T-TEC 4R1P : Temperature Sensor with RS-232

John Steele Scott <john@t-tec.com.au>

31 October 2007

Contents

1 Introduction

This document describes the communications protocol used by the T-TEC 4R1P temperature sensor. It is intended to be read by the engineer who wants to integrate this sensor into a larger system.

The 4R1P device is a digital temperature sensor which measures temperatures using a PT100 probe. The sensor has an RJ45 socket which is used to interface with a host device. Signalling on this interface uses RS-232 levels.

This sensor operates from a 3.6 V Lithium battery, and measures temperature in the range −200 °C–120 °C.

Table 1 below lists which wire is used for each RS-232 signal. Note that the signal names are referenced to the data terminal equipment (DTE), e.g. the RxD signal is data transmitted from the sensor to the receiving equipment. Figure 1 shows the RJ45 cable pin-out.

Signal (referenced to DTE)	RJ45 Pin	TIA-568B Colour	TIA-568A Colour	DB9 Pin
RxD	5	white/blue	white/blue	2
TxD	4	blue	blue	3
GND	6	green	orange	5

Table 1: Wire/Signal Assignments.

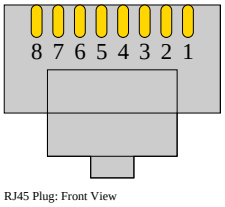

Figure 1: RJ45 cable pin-out.

1.1 Revision History

31 October 2007 Initial release for production version.

2 Message Structure and Request Syntax

Each message has the following form:

`SOH | COMMAND | MSGID | LENGTH | DATA | EOT`

The message components are:

SOH ASCII start-of-header character, has the value 1. (1 byte)

EOT ASCII end-of-transmission character, has the value 4. (1 byte)

COMMAND A command code which specifies how the data in the message should be interpreted. (1 byte)

MSGID Message sequence number, in the range 0–31, increments for each packet. (1 byte)

LENGTH The data length. (1 byte)

DATA Data. (`LENGTH` bytes)

To request a particular message, send the two-byte sequence:

`COMMAND | ?`

where `COMMAND` is the command code corresponding to the message you want, and `?` is the ASCII question mark character (decimal 63, or `0x3f`).

2.1 Information Message

This message has `COMMAND` set to the ASCII character `i`. The data payload is as follows:

`FIRMWARE | SERIAL-HIGH | SERIAL-LOW | TYPE | NUM_PROBES`

where `FIRMWARE` is a single byte firmware version, and `SERIAL` is a two byte device serial number, written out high-byte-first. `TYPE` is an ASCII character which will be `P` for this device. `NUM_PROBES` is a single byte number indicating the number of temperature probes supported by the device. The 4R1P has only one probe, but a future device may have two.

2.2 Temperature Message

This message has `COMMAND` set to the ASCII character `t`. The two byte data payload is as follows:

```
T-HIGH | T-LOW
```

The relationship between `T` and the temperature in degrees is

$$\theta(^{\circ}\mathrm{C}) = (\mathtt{T} - 2733)/10.0$$

A value of 23.6 °C, for example, translates into 2969 decimal, or `0xb99`.

If `T = 0xffff`, the temperature is outside the operating temperature range (too high). If `T = 1`, the temperature is outside the operating temperature range (too low).

If `T = 0`, the temperature probe is damaged or not connected. Please note that it is not possible to detect this condition with 100% accuracy, and the decision has been made to avoid false positives. That is, `T = 0` means a sensor error has occurred, but not all sensor errors will produce `T = 0`.

A future model of this device may support two temperature probes. In this case, it is likely that `COMMAND = t` will be used for the first probe, and `COMMAND = T` will be used for the second probe.

2.3 Battery Message

This message has `COMMAND` set to the ASCII character `b`. The two byte data payload is as follows:

```
B-HIGH | B-LOW
```

The quantity `B` will be the battery voltage in hundredths of a volt. A value of 3.31 V thus translates into 331 decimal or `0x14b`.

3 Example PC Software

An example program is provided as a quick way to see how the sensor works. The program, `rs232-ondemand-sensor-demo.exe`, will run on Windows PCs or compatible systems. It will listen for data from the COM1 serial port. When data arrives, the raw data received will be printed in hexadecimal, followed by the decoded message. To use another serial port, specify that port on the command line, e.g. run `rs232-ondemand-sensor-demo.exe COM6` to listen on COM6.

The source code of `rs232-ondemand-sensor-demo.exe` is supplied in the directory `src` which accompanies the program.