Структура программы приемника версии 0.7

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

Исполнительный файл проекта - reciev.07.uvprojx

Файл конфигурации периферии микроконтроллера - reciev.07.ioc

Для взаимодействия с периферией МК использована библиотека HAL.

Основные программные модули:

main.c – инициализация периферии и переменных

match\_dsp.c – основные функции программы

match\_dsp.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:

Список команд для UART

SET\_NAME\_HPT, // номер метки

SET\_BODRATE, // бодрейт

SET\_REPEATNUM, // количество повторов

SET\_SAMPLENUM, // установка значения выборок на бит

SET\_F1S1, // частота 1 секция 1

SET\_F1S2, // частота 1 секция 2

SET\_F2S1, // частота 2 секция 1

SET\_F2S2, // частота 2 секция 2

SET\_FLP, // частота ФНЧ

SET\_DET\_THRES, // порог обнаружения

SET\_F1, // установка значения f1

SET\_F2, // установка значения f2

SET\_SN, // установка серийного номера

SET\_FW, // установить версью ПО

START\_DTR\_FLP, // запуск передачи данных с фнч

START\_DTR\_F1, // запуск передачи данных с f1

START\_DTR\_F2, // запуск передачи данных с f2

START\_DTR\_IN, // запуск передачи данных с inp

STOP\_DTR, // остановка передачи данных

GET\_SETUP, // передача структуры с настройками

GET\_F1, // передача частоты f1

GET\_F2, // передача частоты f2

GET\_BR, // передача бодрейта

GET\_DET\_THRES, // передача порога обнаружения

GET\_NAME\_HPT, // передача номера

GET\_SN, // получить серийный номер

GET\_FW, // получить версию ПО

GET\_LAST\_CALL, // получить последний номер

GET\_LAST\_CALL\_TYPE, // получить тип последнего вызова

SEND\_CALL\_NAME, // передача полученного номера

SEND\_CALL\_TYPE, // передача типа вызова

SEND\_DTR\_FLP, // передача данных с фнч

SEND\_DTR\_F1, // передача данных с f1

SEND\_DTR\_F2, // передача данных с f2

SEND\_DTR\_IN, // передача данных с inp

DEF\_SETT, // сброс на заводские настройки

CONNECT, // запрос подключения

OK, // ответ ОК

ERR // ответ ERR

Структура настроек SETUP

uint16\_t hpt\_name; // номер метки

uint16\_t samplenum; // число выборок на один бит (BR\*3200)

uint8\_t repeatnum; // количество повторов

uint16\_t f1; // установленная частота f1 в гц

uint16\_t f2; // установленная частота f2 в гц

uint8\_t BR; // установленный бодрейт в 10\*бит/с

uint32\_t serailnum; // серийный номер

uint16\_t firmware; // версия по

float cf1s1[5]; // частота 1 секция 1

float cf1s2[5]; // частота 1 секция 2

float cf2s1[5]; // частота 2 секция 1

float cf2s2[5]; // частота 2 секция 2

float cflp[5]; // частота ФНЧ

float ratio; // порог обнаружения

Структура с типом принимаемых данных UART

{UART2\_RECV\_CMD, UART2\_RECV\_VALUE} UART2Recv\_t

Структура для работы программного UART

uint8\_t tim\_en; // запуск таймера

uint8\_t tim\_cnt; // счетчик

uint8\_t rx\_cnt; // счетчик принятых бит

uint8\_t err\_cnt; // счетчик ошибочных байт

uint8\_t rx\_buff[30]; // приемный буфер для бит

uint8\_t rx\_data[3]; // приемный буфер для байт

uint8\_t rx\_data\_cnt; // счетчик принятых байт

uint16\_t rx\_tmp; // временный буфер для данных

Описание основных функций и переменных.

Описание функций

float IIR\_SOS(float in, float \*coef, float \*his); // рекурсивный фильтр второго порядка

void blink(char mode); // запуск моргания

void DefaultSettings(void); // сброс настроек на заводские

void SaveSetting(SettingParametrs\_t \*Settings); // сохранение настроек

void LoadSetting(SettingParametrs\_t \*Settings); // загрузка настроек

void HPT\_Transmite(uint8\_t type); // ответ HPT метке

void S\_UART(void); // обработчик програмного UART

void StartSUART(void); // запуск програмного UART

uint16\_t dataBuff(uint8\_t data); // кольцевой буфер

Описание переменных и констант

// режим моргания фонаря

#define START 0 // стартовый

#define ALARM 1 // авария

#define PERSONAL 2 // персональный вызов

#define OK\_SET 3 // настройка

// тип запроса для НРТ метки

#define CONFIRM 0 // 195 ms

#define REQUEST 1 // 395 ms

#define TEST 2 // 590 ms

#define ENABLE 3 // включение HPT

// стартовый адрес для сохранения

#define ADR\_START 0x08080000

#define FLASH\_WORD\_SIZE 4UL // размер сохраняемого слова

// расчет размера структуры

#define SETTING\_RECORD\_SIZE ((sizeof(SettingParametrs\_t) + FLASH\_WORD\_SIZE - 1) & (~(FLASH\_WORD\_SIZE - 1)))

// размер структуры настроек в словах

#define SETTING\_WORDS (SETTING\_RECORD\_SIZE / FLASH\_WORD\_SIZE)

#define N 16 // число принимаемых бит

#define ALERT\_H 43690 // аварийный номер

#define ALERT\_L 21845 // аварийный номер

#define TEST\_H 65280 // тестовый номер

#define TEST\_L 255 // тестовый номер

extern uint32\_t buff1; // буфер ацп

static float retf11,retf12,retf21,retf22,retfl, buff0, buffl0; // фильтры

static uint8\_t numbit = 0; // счетчик бит

static uint16\_t j; // счетчик выборок

static uint16\_t bin; // буфер детектируемого номера

// обработка бит

uint16\_t bit\_1 = 0; // количество выборок на '1'

uint16\_t bit\_0 = 0; // количество выборок на '0'

uint16\_t bit\_n = 0; // количество выборок на сигал в неопределенной зоне

uint16\_t binn; // детектированый номер

\_Bool det = 0; // флаг начала детектирования

\_Bool numb[N]; // массив детектируемого номера в битах

static uint8\_t nop = 0; // число пропущеных бит

uint16\_t blink\_cnt = 0; // счетчик для повтора моргания

Описание структуры программы приемника:

Программа состоит из основных модулей, таких как цифровой приемник, управление двумя ключами для моргания фонарем, передача импульсов для метки НРТ, трансляция сигналов управления морганием фонарем с метки НРТ, прием UART с НРТ, обработка команд управления по UART.

При включении в функции main() происходит инициализация периферии, чтение настроек из EEPROM или если EEPROM чистая, то сброс на заводские настройки. Затем происходит запуск приема UART, запуск АЦП через ПДП, запуск команды на моргание фонарем и запуск основного таймера TIM21.

Цифровой приемник устроен по классической схеме выделения частот и детектирования разностного сигнала. Цифровые фильтры реализованы на биквадратном звене канонической формы второго порядка. Функция для решения разностного уравнения - float IIR\_SOS(float in, float \*coef, float \*his). Фильтры для выделения частот f1 и f2 реализованы с аппроксимацией Чебышева второго рода и четвертого порядка. Фильтр для выделения огибающей реализован с аппроксимацией Баттерворта второго порядка. Частота дискретизации 3200 Гц. Обработка сигналов ведется в обработчике прерывания таймера TIM21 void TIM21\_IRQHandler(void). При срабатывании прерывания таймера берется сигнал с АЦП, которое ведет непрерывное преобразование в режиме ПДП для разгрузки процессора. Сигнал с АЦП заводится одновременно на фильтры f1 и f2. После выделения сигнала из фильтров производится их квадратичное вычитание друг из друга с последующим выделением огибающей в фильтре ФНЧ. Далее сигнал с огибающей обрабатывается детектором.

При превышении порога сигналом, детектор начинает производить децимацию сигнала путем накопления трех параметров bit\_1, bit\_0, и bit\_n. bit\_1 накапливается в случае если сигнал выше порога SETUP.ratio, bit\_0 накапливается в случае если сигнал ниже порога -SETUP.ratio, и если сигнал находится в диапазоне между SETUP.ratio и - SETUP.ratio, то накапливается bit\_n. При достижения счетчика децимации j == SETUP.samplenum, производится принятие решения о принятом бите, ‘1’ или ‘0’. Накопление информации ведется в переменной bin. После накопления 16 бит numbit == N ведется сопоставление принятого номера с заданным номером НРТ или аварийным или тестовым. После сопоставления принятого номера запускается команда моргания фонарем и ответа для НРТ метки.

Управление двумя ключами для моргания фонарем реализовано на таймере TIM2. Запуске команды производится переключение порта в режим ШИМ и запускается функция с типом моргания HAL\_TIM\_PWM\_Start\_DMA. Таймер работает через ПДП для разгрузки процессора. В обработчике прерываний таймера TIM21 void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim) ведется переключение дополнительного канала управления фонарем синхронно с основным.

Для ответа метке НРТ формируются импульсы длительность 195, 395 и 590 мс. Формирование происходит в обработчике основного таймера TIM21 void TIM21\_IRQHandler(void). Импульс длительностью 195 мс отправляется при получении номера, 390 мс отправляется при запросе номера из НРТ метки. Импульс длительность 590 мс при получении тестового сигнала. На время передачи импульса отключаются прерывания по входу НРТ.

Трансляция сигналов управления морганием фонаря и программного UART ведется в обработчике внешних прерываний void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin). Если не было команды на отправку импульса запроса номера НРТ, то при срабатывании прерывания переключаются выходы управления ключами для моргания фонарем. После включения приемника и получения первого прерывания по входу НРТ, запускается счетчик, активирующий отправку импульса запроса номера НРТ. При получении прерывания после импульса запроса вызывается функция StartSUART() запускающая приемник программного UART. Обработчик программного UART S\_UART(void) вызывается в обработчике основного таймера TIM21. В нем ведется счетчик времени и считываются состояния входа НРТ каждые 65,5 мс в течении 30 бит. После считывания 30 бит производится сопоставление стартовых и стоповых бит, и если пакет принят верно, происходит его декодирование в номер и запускается повторный прием, после подтверждения которого в настройках записывается номер НРТ и сохраняется в EEPROM. В случае неудачного приема, выполняется повторный запрос. Если происходит 10 неудачных запросов, то приемник начинает моргать в аварийном режиме без остановки.

Для передачи данных на ПК и приема от него команд используется UART2. Обработка команд 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 или передача данных в зависимости от команды.