450081, Республика Башкортостан, г. Уфа
ул. Шота Руставели, д. 51/1, оф. 104
Тел.: 8 (800) 775-74-70
e-mail: support@a-t-tech.ru
custom-eng.ru

Программный комплекс K15.CPU.H7

Руководство пользователя

ООО «Эй Энд Ти Текнолоджис»
ИНН/КПП 0276918662/027601001

Оглавление

- Введение.

Данное руководство является описанием встроенного программного обеспечения серии K15-CPU (далее система). Руководство описывает функциональные возможности системы и методы работы с ними.

Программирование системы осуществляется при помощи программного обеспечения STM32CubeIDE версии не ниже чем 1.6 на языке программирования C. Программирование и отладка программного обеспечения осуществляется при помощи программатора St-link/v2.

Программное обеспечение поставляется в виде проекта для среды разработки STM32CubeIDE. Также в комплект поставки ПО может входить:

- Файлы веб-интерфейса системы;
- ПО GPboot encoder для формирования файлов обновления прошивок;
- Файлы .GPboot с готовыми файлами обновления прошивок;
- Файл .bin содержащий загрузчик системы;
- Дополнительное ПО, необходимое для работы с системой.

Функциональные возможности системы:

- Автоматическое подключение, поддержание связи, автоматическое получение и запись данных в модули серий K15 и K19, подключенные по шине CAN;
- обмен данными по интерфейсу Modbus RTU slave, Modbus TCP slave (до 3х одновременных подключений);
- обмен данными по протоколу http;
- обмен данными по протоколу Modbus RTU master независимо для 2х COM-портов;
- FLASH память объемом 8МБ для K15-CPU.F4 или 64МБ для K15-CPU.H7, с файловой системой FAT32;
- FTP сервер, подключенный к файловой системе;
- Возможность подключения SD-карты для K15-CPU.H7 объемом до 8ГБ с файловой системой FAT32 и размером кластера 4096Б;
- Наличие EEPROM объемом 8кБ или FRAM объемом 64кБ в зависимости от конфигурации платы контроллера
- Доступ к данным модулей серии K15 в виде типизированной структуры;
- Симуляция значений чтения (отключение обновления отдельных значений);
- Функции колбэков для ключевых событий;
- Тайм-менеджер для создания и управления задачами;
- Три дискретных входа и два дискретных выхода на плате контроллера;
- Часы реального времени;
- Обновление ПО системы и модулей серии K15 и K19 через WEB интерфейс и FTP сервер;

- Структура проекта

Дерево проекта содержит следующие директории и файлы:

1. Core – Базовая функциональность системы (инициализации периферии, обработка интерфейсов, таймеров, задач, операционной системы и тд.);
2. Driver – Содержит драйверы периферии и ядра контроллера;
3. Midlewares – Содержит драйверы операционной системы, файловой системы, FTP сервера и драйверы работы с сетью;
4. UnitLayer – Содержит определения и функциональную часть работы с модулями расширения системы K15, пользовательские функции доступа к перифери;
5. UserConnectionLayer – Содержит пользовательские настройки системы, точку входа пользовательского ПО, колбэки событий системы и примеры работы;
    - UCL_Callback.c – Файл содержит в себе обратную связь по ключевым событиям в системе;
    - UCL_Examples.c – Файл содержит в себе функциональные примеры работы со всеми модулями и периферией системы;
    - UCL.c – Основной файл для пользователя системы. Является точной входа для старта разработки, где поток UserLayerTask является точкой входа;
    - UCL.h – Файл содержит все пользовательские настройки системы;
6. Debug – В комплекте поставки ПО директория отсутствует, так как создается при сборке проекта. При передаче проекта или включении его в систему контроля версий данную директорию включать не рекомендуется, так как не несет в себе не возобновляемой информации;
7. K15_CPU_H7_ARM Attach_to_running target.launch или K15_CPU_F4_ARM Attach_to_running target.launch – Файл конфигурации отладки. Данная конфигурация отладки используется **для подключения к запущенному проекту** без его остановки и перезагрузки;
8. K15_CPU_H7_ARM Debug.launch или K15_CPU_F4_ARM Debug.launch- Файл конфигурации отладки. Данная конфигурация используется для перезапуска контроллера, загрузки ПО в контроллер и старта отладки с нулевой точки;
9. STM32H743VITX_FLASH.ld или STM32F427VGTX_FLASH.ld – Файл конфигурации областей памяти контроллера.

Исполнение проекта начинается с файла main.c. В функции main() производятся настройки и инициализация устройства. После инициализации запускается ОСРВ FreeRTOS. Происходит переход в поток UserLayerTask, в котором исполняется код пользовательских настроек после чего поток ожидает завершения инициации всех модулей на шине CAN.

```
/**
 * @brief основной поток для реализации приложения пользователя
 */
void UserLayerTask(void const * argument){
  TickType_t xLastWakeTime = xTaskGetTickCount();
  UL_MB_RTU_SLAVE_CONFIG(9600,UART_STOPBITS_1,UART_PARITY_NONE,17,true);
  UL_MB_TCP_SLAVE_CONFIG(10,true,10000);
  UL_COM1_CONFIG(9600,UART_STOPBITS_1,UART_PARITY_NONE,500);
  UL_COM3_CONFIG(9600,UART_STOPBITS_1,UART_PARITY_NONE,500);
  while(!UL_WorkMode)osDelay(1);
  for(;;){
            //циклическое исполнение пользовательского кода с интервалом, указанным ниже
    vTaskDelayUntil(&xLastWakeTime,25);
  }
}
```

Изменение состояния UL_WorkMode происходит по колбэк функции ModulesInitDoneCallback() в файле UCL_callback.c.

- Пользовательские настройки системы

В файле UCL.h находятся все необходимые базовые настройки системы:

1. MB_SLAVE_HOLDINGS_COUNT - максимальное количество тегов MODBUS slave holdings;
2. MB_SLAVE_INPUTS_COUNT - максимальное количество тегов MODBUS slave inputs;
3. MB_SLAVE_COILS_COUNT - максимальное количество тегов MODBUS slave coils;
4. MB_SLAVE_DISCRETE_COUNT - максимальное количество тегов MODBUS slave descrete;
5. NETWORK_USE_DHCP - флаг использования протокола DHCP или статического IP;
6. NETWORK_HTTP_PORT - порт подключения к HTTP;
7. NETWORK_MODBUS_TCP_PORT - порт подключения к MODBUS TCP slave;
8. NETWORK_FTP_PORT - порт подключения к FTP;
9. IP_ADDR0 - IP_ADDR4 - статический IP адрес;
10. NETMASK_ADDR0 - NETMASK_ADDR4 - статический адрес подсети;
11. GW_ADDR0 - GW_ADDR4 - статический адрес шлюза;
12. GENERAL_MODULES_NUMBER – общее максимальное количество подключаемых модулей.

Также в файле представлены пользовательские настройки каждого доступного типа модулей расширения серий K15 и K19, далее представлены настройки на примере модуля K15_DI16, для остальных модулей настройки подобны:

1. MODULE_K15_DI16_MODULE_NUMBER – максимальное количество подключаемых модулей K15_DI16. Максимальное значение – 8. Если тип модуля не используется в система, можно указать значение 0 для экономии ОЗУ, тогда в процессе компиляции работа с данным типом модуля будет полностью вырезана;
2. MODULE_K15_DI16_USE_SIMULATION – если указано не нулевое значение, то в процессе компиляции будет учитываться возможность программной симуляции отдельных значений модуля. Указав 0 – возможность симуляции будет отключена;
3. MODULE_K15_DI16_PDO_DELAY – интервал в мс для отправки значений управление/настроек в модули данного типа. Данные значения находятся в структуре Write типов модулей, где предусмотрен автоматический метод записи данных;
4. MODULE_K15_DI16_SDO_INP_DELAY – интервал в мс для чтения данных только для чтения, которые не обновляются автоматически. Для каждого типа модуля указаны конкретные значения, которые обновляются при данном событии.
5. MODULE_K15_DI16_SDO_HOL_DELAY – интервал в мс для чтения данных для чтения/записи, которые не обновляются автоматически. Для каждого типа модуля указаны конкретные значения, которые обновляются при данном событии.

- Данные модулей и выбор адреса модуля

Проинициализировав все модули, присутствующие на шине CAN, устройство переходит в рабочий режим. С данного момента можно читать и записывать данные модулей. Каждый подключенный модуль представляет собой типизированную структуру:

```
typedef struct{
  ///информация о номере модуля и рабочем состоянии
  ModuleInfo_str ModuleInfo;
  struct{
     ///прочитанные данные
   struct{
      ///статусы
   }Status;
  }Read;
  struct{
     ///данные автоматической записи.
  }Write;
  ///системная информация
  ModuleSysData_str System;
  struct{
     ///флаги симуляции значений.
  }Simulation;
} str_module;
```

Предусмотрены следующие типы структур (где x – индекс структуры модуля):

- K15_AI6[index]. Структура модулей K15-AI6.

- K15_AI8[index]. Структура модулей K15-AI8.

- K15_AO2[index]. Структура модулей K15-AO2.

- K15_DI16[index]. Структура модулей K15-DI16.

- K15_DO16[index]. Структура модулей K15-DO16.

- K15_DO8pwr[index]. Структура модулей K15-DO8pwr.

- K15_NAM[index]. Структура модулей K15-NAM.

Для однозначного и обращения к конкретному модулю и его структуре, предусмотрены макросы в заголовочных файлах модулей, означающие физический номер модуля, выставленный при помощи DIP переключателя (в старых версиях модулей возможно наличие ротационного переключателя или отсутствие любого переключателя, тогда физический номер задан жестко в прошивке модуля):

- K15_AI6_M0 .. K15_AI6_M7 - файл UL_K15_AI6.h;
- K15_AI8_M0 .. K15_AI8_M7 - файл UL_K15_AI8.h;
- K15_AO2_M0 .. K15_AO2_M7 - файл UL_K15_AO2.h;
- K15_DI16_M0 .. K15_DI16_M7 - файл UL_K15_DI16.h;
- K15_DO16_M0 .. K15_DO16_M7 - файл UL_K15_DO16.h;

- K15_DO8pwr_M0 .. K15_DO8pwr_M7 - файл UL_K15_DO8pwr.h;
- K15_NAM_M0 .. K15_NAM_M7 - файл UL_K15_NAM.h;

Где M0..M7 – Выбранный адрес DIP переключателя, расположенный на задней стенке каждого модуля. Стоит помнить, что положение DIP переключателя может быть выбрано произвольным образом, но не должно быть двух и более одинаковых адресов переключателя для нескольких модулей одного типа (при обнаружении 2х модулей одного типа с одинаковыми физическими адресами, один из них будет отправлен в ошибку до перезагрузки). Таким образом, Подключив модуль, например, K15-NAM, установив адрес на DIP переключателе равным 4, обращение к нему будет выглядеть как: K15_NAM[K15_NAM_M4]. Также, некоторые модули поставляются без DIP переключателя, тогда номер модуля зафиксирован в его прошивке.

При потере связи с модулем произойдет переход в колбэк функцию ModulesLostConnectionCallback, в которую будет передан тип и индекс модуля с которым потеряна связь.

Данные модулей обновляются несколькими способами. Наиболее важные данные модулей обновляются асинхронным методом и передаются на K15-CPU при их изменении или 1 раз в секунду, если данные остаются неизменными. Не критичные данные обновляются синхронным методом с заданным интервалом времени. Передача данных на модули осуществляется синхронным методом для критичных данных с возможность досрочного вызова передачи. Также для некоторых модулей предусмотрены дополнительные функции с целью упрощения обмена данными.

- **Описание работы с модулями K15-AI6**

Модуль K15-AI6 предназначен для чтения токового сигнала -24 - 24 мА или напряжения -10 – 10 В. Содержит 6 каналов.

Настройки работы с модулями K15-AI6 представлены в файле UCL.h

```
///максимальное количество подключаемых модулей K15_AI6. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_AI6_MODULE_NUMBER          1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_AI6_USE_SIMULATION         1
///интервал в мс для передачи данных структуры Write
#define MODULE_K15_AI6_PDO_DELAY              1000
///интервал в мс для обновления температуры модуля, флагов включения передачи АЦП,
статусов
#define MODULE_K15_AI6_SDO_INP_DELAY          1000
```

MODULE_K15_AI6_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-AI6 будет удалено при компиляции проекта. При попытке подключения, модули K15-AI6 будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_AI6_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-AI6. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_AI6_PDO_DELAY – интервал между циклическими отправками данных для записи из структуры Write в мс.

MODULE_K15_AI6_SDO_INP_DELAY - интервал обновления данных температуры модуля, флагов включения передачи АЦП, статусов в мс.

Типизированная структура данных модулей AI6 представлена представлены в файле UL_K15_AI6.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    uint16_t          Adc[6];
    float             Data[6];
    UL_AI6_Source     Source[6];
    UL_AI6_MeasType   Type[6];
    float             Temp;
    UL_AI6_ADC_Status ADCEnable;
    struct{
      uint16_t CAN, Firmware, Hardware;
```

```
  }Status;
 }Read;
 struct{
  UL_AI6_MeasType    Type[6];
 }Write;
 ModuleSysData_str System;
 #if UL_AI6_USE_SIMULATION > 0
  struct{
   uint8_t Adc[8];
   uint8_t Data[8];
  }Simulation;
 #endif
} K15_AI6_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные Adc, Data, Source и Type передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс. Данные data представлены в float формате, приведенные к мА. Данные Adc изначально отключены для обновления, для сокращения трафика на шине CAN. Для включения и отключения обновления данных Adc необходимо воспользоваться функциями:

```
K15_AI6_ADC_Enable(strIndex);
K15_AI6_ADC_Disable(strIndex);
```

Где strIndex – K15_AI6_M0 .. K15_AI6_M7.

Данные Source – это текущий источник сигнала напряжение или ток. Данный параметр только для чтения и источник задается изменением dip переключателя на крышке модуля.

Данные Type – это тип измерения биполярный или униполярный. При биполярном типе измерения производятся измерения -24 – 24 мА или -10 – 10 В. В униполярном режиме отрицательные значения не измеряются, но увеличивается точность дискретизации. Значение Type задается в структуре Write. Для изменения типа необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Temp (значение float, градусы Цельсия) и данные статусов принимаются синхронным образом, изначально раз в 1000мс.

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_AI6_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
 }
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_AI6_M0 .. K15_AI6_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

K15_AI6_GetStructIndex(uint8_t moduleIndex)

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_AI6_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Описание работы с модулями K15-AI8

Модуль K15-AI8 предназначен для чтения токового сигнала 4-24 мА. Содержит 8 каналов.

Настройки работы с модулями K15-AI8 представлены в файле UL_K15_AI8.h

```
///максимальное количество подключаемых модулей K15_AI8. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_AI8_MODULE_NUMBER          1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_AI8_USE_SIMULATION         1
///интервал в мс для обновления температуры модуля, флагов включения передачи АЦП, статусов
#define MODULE_K15_AI8_SDO_INP_DELAY          1000
```

MODULE_K15_AI8_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-AI8 будет удалено при компиляции проекта. При попытке подключения, модули K15-AI8 будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_AI8_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-AI8. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_AI8_SDO_INP_DELAY - интервал обновления данных температуры модуля, флагов включения передачи АЦП, статусов в мс

Типизированная структура данных модулей K15-AI8 представлена представлены в файле UL_K15_AI8.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    uint16_t          Adc[8];
    float             Data[8];
    float             Temp;
    UL_AI8_ADC_Status   ADCEnable;
    struct{
      uint16_t  CAN, Firmware, Hardware;
    }Status;
  }Read;
  ModuleSysData_str System;
  #ifdef UL_AI8_USE_SIMULATION
    struct{
      uint8_t Adc[8];
      uint8_t Data[8];
    }Simulation;
```

```
 #endif
} K15_AI8_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные Adc и Data передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс. Данные data представлены в float формате, приведенные к мА. Данные Adc изначально отключены для обновления, для сокращения трафика на шине CAN. Для включения и отключения обновления данных Adc необходимо воспользоваться функциями:

```
K15_AI8_ADC_Enable(strIndex);
K15_AI8_ADC_Disable(strIndex);
```

Где strIndex - K15_AI8_M0 .. K15_AI8_M7.

Temp (значение float, градусы Цельсия) и данные статусов принимаются синхронным образом изначально раз в 1000мс.

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_AI8_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
}
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_AI8_M0 .. K15_AI8_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

```
K15_AI8_GetStructIndex(uint8_t moduleIndex)
```

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_AI8_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Описание работы с модулями K15-AO2

Модуль K15-AO2 предназначен для задания токового сигнала 0 - 24 мА или напряжения -10 – 10 В. Содержит 2 канала.

Настройки работы с модулями K15-AO2 представлены в файле UL_K15_AO2.h

```
/*K15_AO2*/
///максимальное количество подключаемых модулей K15_AO2. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_AO2_MODULE_NUMBER          1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_AO2_USE_SIMULATION         1
///интервал в мс для передачи данных структуры Write
#define MODULE_K15_AO2_PDO_DELAY              100
///интервал в мс для обновления статусов
#define MODULE_K15_AO2_SDO_INP_DELAY          2000
```

MODULE_K15_AO2_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-AO2 будет удалено при компиляции проекта. При попытке подключения, модули K15-AO2 будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_AO2_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-AO2. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_AO2_PDO_DELAY – интервал между циклическими отправками данных для записи из структуры Write в мс.

MODULE_K15_AO2_SDO_INP_DELAY - интервал обновления статусов в мс

Типизированная структура данных модулей AO2 представлена представлены в файле UL_K15_AO2.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    float             Out[2];
    UL_AO2_ChEnable    Enable[2];
    UL_AO2_ChSource    Source[2];
    UL_AO2_ChTestmode   Test[2];
    float             Calb[2];
    struct{
      uint16_t CAN, Firmware, Hardware, Dac1, Dac2, SPI;
    }Status;
  }Read;
  struct{
    float             Out[2];
```

```
  UL_AO2_ChEnable    Enable[2];
  UL_AO2_ChSource    Source[2];
  UL_AO2_ChTestmode  Test[2];
 }Write;
 ModuleSysData_str System;
 #if UL_AO2_USE_SIMULATION > 0
  struct{
   uint8_t  Out[2];
  }Simulation;
 #endif
} K15_AO2_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные Out, Enable, Source и Test передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс.

Данные Out – это текущее, считанное значение токового выхода в мА. Значение Out задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Данные Enable – считанный параметр включения канала, имеет состояния включено и выключено. Значение Enable задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Данные Source – это текущее, считанное значение типа аналогового выхода, имеет состояния ток или напряжение. Значение Source задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Данные Test – это текущее, считанное значение включения тестового режима канала, имеет состояния включено и выключено. При работе тестового режима канал ступенчато меняет ток/напряжение от минимального до максимального. Значение Test задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Для досрочной отправки данных из структуры Write необходимо вызвать функцию

```
K15_AO2_WriteData(strIndex);
```

Где strIndex - K15_AO2_M0 .. K15_AO2_M7.

Для настройки работы каналов, также, предусмотрены функции:

```
K15_AO2_DAC1_control(uint8_t strIndex,
                     float out,
                    UL_K15_AO2_ChEnable enable,
                   UL_K15_AO2_ChSource type);
K15_AO2_DAC2_control(uint8_t strIndex,
                     float out,
                    UL_K15_AO2_ChEnable enable,
                   UL_K15_AO2_ChSource type);
```

Где:

- strIndex – индекс модуля K15-AO2 (значения K15_AO2_M0 .. K15_AO2_M7);
- out – выходное значение. (значения -10 – 10 В для напряжения, 0 – 24 мА для тока);
- Enable – состояние канала (значения UL_K15_AO2_Disable, UL_K15_AO2_Enable);
- Type – тип канала (значения UL_K15_AO2_Voltage, UL_K15_AO2_Current).

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_AO2_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
 }
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_AO2_M0 .. K15_AO2_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

```
K15_AO2_GetStructIndex(uint8_t moduleIndex)
```

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_AO2_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Описание работы с модулями K15-NAM

Модуль K15-NAM предназначен для получения информации с датчиков типа NAMUR. Содержит 4 канала.

Настройки работы с модулями K15-NAM представлены в файле K15_UL_NAM.h

```
/*K15_NAM*/
///максимальное количество подключаемых модулей K15_DI16. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_NAM_MODULE_NUMBER          1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_NAM_USE_SIMULATION         1
///интервал в мс для передачи данных структуры Write
#define MODULE_K15_NAM_PDO_DELAY              100
///интервал в мс для обновления статусов
#define MODULE_K15_NAM_SDO_INP_DELAY          2000
```

MODULE_K15_NAM_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-NAM будет удалено при компиляции проекта. При попытке подключения, модули K15-NAM будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_NAM_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-NAM. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_NAM_PDO_DELAY – интервал между циклическими отправками данных для записи из структуры Write в мс.

MODULE_K15_NAM_SDO_INP_DELAY - интервал обновления статусов в мс

Типизированная структура данных модулей NAM представлена представлены в файле UL_K15_NAM.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    UL_NAM_ChState     In[4];
    UL_NAM_ScState     Sc[4];
    UL_NAM_WbState     Wb[4];
    UL_NAM_Inversion   Inv[4];
    struct{
      uint16_t  CAN, Firmware, Hardware, I2CErr, StatI2C;
    }Status;
  }Read;
  struct{
    UL_NAM_Inversion   Inv[4];
  }Write;
```

```
    ModuleSysData_str System;
    #if UL_NAM_USE_SIMULATION > 0
     struct{
       uint8_t  In[4];
       uint8_t  Sc[4];
       uint8_t  Wb[4];
     }Simulation;
    #endif
}K15_NAM_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные In, Sc, Wb и Inv передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс.

Данные In – это текущее, считанное значение цифрового входа NAMUR с учетом инверсии канала.

Данные Sc – это текущее, считанное значение короткого замыкания цифрового входа NAMUR.

Данные Wb – это текущее, считанное значение обрыва фазы цифрового входа NAMUR.

Данные Inv – это текущее, считанное значение включения инверсии считанного значения, имеет состояния включено и выключено. Значение Inv задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Для досрочной отправки данных из структуры Write необходимо вызвать функцию

```
K15_NAM_WriteData(strIndex);
```

Где strIndex - K15_NAM_M0 .. K15_NAM_M7.

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_NAM_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
 }
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_NAM_M0 .. K15_NAM_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

```
K15_NAM_GetStructIndex(uint8_t moduleIndex)
```

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_NAM_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Описание работы с модулями K15-DI16

Модуль K15-DI16 предназначен чтения состояния цифровых входов, счета импульсов и измерения частоты (для каналов 1 - 4, 9 - 12). Содержит 16 каналов.

Настройки работы с модулями K15-DI16 представлены в файле UL_K15_DI16.h

```
///максимальное количество подключаемых модулей K15_DI16. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_DI16_MODULE_NUMBER         1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_DI16_USE_SIMULATION        1
///интервал в мс для передачи данных структуры Write
#define MODULE_K15_DI16_PDO_DELAY             1000
///интервал в мс для обновления частот на каналах, статусов
#define MODULE_K15_DI16_SDO_INP_DELAY         500
///интервал в мс для обновления данных счетчиков каналов
#define MODULE_K15_DI16_SDO_HOL_DELAY         1000
```

MODULE_K15_DI16_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-DI16 будет удалено при компиляции проекта. При попытке подключения, модули K15-DI16 будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_DI16_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-DI16. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_DI16_PDO_DELAY – интервал между циклическими отправками данных для записи из структуры Write в мс.

MODULE_K15_DI16_SDO_INP_DELAY - интервал обновления частот на каналах, статусов в мс

MODULE_K15_DI16_SDO_HOL_DELAY - интервал обновления данных счетчиков каналов в мс

Типизированная структура данных модулей DI16 представлена представлены в файле UL_K15_DI16.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    UL_DI16_ChState     In[16];
    UL_DI16_ChFilter    Filter[16];
    uint32_t            Counter[16];
    float               Frequency[12];
    struct{
      uint16_t  CAN, Special, Firmware, Hardware;
    }Status;
  }Read;
```

```
    struct{
      UL_DI16_ChFilter   Filter[16];
    }Write;
    ModuleSysData_str System;
    #if UL_DI16_USE_SIMULATION > 0
      struct{
        uint8_t  In[16];
        uint8_t  Counter[16];
        uint8_t  Frequency[12];
      }Simulation;
    #endif
}K15_DI16_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные In и Filter передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс. Данные Counter обновляются с частотой MODULE_K15_DI16_SDO_HOL_DELAY. Данные Frequency обновляются с частотой MODULE_K15_DI16_SDO_INP_DELAY.

Данные In – это текущее, считанное значение цифрового входа канала.

Данные Filter – считанный параметр фильтра шумов на цифровом входе, имеет состояния 100 Гц, 250 Гц, 1000 Гц, без фильтра. Значение Filter задается в структуре Write. Для изменения выходного значения необходимо записать желаемое значение в структуру Write. При следующем цикле отправки, значение обновится на модуле.

Данные Counter – это текущее, считанное значение количества импульсов, зафиксированных на канале. Для сброса значения Counter, необходимо воспользоваться функцией K15_DI16_ClearCounter

Данные Frequency – это текущее, считанное значение частоты изменения импульсов на канале. Частота считается только для каналов 1 – 4 и 9 – 12.

Для досрочной отправки данных из структуры Write необходимо вызвать функцию

```
K15_DI16_WriteData(strIndex);
```

Где strIndex - K15_DI16_M0 .. K15_DI16_M7.

Для досрочного чтения данных Counter, Frequency, необходимо воспользоваться функцией

```
K15_DI16_ReadData(strIndex);
```

Где strIndex - K15_DI16_M0 .. K15_DI16_M7.

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_DI16_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
}
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_DI16_M0 .. K15_DI16_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

K15_DI16_GetStructIndex(uint8_t moduleIndex)

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_DI16_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Описание работы с модулями K15-DO16

Модуль K15-DO16 предназначен записи цифровых выходов выдачи импульсов с заданной частотой и количество импульсов (для каналов 1 – 4 [до 50000 Гц] и 9 – 12 [до 25000 Гц]). Содержит 16 каналов.

Настройки работы с модулями K15-DO16 представлены в файле UL_K15_DO16.h

```
///максимальное количество подключаемых модулей K15_DO16. 0-8. При задании 0 вся работа
///с данным типом модулей будет вырезана при компиляции
#define MODULE_K15_DO16_MODULE_NUMBER         1
///использование симуляции. Установить 0 чтобы отключить для экономии ОЗУ
#define MODULE_K15_DO16_USE_SIMULATION        1
///интервал в мс для передачи данных структуры Write
#define MODULE_K15_DO16_PDO_DELAY             100
///интервал в мс для обновления статусов
#define MODULE_K15_DO16_SDO_INP_DELAY         2000
///интервал в мс для обновления данных счетчиков и частот каналов
#define MODULE_K15_DO16_SDO_HOL_DELAY         700
```

MODULE_K15_DO16_MODULE_NUMBER – максимальное количество подключаемых модулей. Допустимо использование значений 0 – 8. При значении 0 все, что связанно с работой модулей K15-DO16 будет удалено при компиляции проекта. При попытке подключения, модули K15-DO16 будут отправлены в ошибку до перезагрузки. Также, при попытке подключения большего, чем задано, количества модулей, модули с большим значением физического адреса, при превышении максимального количества, будут отправлены в ошибку до перезагрузки.

MODULE_K15_DO16_USE_SIMULATION - если значение больше 0, то доступно использование симуляции значений K15-DO16. Для отключения использования симуляции значение должно быть равно 0.

MODULE_K15_DO16_PDO_DELAY – интервал между циклическими отправками данных для записи из структуры Write в мс.

MODULE_K15_DO16_SDO_INP_DELAY - интервал обновления статусов в мс

MODULE_K15_DO16_SDO_HOL_DELAY - интервал обновления данных счетчиков и частот каналов в мс

Типизированная структура данных модулей DO16 представлена представлены в файле UL_K15_DO16.h

```
typedef struct{
  ModuleInfo_str ModuleInfo;
  struct{
    UL_DO16_OutState    Out[16];
    UL_DO16_ScState     Sc[16];
    float               Frequency[12];
    uint32_t            Counter[12];
    struct{
      uint16_t  CAN, Firmware, Hardware;
```

```
  }Status;
 }Read;
 struct{
  UL_DO16_OutState   Out[16];
 }Write;
 ModuleSysData_str System;
 #if UL_DO16_USE_SIMULATION > 0
  struct{
   uint8_t  Out[16];
   uint8_t  Sc[16];
  }Simulation;
 #endif
}K15_DO16_str;
```

Структура Read представляет собой все данные, чтение которых происходит асинхронно или циклически. Данные Out и Sc передаются асинхронно при их изменении не реже чем 1 раз в секунду, но не чаще чем каждые 20 мс. Данные Counter и Frequency обновляются с частотой MODULE_K15_DO16_SDO_HOL_DELAY.

Данные Out – это текущее, считанное значение цифрового выхода канала.

Данные Sc – это текущее, считанное значение короткого замыкания цифрового выхода канала.

Данные Counter – это текущее, считанное значение оставшегося для вывода количества импульсов на канале. Доступно только для каналов 1 – 4, 9 – 12.

Данные Frequency – это текущее, считанное значение частоты изменения импульсов на канале. Частота считается только для каналов 1 – 4 (до 50000 Гц) и 9 – 12 (до 25000 Гц).

Для досрочной отправки данных из структуры Write необходимо вызвать функцию

```
K15_DO16_WriteData(strIndex);
```

Где strIndex - K15_DO16_M0 .. K15_DO16_M7.

Для досрочного чтения данных Counter, Frequency, необходимо воспользоваться функцией

```
K15_DO16_ReadData(strIndex);
```

Где strIndex - K15_DO16_M0 .. K15_DO16_M7.

Для настройки работы канала предусмотрена функция:

```
K15_DO16_ChSet (uint8_t strIndex,
        UL_K15_DO16_Channels ch,
        float frequency,
        uint32_t data,
        UL_K!5_DO16_OutState state);
```

Функция содержит следующие параметры:

- strIndex – индекс модуля (значения K15_DO16_M0 .. K15_DO16_M7) ;
- ch – канал (значения UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch16);
- frequency – частота выходного сигнала в Гц

- data – параметр, отвечающий за тип выходных данных (дискретный выход, счетный генератор, непрерывный генератор)
- state – выходное состояние (значения UL_K15_DO16_OutLow, UL_K15_DO16_OutHigh)

Функция поддерживает управление каналом для формирования дискретного выходного значения, счетного генератора с определенной частотой, непрерывного генератора с определенной частотой.

Конфигурация функции для задания выходного состояния дискретного выхода:

```
K15_DO16_ChSet(strIndex, ch, 0, UL_K15_DO16_TYPE_OUTPUT, state);
```

Где strIndex - K15_DO16_M0 .. K15_DO16_M7, ch – канал UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch16, state – выходное состояние UL_K15_DO16_OutLow, UL_K15_DO16_OutHigh.

Конфигурация функции для формирования непрерывного генератора определенной частоты:

```
K15_DO16_ChSet(strIndex, ch, freq, UL_K15_DO16_TYPE_GENERATOR, UL_K15_DO16_OutLow);
```

Где index - K15_DO16_M0 .. K15_DO16_M7, ch – канал UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch4 и UL_K15_DO16_Ch9 .. UL_K15_DO16_Ch12, freq – частота канала (для UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch4 максимальная частота 50000Гц, UL_K15_DO16_Ch9 .. UL_K!5_DO16_Ch12 максимальная частота 25000Гц, для остальных каналов режим генератора не предусмотрен).

Конфигурация функции для формирования счетного генератора определенной частоты:

```
K15_DO16_ChSet(strIndex, ch, freq, imp_cnt, final_state);
```

Где index - K15_DO16_M0 .. K15_DO16_M7, ch – канал UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch4 и UL_K15_DO16_Ch9 .. UL_K15_DO16_Ch12, freq – частота канала (для UL_K15_DO16_Ch1 .. UL_K15_DO16_Ch4 максимальная частота 50000Гц, UL_K15_DO16_Ch9 .. UL_K15_DO16_Ch12 максимальная частота 25000Гц, для остальных каналов режим генератора не предусмотрен), imp_cnt – количество импульсов, final_state – состояние выхода после завершения рабы счетного генератора (значения UL_K15_DO16_OutLow, UL_K15_DO16_OutHigh).

Для использования симуляции значений, нужно установить байт необходимого значения в битовом поле в структуре Simulation. После установки байта данное значение не будет обновляться при чтении.

Перед обращением к структуре, рекомендуется проверить состояние модуля

```
if(K15_DO16_Check(strIndex) == UL_OK){
   /// если модуль доступен и работает, попадаем сюда
}
```

Где strIndex – индекс структуры, к которой относятся данные модуля, который можно получить обращение через макросы K15_DO16_M0 .. K15_DO16_M7 (которые означают физический номер модуля)

Если возвращается значение UL_OK, значит модуль подключен и находится в рабочем состоянии.

Для получения индекса структуры strIndex, к которой относится модуль через числовой индекс номера модуля, необходимо использовать функцию

```
K15_DO16_GetStructIndex(uint8_t moduleIndex)
```

Где moduleIndex – физический номер модуля 0 - 7

Данная функция возвращает strIndex модуля. Если модуль с номером moduleIndex обнаружен, то возвращается его strIndex, иначе возвращается индекс MODULE_K15_DO16_MODULE_NUMBER, который является резервным, для индикации ошибки получения strIndex.

- Часы. Дата и время.

В K15-CPU предусмотрены часы реального времени. Обновление значений происходит автоматически после старта работы программы раз в секунду. Значения времени и даты хранятся в значениях стандарта языка си. В указателе на структуру struct tm * UL_PtrTime и time_t UL_CTime. В переменной UL_BatteryChargePsc хранится значения заряда батареи часов в процентах.

Структура tm:

```
struct tm
{
  int tm_sec;                 // Количество секунд 0 — 61
  int tm_min;                 // Количество минут 0 — 59
  int tm_hour;                // Количество часов 0 — 23
  int tm_mday;                // Количество дней сначала месяца 1 — 31
  int tm_mon;                 // Количество месяцев с 1 января 0 — 11
  int tm_year;                // Количество лет с 1900 года.
  int tm_wday;                // Дней недели, начиная с воскресенья 0 — 6
  int tm_yday;                // 0 — 365
  int tm_isdst;       //
};
```

UL_CTime содержит количество секунд начиная с 1 января 1970 года.

Для задания даты и времени предусмотрена функция:

```
UL_TimeWrite (struct tm * ptr_tm);
```

Доступны стандартные функции библиотеки time.h.

Также, синхронизацию данных времени можно произвести через веб-интерфейс системы.

- Цифровые входы и выходы

В K15-CPU предусмотрено три цифровых входа. Значения текущего состояния дискретных входов хранятся в переменных UL_DI1_state,UL_DI2_state,UL_DI3_state. Предусмотрены функции колбэков по изменению состояния дискретных входов в файле UL_callback.c:

```
UL_DI1_front(UL_DI_STATE_TYPE state)
UL_DI2_front(UL_DI_STATE_TYPE state)
UL_DI3_front(UL_DI_STATE_TYPE state)
```

В K15-CPU предусмотрено два цифровых выхода. Значения текущего состояния дискретных выходов можно получить через функции:

```
DO1 = UL_DO1_get();
DO2 = UL_DO2_get();
```

Запись параметров осуществляется через функции:

```
UL_DO1_set(UL_DO_STATE_HIGH);
UL_DO2_set(UL_DO_STATE_LOW);
```

- FatFs. Работа с Dataflash и MicroSD

Файловая система форматируется при первом включении устройства. Если файловая не была обнаружена, производится ее монтирование. Данная операция занимает примерно 10 секунд. О процессе форматирования информирует светодиодная индикация. В процессе форматирования светодиода RUN, STAT, FAULT будут с высокой скоростью моргать.

Файловая система представляет из себя твердотельный Flash накопитель емкостью 8МБ, файловой системой FAT32, размером сектора 512 байт, размером кластера 1024 байт для K15-CPU.F4. Или Flash накопитель емкостью 64МБ, файловой системой FAT32, размером сектора 4096 байт, размером кластера 8192 байт для K15-CPU.H7

Доступ к файловой системе возможен из нескольких источников: FTP, HTTP, DISPLAY, USER. Для исключения попытки одновременного доступа из нескольких источников к файловой системе предусмотрены пользовательские функции распределенного доступа к файловой системе. Распределенный доступ реализован на семафорах операционной системы. Для начала работы с ФС необходимо забрать семафор доступа к ней. Таким образом, единовременно к файловой системе доступ возможен только из одного источника. При попытке доступа из другого источника, поток из которого производится попытка доступа уйдет в ожидание семафора. После завершения работы с ФС первым источником, он отдаст семафор и его возьмет поток, который находился в ожидании.

Второй функцией библиотек распределенного доступа является вынос непосредственного исполнения доступа к ФС в отдельный поток. Это сделано для снижения требований к выделению памяти в потоках которые работают с ФС. Таким образом, пользователю не нужно беспокоиться о необходимости выделения минимально необходимого количества памяти потока для работы с ФС. Что также экономит общее потребление ОЗУ.

Работа с ФС возможна только внутри потока, доступ к ФС в прерываниях и до запуска FreeRTOS не допускается. Прямая работа с FatFS также нежелательна, поскольку существует вероятность попытки одновременного доступа к ФС из нескольких источников, что приведет к ошибкам исполнения.

Функции работы с файловой системой имеют возвращаемый параметр типа UL_RESULT. Возвращаемый параметр может иметь следующие значения: UL_OK, UL_ERROR, UL_FSM_TIMEOUT, UL_FSM_NOT_AVAIL, UL_FSM_NOT_EDIT_MODE, UL_FSM_NO_FILE, UL_FSM_NO_PATH. Более подробный статус исполнения можно получить, обратившись после вызова к параметру FSM.Result. Данный параметр имеет тип FRESULT, описанный в документации на библиотеку FatFS.

Готовность к работе MicroSD карты можно проверить в SD.Ready. Если она не готова, в данном поле будет 0, в ином случае 1. Необходимо использовать накопители с файловыми системами FAT16/FAT32.

Вход и выход из режима редактирования. Для работы с файловой система сначала необходимо войти в режим распределения доступа, указав тип накопителя (FSM_DATAFLASH или SD).

Для этого используется данная конструкция:

```
//заходим в редактирование файлов
if(UL_FSM_FileEditModeStart(FSM_DATAFLASH)==UL_OK)
```

```
{
  //выходим из редактирования
  UL_FSM_FileEditModeStop();
}
```

Все примеры работы с функциями файловой системы (ФС) находятся в файле UCL_Examples.c **в проекте.**

Функции работы с файловой системой.

1. Вход в режим работы с файловой системой. При входе в режим работы с ФС текущая рабочая папка перемещается в корень "/". При выходе из режима рабочая папка перемещается в исходное положение. Стоит помнить, что в режиме работы с ФС остальные источники не будут иметь доступа к ФС. Поэтому долговременно открывать режим не рекомендуется.

```
UL_RESULT UL_FSM_FileEditModeStart();
```

Принимаемый параметр: drive – выбор диска FSM_DATAFLASH или SD.

Возвращаемые значения: UL_FSM_NOT_AVAIL, UL_OK, UL_ERROR, UL_FSM_TIMEOUT.

2. Выход из режима работы с файловой системой.

```
UL_RESULT UL_FSM_FileEditModeStop();
```

Возвращаемые значения: UL_OK.

3. Создание файла.

```
UL_RESULT UL_FSM_FileCreate(const TCHAR* path);
```

Принимаемый параметр: path – путь к файлу.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

4. Открытие файла.

```
UL_RESULT UL_FSM_FileOpen(const TCHAR* path);
```

Принимаемый параметр: path – путь к файлу.

5. Запись файла.

```
UL_RESULT UL_FSM_FileWrite(char * write_data, uint32_t size_in_byte);
```

Принимаемый параметр: write_data – указатель на данные записи. size_in_byte – количество байт записи.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

6. Чтение файла.

```
UL_RESULT UL_FSM_FileRead(uint32_t max_bytes_to_r, char* data);
```

Принимаемый параметр: data – указатель на данные куда будет считываться файл. max_bytes_to_r – максимальное количество байт для чтения.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

7. Закрыть файл.

```
UL_RESULT UL_FSM_FileClose();
```

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

8. Удаление файла или папки

```
UL_RESULT UL_FSM_FileDelete(const TCHAR* path);
```

Принимаемый параметр: path – путь к удаляемому файлу или папке.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

9. Проверка существования файла или папки
   UL_RESULT UL_FSM_FileCheckExistence(const TCHAR* path);

Принимаемый параметр: path – путь к проверяемому файлу или папке.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR, UL_FSM_NO_FILE.

10. Открытие папки для просмотра содержимого (не переход в папку)
    UL_RESULT UL_FSM_FileOpenDir(const TCHAR* path);

Принимаемый параметр: path – путь к папке.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR, UL_FSM_NO_PATH.

11. Чтение содержимого папки
    UL_RESULT UL_FSM_FileReadDir();

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

12. Закрыть папку
    UL_RESULT UL_FSM_FileCloseDir();

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

13. Переход на определенную позицию в файле или выделение непрерывной области памяти для файла.
    UL_RESULT UL_FSM_Seek(uint32_t position);

Принимаемый параметр: position – задание позиции в байтах в файле.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

14. Обрезание файла по текущей позиции
    UL_RESULT UL_FSM_Truncate();

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

15. Переименование или перемещение файла или папки
    UL_RESULT UL_FSM_FileRenameMove(const TCHAR* old_path, TCHAR* new_path);

Принимаемый параметр: old_path – путь к текущему файлу или папке. new_path – новый путь файла или папки.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

16. Создание папки
    UL_RESULT UL_FSM_MakeDir(const TCHAR* dir);

Принимаемый параметр: dir – путь к создаваемой папке.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

17. Сменить папку. Переход в папку.
    UL_RESULT UL_FSM_ChangeDir(const TCHAR* dir);

Принимаемый параметр: dir – путь к папке, в которую надо перейти.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

18. Получить текущий путь
    UL_RESULT UL_FSM_GetCWD(char * buf, uint32_t len_buf);

Принимаемый параметр: buf – указатель на данные, куда копировать путь. len_buf – максимальная длина данных.

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

19. Получить количество свободного места на диске.
    UL_RESULT UL_FSM_GetFree();

Возвращаемые значения: UL_FSM_NOT_EDIT_MODE, UL_OK, UL_ERROR.

20. Получить размер файла
    FSIZE_t UL_FSM_FileGetSize();

Возвращаемые значения: Текущий размер файла в байтах.

21. Получить текущее положение в файле
    DWORD UL_FSM_FileTell();

Возвращаемые значения: текущее положения указателя в файле в байтах.

Также существуют функции, использование которых независимо от выхода в режим работы с ФС.

UL_RESULT FSM_ChangeDataBetween(char* path, char* key_start, char* key_stop, char* data_replace);

Данная функция производит замену текста между двумя ключами. Длина файла может быть произвольной, длина ключей и заменяемого текста – не более 255 символов.

- FTP сервер

В K15-CPU предусмотрено наличие FTP сервера для хранения данных логов, WEB страниц, изображений (при наличии дисплея). Подключение к FTP серверу осуществляется анонимно. Разрешение на доступ к FTP серверу осуществляется путем ввода пароля через web интерфейс (в следующих версиях ПО). Для подключения к FTP серверу рекомендуется использовать ПО Total commander.

- EEPROM и FRAM

В качестве ПЗУ для хранения настроек выступает EEPROM – 8кБ (последние 256 байт зарезервированы системой) или FRAM– 64кБ (последние 256 байт зарезервированы системой) В зависимости от конфигурации платы контроллера. Определение типа подключенной памяти происходит автоматически, также, как и алгоритм чтения/записи для них.

Для просмотра типа подключенной памяти нужно обратиться к структуре System. System.PeriphAvailable.EEPROM хранит флаг подключения микросхемы EEPROM. System.PeriphAvailable.FRAM хранит флаг подключения FRAM. Одновременное наличие обоих типов памяти невозможен. Если ни одна из микросхем не была обнаружена, то флаги наличия периферии для них будут false, а обращение к чтению/записи вернет ошибку.

Чтение/запись данных для EEPROM/FRAM осуществляется одинаковым набором функций. Количество байт чтения/записи на вызов функции не ограничен, но адрес байт чтения/записи во время исполнения не должен превышать 0x1F00 для EEPROM и 0xFE00 для FRAM.

Чтение осуществляется функцией:

```
UL_MemoryRead(uint16_t addr, uint16_t count, uint8_t * data)
```

В качестве параметров передаются стартовый байт, количество байт чтения и указатель на массив чтения. Адресация не должна выходить за рамки 0x1F00 для EEPROM и 0xFE00 для FRAM байта, количество байт для считывания произвольное. Запись производится функцией:

```
UL_MemoryWrite(uint16_t addr, uint16_t count, uint8_t * data)
```

В качестве параметров передаются стартовый байт, количество байт записи и указатель на массив записи. Адресация не должна выходить за рамки 0x1F00 для EEPROM и 0xFE00 для FRAM байта, количество байт для записи произвольное.

- Описание работы MODBUS MASTER

MODBUS MASTER в K15-CPU поддерживает независимую работу по COM1 и COM3. Поддерживает чтение типов данных HOLDINGS, INPUTS, COILS, DISCRETE с циклическими или разовыми запросами. Запись типов данных HOLDINGS, COILS с циклическими или разовыми запросами.

Инициализация COM портов производится при помощи функций:

```
MB_RTU_Master_Setup(COM_ports COM_port,
        uint32_t baudrate,
        uint32_t stopbits,
        uint32_t parity
        uint32_t wiat_empty);
```

где

- COM_port – Конфигурируемый COM порт _COM1_, _COM2_, _COM3_;
- Baudrate – скорость обмена данными от 4800 до 115200 бод;
- Stopbits – количество стоп битов (значения UART_STOPBITS_1, UART_STOPBITS_2);
- Parity – четность (значения UART_PARITY_NONE, UART_PARITY_EVEN, UART_PARITY_ODD).
- wiat_empty – ожидание таймаута в мс.

Инициализацию COM портов и добавление тэгов для MODBUS MASTER рекомендуется производить до перехода к бесконечному циклу потока UserTask и добавления тэгов.

Добавление тэга для работы в MODBUS MASTER:

```
MB_MASTER_ADD_TAG(COM_ports port,
        R_W_type  r_w,
        TAG_type type,
        uint32_t timeInterval,
        uint8_t addr,
        uint16_t pos,
        uint16_t size,
        void * ptrData,
        uint16_t addit,
        uint32_t *ptrOutIndex,
        void (*callback_fn)(MB_MasterData_str *str));
```

Функция содержит следующие параметры:

1. COM_ports port - выбор COM порта адресации тэга (значения _COM1_, _COM2_, _COM3_);
2. r_w – выбор типа доступа к тэгу (значения _READ_ или _WRITE_);
3. type – тип данных тэга (значения _INPUTS_, _HOLDINGS_, _COILS_, _DISCRETE_);
4. timeInterval – временной интервал циклической обработки тэга. Для циклического чтения/записи задается значение в мс, но не менее 200мс. Максимальное значение временного интервала ограничено размерностью 32х битного значения. Для формирования тэга с чтением/записью по запросу значение time_interval должно быть равно 0;

5. addr – адрес устройства MODBUS (значения 1-255);
6. pos – для HOLDINGS и INPUTS – стартовый регистр. Для COILS и DISCRETE – стартовый бит;
7. size – для HOLDINGS и INPUTS – количество регистров для чтения/записи (не более 96 на тэг). Для COILS и DISCRETE – количество бит для чтения/записи (не более 32 на тэг).
8. ptrData – указатель на данные, куда будут записаны считанные значения для чтения или указатель на данные откуда будут взяты значения для записи. Указатель должен быть приведен к безразмерному типу void.
9. Addit – дополнительный параметр. Для HOLDINGS и INPUTS – параметр перестановки байт (значения _NON_REV_, _REV_). Для COILS и DISCRETE – параметр смещения бита относительно указателя на данные. При чтении - стартовый бит по адресу, откуда начнется заполнение считанных данных. При записи указывает на стартовый бит относительно адреса откуда будут взяты биты для записи.
10. ptrOutIndex – указатель на значение uint32, в которое будет записан индекс тэга, для возможности его запуска и остановки.
11. MB_MasterData_str *str – колбэк функция успешного завершения обмена данными. Записать NULL если не нужен.

При добавлении тэга формируется его структура, после чего при достижении заданного временного интервала происходит запрос на чтение или запись. Работа тэгов для COM1 и COM3 независима друг от друга.
Предусмотрены два колбэка для получения информации о неуспешном выполнении тэга, расположенные в файле UL_callback.c.

```
COMFailReadTagCallback(COM_ports port);
```

Переход по данному колбэку осуществляется в случае неполучения ответа от устройства.

```
COMErrorReadTagCallback(COM_ports port,Master_data_str * tag);
```

Переход по данному колбэку осуществляется в случае получения пакета с кодом ошибки или несовпадающим CRC.

Доступно управление работой тэгов. Возможно запустить или остановить исполнение тэга. Для этого при формировании тэга необходимо получить его индекс.

```
MB_MASTER_TAG_STOP(uint32_t index);
```

Функция остановки исполнения тэга. Для остановки исполнения тэга необходимо вызвать функцию с указанием его индекса.

```
MB_MASTER_TAG_START(uint32_t index);
```

Функция запуска исполнения тэга. Используется в случае, если циклический тэг был остановлен функцией MB_MASTER_TAG_STOP или в случае использования тэга, работающего по запросу. В случае использования такого тэга использование функции MB_MASTER_TAG_START вызовет одно исполнение данного тэга, после чего он будет остановлен. В случае использования циклического тэга использование функции MB_MASTER_TAG_START запустит данный тег и его остановка будет доступна по функции MB_MASTER_TAG_STOP.

Дополнительно есть возможность запуска одиночного запроса, без занесения его в базу тегов. Обработка запроса производится единовременно, то есть после входа в функцию, выход из нее произойдет только поле успешного или неуспешного исполнения:

UL_RESULT MB_MASTER_SEND_PACK(COM_ports port,///<выбор COM порта _COM1_, _COM3_
R_W_type r_w, ///<чтение или запись. _READ_, _WRITE_
TAG_type type,///<выбор типа тэга _INPUTS_, _HOLDINGS_, _COILS_, _DISCRETE_
uint8_t addr,///<адрес устройства 1-255
uint16_t pos, ///<Стартовый регистр для _INPUTS_ , _HOLDINGS_. стартовый бит для _COILS_, _DISCRETE_
uint16_t size,///<Количество регистров чтения для _INPUTS_, _HOLDINGS_, Количество бит чтения для _COILS_, _DISCRETE_
void *ptrData,///<указатель, куда будут записываться считанные данные
uint16_t addit);///<доп данные. для _COILS_, _DISCRETE_ это смещение стартового бита указателя ptrData. для _INPUTS_, _HOLDINGS_ перестановка байт _REV_, _NON_REV_

Так же доступно асинхронное чтение без добавления тега в базу. В файле UCL_Examples.c есть пример использования данной функции.

Начать асинхронный запрос тега:

```
MB_MASTER_SEND_PACK_ASYNC(COM_ports port,
                          R_W_type r_w,
                TAG_type type,
                uint8_t addr,
                uint16_t pos,
                uint16_t size,
                void *ptrData,
                uint16_t addit)
```

Параметры:

1. port - выбор COM порта адресации тэга (значения _COM1_, _COM2_, _COM3_);
2. r_w – выбор типа доступа к тэгу (значения _READ_ или _WRITE_);
3. type – тип данных тэга (значения _INPUTS_, _HOLDINGS_, _COILS_, _DISCRETE_);
4. addr – адрес устройства MODBUS (значения 1-255);
5. pos – для HOLDINGS и INPUTS – стартовый регистр. Для COILS и DISCRETE – стартовый бит;
6. size – для HOLDINGS и INPUTS – количество регистров для чтения/записи (не более 96 на тэг). Для COILS и DISCRETE – количество бит для чтения/записи (не более 32 на тэг).
7. ptrData – указатель на данные, куда будут записаны считанные значения для чтения или указатель на данные откуда будут взяты значения для записи. Указатель должен быть приведен к безразмерному типу void.
8. Addit – дополнительный параметр. Для HOLDINGS и INPUTS – параметр перестановки байт (значения _NON_REV_, _REV_). Для COILS и DISCRETE – параметр смещения бита относительно указателя на данные. При чтении - стартовый бит по адресу, откуда начнется заполнение считанных данных. При записи указывает на стартовый бит относительно адреса откуда будут взяты биты для записи.

Функция, возвращающая текущее состояние работы данной функции:

```
MB_MASTER_ASYNC_GET_STATUS()
```

Возвращает одно из следующих состояний:

- UL_PREPARE – тег передан пользователем, запрос не начал обработку
- UL_IN_WORK – тег в обработке
- UL_NO_OPERATION – тега нет

- UL_OK – успешная обработка тега
- UL_ERROR – в случае ошибки обработки

Так как первые два состояния для пользователя примерно равнозначны, вы можете использовать следующую функцию. Она вернёт true в случае, если в данный момент тег находится в обработке (или пока только передан вами в обработку):

MB_MASTER_ASYNC_GET_IS_IN_WORK()

Если для вашего алгоритма будет удобнее указывать отсутствие тега через состояние UL_NO_OPERATION, можете вызвать MB_MASTER_ASYNC_RESET_STATUS() для сброса состояния.

Так же по завершении работы с тегом вызывается callback функция MB_MASTER_ASYNC_ENDED (в файле UCL_Callback.c), куда передаётся структура асинхронного чтения, содержащую всю информацию.

- Описание работы MODBUS SLAVE RTU/TCP

В K15-CPU предусмотрено добавление тегов MODBUS RTU/TCP, предусмотрена работа с типами данных HOLDINGS, INPUTS, COILS, DISCRETE. Настройки COM-порта для MODBUS SLVE RTU:

```
MB_RTU_Slave_Setup(COM_ports COM_port,
        uint32_t baudrate,
        uint32_t stopbits,
        uint32_t parity,
        uint8_t address,
        bool reversal);
```

Параметры функции:

- COM_port – Конфигурируемый COM порт _COM1_, _COM2_;
- baudrate – скорость передачи данных (значения от 4800 до 115200 бод);
- stopbits – количество стоп бит (значения UART_STOPBITS_1, UART_STOPBITS_2)
- parity – контроль четности (значения UART_PARITY_NONE, UART_PARITY_EVEN, UART_PARITY_ODD);
- address – MODBUS адрес K15-CPU (значения 1-255);
- reversal – перестановка байт (значения true, false).

Настройки MODBUS TCP:

```
UL_MB_TCP_SLAVE_CONFIG(uint8_t address,
        bool reversal);
```

Параметры функции:

- address – MODBUS адрес K15-CPU (значения 1-255);
- reversal – перестановка байт (значения true, false).

Функции настройки MODBUS SLAVE рекомендуется вызывать до старта бесконечного цикла потока UserTask и до добавления тегов.

Для каждого типа данных необходимо задать максимальное количество тегов. Размерность задается макросами в файле UCL.h:

```
#define MB_SLAVE_HOLDINGS_COUNT 64
#define MB_SLAVE_INPUTS_COUNT 64
#define MB_SLAVE_COILS_COUNT 64
#define MB_SLAVE_DISCRETE_COUNT 64
```

Каждый вызов функции создания тега увеличивает общее количество тегов определенного типа. Функции формирования для HOLDINGS и INPUTS обладают встроенной оптимизацией тегов, таким образом, количество зарегистрированных тегов может быть меньше количества вызовов функции создания тегов.
Для получения текущего количества тегов определенного типа необходимо после добавления тегов вызвать функцию:

```
MB_SLAVE_GET_TAG_COUNT(TAG_type type);
```

Где type - _INPUTS_, _HOLDINGS_, _COILS_, _DISCRETE_. Функция вернет количество зарегистрированных тегов.

Создание нового тега для HOLDINGS:

```
MB_SLAVE_ADD_HOLDING(uint16_t * addr,
```

```
uint8_t size_reg,
uint16_t reg,
void (*write_callback_fn)(uint16_t reg, uint8_t * data));
```

Параметры:

- addr – указатель, приведенный к типу uint16_t на данные для чтения/записи;
- size_reg – количество регистров (не более 96 на 1 тэг);
- reg – номер регистра;
- void (*write_callback_fn)(uint16_t reg, uint8_t * data)) – указатель на функцию колбэка по записи значения (если не нужен – значение NULL).

Создание нового тега для INPUTS:

```
MB_SLAVE_ADD_INPUT(uint16_t * addr,
                   uint8_t size_reg,
                   uint16_t reg);
```

Параметры:

- addr – указатель приведенный к типу uint16_t на данные для чтения/записи;
- size_reg – количество регистров (не более 96 на 1 тэг);
- reg – номер регистра;

Создание нового тега для COILS:

```
MB_SLAVE_ADD_COIL(void * addr,
                  uint8_t bit,
                  uint16_t pos,
                  void (*write_callback_fn)(uint16_t coil);
```

Параметры:

- addr – указатель, приведенный к типу void на данные для чтения/записи;
- bit – смещение в битах относительно адреса;
- pos – номер бита в MODBUS;
- void (*write_callback_fn)(uint16_t coil) – указатель на функцию колбэка по записи значения (если не нужен – значение NULL).

Создание нового тега для DISCRETE:

```
MB_SLAVE_ADD_DISCRETE(void * addr,
                      uint8_t bit,
                      uint16_t pos);
```

Параметры:

- addr – указатель, приведенный к типу void на данные для чтения/записи;
- bit – смещение в битах относительно адреса;
- pos – номер бита в MODBUS;

- Free protocol

У вас есть возможность использовать COM порты с реализацией собственных протоколов. Для этого предусмотрены следующие функции.

UL_RESULT UL_FREE_CONFIG(COM_ports port, uint32_t baudrate, uint32_t stopbits, uint32_t parity);

Конфигурирование выбранного порта для работы в качестве мастера произвольного протокола

1. port - порт обмена
2. baudrate - скорость обмена
3. stopbits - стоп-бит
4. parity - чётность

UL_RESULT UL_FREE_Put(COM_ports port, uint16_t count, uint8_t * data);

Отправка пакета данных длиной не более 256Б в выбранный порт

1. port - порт обмена
2. count - длина пакета
3. *data - указатель на байтовый массив данных для отправки

UL_RESULT UL_FREE_Get(COM_ports port, uint16_t count, uint8_t * data);

Прием пакета данных длиной не более 256Б по выбранному порту

1. port - порт обмена
2. count - длина пакета
3. *data - указатель на байтовый массив данных для приема

uint16_t UL_FREE_GetCount(COM_ports port);

Получение количества принятых байт в буфере приема по выбранному порту

- port - порт обмена

UL_RESULT UL_FREE_GetFlush(COM_ports port);

Очистка буфера приема по выбранному порту

- port - порт обмена

- менеджер задач

K15-CPU предусматривает возможность создания пользовательских задач, без создания отдельного потока. Создание задач рекомендуется производить до перехода исполнения в бесконечный цикл потока UserLayerTask. Добавление новой задачи возможно при помощи функции:

```
UL_TaskAdd (uint32_t time_interval,
            uint8_t in_work,
            void (*task_callback_fn)())
```

Параметры:

- time_interval – временной интервал вызова функции колбэка задачи с мс;
- in_work – флаг включения в работу (значения true – сразу включен, false – после создания выключен);
- void (*task_callback_fn)() – указатель на колбэк функцию задачи.

Функция создания задачи возвращает индекс созданной задачи, используя индекс задачи ее можно включать и выключать:

```
UL_TaskStop(uint16_t task_index);
UL_TaskStart(uint16_t task_index);
```

В функции передается индекс задачи, при успешном исполнении возвращает значение 1, если задача с заданным индексом не найдена, возвращает 0.

- Мониторинг производительности

В системе доступно получение текущей загрузки ЦП и приблизительная оценка источников загрузки.

Значение SYSTEM.CPULoad.General содержит общий текущий уровень загрузки ЦП в % * 100.

Значение SYSTEM.CPULoad.User содержит текущий уровень загрузки ЦП в % * 100, вызванный пользовательским потоком.

Значение SYSTEM.CPULoad.Tasks содержит текущий уровень загрузки ЦП в % * 100, вызванный потоком созданных задач.

Значение SYSTEM.CPULoad.Master содержит текущий уровень загрузки ЦП в % * 100, вызванный потоком обработки MODBUS master.

- Обновление ПО Контроллера через веб-интерфейс

K15-CPU предусматривает возможность обновления ПО контроллера без использования программатора st-link. Обновление ПО производится при помощи FTP сервера и WEB интерфейса.

Для обновления ПО через веб-интерфейс необходимо:

1. Соединить контроллер K15-CPU с ПК при помощи Ethernet кабеля. На ПК открыть ПО Total commander и осуществить подключение к FTP серверу контроллера по его IP адресу. Перейти в директорию SET. Рекомендуется удалить другие файлы обновления ПО K15_CPU из папки. Скопировать файл обновления K15-CPU с расширением .gpboot в директорию SET. (рисунок 1).
2. Перейти по IP адресу контроллера в браузере (рекомендуется использовать ПО Google chrome). Авторизоваться с учетной записи super_admin (пароль - 12345678).
3. На главной странице в шапке перейти по пункту Обновление (ранее Система) -> Обновление ПО контроллера K15.CPU. На странице обновления контроллера отобразится информация о текущий версии прошивки контроллера и доступные файлы обновления ПО контроллера из директории SET. (Рисунок 2).
4. Выбрать и подтвердить выбор необходимого файла обновления. После чего отобразится информация о процедуре обновления ПО и кнопка отмены обновления.
5. Для старта обновления перезагрузить контроллер. В ходе обновления веб-интерфейс не может предоставлять информацию о ходе исполнения обновления. Процедура обновления индицируется светодиодами на контроллере. Процедура обновления занимает до трех минут и после обновления веб-интерфейс автоматически перейдет на страницу авторизации.
6. После обновления ПО, авторизоваться и снова перейти на страницу обновления ПО контроллера. Убедиться, что информация о текущий версии ПО соответствует новому ПО. (Рисунок 3)

Траблшутинг.

1. Если в процессе копирования кайла на FTP сервер контроллера возникает ошибка «550 action required», необходимо удалить скопированный файл и произвести копирование повторно. Также, рекомендуется после загрузки убедиться, что размер загруженного файла совпадает с размером исходного файла.
2. Если после обновления ПО контроллера информация о ПО не изменилась, значит процедура обновления не прошла успешно. Для исправления проблемы модно удалить все файлы из директории SET, оставив только нужный файл обновления ПО. Попробовать сократить длину имени файла обновления ПО (не более 25 символов, без спец. символов) и загрузить файл обновления ПО повторно.
3. Если не удается перейти на страницу веб интерфейса и выводится окно ЗАПРАШИВАЕМАЯ СТРАНИЦА НЕ ОБНАРУЖЕНА, необходимо загрузить файлы веб-интерфейса, через FTP сервер (пункт Обновление веб-интерфейса K15_CPU)
   -

-

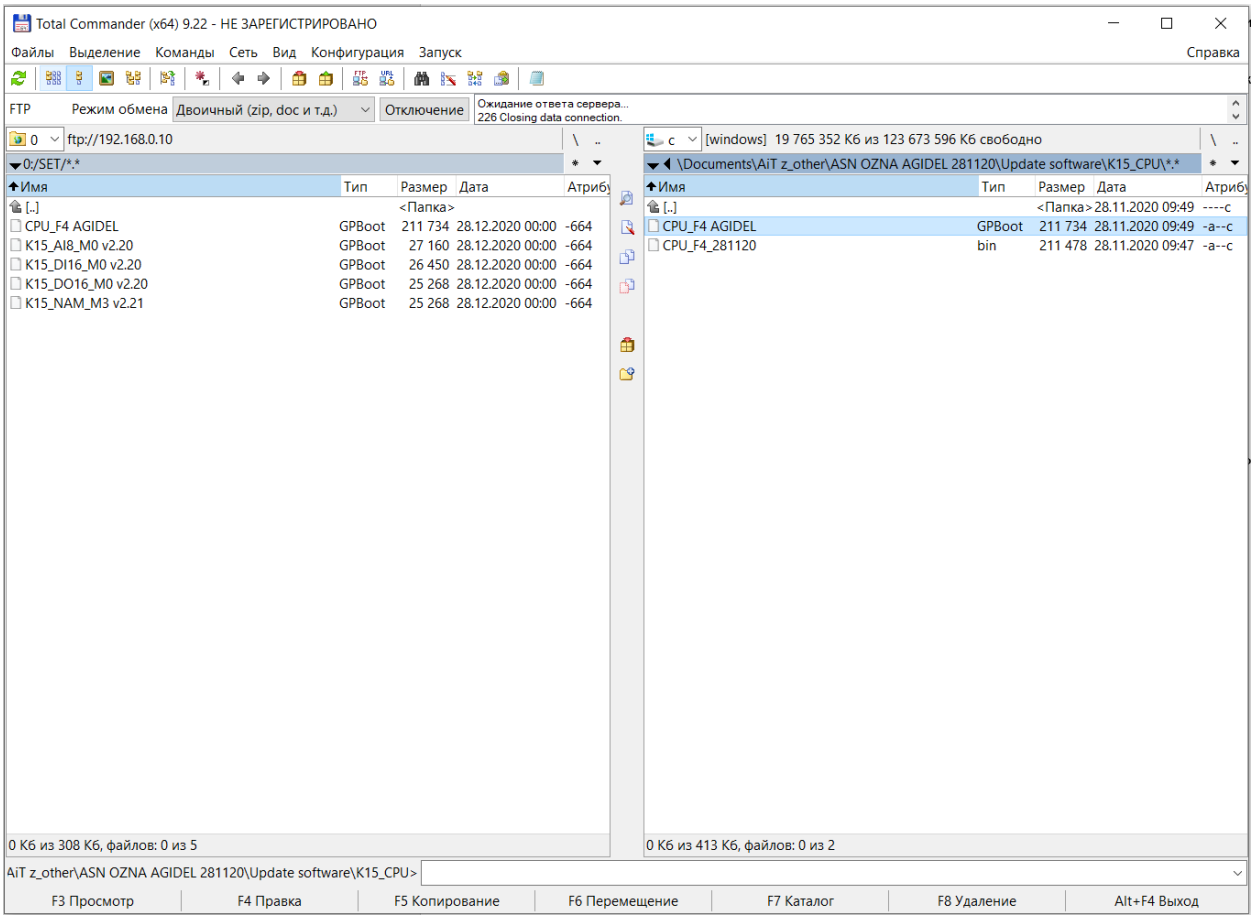

Рисунок 1 Работа с FTP сервером контроллера

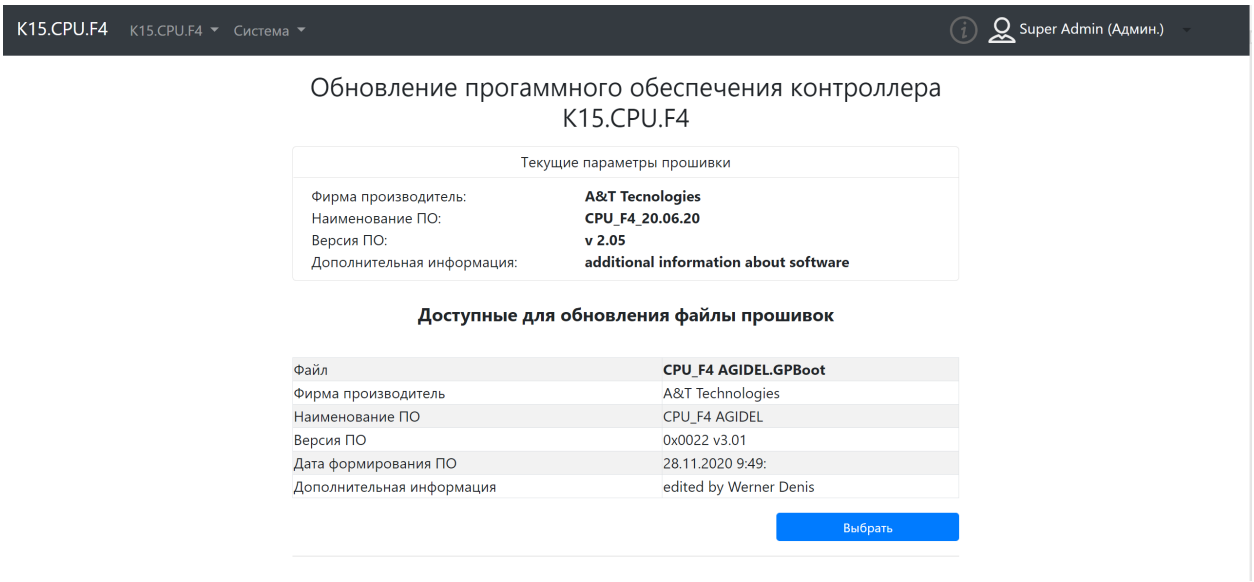

Рисунок 2 Вид страницы обновления ПО контроллера K15

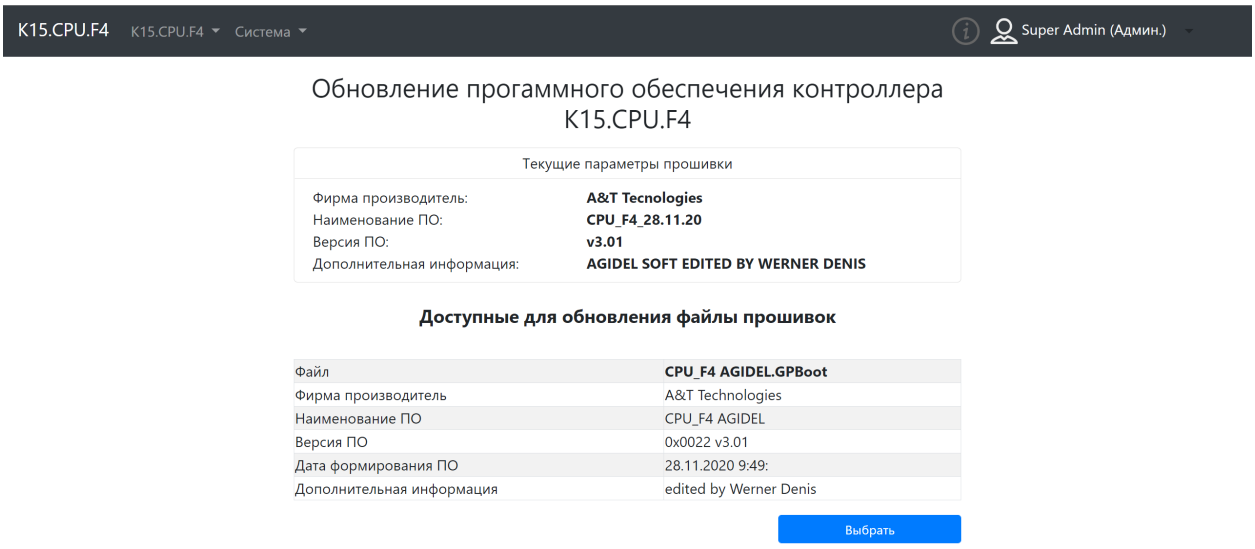

Рисунок 3 Информация о ПО после обновления

- резервное копирование ПО K15-CPU

Для проведения резервного копирования прошивки K15_CPU понадобится ПО ST-Link utility и программатор ST-Link-V2.

1. Подключить программатор к контроллеру через соединительный кабель, который находится под верхней крышкой корпуса. Открыть ПО ST-Link utility и произвести подключение к контроллеру, установив стартовый адрес чтения 0x08000000 и размер чтения 0x8FFFF. (Target->Connect). (Рисунок 4)
2. Сохранить файл прошивки (File->Save file as). После чтения памяти и сохранения должен появиться файл с расширением .bin размером 576 кБ.

Для записи считанного файла прошивки в контроллер необходимо

1. Произвести подключение к контроллеру (Target->Connect).
2. Выбрать «Target->Program…», выставить стартовый адрес 0x08000000, выбрать считанный файл прошивки .bin и нажать «start». (Рисунок 5)

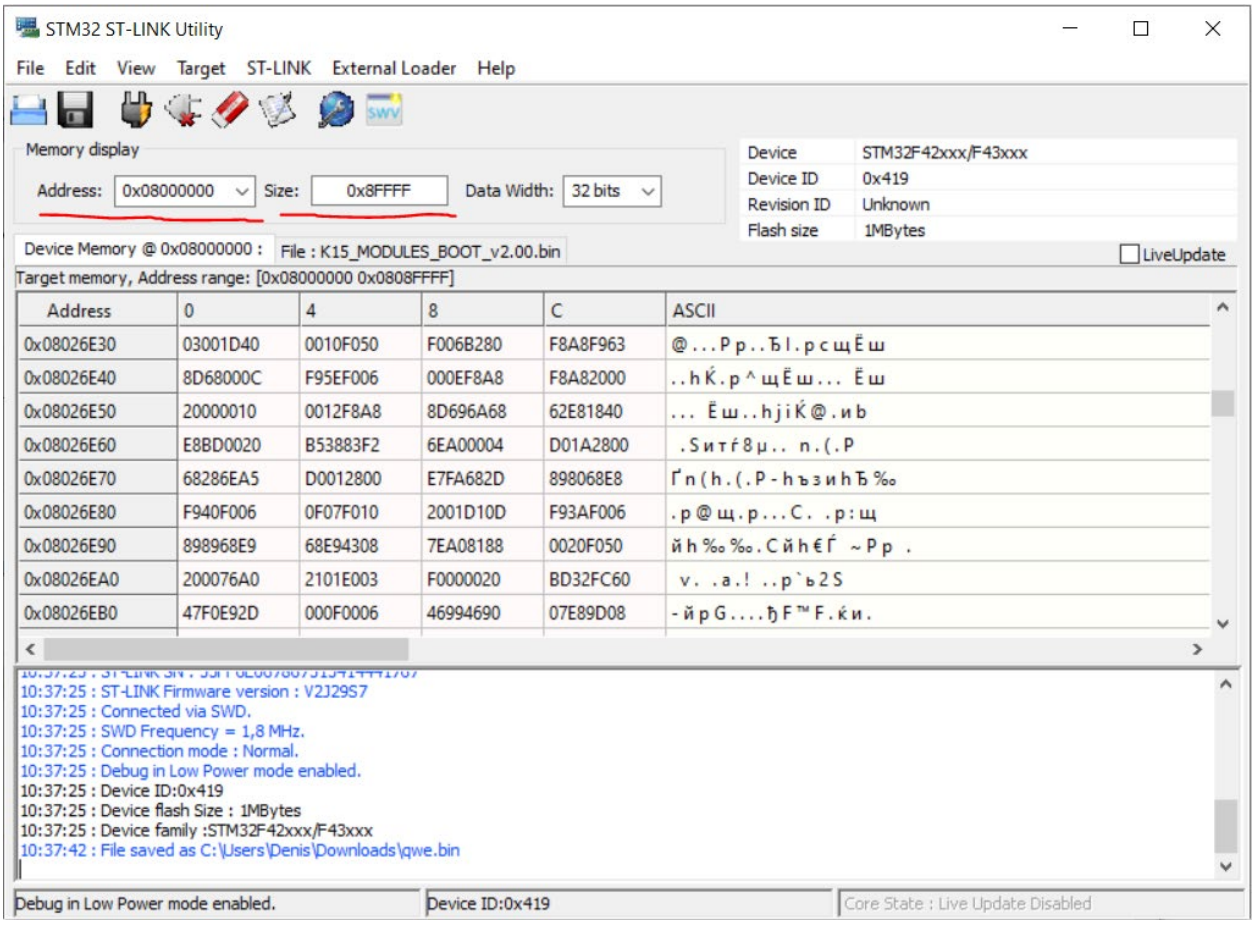

Рисунок 4 Указание адреса чтения и размера

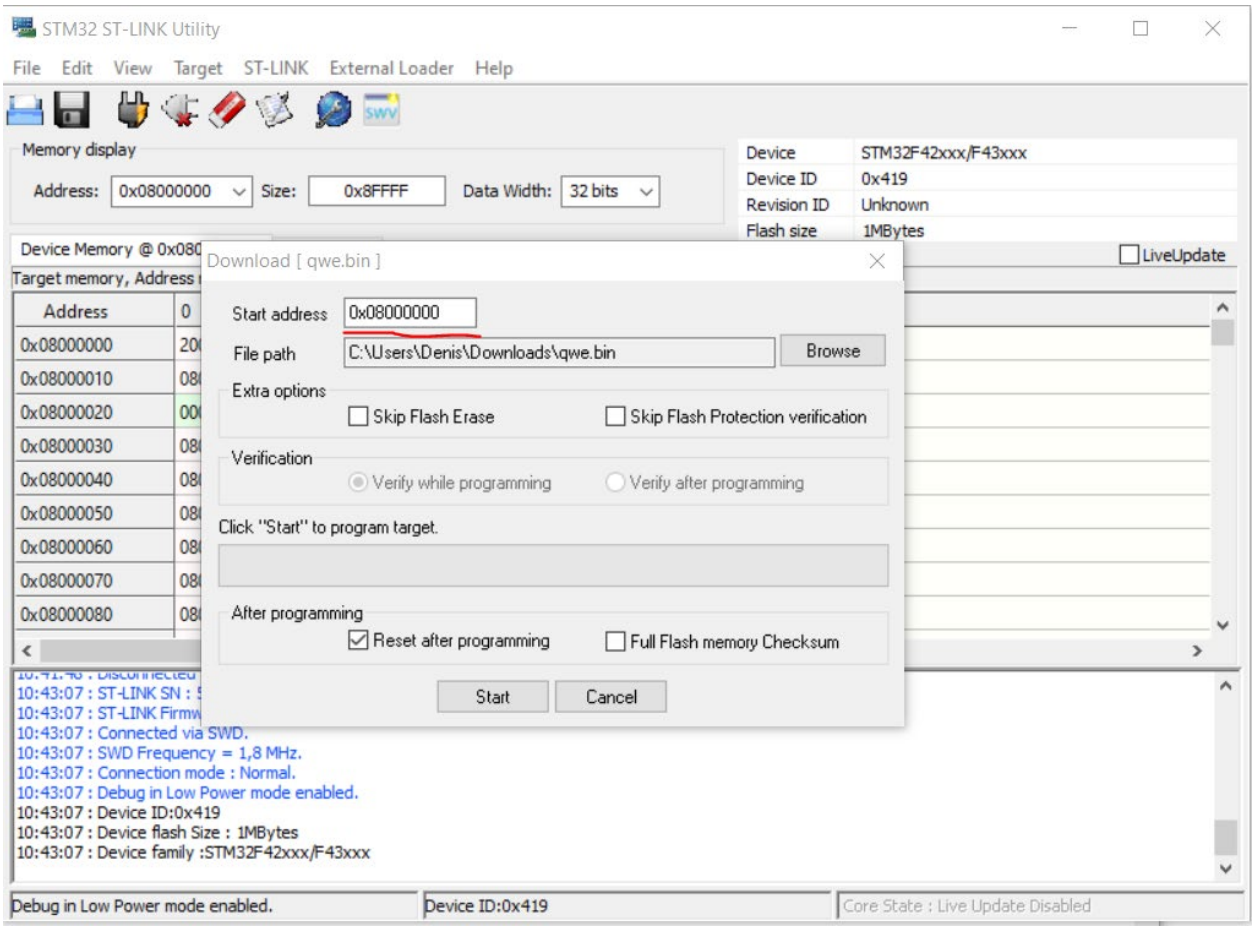

Рисунок 5 Загрузка считанного файла прошивки в контроллер

- Обновление веб-интерфейса K15_CPU

Перед обновлением веб-интерфейса рекомендуется произвести обновление ПО контроллера через веб-интерфейс или st-link до актуальной версии.

Для обновления веб-интерфейса необходимо:

1. Соединить контроллер K15 с ПК при помощи Ethernet кабеля. На ПК открыть ПО Total commander и осуществить подключение к FTP серверу контроллера по его IP адресу. Перейти в директорию SYS. (Рисунок 6).
2. Удалить файлы веб-интерфейса с сервера.
3. Скопировать файлы актуального веб-интерфейса из директории FatFs->SYS поставляемого комплекта ПО.

При возникновении проблем с FTP сервером смотреть раздел «Траблшутинг» в пункте «обновление ПО K15_CPU через веб-интерфейс».

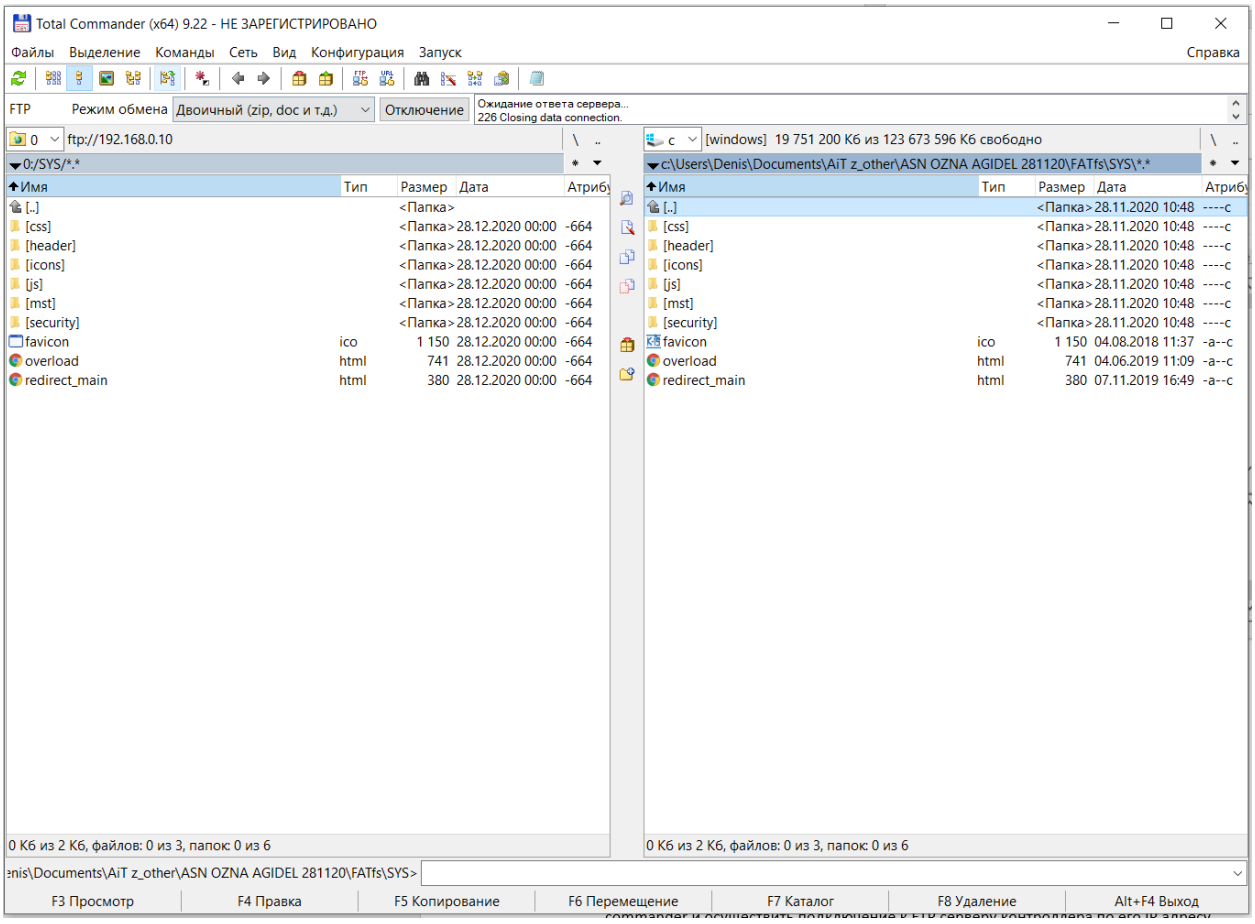

Рисунок 6 Файлы веб-интерфейса

- обновление ПО модулей через веб-интерфейс

Перед обновлением ПО модулей необходимо обновить ПО контроллера и загрузить актуальный веб-интерфейс. Для обновления ПО модулей необходимо:

1. Соединить контроллер К15 с ПК при помощи Ethernet кабеля. На ПК открыть ПО Total commander и осуществить подключение к FTP серверу контроллера по его IP адресу. Перейти в директорию SET. Рекомендуется удалить другие файлы обновления ПО модулей из папки. Скопировать файлы обновления модулей с расширением .gpboot в директорию SET. (рисунок 1).
2. Перейти по IP адресу контроллера в браузере (рекомендуется использовать ПО Google chrome). Авторизоваться с учетной записи super_admin (пароль - 12345678).
3. На главной странице в шапке перейти по пункту “Обновление (ранее Система) -> Обновление ПО модулей К15”.
4. В выпадающем списке выберите необходимый модуль для обновления. Если модуль поддерживает функцию обновления, то отобразится информация о текущей прошивки и доступные файлы обновления из директории SET (рисунок 7). Если модуль имеет устаревшее ПО без возможности обновления через веб-интерфейс, то отобразится надпись «Данный модуль не поддерживает функцию обновления ПО», в данной ситуации необходимо обновить модуль через st-link до актуальной версии (Пункт «обновление ПО модулей через st-link»), после чего модуль будет иметь функцию обновления через веб-интерфейс и его больше не придется обновлять через st-link.
5. Нажать «запустить обновление» в поле с требуемым файлом обновления и подтвердить выбор. Страница обновится и будет отображать статус обновления в строке сверху. По завершению отобразит информацию об успешном или неуспешном обновлении. Перед проверкой информации о прошивке обновленного модуля рекомендуется еще раз **обновить страницу.** Так как, список модулей не обновляется динамически и если после обновления изменился номер модуля, то система отобразит информацию о невозможности обновления. После обновления поля «версия ПО» в файле обновления и самой прошивке должны совпадать (Рисунок 8).

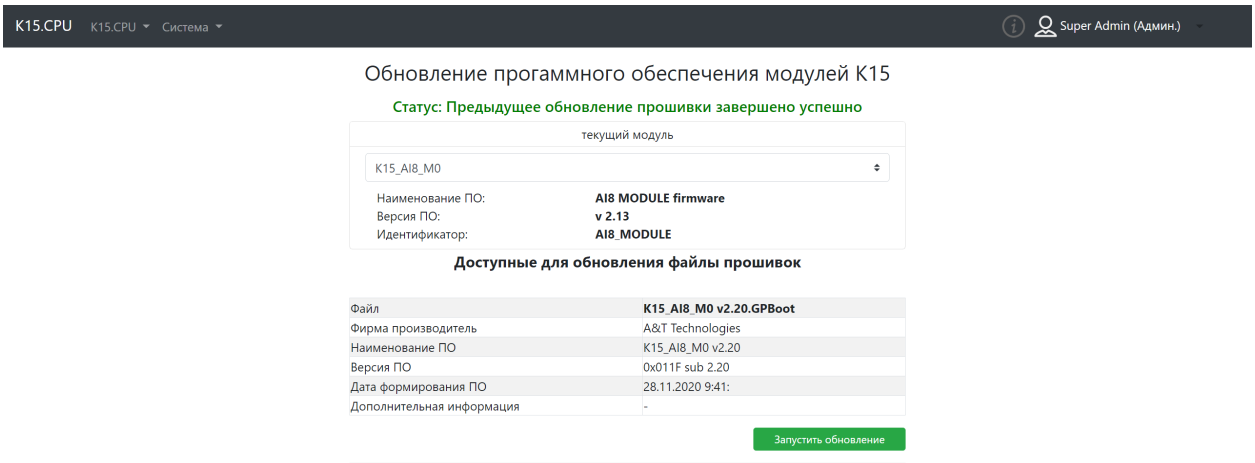

Рисунок 7 Отображение на странице обновления модулей

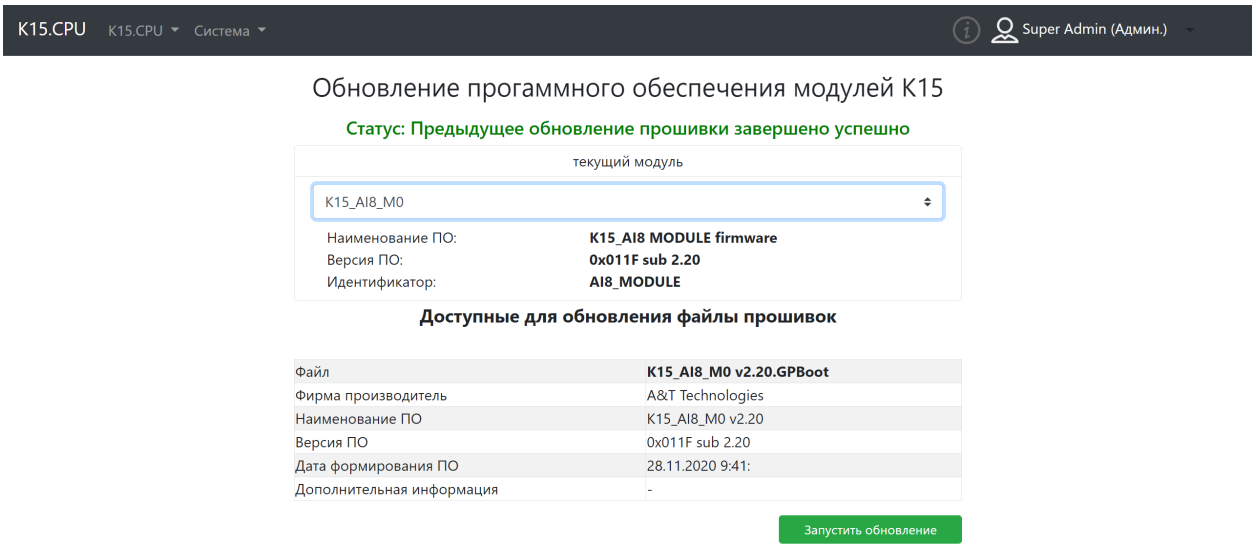

Рисунок 8 Информация о ПО после обновления

- обновление ПО модулей через st-link

Для обновления ПО модулей необходимо разобрать модуль, подсоединить провод программирования к плате, собрать корпус (прижав разъем в плате при помощи подручных стредств). Для обновления ПО необходимо:

1. Подключить программатор к контроллеру через соединительный кабель. Открыть ПО ST-Link utility и произвести подключение к контроллеру.
2. Перейти в Target->Program…, выбрать файл обновления «K15_MODULES_BOOT_v2.00.bin» из поставляемого комплекта ПО и установить стартовый адрес **0x08000000** (это необходимо делать после выбора файла, так как стартовый адрес сбрасывается после выбора файла). (Рисунок 9)
3. Нажать “Start” и дождаться завершения обновления.
4. Перейти в Target->Program…, выбрать файл обновления соответствующего модуля с расширением **.bin** из поставляемого комплекта ПО и установить стартовый адрес **0x08001000** (это необходимо делать после выбора файла, так как стартовый адрес сбрасывается после выбора файла). (Рисунок 10)
5. Нажать “Start” и дождаться завершения обновления.

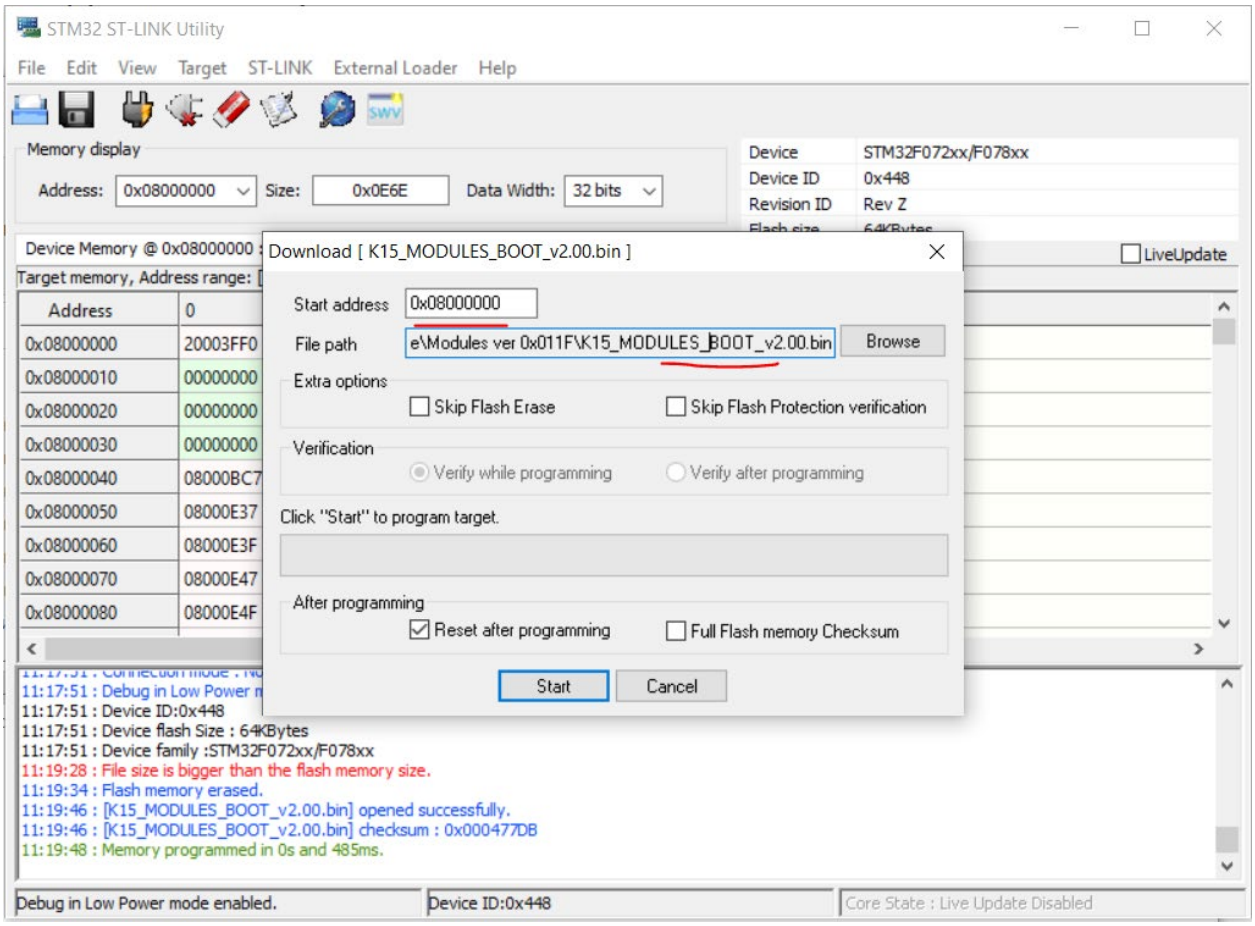

Рисунок 9 Настройки загрузки бутлодера

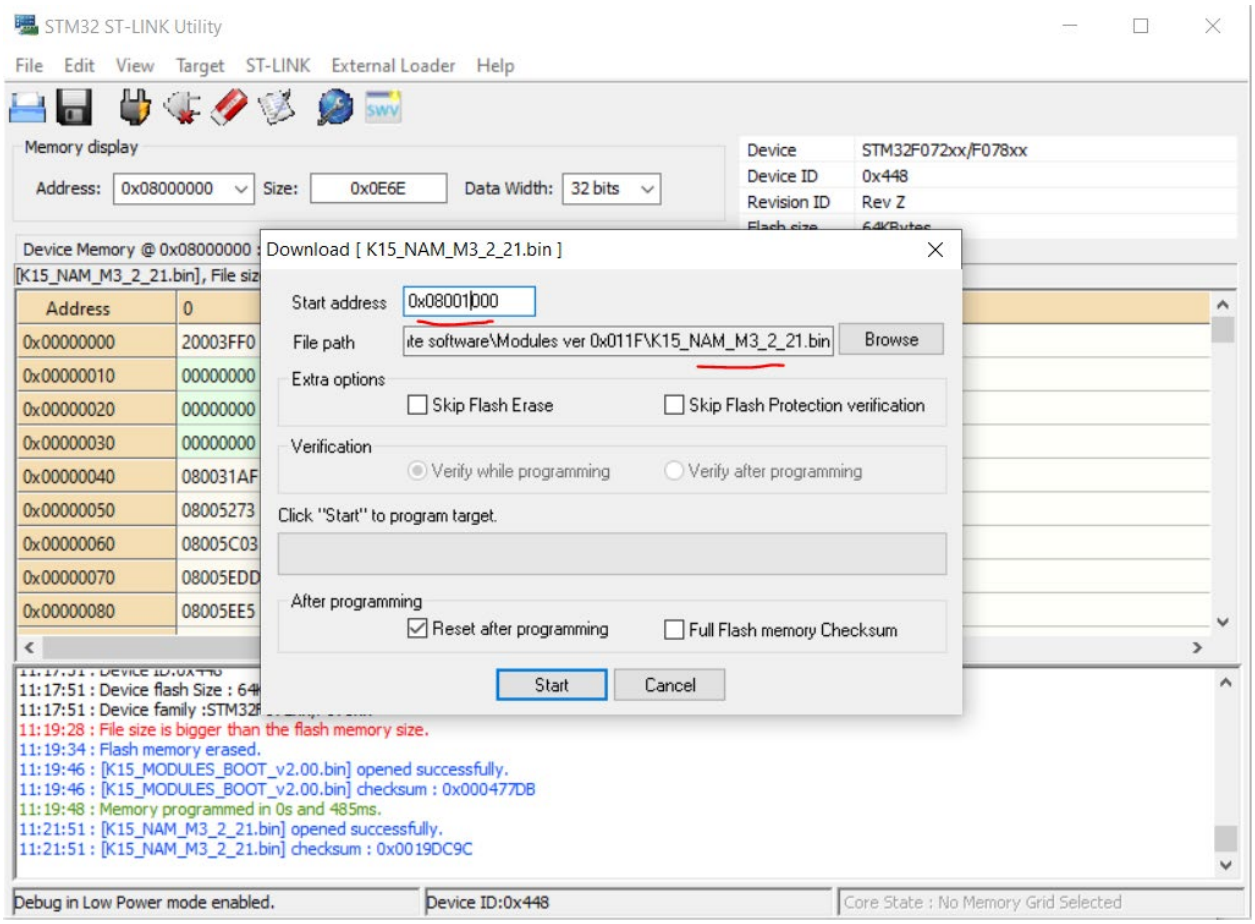

Рисунок 10 Настройки загрузки прошивки модуля