Проект ControlTree

[Общее описание](#_Toc46835009)

[Головное устройство](#_Toc46835010)

[Подчинённые устройства](#_Toc46835011)

[Подключение HMC821](#_Toc46835012)

[Подключение ADF5356](#_Toc46835013)

[Команды: описание](#_Toc46835014)

[Обычная команда](#_Toc46835015)

[Передача файла](#_Toc46835016)

[Контрольная сумма](#_Toc46835017)

[Список прямых команд](#_Toc46835018)

[Подтверждение](#_Toc46835019)

[Задать адрес](#_Toc46835020)

[Задать тип устройства](#_Toc46835021)

[Задать имя устройства](#_Toc46835022)

[Узнать адрес, тип и имя устройства](#_Toc46835023)

[Список команд ГУ](#_Toc46835024)

[Просканировать шину, вывести список найденных устройств](#_Toc46835025)

[Получить список зарегистрированных устройств](#_Toc46835026)

[Добавить устройство в таблицу](#_Toc46835027)

[Поместить устройство в таблицу с перезаписью параметров](#_Toc46835028)

[Удалить устройство из таблицы](#_Toc46835029)

[Вывести состояние всех устройств](#_Toc46835030)

[Список универсальных команд](#_Toc46835031)

[Ping](#_Toc46835032)

[Узнать состояние устройства](#_Toc46835033)

[Управление универсальным портом ввода-вывода](#_Toc46835034)

[Обмен данными по SPI](#_Toc46835035)

[Передача малого количества байт](#_Toc46835036)

[Передача большого количества байт](#_Toc46835037)

[Сохранить настройки линии CS](#_Toc46835038)

[Получить настройки линии CS](#_Toc46835039)

[Обновление прошивки](#_Toc46835040)

[Список команд подчинённого устройства](#_Toc46835041)

[Изменить адрес](#_Toc46835042)

[Изменить тип устройства](#_Toc46835043)

[Изменить имя устройства](#_Toc46835044)

[Узнать тип и имя устройства](#_Toc46835045)

[Управление термостатированием](#_Toc46835046)

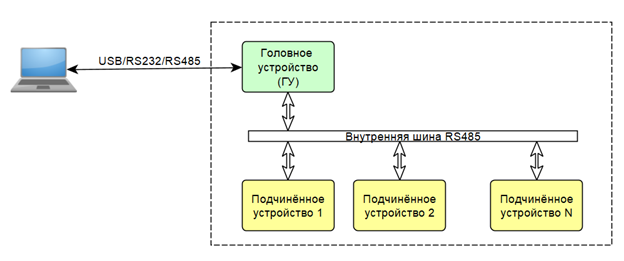
[Управление MRL (HMC821)](#_Toc46835047)

[Управление KuKonv и IKS (ADF5356)](#_Toc46835048)

[Управление частотой KuKonv (ADF5356)](#_Toc46835049)

# Общее описание

Система состоит из нескольких устройств. Система управляется с компьютера по любому из интерфейсов: USB, RS232, RS485. Этими интерфейсами оборудовано головное устройство системы (далее – ГУ). ГУ транслирует команды от компьютера подчинённым устройствам, а также транслирует обратно их ответы. Обмен данными между ГУ и остальными устройствами осуществляется по внутренней шине RS485.



Все внешние интерфейсы логически представляют собой последовательный порт (в частности, USB). Формат команд одинаков для всех интерфейсов.

# Головное устройство

ГУ хранит в энергонезависимой памяти таблицу зарегистрированных устройств с адресами, типами и именами. Это нужно для того, чтобы при сбое/отключении устройства указать, что оно должно быть, но не отвечает. Устройство можно добавить в таблицу либо удалить из неё при помощи команды.

ГУ имеет адрес 0, это не изменяется. Адрес нужен для ряда команд.

# Подчинённые устройства

У подчинённого устройства есть следующие обязательные параметры:

* Уникальный адрес (1 байт)
* Идентификатор типа (1 байт)
* Строковое имя (максимум 15 символов, по 1 байту на символ, без спецсимволов и пробела, кодировка не учитывается).

Тип устройства (и его функционал) зависит от установленных на плате компонентов. Адрес может быть задан путём подачи соответствующей команды; адрес принимает значения от 1 до 32 включительно. Тип и имя задаются подачей адресной команды.

Во всех устройствах есть GPIO и шины SPI1 и SPI2.

Есть следующие типы подчинённых устройств:

|  |  |  |
| --- | --- | --- |
| Устройство | Тип | Дополнительные подсистемы, помимо GPIO, SPI1, SPI2 |
| HF\_Block (ГУ) | 1 | USB, RS232, внешняя RS485, внутренняя RS485 |
| LNA | 2 | 2 термодатчика, нагреватель, датчик тока |
| KUKonv | 3 |  |
| MRL | 4 |  |
| Triplexer | 5 | 2 термодатчика, нагреватель, датчик тока |
| IKS | 6 |  |

## Подключение HMC821

Тип устройства с HMC821: MRL.

HMC821 имеет два режима работы по SPI. Режим выбирается последовательностью прихода логической единицы на линии SEN и SCK, причём изменить это можно только снятием-включением питания. Поэтому линии SEN и SCK должны быть подтянуты к земле через 10к, во избежание непредсказуемого выбора.

По этой же причине может понадобиться выкл-вкл питания после смены типа устройства с какого-то другого на нужный.

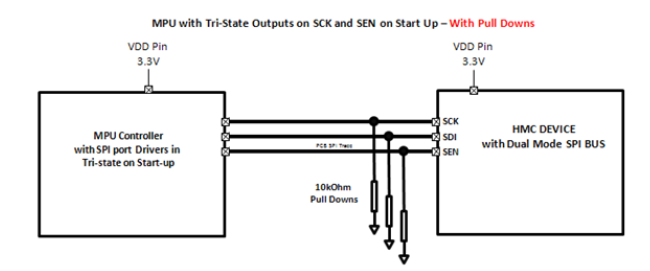
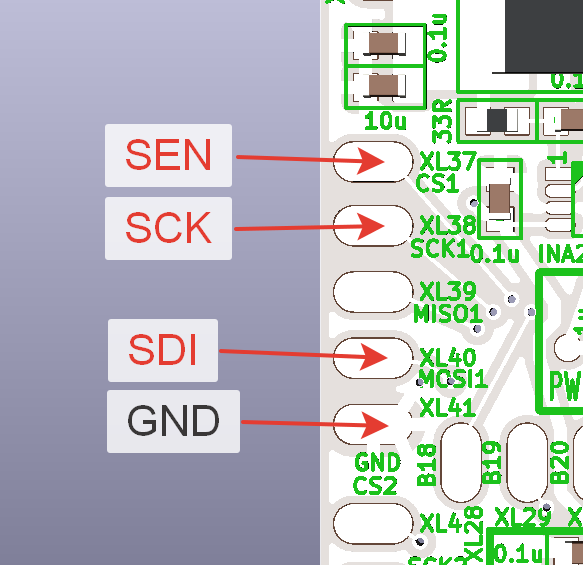


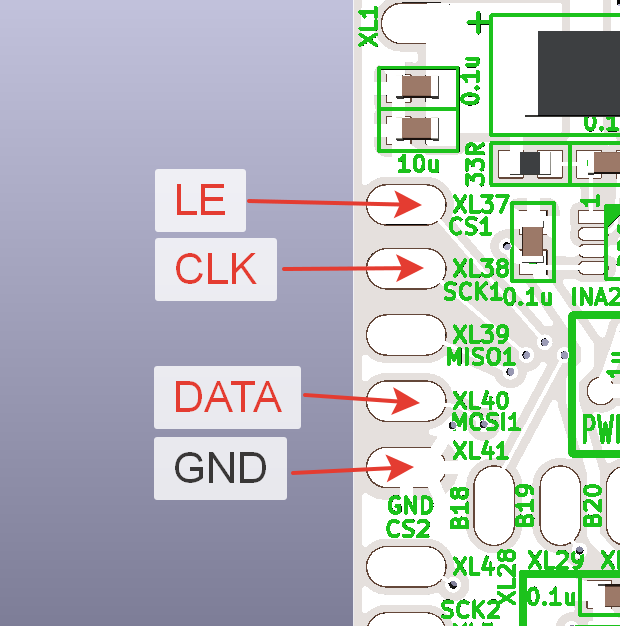
Схема подключения линий к управляющей плате:



## Подключение ADF5356

Тип устройства с ADF5356: KuKonv.

Схема подключения линий к управляющей плате:



# Команды: описание

Все внешние интерфейсы логически представляют собой последовательный порт (в частности, USB). Формат команд одинаков для всех интерфейсов.

Инициатива всегда на стороне компьютера. То есть, компьютер посылает запрос, ГУ на него отвечает. ГУ не посылает данных по своей инициативе.

Управление осуществляется по принципу «команда - ответ». Формат команды и ответа одинаков.

В дальнейшем описании обязательные параметры заключаются в треугольные скобки < >, а необязательные – в квадратные скобки: [ ].

## Обычная команда

В общем виде команда выглядит так:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ИмяКоманды | [Параметр1] | , [Параметр2] | , [Параметр3] | \r\n |

То есть, параметры присутствуют при необходимости, их количество не более 100. Имя команды и параметры разделяются пробелами и/или запятыми. Количество пробелов не имеет значения. Имя команды Case Insensitive, то есть, регистр символов неважен (GETDEVLIST, GetDevList, getdevlist – это одно и то же). Символы \r\n являются признаком конца команды.

Числовые параметры могут быть заданы в форме:

* Десятичной (0, 42, -80)
* Шестнадцатиричной (0x18, 0xDEADBEEF)
* Дробной (3.14159)

Ответом на часть команд является подтверждение получения с кодом ошибки. Код ошибки 0 – это успешное выполнение команды. Не 0 – некий сбой.

## Передача файла

При необходимости передать в устройство большое количество данных имеет смысл использовать передачу файла. В этом случае команда становится двухфазной:

|  |  |  |  |
| --- | --- | --- | --- |
| ИмяКоманды | [Адрес устройства] | <Количество последующих байт> | \r\n |

После получения такой команды приёмник подготавливает всё необходимое и выдаёт приглашение: символ ‘>’. После этого передатчик должен передать нужные байты как есть, без какого-либо кодирования. Максимальный размер данных -- 64 кБайта (65535 байт). Данные должны быть переданы в течение 10 секунд, в противном случае приём прекращается и выдаётся сообщение Timeout. В случае успешного приёма выдаётся ответ Ack 0.

Приём большого объёма данных осуществляется похожим образом. Компьютер передаёт команду в том же виде:

|  |  |  |  |
| --- | --- | --- | --- |
| ИмяКоманды | [Адрес устройства] | <Количество последующих байт> | \r\n |

Затем отправляет приглашающий символ ‘>’ и принимает указанное количество байт.

## Контрольная сумма

Используется CRC16-CCITT с начальным значением 0, а не 0xFFFF.

Алгоритм на C#:

<https://github.com/GataullinRR/CRC16/blob/master/CRC16/CRC16_CCITT.cs>

Алгоритм на C::

#define CRC\_INITVALUE 0x0000U

static const uint16\_t CRCTable[256] = {

0x0000,0x1021,0x2042,0x3063,0x4084,0x50A5,0x60C6,0x70E7,0x8108,0x9129,0xA14A,0xB16B,0xC18C,0xD1AD,0xE1CE,0xF1EF,

0x1231,0x0210,0x3273,0x2252,0x52B5,0x4294,0x72F7,0x62D6,0x9339,0x8318,0xB37B,0xA35A,0xD3BD,0xC39C,0xF3FF,0xE3DE,

0x2462,0x3443,0x0420,0x1401,0x64E6,0x74C7,0x44A4,0x5485,0xA56A,0xB54B,0x8528,0x9509,0xE5EE,0xF5CF,0xC5AC,0xD58D,

0x3653,0x2672,0x1611,0x0630,0x76D7,0x66F6,0x5695,0x46B4,0xB75B,0xA77A,0x9719,0x8738,0xF7DF,0xE7FE,0xD79D,0xC7BC,

0x48C4,0x58E5,0x6886,0x78A7,0x0840,0x1861,0x2802,0x3823,0xC9CC,0xD9ED,0xE98E,0xF9AF,0x8948,0x9969,0xA90A,0xB92B,

0x5AF5,0x4AD4,0x7AB7,0x6A96,0x1A71,0x0A50,0x3A33,0x2A12,0xDBFD,0xCBDC,0xFBBF,0xEB9E,0x9B79,0x8B58,0xBB3B,0xAB1A,

0x6CA6,0x7C87,0x4CE4,0x5CC5,0x2C22,0x3C03,0x0C60,0x1C41,0xEDAE,0xFD8F,0xCDEC,0xDDCD,0xAD2A,0xBD0B,0x8D68,0x9D49,

0x7E97,0x6EB6,0x5ED5,0x4EF4,0x3E13,0x2E32,0x1E51,0x0E70,0xFF9F,0xEFBE,0xDFDD,0xCFFC,0xBF1B,0xAF3A,0x9F59,0x8F78,

0x9188,0x81A9,0xB1CA,0xA1EB,0xD10C,0xC12D,0xF14E,0xE16F,0x1080,0x00A1,0x30C2,0x20E3,0x5004,0x4025,0x7046,0x6067,

0x83B9,0x9398,0xA3FB,0xB3DA,0xC33D,0xD31C,0xE37F,0xF35E,0x02B1,0x1290,0x22F3,0x32D2,0x4235,0x5214,0x6277,0x7256,

0xB5EA,0xA5CB,0x95A8,0x8589,0xF56E,0xE54F,0xD52C,0xC50D,0x34E2,0x24C3,0x14A0,0x0481,0x7466,0x6447,0x5424,0x4405,

0xA7DB,0xB7FA,0x8799,0x97B8,0xE75F,0xF77E,0xC71D,0xD73C,0x26D3,0x36F2,0x0691,0x16B0,0x6657,0x7676,0x4615,0x5634,

0xD94C,0xC96D,0xF90E,0xE92F,0x99C8,0x89E9,0xB98A,0xA9AB,0x5844,0x4865,0x7806,0x6827,0x18C0,0x08E1,0x3882,0x28A3,

0xCB7D,0xDB5C,0xEB3F,0xFB1E,0x8BF9,0x9BD8,0xABBB,0xBB9A,0x4A75,0x5A54,0x6A37,0x7A16,0x0AF1,0x1AD0,0x2AB3,0x3A92,

0xFD2E,0xED0F,0xDD6C,0xCD4D,0xBDAA,0xAD8B,0x9DE8,0x8DC9,0x7C26,0x6C07,0x5C64,0x4C45,0x3CA2,0x2C83,0x1CE0,0x0CC1,

0xEF1F,0xFF3E,0xCF5D,0xDF7C,0xAF9B,0xBFBA,0x8FD9,0x9FF8,0x6E17,0x7E36,0x4E55,0x5E74,0x2E93,0x3EB2,0x0ED1,0x1EF0,

};

uint16\_t CalculateCRC16(uint8\_t \*Buf, uint32\_t Len) {

uint16\_t crc = CRC\_INITVALUE;

for(uint32\_t i=0; i<Len; ++i) {

crc = (crc << 8) ^ CRCTable[(crc >> 8) ^ (0xFF & Buf[i])];

}

return crc;

}

void CRCPrintTable() {

uint16\_t poly = 4129;

uint16\_t temp, a;

uint32\_t Cnt = 0;

for(uint16\_t i = 0; i < 256; ++i) {

temp = 0;

a = (uint16\_t)(i << 8);

for(uint16\_t j=0; j < 8; ++j) {

if(((temp ^ a) & 0x8000) != 0) {

temp = (uint16\_t)((temp << 1) ^ poly);

}

else {

temp <<= 1;

}

a <<= 1;

}

Printf("0x%04X,", temp);

if(Cnt++ == 15) {

Cnt = 0;

Printf("\r");

}

}

}

# Список прямых команд

Эти команды неадресные, то есть, их следует подавать при непосредственном подключении к устройству.

## Подтверждение

Ответ на команду, не требующую развёрнутой информации.

Успешное выполнение:

|  |  |
| --- | --- |
| Ok | \r\n |

Неправильное значение или отсутствие параметра:

|  |  |
| --- | --- |
| BadParam | \r\n |

Команда не выполнена за ожидаемое время:

|  |  |
| --- | --- |
| Timeout | \r\n |

Сбой выполнения:

|  |  |
| --- | --- |
| Failure | \r\n |

Несовпадение контрольной суммы:

|  |  |
| --- | --- |
| CRCError | \r\n |

## Задать адрес

Задаёт адрес устройства. Адрес сохраняется в энергонезависимую память. В случае ГУ адрес считается нулевым независимо от того, что записано в энергонезависимую память.

Формат:

|  |  |  |
| --- | --- | --- |
| SetAddr | <Адрес> | \r\n |

При успехе ответ Ok.

При неправильном адресе (более максимального) ответ BadParam.

При сбое (например, сбой EEPROM) ответ Failure.

## Задать тип устройства

В случае, если тип устройства -- ГУ, то его адрес автоматически становится нулевым.

Формат:

|  |  |  |
| --- | --- | --- |
| SetType | Новый тип | \r\n |

В случае успеха ответ Ok.

При неправильном типе ответ BadParam.

## Задать имя устройства

Формат:

|  |  |  |
| --- | --- | --- |
| SetName | Новое имя | \r\n |

В случае успеха ответ Ok.

При неправильном имени (слишком длинное) ответ BadParam.

## Узнать адрес, тип и имя устройства

Формат:

|  |  |
| --- | --- |
| GetInfo | \r\n |

Ответ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RGetInfo | Адрес | Тип | Имя | \r\n |

# Список команд ГУ

Эти команды выполняются только головным устройством.

## Просканировать шину, вывести список найденных устройств

Формат:

|  |  |
| --- | --- |
| Scan | \r\n |

В ответ выводится список устройств на шине в виде:

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес1 | Тип1 | Имя1 | \n |
| Адрес2 | Тип2 | Имя2 | \n |
| АдресN | ТипN | ИмяN | \n\r |

При отсутствии устройств на шине выводится текст:

|  |  |
| --- | --- |
| NoDevices | \r\n |

## Получить список зарегистрированных устройств

Формат:

|  |  |
| --- | --- |
| GetDeviceList | \r\n |

В ответ выводится список устройств в виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес1 | Тип1 | Имя1 | Статус1 | \n |
| Адрес2 | Тип2 | Имя2 | Статус2 | \n |
| АдресN | ТипN | ИмяN | СтатусN | \n\r\n |

Статус:

* Ok - устройство отзывается, имя и тип соответствуют таблице
* NoAnswer - устройство не отзывается
* DifferentParams - устройство отзывается, но его тип и имя отличаются от таблицы

При отсутствии устройств в таблице выводится текст:

|  |  |
| --- | --- |
| NoDevices | \r\n |

## Добавить устройство в таблицу

Формат:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AddDevice | Адрес | Тип | Имя | \r\n |

В случае успеха ответ Ok.

В случае наличия в таблице устройства с указанным адресом новое не добавляется, ответ DeviceExists.

В случае неправильного адреса, типа или имени устройство не добавляется, ответ BadParam.

В случае заполненности таблицы устройство не добавляется, ответ TableFull.

## Поместить устройство в таблицу с перезаписью параметров

Формат:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PutDevice | Адрес | Тип | Имя | \r\n |

В случае наличия в таблице устройства с указанным адресом его тип и имя перезаписываются. В случае отсутствия - добавляется новое, и команда по сути превращается в AddDevice.

В случае успеха ответ Ok.

В случае неправильного типа или имени устройство не добавляется, ответ BadParam.

В случае заполненности таблицы устройство не добавляется, ответ TableFull.

## Удалить устройство из таблицы

Формат:

|  |  |  |
| --- | --- | --- |
| DelDevice | Адрес | \r\n |

В случае отсутствия в таблице устройства с указанным адресом ответ NoDevices.

В случае успеха ответ Ok.

## Вывести состояние всех устройств

Формат:

|  |  |
| --- | --- |
| GetAllStates | \r\n |

В ответ выводится список устройств в виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес1 | Тип1 | Имя1 | Состояние1 | \n |
| Адрес2 | Тип2 | Имя2 | Состояние2 | \n |
| АдресN | ТипN | ИмяN | СостояниеN | \n\r\n |

Состояние включает в себя параметры, специфичные для данного типа устройства. При отсутствии ответа от устройства ответ NoAnswer, если тип и имя отличается -- DifferentParams.

В списке всегда присутствует головное устройство. Его адрес 0, его параметры -- текущее значение регистра GPIO.

Для устройств с термостатированием (LNA и Triplexer) выдаётся следующая информация:

t=<усреднённая температура> TStating=<0 или 1> t1=<температура датчика или err> t2=<температура датчика или err> t3=<температура датчика или err>

# Список универсальных команд

Эти команды могут быть выполнены и головным устройством, и подчинённым. В случае головного устройства адрес устройства указывается нулевым, и нет ответа NoAnswer.

## Ping

Для получения быстрого отклика от устройства.

|  |  |  |
| --- | --- | --- |
| Ping | Адрес устройства | \r\n |

В случае успеха ответ Ok

В случае отсутствия отклика от устройства ответ NoAnswer.

## Узнать состояние устройства

Формат:

|  |  |  |
| --- | --- | --- |
| GetState | Адрес устройства | \r\n |

В случае отсутствия отклика от устройства ответ NoAnswer.

Штатный ответ:

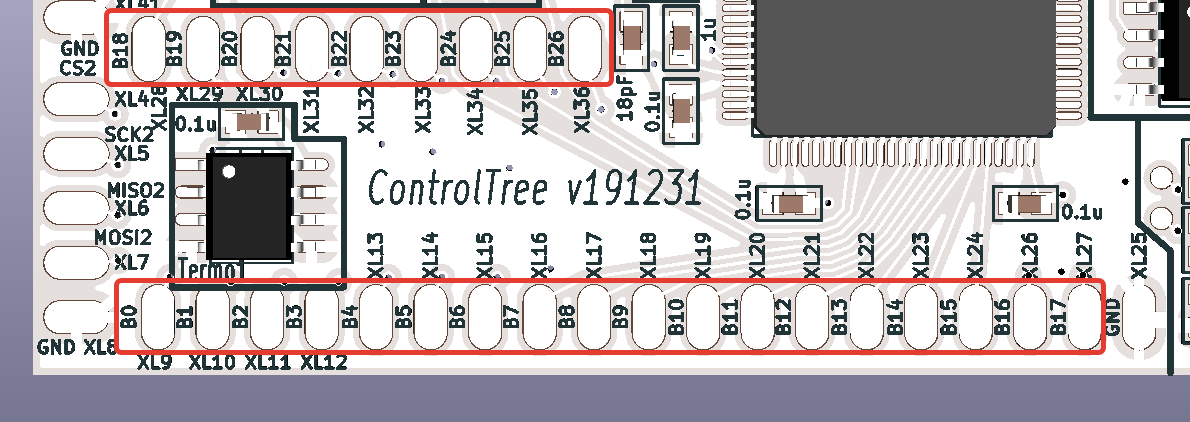
RGetState <Версия прошивки> <GPIO=Текущее значение GPIO> [t=<Текущая температура> TStating=<Термостатирование Вкл/Выкл (1 / 0)>] [SynthFreq=<частота синтезатора> SynthOffset=<сдвиг счастоты>] [Current=<Ток потребления>]

Ответ содержит те или иные параметры в засисимости от типа устройства.

Версия прошивки имеет формат времени: ГГГГММДД\_ЧЧММ, например, 20200505\_2059. Прошивка одинакова для всех типов устройств.

## Управление универсальным портом ввода-вывода

На плате имеется 27 контактов, обозначенных B0, B1, …, B17, B18, …, B26:



Эти контакты работают как выходы Push-Pull. Напряжение логической единицы 3.3В, максимальный втекающий и вытекающий ток 20мА.

Работа с контактами осуществляется путём записи и чтения виртуального 32-битного регистра. Младший бит описывает контакт B0, следующий -- B1, и так далее до B26. Биты 27...31 ничего не делают.

Установить значение регистра:

|  |  |  |  |
| --- | --- | --- | --- |
| SetGPIO | Адрес устройства | Число для записи в регистр | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Прочитать текущее значение регистра:

|  |  |  |
| --- | --- | --- |
| GetGPIO | Адрес устройства | \r\n |

В случае успеха ответ RGetGPIO <Значение регистра>

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Записать в энергонезависимую память значение регистра (будет записано в собственно регистр при включении питания устройства). Выполнение этой команды не меняет текущее значение регистра.

|  |  |  |  |
| --- | --- | --- | --- |
| SetPowerOnGPIO | Адрес устройства | Число для записи в регистр | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Прочитать значение регистра из энергонезависимой памяти:

|  |  |  |
| --- | --- | --- |
| GetPowerOnGPIO | Адрес устройства | \r\n |

В случае успеха ответ RGetPowerOnGPIO <Значение регистра>

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

## 

## Обмен данными по SPI

В устройстве две шины SPI, первая и вторая.

### Передача малого количества байт

Формат:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wrSpi | Адрес устройства | Параметры SPI | Набор байт на отправку | \r\n |

В случае успеха ответ:

RWRSPI <набор ответных байт по количеству отправленных>

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Параметры SPI задаются битовой маской, размер 1 байт:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **b7** | **b6** | **b5** | **b4** | **b3** | **b2** | **b1** | **b0** |
| BitOrder:  0 - MSB  1 - LSB | CPOL:  0 - IdleLow  1 - IdleHigh | CPHA:  0 - FirstEdge  1 - SecondEdge | № SPI:  0 - SPI1  1 - SPI2 | Инверсия CS:  0 – Active Low  1 – Active High | Делитель частоты | | |

Бит инверсии CS определяет поведение этой линии. Active Low, Idle High – это обычное, стандартное поведение CS. Active High, Idle Low – инвертированное.

Делитель частоты определяет скорость передачи данных:

000: 24 МГц

001: 12 МГц

010: 6 МГц

011: 3 МГц

100: 1500 кГц

101: 750 кГц

110: 375 кГц

111: 187.5 кГц

### Передача большого количества байт

При передаче большого количества байт (более несколькоих десятков) может быть удобнее передавать их в бинарном виде, файлом.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| wSpiFile | Адрес устройства | Параметры SPI | Размер данных, байт (не более 65536) | \r\n |

При готовности устройство выдаст приглашающий символ ‘>’, после чего следует передать указанное количество данных. В случае неготовности устройство ответит кодом ошибки, в этом случае передавать данные не надо.

После выполнения передачи данных по SPI устройство ответит Ok. Ответные данные при этом сохраняются в памяти устройства, их можно получить следующей командой:

|  |  |  |  |
| --- | --- | --- | --- |
| rSpiFile | Адрес устройства | Размер данных, байт (не более 65535) | \r\n |

После получения от компьютера приглашающего символа ‘>’ устройство передаёт содержимое буфера в бинарном виде и возвращается в текстовый режим работы. Поскольку буфер для данных SPI один для обеих шин, то будут выданы данные, полученные при выполнении предыдущей команды SPI.

## Сохранить настройки линии CS

Разные устройства требуют различные уровни на линии CS. Нужно выставить правильный уровень при включении питания платы.

|  |  |  |  |
| --- | --- | --- | --- |
| SetCS1 | Адрес устройства | Поведение CS1:  0 – Active Low, Idle High  1 – Active High, Idle Low | \r\n |

|  |  |  |  |
| --- | --- | --- | --- |
| SetCS2 | Адрес устройства | Поведение CS2:  0 – Active Low, Idle High  1 – Active High, Idle Low | \r\n |

В случае успеха ответ Ok.

## Получить настройки линии CS

|  |  |  |
| --- | --- | --- |
| GetCS1 | Адрес устройства | \r\n |

|  |  |  |
| --- | --- | --- |
| GetCS2 | Адрес устройства | \r\n |

В случае успеха ответ RGetCS1 <Поведение CS1>, RGetCS2 <Поведение CS2>

## Обновление прошивки

Бинарный файл прошивки размером не более 65535 байт отправляется устройству. Устройство обновляет содержимое неактивного банка FLASH, выставляет соответствующий номер банка и перезапускается. При старте запускается прошивка из указанного банка.

Передача файла прошивки осуществляется обычным образом. Сначала отправляется команда:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UpdateFW | Адрес устройства | Размер данных, байт (не более 65536) | Контрольная сумма файла прошивки, CRC16 | \r\n |

При готовности устройство выдаст приглашающий символ ‘>’, после чего следует передать файл прошивки в бинарном виде. В случае неготовности устройство ответит кодом ошибки, в этом случае передавать данные не надо.

После получения файла устройство сравнит контрольную сумму. В случае успеха устройство ответит Ok, обновит нужный банк FLASH и перезагрузится.

Описание контрольной суммы см. в соответствующем разделе.

# Список команд подчинённого устройства

Эти команды выполняются только подчинёнными устройствами.

## Изменить адрес

Изменяет адрес устройства. Адрес сохраняется в энергонезависимую память.

Формат:

|  |  |  |  |
| --- | --- | --- | --- |
| ChangeAddr | Адрес устройства | Новый адрес | \r\n |

В случае успеха ответ Ok.

В случае отсутствия отклика от устройства ответ NoAnswer.

## Изменить тип устройства

Формат:

|  |  |  |  |
| --- | --- | --- | --- |
| ChangeType | Адрес устройства | Новый тип | \r\n |

В случае успеха ответ Ok.

В случае отсутствия отклика от устройства ответ NoAnswer.

## Изменить имя устройства

Формат:

|  |  |  |  |
| --- | --- | --- | --- |
| ChangeName | Адрес устройства | Новое имя | \r\n |

В случае успеха ответ Ok.

В случае отсутствия отклика от устройства ответ NoAnswer.

## Узнать тип и имя устройства

Формат:

|  |  |  |
| --- | --- | --- |
| GetTypeName | Адрес устройства | \r\n |

В случае успеха ответ RGetTypeName <Тип устройства> <Имя устройства>.

В случае отсутствия отклика от устройства ответ NoAnswer.

## Управление термостатированием

Задать температуру термостатирования в градусах Цельсия

(сохраняется в EEPROM):

|  |  |  |  |
| --- | --- | --- | --- |
| SetTTS | Адрес устройства | Температура | \r\n |

В случае успеха ответ Ok.

В случае отсутствия отклика от устройства ответ NoAnswer.

Прочитать значение температуры термостатирования:

|  |  |  |
| --- | --- | --- |
| GetTTS | Адрес устройства | \r\n |

В случае успеха ответ RGetTTS <Значение температуры>

В случае отсутствия отклика от устройства ответ NoAnswer.

Включить / выключить термостатирование (сохраняется в EEPROM):

|  |  |  |  |
| --- | --- | --- | --- |
| SetTstating | Адрес устройства | Вкл/выкл (1 / 0) | \r\n |

В случае успеха ответ Ok.

В случае отсутствия отклика от устройства ответ NoAnswer.

Прочитать значение температуры термостатирования:

|  |  |  |
| --- | --- | --- |
| GetTstating | Адрес устройства | \r\n |

В случае успеха ответ RGetTstating <Вкл/выкл (1 / 0)>

В случае отсутствия отклика от устройства ответ NoAnswer.

## Управление MRL (HMC821)

Установить значение регистров (без записи в энергонезависимую память):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SetRegs | Адрес устройства | Адрес регистра1 | Значение регистра1 | Адрес регистраN | Значение регистраN | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Сохранить значения регистров в энергонезависимой памяти (без немедленной установки). Максимальное количество регистров 22, повторы вполне возможны.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SaveRegs | Адрес устройства | Адрес регистра1 | Значение регистра1 | Адрес регистраN | Значение регистраN | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Вывести значения сохранённых в энергонезависимой памяти регистров:

|  |  |
| --- | --- |
| GetRegs | \r\n |

Ответ:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RGetRegs | Адрес регистра1 | Значение регистра1 | Адрес регистраN | Значение регистраN | \r\n |

В случае отсутствия сохранённых регистров ответ RGetRegs Empty\r\n

Очистить сохранённые в энергонезависимой памяти данные (при включении питания регистры не будут загружаться).

|  |  |
| --- | --- |
| ClearRegs | \r\n |

В случае успеха ответ Ok.

## Управление KuKonv и IKS (ADF5356)

Установить значение регистров (без записи в энергонезависимую память):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SetRegs | Адрес устройства | Значение регистра1 | Значение регистра2 | Значение регистраN | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Сохранить значения регистров в энергонезависимой памяти (без немедленной установки). Максимальное количество регистров 22, повторы вполне возможны.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SaveRegs | Адрес устройства | Значение регистра1 | Значение регистра2 | Значение регистраN | \r\n |

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Очистить сохранённые в энергонезависимой памяти данные (при включении питания регистры не будут загружаться).

|  |  |
| --- | --- |
| ClearRegs | \r\n |

В случае успеха ответ Ok.

### Управление частотой KuKonv (ADF5356)

Для вычисления значений регистров исходя из заданных частот используется алгоритм, представленный здесь: <https://bitbucket.org/semen966/registers/src/metrology/>

Как описано на страницах 32 и 33 даташита, последовательность записи в регистры разная для первого включения и для изменения частоты. Нужная последовательность – первоначальная или для изменения частоты – выбирается автоматически.

Также последовательность зависит от частоты *fPFD*.. Последовательность реализована только для частоты *fPFD* <= 75МГц.

Вычислить значения регистров и установить их (без записи в энергонезависимую память):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CalcRegsAndSet | Адрес устройства | fref | step | fd | fvco | \r\n |

Параметры дробные.

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.

Сохранить в энергонезависимую память параметры синтезатора, для применения при включении питания:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SaveSynthParams | Адрес устройства | fref | step | fd | fvco | \r\n |

Параметры дробные.

В случае успеха ответ Ok.

Для подчинённого устройства в случае отсутствия отклика от устройства ответ NoAnswer.