Структура программы Интеллектуального драйвера версии 3.0

Программа представлена в виде проекта для IDE Keil uVision v5.2. Настройка перифирии микроконтроллера осуществлена при помощи IDE STM32CubeMX.

Исполнительный файл проекта - Driver\_rev3.0.uvprojx

Файл конфигурации периферии микроконтроллера - Driver\_rev3.0.ioc

Для взаимодействия с периферией МК использована библиотека HAL.

Основные программные модули:

main.c – инициализация периферии и переменных

general\_math.c – основные функции программы

general\_math.h – структуры, функции прототипы и константы

stm32f3xx\_it.c – обработчик прерываний

В остальных модулях представлены параметры инициализации периферии.

Описание основных структур, переменных и функций для работы модулей:

uint32\_t psk = 0; // рабочая длительность импульса

uint32\_t psk\_buff = 0; // временная длительность импульса

uint32\_t Period; // длительность периода

uint32\_t Pulse\_1; // длительность верхнего ключа диагональ В

uint32\_t Pulse\_2; // длительность нижнего ключа диагональ В

uint32\_t Pulse\_3; // длительность верхнего ключа диагональ А

uint32\_t Pulse\_4; // длительность нижнего ключа диагональ А

uint32\_t DeathTime\_u = 50; // длительность паузы между диагоналями в мкс

uint32\_t DeathTime; // рабочая длительность паузы в тиках таймера

int32\_t Overlap\_u = -50; // длительность перекрытия нижних ключей в мкс

uint32\_t Overlap; // рабочая длительность паузы в тиках таймера

uint32\_t TimeLimit\_u = 650; // максимальная длительность импульса верхних ключей

uint32\_t TimeLimit; // рабочая длительность импульса верхних ключей

uint32\_t Tick = 8; // количество тиков таймера в 1 мкс

uint8\_t SoftStart = ON; // плавный запуск вкл/выкл

uint8\_t State = pwmSTART; // состояние передачи 0 - запуск, 1 - передача, 2 - останов

uint16\_t DataLenght = 8; // длина сообщения

uint8\_t Position = 0; // позиция в сообщении

uint8\_t Data[MAXDAT]; // данные

uint32\_t f1 = 719; // передача "1"

uint32\_t f2 = 769; // передача "0"

float BR = 1; // скорость передачи

uint8\_t TransMode = 0; // режим передатчика: 0 - передача сообщения, 1 - размагничивание, 2 - контроль, 3 - генератор

uint8\_t f\_change = 0; // триггер смены частоты для диагностики

// буферы значений данных с АЦП

uint32\_t ADC1buff;

uint32\_t ADC2buff;

uint32\_t ADC3buff;

uint32\_t ADC4buff;

// значения напряжения на трансформаторе тока

float U1rise1 = CONST; // после переключения

float U1fall1 = CONST; // перед переключением

float U2rise1 = CONST;

float U2fall1 = CONST;

float U3rise1 = CONST;

float U3fall1 = CONST;

float U4rise1 = CONST;

float U4fall1 = CONST;

float U1rise2 = CONST;

float U1fall2 = CONST;

float U2rise2 = CONST;

float U2fall2 = CONST;

float U3rise2 = CONST;

float U3fall2 = CONST;

float U4rise2 = CONST;

float U4fall2 = CONST;

// измерение средней точки

float mean1 = 0;

float mean2 = 0;

float mean3 = 0;

float mean4 = 0;

uint8\_t defcnt = 0; // счетчик срабатывания защиты

// емкость в антеннах

float C1;

float C2;

float C3;

float C4;

// уточненное значение емкости в антеннах

float C1corr;

float C2corr;

float C3corr;

float C4corr;

// среднее уточненное значение емкости в антеннах

float C1cra;

float C2cra;

float C3cra;

float C4cra;

// индуктивность антенн

float L1;

float L2;

float L3;

float L4;

// средняя индуктивность антенн

float L1a;

float L2a;

float L3a;

float L4a;

// сопротивление антенн

float R1;

float R2;

float R3;

float R4;

// среднее сопротивление антенн

float R1a;

float R2a;

float R3a;

float R4a;

// мощность в антеннах

float P1;

float P2;

float P3;

float P4;

// средняя мощность в антеннах

float P1a;

float P2a;

float P3a;

float P4a;

// максимальная мощность в антеннах

float P1m;

float P2m;

float P3m;

float P4m;

// напряжение на индуктивнсти антенны

float UL1;

float UL2;

float UL3;

float UL4;

// напряжение питания антенны

float Us1;

float Us2;

float Us3;

float Us4;

// ток в антеннах

float I1;

float I2;

float I3;

float I4;

// средний ток

float I1a;

float I2a;

float I3a;

float I4a;

// максимальный ток

float I1m;

float I2m;

float I3m;

float I4m;

// период

float T;

// индуктивность трансформатора напряжения

float L0 = 3200e-9f; // 40

float Ld = 0.0e-3f; // индуктивность дросселя

// напряжение питания

float Us = 310; // исходное напряжение питания

// напряжение на антеннах

float Us1 = 98;

float Us2 = 98;

float Us3 = 48;

float Us4 = 48;

// переменные для диагностики

uint32\_t DiagTime = 1; // период диагностики

uint32\_t DiagCNT = 0; // количество измерений диагностики

uint32\_t DiagPulse = 0;

uint8\_t RiseDelay = 80; // задержка перед измерением в момент переключения

uint8\_t FallDelay = 80; // задержка перед измерением в момент перед переключением

uint8\_t PauseDelay = 20; // задержка перед измерением в момент после паузы

// переменные для синхронизации

uint8\_t TimeOut\_en; // запуск таймера для синхронизации

uint16\_t TimeOut\_cnt; // счетчик таймера

uint8\_t Uart\_RX\_TimeOut\_cnt = 0; // счетчик таймаута UART

// параметры для контроля в режиме работы

float UIrise[4]; // усредненное значение напряжения контроля

float UI\_break[4]; // порог срабатывания защиты по обрыву

float UI\_fuse[4]; // порог срабатывания защиты по кз

uint16\_t mean\_cnt = 0; // счетчик измерений для защиты по выходу

uint16\_t mean\_value = 32; // количество измерений для защиты по выходу

uint8\_t ant\_nwork = 0; // количество неработающих антенн

uint8\_t ant\_work = 0; // количество работающих антенн

uint8\_t sec = 0; // секунды

uint32\_t min = 0; // минуты

uint16\_t alarm\_trans\_cnt = 0; // счетчик времени передачи аварийного оповещения

uint16\_t alarm\_pause\_cnt = 0; // счетчик паузы аварийного оповещения

uint8\_t n = 0; // счет момента измерения

uint8\_t n\_IC = 0; // режим захвата частоты внешнего запуска 1 - захват, 0 - нет

uint16\_t CCR1tmp = 0; // регистр сравнения частоты

uint8\_t n\_stop = 0; // счет импульсов плавного останова

uint8\_t n\_s; // триггер смены длительности плавного запуска и останова

uint8\_t diag = 0; // триггер диагностики

uint8\_t def = 0; // триггер сработки защиты по току

uint8\_t CodeMode = 0; // кодировка

uint8\_t SendForm = 0; // 0 - шахтер, 1 - команда, 2 - шахтер и команда, 3 - авария, 4 - тест меток

uint8\_t RepeatNum = 0; // число повторов

uint16\_t freq; // рабочая частота

uint32\_t tmpData = 43690; // вызываемый номер

uint8\_t tmpComand = 0; // вызываемая команда

\_Bool StartSend = 0; // запуск передачи

\_Bool StopDiag = 0; // остановка диагностики

StatusSystem\_t STATUS; // структура статуса

SettingParametrs\_t SETUP; // структура настроек

UART2\_Queue\_Data UART2RecvData; // структура обмена данными

UART2Recv\_t UART2\_RecvType; // тип принимаемых данных

Для обмена данными с БУ используются структуры STATUS и SETUP. Команды управления используются глобальные.

Описание структур:

STATUS

// состояния на отправку в передатчик

uint8\_t overload\_cnt; // кличество срабатываний защиты

float overload\_curr; // ток срабатывания защиты

uint8\_t bitnum; // текущий бит

uint8\_t repeatcur; // текущий повтор

float ia[4]; // усредненный амплетудный ток в антенне

float im[4]; // максимальное значение тока в антенне

float pa[4]; // усредненное значение мощности в антенне

float pm[4]; // максимальное значение мощности в антенне

float ra[4]; // сопротивление антенны

float c[4]; // емкость антенны

float l[4]; // индуктивность антенны

uint8\_t trans\_ok;// состояние целостности передатчика 1 - рабочий, 0 - сработало ограничение по защите

uint8\_t trans\_state; // состояние работы передачи 1 - передача, 2 - контроль, 0 - ожидание

uint8\_t drvenable; // состояние активности драйвера 1 - драйвер включен, 0 - драйвер в режиме ожидания

uint8\_t ant\_fuse[4]; // костояние кз в антенне, 0 - нет, 1 - кз

uint8\_t ant\_break[4]; // состояние обрыва антенны, 0 - нет, 1 - обрыв

uint16\_t driver\_fw; //версия по драйвера

uint16\_t driver\_hw; //версия блока драйвера

SETUP

// настройки передатчика

uint32\_t data; // вызываемый номер метки

uint16\_t freq1; // частота "1"

uint16\_t freq2; // частота "0"

uint16\_t boudrate; // бодрейт

uint8\_t deadtime; // пауза между открытиями ключей в мкс

int8\_t overlap; // перекрытие нижних ключей в мкс

uint16\_t duration\_limit; //ограничение длительности импульса в мкс

uint16\_t check\_ant\_period; // период контроля в мин

uint8\_t current\_limit; // ограничение тока(А)

uint8\_t count\_halt\_limit; // число срабатываний защиты

float capatity\_ant[4]; // ёмкость в антеннах

uint8\_t i\_break[4]; // ток обрыва антенны

uint8\_t i\_fuse[4]; // ток короткого замыкания антенны

uint8\_t repeatnum; // число повторов передачи

uint8\_t enable; // запуск передачи

uint8\_t mode; // режим работы 1 - основной, 0 - внешний, 2 - генератор

uint8\_t softsrart; // плавный запуск

uint8\_t currentlimit; // режим ограничения тока

uint8\_t diag; // запуск диагностики

uint8\_t cap; // емкость подлючена/отключена

uint8\_t modulation; // модуляция psk/fsk

uint8\_t codemode; // кодировка 0 - бинарная, 1 - радиус 1, 2 - радиус 2

uint8\_t overloadmode; // режим перегрузки по току 1 - автоматический, 0 - одиночный

uint8\_t suppvoltage; // напряжение питания 540 - высокое, 310 - низкое

uint8\_t alarm\_msg; // запуск аварийного оповещения

uint8\_t test\_tag; // запуск тестового сообщения

uint8\_t command; // запуск передачи команды (радиус)

uint8\_t ant[4]; // состояние антенн 1 - подключена, 0 отключена, -1 - отсутствует

uint8\_t antlevel[4]; // выход антенны 1 - высокое, 0 - низкое

uint16\_t turns\_ant[4]; // количество витков во вторичной обмотке (полное)

uint16\_t turns\_prim; // количество витков в первичной обмотке

uint8\_t supplevel; //напряжение первичной обмотки относительно выхода 1 -высокое, 0 низкое

uint8\_t standby; //режим ожидания вкл/выкл

список команд

CT\_UNDEFINED = -1,

// I2C

I2C\_GET\_TEMP\_LVL\_1\_ON, // команды ля i2c датчика температуры (не используются в драйвере)

I2C\_GET\_TEMP\_LVL\_2\_ON,

I2C\_GET\_TEMP\_LVL\_1\_OFF,

I2C\_GET\_TEMP\_LVL\_2\_OFF,

I2C\_SET\_TEMP\_LVL\_1\_ON,

I2C\_SET\_TEMP\_LVL\_2\_ON,

I2C\_SET\_TEMP\_LVL\_1\_OFF,

I2C\_SET\_TEMP\_LVL\_2\_OFF,

I2C\_LED\_ON, // команды для i2c индикации (не используются в драйвере)

I2C\_LED\_OFF,

// драйвер передатчика

U2CT\_DATA,//i // передача номера в драйвер

U2CT\_F1,//i16 // передача f1 в драйвер

U2CT\_F2,//i16 // передача f2 в драйвер

U2CT\_BR,//i16 // передача бодрейта в драйвер

U2CT\_DEATH\_TIME,//i8 // передача паузы в драйвер

U2CT\_OVERLAP,//i8 // передача перекрытия в драйвер

U2CT\_TIME\_LIMIT,//i16 // передача ограничения длительности импульса в драйвер

U2CT\_DIAG\_PERIOD,//i16 // передача периода диагностики в драйвер

U2CT\_CURRENT\_LIMIT\_A,//i8 // передача значения ограничения тока в драйвер

U2CT\_OVER\_TICK,//i8 // передача количества срабатывания защиты в драйвер

U2CT\_C1,//f // передача емкости в антенне в драйвер

U2CT\_C2,//f

U2CT\_C3,//f

U2CT\_C4,//f

U2CT\_I\_BREAK\_ANT\_1,//i8 //передача тока обрыва антенны в драйвер

U2CT\_I\_BREAK\_ANT\_2,//i8

U2CT\_I\_BREAK\_ANT\_3,//i8

U2CT\_I\_BREAK\_ANT\_4,//i8

U2CT\_I\_FUSE\_ANT\_1,//i8 // передача тока замыкания антенны в драйвер

U2CT\_I\_FUSE\_ANT\_2,//i8

U2CT\_I\_FUSE\_ANT\_3,//i8

U2CT\_I\_FUSE\_ANT\_4,//i8

U2CT\_REPEATNUM,//i8 // передача числа повторов в драйвер

U2CT\_ENABLE,//i8 // команда запуска передачи в драйвер

U2CT\_MODE,//i8 // передача режима работы генератора в драйвер

U2CT\_SOFT\_START,//i8 // передача режима плавного запуска в драйвер

U2CT\_CURRENT\_LIMIT,//i8 // передача режима ограничкения тока в драйвер

U2CT\_DIAG,//i8 // команда запуска диагностики в драйвер

U2CT\_CAP,//i8 // передача режима емкости в драйвер

U2CT\_MODUlATION,//i8 // передача режима модуляции в драйвер

U2CT\_CODE\_MODE,//i8 // передача кодировки в драйвер

U2CT\_OVERLOAD\_MODE,//i8 // передача режима перегрузки по току в драйвер

U2CT\_SUPP\_VOLTAGE,//i8 // передача режима напряжения питания в драйвер

U2CT\_ALARM\_MSG,//i8 // передача команды на запуск аварийного оповещения в драйвер

U2CT\_TEST\_TAG,//i8 // перердача команды на тест меток в драйвер

U2CT\_COMMAND,//i8 // передача отправки команды в драйвер

U2CT\_ANTENNA\_1,//i8 // передача режима работы антенн в драйвер

U2CT\_ANTENNA\_2,//i8

U2CT\_ANTENNA\_3,//i8

U2CT\_ANTENNA\_4,//i8

U2CT\_ANT\_LEVEL\_1,//i8 // передача выходного напряжения антенн в драйвер

U2CT\_ANT\_LEVEL\_2,//i8

U2CT\_ANT\_LEVEL\_3,//i8

U2CT\_ANT\_LEVEL\_4,//i8

U2CT\_TURNS\_ANT\_1,//i16 // передача кол-ва витков в антенне в драйвер

U2CT\_TURNS\_ANT\_2,//i16

U2CT\_TURNS\_ANT\_3,//i16

U2CT\_TURNS\_ANT\_4,//i16

U2CT\_TURNS\_PRIM,//i16

U2CT\_SUPPLY\_LEVEL,//i8 //передача напряжения первичной обмотки в драйвер

U2CT\_STANDBY,//i8 // передача режима ожидания в драйвер

U2CT\_OVERLOAD\_CNT,//i8 // передача количества срабатывания защиты в бу

U2CT\_OVERLOAD\_CURR,//f // передача тока срабатываня защиты в бу

U2CT\_BITNUM,//i8 // передача текущего бита в бу

U2CT\_REPEAT\_CUR,//i8 // передача текущего повтора в бу

U2CT\_IA\_1,//f // передача усредненного значения амплитуды тока в бу

U2CT\_IM\_1,//f // перердача макимального значения амплитуды тока в бу

U2CT\_IA\_2,//f

U2CT\_IM\_2,//f

U2CT\_IA\_3,//f

U2CT\_IM\_3,//f

U2CT\_IA\_4,//f

U2CT\_IM\_4,//f

U2CT\_PA\_1,//f // передача усредненного значения мощности в бу

U2CT\_PM\_1,//f // перердача макимального значения мощности в бу

U2CT\_PA\_2,//f

U2CT\_PM\_2,//f

U2CT\_PA\_3,//f

U2CT\_PM\_3,//f

U2CT\_PA\_4,//f

U2CT\_PM\_4,//f

U2CT\_RA\_1,//f //перердача сопротивления антенны в бу

U2CT\_RA\_2,//f

U2CT\_RA\_3,//f

U2CT\_RA\_4,//f

U2CT\_C\_1,//f // передача расчитанного значения емкости в бу

U2CT\_C\_2,//f

U2CT\_C\_3,//f

U2CT\_C\_4,//f

U2CT\_L\_1,//f // передача расчитанного значения индуктивности в бу

U2CT\_L\_2,//f

U2CT\_L\_3,//f

U2CT\_L\_4,//f

U2CT\_TRANS\_OK,//i8 // передача целостности передатчика в бу

U2CT\_STATE,//i8 // передача состояниня передатчика в бу

U2CT\_DRVENABLE,//i8 // передача состояния режима ожидания в бу

U2CT\_ANT\_FUSE\_1,//i8 // передача аварии по кз в антенне в бу

U2CT\_ANT\_BREAK\_1,//i8 // передача аварии по обрыву в антенне в бу

U2CT\_ANT\_FUSE\_2,//i8

U2CT\_ANT\_BREAK\_2,//i8

U2CT\_ANT\_FUSE\_3,//i8

U2CT\_ANT\_BREAK\_3,//i8

U2CT\_ANT\_FUSE\_4,//i8

U2CT\_ANT\_BREAK\_4,//i8

U2CT\_DRIVER\_FW,//i16 // перердача версии По драйвера в бу

U2CT\_DRIVER\_HW,//i16 // передача версии платы драйвера в бу

U2CT\_SETUP, // передача структуры настроек в драйвер в бу

U2CT\_STATUS, // передача структуры статуса в БУ в бу

// разное

CT\_SETTINGS\_STORE, // не используются в драйвере в бу

CT\_GET\_STATUS // не используются в драйвере в бу

// структура передаваемых значений для типа Queue\_Object\_Data

// размер структуры - 4 байта

typedef union {

uint32\_t u32;

uint16\_t u16;

uint8\_t u8;

int32\_t i32;

int16\_t i16;

int8\_t i8;

int i;

float f;

} Queue\_Object\_Data\_Value;

// структура данных

// Cmd\_Type - 1 байт

// Queue\_Object\_Data\_Value - 4 байта

typedef struct {

// команда

Cmd\_Type cmd;

// данные

Queue\_Object\_Data\_Value value;

} UART2\_Queue\_Data;

// тип принимаемых данных в uart

typedef enum {UART2\_RECV\_CMD, UART2\_RECV\_VALUE} UART2Recv\_t;

Описание основных функций

void TimingCalc(void); // расчет длительностей управляющих импульсов

void SetTiming(void); // установка регистров

void StartPWM(void); // запуск шим

void StopPWM(void); // остановка передачи

void CommandReply(Cmd\_Type cmd, const char fmt, ...); // отправка команды по уарт

void SendPacketUART(void); // отправка статуса

void FormSTATUS(void); // формирование статуса

float CalculateSupply(int ANTlvl, int TRAlvl, int SUPlvl, char ANTnum); // расчет выходного напряжения питания на антенне

float CalculateTransVoltage(uint8\_t I\_forb, uint8\_t Utype); // расчет порога срабатывания защиты по току в антеннах

// вызов метки

void SendData(uint32\_t data); // вызов метки

void ProcData(void); // обработчик вызова

void DiagnSendData(void); // запуск диагностики перед вызовом

// тестирование меток

void TestTag(uint32\_t data); // тест меток

void ProcTestTag(void); // обработчик тестового вызова

void DiagnSendTestTag(void); // запуск диагностики перед тестовывм вызовом

// аварийный вызов

void SendAlarm(uint32\_t data); // запуск аварийного оповещения

void ProcAlarm(void); // обработчик аварийного оповещения

void DiagnSendAlarm(void); // запуск диагностики перед оповещением

void StopAlarm(void); // остановка аварийного оповещения

// вызов команды

void SendComand(uint8\_t comand); // отправка команды в метку (радиус-2)

void DiagnSendComand(void); // запуск диагностики перед отправкой

// вызов команды с номером

void SendComandnData(uint32\_t data, uint8\_t comand); // отправка команды в метку с номером (радиус-2)

void DiagnSendComandnData(void); // запуск диагностики перед отправкой

// парсинг номера в массив

void Parsing(uint32\_t data, uint8\_t command); // преобразования номера в кодированный массив бит

// расчет параметров I,L,R,P,C

float CalculateL(float Ur, float Uf, float Usupp);

float CalculateI(float Ur);

float CalculateR(float Ur, float Uf, float Cap, float L);

float CalculateP(float I, float R);

float CalculateC(float L);

// работа с антеннами

void CheckAntState(float U1r, float U2r, float U3r, float U4r); // проверка состояния антенн

void CalibrateMean(void); // калибровка средней точки

void TransferInterrupt(void); // прерывание передачи для отключения контактора

// работа с внешним генератором

void StartExternPWM(void); // запуск от внешнего генератора

void ExternPWM(void); // обработка сигналов с внешнего генератора

// обработка измерений ацп

void GetVoltage(void); // измерение сигналов с трансформаторов напряжения

void CorrectTIM(void); // корректировка момента измерения

// проведение диагностики антенн

void Diag(void); // диагностика антенн

void ProcDiag(void); // обработка диагностики антенн

// размагничивание (не используется)

void Demagnetization(void); // запуск размагничивания

void ProcDemagnetization(void); // обработка размагичивания

// обработка данных с UART

void HAL\_UART1\_RxCpltCallback(UART\_HandleTypeDef \*huart);

// обработка прерываний

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin);

void SaveSetting(void); // сохранение настроек

void LoadSetting(void); // загрузка настроек

void Generator(void); // запуск генератора

Описание структуры программы интеллектуального драйвера:

Программа состоит из основных модулей, таких как формирование сигналов для управления транзисторным мостом, измерение и расчет параметров антенн, контроль тока в антеннах, запуск периодического контроля параметров антенн, контроль состояния защиты по току, синхронизация запуска команд, обработка кнопок управления, обработка команд управления по UART.

Основное управление Интеллектуальным драйвером осуществляется через интерфейс UART2 путем обработки команд от блока управления в обработчике HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart). В зависимости от типа UART2\_RecvType команды происходит выполнение соответствующих функций и установка значений переменных.

Формирование сигналов управления транзисторным мостом происходит при помощи четырех каналов таймера TIM1. Перед запуском таймера производится расчет и настройка делителей частоты и регистров сравнения для задержек между влечениями транзисторов с заданной частотой и скорость передачи данных. Настройка параметров производится в следующих функциях в зависимости от команды: void SendData(uint32\_t data), void TestTag(uint32\_t data), void SendAlarm(uint32\_t data), SendComand(uint8\_t comand), SendComandnData(uint32\_t data, uint8\_t comand), Diag(void), Demagnetization(void), Generator(void). Тип команды задается переменной uint8\_t TransMode, которая задает режим работы передатчика. В этих функциях рассчитывается длительность паузы, перекрытия и длительности импульса в системных тиках таймера. Происходит включение прерывания от синхроимпульса для синхронного запуска передачи. Запускается таймаут длительностью 1000 мс и 500 мс, основанный на системном таймере и обрабатывающийся в обработчике HAL\_SYSTICK\_Callback(void) в ожидании синхроимпульса. Если поступает синхроимпульс или истекает таймаут выполняется дальнейшая работа формирователя сигналов управления транзисторным мостом. Запускается таймер TIM6, по прерываниям которого вызываются функции расчета и установки значений регистров таймера TIM1 согласно передаваемым данным в зависимости от режима - ProcData(void), ProcTestTag(void), ProcAlarm(void), ProcDiag(void), ProcDemagnetization(void). Конкретный расчет значений регистров TIM1 осуществляется в функции TimingCalc(void), установка значений в регистры SetTiming(void). Запуск таймера производится функцией StartPWM(void), где происходит запись управляющих регистров, останов таймера функцией StopPWM(void). Смена частоты модуляции зависит от предачи “0” и “1”. Длительность передачи зависит от типа кодировки и количества повторов. Вызываемый номер преобразуется в последовательность бит в массив Data[MAXDAT]. Размерность массива определена 256 битами. По окончанию передаваемых данных останавливается работа передатчика командой StopPWM(). После каждого преданного бита формируется структура статуса FormSTATUS(void), и предается в БУ функцией SendPacketUART(void).

Измерение параметров антенн происходит после запуска функции Diag(void). В этой функции задаются параметры таймера TIM1 для формирования одиночного колебания. По прерываниям первого канала таймера TIM1 происходит вызов функций корректировки таймера TIM7 CorrectTIM(void). В этой функции изменяются значения регистров сравнения и по прерываниям таймера TIM7 происходит вызов функции GetVoltage(void) в моменты переключения транзисторных ключей. В этой функции происходит измерение напряжения сигналов с трансформаторов напряжения по каналам ADC1\_CH1, ADC2\_CH3, ADC3\_CH1, ADC4\_CH3. По измеренным значениям производится расчет функциями CalculateL(float Ur, float Uf, float Usupp), CalculateI(float Ur), CalculateR(float Ur, float Uf, float Cap, float L), CalculateP(float I, float R), CalculateC(float L). Для более точного измерения параметров индуктивности и сопротивления выполняется усреднение 15-ти измеренных значений сигналов с трансформатора напряжения. Измерения производятся с частотой 15 Гц. После 15-ти измерений производится вычисление параметров L и R и определение обрыва или короткого замыкания в антенне. Во время передачи основных измерение напряжения сигналов с трансформатора напряжения осуществляется так же по прерываниям таймера TIM7, только производится расчет тока и мощности функциями CalculateI(float Ur) и CalculateP(float I, float R) с дальнейшим усреднением значений за время передачи одного бита, а затем формируются значения статуса и отправляются в БУ.

Во время передачи основных сигналов ведется контроль тока в антеннах, путем усреднения значений сигналов с трансформатора напряжения за период накопления mean\_value и производится сравнение с порогами UI\_fuse[4] и UI\_break[4]. При выходе значения тока за пороговое значение определяется тип аварии в антенне в статусе STATUS.ant\_fuse[4] и STATUS.ant\_break[4]. И производится прерывание передачи TransferInterrupt(void) для отключения контакторов с последующим перезапуском на оставшихся антеннах или остановка передачи StopPWM(void).

Для точного измерения параметров антенн каждый запуск диагностики происходит пересчет значения средней точки CalibrateMean(void) с последующим вычитанием её из полученных значений.

Периодический контроль антенн запускается по прерывания таймера TIM16 в зависимости от настроенного периода диагностики антенн. Таймер TIM16 срабатывает каждую секунду с инкрементированием счетчика, и останавливается только на время передачи основных сигналов. Так же каждую секунду производится отправка переменной статуса trans\_ok в БУ. По таймеру TIM16 ведется задержка в 1 секунду между диагностикой и передачей основных сигналов, рассчитывается время паузы между аварийным оповещением.

Обработка нажатия кнопок ведется в обработчике прерываний HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin). При нажатии на кнопку старт происходит вызов функции DiagnSendData(void), StartExternPWM(void) и Generator(void) в зависимости от режима работы которая запускает процедуру формирования сигналов передачи. Кнопка сброс останавливает передачу вызовом функции StopPWM(void) и запускает таймер TIM15, который сбрасывает триггер защиты.

В обработчике прерываний ведется ожидание синхроимпульса для синхронного запуска двух и более передатчиков. При получении команды на запуск передатчика запускается таймаут на системном таймере длительностью 1000 мс и включается разрешение прерывания по каналу синхронизации. При получении прерывания, останавливается таймер и запускается таймер TIM6 для запуска команды Diag(void).

Обработка команд с БУ по UART ведется в обработчике HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart). Обработчик вызывается системными прерываниями UART. Обмен данными выполняется путем отправки структуры типа UART2\_Queue\_Data. В исходном состоянии приемник UART ожидает прием команды cmd. После приема команды ведется прием 4 байт с данными. В отдельных случаях, при получении команды U2CT\_SETUP ожидается прием целой структуры SETUP. Передача данных осуществляется таким же способом. Сначала передается команда cmd, затем 4 байта данных. При отправке команды U2CT\_STATUS отправляется вся структура STATUS.