**Общество с ограниченной ответственностью «ДЕЦИМА»**

УТВЕРЖДАЮ

Директор ООО «ДЕЦИМА»

/Шкляев А.А./

13.04. 2020г.

**ПРОГРАММА ДЛЯ ЭВМ**

**Программа обработки команд, кодирования и передачи информации**

**УРОВ**

**Авторы**: Порохня Дмитрий Викторович

Исходный текст программы

11 Листов

© ООО «ДЕЦИМА» 2020

**Москва 2020г.**

//------------------------------------------------------------------------------/\*

BLACK F407VG

Serial commanication: SerialUART1

Upload method: STLink

\*/

//------------------------------------------------------------------------------#include <Arduino.h>

#include "CONFIG.h"

#include "TFTMenu.h"

#include "DS3231.h" // подключаем часы

#include "ConfigPin.h"

#include "AT24CX.h"

#include "InterruptHandler.h"

#include "DelayedEvents.h"

//------------------------------------------------------------------------------// подключаем наши экраны

#include "Screen1.h" // Главный экран

#include "Screen2.h" // Вызов меню настроек

#include "Screen3.h" //

#include "Screen4.h" // Вызов меню установки времени и даты

#include "Screen5.h" // Вызов установки времени

#include "Screen6.h" // Вызов установки даты

#include "InterruptScreen.h" // экран с графиком прерывания

#include "Buttons.h" // наши железные кнопки

#include "Feedback.h" // обратная связь (диоды и прочее)

#include "FileUtils.h"

#include "Settings.h"

#include "CoreCommandBuffer.h"

#include <Wire.h>

#include "Endstops.h"

#include "RS485.h"

//------------------------------------------------------------------------------TwoWire Wire1 **=** TwoWire**(**I2C2**,** PB11**,** PB10**);** // второй I2C

Vector**<**uint8\_t**>** LastTriggeredInterruptRecord**;** // список последнего сработавшего прерывания

//------------------------------------------------------------------------------#ifdef USE\_EXTERNAL\_WATCHDOG

ExternalWatchdogSettings watchdogSettings**;**

#endif

//------------------------------------------------------------------------------uint32\_t screenIdleTimer **=** 0**;**

bool setupDone **=** **false;**

//------------------------------------------------------------------------------void screenAction**(**AbstractTFTScreen**\*** screen**)**

**{**

// какое-то действие на экране произошло.

// тут просто сбрасываем таймер ничегонеделанья.

screenIdleTimer **=** millis**();**

**}**

//------------------------------------------------------------------------------// РАБОТА С RS-485

//------------------------------------------------------------------------------RS485 rs485**(**RS485\_SERIAL**,**Upr\_RS485**,**RS485\_READING\_TIMEOUT**);**

uint32\_t rs485RelayTriggeredTime **=** 0**;** // время срабатывания защиты

uint32\_t rs485DataArrivedTime **=** 0**;**

DS3231Time rsRelTrigTime**;** // время срабатывания защиты

bool HasRS485Link **=** **false;**

uint32\_t lastRS485PacketSeenAt **=** 0**;**

//------------------------------------------------------------------------------void processInterruptFromModule**(**uint32\_t dataArrivedTime**,** DS3231Time**&** tm**,** InterruptTimeList**&** interruptsList**,** bool endstopUpTriggered**,** bool endstopDownTriggered**)**

**{**

// тут обрабатываем результаты срабатывания защиты от модуля

// обновляем моторесурс, т.к. было срабатывание защиты

//DBGLN(F("processInterruptFromModule: INC motoresource!"));

uint32\_t motoresource **=** Settings**.**getMotoresource**(**0**);**

motoresource**++;**

Settings**.**setMotoresource**(**0**,** motoresource**);**

bool hasAlarm **=** **!**interruptsList**.**size**();** // авария, если в списке нет данных

**if** **(**hasAlarm**)**

**{**

// DBGLN(F("processInterruptFromModule: HAS ALARM FLAG!"));

// сделал именно так, поскольку флаг аварии сбрасывать нельзя, плюс могут понадобиться дополнительные действия

Feedback**.**alarm**(true);**

**}**

// начинаем сравнивать с эталоном

EthalonCompareResult compareRes1 **=** COMPARE\_RESULT\_NoSourcePulses**;**

EthalonCompareNumber compareNumber1**;**

InterruptTimeList ethalonData1**;**

bool needToLog **=** **false;**

// теперь смотрим - надо ли нам самим чего-то обрабатывать?

**if** **(**interruptsList**.**size**()** **>** 1**)**

**{**

// DBG("processInterruptFromModule: INTERRUPT HAS DATA COUNT: ");

// DBGLN(interruptsList.size());

// зажигаем светодиод "ТЕСТ"

Feedback**.**testDiode**();**

needToLog **=** **true;**

// здесь мы можем обрабатывать список сами - в нём ЕСТЬ данные

compareRes1 **=** EthalonComparer**::**Compare**(**interruptsList**,** 0**,** compareNumber1**,** ethalonData1**);**

**if** **(**compareRes1 **==** COMPARE\_RESULT\_MatchEthalon**)**

**{**

// DBGLN(F("processInterruptFromModule: MATCH ETHALON!"));

**}**

**else** **if** **(**compareRes1 **==** COMPARE\_RESULT\_MismatchEthalon **||** compareRes1 **==** COMPARE\_RESULT\_RodBroken**)**

**{**

// DBGLN(F("processInterruptFromModule: MISMATCH ETHALON!"));

Feedback**.**failureDiode**();**

Feedback**.**alarm**();**

**}**

**}**

**else**

**{**

// DBGLN(F("processInterruptFromModule: INTERRUPT HAS NO DATA!!!"));

**}**

// ИЗМЕНЕНИЯ ПО ТОКУ - НАЧАЛО //

// получаем список данных по току

CurrentOscillData oscillData **=** InterruptHandlerClass**::**getCurrentData**();**

// нормализуем список времён

InterruptHandlerClass**::**normalizeList**(**oscillData**.**times**);**

// ИЗМЕНЕНИЯ ПО ТОКУ - КОНЕЦ //

bool wantToInformSubscriber **=** **(**hasAlarm **||** **(**interruptsList**.**size**()** **>** 1**));**

**if** **(**wantToInformSubscriber**)**

**{**

// DBGLN(F("processInterruptFromModule: WANT TO INFORM SUBSCRIBER!"));

InterruptHandler**.**informSubscriber**(**oscillData**,**interruptsList**,** compareRes1/\*, millis() - rs485RelayTriggeredTime, rs485RelayTriggeredTime\*/**);**

// обновляем экран, чтобы график появился сразу

Screen**.**update**();**

**}**

**if** **(**needToLog**)**

**{**

// записываем в EEPROM

InterruptHandlerClass**::**writeToLog**(**dataArrivedTime**,** tm**,** oscillData**,**interruptsList**,** compareRes1**,** compareNumber1**,** ethalonData1**,true);**

#ifndef \_SD\_OFF

// DBGLN(F("processInterruptFromModule: WANT TO LOG ON SD!"));

// надо записать в лог на SD дату срабатывания системы

InterruptHandlerClass**::**writeToLog**(**dataArrivedTime**,** tm**,** oscillData**,**interruptsList**,** compareRes1**,** compareNumber1**,** ethalonData1**);**

#endif // !\_SD\_OFF

**}** // needToLog

**}**

//------------------------------------------------------------------------------void OnRS485IncomingData**(**RS485**\*** Sender**)**

**{**

// пришёл пакет от модуля по RS-485 (не обязательно от модуля, но в целом - пришёл какой-то пакет от кого-то)

HasRS485Link **=** **true;** // обновляем флаг, что есть связь

lastRS485PacketSeenAt **=** millis**();**

uint8\_t**\*** data**;**

RS485Packet packet **=** rs485**.**getDataReceived**(**data**);**

**switch** **(**packet**.**packetType**)**

**{**

**case** rs485Ping**:** // сообщение вида "ПРОВЕРКА СВЯЗИ", посылает модуль периодически для проверки связи

**{**

// пришёл пакет пинга

// DBGLN(F("[RS-485] RECEIVE PING PACKET FROM MODULE !!!"));

**}**

**break;** // rs485Ping

**case** rs485HasInterrupt**:** // сообщение вида "ЕСТЬ СРАБАТЫВАНИЕ ЗАЩИТЫ", посылает модуль по факту срабатывания защиты

**{**

// DBGLN(F("[RS-485] HAS INTERRUPT FROM MODULE !!!"));

rs485RelayTriggeredTime **=** millis**();** // запоминаем время срабатывания защиты

rsRelTrigTime **=** RealtimeClock**.**getTime**();**

// ИЗМЕНЕНИЯ ПО ТОКУ - НАЧАЛО //

InterruptHandlerClass**::**startCollectCurrentData**();** // начинаем собирать данные по току

// ИЗМЕНЕНИЯ ПО ТОКУ - КОНЕЦ //

//TODO: ТУТ МОЖНО ЧТО-ТО ДЕЛАТЬ ПО ФАКТУ СРАБАТЫВАНИЯ ЗАЩИТЫ, НАПРИМЕР - НАЧИНАТЬ ПРОВЕРКУ ТОКОВЫХ ТРАНСФОРМАТОРОВ.

**}**

**break;** // rs485HasInterrupt

**case** rs485InterruptData**:** // сообщение вида "ДАННЫЕ ПО ПРЕРЫВАНИЮ", посылает модуль по факту собирания списка прерываний

**{**

// DBGLN(F("[RS-485] HAS INTERRUPT LIST FROM MODULE !!!"));

// пришли данные по прерыванию, собираем их

// первый байт в пакете - состояние верхнего концевика на момент окончания сбора данных

bool endstopUpTriggered **=** **\***data**++;**

// второй байт в пакете - состояние нижнего концевика на момент окончания сбора данных

bool endstopDownTriggered **=** **\***data**++;**

// потом идёт список прерываний

// 2 байта - служебный заголовок пакета (концевик 1 + концевик 2), потом N записей по 4 байта. Высчитываем это из длины пакета в байтах

uint16\_t recordsCount **=** **(**packet**.**dataLength **-** 2**)** **/** **sizeof(**uint32\_t**);**

uint32\_t**\*** rec **=** **(**uint32\_t**\*)**data**;**

// DBG(F("[RS-485] INTERRUPTS COUNT: "));

// DBGLN(recordsCount);

// сохраняем записи

InterruptTimeList interruptsList**;**

interruptsList**.**reserve**(**recordsCount**);**

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

**{**

interruptsList**.**push\_back**(\***rec**++);**

**}**

// запоминаем время, когда данные пришли

rs485DataArrivedTime **=** millis**();**

uint32\_t dataArriveTime **=** rs485DataArrivedTime **-** rs485RelayTriggeredTime**;**

// выводим их для теста

/\*

#ifdef \_MY\_DEBUG

DBGLN(F("-----INTERRUPTS LIST FROM MODULE -----"));

for (size\_t i = 0; i<interruptsList.size(); i++)

{

DBGLN(interruptsList[i]);

}

DBGLN(F("-----INTERRUPTS LIST END -----"));

#endif

\*/

// ИЗМЕНЕНИЯ ПО ТОКУ - НАЧАЛО //

// говорим, что хватит нам собирать данные по току

InterruptHandlerClass**::**stopCollectCurrentData**();**

// ИЗМЕНЕНИЯ ПО ТОКУ - КОНЕЦ //

// обрабатываем список прерываний

processInterruptFromModule**(**dataArriveTime**,** rsRelTrigTime**,**interruptsList**,** endstopUpTriggered**,** endstopDownTriggered**);**

**}**

**break;** // rs485InterruptData

**default:**

**{**

//DBG(F("[RS-485] MAIN HANDLER, UNHANDLED PACKET TYPE: "));

//DBGLN(packet.packetType);

**}**

**break;**

**}** // switch

rs485**.**clearReceivedData**();** // очищаем входящие данные

**}**

//------------------------------------------------------------------------------void SwitchRS485MainHandler**(**bool on**)**

**{**

**if** **(!**on**)** // нас выключили

**{**

// DBGLN(F("Main handler, release RS-485..."));

rs485**.**setHandler**(NULL);**

rs485**.**clearReceivedData**();**

// DBGLN(F("Main handler, RS-485 released."));

**}**

**else** // включили

**{**

// DBGLN(F("Main handler, own RS-485..."));

rs485**.**setHandler**(**OnRS485IncomingData**);**

// DBGLN(F("Main handler, RS-485 owned."));

**}**

**}**

//------------------------------------------------------------------------------#ifdef USE\_EXTERNAL\_WATCHDOG

//------------------------------------------------------------------------------void updateExternalWatchdog**()**

**{**

static unsigned long watchdogLastMillis **=** millis**();**

unsigned long watchdogCurMillis **=** millis**();**

uint16\_t dt **=** watchdogCurMillis **-** watchdogLastMillis**;**

watchdogLastMillis **=** watchdogCurMillis**;**

watchdogSettings**.**timer **+=** dt**;**

**switch(**watchdogSettings**.**state**)**

**{**

**case** WAIT\_FOR\_TRIGGERED**:**

**{**

**if(**watchdogSettings**.**timer **>=** WATCHDOG\_WORK\_INTERVAL**)**

**{**

watchdogSettings**.**timer **=** 0**;**

watchdogSettings**.**state **=** WAIT\_FOR\_NORMAL**;**

digitalWrite**(**WATCHDOG\_REBOOT\_PIN**,** WATCHDOG\_TRIGGERED\_LEVEL**);**

**}**

**}**

**break;**

**case** WAIT\_FOR\_NORMAL**:**

**{**

**if(**watchdogSettings**.**timer **>=** WATCHDOG\_PULSE\_DURATION**)**

**{**

watchdogSettings**.**timer **=** 0**;**

watchdogSettings**.**state **=** WAIT\_FOR\_TRIGGERED**;**

digitalWrite**(**WATCHDOG\_REBOOT\_PIN**,** WATCHDOG\_NORMAL\_LEVEL**);**

**}**

**}**

**break;**

**}**

**}**

//------------------------------------------------------------------------------#endif // USE\_EXTERNAL\_WATCHDOG

//------------------------------------------------------------------------------void setup**()**

**{**

Serial**.**begin**(**SERIAL\_SPEED**);**

**while(!**Serial **&&** millis**()** **<** 2000**);**

DBGLN**(**F**(**"Init I2C..."**));**

//Wire.begin();

Wire1**.**begin**();**

DBGLN**(**F**(**"I2C inited."**));**

ConfigPin**::**setup**();**

#ifdef USE\_EXTERNAL\_WATCHDOG

pinMode**(**WATCHDOG\_REBOOT\_PIN**,**OUTPUT**);**

digitalWrite**(**WATCHDOG\_REBOOT\_PIN**,**WATCHDOG\_NORMAL\_LEVEL**);**

watchdogSettings**.**timer **=** 0**;**

watchdogSettings**.**state **=** WAIT\_FOR\_TRIGGERED**;**

#endif

DBGLN**(**F**(**"Init settings..."**));**

Settings**.**begin**();**

DBGLN**(**F**(**"Settings inited."**));**

DBGLN**(**F**(**"Init RTC..."**));**

RealtimeClock**.**begin**();**

// RealtimeClock.setTime(0,1,11,1,7,2,2018);

DBGLN**(**F**(**"RTC inited."**));**

DBGLN**(**F**(**"Init endstops..."**));**

SetupEndstops**();**

DBGLN**(**F**(**"Endstops inited."**));**

DBGLN**(**F**(**"Init RS-485..."**));**

// инициализируем RS-485

RS485\_SERIAL**.**begin**(**RS485\_SPEED**);**

rs485**.**setHandler**(**OnRS485IncomingData**);**

rs485**.**begin**();**

DBGLN**(**F**(**"RS-485 inited."**));**

#ifndef \_SD\_OFF

DBGLN**(**F**(**"INIT SD..."**));**

**if** **(**SDInit**::**InitSD**())**

**{**

DBGLN**(**F**(**"SD inited."**));**

**}**

**else**

**{**

DBGLN**(**F**(**"ОШИБКА ИНИЦИАЛИЗАЦИИ SD!!"**));**

**}**

#endif // !\_SD\_OFF

DBGLN**(**F**(**"Init screen..."**));**

Screen**.**setup**();**

DBGLN**(**F**(**"Add screen1..."**));**

Screen**.**addScreen**(**Screen1**::**create**());** // первый экран покажется по умолчанию

#ifndef \_SCREEN\_2\_OFF

DBGLN**(**F**(**"Add screen2..."**));**

// добавляем второй экран

Screen**.**addScreen**(**Screen2**::**create**());**

#endif // !\_SCREEN\_2\_OFF

#ifndef \_SCREEN\_3\_OFF

DBGLN**(**F**(**"Add screen3..."**));**

// добавляем третий экран. Переход в меню настройки

Screen**.**addScreen**(**Screen3**::**create**());**

#endif // !\_SCREEN\_3\_OFF

#ifndef \_SCREEN\_4\_OFF

DBGLN**(**F**(**"Add screen4..."**));**

// добавляем четвертый экран. Меню установки даты и времени

Screen**.**addScreen**(**Screen4**::**create**());**

#endif // !\_SCREEN\_4\_OFF

#ifndef \_SCREEN\_5\_OFF

DBGLN**(**F**(**"Add screen5..."**));**

// добавляем 5 экран. Установка времени

Screen**.**addScreen**(**Screen5**::**create**());**

#endif // !\_SCREEN\_5\_OFF

#ifndef \_SCREEN\_6\_OFF

DBGLN**(**F**(**"Add screen6..."**));**

// добавляем 6 экран. Установка даты

Screen**.**addScreen**(**Screen6**::**create**());**

#endif // !\_SCREEN\_6\_OFF

DBGLN**(**F**(**"Add interrupt screen..."**));**

// добавляем экран с графиком прерываний

Screen**.**addScreen**(**InterruptScreen**::**create**());**

// переключаемся на первый экран

Screen**.**switchToScreen**(**"Main"**);**

// настраиваем обратную связь (информационные диоды и пр.)

Feedback**.**begin**();**

// настраиваем железные кнопки

Buttons**.**begin**();**

// поднимаем наши прерывания

InterruptHandler**.**begin**();**

screenIdleTimer **=** millis**();**

Screen**.**onAction**(**screenAction**);**

DBGLN**(**F**(**"Inited."**));**

CommandHandler**.**getVER**(&**Serial**);**

setupDone **=** **true;**

**}**

//------------------------------------------------------------------------------void loop**()**

**{**

#ifndef \_DELAYED\_EVENT\_OFF

CoreDelayedEvent**.**update**();**

#endif // \_DELAYED\_EVENT\_OFF

#ifdef USE\_EXTERNAL\_WATCHDOG

updateExternalWatchdog**();**

#endif // USE\_EXTERNAL\_WATCHDOG

Settings**.**update**();**

// обновляем кнопки

Buttons**.**update**();**

Screen**.**update**();**

InterruptHandler**.**update**();**

// проверяем, какой экран активен. Если активен главный экран - сбрасываем таймер ожидания. Иначе - проверяем, не истекло ли время ничегонеделанья.

AbstractTFTScreen**\*** activeScreen **=** Screen**.**getActiveScreen**();**

**if(**activeScreen **==** mainScreen**)**

**{**

screenIdleTimer **=** millis**();**

**}**

**else**

**{**

**if(**millis**()** **-** screenIdleTimer **>** RESET\_TO\_MAIN\_SCREEN\_DELAY**)**

**{**

// DBGLN(F("ДОЛГОЕ БЕЗДЕЙСТВИЕ, ПЕРЕКЛЮЧАЕМСЯ НА ГЛАВНЫЙ ЭКРАН!"));

screenIdleTimer **=** millis**();**

Screen**.**switchToScreen**(**mainScreen**);**

**}**

**}** // else

#ifndef \_COM\_COMMANDS\_OFF

// обрабатываем входящие команды

CommandHandler**.**handleCommands**();**

#endif // \_COM\_COMMANDS\_OFF

// обновляем RS-485

rs485**.**update**();**

**if** **(**millis**()** **-** lastRS485PacketSeenAt **>=** **(**RS485\_PING\_PACKET\_FREQUENCY**)\***3**)**

**{**

HasRS485Link **=** **false;** // долго не было пакетов по RS-485

lastRS485PacketSeenAt **=** millis**();** // чтобы часто не дёргать

**}**

**}**

//------------------------------------------------------------------------------bool nestedYield **=** **false;**

//------------------------------------------------------------------------------#ifndef \_YIELD\_OFF

//------------------------------------------------------------------------------void yield**()**

**{**

**if(**nestedYield **||** **!**setupDone**)**

**return;**

nestedYield **=** **true;**

// обновляем прерывания

InterruptHandler**.**update**();**

#ifndef \_DELAYED\_EVENT\_OFF

CoreDelayedEvent**.**update**();**

#endif // \_DELAYED\_EVENT\_OFF

Buttons**.**update**();**

#ifdef USE\_EXTERNAL\_WATCHDOG

updateExternalWatchdog**();**

#endif // USE\_EXTERNAL\_WATCHDOG

nestedYield **=** **false;**

**}**

//------------------------------------------------------------------------------#endif // \_YIELD\_OFF

//------------------------------------------------------------------------------