DE/15/161

TC_TM66xx*

* Available for TC6621, TC6622 TM6602, TM6612 and TM6630.

Temperature calibrator

SCPI Commands

Version	Date	Changes
1.0	16/12/2015	Original version
1.1	17/12/2015	TC measurement update
1.2	18/12/2015	Simulation commands update

1. General Protocol

Transmission format: 8 bits, 1 stop, No parity, no flow control
Baud rate: 115200

Commands format: type IEEE 488-2, 'SCPI-like'

- Last character command line is **'\n'** (decimal code10, hexadecimal 0x0A)
- It can have one or more commands separated by character **';'**
- Each command has a header followed by one or more spaces and a 0 or several arguments, separated by character **','**
- Command header can be built by or more key words separated by character **':'**, succession of these key words are defining a tree diagram allowing classifying all the commands in subsets,
- All command header asking for a response are finished by character **'?'.** This character belongs to the header (no space required)
- When wrong command is sent, instrument does not reply (for commands requiring a response).
- Instrument can propose a table of the last 5 error codes, managed in the FIFO. Command « ERR? » which allows the extraction from the older error code. « *CLS » erase FIFO content.

Examples:

REM	Place instrument as remote use (lock keypad)
SENS:FUNC RTD	Select function OHM
SENS:RES:RANG PT100	Define range of function OHM
LOC	Place instrument as local use (unlock keypad)
*IDN?	Instrument identification request. The command is finished by **'?'**, therefore a response is needed.
MEAS:RES? 400,10	Realize 10 measurements et do the average (Range 400 ohm=PT100)

Instrument responses are finished by « \r\n »

1.1 Communication general principles

- Switch the instrument in Remote sending « REM » before sending other commands.
- Erase FIFO of old error codes using command « *CLS »
- Send a command
- If command is not a request, verify it has been performed sending command ERR? then wait for the response.
- Before closing the connexion, switch the instrument in Local mode (command 'LOC')

2. General commands for instrument control

REMote

Switch the instrument as remote (keypads is locked).

LOCal

Switch the instrument to LOCAL and and lock the keypad

***CLS**

Erase FIFO of old error codes.

ERRor?

Extract from FIFO the older error code.
Instrument response:
<digital code>, <error text>
Example:
0, "No error"

***IDN?**

Identify instrument
Instrument response:
Example:
AOIP, TM6612 , 1234A A00 4567 A

3. Setting and exploitation of measurement

3.1 Measurement function setting selection

```
SENSe
        :TCouple
                :TYPE      < TC >
                :DISPlay { MV | CEL | K | FAR }
                :RJUNction    <Value>
                        :TYPE {INTernal | DISabled | FIXed

        :RTD
                :TYPE   < RTD >
                :DISPlay { OHM | CEL | K | FAR }
```

Thermocouples mnemonic list:

K, T, J, E, N, U, L, S, R, B, C, PL, MO, XA_K, XK_L, XK68

RTD mnemonic list :

PT50, PT100, PT200, PT500, PT1000,
PT100_3916, PT100_3926,
NI100, NI120, NI1000,
CU10, CU50,
PTP46_1_3910, PTP50_1_3911, P_50P_1_3911, PTP100_1_3910, P_100P_1_3911, PTP500_1_3910,
CU50_1_4260, CUP50_1_4280, P_50M_1_4280, CU53_1_4260, CU100_1_4260, CUP100_1_4280,
P_100M_1_4280

3.2 Annex functions settings

```
SENSe
        :FILTer { ON | OFF }
                :COUNT <measurement number>

        :NULL { ON | OFF }
                :AMPLitude  { < valeur>}
                :TARE                                   // ->Display to 0

        : SCALing { ON | OFF }
                :SIZE <number of points of the scaling: 2 to 10>
                :SIZE?
                :POINt <num(1 to 10)>, <Xvalue Nrf>, <Yvalue Nrf>
                :POINt? <num>                          // Response : Value of X, Value of Y
                :UNIT <chain of 1 of 4 characters between “”>
                :UNIT?
                :ACCuracy <number of resolution>
                :ACC?
```

3.3 Exploitation Commands

```
SENS
        :HOLD
        :TRIG
        :RUN
```

3.4 Measurement request

MEASure? [<*number of meas to be averaged*>]

MEASure
:VOLTage? [{78MV}] [,<*number of meas to be averaged* >]]
:RESistance? [{ 400 OHM | 3600 OHM }[,< *number of meas to be averaged* >]]
:TEMPerature? { TC | RTD }[, <*probe type*>[,< *number of meas to be averaged*]]]

4. Declaration of calibrated sensors

(Calibrated Sensors)

CSENsor
:LOAD <*number (1 to 5) of the probe*> (load the current setting into working memory)
:NAME ''<*name of 15 caracters maximum*> ''
:CDATe <*year>,<month>,<day*> *(calibration date)*
:TYPE { RTD}, <*sensor type*>
:**SIZE** <*number of point in the calibration table : 1 to 4*>
:**UNIT** { TEMPerature | RESistance }
:POINt <*n° of point(1 to Size)*> **,** <*Tempér. true in ℃>, <read value (in ℃, Ohm or V according to unit UNIT) >*
:SAVE <*number (1 to 5) of the sensor*> (save working memory)

Examples:

Declaration of Pt100 calibrated in 2 points:

```
CSEN :NAME ''MY_PT100'' ; CDATE 2007,2,15 ; TYPE RTD, PT100 ; UNIT RESISTANCE ; SIZE 2
CSEN :POINT 1,  0 CEL, 101.234 OHM; POINT 2, 100.5 CEL, 140.162 OHM
CSEN :SAVE 2
```

To use one of the above mentioned sensor, indicate its mnemonic in the command SENSe : TYPE
Mnemonic of the 5 sensors: CSENSOR1, CSENSOR2, CSENSOR3, CSENSOR4, CSENSOR5

Example :

```
SENS:RTD:TYPE CSENSOR2
SENS2:TC:TYPE CSENSOR1
```

CSENsor?
Send back settings of the current loaded sensor

Example:

```
#0\r\n
#0
NAME 'MY_PT100'
     CDATE 2007,2,15
               TYPE RTD,PT100
                        SIZE 2
                            UNIT RESISTANCE
                                     POINT 1,0.00 CEL,0.0M
                                                 M

\r\n
```

5. Setting and exploitation of simulation

5.1 Definition of each parameter

```
SOURce
        :VOLTage
                :RANGe { 100MV }


        :RESistance                                         For WEM41021-022A/B
                :RANGe { 400OHM | 3600OHM}
                :CURRENT { PULSed | CONTinuous}


        :RESistance                                         For WEM41021-022B1/C
                :RANGe { 400OHM | 3600OHM | 100KOHM } , { PULSed | CONTinuous} (1) , {1MA | 4MA} (2)
                :CURRENT { PULSed | CONTinuous} , {1MA | 4MA} (2),(3)

        :TCouple
                :TYPE  <thermocouple mnemonic>
                :DISPlay { MV | CEL | K | FAR }
                :RJUNction
                        :TYPE {INTernal | DISabled | FIXed}
                :RJUNction    <Value>                       // when type = FIXED, in ℃ if not specified

        :RTD
                :TYPE  < RTD mnemonic >
                :DISPlay {OHM | CEL | K | FAR }
                :CURRENT { PULSed | CONTinuous}
```

Thermocouples mnemonic list:

K, T, J, E, N, U, L, S, R, B, C, PL, MO, XA_K, XK_L, XK68

RTD mnemonic list :

PT50, PT100, PT200, PT500, PT1000,
PT100_3916, PT100_3926,
NI100, NI120, NI1000,
CU10, CU50,
PTP46_1_3910, PTP50_1_3911, P_50P_1_3911, PTP100_1_3910, P_100P_1_3911, PTP500_1_3910,
CU50_1_4260, CUP50_1_4280, P_50M_1_4280, CU53_1_4260, CU100_1_4260, CUP100_1_4280,
P_100M_1_4280

5.2 Selection of emission function

```
SOURce
        :FUNCtion {VOLTage | |RESistance |TCouple |RTD | }
```

5.3 Setting of annex functions

```
SOURce
        :SCALing { ON | OFF }
                :SIZE <number of point of scaling : 2 to 10>
                :POINt <num(1 to 10)>, <Xvalue Nrf>, <Yvalue Nrf>
                :POINt? <num>                        // Response :  X value, Y Value
```

```
:UNIT <chain of 4 characters “”>
:ACCuracy <resolution points>
```

5.4 Sending a value

SOURce

```
    :VOLTage      <value>                     // in V if no unit specified
    :RESistance   < value >                   // in Ohm if no unit specified
    :TCouple      < value >             // in ℃ if no unit specified
    :RTD          < value >             // in ℃ if no unit specified
```

Examples:

SOUR:VOLT 0.08	
SOUR:VOLT 80 mV	Send voltage 80 mV
SOUR:RTD 123	Simulate temperature 123℃ (whatever i s the unit defined in setup) If temperature instrument unit is ℉, value 123℃ is converted to ℉ for display.
SOUR:RTD 123 FAR	Simulate temperature 123℉ (whatev er is the unit defined in setup)

6. Setting and exploitation of data memory

6.1 Setting

TRACe

: SIZE	<number of measurements>	memory size
:TIMer	<period>	in seconds
:TRIGger		
:SOURce	**{ IMMediate | MANual | INTernal}**	Event triggering POST counting
:LEVel	<level>	(If SOURce = INTernal)
:SLOPe	**{ POSitive | NEGative }**	
:POST	< number of measurements >	N° of measurement count ed after the Trig

TIMer period can only have the following values:
0.5s, 1s, 2s, 5s, 10s, 20s, 30s, 1mn, 2mn, 5mn, 10mn, 20mn, 30mn
If another value is indicated, it will be the lower valid value that will be used.

Example: TRACe :TIMer 3mn -> Used period will be 2 and not 3

LEVel and SLOPe are taken into account only if SOURce = INTernal.
LEVel the input level (in current unit) corresponding to the trigger.
If SLOPE = POSitive, trigger is performed when measurement is upper or = to the LEVel value
If SLOPE = NEGative, trigger is performed when measurement is lower or = to the LEVel value

POST is taking into account when SOURce = MANual or INTernal
Programmed value is the number of measurements to be stored after TRIG detection.

Example:
TRAC :SIZE 100 ; TIM 0.5s ; TRIG :SOUR INT; LEV 100.5; SLOP POS; POST 50

- Buffer size is 100 measurements
- A value is stored every 0.5 seconds
- Trigger is performed when measurement is upper than 100,5
- 50 measurements have to be stored before stopping the record.

6.2 Acquisition

INITiate

Erase trace and start recording ad trigger monitoring (if TRIG :SOUR = MAN or INT)
If TRIG :SOUR = IMM, POST value is not taken into account and trace records exactly before stop.
(Equal keyboard command **Measures | Run**)

ABORt

Stops records in the trace buffer
(Equal keyboard command **Measures | Stop**

***TRG**

If TRIGger:SOURce = MANual, start counting POST records before stop.

6.3 Measurement readings

DATA: POINts? Give the number of measurements available in the trace buffer.

DATA: HEADer?

Give the trace header as binary block with a defined length.

Example:

#297\n	97 characters
W/O NAME\n	Burst name (15 characters max)
300 POINTS\n	Number of values
PROG\n	Recording type : PROG or FREE
10/05/2005 14 :40 :00\n	Date and time of 1st record
10/05/2005 14 :45 :00\n	Date and time of last record
TC K\n	Measurement Function and range
℃\n	Measurement Unit (to be confirmed line af ter line if Vdc Auto)
2\n	Number of resolution point (to be confirmed by the range if Vdc Auto)
SCALING OFF\n	Scaling: SCALING ON or SCALING OFF
TARE OFF\n	TARE ON or TARE OFF
\n	

DATA? [<*1st measurement index*>, [<*number of measurement*>]]

<1st measurement index > : between 1 and the number returned by DATA :POINts ? (1 if not specified)

< number of measurement > : number of measurement specified (1 if nothing specified)

Return timed and dated measurements as a binary block with a defined length.

Example:

```
#273\n
000000.0\t123.56789\tUNIT\n
000000.5\t123.56789UNIT\n
000001.0\t123.56789UNIT\n
\n
```

Each measurement with time and date is 24 bytes:

- 8 bytes time in seconds : 0.1 to 999999.9s
- 1 tabulation
- 9 bytes for measurement value
- 1 tabulation
- 4 bytes for unit
- 1 end of line

1 character ‘end of line' is sent between size indicator #ndddd and first measurement result. This character is numbered in the indicated number ddd.

Character 'End of line' ‘eventually sent after the last measurement (in case instrument has nothing else to send) is not numbered in the number ddd.

Recording in eeprom and reading

MEMory:DATA:SAVE "<Name>"
Record Trace in EEProm as a specified name

MEMory:DATA: COUNt?
Return number of traces in memory

MEMory:DATA: HEADer? *<number>*
Return trace header position <number>
<number> is a recording rank number in the memory trace
Most recent trace is number 1.
The older trace has the number returned by command MEMory :DATA?

Trace header is transmitted as a binary block with a defined length with the same structure as the one returned in response to DATA :HEADer ?

MEMory :DATA:LOAD *<number>*
Load the trace with the number indicated in the trace buffer.
<number> is a recording rank number in the memory trace.
Most recent trace is number 1.
The older trace has the number returned by command MEMory :DATA?
Once loaded in the trace buffer, burst is read using command DATA as described in chapter9.3

MEMory FREE?
Return the number of free bytes in the memory and the number of occupied bytes.
<Free Bytes>,*<occupied bytes>*

MEMory :DATA:DELete *<number>*
Delete from memory the trace with this number
(Rank number of all burst is then decremented)

MEMory :DATA: DELete ALL
Delete all records of the memory traces.

7. Adjustment

7.1 « IN » channel adjustment

```
CALibration
        [:SENSe]
                :VOLTage
                        :MEASure? {100MV} [,<n°of measurement to average > ]
                        :GAIN   {100MV},  < gain value >
                        :GAIN? {100MV}
                        :OFFSet {100MV},  < offset value >
                        :OFFSet? {100MV}
                        :DATE ? {100MV}                    // Dernière date de reg. du calibre

        [:SENSe]
                :RESistance
                        :MEASure? { 400 OHM | 3600 OHM}  [,<n°of measurement to average  >]
                        :GAIN   { 400 OHM | 3600 OHM} , < gain value >
                        :GAIN? { 400 OHM | 3600 OHM }
                        :OFFSet { 400 OHM | 3600 OHM },  < offset value >
                        :OFFSet? { 400 OHM | 3600 OHM}
                        :DATE? { 400 OHM | 3600 OHM | }
```

7.2 « OUT » channel adjustment

```
CALibration
        :SOURce
                :VOLTage
                        :MEASure? {100MV} [,<n°of measurement to average > ]
                        :GAIN   {100MV},  < gain value >
                        :GAIN? {100MV}
                        :OFFSet {100MV},  < offset value >
                        :OFFSet? {100MV}
                        :DATE ? {100MV}

               :RESistance
                       :ADJust
                                :OFFSet
                                        :ADJ    ADC                        Input Offsets adjustment
                                        :ADJ    DAC                        output Offsets adjustment

:                       :ADJust?  { 400OHM | 1000OHM | 3600OHM | },        Auto adjustment (linearization)

                        :GAIN { 400OHM | 1000OHM | 600OHM},{PULSed | CONTinuous},{1MA | 4MA}, (1)< gain
value >
                        :GAIN? { 400OHM | 3600OHM},{PULSed | CONTinuous},{1MA | 4MA} (1)
                        :OFFSet { 400OHM | 1000OHM | 3600OHM},{PULSed | CONTinuous},{1MA | 4MA}, (1)<
offset value >
                        :OFFSet? { 400OHM | 1000OHM | 3600OHM},{PULSed | CONTinuous},{1MA | 4MA} (1)
                        :DATE ? { 400OHM | 1000OHM | 3600OHM} ,{PULSed | CONTinuous},{1MA | 4MA} (1)
```

Other commands relatives to instrument adjustment

CALibration

```
        :DATE <year>, <month>,<day>                          // calibration date
        :DATE?
        :REPort  <Chain of 0 to 50 characters > // Reference of calibration certificate
        :REPort?
        :SECure :STATe {OFF|ON}, <code 5930>                 // lock/unlock eeprom
        :SECure :STATe?
```