```
//
// 2.4インチ タッチパネル付き液晶を使ったフォトフレーム
//   画像は320x240の24bitBMPファイルでマイクロSDカードに入れておく
//   画像はフォルダ単位で読み出すので、種類別にフォルダに入れる
//
//    Ver-1.0 2016/3/28 Susumu Shikata
//       "tftbmp"をベースに作成
//    Ver-2.0 2016/3/29
//       温湿度センサ(DHT-11)を追加
//

#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <SD.h>
#include <SPI.h>
#include <TouchScreen.h>.
#include "DHT.h"

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
//
// SDカードリーダ
//    CS   --> digital pin 10
//    MOSI --> digital pin 11
//    MISO --> digital pin 12
//    SCK  --> digital pin 13
//
//  Humididty sensor(DHT11) definitions
#define DHTPIN 19        // what digital pin we're connected to DHT11
#define DHTTYPE DHT11    // DHT 11

#define YP A3  // must be an analog pin, use "An" notation!
```

```
#define XM A2  // must be an analog pin, use "An" notation!
#define YM 9   // can be a digital pin
#define XP 8   // can be a digital pin

// タッチパネルの座標を指定する
// 但し供給電圧などで微妙に位置がずれるので注意
#define TS_MINX 150
#define TS_MINY 160
#define TS_MAXX 920
#define TS_MAXY 925

// For better pressure precision, we need to know the resistance
// between X+ and X- Use any multimeter to read it
// For the one we're using, its 300 ohms across the X plate
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

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

// 液晶表示のカラーを定義
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

// ボタンのサイズとペンの幅のデフォルト値
#define BOXSIZE 40
#define PENRADIUS 3

// SDカードリーダのCSに対するピンを定義
#define SD_CS 10

// our TFT wiring
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, A4);

int  dir = 0;
int  draw = 0;

char c = 0x00;
float tmp;            //  温度
float hum;             //  湿度
int array_time[ 3 ];  //  時刻設定用配列  時・分・秒
int mode = 0;          //  動作モード 0:通常時計表示／1:設定モード

DHT dht(DHTPIN, DHTTYPE);                //  温湿度センサにDHT11を使用

void setup()
{
  Serial.begin(9600);
  dht.begin();

  // 液晶の初期化
  //  今回使用したTFT液晶のドライバIDは「0x9325」だった
  tft.reset();
  uint16_t identifier = tft.readID();
  //Serial.println(identifier, HEX);
  tft.begin(identifier);
  SD.begin(SD_CS);

  tft.setRotation(1);
  DrawDHT("0.bmp", 0, 0);
  //pinMode(13, OUTPUT);
}

#define MINPRESSURE 10
#define MAXPRESSURE 1000

void loop()
{
  char fn[16];
  int  i;

  // ボタンを表示して、タッチ待ちになる
  tft.setRotation(1);
  tft.setTextColor(WHITE);  tft.setTextSize(3);
  tft.setCursor(120, 10); tft.print( F("ISHIGAKI") );
  tft.setCursor(120, 50); tft.print( F("CAT's") );
  tft.setRotation(2);     //
なぜか液晶とタッチパネルの座標が180度ずれていたので、回転して補正
  tft.fillRect(0, 0, BOXSIZE, BOXSIZE, BLUE);
  tft.fillRect(BOXSIZE, 0, BOXSIZE, BOXSIZE, CYAN);

  // タッチパネル入力の座標を取り込む
  TSPoint p = ts.getPoint();

  // if sharing pins, you'll need to fix the directions of the touchscreen pins
  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);
  // タッチパネルの入力をチェックする
  if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
    // scale from 0->1023 to tft.width
    p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
    p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);

    if (p.y < BOXSIZE) {
      if (p.x < BOXSIZE) {
        dir = 0;
        draw = 1;
      } else if (p.x < BOXSIZE * 2) {
        dir = 1;
        draw = 1;
      }
    }
  }

  // 画像を表示
  if ( draw == 1 ) {       // 約5秒毎に画像を更新する
    tft.fillScreen(0);
    for ( i = 1; i <= 350; i++) {   // 1つのフォルダ内の画像は350まで
      // 取りあえずフォルダ名は固定(2つだけ)
      if ( dir == 0 ) sprintf(fn, "ishigaki/%d.bmp", i);
      if ( dir == 1 ) sprintf(fn, "cat/%d.bmp", i);
      if ( DrawDHT(fn, 0, 0) ) {
        for (int j = 0; j <= 50; j++ ) {
          delay(100);
          TSPoint p = ts.getPoint();
          pinMode(XM, OUTPUT);
          pinMode(YP, OUTPUT);
          if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
            i = 350;
            draw = 0;
          }
        }
      }
      Serial.println(i);Serial.println(fn);
    }
  }
}

boolean DrawDHT(char *filename, int x, int y) {
  char StrBuf[2];

  boolean resp = bmpDraw(filename, x, y);   // BMP画像の表示

  if ( resp ) {
    tft.setRotation(1);
    tft.setTextColor(YELLOW);  tft.setTextSize(5);

    // 温度･湿度を表示
    tmp = dht.readTemperature();  // DHT11から温度を読む
    hum = dht.readHumidity();     // DHT11から湿度を読む
    tft.setCursor(100, 200);
    //  温度
    sprintf(StrBuf, "%02d", int(tmp));
    tft.print( StrBuf );
    tft.setTextSize(2); tft.print("'C");
    tft.print( "   " );

    //  湿度
    tft.setTextSize(5);
    sprintf(StrBuf, "%02d", int(hum));
    tft.print( StrBuf );
    tft.setTextSize(2); tft.print('%');
  }
  return resp;
}

//
// 画像をSDカードから読み出して表示する
//  引数：ファイル名、表示開始座標(x,y)
//  戻り値が"false"の場合はエラー(ファイルが無いとかbmpでは無い時など)
//
#define BUFFPIXEL 20

boolean bmpDraw(char *filename, int x, int y) {

  File     bmpFile;
  int      bmpWidth, bmpHeight;   // W+H in pixels
  uint8_t  bmpDepth;              // Bit depth (currently must be 24)
  uint32_t bmpImageoffset;        // Start of image data in file
  uint32_t rowSize;               // Not always = bmpWidth; may have padding
  uint8_t  sdbuffer[3 * BUFFPIXEL]; // pixel in buffer (R+G+B per pixel)
  uint16_t lcdbuffer[BUFFPIXEL];  // pixel out buffer (16-bit per pixel)
  uint8_t  buffidx = sizeof(sdbuffer); // Current position in sdbuffer
  boolean  goodBmp = false;       // Set to true on valid header parse
  boolean  flip    = true;        // BMP is stored bottom-to-top
  int      w, h, row, col;
  uint8_t  r, g, b;
  uint32_t pos = 0, startTime = millis();
  uint8_t  lcdidx = 0;
  boolean  first = true;

  if ((x >= tft.width()) || (y >= tft.height())) return goodBmp;
```

```
/*
  Serial.println();
  Serial.print(F("Loading '"));
  Serial.print(filename);
  Serial.println('\'');
*/
// Open requested file on SD card
if ((bmpFile = SD.open(filename)) == NULL) {
  Serial.println(F("Not found"));
  return goodBmp;
}

// Parse BMP header
if (read16(bmpFile) == 0x4D42) { // BMP signature
  (void)read32(bmpFile);
  //Serial.println(F("File size: ")); Serial.println(read32(bmpFile));
  (void)read32(bmpFile); // Read & ignore creator bytes
  bmpImageoffset = read32(bmpFile); // Start of image data
  //Serial.print(F("Image Offset: ")); Serial.println(bmpImageoffset, DEC);
  // Read DIB header
  (void)read32(bmpFile);
  //Serial.print(F("Header size: ")); Serial.println(read32(bmpFile));
  bmpWidth  = read32(bmpFile);
  bmpHeight = read32(bmpFile);
  if (read16(bmpFile) == 1) { // # planes -- must be '1'
    bmpDepth = read16(bmpFile); // bits per pixel
    //Serial.print(F("Bit Depth: ")); Serial.println(bmpDepth);
    if ((bmpDepth == 24) && (read32(bmpFile) == 0)) { // 0 = uncompressed

      goodBmp = true; // Supported BMP format -- proceed!

      if ( bmpWidth == 240 ) tft.setRotation(0);  // Portlait
      else tft.setRotation(1);                    // Landscape


      // BMP rows are padded (if needed) to 4-byte boundary
      rowSize = (bmpWidth * 3 + 3) & ~3;

      // If bmpHeight is negative, image is in top-down order.
      // This is not canon but has been observed in the wild.
      if (bmpHeight < 0) {
        bmpHeight = -bmpHeight;
        flip      = false;
      }

      // Crop area to be loaded
      w = bmpWidth;
      h = bmpHeight;
      if ((x + w - 1) >= tft.width())  w = tft.width()  - x;
      if ((y + h - 1) >= tft.height()) h = tft.height() - y;

      // Set TFT address window to clipped image bounds
      tft.setAddrWindow(x, y, x + w - 1, y + h - 1);

      for (row = 0; row < h; row++) { // For each scanline...
        // Seek to start of scan line.  It might seem labor-
        // intensive to be doing this on every line, but this
        // method covers a lot of gritty details like cropping
        // and scanline padding.  Also, the seek only takes
        // place if the file position actually needs to change
        // (avoids a lot of cluster math in SD library).
        if (flip) // Bitmap is stored bottom-to-top order (normal BMP)
          pos = bmpImageoffset + (bmpHeight - 1 - row) * rowSize;
        else     // Bitmap is stored top-to-bottom
          pos = bmpImageoffset + row * rowSize;
        if (bmpFile.position() != pos) { // Need seek?
          bmpFile.seek(pos);
          buffidx = sizeof(sdbuffer); // Force buffer reload
        }

        for (col = 0; col < w; col++) { // For each column...
          // Time to read more pixel data?
          if (buffidx >= sizeof(sdbuffer)) { // Indeed
            // Push LCD buffer to the display first
            if (lcdidx > 0) {
              tft.pushColors(lcdbuffer, lcdidx, first);
              lcdidx = 0;
              first  = false;
            }
            bmpFile.read(sdbuffer, sizeof(sdbuffer));
            buffidx = 0; // Set index to beginning
          }

          // Convert pixel from BMP to TFT format
          b = sdbuffer[buffidx++];
          g = sdbuffer[buffidx++];
          r = sdbuffer[buffidx++];
          lcdbuffer[lcdidx++] = tft.color565(r, g, b);
        } // end pixel
      } // end scanline
      // Write any remaining data to LCD
      if (lcdidx > 0) {
        tft.pushColors(lcdbuffer, lcdidx, first);
      }
    } // end goodBmp
  }
```

```
  }

  bmpFile.close();
  //if (!goodBmp) Serial.println(F("BMP format not recognized."));
  return goodBmp;
}

// These read 16- and 32-bit types from the SD card file.
// BMP data is stored little-endian, Arduino is little-endian too.
// May need to reverse subscript order if porting elsewhere.

uint16_t read16(File f) {
  uint16_t result;
  ((uint8_t *)&result)[0] = f.read(); // LSB
  ((uint8_t *)&result)[1] = f.read(); // MSB
  return result;
}

uint32_t read32(File f) {
  uint32_t result;
  ((uint8_t *)&result)[0] = f.read(); // LSB
  ((uint8_t *)&result)[1] = f.read();
  ((uint8_t *)&result)[2] = f.read();
  ((uint8_t *)&result)[3] = f.read(); // MSB
  return result;
}
```