**ТЕПЛОВЫЧИСЛИТЕЛИ**

**АКСИОМА**

Модификация: Аксиома-03

Протокол обмена

Версия протокола 1

ООО «Аква Сити Сервис»

Россия, 456209, Челябинская область, г. Златоуст, ул. Металлистов, 14А

Телефон: +7(951) 485-56-26, +7(950) 735-42-08

e-mail: AquaCity@list.ru

**Введение**

Настоящий документ предназначен для специалистов, осуществляющих программирование и обслуживание систем диспетчеризации.

Тепловычислители Аксиома соответствуют требованиям постановления Правительства РФ от 18 ноября 2013 г. № 1034 «О коммерческом учете тепловой энергии, теплоносителя». Значения плотности и энтальпии воды вычисляются согласно МИ 2412 (МИ 2412-97) по измеренной температуре в диапазоне от 0 °С до 180 °С и давлению в диапазоне от 0 до 1,6 МПа.

Настоящая редакция протокола обмена актуальна для тепловычислителей модификации Аксиома-03 и программной версии интерфейсной части не менее 3.0.2. Дальнейшая постоянная работа изготовителя над совершенствованием возможностей, повышением надежности и удобства пользования может привести к некоторым непринципиальным изменениям в тепловычислителях Аксиома, не отраженным в настоящей редакции.

1. **Описание и принцип работы**

**1.1 Назначение**

Тепловычислители предназначены для измерений выходных сигналов измерительных преобразователей параметров теплоносителя (расхода, температуры, давления) и вычислений по результатам измерений количества теплоты (тепловой энергии), потребляемой в закрытых и открытых водяных системах теплоснабжения в составе теплосчетчиков.

Принцип действия тепловычислителей основан на измерении первичных параметров теплоносителя с помощью преобразователей расхода, температуры и давления, с последующей обработкой результатов измерений в соответствии с установленным алгоритмом учета тепловой энергии.

Тепловычислители обеспечивают:

* измерение текущих значений расхода теплоносителя, температуры и давления с помощью первичных преобразователей. Данное измерение может происходить одновременно во всех трубопроводах (до восьми), с определением текущих и средних значений параметров теплоносителя.
* вычисление значений полной тепловой энергии и количество потребленной тепловой энергии в отдельном трубопроводе.
* архивирование результатов измерений, вычислений, диагностики и установочных параметров в энергонезависимой памяти.
* защищенность архивных данных и установочных параметров от несанкционированного доступа.
* индикацию измеряемых, расчетных, архивированных данных и установочных параметров.
* обмен данными с внешними устройствами через проводные интерфейсы: USB, RS-485, Ethernet и беспроводные интерфейсы: Wi-Fi, Bluetooth, LoRa, GSM.
* ввод и использование в расчетах договорных значений расхода, температуры и давления теплоносителя в трубопроводах, а также договорных значений расхода и тепловой мощности в теплосистеме.
* учет до восьми полностью независимых контуров теплоносителя.
* возможность программного конфигурирования каждого контура системы измерений и расчетов, с учетом вида контролируемой теплосистемы и набора используемых первичных параметров расхода, температуры и давления теплоносителя.

Кроме того, тепловычислители позволяют:

* использовать в расчетах текущие параметры холодной воды (температура, давление) и представлять их в отчетных формах;
* принимать сигналы от внешних датчиков, входящих в систему безопасности;
* функционировать как при автономном, так и при внешнем питании в штатном режиме.

Согласно сконфигурированному контуру, тепловычислители измеряют тепловую энергию по тепловому вводу, который может включать в себя следующие трубопроводы:

* подающий;
* обратный;
* горячего водоснабжения или подпитки;
* холодного водоснабжения.

Рассчитанные значения тепла, объема и массы в архивах (часовом, суточном и месячном) в тепловычислителях формируются нарастающим итогом.

**1.2 Интерфейсы связи**

Тепловычислители имеют встроенные проводные интерфейсы USB, RS-485, Ethernet и беспроводные интерфейсы: Wi-Fi, Bluetooth, LoRa, GSM для связи с внешними устройствами и передачи архивной информации.

Встроенный GSM модем дает возможность тепловычислителю передавать информацию по каналу сотовой связи, в том числе и в сеть Интернет. Дальность связи по GSM-каналу определяется характеристиками сотовой связи.

На базе программного комплекса «Аксиома-КС» можно организовать диспетчерскую сеть для тепловычислителей Аксиома.

Все встроенные интерфейсы обеспечивает возможность доступа к измерительным, расчетным и установочным параметрам, включая архивы.

**2. Протокол обмена**

**2.1 Пакет данных**

Пакет данных состоит из следующей последовательности:

- Начало пакета 0xEF (1 байт)

- Длина данных (2 байта)

- Команда (2 байта)

- Данные

- Контрольная сумма CRC16 (2 байта)

- Параметр 1 (1 байт)

- Параметр 2 (1 байт)

- Параметр 3 (1 байт)

- Параметр 4 (1 байт)

- Конец пакета 0xEE (1 байт)

Структура пакета:

public struct DataPacketStruct

 {

 public byte Preamble; //0xEF

 public UInt16 DataLen; //длина данных

 public UInt16 Command; //команда

 public byte[] Data; //данные

 public UInt16 CRC;

 public byte Option1; //параметр 1

 public byte Option2; //параметр 2

 public byte Option3; //параметр 3

 public byte Option4; //параметр 4

 public byte End; //0xEE

 }

**2.2 Команды**

Тепловычислитель принимает следующие команды:

#define CheckConnectionCommand 1 //ответ, что вход выполнен

#define ReadIdentificationTable 2 //чтение идентификационных данных

#define ReadDateTime 26 //чтение даты и времени

#define ReadLoadTable 4 //чтение настроек

#define ReadPodSystemaYchetaTY 5 //чтение подсистемы учета

#define ReadStatusTable 6 //чтение статусной таблицы

#define ReadConfigurationVaribleString 7 //чтение конфигурационной строки

#define ReadTekyshieTable 30 //чтение текущих показаний

#define ReadSummarnueTable 32 //чтение накопленных показаний

#define ReadMesyachnueArxivFromDateTime 54 //чтение архива с месячными показаниями по определенной дате

#define ReadSytochnueArxivFromDateTime 52 //чтение архива с суточными показаниями по определенной дате

#define ReadChasovueArxivFromDateTime 50 //чтение архива с часовыми показаниями по определенной дате

#define ReadSummarnueArxivFromDateTime 56 //чтение архива с накопленными показаниями по определенной дате

**2.3 Структуры данных**

**2.3.1 Индентификационные данные**

 public const int ID\_name\_Len = 16;

 public const int ID\_METRO\_serial\_number\_Len = 6;

 public const int ID\_ITERF\_serial\_number\_Len = 10;

 public const int VersionESP32\_Len = 10;

 public const int VersionBootLoader\_Len = 10;

 [StructLayout(LayoutKind.Sequential, Pack = 2)]

 unsafe public struct IdentificationTable

 {

 public fixed sbyte ID\_name[ID\_name\_Len]; //Наименование ПО вычислителя

 public fixed sbyte ID\_METRO\_serial\_number[ID\_METRO\_serial\_number\_Len]; //Версия ПО метрологически значимой части

 public fixed sbyte ID\_ITERF\_serial\_number[ID\_ITERF\_serial\_number\_Len]; //Версия ПО микроконтроллера

 public fixed sbyte VersionESP32[VersionESP32\_Len]; //Версия ПО на ESP32

 public fixed sbyte VersionBootLoader[VersionBootLoader\_Len]; //Версия ПО загрузчика для микроконтроллера

 public UInt16 VersionProtocol; //версия протокола обмена

 public UInt16 METROLOGcrc; //Контрольная сумма метрологически значимой части

 }

**2.3.2 Таблица с настройками**

Настройки защищаются перемычкой и контрольной суммой

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode, Pack = 2)]

 public struct LoadTable

 {

 public UInt16 Serial\_number; ///серийный номер устройства

 ///настройки работы вычислителя

 public UInt16 time\_integrirov; //период считывания

 public byte work\_function; ///режим работы

 public byte ActionsOnSwitchOffPitanie; ///действие при отключения внешнего питания: 0 - Отключено, 1 - Авария, 2 - Уведомление

 //Параметры датчиков температур

 public fixed byte Ro\_datchikov\_temperatur[9]; //номинальное сопротивление датчиков

 public fixed byte type\_datchikov\_temperatur[9]; //тип датчиков температур

 public fixed float temperatura\_value[8]; ///заданное значение температуры для t1,t2,t3,t4,t5,t6,t7,t8

 public fixed float temperatura\_MaxValue[9]; ///придел максимальной температуры

 public fixed float temperatura\_MinValue[9]; ///минимальный придел температуры

 //Параметры датчиков давления

 public fixed float davlenie\_P\_max[8]; //максимальное давление

 public fixed float davlenie\_Io[8]; //значение тока при P=0

 public fixed float davlenie\_Imax[8]; //значение тока при P=Pmax

 public fixed float davlenie\_value[8]; ///заданное значение давления для P1,P2,P3,P4,P5,P6,P7,P8

 public fixed float davlenie\_value\_ON\_NS[8]; //давление при НС

 //Параметры датчиков расхода

 public fixed float rasxod\_Bimpulsa[8]; //вес импульса

 public fixed float rasxod\_MaxValue[8]; //максимальный расход

 public fixed float rasxod\_MinValue[8]; //минимальный расход

 public fixed float rasxod\_PerValue[8]; //переходный расход

 //Параметры холодной воды

 public byte switch\_TemperaturaColdWater; //0 - отключен (не используется в силу исп. схемы) 1 - исп. переменная 2 - вручную

 public byte NumChenal\_TemperaturaColdWater; //номер канала температуры, если исп. переменная

 public byte switch\_DavlenieColdWater; //0 - отключен (не используется в силу исп. схемы) 1 - исп. переменная 2 - вручную

 public byte NumChenal\_DavlenieColdWater; //номер канала давления, если исп. переменная

 public byte switch\_RasxodColdWater; // 0 - отключен (не используется в силу исп. схемы) , 1 - исп. переменная

 public byte NumChenal\_RasxodColdWater; //номер канала расхода, если исп. переменная

 //Настройки датчиков

 public fixed byte switch\_temperatura[9]; //вкл-выкл каналов сопротивлений

 public fixed byte switch\_davlenie[8]; //вкл-выкл каналов тока

 public fixed byte switch\_volume[8]; //вкл-выкл каналов измерения частоты

 ///описание всех подсистем учета тепловой энергии

 public byte CountUsedNumberChenal; //общее количество используемых груп каналов (группа это t1,P1,V1)

 public byte CountUsedDeltaTemperaturaDavleniePeremen; //общее количество разности температуры и давления

 public byte CountUsedDeltaMassaPeremen; //общее количество разности массы переменных

 public byte CountUsedQperemen;//общее количество Q переменных (a,b,c,d,e,f)

 public byte CountPodSystemsYchetaTY; //количество подсистем учета ТЭ

 ///настройки отопительного сезона

 public datetime StartHeatdate; //начало отопительного сезона

 public datetime EndHeatdate; //конец отопительного сезона

 //заданное значение

 public float tx\_summer; //заданное значение температуры в летнее время

 public float tx\_winter; //заданное значение температуры в зимнее время

 public float Px\_summer; //заданное значение давления в летнее время

 public float Px\_winter; //заданное значение давления в зимнее время

 };

**2.3.3 Конфигурационная строка использующихся переменных**

 [StructLayout(LayoutKind.Sequential, Pack = 1)]

 unsafe public struct configurationVarible

 {

 public byte[] ConfigurationVaribleString;

 }

**2.3.4 Структура с показаниями**

 [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode, Pack = 2)]

 public struct PokazaniyaTable

 {

 public datetime OnDateTime; //время и дата

 //нештатные ситуации

 public fixed byte errors\_temperatura[14]; // t1, t2, t3, t4, t5, t6, t7, t8, tx, dt1, dt2, dt3, dt4, tokr

 public fixed byte errors\_davlenie[13]; // P1, P2, P3, P4, P5, P6, P7, P8, Px, dP1, dP2, dP3, dP4

 public fixed byte errors\_volume[13]; //Go1, Go2, Go3, Go4, Go5, Go6, Go7, Go8, Gox, dGo1, dGo2, dGo3, dGo4

 public fixed byte errors\_massa[13]; //Gm1, Gm2, Gm3, Gm4, Gm5, Gm6, Gm7, Gm8, Gmx, dGm1, dGm2, dGm3, dGm4

 public fixed byte errors\_energi[12]; //Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Qa, Qb, Qc, Qd

 //контура которые находятся в аварии

 public fixed byte avariya\_contur[8];

 //добивка для четности

 public byte temp;

 //значения

 public fixed float timeWork[8]; //время наработки в минутах для 8 контуров

 public fixed float timeNotWork[8]; //время простоя в минутах для 8 контуров

 public fixed float temperatura[14]; // t1, t2, t3, t4, t5, t6, t7, t8, tx, dt1, dt2, dt3, dt4, tokr

 public fixed float davlenie[13]; // P1, P2, P3, P4, P5, P6, P7, P8, Px, dP1, dP2, dP3, dP4

 public fixed float obemnui\_rasxod[13]; //Go1, Go2, Go3, Go4, Go5, Go6, Go7, Go8, Gox, dGo1, dGo2, dGo3, dGo4

 public fixed float massovui\_rasxod[13]; //Gm1, Gm2, Gm3, Gm4, Gm5, Gm6, Gm7, Gm8, Gmx, dGm1, dGm2, dGm3, dGm4

 public fixed float energi[12]; //Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Qa, Qb, Qc, Qd

 };

**2.3.5 Структура с накопленными показаниями**

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode, Pack = 2)]

 public struct SummarnueTable

 {

 public datetime OnDateTime; //время и дата

 public fixed byte temp[2]; //не используется

 public fixed double timeWork[8]; //время наработки в минутах для 8 контуров

 public fixed double timeNotWork[8]; //время простоя в минутах для 8 контуров

 public fixed double volume[13]; //V1, V2, V3, V4, V5, V6, V7, V8, Vx, dV1, dV2, dV3, dV4

 public fixed double massa[13]; //M1, M2, M3, M4, M5, M6, M7, M8, Mx, dM1, dM2, dM3, dM4

 public fixed double energi[12]; //Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Qa, Qb, Qc, Qd

 };

**2.3.6 Статусная таблица**

[StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode, Pack = 2)]

 public struct StatusTable

 {

 public byte SwitchSaveButton; //кнопка защиты от записи

 public byte PowerBatPercent; ///уровень заряда батареи

 public byte PowerAccumPercent; ///уровень заряда аккумулятора

 public byte PowerSource; //источник питания

 public float TemperaturaCore; //температура устройства

 public UInt16 CountChasovueArxivZap; ///количество часовых записей в журнале

 public UInt16 CountSytochnueArxivZap; ///количество суточных записей в журнале

 public UInt16 CountMesyachnueArxivZap; ///количество месячных записей в журнале

 public UInt16 CountSummarnueArxivZap; ///количество суммарных записей в журнале

 public datetime LastModDateTime; //дата последних изменений настроек (по аналогии с CRC)

 public datetime StartMagazinZapDateTime; //дата начала ведения всего журнала

 //даты начала журнала

 public datetime StartChasovueMagazinZapDateTime; //дата начала часового журнала

 public datetime StartSytochnueMagazinZapDateTime; //дата начала суточного журнала

 public datetime StartMesyachnueMagazinZapDateTime; //дата начала месячного журнала

 public datetime StartSummarnueMagazinZapDateTime; //дата начала сумарного журнала

 //даты конца журнала

 public datetime EndChasovueMagazinZapDateTime; //дата конца часового журнала

 public datetime EndSytochnueMagazinZapDateTime; //дата конца суточного журнала

 public datetime EndMesyachnueMagazinZapDateTime; //дата конца месячного журнала

 public datetime EndSummarnueMagazinZapDateTime; //дата конца сумарного журнала

 };

**2.3.7 Структура подсистемы учета тепловой энергии**

public const int NamePodSystem\_Len = 9; //длина имени подсистемы в UTF-8 (т.к. char кодируется 2мя байтами, то в byte будет 18 байт)

 [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode, Pack = 2)]

 unsafe public struct PodSystemaYchetaTY

 {

 public fixed char NamePodSystem[NamePodSystem\_Len]; //имя подсистемы в UTF-8 (кодируется 2 байтами 1 символ)

 public byte FirstUsedNumberChenal; //начальная цифра переменных например 1

 public byte FirstUsedTemperaturaDavleniePeremen; //начальная цифра разности температуры и давления например 1

 public byte FirstUsedDeltaMassaPeremen; //начальная цифра разности массы

 public byte FirstUsedQperemen; //буква начала переменных например b

 public byte CountChenals; //количество используемых каналов

 public byte CountDeltaTemperaturaDavleniePeremen; //количество используемых разности температуры и давления переменных

 public byte CountDeltaMassaPeremen; //количество используемых разности массы переменных

 public byte CountQperemen; //количество используемых Q в подсистеме Qa, Qb, Qc...

 public byte UseTemperatOkrSredu; //использовать датчик окружающей среды

 public fixed byte AlgoritmQ[5]; //индекс алгоритма счета Q

 public fixed byte AlgoritmMassa[5]; //индекс алгоритма счета массы

 public byte UseTemperatAndDavlenColdWater; //флаг использования температуры и давления холодной воды

 }

**2.3.8 Структура для доступа к серверу**

 public const int INF\_Login\_MaxLen = 12;

 public const int INF\_Password\_MaxLen = 12;

 [StructLayout(LayoutKind.Sequential, Pack = 1)]

 unsafe public struct loginAndPassForEnterToServerTable

 {

 public fixed byte INF\_Login[INF\_Login\_MaxLen];

 public fixed byte INF\_Password[INF\_Password\_MaxLen];

 public UInt16 VersionProtocol; //версия протокола связи

 }

**2.3.9 Структура даты и времени**

 public struct datetime

 {

 public byte Second;

 public byte Minute;

 public byte Hour;

 public byte Day;

 public byte Month;

 public byte Year;

 };

1. **Основные алгоритмы**
	1. **Разбор конфигурационной строки**

Конфигурационная строка считывается с прибора командой:

#define ReadConfigurationVaribleString 7

Алгоритм разбора конфигурационной строки:

public void PriparePokazViewFromConfigurationString()

 {

 UTF8Encoding uTF8Encoding = new UTF8Encoding();

 PodSytemStruct PodSytem = new PodSytemStruct();

 if (ConfigurationVarible.ConfigurationVaribleString.Length < 18) return; //если конфигурация отсутствует, то выход

 int IndexInPodsystem = 0; //индекс в подсистеме

 for (int i = 0; i < ConfigurationVarible.ConfigurationVaribleString.Length; i++)

 {

 if (IndexInPodsystem == 0)

 {

 string NamePodSystem = uTF8Encoding.GetString(ConfigurationVarible.ConfigurationVaribleString, i, 18);

 PodSytem = new PodSytemStruct();

 PodSytem.Name = NamePodSystem;

 PodSytem.config = new byte[256];

 PodSytem.configLen = 0;

 i += 17;

 IndexInPodsystem = 17;

 }

 else

 {

 PodSytem.config[PodSytem.configLen++] = ConfigurationVarible.ConfigurationVaribleString[i];

 }

 ///

 if (ConfigurationVarible.ConfigurationVaribleString[i] == 0xFF)

 {

 IndexInPodsystem = 0;

 PodSystemsList.Add(PodSytem);

 }

 }

 }

* 1. **Расчет контрольной суммы**

public enum InitialCrcValue { Zeros, NonZero1 = 0xffff, NonZero2 = 0x1D0F }

const ushort poly = 0x1021;

 ushort[] table = new ushort[256];

 ushort initialValue = 0;

public ushort ComputeChecksum(byte[] bytes, int len, int offset)

 {

 ushort crc = this.initialValue;

 for (int i = offset; i < len; ++i)

 {

 crc = (ushort)((crc << 8) ^ table[((crc >> 8) ^ (0xff & bytes[i]))]);

 }

 return crc;

 }

public CRC16(InitialCrcValue initialValue)

 {

 this.initialValue = (ushort)initialValue;

 ushort temp, a;

 for (int i = 0; i < table.Length; ++i)

 {

 temp = 0;

 a = (ushort)(i << 8);

 for (int j = 0; j < 8; ++j)

 {

 if (((temp ^ a) & 0x8000) != 0)

 {

 temp = (ushort)((temp << 1) ^ poly);

 }

 else

 {

 temp <<= 1;

 }

 a <<= 1;

 }

 table[i] = temp;

 }

 }

* 1. **Работа с архивом**

Запрос месячных, суточных, часовых и накопленных показаний по определенной дате производится по следующему алгоритму:

 Архивные константы:

 public const byte MoreOrEqual = 1; //больше или равно

 public const byte MoreNotEqual = 2; //больше

 public const byte OnlyEqual = 3; //равно

Пример:

 case 54: //пришла структура с месячными показаниями по определенной дате

 if (Option1 == 1)

 {

 //занесем в структуру

 ReadArxivPokazaniyaTable = (Aksioma03Structures\_v1.PokazaniyaTable)Net.CopyDataToSructures(ReadArxivPokazaniyaTable, DataReceive);

 //напишем

 DateTime DT = ConvertDTStructToDateTime(ReadArxivPokazaniyaTable.OnDateTime);

 FUNCTIONS.WriteToConsolAndLogFile("Device: " + Net.Login + " :: Get MesyachnueTable on Date = " + DT.ToString("yyyy-MM-dd"));

 //запросим следующую структуру

 DATA = Net.CopyStructuraToByte(ReadArxivPokazaniyaTable.OnDateTime);

 if (Net.SendData(DATA, 54, MoreNotEqual, 0, 0, 0) == false) return Aksioma03Net\_v1.SendERROR;

 }

 else

 {

 //больше данных нет, запросим суточные показания

 FUNCTIONS.WriteToConsolAndLogFile("Device: " + Net.Login + " :: Look Sytochnue arxiv");

 ReadArxivData(SytochnueRow);

 }

 break;