ВВЕДЕНИЕ В MODBUS ПРОТОКОЛ.

Передача в сети MODBUS.

Стандартные MODBUS-порты в контроллерах MODICON используют RS-232C совместиимый последовательный интерфейс . Контроллеры могут быть соединены на прямую или через модем.

Контроллеры соединяются используя технологию главный-подчиненный, при которой только одно устройство (главный) может инициировать передачу (сделать запрос). Другие устройства (подчиненные) передают запрашиваемые главным устройством данные, или производят запрашиваемые действия. Типичное главное устройство включает в себя ведущий (HOST) процессор и панели программирования. Типичное подчинненое устройство - программируемый контроллер.

Главный может адресоваться к индивидуальному подчиненному или может инициировать широкую передачу сообщения на все подчиненные устройства. Подчиненное устройство возвращает сообщение в ответ на запрос, адресуемый именно ему. Ответы не возвращаются при широковещательном запросе от главного.

Цикл запрос - ответ.

Запрос от главного	Ответ подчиненного
Адрес устройства	Адрес устройства
Код функции	Код функции
8 - битные	8 - битные
байты данных	байты данных
Контрольная сумма	Контрольная сумма

Запрос : Код функции в запросе говорит подчиненному устройству какое действие необходимо провести. Байты данных содержат информацию необходимую для выполнения запрошенной функции. Например, код функции 3 подразумевает запрос на чтение содержимого регистров подчиненного.

Ответ : Если подчиненный дает нормальный ответ, код функции в ответе повторяет код функции в запросе. В байтах данных содержится затребованная информация. Если имеет место ошибка, то код функции модифицируется, и в байтах данных передается причина ошибки.

Два режима последовательной передачи.

В сетях MODBUS может быть использован один из двух способов передачи: ASCII или RTU. Пользователь выбирает необходимый режим вместе с другими параметрами (скорость передачи, режим паритета и т.д.) во время конфигурации каждого контроллера.

Режим ASCII.

При использовании ASCII - режима каждый байт сообщения передается как два ASCII символа. Главное преимущество этого способа время между предачей символов может быть до 1 сек. без возникновения ошибок при передаче.

Формат каждого байта в ASCII-режимме:

Система кодировки: Шестнадцатиричная, ASCII-символы 0-9, A-F

Назначение битов:	1 старт бит
	7 бит данных, младшим битом вперед
	1 бит паритета; нет бита паритета
	1 стоп бит если есть паритет; 2 бита если
	нет паритета
	Контрольная сумма: Longitudinal Redundancy Chek (LRC)
	Режим RTU.

При использовании ASCII - режима каждый байт сообщения содержит два 4-х битных шестнадцатиричных числа.

Каждое сообщение передается непрерывным потоком.

Формат каждого байта в RTU-режиме:

Система кодировки: 8-ми битовая двоичная, шестнадцатиричная	
	0-9, A-F
	Две шестнадцатиричные цифры содержатся в
	каждом 8-ми битовом байте сообщения.
Назначение битов:	1 старт бит
	8 бит данных, младшим значащим разрядом
	вперед

1 бит паритета; нет бита паритета

1 стоп бит если есть паритет; 2 бита если нет паритета

Контрольная сумма: Cyclical Redundancy Check (CRC)

Содержание сообщения MODBUS.

ASCII фрейм.

В ASCII-режиме, сообщение начинается с "двоеточия" (:, ASCII 3A hex), и заканчивается последовательностью "возврат каретки-перевод строки" (CRLF, ASCII 0D и 0A hex).

Допустимые символы для передачи - это шестнадцатиричные цифры 0-9, A-F. Монитор сетевого устройства в сети непрерывно отслеживает символ "двоеточие". Когда он принят, каждое устройство декодирует следующие поле сообщения (поле адреса) и т.д.

Интервалы между символами сообщения могут быть до 1 сек. Если интервал больше, то принимающее устройство распознает это как ошибку. Типичный фрейм сообщения показан ниже.

старт	адрес	ф-ия	данные	LRC	конец
1 сим	2 сим	2 сим	n сим	2 сим	2 сим
:					CR LF

Исключение: В контроллерах типа 584 и 984A/B/X ASCII-сообщение может нормально заканчиваться после контрольной суммы без CRLF последовательности. Интервалы меньше 1 сек допускаются.

RTU фрейм.

В RTU режиме сообщение начинается с интервала тишины равного времени передачи 3.5 символов при данной скорости передачи в сети. Первым полем затем передается адрес устройства.

Вслед за последним передаваемым символом также следует интервал тишины продолжительностью не менее 3.5 символов. Новое сообщение может начинаться после этого интервала.

Фрейм сообщения передается непрерывно. Если интервал тишины продолжительностью 1.5 возник во время передачи фрейма, принимающее устройство заканчивает прием сообщения и следующий байт будет воспринят как начало следующего сообщения.

Таким образом, если новое сообщение начнется раньше 3.5 интервала, принимающее устройство воспримет его как продолжение предыдущего сообщения. В этом случае устанавливается ошибка, так как будет несовпадение контрольных сумм. Типичный фрейм сообщения показан ниже.

старт	адрес	функция	данные	CRC	конец
T1-T2-T3-T4	8 бит	8 бит	n x бит	16 бит	T1-T2-T3-T4

Содержание адресного поля.

Адресное поле фрейма содержит два символа (ASCII) или 8 бит (RTU). Допустимый адрес передачи находится в диапазоне 0 - 247. Каждому подчиненному устройству присваивается адрес в пределах от 1 до 247.

Адрес 0 используется для широковещательной передачи, его распознает каждое устройство. Когда MODBUS протокол используется на более высоком уровне сети, широковещательная передача может не поддерживаться или может быть реализована другими методами.

Содержание поля функции.

Поле функции фрейма содержит два символа (ASCII) или 8 бит (RTU). Диапазон числа 1 -255. Некоторые функции работают на всех контроллерах MODICON, некоторые - на определенных моделях, другие же коды зарезервированы для будущего использования. Имеющийся набор функций описан в приложении 2.

Когда подчиненный отвечает главному, он использует поле кода функции для фиксации ошибки. В случае нормального ответа подчиненный повторяет оригинальный код функции. Если имеет место ошибка, возвращается код функции с установленным в 1 старшим битом.

Например, сообщение от главного подсиненному прочитать группу регистров имеет следующий код функции:

0000 0011 (03 hex) Если подчиненный выполнил затребованное действие без ошибки, он возвращает такой же код. Если имеет место ошибка, то он возвращает:

1000 0011 (83 hex) В добавление к изменению кода функции, подчиненный размещает в поле данных уникальный код, который говорит главному какая именно ошибка произошла или причину ошибки.

Содержание поля данных.

Поле данных в сообщении от главного к подчиненному содержит дополнительную информацию, которая необходима подчиненному для выполнения указанной функции. Оно может содержать адреса регистров или выходов, их количество, счетчик передаваемых байтов данных.

Например, если главный запрашивает у подчиненного прочитать группу регистров (код функции 03), поле данных содержит адрес начального регистра и количество регистров. Если главный хочет записать группу регистров (код функции 10 hex), поле данных содержит адрес начального регистра, количество регистров, счетчик количества байтов данных и данные для записи в регистры.

Поле данных может не существовать (иметь нулевую длину) в определенных типах сообщений.

Содержание поля контрольной суммы.

В MODBUS - сетях используются два метода контроля ошибок передачи. Содержание поля контрольной суммы зависит от выбранного способа передачи. ASCII Когда используется ASCII-режим поле контрольной суммы содержит два ASCII-символа. Контрольная сумма является результатом вычисления Longitudinal Redundancy Check (LRC) сделанного над содержанием сообщения начиная с ":" и заканчивая CRLF. RTU Когда используется RTU-режим поле контрольной суммы содержит 16-ти битовую величину. Контрольная сумма является результатом вычисления Cyclical Redundancy Check сделанного над содержанием сообщения. CRC добавляется к сообщению последним полем младшим байтом вперед.

Формат передачи символов.

Передача символов идет младшим битом вперед.

ASCII фрейм

С контролем четности

```
------T-----T-----T-----T-----T-----T-----T-----T-----T------¬
¦старт¦  1  ¦  2  ¦  3  ¦  4  ¦  5  ¦  6  ¦  7  ¦ Пар ¦Стоп ¦
L-----+-----+-----+-----+-----+-----+-----+-----+-----+------
```

Без контроля четности

```
------T-----T-----T-----T-----T-----T-----T-----T-----T------¬
¦старт¦  1  ¦  2  ¦  3  ¦  4  ¦  5  ¦  6  ¦  7  ¦Стоп ¦Стоп ¦
L-----+-----+-----+-----+-----+-----+-----+-----+-----+------
```

RTU фрейм

С контролем четности

```
------T-----T-----T-----T-----T-----T-----T-----T-----T-----T------¬
¦старт¦  1  ¦  2  ¦  3  ¦  4  ¦  5  ¦  6  ¦  7  ¦  8  ¦ Пар ¦Стоп ¦
L-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+------
```

Без контроля четности

```
------T-----T-----T-----T-----T-----T-----T-----T-----T-----T------¬
¦старт¦  1  ¦  2  ¦  3  ¦  4  ¦  5  ¦  6  ¦  7  ¦  8  ¦Стоп ¦Стоп ¦
L-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+------
```

Методы контроля ошибок.

Стандартная MODBUS сеть использует два метода контроля ошибок. Контроль паритета (even/odd) и контрольная сумма. Обе эти проверки генерируются в головном устройстве. Подчиненное устройство проверяет каждый байт и все сообщение в процессе приема.

Пользователь может устанавливать продолжительность интервала таймаута в течении которого головное устройство будет ожидать ответа от подчиненного. Если подчиненный обнаружил ошибку передачи, то он не формирует ответ главному.

Контроль паритета.

Пользователь может конфигурировать контроллеры на проверку четного или нечетного паритета (Even/Odd).

Например, 8 бит RTU-режима содержат следующую информацию:

1100 0101 Общее количество единиц - 4. Если используется четный паритет, то бит паритета будет равен 0, и общее количество 1-иц будет по прежнему четным числом. Если используется нечетный паритет, то бит паритета будет равен 1, тогда общее количество 1-иц вместе с битом паритета будет равно 5, т.е. нечетному числу.

Контрольная сумма LRC.

Метод LRC проверяет содержание сообщения исключая начальный символ ":" и пару CRLF.

LRC это 1 байт. LRC вычисляется передающим устройством и добавляется в конец сообщения. Принимающее устройство вычисляет LRC в процессе приема сообщения и сравнивает его с принятым от главного. Если есть несовпадение, то имеет место ошибка.

Контрольная сумма CRC.

Контрольная сумма CRC состоит из двух байт. Контрольная сумма вычисляется передающим устройством и добавляется в конец сообщения. Принимающее устройство вычисляет контрольную сумму в процессе приема и сравнивает ее с полем CRC принятого сообщения.

Счетчик контрольной суммы предварительно инициализируется числом FF hex. Только восемь бит данных используются для вычисления контрольной суммы CRC. Старт и стоп биты, бит паритета, если он используется, не учитываются в контрольной сумме.

Во время генерации CRC каждый байт сообщения складывается по исключающему ИЛИ с текущим содержимым регистра контрольной суммы. Результат сдвигается в направлении младшего бита, с заполнением нулем старшего бита. Если младший бит равен 1, то производится исключающее ИЛИ содержимого регистра контрольной суммы и определенного числа. Если младший бит равен 0, то исключающее ИЛИ не делается.

Процесс сдвига повторяется восемь раз. После последнего (восьмого) сдвига, следующий байт складывается с текущей величиной регистра контрольной суммы, и процесс сдвига повторяется восемь раз как описано выше. Конечное содержание регистра и есть контрольная сумма CRC.

Функции контроля и обработки данных.

01 Чтение статуса выходов

ОПИСАНИЕ

Читает статуса ON/OFF дискретных выходов в подчиненном.

ЗАПРОС

Запрос содержит адрес начального выхода и количество выходов для чтения. Выхода адресуются начиная с нуля: выхода 1-16 адресуются как 0-15.

Ниже приведен пример запроса на чтение выходов 20-56 с подчиненного устройства 17.

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	01	
Начальный адрес Hi		00
Начальный адрес Lo		13
Количество Hi		00
Количество Lo		25
Контрольная сумма (CRC или LRC)	--	

ОТВЕТ

Статус выходов в ответном сообщении передается как один выход на бит.

Если возвращаемое количество выходов не кратно восьми, то оставшиеся биты в последнем байте сообщения будут установлены в 0. Счетчик байт содержит количество байт передаваемых в поле данных.

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	01	
Счетчик байт 4		005
Данные(Выхода 27-20)		CD
Данные(Выхода 35-28)		6B
Данные(Выхода 43-36)		B2
Данные(Выхода 51-44)		0E
Данные(Выхода 56-52)		1B
Контрольная сумма (CRC или LRC)	--	

02 Read Input Status

ОПИСАНИЕ

Чтение ON/OFF состояния дискретных входов (ссылка 1X) в пдчиненном.

ЗАПРОС

Запрос содержит номер начального входа и количество входов для чтения. Входа адресуются начиная с 0.

Ниже приведен пример запроса на чтение входов 10197-10218 с подчиненного устройства 17.

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	02	
Начальный адрес ст.		00
Начальный адрес мл.		C4
Кол-во входов ст.	00	
Кол-во входов мл.	16	
Контрольная сумма		--

ОТВЕТ

Статус входов в ответном сообщении передается как один выход на бит.

Если возвращаемое количество входов не кратно восьми, то оставшиеся биты в последнем байте сообщения будут установлены в 0. Счетчик байт содержит количество байт передаваемых в поле данных.

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	01	
Счетчик байт 4		005
Данные(Входы 10204-10197)		AC
Данные(Входы 10212-10205)		DB
Данные(Входы 10218-10213)		35
Контрольная сумма (CRC или LRC)	--	

03 Read Holding Registers

ОПИСАНИЕ

Чтение двоичного содержания регистров (ссылка 4X) в подчиненном.

ЗАПРОС

Сообщение запроса специфицирует начальный регистр и количество регистров для чтения. Регистры адресуются начина с 0: регистры 1-16 адресуются как 0-15.

Ниже приведен пример чтения регистров 40108-40110 с подчиненного устройства 17.

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	03	
Начальный адрес ст.		00

Начальный адрес мл.	6B
Кол-во регистров ст.	00
Кол-во регистров мл.	03
Контрольная сумма	--

ОТВЕТ

Данные регистров в ответе передаются как два бйта на регистр. Для каждого регистра, первый байт содержит старшие биты второй байт содержит младшие биты.

За одно обращение может считываться 125 регистров для контроллеров 984-X8X (984-685 и т.д.), и 32 регистра для других контроллеров. Ответ дается когда все данные укомплектованы.

Это пример ответа на запрос представленный выше:

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	03
Счетчик байт	06
Данные (регистр 40108) ст.	02
Данные (регистр 40108) мл.	2B
Данные (регистр 40109) ст.	00
Данные (регистр 40109) мл.	00
Данные (регистр 40110) ст.	00
Данные (регистр 40110) мл.	64
Контрольная сумма	--

04 Read Input Registers

СОДЕРЖАНИЕ

Чтение двоичного содержания входных регистров (ссылка 3X) в подчиненном.

ЗАПРОС

Запрос содержит номер начального регистра и количество регистров для чтения.

Ниже приведен пример запроса для чтения регистра 30009 с подчиненного устройства 17.

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	03
Начальный адрес ст.	00
Начальный адрес мл.	6B
Кол-во регистров ст.	00
Кол-во регистров мл.	03
Контрольная сумма	--

ОТВЕТ

Данные регистров в ответе передаются как два бйта на регистр. Для каждого регистра, первый байт содержит старшие биты второй байт содержит младшие биты.

За одно обращение может считываться 125 регистров для контроллеров 984-X8X (984-685 и т.д.), и 32 регистра для других контроллеров. Ответ дается когда все данные укомплектованы.

Это пример ответа на запрос представленный выше:

Ответ

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	03	
Счетчик байт		02
Данные (регистр 30009) ст.	00	
Данные (регистр 30009) мл.		2A
Контрольная сумма		--

05 Force Single Coil

ОПИСАНИЕ

Установка единичного выхода (ссылка 1X) в ON или OFF. При широковещательной передаче функция устанавливает все выходы с данным адресом во всех подчиненных контроллерах.

ЗАМЕЧАНИЕ Функция может пересекаться с установкой защиты

памяти и установкой недоступности выходов.

ЗАПРОС

Запрос содержит номер выхода для установки. Выходы адресуются начиная с 0. Выход 1 адресуется как 0.

Состояние, в которое необходимо установить выход (ON/OFF) описывается в поле данных. Величина FF00 Hex - ON. Величина 0000 - OFF. Любое другое число неверно и не влияет на выход.

В приведенном ниже примере устанавливается выход 173 в состояние ON в подчиненном устройстве 17.

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	05	
Адрес выхода мл.	00	
Адрес выхода ст.		AC
Данные ст.		FF
Данные мл.		00
Контрольная сумма		--

ОТВЕТ

Нормальный ответ повторяет запрос.

Ответ

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	05	
Адрес выхода мл.	00	
Адрес выхода ст.		AC
Данные ст.		FF
Данные мл.		00

Контрольная сумма	--

06 Preset Single Register

ОПИСАНИЕ

Записывает величину в единичный регистр (ссылка 4X). При широковезательной передаче на всех подчиненных устройствах устанавливается один и тот же регистр.

ЗАМЕЧАНИЕ

Функция может пересекаться с установленной защитой памяти.

ЗАПРОС

Запрос содержит ссылку на регистр, который необходимо установить. Регистры адресуются с 0.

Величина, в которую необходимо установить регистр передается в поле данных. Контроллеры M84 и 484 используют 10-ти битную величину, старшие шесть бит заполняются 0. Все другие контроллерыиспользуют 16 бит.

В приведенном ниже примере в регистр 40002 записывается величина 0003 Hex в подчиненном устройстве 17.

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	06
Адрес регистра мл.	00
Адрес регистра ст.	01
Данные ст.	00
Данные мл.	03
Контрольная сумма	--

ОТВЕТ

Нормальный ответ повторяет запрос.

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	06
Адрес регистра мл.	00
Адрес регистра ст.	01
Данные ст.	00
Данные мл.	03
Контрольная сумма	--

07 Read Exception Status

ОПИСАНИЕ

Чтение статусных выходов в подчиненном контроллере. Назначение некоторых выходов в контроллерах заранее определено. Другие же могут программироваться пользователем для вывода информации о контроллере, например "машина ВКЛ/ВЫКЛ" и др.

Функция обеспечивает простой метод доступа к данной информации, потому что адрес статусных выходов известен (нет необходимости указывать адрес выхода).

Статусные выходы имеют следующие предопределения:

Модель контроллера	Выход	Назначение

M84,184/384,584,984 1-8		Определяемые пользователем
484	257	Состояние батареи
	258-264	Определяемые пользователем
884	761	Состояние батареи
	762	Статус Защиты памяти
	763	RIO Health Status
	764-768	Определяемые пользователем

ЗАПРОС

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	07
Контрольная сумма	--

ОТВЕТ

Нормальный ответ содержит состояние восьми выходов статуса.

Пример ответа на запрос описанный выше:

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	06
Данные выходов	6D
Контрольная сумма	--

11 (0B Hex) Fetch Comm Event Ctr

ОПИСАНИЕ

Возвращает слово состояния и счетчик событий из коммуникационного счетчика событий подчиненного устройства. Просматривая текущий счетчик перед и после серии сообщений , главный может увидеть какие сообщения были нормально приняты подчиненным.

Счетчик событий инкрементируется после каждого удачного приема сообщения. Он не инкрементируется после сообщений об ошибке, команд регистрации, или команд просмотра счетчика событий.

Счетчик событий может быть переустановлен посредством Диагностической функции (код 08), подфункциями Restart Communication Option (код 00 01) или Clear Counters and Diagnostic Register (код 00 0A).

ЗАПРОС

Ниже приведен пример запроса на чтение коммуникационного счетчика событий в подчиненном устройстве 17:

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	0B
Контрольная сумма	--

ОТВЕТ

Нормальный ответ содержит два байта слова состояния, и два байта счетчика событий. Слово состояния содержит все 1 если предыдущая команда еще выполняется. Иначе слово состояния содержит все нули.

Ниже приведен пример ответа на описанный выше запрос:

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	0B
Состояние ст.	FF
Состояние мл.	FF
Счетчик событий ст.	01
Счетчик событий мл.	08
Контрольная сумма	--

В этом примере слово состояния содержит число FFFF Hex, что говорит о том, что в подчиненном все еще выполняется функция. Счетчик событий содержит число 264, что говорит о том, что на данный момент произошло 264 события.

12 (0C Hex) Fetch Comm Event Log

ОПИСАНИЕ

Возвращает слово состояния, счетчик событий, счетчик сообщений, и поле байтов событий.

Слово состояния и счетчик событий идентичны возвращаемым функцией 11 (0B Hex).

Счетчик сообщений обработанных подчиненным со времени последнего рестарта, операции очистки счетчиков или с момента включения питания. Этот счетчик идентичен возвращаемому функцией диагностики (код 08), подфункцией Return Bus Message Count (код 11).

Поле байтов событий содержит 0-64 байта, каждый байт содержит статус одной посылки или приема сообщения подчиненным. События размещаются подчиненным в поле в хронологическом порядке. Байт 0 содержит последнее по времени событие.

ЗАПРОС

Ниже приведен пример запроса на посылку коммуникационного журнала сообщений главному с подчиненного устройства 17:

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	0C
Контрольная сумма	--

ОТВЕТ

Нормальный ответ содержит два байта слова состояния, два байта счетчика событий, два байта счетчика сообщений, и поля содержащего 0-64 байта описания событий. Поле счетчика данных определяет общую длину данных в четырех полях.

Это пример ответа на запрос представленный выше:

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	0C
Счетчик байтов	08
Состояние ст.	00
Состояние мл.	00
Счетчик событий ст.	01
Счетчик событий мл.	08
Счетчик сообщений ст.	01

Счетчик сообщений мл.	21
Событие 0	20
Событие 1	00
Контрольная сумма	--

Вданном примере, слово состояния содержит 00 00 Hex, это говорит о том, что подчиненный не занят выпонением программной функции. Счетчик событий содержит 264. Счетчик сообщений показывает, что 289 сообщений было обработано.

Последнее коммуникационное событие показано в 0-ом байте событий. Его содержимое говорит о том, что подчиненный был введен в режим Listen Only Mode.

Предыдущее событие показано в 1-ом байте событий. Его содержимое (00) говорит о том, что подчиненный принял Communications Restart.

Содержимое байтов событий описано ниже.

ЧТО СОДЕРЖАТ БАЙТЫ СОБЫТИЙ

Байты событий, возвращаемые функцией Fetch Communication Event Log, могут быть одним из четырех типов. Тип определяется старшим битом в каждом байте. Кроме того, он может определятся битом 6.

Slave Modbus Receive Event

Этот тип байта события сохраняется подчиненным когда сообщение-запрос было принято. Он сохраняется перед тем как подчиненный обработает сообщение. Этот тип события определяется установкой бита 7 в "1". Остальные биты устанавливаются в "1" если произошли описанные ниже события:

Бит	Описание
0	Не используется
1	Коммуникационная ошибка
2	Не используется
3	Не используется
4	Character Overrun
5	Currently in Listen Only Mode
6	Широковещательный прием
7	1

Slave Modbus Send Event

Этот тип байта события сохраняется подчиненным когда он закончил обработку сообщения запроса. Он сохраняется если подчиненный возвращает нормальный ответ или возвращает сообщение об ошибке, или не отвечает на запрос. Это событие определяется установкой бита 7 в 0, с установкой бита 6 в 1. Остальные биты устанавливаются в 1 если произошли описанные ниже события:

Бит	Описание
0	Read Exeption Sent (Код ошибки 1-3)
1	Slave Abort Exeption Sent (Код ошибки 4)
2	Slave Busy Exeption Sent (Код ошибки 5-6)
3	Slave Program NAK Exeption Sent (Код ошибки 7)
4	Write Timeout Error Occured
5	Currently in Listen Mode
6	1
7	0

Slave Entered Listen Only Mode

Этот тип байта события сохраняется подчиненным когда он вводится в режим Listen Only Mode.

Описание битов:

Бит	Описание
0	0

1	0
2	1
3	0
4	0
5	0
6	0
7	0

Slave Initiated Communication Restart

Этот тип байта события сохраняется подчиненным когда был сделан рестарт коммуникационного порта. Рестарт подчиненного может быть сделан посредством диагностической функции (код 08 Hex) подфункция Restart Communication Option (код 00 01).

Эта функция также может переводить подчиненного в режим 'Continue on Error' или 'Stop on Error'. Если подчиненный находится в режиме 'Continue on Error Mode', байт события добавляется к существующему журналу событий. Если подчиненный находится в режиме 'Stop on Error' байт добавляется в журнал и остаток журнала сбрасывается в 0.

Описание битов

Бит	Содержание
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

15 (0F Hex) Force Multiple Coils

ОПИСАНИЕ

Устанавливает каждый выход (ссылка 0X) последовательности выходов в одно из состояний ON или OFF. При широковещательной передаче функция устанавливает подобные выходы на всех всех подчиненных.

ЗАМЕЧАНИЕ Функция может пересекаться с установкой защиты памяти и установкой недоступности выходов.

ЗАПРОС

Запрос специфицирует выходы для установки. Выходы адресуются начиная с 0.

Ниже показан пример запроса на установку последовательности выходов начиная с 20 (адресуется как 19) в подчиненном устройстве 17.

Поле данных запроса содержит 2 байта: CD 01 Hex (1100 1101 0000 0001 двоичное). Соответствие битов и выходов представлено ниже:

Бит: 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0 1

Выход: 27 26 25 24 23 22 21 20 - - - - - - 29 28

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	0F	
Адрес выхода ст.		00
Адрес выхода мл.	13	
Кол-во выходов ст.		00
Кол-во выходов мл.		0A

Счетчик байт	02
Данные для установки (Выходы 27-20)	CD
Данные для установки (Выходы 29-28)	01
Контрольная сумма	--

ОТВЕТ

Нормальный ответ возвращает адрес подчиненного, код функции, начальный адрес, и количество установленных выходов.

Это пример ответа на представленный выше запрос.

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	0F
Адрес выхода ст.	00
Адрес выхода мл.	13
Кол-во выходов ст.	00
Кол-во выходов мл.	0A
Контрольная сумма	--

16 (10 Hex) Preset Multiple Regs

ОПИСАНИЕ

Запись данных в последовательность регистров (ссылка 4X). При широковещательной передаче, функция устанавливает подобные регистры во всех подчиненных устройствах.

ЗАМЕЧАНИЕ

Функция может пересекаться с установленной защитой памяти.

ЗАПРОС

Запрос специфицирует регистры для записи. Регистры адресуются начиная с 0.

Данные для записи в регистры содержатся в поле данных запроса. Контроллеры M84 и 484 используют 10-битовую величину, со старшими шестью битами установленными в 0. Все остальные контроллеры используют 16 бит.

Ниже приведен пример запроса на установку двух регистров начиная с 40002 в 00 0A и 01 02 Hex, в подчиненном устройстве 17:

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	10
Начальный адрес	00
Начальный адрес	01
Кол-во регистров ст.	00
Кол-во регистров мл.	02
Счетчик байт	02
Данные ст.	00
Данные мл.	0A
Данные ст.	01
Данные мл.	02

Контрольная сумма	--

ОТВЕТ

Нормальный ответ содержит адрес подчиненного, код функции, начальный адрес, и количество регистров.

17 (11 Hex) Чтение идентификатора подчиненного

ОПИСАНИЕ

Возвращает описание типа контроллера представленного по данному адресу, текущее состояние индикатора пуска, и другую информацию о подчиненном устройстве.

ЗАПРОС

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	11	
Контрольная сумма (CRC или LRC)	--	

ОТВЕТ

Формат нормального ответа показан ниже. Содержание байтов данных специфично для каждого типа контроллера.

Имя поля		Пример (Hex)
Адрес подчиненного		Возврат адреса подчиненного
Функция	11	
Счетчик байт 4		0Зависит от устройства
Идентификатор у-ва		Зависит от устройства
Индикатор пуска		00=OFF, FF=ON
Добавочная информация		Зависит от устройства ...
Контрольная сумма (CRC или LRC)	--	

Список идентификаторов устройств

Идентификатор	Контроллер
0	Micro 84
1	484
2	184/384
3	584
8	884
9	984

184/384

184 или 384 контроллеры возвращают счетчик байт 4 или 74. Если J347 Modbus Slave Interface инициализирован правильно, и если внутренняя PIB-таблица нормальная, счетчик байт будет содержать 74. Иначе - 4.

Эти четыре байта будут возвращены в любом случае:

Байт	Содержимое
1	Идентификатор (2 для 184/384)
2	Индикатор пуска (0=OFF, FF=ON)
3,4	Слово статуса:
	Бит 0 = 0

Бит 1 = Защита памяти (0=OFF, 1=ON)

Бит 2,3 = Тип контроллера :

Бит 2=0 и Бит 3=0 - 184

Бит 2=1 и Бит 3=0 - 384

Биты 4-15 = не используются

Дополнительные 70 байт:

Байт	Содержание
5,6	Начальный адрес PIB-таблицы
7,8	Серийный номер контроллера
9,10	Идентификатор исполнителя
	Байты 11-74 содержат PIB-таблицу.
	Эти данные верны если контроллер работает
	(см. байт 2).
11,12	Максимальное количество выходов
13,14	Таблица доступных выходов
15,16	Адрес таблицы входов coil/run
17,18	Количество входов
19,20	Таблица доступных входов
21,22	Первое ключевое число
	(должно быть кратно 16)
23,24	Последнее ключевое число
	(должно быть кратно 16)
25,26	Адрес входных регистров
27,28	Количество входных регистров
29,30	Количество выходных и захваченных
	регистров
31,32	Адрес пользовательской логики
33,34	Адрес output coil RAM table
35,36	Маска недоступных функций
37,38	Адрес программы расширенной функции
39,40	Адрес программы передачи данных
41,42	Адрес traffic cop
43,44	Не используются
45,46	Маска недоступных функций
47,48	Адрес таблицы истории "A"-режима
49,50	Таблица запросов для DX принтера
51,52	Количество групп последовательностей
53,54	Адрес таблицы последовательностей отображения
55,56	Адрес последовательности RAM (адрес ОЗУ)
57,58	Количество 50XX регистров
59,60	Адрес таблицы 50XX
61,62	Адрес output coil RAM image
63,64	Адрес input RAM image
65,66	Delayed output start group
67,68	Delayed output end group

69,70	Сторожевой таймер
71,72	RAM adress of latches (адрес ключей в ОЗУ)
73,74	Адрес delayed output groups

584

584 контроллер возвращает счетчик байт = 9.

Байт	Содержание
1	Идентификатор (3 для 584)
2	Индикатор пуска (0=OFF, 1=ON)
3	Количество 4К секций нулевой страницы памяти
4	Количество 1К секций в статическом ОЗУ
5	Количество сегментов пользовательской логики
6,7	Слово состояния машины (слово таблицы конфигурации 101, 65 Hex) Организация этого слова показана ниже: Байт 6: Бит 15 = Port 1 setup Бит 14 = Port 2 setup Бит 13 = Port 1 adress set Бит 12 = Port 2 adress set Бит 11 = Не присвоен Бит 10 = Статус Constant Sweep (0=Constatnt Sweep OFF, 1=ON) Бит 9 = Статус Single Sweep (0=Single Sweep OFF, 1=ON) Бит 8 = 16/24-битовый узел (0=24-битовый, 1=16-битовый) Байт 7: Бит 7 = Питание ON (1=ON, никогда OFF) Бит 6 = RUN - индикатор (0=ON, 1=OFF) Бит 5 = Защита памяти (0=ON, 1=OFF) Бит 4 = Состояние батареи (0=OK, 1=Not OK) Биты 3-0 Не используются
8,9	Машинный код стопа (таблица конфигурации слово 105, 69 Hex) Организация этого слова показана ниже: Байт 8: Бит 15 = Стоп переферийного порта (стоп контроллера) Бит 14 = Не используется Бит 13 = Dim awareness Бит 12 = Недопустимое вторжение перифе- рии

Бит 11 = Неправильная таблица задач

Бит 10 = Старт узла не является стартом сегмента

Бит 9 = Ошибка теста статического ОЗУ

Бит 8 = Не обнаружен конец логики, или неверное количество сегментов

Бит 9:

Бит 7 = Сторожевой таймер накрылся

Бит 6 = Ошибка часов реального времени

Бит 5 = Ошибка диагностики процессора

Бит 4 = Неправильный тип traffic cop

Бит 3 = Неправильный тип узла

Бит 2 = Ошибка контрольной суммы логики

Бит 1 = Ошибка резервной контрольной суммы

Бит 0 = Неверная конфигурация

984

984 контроллер возвращает счетчик байт = 9.

Байт	Содержание
1	Идентификатор подчиненного (9 для 984)
2	Статус RUN-индикатора (0=OFF, 1=ON)
3	Количество 4К секций нулевой страницы памяти
4	Количество 1К секций в статическом ОЗУ
5	Количество сегментов пользовательской логики
6,7	Слово состояния машины (слово таблицы конфигурации 101, 65 Hex)

Организация этого слова показана ниже:

Байт 6:

Бит 15 = Не используется

Бит 14-11 = Не используется

Бит 10 = Статус Constant Sweep (0=Constatnt Sweep OFF, 1=ON)

Бит 9 = Статус Single Sweep (0=Single Sweep OFF, 1=ON)

Бит 8 = 16/24-битовый узел (0=24-битовый, 1=16-битовый)

Байт 7:

Бит 7 = Питание ON (1=ON, никогда OFF)

Бит 6 = RUN - индикатор (0=ON, 1=OFF)

Бит 5 = Защита памяти (0=ON, 1=OFF)

Бит 4 = Состояние батареи (0=OK, 1=Not OK)

Биты 3-1 = Не используются

Бит 0 = Флаг размера памяти

Размер памяти : Бит 0 слова состояния машины определяет использование величины размера памяти в словах 00, 100, 175 (63, 64, AF Hex) таблицы конфигурации. Если бит 0 = логической 1, размер памяти вычисляется следующим образом:

Размер 0 страницы(16 бит)=(Слово 99*4096)-(Слово 175 мл.байт*16)

Размер статической таблицы(16 бит) =

(Слово 100*1024)-(Слово 175 ст. байт*16)

8,9	Машинный код стопа (таблица конфигурации слово 105, 69 Hex) Организация этого слова показана ниже: Байт 8: Бит 15 = Стоп периферийного порта (стоп контроллера) Бит 14 (984A,B,X) = Ошибка паритета расширенной памяти Бит 13 = Dim awarness Бит 12 = Недопустимое вторжение периферии Бит 11 = Неверная таблица списков сегментов Бит 10 = Старт узла не является стартом сегмента Бит 9 = Ошибка теста статического ОЗУ Бит 8 = Не обнаружен конец логики, или неверное количество сегментов Байт 9: Бит 7 = Сторожевой таймер накрылся Бит 6 = Ошибка часов реального времени Бит 5 (984A, B, X) = Ошибка диагностики процессора Бит 5 (Другие 984) = Плохая таблица использования выходов Бит 4 = S908 remote IO head failure Бит 3 = Неправильный тип узла Бит 2 = Ошибка контрольной суммы логики Бит 1 = Выхода недоступны пока контроллер работает Бит 0 = Неверная конфигурация

Micro 84

Micro 84 возвращает счетчик байт 8:

Байт	Содержание
1	Идентификатор подчиненного (0 для Micro 84)
2	Статус индикатора RUN (0=OFF, FF=ON)
3	Текущий номер порта
4	Размер памяти (1 = 1K, 2 = 2K)
5	Не используется (все 0)

484

484 контроллер возвращает счетчик байт 5:

Байт	Содержание

1	Идентификатор подчиненного (1 для 484)
2	Статус индикатора RUN (0=OFF, FF=ON)
3	Установка системы
4	Первый байт конфигурации
5	Второй байт конфигурации

884

884 контроллер возвращает счетчик байт 8:

Байт	Содержание
1	Идентификатор подчиненного (0 для Micro 84)
2	Статус индикатора RUN (0=OFF, FF=ON)
3	Текущий номер порта
4	Размер пользовательской логики плюс статическая память, в килобайтах
5	Зарезервирован
6	Биты: Бит 0-2 = Зарезервированы Бит 3 = Обход mapper: 1 = не использовать стандартный mapper Бит 4 = Конец теста Scan 1 = Конец теста сканирования Бит 5 = Зарезервирован Бит 6 = Обход решателя логики: 1 = не использовать стандартный решатель Бит 7 = зарезервирован
7,8	Зарезервированы

20 (14 Hex) Read General Reference

ОПИСАНИЕ

Возвращает содержание регистров файла расширенной памяти (6XXXX). Широкое вещание не поддерживается.

Функция может читать несколько групп. Группы могут быть разделены, но посылка внутри каждой группы должна быть непрерывной.

ЗАПРОС

Запрос специфицирует группу или группы для чтения. Каждая группа определяется в поле "суб-запроса" которое содержит 7 байт:

- Тип ссылки : 1 байт (должен быть специфицирован как 6)

- Номер файла расширенной памяти: 2 байта (от 1 до 10)

- Начальный адрес регистра внутри файла: 2 байта

- Количество регистров для чтения: 2 байта.

Количество регистров для чтения, вместе с другими полями в ответе, не должно превышать допустимую длину MODBUS-сообщения: 256 байт.

Доступное количество файлов расширенной памяти зависит от установленного размера расширенной памяти в подчиненном контроллере. Каждый файл, исключая последний, содержит 10000 регистров, адресуемых как 0000-270F Hex (0000-9999).

Для контроллеров кроме 984-785:

Размер расш. памяти	Кол-во файлов	Остаточные регистры
16K	2	6383
32K	4	2767

64K	7	5535
96K	10	8303

Для контроллеров 984-785:

984-785 с картриджем AS-M785-032:

Польз. логика	Статич. ОЗУ	Размер расш. пам.	Кол. файлов	Ост. рег.
32K	32K	0	0	0
16K	64K	72K	8	3727

984-785 с картриджем AS-M785-048:

Польз. логика	Статич. ОЗУ	Размер расш. пам.	Кол. файлов	Ост. рег.
48K	32K	24K	3	4575
32K	64K	96K	10	8303

Пример чтения двух групп с подчиненного устройства 17 показан ниже.

Группа 1 состоит из двух регистров из файла 4, начиная с регистра 2 (адрес 0001).

Группа 2 состоит из двух регистров из файла 3, начиная с регистра 10 (адрес 0009).

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	14	
Счетчик байт		0E
Суб-запрос 1, Тип ссылки	06	
Суб-запрос 1, Номер файла ст.		00
Суб-запрос 1, Номер файла мл.		04
Суб-запрос 1, Стартовый адрес ст	.	00
Суб-запрос 1, Стартовый адрес мл.	01	
Суб-запрос 1, Кол-во регистров ст.	00	
Суб-запрос 1, кол-во регистров мл.	02	
Суб-запрос 2, тип ссылки		06
Суб-запрос 2, Номер файла ст.		00
Суб-запрос 2, Номер файла мл.		03
Суб-запрос 2, Стартовый адрес ст.		00
Суб-запрос 2, Стартовый адрес мл.	09	
Суб-запрос 2, Кол-во регистров ст.	00	
Суб-запрос 2, кол-во регистров мл.	02	
Контрольная сумма (LRC или CRC)	--	

ОТВЕТ

Нормальный ответ состоит из серии суб-ответов, один на каждый суб-запрос.

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	14	
Счетчик байт		0E
Суб-ответ 1, Счетчик байт	0C	
Суб-ответ 1, Тип ссылки		06

Суб-ответ 1, Данные регистра ст.		0D
Суб-ответ 1, Данные регистра мл.		FE
Суб-ответ 1, Данные регистра ст.		00
Суб-ответ 1, Данные регистра мл.		20
Суб-ответ 2, Счетчик байт	05	
Суб-ответ 2, Тип ссылки		06
Суб-ответ 2, Данные регистра ст.		33
Суб-ответ 2, Данные регистра мл.		CD
Суб-ответ 2, Данные регистра ст.		00
Суб-ответ 2, Данные регистра мл.		40
Контрольная сумма		--

21 (15 Hex) Write general reference

Описание

Запись содержимого регистров в файл расширенной память (6XXXX).

ЗАПРОС

Запрос специфицирует группу или группы для записи, и данные которые содержатся в них.

Запрос специфицирует группу или группы для чтения. Каждая группа определяется в поле "суб-запроса" которое содержит 7 байт:

- Тип ссылки : 1 байт (должен быть специфицирован как 6)

- Номер файла расширенной памяти: 2 байта (от 1 до 10)

- Начальный адрес регистра внутри файла: 2 байта

- Количество регистров для чтения: 2 байта.

Количество регистров для записи, вместе с другими полями в ответе, не должно превышать допустимую длину MODBUS-сообщения: 256 байт.

Доступное количество файлов расширенной памяти зависит от установленного размера расширенной памяти в подчиненном контроллере. Каждый файл, исключая последний, содержит 10000 регистров, адресуемых как 0000-270F Hex (0000-9999).

Для контроллеров кроме 984-785:

Размер расш. памяти	Кол-во файлов	Остаточные регистры
16K	2	6383
32K	4	2767
64K	7	5535
96K	10	8303

Для контроллеров 984-785:

984-785 с картриджем AS-M785-032:

Польз. логика	Статич. ОЗУ	Размер расш. пам .	Кол. файлов	Ост. рег.
32K	32K	0	0	0
16K	64K	72K	8	3727

984-785 с картриджем AS-M785-048:

Польз. логика	Статич. ОЗУ	Размер расш. пам.	Кол. файлов	Ост. рег.
48K	32K	24K	3	4575
32K	64K	96K	10	8303

Пример запроса и ответа приведен ниже.

В данном примере требуется записать одну группу в подчиненное устройство 17.

Группа состоит из трех регистров в файле 4, начиная с регистра 8 (адрес 0007).

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	15	
Счетчик байт		0D
Суб-запрос 1, тип ссылки		06
Суб-запрос 1, Номер файла ст.		00
Суб-запрос 1, Номер файла мл.		04
Суб-запрос 1, Начальный адрес ст.	00	
Суб-запрос 1, Начальный адрес мл.	07	
Суб-запрос 1, Кол-во регистров ст.	00	
Суб-запрос 1, Кол-во регистров мл.	03	
Суб-запрос 1, Данные регистра ст.		06
Суб-запрос 1, Данные регистра мл.	AF	
Суб-запрос 1, Данные регистра ст.		04
Суб-запрос 1, Данные регистра мл.	BE	
Суб-запрос 1, Данные регистра ст.		10
Суб-запрос 1, Данные регистра мл.	0D	
Контрольная сумма		--

Ответ

Нормальный ответ повторяет запрос.

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	15	
Счетчик байт		0D
Суб-ответ 1, тип ссылки		06
Суб-ответ 1, Номер файла ст.		00
Суб-ответ 1, Номер файла мл.		04
Суб-ответ 1, Начальный адрес ст.		00
Суб-ответ 1, Начальный адрес мл.	07	
Суб-ответ 1, Кол-во регистров ст.		00
Суб-ответ 1, Кол-во регистров мл.		03
Суб-ответ 1, Данные регистра ст.		06
Суб-ответ 1, Данные регистра мл.		AF
Суб-ответ 1, Данные регистра ст.		04
Суб-ответ 1, Данные регистра мл.		BE
Суб-ответ 1, Данные регистра ст.		10
Суб-ответ 1, Данные регистра мл.		0D
Контрольная сумма		--

22 (16 Hex) Mask Write 4X Register

Описание

Модифицирует содержание регистров 4XXXX используя комбинацию OR-маску, AND-маску и текущего содержимого регистра. Функция может использоваться для установки или сброса отдельного бита в регистре.

Функция поддерживается только 984-785 контроллерами.

ЗАПРОС

Алгоритм функции следующий:

Результат=(Текущ. знач. AND Маска_И) OR (Маска_ИЛИ AND ~Маска_И)

Например:	Hex	Двоичное
Текущее значение	12	0001 0010
Маска_И	F2	1111 0010
Маска_ИЛИ	25	0010 0101
~Маска_И	0D	0000 1101
Результат	17	0001 0111

Пример записи с маской в регистр 5 в подчиненное устройство 17, с приведенной выше маской описан ниже.

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	17
Функция	16
Адрес ссылки ст.	00
Адрес ссылки мл.	04
Маска_И ст.	00
Маска_И мл.	F2
Маска_ИЛИ ст.	00
Маска_ИЛИ мл.	25
Контрольная сумма	--

ОТВЕТ

Нормальный ответ повторяет запрос. Ответ возвращается после записи в регистр.

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	17
Функция	16
Адрес ссылки ст.	00
Адрес ссылки мл.	04
Маска_И ст.	00
Маска_И мл.	F2
Маска_ИЛИ ст.	00
Маска_ИЛИ мл.	25
Контрольная сумма	--

23 (17 Hex) Read/Write 4X Registers

ОПИСАНИЕ

Производит операцию чтения и записи за одну MODBUS транзакцию. Функция может записывать новое содержание регистров 4XXXX и возвращать содержание другой группы регистров 4XXXX. Эта функция поддерживается только на 984-785 контроллерах.

ЗАПРОС

Запрос специфицирует начальный адрес и количество регистров группы для чтения. И также специфицирует начальный адрес, количество регистров, и данные для записи в группу регистров. Счетчик байтов содержит количество байт передаваемых в поле данных.

В примере, приведенном ниже требуется прочитать шесть регистров начиная с пятого, и записать три регистра начиная с 16, в подчиненном устройстве 17:

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	17
Функция	17
Начальный регистр чтения ст.	00
Начальный регистр чтения мл.	04
Кол-во регистров для чтения ст.	00
Кол-во регистров для чтения мл.	06
Начальный регистр записи ст.	00
Начальный регистр записи мл.	0F
Кол-во регистров для записи ст.	00
Кол-во регистров для записи мл.	03
Счетчик байтов	06
Данные для записи 1 ст.	00
Данные для записи 1 мл.	FF
Данные для записи 2 ст.	00
Данные для записи 2 мл.	FF
Данные для записи 3 ст.	00
Данные для записи 3 мл.	FF
Контрольная сумма	--

ОТВЕТ

Нормальный ответ содержит данные прочитанных регистров.

Ответ

Имя поля	Пример (Hex)
Адрес подчиненного	17
Функция	17
Счетчик байт	0C
Считанные данные 1 ст.	00
Считанные данные 1 мл.	FE
Считанные данные 2 ст.	0A
Считанные данные 2 мл.	CD
Считанные данные 3 ст.	00
Считанные данные 3 мл.	01
Считанные данные 4 ст.	00
Считанные данные 4 мл.	03
Считанные данные 5 ст.	00
Считанные данные 5 мл.	0D
Считанные данные 6 ст.	00
Считанные данные 6 мл.	FF
Контрольная сумма	--

24 (18 Hex) Read FIFO Queue

ОПИСАНИЕ

Чтение содержимого очереди FIFO (регистры 4XXXX). Функция возвращает счетчик регистров в очереди, следом идут данные очереди. До 32 регистров могут быть считаны: счетчик, плюс 31 регистр данных очереди.

Функция читает содержимое очереди, но не очищает ее.

Функция поддерживается только на 984-785 контроллерах.

ЗАПРОС

Запрос специфицирует начальный регистр 4XXXX для чтения FIFO очереди. Это адрес регистра указателя использующегося в функциональных блоках FIN и FOUT контроллеров.

Ниже показан пример чтения FIFO очереди с подчиненного устройства 17. Чтение очереди начинается с регистра указателя 41247 (04DE Hex).

Запрос

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	18	
Адрес указателя FIFO ст.		04
Адрес указателя FIFO мл.		DE
Контрольная сумма		--

ОТВЕТ

В нормальном ответе, счетчик байт содержит количество следующих за ним байтов, включая счетчик байтов очереди и регистры данных (но не включая поле контрольной суммы).

Счетчик байтов очереди содержит количество регистров данных в очереди (не включая счетчик).

Если счетчик очереди содержит число больше 31, то в ответе возвращается код ошибки 03 (Недопустимая величина данных).

Ниже показан пример нормального ответа на запрос представленный выше:

Ответ

Имя поля		Пример (Hex)
Адрес подчиненного		11
Функция	18	
Счетчик байт ст.		00
Счетчик байт мл.	08	
Счетчик FIFO ст.		00
Счетчик FIFO мл.	03	
Регистр данных FIFO 1 ст.	01	
Регистр данных FIFO 1 мл.	B8	
Регистр данных FIFO 2 ст.	12	
Регистр данных FIFO 2 мл.	84	
Регистр данных FIFO 3 ст.	13	
Регистр данных FIFO 3 мл.	22	
Контрольная сумма		--

ФУНКЦИЯ ДИАГНОСТИКИ

Функция 08 - Диагностическая

ОПИСАНИЕ

Функция 08 обеспечивает серию тестов для проверки системы коммуникации между главным и подчиненным, или проверку на различные внутренние ошибки в подчиненном. Широкое вещание не поддерживается.

Функция использует два байта кода подфункции в запросе для определения типа теста который необходимо провести. Подчиненный возвращает оба кода функции и подфункции в нормальном ответе.

Большинство диагностических запросов используют два байта поля данных для посылки диагностических данных или контрольной информации подчиненному. Некоторые результаты диагностики могут возвращаться подчиненным в поле данных нормального ответа.

ЭФФЕКТ ДИАГНОСТИКИ НА ПОДЧИНЕННОМ

В общем, работа диагностической функции в подчиненном устройстве не влияет на работу пользовательской программы. Пользовательская логика, такая как дискретные выходы и регистры, не доступна для диагностики. Определенные функции могут выборочно сбросить счетчик ошибок в подчиненном.

КАК ОРГАНИЗОВАНА ИНФОРМАЦИЯ В ДАННОМ РУКОВОДСТВЕ

Примеры диагностических запросов и ответов показаны на последующих страницах. В них показано размещение кода функции, кода подфункции, и поля данных внутри сообщения.

Список кодов подфункций поддерживаемых различными контроллерами приведен ниже после примера ответа на запрос.

ЗАПРОС

Это пример запроса подчиненному устройству на возврат данных переданных в запросе. Здесь используется код подфункции 0.

Запрос

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	08
Подфункция ст.	00
Подфункция мл.	00
Данные ст.	A5
Данные мл.	37
Контрольная сумма	--

ОТВЕТ

Нормальный ответ возвращает те же данные.

Имя поля	Пример (Hex)
Адрес подчиненного	11
Функция	08
Подфункция ст.	00
Подфункция мл.	00
Данные ст.	A5
Данные мл.	37
Контрольная сумма	--

ДИАГНОСТИЧЕСКИЕ КОДЫ ПОДДЕРЖИВАЕМЫЕ КОНТРОЛЛЕРАМИ

Код	Имя	383	484	584	884	M84	984	
00	Return Query Data	Y	Y	Y	Y	Y	Y	
01	Restart Comm Option		Y	Y	Y	Y	Y	Y
02	Return Diagnostic Register		Y	Y	Y	Y	Y	Y
03	Change ASCII Input Delimiter		Y	Y	Y	N		Y
04	Force Listen Only Mode		Y	Y	Y	Y	Y	Y
05-09	Зарезервированы							

10	Clear Ctrs and Diagnostic Reg.		Y	Y	(1)	N	N	(1)
11	Return Bus Message Count	Y	Y	Y	N	N	Y	
12	Return Bus Comm. Error Count		Y	Y	Y	N	N	Y
13	Return Bus Exeption Error Count		Y	Y	Y	N	N	Y
14	Return Slave Message Count	Y	Y	Y	N	N	Y	
15	Return Slave No Response Cnt		Y	Y	Y	N	N	Y
16	Return Slave NAK Count		Y	Y	Y	N	N	Y
17	Return Slave Busy Count		Y	Y	Y	N	N	Y
18	Return Bus Char. Overrun Cnt		Y	Y	Y	N	N	Y
19	Return Overrun Error Count	N	N	N	Y	N	N	
20	Clear Overrun Counter and Flag		N	N	N	Y	N	N
21	Get/Clear Modbus Plus Statistic		N	N	N	N	N	Y

22 и далее Зарезервированы

Примечание:

(1) Только очистка счетчиков.

ДИАГНОСТИЧЕСКИЕ ПОДФУНКЦИИ

00 Return Query Data

Данные расположенные в поле данных запроса должны быть возвращены в ответе. Ответ должен быть идентичен запросу.

Подфункция	Поле данных(Запрос)	Поле данных(Ответ)
00 00	Любое значение	Повтор данных запроса

01 Restart Communication Option

Периферийный порт подчиненного инициализируется и перезапускается, все коммуникационные счетчики очищаются. Если порт находится в режиме 'Listen Only Mode', ответ не возвращается. Если порт не находится в данном режиме, то возвращается нормальный ответ. Это делается перед рестартом.

Когда подчиненный принимает запрос, он делает рестарт и запускает тесты, которые проводятся при включении питания. После удачного прохождения тестов порт переходит в режим готовности.

Поле данных запроса содержит FF 00 Hex для очистки Коммуникационного Журнала Событий. Содержимое поля данных 00 00 оставляет журнал без изменений.

Подфункция	Поле данных(Запрос)	Поле данных(Ответ)
00 01	00 00	Повтор данных запроса
00 01	FF 00	Повтор данных запроса

02 Return Diagnostic Register

Содержимое регистра диагностики подчиненного возвращается в ответе.

Подфункция	Поле данных(Запрос)	Поле данных(Ответ)
00 02	00 00	Содержимое регистра диагностики

КАК ОРГАНИЗОВАНЫ ДАННЫЕ РЕГИСТРА

Назначение битов регистра диагностики показано ниже.

184/384 Регистр диагностики

Бит	Описание
0	Continue on Error
1	Run Light Filed
2	T-Bus Test Filed
3	Asynchronous Bus Test Failed
4	Force Listen Only Mode

5	Не используется
6	Не используется
7	ROM Chip 0 Test Failed
8	Continuos ROM Cheksum Test in Execution
9	ROM Chip 1 Test Failed
10	ROM Chip 2 Test Failed
11	ROM Chip 3 Test Failed
12	RAM Chip 5000-53FF Test Failed
13	RAM Chip 6000-67FF Test Failed, Even Adresess
14	RAM Chip 6000-67FF Test Failed, Odd Adresess
15	Timer Chip Test Filed

484 Регистр диагностики

Бит	Описание
0	Continue on Error
1	CPU Test or Run Light Failed
2	Parallel Port Test Failed
3	Asynchronous Bus Test Failed
4	Timer 0 Test Filed
5	Timer 1 Test Filed
6	Timer 2 Test Filed
7	ROM Chip 0000-07FF Test Failed
8	Continuous ROM Checksum Test in Execution
9	ROM Chip 0800-0FFF Test Failed
10	ROM Chip 1000-17FF Test Failed
11	ROM Chip 1800-1FFF Test Failed
12	RAM Chip 4000-40FF Test Failed
13	RAM Chip 4100-41FF Test Failed
14	RAM Chip 4200-42FF Test Failed
15	RAM Chip 4300-43FF Test Failed

584/984 Регистр диагностики

Бит	Описание
0	Illegal Configuration
1	Backup Cheksum Error in High-Speed RAM
2	Logic Checksum Error
3	Invalid Node Type
4	Invalid Traffic Cop Type
5	CPU/Solve Diagnostic Failed
6	Real Time Clock Failed
7	Watchdog Timer Failed-Scan Time exeeded 250ms
8	No End of Logic Node detected, or quantity of end of segment words (DOIO) does not match quantity of segments configured
9	Sate RAM Test Failed
10	Start of Network (SON) did not begin network
11	Bad Order of Solve Table
12	Illegal Peripherial Intervention

13	Dim Awareness Flag
14	Не используется
15	Peripherial Port Stop Executed, not an error

884 Регистр диагностики

Бит	Описание
0	Modbus IOP Overrun Errors Flag
1	Modbus Option Overrun Errors Flag
2	Modbus IOP Failed
3	Modbus Option Failed
4	Ourbus IOP Failed
5	Remote IO Failed
6	Main CPU Failed
7	Table RAM Checksum Failed
8	Scan Task exceeded its time limit-too much user logic
9	Не используется
10	Не используется
11	Не используется
12	Не используется
13	Не используется
14	Не используется
15	Не используется

03 Change ASCII Input Delimeter

Символ 'CHAR' Размещенный в поле данных запроса становится признаком конца сообщения в последующих обменах (заменяя символ LF).

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 03	CHAR 00	Повтор данных запроса

04 Force Listen Only Mode

Установка адресуемого подчиненного в режим Listen Only Mode. Это изолирует его от других устройств сети, позволяя иметь связь без прерывания с адресуемого подчиненного.

Когда подчиненный введен в данный режим, весь активный коммуникационный контроль выключается. Пока устройство находится в данном режиме, любые сообщения адресуемые подчиненному или широковещательная передача отслеживаются, но не выполняется никаких действий и ответы не возвращаются.

Только одна функция может быть выполнена - это функция Restart Communication Option (код функции 8, подфункция 1).

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 04	00 00	Ответ не возвращается

10 (0A Hex) Clear Counters and Diagnostic Register

Для контроллеров кроме 584 или 984, очищаются все счетчики и регистр диагностики. Для 584 или 984 очищаются только счетчики. Счетчики также очищаются при включении питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0A	00 00	Повтор данных запроса

11 (0B Hex) Return Bus Message Count

Поле данных ответного сообщения содержит количество сообщений обнаруженных коммуникационной системой после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0B	00 00	Повтор данных запроса

12 (0C Hex) Return Bus Communication Error Count

Поле данных ответного сообщения содержит количество ошибок контрольной суммы насчитанных после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0C	00 00	Счетчик ошибок контр. суммы

13 (0D Hex) Return Bus Exeption Error Count

Поле данных ответного сообщения содержит количество сообщений об ошибках насчитанных подчиненным после последнего рестарта, операции очистки счетчиков, или включения питания.

Сообщения об ошибках описаны в приложении А.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0D	00 00	Счетчик ошибок

14 (0E Hex) Return Slave Message Count

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному или широковещательных сообщений насчитанных подчиненным после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0E	00 00	Счетчик сообщений подчиненного

15 (0F Hex) Return Slave No Response Count

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному которые остались без ответа после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 0F	00 00	Счетчик сообщений подчиненного

16 (10 Hex) Return Slave NAK Count

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному, для которых был возвращен ответ с сообщением об ошибке типа Negative Acknowledge (NAK), после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 10	00 00	Счетчик NAK сообщений

17 (11 Hex) Return Slave Busy Count

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному, для которых был возвращен ответ с сообщением об ошибке типа Slave Device Busy, после последнего рестарта, операции очистки счетчиков, или включения питания.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 11	00 00 Счетчик ответов "	Устройство занято"

18 (12 Hex) Return Bus Character Overrun Count

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному, в которых был обнаружен пропуск символа, после последнего рестарта, операции очистки счетчиков, или включения питания. Пропуск символа может возникать в случае если данные поступают в порт слишком быстро, или в случае аппаратных сбоев.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 12	00 00	Счетчик Slave Overrun

19 (13 Hex) Return IOP Overrun Count (884)

Поле данных ответного сообщения содержит количество сообщений адресованных подчиненному, в которых был обнаружен пропуск символа (для 884 IOP), после последнего рестарта, операции очистки счетчиков, или включения питания. Пропуск символа может возникать в случае если данные поступают в порт слишком быстро, или в случае аппаратных сбоев. Эта функция специфична для 884.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 13	00 00	Счетчик Slave IOP Overrun

20 (14 Hex) Clear Overrun Counter and Flag (884)

Очищает счетчик 884-го Overrun Counter и сбрасывает флаг ошибки. Текущее значение флага располагается в бите 0 регистра диагностики (смотри подфункцию 02). Эта функция специфична для 884.

Подфункция	Поле данных (Запрос)	Поле данных (Ответ)
00 14	00 00	Повтор данных запроса

21 (15 Hex) Get/Clear Modbus Statistic

Возвращает последовательность из 54 слов (108 байтов) в поле данных ответа. Данные содержат статистику Modbus Plus процессора подчиненного.

В добавление к коду функции (08) и коду подфункции (00 15 Hex) следующие два байта в запросе специфицируют тип операции: 'Взять статистику' или 'Очистить статистику'. Статистика может бать также очищена выключением питания.

Поле типа операции:

- 00 03 специфицирует операцию 'Взять Статистику'

- 00 04 специфицирует операцию 'Очистить Статистику'

ЗАПРОС: Это последовательность полей в запросе:

Функция	Подфункция	Операция	
08	00 15	00 03	(Get Statistic)
08	00 15	00 04	(Clear Statistic)

ОТВЕТ НА ОПЕРАЦИЮ ВЗЯТЬ СТАТИСТИКУ: Это последовательность полей нормального ответа на операцию 'Взять статистику':

Функция	Подфункция	Операция	Данные(108 байт)
08	00 15	00 03	Слова 00-53

ОТВЕТ НА ОПЕРАЦИЮ ОЧИСТИТЬ СТАТИСТИКУ: Нормальный ответ на эту операцию повторяет запрос.

Функция	Подфункция	Операция
08	00 15	00 04

Сетевая статистика Modbus Plus

Слово	Биты	начение
00		Тип Узла
	0	Незнакомый тип узла
	1	Программируемый контроллер
	2	Modbus Brige
	3	Host - компьютер
	4	Brige Plus
	5	I/O узел
01	0..11	Номер версии программного обеспечения
	12..14	Зарезервировано
	15	Определяет значение слова 15
		(см. слово 15)
02		Сетевой адрес для данной станции
03		Установки MAC переменных
	0	Состояние питания
	1	Monitor offline state
	2	Duplicate offline state
	3	Состояние простоя
	4	Use token state
	5	Work response state
	6	Pass token state

	7	Solicit response state
	8	Check pass state
	9	Claim token state
	10	Claim response state
04		Peer status(LED code); определяет статус данного блока относительно сети.
	0	Monitor link operation
	32	Normal link operation
	64	Never getting token
	96	Sole station
	128	Duplicate station
05		Счетчик символов; инкрементируется каждый раз когда станция принимает символ.
06		Token rotation time in ms.
07	Мл	Data master failed during token owership bit map
	Ст	Program master failed during token owership bit map
08	Мл	Data master token owner work bit map
	Ст	Program master token owner work bit map
09	Мл	Data slave token owner work bit map
	Ст	Program slave token owner work bit map
10	Ст	Data slave/get slave command transfer request bit map
11	Мл	Program master/get master rsp transfer request bit map
	Ст	Program slave/get slave command transfer request bit map
12	Мл	Program master connect status bit map
	Ст	Program slave automatic logout request bit map
13	Мл	Pretransmit deferral error counter
	Ст	Receive buffer DMA overrun error counter
14	Мл	Repeated command received counter
	Ст	Frame size error counter
15	Если бит 15 слова 1 не установлен, содержание слова 15 следующее:	
	Мл	Receiver collision-abort error counter.
	Ст	Receiver alignment error counter
	Если бит 15 слова 1 установлен, содержание слова 15 следующее:	
	Мл	Cable A framing error
	Ст	Cable B framing error
16	Мл	Receiver CRC error counter
	Ст	Bad packet-lenght error counter
17	Мл	Bad link-address error counter
	Ст	Transmit buffer DMA-underrun error counter
18	Мл	Bad internal packet lenght error counter
	Ст	Bad MAC function error counter
19	Мл	Communication retry counter
	Ст	Communication failed error counter
20	Мл	Good receive packet error counter
	Ст	No response received error counter
21	Мл	Exeption response received error counter
	Ст	Unexpected path error counter

22	Мл	Unexpected response error counter
	Ст	Forgotten transaction error counter
23	Мл	Active station table bit map, nodes 1..8
	Ст	Active station table bit map, nodes 9..16
24	Мл	Active station table bit map, nodes 17..24
	Ст	Active station table bit map, nodes 25..32
25	Мл	Active station table bit map, nodes 33..40
	Ст	Active station table bit map, nodes 41..48
26	Мл	Active station table bit map, nodes 49..56
	Ст	Active station table bit map, nodes 57..64
27	Мл	Token station table bit map, nodes 1..8
	Ст	Token station table bit map, nodes 9..16
28	Мл	Token station table bit map, nodes 27..24
	Ст	Token station table bit map, nodes 25..32
29	Мл	Token station table bit map, nodes 33..40
	Ст	Token station table bit map, nodes 41..48
30	Мл	Token station table bit map, nodes
	Ст	Token station table bit map, nodes 49..56
31	Мл	Global data present table bit map, nodes 1..8
	Ст	Global data present table bit map, nodes 9..16
32	Мл	Global data present table bit map, nodes 17..24
	Ст	Global data present table bit map, nodes 25..32
33	Мл	Global data present table bit map, nodes 33..40
	Ст	Global data present table bit map, nodes 41..48
34	Мл	Global data present table bit map, nodes 49..56
	Ст	Global data present table bit map, nodes 57..64
35	Мл	Receive buffer in use bit map, buffer 1..8
	Ст	Receive buffer in use bit map, buffer 9..16
36	Мл	Receive buffer in use bit map, buffer 17..24
	Ст	Receive buffer in use bit map, buffer 25..32
37	Мл	Receive buffer in use bit map, buffer 33..40
	Ст	Station managment command processed initiation counter
38	Мл	Data master output path 1 command initiation counter
	Ст	Data master output path 2 command initiation counter
39	Мл	Data master output path 3 command initiation counter
	Ст	Data master output path 4 command initiation counter
40	Мл	Data master output path 5 command initiation counter
	Ст	Data master output path 6 command initiation counter
41	Мл	Data master output path 7 command initiation counter
	Ст	Data master output path 8 command initiation counter
42	Мл	Data slave input path 41 command processed counter
	Ст	Data slave input path 42 command processed counter
43	Мл	Data slave input path 43 command processed counter
	Ст	Data slave input path 44 command processed counter
44	Мл	Data slave input path 45 command processed counter

	Ст	Data slave input path 46command processed counter
45	Мл	Data slave input path 47 command processed counter
	Ст	Data slave input path 48 command processed counter
46	Мл	Program master output path 81 command initiation counter
	Ст	Program master output path 82 command initiation counter
47	Мл	Program master output path 83 command initiation counter
	Ст	Program master output path 84 command initiation counter
48	Мл	Program master output path 85 command initiation counter
	Ст	Program master output path 86 command initiation counter
49	Мл	Program master output path 87 command initiation counter
	Ст	Program master output path 88 command initiation counter
50	Мл	Program slave input path C1 command processed counter
	Ст	Program slave input path C2 command processed counter
51	Мл	Program slave input path C3 command processed counter
	Ст	Program slave input path C4 command processed counter
52	Мл	Program slave input path C5 command processed counter
	Ст	Program slave input path C6 command processed counter
53	Мл	Program slave input path C7 command processed counter
	Ст	Program slave input path C8 command processed counter

ПРИЛОЖЕНИЕ А

СООБЩЕНИЯ ОБ ОШИБКАХ

Одна из четырех ситуаций может иметь место при запросе главного к подчиненному:

- Если подчиненное устройство приняло запрос без коммуникационных ошибок, и может нормально распознать запрос, оно возвращает нормальный ответ.

- Если подчиненное устройство не приняло запрос, ответ не возвращается. Главный ожидает ответа на запрос в течении определенного таймаута.

- Если подчиненный принял запрос, но обнаружил коммуникационную ошибку(паритет, ошибка контрольной суммы), то ответ не возвращается. Главный ожидает ответа на запрос в течении определенного таймаута.

- Если подчиненный принял запрос без коммуникационной ошибки, но не может выполнить затребованную функцию(например, чтение несуществующих выходов или регистров), подчиненный возвращает сообщение об ошибке и ее причинах.

Сообщение об ошибке имеет два поля которые отличаются от полей нормального ответа:

ПОЛЕ КОДА ФУНКЦИИ: В нормальном ответе, подчиненный повторяет код функции содержащийся в поле кода функции запроса. Во всех кодах функций старший значащий бит установлен в 0. При возврате сообщения об ошибке подчиненный устанавливает этот бит в 1.

При установленному старшему биту в коде функции главный распознает сообщение об ошибке, и может проанализировать поле данных сообщения.

ПОЛЕ ДАННЫХ: В нормальном ответе, подчиненный может может возвращать данные или статистику в поле данных(любую информацию которая затребована в запросе). В сообщении об ошибке, подчиненный возвращает код ошибки в поле данных.

Ниже показан пример запроса главного и сообщения об ошибке подчиненного:

ЗАПРОС

Имя поля	Пример (Hex)
Адрес подчиненного	0A
Функция	01
Начальный адрес ст.	04
Начальный адрес мл.	A1

Кол-во входов ст.	00	
Кол-во входов мл.	01	
Контрольная сумма (LRC)	4F	
СООБЩЕНИЕ ОБ ОШИБКЕ		
Адрес подчиненного		0A
Функция	81	
Код ошибки		02
Контрольная сумма (LRC)	73	

Вданном примере главный адресует подчиненное устройство 10. Код функции (01) - Read Coil Status. В запросе требуется прочитать выход с адресом 1245.

Если указанный выход не существует подчиненный возвращает сообщение об ошибке с кодом ошибки (02). Этот код специфицирует несуществующий адрес данных в подчиненном. Например если подчиненный это 984-385 с 512 выходами, то этот код ошибки будет возвращаться при обращении к несуществующим выходам.

Список кодов ошибок представлен ниже.

Код	Название	Описание
01	ILLEGAL FUNCTION	Принятый код функции не может быть обработан на подчиненном.
02	ILLEGAL DATA ADDRESS	Адрес данных указанный в запросе не доступен данному подчиненному.
03	ILLEGAL DATA VALUE	Величина содержащаяся в поле данных запроса является не допустимой величиной для подчиненного.
04	SLAVE DEVICE FAILURE	Невосстанавливаемая ошибка имела место пока подчиненный пытался выполнить затребованное действие.
05	ACKNOWLEDGE	Подчиненный принял запрос и обрабатывает его, но это требует много времени. Этот ответ предохраняет главного от генерации ошибки таймаута. Главный может выдать команду Poll Program Complete для обнаружения завершения обработки команды.
06	SLAVE DEVICE BUSY	Подчиненный занят обработкой команды. Главный должен повторить сообщение позже, когда подчиненный освободится.
07	NEGATIVE ACKNOWLEDGE	Подчиненный не может выполнить программную функцию, принятую в запросе. Этот код возвращается для неудачного программного запроса, использующего функции с номерами 13 или 14. Главный должен запросить диагностическую информацию или информацию обошибках с подчиненного.
08	MEMORY PARITY ERROR	Подчиненный пытается читать расширенную память, но обнаружил ошибку паритета. Главный может повторить запрос, но обычно в таких случаях требуется ремонт.

ПРИЛОЖЕНИЕ В

ПРИМЕЧАНИЯ К ПРИМЕНЕНИЮ

МАКСИМАЛЬНЫЕ ПАРАМЕТРЫ ДЛЯ ЗАПРОСА/ОТВЕТА

В списке, представленном в данной секции, показано максимально возможное количество данных для каждого контроллера, которые могут быть использованы в запросе главного устройства или в ответе подчиненного.

184/384 Функция	Описание	Запрос	Ответ
1	Read Coil Status	800 Выходов	800 Выходов
2	Read Input Status	800 Входов	800 Входов
3	Read Holding Registers	100 Регистров	100 Регистров
4	Read Input Registers	100 Регистров	100 Регистров
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов
8	Diagnostics	Не доступна	Не доступна
9	Program 484	Не поддерживается	Не поддерживается

10	Poll 484	Не поддерживается	Не поддерживается
11	Fetch. Comm. Event Ctr.	Не доступна	Не доступна
12	Fetch. Comm. Event Log	Не доступна	70 байтов данных
13	Program Controller	32 байта данных	32 байта данных
14	Poll Controller	Не доступна	32 байта данных
15	Force Multiply Coils	800 Выходов	800 Выходов
16	Preset Multiply Regs	100 Регистров	100 Регистров
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	Не поддерживается	Не поддерживается
19	Preset Comm. Link	Не поддерживается	Не поддерживается
20	Read General Reference	Не поддерживается	Не поддерживается
21	Write General Reference	Не поддерживается	Не поддерживается
484 Функция	Описание	Запрос	Ответ
1	Read Coil Status	512 Выходов	512 Выходов
2	Read Input Status	512 Входов	512 Входов
3	Read Holding Registers	254 Регистра	254 Регистра
4	Read Input Registers	32 Регистра	32 Регистра
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов
8	Diagnostics	Не доступна	Не доступна
9	Program 484	16 байтов данных	16 байтов данных
10	Poll 484	Не доступна	16 байтов данных
11	Fetch. Comm. Event Ctr.	Не поддерживается	Не поддерживается
12	Fetch. Comm. Event Log	Не поддерживается	Не поддерживается
13	Program Controller	Не поддерживается	Не поддерживается
14	Poll Controller	Не поддерживается	Не поддерживается
15	Force Multiply Coils	800 Выходов	800 Выходов
16	Preset Multiply Regs	60 Регистров	60 Регистров
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	Не поддерживается	Не поддерживается
19	Preset Comm. Link	Не поддерживается	Не поддерживается
20	Read General Reference	Не поддерживается	Не поддерживается
21	Write General Reference	Не поддерживается	Не поддерживается
584 Функция	Описание	Запрос	Ответ
1	Read Coil Status	2000 Выходов	2000 Выходов
2	Read Input Status	2000 Входов	2000 Входов
3	Read Holding Registers	125 Регистров	125 Регистров
4	Read Input Registers	125 Регистров	125 Регистров
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов
8	Diagnostics	Не доступна	Не доступна
9	Program 484	Не поддерживается	Не поддерживается
10	Poll 484	Не поддерживается	Не поддерживается

11	Fetch. Comm. Event Ctr.	Не доступна	Не доступна
12	Fetch. Comm. Event Log	Не доступна	70 байтов данных
13	Program Controller	33 байта данных	33 байта данных
14	Poll Controller	Не доступна	33 байта данных
15	Force Multiply Coils	800 Выходов	800 Выходов
16	Preset Multiply Regs	100 Регистров	100 Регистров
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	Не поддерживается	Не поддерживается
19	Preset Comm. Link	Не поддерживается	Не поддерживается
20	Read General Reference	(1)	(1)
21	Write General Reference	(1)	(1)

Примечание:

(1) Максимальная длина сообщения не должна превышать 256.

884 Функция	Описание	Запрос	Ответ
1	Read Coil Status	2000 Выходов	2000 Выходов
2	Read Input Status	2000 Входов	2000 Входов
3	Read Holding Registers	125 Регистров	125 Регистров
4	Read Input Registers	125 Регистров	125 Регистров
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов
8	Diagnostics	Не доступна	Не доступна
9	Program 484	Не поддерживается	Не поддерживается
10	Poll 484	Не поддерживается	Не поддерживается
11	Fetch. Comm. Event Ctr.	Не поддерживается	Не поддерживается
12	Fetch. Comm. Event Log	Не поддерживается	Не поддерживается
13	Program Controller	Не поддерживается	Не поддерживается
14	Poll Controller	Не поддерживается	Не поддерживается
15	Force Multiply Coils	800 Выходов	800 Выходов
16	Preset Multiply Regs	100 Регистров	100 Регистров
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	(1)	(1)
19	Preset Comm. Link	(1)	(1)
20	Read General Reference	Не поддерживается	Не поддерживается
21	Write General Reference	Не поддерживается	Не поддерживается

Примечание:

(1) Максимальная длина сообщения не должна превышать 256.

M84 Функция	Описание	Запрос	Ответ
1	Read Coil Status	64 Выхода	64 Выхода
2	Read Input Status	64 Входа	64 Входа
3	Read Holding Registers	32 Регистра	32 Регистра
4	Read Input Registers	4 Регистров	4 Регистров
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов

8	Diagnostics	Не доступна	Не доступна
9	Program 484	Не поддерживается	Не поддерживается
10	Poll 484	Не поддерживается	Не поддерживается
11	Fetch. Comm. Event Ctr.	Не поддерживается	Не поддерживается
12	Fetch. Comm. Event Log	Не поддерживается	Не поддерживается
13	Program Controller	Не поддерживается	Не поддерживается
14	Poll Controller	Не поддерживается	Не поддерживается
15	Force Multiply Coils	64 Выхода	64 Выхода
16	Preset Multiply Regs	32 Регистра	64 Регистра
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	(1)	(1)
19	Preset Comm. Link	Не доступна	Не доступна
20	Read General Reference	Не поддерживается	Не поддерживается
21	Write General Reference	Не поддерживается	Не поддерживается

Примечание:

(1) Максимальная длина сообщения не должна превышать 256.

984 Функция	Описание	Запрос	Ответ
1	Read Coil Status	2000 Выходов	2000 Выходов
2	Read Input Status	2000 Входов	2000 Входов
3	Read Holding Registers	125 Регистров	125 Регистров
4	Read Input Registers	125 Регистров	125 Регистров
5	Force Single Coil	1 Выход	1 Выход
6	Preset Single Register	1 Регистр	1 Регистр
7	Read Exeption Status	Не доступна	8 Выходов
8	Diagnostics	Не доступна	Не доступна
9	Program 484	Не поддерживается	Не поддерживается
10	Poll 484	Не поддерживается	Не поддерживается
11	Fetch. Comm. Event Ctr.	Не доступна	Не доступна
12	Fetch. Comm. Event Log	Не доступна	70 байтов данных
13	Program Controller	33 байта данных	33 байта данных
14	Poll Controller	Не доступна	33 байта данных
15	Force Multiply Coils	800 Выходов	800 Выходов
16	Preset Multiply Regs	100 Регистров	100 Регистров
17	Report Slave ID	Не доступна	Не доступна
18	Program 884/M84	Не поддерживается	Не поддерживается
19	Preset Comm. Link	Не поддерживается	Не поддерживается
20	Read General Reference	(1)	(1)
21	Write General Reference	(1)	(1)

Примечание:

(1) Максимальная длина сообщения не должна превышать 256.

ОЦЕНКА ВРЕМЕНИ ПОСЛЕДОВАТЕЛЬНОЙ ПЕРЕДАЧИ

ПОСЛЕДОВАТЕЛЬНОСТЬ ПЕРЕДАЧИ

Ниже приведена действий при последовательной передаче Modbus. Символы указанные в скобках ссылаются на примечания, перечисленные после списка действий.

1. Главное устройство Modbus Формирует сообщение.

2. Проверяются статусы RTS и CTS модема главного. (A)

3. Сообщение запроса передается подчиненному.(B)

4. Подчиненный принимает сообщение запроса. (C)(D)

5. Подчиненный вычисляет контрольную сумму. (E)

6. Проверяются статусы RTS и CTS модема подчиненного. (F)

7. Ответное сообщение передается главному. (B)

8. Главный производит необходимые действия над принятым ответом и данными.

ПРИМЕЧАНИЯ

(A) Если выводы RTS и CTS соединены вместе, то это время незначительно. Для модема J478, это время около 5 мс.

(B) Следующая формула оценивает время передачи:

$$\text{Время} = \frac{1000 \times (\text{счетчик символов}) \times (\text{битов в символе})}{\text{Скорость передачи}}$$

(C) Modbus сообщение обрабатывается в конце скана контроллера. В худшем случае задержка будет равна времени одного скана. В среднем задержка составляет 0.5 скана.

Время, выделяемое для обслуживания портов Modbus в конце каждого скана контроллера(перед началом нового), зависит от модели контроллера. Время для каждой модели описано ниже.Для 484 контроллера это время составляет приблизительно 1.5 мс.

Для 584 и 984 контроллера время составляет приблизительно 1.5 мс. Порты обслуживаются последовательно, начиная с порта 1.

Для 184/384 контроллеров время зависит от количества принимаемых данных. Это время варьируется от минимального 0.5 мс до максимального около 6.0 мс (для 100 регистров), или 7.0 мс (для 800 выходов).Если с контроллером используется программная панель, то Modbus порт отключается.

(D) Modbus Функции с 1 по 4, 15 и 16 позволяет главному устройству запросить больше данных чем то количество, которое может обработать контроллер в течении времени выделяемого на обслуживание портов. Если подчиненный не успел обработать все данные, он может буферизировать их и обработать позже.

Ниже приведены данные, которые могут быть обработаны за один цикл обслуживания Modbus порта.

	Дискреты	Регистры
Micro 84	16	4
184/384	800	100
484	32	16
584	64	32
984A/B/X	64	32
984-X8X	1000	125

Примечание: '984-X8X' ссылается на slot-mount модели(984-385, -685 и т.д.).

Для 884, время обработки множественных данных следующее: Чтение 700 выходов: 14 сканов Установка единич. выхода: 3 скана Чтение 256 входов: 7 сканов Установка регистра: 3 скана Чтение 125 вых. рег-ов: 5 сканов Установка 768 выходов: 18 сканов Чтение 125 вх. рег-ов: 8 сканов Установка 100 рег.: 10 сканов

(E) Вычисление контрольной суммы LRC - около 1 мс. Вычисление контрольной суммы CRC - около 0.3 мс для каждых 8-ми бит данных возвращаемых в ответе.

ПРИМЕЧАНИЯ ДЛЯ 584 И 984A/B/X

Это примечание касается только 584 и 984A/B/X контроллеров.

СКОРОСТЬ ПЕРЕДАЧИ: Когда используются оба Modbus порта 1 и 2, максимальная доступная скорость передачи - 19 200 бод.

БЛОКИРОВКА ПОРТА: Когда вы используете ASCII режим, избегайте посылки сообщений с 'нулевой длиной данных', или сообщений без адреса устройства. Например, это неправильное сообщение:

: CR LF

При использовании такого рода сообщений может иметь место случайная блокировка порта.

ПРИЗНАК КОНЦА ASCII СООБЩЕНИЯ: Нормальное ASCII сообщение должно заканчиваться парой CRLF. При использовании контроллеров 584 и 984A/B/X, ASCII сообщение может заканчиваться после поля контрольной суммы LRC (без установки символов CRLF), если после поля LRC имеет место интервал по крайней мере 1 сек. Если это случилось, контроллер считает что сообщение закончилось нормально.

ПРИЛОЖЕНИЕ С

ГЕНЕРАЦИЯ LRC/CRC

Генерация LRC

Longitudinal Redundancy Check(LRC) это один байт. LRC вычисляется передающим устройством и добавляется к концу сообщения. Принимающее устройство также вычисляет LRC в процессе приема и сравнивает вычисленную величину с полем контрольной суммы пришедшего сообщения. Если суммы не совпали - то имеет место ошибка.

LRC вычисляется сложением последовательности байтов сообщения, отбрасывая все переносы, и затем двойным дополнением результата. LRC - это 8-ми битовое поле, где каждое новое прибавление символа, приводящее к результату более чем 255, приводит к простому перескакиванию через 0. Так как это поле не является 9-ти битовым, перенос отбрасывается автоматически.

Алгоритм генерации LRC:

1. Сложить все байты сообщения, исключая стартовый символ ':' и конечные CRLF, складывая их так, чтобы перенос отбрасывался.

2. Отнять получившееся значение от числа FF(Hex) - это является первым дополнением.

3. Прибавить к получившемуся значению 1 - это второе дополнение.

РАЗМЕЩЕНИЕ LRC В СООБЩЕНИИ

Когда 8-ми битовое поле LRC (2 ASCII символа) передается в сообщении, то старший символ будет передан первым, а за ним - младший. Например, если значение LRC 61 hex(0110 0001):

':'	Адрес	Функция	Сч-к байт	Байт	Байт	Байт	Байт	LRC Ст.	LRC Мл.	CR	LF	
								6	1			

Пример функции на языке С реализующей генерацию LRC приведен ниже. Функция принимает два аргумента:

```
unsigned char *auchMsg; Указатель на буфер данных
unsigned short usDataLen; Количество байт в буфере
Функция возвращает LRC как тип unsigned char.
ПРИМЕР
static unsigned char LRC(auchMsg, usDataLen)
unsigned char *auchMsg;/* Сообщение над которым */
                       /* вычисляется LRC */
unsigned char usDataLen; /* Количество байт в сообщении */
{
unsigned char uchLRC=0; /* Инициализация LRC */
while(usDataLen)
uchLRC+=*auchMsg++;
return((unsigned char)(-((char uchLRC)));
}
```

Генерация CRC

CRC это 16-ти разрядная величина т.е. два байта. CRC вычисляется передающим устройством и добавляется к сообщению.Принимающее устройство также вычисляет CRC в процессе приема и сравнивает вычисленную величину с полем контрольной суммы пришедшего сообщения. Если суммы не совпали - то имеет место ошибка.

16-ти битовый регистр CRC предварительно загружается числом FF hex. Процесс начинается с добавления байтов сообщения к текущему содержимому регистра. Для генерации CRC используются только 8 бит данных. Старт и стоп биты, бит паритета, если он используется, не учитываются в CRC.

В процессе генерации CRC, каждый 8-ми битовый символ складывается по ИСКЛЮЧАЮЩЕМУ ИЛИ с содержимым регистра. Результата сдвигается в направлении младшего бита, с заполнением 0 старшего бита. Младший бит извлекается и проверяется. Если младший бит равен 1, то содержимое регистра складывается с определенной ранее, фиксированной величиной, по ИСКЛЮЧАЮЩЕМУ ИЛИ. Если младший бит равен 0, то ИСКЛЮЧАЮЩЕЕ ИЛИ не делается.

Этот процесс повторяется пока не будет сделано 8 сдвигов. После последнего (восьмого) сдвига, следующий байт складывается с содержимым регистра и процесс повторяется снова. Финальное содержание регистра, после обработки всех байтов сообщения и есть контрольная сумма CRC.

Алгоритм генерации CRC:

1. 16-ти битовый регистр загружается числом FF hex (все 1), и используется далее как регистр CRC.

2. Первый байт сообщения складывается по ИСКЛЮЧАЮЩЕМУ ИЛИ с содержимым регистра CRC. Результат помещается в регистр CRC.

3. Регистр CRC сдвигается вправо(в направлении младшего бита) на 1 бит, старший бит заполняется 0.

4. (Если младший бит 0): Повторяется шаг 3 (сдвиг)

(Если младший бит 1): Делается операция ИСКЛЮЧАЮЩЕЕ ИЛИ регистра CRC и полиномиального числа A001 hex.

5. Шаги 3 и 4 повторяются восемь раз.

6. Повторяются шаги со 2 по 5 для следующего сообщения. Это повторяется до тех пор пока все байты сообщения не будут обработаны.

7. Финальное содержание регистра CRC и есть контрольная сумма.

РАЗМЕЩЕНИЕ CRC В СООБЩЕНИИ

При передаче 16 бит контрольной суммы CRC в сообщении, сначала передается младший байт, затем старший. Например, если CRC равна 1241 hex :

Адрес	Функция	Счетчикк байт	Данные	Данные	Данные	Данные	CRC Ст.	CRC Мл.
							41	12

ПРИМЕР

Пример функции на языке С реализующей генерацию CRC приведен ниже. Все возможные величины CRC загружены в два массива. Один массив содержит все 256 возможных комбинаций CRC для старшего байта поля CRC, другой массив содержит данные для младшего байта. Идексация CRC в этом случая обеспечивает быстрое выполнение вычислений новой величины CRC для каждого нового байта из буфера сообщения.

Функция принимает два аргумента:

```
unsigned char *puchMsg; /* Указатель на буфер */

unsigned short usDataLen; /* Количество байтов в буфере */

Функция возвращает CRC как тип unsigned short.

static 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,
0x22,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
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,
0x40,0x01,0xC0,0x80,0x41,0x00,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
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

 }
```

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