ООО «НПП Электромеханика»

Модуль аналогового ввода МС1210. Описание протокола обмена данными ModBus RTU

разработано на основе Modbus Application Protocol Specification V1.1b

Modbus over Serial Line Specification and Implementation Guide V1.02

http://www.Modbus-IDA.org

13 апреля 2012 г.

ОГЛАВЛЕНИЕ

ОБЩИЕ ПРИНЦИПЫ ПЕРЕДАЧИ ДАННЫХ В ПРОТОКОЛЕ MODBUS RTU

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 подчиненное 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 подчинённого
Адрес	Адрес
Код функции	Код функции
8 битные байты данных	8 битные 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 4 подразумевает запрос на чтение содержимого регистров подчиненного.

Ответ: Если подчиненный дает 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.

СОДЕРЖАНИЕ СООБЩЕНИЯ MODBUS

Режим ASCII не используется.

RTU ФРЕЙМ

В RTU режиме сообщение начинается с интервала тишины продолжительностью более 3.5 символа при данной скорости передачи в сети. Первым байтом 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. Интервал тишины продолжительностью более 1.5 символа во время передачи фрейма, воспринимается устройством как ошибка. Типичный фрейм сообщения показан ниже.

Старт	Адрес	Функция	Данные	CRC	Конец
T1-T2-T3-T4	8 бит	8 бит	N x 8 бит	16 бит	T1-T2-T3-T4

СОДЕРЖАНИЕ АДРЕСНОГО ПОЛЯ

Адресное поле фрейма содержит 8 бит. 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.

СОДЕРЖАНИЕ ПОЛЯ ФУНКЦИИ

Поле функции фрейма содержит 8 бит. Диапазон числа 1–255. Имеющийся набор функций описан в разделе «Функции контроля и обработки данных».

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 подчиненному прочитать группу регистров имеет следующий код функции:

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

1000 0011 (83h) 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 04h), поле данных содержит адрес начального регистра и количество регистров. Если главный хочет записать группу регистров (код функции 10h), 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

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.

ФОРМАТ КАЖДОГО БАЙТА В RTU-РЕЖИМЕ

старт	1	2	3	4	5	6	7	8	паритет	стоп

Система кодировки: 8-ми битовая двоичная, шестнадцатеричная
0-9, A-F
Две шестнадцатеричные цифры содержатся в каждом 8-ми битовом байте сообщения.

Назначение битов:
1 стартовый бит
8 бит данных, младшим значащим разрядом вперед
1 бит паритета с контролем четности
1 стоповый бит

МЕТОДЫ КОНТРОЛЯ ОШИБОК

Контроль паритета и контрольная сумма. Главное и подчинённое устройства проверяют каждый байт и всё сообщение в процессе приема. Если подчиненный обнаружил ошибку передачи, то он не формирует ответ главному.

КОНТРОЛЬ ПАРИТЕТА

Устройства используют чётный (Even) паритет. Например, 8 бит RTU фрейма содержат биты 1100 0101. Общее количество единиц – 4. Бит паритета будет равен 0, так чтобы общее количество единиц вместе с битом паритета было чётным числом.

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 0xFFFF. 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 МОДУЛЯ АНАЛОГОВОГО ВВОДА МС1210

СПИСОК ФУНКЦИЙ МС1210

Код функции	Описание
04h	Чтение регистров
06h	Запись в единичный регистр
07h	Чтение регистра статуса
08h	Чтение регистров диагностики
10h	Запись нескольких регистров
2Bh/0Eh	Чтение идентификации устройства

04H ЧТЕНИЕ РЕГИСТРОВ

Функция «Чтение регистров» используется для считывания регистров преобразователя, содержащих значения параметров, указанных в таблице 3.

Запрос содержит номер начального регистра и количество регистров для чтения.

Данные регистров в ответе передаются как два байта на регистр. Для каждого регистра первый байт содержит старшие биты, второй байт содержит младшие биты. Ответ дается незамедлительно, после выдержки 3.5 интервала от последнего байта запроса.

Запрос

Код функции	1 байт	0x04
Стартовый адрес	2 байта	0x0000 – 0xFFFF
Количество регистров	2 байта	1 – 125 (0x7D)

Ответ

Код функции	1 байт	0x04
Количество байт	1 байт	2 x N*
Значения регистров	N* x 2 байта	

*N = количество регистров

Сообщение об ошибке

Код ошибки	1 байт	0x84
Причина ошибки	1 байт	02 или 03 или 04

Пример

Запрос для чтения 1 регистра «Мгновенное действующее значение напряжения фазы А (U1a)»

Запрос		Ответ	
Поле	Hex	Поле	Hex
Адрес	01	Адрес	01
Код функции	04	Код функции	04
Начальный адрес ст.	02	Кол-во байт	02
Начальный адрес мл.	00	Значение регистра ст.	00
Кол-во регистров ст.	00	Значение регистра мл.	02
Кол-во регистров мл.	01	CRC ст.	38

Запрос		Ответ	
Поле	Hex	Поле	Hex
CRC ст.	30	CRC мл.	F1
CRC мл.	72		

Содержимое регистра 0x0200 имеет значение 0x0002. Значит, текущий диапазон ±5В.

06H ЗАПИСЬ В ЕДИНИЧНЫЙ РЕГИСТР

Эта функция используется для записи одного регистра в удалённом устройстве. Запрос содержит адрес регистра и значение, которое должно быть записано. Нормальный ответ – эхо запроса - возвращается после того как содержимое регистра в удалённом контроллере перезаписано.

Запрос

Код функции	1 байт	0x06
Адрес регистра	2 байта	0xXXXX*
Значение регистра	2 байта	0xXXXX*

Ответ

Код функции	1 байт	0x06
Адрес регистра	2 байта	0xXXXX*
Значение регистра	2 байта	0xXXXX*

Сообщение об ошибке

Код ошибки	1 байт	0x86
Причина ошибки	1 байт	02 или 03 или 04

*Допустимые адреса и значения регистров для записи

Адрес	Действие	Возможные значения
0x0100 – 0x01FF	Карта регистров	0x0200 – 0x02FF
0x1100	Конфигурация устройства	SnConf
0x1101	Внешняя калибровка смещения нуля[1]	0x000F
0x1102	Внешняя калибровка усиления[1]	0x000F
0x1103	Внутренняя калибровка[1]	0x000F
0x1104	Отменить внешнюю калибровку[1]	0x000F
0x1105	Регистр НЧ фильтра	13 – 255
0x1106 – 0x1209	Таблица термопары	TabT
0x7FD0 – 0x7FD1	Пароль на калибровку	0x00000000 – 0xFFFFFFFF
0x8000	Фиксация данных	0x000F
0x8001	Сброс регистра состояния	0x000F
0x8010	Разрешение установки параметра, сбрасывается после	0x0000 (адрес) 0x0001 (скорость)

Адрес	Действие	Возможные значения
	записи любого параметра или чтения данных[2]	0x0003 (пароль) 0x0004 (протокол) 0x0006 (калибровка)
0x8011	Установка адреса устройства	0x0001 – 0x00F7
0x8012	Установка скорости текущего канала	0x0001 (19200 бит/с) 0x0002 (9600 бит/с) 0x0003 (4800 бит/с) 0x0004 (2400 бит/с) 0x0005 (1200 бит/с) 0x0011 (38400 бит/с) 0x0012 (57600 бит/с) 0x0013 (115200 бит/с)
0x8013	Установка протокола интерфейса текущего канала	0x0001 (FT3) 0x0002 (MODBUS)

[1] Защищено паролем. См. «Выполнение калибровки».

[2] Непосредственно перед изменением параметра (например, скорости) необходимо предварительно записать константу (например, 1 для скорости) в регистр 0x8010.

Пример. Сброс регистра состояния

Запрос		Ответ	
Поле	Hex	Поле	Hex
Адрес	01	Адрес	01
Код функции	06	Код функции	06
Адрес регистра ст.	80	Адрес регистра ст.	80
Адрес регистра мл.	01	Адрес регистра мл.	01
Значение регистра ст.	00	Значение регистра ст.	00
Значение регистра мл.	0F	Значение регистра мл.	0F
CRC ст.	B1	CRC ст.	B1
CRC мл.	CE	CRC мл.	CE

07H ЧТЕНИЕ РЕГИСТРА СТАТУСА

Эта функция используется для чтения восьми статусных битов в удалённом устройстве. Нормальный ответ содержит 1 байт статусного регистра.

Запрос

Код функции	1 байт	0x07

Ответ

Код функции	1 байт	0x07
Данные регистра	1 байт	0x00 – 0xFF

Сообщение об ошибке

Код ошибки	1 байт	0x87
Причина ошибки	1 байт	04

Байт регистра статуса равен младшему байту в структуре MS1210STATE

Пример запроса на чтение статусного регистра

Запрос		Ответ	
Поле	Hex	Поле	Hex
Адрес	01	Адрес	01

Запрос		Ответ	
Поле	Hex	Поле	Hex
Код функции	07	Код функции	07
CRC ст.	41	Данные регистра	C1
CRC мл.	E2	CRC ст.	E3
		CRC мл.	A0

08H ЧТЕНИЕ РЕГИСТРОВ ДИАГНОСТИКИ

Эта функция обеспечивает проверку коммуникации между главным и подчинённым и выявляет внутренние ошибки в подчинённом. Широкое вещание не поддерживается.

Функция использует два байта кода подфункции в запросе для определения типа диагностики. Подчинённый возвращает оба кода функции и подфункции в ответе.

Запрос

Код функции	1 байт	0x08
Код подфункции	2 байта	0x00XX*
Поле данных	2 байта	

Ответ

Код функции	1 байт	0x08
Код подфункции	2 байта	0x00XX*
Поле данных	2 байта	

*см. в таблице диагностических подфункций функции 08

Сообщение об ошибке

Код ошибки	1 байт	0x88
Причина ошибки	1 байт	01 или 03 или 04

Таблица диагностических подфункций функции 08

Код подфункции (Dec)	Применение
00	эхо, возвращает принятые данные
01	инициализация UART
02	возвращает регистр диагностики
04	переводит в режим прослушивания сети
10	очистка счетчиков и регистра диагностики
11	возвращает счетчик принятых сообщений
12	возвращает счетчик ошибок CRC и паритета
13	возвращает счетчик неправильных запросов
14	возвращает счетчик запросов устройства и широковещательных запросов
15	возвращает счетчик запросов без ответа
16	возвращает счетчик NAK
17	возвращает счетчик ответов «занят»
18	возвращает счетчик ошибок переполнения буфера
20	очищает счетчик и флаг ошибок переполнения буфера

00 Возвращает принятые данные. Ответ идентичен запросу.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 00	Любые 2 байта	Повтор данных запроса

01 Инициализируется UART, все коммуникационные счётчики очищаются. Если порт находится в режиме прослушивания сети, то ответ не возвращается, но порт переходит в обычный режим. Если поле данных содержит FF00h, то происходит очистка коммуникационного журнала событий.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 01	00 00	Повтор данных запроса
00 01	FF 00	Повтор данных запроса

02 Возвращает содержимое регистра диагностики.

Младший байт регистра диагностики равен байту регистра статуса устройства. Старший байт содержит дополнительные флаги состояния устройства.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 02	00 00	Содержимое регистра статуса

04 Устанавливает устройство в режим прослушивания сети. Все сообщения, адресуемые подчинённому или широковещательно, отслеживаются, но не выполняется никаких действий и ответы не возвращаются. Только одна функция может быть выполнена – рестарт UART, что вернёт устройство в нормальный режим.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 04	00 00	не возвращается

10 Очистка всех счётчиков и регистра диагностики

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0A	00 00	Эхо данных запроса

11 Возвращает счётчик принятых сообщений после последнего рестарта UART, операции очистки счётчиков или включения питания.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0B	00 00	Счётчик принятых сообщений

12 Возвращает счётчик ошибок CRC после последнего рестарта, операции очистки счётчиков или включения питания.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0C	00 00	Счётчик ошибок CRC

13 Возвращает счётчик сообщений об ошибках, насчитанных после последнего рестарта, операции очистки счётчиков или включения питания.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0D	00 00	Счётчик неправильных запросов

14 Возвращает счетчик запросов, адресованных устройству и широковещательных запросов, насчитанных после последнего рестарта, операции очистки счётчиков или включения питания.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0E	00 00	Счётчик запросов

15 Возвращает счетчик запросов подчинённому, которые остались без ответа, насчитанных после последнего рестарта, операции очистки счётчиков или включения питания.

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 0F	00 00	Счётчик запросов без ответа

16 Возвращает счетчик сообщений, адресованных устройству, для которых был возвращён ответ с сообщением об ошибке типа Negative Acknowledge (NAK)

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 10	00 00	Счётчик NAK

17 Возвращает счетчик ответов «занят»

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 11	00 00	Счётчик ответов «занят»

18 Возвращает счетчик ошибок переполнения буфера

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 12	00 00	Счётчик переполнения

20 Очищает счётчик переполнений и флаг ошибок переполнения буфера

Подфункция	Поле данных (запрос)	Поле данных (ответ)
00 14	00 00	00 00

10H ЗАПИСЬ НЕСКОЛЬКИХ РЕГИСТРОВ

Эта функция используется для записи непрерывного блока регистров (от 1 до 123 регистров) в удалённом устройстве. Запрос содержит стартовый адрес регистра, количество записываемых регистров, число передаваемых байт и данные регистров. Нормальный ответ возвращает код функции, стартовый адрес и количество записанных регистров.

Запрос

Код функции	1 байт	0x10
Стартовый адрес	2 байта	**
Количество регистров	2 байта	0x0001 – 0x007B
Количество байт	1 байт	2 x N*
Значения регистров	N* x 2 байта	

*N – число регистров

Ответ

Код функции	1 байт	0x10
Стартовый адрес	2 байта	**
Количество регистров	2 байта	0x0001 – 0x007B

Сообщение об ошибке

Код ошибки	1 байт	0x90
Причина ошибки	1 байт	02 или 03 или 04

**Допустимые адреса и значения регистров для записи см. в таблице, указанной в описании функции 06.

ИЗМЕНЕНИЕ ПАРОЛЕЙ

Для изменения пароля доступа к уставкам или пароля доступа к счётчикам и журналам необходимо выполнить последовательно три запроса к устройству.

1. Разрешение установки пароля путём записи константы в регистр функцией 06h.
2. Запись старого пароля в 2 регистра пароля, используя функцию 10h. См. в таблице, указанной в описании функции 06h.
3. Запись нового пароля в те же регистры пароля, используя функцию 10h.

При несовпадении отправленного старого пароля с хранящимся в ПЦ, либо неверной последовательности действий на третий запрос будет выдано сообщение об ошибке с причиной ошибки 04. Пароль в случае любых ошибок останется прежний.

ВЫПОЛНЕНИЕ КАЛИБРОВКИ

Выполнение калибровки производится в следующей последовательности:

1. Разрешить доступ к выполнению калибровки путём записи константы в регистр функцией 06h.
2. Записать пароль доступа к калибровке (2 регистра), используя функцию 10h.

Для калибровки смещения нуля:

- закоротить выводы входа текущего диапазона измерений;
- записать 0x000F в регистр внешней калибровки смещения нуля, см. таблицу, указанную в описании функции 06h.

Для калибровки усиления:

- подать на вход текущего диапазона измерений сигнал максимального значения, для входа 1 диапазона ± 80 мВ это значение равно 80 мВ, для входа 2 диапазона ± 5.12 В – это 5,12 В, для входа 3 диапазона ±20,48 мА – это 20,48 мА, для входа 4 (термопара) – это 80 мВ;
- записать 0x000F в регистр внешней калибровки усиления.

Для внутренней калибровки:

- записать 0x000F в регистр внутренней калибровки.

Для отмены внешней калибровки:

- записать 0x000F в регистр отмены внешней калибровки.

При несовпадении отправленного пароля с хранящимся в устройстве, либо неверной последовательности действий на третий запрос будет выдано сообщение об ошибке с причиной ошибки 04.

2BH/0EH (43/14) ЧТЕНИЕ ИДЕНТИФИКАЦИИ УСТРОЙСТВА

Эта функция используется для чтения идентификационной и дополнительной информации конфигурации устройства.

Запрос

Код функции	1 байт	0x43
Считывание идентификации	1 байт	0x0E
Тип считывания	1 байт	0x01 или 0x02
Id объекта	1 байт	0x00

Ответ

Код функции	1 байт	0x43
Считывание идентификации	1 байт	0x0E
Тип считывания	1 байт	0x01 или 0x02
Уровень соответствия	1 байт	0x02
Кадр последовательности	1 байт	0x00
Резерв	1 байт	0x00
Количество объектов	1 байт	
номер объекта	1 байт	0x00

длина объекта	1 байт	
ASCII строка	длина объекта	зависит от ID объекта
…	…	…
номер n объекта	1 байт	n
длина n объекта	1 байт	
ASCII строка n объекта	длина объекта	зависит от ID n объекта

В случае ошибки при обработке запроса выдается специальный исключительный ответ

Код ошибки	1 байт	0xAB (0x2B + 0x80)
Считывание идентификации	1 байт	0x0E
Причина ошибки	1 байт	01

В ответе может содержаться либо упрощённая идентификация, либо полная. Упрощённый ответ содержит только первые три базовых объекта с 0x00 по 0x02. Полный ответ содержит все семь объектов идентификации устройства с 0x00 по 0x06.

Номер объекта	Название объекта	строка ASCII	Категория объекта
0x00	торговое наименование	"ООО НПП Электромеханика"	базовый
0x01	код изделия	“1210”	
0x02	версия ПО	"6"	
0x03	электронный адрес производителя	"www.npp-em.ru"	стандарт
0x04	наименование изделия	"MC1210"	
0x05	наименование модели	"Модуль аналогового ввода"	
0x06	серийный номер	"0001100058"	

РЕГИСТРЫ ПРЕОБРАЗОВАТЕЛЯ MC1210

Адреса регистров, приведены в таблице 3.

Физически регистры измеренных параметров расположены в области с адреса 0x0200. Начиная с адреса 0x8000, размещены регистры конфигурации и управления. Для удобства и совместимости с другими устройствами в MODBUS сети, пространство 0x0000–0x00FF используется для отображения физических регистров. То есть, физическим регистрам можно назначать адреса-синонимы из диапазона 0x0000–0x00FF. Для этого необходимо в карту регистров (0x0100–0x01FF) записать адреса физических регистров, в любой последовательности, как это удобно для конкретной задачи. В карту регистров нельзя записать адреса вне диапазона физических регистров. Например, если записать в регистр 0x0100 значение 0x0200, то мгновенное действующее значение напряжения фазы A (U1a) можно читать и по адресу 0x0000, и, конечно, по неизменному 0x0200. В карте регистров значение, записанное по адресу 0x0100, соответствует отображаемому регистру по адресу 0x0000. Значение, записанное по адресу 0x0101, соответствует отображаемому регистру по адресу 0x0001, и так далее.

Регистры, фиксированные функцией 06h, доступны по тем же адресам, что и текущие. Их содержимое можно получить и из физической области (0x0200–0x02FF), и из области отображаемых регистров (0x0000–0x00FF).

Карта регистров (0x0100–0x01FF) едина и для фиксированных, и для текущих регистров.

Таблица 3

Имя регистра	Адрес	Чтение/ Запись
Отображаемые регистры в соответствии с картой регистров	0x0000	+/-

Имя регистра	Адрес	Чтение/ Запись
	…	+/-
	0x00FF	+/-
Карта регистров	0x0100	+/+
	…	+/+
	0x01FF	+/+
Конфигурация устройства, SnConf	0x0200	+/-
Регистр данных, SnRes мл. слово	0x0201	+/-
Регистр данных, SnRes ст. слово	0x0202	+/-
Регистр состояния, MS1210STATE	0x0203	+/-
Регистр НЧ фильтра, SF	0x0204	+/-
Конфигурация		
Конфигурация устройства, SnConf	0x1100	-/+
Внешняя калибровка смещения нуля	0x1101	-/+
Внешняя калибровка усиления	0x1102	-/+
Внутренняя калибровка	0x1103	-/+
Отменить внешнюю калибровку	0x1104	-/+
Регистр НЧ фильтра, SF	0x1105	-/+
Таблица термопары, 260 слов	0x1106	+/+
	…	+/+
	0x1209	+/+
Информация об устройстве, структура TUsoType		
Модель = 0x1012	0x2000	+/-
Аппаратная версия	0x2001	+/-
Версия программы	0x2002	+/-
Серийный номер, мл. слово	0x2003	+/-
Серийный номер, ст. слово	0x2004	+/-
Контрольная сумма ПО общая, мл. слово	0x2005	+/-
Контрольная сумма ПО общая, ст. слово	0x2006	+/-
Контрольная сумма метрологически значимой части ПО, мл. слово	0x2007	+/-
Контрольная сумма метрологически значимой части ПО, ст. слово	0x2008	+/-
Пароль на калибровку, мл. слово	0x7FD0	-/+
Пароль на калибровку, ст. слово	0x7FD1	-/+
Защёлка для фиксации данных	0x8000	-/+
Сброс регистра состояния	0x8001	-/+
Разрешение установки параметра	0x8010	-/+
Адрес устройства	0x8011	+/+
Скорость интерфейса	0x8012	+/+
Протокол связи канала	0x8013	+/+

ПРИМЕР И ИНТЕРПРЕТАЦИЯ

Правила перевода значений регистров в единицы физических величин

Примеры

Регистр	адрес	Значение	Интерпретация
Конфигурация устройства	0x0200	0x02	измерение напряжения на пределе 5 В
Регистр данных	0x0201 –0x0202	0x0044AA20 (4500000)	4500000 мкВ = 4.5 В

ТИПЫ ДАННЫХ

SNCONF

```
// Конфигурация устройства
int SnConf;
```

Допустимые значения SnConf	
0x01	измерение напряжения на пределе ±10 В
0x02	измерение напряжения на пределе ±5 В
0x03	измерение напряжения на пределе ±2.56 В
0x04	измерение напряжения на пределе ±1.28 В
0x05	измерение напряжения на пределе ±640 мВ
0x06	измерение напряжения на пределе ±320 мВ
0x07	измерение напряжения на пределе ±80 мВ
0x08	измерение напряжения на пределе ±20 мВ
0x10	измерение тока на пределе ±20 мА
0x21	измерение температуры

ТАБЛИЦА ТЕРМОПАРЫ

```
struct
{
  short int u[256];
  byte nT;  /* начальная (наиболее низкая) температура в градусах в виде целого
            числа со знаком*/
  byte dT;  /* шаг по температуре между элементами таблицы в виде целого
            положительного числа*/
  unsigned char Name[6]; //имя таблицы
} TabT;

/*Таблица термопары представлена в виде 16-ти разрядных нормализованных значений
со знаком, расположенных в порядке возрастания температуры и с постоянным шагом
по температуре.
[нормализованное значение] = [напряжение(мВ)]*32767/80
Значения таблицы занимают 256 слов. Неиспользованные элементы нижней части
таблицы (наиболее высокая температура) заполняются кодом 32767 (0x7FFF).
Последние 4 слова формируют заголовок таблицы.*/
```

MS1210STATE

```
// Регистр состояния
typedef struct _MS1210STATE
{
 unsigned char errRes:1;    // сброс устройства
 unsigned char errCSee:1;   // ошибка КС EEPROM
 unsigned char errCSprog:1; // ошибка КС памяти программ
 unsigned char errCRC:1;    // ошибка CRC канальной связи
```

```
 unsigned char errOV_FE:1;  // ошибка приёма данных(OERR или FERR или PERR)
 unsigned char errBREAK:1;  // обрыв термопары
 unsigned char errTabT:1;   // таблица нормализации не загружена
 unsigned char errRange:1;  // превышение диапазона АЦП

 unsigned char errDelay:1;  // опоздание записи во внешнюю энергонезависимую
память данных
 unsigned char errReady:1;  // нет готовности внешней энергонезависимой памяти
данных
 unsigned char errCSfram:1; // ошибка КС внешней энергонезависимой памяти данных
 unsigned char errT:1;      // ошибка датчика температуры холодного спая
термопары
 unsigned char errTanot:1;  // таблица линеаризации термопары "чужая"
 unsigned char errCSram:1;  // ошибка контрольной суммы оперативной памяти
процессора
 unsigned char stOffs:1;    // на текущем диапазоне произведена калибровка
смещения
 unsigned char stGain:1;    // на текущем диапазоне произведена калибровка
усиления unsigned char
} MS1210STATE;
```

РЕГИСТР ДАННЫХ

```
long SnRes;       // 32-х разрядное число со знаком
// либо напряжение в микровольтах,
// либо ток в микроамперах,
// либо температура в градусах Цельсия с шагом 0.1°
```

TUSOTYPE

```
// Информация об устройстве
typedef struct
{
unsigned int      Mod;              // Модель const=0x0112
unsigned int  HardwareVersion;      // Аппаратная версия
unsigned int  SoftwareVersion;      // Программная версия
unsigned byte     Ser_Numb[4];      // Серийный номер
} TUsoType;
```

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

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

ПОЛЕ ДАННЫХ: В нормальном ответе, подчиненный 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		Ответ	
Поле	Hex	Поле	Hex
Адрес	01	Адрес	01
Код функции	04	Код ошибки	84
Начальный адрес ст.	00	Причина ошибки	02
Начальный адрес мл.	2E	CRC ст.	C2
Кол-во регистров ст.	00	CRC мл.	C1
Кол-во регистров мл.	01		
CRC ст.	51		
CRC мл.	C3		

В данном примере главный требует прочитать несуществующий регистр с адресом 0x002E. Подчиненный возвращает сообщение об ошибке с кодом ошибки (02). Этот код специфицирует несуществующий адрес данных в подчиненном.

Список кодов ошибок представлен ниже.

Код	Название	Описание
01	ILLEGAL FUNCTION	Принятый код функции не поддерживается 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 недопустимой величиной для подчиненного.
04	SLAVE DEVICE FAILURE	Невосстанавливаемая ошибка имела место, пока подчиненный пытался выполнить затребованное действие.

ПРИЛОЖЕНИЕ В. 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 0xFFFF. 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. Результат 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 0xFFFF, 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 0xA001.

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 имеет значение 0x1241:

Адрес	Функция	Счетчик байт	Данные	Данные	Данные	Данные	CRC Мл.	CRC Ст.
							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.

```
const char auchCRCHi[256] = {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
    0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
```

```
    0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
    0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
    0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
    0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
    0x40};
const char auchCRCLo[256] = {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
    0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
    0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
    0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
    0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
    0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
    0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
    0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
    0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
    0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
    0x40};

unsigned short CRC16MOD(puchMsg, usDataLen)
char *puchMsg; /* Указатель на буфер */
unsigned short usDataLen; /* Количество байтов в буфере */
{
  struct {
          char uchCRCHi;  // здесь специально такой порядок байт,
          char uchCRCLo;  // т.к. в MODBUS сначала передаётся младший байт CRC
         } uchCRC;
  unsigned uIndex;

  *(word*)&uchCRC=0xFFFF;
  while (usDataLen--)
      {
        uIndex = uchCRC.uchCRCHi^ *puchMsg++;
        uIndex&=0x00FF;
        uchCRC.uchCRCHi = uchCRC.uchCRCLo^auchCRCHi[uIndex];
        uchCRC.uchCRCLo = auchCRCLo[uIndex];
      }
  return *(word*)&uchCRC;
}
```