ТЕРМОТРОНИК

Описание протокола Modbus RTU расходомера-счетчика «Питерфлоу Т» модификации 1 с системой верхнего уровня

ТРОН.00019-01 90 01

Редакция 1.02

ООО «ТЕРМОТРОНИК»
193318, Россия, Санкт-Петербург, ул. Ворошилова, д.2
Телефон, факс: +7 (812) 326-10-50
Сайт ООО «ТЕРМОТРОНИК»: **www.termotronic.ru**
Служба технической поддержки: **support@termotronic.ru**
тел. 8-800-333-10-34

Содержание

1 Введение 4
1.1 Общие сведения 4
1.2 Физический интерфейс 4
1.3 Уровни протокола обмена 4
1.3.1 Логический уровень 4
1.3.2 Коммуникационный уровень 4
ADU «Modbus RTU» 4
2 Реализованные функции протокола 6
2.1 Функция Modbus 03 (0x03 hex) (Read Holding Registers) и 04 (0x04 hex) (Read Input Registers) 6
2.2 Функция Modbus 06 (0x06 hex) (Write Single Register) 6
2.3 Функция Modbus 16 (0x10 hex) (Write Multiple Registers) 7
2.4 Функция Modbus 20 (0x14 hex) (Read File Record) 7
3 Организация доступа к данным 9
3.1 Используемые типы данных и условные обозначения 9
3.2 Регистровый доступ и характеристики параметров 9
3.3 Порядок хранения и передачи байт данных 9
3.4 Чтение/запись текущих и настроечных параметров 10
3.5 Организация архивов и считывание архивных данных 10
3.5.1 Интерпретация минутного, часового и суточного архивов 11
3.5.2 Интерпретация архива событий 12
4 Карта переменных 14
Приложение 1. Функция расчета контрольной суммы LRC 16
Приложение 2. Функции преобразования в ASCII и обратно 17
Приложение 3. Коды ошибок, возвращаемые прибором 18
Приложение 4. Функция расчета контрольной суммы Crc32 19
Приложение 5. Функции преобразования в BCD и обратно 20
Приложение 6. Оптимизация алгоритма считывания архивов 21
Приложение 7. Функция расчета контрольной суммы Cyclical Redundancy Checking (CRC16) 22

История редактирования

- 20.03.2020 создана редакция 1.00;
- 13.01.2021 создана редакция 1.01:
  - Исправлено обозначение типа double (32 бита исправлено на 64 бита);
- 15.01.2025 создана редакция 1.02:
  - Исправлено обозначение типа double (32 бита исправлено на 64 бита);
  - Добавлено описание целочисленных интегралов V+, V- (адреса 182, 184);
  -

1 Введение

1.1 Общие сведения

Данное описание распространяется на расходомеры-счётчики электромагнитные «Питерфлоу Т» модификации 1 (далее - Прибор), оснащаемые адаптером RS-485 и реализующие протокол Modbus RTU. Прибор позволяет получать текущие и архивные параметры, а также предоставляет доступ к чтению и изменению настроечных параметров через коммуникационный интерфейс. Прибор реализует протокол Modbus RTU в режиме «ведомый».

1.2 Физический интерфейс

Для связи с системой верхнего уровня прибор оснащен последовательным интерфейсом RS-485. Для уточнения характеристик последовательного интерфейса следует обращаться к руководству по эксплуатации на прибор.

1.3 Уровни протокола обмена

Этот раздел содержит краткие сведения из стандарта «Modbus». Подробную информацию можно получить из документов, размещенных на сайте www.modbus.org:

- Modbus Application Protocol;
- Modbus over serial line;
- MODBUS Messaging on TCP/IP Implementation Guide.

1.3.1 Логический уровень

Логический уровень протокола отвечает за способ доступа к данным. Протокол «Modbus» определяет понятие PDU (Protocol Data Unit), независимое от используемого коммуникационного протокола. PDU содержит 2 поля: код функции (длина 1 байт) и данные (длина не более 252 байт). Подробное описание логического уровня приведено в разделе Реализованные функции протокола.

Рисунок 4: Состав PDU

1.3.2 Коммуникационный уровень

Коммуникационный уровень протокола отвечает за доставку передаваемой информации между двумя совместимыми «Modbus». В различных коммуникационных протоколах PDU дополняется полями сетевой адрес, контрольная сумма, заголовок и т.д., образуя при этом, понятие ADU (Application Data Unit). Дополнительные поля требуются для адресации, идентификации и контроля целостности данных.

Независимо от коммуникационного протокола прибор работает только в режиме «ведомый». Это означает, что прибор может выполнить посылку только в ответ на запрос системы верхнего уровня. Время ответа прибора не превышает 500 мс. В случае отсутствия ответа от прибора система верхнего уровня должна выполнить повтор запроса.

ADU «Modbus RTU»

В случае использования коммуникационного протокола «Modbus RTU» PDU дополняется полями сетевой адрес и контрольная сумма.

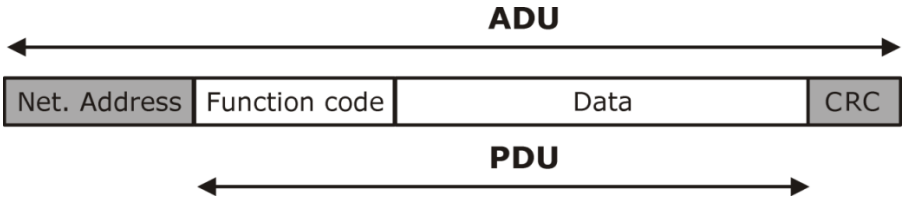

Рисунок 5: ADU для «Modbus RTU»

Сетевой адрес служит для адресации прибора в сети. Контрольная сумма служит для проверки целостности данных. Передающее устройство вычисляет контрольную сумму над всеми полями посылки, и затем результат вычисления добавляет в конец посылки. Принимающее устройство, получив всю посылку, вычисляет контрольную сумму кадра для всех байтов сообщения, исключая байты контрольной суммы. В случае, если принятая и вычисленная контрольные суммы равны, принимается решение о достоверности принятого кадра. В противном случае кадр считается недостоверным. Если прибор получает недостоверный кадр, он его игнорирует и не посылает каких-либо ответных сообщений. Это означает, что система верхнего уровня не получит ответа в течение ожидаемого времени и должна сделать повтор запроса. Если же факт получения недостоверной посылки обнаружен системой верхнего уровня, то она может выполнить повтор запроса.

При передаче используется двоичная система кодирования, начало и конец посылки определяются по длительности «интервала тишины» - времени, в течение которого не происходило передачи. Т.е. кадр сообщения «заключен» между двумя интервалами тишины. Байты сообщения должны передаваться непрерывным потоком. Длительность интервала тишины по стандарту «Modbus RTU» определяется как 3,5 длительности передачи байта на заданной скорости.

В качестве функции расчета контрольной суммы в режиме RTU используется Cyclical Redundancy Checking (CRC16). Пример функции расчета CRC16 приведен в Приложении 7, а описание генерации контрольной суммы может быть найдено в документации на сайте www.modbus.org.

2 Реализованные функции протокола

2.1 Функция Modbus 03 (0x03 hex) (Read Holding Registers) и 04 (0x04 hex) (Read Input Registers)

Функции предназначены для чтения двоичного содержимого шестнадцатиразрядных регистров прибора. Отличие функции 0x04 от 0x03 в том, что она применяется только для чтения параметров, недоступных для записи. В общем виде структура запроса и ответа имеет следующий вид:

PDU запроса:

Функция **0x03/0x04**	Начальный адрес (старший байт)	Начальный адрес (младший байт)	Количество регистров (старший байт)	Количество регистров (младший байт)

Поле Data PDU содержит поля «Начальный адрес», указывающий с какого регистра начинать чтение, и «Количество регистров», указывающее, сколько регистров следует прочитать.

PDU ответа в случае выполнения без ошибок:

Функция **0x03/0x04**	Количество байт данных в ответе	1-ый регистр (старший байт)	1-ый регистр (младший байт)	Байты регистров 2,3…N

В случае успешного выполнения в ответе присутствует содержимое запрошенных регистров. Поле «Количество байт данных в ответе» будет равно количеству запрошенных регистров, умноженному на два. Прочитанное содержимое регистров начинается с байта, следующего за полем «Количество байт данных в ответе».

PDU ответа при возникновении ошибки:

Функция 0x83/0x84 (установлен старший бит)	Код ошибки

Для информирования ведущего о том, что операция не выполнена или выполнена с ошибкой, прибор устанавливает старший бит поля «Функция» в ответе. Байт, следующий за полем «Функция», будет содержать код ошибки (значения кодов ошибок приведены в Приложении 3).

2.2 Функция Modbus 06 (0x06 hex) (Write Single Register)

Функция предназначена для записи двоичного содержимого одного шестнадцатиразрядного регистра прибора.

PDU запроса:

Функция 0x06	Адрес (старший байт)	Адрес (младший байт)	Значение (старший байт)	Значение (младший байт)

Поле Data PDU запроса содержит поля «Адрес», указывающий в какой регистр выполняется запись, и значение записываемого регистра.

PDU ответа в случае выполнения без ошибок:

Функция 0x06	Адрес (старший байт)	Адрес (младший байт)	Значение (старший байт)	Значение (младший байт)

В случае успешного выполнения PDU ответа содержит копию первых пяти байт PDU запроса.

PDU ответа при возникновении ошибки:

Функция 0x86 (установлен старший бит)	Код ошибки

Для информирования ведущего о том, что операция не выполнена или выполнена с ошибкой, прибор устанавливает старший бит поля «Функция» в ответе. Байт, следующий за полем «Функция», будет содержать код ошибки (значения кодов ошибок приведены в Приложении 3).

2.3 Функция Modbus 16 (0x10 hex) (Write Multiple Registers)

Функция предназначена для записи двоичного содержимого шестнадцатиразрядных регистров прибора.

PDU запроса:

Функция 0x10	Нач-ый адрес (старший байт)	Нач-ый адрес (младший байт)	Кол-во рег-ов (старший байт)	Кол-во рег-ов (младший байт)	Кол-во байт для записи	1-ый регистр (старший байт)	1-ый регистр (младший байт)	Байты рег-ов 2,3…N

Поле Data PDU запроса содержит поля «Начальный адрес», указывающий с какого регистра начинать запись, «Количество регистров», указывающее, сколько регистров следует записать, «Количество байт для записи» и непосредственно значения записываемых регистров.

PDU ответа в случае выполнения без ошибок:

Функция 0x10	Нач-ый адрес (старший байт)	Нач-ый адрес (младший байт)	Кол-во рег-ов (старший байт)	Кол-во рег-ов (младший байт)

В случае успешного выполнения PDU ответа содержит копию первых пяти байт PDU запроса.

PDU ответ при возникновении ошибки:

Функция 0x90 (установлен старший бит)	Код ошибки

Для информирования ведущего о том, что операция не выполнена или выполнена с ошибкой, прибор устанавливает старший бит поля «Функция» в ответе. Байт, следующий за полем «Функция», будет содержать код ошибки (значения кодов ошибок приведены в Приложении 3).

2.4 Функция Modbus 20 (0x14 hex) (Read File Record)

Функция предназначена для чтения регистров файла. Файл организован как набор записей с номерами от 0000 до 9999. Функция может читать несколько различных групп регистров. Группы могут быть непоследовательными, но регистры внутри группы должны быть последовательными.

PDU запроса:

Функция 0x14	Длина запроса (7÷245)	Группа X, тип (всегда равен 6)	Группа X, номер файла (старш. байт)	Группа X, номер файла (младш. байт)	Группа X, номер записи (старш. байт)	Группа X, номер записи (младш. байт)	Группа X, длина записи (старш. байт)	Группа X, длина записи (младш. байт)	Группа X+1, тип (всегда равен 6)	Группа X+1, номер файла (старш. байт)	Группа X+1, номер файла (младш. байт)	…

Поле Data PDU запроса содержит:

- Поле «длина запроса»= 7 * N, где N – количество групп запрашиваемых регистров;
- Описание групп запрашиваемых регистров, каждое из которых имеет поля:
  - Тип. Всегда равен 6;
  - Номер файла;
  - Номер записи внутри файла;
  - Длина записи. Определяет количество запрашиваемых регистров группы.

PDU ответа в случае выполнения без ошибок:

Функция 0x14	Длина ответа	Группа X, длина (байт)	Группа X, тип (всегда равен 6)	Группа X, регистр 1 (старш. байт)	Группа X, регистр 1 (младш. байт)	Группа X, регистр 2 (старш. байт)	Группа X, регистр 2 (младш. байт)	...	Группа X+1, длина (байт)	Группа X+1, тип (всегда равен 6)	Группа X+1, регистр 1 (старш. байт)	Группа X+1, регистр 1 (младш. байт)	Группа X+1, регистр 2 (старш. байт)	Группа X+1, регистр 2 (младш. байт)	…

Поле Data PDU запроса содержит:

- Поле «длина ответа»=((Группа X, длина) + 2)+((Группа X+1, длина)+2)+…;
- Группы регистров, каждая из которых имеет:
  - Длина группы (байт);
  - Тип. Всегда равен 6;
  - Регистры данных.

PDU ответ при возникновении ошибки:

Функция 0x94 (установлен старший бит)	Код ошибки

Для информирования ведущего о том, что операция не выполнена или выполнена с ошибкой, прибор устанавливает старший бит поля «Функция» в ответе. Байт, следующий за полем «Функция», будет содержать код ошибки (значения кодов ошибок приведены в Приложении 3).

3 Организация доступа к данным

3.1 Используемые типы данных и условные обозначения

- unsigned char –беззнаковое целое число (8 бит);
- signed char –знаковое целое число (8 бит);
- unsigned short –беззнаковое целое число (16 бит);
- signed short –знаковое целое число (16 бит);
- unsigned long –беззнаковое целое число (32 бита);
- signed long –знаковое целое число (32 бита);
- unsigned __int64 –беззнаковое целое число (64 бита);
- signed __int64 –знаковое целое число (64 бита);
- float – вещественное число одинарной точности (32 бита) с плавающей запятой, соответствующее стандарту «IEEE 754»;
- double – вещественное число двойной точности (64 бита) с плавающей запятой, соответствующее стандарту «IEEE 754»;
- R/O – доступ только на чтение;
- R/W – доступ на чтение и запись (безусловный);
- PAS - доступ на чтение, доступ на запись при предъявлении пароля;
- KEY - доступ на чтение, доступ на запись при наличии электронного ключа;
- JUM - доступ на чтение, доступ на запись при установленной перемычке «Защита»;
- KEY+JUM - доступ на чтение, доступ на запись при наличии электронного ключа и при установленной перемычке «Защита»;

3.2 Регистровый доступ и характеристики параметров

Доступ к текущим и настроечным параметрам прибора реализован через функции чтения и записи регистров – переменных, имеющих тип шестнадцатиразрядное беззнаковое целое. При организации регистрового доступа делается допущение, что все многообразные структуры данных располагаются в памяти, элементарной ячейкой которой является один шестнадцатиразрядный регистр типа «беззнаковое целое». Физически данные могут находиться в совершенно разных участках памяти прибора и даже в разных типах памяти (оперативная, энергонезависимая и т.д.), но для системы верхнего уровня данные «выглядят» как единое адресное пространство. В этом случае все доступные данные можно представить как массив шестнадцатиразрядных регистров, каждый из которых характеризуется номером в массиве (далее адресом). Каждый параметр прибора может занимать часть регистра, весь регистр целиком или несколько регистров. Таким образом, параметр характеризуется собственным типом и расположением внутри массива регистров.

Доступ к архивам организован через стандартные функции чтения файлов, предусмотренные протоколом «Modbus».

3.3 Порядок хранения и передачи байт данных

Для чтения и записи регистров в стандарте «Modbus» предусмотрены специальные функции, которые оперируют содержимым шестнадцатиразрядных регистров. Эти функции предполагают, что прибор хранит данные только типа шестнадцатиразрядное беззнаковое целое и ничего не «знают» о тех типах данных, которыми действительно представлены параметры прибора. Таким образом, получается, что в приборе данные хранятся в некоем исходном формате, а передаются по сети в виде набора шестнадцатиразрядных регистров. При передаче данных, чей размер в исходном формате превышает 16 бит (long, float, double и т.д.), используются несколько последовательных регистров. При этом младшие слова передаются в первую очередь, старшие - в последнюю. Т.о., для преобразования к порядку байт, естественному для

платформы PC, требуется для каждого прочитанного/записываемого регистра изменить порядок байт.

Пример размещения данных для типа **long** (MSB-most significant byte, LSB-least significant byte):

B3	B2	B1	B0
MSB			LSB

Регистр	Регистр A0		Регистр A1	
Порядок передачи	первый			последний
Байт	B1	B0(LSB)	B3	B2(MSB)

Пример размещения данных для типа **float**:

B3	B2	B1	B0
SEEEEEEE	EMMMMMMM	MMMMMMMM	MMMMMMMM

Регистр	Регистр A0		Регистр A1	
Порядок передачи	первый			последний
Байт	B1	B0(LSB)	B3	B2(MSB)

Пример размещения данных для типа **double**:

B7	B6	B5-B1	B0
SEEEEEEE	EEEEMMMM	MMMMMMMM	MMMMMMMM

Регистр	Регистр A0		Регистр A1		Регистр A2		Регистр A3	
Порядок передачи	первый							последний
Байт	B1	B0(LSB)	B3	B2	B5	B4	B7(MSB)	B6

3.4 Чтение/запись текущих и настроечных параметров

Чтение/запись текущих и настроечных параметров производится при помощи функций чтения/записи регистров (функции чтения 0x03, 0x04, функции записи 0x06, 0x10). Параметры, доступные только на чтение, могут быть прочитаны только функцией 0x04. Параметры, доступные как на чтение, так и на запись, могут быть прочитаны обеими функциями по одинаковым адресам. Распределение переменных в адресном пространстве описано в разделе Карта переменных.

3.5 Организация архивов и считывание архивных данных

Прибор содержит архивы следующих типов:

- События;
- Минутный архив;
- Часовой архив;
- Суточный архив;

Архивы имеют кольцевую структуру, т.е. при полном заполнении архива новые записи пишутся на место самых старых.

Архивы хранятся в приборе в виде файлов. Каждый файл содержит дескриптор файла (в первой записи), а также записи, представляющие одно событие или один «временной срез» данных. Для чтения содержимого архивов используется функция чтения файлов (функция 0x14).

Каждый файл в первой записи (номер 0) хранит дескриптор файла. Остальные записи представляют данные файла. Дескриптор файла содержит:

- Длина первой записи (длина дескриптора, 2 байта);

- Тип первой записи (тип дескриптора, 2 байта, всегда равен 1);
- Длина файла в количестве записей (без учета первой записи, 2 байта);
- Длина каждой записи (кроме первой записи, 2 байта);
- Тип содержимого файла (2 байта). Может принимать значения: 4 – файл событий, 3 или 5 – файл измерительного архива (минутный, часовой или суточный).

Алгоритм доступа к файлам:

1. Чтение записи с номером 0 из файла с номером 1 (получение дескриптора файла). Если прибор вернул ошибку с кодом 2, значит, файлов больше нет;
2. Анализ дескриптора на предмет длины основных записей и длины файла;
3. Чтение оставшихся записей файла и их интерпретация;
4. Повторение пунктов 1-3 для следующего файла.

Каждая архивная запись, независимо от типа файла, имеет следующую структуру:

- Номер записи. Счетчик, который увеличивается на единицу при формировании новой записи внутри данного файла. Имеет тип unsigned __int64;
- Массив из 52 байт. В нем размещается структура данных архивной записи (см. Интерпретация архивов);
- Контрольная сумма записи. Имеет тип unsigned long и рассчитывается по алгоритму Crc32 (Проверка целостности данных должна выполняться над всем блоком данных длиной 64 байта включая контрольную сумму. Функция проверки контрольной суммы в Приложении 4).

Алгоритм оптимизированного считывания архива приведен в Приложении 6.

3.5.1 Интерпретация минутного, часового и суточного архивов

Минутный, часовой и суточный архивы ведутся всегда. Хронологическая метка записи содержит время наработки прибора и может содержать дату/время при наличии в приборе часов реального времени (опция). Наличие часов реального времени можно определить по значению переменной «Регистр состояния прибора» (см. Карта переменных).

Минутный архив хранится в файле номер 6, часовой в файле номер 7, суточный в файле номер 8.

Необходимо прочитать все записи файла, выстроить их по признаку нарастания значения поля «Номер записи» и интерпретировать те записи, у которых сходится контрольная сумма.

Архивная запись содержит следующие поля:

- Интегральное время наработки на момент записи (минут). Тип unsigned long;
- Показания часов реального времени на момент архивирования. Если плата не имеет часов реального времени, то анализировать данное поле не следует. Значение представлено в виде массива unsigned char из 6-ти элементов. Каждый байт содержит число в формате BCD (функции преобразования представлены в Приложении 5). Порядок следования: секунда, минута, час, день, месяц, год;
- Интегральное значение **V+** (м3). Тип double;
- Интегральное значение **V-** (м3). Тип double;
- Мгновенное значение расхода на момент архивирования (м3/ч). Тип float;

- Коды ошибок. Тип unsigned long. Коды ошибок и наименования ошибок описаны в руководстве по эксплуатации на прибор. Младшие 12 бит (биты 0-11) хранят описание самой серьезной ошибки за интервал архивирования. При этом биты 0-7 хранят значение ошибки, биты 8-11 – расширение ошибки. Ненулевое значение в битах 0-7 означает наличие ошибки. В случае наличия ошибки следует проанализировать расширение ошибки (биты 8-11). Код ошибки должен быть представлен в виде «X.Y», где X - значение ошибки в шестнадцатеричном представлении, Y - расширение ошибки в шестнадцатеричном представлении. Если расширение ошибки равно 0, то код ошибки представляется только значением ошибки в виде «X». Старшие 2 байта (биты 16-31) хранят признаки наличия ошибок и флаги состояний прибора. Установленный бит означает наличие ошибки или установленного флага. Возможны сочетания следующих значений:
  - (0x00000001 << 30) – должен отображаться как «RTC»;
  - (0x00000001 << 29) – должен отображаться как «MG»;
  - (0x00000001 << 28) – должен отображаться как «W»;
  - (0x00000001 << 27) – должен отображаться как «IN»;
  - (0x00000001 << 26) – должен отображаться как «IC»;
  - (0x00000001 << 25) – должен отображаться как «MX»;
  - (0x00000001 << 24) – должен отображаться как «OF»;
  - (0x00000001 << 20) – должен отображаться как «GR»;
  - (0x00000001 << 19) – должен отображаться как «WC»;
  - (0x00000001 << 18) – должен отображаться как «FR»;
  - (0x00000001 << 17) – должен отображаться как «RR»;
  - (0x00000001 << 16) – должен отображаться как «K».
- Интегральное время наработки при ошибках на момент записи (минут). Тип unsigned long;
- Минимальное значение расхода за интервал архивирования (м3/ч). Тип float;

Максимальное значение расхода за интервал архивирования (м3/ч). Тип float;

- Сопротивление жидкости (кОм). Тип float. Присутствует только в архивной записи типа 5 (см. описание дескриптора файла).

3.5.2 Интерпретация архива событий

Архив событий может содержать более одного файла. Поэтому следует прочитать все записи всех файлов архива событий. Затем записи, у которых сходится контрольная сумма отсортировать по времени и затем интерпретировать.

Архивная запись содержит следующие поля:

- Тип события. Тип unsigned char. Определяет тип произошедшего события. Может принимать следующие значения:
  - 0x00 - Нет события;
  - 0x02 – Изменение параметра;
  - 0x03 – Вкл. питания;
  - 0x04 – Перезапуск;
  - 0x10 - Ошибка стека;
  - 0x11 – Ошибка измерения;
  - 0x12 – Ошибка часов;
  - 0x13 – Первый запуск;
- Интегральное время наработки на момент записи (минут). Тип unsigned long;
- Внешняя временная метка. Тип signed long. Представляет количество секунд, прошедшее от 01.01.1970 00:00:00;
- Показания часов реального времени на момент архивирования. Если плата не имеет часов реального времени, то анализировать данное поле не следует.

Значение представлено в виде массива unsigned char из 6-ти элементов. Каждый байт содержит число в формате BCD (функции преобразования представлены в Приложении 5). Порядок следования: секунда, минута, час, день, месяц, год;

- Тип значения. Тип unsigned char. Определяет тип данных значений, хранимых в полях «Старое значение» и «Новое значение». Может принимать следующие значения:
  - 0 – нет значения;
  - 1 – unsigned char;
  - 2 – signed char;
  - 3 – unsigned short;
  - 4 – signed short;
  - 5 – unsigned long;
  - 6 – signed long;
  - 7 - float
  - 8 – unsigned __int64;
  - 9 – signed __int64;
  - 10 – double;
  - 17 – дата/время. Массив из 6-ти байт в формате BCD (секунда, минута, час, день, месяц, год).
- Тип переменной для события «Изменение параметра». Тип unsigned short. Содержит значение начального адреса регистра «Modbus» измененного параметра:
- Старое значение. Массив из 8-ми байт. Содержит значение в соответствии с типом, указанным в поле «Тип значения»;
- Новое значение. Массив из 8-ми байт. Содержит значение в соответствии с типом, указанным в поле «Тип значения»;
- Идентификатор ключа доступа. Массив типа unsigned char из 4-х элементов;
- Коды ошибок. Тип unsigned long. Интерпретация аналогична интерпретации одноименного поля в минутном архиве;
- Номер события. Тип unsigned long. Счетчик, который увеличивается на единицу при формировании новой записи (сквозная нумерация событий, общая для всех файлов архива событий).

4 Карта переменных

Название	Адрес	Размер (байт)	Тип	Доступ	Примечание
Тип прибора	**0**	2	unsigned short	R/O	0x170E – Питерфлоу Т1
Версия программы	**1**	2	unsigned short	R/O	старший байт версия, младший-редакция
Порядок байт и слов	**3**	2	unsigned short	R/O	Не используется
CRC прошивки	**4**	2	unsigned short	R/O	Индицировать в шестнадцатеричном виде
Регистр состояния прибора	**6**	2	unsigned short	R/O	Младший бит определяет наличие часов реального времени
Калибр. коэффициент A	**7**	4	float	R/O	
Калибр. коэффициент B	**9**	4	float	R/O	
Текущий расход (м3)	**100**	4	float	R/O	
Код ошибки	**102**	4	unsigned long	R/O	Интерпретация аналогична одноименному полю минутного архива
Интеграл V+ (м3)	**104**	8	double	R/O	
Интеграл V- (м3)	**108**	8	double	R/O	
Время наработки (мин.)	**112**	4	unsigned long	R/O	
Интеграл V+ ($м^3$)	**182**	4	unsigned long	R/O	Целая часть параметра с адресом 104
Интеграл V- ($м^3$)	**184**	4	unsigned long	R/O	Целая часть параметра с адресом 108
Уровень доступа	**200**	2	unsigned short	R/W	Запись сбрасывает доступ в ноль. Битовая маска: бит 0-PAS; бит 1-KEY; 2-JUM
Авторизация	**201**	20	unsigned char[16] + unsigned long	R/W	Байты 0..15: строковый пароль + метка времени
Предел коррекции часов реального времени (сек.)	**410**	2	unsigned short	R/O	Разрешенные пределы коррекции часов
Коррекция часов реального времени (сек.)	**411**	2	signed short	PAS	
Уставка компаратора F1 (м3/ч)	**420**	4	float	PAS	
Уставка компаратора F2 (м3/ч)	**430**	4	float	PAS	
Сетевой адрес	**440**	2	unsigned short	PAS	
Тип выхода F1	**500**	2	unsigned short	KEY	
Тип выхода F2	**510**	2	unsigned short	KEY	
Вес импульса (л)	**520**	4	float	KEY	
ДУ (мм)	**540**	2	unsigned short	KEY+JUM	
Класс расходомера	**550**	2	unsigned short	KEY+JUM	Символы 'A', 'B', 'C', ' '
Максимальный расход (м3/ч)	**560**	2	unsigned short	KEY+JUM	
Серийный номер	**570**	4	unsigned long	KEY+JUM	
Уровень отсечки	**580**	4	float	KEY*	

Часы реального времени	**590**	6	unsigned char[6]	KEY	Сек., мин., час, день, месяц, год в формате BCD

Примечание (*): уровень доступа к параметру «Уровень отсечки» для приборов с версией ПО 3.12 KEY+JUM.

Переменная «**Регистр состояния прибора**» содержит информацию о наличии на плате часов реального времени. Установленный младший бит говорит о наличии часов на плате.

Переменную «**Код ошибки**» следует интерпретировать так же, как описано для одноименного поля минутного архива.

Переменная «**Уровень доступа**» позволяет прочитать текущий уровень доступа к изменению параметров. Три младших бита кодируют наличие доступа. Бит 0 – PAS, бит 1 – KEY, бит 2 – JUM.

Переменная «**Авторизация**» необходима для передачи пароля при попытке изменения параметра, требующего доступа по паролю. Если текущий уровень доступа менее уровня PAS, перед изменением параметра система верхнего уровня должна выполнить запись в переменную «Авторизация». В байты 0÷15 массива авторизации копируется строковый пароль (с завершающим нулем на конце), в поле временной метки копируется значение локального времени (в формате количества секунд, прошедших с 01.01.1970 00:00:00). Переменная «Авторизация» должна быть записана единым массивом (20 байт).

Переменные «Тип выхода Fx» могут принимать значения:

- Расход прямой и обратный: (0 | (0<<7) | (1<<6) | (1<<5));
- Расход прямой и обратный, инверсный выход: (0 | (1<<7) | (1<<6) | (1<<5));
- Расход прямой: (0 | (0<<7) | (0<<6) | (1<<5));
- Расход прямой, инверсный выход: (0 | (1<<7) | (0<<6) | (1<<5));
- Расход обратный: (0 | (0<<7) | (1<<6) | (0<<5));
- Расход обратный, инверсный выход: (0 | (1<<7) | (1<<6) | (0<<5));
- Компаратор: (1 | (0<<7));
- Компаратор, инверсный выход: (1 | (1<<7));
- Ошибка: (2 | (0<<7)).

Приложение 1. Функция расчета контрольной суммы LRC

Пример функции расчета контрольной суммы кадра на языке СИ:

```
   unsigned char Lrc(unsigned char * pSrc, int length) {
     unsigned char locLrc=0;
     for(int i=0;i<length;i++)
            locLrc += *(pSrc+i);
     return locLrc = ~locLrc + 1;
}
```

где:

- pSrc – указатель на буфер, содержащий сообщение;
- length – количество байт данных, для которых требуется произвести подсчет LRC.

Приложение 2. Функции преобразования в ASCII и обратно

Ниже приведены примеры на языке СИ функций преобразования из ASCII формата в двоичный и обратного преобразования из двоичного в ASCII.

```
const       unsigned char CharToBin[23]={
0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,0,10,11,12,13,14,15};
const       char    BinToChar[16]={
'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

char TwoASCIItoBIN(char *cptr, unsigned char *bptr) {
     char    ca,i;
     unsigned char  cb;

     cb=0;
     for(i=0; i<2; i++) {
            ca=*cptr++;
            cb<<=4;
            if((ca >= '0') && (ca <= '9') || (ca >= 'A') && (ca <= 'F'))
                   cb|=CharToBin[ca-0x30];
            else
                   return(0);
     }
     *bptr=cb;
     return(1);
}
```

где:

- cptr – указатель на буфер, содержащий символы ASCII;
- bprt - указатель на буфер, куда записываются двоичные байты.

```
void BINtoTwoASCII(unsigned char *bptr, char *cptr) {
     unsigned char  cb;

     cb=*bptr;
     *cptr=BinToChar[(cb>>4) & 0x0F];
     cptr++;
     *cptr=BinToChar[cb & 0x0F];
}
```

где:

- bptr – указатель на буфер, содержащий двоичные байты;
- cprt - указатель на буфер, куда записываются символы ASCII.

Приложение 3. Коды ошибок, возвращаемые прибором

0 - нет ошибок;
1 - недопустимая функция;
2 - недопустимый адрес в запросе;
3 - недопустимые значения в поле данных запроса;
4 – непоправимая ошибка возникла во время выполнения операции;
5 – подтверждение выполнения операции (не используется);
6 - запрос не может быть обработан сейчас, необходимо повторить запрос позднее;

Приложение 4. Функция расчета контрольной суммы Crc32

```
#define     CRC32_Pre()     0xFFFFFFFF
#define     CRC32_Pnom()    0xEDB88320
#define     CRC32_XOR()     0xFFFFFFFF
#define     CRC32_Check(_C)         (_C == 0xDEBB20E3)

unsigned char CRC32(void * pBuf, unsigned long len) {
     unsigned long  j;
     unsigned long _CRC;

     _CRC = CRC32_Pre();
     for (j=0; j < len; j++) {
            _CRC = CRC32_Calc_Byte(*((BYTE*)(pBuf)+j), _CRC);
     }

     if (CRC32_Check(_CRC)) {
            return 1;
     } else {
            return 0;
     }
}

unsigned long CRC32_Calc_Byte(unsigned char _D, unsigned long _C) {
     unsigned char  _i = 8;
     _C ^= (_D & 0xFF);
     do {
            if (_C & 1) { _C=(_C>>1)^CRC32_Pnom(); } else { _C>>=1; }
     } while (--_i);
     return (_C);
}
```

Приложение 5. Функции преобразования в BCD и обратно

```
   unsigned char BinByteToBcdByte(unsigned char Src) {
        unsigned char res;
        res = (unsigned char)( ((Src/10)<<4) | (Src % 10) );
        return res;
   }

   unsigned char BcdByteToBinByte(unsigned char Src) {
        unsigned char res;
        res = (unsigned char)( ((Src>>4) * 10) + (Src & 0x0f) );
        return res;
}
```

Приложение 6. Оптимизация алгоритма считывания архивов

Доступ к архивам прибора реализован через считывание файлов по записям без возможности указания желаемой даты архивной записи. Это предполагает считывание всего файла архива целиком в каждом сеансе связи. При регулярном считывании архива это приводит к дополнительным временным затратам на получение архивных записей, которые были прочитаны во время предыдущих сеансов связи. Для экономии времени считывания алгоритм может быть оптимизирован на стороне программного обеспечения верхнего уровня, если оно имеет возможность сохранять прочитанные данные в каком-нибудь хранилище, например, базе данных.

Алгоритм основан на том, что база данных (БД) хранит как содержимое архивных записей, так и служебную информацию о записях (номер файла и индекс записи, из которых запись была прочитана).

Если для данного файла в БД нет информации, то производится считывание всех имеющихся в приборе записей. При считывании каждой записи имеет смысл проверять контрольную сумму. При последовательном несовпадении контрольной суммы 5 и более раз можно прекратить считывание данного файла, т.к. данная ситуация может указывать на то, что файл не заполнен полностью.

Если БД содержит информацию о предыдущем считывании, то необходимо произвести считывание из прибора, указав номер файла и индекс записи такие же, как в последней записи БД (т.о. выполняется контрольное считывание последней известной записи).

Несовпадение инкрементного номера записи (тип unsigned __int64) или контрольной суммы прочитанной из прибора записи с аналогичными полями записи из БД говорит о том, что архивная запись была переписана с момента последнего считывания (архив «закольцевался»). В этом случае следует произвести считывание данных так, как будто БД не содержала информации об архиве.

В противном случае считывание данных выполняется, начиная со следующей записи. Считывание продолжается до возникновения одного из событий:

- Прочитаны все записи файла;
- Прочитано 5 или более записей с несовпадающей контрольной суммой;
- Прочитана запись, имеющая значение инкрементного номер меньше максимального известного значения инкрементного номера записи (хранящегося в БД или прочитанного во время текущего сеанса связи). Данная ситуация говорит о том, что прочитана запись, которая уже должна находиться в БД.

Приложение 7. Функция расчета контрольной суммы Cyclical Redundancy Checking (CRC16)

Расчет контрольной суммы кадра RTU может выполняться по следующему алгоритму (текст на языке программирования СИ):

```
WORD Crc16(BYTE *Data, ULONG size) {
    union
    {BYTE b[2]; unsigned short w;} Sum;
    char shift_cnt;
    BYTE *ptrByte;
    ULONG  byte_cnt = size;

    ptrByte=Data;
    Sum.w=0xffffU;
    for(; byte_cnt>0; byte_cnt--) {
        Sum.w=(unsigned short)((Sum.w/256U)*256U+((Sum.w%256U)^(*ptrByte++)));
        for(shift_cnt=0; shift_cnt<8; shift_cnt++) {
            if((Sum.w&0x1)==1)
                Sum.w=(unsigned short)((Sum.w>>1)^0xa001U);
            else
                Sum.w>>=1;
        }
    }
    return Sum.w;
}
```

Пусть какое-либо сообщение, имеющего длину N, записано в массиве Data[N+2] типа unsigned char. Тогда для этого сообщения контрольную сумму следует формировать следующим образом:

```
WORD CheckSumm = Crc16(Data, N);
Data[N] = CheckSumm;
Data[N+1] = CheckSumm>>8;
```