

KLINKMANN

MultIO application

User Instruction

Version 1.5

Klinkmann Automation

29.08.2007

Table of Contents

Table of Contents	ii
Revision History	ii
1. Operation	3
1.1 Temperature measuring	3
2. Contents of package	4
3. Configuring and running the MultiIO application	4
4. Configuration by using TCConfigurator	5
4.1 Autostart off/on	6
4.2 SETTINGS	6
4.3 INPUT LOW MESSAGES	8
4.4 INPUT HIGH MESSAGES	8
4.5 OUTPUT LOW MESSAGES and OUTPUT HIGH MESSAGES	9
4.6 REQUEST MESSAGES	9
4.7 REPLY MESSAGES	9
4.8 BATTERY	9
4.9 INPUT LOW DELAY	9
4.10 INPUT HIGH DELAY	9
4.11 ADC LIMITS	9
4.12 ADC EU	10
4.13 INPUT LOW NUMBER	10
4.14 INPUT HIGH NUMBER	10
4.15 INPUT LOW CALL NUMBER	10
4.16 INPUT HIGH CALL NUMBER	10
4.17 PHONES	10
4.18 SENSORS	10
5. Configuration and control by SMS	12
6. Related documentation	16

Revision History

Date	Reason For Changes	Version	Approved
17.01.2007	Initial document	1.0	
12.03.2007	“TCConfigurator” and “Configuration and control by SMS” sections modified	1.1	
02.04.2007	Description how to use thermo sensors added. “Contents of package” and “Configuring and running the MultiIO application” sections added	1.2	
12.04.2007	Description of SMS-commands for thermo sensors configuration added.	1.3	
06.07.2007	ADC LIMITS modified and ADC EU added. Support for decimal point values added in ADC REPLAY MESSAGES.	1.4	
29.08.2007	Example SMS for monitoring the temperature of specified sensor corrected.	1.5	

1. Operation

The purpose of TC65T's MultilO embedded application is:

- 1) detection of state changes of digital inputs and sending corresponding notification SMS-messages;
- 2) sending SMS-messages with analog inputs voltage values on request SMS-message or timer basis;
- 3) changing the state of outputs by sending the corresponding SMS-messages to TC65T modem;
- 4) voltage measuring on analog inputs and sending notification messages if voltage is out of limits;
- 5) processing of incoming voice calls to check the state of MultilO application;
- 6) detection of undervoltage and overvoltage conditions;
- 7) temperature measuring by using 1-Wire® Digital Thermometers (thermo sensors).

On startup modem searches its flash memory for a file named "MultilO.ini". If it doesn't find this file, it creates one with current (default) parameters; otherwise it reads parameters from this file.

The configuration of MultilO application can be done by using the configuration utility named "TCConfigurator" or by using SMS-messages.

1.1 Temperature measuring

The temperature measuring is implemented by using Dallas Semiconductor 1-Wire® Digital Thermometers (e.g. DS18S20), with variable number of monitored sensors possible. Operating temperature range is $-55^{\circ}\text{C} \dots +125^{\circ}\text{C}$ and accuracy is $\pm 0.5^{\circ}\text{C}$ over the range of $-10^{\circ}\text{C} \dots +85^{\circ}\text{C}$. The MultilO Java application operates with 1-Wire® Digital Thermometers using RS232 serial interface of TC65T modem. Modem sends control signals and receives responses through RS232 to 1-Wire® adapter.

During the MultilO application start-up procedure the following is done in case 1-Wire® net is connected:

1. Searching the TC65T modem flash memory for a file named "MultilO.ini" (MultilO configuration residing on modem). If this file does not exist, the new one is created with current (default) parameters; otherwise it reads parameters from this file.
2. Searching for RS232 to 1-Wire® adapter presence. If it is not detected, MultilO application continues to work without 1-Wire® functionality.
3. Setting the parameters of DS2480B - serial 1-Wire® line driver used in RS232 to 1-Wire® adapter. See adapter documentation [3]...[6] for more information.
4. Searching for 1-Wire® Digital thermo sensors.
5. Writing the alarm levels of 1-Wire® Digital thermo sensors detected into their internal nonvolatile EEPROM.
6. Reading the current temperatures of all 1-Wire® Digital thermo sensors detected and monitored.
7. Updating the modem MultilO.ini file by writing the data about sensors detected, including their current temperature(s).
8. If RS232 to 1-Wire® adapter was detected, but no 1-Wire® sensor is connected, the notification SMS-message is sent to phone number(s) specified in MultilO configuration "PHONES/sensors" section (see below).
9. Executing a modem Java application task that searches for 1-Wire® Digital thermo sensors within specified time interval (defined in MultilO configuration "SETTINGS/Temperature scanning interval" field). Updating the modem MultilO.ini file accordingly to current situation on 1-Wire® net.

Because MultilO application regularly searches the 1-Wire® net for new devices, it is possible to add new devices on the fly - MultilO application will find them and update the modem MultilO.ini file accordingly. After adding new 1-Wire® sensors, you only might want to modify the high and low alarm temperature values according to your needs. It is also possible to read the current temperature of any sensor by sending the "read temperature" SMS-message to modem.

2. Contents of package

There are following files in MultiIO application delivery package:

1. MultiIO vx.x.pdf - MultiIO application User Instruction (this file) (x.x - the current version of instruction).
2. TCConfigurator.exe - Windows application program for Unitronics GPRS application configuration and enabling/disabling the modem autostart mode.
3. Empty.ini - TCConfigurator default configuration file.
4. TCConfiguratorMxx.pdf - TCConfigurator User Instruction (xx - the current version of instruction).

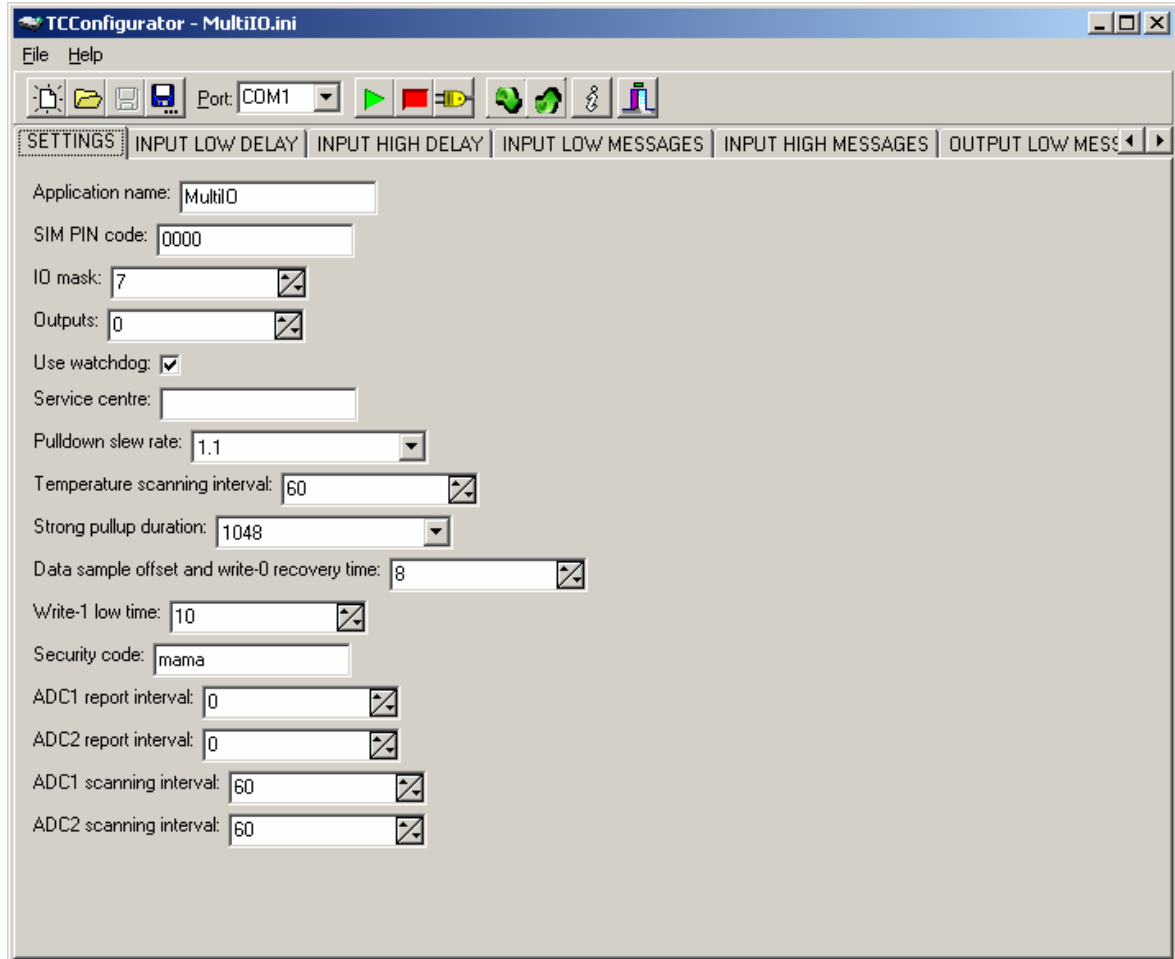
3. Configuring and running the MultiIO application

The TC65T modem is delivered with MultiIO application and default MultiIO configuration already installed on the modem. The modem is delivered with "autostart on" disabled. The following steps are necessary to adjust MultiIO application to current environment and start the MultiIO application:

1. Connect modem and computer COM ports via standard PC/modem cable.
2. Copy the TCConfigurator.exe and Empty.ini files (included in MultiIO application delivery package) to some location on computer.
3. Start TCConfigurator application - the TCConfigurator main window, containing only one parameter in SETTINGS area - the name of Java application ("MultiIO"), should appear.
4. Press the "Autostart off" or "Power down" button in TCConfigurator - the modem should power in "autostart off" mode.
5. Read the default configuration from modem - by pressing "Read settings from TC65..." button in TCConfigurator.
6. Adjust MultiIO application settings to current environment.
7. Save the modified configuration to file (under any name) on computer.
8. Download the modified configuration to the modem by pressing "Write settings to TC65..." button.
9. Press the "Autostart on" button in TCConfigurator - the "autostart enabled" message should appear. In case the "autostart on" mode is not enabled at first attempt, press the "Autostart on" button again.
10. Turn off the modem - by pressing the TCConfigurator "Power down" button or manually by pressing modem ON/OFF push button.
11. If 1-Wire® Digital thermo sensors are used, the modem/PC cable must be disconnected from modem and 1-Wire® net connected to modem COM port.
12. Turn on the modem - by pressing again the "Power down" button in TCConfigurator (if thermo sensors not used) or manually by pressing modem ON/OFF push button.
13. Now modem Java application is started and it will start run automatically after each modem power off/on.

4. Configuration by using TCConfigurator

The TCConfigurator program is a software utility that simplifies the configuration parameter entry for Java application (MultiIO) running on TC65T-modem and enables user to set the modem Java application to run automatically on startup. TCConfigurator program needs the modem and computer COM ports are connected via standard PC/modem cable. The TCConfigurator main window (see below) consists of a menu bar, tool bar and configuration area:



The Configuration area consists of multiple tabs that groups together correlated parameters.

After TCConfigurator application is started, at first it reads the default configuration file (Empty.ini), containing only one parameter – the name of Java application (“MultiIO”); this Empty.ini configuration file is supplied together with TCConfigurator application and must reside in same directory as TCConfigurator executable.

After default configuration file with Java application name is opened, TCConfigurator is ready for work. It is possible to read the existing configuration - from already saved configuration file residing in PC or from modem - by pressing “Read settings from TC65...” button. The configuration file contains information about modem Java application parameters and their current (or default) values. The values can be edited and saved on PC (under same or other name) and/or transferred to modem - after you have finished the editing of configuration, you can download it to the modem by pressing “Write settings to TC65...” button. The configuration files are ASCII format text files and can be edited also by any text editor, however it is not recommended to edit these files manually, because you can mistype parameter name or enter an unacceptable value. This will cause modem application to malfunction.

4.1 Autostart off/on

The modem Java application configuration file upload and download will work only if modem responds to computer AT commands. That means Java application must be not running on the modem – that means modem must be in “autostart off” mode. If “autostart on” is enabled, it must first be turned off to allow the execution of download/upload operations.

To turn autostart off (disable autostart), you must do the following actions:

1. Modem must be turned off manually – by pressing the modem ON/OFF push button.
2. If 1-Wire® Digital thermo sensors are used, the 1-Wire® net must be disconnected from modem and modem/PC cable connected to modem COM port.
3. Press the “Autostart off” button in TCConfigurator - the modem should power on and “autostart off” mode should be enabled (TCConfigurator should display “autostart disabled” message). In case the “autostart off” mode is not enabled at first attempt, repeat the sequence - press “on/off” button on modem and then “Autostart off” button in TCConfigurator.
4. Now you should be able to execute AT commands and upload/download configuration from/to modem.

To enable the “autostart on” mode (enable the modem Java application will start run automatically after modem power on), the following should be done in “autostart off” mode:

1. Press the “Autostart on” button - TCConfigurator should display the “autostart enabled” message. In case the “autostart on” mode is not enabled at first attempt, press the “Autostart on” button again.
2. Turn off the modem - by pressing the “Power down” button or manually by pressing modem ON/OFF push button.
3. If 1-Wire® Digital thermo sensors are used, the modem/PC cable must be disconnected from modem and 1-Wire® net connected to modem COM port.
4. Turn on the modem - by pressing again the “Power down” button in TCConfigurator (if thermo sensors not used) or manually by pressing modem ON/OFF push button.
5. Now modem Java application is started and it will start run automatically after each modem power off/on.

4.2 SETTINGS

This section contains Java application general parameters. The following settings can be entered:

Application name

The name of Java application (“MultiIO”) running on the modem.

SIM PIN code

The SIM card PIN code. This setting is necessary if SIM PIN security is enabled. If SIM PIN security is disabled, modem Java application ignores this setting.

IO mask

The active (currently used) digital inputs/outputs and analog inputs. The TC65T modem has 10 switchable (shared) digital inputs/outputs and 2 analog inputs. The IO mask value is the sum of following values:

INPUT 1 / OUTPUT 1 = 1
 INPUT 2 / OUTPUT 2 = 2
 INPUT 3 / OUTPUT 3 = 4
 INPUT 4 / OUTPUT 4 = 8
 INPUT 5 / OUTPUT 5 = 16
 INPUT 6 / OUTPUT 6 = 32
 INPUT 7 / OUTPUT 7 = 64
 INPUT 8 / OUTPUT 8 = 128
 INPUT 9 / OUTPUT 9 = 256
 INPUT 10 / OUTPUT 10 = 512
 ANALOG INPUT 1 = 1024

ANALOG INPUT 2 = 2048

For example, if used I/Os are: INPUT 1, OUTPUT 2, INPUT 6 and ANALOG INPUT 1, then IO mask = $1 + 2 + 32 + 1024 = 1059$

Outputs

The active (currently used) digital outputs. This value is the sum of following values (the same way as **IO mask**):

OUTPUT 1 = 1

OUTPUT 2 = 2

OUTPUT 3 = 4

OUTPUT 4 = 8

OUTPUT 5 = 16

OUTPUT 6 = 32

OUTPUT 7 = 64

OUTPUT 8 = 128

OUTPUT 9 = 256

OUTPUT 10 = 512

If pin is input or not used, add the value of **0**; if pin is output, add the corresponding value. For example, if you use pins 2 and 5 as inputs and 4 and 9 as outputs, then **Outputs** = $8 + 256 = 264$.

Use watchdog

Specifies if external hardware watchdog is used. Values: yes/no (checked/unchecked). If set to "yes", modem Java application will set GPIO1 to be used as output (changing "io mask" and "outputs" variables accordingly) and toggle it every 30 seconds to reset external hardware watchdog.

Service centre

Read Only field, where SIM card SMSC phone number is displayed after configuration is downloaded from modem. Modem Java application always uses SIM card SMSC address and ignores the setting in case entered here manually in TCCConfigurator.

The following 5 fields are reserved for common settings for 1-Wire® Digital Thermometers:

Pulldown slew rate

The PDSRC parameter of DS2480B - serial 1-Wire® line driver used in RS232 to 1-Wire® adapter. See related documentation [3]...[6] for details.

Temperature scanning interval

The 1-Wire® temperature sensors network check interval. This is a delay between successive checks, because checking duration itself is sensor count dependant and can take considerable time on great 1-Wire® nets. If set to 0 – no checking.

Strong pullup duration

The SPUD parameter of DS2480B - serial 1-Wire® line driver used in RS232 to 1-Wire® adapter. See related documentation [3]...[6] for details.

Data sample offset and write-0 recovery time

The DSO/W0RT parameter of DS2480B - serial 1-Wire® line driver used in RS232 to 1-Wire® adapter. See related documentation [3]...[6] for details.

Write-1 low time

The W1LT parameter of DS2480B - serial 1-Wire® line driver used in RS232 to 1-Wire® adapter. See related documentation [3]...[6] for details.

The recommended values for **Pulldown slew rate**, **Write-1 low time** and **Data sample offset and write-0 recovery time** parameters is given in table below:

Parameter	Active cable length					
	< 50 m	0 to 100 m	0 to 150 m	0 to 200 m	0 to 250 m	0 to 300 m
Slew rate	2.2 V/ μ s	2.2 V/ μ s	1.65 V/ μ s	1.37 V/ μ s	1.1 V/ μ s	0.83 V/ μ s
W1LT	8 μ s	8 μ s	9 μ s	10 μ s	11 μ s	12 μ s
DSO/W0RT	5 μ s	6 μ s	7 μ s	8 μ s	9 μ s	10 μ s

The W1LT + DSO/W0RT summary value cannot be specified greater as 22 μ s. If it is so, the W1LT parameter will be decreased (by RS232 to 1-Wire® adapter) to have W1LT + DSO/W0RT equal to 22 μ s. See related documentation [3]...[6] for details.

Security code

The access code required for configuration via SMS-messages. The default setting is "0000".

ADC1 report interval

Analog input 1 report interval. Reports are sent to specified phones, using SMS. If you don't want to receive report messages, set it to 0.

ADC2 report interval

Analog input 2 report interval. Reports are sent to specified phones, using SMS. If you don't want to receive report messages, set it to 0.

ADC1 scanning interval

Analog input 1 check interval in seconds. Must be greater than 0. Has significance only if analog inputs are used (IO mask).

ADC2 scanning interval

Analog input 2 check interval in seconds. Must be greater than 0. Has significance only if analog inputs are used (IO mask).

4.3 INPUT LOW MESSAGES

This section contains notify messages sent when digital inputs change from high to low state and when analog input values go below low limit. The "%d" in ADC message will be replaced with current analog input value in millivolts. If message for corresponding input is not specified, modem will not notify about this event.

The ADC1 and ADC2 INPUT LOW MESSAGE can contain "%dn" - it will be replaced with analog input current value in engineering units; the "n" specifies the number of digits after decimal point – for example, if ADC1 current value is 1.34 (scaled in engineering units, not in millivolts) and configured ADC1 INPUT LOW MESSAGE is "ADC1:%d1m - below low limit", then real SMS-message will contain "ADC1:1.3m - below low limit".

4.4 INPUT HIGH MESSAGES

This section contains notify messages sent when digital inputs change from low to high state and when analog input values go above high limit. The "%d" in ADC message will be replaced with current analog input value in millivolts. If message for corresponding input is not specified, modem will not notify about this event.

The ADC1 and ADC2 INPUT HIGH MESSAGE can contain "%dn" - it will be replaced with analog input current value in engineering units; the "n" specifies the number of digits after decimal point – for example, if ADC1 current value is 10.78 (scaled in engineering units, not in millivolts) and configured ADC1 INPUT HIGH MESSAGE is "ADC1:%d1m - above high limit", then real SMS-message will contain "ADC1:10.8m - above high limit".

4.5 OUTPUT LOW MESSAGES and OUTPUT HIGH MESSAGES

The messages configured in these sections can be used as SMS-messages to set outputs accordingly to low and high state.

4.6 REQUEST MESSAGES

The messages configured in this section can be used as SMS-messages to get current state of inputs and outputs. When sending the REQUEST MESSAGE to MultiIO application, it must precede with **Security code** and character #; for example, if **Security code** is 0000 and configured REQUEST MESSAGE is "ai1?" then SMS-message to be sent is: "0000#ai1?".

Important!

It is necessary to carefully check if configured REQUEST MESSAGE does not match with some of SMS commands used to configure the MultiIO application. For example, if configured REQUEST MESSAGE is "adc1?" then MultiIO application will try to recognize this SMS-message as a wrong SMS command for setting the "ADC out of limits" notification message – as this SMS command starts with 0000#adc1# (Security code 0000 is used); in this case MultiIO application will reply with error response, like "ADC number invalid: 1?".

4.7 REPLY MESSAGES

The messages configured in this section will be sent as response SMS-messages to "request messages" (for analog inputs as "report messages" in case analog input reporting is enabled).

The "%s" in input "reply message" will be replaced with the following string:

IN1=X,IN2=X, ...,IN10=X
where "X" = 0/1 – state of corresponding input.

The "%s" in output "reply message" will be replaced with the following string:

OUT1=X,OUT2=X, ...,OUT10=X
where "X" = 0/1 – state of corresponding output

The "%dn" in ADC "report message" will be replaced with analog input current value in engineering units; the "n" specifies the number of digits after decimal point – for example, if ADC1 current value is 32.344 (scaled in engineering units, not in millivolts) and configured ADC1 REPLAY MESSAGE is "ADC1=%d2 kg", then real SMS-message will contain "ADC1=32.34 kg".

4.8 BATTERY

This section contains notify messages sent when modem battery supply voltage goes below low or above high limit. The "%d" in messages will be replaced with current voltage values in millivolts. If message for corresponding condition is not specified, modem will not notify about this event.

4.9 INPUT LOW DELAY

Delay in seconds how long the low voltage must persist on input to be verified as valid and reported about it. Specified separately for each input.

4.10 INPUT HIGH DELAY

Delay in seconds how long the high voltage must persist on input to be verified as valid and reported about it. Specified separately for each input.

4.11 ADC LIMITS

This section is used to configure the low and high limits for analog inputs. In case ADC current value exceeds low or high limit, the corresponding INPUT LOW MESSAGE or INPUT HIGH MESSAGE will be sent to phone number(s) configured for this ADC in PHONES SECTION. The values entered in ADC LIMITS section are in engineering units (EU, same as configured in ADC EU section), not in mV (millivolts).

4.12 ADC EU

This section is used to configure the range of possible values for analog inputs ADC1 and ADC2. The minimum and maximum values must be entered in engineering units (EU) and in mV (millivolts).

The scaling between millivolts and EUs is done in the following way:

At MultiIO application startup, the analog input real physical range in millivolts supported by TC65T is determined - usually the minimum value is about -56...-57 mV and maximum value is about 5040...5070 mV. Then this real physical range and settings entered in ADC EU section are used to calculate the analog input current value in engineering units; for example, if "ADC EU low" is 1, "ADC EU high" is 15, "ADC mV low" is 100 mV and "ADC mV high" is 4900 mV, then in case analog input current measurement is 3200 mV (3.2V), the corresponding value in EU will be calculated as 10.17:

$$\left(\frac{3200 - 100}{4900 - 100} \right) \times (15 - 1) + 1 = \frac{3100}{4800} \times 14 + 1 = 0.6458 \times 14 + 1 = 9.1672 + 1 = 10.1672 \rightarrow \text{rounded to } 10.17$$

4.13 INPUT LOW NUMBER

The number of input state changes from high to low before the notification SMS is sent and/or call is performed. This is called "low alarm state". Default and minimum value is 1. Specified for each input.

4.14 INPUT HIGH NUMBER

The number of input state changes from low to high before the notification SMS is sent and/or call is performed. This is called "high alarm state". Default and minimum value is 1. Specified for each input.

4.15 INPUT LOW CALL NUMBER

How many call attempts to be done for every phone number specified in PHONES section for "low alarm state".

4.16 INPUT HIGH CALL NUMBER

How many call attempts to be done for every phone number specified in PHONES section for "high alarm state".

4.17 PHONES

The list of phone numbers for input alarms – phone numbers where to send SMS-messages and/or make a voice call. Phone numbers are defined separately for each input, for each ADC, for temperature sensors, for battery voltage, plus there are additional phone numbers common for all inputs ("inputs" field) and phone numbers for other alerts (like runtime errors, "other" field).

Sending SMS-messages and calling sequence is following:

- modem Java application at first sends SMS-message and calls to phone numbers specified in "1"- "10" field for corresponding input;
- then if it is an input alert, SMS-message is sent and call done to additional phone numbers specified in "inputs" field (only in case same phone number is not already listed in "1"- "10" field for specific input);

The calling sequence is following:

- no calling is done if corresponding INPUT LOW CALL NUMBER or INPUT HIGH CALL NUMBER is 0;
- at first a call is done to the first phone number listed in "1"- "10" field for corresponding input;
- if call is answered then no more calls are made; if not then modem calls to next phone number listed in "1"- "10" field for corresponding input and so on, until parameter INPUT LOW CALL NUMBER or INPUT HIGH CALL NUMBER is reached for all phone numbers;
- then if it is an input alert, call is done to additional phone numbers specified in "inputs" field (only in case same phone number is not already listed in "1"- "10" field for specific input);
- no calls are done to phone numbers listed in "other" field.

4.18 SENSORS

Sections starting with key sequence "SENSOR" specify the 1-Wire® temperature sensor information, one section per sensor. The sensor sections are named "SENSOR 1", "SENSOR 2", etc.

ROM code

Each 1-Wire® Digital Thermometer has a unique 64-bit serial code (factory written), which allows multiple sensors to function on the same 1-Wire® bus. Displayed in hexadecimal notation, MSB first. This is a mandatory parameter of each section; MultiIO application uses it to find 1-Wire® sensors on 1-Wire® net.

Name

User assigned descriptive sensor name. Default names are "SENSOR 1", "SENSOR 2", etc.

Found

The state of this checkbox indicates if sensor was found on 1-Wire® bus during last temperature scanning cycle.

Monitored

Specifies whether to monitor the temperature of this sensor. Default setting is YES (checkbox checked).

Temperature out of range

Indicates whether temperature of this sensor was out of defined normal range during last temperature scanning cycle.

Temperature

The current temperature of this sensor. The value is obtained during last temperature scanning cycle.

High temperature

The upper limit of normal temperature range. The high temperature alarm message will be sent when temperature of this sensor goes above this value. Default value is 125.

High temperature message

The text of SMS-message, what will be sent to phone number(s) specified in "PHONES/sensors" section when temperature of this sensor goes above upper limit of normal temperature. The name of sensor can be included in message by using "%NAME", the ROM code – by using %ROM and the current temperature of sensor – by using %TEMP. Default text of this message is "*%NAME" %ROM %TEMP deg C - too high.*"

Low temperature

The lower limit of normal temperature range. The low temperature alarm message will be sent when temperature of this sensor goes below this value. Default value is -55.

Low temperature message

The text of SMS-message, what will be sent to phone number(s) specified in "PHONES/sensors" section when temperature of this sensor goes below upper limit of normal temperature. The name of sensor can be included in message by using "%NAME", the ROM code – by using %ROM and the current temperature of sensor – by using %TEMP. Default text of this message is "*%NAME" %ROM %TEMP deg C - too low.*"

Normal temperature message

The text of SMS-message, what will be sent to phone number(s) specified in "PHONES/sensors" section, when temperature of this sensor returns back to normal range. The name of sensor can be included in message by using "%NAME", the ROM code – by using %ROM and the current temperature of sensor – by using %TEMP. Default text of this message is "*%NAME" %ROM %TEMP deg C –returned to normal.*"

The following notification SMS-message is sent to phone number(s) specified in MultiIO configuration "PHONES/sensors" section in case RS232 to 1-Wire® adapter was detected, but no 1-Wire® sensor is connected:

"Adapter found but could not find 1-wire temperature sensors"

After setting the parameters in TCCConfigurator is finished, you have to save them to file (under any name) and write configuration to the modem. In the modem Flash file system the name of configuration file always is MultiIO.ini.

If MultiIO.ini file does not present in the root directory of TC65T (Module) file system, it is created on modem application start-up with default settings.

5. Configuration and control by SMS

The modem Java application recognizes a set of SMS-messages (SMS commands) to tune up the modem Java application for specific needs. There is no limitation from which phone or GSM-modem a configuration SMS-message is sent: if security (access) code matches with current access code used on modem, the received SMS-message will be processed. In case of successful or unsuccessful message processing, response SMS-message will be sent back to the sender, containing relevant information pertaining to this message. The new value of modified parameter will be saved in MultiIO.ini file. The "#" character is used as a delimiter between configuration SMS-message fields. The "#" character at the start and end of SMS-message is optional. Most SMS commands are **case sensitive**. MultiIO.ini resides in flash memory of modem, therefore after modem switching off and on, all settings will be restored and it is not necessary to send the configuration SMS-messages again.

Here is the brief description of all SMS commands available for modem Java application configuration:

access#rst - modem reset. All settings saved. Case insensitive, except access code part. Modem Java application should respond with "OK, reset" SMS-message. Example: 0000#rst

Note – after modem reset, the modem Java application will start to work and respond to SMS commands (e.g. ping) 2-3 minutes after restart.

access#def – setting the modem Java application to default settings. Case insensitive, except access code part. Example: 0000#def

access#newaccess#newaccess - SMS command to set or change the security (access) code, where "access" is the old code and "newaccess" - the new code. Case sensitive. Example: 0000#abcd#abcd

access#svcno=XXXXXX - SMS command to set and change GSM Service Center number. Case sensitive. Example: 0000#svcno=+358405202000

access#phoneinX=XXXXXX - SMS command to add input state change notification phone number. "X" – input number [fields 1...10 in MultiIO configuration PHONES section]. Example: 0000#phonein1=+35812345678

access#phoneadcX=XXXXXX - SMS command to add analog input alert notification phone number. "X" – analog input number 1 or 2. Example: 0000#phoneadc1=+35812345678

access#phonesensors=XXXXXX - SMS command to add temperature sensor temperature alert notification phone number. Example: 0000#phonesensors=+35812345678

access#phonebattery=XXXXXX - SMS command to add modem battery supply voltage low/high limit alert notification phone number. Example: 0000#phonebattery=+35812345678

access#phoneinput=XXXXXX - SMS command to add input state change notification phone number. Common for all inputs (goes to MultiIO configuration PHONES section "inputs" field). Example: 0000#phoneinput=+35812345678

access#phoneother=XXXXXX - SMS command to add notification phone number for "other" alerts. Example: 0000#phoneother=+35812345678

access#phonedel...=XXXXXX - SMS command to remove notification phone number. Syntax is the same as for all adding phone number messages, simply replace "phone" with "phonedel". Examples:

```
0000#phonedelin1=+35812345678
0000#phonedeladc1=+35812345678
0000#phonedelsensors=+35812345678
0000#phonedelbattery=+35812345678
0000#phonedelininput=+35812345678
0000#phonedelother=+35812345678
```

access#ping – request of general script settings. Case insensitive, except access code part.
Example: 0000#ping

The response on this SMS-message is following:

```
PIN=SIM_PIN#ACCESS#SERVICE_CENTER_NR# iomask=IO_MASK#adcscan=
ADC1SCAN,ADC2SCAN#adcrep=ADC1REP,ADC2REP
```

access#pin=xxx – for setting the SIM PIN code. Example: 0000#pin=1234

access#iomask=xxx – for setting the “IO mask” parameter. Example: 0000#iomask=7

access#out=xxx – for setting the “Outputs” parameter. Example: 0000#out=264

access#scan1=xxx – for setting the ADC1 scanning interval parameter. Example: 0000#scan1=60

access#scan2=xxx – for setting of ADC2 scanning interval parameter. Example: 0000#scan2=60

access#inX#MsgInpLow#MsgInpHigh – for setting the input state change SMS-message texts for digital inputs, where X is the input number in range 1...10 (goes to corresponding fields in MultiIO configuration INPUT LOW MESSAGES and INPUT HIGH MESSAGES sections). If MsgInpLow or MsgInpHigh field is omitted, corresponding input state change notification message will not be sent.
Example: 0000#in1# Input1 Off# Input1 On

access#outX#MsgOutLow#MsgOutHigh – for setting SMS-messages for changing output state to Low or High. “X” is the output number in range 1...10 (goes to corresponding fields in MultiIO configuration OUTPUT LOW MESSAGES and OUTPUT HIGH MESSAGES sections). If MsgOutLow or MsgOutHigh field is omitted, corresponding output change will not be possible.

Example:

If for output 2 the following configuration SMS command was sent:

```
0000#out2##Start motor 2
```

Then to set output 2 to high state, the following SMS-message can be sent to modem:

```
1234#Start motor 2
```

It also a possibility to set output to high or low state for specified time interval, in seconds. To do that, use the same output state set command, followed by “=time”, where time – duration of output pulse in seconds. For example, to generate 30 seconds high level pulse on output 2, the following SMS-command can be sent to modem: 1234#Start motor 2=30

access#battery#MsgBatteryLow#MsgBatteryHigh – for setting “battery voltage out of limits” notification message. If MsgBatteryLow or MsgBatteryHigh field is omitted, corresponding “battery voltage out of limits” notification message will not be sent. Example: 0000#battery#Battery low#Battery high

access#adcX#MsgADCLow#MsgADCHigh – for setting “ADC out of limits” notification messages. “X” – ADC number, can be 1 or 2. If MsgADCLow or MsgADCHigh field is omitted, corresponding “ADC out of limits” notification message will not be sent. Example: 0000#adc2#ADC2 low# ADC2 high

access#delayX#HighStateDelay#LowStateDelay – for setting “High state stable” interval and “Low state stable” interval (goes to corresponding fields in MultiIO configuration INPUT HIGH DELAY and INPUT LOW DELAY sections). If HighStateDelay or LowStateDelay is omitted, the corresponding parameter will be set to 0. “X” – the input number [1...10]. Example: 0000#delay2#30

access#repin#RequestMsg#ReplyMsg – for setting the digital input request and reply messages. Both RequestMsg and ReplyMsg must present, otherwise digital input “read on request” will not be possible. Example: 0000#repin2#inputs?#inputs: %s (“%s” in response SMS-message will be replaced by “IN1=X,IN2=X, ...,IN10=X” where X = 0/1 – state of corresponding input)

access#reput#RequestMsg#ReplyMsg – for setting the output request and reply messages. Both RequestMsg and ReplyMsg must present, otherwise output “read on request” will not be possible. Example: 0000#reput2#outputs?#outputs: %s (“%s” in response SMS-message will be replaced by “OUT1=X,OUT2=X, ...,OUT10=X” where X = 0/1 – state of corresponding output)

access#repadcX=Y#RequestMsg#ReplyMsg – for setting the analog input request and reply messages and analog input auto report interval. “X” – ADC number, 1 or 2; “Y” – auto report interval in seconds. If Y>0, ReplyMsg will be automatically sent in specified update interval Y. Example (for requesting analog input report each 1 hour): 0000#repadc1=3600#ai1?#ai1: %d Mv (“%d” in response SMS-message will be replaced with analog input current value in millivolts)

access#limadcX=Lo,Hi – for setting the analog input value limits. “X” – ADC number, 1 or 2. If input value goes below specified Lo value, MsgADCLow message is sent once. If input value goes above specified Hi value, MsgADCHigh message is sent once. On returning back to normal range (value inside limits), the notification message (“ADCX returned to normal”) will be sent. Up to 2 digits after decimal point (e.g. 300.55) is supported. Example: 0000#limadc2=100,5000

access#numberinX#Y#Z – for setting the number of input X state changes (Y – number of “low alarm” state changes, Z – number of “high alarm” state changes) before the notification SMS-message is sent and/or call is performed. Will go to corresponding fields in MultiIO configuration INPUT LOW NUMBER and INPUT HIGH NUMBER sections. Example: 0000#numberin2#3#4 – for input 2 the SMS-message will be sent and/or call performed after 3 times “low alarm” state changes or 4 times “high alarm” state changes.

access#callinX#Y#Z

access#calladcX#Y#Z

access#callbattery#Y#Z

access#callsensorX#Y#Z

Above SMS-commands can be used for setting the number of call attempts to be done for every phone number specified in PHONES section. X – the number of input, ADC or sensor, Y – the number of “low alarm” state calls, Z – number of “high alarm” state calls. Will go to corresponding fields in MultiIO configuration INPUT LOW CALL NUMBER and INPUT HIGH CALL NUMBER sections. Example: 0000#callin2#0#1 – for input 2 no call will be done for “low alarm” state and 2 calls will be done for “high alarm” state.

The following SMS-commands are available for 1-Wire® Digital Thermometers online configuration:

access#sensor query – to request a list of names for all sensors detected.

Example: 0000#sensor query

access#sensor name#new=newname – to change the sensor name.

Example: 0000#SENSOR 2#new=sens2

access#sensor name – for reading the current temperature value of 1-Wire® sensor with name “sensor name”. The “read temperature” response SMS-message (currently the format of this response is unchangeable) will contain the sensor name and temperature value.

Example: 0000#SENS2

Example of response: Sensor SENS2 temperature is 22.5 deg C

access#sensorscan=scanrate – for setting the temperature scanning interval (in seconds).

Example: 0000#sensorscan=300

access#slew=value – for setting the pull-down slew rate. The allowed values are: 15, 2.2, 1.65, 1.37, 1.1, 0.83, 0.7, 0.55 (unit: V/μs).

Example: 0000#slew=2.2

access#w1lt=value – for setting the write-1 low time. The allowed values are from 8...15 (unit: μs).

Example: 0000#w1lt=8

access#dso=value – for setting the data sample offset and write-0 recovery time. The allowed values are from 3...10 (unit: μs).

Example: 0000#dso=8

access#spud=value – for setting the strong pullup duration time. The allowed values are: 16.4, 65.5, 131, 262, 524, 1048 (unit: ms).

Example: 0000#spud=524

access#sensor name#monitor=yes/no – to specify whether to monitor the temperature of specified sensor.

Example: 0000#sens3#monitor=no

access#sensor name#high=value – for setting the high alarm temperature level of specified sensor. The **value** – integer in range -55...125.

Example: 0000#sens2#high=35

access#sensor name#low=value – for setting the low alarm temperature level of specified sensor. The **value** – integer in range -55...125.

Example: 0000#sens2#low=5

access#sensor name#high#high temperature message – for setting the high temperature alarm SMS-message.

Example: 0000#sens2#high#"%NAME" - high temp. alarm %TEMP deg C

access#sensor name#low#low temperature message – for setting the low temperature alarm SMS-message.

Example: 0000#sens2#low#"%NAME" - low temp. alarm %TEMP deg C

access#sensor name#normal#return to normal temperature message – for setting the “return to normal temperature” SMS-message.

Example: 0000#sens2#normal#"%NAME" – temp. back to normal

Note!

Configuration SMS-messages, output change SMS-messages and "ping" SMS-messages can be sent from **any** phone number (not only from phone number configured as **PhoneNo**) - the SMS-message sent to TC65T should contain the proper current **access code**.

6. Related documentation

1. TCConfigurator user instruction.
2. TC65 Terminal. Siemens Cellular Engine. Hardware Interface Description.
3. Application note 148: "Guidelines for Reliable 1-Wire Networks".
4. DS2480B Serial 1-Wire Line Driver with Load Sensor datasheet.
5. Application note 192: "Using the DS480B Serial 1-Wire Line Driver".
6. DS18S20 High Precision 1-Wire Digital Thermometer datasheet.

Can be sent on request.

Technical support: e-mail to support@klinkmann.com.