vit
зарождая технологии™

AVMOD
IO44D, IO44DU

Модули дискретного управления и ввода

ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ:

- 4 группы контактов(реле) и 4 дискретных входа.
- Максимальные коммутируемые токи и напряжения 5А, 240В.
- Гальванически развязанные интерфейсы RS485 и USB 2.0.
- Возможность использования в качестве преобразователя интерфейсов USB-RS485.
- Возможность подключения нескольких устройств к одной шине.
- Управление по стандартному промышленному протоколу ModBus.
- Средства асинхронной регистрации событий входов.
- Возможность переключения реле на заданный временной интервал одной командой.
- Доступ через виртуальный COM-порт при подключении к ПК по USB.
- Гальванически развязанные входы с внутренними подтягивающими резисторами либо независимые входы по напряжению.
- Надежный и удобный корпус с возможностью установки на DIN-рейку.
- Настройка и тестирование модуля с помощью ПО поставляемого в комплекте.

Оглавление

АППАРАТНОЕ ОБЕСПЕЧЕНИЕ 3
Технические характеристики 3
Подключение внешних цепей 4
Интерфейс RS485 5
Контакты реле 6
Дискретные входы 6
Интерфейс USB (IO44DU) 7
УТИЛИТА КОНФИГУРАЦИИ AVMOD 8
Выбор порта 8
Поиск по адресам 8
Поиск по скоростям 8
Поиск по скоростям и адресам 8
Конфигурирование устройств 9
Режим управления модулями IO44D и IO44DU 9
Режим отображения терминала 9
ПРОГРАМНАЯ МОДЕЛЬ IO44D И IO44DU 10
Описание протокола связи 10
Функциональные коды 10
Коды ошибок 14
Описание регистров 14
Битовые переменные и входы 16
Пример подпрограммы расчета контрольной суммы на языке C 17
Пример подпрограммы расчета контрольной суммы на языке Pascal 18

АППАРАТНОЕ ОБЕСПЕЧЕНИЕ

Технические характеристики

ПАРАМЕТРЫ КОНТАКТНЫХ ПАР

Количество контактных пар	4
Максимальное коммутируемое напряжение	240В
Максимальный коммутируемый ток	5А
Гальваническая развязка от остальных узлов схемы	>1000В
Механическая надежность контактов	10 000 000 срабатываний
Электрическая прочность контактов	10 000 срабатываний
Варианты исполнения контактных пар	н. замк./н. раз.

ПАРАМЕТРЫ ДИСКРЕТНЫХ ВХОДОВ

Количество входов	4
Варианты исполнения входов	
Коммутация на общую землю:	
Внутренние подтягивающие резисторы	2.2кОм
Напряжение внутреннего источника	+4В
Гальванически независимые входы управляемые напряжением	
Напряжение	+5/+12/+24В
Защита от перенапряжений	<2Uном.
Гальваническая развязка от остальных узлов схемы	>1000В

ПАРАМЕТРЫ ИНТЕРФЕЙСА RS485

Поддерживаемые скорости, Бод	4800, 9600, 14400, 57600, 19200, 38400, 115200
Режимы проверки четности	EVEN, ODD, NO
Гальваническая развязка от остальных узлов схемы	1000В
Защита от электростатических разрядов	20кВ

ПАРАМЕТРЫ ИНТЕРФЕЙСА USB*

Спецификация	2.0
Защита от электростатических разрядов	20кВ
Гальваническая развязка от остальных узлов схемы	>1000В

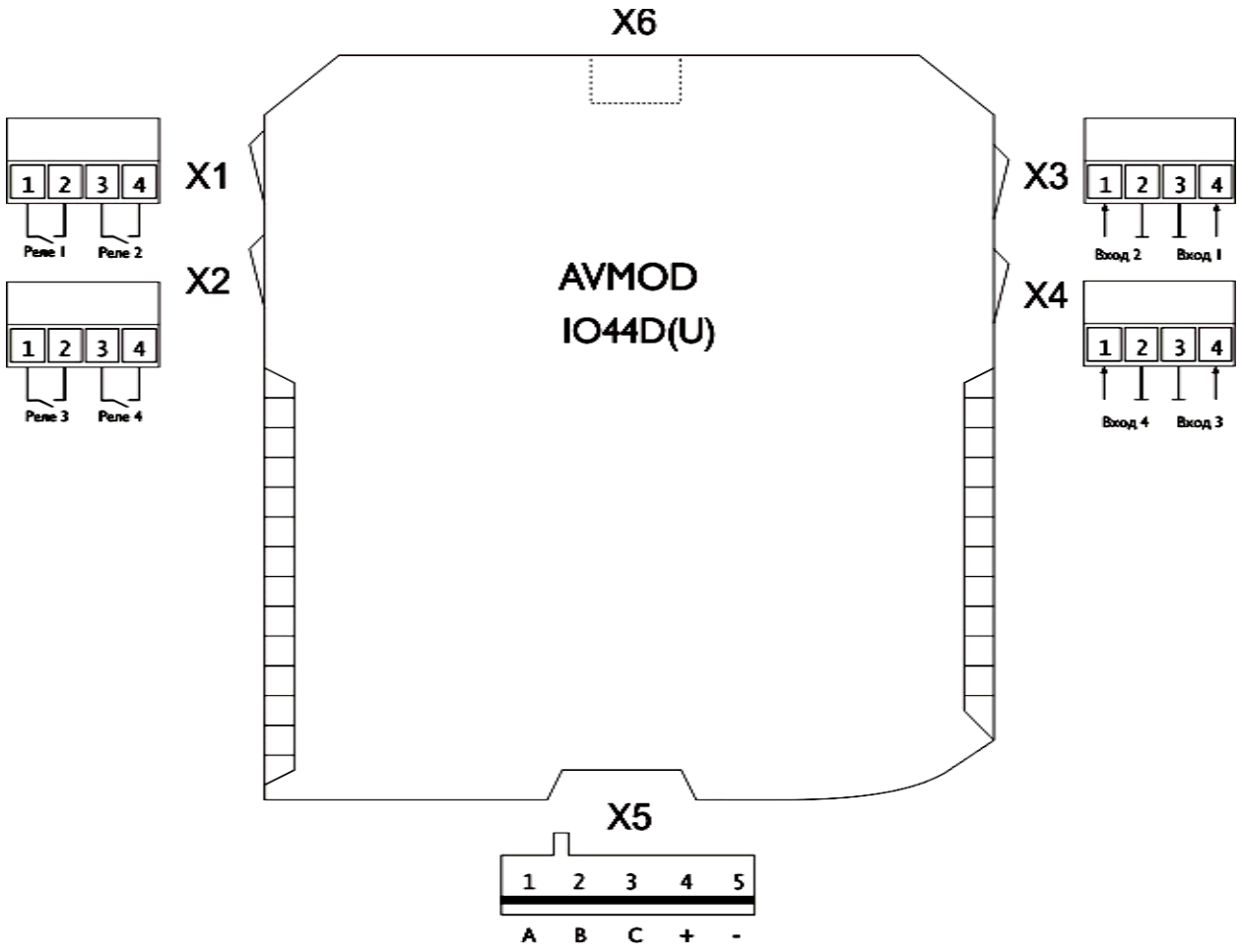
ПАРАМЕТРЫ ПИТАНИЯ

Напряжение питания	10…30В
Максимальный потребляемый ток	150мА
Защита от переполюсовки питания	есть

*Только для модуля IO44DU

Подключение внешних цепей

Схема подключения внешних разъемов модуля IO44D/ IO44DU имеет следующий вид:



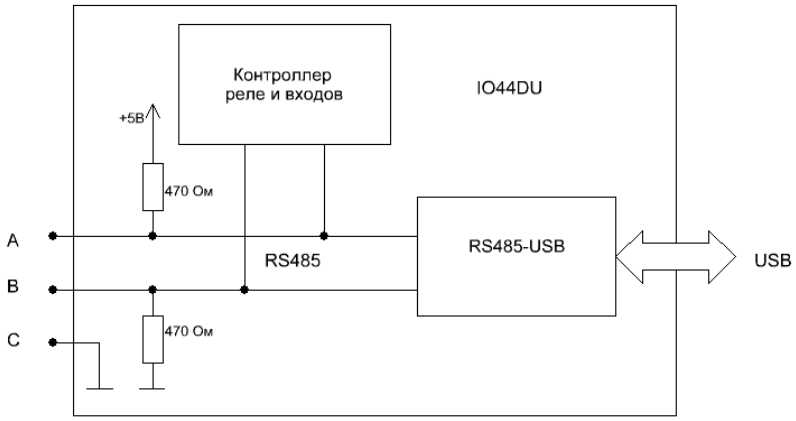
Разъем	Описание	Контакты				
		1	2	3	4	5
X1	Реле 1,2	Реле 1	Реле 1	Реле 2	Реле 2	-
X2	Реле 3,4	Реле 3	Реле 3	Реле 4	Реле 4	-
X3	Входы 1,2	Вход 2	Вход 2 общ.	Вход 1 общ.	Вход 1	-
X4	Входы 3,4	Вход 4	Вход 4 общ.	Вход 3 общ.	Вход 3	-
X5	RS485, питание	RS485 A	RS485 B	RS485 C (общ.)	+Uпит	-Uпит
X6*	USB гнездо тип B	-				

*Только для модуля IO44DU

Интерфейс RS485

IO44DU

Внутренняя логическая организация модуля IO44DU имеет следующий вид:



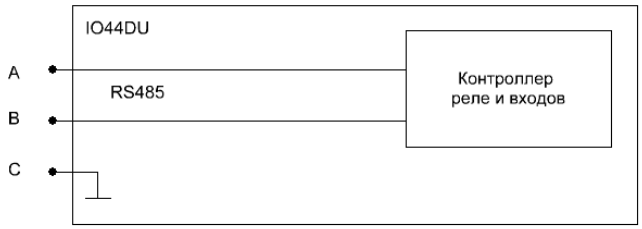
Модуль имеет внутреннюю шину RS485, к которой подключен контроллер управления реле и дискретных входов. Доступ к шине RS485 можно получить двумя способами: непосредственно подключившись к её линиям, выведенным на внешний разъем устройства, либо через встроенный преобразователь интерфейсов RS485-USB (см. раздел «Интерфейс USB»).

При подключении устройства через переходник RS485-RS232 номер порта определяется физическим номером COM порта, к которому подключен переходник.

ВНИМАНИЕ! Модуль IO44DU предполагается использовать в качестве ведущего (master) устройства на шине RS485, поэтому он имеет внутренние подтягивающие резисторы на линиях A и B. Другие устройства, подключенные к шине, должны быть без подтягивающих резисторов.

IO44D

Внутренняя логическая организация модуля IO44D имеет следующий вид:



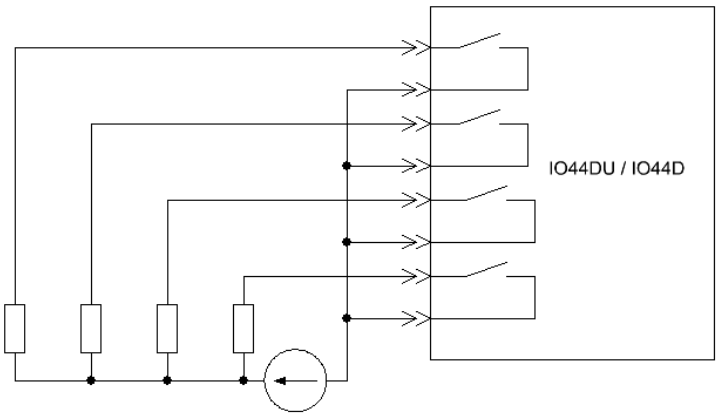
Управления модулем может осуществляться только с помощью внешнего преобразователя интерфейсов или master-устройства, подключенного к шине RS485.

ВНИМАНИЕ! Модуль IO44D может быть использован только в качестве ведомого (slave) устройства на шине RS485, поэтому он не имеет внутренних подтягивающих резисторов на линиях A и B. Ведущее устройство на шине должно иметь подтягивающие резисторы соответствующих номиналов.

К шине RS485 может быть подключено несколько модулей IO44D, при условии наличия уникального адреса у каждого из них.

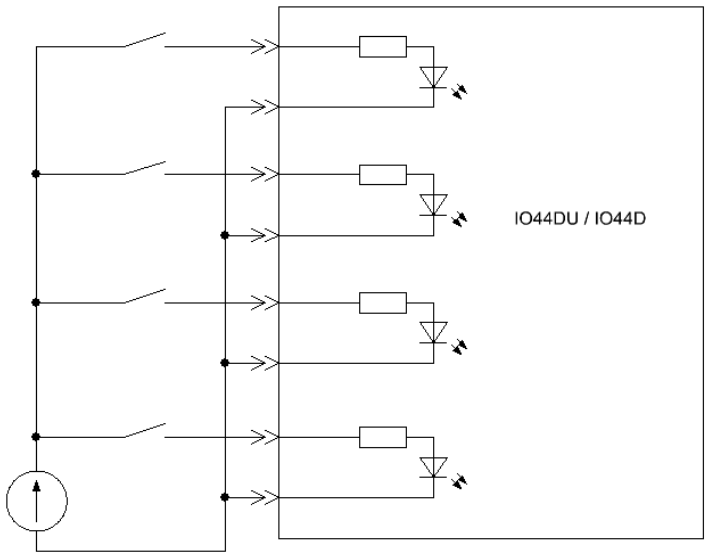
Контакты реле

Модули AVMOD IO44DU и IO44D содержат четыре гальванически независимые друг от друга группы контактов реле, выведенные на внешние клемники. Типовая схема управления внешними нагрузками имеет вид:

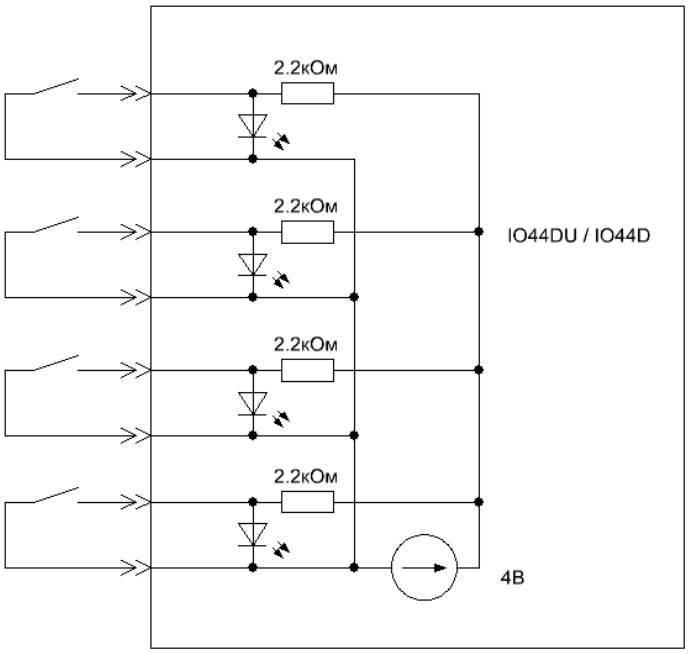


Дискретные входы

Модули IO44DU/IO44D могут иметь два различных типа дискретных входов. Первый тип - **гальванически развязанные входы управляемые напряжением**. Для подачи логической единицы на вход, необходимо подключить его к внешнему источнику напряжения. Номиналы внутренних ограничивающих резисторов зависят от входного напряжения, на которое был рассчитан данный модуль (+5, +12 или +24В). В данном исполнении все входы гальванически не связанны друг от другом и с остальными узлами схемы.



Второй вариант входов – **входы с коммутацией на общую землю**. Для подачи логической единицы необходимо замкнуть клеммы соответствующего входа. В данном варианте исполнения, входы гальванически отвязаны от остальных узлов схемы, но имеют общую землю. Данный вариант исполнения входов используется при соединении внешних ключей по схеме «сухой контакт»



Интерфейс USB (IO44DU)

Блок USB фактически является встроенным преобразователем интерфейсов RS485-USB, дающим возможность получить доступ к внутренней шине RS485 устройства. При подключении к порту USB, устройство определяется системой как виртуальный COM порт. Номер, присвоенный порту можно определить следующим образом: откройте **Пуск** -> **Панель управления** -> **Система** -> **Оборудование** -> **Диспетчер устройств** -> **Порты COM и LPT**. В списке портов найдите порт с названием **Silicon Labs CP2103 USB to UART Bridge**. Номер этого порта и будет искомым.

Питание

Модуль может работать в диапазоне питающих напряжений от +10 до +30В. Рекомендуемый рабочий диапазон 12 - 24В. Максимальный ток, потребляемый модулем, составляет 150мА при четырех включенных реле и напряжении питания 10В. Предусмотрена защита от переполюсовки питания.

УТИЛИТА КОНФИГУРАЦИИ AVMOD

Утилита предназначена для конфигурации и проверки работоспособности различных устройств серии AVMOD. С её помощью можно осуществлять поиск устройств, подключенных к последовательным портам ПК (в том числе виртуальным).

Выбор порта

Для начала работы с утилитой необходимо выбрать последовательный порт, к которому подключено устройство (см. раделы «Интерфейс USB» и «Интерфейс RS485»). Порт выбирается из выпадающего списка «Порт», в котором перечислены доступные последовательные порты.

Поиск по адресам

Данный тип поиска следует использовать в случае, если к шине подключен один или несколько модулей, имеющих одинаковую и известную скорость обмена и тип проверки четности, но различные адреса. Для поиска следует установить переключатели «по скоростям» и «по адресам» в выключенное и включенное состояния соответственно. Также необходимо задать начальный (текущий) и конечный адреса, скорость и четность, на которых будет осуществляться поиск. При поиске по адресам скорость и четность остаются неизменными. Поиск может осуществляться до первого найденного устройства, или по всем адресам диапазона.

Поиск по скоростям

Данный тип поиска следует использовать в случае, если адрес устройства известен, но неизвестны его скорость обмена и тип проверки четности. Для поиска следует установить переключатели «по скоростям» и «по адресам» во включенное и выключенное состояния соответственно. Также необходимо задать текущий адрес, на котором будет осуществляться поиск. При поиске по скоростям, текущий адрес остается неизменными. Поиск может осуществляться до первого найденного устройства, или по всем возможным комбинациям скоростей и четностей.

ВНИМАНИЕ! В случае подключения к одной шине устройств, имеющих одинаковые адреса, скорости и четности, или одинаковые скорости и адреса, но различные четности, может возникнуть конфликт, приводящий к неверному определению устройств.

Поиск по скоростям и адресам

Следует использовать в случае, если адреса, скорости и четности устройств, подключенных к шине, заранее неизвестны. Данный тип поиска является наиболее медленным, поскольку предполагает последовательную проверку всех возможных комбинаций скоростей и четностей для всех адресов. Для поиска следует установить переключатели «по скоростям» и «по адресам» во включенное состояние. Также необходимо задать начальный (текущий) и конечный адреса диапазона, в котором будет осуществляться поиск. Поиск может осуществляться до первого найденного устройства или по всем адресам диапазона для каждой возможной комбинации скорости и типа проверки четности.

Конфигурирование устройств

Конфигурирование найденных устройств производится непосредственно в окне списка устройств. Для изменения скорости обмена устройства или типа проверки четности необходимо кликнуть на соответствующей ячейке и из выпадающего списка выбрать требуемые значения. Для изменения адреса устройства необходимо в ячейку адреса ввести требуемое значение.

Режим управления модулями IO44D и IO44DU

Для перехода к режиму управления следует на вкладке «Поиск» выбрать устройство из списка устройств, при этом появится вкладка «Управление…». На ней будут размещены элементы управления, соответствующие выбранному типу устройства. Для данных модулей доступны следующие элементы управления:

- Переключатели состояния реле.
- Переключатели реле на заданный временной интервал.
- Флажки состояний входов и кнопка их обновления(« <- »);
- Флажки регистрации переходов из 1 в 0 и кнопки их обновления и сброса (« <- » и «0»);
- Флажки регистрации переходов из 0 в 1 и кнопки их обновления и сброса (« <- » и «0»);
- Флажки регистрации любых переходов и кнопки их обновления и сброса (« <- » и «0»);

Режим отображения терминала

Терминал позволяет контролировать процесс обмена пакетами между программным обеспечением и модулями. Для включения режима отображения терминала следует в пункте меню «**Вид**» выбрать «**Показать терминал**». Для скрытия терминала следует выбрать «**Скрыть терминал**».

ПРОГРАМНАЯ МОДЕЛЬ IO44D И IO44DU

Описание протокола связи

Обмен данными с устройством происходит посредством пакетов. Каждый пакет состоит из одного байта адреса, байта функционального кода, последовательности байтов данных и двух байтов контрольной суммы. Функциональный код и байты данных формируют элементарный пакет протокола (PDU).

Адрес на шине	Функциональный код	Данные	CRC16

Адрес устройства определяет, какому устройству на шине предназначен данный пакет или от какого устройства он получен (в случае ответной посылки). Адрес может находится в диапазоне 1…255 (рекомендованные в спецификации протокола ModBus адреса – 1...247). Адрес 0 – широковещательный: при обращении по нему все устройства подключенные к шине выполняют соответствующую инструкцию, не отправляя при этом ответной посылки. Посылки, адрес которых не соответствует адресу устройства или не равен нулю, данным устройством игнорируются. Функциональный код определяет предназначение данных, которые за ним следуют. Контрольная сумма вычисляется на основе стандартного полинома ModBus При несовпадении контрольной суммы устройство игнорирует посылку.

Доступ к ресурсам модуля AVMOD IO44D(U) в соответствии с протоколом ModBus осуществляется битовым или регистровым способом, то есть обращение к различным ресурсам, таким как реле или входы, в большинстве случаев может происходить как посредством чтения-записи двоичных (битовых) переменных, так и через разряды соответствующих регистров.

Функциональные коды

Устройство AVMOD IO44DU поддерживает следующие функциональные коды:

Функциональный код	Описание
0x01	чтение битовых переменных
0x02	чтение дискретных входов
0x03	чтение регистров
0x05	запись одной битовой переменной
0x06	запись одного регистра
0x0F	запись битовых переменных
0x10	запись регистров

0x01 (1) – Позволяет считать значения нескольких битовых переменных. В запросе указывается адрес первой переменной, с которой начинается чтение и количество переменных, которые необходимо считать. В ответной посылке значения переменных представляются побитно, т.е. первой переменной соответствует первый (младший) бит младшего байта, второй – второй бит и т.д. Количество байт данных n в ответной посылке равно количеству переменных деленному на 8 и округленному до ближайшего большего целого числа.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x01
Стартовый адрес	2	0x0000 - 0x000F
Количество переменных	2	0x0001 - 0x000F

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x01
Количество байт	1	0x01 - 0x02
Значения переменных	n	-

Пример: считать состояния четырех дискретных переменных, начиная с адреса 0x0000 (переменные с адресами 0 и 2 - лог. 1, 1 и 3 – лог. 0)

Запрос: 01 **01 00 00 00 04** 3D CD Ответ: 01 **01 01 05** 91 8B

0x02 (2) – Позволяет считать значения нескольких дискретных входов. В запросе указывается адрес входа, с которого начинается чтение, и количество входов, которые необходимо считать. В ответной посылке значения входов представляются побитно, т.е. первому входу соответствует первый (младший) бит младшего байта, второму – второй бит и т.д. Количество байт данных n в ответной посылке равно количеству входов деленному на 8 и округленному до ближайшего большего целого числа.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x02
Стартовый адрес	2	0x0000 - 0x000F
Количество переменных	2	0x0001 - 0x0010

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x01
Количество байт	1	0x01 - 0x02
Значения переменных	n	-

Пример: считать состояния четырех дискретных входов начиная с адреса 0x0004 (вход 4 – лог.1)

Запрос: 01 **02 00 04 00 04** 38 08 Ответ: 01 **02 01 08** A0 4E

0x03 (3) – Позволяет считать значения нескольких регистров. В запросе указывается адрес первого регистра, с которого начинается чтение, и количество регистров, которые необходимо считать. В ответной посылке каждому регистру соответствует 2 байта данных.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x03
Стартовый адрес	2	0x0000 - 0x000C
Количество переменных	2	0x0001 - 0x000D

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x03
Количество байт	2	0x0001 - 0x000D
Значения переменных	n	-

Пример: Считать содержимое двух регистров начиная с адреса 0x0000 (содержимое 0x0001 и 0x0222)

Запрос: 01 **03 00 00 00 02** C4 0B Ответ: 01 **03 04 02 22 00 01** 9A 41

0x05 (5) – Позволяет изменить состояние одной битовой переменной по заданному адресу. Переменная устанавливается в лог. 1, если переданное значение равно 0xFF00, или в лог. 0, если 0x0000. Все остальные значения приводят к генерации кода ошибки.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x05
Адрес	2	0x0000 - 0x000F
Значение	2	0x0000 или 0xFF00

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x05
Адрес	2	0x0000 - 0x000F
Значение	2	0x0000 или 0xFF00

Пример: Записать в переменную с адресом 0x0000 лог. 1.

Запрос: 01 **05 00 00 FF 00** 8C 3A Ответ: 01 **05 00 00 FF 00** 8C 3A

0x06 (6) – Позволяет изменить значение одного регистра по заданному адресу.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x06
Адрес	2	0x0002 - 0x000C
Значение	2	0x0000 - 0xFFFF

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x06
Адрес	2	0x0002 - 0x000
Значение	2	0x0000 - 0xFFFF

Пример: Записать в регистр с адресом 0x0009 значение 0x0010

Запрос: 01 **06 00 09 00 10** 58 04 Ответ: 01 **06 00 09 00 10** 58 04

0x0F (15) - Позволяет изменить состояние нескольких битовых переменных. В посылке указывается адрес первой переменной и количество переменных, которые необходимо записать. Значения переменных задаются соответствующими разрядами байтов данных, т.е. первой переменной соответствует первый (младший) бит младшего байта, второй – второй бит и т.д. Количество байтов данных n должно быть равно количеству переменных деленному на 8 и округленному до ближайшего большего целого числа.

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x0F
Стартовый адрес	2	0x0000 - 0x000F
Количество переменных	2	0x0001 - 0x0010
Количество байт	1	n
Значение	n	-

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x0F
Стартовый адрес	2	0x0000 - 0x000F
Количество переменных	2	0x0001 - 0x0010

Пример: записать четыре битовых переменных начиная с адреса 0x0000 (0 и 2 – лог. 1)

Запрос: 01 **0F 00 00 00 04 01 05** FE 95 Ответ: 01 **0F 00 00 00 04** 54 08

0x10 (16) - Позволяет изменить значения нескольких регистров. В посылке указывается адрес первого регистра и количество регистров, которые необходимо записать. Количество байт данных n должно быть равно количеству регистров умноженному на 2

Запрос:

Описание	Байт	Диапазон значений
Функциональный код	1	0x10
Стартовый адрес	2	0x0002 - 0x000C
Количество переменных	2	0x0001 - 0x000A
Количество байт	1	n
Значение	n	-

Ответ:

Описание	Байт	Диапазон значений
Функциональный код	1	0x10
Стартовый адрес	2	0x0002 - 0x000C
Количество переменных	2	0x0001 - 0x000A

Пример: Записать в четыре регистра, начиная с адреса 0x0009 значение 0x0010

Запрос: 01 **10 00 09 00 04 08 00 10 00 10 00 10 00 10** 7A 6D Ответ: 01 **10 00 09 00 04** 11 C8

Коды ошибок

В случае возникновения ошибки при попытке получить доступ к ресурсам устройства, ответный пакет будет содержать функциональный код ошибки и код исключения. Код ошибки формируется путем сложения функционального кода запроса с константой 0x80.

Ошибка:

Описание	Байт	Диапазон значений
Функциональный код	1	Функциональный код запроса + 0x80
Исключение	1	0x01 или 0x02 или 0x03 или 0x04

Коды исключения указывают на тип ошибки:

0x01 – неверный функциональный код
0x02 – неверный адрес данных
0x03 – неверное значение данных
0x04 – ошибка при попытке выполнить функцию

Описание регистров

Доступ к регистрам осуществляется при обращении с функциональными кодами 0x03, 0x06 и 0x10

Адрес регистра	Описание
0x00	Старшие два байта серийного номера
0x01	Младшие два байта серийного номера
0x02	Адрес устройства
0x03	Скорость|четность
0x04	Реле (младшая тетрада)
0x05	Входы (младшая тетрада)
0x06	Переходы из высокого в низкий (младшая тетрада)
0x07	Переходы из низкого в высокий (младшая тетрада)
0x08	Любые переходы (младшая тетрада)
0x09	Переключение реле 1 на заданное количество десятых секунды
0x0A	Переключение реле 2 на заданное количество десятых секунды
0x0B	Переключение реле 3 на заданное количество десятых секунды
0x0C	Переключение реле 4 на заданное количество десятых секунды

0x00, 0x01 (только чтение) – регистры, содержащие тип устройства и серийный номер.
0x02 – регистр адреса устройства на шине ModBus.
0x03 – регистр установки скорости обмена и проверки четности на линии RS485.
Младший байт регистра определяет скорость обмена:

Младший байт регистра 0x03	Скорость, бод
0	4800
1	9600
2	14400
3	**19200***
4	38400
5	57600
6	115200

Старший байт определяет тип или наличие бита четности в посылке:

Старший байт регистра 0x03	Четность
0	**EVEN***
1	ODD
2	NO

*Заводские значения скорости и четности (0x0003).

0x04 – регистр управления реле. Биты младшей тетрады данного регистра определяют состояние реле: в случае, если контакты нормально разомкнуты, при записи единицы в соответствующий бит контакты замыкаются, в противном случае – размыкаются.

0x05 – регистр дискретных входов. Биты младшей тетрады данного регистра соответствуют дискретным входам: в случае наличия подтягивающих резисторов при замыкании входа на землю в соответствующем разряде регистра будет установлена единица. Если же входы не имеют подтягивающих резисторов, единица будет установлена при подаче на вход напряжения, превышающего пороговое.

0x06 – регистр-защелка, соответствующие биты которого (младшая тетрада) устанавливаются в 1, если на одном или нескольких входах произошел переход из высокого уровня в низкий. Биты можно сбросить в изначальное состояние только программной записью 0.

0x07 – регистр-защелка, соответствующие биты которого (младшая тетрада) устанавливаются в 1, если на одном или нескольких входах произошел переход из низкого уровня в высокий. Биты можно сбросить в изначальное состояние только программной записью 0.

0x08 – регистр-защелка, соответствующие биты которого (младшая тетрада) устанавливаются в 1, если на одном или нескольких входах произошло любое изменение состояния. Биты можно сбросить в изначальное состояние только программной записью 0.

0x09, 0x0A, 0x0B, 0x0C – 2-х байтные регистры, отвечающие за переключение соответствующих реле (0x09 соответствует реле 1 и т.д.) на заданный временной интервал. При записи в регистр числа N, отличного от 0 происходит переключение реле в состояние, противоположное текущему на временной интервал, определяемый записанным числом. Число имеет размерность десятых секунды, т.е. временную задержку T в секундах можно определить по формуле T=N*10. После истечения заданного временного интервала, реле переключается в исходное состояние.

Битовые переменные и входы

Доступ к битовым переменным осуществляется при обращении по функциональным кодам 0x01, 0x05, 0x0F; к дискретным входам – 0x02

Адрес	Описание	
	Битовые переменные	Входы
0x00	Реле 1	Вход 1
0x01	Реле 2	Вход 2
0x02	Реле 3	Вход 3
0x03	Реле 4	Вход 4
0x04	Переход В-Н вход 1	Переход В-Н вход 1
0x05	Переход В-Н вход 2	Переход В-Н вход 2
0x06	Переход В-Н вход 3	Переход В-Н вход 3
0x07	Переход В-Н вход 4	Переход В-Н вход 4
0x08	Переход Н-В вход 1	Переход Н-В вход 1
0x09	Переход Н-В вход 2	Переход Н-В вход 2
0x0A	Переход Н-В вход 3	Переход Н-В вход 3
0x0B	Переход Н-В вход 4	Переход Н-В вход 4
0x0C	Любой переход вход 1	Любой переход вход 1
0x0D	Любой переход вход 2	Любой переход вход 2
0x0E	Любой переход вход 3	Любой переход вход 3
0x0F	Любой переход вход 4	Любой переход вход 4

0x00, 0x01, 0x02, 0x03 – дискретные переменные реле или входов. При обращении с функциональными кодами 0x01, 0x05, 0x0F соответствуют битам первой тетрады регистра 0x04 (реле). При обращении с кодом 0x02 позволяют считывать состояния соответствующих входов.
0x04, 0x05, 0x06, 0x07 – соответствуют первым четырем битам регистра-защелки переходов из высокого уровня в низкий (0x06). Доступны по функциональным кодам 0x01, 0x05, 0x0F.
0x08, 0x09, 0x0A, 0x0B – соответствуют первым четырем битам регистра-защелки переходов из низкого уровня в высокий (0x07). Доступны по функциональным кодам 0x01, 0x05, 0x0F.
0x0C, 0x0D, 0x0E, 0x0F – соответствуют первым четырем битам регистра-защелки переходов из высокого уровня в низкий (0x08). Доступны по функциональным кодам 0x01, 0x05, 0x0F.

Пример подпрограммы расчета контрольной суммы на языке С

```
unsigned char auchCRCHi[] = {
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
} ;

unsigned char auchCRCLo[] = {
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
};

unsigned short CRC16 ( unsigned char *puchMsg, unsigned short usDataLen)
{
      unsigned char uchCRCHi = 0xFF ;
      unsigned char uchCRCLo = 0xFF ;
      unsigned uIndex ;
      while (usDataLen--) {
            uIndex = uchCRCLo ^ *puchMsg++ ;
            uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
            uchCRCHi = auchCRCLo[uIndex] ;
      }
      return (uchCRCHi << 8 | uchCRCLo) ;
}
```

Пример подпрограммы расчета контрольной суммы на языке Pascal

```
const

CRChi : array[0..255] of byte =
  (
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $01,$C0,$80,$41,$00,$C1,$81,$40,$00,$C1,$81,$40,$01,$C0,$80,$41,
  $00,$C1,$81,$40,$01,$C0,$80,$41,$01,$C0,$80,$41,$00,$C1,$81,$40 );

CRClo : array[0..255] of byte =
  (
  $00,$C0,$C1,$01,$C3,$03,$02,$C2,$C6,$06,$07,$C7,$05,$C5,$C4,$04,
  $CC,$0C,$0D,$CD,$0F,$CF,$CE,$0E,$0A,$CA,$CB,$0B,$C9,$09,$08,$C8,
  $D8,$18,$19,$D9,$1B,$DB,$DA,$1A,$1E,$DE,$DF,$1F,$DD,$1D,$1C,$DC,
  $14,$D4,$D5,$15,$D7,$17,$16,$D6,$D2,$12,$13,$D3,$11,$D1,$D0,$10,
  $F0,$30,$31,$F1,$33,$F3,$F2,$32,$36,$F6,$F7,$37,$F5,$35,$34,$F4,
  $3C,$FC,$FD,$3D,$FF,$3F,$3E,$FE,$FA,$3A,$3B,$FB,$39,$F9,$F8,$38,
  $28,$E8,$E9,$29,$EB,$2B,$2A,$EA,$EE,$2E,$2F,$EF,$2D,$ED,$EC,$2C,
  $E4,$24,$25,$E5,$27,$E7,$E6,$26,$22,$E2,$E3,$23,$E1,$21,$20,$E0,
  $A0,$60,$61,$A1,$63,$A3,$A2,$62,$66,$A6,$A7,$67,$A5,$65,$64,$A4,
  $6C,$AC,$AD,$6D,$AF,$6F,$6E,$AE,$AA,$6A,$6B,$AB,$69,$A9,$A8,$68,
  $78,$B8,$B9,$79,$BB,$7B,$7A,$BA,$BE,$7E,$7F,$BF,$7D,$BD,$BC,$7C,
  $B4,$74,$75,$B5,$77,$B7,$B6,$76,$72,$B2,$B3,$73,$B1,$71,$70,$B0,
  $50,$90,$91,$51,$93,$53,$52,$92,$96,$56,$57,$97,$55,$95,$94,$54,
  $9C,$5C,$5D,$9D,$5F,$9F,$9E,$5E,$5A,$9A,$9B,$5B,$99,$59,$58,$98,
  $88,$48,$49,$89,$4B,$8B,$8A,$4A,$4E,$8E,$8F,$4F,$8D,$4D,$4C,$8C,
  $44,$84,$85,$45,$87,$47,$46,$86,$82,$42,$43,$83,$41,$81,$80,$40 );


procedure CRC16(Data: array of byte;  len : word; var Hi : byte; var Lo : byte);
var Index : word;
    i : word;
begin
   Hi:=$FF;
   Lo:=$FF;
   i:=0;
   While (i<Len) do
   begin
     Index:=Hi xor Data[i];
     Hi:=Lo xor CRCHi[Index];
       Lo:=CRCLo[Index];
       i:=i+1;
   end;
end;
```