_Disp ( 0, 1, mdisp ) ;
      break;
```

```
  }
  case 8: {                             //  AUTO SCAN mode from M-chan
    scandisp[1] = 0x30 | (scan_ad/10) ;     //  display scan chan
    scandisp[2] = 0x30 | (scan_ad%10) ;     //  display scan chan
    LCD_Disp ( 0, 1, scandisp ) ;
    EEPROM.get( MCHAN0 + scan_ad*4, freq1); // read out MEM frequency
     scan_ad++;                         //  Changed 20201013
    if (scan_ad > 49) scan_ad = 0;
    delay(200);
    if (freq1==100000000) break;        // If 100MHz, then skip.
    if (freq1==freq_0) break;           // If same as with last one, then skip
    Dsp_Disp();
    freq_0=freq1;
    if (if_mode==1) freq_0=freq1+10700000 ; //  IFモード時は10.7MH上位を発振
    set_freq(freq_0);
    freq_0=freq1;
    Serial.println("freq SCAN");

    if ( agc_volt > 200) delay(1000);  // If squelch off, wait 1 sec
    break;
     }
  case 5:{                              //  FREQ button long pus

      LCD_Disp ( 0, 1, spdisp) ;        //  2行目1文字目space表示
      delay(500);
      LCD_Disp ( 0, 1, freqdisp) ;      //  2行目1文字目freq表示
      EEPROM.put( FREQ_AD, freq1 );     //  freq data into EEPROM
      Serial.println("freq written");
      delay(500);
      mode=1;
      break;
  }
  case 6:{                              //  STEP written

      LCD_Disp ( 0, 1, spdisp) ;        //  2行目1文字目space表示
      EEPROM.put( FSTEP_AD, fstep );    //  fstep data 100kHz for EEPROM
      LCD_Disp ( 0, 1, spdisp) ;        //  2行目1文字目spacep表示
      delay(500);
      LCD_Disp ( 0, 1, stepdisp) ;        //  2行目1文字目step表示
      Serial.println("fstep written");
      delay(500);
      mode=2;
      break;
  }
  case 7:{                              //  MEMORY write

      LCD_Disp ( 0, 1, mdisp) ;         //  2行目1文字目m表示
      EEPROM.put( MCHAN0 + memo_ad*4, freq1);   // Write FREQ data on defined
EEPROM
```

```
      delay(500);
      LCD_Disp ( 0, 1, spdisp) ;       //  2行目1文字目space表示
      delay(500);
      LCD_Disp ( 0, 1, mdisp) ;        //  2行目1文字目mem表示
     Serial.println("memory write=");
     Serial.println( mdisp);
      mode=3;
      delay(500);
      break;

    }
    case 4:{                            //  Manual step SCAN
      scandisp[1] = 0x30 | (scan_ad/10) ;//  display scan chan
      scandisp[2] = 0x30 | (scan_ad%10) ;
      LCD_Disp ( 0, 1, scandisp ) ;
      EEPROM.get( MCHAN0 + scan_ad*4, freq1);  // read out MEM frequency
      if (freq_0 == freq1) break;
      Dsp_Disp();
      freq_0=freq1;
      if (if_mode==1) freq_0=freq1+10700000 ;  //  IFモード時は10.7MH上位を発振
      set_freq(freq_0);
      freq_0=freq1;
      break;
    }
  }

if(mode==0) mode=1;

// led
   digitalWrite(led_pin, k=!k );   //  Led blinked

// もしロータリーエンコーダーが変化したのなら、モードに応じて
// freqの増減、step周波数の増減、指定メモリーchの増減を行う
   re_result = r.process();

   if (re_result  )  { // Left 0x20 ? Right 0x10?
      Serial.println(re_result == DIR_CW ? "Right" : "Left");
      Serial.println(mode);
      if ((mode == FREQ) && (re_result == 0x10 ) ) {  // clockwise FREQ
         freq1 = freq1 + fstep;
         if(freq1 > (long) 200000000) {          //  200MHz -> 100MHz
             freq1 = (long) 100000000;
         }
         freq_0=freq1;
        if (if_mode==1) freq_0=freq1+10700000 ; // on IF mode 10.7MH upper
         set_freq(freq_0);
         Dsp_Disp();
         freq_0=freq1;
         digitalWrite(led_pin, k=!k );       // #13 LED on/off in turn;
```

```
      }
      else if ((mode == FREQ) && (re_result == 0x20) ) { //counter clockwise
FREQ

         freq1 = freq1 - fstep;
         if(freq1 < 100000000) {            // 100MHz->200MHz
            freq1 = 200000000;
         }
         freq_0=freq1;
         if (if_mode==1) freq_0=freq1+10700000 ; //  on IF mode 10.7MH upper
         set_freq(freq_0);
         Dsp_Disp();
         freq_0=freq1;
         digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn
      }

      else if ((mode == STEP) && (re_result == 0x10) ) { // clockwise STEP
         fstep = fstep*10;
         if (fstep > 10000000)  fstep=1000; // 100-->1000
         Step_Disp();
         digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn

      }
      else if ((mode == STEP) && (re_result == 0x20 )) { // counter clockwise
STEP
         fstep = fstep/10;
         if (fstep < 1000) fstep=10000000;  //  100-->1000
         Step_Disp();
         digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn
      }
      else if ((mode == MEMORY) && (re_result == 0x10 )) { // clockwise STEP
         memo_ad ++ ;
         if (memo_ad > 49)  memo_ad = 0;
         mdisp[1] = 0x30 | (memo_ad/10) ;
         mdisp[2] = 0x30 | (memo_ad%10) ;
         LCD_Disp ( 0, 1, mdisp) ;          // #13 LED on/off in turn
      }
      else if ((mode == MEMORY) && (re_result == 0x20 )) {// Counter clockwise
STEP
         memo_ad --;
         if (memo_ad < 0)  memo_ad = 49;
         mdisp[1] = 0x30 | (memo_ad/10) ;
         mdisp[2] = 0x30 | (memo_ad%10) ;
         LCD_Disp ( 0, 1, mdisp) ;
         digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn
      }
      else if ((mode == SCAN) && (re_result == 0x10 )) {// clockwise STEP
         scan_ad ++;
         if (scan_ad > 49)  scan_ad = 0;
```

```
        scandisp[1] = 0x30 | (scan_ad/10) ;
        scandisp[2] = 0x30 | (scan_ad%10) ;
        LCD_Disp ( 0, 1, scandisp) ;
        digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn
      }
     else if ((mode == SCAN) && (re_result == 0x20 )) {// Counter clockwise
STEP
        scan_ad --;
        if (scan_ad < 0)  scan_ad = 49;
        scandisp[1] = 0x30 | (scan_ad/10) ;
        scandisp[2] = 0x30 | (scan_ad%10) ;
        LCD_Disp ( 0, 1, scandisp) ;
        digitalWrite( led_pin, k=!k );     // #13 LED on/off in turn
      }
    }
    s_meter_disp();                        // Get AGC voltage & change S-meter
}

// i2c LCD & 4bit parallel LCD display commonly
void LCD_Disp ( char column, char line, char* charbuf) {
  LCD.SetCursor( column, line) ;           // Set column position
  LCD.Puts( charbuf ) ;                    // Let to display
  lcd.setCursor( column, line);
  lcd.print( charbuf ) ;
}

//***** fstep displaying ***** 1kHz,10kHz,100kHz,1MHz,10MHz
void Step_Disp() {                         // display fstep on LCD
    if (fstep == 1000)  {
        LCD_Disp ( 10, 0, " 1kHz ") ;
        Serial.println("Step 1kHz   ");
    }
    else if (fstep == 10000) {
        LCD_Disp ( 10, 0, " 10kHz") ;
        Serial.println("Step 10kHz   ");
    }
    else if (fstep == 100000) {
        LCD_Disp ( 10, 0, "100kHz") ;
        Serial.println("Step 100kHz   ");
    }
    else if (fstep == 1000000) {
        LCD_Disp ( 10, 0, " 1MHz ") ;
        Serial.println("Step 1MHz   ");
    }
    else {
        LCD_Disp ( 10, 0, " 10MHz") ;
        Serial.println("Step 10MHz   ");
    }
    delay(100);
```

```
}


//********* display frequency *********
    char fdds[16];
void Dsp_Disp() {                // Format freq data for display & display
    char fdds1[16];
    const unsigned char fdds2[] = "test";
    LongToStr(freq1, fdds);      // convert Long data 16 columns
    Fdds_Space(fdds);            // change leading "0" to space

    fdds1[11] =  '\0';
    fdds1[10] =  '\0';
    fdds1[9] = 'z';
    fdds1[8] =  'H';
    fdds1[7] =  'M';
    fdds1[6] = fdds[7];
    fdds1[5] = fdds[6];
    fdds1[4] = fdds[5];
    fdds1[3] = '.';
    fdds1[2] = fdds[4];
    fdds1[1] = fdds[3];
    fdds1[0] = fdds[2];

    LCD_Disp ( 0, 0, fdds1) ;
    if (if_mode==1) LCD_Disp ( 14, 1, "IF") ;   //  IF mode
}

// Long FREQ data changes to 10 digit numerics
void LongToStr(long dat, char *fm) {
    long fmdat;
    int i;
    fmdat = dat;
    for (i=0; i<11; i++) {
        fm[10-i] =( fmdat % 10 ) | 0x30;
        fmdat = fmdat / 10;
    }
}

void  Fdds_Space(char *fm) {   // Reducing preceding zero into space
    int i;
    for (i=0; i<11; i++) {
        if( fm[i] == 0x30) {
            fm[i]=0x20;
        }
        else i=11;
    }
}
```

```
void monit(long on_time, long off_time) {  // #13 LED monit on_off
      digitalWrite(led_pin, k=!k );       // Mon Led on/off
      delay(on_time);
      digitalWrite(led_pin, k=!k );
      delay(off_time);
}

/* Frequency setting on Si5351a via i2c
Thanks Dr.Uebo
                                        */

void set_freq(unsigned long freqs) {
//    freqs [Hz]
//
//    fvco= fxtal*(a+b/c)  ( a:15 -- 90,   b:0 -- 1048575, c:1 -- 1048575 )
//    freq= fvco /(a+b/c)  ( a:4, 6--1800, b:0 -- 1048575, c:1 -- 1048575 )
//
//    P1= 128*a +   floor(128*b/c) - 512
//    P2= 128*b - c*floor(128*b/c)
//    P3= c
//

  int k;
  unsigned long M;
  unsigned int R;

 if(freqs<1500) freqs=1500; else if(freqs>280000000) freqs=280000000;

  if(     freqs> 150000000){M=4; R=0;}
  else if(freqs>=63000000){M=6; R=0;}
  else if(freqs>=27500000){M=14; R=0;}
  else if(freqs>=13000000){M=30; R=0;}
  else if(freqs>= 6500000){M=62; R=0;}
  else if(freqs>= 3000000){M=126; R=0;}
  else if(freqs>= 1500000){M=280; R=0;}
  else if(freqs>=  700000){M=600; R=0;}
  else if(freqs>=  330000){M=1280; R=0;}
  else if(freqs>=  150000){M=1300; R=1;}
  else if(freqs>=   67000){M=1500; R=2;}
  else if(freqs>=   30300){M=1600; R=3;}
  else if(freqs>=   14000){M=1800; R=4;}
  else if(freqs>=    7000){M=1800; R=5;}
  else if(freqs>=    3500){M=1800; R=6;}
  else{M=1800; R=7;}

  freqs*=M;
  freqs<<=R;

  unsigned long c=0xFFFFF;
```

```
 unsigned long a=freqs/25000000;
 unsigned long b=(long)((double)(freqs-a*25000000)*(double)c/(double)25000000);
 unsigned long dd=(128*b)/c;
 unsigned long P1=128*a+dd-512;
 unsigned long P2=128*b-c*dd;
 unsigned long P3=c;


 //Set fvco of PLL_A
   cmd_si5351(26,(P3>>8)&0xFF);        //MSNA_P3[15:8]
   cmd_si5351(27,P3&0xFF);             //MSNA_P3[7:0]
   cmd_si5351(28,(P1>>16)&0x03);       //MSNA_P1[17:16]
   cmd_si5351(29,(P1>>8)&0xFF);        //MSNA_P1[15:8]
   cmd_si5351(30,P1&0xFF);             //MSNA_P1[7:0]
  cmd_si5351(31,(P3>>12)&0xF0|(P2>>16)&0x0F);//MSNA_P3[19:16], MSNA_P2[19:16]
   cmd_si5351(32,(P2>>8)&0xFF);        //MSNA_P2[15:8]
   cmd_si5351(33,P2&0xFF);             //MSNA_P2[7:0]

 // Set MS0, MS1
 // a=M, b=0, c=1 ---> P1=128*M-512, P2=0, P3=1
 if(M==4){
   P1=0;
   cmd_si5351(42,0);                   //MS0_P3[15:8]
   cmd_si5351(43,1);                   //MS0_P3[7:0]
   cmd_si5351(44,0b00001100);         //0,R0_DIV[2:0],MS0_DIVBY4[1:0],
MS0_P1[17:16]
   cmd_si5351(45,0);                   //MS0_P1[15:8]
   cmd_si5351(46,0);                   //MS0_P1[7:0]
   cmd_si5351(47,0);                  //MS0_P3[19:16], MS0_P2[19:16]
   cmd_si5351(48,0);                   //MS0_P2[15:8]
   cmd_si5351(49,0);                   //MS0_P2[7:0]

   cmd_si5351(50,0);                   //MS1_P3[15:8]
   cmd_si5351(51,1);                   //MS1_P3[7:0]
   cmd_si5351(52,0b00001100);         //0,R1_DIV[2:0],MS1_DIVBY4[1:0],
MS1_P1[17:16]
   cmd_si5351(53,0);                   //MS1_P1[15:8]
   cmd_si5351(54,0);                   //MS1_P1[7:0]
   cmd_si5351(55,0);                  //MS1_P3[19:16], MS0_P2[19:16]
   cmd_si5351(56,0);                   //MS1_P2[15:8]
   cmd_si5351(57,0);                   //MS1_P2[7:0]
  }
  else{
   P1=128*M-512;
   cmd_si5351(42,0);                    //MS0_P3[15:8]
   cmd_si5351(43,1);                    //MS0_P3[7:0]
 cmd_si5351(44,(R<<4)&0x70|(P1>>16)&0x03);//0,R0_DIV[2:0],MS0_DIVBY4[1:0],
MS0_P1[17:16]
   cmd_si5351(45,(P1>>8)&0xFF);         //MS0_P1[15:8]
```

```
    cmd_si5351(46,P1&0xFF);             //MS0_P1[7:0]
    cmd_si5351(47,0);                   //MS0_P3[19:16], MS0_P2[19:16]
    cmd_si5351(48,0);                   //MS0_P2[15:8]
    cmd_si5351(49,0);                   //MS0_P2[7:0]

    cmd_si5351(50,0);                   //MS1_P3[15:8]
    cmd_si5351(51,1);                   //MS1_P3[7:0]
  cmd_si5351(52,(R<<4)&0x70|(P1>>16)&0x03);//0,R1_DIV[2:0],MS1_DIVBY4[1:0],
MS1_P1[17:16]
    cmd_si5351(53,(P1>>8)&0xFF);        //MS1_P1[15:8]
    cmd_si5351(54,P1&0xFF);             //MS1_P1[7:0]
    cmd_si5351(55,0);                   //MS1_P3[19:16], MS0_P2[19:16]
    cmd_si5351(56,0);                   //MS1_P2[15:8]
    cmd_si5351(57,0);                   //MS1_P2[7:0]
  }
 cmd_si5351(165,0);
 cmd_si5351(166,M);

  cmd_si5351(177,0xA0);    // Reset PLL_A.
Mが変化（特にM=4からM>=6に変化）するときは必須かも…
}

// Si5351a set up at 100MHz
void si5351_init(void) {
    cmd_si5351(183,0b10010010);         // CL=8pF
    cmd_si5351(16,0x80);                // Disable CLK0
    cmd_si5351(17,0x80);                // Disable CLK1
    set_freq(100000000);                // 100MHz
    cmd_si5351(177,0xA0);               // Reset PLL_A
    cmd_si5351(16,0x4F);               // Enable CLK0 (MS0=Integer Mode,
Source=PLL_A)
    cmd_si5351(17,0x4F);               // Enable CLK1 (MS1=Integer Mode,
Source=PLL_A)
}

//レジスタに１バイトデータを書き込む。
void cmd_si5351(char Reg , char Data) {
  Wire.beginTransmission(Si5351_ADDR);
   Wire.write(Reg);
   Wire.write(Data);
   Wire.endTransmission();
}

// Function SW determining  A0 port: function SW
// Vcc 2k SW1 330 SW2 630 SW3 1k SW4 Gnd
int function_key() {

   ADCdata = analogRead(A0);       // define sw as voltage
   result = NONE ;                 //
```

```
//   if ( abs(ADCdata - analogRead(A0)) <5 )   //  To avoid chatter
//   {
     if (ADCdata < 50) {           // SW4
       result = FREQ ;
     }
     else if (ADCdata < 150) {     // SW3
        result = STEP ;
     }
     else if (ADCdata < 350) {     // SW2
        result = MEMORY ;
     }
     else if (ADCdata < 600) {     // SW1
        result = SCAN ;
     }
     else {
        result = NONE ;            // over 600 NONE
        sw_pending=0;              // 20201002
        sw_result = NONE ;         // 20201002
     }
//   }
   return result ;                 // Switch not settled 20201002
}


unsigned char mode_define(unsigned char moded) {
  if ( sw_result0 ) {              // NONEでなかったらSW on
     if ( sw_result0 != sw_result ) {  //  前のスイッチから変化
       time0 = millis();           //  SWが変化した時間を覚える
       moded = sw_result0;         // sw shall be accepted
       sw_pending=1;               // Sw changed, and on pending.
       sw_result = sw_result0;     // Remember sw
       return moded;
     }
     else{                         //  前のスイッチから変化せず
       time1=millis();             //  長押しチェックのために時間覚え
      if ((( time1 - time0) > 800 )& sw_pending==1 ){ //  800ms超えた, sw shall
be accepted as long pushed
         moded = sw_result0 + 4 ;  //  長押しモード受付、数字はそれぞれの4まし
         sw_pending=0;
         return moded;
       }
     }
   }
   return moded;
}

void s_meter_disp(){               //  Pendig as debugging
  char s_disp[11];
  unsigned char i;
```

```
 unsigned char s_value;
 agc_volt = analogRead(SQUELTCH);
 if (agc_volt<143) s_value=0;
 else if (agc_volt<168) s_value=1;
 else if (agc_volt<205) s_value=2;
 else if (agc_volt<254) s_value=3;
 else if (agc_volt<328) s_value=4;
 else if (agc_volt<426) s_value=5;
 else if (agc_volt<573) s_value=6;
 else if (agc_volt<770) s_value=7;
 else if (agc_volt<1000) s_value=8;
 else s_value=9;

 s_disp[0] = 0x53;
 for ( i=0; i<10; i++){
   if (s_value>i){                  //  i=0 s_value=1 set 1 S_MAX=8
     s_disp[i+1] = 0xff;
   }
   else s_disp[i+1] = 0x20;
 }
 LCD_Disp ( 3, 1, s_disp) ;
}
```