TERMEX

ТЕРМОМЕТРЫ МНОГОФУНКЦИОНАЛЬНЫЕ ТмК

Протокол связи с компьютером

СОДЕРЖАНИЕ

Настоящее описание распространяется на «Термометры многофункциональные ТмК» (далее по тексту — термометры) и содержит сведения, необходимые для разработки прикладного программного обеспечения (ПО), предназначенного для удаленного управления работой термометров в составе программно-аппаратных комплексов.

Изготовитель оставляет за собой право вносить в протокол изменения, не затрагивающие описанные ниже функции.

1 ИНТЕРФЕЙСЫ ТЕРМОМЕТРОВ ДЛЯ УДАЛЕННОГО УПРАВЛЕНИЯ

На рисунке 1 показана задняя панель термометров, на которой расположены разъемы интерфейсов USB (1), RS-232 (2), RS-485 (3).

Для опробования команд из протокола можно воспользоваться программой TmKGraph. TmKGraph содержит командную консоль, с помощью которой можно отправлять любые команды подключенному термометру.

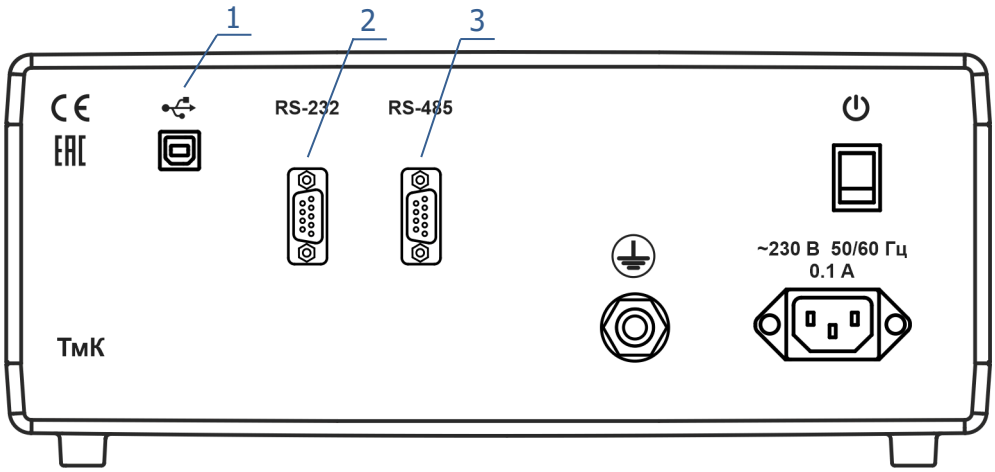

Рисунок 1

1.1 USB

1.1.1 Термометры подключаются к USB порту компьютера с помощью стандартного кабеля для периферийных устройств USB 2.0 Type-A — Type-B (рисунок 2).

Рисунок 2

1.1.2 Настройка драйверов операционной системы (ОС) выполняется автоматически при первом подключении термометра к USB порту компьютера.

1.1.3 Для облегчения написания прикладного ПО следует использовать распространяемую по запросу библиотеку `hidt.dll`, в которой находятся необходимые функции для обмена данными с термометром.

1.1.4 Если прикладное ПО создается без использования библиотеки `hidt.dll`, то необходимо учитывать следующее: для ОС подключенный термометр является HID-устройством, для обмена данными с которым следует использовать соответствующее API. Термометр (HID-устройство) ведет обмен данными с помощью так называемого репорта. Репорт — это массив байт размером 64 байта: `uint8_t report[64]`. Данные, сформированные в соответствии с протоколом, должны быть расположены внутри репорта. Если размера репорта не хватает, чтобы вместить все данные, то остаток данных досылается следующим репортом. За символом завершения пакета `'\n'` (2.1) должен следовать символ `'\0'` (символ с кодом 0).

1.2 RS-232

1.2.1 Термометры подключаются к интерфейсу RS-232 компьютера с помощью стандартного нуль-модемного кабеля, схема распайки которого приведена на рисунке 3. Достаточен минимальный вариант из трех сигналов TxD, RxD и SG.

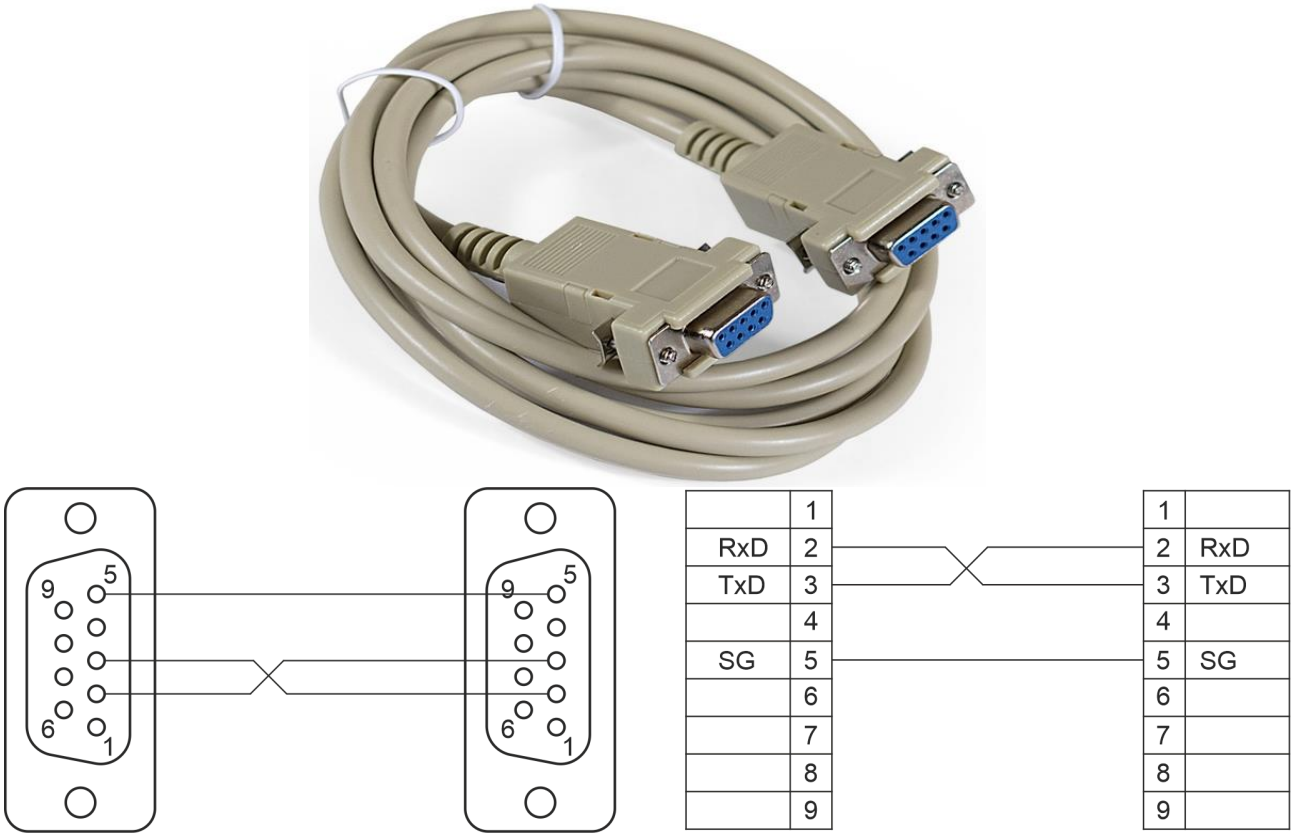

Рисунок 3

1.2.2 Для правильной работы интерфейса необходимо установить одинаковые настройки скорости, паритета и количества стоп-битов как на термометре, так и в компьютере. По умолчанию в термометре используются следующие настройки: скорость 115200, паритет — нет, стоп-биты — 1. Изменить эти настройки можно через меню прибора.

1.3 RS-485

1.3.1 Распайка контактов разъема RS-485, установленного в термометре показана на рисунке 4. Для подключения термометра необходимо самостоятельно изготовить соответствующий кабель.

1	GND
2	GND
3	GND
4	GND
5	GND
6	+5V
7	B
8	A
9	

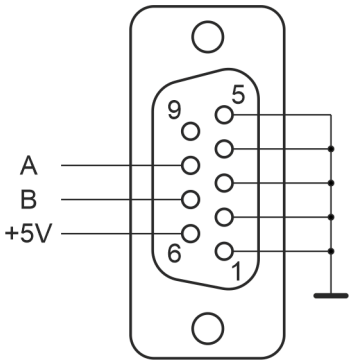

Рисунок 4

1.3.2 Для правильной работы интерфейса необходимо установить одинаковые настройки скорости, паритета и количества стоп-битов как на термометре, так и в компьютере. По умолчанию в термометре используются следующие настройки: скорость 9600, паритет — нет, стоп-биты — 1. Изменить эти настройки можно через меню прибора.

2 СИНТАКСИС КОМАНД

2.1 Формат сообщений

2.1.1 Обмен данными с термометром выполняется по инициативе компьютера (хоста) путем передачи сообщения, которое представляет собой строку ASCII символов, содержащую команду и ее параметры, и заканчивающуюся символом новой строки (`'\n'`, код `10`). Ответ на команду термометр отправляет строкой, также заканчивающейся символом новой строки. В дальнейшем при описании команд, завершающий символ `'\n'` будет опускаться.

2.2 Принятые соглашения

2.2.1 При описании синтаксиса команд используются следующие соглашения.

- в квадратных скобках [] указываются необязательные параметры;
- в угловых скобках <> указываются параметры, требующие определенного значения;
- в фигурных скобках {} указывается допустимый набор параметров для данной команды, разделенных символом '|', из которых нужно выбрать один.

При составлении команды включать какие-либо скобки в команду не нужно.

2.3 Чувствительность к регистру

2.3.1 Команды не чувствительны к регистру символов. Можно произвольно использовать заглавные и строчные символы.

2.4 Формат команд

2.4.1 В общем случае команда состоит из собственно команды и набора параметров.

```
SENSor:FILTer:SIZE 10
```

Здесь `SENSor:FILTer:SIZE` команда, `10` — значение параметра.

Параметры должны отделяться от команды символом пробела `' '` (код 32).

Если у команды несколько параметров, то между собой они отделяются символом `','` (запятая). Например, `CLB:VCORection 1, 1000.0`

2.4.2 Команда может состоять как из одного слова, так и из нескольких. В этом случае слова команды разделяются символом `':'` (двоеточие).

2.4.3 В конце любого слова команды может быть числовой суффикс, уточняющий область действия команды. Например, `SENSor3:FILTer:SIZE 10`. Здесь число `3` в конце слова `SENSor` уточняет, что действие команды по установке глубины фильтра относится к результатам измерений датчика, подключенного к *третьему* каналу измерительного модуля.

2.4.4 Команда может быть записана в сокращенной или полной формах. В данном документе формы команд разделяются с помощью регистра символов. Заглавными буквами записана сокращенная форма. Например, команда `SENSor` имеет сокращенную форму `SENS`, а полную — `SENSOR`. Сокращенная форма используется для уменьшения размера сообщения, отправляемого термометру, а полная — для удобства чтения человеком.

2.4.5 Если в конце команды нераздельно от нее стоит знак вопроса `'?'`, то это команда-запрос, результатом которой является значение запрашиваемого параметра. Например, команда-запрос `SENSor<n>:FILTer:SIZE?` возвращает значение глубины фильтра в канале, а та же команда `SENSor<n>:FILTer:SIZE {<val>}` устанавливает значение глубины фильтра.

2.4.6 В числах разделителем целой и дробной части является точка `'.'`.

2.5 Ответы термометра

2.5.1 Приняв и обработав команду, термометр отправляет один из следующих ответов:

- значение запрошенного параметра, в ответ на команду-запрос;
- строка `ok`, если команда выполнена;
- строка `failed`, если команда не выполнена;
- сообщение об ошибке, если команда составлена неправильно.

2.6 Сообщения об ошибках

2.6.1 Если команда составлена неправильно, то термометр возвращает соответствующее сообщение об ошибке.

2.6.2 Формат сообщения имеет вид: `!,<код ошибки>,<краткое описание>`.

Сообщение об ошибке начинается с символа `'!'` (восклицательный знак), затем через запятую идут код ошибки и краткое описание ошибки на английском языке. Список возможных ошибок представлен в таблице 1.

Таблица 1

Ошибка	Описание
-109, Missing parameter	Не указан параметр(ы), требующийся для команды.
-114, Header suffix out of range	Числовой суффикс в команде (2.4.3) вне допустимого диапазона.
-224, Illegal parameter value	Указано некорректное значение параметра. Например, вместо ожидаемого числа указана строка символов.

2.7 Термометр с точки зрения компьютера (хоста)

2.7.1 Со стороны компьютера термометр видится как плата человеко-машинного интерфейса (HMI), которая взаимодействует с компьютером посредством USB, RS-232, RS-485 с одной стороны, и с измерительными модулями с другой.

2.7.2 У платы HMI свой набор команд. У измерительных модулей — свой, одинаковый для всех модулей.

2.7.3 Плата HMI может пересылать команды от компьютера к указанному измерительному модулю и возвращать ответ обратно.

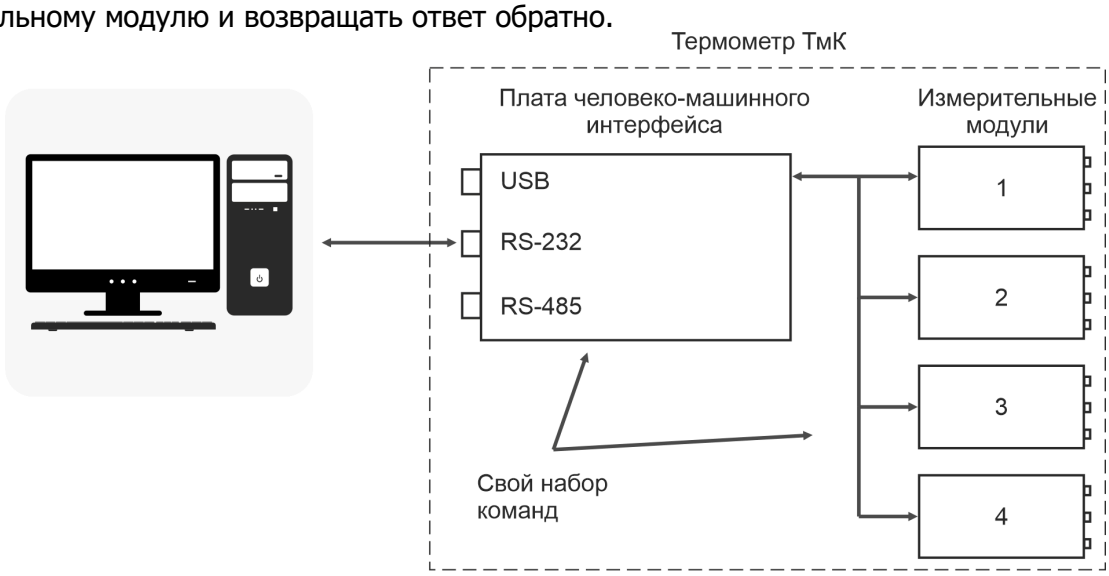

Рисунок 5

3 КОМАНДЫ HMI

3.1 *IDN? — чтение идентификационной строки

`*IDN?`

Читает идентификационную строку, состоящую из нескольких полей, разделенных запятой: `TmK,<serial>,<version>,<build-date>`

Здесь `TmK` — наименование термометра,

`<serial>` — заводской номер термометра,

`<version>` — версия прошивки термометра,

`<build-date>` — дата и время создания прошивки.

Пример.

`*idn?`

`TmK,00000000,2.4.3/3,11:15:38 Aug 29 2022`

3.2 *RST — сброс термометра

`*RST`

Выполняет полный сброс термометра.

На эту команду термометр не отвечает.

3.3 PASS — передает команду измерительному модулю

`PASS<m> 'command for module n'`

Передает измерительному модулю с номером `<m>` команду, записанную внутри одинарных кавычек. Ответ измерительного модуля передает обратно хосту.

`<m>` — номер измерительного модуля. Число в диапазоне от 1 до 4.

`command for module` — команда для модуля.

Пример.

`pass2 'tstat:t?'` (запрос температуры внутреннего термостата у второго модуля)

`40.01`

3.4 ConFiG? — возвращает список готовых к работе модулей

`ConFiG?`

Возвращает список готовых к работе измерительных модулей в виде строки `x,x,x,x`.

Где `x` — номер готового к работе модуля.

Пример.

`cfg?`

`1,2` (модули номер один и два готовы к работе)

3.5 ModuleSTAte? — возвращает состояние всех модулей

`ModuleSTAte?`

Возвращает состояние всех измерительных модулей в виде строки `S1,S2,S3,S4`.

Где `Sx` — число от 0 до 3, обозначающее состояние соответствующего модуля.

Состояние модуля может принимать следующие значения:

`0` — модуль не инициализирован;

`1` — в процессе инициализации модуль не обнаружен;

`2` — модуль готов к работе;

`3` — модуль отключен из-за возникших в нем ошибок.

Пример.

`msta?`

`2,2,1,1` (модули номер один и два готовы к работе, модули три и четыре не обнаружены).

4 КОМАНДЫ ИЗМЕРИТЕЛЬНОГО МОДУЛЯ

Команды выбранному измерительному модулю отправляются с использованием команды `PASS` (3.3).

Далее запись `<n>` в командах обозначает число от 1 до 3, указывающее номер канала, которому адресована команда.

4.1 *IDN — чтение идентификационной строки

`*IDN?`

Читает идентификационную строку, состоящую из нескольких полей, разделенных запятой: `TERMEX,MPSU,<serial>,<version>,<build-date>`

Здесь `TERMEX` — наименование предприятия-изготовителя,

`MPSU` — наименование измерительного модуля,

`<serial>` — заводской номер модуля,

`<version>` — версия прошивки модуля,

`<build-date>` — дата и время создания прошивки.

Пример.

```
pass1 '*idn?'
TERMEX,MPSU,220601,2.4.5/5,09:04:25 Aug 26 2022
```

4.2 *RST — сброс измерительного модуля

`*RST`

Выполняет сброс измерительного модуля.

4.3 MEAS? — чтение результатов измерений

`MEASurement<n>? [flags]`

Читает результаты измерений в указанном канале.

`flags` — необязательный параметр, набор битовых полей, уточняющих какой именно результат(ы) надо предоставить. Если параметр не указан, то выводится фильтрованное значение температуры в канале. Значение `flags` указывается в десятичном виде.

Поля параметра `flags` (для удобства указаны в шестнадцатеричном виде). Если соответствующий бит установлен, то к результирующей строке через пробел добавляется соответствующее значение.

`0x01` — фильтрованное значение температуры;

`0x02` — значение температуры;

`0x04` — фильтрованное значение измеряемой величины (напряжение или сопротивление);

`0x08` — значение измеряемой величины;

`0x10` — признак установился ли фильтр (1) или еще идет переходный процесс (0) в нем;

`0x20` — измерительный статус.

Измерительный статус — набор битовых полей, несущих информацию о корректности работы микросхемы АЦП во время проведения измерения в канале.

`bit0` — был (1) или не был (0) сбой в работе АЦП;

`bit1` — была (1) или не была (0) перегрузка входов АЦП. Если `bit0` установлен, то `bit1` может принимать произвольные значения.

Если значение измерительного статуса не нулевое, то результат измерений не корректен.

Пример.

`pass1 'meas2?'` (запрос значения фильтрованной температуры в канале 2 модуля 1)

`-0.002`

`pass1 'meas3? 49'` (49d = 0x31 запрос значений фильтрованной температуры (0x01), установлен ли фильтр (0x10), измерительный статус (0x20) для канала 3 модуля 1)

`100.015 0 0` (температура 100.015°C, фильтр не установился, измерительный статус — ошибок АЦП нет, результат корректен)

4.4 SENS:EN — включение канала

`SENSor<n>:ENable?`

`SENSor<n>:ENable {0|1}`

Первый вариант команды читает текущее состояние указанного канала. Второй — включает (1) или отключает (0) канал.

Пример.

`pass1 'sens2:en?'` (второй канал первого модуля включен?)

`0` (нет)

`pass1 'sens2:en 1'` (включить второй канал первого модуля)

`ok`

4.5 SENS:FUNC — режим работы канала

SENSor<n>:FUNCtion?

SENSor<n>:FUNCtion { V|R1|R2 }

Первый вариант команды читает текущий режим измерений указанного канала. Второй — устанавливает режим измерений.

Установленный режим измерений может противоречить назначенному для канала типу датчика. В этом случае командами **MEM:SENS:TYPE** (4.7) и **MEM:SENS:COEF** (4.8) следует установить необходимые тип датчика и его коэффициенты.

При изменении режима измерений установка нуля и относительные измерения в канале отключаются.

V — режим измерений напряжения;

R1 — режим измерений сопротивления, с током питания датчика не более 1.0 мА;

R2 — режим измерений сопротивления, с током питания датчика не более 0.1 мА;

Пример.

pass1 'sens2:func?' (режим измерений второго канала первого модуля?)

r1 (режим измерений сопротивления с током 1.0 мА)

pass1 'sens2:func v' (установить режим измерений напряжения)

ok

4.6 SENS:FILT — управление фильтрацией

SENSor<n>:FILTer:SIZE?

SENSor<n>:FILTer:SIZE {<val>}

Первый вариант команды читает текущую глубину фильтра указанного канала. Второй — устанавливает глубину фильтра.

<val> — целое число в диапазоне от 1 до 100.

Пример.

```
pass1 'sens2:filt:size?'
10
pass1 'sens2:filt:size 20'
ok
```

SENSor<n>:FILTer:LEVel?

SENSor<n>:FILTer:LEVel {<val>}

Первый вариант команды читает текущий порог фильтра указанного канала. Второй — устанавливает порог фильтра.

<val> — вещественное число в диапазоне от 0 до 1.0E6.

Пример.

```
pass1 'sens2:filt:lev?'
1.0e-01
pass1 'sens2:filt:lev 2.0'
ok
```

SENSor<n>:FILTer:SET?

Читает признак того установился (1) или нет (0) фильтр в указанном канале.

Пример.

```
pass1 'sens2:filt:set?'
1
```

SENSor<n>:FILTer:FLUSh

Выполняет сброс фильтра в указанном канале.

Пример.

```
pass1 'sens2:filt:flush'
ok
```

4.7 MEM:SENS:TYPE — тип датчика

MEMory:SENSor<n>:TYPE?

MEMory:SENSor<n>:TYPE {<val>}?

Первый вариант команды читает тип датчика, назначенного указанному каналу. Второй — устанавливает новый тип датчика.

Установленный новый тип датчика может не соответствовать текущему режиму измерений (4.5). В этом случае командой **SENS:FUNC** следует установить режим измерений, соответствующий установленному типу датчика.

<val> — целое число от 0 до 22, которым кодируется тип датчика, как указано в таблице 2.

Таблица 2

Код	Тип датчика	Код	Тип датчика
0	не назначен	12	ТП S
1	термопара (ТП) A-1	13	ТП T
2	ТП A-2	14	ТП Au/Pt
3	ТП A-3	15	ТП Pt/Pd
4	ТП B	16	ТП ППО
5	ТП E	17	ТП ПРО
6	ТП J	18	термометр сопротивления (ТС) платиновый
7	ТП K	19	ТС медный (ТСМ)
8	ТП L	20	ТС никелевый (ТСН)
9	ТП M	21	эталонный термометр сопротивления (ЭТС)
10	ТП N	22	термистор NTC
11	ТП R		

Пример.

```
pass1 'mem:sens2:type?'
18                          (текущий тип датчика ТСП)
pass1 'mem:sens2:type 1'    (установить в качестве датчика ТП A-1)
ok
pass1 'sens2:func v'        (приводим режим измерений в соответствие с датчиком)
ok
```

4.8 MEM:SENS:COEF — коэффициенты датчика

MEMory:SENSor<n>:COEFficient<c>?

MEMory:SENSor<n>:COEFficient<c> {<val>}

Первый вариант команды читает текущее значение указанного коэффициента датчика. Второй — устанавливает новое значение коэффициента.

<c> — индекс коэффициента. В таблице 3 показаны наборы коэффициентов и соответствующих индексов для каждого типа датчика. Первому коэффициенту соответствует индекс [1], второму — индекс [2] и т.д.

! *Наличие константных значений 0.0 и 1.0 в наборе коэффициентов ТСП (КВД и полином) обязательно.*

! *Команда* **MEM:SENS:COEF** *должна использоваться в паре с командой* **MEM:SENS:TYPE**

Таблица 3

Тип датчика	Набор коэффициентов	Индексы
Все ТП, кроме ППО и ПРО	$T_{\text{ск}}$	[1]
ТП ППО	$T_{\text{ск}}, E_{300}, E_{400}, E_{500}, \dots, E_{1000}, E_{1100}, E_{1200}$	[1 – 11]
ТП ПРО	$T_{\text{ск}}, E_{600}, E_{700}, E_{800}, \dots, E_{1600}, E_{1700}, E_{1800}$	[1 – 14]
ТСМ, ТСН	R_0, A, B, C	[1 – 4]
ТСП (КВД)	$R_0, A, B, C, 0.0, 0.0$	[1 – 6]
ТСП (полином)	$a_0, \mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3, \mathrm{a}_4, 1.0$	[1 – 6]
ЭТС	$R_{0.01}, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{W}_{660}, M$	[1 – 7]
Термистор	$\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}$	[1 – 4]

Пример.

pass1 'mem:sens1:type 18' (устанавливаем тип датчика ТСП для 1-го канала 1-го модуля)

ok

pass1 'mem:sens1:coef1 1000.0' (устанавливаем весь набор коэффициентов)

ok

pass1 'mem:sens1:coef2 3.9083e-3'

ok

pass1 'mem:sens1:coef3 -5.775e-7'

ok

pass1 'mem:sens1:coef4 -4.183e-12'

ok

pass1 'mem:sens1:coef5 0'

ok

pass1 'mem:sens1:coef6 0'

ok

pass1 'mem:store3' (сохраняем внесенные изменения)

ok

! *Команды* **MEM:SENS:TYPE** *и* **MEM:SENS:COEF** *не сохраняют изменения в энергонезависимой памяти термометра. Установленные коэффициенты и тип датчика будут действовать до выключения прибора. После его включения, установки вернуться к тем, что были сделаны с помощью органов управления на термометре. Чтобы сохранить изменения, сделанные данными командами, надо выполнить команду* **MEMory:STORe3** *для того модуля, в канал которого вносились изменения.*

4.9 CLB:VCOR — коррекция по напряжению

`CLB:VCORrection <n>, <vext>`

Выполняет коррекцию измерительного модуля по напряжению.

`<n>` — номер канала, к которому подключено образцовое напряжение;

`<vext>` — значение образцового напряжения, мВ.

Команда возвращает ноль, если коррекция выполнена без ошибок или код ошибки (таблица 4), если коррекция не выполнена.

Таблица 4

Код ошибки	Описание
0	Команда выполнена, ошибок нет.
1	Указанный канал отключен.
2	Измеренный сигнал в указанном канале отличается от заданного больше чем на 10%.
3	В канале установлены неподходящие параметры фильтра. Глубина фильтра должна быть больше 10, порог больше 0.05.
4	Фильтр не установился.
5	В расчетах появились недопустимые значения.
6	Перегрузка входов.
7	Режим измерений в канале не соответствует команде.

Пример.

`pass1 'clb:vcor 1, 1000'` (выполнить коррекцию по 1-му каналу с напряжением 1000 мВ)

`0` (коррекция выполнена)

4.10 CLB:RCOR — коррекция по сопротивлению

`CLB:RCORrection <r>, <n>, <rext>`

Выполняет коррекцию измерительного модуля по сопротивлению.

`<r>` — значение в диапазоне [1, 2], обозначающее режим измерений, для которого выполняется коррекция. Значение 1 соответствует режиму `R1`, значение 2 — режиму `R2` (4.5).

`<n>` — номер канала, к которому подключено образцовое сопротивление;

`<rext>` — значение образцового сопротивления, Ом.

Команда возвращает ноль, если коррекция выполнена без ошибок или код ошибки, если коррекция не выполнена. Описание кодов ошибок указано в таблице 4.

Пример.

`pass1 'clb:rcor 2, 1, 1000'` (выполнить коррекцию по 1-му каналу с образцовым сопротивлением 1000 Ом для режима измерений R2)

`0` (коррекция выполнена)

4.11 TSTAT:T? — чтение температуры термостата

```
TSTAT:T?
```

Читает значение текущей температуры внутреннего термостата измерительного модуля.

Пример.

```
pass1 'tstat:t?'
40.002
```

4.12 TSTAT:P? — чтение мощности нагрева термостата

```
TSTAT:P?
```

Читает значение текущей мощности нагрева (от 0 до 100%) внутреннего термостата измерительного модуля.

Пример.

```
pass1 'tstat:p?'
52.7
```

4.13 RTD:KVD — вычисление температуры с использованием функции КВД

```
RTD:KVD <R0>,<A>,<B>,<C>,<Rx>
```

Вычисляет значение температуры, используя функцию Каллендара-Ван Дюзена (КВД) с указанными значениями коэффициентов.

`<R0>,<A>,<B>,<C>` — значения коэффициентов функции КВД;

`<Rx>` — значение сопротивления (Ом), для которого нужно рассчитать соответствующую температуру.

Пример.

```
pass1 'rtd:kvd 1000, 3.9083E-3, -5.7750E-7, -4.1830E-12, 1089.63'
23.011
```

4.14 RTD:POLY — вычисление температуры с использованием полинома

```
RTD:POLY <a0>,<a1>,<a2>,<a3>,<a4>,<Rx>
```

Вычисляет значение температуры, используя полиномиальную функцию с указанными значениями коэффициентов.

`<a0>…<a4>` — значения коэффициентов полинома. `<a0>` — коэффициент при нулевой степени аргумента, `<a1>` — при первой и т.д.

`<Rx>` — значение сопротивления (Ом), для которого нужно рассчитать соответствующую температуру.

Пример.

```
pass1 'rtd:poly -243.91, 2.3247, 1.1942E-03, -5.3349E-07, 1.8427E-09, 110.01'
25.842
```

4.15 RTD:ITS — вычисление температуры с использованием МТШ-90

`RTD:ITS <R0.01>,<a>,<b>,<c>,<d>,<W660>,<M>,<Rx>`

Вычисляет значение температуры, используя интерполяционное уравнение МТШ-90 с указанными значениями коэффициентов.

`<R0.01>,<a>,<b>,<c>,<d>,<W660>,<M>` — значения коэффициентов. Если какой-либо коэффициент отсутствует, то его значение принимают равным нулю.

`<Rx>` — значение сопротивления (Ом), для которого нужно рассчитать соответствующую температуру.

Пример.

```
pass1 'rtd:its 100.0164, -0.002091, -0.000481, 0, 0, 0, -0.002430, 100.36'
0.873
```

4.16 TC:CALCTEMP — вычисление температуры ТП

`TCouple:CALCTEMP <type>, <Tcj>, <mV>`

Вычисляет значение температуры ТП для указанных значений термоэдс (ТЭДС) и температуры свободных концов.

`<type>` — число в диапазоне от 1 до 15 (см. 4.7), определяющее тип ТП;

`<Tcj>` — температура свободных концов, °С;

`<mV>` — значение ТЭДС, мВ.

Пример.

```
pass1 'tc:calctemp 7, 0.0, 10.000' (ТП типа K)
246.230
```

4.17 TC:CALCEMF — вычисление термоэдс ТП

`TCouple:CALCEMF <type>, <t>`

Вычисляет значение ТЭДС (мВ) для указанного значения температуры, принимая, что температура свободных концов равна нулю.

`<type>` — число в диапазоне от 1 до 15 (см. 4.7), определяющее тип ТП;

`<t>` — температура ТП, °С.

Пример.

```
pass1 'tc:calcemf 7, 246.230' (ТП типа K)
10.0000
```