ООО «НПП Электромеханика»

Преобразователь измерительный ПИ849Ц. Описание протокола обмена данными стандарта ГОСТ Р МЭК-870-5-1-95 формата FT3

19 сентября 2013 г.

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). ПИ849Ц 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 0x64
DataLen	Длина данных	1 байт	0x00
ControlByte	Контрольный байт	1 байт	0x00
Address	Адрес	2 байта, младший байт передается первым	0x0000– 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 необходимо 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.

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).

Кадр ответа с одним блоком данных имеет вид:

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

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 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– 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 ИЗМЕРИТЕЛЬНОГО ПРЕОБРАЗОВАТЕЛЯ ПИ849Ц

СПИСОК КОМАНД ПИ849Ц

Код команды	Наименование
0x01	Подготовка к записи данных во флеш-память ПИ
0x02	Изменение адреса ПИ
0x03	Чтение адреса ПИ
0x04	Запись паролей
0x05	Телеуправление
0x06	Разрешить запись уставок
0x07	Получить данные (шаблон)
0x08	Получить типизацию ПИ
0x12	Запись конфигурации уставок
0x14	Сброс счетчиков импульсов
0x15	Установка скорости обмена данными
0x16	Фиксировать данные
0x17	Установить дату и время
0x18	Прочитать дату и время
0x19	Установить метки смены сезонов
0x1A	Прочитать метки смены сезонов
0x1B	Синхронизация времени
0x1C	Чтение конфигурации уставок/ТУ
0x1D	Очистка журналов
0x1E	Получить конфигурацию журналов
0x1F	Получить данные журналов
0x2B	Записать конфигурацию устройства
0x2C	Прочитать конфигурацию устройства
0x2E	Запись конфигурации ТУ
0x2F	Получить точные данные
0x30	Запись конфигурации в журнал профилей
0x31	Чтение конфигурации журнала профилей
0x32	Запуск журнала профилей
0x33	Останов журнала профилей

0X01 ПОДГОТОВКА К ЗАПИСИ ДАННЫХ ВО ФЛЕШ-ПАМЯТЬ ПИ

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 ПЗУ ПИ849Ц.

0X02 ИЗМЕНЕНИЕ АДРЕСА ПИ

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память устройства

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 ПИ

Параметры: нет

Возвращаемые данные: считанный адрес в поле Address структуры PKTREADHEAD.

0X04 ЗАПИСЬ ПАРОЛЕЙ

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память ПИ

Параметры	Байты структуры PARAMETRS
Тип пароля	P1
Старый пароль	P2-P5, младший байт передается первым
Новый пароль	P6-P9, младший байт передается первым

Константы типа паролей:

Константа	Тип пароля
0x01	Пароль для изменения уставок
0x02	Пароль для сброса счетчиков

Возвращаемые данные: нет

0X05 ТЕЛЕУПРАВЛЕНИЕ

Параметры	Байты структуры PARAMETRS
Состояние ТУ (SETTU)	P1-P5, младший байт передается первым
Защитный код (PCODE)	P6=0x9C P7=0x39

Возвращаемые данные: нет

Примечание: Команда 0x05 "Телеуправление" требует передачи защитной кодовой комбинации.

0X06 РАЗРЕШИТЬ ЗАПИСЬ УСТАВОК

Параметры	Байты структуры PARAMETRS
Пароль для изменения уставок	P1-P4
0	P5
Пароль для сброса счетчиков	P1-P4
1	P5

Возвращаемые данные: нет

0X07 ПОЛУЧИТЬ ДАННЫЕ (ШАБЛОН)

Параметры	Байты структуры PARAMETRS
Маска данных	P1-P3, младший байт передается первым
Управляющий контрольный байт (CONTROLBYTE)	P9

Константы для маски запроса данных	Наименование запрашиваемых данных	Структура
0x000001	Мгновенные значения: Фаза A	PHASE
0x000002	Мгновенные значения: Фаза B	PHASE
0x000004	Мгновенные значения: Фаза C	PHASE
0x000008	Интегрированные значения: Фаза A	PHASE
0x000010	Интегрированные значения: Фаза B	PHASE

Константы для маски запроса данных	Наименование запрашиваемых данных	Структура
0x000020	Интегрированные значения: Фаза С	PHASE
0x000040	Счетчики ТС	ENERGY
0x000080	Частота, состояние ТС, ТУ, температура	FREQDAT
0x000100	Фиксированные данные	FIXDATA
0x000200	2 предыдущих состояния ТС	PREVTC
0x000400	Метка времени измерения	TIMERSTAMP
0x000800	Два байта состояния процессоров	SENSORSTATE
0x001000	Уставки	ACTSTAT
0x002000	Мощность суммарная активная, мощность суммарная реактивная	POWERSUMM
0x004000	Фиксированная частота, ТУ, ТС	FIXDATA2
0x008000	Линейное напряжение между фазами, ток и напряжение нулевой последовательности	Add_Val
0x010000	Интегрированные значения линейных напряжений между фазами, тока и напряжения нулевой последовательности	Add_Val
0x020000	Средние значения тока и напряжения по фазам	Aver_Val
0x040000	Интегрированные средние значения тока и напряжения по фазам	Aver_Val

Примечание: команда 0x07 "Получить данные" принимает в виде параметра двухбайтовую маску, определяющую, какие данные будут переданы контроллеру верхнего уровня. Порядок передачи запрошенных структур данных определяется по возрастанию величины маски. Маска может иметь любую возможную комбинацию по логической операции "ИЛИ" из указанного набора констант.

Возвращаемые данные:

- Данные по фазе
- Счетчики ТС
- Частота и другие данные
- Фиксированные данные
- Состояние процессоров
- Уставки
- Напряжение между фазами, ток и напряжение нулевой последовательности

Примечание: здесь и далее, если это не оговорено особо, для пересчета полученных значений следует использовать таблицу F1.

0X08 ПОЛУЧИТЬ ТИПИЗАЦИЮ ПИ

Параметры: нет

Возвращаемые данные: информация о ПИ IPCINFO

Примечание: команда 0x08, выдающая информацию о ПИ, может применяться в том случае, если контроллер верхнего уровня обслуживает разные типы ПИ. Серия ПИ возвращается в шестнадцатеричном формате в поле Model структуры IPCINFO (с измененным порядком байт), а номер модели – в поле ModNumber также в шестнадцатеричном формате.

0X12 ЗАПИСЬ КОНФИГУРАЦИИ УСТАВОК

Предварительные команды:

- 0x01 Подготовка к записи данных во флеш-память ПИ,
- 0x06 Разрешить запись уставок.

Примечание: в предварительной команде 0x06 необходимо передавать пароль на запись уставок (параметр P5=0).

Параметры	Байты структуры PARAMETRS
Номер уставки	P1
Уставка (UST_CONFIG.)	P2-P9

Примечание: команда 0x12 "Запись конфигурации уставок" принимает номер уставки и данные для уставки с указанным номером.

Нумерация уставок начинается с нуля, максимальное количество – 16.

Возвращаемые данные: нет

Возвращаемые данные: нет

0X14 СБРОС СЧЕТЧИКОВ ИМПУЛЬСОВ

Параметры	Байты структуры PARAMETRS
Пароль для сброса счетчиков	P1-P4

Возвращаемые данные: нет

0X15 УСТАНОВКА СКОРОСТИ ОБМЕНА ДАННЫМИ

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память ПИ

Параметры	Байты структуры PARAMETRS
Константа скорости	P1
Модификатор команды	P2

Константы скоростей ПИ (SENSORSPEED)

Константа	Скорость ПИ
0x01	19200
0x02	9600 при первом включении
0x03	4800
0x04	2400
0x05	1200
0x11	38400
0x12	57600
0x13	115200

Возвращаемые данные: нет

Примечание: команда 0x15 "Установка скорости обмена" для ПИ, имеющих два канала RS-485, может принимать модификатор (параметр P2), который позволяет изменить скорость двух каналов связи ПИ одновременно. Если параметр модификатора не равен нулю - скорость будет изменена для двух каналов сразу, в противном случае - только для канала, по которому пришел запрос.

0X16 ФИКСИРОВАТЬ ДАННЫЕ

Параметры	Байты структуры PARAMETRS
Таймерная метка	P1-P4

Возвращаемые данные: нет

Примечание: Команда 0x16 "Фиксировать данные" принимает четырехбайтовую метку времени, которая в произвольном формате может быть передана для ПИ контроллером верхнего уровня. При чтении фиксированных данных данная метка времени будет возвращена ПИ обратно, смотрите FIXDATA .

0X17 УСТАНОВИТЬ ДАТУ И ВРЕМЯ

Параметры	Байты структуры PARAMETRS
Дата и время ПИ (SENSORTIME)	P1-P9

Возвращаемые данные: нет

Примечание: установка времени осуществляется с точностью до секунд. По приходу команды значение параметра SubSec структуры SENSORTIME сбрасывается. Подавать команду следует при переходе через секунду с учетом скорости передачи данных.

0X18 ПРОЧИТАТЬ ДАТУ И ВРЕМЯ

Параметры	Байты структуры PARAMETRS
Тип запрашиваемого времени	P1

Код	Тип запрашиваемого времени
0	Текущее время ПИ
1	Время последнего включения ПИ
2	Время последнего выключения ПИ

Возвращаемые данные: дата и время ПИ (SENSORTIME).

Для ПИ849Ц 0-й бит байта Season (SENSORTIME) указывает на летнее или зимнее время (0 – зимнее, 1 – летнее).

0X19 УСТАНОВИТЬ МЕТКИ СМЕНЫ СЕЗОНОВ

Параметры	Байты структуры PARAMETRS
Сезоны (SENSORSEASON)	P1-P6

Возвращаемые данные: нет

0X1A ПРОЧИТАТЬ МЕТКИ СМЕНЫ СЕЗОНОВ

Параметры: нет

Возвращаемые данные: SENSORSEASON

0X1B СИНХРОНИЗАЦИЯ ВРЕМЕНИ

Параметры: нет

Возвращаемые данные: нет

Команда широковещательная – отсылается по адресу 0xFF (всем подключенным ПИ). По этой команде сбрасываются счетчики секунд и миллисекунд. Если в момент получения команды значение

счетчика секунд меньше 30, то происходит просто очищение, если больше либо равно 30, то счетчик секунд очищается, а счетчик минут увеличивается на 1.

Примечание: если команда подана на 59 или 0 мин, то она будет проигнорирована.

0X1C ЧТЕНИЕ КОНФИГУРАЦИИ УСТАВОК/ТУ

Параметры	Байты структуры PARAMETRS
Номер уставки или ТУ	P1
Код	P2

Возвращаемые данные:

- если P2=0, возвращается конфигурация уставки под номером P1 (UST_CONFIG)
- если P2=1, возвращается конфигурация ТУ под номером P1 (TU_MASK).

0X1D ОЧИСТКА ЖУРНАЛОВ

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память ПИ.

Параметры	Байты структуры PARAMETRS
Код журнала	P1
Пароль на стирание счетчиков	P2-P5

Номера журналов

Код журнала	Журнал
0x04	Журнал событий
0x0E	Журнал включений-выключений
0x10	Журнал профилей

Возвращаемые данные: нет

0X1E ПОЛУЧИТЬ КОНФИГУРАЦИЮ ЖУРНАЛОВ

Параметры	Байты структуры PARAMETRS
Код журнала	P1

Возвращаемые данные: информация о журнале (MAGAZINEINFO)

0X1F ПОЛУЧИТЬ ДАННЫЕ ЖУРНАЛОВ

Параметры	Байты структуры PARAMETRS
Код журнала	P1
Индекс записи	P2-P3
Количество записей	P4-P5

Возвращаемые данные:

- Журнал событий (структура MAGAZINECRASH)
- Журнал включений/выключений (структура MAGAZINEINTERCEPT)
- Журнал профилей (структура MAGAXINEPROF)

Примечание: если индекс записи превышает количество записей в журнале (в случае, когда журнал переполнен, т.е. превышен объем журнала), то команда игнорируется.

Чтение осуществляется следующим образом: первой передается запись с индексом P2-P3, второй P2-P3+1… и т.д., равное количеству записей для чтения*.

*Если количество записей для чтения превышает количество записей в журнале, то оно считается равным количеству записей в журнале минус индекс записи (в случае, когда журнал переполнен – равным объему журнала).

*Если произведение количества записей для чтения на длину записи для данного журнала превосходит максимальное число байт, которые может передать ПИ за одну посылку, то оно усекается до наибольшего числа (целого кол-ва записей), которое ПИ может передать.

Перед каждым выполнением команды 0x1F рекомендуется выполнить команду 0x1E “Получить конфигурацию журналов”, т.к. при ее выполнении в ПИ фиксируются количество записей в журнале и индекс начальной записи на момент прихода команды. Это позволяет избежать потери записи при чтении в случае добавления записи в журнал в момент чтения. Обратите внимание, что фиксация параметров производится только для журнала, соответствующего переданному в команде 0x1E коду.

0X2B ЗАПИСАТЬ КОНФИГУРАЦИЮ УСТРОЙСТВА

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память ПИ

Параметры	Байты структуры PARAMETRS
Конфигурация (SENSORCONFIG)	P1-P9

Возвращаемые данные: нет

0X2C ПРОЧИТАТЬ КОНФИГУРАЦИЮ УСТРОЙСТВА

Параметры: нет

Возвращаемые данные: структура SENSORCONFIG

Поле SENSORCONFIG.Operation в команде 0x2C не несет значащей информации.

0X2E ЗАПИСЬ КОНФИГУРАЦИИ ТУ

Предварительные команды:

- 0x01 Подготовка к записи данных во флеш-память ПИ,
- 0x06 Разрешить запись уставок.

Примечание: в предварительной команде 0x06 необходимо передавать пароль на запись уставок (параметр P5=0).

Параметры	Байты структуры PARAMETRS
Номер ТУ	P1
Маска (TU_MASK)	P2-P9

Маски NOT, AND, OR в структуре TU_MASK служат для связи ТУ номер P1 с уставками.

Возвращаемые данные: нет

0X2F ПОЛУЧИТЬ ТОЧНЫЕ ДАННЫЕ

Параметры	Байты структуры PARAMETRS
Маска данных	P1

Возвращаемые данные:

Маска данных	Наименование запрашиваемых данных	Структура
0x01	Точные данные	ACCData
0x02	Линейные напряжения	ACCLineVoltage

Маска данных	Наименование запрашиваемых данных	Структура
0x04	Ток и напряжение нулевой последовательности	ACCNull
0x08	Полная мощность	ACCApparentPower

Примечание: команда принимает в виде параметра однобайтовую маску, определяющую, какие данные будут переданы контроллеру верхнего уровня. Порядок передачи запрошенных структур данных определяется по возрастанию величины маски. Маска может иметь любую возможную комбинацию по логической операции "ИЛИ" из указанного набора констант.

Примечание: для пересчета полученных значений следует использовать таблицу F2.

0X30 ЗАПИСЬ КОНФИГУРАЦИИ ЖУРНАЛА ПРОФИЛЕЙ

Предварительная команда: 0x01 Подготовка к записи данных во флеш-память ПЦ.

0x06 Разрешить запись уставок

Примечание: в предварительной команде 0x06 необходимо передавать пароль на стирание счетчиков энергии (параметр P5=1).

Параметры	Байты структуры PARAMETRS
Конфигурация журнала профилей (TJprof_config, в зависимости от P1)	P1

- при P1=1 – начиная с P2 передаются parmask, fsec, interval структуры TJprof_config;
- при P1=2 – начиная с P2 передаются ustmask, after_cnt структуры TJprof_config.

Возвращаемые данные: нет.

Примечание: смена конфигурации очищает журнал профилей.

0X31 ЧТЕНИЕ КОНФИГУРАЦИИ ЖУРНАЛА ПРОФИЛЕЙ

Параметры: нет

Возвращаемые данные: TJprof_config

0X32 ЗАПУСК ЖУРНАЛА ПРОФИЛЕЙ

Параметры: нет

Возвращаемые данные: нет

Примечание: команда очищает журнал, если тип интервала задан в периодах измерения.

0X33 ОСТАНОВ ЖУРНАЛА ПРОФИЛЕЙ

Параметры: нет

Возвращаемые данные: нет

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.

SETTU

```
//Состояние ТУ
typedef struct _SETTU
{
 unsigned  char   Active_TU1  :1;   //Активизировать ТУ1 (true/false)
 unsigned  char   Active_TU2  :1;   //Активизировать ТУ2 (true/false)
 unsigned  char   Active_TU3  :1;   //Активизировать ТУ3 (true/false)
 unsigned  char   Active_TU4  :1;   //не используется
 unsigned  char   Free4_Pos5  :4;   //Свободные биты (резерв)
 unsigned  char   WrkTimeTU1;       //Время удержания ТУ1 (сек)
 unsigned  char   WrkTimeTU2;       //Время удержания ТУ2 (сек)
 unsigned  char   WrkTimeTU3;       //Время удержания ТУ3 (сек)
 unsigned  char   WrkTimeTU4;       //не используется
}SETTU;
/*Примечание: если время удержания ТУ равно 0, ТУ включается и остается в этом состоянии до тех пор, пока не будет подана команда сверху или не сработает уставка на выключение.*/
```

PCODE

```
//Защитный код
typedef struct _PCODE
{
 unsigned  char  ByteN6 = 0x9C;
 unsigned  char  ByteN7 = 0x39;
}PCODE;
```

CONTROLBYTE

```
//Управляющий контрольный байт
typedef struct _CONTROLBYTE
{
 unsigned char   PicClearRegTU      :1;   //Очистка регистра-защелки ТУ
 unsigned char   PicClearError      :1;   //Очистить регистр ошибок ПИ
 unsigned char   Reserv             :6;   //Резерв
} CONTROLBYTE;
/*Контрольный байт управления используется для дополнительных операций с ПИ, а именно: очистки регистра-защелки состояния ТУ и очистки регистра ошибок. Для выполнения данных операций ПИ должен получить команду 0x07 "Получить данные (шаблон)"  с любой маской данных и активизированными битами PicClearRegTU=1 и (или) PicClearError=1*/
```

PHASE

```
typedef struct PHASE
{
 unsigned short   Current;              //Ток
 unsigned short   Voltage;              //Напряжение
 short            PowerActive;          //Мощность активная
 short            PowerReactive;        //Мощность реактивная
}PHASE;
/*Для преобразования величин, приведенных в данной таблице, к реальным значениям с плавающей точкой используйте формулы, приведенные в табл.F1*/
```

ENERGY

```
typedef struct _ENERGY
{
 unsigned long  EnActiveUse;        //Резерв
 unsigned long  EnActiveReturn;     //Резерв
 unsigned long  EnReactivePlus;     //Резерв
 unsigned long  EnReactiveMinus;    //Резерв
 unsigned long  CountTC1;           //Счетчик TC1
 unsigned long  CountTC2;           //Счетчик TC2
}ENERGY;
/*Два входа TC – TC1 и TC2 являются счетчиками импульсов. Таким образом, поля CountTC1 и CountTC2 отражают количество активизаций данных входов*/
```

FREQDAT

```
typedef struct _FREQDAT
{
 unsigned short   Freq;        //Период
 STATETU    StateTU;           //Состояние ТУ
 STATETC    StateTC;           //Состояние ТС
 ACTSTAT    ActStatus;         //Активные уставки (позиционный код)
 TULATCH    StateRegisterTU;   //Состояние  регистра – защелки ТУ
 short      T;                 //Значение температуры в градусах Цельсия* 32
 ERRORPIC   ErrorPIC;          //Ошибки контроллера PIC
}FREQDAT;
/*Для преобразования величины Freq (период) к реальным значениям с плавающей точкой используйте формулы, приведенные в табл.F1*/
```

STATETU

```
//Состояние ТУ
typedef struct   _STATETU
{
unsigned char   StateTU1      :1;//Состояние ТУ1 (1-выход ТУ активен, 0-нет)
 unsigned char   StateTU2     :1;//Состояние ТУ2
 unsigned char   StateTU3     :1;//Состояние ТУ3
 unsigned char   StateTU4     :1;//не используется
 unsigned char   FreeByte     :4;//Свободные биты
}STATETU;
```

STATETC

```
typedef struct   _STATETC
{
 unsigned char   StateTC1     :1;   //Состояние ТС1
 unsigned char   StateTC2     :1;   //Состояние ТС2
 unsigned char   StateTC3     :1;   //Состояние ТС3
 unsigned char   StateTC4     :1;   //Состояние ТС4
 unsigned char   StateTC5     :1;   //не используется
 unsigned char   StateTC6     :1;   //не используется
 unsigned char   StateTC7     :1;   //не используется
 unsigned char   StateTC8     :1;   //не используется
}STATETC;
```

ACTSTAT

```
typedef struct   _ACTSTAT
{
 unsigned char   UST1   :1;   //Состояние уставки 1
 unsigned char   UST2   :1;   //Состояние уставки 2
 unsigned char   UST3   :1;   //Состояние уставки 3
 unsigned char   UST4   :1;   //Состояние уставки 4
 unsigned char   UST5   :1;   //Состояние уставки 5
 unsigned char   UST6   :1;   //Состояние уставки 6
 unsigned char   UST7   :1;   //Состояние уставки 7
```

```
 unsigned char   UST8   :1;   //Состояние уставки 8
 unsigned char   UST9   :1;   //Состояние уставки 9
 unsigned char   UST10  :1;   //Состояние уставки 10
 unsigned char   UST11  :1;   //Состояние уставки 11
 unsigned char   UST12  :1;   //Состояние уставки 12
 unsigned char   UST13  :1;   //Состояние уставки 13
 unsigned char   UST14  :1;   //Состояние уставки 14
 unsigned char   UST15  :1;   //Состояние уставки 15
 unsigned char   UST16  :1;   //Состояние уставки 16
}ACTSTAT;
/*  Уставка активна, то есть находиться в сработавшем состоянии, если
соответствующее ей битовое поле равно единице, в противном случае данная уставка
не работает*/
```

TULATCH

```
//Состояние регистра-защелки ТУ
typedef struct
{
 unsigned char   TU1Changed   :1; // Признак срабатывания ТУ1 (1-срабатывало,
                                  // 0- не срабатывало)
 unsigned char   TU2Changed   :1; // Признак срабатывания ТУ2
 unsigned char   TU3Changed   :1; // Признак срабатывания ТУ3
 unsigned char   TU4Changed   :1; // Не используется
 unsigned char   FreeByte     :4; // Свободные биты
} TULATCH;
/* Наличие в битовом поле единицы говорит о том, что ТУ срабатывало. Сброс
регистра-защелки состояния ТУ осуществляется посылкой в ПИ управляющего
контрольного байта с активизированным полем PicClearRegTU=1 структуры
CONTROLBYTE, смотрите команду 0x07 "Получить данные (шаблон)"*/
```

ERRORPIC

```
//Ошибки контроллера PIC
typedef struct _ERRORPIC
{
 unsigned char ProcReset      :1;// сброс процессора по включению питания
 unsigned char ErrCS_Adr      :1;// ошибка КС блока информации ПИ (адрес
                                 // скорость)
 unsigned char ErrCS_Type     :1;// ошибка КС блока информации ПИ (тип,
                                 // пароли)
 unsigned char ErrCS_K        :1;// ошибка КС блока информации ПИ
                                 // (коэффициенты)
 unsigned char ErrCS_U        :1;// ошибка КС блока информации ПИ
                                 // (конфигурация //уставок)
 unsigned char ErrCS_C        :1;// ошибка КС блока информации ПИ (счетчики ТС,
                                 // время выключения)
 unsigned char ErrFrame       :1;// Ошибка кадровой синхронизации при приеме
                                 // данных
 unsigned char ErrCRC         :1;// Ошибка CRC принятого пакета
}ERRORPIC;
/*Наличие ошибок процессора ПИ отражаются в приведенной выше битовой структуре:
если битовое поле равно единице - ошибка присутствует, в противном случае - нет.
Сброс процессора ошибкой не является, данное поле идентифицирует факт выключения
ПИ. Очистка регистра ошибок может быть выполнена посылкой в ПИ управляющего
контрольного байта с активизированным полем PicClearError=1 структуры
CONTROLBYTE, смотрите команду 0x07 "Получить данные (шаблон)"*/
```

FIXDATA

```
typedef struct _FIXDATA
{
 unsigned long    TimeStamp;        //Временной штамп
 PHASE            IntegrPhaseA;     //Интегрированные значения фазы A
 PHASE            IntegrPhaseB;     //Интегрированные значения фазы B
```

```
 PHASE            IntegrPhaseC;     //Интегрированные значения фазы C
 unsigned long    EnActiveUse;      //не используется
 unsigned long    EnActiveReturn;   //не используется
 unsigned long    EnReactivePlus;   //не используется
 unsigned long    EnReactiveMinus;  //не используется
}FIXDATA;
/*В поле TimeStamp возвращается таймерная метка, полученная от контроллера верхнего уровня по команде 0x16 "Фиксация данных". Для преобразования величин интегрированных значений по фазам, приведенных в данной таблице, к реальным значениям с плавающей точкой используйте формулы в табл.F1*/
```

PREVTC

```
typedef struct _ PREVTC
{
 unsigned short   PreviousTC; // 2 предыдущих состояния ТС
} PREVTC;
/*В этой структуре хранятся 2 предыдущих состояния входов телесигнализации.
При получении новых значений ТС значения из младшего байта структуры PREVTC переписываются в старший, из соответствующих битов FREQDAT в младший байт PREVTC, а новое состояние уже соответственно в FREQDAT*/
```

TIMERSTAMP

```
typedef struct _TIMERSTAMP
{
 unsigned long   Sec2000;     //Число секунд с начала 2000 года
 unsigned char    SubSec;     //1/256 с
}TIMERSTAMP;
```

SENSORSTATE

```
typedef struct
{
 unsigned char ProcReset      :1;// сброс процессора по включению питания
 unsigned char ErrCS_Adr      :1;// ошибка КС блока информации ПИ (адрес
                                 // скорость)
 unsigned char ErrCS_Type     :1;// ошибка КС блока информации ПИ (тип,
                                 // пароли)
 unsigned char ErrCS_K        :1;// ошибка КС блока информации ПИ
                                 // (коэффициенты)
 unsigned char ErrCS_U        :1;// ошибка КС блока информации ПИ
                                 // (конфигурация //уставок)
 unsigned char ErrCS_C        :1;// ошибка КС блока информации ПИ (счетчики ТС,
                                 // время выключения)
 unsigned char ErrFrame       :1;// Ошибка кадровой синхронизации при приеме
                                 // данных
 unsigned char ErrCRC         :1;// Ошибка CRC принятого пакета
 unsigned char ProcReset2     :1;// Сброс процессора сторожевым таймером
 unsigned char ErrLCD         :1;// Не используется
 unsigned char StartRTC       :1;// Произведен запуск внутренних часов ПИ
 unsigned char ErrRTC         :1;// Ошибка в работе часов
 unsigned char ErrOver        :1;// Переполнение буфера приема
 unsigned char ErrCS_TU       :1;// Ошибка КС блока информации ПИ
                                 // (конфигурация ТУ)
 unsigned char ErrCS_Config   :1;// Ошибка КС блока информации ПИ
                                 // (конфигурация ПИ)
 unsigned char Jprof_Run      :1;// Идет запись в журнал профилей
}SENSORSTATE;
```

POWERSUMM

```
typedef struct _POWERSUMM
{
  signed    summPowerActive:24;     // Мощность суммарная активная
  signed    summPowerReactive:24;   // Мощность суммарная реактивная
} POWERSUMM;
/*Для преобразования величин к реальным значениям с плавающей точкой используйте
формулы, приведенные в табл.F1*/
```

FIXDATA2

```
typedef struct _FIXDATA2
{
  unsigned short  Frequency;   // Фиксированная частота
  unsigned char   StateTU;     // Телеуправление
  unsigned char   StateTC;     // Телесигнализация
} FIXDATA2;
```

ADD_VAL

```
typedef struct
{
      unsigned short U_PhAB;  // линейное напряжение между фазами A и B
      unsigned short U_PhBC;  // линейное напряжение между фазами B и C
      unsigned short U_PhCA;  // линейное напряжение между фазами C и A
      unsigned short I_3Ph;   // ток нулевой последовательности 3I0
      unsigned short U_3Ph;   // напряжение нулевой последовательности 3U0
}Add_Val;
```

AVER_VAL

```
typedef struct
{
      unsigned short Aver_I;  // среднее значение тока по фазам
      unsigned short Aver_U;  // среднее значение напряжения по фазам
}Aver_Val;
```

IPCINFO

```
typedef struct _IPCINFO
{
 unsigned  short  Model;            //Модель ПИ    (Hex)
 unsigned  char   ModNumber;        //Номер модели      (Hex)
 unsigned  char   PowerVType  :4;   //Тип питания
 unsigned  char   InputVType  :4;   //Тип входного напряжения
 unsigned  char   AcurrAType  :3;   //Не используется
 unsigned  char   CurveIsYes  :1;   //Не используется
 unsigned  char   SubModType  :4;   //Модификация модели
 unsigned  char   SoftVersion;      //Программная версия
 unsigned  char   ResForUse;        //Резерв
 unsigned  char   SerialNumberHight;//Серийный номер – старший байт
 unsigned  short  SerialNumber;     //Серийный номер
}IPCINFO;

//Расшифровка поля InputVType
```

InputVType	Расшифровка
1	Количество фаз = 3 Входное напряжение = 60V Входной ток = 1A
2	Количество фаз = 2 Входное напряжение = 100V Входной ток = 1A
3	Количество фаз = 3 Входное напряжение = 60V Входной ток = 5A
4	Количество фаз = 2 Входное напряжение = 100V Входной ток = 5A
5	Количество фаз = 3 Входное напряжение = 220V Входной ток = 5A

//Расшифровка поля PowerVType

PowerVType	Расшифровка
1	~80…260 В, =100…300 В;
2	Не используется

//Расшифровка поля SubModType

SubModType	Выполняемые функции
0	• измерение силы тока, фазного напряжения по каждой фазе сети и линейных напряжений; активной и реактивной мощности по каждой фазе сети; • вычисление напряжения переменного тока нулевой последовательности, силы переменного тока нулевой последовательности; • передача результатов измерений по гальванически изолированным интерфейсам связи RS-485.
1	• функции ПИ849Ц/XX-0 • функции телеуправления и телесигнализации; • подсчет количества импульсов, поступивших на входы телесигнализации ТС1 и ТС2; • включение выходов телеуправления в случае выхода измеряемых параметров за установленные пределы, при появлении сигнала на входах телесигнализации или по команде с верхнего уровня.
2	• функции ПИ849Ц/XX-1; • ведение часов реального времени; • присвоение метки времени измерениям параметров сети; • автоматический переход на летнее/зимнее время (с возможностью отключения данной функции); • архивирование событий с метками времени (журнал вкл/выкл, журнал событий и журнал профилей).

SENSORTIME

```
typedef struct _SENSORTIME
{
  unsigned char    Year;      //2000+Year  год
  unsigned char    Month;     //Месяц   1...12
  unsigned char    Day;       //День    1...31
  unsigned char    Hour;      //Час     0...23
  unsigned char    Min;       //Минуты  0...59
  unsigned char    Sec;       //Секунды 0...59
  unsigned char    SubSec;    //1/256 секунды
  unsigned char    DayOfWeek; //День недели
  unsigned char    Season;    //
}SENSORTIME;
```

SENSORSEASON

```
typedef struct _SENSORSEASON
{
//Переход на лето
 unsigned char   SummerMonth;       //Месяц  1-12
 unsigned char   SummerDay;         //Воскресенье месяца
 unsigned char   SummerHour;        //Час 1...24
//Переход на зиму
 unsigned char   WinterMonth;       //Месяц  1-12
 unsigned char   WinterDay;         //Воскресенье месяца
 unsigned char   WinterHour;        //Час 1...24
}SENSORSEASON;
```

TJPROF_CONFIG

```
typedef struct {

      unsigned long     parmask;    //Маска параметров
      unsigned short    fsec;       //Тип интервала времени записи
      unsigned short    interval;   //Интервал времени между записями
      TUSTMASK          ustmask;    //Маска: связь уставки с остановом
      unsigned short    after_cnt;  //Количество записей после останова

}TJprof_config;

//Маска параметров
```

Код	Наименование параметра	Тип
0x00000001	Мгновенный ток по фазе А	unsigned short
0x00000002	Мгновенное напряжение по фазе А	unsigned short
0x00000004	Мгновенная активная мощность по фазе А	short
0x00000008	Мгновенная реактивная мощность по фазе А	short
0x00000010	Мгновенный ток по фазе В	unsigned short
0x00000020	Мгновенное напряжение по фазе В	unsigned short
0x00000040	Мгновенная активная мощность по фазе В	short
0x00000080	Мгновенная реактивная мощность по фазе В	short
0x00000100	Мгновенный ток по фазе С	unsigned short
0x00000200	Мгновенное напряжение по фазе С	unsigned short
0x00000400	Мгновенная активная мощность по фазе С	short
0x00000800	Мгновенная реактивная мощность по фазе С	short
0x00001000	Мгновенные значения линейного напряжения АВ	unsigned short
0x00002000	Мгновенные значения линейного напряжения ВС	unsigned short
0x00004000	Мгновенные значения линейного напряжения СА	unsigned short
0x00008000	Мгновенные значения тока нулевой последовательности ($3I_0$)	unsigned short
0x00010000	Мгновенные значения напряжения нулевой последовательности ($3U_0$)	unsigned short
0x00020000	Частота	unsigned short
0x00040000	Мощность суммарная активная (мгновенная)	long
0x00080000	Мощность суммарная реактивная (мгновенная)	long
0x00100000	Состояние ТУ/ТС	STATETU, STATETC

/*Для преобразования величин, приведенных в данной таблице, к реальным значениям с плавающей точкой используйте формулы, приведенные в табл.F1*/
/*Маска параметров определяет, какие данные будут записаны в журнал профилей. Порядок записи параметров определяется по возрастанию величины маски. Маска может иметь любую возможную комбинацию по логической операции "ИЛИ" из указанного набора констант.*/
/*Если fsec=0, интервал между записями будет отсчитываться в периодах измерения. Структура записи – MAGAZINEPROF_PERIOD;
fsec=1, интервал между записями будет отсчитываться в секундах; в начало записи будет помещаться метка времени измерения. Структура записи – MAGAZINEPROF_SEC.*/
/*interval – интервал времени между записями, который отсчитывается в зависимости от значения переменной fsec в периодах измерения или секундах. Производится усреднение данных на этом интервале*/
/*after_cnt – количество записей в журнале, которое будет сделано после автоматического останова записи в журнал профилей*/

TUSTMASK

```
typedef struct
{
 unsigned char   UST1   :1;   //Останов по уставке 1
 unsigned char   UST2   :1;   //Останов по уставке 2
 unsigned char   UST3   :1;   //Останов по уставке 3
 unsigned char   UST4   :1;   //Останов по уставке 4
```

```
 unsigned char   UST5   :1;   //Останов по уставке 5
 unsigned char   UST6   :1;   //Останов по уставке 6
 unsigned char   UST7   :1;   //Останов по уставке 7
 unsigned char   UST8   :1;   //Останов по уставке 8
 unsigned char   UST9   :1;   //Останов по уставке 9
 unsigned char   UST10  :1;   //Останов по уставке 10
 unsigned char   UST11  :1;   //Останов по уставке 11
 unsigned char   UST12  :1;   //Останов по уставке 12
 unsigned char   UST13  :1;   //Останов по уставке 13
 unsigned char   UST14  :1;   //Останов по уставке 14
 unsigned char   UST15  :1;   //Останов по уставке 15
 unsigned char   UST16  :1;   //Останов по уставке 16
} TUSTMASK;
/*Останов происходит по срабатыванию уставки, если соответствующее ей битовое
поле равно единице.*/
```

MAGAZINEINFO

```
typedef struct _MAGAZINEINFO
{
 unsigned  short   RecordCount;          //Количество накопленных записей
 unsigned  short   RecordMax;            //Максимальное количество записей
 unsigned  char    RecordSize;           //Размер записи в байтах
 SENSORTIME        LastTime;
 unsigned  char    ClearCode;
}MAGAZINEINFO;
/*Расшифровка поля ClearCode: 0 - Очистка запрещена, 1 - Очистка с паролем, 2 -
Очистка без пароля. Как только количество накопленных записей в журнале превысит
максимальное количество записей   (RecordMax), старший бит RecordMax будет
установлен в 1, а счетчик RecordCount обнулится и подсчет накопленных записей
начнется заново. 1 в старшем бите RecordMax будет храниться до тех пор, пока
журнал не будет очищен. Поле LastTime – время последней очистки журнала*/
```

MAGAZINEINTERCEPT

```
typedef struct _MAGAZINEINTERCEPT
{
 unsigned long   PowerOFF;    //Время выключения, количество секунд, прошедших
                              //с 2000 г.
 unsigned long   PowerON;     //Время включения, количество секунд, прошедших
                              //с 2000 г.
 unsigned char   CSum;        //Контрольная сумма
}MAGAZINEINTERCEPT;
/*Контрольная сумма есть инвертированная сумма всех байт, кроме CSum, структуры
MAGAZINEINTERCEPT.*/
```

MAGAZINECRASH

```
typedef struct _MAGAZINECRASH
{
 unsigned char   CrashReason; //Код события
 unsigned long CrashTime;     //Время регистрации в формате – количество секунд,
                              //прошедших с 2000 г.
 unsigned char SubSec         //Время регистрации с точностью до 1/256 с
 STATETU         StateTU;     //Состояние выходов телеуправления
 STATETC         StateTC;     //Состояние входов телесигнализации
 unsigned short  Value;       //Величина аварийного параметра

 unsigned short  Freq;        //Период
 unsigned char  CSum;         //Контрольная сумма
}MAGAZINECRASH;
```

//Коды события (поля CrashReason)

Код	Наименование аварийной ситуации
0x01	Срабатывание уставки 1
0x02	Срабатывание уставки 2
…	…
0x10	Срабатывание уставки 16
0x80	Изменение состояния ТС
0x81	Изменение состояния ТУ
0x82	Одновременное срабатывание ТУ и ТС
0x83	Останов журнала профилей

/*Контрольная сумма есть инвертированная сумма всех байтов (кроме CSum) структуры MAGAZINECRASH*/

MAGAZINEPROF

В зависимости от значения переменной fsec структуры TJprof_config запись в журнал профилей имеет один из двух форматов: MEGAZINEPROF_SEC или MAGAZINEPROF_PERIOD.

MEGAZINEPROF_SEC

```
typedef struct
{
 CP56_time MeazureTime;       //Метка времени измерения
 unsigned char Not_used;      //не используется
 parameter1;                  //Количество и тип параметров зависит от маски
 …                            //параметров структуры команды 0x30
 parameterN;
 TWRITEBREAK WriteBreak;      //Байт неполноты записи
 unsigned char CSum;          //Контрольная сумма
}MAGAZINEPROF_SEC;
```

MEGAZINEPROF_PERIOD

```
typedef struct
{
 parameter1;                  //Количество и тип параметров зависит от маски
 …                            //параметров структуры команды 0x30
 parameterN;
 TWRITEBREAK WriteBreak;      //Байт неполноты записи
 unsigned char CSum;          //Контрольная сумма
}MAGAZINEPROF_PERIOD;
```

CP56_time

```
typedef struct
{
  unsigned short msec;        //0..59999 миллисекунд

  unsigned char min:6;        //0..59 минут
  unsigned char res1:1;
  unsigned char iv:1;

  unsigned char hour:5;       //0..23 часов
  unsigned char res2:2;
  unsigned char su:1;         //сезон

  unsigned char date:5;       //1..31 дней месяца
  unsigned char day:3;        //1..7 дней недели

  unsigned char month:4;      //1..12 месяцев
  unsigned char res3:4;

  unsigned char year:7;       //0..99 лет
  unsigned char res4:1;
} cp56_time;                        //7 байт
```

TWRITEBREAK

```
typedef struct
{
 unsigned char b0:1; // не прошел заданный интервал между записями, считанная
                     // запись не полная
 unsigned char b1:1; // произошла установка или синхронизация времени ПИ
 unsigned char b2:1; // первая запись после включения питания
 unsigned char b3:1; // резерв
 unsigned char b4:1; // резерв
 unsigned char b5:1; // резерв
 unsigned char b6:1; // резерв
 unsigned char b7:1; // выполнилось условие останова
}TWRITEBREAK;
```

КОНФИГУРАЦИЯ УСТРОЙСТВА

```
//CRASH_MAGAZIN_MASK
typedef struct _CRASH_MAGAZIN_MASK /*Уставка, срабатывание которой
                                     фиксируется в журнале событий*/
{
 unsigned char   Crash_UST1   :1;   //Сработала уставка 1
 unsigned char   Crash_UST2   :1;   //Сработала уставка 2
 unsigned char   Crash_UST3   :1;   //Сработала уставка 3
 unsigned char   Crash_UST4   :1;   //Сработала уставка 4
 unsigned char   Crash_UST5   :1;   //Сработала уставка 5
 unsigned char   Crash_UST6   :1;   //Сработала уставка 6
 unsigned char   Crash_UST7   :1;   //Сработала уставка 7
 unsigned char   Crash_UST8   :1;   //Сработала уставка 8
 unsigned char   Crash_UST9   :1;   //Сработала уставка 9
 unsigned char   Crash_UST10  :1;   //Сработала уставка 10
 unsigned char   Crash_UST11  :1;   //Сработала уставка 11
 unsigned char   Crash_UST12  :1;   //Сработала уставка 12
 unsigned char   Crash_UST13  :1;   //Сработала уставка 13
 unsigned char   Crash_UST14  :1;   //Сработала уставка 14
 unsigned char   Crash_UST15  :1;   //Сработала уставка 15
 unsigned char   Crash_UST16  :1;   //Сработала уставка 16
} CRASH_MAGAZIN_MASK;

//CRASH_STATETC
typedef struct  /*Номера ТС, изменение состояния которых фиксируется в
                  журнале событий*/
{
 unsigned char   Crash_StateTC1     :1;   //изменил состояние ТС1
 unsigned char   Crash_StateTC2     :1;   //изменил состояние ТС2
 unsigned char   Crash_StateTC3     :1;   //изменил состояние ТС3
 unsigned char   Crash_StateTC4     :1;   //изменил состояние ТС4
 unsigned char   Crash_StateTC5     :1;   //не используется
 unsigned char   Crash_StateTC6     :1;   //не используется
 unsigned char   Crash_StateTC7     :1;   //не используется
 unsigned char   Crash_StateTC8     :1;   //не используется
}CRASH_STATETC;

//SENSORCONFIG
typedef struct  _SENSORCONFIG
{
 unsigned char Operation;
 CRASH_MAGAZIN_MASK CrashMagazinMask;
 unsigned char FixPeriod;      //Не используется
 unsigned char PowerPeriod;    //Не используется
 unsigned char TaxCount;       //Не используется
 CRASH_STATETC CrashMagazinTCMask;
 unsigned char TCtime;         /*Время дребезга для входов ТС (единица равна
                                 1/256 с), если 0, то устанавливается равной 6
```

```
                                    (т.е. 20 мс)*/
 unsigned char Unused;          //Не используется
}SENSORCONFIG;
```

/*В зависимости от параметра P1 (поле Operation структуры SENSORCONFIG) заполняются соответствующие поля этой структуры. Поле Operation принимает битовую маску выполняемой операции. Маска формируется по логической операции "ИЛИ". */

//Расшифровка поля Operation:

Код	Расшифровка
0x01	Установка масок для журнала событий
0x10	Установка времени дребезга ТС

TU_MASK

```
typedef struct
{
  unsigned short no;          // маска no
  unsigned short and;         // маска and
  unsigned short or;          // маска or
  unsigned short TimeAfter;   //Время удержания ТУ в секундах (0 – бесконечное)
} TU_MASK;
```

ACCDATA

```
 typedef struct
{
  unsigned Current_A          :24;        // Ток по фазе A
  unsigned Voltage_A          :24;        // Напряжение по фазе A
  signed Active_Power_A       :24;        // Активная мощность по фазе A
  signed Reactive_Power_A     :24;        // Реактивная мощность по фазе A

  unsigned Current_B          :24;        // Ток по фазе B
  unsigned Voltage_B          :24;        // Напряжение по фазе B
  signed Active_Power_B       :24;        // Активная мощность по фазе B
  signed Reactive_Power_B     :24;        // Реактивная мощность по фазе B

  unsigned Current_C          :24;        // Ток по фазе C
  unsigned Voltage_C          :24;        // Напряжение по фазе C
  signed Active_Power_C       :24;        // Активная мощность по фазе C
  signed Reactive_Power_C     :24;        // Реактивная мощность по фазе C
} ACCData;
```

ACCLINEVOLTAGE

```
 typedef struct
{
  unsigned Voltage_AB         :24;        // Напряжение между фазами A и B
  unsigned Voltage_BC         :24;        // Напряжение между фазами B и C
  unsigned Voltage_CA         :24;        // Напряжение между фазами C и A
} ACCLineVoltage;
```

ACCNULL

```
typedef struct
{
  unsigned _3I0               :24;        // Ток нулевой последовательности
  unsigned _3U0               :24;        //Напряжение нулевой последовательности
} ACCNull;
```

ACCAPPARENTPOWER

```
 typedef struct
{
  unsigned S_A                :24;        // Полная мощность по фазе A
  unsigned S_B                :24;        // Полная мощность по фазе B
  unsigned S_C                :24;        // Полная мощность по фазе C
} ACCApparentPower;
```

UST_CONFIG

```
typedef struct
{  //
   unsigned char Type;              // Тип уставки
   unsigned char OnOffTU;           // 1-вкл; 0-выкл ТУ
   unsigned short Value;            // Значение для данного типа уставки
   unsigned short TimeTo;           // Время с момента возникновения условия на
                                    //  срабатывание уставки до ее фактического
                                    // срабатывания(1/256 с)
   unsigned short ReturnValue;      // Резерв
} UST_CONFIG;

//Типы уставок
#define UST_NOTYPE                  0 // Уставка отсутствует
#define UST_MAX_CURRENT             1 // Уставка по макс. току
#define UST_MIN_CURRENT             2 // Уставка по мин. току
#define UST_MAX_VOLTAGE             3 // Уставка по макс. напряжению

#define UST_MIN_VOLTAGE             4 // Уставка по мин. напряжению
#define UST_MAX_ACTIVE_POWER        5 // Уставка по макс. активной мощности
#define UST_MIN_ACTIVE_POWER        6 // Уставка по мин. активной мощности
#define UST_MAX_REACTIVE_POWER      7 // Уставка по макс. реактивной
#define UST_MIN_REACTIVE_POWER      8 // Уставка по мин. реактивной
#define UST_MAX_FREQUENCY           9 // Уставка по макс. частоте
#define UST_MIN_FREQUENCY           10// Уставка по мин. частоте
#define UST_MAX_NULL_CURRENT        11// Уставка по макс. току нулевой
                                      // последовательности
#define UST_MIN_NULL_CURRENT        12// Уставка по мин. току нулевой
                                      // последовательности
#define UST_MAX_NULL_VOLTAGE        13// Уставка по макс. напряжению нулевой
                                      // последовательности
#define UST_MIN_NULL_VOLTAGE        14// Уставка по мин. напряжению нулевой
                                      // последовательности
#define UST_MAX_TEMP                15// Уставка по макс. внутренней температуре
#define UST_MIN_TEMP                16// Уставка по мин. внутренней температуре
#define UST_REMOTE_SIGNALING        128//Уставки по ТС
```

ФОРМУЛЫ ДЛЯ РАСЧЕТНЫХ ВЕЛИЧИН

Таблица F1

N	Формула	Применение	Единица измерения
1	2457600.0 / Freq	Значение частоты	Гц
2	Current/1000.0	Значение тока	А
3	Voltage/10.0	Значение напряжения	В
4	Power/10.0	Значение активной мощности	Вт
5	Power/10.0	Значение реактивной мощности	вар
6	Power/10.0	Значение полной мощности	В·А
7	SumPower/100.0	Значение суммарных мощностей	Вт, вар, В·А

Таблица F2 (для пересчета точных величин)

N	Формула	Применение	Единица измерения
1	Current/10000.0	Значение тока	А
2	Voltage/100.0	Значение напряжения	В
3	Power/100.0	Значение активной мощности	Вт
4	Power/100.0	Значение реактивной мощности	вар
5	Power/100.0	Значение полной мощности	В·А

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);
}
```