**СПЕЦИФИКАЦИЯ ПРИЛОЖЕНИЯ ПРОТОКОЛА MODBUS V1.1b**

Содержание

1 Введение

* 1. Область применения данного документа

1. Аббревиатуры
2. Контекст
3. Основное описание
   1. Описание протокола
   2. Кодирование данных
   3. Модель данных MODBUS
   4. Модель адресации MODBUS
   5. Определение передачи по MODBUS
4. Категории кодов функций
   1. Описание общедоступных кодов функций
5. Описание кодов функций
   1. 01 (0x01) Read Coils (Чтение ячеек)
   2. 02 (0x02) Read Discrete Inputs (Чтение дискретного входа)
   3. 03 (0x03) Read Holding Registers (Чтение сохраняющих регистров)
   4. 04 (0x04) Read Input Registers (Чтение входных регистров)
   5. 05 (0x05) Write Single Coil (Перезаписать единственную ячейку)
   6. 06 (0x06) Write Single Register (Перезаписать единственный регистр)
   7. 07 (0x07) Read Exception status (Serial Line only) (Считать статус ошибки - Для подключения через последовательный порт)

6.8.1 Коды подфункций, поддерживаемые промышленной линейкой устройств

6.8.2 Пример и диаграмма состояния

6.9 11 (0x0B) Get Comm Event Counter (Serial Line only) (Получить общий счетчик событий – Только при подключении через последовательный порт)

6.10 12 (0x0C) Get Comm Event Log (Serial Line only) (Получить общий лог событий – Для подключения через последовательный порт)

6.11 15 (0x0F) Write Multiple Coils (Запись в несколько ячеек)

6.12 16 (0x10) Write Multiple registers (Запись в несколько регистров)

6.13 17 (0x11) Report Slave ID (Serial Line only) (Сообщить ID подчиненного устройства)

6.14 20 (0x14) Read File Record (Чтение файловой записи)

6.15 21 (0x15) Write File Record (Создание файловой записи)

6.16 22 (0x16) Mask Write Register (Запись регистра через маску)

6.17 23 (0x17) Read/Write Multiple registers (Прочитать/Записать множество регистров)

6.18 24 (0x18) Read FIFO Queue (Прочитать очередь вызовов)

6.19 43 (0x2B) Encapsulated Interface Transport (Инкапсулированный транспорт через интерфейс)

6.20 43 / 13 (0x2B / 0x0D) CANopen General Reference Request and Response PDU (Запрос на возможность установки цепочки Запросов и Ответов)

6.21 43 / 14 (0x2B / 0x0E) Read Device Identification (Запросить состояние идентификации устройства)

7 Ошибочные ответы протокола – MODBUS

Annex A (Informative): Зарезервированные коды функций, подфункций и MEI типов MODBUS

Annex B (Informative): Основная ссылочная команда установки связи

1 Введение

* 1. Область применения этого документы

MODBUS – это протокол отправки сообщений уровня приложений, располагаемый на 7 уровне модели OSI, что обеспечивает клиент/серверное взаимодействие между устройствами, соединенными разными типами шин и сетей.

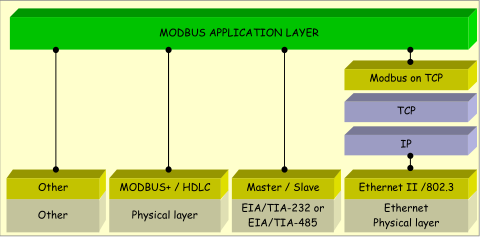
Являясь стандартом дэ факто в промышленном производстве с 1979, MODBUS продолжает обеспечивать миллионы автоматизированных устройств возможностью взаимодействия друг с другом. Сегодня, повсеместность распространения простой и элегантной структуры протокола продолжает расти. Мировое сообщество может получить доступ к протоколу MODBUS по зарезервированному системой порту 502 в TCP/IP стеке.

MODBUS – это двунаправленный протокол обмена, который предлагает возможности, предоставляемые по кодам функций. Функции кодов MODBUS – являются элементами MODBUS запрос/ответ PDUs (фрагмент данных протокола). Цель этого документа – описать коды функций, используемые совместно с каркасом MODBUS – транзакции.

MODBUS – это протокол обмена сообщениями по типу клиент/сервер уровня приложений, обеспечивающий связь между устройствами, соединенными разными типами шин и сетей.

На текущий момент реализовано использование:

* TCP/IP поверх Ethernet. Подробнее в “Описании реализации обмена сообщениями по MODBUS V1.0a
* Асинхронное серийное сообщение поверх множества сред (проводные: EIA/TIA-232-E, EIA-422, EIA/TIA-485-A; волокно, радио, и т.д.)
* MODBUS PLUS, высокоскоростная сеть на основе передачи токена.



Фигура 1. Стек передачи протокола MODBUS

Ссылки

1. RFC 791, Интернет Протокол, Sep81 DARPA
2. Аббревиатуры

**ADU** Application Data Unit (Фрагмент данных приложения)

**HDLC** High level Data Link Control (Высокоуровневое управление связью данных)

**HMI** Human Machine Interface (Человеко – машинный интерфейс)

**IETF** Internet Engineering Task Force ()

**I/O** Input/Output (Ввод/Вывод)

**IP** Internet Protocol (Интернет протокол)

**MAC** Medium Access Control (Контроль доступа устройств)

**MBAP** MODBUS Application Protocol (Протокол приложений MODBUS)

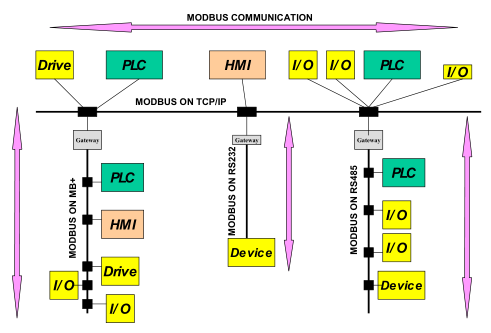
**PDU** Protocol Data Unit (Фрагмент данных протокола)

**PLC** Programmable Logic Controller (Программируемый логический контроллер)

**TCP** Transport Control Protocol (Транспортный протокол обмена)

1. Контекст

MOBDUS – протокол позволяет легко осуществлять взаимодействие через все типы сетевых архитектур.



Фигура 2. Пример архитектуры сети MODBUS

1. **Описание кодов функций**
   1. **01 (0x01) Read Coils (Считать значения ячеек)**

Этот код функции используется для считывания от 0 до 2000 смежных статусов ячеек на удаленном устройстве. Запрос фрагмента данных протокола (PDU) определяет начальный адрес, например, адрес первой ячейки; и количество ячеек. В PDU нумерация ячеек начинается с 0. Таким образом, ячейки пронумерованные как 1-16 адресно описываются как 0-15.

Ячейки в ответном сообщении упаковываются как одна ячейка на бит в поле данных. Статус определяется как 1= ВКЛЮЧЕНО и 0 = ВЫКЛЮЧЕНО. Меньший значащий бит первого байта данных содержит выход, адресованный и очереди. Другие ячейки следуют дальше в возрастающем порядке до конца байта и от нижнего порядка до высокого порядка во всех байтах последовательности.

Если возвращенное количество не делится на 8 без остатка, оставшиеся биты в последнем байте данных до заполняются нулями (в направлении к началу старшего бита). Поле Byte Count определяет количество заполненных данными байт.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | 0x01 |
| Начальный адрес | 2 байта | 0x0000 до 0xFFFF |
| Количество ячеек | 2 байта | 1 до 2000 (0x7D0) |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | 0x01 |
| Кол-во байт | 1 байт | N\* |
| Состояние ячеек | n байт | n = N или N+1 |

**\*N** = Количество ответов / 8, если остаток отличается от 0 => N = N+1

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | Код функции + 0x80 |
| Код исключения | 1 байт | 01 или 02 или 03 или 04 |

Ниже приведен пример запроса на чтение дискретных выходов 20-38:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | |  | |
| *Имя поля* | *(Hex – запись)* | *Имя поля* | *(Hex – запись)* |
| Функция | **01** | Функция | **01** |
| Начальный адрес Hi | **00** | Кол-во байт | **03** |
| Начальный адрес Lo | **13** | Состояние выходов 27-20 | **CD** |
| Кол-во ответов Hi | **00** | Состояние выходов 35-28 | **6B** |
| Кол-во ответов Lo | **13** | Состояние выходов 38-36 | **05** |

Состояние выходов с 27 – 20 представлено шестнадцатеричным значением CD, или бинарным 1100 1101. Выход 27 является MSB (наибольшим значащим битом) байта, и выход 20 является LSB (младшим битом).

Согласно соглашению, биты в байте отображаются по направлению от MSB старшего слева до LSB – младшего справа. Таким образом, выходы в первом байте отображаются от «27 к 20» слева направо. Следующий байт содержит выходы «от 35 к 28», слева направо. Так как биты передаются последовательно, они идут от LSB к MSB: 20 … 27, 28 … 35, и т.д.

В последнем байте данных статус выходов 38-36 представлены как шестнадцатеричное значение 05 или бинарное 0000 0101. Выход 38 находится на 6 битовой позиции слева, и выход 36 является LSB младшим значением в байте. Пять оставшихся выше расположенных по порядку битов заполнены нулями.

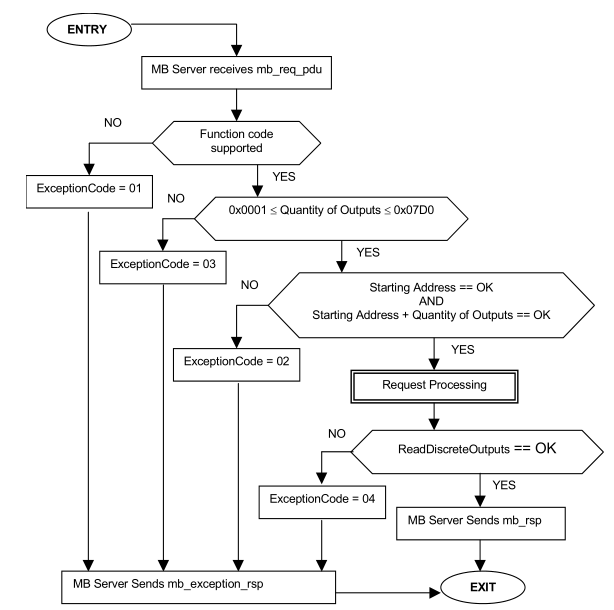


Рисунок 11. Диаграмма считывания состояния ячеек

Обобщенно коды исключений соответствуют следующим состояниям в протоколе:

ExceptionCode = 01 Код функции не поддерживается

ExceptionCode = 02 Начальный адрес задан правильно и Начальный адрес совместно с кол-вом выходов заданы правильно

ExceptionCode = 03 Количество запрошенных состояний цифровых выходов не находится в диапазоне от 0x0001 до 0x07D0

ExceptionCode = 04 Запрос на состояние цифровых выходов завершился неудачно

* 1. **02(0x02) Read Discrete Inputs (Считать значения цифровых входов)**

Этот функциональный код используется для считывания от 1 до 2000 статусов близкорасположенных дискретных входов на удаленном устройстве. Запрос данных протокола определяет начальный адрес, например, адрес первого входа, и кол-во входов. В передаваемом запросе адресация дискретных входов начинается с нуля. Таким образом, дискретные входы, пронумерованные с 1 – 16 адресно записываются как 0 – 15.

Дискретные входы в ответном сообщении записываются в форме – один вход на один бит в поле данных. Статус соответствует: 1 = ВКЛЮЧЕНО; 0 = ВЫКЛЮЧЕНО. Младшие биты в первом байте передаваемых данных сообщают о том, что запрошено состояние цифровых входов. Состояние входов записывается далее по направлению к старшему биту до конца текущего байта, и от младшего до старшего бита в каждом последующем байте последовательности.

Если возвращенное количество входов не является множителем 8-ми, оставшиеся биты в последнем байте заполняются нулями (по направлению к старшему биту в байте). Поле «Кол-во байтов» определяет кол – во полностью заполненных данными байтов.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x02** |
| Начальный адрес | 2 байта | 0x0000 до 0xFFFF |
| Количество входов | 2 байта | 1 до 2000 (0x7D0) |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x02** |
| Начальный адрес | 1 байт | **N\*** |
| Состояние входа | **N\*** x 1 байтов |  |

**\*N** = Количество входов / 8 если остаток отличается от 0 => N = N+1

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код ошибки | 1 байт | **0x82** |
| Код исключения | 1 байт | 01 или 02 или 03 или 04 |

Ниже приведен пример запроса на считывание состояния дискретных входов с 197 – 218:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Название поля* | *(Hex – запись)* | *Название поля* | *(Hex – запись)* |
| Функция | **02** | Функция | **02** |
| Начальный адрес Hi | **00** | Кол-во байт | **03** |
| Начальный адрес Lo | **C4** | Статус д. входов 204-197 | **AC** |
| Кол-во входов Hi | **00** | Статус д. входов 212-205 | **DB** |
| Кол-во входов Lo | **16** | Статус д. входов 218-213 | **35** |

Состояние дискретных входов 204-197 показано как запись в 1 байте AC значения в hex – форме, или как бинарное 1010 1100. Вход 204 является наибольшим в байте, вход 197 – наименьшим.

Состояние дискретных входов 218-213 представлено как запись в байте 35 hex – значения или бинарного представления 0011 0101. Вход 218 находится на месте 3-го бита слева, состояние входа 213 записывается в младшем бите.

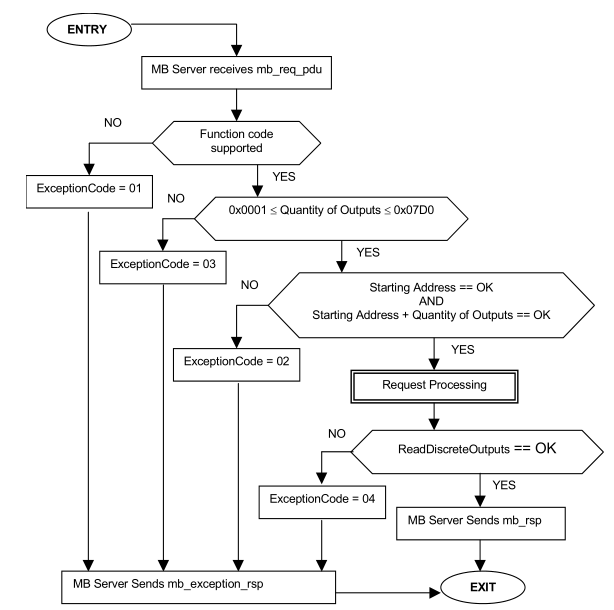
**Заметка:** 2 оставшихся бита (по направлению к старшему в конце) до заполняются нулями. ****

Рисунок 12. Диаграмма чтения состояния цифровых входов

* 1. **03 (0x03) Read Holding Registers (Считать состояние регистров хранения)**

Этот код функции используется для считывания данных близко расположенных блоков регистров хранения на удаленном устройстве. Фрагмент данных запроса определяет адрес начального регистра и кол-во регистров. В фрагменте данных адресация к регистрам выполняется, начиная с 0. Таким образом, регистры, пронумерованные с 1-16 адресно доступны как 0-15.

В ответном сообщении под данные каждого из регистров отводится по 2 байта памяти, заполненных бинарным содержимым, выровненном в каждом байте. Для каждого регистра первый байт содержит наибольшие значащие биты – второй содержит наименьшие значащие биты.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x03** |
| Начальный адрес | 2 байта | 0x0000 до 0xFFFF |
| Количество регистров | 2 байта | 1 до 125 (0x7D) |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x03** |
| Кол-во байтов | 1 байт | 2 x **N\*** |
| Значения регистров | **N\*** x 2 байта |  |

**N\* =** Количество регистров

**Ошибка**

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

Ниже приведен пример запроса на чтение регистров 108-110:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Название поля* | *(Hex – запись)* | *Название поля* | *(Hex – запись)* |
| Функция | **03** | Функция | **03** |
| Начальный адрес старшие биты | **00** | Кол-во бит | **06** |
| Начальный адрес младшие биты | **6B** | Значение регистра Hi (108) | **02** |
| Кол-во регистров старшие биты | **00** | Значение регистра Lo (108) | **2B** |
| Кол-во регистров младшие биты | **03** | Значение регистра Hi (109) | **00** |
|  |  | Значение регистра Lo (109) | **00** |
|  |  | Значение регистра Hi (110) | **00** |
|  |  | Значение регистра Lo (110) | **64** |

Содержимое регистра 108 записывается как значение 2-х байтов 02 2B в hex – виде, или 555 – в десятичной форме записи. Содержимое регистров 109-110 00 00 и 00 64 в hex – виде, или 0 и 100 в десятичной форме соответственно.

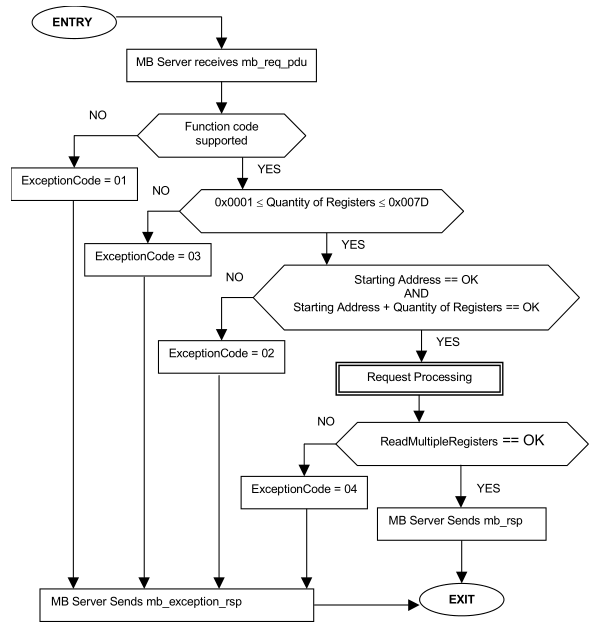


Рисунок 13. Диаграмма чтения состояний регистров хранения

* 1. **04 (0x04) Read Input Registers (Чтение входных регистров)**

Код функции используется, чтобы считывать от 1 до 125 значений близко расположенных входных регистров на удаленном устройстве. Во фрагменте данных протокола определяются начальный адрес регистра и количество регистров. Во фрагменте данных регистры к регистрам обращаются по адресу, начиная с 0. Так получается, что к входным регистрам, пронумерованным как 1-16, можно обратиться через адреса 0-15.

Данные регистров в ответном сообщении представлены двумя байтами данных на один регистр, содержат бинарное содержимое полностью заполняющее каждый из байтов. Для каждого регистра первый байт содержит наибольшие значащие биты, и второй содержит наименьшие значащие биты.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x04** |
| Начальный адрес | 2 байта | 0x0000 до 0xFFFF |
| Количество входных регистров | 2 байта | 0x0001 до 0x007D |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x04** |
| Кол-во байт | 1 байт | 2 x **N\*** |
| Входные регистры | **N\*** x 2 байта |  |

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код ошибки | 1 байт | **0x84** |
| Код исключения | 1 байт | 01 или 02 или 03 или 04 |

Ниже приведен пример запроса на чтение регистра 9:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Имя поля* | *(Hex – вид)* | *Имя поля* | *(Hex – вид)* |
| Функция | **04** | Функция | **04** |
| Начальный адрес Hi | **00** | Кол-во байт | **02** |
| Начальный адрес Lo | **08** | Входной регистр 9 (наибольший знач. бит) | **00** |
| Кол-во входных регистров Hi | **00** | Входной регистра 9 (наименьш. знач. бит) | **0A** |
| Кол-во входных регистров Lo | **01** |  |  |

Содержимое входного регистра 9 представлено 2 – мя байтами значений 00 0A в hex – записи, или 10 – в десятичной записи.

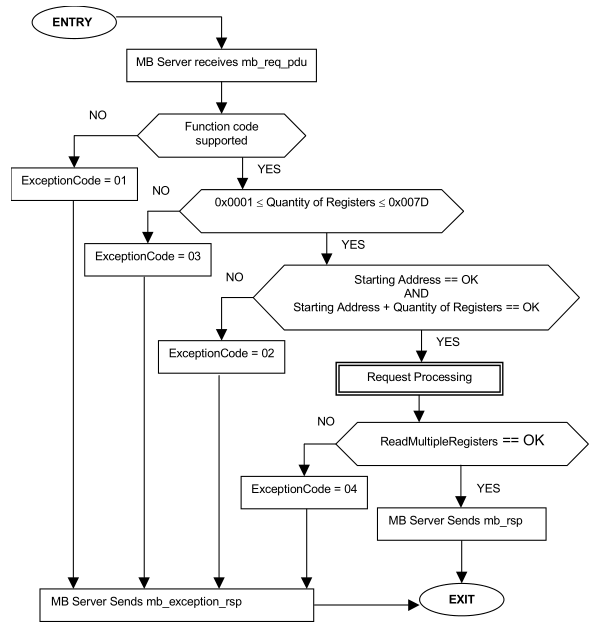


Рисунок 14. Диаграмма считывания значений входных регистров

* 1. **05 (0x05) Write Single Coil (Запись значения в одну ячейку)**

Этот код функции используется, чтобы записывать любое из 2 – х состояний ВКЛЮЧЕНО или ВЫКЛЮЧЕНО в удаленном устройстве.

Запрашиваемое состояние ВКЛЮЧЕНО/ВЫКЛЮЧЕНО определяется константой в поле данных запроса. Значение FF 00 в hex – записи запрашивает установку цифрового выхода в ВКЛЮЧЕНО. Значение 00 00 в hex – виде запрашивает установку значения в ВЫКЛЮЧЕНО. Все другие значения являются неправильными и не повлияют на состояние цифрового выхода.

Фрагмент данных в запросе задает определение для адреса ячейки, которая должна быть изменена. Ячейки нумеруются, начиная с 0. Таким образом, цифровой выход с номером 1 доступен по адресу 0. Запрашиваемое состояние ВКЛЮЧЕНО/ВЫКЛЮЧЕНО определяется константным значением в поле «Значение цифрового выхода». Значение 0XFF00 запрашивает изменение состояния ячейки во ВКЛЮЧЕНО. Значение 0X0000 запрашивает изменение состояния в ВЫКЛЮЧЕНО. Все другие значения являются не валидными и не окажут эффекта на ячейку.

Нормальным ответом на запрос является отклик, возвращаемый после того, как состояние цифрового выхода перезапишется.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x05** |
| Адрес цифрового выхода | 2 байта | 0x0000 до 0xFFFF |
| Устанавливаемое значение | 2 байта | 0x0000 до 0xFF00 |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x05** |
| Адрес цифрового выхода | 2 байта | 0x0000 до 0xFFFF |
| Значение цифрового выхода | 2 байта | 0x0000 до 0xFF00 |

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код ошибки | 1 байт | **0x85** |
| Код исключения | 1 байт | 01, 02, 03 или 04 |

Ниже приведен пример запроса на запись Цифрового выхода № 173 в состояние ВКЛЮЧЕНО:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Имя поля* | *(Hex – вид)* | *Имя поля* | *(Hex – вид)* |
| Функция | **05** | Функция | **05** |
| Адрес цифр. выхода Hi | **00** | Адрес цифр. выхода Hi | **00** |
| Адрес цифр. Выхода Lo | **AC** | Адрес цифр. выхода Lo | **AC** |
| Значение цифр. выхода Hi | **FF** | Значение цифр. выхода Hi | **FF** |
| Значение цифр. выхода Lo | **00** | Значение цифр. выхода Lo | **00** |

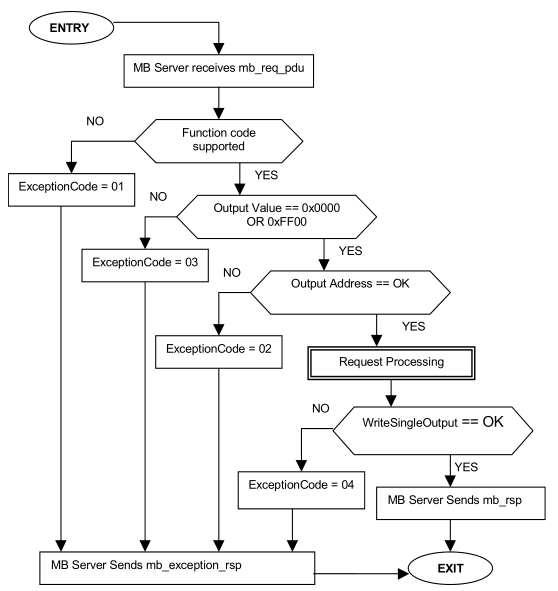


Рисунок 15. Блок – схема алгоритма изменения состояния одного цифрового выхода

* 1. **06 (0x06) Write Single Register (Запись в один регистр)**

Код функции используется для записи одного регистра хранения на удаленном устройстве. Фрагмент данных запроса определяет адрес регистра, который должен быть записан. Адресация осуществляется, начиная с нуля. Регистр с номером 1 доступен по адресу 0.

Нормальным ответом на запрос является ответ, возвращаемый после того, как содержимое регистра перезаписано.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x06** |
| Адрес регистра | 2 байта | 0x0000 до 0xFFFF |
| Значение регистра | 2 байта | 0x0000 до 0xFFFF |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x06** |
| Адрес регистра | 2 байта | 0x0000 до 0xFFFF |
| Значение регистра | 2 байта | 0x0000 до 0xFFFF |

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код ошибки | 1 байт | **0x86** |
| Код исключения | 1 байт | 01, 02, 03 или 04 |

Пример запроса на запись в регистр 2 значения 00 03 в hex – виде выглядит как:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Название поля* | *(hex – запись)* | *Название поля* | *(hex – запись)* |
| Функция | **06** | Функция | **06** |
| Адрес регистра Hi | **00** | Адрес регистра Hi | **00** |
| Адрес регистра Lo | **01** | Адрес регистра Lo | **01** |
| Значение регистра Hi | **00** | Значение регистра Hi | **00** |
| Значение регистра Lo | **03** | Значение регистра Lo | **03** |

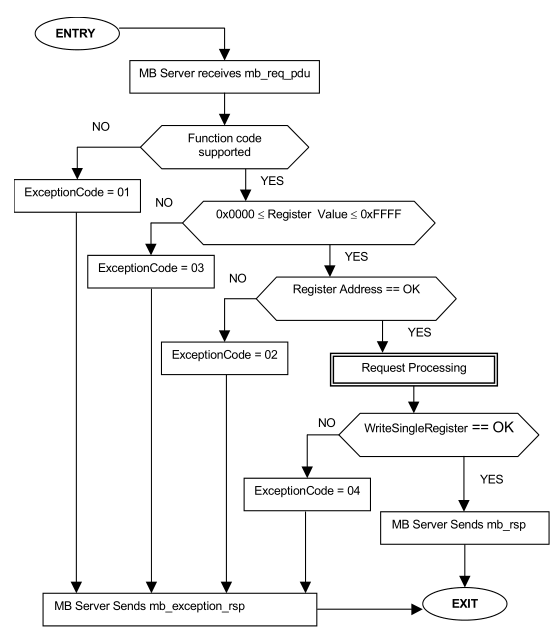


Рисунок 16. Блок – схема перезаписи значения одного регистра хранения

* 1. **07 (0x07) Read Exception Status (Serial Line Only) Считать состояние исключений (Только для подключения через последовательный порт)**

Этот функциональный код служит для чтения выходных значений 8 – ми Статусов Исключений на удаленном устройстве.

Функция предоставляет простой метод для доступа к этой информации, т.к. ссылки к Выходам Исключений известны (в указании в функции ссылки до выходного адреса нет необходимости).

Правильный ответ на запрос содержит состояние 8 –ми выходных значений исключений. Выходы группируются в один байт данных – по одному биту на каждый из выходов. Состояние самого младшего выхода содержится в наименьшем значащем бите байта.

Содержимое всех 8 – ми выходов Состояний Исключений специфичны и зависят от устройства.

**Запрос**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x07** |

**Ответ**

|  |  |  |
| --- | --- | --- |
| Код функции | 1 байт | **0x07** |
| Выходные данные | 1 байт | 0x00 до 0xFF |

**Ошибка**

|  |  |  |
| --- | --- | --- |
| Код ошибки | 1 байт | **0x87** |
| Код исключения | 1 байт | 01 или 04 |

Ниже приведен пример запроса на чтение состояния исключений:

|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | | **Ответ** | |
| *Имя поля* | *(Hex – вид)* | *Имя поля* | *(Hex – вид)* |
| Функция | **07** | Функция | **07** |
|  |  | Выходные данные | **6D** |

Это пример, где выходными данными является значение 6D (hex), или 0110 1101 в бинарном виде. Слева направо, выходами являются: ВЫКЛЮЧЕНО-ВКЛЮЧЕНО-ВКЛЮЧЕНО-ВЫКЛЮЧЕНО-ВКЛЮЧЕНО-ВЛКЮЧЕНО-ВЫКЛЮЧЕНО-ВКЛЮЧЕНО. Состояния выводятся от самого старшего до самого младшего адреса выхода.

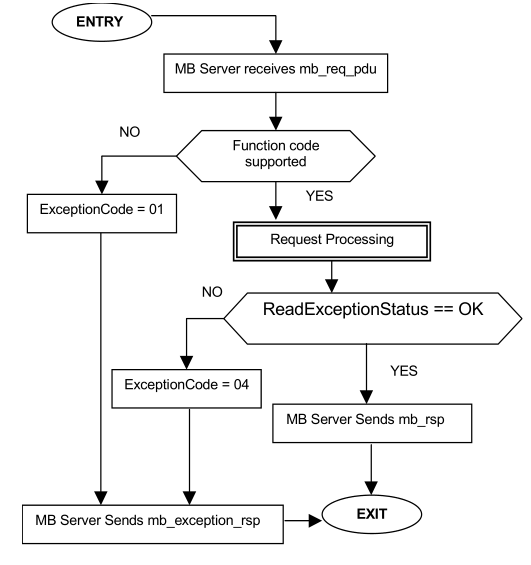


Рисунок 17. Блок – схема считывания состояния исключений