ООО «НПП Электромеханика»

# Описание протокола обмена данными стандарта МЭК-870-5-1-95 формата FT3 МС1218Ц

Данное описание применимо к МС1218Ц с программной версией от 5. Информация о модулях с более ранними версиями содержится в архиве МС12Х_СК.HLP.

10.03.2011

# ОГЛАВЛЕНИЕ

# ОБЩИЕ ПРИНЦИПЫ ПЕРЕДАЧИ ДАННЫХ ПО СТАНДАРТУ МЭК-870-5-1-95 ФОРМАТ КАДРА FT3.

## ПЕРЕДАЧА В СЕТИ

Устройства в сети отвечают на запросы главного контроллера. Байты идут непрерывным потоком. Запрос – ответ. Начало кадра запроса и ответа идентифицируется маркером (двумя специальными байтами). МС1218Ц начинает отвечать через 2 мс после получения последнего байта запроса.

## ФРЕЙМ

**Назначение битов**: 1 стартовый бит; 8 бит данных, младшим значащим разрядом вперед, паритет отсутствует; 1 стоповый бит.

| старт | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | стоп |
|---|---|---|---|---|---|---|---|---|---|

## ФОРМАТ КАДРА ЗАПРОСА

Кадр запроса состоит из стартовой последовательности длиной 2 байта, одного блока данных длиной 14 байт и двух байт CRC в конце. CRC рассчитывается для 14 байт, начиная с длины.

Кадр запроса содержит следующие поля:

| Наименование | Описание | Размер | Значение |
|---|---|---|---|
| Head | Стартовая последовательность | 2 байта | 0x05<br>0x64 |
| DataLen | Длина данных | 1 байт | 0x00 |
| ControlByte | Контрольный байт | 1 байт | 0x00 |
| Address | Адрес | 2 байта, младший байт передается первым | 0x0000–<br>0xFFFF* |
| Command | Команда для устройства | 1 байт | |
| Parameters | Параметры команды | 9 байт | |
| CRC | Контрольная сумма | 2 байта, старший байт передается первым | |

*Address = 0x00FF - широковещательный адрес. При совместном использовании с протоколом MODBUS помнить, что в MODBUS допустимый адрес устройства ограничен значением 0x01 – 0xF7.

См. структуру PKTSEND.

## ФОРМАТ КАДРА ОТВЕТА

Кадр ответа состоит из стартовой последовательности длиной 2 байта и одного или нескольких блоков данных.

Кадр ответа с одним блоком данных имеет вид:

| Наименование | Описание | Размер | Значение |
|---|---|---|---|
| Head | Стартовая последовательность | 2 байта | 0x05<br>0x64 |
| DataLen | Длина данных | 1 байт | 0x0E |
| ControlByte | Контрольный байт | 1 байт | 0x00 |
| Address | Адрес | 2 байта, младший байт передается первым | 0x0000–<br>0xFFFF |
| Data | Данные | 10 байт, младший байт передается первым | |
| CRC | Контрольная сумма | 2 байта, старший байт передается первым | |

Если число передаваемых данных не более 10 байт, то кадр ответа содержит 1 блок данных, фиксированной длины - 16 байт (из них 4 байта – заголовочная часть, 2 байта - CRC). В поле длины DataLen, независимо от количества байт данных в блоке, передается 14. Содержимое незадействованных байт данных может быть произвольным, CRC считается для всех 14 байт, начиная с поля длины.

Если число передаваемых данных более 10 байт, то кадр ответа содержит несколько блоков данных. Каждый блок данных заканчивается двумя байтами CRC. Первый блок данных также имеет заголовочную часть (4 байта), которая является заголовочной частью для всего кадра (последующие блоки не содержат заголовочной части). В поле длины DataLen указывается количество байт данных в кадре (без стартовой последовательности и CRC).

Длина первого блока всегда 16 байт (с учетом заголовочной части и 2 байт CRC), длина последнего блока определяется количеством байт данных в нем и может находиться в пределах от 3 (1 байт данных, 2 байта CRC) до 16, все промежуточные блоки имеют длину 16 байт (14 байт данных, 2 байта CRC).

Кадр ответа из нескольких блоков содержит следующие поля:

| Наименование | Описание | Размер | Значение |
|---|---|---|---|
| Head | Стартовая последовательность | 2 байта | 0x05<br>0x64 |
| DataLen | Длина данных в кадре | 1 байт | 0x0F – 0xFF |
| ControlByte | Контрольный байт | 1 байт | 0x00 |
| Address | Адрес | 2 байта, младший байт передается первым | 0x0000–<br>0xFFFF |
| Data | Данные | 10 байт, младший байт передается первым | |
| CRC | Контрольная сумма | 2 байта, старший байт передается первым | |
| Data | Данные | 14 байт, младший байт передается первым | |
| CRC | Контрольная сумма | 2 байта, старший байт передается первым | |
| … | … | … | … |
| Data | Данные | 1-14 байт, младший байт передается первым | |
| CRC | Контрольная сумма | 2 байта, старший байт передается первым | |

В поле DataLen указывается длина данных Data плюс 4 байта, учитывающие размер полей DataLen, ControlByte и Address. Длина кадра ответа (исключая поле Head и поля CRC) не должна превышать 255 байт.

См. структуры PKTHEAD, PKTREADHEAD, PKTREADDATA.

# СИСТЕМА КОМАНД ПРЕОБРАЗОВАТЕЛЯ ТЕМПЕРАТУРЫ МС1218Ц

## СПИСОК КОМАНД МС1218Ц

| Код команды | Наименование |
|---|---|
| 0x01 | Подготовка к записи данных |
| 0x02 | Запись адреса |
| 0x03 | Чтение адреса |
| 0x08 | Считать информацию об устройстве |
| 0x15 | Установка скорости обмена данными |
| 0x86 | Поиск датчиков температуры |
| 0x87 | Калибровать датчики температуры |
| 0x88 | Получить количество датчиков температуры |
| 0x89 | Получить температуру со всех датчиков |
| 0x8A | Задать уставку |
| 0x8B | Прочитать уставку |
| 0x8C | Задать состояние выхода телеуправления |
| 0x8D | Читать состояние выхода телеуправления |
| 0xFF | Задать протокол обмена данными |

## 0X01 ПОДГОТОВКА К ЗАПИСИ ДАННЫХ

| **Параметры** | **Байты структуры** PARAMETRS |
|---|---|
| Признак команды | P1=0xA5 |

**Возвращаемые данные:** нет

Команда 0x01 «Подготовка к записи данных» является предварительной для любой команды, изменяющей внутренние данные энергонезависимой памяти МС1218Ц.

## 0X02 ЗАПИСЬ АДРЕСА

**Предварительная команда:** 0x01 Подготовка к записи данных

| **Параметры** | **Байты структуры** PARAMETRS |
|---|---|
| Старый адрес | P1-P2, младший байт передается первым |
| Новый адрес | P3-P4, младший байт передается первым |

**Возвращаемые данные:** нет

## 0X03 ЧТЕНИЕ АДРЕСА

МС1218Ц поставляется с установленным адресом, равным 1.

| **Параметры** | **нет** |
|---|---|

**Возвращаемые данные:** (см. поле Data в разделе “Формат кадра ответа”)

- 0-1 байт Data: Адрес (младший байт передается первым).

## 0X08 СЧИТАТЬ ИНФОРМАЦИЮ ОБ УСТРОЙСТВЕ

| Параметры | нет |
|---|---|

**Возвращаемые данные:** структура TUsoType

Для МС1218Ц, модель = 0x1812, серийный номер - трёхбайтовый.

## 0X15 УСТАНОВКА СКОРОСТИ ОБМЕНА ДАННЫМИ

**Предварительная команда:** 0x01 Подготовка к записи данных

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Константа скорости | P1 |

Константы скоростей (SENSORSPEED)

| Константа скорости | Скорость RS485, бит/с |
|---|---|
| 0x01 | 19200 |
| 0x02 | 9600 |
| 0x03 | 4800 |
| 0x04 | 2400 |
| 0x05 | 1200 |
| 0x11 | 38400 |
| 0x12 | 57600 |
| 0x13 | 115200 |

**Возвращаемые данные:** нет

МС1218Ц поставляется с установленной скоростью 9600 бит/с.

## 0X86 ПОИСК ДАТЧИКОВ ТЕМПЕРАТУРЫ

**Предварительная команда:** 0x01 Подготовка к записи данных

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Режим поиска | P1 |

**Возвращаемые данные:** нет

По команде 0x86 Поиск датчиков температуры МС1218Ц сканирует все присоединённые к нему датчики температуры и в зависимости от параметра P1 выполняет следующие действия:

| | |
|---|---|
| P1=0 | Осуществляется поиск всех подключенных датчиков температуры. Серийные номера найденных датчиков записываются в память прибора в виде таблицы MC1218Data. При этом, вся информация о неподсоединенных датчиках стирается. Калибровочные константы обнуляются. |
| P1=1 | Осуществляется поиск новых датчиков температуры, с серийными номерами которые не записаны в память устройства. При этом сохраняется старый порядок присоединенных датчиков в таблице MC1218Data. Новые датчики добавляются в таблицу на места отсоединенных (старый удаляется), либо в конец таблицы MC1218Data. Информация о калибровке для датчиков с известными ранее серийными номерами не теряется. |

## 0X87 КАЛИБРОВАТЬ ДАТЧИКИ ТЕМПЕРАТУРЫ

**Предварительная команда:** 0x01 Подготовка к записи данных

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Эталонная температура | P1-P2, младший байт передается первым |

**Возвращаемые данные:** нет

При получении команды 0x87 Калибровать датчики температуры MC1218Ц определяет поправку для каждого присоединённого датчика в соответствии с полученной эталонной температурой. Эта поправка сохраняется в энергонезависимой памяти преобразователя и учитывается для всех дальнейших измерений. Эталонная температура имеет формат t(°C)·16 со знаком.

Примечание: перед подачей команды 0x87 все присоединённые датчики должны иметь температуру, равную эталонной.

## 0X88 ПОЛУЧИТЬ КОЛИЧЕСТВО ДАТЧИКОВ ТЕМПЕРАТУРЫ

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Количество сенсоров | P1 |

**Возвращаемые данные:** в 0 байте поля Data возвращается количество датчиков температуры (SensorCount), обнаруженных по команде 0x86 Поиск датчиков температуры

## 0X89 ПОЛУЧИТЬ ТЕМПЕРАТУРУ СО ВСЕХ ДАТЧИКОВ

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Формат считывания | P1 |

**Возвращаемые данные:**

- если P1 = 1, то возвращается только температура и байт флага успешности считывания температуры, структура MC1218TEMPERATURE1
- если P1 = 0, то возвращаются значения температуры датчиков вместе с их уникальными серийными номерами (ROM code) и байтами флагов успешности считывания температуры, т.е. массив MC1218Data структур типа MC1218TEMPERATURE0 размером SensorCount.

Для перевода температуры в шкалу Цельсия необходимо полученные 16-битные числа со знаком делить на 16. Таким образом, разрешающая способность температурного сенсора равна 0.0625 °C.

## 0X8A ЗАДАТЬ УСТАВКУ

**Предварительная команда:** 0x01 Подготовка к записи данных

| **Параметры** (структура TUst) | Байты структуры PARAMETRS |
|---|---|
| Верхний предел температуры TUst.TempHi | P1 – P2, младший байт передается первым |
| Нижний предел температуры TUst.TempLo | P3 – P4, младший байт передается первым |

Уставка предназначена для включения-выключения ТУ по следующему алгоритму:

1. Если температура сенсора номер 0 превысит верхний предел температуры, то ТУ выключается.
2. Если температура сенсора номер 0 становится меньше нижнего предела температуры, ТУ включается.
3. Если при включении питания преобразователя МС1218Ц температура, измеренная сенсором номер 0 ниже верхнего предела температуры, то ТУ включается. И далее работает по правилам 1 и 2.

**Возвращаемые данные**: нет

## 0X8B ПРОЧИТАТЬ УСТАВКУ

| **Параметры** | **нет** |
|---|---|

**Возвращаемые данные:** структура TUst

## 0X8C ЗАДАТЬ СОСТОЯНИЕ ВЫХОДА ТЕЛЕУПРАВЛЕНИЯ

| Параметры | Байты структуры PARAMETRS |
|---|---|
| Состояние ТУ | P1 |

| Код состояния ТУ | Состояние ТУ |
|---|---|
| 0 | ТУ выключить (контакт разомкнуть) |
| 1 | ТУ включить (контакт замкнуть) |

**Возвращаемые данные:** нет

Команда 0x8C имеет более высокий приоритет, чем срабатывание ТУ по уставке.

## 0X8D ЧИТАТЬ СОСТОЯНИЕ ВЫХОДА ТЕЛЕУПРАВЛЕНИЯ

| **Параметры** | **нет** |
|---|---|

**Возвращаемые данные:** (0 байт в поле Data равен)

- 1 – ТУ включено, контакт замкнут;
- 0 – ТУ выключено, контакт разомкнут.

## 0XFF ЗАДАТЬ ПРОТОКОЛ ОБМЕНА ДАННЫМИ

**Предварительная команда:** 0x01 Подготовка к записи данных

| **Параметры** | **Байты структуры** PARAMETRS |
|---|---|
| Константа протокола | P1 |
| Контрольный байт | P2=0x22 |
| Контрольный байт | P3=0xBA |
| Контрольный байт | P4=0x1E |
| Контрольный байт | P5=0x45 |

Константы протоколов:

| Константа протокола | Протокол |
|---|---|
| 0x01 | FT3 |
| 0x02 | MODBUS RTU |

**Возвращаемые данные:** нет

Команда 0xFF “Задать протокол” устанавливает канал связи в режим работы по протоколу обмена данными MODBUS RTU.

# ПРИЛОЖЕНИЕ А. СТРУКТУРЫ ДАННЫХ

## PKTSEND

Пакет для передачи

```
typedef struct _PKTSEND
{
 PKTHEAD        Head;          //Заголовок пакета
 unsigned  char   DataLen;     //Длина данных
 unsigned  char   ControlByte; //Контрольный байт = 0x00
 unsigned  short  Address;     //Адрес устройства
 unsigned  char   Command;     //Команда для устройства
 PARAMETRS P1P9;               //Параметры
 unsigned  short  CRC;         //Контрольная сумма
} PKTSEND;
```

## PKTHEAD

Заголовок пакета

```
typedef struct _PKTHEAD
{
 unsigned  char   HeadByte1;   //Сигнатура заголовка: Байт N1 = 0x05
 unsigned  char   HeadByte2;   //Сигнатура заголовка: Байт N2 = 0x64
}PKTHEAD;
```

## PARAMETRS

Параметры пакета передачи

```
typedef struct _PARAMETRS
{
 unsigned  char  P1;      //Параметр N1
 unsigned  char  P2;      //Параметр N2
 unsigned  char  P3;      //Параметр N3
 unsigned  char  P4;      //Параметр N4
 unsigned  char  P5;      //Параметр N5
 unsigned  char  P6;      //Параметр N6
 unsigned  char  P7;      //Параметр N7
 unsigned  char  P8;      //Параметр N8
 unsigned  char  P9;      //Параметр N9
} PARAMETRS;
```

## PKTREADHEAD

Стартовый пакет приема

```
typedef struct _PKTREADHEAD
{
 unsigned  char   DataLen;     //Длина данных
 unsigned  char   ControlByte; //Контрольный байт
 unsigned  short  Address;     //Адрес устройства
```

```
 unsigned  char   Data[10];       //Данные
 unsigned  short  CRC;            //Контрольная сумма
} PKTREADHEAD;
```

### PKTREADDATA

Пакет приема данных

```
typedef struct _PKTREADDATA
{
 unsigned  char   Data[14];       //Данные
 unsigned  short  CRC;            //Контрольная сумма
} PKTREADDATA;
```

Примечание: Длина поля Data в зависимости от размера кадра может варьироваться // от 1 до 14.

### TUSOTYPE

Идентификация устройства

```
typedef struct {
    unsigned  short  Model;                 // Модель устройства
    unsigned  char  HardwareVersion;        // Аппаратная версия
    unsigned  char  SoftwareVersion;        // Версия ПО
    unsigned  char  Reserv4;                // Не используется
    unsigned  char  Reserv5;                // Не используется
    unsigned  char  Reserv6;                // Не используется
    unsigned  char  SerialNumberHigh;       // Серийный номер – старший байт
    unsigned  short  SerialNumber;          // Серийный номер
}TUsoType;

typedef struct
{
  int short Temperature[SensorCount];   // Температура(°C) = Temperature[i]/16.0
  unsigned char  Status;                // флаг успешности считывания
} MC1218TEMPERATURE1;

// SensorCount – количество датчиков температуры, возвращаемое командой 0x88
// ”Получить количество датчиков температуры”
// Status – флаг успешности считывания температуры. Наличие единицы в нулевом бите
// означает корректное считывание температуры с нулевого датчика. Соответственно
// единица в первом бите означает корректное считывание температуры с первого
// датчика, и т.д.

typedef struct
{
 int      short Temperature;             // Температура(°C) = Temperature/16.0
 unsigned char  Address[7];              // серийный номер сенсора (56-bit ROM code)
 unsigned char  Status;                  // флаг успешности считывания
} MC1218TEMPERATURE0;                    // Status=1 - корректное считывание температуры
                                         // Status=0 - произошла ошибка считывания
MC1218TEMPERATURE0 MC1218Data[SensorCount];

typedef struct
{
   int short TempHi; // температура отключения ТУ (Температура(°C)=TempHi/16.0)
   int short TempLo; // температура включения ТУ  (Температура(°C)=TempLo/16.0)
} TUst;
```

## ПРИЛОЖЕНИЕ Б. ПРИМЕР ПРОГРАММЫ РАСЧЕТА CRC

```
const unsigned short crctable_ft3[256] = {
0x0000, 0x9EB3, 0xA3D5, 0x3D66, 0xD919, 0x47AA, 0x7ACC, 0xE47F,
0x2C81, 0xB232, 0x8F54, 0x11E7, 0xF598, 0x6B2B, 0x564D, 0xC8FE,
0x5902, 0xC7B1, 0xFAD7, 0x6464, 0x801B, 0x1EA8, 0x23CE, 0xBD7D,
0x7583, 0xEB30, 0xD656, 0x48E5, 0xAC9A, 0x3229, 0x0F4F, 0x91FC,
0xB204, 0x2CB7, 0x11D1, 0x8F62, 0x6B1D, 0xF5AE, 0xC8C8, 0x567B,
0x9E85, 0x0036, 0x3D50, 0xA3E3, 0x479C, 0xD92F, 0xE449, 0x7AFA,
0xEB06, 0x75B5, 0x48D3, 0xD660, 0x321F, 0xACAC, 0x91CA, 0x0F79,
0xC787, 0x5934, 0x6452, 0xFAE1, 0x1E9E, 0x802D, 0xBD4B, 0x23F8,
0xFABB, 0x6408, 0x596E, 0xC7DD, 0x23A2, 0xBD11, 0x8077, 0x1EC4,
0xD63A, 0x4889, 0x75EF, 0xEB5C, 0x0F23, 0x9190, 0xACF6, 0x3245,
0xA3B9, 0x3D0A, 0x006C, 0x9EDF, 0x7AA0, 0xE413, 0xD975, 0x47C6,
0x8F38, 0x118B, 0x2CED, 0xB25E, 0x5621, 0xC892, 0xF5F4, 0x6B47,
0x48BF, 0xD60C, 0xEB6A, 0x75D9, 0x91A6, 0x0F15, 0x3273, 0xACC0,
0x643E, 0xFA8D, 0xC7EB, 0x5958, 0xBD27, 0x2394, 0x1EF2, 0x8041,
0x11BD, 0x8F0E, 0xB268, 0x2CDB, 0xC8A4, 0x5617, 0x6B71, 0xF5C2,
0x3D3C, 0xA38F, 0x9EE9, 0x005A, 0xE425, 0x7A96, 0x47F0, 0xD943,
0x6BC5, 0xF576, 0xC810, 0x56A3, 0xB2DC, 0x2C6F, 0x1109, 0x8FBA,
0x4744, 0xD9F7, 0xE491, 0x7A22, 0x9E5D, 0x00EE, 0x3D88, 0xA33B,
0x32C7, 0xAC74, 0x9112, 0x0FA1, 0xEBDE, 0x756D, 0x480B, 0xD6B8,
0x1E46, 0x80F5, 0xBD93, 0x2320, 0xC75F, 0x59EC, 0x648A, 0xFA39,
0xD9C1, 0x4772, 0x7A14, 0xE4A7, 0x00D8, 0x9E6B, 0xA30D, 0x3DBE,
0xF540, 0x6BF3, 0x5695, 0xC826, 0x2C59, 0xB2EA, 0x8F8C, 0x113F,
0x80C3, 0x1E70, 0x2316, 0xBDA5, 0x59DA, 0xC769, 0xFA0F, 0x64BC,
0xAC42, 0x32F1, 0x0F97, 0x9124, 0x755B, 0xEBE8, 0xD68E, 0x483D,
0x917E, 0x0FCD, 0x32AB, 0xAC18, 0x4867, 0xD6D4, 0xEBB2, 0x7501,
0xBDFF, 0x234C, 0x1E2A, 0x8099, 0x64E6, 0xFA55, 0xC733, 0x5980,
0xC87C, 0x56CF, 0x6BA9, 0xF51A, 0x1165, 0x8FD6, 0xB2B0, 0x2C03,
0xE4FD, 0x7A4E, 0x4728, 0xD99B, 0x3DE4, 0xA357, 0x9E31, 0x0082,
0x237A, 0xBDC9, 0x80AF, 0x1E1C, 0xFA63, 0x64D0, 0x59B6, 0xC705,
0x0FFB, 0x9148, 0xAC2E, 0x329D, 0xD6E2, 0x4851, 0x7537, 0xEB84,
0x7A78, 0xE4CB, 0xD9AD, 0x471E, 0xA361, 0x3DD2, 0x00B4, 0x9E07,
0x56F9, 0xC84A, 0xF52C, 0x6B9F, 0x8FE0, 0x1153, 0x2C35, 0xB286};

unsigned short crc_ft3(unsigned char *Data, unsigned char DataLen)
{
  unsigned short crc = 0;
  unsigned char uIndex;

  while (DataLen--)
  {
    uIndex= ((crc>>8) ^ *Data++);
    crc<<=8;
    crc ^= crctable_ft3[uIndex];
  }
return (crc>>8)|(crc<<8);
}
```