/\*

TFT Touch Shield 2.0

BetterMed

Matthew Horger

\*/

#include <SD.h> // our libraries

#include <SPI.h>

#include "TFTv2.h"

#define MAX\_BMP 10 // bmp file num

#define FILENAME\_LEN 20 // max file name length

const int PIN\_SD\_CS = 4; // pin of sd card

const int \_\_Gnbmp\_height = 320; // bmp hight

const int \_\_Gnbmp\_width = 240; // bmp width

unsigned char \_\_Gnbmp\_image\_offset = 0; // offset

int \_\_Gnfile\_num = 0; // num of file in array

char \_\_Gsbmp\_files[1][FILENAME\_LEN] =

{

"BetterMed.BMP",

};

File bmpFile; // our actual file

void setup()

{

Serial.begin(9600);

pinMode(PIN\_SD\_CS,OUTPUT);

digitalWrite(PIN\_SD\_CS,HIGH);

Tft.TFTinit();

Sd2Card card;

card.init(SPI\_FULL\_SPEED, PIN\_SD\_CS);

if(!SD.begin(PIN\_SD\_CS))

{

Serial.println("failed!");

while(1); // init fail, die here

}

Serial.println("SD OK!");

TFT\_BL\_ON;

}

void loop()

{

for(unsigned char i=0; i<\_\_Gnfile\_num; i++)

{

bmpFile = SD.open(\_\_Gsbmp\_files[i]); // open our files

if (! bmpFile) // if we couldn't find the image

{

Serial.println("didnt find image");

while (1);

}

if(! bmpReadHeader(bmpFile)) // if the file extension is bad

{

Serial.println("bad bmp");

return;

}

bmpdraw(bmpFile, 0, 0);

bmpFile.close();

delay(1000);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// This procedure reads a bitmap and draws it to the screen

// its sped up by reading many pixels worth of data at a time

// instead of just one pixel at a time. increading the buffer takes

// more RAM but makes the drawing a little faster. 20 pixels' worth

// is probably a good place

#define BUFFPIXEL 60 // must be a divisor of 240

#define BUFFPIXEL\_X3 180 // BUFFPIXELx3

void bmpdraw(File f, int x, int y)

{

bmpFile.seek(\_\_Gnbmp\_image\_offset);

uint32\_t time = millis();

uint8\_t sdbuffer[BUFFPIXEL\_X3]; // 3 \* pixels to buffer

for (int i=0; i< \_\_Gnbmp\_height; i++)

{

for(int j=0; j<(240/BUFFPIXEL); j++)

{

bmpFile.read(sdbuffer, BUFFPIXEL\_X3);

uint8\_t buffidx = 0;

int offset\_x = j\*BUFFPIXEL;

unsigned int \_\_color[BUFFPIXEL];

for(int k=0; k<BUFFPIXEL; k++)

{

\_\_color[k] = sdbuffer[buffidx+2]>>3; // read

\_\_color[k] = \_\_color[k]<<6 | (sdbuffer[buffidx+1]>>2); // green

\_\_color[k] = \_\_color[k]<<5 | (sdbuffer[buffidx+0]>>3); // blue

buffidx += 3;

}

Tft.setCol(offset\_x, offset\_x+BUFFPIXEL);

Tft.setPage(i, i);

Tft.sendCMD(0x2c);

TFT\_DC\_HIGH;

TFT\_CS\_LOW;

for(int m=0; m < BUFFPIXEL; m++)

{

SPI.transfer(\_\_color[m]>>8);

SPI.transfer(\_\_color[m]);

}

TFT\_CS\_HIGH;

}

}

Serial.print(millis() - time, DEC);

Serial.println(" ms");

}

boolean bmpReadHeader(File f)

{

// read header

uint32\_t tmp;

uint8\_t bmpDepth;

if (read16(f) != 0x4D42) {

// magic bytes missing

return false;

}

// read file size

tmp = read32(f);

Serial.print("size 0x");

Serial.println(tmp, HEX);

// read and ignore creator bytes

read32(f);

\_\_Gnbmp\_image\_offset = read32(f);

Serial.print("offset ");

Serial.println(\_\_Gnbmp\_image\_offset, DEC);

// read DIB header

tmp = read32(f);

Serial.print("header size ");

Serial.println(tmp, DEC);

int bmp\_width = read32(f);

int bmp\_height = read32(f);

if(bmp\_width != \_\_Gnbmp\_width || bmp\_height != \_\_Gnbmp\_height) // if image is not 320x240, return false

{

return false;

}

if (read16(f) != 1)

return false;

bmpDepth = read16(f);

Serial.print("bitdepth ");

Serial.println(bmpDepth, DEC);

if (read32(f) != 0) {

// compression not supported!

return false;

}

Serial.print("compression ");

Serial.println(tmp, DEC);

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// These read data from the SD card file and convert them to big endian

// (the data is stored in little endian format!)

// LITTLE ENDIAN!

uint16\_t read16(File f)

{

uint16\_t d;

uint8\_t b;

b = f.read();

d = f.read();

d <<= 8;

d |= b;

return d;

}

// LITTLE ENDIAN!

uint32\_t read32(File f)

{

uint32\_t d;

uint16\_t b;

b = read16(f);

d = read16(f);

d <<= 16;

d |= b;

return d;

}