//#include <FRAM\_MB85RC\_I2C.h>

// ToyotaOBD1\_Reader

// In order to read the data from the OBD connector, short E1 + TE2. then to read the data connect to VF1.

// Note the data line output is 12V - connecting it directly to one of the arduino pins might damage (proabably) the board

// This is made for diaply with an OLED display using the U8glib - which allow wide range of display types with minor adjusments.

// Many thanks to GadgetFreak for the greate base code for the reasding of the data.

// If you want to use invert line - note the comments on the MY\_HIGH and the INPUT\_PULLUP in the SETUP void.

#include "SPI.h"

#include "Adafruit\_GFX.h"

#include "Adafruit\_ILI9341\_STM.h"

#include "Orbitron9pt7b.h"

int charWd = 10;

int charHt = 16;

int charYoffs = 0;

#include <MD\_KeySwitch.h>

#define VREF\_MEASURED 3.32 //Измеренное опорное напряжение с 3.3В стабилизатора

#define TT\_PIN\_DIVIDER 3.13

// выбрать один вариант логирования

#define LOGGING\_MINIMAL //запись минимума данных

//#define LOGGING\_FULL //Запись на SD всех данных

//#define LOGGING\_DEBUG //Запись на SD всех данных + статистику расхода и пробега

//#define SECOND\_O2SENS // Включение 2го сенсора кислорода для V движков

#define DEBUG\_OUTPUT true // for debug option - swith output to Serial

//DEFINE пинов под входы-выходы

//#define LED\_PIN 33 //встроенный светодиод

//#define OX\_PIN 3 //D3-PB0 для сенсора кислорода

//#define TT\_PIN 8 //D8-PA3 для сенсора ТТ АКПП

#define ENGINE\_DATA\_PIN 18 //D18-PB5-EXTI5 VF1 PIN

//#define TOGGLE\_BTN\_PIN 18 // D4 screen change PIN

#define TFT\_CS 7

#define TFT\_DC 12

Adafruit\_ILI9341\_STM tft = Adafruit\_ILI9341\_STM(TFT\_CS, TFT\_DC); // Use hardware SPI

//DEFINE констант расходомера

#define Ls 0.004020653 //производительсность форсунки литров в секунду // базовый 0.004 или 240cc

#define Ncyl 6 //кол-во цилиндров

//DEFINE OBD READER

#define MY\_HIGH HIGH //LOW // I have inverted the Eng line using an Opto-Coupler, if yours isn't then reverse these low & high defines.

#define MY\_LOW LOW //HIGH

#define TOYOTA\_MAX\_BYTES 24

#define OBD\_INJ 1 //Injector pulse width (INJ)

#define OBD\_IGN 2 //Ignition timing angle (IGN)

#define OBD\_IAC 3 //Idle Air Control (IAC)

#define OBD\_RPM 4 //Engine speed (RPM)

#define OBD\_MAP 5 //Manifold Absolute Pressure (MAP)

#define OBD\_ECT 6 //Engine Coolant Temperature (ECT)

#define OBD\_TPS 7 // Throttle Position Sensor (TPS)

#define OBD\_SPD 8 //Speed (SPD)

#define OBD\_OXSENS 9 // Лямбда 1

#ifdef SECOND\_O2SENS

#define OBD\_OXSENS2 10 // Лямбда 2 на V-образных движка. У меня ее нету.

#endif

#define FONT\_X 18

#define FONT\_Y 18

//MD\_KeySwitch S(TOGGLE\_BTN\_PIN, HIGH);

byte CurrentDisplayIDX = 1, TT\_last = 0, TT\_curr = 0;

float total\_fuel\_consumption = 0, trip\_fuel\_consumption = 0;

float trip\_avg\_fuel\_consumption;

float cycle\_obd\_inj\_dur = 0;

float cycle\_trip = 0;

float trip\_inj\_dur = 0;

float total\_inj\_dur\_ee = 0;

float current\_trip = 0;

float total\_trip = 0;

float total\_avg\_consumption;

float total\_avg\_speed;

float trip\_avg\_speed;

unsigned long current\_time = 0;

unsigned long total\_time = 0;

unsigned long t;

unsigned long last\_log\_time = 0;

unsigned long odometer;

bool flagNulSpeed = true;

unsigned int OX, TT;

volatile uint8\_t ToyotaNumBytes, ToyotaID, ToyotaData[TOYOTA\_MAX\_BYTES];

volatile uint16\_t ToyotaFailBit = 0;

volatile float DATA[18];

boolean LoggingOn = false; // dfeine connection flag and last success packet - for lost connection function.

void setup() {

Serial.begin(9600);

tft.begin();

tft.setRotation(3);

tft.fillScreen(ILI9341\_BLACK);

//EEPROM.get(104, total\_trip);

//EEPROM.get(108, total\_time);

//EEPROM.get(200, odometer);

//EEPROM.get(204, total\_inj\_dur\_ee);

/\*

S.begin();

S.enableDoublePress(true);

S.enableLongPress(true);

S.enableRepeat(false);

S.enableRepeatResult(false);

S.setDoublePressTime(300);

S.setLongPressTime(2000);

\*/

//u8g.setFont(u8g\_font\_profont15r);

/\*if (DEBUG\_OUTPUT) {

Serial.println("system Started");

Serial.print("Read float from EEPROM: ");

Serial.println(total\_trip, 3);

Serial.println(total\_time, 3);

Serial.println(odometer, 1);

Serial.println(total\_inj\_dur\_ee, 3);

}\*/

// writeHeader();

pinMode(ENGINE\_DATA\_PIN, INPUT); // VF1 PIN

// pinMode(LED\_PIN, OUTPUT);

// pinMode(TT\_PIN, INPUT\_ANALOG);

// pinMode(OX\_PIN, INPUT\_ANALOG);

attachInterrupt(ENGINE\_DATA\_PIN, ChangeState, CHANGE); //setup Interrupt for data line

// pinMode(TOGGLE\_BTN\_PIN, INPUT); // кнопка СЛЕД. ЭКРАН

CurrentDisplayIDX = 1; // set to display 1

//drawScreenSelector();

//Расходомер

t = millis();

last\_log\_time = millis();

delay(1000);

} // END VOID SETUP

void loop(void) {

unsigned long new\_t;

unsigned int diff\_t;

/\* switch (S.read())

{

case MD\_KeySwitch::KS\_NULL: break;

case MD\_KeySwitch::KS\_PRESS: ent(); break;

case MD\_KeySwitch::KS\_DPRESS: {

if (LoggingOn == false) LoggingOn = true; else LoggingOn = false;

} break;

case MD\_KeySwitch::KS\_LONGPRESS: cleardata(); break;

case MD\_KeySwitch::KS\_RPTPRESS: break;

}

\*/

if (ToyotaNumBytes > 0) { // if found bytes

new\_t = millis();

if (new\_t > t && getOBDdata(OBD\_RPM) > 100 ) {// выполняем только когда на работающем двигателе

diff\_t = new\_t - t;

cycle\_obd\_inj\_dur = getOBDdata(OBD\_RPM) \* Ncyl \* (float)diff\_t \* getOBDdata(OBD\_INJ) / 120000; //Время открытых форсунок за 1 такт данных. В МС

//ОБ/М ОБ/С

//форсунка срабатывает раз в 2 оборота КВ

//6форсунок в с

//время цикла мс в с. Получаем кол-во срабатываний за время цикла. Умножаем на время открытия форсунки, получаем время открытия 6 форсунок В МИЛЛИСЕКУНДАХ

trip\_inj\_dur += cycle\_obd\_inj\_dur; //Время открытых форсунок за поездку В МС

total\_inj\_dur\_ee += cycle\_obd\_inj\_dur; //Время открытых форсунок за все время. EEPROM В МС

trip\_fuel\_consumption = trip\_inj\_dur \* Ls / 1000; //потребление топлива за поездку в литрах

total\_fuel\_consumption = total\_inj\_dur\_ee \* Ls / 1000; //потребление топлива за все время. Из ЕЕПРОМ в литрах

cycle\_trip = (float)diff\_t \* getOBDdata(OBD\_SPD) / 3600000; //расстояние пройденное за такт обд данных

current\_trip += cycle\_trip; //Пройденное расстояние с момента включения. В КМ

total\_trip += cycle\_trip; //Полное пройденное расстояние. EEPROM. В КМ

odometer += cycle\_trip; //электронный одометр. Хранится в еепром и не стирается кнопкой

current\_time += diff\_t; //Время в пути в миллисекундах с момента включения

total\_time += diff\_t; //полное пройденное время в миллисекундах лимит ~49 суток. EEPROM

trip\_avg\_speed = current\_trip \* 3600000 / (float)current\_time ; //средняя скорость за поездку

total\_avg\_speed = total\_trip \* 3600000 / (float)total\_time; // средняя скорость за все время. км\ч

trip\_avg\_fuel\_consumption = 100 \* trip\_fuel\_consumption / current\_trip; //средний расход за поездку

total\_avg\_consumption = 100 \* total\_fuel\_consumption / total\_trip; //среднее потребление за все время - Л на 100км

t = new\_t;//тест

// if (LoggingOn == true) logData(); //запись в лог данных по двоному нажатию на кнопку

//updateEepromData(); //запись данных при остановке

if (millis() - last\_log\_time > 180000) { //Запись данных в EEPROM каждые 3 минуты. Чтобы не потерять данные при движении на трассе

// rtc.eeprom\_write(104, total\_trip);

// rtc.eeprom\_write(108, total\_time);

// rtc.eeprom\_write(200, odometer);

// rtc.eeprom\_write(204, total\_inj\_dur\_ee);

last\_log\_time = millis();

}

}

//drawScreenSelector(); // draw screen

drawAllData();

ToyotaNumBytes = 0; // reset the counter.

} // end if (ToyotaNumBytes > 0)

if (millis() % 50 == 0 && LoggingOn == true && CurrentDisplayIDX == 6) { //каждые 50мс, когда включено логирование и выбран экран с флагами(!)

// OX = analogRead(OX\_PIN);

if (OX < 400) { //исключаю ложные показание > ~1.3В

// file.write(';');

// file.print(((float)OX \* VREF\_MEASURED) / 1024, 3 );

// file.println();

}

}

if (millis() % 500 == 0) { //каждые пол секунды читаем состояние АКПП

// TT = analogRead(TT\_PIN);

// TT\_curr = (int)(TT \* VREF\_MEASURED \* TT\_PIN\_DIVIDER / 1024 + 0.5);

// drawScreenSelector();

// if (TT\_last != TT\_curr) {

// drawScreenSelector();

// TT\_last = TT\_curr;

//}

//Serial.println((float)TT \* VREF\_MEASURED / 1024 \* 3.13, 3);

// Serial.println((int)(TT \* VREF\_MEASURED / 1024 \* 3.13+0.5));

}

}

// if (millis() % 5000 < 50) autoscreenchange(); // ротация экранов

void updateEepromData() {

if (getOBDdata(OBD\_SPD) == 0 && flagNulSpeed == false) { //Запись данных в еепром когда остановка авто

// EEPROM.put(104, total\_trip);

// EEPROM.put(108, total\_time);

// EEPROM.put(200, odometer);

//EEPROM.put(204, total\_inj\_dur\_ee);

flagNulSpeed = true; //запрет повторной записи

last\_log\_time = millis(); //чтобы не писать лишний раз

}

if (getOBDdata(OBD\_SPD) != 0) flagNulSpeed = false; //начали двигаться - разрешаем запись

}

void cleardata() {

int i;

for (i = 104; i <= 112; i++) {

// EEPROM.update(i, 0);

}

for (i = 204; i <= 208; i++) {

// EEPROM.update(i, 0);

}

// EEPROM.get(104, total\_trip);

// EEPROM.get(108, total\_time);

// EEPROM.get(204, total\_inj\_dur\_ee);

}

/\*

void writeHeader() {

#ifdef LOGGING\_FULL

file.print(F(";OX\_RAW;INJ TIME;IGN;IAC;RPM;MAP;ECT;TPS;SPD;VF1;OX;ASE;COLD;KNOCK;OPEN LOOP;Acceleration Enrichment;STARTER;IDLE;A/C;NEUTRAL;AVG SPD;LPK\_OBD;LPH\_OBD;TOTAL\_OBD;AVG\_OBD;CURR\_OBD;CURR\_RUN;total\_trip"));

#endif

#ifdef LOGGING\_DEBUG

file.print(F(";OX\_RAW;INJ TIME;IGN;IAC;RPM;MAP;ECT;TPS;SPD;VF1;OX;AVG SPD;LPK\_OBD;LPH\_OBD;TOTAL\_OBD;AVG\_OBD;CURR\_OBD;CURR\_RUN;total\_trip"));

#endif

#ifdef LOGGING\_MINIMAL

file.print(F(";OX\_RAW;INJ TIME;IGN;IAC;RPM;MAP;ECT;TPS;SPD;VF1;OX;LPH\_OBD;TT"));

#endif

file.println();

file.sync();

}

\*/

//обнуление данных

/\*

void logData() {

file.print(float(millis()) / 60000, 3); file.write(';') ; file.write(';');

file.print(getOBDdata(OBD\_INJ)); file.write(';'); file.print(getOBDdata(OBD\_IGN)); file.write(';'); file.print(getOBDdata(OBD\_IAC)); file.write(';');

file.print(getOBDdata(OBD\_RPM)); file.write(';'); file.print(getOBDdata(OBD\_MAP)); file.write(';'); file.print(getOBDdata(OBD\_ECT)); file.write(';');

file.print(getOBDdata(OBD\_TPS)); file.write(';'); file.print(getOBDdata(OBD\_SPD)); file.write(';'); file.print(getOBDdata(OBD\_OXSENS)); file.write(';'); file.print(getOBDdata(20)); file.write(';');

#ifdef LOGGING\_FULL

file.print(getOBDdata(11)); file.write(';'); file.print(getOBDdata(12)); file.write(';'); file.print(getOBDdata(13)); file.write(';'); file.print(getOBDdata(14)); file.write(';');

file.print(getOBDdata(15)); file.write(';'); file.print(getOBDdata(16)); file.write(';'); file.print(getOBDdata(17)); file.write(';'); file.print(getOBDdata(18)); file.write(';');

file.print(getOBDdata(19)); file.write(';');

#endif

#ifdef LOGGING\_DEBUG

file.print(total\_avg\_speed); file.write(';'); //AVG\_SPD ok

file.print(100 / getOBDdata(OBD\_SPD) \* (getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM)\*Ls \* 0.18)); file.write(';'); //LPK\_OBD ok

#endif

file.print(getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM)\*Ls \* 0.18); file.write(';'); //LPH\_OBD ok

file.print((float)TT \* VREF\_MEASURED \* TT\_PIN\_DIVIDER / 1024, 2); file.write(';'); //ТТ пин напряжение

#ifdef LOGGING\_DEBUG

file.print(total\_fuel\_consumption); file.write(';'); //TOTAL\_OBD ok

file.print(trip\_avg\_fuel\_consumption); file.write(';'); //!AVG\_OBD

file.print(trip\_fuel\_consumption); file.write(';'); //!CURR\_OBD

file.print(current\_trip); file.write(';'); //CURR\_RUN ok

file.print(total\_trip); file.write(';');//RUN\_TOTAL ok

#endif

file.println();

file.sync();

}

\*/

/\*

void drawScreenSelector(void) {

if (CurrentDisplayIDX == 1) DrawCurrentFuelConsuption();

else if (CurrentDisplayIDX == 2) DrawTotalFuelConsuption();

else if (CurrentDisplayIDX == 3) drawTimeDistance();

else if (CurrentDisplayIDX == 4) drawTripTimeDistance();

else if (CurrentDisplayIDX == 5) drawAllData();

else if (CurrentDisplayIDX == 6) drawExtraData();

} // end drawScreenSelector()

\*/

/\*

void DrawCurrentFuelConsuption(void) {

u8g.setFont(u8g\_font\_profont15r);

u8g.firstPage();

do {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 15, "TRIP" );

u8g.drawStr( 74, 15, "L" );

u8g.setPrintPos(35, 15) ;

u8g.print(trip\_fuel\_consumption, 1);

if (LoggingOn == true) u8g.drawHLine(20, 24, 88);

u8g.setPrintPos(90, 20) ;

u8g.print((float)TT \* VREF\_MEASURED \* TT\_PIN\_DIVIDER / 1024, 2);

u8g.setFont(u8g\_font\_profont22r);

switch (TT\_curr) { //для делителя 10k + 4.7k

case 0: u8g.drawStr( 95, 20, "1" ); break;

case 2: u8g.drawStr( 95, 20, "2" ); break;

case 4: u8g.drawStr( 95, 20, "3" ); break;

case 5: u8g.drawStr( 95, 20, "3L" ); break;

case 6: u8g.drawStr( 95, 20, "4" ); break;

case 7: u8g.drawStr( 95, 20, "4L" ); break;

}

if (getOBDdata(OBD\_SPD) > 1)

{

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "L/100Km" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print( 100 / getOBDdata(OBD\_SPD) \* (getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM)\*Ls \* 0.18), 1);

} else {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "L/Hour" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print(getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM)\*Ls \* 0.18, 1);

}

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 60, 42, "Average" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(60, 60) ;

if (trip\_avg\_fuel\_consumption < 100)

u8g.print( trip\_avg\_fuel\_consumption, 1);

else u8g.drawStr( 60, 60, "---" );

}

while ( u8g.nextPage() );

}

\*/

/\*

void DrawTotalFuelConsuption(void) {

u8g.setFont(u8g\_font\_profont15r);

u8g.firstPage();

do {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 15, "TOTAL" );

u8g.drawStr( 74, 15, "L" );

u8g.setPrintPos(42, 15) ;

u8g.print(total\_fuel\_consumption, 1);

if (LoggingOn == true) u8g.drawHLine(20, 24, 88);

u8g.setFont(u8g\_font\_profont22r);

switch (TT\_curr) { //для делителя 10k + 4.7k

case 0: u8g.drawStr( 95, 20, "1" ); break;

case 2: u8g.drawStr( 95, 20, "2" ); break;

case 4: u8g.drawStr( 95, 20, "3" ); break;

case 5: u8g.drawStr( 95, 20, "3L" ); break;

case 6: u8g.drawStr( 95, 20, "4" ); break;

case 7: u8g.drawStr( 95, 20, "4L" ); break;

}

if (getOBDdata(OBD\_SPD) > 1)

{

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "L/100Km" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print( 100 \* (getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM) / getOBDdata(OBD\_SPD)\*Ls \* 0.18), 1);

} else {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "L/Hour" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print(getOBDdata(OBD\_INJ) \* getOBDdata(OBD\_RPM)\*Ls \* 0.18, 1);

}

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 60, 42, "Average" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(60, 60) ;

if (total\_avg\_consumption < 100)

u8g.print( total\_avg\_consumption, 1);

else u8g.drawStr( 60, 60, "---" );

}

while ( u8g.nextPage() );

}

\*/

/\*

void drawTimeDistance(void) {

u8g.setFont(u8g\_font\_profont15r);

u8g.firstPage();

do {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 15, "TOTAL" );

//u8g.drawStr( 90, 15, "KM" );

u8g.setPrintPos(44, 15) ;

u8g.print(total\_trip, 1);

if (LoggingOn == true) u8g.drawHLine(20, 24, 88);

u8g.setFont(u8g\_font\_profont22r);

switch (TT\_curr) { //для делителя 10k + 4.7k

case 0: u8g.drawStr( 95, 20, "1" ); break;

case 2: u8g.drawStr( 95, 20, "2" ); break;

case 4: u8g.drawStr( 95, 20, "3" ); break;

case 5: u8g.drawStr( 95, 20, "3L" ); break;

case 6: u8g.drawStr( 95, 20, "4" ); break;

case 7: u8g.drawStr( 95, 20, "4L" ); break;

}

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "Avg SPD" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print(total\_avg\_speed, 1);

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 60, 42, "Time (M)" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(60, 60) ;

u8g.print( float(total\_time) / 60000, 1);

}

while ( u8g.nextPage() );

}

\*/

/\*

void drawTripTimeDistance(void) {

u8g.setFont(u8g\_font\_profont15r);

u8g.firstPage();

do {

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 15, "TRIP" );

//u8g.drawStr( 90, 15, "KM" );

u8g.setPrintPos(44, 15) ;

u8g.print(current\_trip, 1);

if (LoggingOn == true) u8g.drawHLine(20, 24, 88);

u8g.setFont(u8g\_font\_profont22r);

switch (TT\_curr) { //для делителя 10k + 4.7k

case 0: u8g.drawStr( 95, 20, "1" ); break;

case 2: u8g.drawStr( 95, 20, "2" ); break;

case 4: u8g.drawStr( 95, 20, "3" ); break;

case 5: u8g.drawStr( 95, 20, "3L" ); break;

case 6: u8g.drawStr( 95, 20, "4" ); break;

case 7: u8g.drawStr( 95, 20, "4L" ); break;

}

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 0, 42, "Avg SPD" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(0, 60) ;

u8g.print(trip\_avg\_speed, 1);

u8g.setFont(u8g\_font\_profont15r);

u8g.drawStr( 60, 42, "Time (M)" );

u8g.setFont(u8g\_font\_profont22r);

u8g.setPrintPos(60, 60) ;

u8g.print( float(current\_time) / 60000, 1);

}

while ( u8g.nextPage() );

}

\*/

void drawAllData(void) {

int16\_t x1, y1;

uint16\_t w, h;

float inj\_last, ign\_last, iac\_last,

// graphic commands to redraw the complete screen should be placed here

tft.fillScreen(ILI9341\_BLACK);

tft.setTextColor(ILI9341\_GREEN);

tft.setFont(&Orbitron\_Medium9pt7b);

tft.setCursor(15, FONT\_X);

tft.print("INJ");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X);

tft.println(getOBDdata(OBD\_INJ));

tft.setCursor(15, FONT\_X \* 2);

tft.print("IGN");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 2);

tft.print(int(getOBDdata(OBD\_IGN)));

tft.setCursor(15, FONT\_X \* 3);

tft.print("IAC");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 3);

tft.print(int(getOBDdata(OBD\_IAC)));

tft.setCursor(15, FONT\_X \* 4);

tft.print("RPM");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 4);

tft.print(int(getOBDdata(OBD\_RPM)));

tft.setCursor(15, FONT\_X \* 5);

tft.print("MAP");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 5);

tft.print(int(getOBDdata(OBD\_MAP)));

tft.setCursor(15, FONT\_X \* 6);

tft.print("ECT");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 6);

tft.print(int(getOBDdata(OBD\_ECT)));

tft.setCursor(15, FONT\_X \* 7);

tft.print("TPS");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 7);

tft.print(int(getOBDdata(OBD\_TPS)));

tft.setCursor(15, FONT\_X \* 8);

tft.print("SPD");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 8);

tft.print(int(getOBDdata(OBD\_SPD)));

tft.setCursor(15, FONT\_X \* 9);

tft.print("VF");

tft.setCursor(15 + FONT\_X \* 3, FONT\_X \* 9);

tft.print(getOBDdata(OBD\_OXSENS), 1);

if (int(getOBDdata(11)) == 1) {

tft.setCursor(160, FONT\_X);

tft.print("START");

tft.setCursor(160 + 26, FONT\_X \* 2);

tft.print("ENRICH");

}

if (int(getOBDdata(12)) == 1) {

tft.setCursor(160, FONT\_X \* 3);

tft.print("COLD");

}

if (int(getOBDdata(13)) == 1) {

tft.setCursor(160, FONT\_X \* 4);

tft.print("KNOCKING");

}

if (int(getOBDdata(14)) == 1) {

tft.setCursor(160, FONT\_X \* 5);

tft.print("OPEN LOOP");

}

if (int(getOBDdata(15)) == 1) {

tft.setCursor(160, FONT\_X \* 6);

tft.print("ACCEL");

tft.setCursor(160 + 26, FONT\_X \* 7);

tft.print("ENRICH");

}

if (int(getOBDdata(16)) == 1) {

tft.setCursor(160, FONT\_X \* 8);

tft.print("STARTER");

}

if (int(getOBDdata(17)) == 1) {

tft.setCursor(160, FONT\_X \* 9);

tft.print("IDLE");

}

if (int(getOBDdata(18)) == 1) {

tft.setCursor(160, FONT\_X \* 10);

tft.print("AIR COND");

}

if (int(getOBDdata(19)) == 1) {

tft.setCursor(160, FONT\_X \* 11);

tft.print("NEUTRAL");

}

tft.setCursor(160, FONT\_X \* 12);

if (int(getOBDdata(20)) == 0) {

tft.print("LEAN");

} else {

tft.print("RICH");

}

} // end void drawalldata

/\*

void autoscreenchange() {

CurrentDisplayIDX++;

if (CurrentDisplayIDX > 3) CurrentDisplayIDX = 1;

drawScreenSelector();

}

\*/

void ent() {//ПЕРЕКЛЮЧЕНИЕ ЭКРАНОВ

CurrentDisplayIDX++;

if (CurrentDisplayIDX > 6) CurrentDisplayIDX = 1;

// drawScreenSelector();

}

void getOBD(byte OBDdataIDX) {

DATA[OBD\_INJ] = ToyotaData[OBD\_INJ] \* 0.125;

DATA[OBD\_IGN] = ToyotaData[OBD\_IGN] \* 0.47 - 30;

DATA[OBD\_IAC] = ToyotaData[OBD\_IAC] \* 0.39215;

DATA[OBD\_RPM] = ToyotaData[OBD\_RPM] \* 25;

DATA[OBD\_MAP] = ToyotaData[OBD\_MAP] \* 2; //MAF

if (ToyotaData[OBD\_ECT] >= 243)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 243) \* 9.8) + 122;

else if (ToyotaData[OBD\_ECT] >= 237)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 237) \* 3.83) + 99;

else if (ToyotaData[OBD\_ECT] >= 228)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 228) \* 2.11) + 80.0;

else if (ToyotaData[OBD\_ECT] >= 210)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 210) \* 1.11) + 60.0;

else if (ToyotaData[OBD\_ECT] >= 180)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 180) \* 0.67) + 40.0;

else if (ToyotaData[OBD\_ECT] >= 135)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 135) \* 0.44) + 20.0;

else if (ToyotaData[OBD\_ECT] >= 82)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 82) \* 0.38);

else if (ToyotaData[OBD\_ECT] >= 39)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 39) \* 0.47) - 20.0;

else if (ToyotaData[OBD\_ECT] >= 15)

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 15) \* 0.83) - 40.0;

else

DATA[OBD\_ECT] = ((float)(ToyotaData[OBD\_ECT] - 15) \* 2.0) - 60.0;

//endif OBD\_ECT

DATA[OBD\_TPS] = ToyotaData[OBD\_TPS] / 1.8;

DATA[OBD\_SPD]ToyotaData[OBD\_SPD];

DATA[OBD\_OXSENS](float)ToyotaData[OBD\_OXSENS] \* 0.01953125;

#ifdef SECOND\_O2SENS

DATA[OBD\_OXSENS2] = (float)ToyotaData[OBD\_OXSENS2] \* 0.01953125;

#endif

DATA[OBD\_IAC]

}

float getOBDdata(byte OBDdataIDX) {

float returnValue;

switch (OBDdataIDX) {

case 0:// UNKNOWN

returnValue = ToyotaData[0];

break;

case OBD\_INJ: // Время впрыска форсунок =X\*0.125 (мс)

returnValue = ToyotaData[OBD\_INJ] \* 0.125; //Время впрыска форсунок

break;

case OBD\_IGN: // Угол опережения зажигания X\*0.47-30 (град)

returnValue = ToyotaData[OBD\_IGN] \* 0.47 - 30;

break;

case OBD\_IAC: // Состояние клапана ХХ Для разных типов КХХ разные формулы: X/255\*100 (%)

// X (шаг)

returnValue = ToyotaData[OBD\_IAC] \* 0.39215; ///optimize divide

break;

case OBD\_RPM: //Частота вращения коленвала X\*25(об/мин)

returnValue = ToyotaData[OBD\_RPM] \* 25;

break;

case OBD\_MAP: //Расходомер воздуха (MAP/MAF)

// X\*0.6515 (кПа)

// X\*4.886 (мм.ртут.столба)

// X\*0.97 (кПа) (для турбомоторов)

// X\*7.732 (мм.рт.ст) (для турбомоторов)

// (гр/сек) (данная формула для MAF так и не найдена)

// X/255\*5 (Вольт) (напряжение на расходомере)

returnValue = ToyotaData[OBD\_MAP] \* 2; //MAF

break;

case OBD\_ECT: // Температура двигателя (ECT)

// В зависимости от величины Х разные формулы:

// 0..14: =(Х-5)\*2-60

// 15..38: =(Х-15)\*0.83-40

// 39..81: =(Х-39)\*0.47-20

// 82..134: =(Х-82)\*0.38

// 135..179: =(Х-135)\*0.44+20

// 180..209: =(Х-180)\*0.67+40

// 210..227: =(Х-210)\*1.11+60

// 228..236: =(Х-228)\*2.11+80

// 237..242: =(Х-237)\*3.83+99

// 243..255: =(Х-243)\*9.8+122

// Температура в градусах цельсия.

if (ToyotaData[OBD\_ECT] >= 243)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 243) \* 9.8) + 122;

else if (ToyotaData[OBD\_ECT] >= 237)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 237) \* 3.83) + 99;

else if (ToyotaData[OBD\_ECT] >= 228)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 228) \* 2.11) + 80.0;

else if (ToyotaData[OBD\_ECT] >= 210)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 210) \* 1.11) + 60.0;

else if (ToyotaData[OBD\_ECT] >= 180)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 180) \* 0.67) + 40.0;

else if (ToyotaData[OBD\_ECT] >= 135)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 135) \* 0.44) + 20.0;

else if (ToyotaData[OBD\_ECT] >= 82)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 82) \* 0.38);

else if (ToyotaData[OBD\_ECT] >= 39)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 39) \* 0.47) - 20.0;

else if (ToyotaData[OBD\_ECT] >= 15)

returnValue = ((float)(ToyotaData[OBD\_ECT] - 15) \* 0.83) - 40.0;

else

returnValue = ((float)(ToyotaData[OBD\_ECT] - 15) \* 2.0) - 60.0;

break;

case OBD\_TPS: // Положение дроссельной заслонки

// X/2(градусы)

// X/1.8(%)

returnValue = ToyotaData[OBD\_TPS] / 1.8;

break;

case OBD\_SPD: // Скорость автомобиля (км/час)

returnValue = ToyotaData[OBD\_SPD];

break;

// Коррекция для рядных/ коррекция первой половины

case OBD\_OXSENS:

returnValue = (float)ToyotaData[OBD\_OXSENS] \* 0.01953125;

break;

#ifdef SECOND\_O2SENS

case OBD\_OXSENS2:// Lambda2 tst

returnValue = (float)ToyotaData[OBD\_OXSENS2] \* 0.01953125;

break;

#endif

// читаем Байты флагов побитно

case 11:

returnValue = bitRead(ToyotaData[11], 0); // Переобогащение после запуска 1-Вкл

break;

case 12:

returnValue = bitRead(ToyotaData[11], 1); //Холодный двигатель 1-Да

break;

case 13:

returnValue = bitRead(ToyotaData[11], 4); //Детонация 1-Да

break;

case 14:

returnValue = bitRead(ToyotaData[11], 5); //Обратная связь по лямбда зонду 1-Да

break;

case 15:

returnValue = bitRead(ToyotaData[11], 6); //Дополнительное обогащение 1-Да

break;

case 16:

returnValue = bitRead(ToyotaData[12], 0); //Стартер 1-Да

break;

case 17:

returnValue = bitRead(ToyotaData[12], 1); //Признак ХХ (Дроссельная заслонка) 1-Да(Закрыта)

break;

case 18:

returnValue = bitRead(ToyotaData[12], 2); //Кондиционер 1-Да

break;

case 19:

returnValue = bitRead(ToyotaData[12], 3); //Нейтраль 1-Да

break;

case 20:

returnValue = bitRead(ToyotaData[12], 4); //Смесь первой половины 1-Богатая, 0-Бедная

break;

#ifdef SECOND\_O2SENS //Вторая лябмда для Vобразных движков

case 21:

returnValue = bitRead(ToyotaData[12], 5); //Смесь второй половины 1-Богатая, 0-Бедная

break;

#endif

default: // DEFAULT CASE (in no match to number)

// send "error" value

returnValue = 9999.99;

} // end switch

// send value back

return returnValue;

} // end void getOBDdata

void ChangeState() {

static uint8\_t ID, EData[TOYOTA\_MAX\_BYTES];

static boolean InPacket = false;

static unsigned long StartMS;

static uint16\_t BitCount;

int state = digitalRead(ENGINE\_DATA\_PIN);

// digitalWrite(LED\_PIN, state);

if (InPacket == false) {

if (state == MY\_HIGH) {

StartMS = millis();

} else { // else if (state == MY\_HIGH)

if ((millis() - StartMS) > (15 \* 8)) {

StartMS = millis();

InPacket = true;

BitCount = 0;

} // end if ((millis() - StartMS) > (15 \* 8))

} // end if (state == MY\_HIGH)

} else { // else if (InPacket == false)

uint16\_t bits = ((millis() - StartMS) + 1 ) / 8; // The +1 is to cope with slight time errors

StartMS = millis();

// process bits

while (bits > 0) {

if (BitCount < 4) {

if (BitCount == 0)

ID = 0;

ID >>= 1;

if (state == MY\_LOW) // inverse state as we are detecting the change!

ID |= 0x08;

} else { // else if (BitCount < 4)

uint16\_t bitpos = (BitCount - 4) % 11;

uint16\_t bytepos = (BitCount - 4) / 11;

if (bitpos == 0) {

// Start bit, should be LOW

if ((BitCount > 4) && (state != MY\_HIGH)) { // inverse state as we are detecting the change!

ToyotaFailBit = BitCount;

InPacket = false;

break;

} // end if ((BitCount > 4) && (state != MY\_HIGH))

} else if (bitpos < 9) { //else TO if (bitpos == 0)

EData[bytepos] >>= 1;

if (state == MY\_LOW) // inverse state as we are detecting the change!

EData[bytepos] |= 0x80;

} else { // else if (bitpos == 0)

// Stop bits, should be HIGH

if (state != MY\_LOW) { // inverse state as we are detecting the change!

ToyotaFailBit = BitCount;

InPacket = false;

break;

} // end if (state != MY\_LOW)

if ( (bitpos == 10) && ((bits > 1) || (bytepos == (TOYOTA\_MAX\_BYTES - 1))) ) {

ToyotaNumBytes = 0;

ToyotaID = ID;

for (uint16\_t i = 0; i <= bytepos; i++)

ToyotaData[i] = EData[i];

ToyotaNumBytes = bytepos + 1;

if (bits >= 16) // Stop bits of last byte were 1's so detect preamble for next packet

BitCount = 0;

else {

ToyotaFailBit = BitCount;

InPacket = false;

}

break;

}

}

}

++BitCount;

--bits;

} // end while

} // end (InPacket == false)

} // end void change