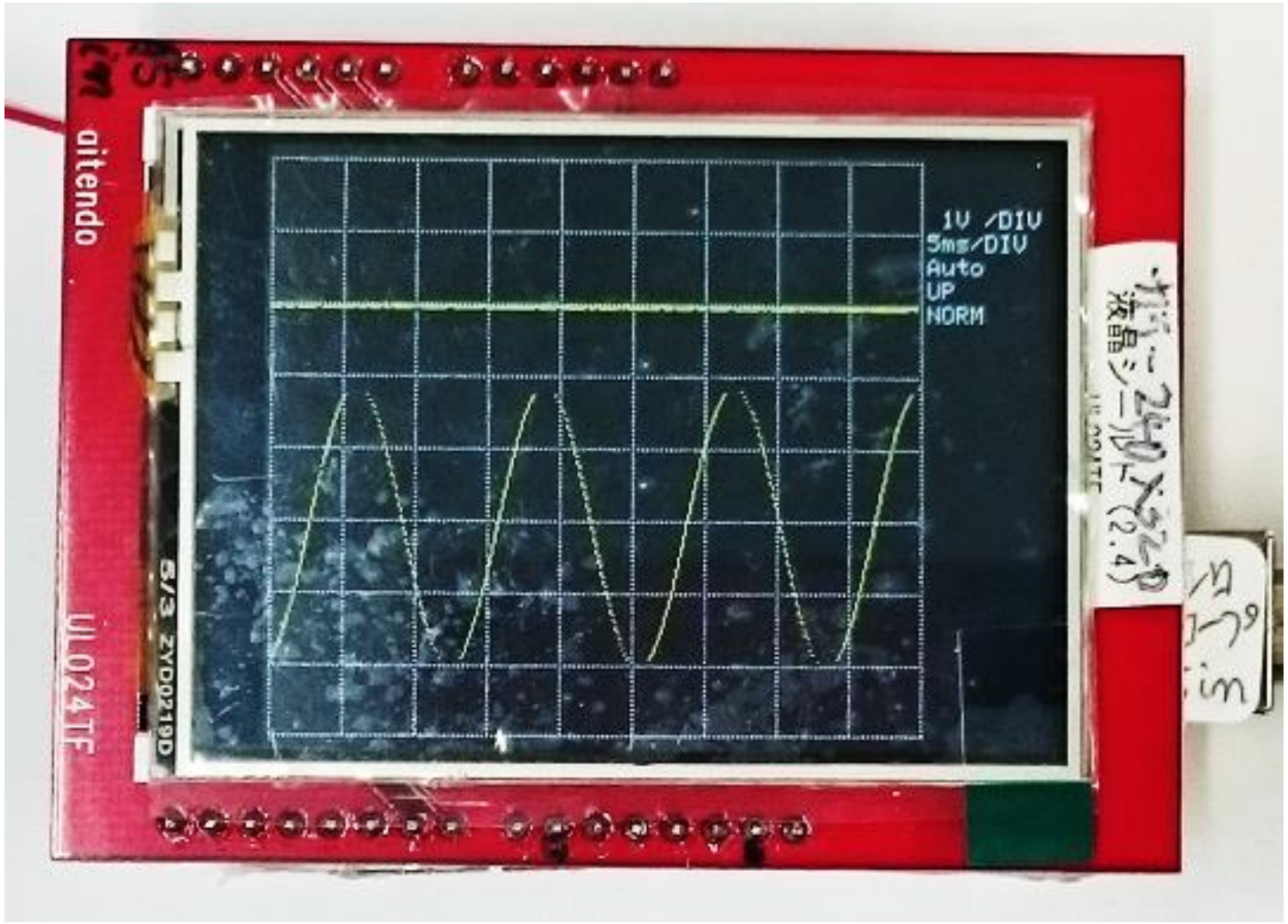
# “びんぼうでいいの”によるLCDシールド aitendo UL024TF のオシロスコープモード試験

2016.7.7




Aitendo UL024TF 2.4吋 （240x320）　　aitendou http://www.aitendo.com/product/10471

試験信号：50Hz, 4Vp-p,Offset=2V
nf=WF1973ファンクションジェネレータ

100Hz が限界周波数

導入案内：http://blog.boochow.com/archives/20150718-1.html

# 大元スケッチソース　http://n.mtng.org/ele/arduino/oscillo-j.html　http://researchmap.jp/read0054750/　MITSUNAGA Noriaki　先生
# 参考スケッチソース　http://make.kosakalab.com/make/tft_lcd_arduino/

```
/* オシロ風表示、aitendo UL024TF 240x320  mtsw
 *  入力は  A5 （左下、端）
 *  2016.7.6　ライブラリのヘッダ を修正して動作する

  大元は  2.6インチ液晶 for Arduino UNO  である。
  しかし、2.4インチでも無事に表示できることが分かった。
 ------------------------------------------------------------
 *  static unsigned char  font[] PROGMEM = {                ^
exit status 1
コンパイル時にエラーが発生しました。
↓ライブラリ記述を修正して、エラー取れる
 http://make.kosakalab.com/make/tft_lcd_arduino/
         static unsigned char  font[] PROGMEM = {  を
         const unsigned char  font[] PROGMEM = {  と修正
 ------------------------------------------------------------
オシロっぽい表示が可能となった。
大変、有難い！！！！
但し以下のメッセージをもらう！！
コンパイル！書き込み時！

  最大32,256バイトのフラッシュメモリのうち、スケッチが10,548バイト（32%）を使っています。
  最大2,048バイトのRAMのうち、グローバル変数が1,659バイト（81%）を使っていて、ローカル変数
  で389バイト使うことができます。
  スケッチが使用できるメモリが少なくなっています。動作が不安定になる可能性があります。
*/

/*
 * Arduino Oscilloscope using a graphic LCD
 * The max sampling rates are 4.3ksps with 2 channels and 8.6ksps with a channel.
 * Copyright (c) 2009-2014, Noriaki Mitsunaga
 */

#include "TFTLCD.h"

#define LCD_CS A3
#define LCD_CD A2
#define LCD_WR A1
#define LCD_RD A0
#define LCD_RESET A4

TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);

// #include <Arial14.h>  // font definitions

#define txtLINE0   0
#define txtLINE1   16
#define txtLINE2   30
#define txtLINE3   46

const int LCD_WIDTH = 320;
const int LCD_HEIGHT = 240;
const int SAMPLES = 270;
const int DOTS_DIV = 30;

const int ad_sw = 3;                // Analog 3 pin for switches
const int ad_ch0 = 4;               // Analog 4 pin for channel 0
const int ad_ch1 = 5;               // Analog 5 pin for channel 1
const unsigned long VREF[] = {150, 300, 750, 1500, 3000}; // reference voltage 5.0V ->  150 :   1V/div range (100mV/dot)
                                    // It means 5.0 * DOTS_DIV = 150. Use 4.9 if reference voltage is 4.9[V]
                                    //                  -> 300 : 0.5V/div
                                    //                  -> 750 : 0.2V/div
                                    //                  ->1500 : 100mV/div
                                    //                 -> 3000 :  50mV/div
const int MILLIVOL_per_dot[] = {33, 17, 6, 3, 2}; // mV/dot
const int MODE_ON = 0;
const int MODE_INV = 1;
const int MODE_OFF = 2;
const char *Modes[] = {"NORM", "INV", "OFF"};
const int TRIG_AUTO = 0;
const int TRIG_NORM = 1;
const int TRIG_SCAN = 2;
const int TRIG_ONE  = 3;
const char *TRIG_Modes[] = {"Auto", "Norm", "Scan", "One"};
const int TRIG_E_UP = 0;
const int TRIG_E_DN = 1;
#define RATE_MIN 0
#define RATE_MAX 13
const char *Rates[] = {"F1-1", "F1-2 ", "F2  ", "5ms", "10ms", "20ms", "50ms", "0.1s", "0.2s", "0.5s", "1s", "2s", "5s",
"10s"};
#define RANGE_MIN 0
#define RANGE_MAX 4
const char *Ranges[] = {" 1V ", "0.5V", "0.2V", "0.1V", "50mV"};
unsigned long startMillis;
byte data[4][SAMPLES];              // keep twice of the number of channels to make it a double buffer
byte sample=0;                      // index for double buffer

////////////////////////////////////////////////////////////////////////////////////////////////
// Define colors here
#define                     BGCOLOR  BLACK
#define                     GRIDCOLOR WHITE
#define CH1COLOR  BLUE
#define CH2COLOR  YELLOW

// Declare variables and set defaults here
// Note: only ch1 is available with Aitendo's parallel 320x240 TFT LCD
byte range0 = RANGE_MIN, ch0_mode = MODE_OFF;  // CH0
short ch0_off = 204;
byte range1 = RANGE_MIN, ch1_mode = MODE_ON;  // CH1
short ch1_off = 204;
byte rate = 3;                      // sampling rate
byte trig_mode = TRIG_AUTO, trig_lv = 30, trig_edge = TRIG_E_UP, trig_ch = 1; // trigger settings
byte Start = 1;  // Start sampling
byte menu = 0;  // Default menu
////////////////////////////////////////////////////////////////////////////////////////////////

void setup(){
  tft.reset();
  tft.initDisplay();
  tft.setRotation(3);

  tft.fillScreen(BGCOLOR);

  tft.setTextColor(WHITE);
  tft.setTextSize(1);
  tft.setCursor(75, 100);
  tft.print("Arduino Oscillo Scope");
  tft.setCursor(75, 120);
  tft.print("with Aitendo TFT LCD (320x240)");
  tft.setCursor(75, 140);
  tft.print("(c) 2009-2014 non");
  delay(2000);

  tft.fillScreen(BGCOLOR);

  Serial.begin(9600);
  DrawGrid();
  DrawText();
}

void CheckSW() {
  static unsigned short oain[2];
  static unsigned long Millis = 0, oMillis = 0;
  unsigned long ms;
  unsigned short ain = analogRead(ad_sw);

  return;

  ms = millis();
  if ((ms - Millis)<5)
    return;
  Millis = ms;

  if (!(abs(oain[0] - oain[1])>10 && abs(oain[1] - ain)<2)) {
    oain[0] = oain[1];
    oain[1] = ain;
    return;
  }
  oain[0] = oain[1];
  oain[1] = ain;

  if (ain > 950 || (Millis - oMillis)<200)
    return;
  oMillis = Millis;

  // Serial.println(ain);

  int sw;
  for (sw = 0; sw < 10; sw ++) {
    const int sw_lv[] = {889, 800, 700, 611, 514, 419, 338, 231, 132, 70};
    if (ain > sw_lv[sw])
      break;
  }
```

```
// Serial.println(sw);

  switch (menu) {
  case 0:
  default:
    menu0_sw(sw);
    break;
  case 1:
    menu1_sw(sw);
    break;
  case 2:
    menu2_sw(sw);
    break;
  }

  DrawText();
}

void menu0_sw(int sw) {
  switch (sw) {
  case 0:
    // START/HOLD
    if (Start)
      Start = 0;
    else
      Start = 1;
    break;
  case 1:
    // CH0 RANGE -
    if (range0 < RANGE_MAX)
      range0 ++;
    break;
  case 2:
    // CH1 RANGE -
    if (range1 < RANGE_MAX)
      range1 ++;
    break;
  case 3:
    // RATE FAST
    if (rate > 0)
      rate --;
    break;
  case 4:
    // TRIG MODE
    if (trig_mode < TRIG_ONE)
      trig_mode ++;
    else
      trig_mode = 0;
    break;
  case 5:
    // SEND
    SendData();
    break;
  case 6:
    // TRIG MODE
    if (trig_mode > 0)
      trig_mode --;
    else
      trig_mode = TRIG_ONE;
    break;
  case 7:
    // RATE SLOW
    if (rate < RATE_MAX)
      rate ++;
    break;
  case 8:
    // CH1 RANGE +
    if (range1 > 0)
      range1 --;
    break;
  case 9:
    // CH0 RANGE +
    if (range0 > 0)
      range0 --;
    break;
  case 10:
  default:
    // MENU SW
    menu ++;
    break;
  }
}

void menu1_sw(int sw) {
  switch (sw) {
  case 0:
    // START/HOLD
    if (Start)
      Start = 0;
    else
      Start = 1;
    break;
  case 1:
    // CH0 offset +
    if (ch0_off < 1023)
      ch0_off += 1024/VREF[range0];
    break;
  case 2:
    // CH1 offset +
    if (ch1_off < 1023)
      ch1_off += 1024/VREF[range1];
    break;
  case 3:
    // trigger level +
    if (trig_lv < 60)
      trig_lv ++;
    break;
  case 4:
  case 6:
    // TRIG EDGE
    if (trig_edge == TRIG_E_UP)
      trig_edge = TRIG_E_DN;
    else
      trig_edge = TRIG_E_UP;
    break;
  case 5:
    // SEND
    SendData();
    break;
  case 7:
    // trigger level -
    if (trig_lv > 0)
      trig_lv --;
    break;
  case 8:
    // CH1 OFF -
    if (ch1_off > -1023)
      ch1_off -= 1024/VREF[range1];
    break;
  case 9:
    // CH0 OFF -
    if (ch0_off > -1023)
      ch0_off -= 1024/VREF[range0];
    break;
  case 10:
  default
    // MENU SW
    menu ++;
    break;
  }
}

void menu2_sw(int sw) {
  switch (sw) {
  case 0:
    // START/HOLD
    if (Start)
      Start = 0;
    else
      Start = 1;
    break;
  case 1:
    if (ch0_mode < 2)
      ch0_mode ++;
    break;
  case 2:
    if (ch1_mode < 2)
      ch1_mode ++;
    break;
  case 3:
  case 7:
    // TRIG channel
    if (trig_ch == 0)
      trig_ch = 1;
    else
      trig_ch = 0;
    break;
  case 5:
    // SEND
    SendData();
    break;
  case 8:
    if (ch1_mode > 0)
      ch1_mode --;
    break;
  case 9:
    if (ch0_mode > 0)
      ch0_mode --;
    break;
  case 10:
    // MENU SW
    menu = 0;
    break;
  case 4:
  case 6:
  default:
    // none
    break;
  }
}

void SendData() {
  Serial.print(Rates[rate]);
  Serial.println("/div (30 samples)");
  for (int i=0; i<SAMPLES; i ++) {
    Serial.print(data[sample + 0][i]*MILLIVOL_per_dot[range0]);
    Serial.print(" ");
    Serial.println(data[sample + 1][i]*MILLIVOL_per_dot[range1]);
  }
}

void DrawGrid() {
  for (int x=0; x<=SAMPLES; x += 2) { // Horizontal Line
    for (int y=0; y<=LCD_HEIGHT; y += DOTS_DIV) {
      tft.drawPixel(x, y, GRIDCOLOR);
      CheckSW();
    }
    if (LCD_HEIGHT == 240)
      tft.drawPixel(x, LCD_HEIGHT-1, GRIDCOLOR);
  }
  for (int x=0; x<=SAMPLES; x += DOTS_DIV ) { // Vertical Line
    for (int y=0; y<=LCD_HEIGHT; y += 2) {
      tft.drawPixel(x, y, GRIDCOLOR);
      CheckSW();
    }
  }
}

void DrawText() {
  tft.setTextColor(WHITE);
  tft.setTextSize(1);
  tft.setCursor(SAMPLES+3, 20);
  tft.print(Ranges[range1]);
  tft.println("/DIV");
  tft.setCursor(SAMPLES+3, 30);
  tft.print(Rates[rate]);
  tft.println("/DIV");
  tft.setCursor(SAMPLES+3, 40);
  tft.println(TRIG_Modes[trig_mode]);
  tft.setCursor(SAMPLES+3, 50);
  tft.println(trig_edge == TRIG_E_UP ? "UP" : "DN");
  tft.setCursor(SAMPLES+3, 60);
  tft.println(Modes[ch1_mode]);
//  tft.setCursor(SAMPLES, 70);
//  tft.println(trig_ch == 0 ? "T:1" : "T:2");
#if 0
  GLCD.FillRect(101,txtLINE0,28,64, WHITE);  // clear text area that will be drawn below
```

```
switch (menu) {
    case 0:
      GLCD.GotoXY(SAMPLES + 1,txtLINE0);       // locate curser for printing text
      GLCD.Puts(Ranges[range0]);
      GLCD.GotoXY(SAMPLES + 1,txtLINE1);       // locate curser for printing text
      GLCD.Puts(Ranges[range1]);
      GLCD.GotoXY(SAMPLES + 1,txtLINE2);       // locate curser for printing text
      GLCD.Puts(Rates[rate]);
      GLCD.GotoXY(SAMPLES + 1,txtLINE3);       // locate curser for printing text
      GLCD.Puts(TRIG_Modes[trig_mode]);
      break;
    case 1:
      GLCD.GotoXY(SAMPLES + 1,txtLINE0);       // locate curser for printing text
      GLCD.Puts("OF1");
      GLCD.GotoXY(SAMPLES + 1,txtLINE1);       // locate curser for printing text
      GLCD.Puts("OF2");
      GLCD.GotoXY(SAMPLES + 1,txtLINE2);       // locate curser for printing text
      GLCD.Puts("Tlv");
      GLCD.GotoXY(SAMPLES + 1,txtLINE3);       // locate curser for printing text
      GLCD.Puts(trig_edge == TRIG_E_UP ? "UP" : "DN");
      break;
    case 2:
      GLCD.GotoXY(SAMPLES + 1,txtLINE0);       // locate curser for printing text
      GLCD.Puts(Modes[ch0_mode]);
      GLCD.GotoXY(SAMPLES + 1,txtLINE1);       // locate curser for printing text
      GLCD.Puts(Modes[ch1_mode]);
      GLCD.GotoXY(SAMPLES + 1,txtLINE2);       // locate curser for printing text
      GLCD.Puts(trig_ch == 0 ? "T:1" : "T:2");
      break;
   }
#endif
}

void DrawGrid(int x) {
   if ((x % 2) == 0)
     for (int y=0; y<=LCD_HEIGHT;  y += DOTS_DIV)
       tft.drawPixel(x, y, GRIDCOLOR);
   if ((x % DOTS_DIV) == 0)
     for (int y=0; y<=LCD_HEIGHT;  y += 2)
       tft.drawPixel(x, y, GRIDCOLOR);
}

void ClearAndDrawGraph() {
  int clear = 0;

  if (sample == 0)
    clear = 2;
#if 0
  for (int x=0; x<SAMPLES; x++) {
    GLCD.SetDot(x, LCD_HEIGHT-data[clear+0][x], WHITE);
    GLCD.SetDot(x, LCD_HEIGHT-data[clear+1][x], WHITE);
    GLCD.SetDot(x, LCD_HEIGHT-data[sample+0][x], BLACK);
    GLCD.SetDot(x, LCD_HEIGHT-data[sample+1][x], BLACK);
  }
#else
  for (int x=0; x<(SAMPLES-1); x++) {
    tft.drawLine(x, LCD_HEIGHT-data[clear+0][x], x+1, LCD_HEIGHT-data[clear+0][x+1], BGCOLOR);
    tft.drawLine(x, LCD_HEIGHT-data[clear+1][x], x+1, LCD_HEIGHT-data[clear+1][x+1], BGCOLOR);
    if (ch0_mode != MODE_OFF)
      tft.drawLine(x, LCD_HEIGHT-data[sample+0][x], x+1, LCD_HEIGHT-data[sample+0][x+1], CH1COLOR);
    if (ch1_mode != MODE_OFF)
      tft.drawLine(x, LCD_HEIGHT-data[sample+1][x], x+1, LCD_HEIGHT-data[sample+1][x+1], CH2COLOR);
    CheckSW();
  }
#endif
}

void ClearAndDrawDot(int i) {
  int clear = 0;

  if (i <= 1)
    return;
  if (sample == 0)
    clear = 2;
#if 0
  for (int x=0; x<SAMPLES; x++) {
    GLCD.SetDot(x, LCD_HEIGHT-data[clear+0][x], WHITE);
    GLCD.SetDot(x, LCD_HEIGHT-data[clear+1][x], WHITE);
    GLCD.SetDot(x, LCD_HEIGHT-data[sample+0][x], BLACK);
    GLCD.SetDot(x, LCD_HEIGHT-data[sample+1][x], BLACK);
  }
#else
  tft.drawLine(i-1, LCD_HEIGHT-data[clear+0][i-1], i, LCD_HEIGHT-data[clear+0][i], BGCOLOR);
  tft.drawLine(i-1, LCD_HEIGHT-data[clear+1][i-1], i, LCD_HEIGHT-data[clear+1][i], BGCOLOR);
  if (ch0_mode != MODE_OFF)
    tft.drawLine(i-1, LCD_HEIGHT-data[sample+0][i-1], i, LCD_HEIGHT-data[sample+0][i], CH1COLOR);
  if (ch1_mode != MODE_OFF)
    tft.drawLine(i-1, LCD_HEIGHT-data[sample+1][i-1], i, LCD_HEIGHT-data[sample+1][i], CH2COLOR);
#endif
  DrawGrid(i);
}

void DrawGraph() {
  for (int x=0; x<SAMPLES; x++) {
    tft.drawPixel(x, LCD_HEIGHT-data[sample+0][x], CH1COLOR);
    tft.drawPixel(x, LCD_HEIGHT-data[sample+1][x], CH2COLOR);
  }
}

void ClearGraph() {
  int clear = 0;

  if (sample == 0)
    clear = 2;
  for (int x=0; x<SAMPLES; x++) {
    tft.drawPixel(x, LCD_HEIGHT-data[clear+0][x], BGCOLOR);
    tft.drawPixel(x, LCD_HEIGHT-data[clear+1][x], BGCOLOR);
  }
}

inline unsigned long adRead(byte ch, byte mode, int off)
{
  unsigned long a = analogRead(ch);
  a = ((a+off)*VREF[ch == ad_ch0 ? range0 : range1]+512) >> 10;
  a = a>=(LCD_HEIGHT+1)  ? LCD_HEIGHT : a;
  if (mode == MODE_INV)
    return LCD_HEIGHT - a;
  return a;
}

void  loop() {
  if (trig_mode != TRIG_SCAN) {
     unsigned long st = millis();
     byte oad;
     if (trig_ch == 0)
       oad = adRead(ad_ch0, ch0_mode, ch0_off);
     else
       oad = adRead(ad_ch1, ch1_mode, ch1_off);
     for (;;) {
       byte ad;
       if (trig_ch == 0)
         ad = adRead(ad_ch0, ch0_mode, ch0_off);
       else
         ad = adRead(ad_ch1, ch1_mode, ch1_off);

       if (trig_edge == TRIG_E_UP) {
         if (ad >= trig_lv && ad > oad)
           break;
       } else {
         if (ad <= trig_lv && ad < oad)
           break;
       }
       oad = ad;

       CheckSW();
       if (trig_mode == TRIG_SCAN)
         break;
       if (trig_mode == TRIG_AUTO && (millis() - st) > 100)
         break;
     }
  }

  // sample and draw depending on the sampling rate
   if (rate <= 5 && Start) {
   // change the index for the double buffer
   if (sample == 0)
     sample = 2;
   else
     sample = 0;

   if (rate == 0) { // full speed, channel 0 only
     unsigned long st = millis();
     for (int i=0; i<SAMPLES; i ++) {
       data[sample+0][i] = adRead(ad_ch0, ch0_mode, ch0_off);
     }
```

```
  for (int i=0; i<SAMPLES; i ++)
    data[sample+0][i] = 0;
  // Serial.println(millis()-st);
 } else if (rate == 2) { // full speed, dual channel
  unsigned long st = millis();
  for (int i=0; i<SAMPLES; i ++) {
    data[sample+0][i] = adRead(ad_ch0, ch0_mode, ch0_off);
    data[sample+1][i] = adRead(ad_ch1, ch1_mode, ch1_off);
  }
  // Serial.println(millis()-st);
 } else if (rate >= 3 && rate <= 5) { // .5ms, 1ms or 2ms sampling
  const unsigned long r_[] = {5000/DOTS_DIV, 10000/DOTS_DIV, 20000/DOTS_DIV};
  unsigned long st0 = millis();
  unsigned long st = micros();
  unsigned long r = r_[rate - 3];
  for (int i=0; i<SAMPLES; i ++) {
    while((st - micros())<r) ;
    st += r;
    data[sample+0][i] = adRead(ad_ch0, ch0_mode, ch0_off);
    data[sample+1][i] = adRead(ad_ch1, ch1_mode, ch1_off);
  }
  // Serial.println(millis()-st0);
 }
 ClearAndDrawGraph();
 CheckSW();
 DrawGrid();
 DrawText();
} else if (Start) { // 5ms - 500ms sampling
// copy currently showing data to another
 if (sample == 0) {
  for (int i=0; i<SAMPLES; i ++) {
    data[2][i] = data[0][i];
    data[3][i] = data[1][i];
  }
 } else {
  for (int i=0; i<SAMPLES; i ++) {
    data[0][i] = data[2][i];
    data[1][i] = data[3][i];
  }
 }

 const unsigned long r_[] = {50000/DOTS_DIV, 100000/DOTS_DIV, 200000/DOTS_DIV,
              500000/DOTS_DIV, 1000000/DOTS_DIV, 2000000/DOTS_DIV,
              5000000/DOTS_DIV, 10000000/DOTS_DIV};
 unsigned long st0 = millis();
 unsigned long st = micros();
 for (int i=0; i<SAMPLES; i ++) {
  while((st - micros())<r_[rate-6]) {
    CheckSW();
    if (rate<6)
      break;
  }
  if (rate<6) { // sampling rate has been changed
    tft.fillScreen(BGCOLOR);
    break;
  }
  st += r_[rate-6];
  if (st - micros()>r_[rate-6])
      st = micros(); // sampling rate has been changed to shorter interval
  if (!Start) {
    i --;
    continue;
  }
  data[sample+0][i] = adRead(ad_ch0, ch0_mode, ch0_off);
  data[sample+1][i] = adRead(ad_ch1, ch1_mode, ch1_off);
  ClearAndDrawDot(i);
 }
 // Serial.println(millis()-st0);
 DrawGrid();
 DrawText();
} else {
 CheckSW();
}
if (trig_mode == TRIG_ONE)
 Start = 0;
}
```