**Ч А С Т О Т Н Ы Й П Р Е О Б Р А З О В А Т Е Л Ь**

РЕГУЛЯТОР СКОРОСТИ ЭЛЕКТРОДВИГАТЕЛЯ ПЕРЕМЕННОГО ТОКА

# Протокол
# управления ПЧ серии E5-Mini
# по последовательной линии связи

## MODBUS

# Руководство пользователя

# ВЕСПЕР

**Версия 1.0** **Сентябрь 2024**

Оглавление

# 1 Описание протокола

Modbus представляет собой промышленный протокол промышленной связи «ведущий/ведомый». К линии подключено только одно ведущее устройство и один или несколько ведомых (максимум 247).

Ведущее устройство управляет обменом информации в линии связи, посылая запросы на ведомые устройства. Ведомое устройство не может отправлять информацию по линии без запроса ведущего устройства. Ведомые устройства никак не взаимодействуют друг с другом.

Если ведомое устройство не может распознать сообщение от ведущего, оно отправляет сообщение об ошибке.

Между ведущим и ведомым устройством осуществляется обмен данными двух типов:

- Ведущее устройство отправляет запрос на отдельное ведомое устройство. После получения и обработки запроса ведомое устройство возвращает ведущему сообщение «ответ». В таком режиме транзакция Modbus состоит из двух сообщений запроса от ведущего и ответа от ведомого устройства. Каждое подчиненное устройство должно иметь уникальный адрес (от 1 до 247), чтобы к нему можно было обращаться независимо от других устройств в линии связи.
- Ведущее устройство отправляет запрос всем ведомым устройствам. На широковещательные запросы, отправленные ведущим устройством, не возвращается ответ. Широковещательные команды обязательно являются командами записи. Все устройства должна принимать функцию широковещательной передачи для записи. Адрес 0 зарезервирован для идентификации широковещательного обмена.

### 1.1 Правила адресации Modbus

Адресное пространство Modbus содержит 256 различных адресов.

| 0 | От 1 до 247 | От 248 до 255 |
|---|---|---|
| Широковещательная рассылка | Адреса подчиненных устройств | Резерв |

Адрес 0 зарезервирован в качестве широковещательного адреса. Все подчиненные узлы должны распознавать широковещательный адрес.

Главный узел MODBUS не имеет определенного адреса, адрес должен быть только у подчиненных узлов.

### 1.2 Структура сообщений Modbus

Независимо от базовых уровней связи протокол Modbus определяет протокольный блок данных, **PDU** (**P**rotocol **D**ata **U**nit):

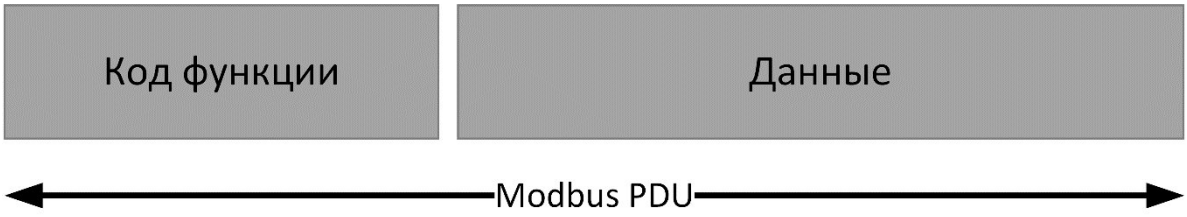


Рисунок 1.1 – Modbus PDU (Protocol Data Unit)

При работе протокола Modbus на конкретной шине или сети в блоке данных протокола PDU вводятся дополнительные поля. Клиент, который инициирует транзакцию Modbus, создает PDU, а затем добавляет поля для создания коммуникации, таким создается ADU (**A**pplication **D**ata **U**nit).

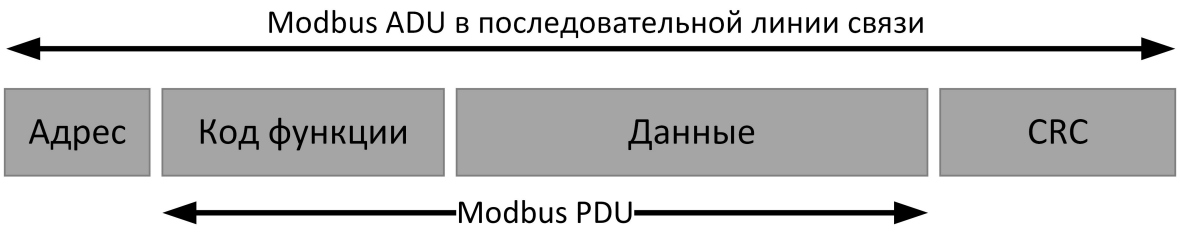


Рисунок 1.2 – Modbus ADU (Application Data Unit)

В последовательной линии Modbus поле Адреса содержит только адрес ведомого устройства.

Как описано вы предыдущем разделе, адреса ведомых устройств лежат в пределе от 1 до 247. Ведущее устройство обращается к ведомому устройству помещая адрес ведомого в поле адреса сообщения. Когда ведомое устройство возвращает свой ответ, оно помещает свой адрес в поле адреса ответа, чтобы сообщить ведущему устройству какое ведомое устройство отвечает.

Код функции сообщает какое действие необходимо выполнить. За кодом функции следует поле данных, которое содержит параметры запроса и ответа.

После поля данных идет поле проверки на наличие ошибок CRC.

| Адрес | Код функции | Данные | CRC |
|---|---|---|---|
| 1 байт | 1 байт | От 0 до 252 байт | 2 байта<br>CRC Low; CRC Hi |

Рисунок 1.3 – Размеры элементов ADU Modbus

Максимальный размер ADU Modbus RTU 256 байт.

Передающее устройство помещает ADU Modbus RTU в **кадр**, который имеет известные начальные и конечные точки. Это позволяет устройствам, получающим новый кадр, начинать с начала сообщения и знать, когда сообщение будет завершено.

Передача байт данных в пределах кадра производится последовательно с промежутком времени между передачей не более 1,5 времени передачи одного символа при текущей скорости. В протоколе используется повременная синхронизация начала и завершения передачи.

Перед началом передачи очередного кадра должна быть задержка по времени равная 3,5 времени передачи одного символа (3,5T) или более после завершения передачи предыдущего кадра. Если интервал меньше 3,5T преобразователь воспринимает второй кадр, как часть первого. В результате кадры смешиваются, CRC покажет ошибку, что приводит к сбою связи.

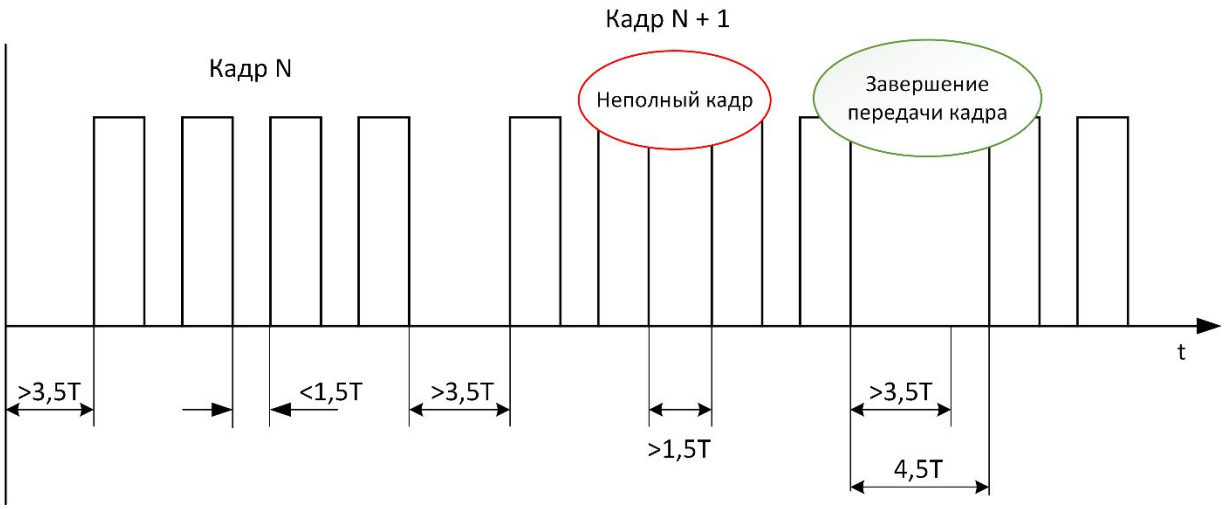


Рисунок 1.4 – Интервалы при передаче кадров Modbus

Завершением передачи кадра является 1,5 времени передачи одного символа (1,5T). Если по истечении времени 1,5T в течении времени 3,5T возобновится передача данных, кадр считается неполным и отклоняется получателем.

Сообщение Modbus

| Старт | Адрес | Код функции | Данные | CRC | Конец |
|---|---|---|---|---|---|
| ≥3,5 символа | 8 бит | 8 бит | N x 8 бит | 16 бит | ≥3,5 символа |

Рисунок 1.5 – Формат кадра Modbus RTU

### 1.3 Формат сообщений RTU

Когда устройства обмениваются данными по последовательной линии MODBUS в режиме RTU (Remote Terminal Unit), каждый 8–разрядный байт в сообщении содержит два 4–разрядных шестнадцатеричных символа. Основное преимущество этого режима заключается в том, что его большая плотность символов обеспечивает более высокую пропускную способность по сравнению с режимом ASCII при той же скорости передачи данных в бодах. Каждое сообщение должно передаваться непрерывным потоком символов.

Формат для каждого байта (11 бит) в RTU режиме:

1 стартовый бит

8 бит с данными

1 бит проверки четности

1 стоповый бит

Последовательная передача символов:

| | | | |
|---|---|---|---|
| Старт бит | 8 бит данных | Бит проверки четности | Стоп бит |

Для согласования с другими устройствами, возможно работа без бита четности, при этом используется два стоп-бита.

| | | | |
|---|---|---|---|
| Старт бит | 8 бит данных | Стоп бит | Стоп бит |

Все устройства, подключенные к одной сети, должны иметь одинаковую скорость передачи данных [F10.01] и один формат передачи данных [F10.02]

## 2 Подключение E5-mini

Обмен данными по протоколу Modbus осуществляется по интерфейсу RS485, клеммы «A+» и «A-». На концах линий необходимо устанавливать согласующие резисторы. Чтобы включить согласующие резисторы на ПЧ E5-mini, нужно поставить перемычки «RS485» в верхнее положение. Схема подключения на рисунке 2 и 3.

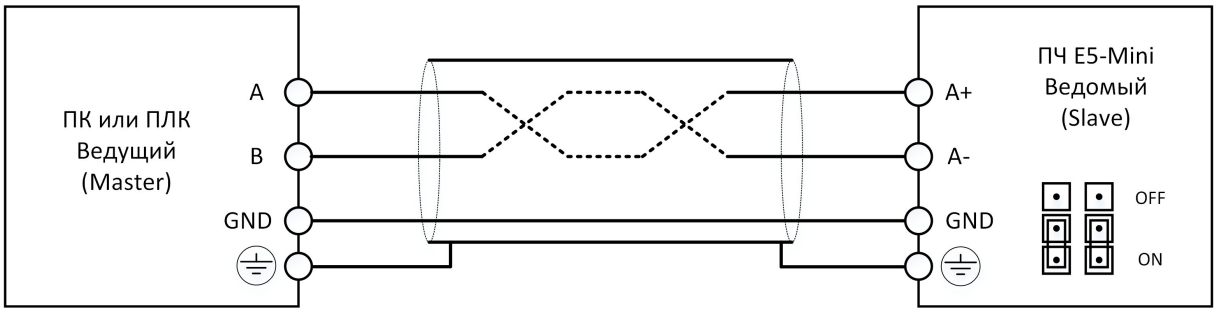


Рисунок 2.1 – Подключение в сеть одного преобразователя E5-mini

Согласующие резисторы нужны для предотвращения отражения сигнала от конца линии связи. Рекомендуется использовать экранированную витую пару с волновым сопротивлением 120 Ом. Расстояние между витой парой и силовыми кабелями должно быть не менее 50 см. Пересекаться витая пара с силовыми кабелями должна строго под прямым углом. Максимальная длина линии 1200 м.

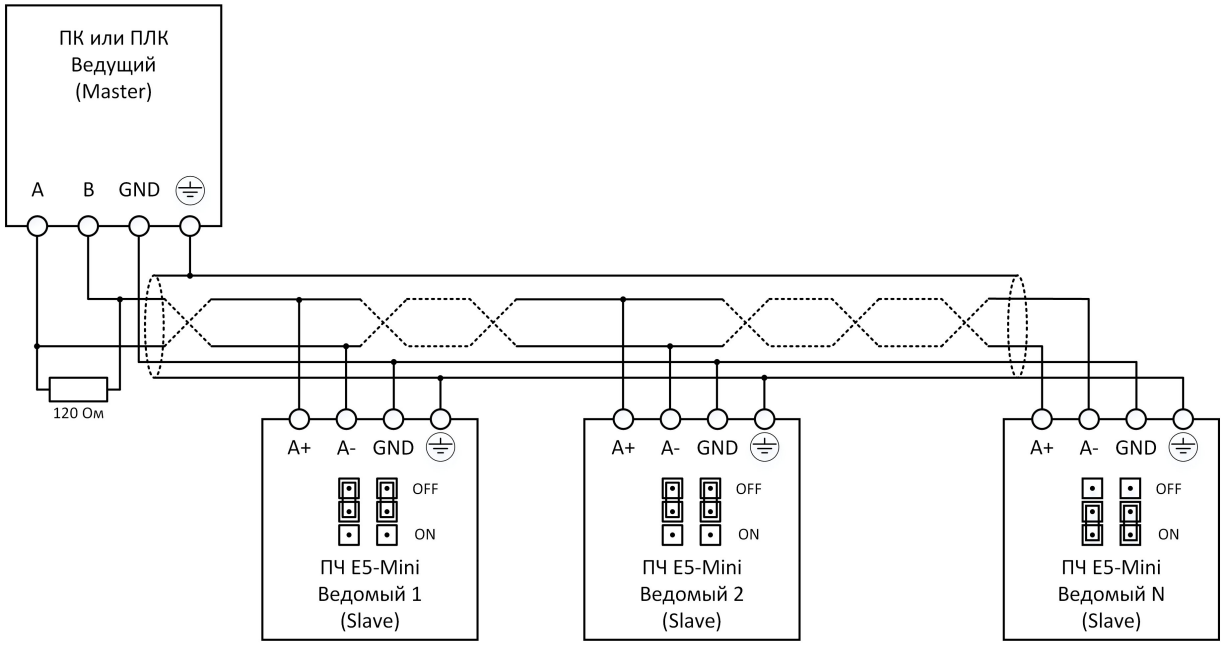


Рисунок 2.2 – Подключение в сеть нескольких преобразователей E5-mini

## 3 Основные настройки преобразователя

Для настройки связи необходимо настроить группу параметров F10. Преобразователю необходимо задать свой адрес в сети, скорость, формат данных.

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F10.00** | Адрес станции в сети связи Modbus | 1~247 (0 — одновременное обращение ко всем устройствам в сети) | – | 1 |

Для работы в сети преобразователь должен иметь свой уникальный адрес. Адрес «0» используется для широковещательной рассылки.

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F10.01** | Скорость передачи данных Modbus | 0: 4800<br>1: 9600<br>2: 19200<br>3: 38400<br>4: 57600<br>5: 115200 | бод/с | 1 |

Преобразователь E5-mini поддерживает 6 скоростей передачи данных. Скорости всех устройств в линии связи должны быть одинаковы.

В качестве примера возьмем [F10.01] = 9600 бод/с. По умолчанию каждый байт состоит из допустимы восьмиразрядных данных (например 0x01). Если в реальной ситуации требуется передать 10-разрядные данные, то время передачи будет примерно 1.04 микросекунды (10бит/9600).

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F10.02** | Формат данных Modbus | **0:** 1-8-N-1 (1 стартовый бит + 8 бит данных + 1 стоповый бит)<br>**1:** 1-8-E-1 (1 стартовый бит + 8 бит данных + 1 четность + 1 стоповый бит)<br>**2:** 1-8-O-1 (1 стартовый бит + 8 бит данных + 1 бит четности + 1 стоповый бит)<br>**3:** 1-8-N-2 (1 стартовый бит + 8 бит данных + 2 стоповых бита)<br>**4:** 1-8-E-2 (1 стартовый бит + 8 бит данных + 1 четность + 2 стоповых бита)<br>**5:** 1-8-O-2 (1 стартовый бит + 8 бит данных + 1 бит четности + 2 стоповых бита) | – | 0 |

При передаче по протоколу Modbus данные обычно состоят из начального бита, данных (8 бит), контрольного бита (опционально) и стоп-бита. Преобразователь частоты E5-mini поддерживает 6 форматов данных в соответствии с используемыми при передаче комбинациями Modbus-RTU.

| Старт бит | Данные | | | | | | | | Бит проверки | Стоп бит |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | N/O/E | 1 |

Если [F10.02=0], означает что текущие данные состоят из одного начального бита + восьми бит данных + нет контрольного бита + одного стоп бита.

★ N (НЕТ), нет проверки четности; E (ЧЕТ), бит проверки четности; O (НЕЧЕТ), бит проверки нечетности.

Формат данных для всех устройств в одной сети должен быть одинаковым.

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F10.03** | Тайм-аут связи Modbus | 0.0～60.0;<br>0.0: не активно | с | 0.0 |

Эта функция применяется для контроля линии связи в случаях, когда ведется непрерывный обмен данными. Если интервал с последнего полученного сообщения превысит время, указанное в [F10.03]. То ПЧ выдаст ошибку «E16».

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F10.04** | Задержка ответа Modbus | 1 ~ 20 | мс | 2 |

Задержка ответа определяется, как интервал времени с момента получения преобразователем пакета данных до момента начала анализа данных и их отсылки, чтобы обеспечить стабильную работу чипа протокола, настройка задержки ответа должна быть в пределах 1–20 мс (не может быть 0). Если данные связи включают работу EEPROM, фактическое время задержки ответа будет увеличено, а именно «время работы EEPROM + [F10.04]».

После настройки связи в ПЧ E5-Mini, для непосредственного управления нужно настроить следующие параметры:

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F00.02** | Источник команды пуск | 0: Пульт управления<br>1: Клеммы<br>**2: Порт-RS485** | – | 0 |

Пуск/Стоп и другие команды отправляются в регистр 7000h в соответствии с Таблицей 5.1

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F00.04** | Источник основной частоты A | 0: Кнопки пульта управления<br>1: Клемма AI1<br>**6: Порт-RS485 (Проценты)**<br>**7: Порт-RS485 (Значение)**<br>8: Цифровой потенциометр | – | 8 |

При значении «6» используется регистр 7001h. Максимальная частота 100.00% равна частоте, указанной в параметре [F00.16]. Пример задания 100.00% = 10000(dec) = 2710(hex)

При значении «7» используется регистр 7015h. Максимальная частота по умолчанию 50.00 Гц.  Пример задания 42.00 Гц = 4200(dec) = 1068(hex)

| Код параметра | Наименование | Описание параметра, диапазон | Единицы измерения | Значение по умолчанию |
|---|---|---|---|---|
| **F09.00** | Источник задания ПИД | 0: Цифровое задание F09.01<br>1: Клемма AI1<br>**6: Порт-RS485** | – | 0 |

При задании ПИД по последовательной линии связи используется регистр 7004h. Значение задается в процентах от предела измерения датчика. Если датчик давления на 10 бар, а поддерживать нужно 4 бара, то в регистр отправляется 40.00%. 40.00% = 4000(dec) = 0FA0(hex)

# 4 Описание функциональных кодов

Основная функция протокола Modbus – считывание и запись параметров. Различные коды функций обрабатывают различные запросы.

В таблице ниже приведены функции Modbus, управляемые при помощи преобразователей серии E5-Mini и их ограничения. Считывание и запись рассматриваются с позиции ведущего устройства.

Таблица 4.1 – Коды функций в преобразователе частоты E5-Mini

| Код функции | Функциональный код неисправности | Описание |
|---|---|---|
| 03 | 83 | Считывает несколько регистров |
| 41 | C1 | Записывает один регистр или команду без сохранения в EEPROM |
| 42 | C2 | Записывает несколько регистров без сохранения в EEPROM |
| 08 | 88 | Проверка линии связи |
| 06 | 86 | Записывает один регистр или команду |
| 10 | 90 | Записывает несколько регистров или команд |

### 4.1 Функциональный код 0x03

PDU запроса описывает адрес начального регистра и количество регистров для считывания. Данные регистра в ответном сообщении разделены на два байта в каждом регистре. Первый байт каждого регистра содержит старшие биты, а второй байт содержит младшие биты.

PDU запрос:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x03 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 1-16 |

PDU ответ:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x03 |
| Адрес стартового регистра | 2 байта | 2*N |
| Количество регистров | N*2 байта | |

N = количество регистров.

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0x83 |
| Код исключения | 1 байт | 01, 02, 03 или 04 |

Ниже приведен пример чтения регистров с F19.00 по F19.05 (состояние ПЧ при последней ошибке):

Таблица 4.2 – Пример чтения регистров E5-Mini с помощью функции 0x03

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 03 | Код функции | 03 | Код ошибки | 83 |
| Адрес стартового регистра (Hi) | 13 | Количество байт | 0C | Код исключения | 03 |
| Адрес стартового регистра (Lo) | 00 | Значение регистра 1 [F19.00] (Hi) | 00 | | |
| Количество регистров (Hi) | 00 | Значение регистра 1 [F19.00] (Lo) | 11 | | |
| Количество регистров (Lo) | 06 | Значение регистра 2 [F19.01] (Hi) | 00 | | |
| | | Значение регистра 2 [F19.01] (Lo) | 00 | | |
| | | Значение регистра 3 [F19.02] (Hi) | 00 | | |
| | | Значение регистра 3 [F19.02] (Lo) | 00 | | |
| | | Значение регистра 4 [F19.03] (Hi) | 01 | | |
| | | Значение регистра 4 [F19.03] (Lo) | 2C | | |
| | | Значение регистра 5 [F19.04] (Hi) | 00 | | |
| | | Значение регистра 5 [F19.04] (Lo) | 00 | | |
| | | Значение регистра 6 [F19.05] (Hi) | 00 | | |
| | | Значение регистра 6 [F19.05] (Lo) | 00 | | |

Согласно полученным из регистров данным, последнее аварийное сообщение E17(0011H): защита по перегреву, при котором выходная частота составляет 0.00 Гц [F19.01], выходной ток 0.00 А [F19.02], напряжение на звене постоянного тока 300 В [012CH], статус ПЧ – ожидание [F19.04], а рабочее время 0.0 часов [F19.05].

### 4.2 Функциональный код 0x41

Используется для записи регистра без использования энергонезависимой памяти. PDU запроса описывает адрес регистра и его данные.

PDU запрос:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x41 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 0x0000 – 0xFFFF |

PDU ответ:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x41 |
| Адрес стартового регистра | 2 байта | 2*N |
| Количество регистров | 2 байта | 0x0000 – 0xFFFF |

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0xC1 |
| Код исключения | 1 байт | См. таблицу 4.8 |

Ниже приведен пример запроса на изменение основной частоты A (7001H) на – 50.00%:

Таблица 4.3 – Пример записи регистра E5-Mini с помощью функции 0x41

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 41 | Код функции | 41 | Код ошибки | C1 |
| Адрес регистра (Hi) | 70 | Адрес регистра (Hi) | 70 | Код исключения | 03 |
| Адрес регистра (Lo) | 01 | Адрес регистра (Lo) | 01 | | |
| Значение регистра (Hi) | EC | Значение регистра 1 [F19.00] (Hi) | EC | | |
| Значение регистра (Lo) | 78 | Значение регистра 2 [F19.01] (Lo) | 78 | | |

Этот функциональный код нельзя использовать для изменения параметров атрибута “x” (он не может быть изменен во время работы). То есть могут быть изменены только параметры атрибута “o” (он может быть изменен во время работы). В противном случае будет возвращен код ошибки 1.

Код функции 0x41 соответствует стандартному коду функции 0x06 (такой же запрос, ответ и ошибка PDU). Разница состоит лишь в том, что ведомое устройство реагируя на этот функциональный код изменяет соответствующее значение только в оперативной памяти и не задействует EEPROM. Для параметров, которые часто перезаписываются (например, [F00.14] время разгона) рекомендуется использовать код функции 0x41.

### 4.3 Функциональный код 0x42

Данный функциональный код используется для записи последовательных регистров без использования постоянной энергонезависимой памяти. PDU запроса содержит стартовый регистр, после него идут данные, которые необходимо записать по 2 байта для каждого регистра. В обычном ответе ведомое устройство отправит код функции, начальный адрес и количество записанных регистров.

PDU запрос:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x42 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 1–16 |
| Количество байт | N*2 байта | |

N = количество регистров.

PDU ответ:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x41 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 1–16 |

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0xC2 |
| Код исключения | 1 байт | См. таблицу 4.8 |

Ниже приведен пример для изменения времени разгона 1 [F00.14] до 5.00 секунд и времени разгона 2 [F00.15] до 6.00 секунд.

Таблица 4.4 – Пример записи регистров E5-Mini с помощью функции 0x42

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 42 | Код функции | 42 | Код ошибки | C2 |
| Адрес стартового регистра (Hi) | 00 | Адрес стартового регистра (Hi) | 00 | Код исключения | 03 |
| Адрес стартового регистра (Lo) | 0E | Адрес стартового регистра (Lo) | 0E | | |
| Количество регистров (Hi) | 00 | Количество регистров (Hi) | 00 | | |
| Количество регистров (Lo) | 02 | Количество регистров (Lo) | 02 | | |
| Количество байт | 04 | | | | |
| Значение регистра 1 (Hi) [F00.14] | 01 | | | | |
| Значение регистра 1 (Lo) [F00.14] | F4 | | | | |
| Значение регистра 2 (Hi) [F00.15] | 02 | | | | |
| Значение регистра 2 (Lo) [F00.15] | 58 | | | | |

Этот функциональный код нельзя использовать для изменения параметров атрибута “**x**” (он не может быть изменен во время работы). То есть могут быть изменены только параметры атрибута “**o**” (он может быть изменен во время работы). В противном случае будет возвращен код ошибки 1.

### 4.4 Функциональный код 0x08

Данный функциональный код включает в себя тест для проверки связи между ведущим и ведомым устройством. С помощью кода диагностики невозможно получить доступ логике и значениям регистров. Счетчик ошибок в ведомом устройстве может быть удаленно сброшен с помощью некоторых функций.

Основной функцией диагностики в ПЧ E5-mini, является линейная диагностика (0000), которая используется для проверки нормальной связи между хостом и подчиненным устройством. Нормальным ответом на запрос о возврате данных запроса является возврат тех же данных. В то же время коды функций и подфункций также копируются.

PDU запрос:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x08 |
| Код подфункции | 2 байта | 0x0000 – 0xFFFF |
| Данные | 2 байта | 0x0000 – 0xFFFF |

PDU ответ:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x08 |
| Код подфункции | 2 байта | 0x0000 – 0xFFFF |
| Данные | 2 байта | 0x0000 – 0xFFFF |

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0x88 |
| Код исключения | 1 байт | См. таблицу 4.8 |

Код подфункции:

| Код подфункции | Значение | Поле запроса | Поле ответа |
|---|---|---|---|
| 0000 | Возврат данных запроса | Любое | Копирование данных запроса |
| …. | | | |

0000: возврат данных, переданных в поле запроса в ответ. Все сообщения должны соответствовать сообщению запроса.

В таблице ниже приведен пример запроса к ведомому устройству с использованием подфункции 0000. Ответ устройства представляет собой поле данных размером 2 байта.

Таблица 4.5 – Пример диагностики линии E5-Mini с помощью функции 0x08

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 08 | Код функции | 08 | Код ошибки | 88 |
| Код подфункции (Hi) | 00 | Код подфункции (Hi) | 00 | Код исключения | 03 |
| Код подфункции (Lo) | 00 | Код подфункции (Lo) | 00 | | |
| Данные (Hi) | A5 | Данные (Hi) | A5 | | |
| Данные (Lo) | 37 | Данные (Lo) | 37 | | |

### 4.5 Функциональный код 0x06

Этот функциональные код используется для записи в единый регистр хранения. В PDU содержатся адрес ведомого устройства, адрес регистра и данные. Обычный ответ на запрос, это возврат PDU после записи содержимого регистра.

PDU запроса:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x06 |
| Адрес регистра | 2 байта | 0x0000 – 0xFFFF |
| Данные | 2 байта | 0x0000 – 0xFFFF |

PDU ответа:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x06 |
| Адрес регистра | 2 байта | 0x0000 – 0xFFFF |
| Данные | 2 байта | 0x0000 – 0xFFFF |

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0x86 |
| Код исключения | 1 байт | См. таблицу 4.8 |

Функциональный код 0x06 нельзя использовать при частом изменении данных в регистрах.

Ниже приведен пример для изменения частоты в параметре [F00.07] до 50.00 Гц. Как только ведомое устройство получит команду с кодом функции 0x06, данные запишутся в EEPROM и сохранятся после отключения питания.

Таблица 4.6 – Пример записи данных в регистр E5-Mini с помощью функции 0x06

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 06 | Код функции | 06 | Код ошибки | 86 |
| Адрес регистра (Hi) | 00 | Адрес регистра (Hi) | 00 | Код исключения | См. таблицу 4.8 |
| Адрес регистра (Lo) | 07 | Адрес регистра (Lo) | 07 | | |
| Значение регистра (Hi) | 13 | Значение регистра (Hi) | 13 | | |
| Значение регистра (Lo) | 88 | Значение регистра (Lo) | 88 | | |

### 4.6 Функциональный код 0x10

Этот функциональный код используется для записи последовательных регистров в ведомом устройстве. Значение, которое требуется записать находится поле данных запроса. Данные каждого регистра делятся на два байта. В обычном ответе будет возвращен код функции, начальный адрес и количество записанных регистров.

PDU запрос:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x10 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 1–16 |
| Количество байт | 1 байт | 2*N |
| Значение регистра | N*2 байта | |

N = количество регистров.

PDU ответ:

| | | |
|---|---|---|
| Код функции | 1 байт | 0x41 |
| Адрес стартового регистра | 2 байта | 0x0000 – 0xFFFF |
| Количество регистров | 2 байта | 1–16 |

PDU сообщающий об ошибке:

| | | |
|---|---|---|
| Код ошибки | 1 байт | 0x90 |
| Код исключения | 1 байт | См. таблицу 4.8 |

Ниже приведен пример запроса для изменения данных в регистрах, отвечающих за параметры [F02.02] «Дискретный вход X3» и [F02.03] «Дискретный вход X4». На дискретные входы будут назначены функции «Команда больше» и «Команда меньше».

Таблица 4.7 – Пример записи данных в регистры E5-Mini с помощью функции 0x10

| Запрос | | Ответ | | | |
|---|---|---|---|---|---|
| Имя домена | (0x) | Имя домена | (0x) | Имя домена | (0x) |
| Код функции | 10 | Код функции | 10 | Код ошибки | 90 |
| Адрес стартового регистра (Hi) | 02 | Адрес стартового регистра (Hi) | 02 | Код исключения | 03 |
| Адрес стартового регистра (Lo) | 02 | Адрес стартового регистра (Lo) | 02 | | |
| Количество регистров (Hi) | 00 | Количество регистров (Hi) | 00 | | |
| Количество регистров (Lo) | 02 | Количество регистров (Lo) | 02 | | |
| Количество байт | 04 | | | | |
| Значение регистра 1 (Hi) [F00.14] | 00 | | | | |
| Значение регистра 1 (Lo) [F00.14] | 06 | | | | |
| Значение регистра 2 (Hi) [F00.15] | 00 | | | | |
| Значение регистра 2 (Lo) [F00.15] | 07 | | | | |

Функциональный код 0x10 так же, как и код 0x06 нежелательно использовать для частой перезаписи параметров ПЧ.

### 4.7 Сообщения с кодами исключения

Когда ведущее устройство отправляет запрос на ведомые, главная станция ожидает нормального ответа. Запрос главной станции может привести к одному из событий:

- Ведомое устройство получило запрос без ошибок, и он может быть обработан должным образом, ведомое устройство вернет обычный ответ
- Если ведомое устройство не получит запрос из-за ошибок связи, сообщение не будет возвращено. Это будет расценено ведомым устройством как тайм-аут.
- Если ведомое устройство получает запрос без ошибок связи, но не может обработать запрос (например, запрос на считывание несуществующего регистра), то ведомое устройство возвращает ответ с кодом исключения и ведущее устройство будет проинформировано об ошибке. Ответное сообщение об ошибке содержит два поля, отличные от полей.
- Поле функционального кода: в обычном ответе ведомое устройство копирует функциональный код исходного запроса в соответствующее поле функционального кода. Значения MSB для всех функциональных кодов равны 0. В сообщении об ошибке ведомое устройство устанавливает значение MSB для кода функции равным 1. То есть код функции реагирования на исключение = код функции нормального реагирования + 0x80.
- Поле данных: Ведущее устройство может возвращать данные из поля данных в обычном ответе и код исключения в ответе на исключение. Определенные коды исключений подробно описаны в таблице 4.8 «Определения кодов исключений»

Таблица 4.8 – Значение кодов исключения в преобразователе частоты E5-Mini

| Код ошибки | Название | Значение |
|---|---|---|
| 01h | Некорректная функция | Функциональный код, полученный ведомым устройством за пределами заданного диапазона. |
| 02h | Некорректный адрес для передачи данных | Адрес данных, полученный ведомым устройством, недопустим. В частности, недопустима комбинация стартового регистра и длины данных. |
| 03h | Некорректный кадр данных | Ведущее устройство обнаружило неправильную длину кадра данных запроса или проверку CRC. |
| 04h | Защита ведомого устройства | Когда ведомое устройство пытается выполнить запрошенную операцию, возникает неустранимая ошибка. Это может быть вызвано логической ошибкой, невозможностью записи в EEPROM и т.д. |
| 05h | Превышение диапазона передачи данных | Данные, полученные ведомым устройством, не находятся между минимальным и максимальным значениями соответствующего регистра. |
| 06h | Параметр только для чтения | Запрашиваемый регистр предназначен только для чтения. |
| 07h | Неизменяемый параметр при запуске | Когда преобразователь частоты находится в работе, данный регистр запрещено записывать. Необходимо остановить преобразователь. |
| 08h | Параметр защищен паролем | Запрашиваемый регистр защищен паролем. |

### 4.8 Определение длины данных кадра

Часть PDU кадра RTU сообщения Modbus способная считывать/записывать 1-16 регистров. Для различных кодов функции фактическая длина кадра RTU варьируется в соответствии с таблицей ниже:

Таблица 4.9 – Соотношение длины кадра между функциональными кодами и длиной кадра RTU в преобразователе E5-Mini

| Код функции (0x) | Длина кадра RTU (байт) | | | Максимальная длина (Байты) |
|---|---|---|---|---|
| | Запрос | Ответ | Ответ с кодом исключения | |
| 03 | 8 | $5+2N_r$ | 5 | 37 |
| 41(06) | 8 | 8 | 5 | 8 |
| 08 | 8 | 8 | 5 | 8 |
| 42(10) | $9+N_w$ | 8 | 5 | 41 |

# 5 Назначение адресов регистров в ПЧ E5-Mini

Таблица 5.1 – подробная информация адресов регистров преобразователя E5-Mini

| Адреса регистров | | Описание | |
|---|---|---|---|
| Регистры 0000h–6F63h | | Для параметров FXX.YY старший байт равен шестнадцатеричному значению XX и младший байт шестнадцатеричному значению YY. Например, адрес параметра [F00.14] равен 000EH (00d=00h/14d=0Eh) | |
| Регистры 8000h–EF63h (не сохраняются после отключения питания) | | Если параметры заданы с помощью функционального кода 0x06 или 0x10 | |
| Команды управления ПЧ 7000h – 71FFh **(Только запись)** | 7000h | 0000h | Нет команды |
| | | 0001h | Пуск вперед |
| | | 0002h | Пуск назад |
| | | 0003h | Шаговая скорость вперед |
| | | 0004h | Шаговая скорость назад |
| | | 0005h | Торможение до останова |
| | | 0006h | Экстренное торможение [F15.40] |
| | | 0007h | Останов выбегом |
| | | 0008h | Сброс аварии |
| | | 0009h | Изменение направления вращения |
| | | 000Bh | Останов шагового скорости |
| | | 000Ch–00FFh | Резерв |
| | 7001h | Задание основной частоты «A» в процентах | -100.00% – 100.00% (100% = максимальная частота [F00.16]), если [F00.04]=6. |
| | 7002h | Задание вспомогательной частоты «B» в процентах | -100.00% – 100.00% (100% = максимальная частота [F00.16]), если [F00.05=6] |
| | 7004h | Опорное значение ПИД | -100.00% – 100.00% |
| | 7005h | Обратная связь ПИД | -100.00% – 100.00% |
| | 7006h | Опорное значение напряжения в режиме разделения V/f | 0.00% – 100.00% |
| | 7007h-700Eh | Резерв | |
| | 700Fh | Настройка связи между ведущим и ведомым устройством | -100.00% – 100.00% |
| | 7010h-7013h | Резерв | |
| | 7014h | Внешняя защита | Регистр для сигнала защиты с внешнего устройства |
| | 7015h | Задание основной частоты «A» | 0.00 – Максимальная частота |
| | 7016h | Задание вспомогательной частоты «B» | 0.00 – Максимальная частота |
| | 7017h-70FFh | Резерв | |

| | | | | |
|---|---|---|---|---|
| | 7200h Состояние 1 ПЧ | Бит 7-0 Состояние ПЧ при работе | 00h | Ожидание команды |
| | | | 01h | ПЧ в работе |
| | | | 02h | Работа с шаговой скоростью |
| | | | 03h | Автонастройка |
| | | | 04h | Останов ПЧ |
| | | | 05h | Останов шаговой скорости |
| | | | 06h | Авария ПЧ |
| | | | 07h-0FFh | Резерв |
| | | Бит 15-8 Информация об неисправностях | 00h | Нормальная работы ПЧ |
| | | | xxh | Аварийное состояние ПЧ, где «xx» код неисправности |
| | 7201h Состояние 2 ПЧ | Бит 0 направление установки частоты | 1 | Отрицательное направление |
| | | | 0 | Положительное направление |
| | | Бит 1 направление вращения | 1 | Отрицательная выходная частота |
| | | | 0 | Положительная выходная частота |
| | | Бит 3-2 режим работы | 00 | Режим управления скоростью |
| | | | 10 | Резерв |
| | | | 11 | Резерв |
| | | Бит 4 Защита параметров | 1 | Защита параметров |
| | | | 0 | Нет защиты параметров |
| | | Бит 6-5 | Резерв | |
| | | Бит 8-7 Источник команды «Пуск» | 00 | Пульт ПЧ |
| | | | 01 | Дискретные входы |
| | | | 10 | Последовательный порт |
| | | | 11 | Резерв |
| | | Бит 9 | Резерв | |
| | | Бит 10 | 0 | Нет предупреждений |
| | | | 1 | Предупреждение [7230h] |
| | | Бит 15-10 | Резерв | |
| | 7202h | Резерв | | |
| | 7203h | Выходная частота | | xx.xx Гц |
| | 7204h | Выходное напряжение | | xxx.x В |
| | 7205h | Выходная мощность | | |
| | 7206h | Скорость вращения об/мин | | xxxx об/мин |
| | 7207h | Напряжение ЗПТ | | xxx В |
| | 7208h | Резерв | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 7209h | Дискретные входы | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | * | * | * | * | * | * | * | * |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | * | * | * | * | X4 | X3 | X2 | X1 |
| | 720Ah | Виртуальные входы | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | VX8 | VX7 | VX6 | VX5 | VX4 | VX3 | VX2 | VX1 |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | * | * | * | * | * | * | * | * |
| | 720Bh | Дискретные выходы | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | * | * | * | * | * | * | * | * |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | * | * | * | * | * | Y1 | * | R1 |
| | 720Ch | Виртуальные дискретные выходы | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | VY8 | VY7 | VY6 | VY5 | VY4 | VY3 | VY2 | VY1 |
| | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | * | * | * | * | * | * | * | * |

| | | | |
|---|---|---|---|
| | 720Dh | Предпоследняя ошибка (1) | [F19.06] |
| | 720Eh | Наиболее давняя ошибка (2) | [F19.12] |
| | 720Fh | Последняя ошибка (3) | [F19.00] |
| | 7210h | Выходная частота при последней неисправности (1) | [F19.01] |

| | | | |
|---|---|---|---|
| | 7211h | Выходной ток при последней неисправности (1) | [F19.02] |
| | 7212h | Напряжение ЗПТ при последней неисправности (1) | [F19.03] |
| | 7213h | Состояние ПЧ при последней неисправности (1) | [F19.04] |
| | 7214h | Время наработки при последней неисправности (1) | [F19.05] |
| | 7215h | Установленное время разгона | [F00.14] |
| | 7216h | Установленное время торможения | [F00.15] |
| | 7217h - 7219h | Резерв | |
| | 7224h | Выходной ток | |
| | 7225h | Установленная частота | |
| | 7228h | Суммарное время работы | |
| | 722Fh | Код сообщения о неисправности | |
| | 7230h | Код сообщения предупреждения | 0: нет предупреждений; остальное: текущий код предупреждения |
| | 7231h-73FFh | Резерв | |
| Остальное | Резерв | | |

## 6 проверка CRC

Младший байт проверки CRC стоит перед высоким байтом.

Сначала передающее устройство вычисляет значение CRC, которое включается в отправляемое сообщение. Получающее устройство повторно рассчитывает значение CRC и сравнивает вычисленное значение с полученным CRC. Если эти два значения не равны, значит в процессе передачи произошла ошибка.

Процесс расчета проверки CRC:

[1] Определение регистра CRC и присваивание ему начального значения, FFFFh.

[2] Выполните вычисление исключения с использованием первого байта переданного сообщения и значения регистра CRC и сохраните результат в регистре CRC. Начиная с кода адреса, начальный и конечный биты в вычислении не участвуют.

[3] Извлеките и проверьте LSB (младший значащий бит регистра CRC).

[4] Если LSB равен 1, то каждый бит регистра CRC сдвигается вправо на один бит, а к старшему значащему биту добавляется 0. Выполните исключающее вычисление значения регистра CRC и A001H и сохраните результат в регистре CRC.

[5] Если LSB равен 0, то каждый бит регистра CRC сдвигается вправо на один бит, а к старшему значащему биту добавляется 0.

[6] Повторяйте шаги 3, 4 и 5, пока не завершите 8 смен.

[7] Повторяйте шаги 2, 3, 4, 5 и 6 для обработки следующего байта передаваемого сообщения, пока не будут обработаны все байты передаваемого сообщения. пока все байты информации не будут обработаны и переданы.

[8] После вычисления содержимое регистра CRC является значением проверки CRC.

[9] В системе с ограниченными временными ресурсами рекомендуется выполнять проверку CRC методом табличного поиска.

Простая функция CRC заключается в следующем (запрограммирована на языке Си):

```
unsigned int CRC_Cal_Value(unsigned char *Data, unsigned char Length)
{
    unsigned int crc_value = 0xFFFF;
    int i = 0;
    while(Length--)
    {
    crc_value ^= *Data++;
    for(i=0;i<8;i++)
    {
    if(crc_value & 0x0001)
    {
        crc_value = (crc_value>>1)^ 0xa001;
```

```
                }
        else
        {
              crc_value = crc_value>>1;
                }
             }
        }
        return(crc_value);
    }
```

Это описывает только теорию проверки CRC и требует длительного времени выполнения. Особенно если данные проверки большие, время вычисления будет слишком большим. Таким образом, следующие два метода поиска в таблице применяются для 16-разрядных и 8-разрядных контроллеров соответственно.

Поисковая таблица CRC16 для 8-разрядного процессора: (Старший байт в конечном результате работы этой программы находится впереди. Пожалуйста, измените его во время отправки.)

```
const Uint8 crc_l_tab[256] = {
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40
};
constUint8 crc_h_tab[256] = {
0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,0x05,0xC5,0xC4,0x04,
```

```
0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09,0x08,0xC8,
0xD8,0x18,0x19,0xD9,0x1B,0xDB,0xDA,0x1A,0x1E,0xDE,0xDF,0x1F,0xDD,0x1D,0x1C,0xDC,
0x14,0xD4,0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,0x11,0xD1,0xD0,0x10,
0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,0x37,0xF5,0x35,0x34,0xF4,
0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,0x3B,0xFB,0x39,0xF9,0xF8,0x38,
0x28,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,0xEA,0xEE,0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,
0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26,0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,
0xA0,0x60,0x61,0xA1,0x63,0xA3,0xA2,0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,0x64,0xA4,
0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68,
0x78,0xB8,0xB9,0x79,0xBB,0x7B,0x7A,0xBA,0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,
0xB4,0x74,0x75,0xB5,0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,
0x50,0x90,0x91,0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,
0x9C,0x5C,0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,0x59,0x58,0x98,
0x88,0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,
0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,0x40
};
Uint16CRC(Uint8 * buffer, Uint8 crc_len)
{
        Uint8 crc_i,crc_lsb,crc_msb;
        Uint16 crc;
        crc_msb = 0xFF;
        crc_lsb = 0xFF;
        while(crc_len--)
        {
                crc_i = crc_lsb ^ *buffer;
                buffer ++;
                crc_lsb = crc_msb ^ crc_l_tab[crc_i];
                crc_msb = crc_h_tab[crc_i];
        }
                crc = crc_msb;
                crc = (crc << 8) + crc_lsb;
                return crc;
        }
```

Таблица поиска CRC16 для 16-разрядного процессора: (Старший байт в конечном результате работы этой программы находится впереди. Пожалуйста, измените его во время отправки.)

```
const Uint16 crc_table[256] = {
0x0000,0xC1C0,0x81C1,0x4001,0x01C3,0xC003,0x8002,0x41C2,0x01C6,0xC006
,0x8007,0x41C7,0x0005,0xC1C5,0x81C4,0x4004,0x01CC,0xC00C,0x800D,0x41CD
,0x000F,0xC1CF,0x81CE,0x400E,0x000A,0xC1CA,0x81CB,0x400B,0x01C9,0xC009
,0x8008,0x41C8,0x01D8,0xC018,0x8019,0x41D9,0x001B,0xC1DB,0x81DA,0x401A
,0x001E,0xC1DE,0x81DF,0x401F,0x01DD,0xC01D,0x801C,0x41DC,0x0014,0xC1D4
,0x81D5,0x4015,0x01D7,0xC017,0x8016,0x41D6,0x01D2,0xC012,0x8013,0x41D3
,0x0011,0xC1D1,0x81D0,0x4010,0x01F0,0xC030,0x8031,0x41F1,0x0033,0xC1F3
,0x81F2,0x4032,0x0036,0xC1F6,0x81F7,0x4037,0x01F5,0xC035,0x8034,0x41F4
,0x003C,0xC1FC,0x81FD,0x403D,0x01FF,0xC03F,0x803E,0x41FE,0x01FA,0xC03A
,0x803B,0x41FB,0x0039,0xC1F9,0x81F8,0x4038,0x0028,0xC1E8,0x81E9,0x4029
,0x01EB,0xC02B,0x802A,0x41EA,0x01EE,0xC02E,0x802F,0x41EF,0x002D,0xC1ED
,0x81EC,0x402C,0x01E4,0xC024,0x8025,0x41E5,0x0027,0xC1E7,0x81E6,0x4026
,0x0022,0xC1E2,0x81E3,0x4023,0x01E1,0xC021,0x8020,0x41E0,0x01A0,0xC060
,0x8061,0x41A1,0x0063,0xC1A3,0x81A2,0x4062,0x0066,0xC1A6,0x81A7,0x4067
,0x01A5,0xC065,0x8064,0x41A4,0x006C,0xC1AC,0x81AD,0x406D,0x01AF,0xC06F
,0x806E,0x41AE,0x01AA,0xC06A,0x806B,0x41AB,0x0069,0xC1A9,0x81A8,0x4068
,0x0078,0xC1B8,0x81B9,0x4079,0x01BB,0xC07B,0x807A,0x41BA,0x01BE,0xC07E
,0x807F,0x41BF,0x007D,0xC1BD,0x81BC,0x407C,0x01B4,0xC074,0x8075,0x41B5
,0x0077,0xC1B7,0x81B6,0x4076,0x0072,0xC1B2,0x81B3,0x4073,0x01B1,0xC071
,0x8070,0x41B0,0x0050,0xC190,0x8191,0x4051,0x0193,0xC053,0x8052,0x4192
,0x0196,0xC056,0x8057,0x4197,0x0055,0xC195,0x8194,0x4054,0x019C,0xC05C
,0x805D,0x419D,0x005F,0xC19F,0x819E,0x405E,0x005A,0xC19A,0x819B,0x405B
,0x0199,0xC059,0x8058,0x4198,0x0188,0xC048,0x8049,0x4189,0x004B,0xC18B
,0x818A,0x404A,0x004E,0xC18E,0x818F,0x404F,0x018D,0xC04D,0x804C,0x418C
,0x0044,0xC184,0x8185,0x4045,0x0187,0xC047,0x8046,0x4186,0x0182,0xC042
,0x8043,0x4183,0x0041,0xC181,0x8180,0x4040};
Uint16 CRC16(Uint16 *msg , Uint16 len){
        Uint16 crcL = 0xFF , crcH = 0xFF;
        Uint16 index;
        while(len--){
        index = crcL ^ *msg++;
```

```
                crcL = ((crc_table[index] & 0xFF00) >> 8) ^ (crcH);
                crcH = crc_table[index] & 0xFF;
                }
        return (crcH<<8) | (crcL);
}
```