Автор- Иржи Антош, Орбит Меррет, Прага, Чехия
Перевод с английского – Герман Цимерман, АКС-Сервис, Орел, Россия

Описание протокола MODBUS приборов Орбит Меррет

Введение

В данном документе описывается реализация протокола MODBUS. Для начала, следует сказать, что MODBUS является третьим протоколом, внедренным в приборы Орбит Меррет. Он предназначен для организации связи с системами управления и диспетчеризации.
Орбит Меррет поставляет программное обеспечение OM-Link для простого управления приборами. Это ПО работает только по протоколу MODBUS ASCII.

Параметры связи – нижний уровень протокола

Интерфейс : RS485
Среда передачи данных: витая пара
Максимальная длина :1200 м
Максимальное количество устройств: 32 (для большего количества устройств требуется повторитель)
Скорость обмена: 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400 бит/с
Количество старт-бит: 1
Количество бит данных: 8
Количество стоп-бит: 1
Контроль честности: None
Расстояние между сообщениями: >3,5 символов*
Расстояние между символами: не контролируется

* Сеанс связи прерывается, если не было обмена данными в течение 3,5 символов. Этот период определяется с допуском ± 250 мкс. Протокол MODBUS имеет стандартные скорости обмена до 19200 бит/с. Для более высоких скоростей необходимо учитывать этот допуск, например для скорости обмена 115200 бит/с, расстояние между посылками составляет 500±250 мкс, а для скорости 230400 - 250±250 мкс.

Формат сообщения MODBUS RTU

<ADR> <FNC> <DATA> <CRC-L> <CRC-H>
<ADR> - поле адреса, длиной 1 байт, в диапазоне от 1 до 247. Широковещательный адрес 0 приборами Орбит Меррет не поддерживается
<FNC> - поле команды, длиной 1 байт, которое описывает содержимое поля <DATA>. Подробное описание команд и их данных приведено ниже.
<CRC> - поле контрольной суммы, длиной 2 байта. Описание контрольной суммы приведено ниже.

Таблица 1. Поддерживаемые типы данных

Тип данных	Сокращенное название	Диапазон	Размещение в теле сообщения
длина 1 байт			
index of List (индекс списка)	List	0..255	<0x00><Byte>
unsigned char (беззнаковый символ)	Char	0..255	<0x00><Byte>
длина 2 байта			
unsigned integer (беззнаковое целое)	Int	0..65535	<High><Low>

2-character description (2-символьное описание)	Desc	'..'', 0x20 .. 0x7F	<First><Second>
длина 3 байта			
3-character description (3-символьное описание)	Desc3	'..'', 0x20 .. 0x7F	<0x00><First><Second><Third>
длина 4 байта			
unsigned long integer (беззнаковое длинное целое)	Long	0..4294967295	<0x00><First><Second><Third>
IEEE Floating poin (число с плавающей точкой в стандарте IEEE 754)	Float	±6,80564693277E+38	<SE><EM><MM><MM>

Таблица 2. Формат числа с плавающей точкой

Байт	Обозначение	Биты	Пояснение
байт 1	<SE>	SEEE EEEE	S(знак) 0=>1, 1=>-1 E(порядок) 0x00=>-127, 0x7F=>0, 0xFF=>128
байт 2	<EM>	EMMM MMMM	Нормализованная мантисса (1,0 ≤ Мантисса ≤ 2,0). Старший бит мантиссы всегда = 1
байты 3 и 4	<MM><MM>		Оставшиеся биты мантиссы. Например, 0x3F80 0000 = S*2^E*M=1*2^(0)*1=1

Подробное описание команд

Команда 0x01 – Read coils, State of relays *(считать катушки, состояния реле)*

Запрос: <ADR><0x01><0x00><0x00><0x00><0x08><CRC Lo><CRC Hi>
Ответ: <ADR><0x01><0x01><RR><CRC Lo><CRC Hi>
<RR> - это двоично-кодированное состояние реле.
Отсутствующие реле возвращают 0.

0x80 – включено реле 8
0x40 – включено реле 7
0x20 – включено реле 6
0x10 – включено реле 5
0x08 – включено реле 4
0x04 – включено реле 3
0x02 – включено реле 2
0x01 – включено реле 1

Команда 0x02 – Read external inputs *(считать внешние входы)*

Запрос: <ADR><0x02><0x00><0x00><0x00><0x08><CRC Lo><CRC Hi>
Ответ: <ADR><0x02><0x01><Inputs><CRC Lo><CRC Hi>
<Inputs> - это двоично-кодированное состояние внешних входов.
Отсутствующие входы возвращают 0.

0x80 – включен вход 8
0x40 – включен вход 7
0x20 – включен вход 6
0x10 – включен вход 5
0x08 – включен вход 4
0x04 – включен вход 3
0x02 – включен вход 2
0x01 – включен вход 1

Команда 0x03 – Read menu items *(считать пункты меню)*
Эта команда имеет различный формат, в зависимости от пункта меню.
Формат с 1-им или 2-мя байтами
```
Запрос:  <ADR><0x03><AA High><AA Low><0x00><0x01><CRC Lo><CRC Hi>
Ответ:   <ADR><0x03><Data Hi><Data Lo><CRC Lo><CRC Hi>
```
Формат с 3-мя или 4-мя байтами
```
Запрос:  <ADR><0x03><AA High><AA Low><0x00><0x02><CRC Lo><CRC Hi>
Ответ:   <ADR><0x03><MSB of data><data><data><LSB of data><CRC Lo><CRC Hi>
```
→ Предупреждение! Некоторые пункты меню не могут быть прочитаны!

Команда 0x04 – Read measured or evaluated values *(Считать измеренное или вычисленное значение)*
Данные в этой команде всегда представлены в формате Float.
```
Запрос:  <ADR><0x04><AA High><AA Low><0x00><0x02><CRC Lo><CRC Hi>
Ответ:   <ADR><0x04><MSB of data><data><data><LSB of data><CRC Lo><CRC Hi>
```
→ Предупреждение! Некоторые параметры не могут быть прочитаны!

Команда 0x05 – Do Action *(Выполнить действие)*
```
Запрос:  <ADR><0x05><AA High><AA Low><0xFF><0x00><CRC Lo><CRC Hi>
Ответ:   <ADR><0x05><AA High><AA Low><0xFF><0x00><CRC Lo><CRC Hi>
```
→ Состояние реле всегда определяется самим прибором. Описываемая команда только запускает действие. Многократное выполнение действия не поддерживается.

Команда 0x06 – Set menu items *(Изменить пункт меню)*
Эта команда предназначена для работы с пунктами меню длиной 1 или 2 байта
```
Запрос:  <ADR><0x06><AA High><AA Low><Data Hi><Data Lo><CRC Lo><CRC Hi>
Ответ:   <ADR><0x06><AA High><AA Low><Data Hi><Data Lo><CRC Lo><CRC Hi>
```
→ Предупреждение! Изменение некоторых пунктов меню этой командой не поддерживается.

Команда 0x10 – Set menu items *(Изменить пункт меню)*
Эта команда предназначена для работы с пунктами меню длиной 3 или 4 байта
```
Запрос:  <ADR><0x10><AA High><AA Low><0x00><0x02><0x04>
         <MSB of data><data><data><LSB of data><CRC Lo><CRC Hi>
Ответ:   <ADR><0x10><AA High><AA Low><0x00><0x02><CRC Lo><CRC Hi>
```
→ Предупреждение! Изменение некоторых пунктов меню этой командой не поддерживается.

Команда 0x11 – Slave identification *(Идентификация подчиненного устройства)*
Эта команда предназначена для работы с пунктами меню длиной 3 или 4 байта
```
Запрос:  <ADR><0x11><CRC Lo><CRC Hi>
Ответ:   <ADR><0x11><0x1C><0xFF><0xFF>
         <data>..<data> - 12 символов названия прибора, например
                          «OM 402UNI   »
         <'-'><data>..<data> - 6 символов версии встроенного ПО, например
                               «62-001»
         <'-'><data>..<data> - 6 символов названия режима измерения, например
                               «  40 V»
         <CRC Lo><CRC Hi>
```

Используемые адреса регистров *(поля <AA High><AA Low>)*
Список этих адресов можно получить по запросу на электронную почту orbit@merret.cz или на сайте компании http://www2.merret.cz/podpora/Rs/Index_en.htm. Адреса нужно использовать без первой цифры, например 40012 нужно использовать как <ADR><0x06><0x00><0x0C><0x02>.

Ответ на ошибочный запрос
В случае неправильного адреса или контрольной суммы ответа не последует.
В случае любой другой ошибки ответы будут следующими:
```
<ADR> <FNC + 0x80> 01 <CRC Lo> <CRC Hi>   - получена неподдерживаемая команда
<ADR> <FNC + 0x80> 02 <CRC Lo> <CRC Hi>   - неправильный адрес регистра
<ADR> <FNC + 0x80> 03 <CRC Lo> <CRC Hi>   - неправильное количество обмоток
```

```
                                                  или регистров
<ADR> <FNC + 0x80> 04 <CRC Lo> <CRC Hi> - команда не может быть выполнена
                                          (неправильные данные или пункт меню
                                          динамически заблокирован).
```

Генерирование контрольной суммы (CRC)

Циклический избыточный код (CRC) – это код для обнаружения ошибок данных, применяемый в цифровых сетях и устройствах хранения данных. В приборах Орбит Меррет для передачи CRC используется поле длиной в 2 байта, содержащее 16-битное двоичное число. CRC генерируется передающим устройством, которое добавляет его к сообщению. Получающее устройство перерасчитывает CRC во время обработки полученного сообщения, и сравнивает рассчитанную величину с величиной, полученной в поле CRC. Если их величины не равны, то возникает ошибка. Процесс вычисления CRC начинается с установки всех его 16-ти битов в 1. Затем процесс начинается с последовательного наложения байтов сообщения на текущее содержимое регистра, в котором ведется расчет CRC. При этом используются только 8 бит данных, старт-бит, стоп-бит, бит контроля четности не используются. Во время генерирования CRC, каждый 8-битный символ накладывается на содержимое регистра с помощью операции «Исключающее ИЛИ». Затем результат сдвигается в сторону младшего разряда (LSB) с установкой в 0 старшего разряда (MSB). Затем младший разряд извлекается и проверяется. Если он равен 1, то регистр проходит через операцию «Исключающее ИЛИ» с предустановленной фиксированной величиной, если же он равен 0, то такая операция не выполняется. Эта процедура выполняется пока не будут выполнены 8 сдвигов. После последнего, 8-го сдвига, следующий 8-битный символ накладывается на текущее содержимое регистра с помощью операции «Исключающее ИЛИ» и процесс повторяется на протяжении еще 8 сдвигов, как описано выше. После такой обработки всех символов сообщения получается окончательная величина CRC.

Процедура генерирования CRC:

1. Загружаем 16-битный регистр числом 0xFFFF(все биты=1). Назовем его регистром CRC.
2. Проводим «Исключающее ИЛИ» между первым байтом сообщения и младшим байтом CRC, помещая результат в этот регистр.
3. Сдвигаем регистр вправо на 1 разряд, в старший бит помещаем 0. Извлекаем и проверяем младший бит.
4. Если младший бит=0, то повторяем шаг 3. Если младший бит=1, проводим операцию «Исключающее ИЛИ» регистра CRC с полиномиальной величиной 0xA001.
5. Повторяем шаги 3-4 пока не будут выполнены 8 сдвигов. После их завершения весь байт данных будет обработан.
6. Повторяем шаги 2-5 для следующего байта сообщения. Продолжаем эти операции пока все байты сообщения не будут обработаны.
7. Окончательное значение, полученное в регистре CRC и будет величиной CRC сообщения.
8. При помещении CRC в сообщение, старший и младший байт меняются местами, как указано выше в структуре сообщений.

Ниже приводится пример на языке С, в котором описана функция генерирования CRC. Все возможные величины CRC предустановленны в двух массивах, которые просто индексируются как функция приращения буфера сообщений. Один массив содержит все 256 возможных значения старшего байта 16-битного CRC, другой массив содержит все возможные значения младшего байта. Индексирование CRC, в данном случае, обеспечивает более быстрое получение нужного значения, нежели прямой расчет по вышеприведенному алгоритму. Так же эта функция обеспечивает перестановку байтов согласно его пункта 8. Таким образом, CRC возвращаемый функцией, может быть непосредственно добавлен к передаваемому сообщению.

Функция имеет два аргумента:
*unsigned char *puchMsg ;* /* указатель на буфер сообщений, в котором находятся двоичные данные, исползуемые для генерирования CRC */
unsigned short usDataLen ; /*количество байт в буфере сообщений*/
Функция возвращает CRC с типом *unsigned short.*

Функция генерирования 16-битного CRC

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ; /* сообщение для генерирования CRC */
unsigned short usDataLen ; /* количество байт в сообщении */
{
  unsigned char uchCRCHi = 0xFF ; /* старший байт для инициализации регистра CRC */
  unsigned char uchCRCLo = 0xFF ; /* младший байт для инициализации регистра CRC */
  unsigned uIndex ; /* индекс в таблице байтов CRC */
  while (usDataLen—) /* цикл прохода по буферу сообщений */
  {
     uIndex = uchCRCHi ^ *puchMsgg++ ; /* расчет CRC */
     uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
     uchCRCLo = auchCRCLo[uIndex] ;
   }
  return (uchCRCHi << 8 | uchCRCLo) ;
}
/* Таблица старших байтов CRC */
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, 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};
/* Таблица младших байтов CRC */
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};
```