Application note

11.5.2009

Firmware V3.0

MTR970
Data format

Nokeval

INTRODUCTION

MTR970 is an entry-level radio receiver with limited processing capability. It is intended to be used with a PC software or other external intelligence that will process the data. When this is not possible, a receiver with more processing capability RTR970-PRO is recommended.

This document supplements the actual MTR970 user manual and gives information for those who intend to write software to read the MTR970.

MTR970 supports only Nokeval SCL protocol, and within it, two languages can be used: SCL commands, and Nopsa commands encapsulated in SCL. The protocol is described in a separate document "SCL user manual".

The RTR970 (without -PRO) is a rail-mounted version of MTR970, with equivalent firmware capabilities.

Table of contents

Manufacturer

Nokeval Oy
Yrittäjäkatu 12
FIN-37100 Nokia
Finland

Tel +358 3 3424800
Fax +358 3 3422066
WWW: www.nokeval.com

LANGUAGES

MTR970 can be read with two command languages.

Native SCL commands

Nokeval SCL is usually used with human-readable commands, parameters, and responses. E.g. command "TYPE ?" will give a response "MTR970 V3.0". This is convenient for debugging, but inefficient for machine processing, since several data-to-string and string-to-data conversions are needed.

Nopsa

Nokeval has specified a language called Nopsa, which consists of bytes and bits and fixed field lengths when possible. In that sense it resembles much Modbus. However Nopsa is a language only, so it needs a protocol to make it transferable over a serial bus. When transfered over SCL protocol, the Nopsa command and response packets are converted to a hexadecimal string. Additionally, a 'N' and a space is preceded in the command packet to distinguish Nopsa commands from other commands like "TYPE" and "MEA".

Nopsa is specified in detail in a separate document.

RTR970-PRO ring buffers can be read with identical Nopsa commands with MTR970, so Nopsa is preferred over native SCL commands in most cases.

SCL COMMANDS

Command set

TYPE ?

Returns the model name and firmware version: "MTR970 V3.0"

SN ?

Returns the circuit board or device serial number, for example ”P123456” for circuit board or "A123456" for device.

MEA CH 1 ?

Returns the last received value from the device that is defined as ”channel 1”. Channels 1…4 can be read this way. This is available for the following transmitters only:

- MTR262 with other than thermocouple sensors
- MTR265 with other than thermocouple sensors
- MTR165

There is no timeout involved. If the transmitter is not heard, the same reading will be returned forever. It is not recommended to use this poor implementation of MEA command. If this kind of functionality is needed, it is highly recommended to use the RTR970-PRO receiver.

DBG 1 ?

Returns the oldest unread data packet from the buffer. This is the normal way of reading the buffer. If there is no unread packets, "#" is returned. Otherwise a list of decimal numbers is returned, see below.

DBR 1 xx ?

Returns a data packet from the ring buffer location xx. This is an alternative way to read the buffer, when the index is desired to keep at the reader. The response is similar to DBG response. "#" is returned when the given index is equivalent to the internal write index of MTR970.

xx is the location 0…(size-1) in decimal.

DBX

Clears the ring buffer and discards stored packets.

DBS 1 ?

Returns the size of the ring buffer in decimal. In MTR970 firmware version 3.0, the size is 96 locations. This is a meaningful information only when using the DBR command.

Reading the ring buffer with DBG

Reading the buffer with DBG command is easy and straightforward. DBG command is sent repeatedly to MTR970 and the response is examined. How often the command should be sent, depends on the number and transmit interval of the nearby transmitters, so that the packets are read out of MTR970 before its buffer starts overwriting the oldest packets. Once a second is a good interval, especially if the buffer is read until empty (DBG repeated until a "#" is returned).

When the system is started, it is up to the programmer whether he wants to erase all the older packets from the buffer by sending DBX command or not.

DBG and DBR response format

The response consists of space-separated fields, that consist of integers in decimal format:

<type> <bytesandbatt> <rsl> <id> <data0> <data1> …

Field	Range	Description
<type>	0…255	Transmitter type. See the table below.
<bytesandbatt>	0…255	Bits 7:5: Number of payload data bytes after <id>, 0…7 Bits 4:0: Transmitter battery voltage in decivolts, 0…31
<rsl>	0…255	Bit 7: CRC error. The receiver does not give this kind of packets unless the CRC checking is disabled in the configuration menu. Bits 6:0: Received signal level. Subtract 127 from this to get dBm. The noise floor is about -100dBm.
<id>	1…65535	Transmitter ID number. ID 0 is reserved for factory use.
<data0>…	0…255	Payload data bytes. There may be 0 to 7 bytes depending on the transmitter type. See chapter Payload data.

Example: MTR970 returns

0 91 33 2378 200 9

First number tells the transmitter is MTR260. The next says 2 payload bytes and 2.7 volts. 2378 is the ID, corresponds to the sticker on the transmitter. 200 and 9 form a 16-bit integer least significant byte first: 200+256*9=2504 decikelvins = -22.8°C.

Transmitter types

Type	Name
0	MTR260 (including 260B, 260C, 260D…) and FT10-RT433-IS
2	MTR262 (including 262B) and FTR262
4	MTR264
5	MTR265
6	MTR165
7	FTR860
8	CSR264 Start
9	CSR264 Limit
10	CSR264 Arrive
11	CSR260
12	KMR260
15	Utility packet, may be transmitted by any transmitter

NOPSA LANGUAGE

Command set

MTR970 supports the following Nopsa commands:

- 1/0 Get type
- 1/1 Get version
- 1/2 Get serial number
- 1/3 Get description
- 1/4 Get command set
- 1/5 Get serial buffer size
- 1/16 Reset
- 1/32 Meku
- 4/0 Get buffer info
- 4/1 Go to oldest
- 4/2 Go to newest
- 4/3 Read buffer by index
- 4/4 Read buffer and autoincrement
- 4/5 Read again

Reading the ring buffer

Like with native SCL commands, Nopsa provides two ways to read the ring buffer. The recommended way is to let the MTR970 to handle all the buffer indices, and only that way is presented here.

Start-up

When the system is started, the read index inside MTR970 points to the oldest available unread packet. If the buffer is then read, those old packets are returned first.

If the old packets are not desired, Nopsa command 4/2 can be sent. It moves the index to the newest packet, so older packets are not returned.

Reading

Send command 4/4 (<id>N 0404<etx><bcc>). It does not have any parameters.

The response is a string of hexadecimal characters. Convert them to a binary array.

Examine the result. Its bytes are as follows:

Byte(s)	Description
0	Nopsa status byte. The lowest three bits should be 0.
1:2	Buffer index. May be neglected but is useful when recovering from a serial error as described below.
3	Rollover count. Tells how many times the index has rolled over to zero. This count will roll over after 255. May be neglected.
4:7	Timestamp. MTR970 does not have clock, so this is always 0x00000000.
8:9	Transmitter ID.
10	Data type 0x20 = STRUCT
11	Struct type 0x00 = raw radio packet (RTR970-PRO may return type 0x01 = processed radio packet too, see Nopsa struct documentation).
12	Transmitter type, see page 5.

13	Signal strength. Subscribe 127 to get dBm.
14	Bytesandbatt. Bits 7:5: Number of payload data bytes, 0…7. Bits 4:0: Transmitter battery voltage in decivolts, 0…31. MTR260C uses a 3.6V battery, so this will be leaning to 3.1V most of time.
15…	Payload data, see chapter Payload data.

If there is bytes after the payload, they can be neglected.

Serial errors

If the command or response gets corrupted due to a disturbance on the serial bus, we can not know whether the MTR970 has got the command or not. This is important because we do not know if it has incremented the buffer index and skipped one packet or not.

The correct way to proceed after a serial error is to re-read a packet with command 4/5. If it returns the same packet as the previous successful command-response pair did, we can discard this repeated response. If this is a new packet, then the MTR970 did increment its index and this is a new packet.

To see if we got the same packet with command 4/5, we can examine the buffer index at bytes 1:2.

PAYLOAD DATA

MTR260 and FT10-RT433-IS

Internal temperature sensor.

Bytes	Description
0:1	Temperature reading in decikelvins. Treat these two bytes as a 16-bit integer, least significant byte first. Divide by 10 and subtract 273.2 to get Celsius.

MTR262 and MTR265

Universal input: thermocouples, RTD's, mV, V, mA.

Bytes	Description
0:3	Reading, IEEE-754 floating point number, least significant byte first.
4:5	Cold junction temperature in decikelvins. Treat these two bytes as a 16-bit integer, most significant byte first (sorry). Divide by 10 and subtract 273.2 to get Celsius.

With other sensors than thermocouples, the reading is the reading as measured and scaled by the transmitter. RTD's give °C, and mV, V, and mA gives the unit suggested by the range name.

With thermocouples, the first four bytes give the sensor signal in millivolts. The last bytes indicate the temperature of the transmitter connector block.

It should be checked that the reading is within a reasonable range, otherwise the sensor or the wires are probably broken.

MTR264

Four thermocouple inputs. MTR264 sends four independent packets, each corresponding to one thermocouple input. The transmissions use radio id's ID, ID+1, ID+2, and ID+3, where ID is the radio ID marked in the label.

Bytes	Description
0:3	Sensor voltage in mV. IEEE-754 floating point number, least significant byte first.
4:5	Cold junction temperature in decikelvins. Treat these two bytes as a 16-bit integer, most significant byte first. Divide by 10 and subtract 273.2 to get Celsius.

MTR165

Process signal inputs: mV, V, or mA. The signals can be scaled to an engineering value at the transmitter, and this engineering value will be transmitted.

Bytes	Description
0:3	Reading in mV, V, or mA. IEEE-754 floating point number, least significant byte first.

FTR860

Two thermocouple/RTD/mA inputs, and two digital inputs for controlling the transmission. FTR860 will send the two input channels in separate radio packets. Channel 1 is sent in a radio packet having the radio ID marked in the label; channel 2 is sent using ID+1.

Bytes	Description
0:3	Reading in mV, V, or mA. IEEE-754 floating point number, least significant byte first.

KMR260

Hand-held thermometer.

Bytes	Description
0:1	Temperature reading in decikelvins. Treat these two bytes as a 16-bit integer, least significant byte first. Divide by 10 and subtract 273.2 to get Celsius.
2	User (index to a table)
3:4	Location
5:6	Food

Utility data

Some transmitters may intermittently send some additional information using transmitter type 15. After that, comes a byte indicating the type of utility data, and after that, some data.

Utility type 0: Calibration date

Tells when this transmitter has been calibrated. Can be used to warn the user that the transmitter should be recalibrated.

Bytes including type byte 0:

Bytes	Description
0	Value 0: calibration date
1:2	Calibration date. 16 bit word, least significant byte first. Value 0 corresponds to 1.1.2000, and every day increments by one.