Kelvin NTC10KA

Smart VSCP NTC10KA temperature controller

Reversion 1.1 - 2012-05-02

Abstract

Kelvin NTC10KA is a temperature module that connects to a CAN4VSCP bus and can sense one internal and up to five external NTC temperature sensors. The module can be attached to a standard DIN Rail or be mounted directly on a wall (ordered separately). The module fully adopts to the CAN4VSCP specification and can be powered directly over the bus with a 9 - 28VDC power source. It has a rich register set for configuration and many information events defined. It also have a decision matrix for easy dynamic event handling.

Grodans Paradis AB
Brattbergavägen 17
820 50 LOS
SWEDEN
web: http://www.auto.grodansparadis.com
email: info@grodansparadis.com
phone: +46 8 40011835

All boards produced by *Grodans Paradis AB* are ROHS compliant.

Contents

List of Figures

List of Tables

Chapter 1

Kelvin NTC10KA - Temperature measurement module

1.1 Most current information

You can find the most current information about the Kelvin NTC10KA module at http://www.auto.grodansparadis.com/kelvin_ntc/kelvin_ntc.html. On the site you can also find links to the latest firmware and Module Description File (MDF) for the device as well as schematics and recipes for its use. This information is of course pointed to from the MDF file which you can locate from the module itself reading it's standard registers.

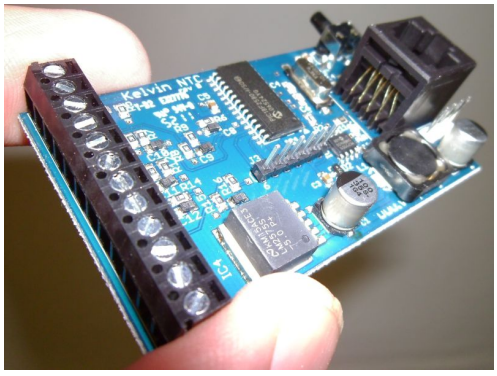


Figure 1.1: Kelvin NTC10KA

1.2 The raw facts

Parameter	Value
Supply voltage	9-28VDC
PCB Size	42 mm x 72mm
Power requirements	0.1W
Communication	VSCP4CAN (CAN), 125kbps
Measurement points	6 (one on board, five external)

Table 1.1: The raw facts

1.3 Hardware

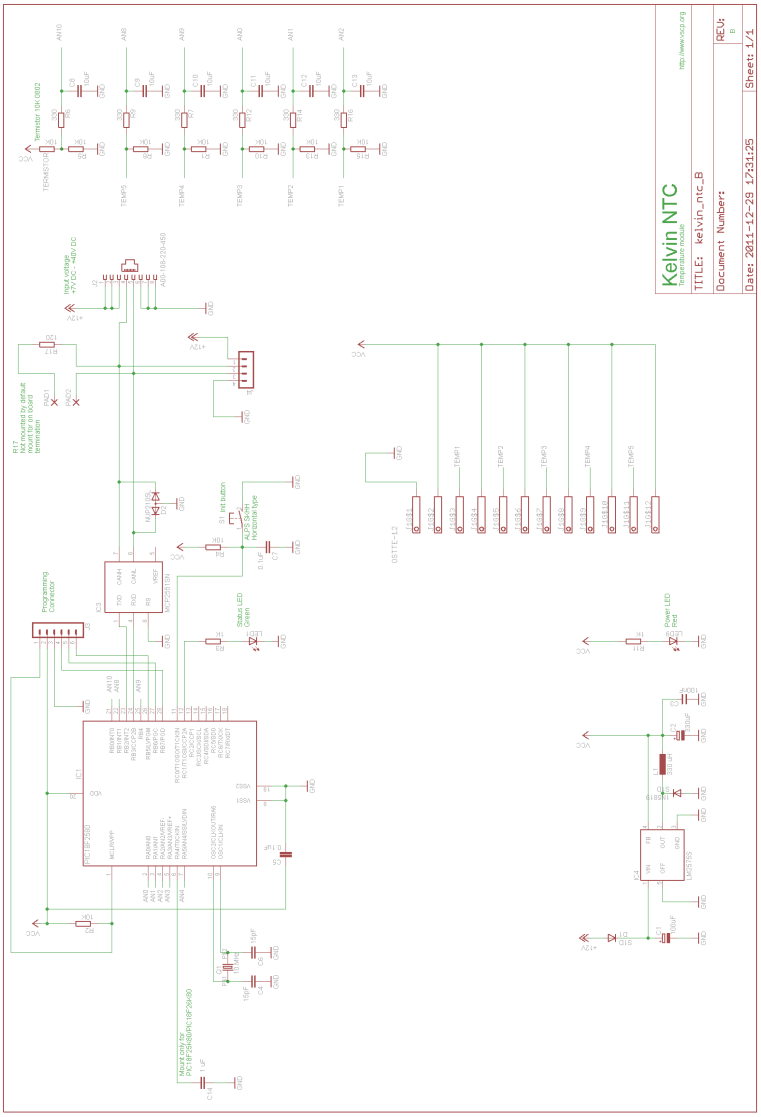


Figure 1.2: Schema for the Kelvin NTC10KA module

Some key positions on the module is outlined in the figure below

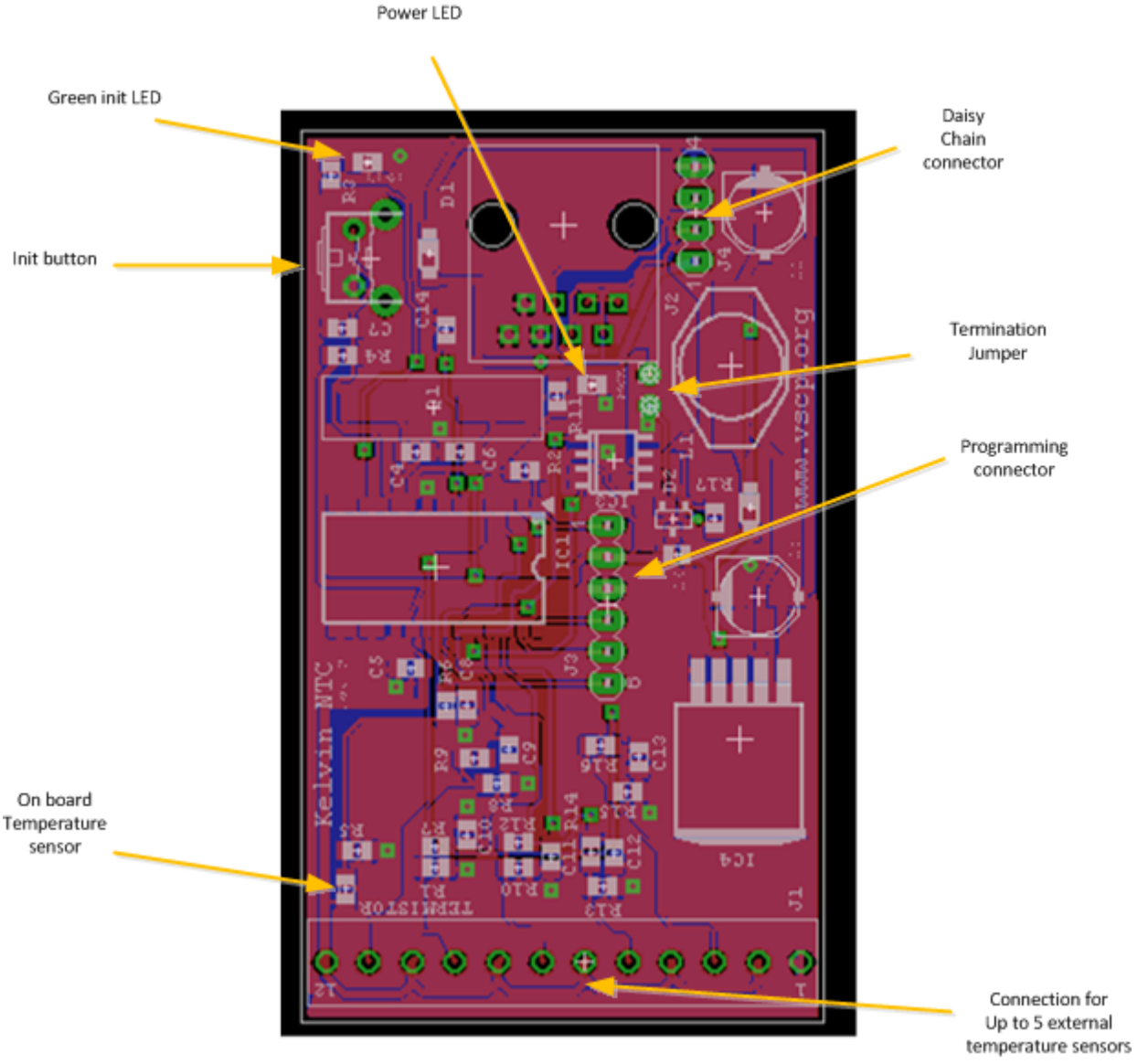


Figure 1.3: Road map to module

1.4 Cable and connectors

The unit is powered over the CAN4VSCP bus. The CAN4VSCP normally uses CAT5 or better twisted pair cable. You can use other cables if you which. The important thing is that the CANH and CANL signals uses a twisted cable. For connectors you can use RJ10, RJ11, RJ12 or the most common RJ45 connectors. There are different versions

1.4.0.1 RJ-XX pin-out

Pin	Use	RJ-11	RJ-12	RJ-45	Patch Cable wire color T568B
1	+9-28V DC			RJ-45	Orange/White
2 1	+9-28V DC		RJ-12	RJ-45	Orange
3 2 1	+9-28V DC	RJ-11	RJ-12	RJ-45	Green/White
4 3 2	CANH	RJ-11	RJ-12	RJ-45	Blue
5 4 3	CANL	RJ-11	RJ-12	RJ-45	Blue/White
6 5 4	GND	RJ-11	RJ-12	RJ-45	Green
7 6	GND		RJ-12	RJ-45	Brown/White
8	GND			RJ-45	Brown

Table 1.2: RJ-XX pin-out

RJ-11/12/45 pin-out

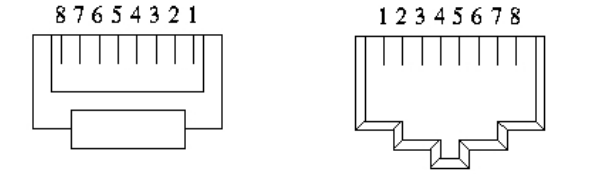


Figure 1.4: RJ-45 pin out

Always use a pair of wires for CANH/CANL fort best noise immunity. If the EIA/TIA 56B standard is used this condition will be satisfied. This is good as most Ethernet networks already is wired this way.

1.4.1 Cable length

CAN4VSCP always communicate with 125kbps. This means that if you use a good quality cable it can be up to a maximum of 500 meters using AWG24 or similar (CAT5) . Actual length depend on the environment and other parameters. Drops with a maximum length of 24 meters can be taken from this cable and the sum of all drops must not exceed a total of 120 meters.

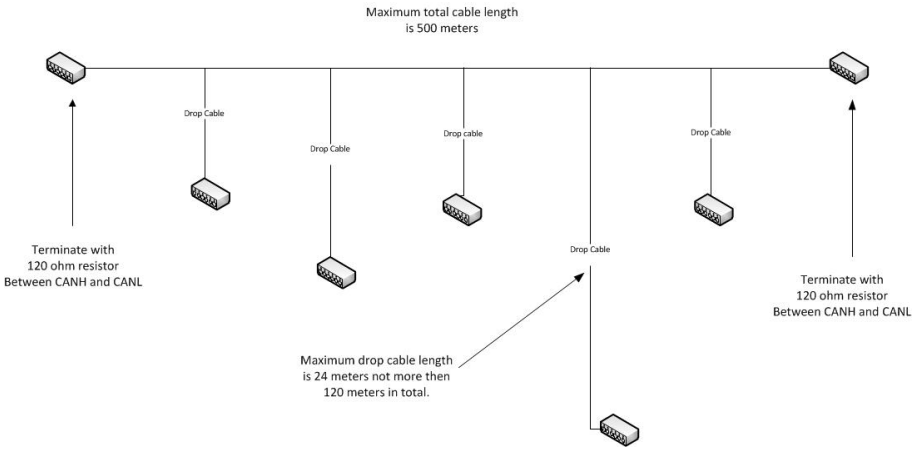


Figure 1.5: CAN4VSCP bus with drops and terminations

1.4.2 Termination

The CAN4VSCP bus, as all CAN based networks, should be terminated with a 120 ohms resistor between CANH and CANL at both ends of the cable.

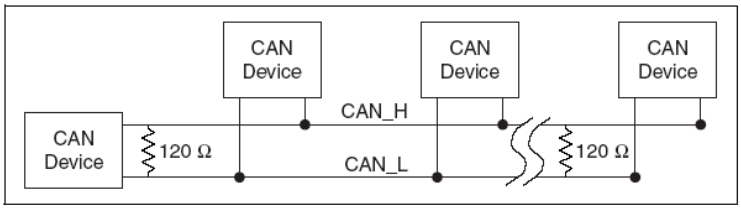


Figure 1.6: Termination

If you use CAT5 this termination should be placed between the blue - blue/white cables at both end of your bus.

On the board there is a jumper for an on-board terminator. See figure above.

1.4.2.1 Why are terminators required?

Terminators are needed to cancel signal echos in the cable. In short you get less noice in the cable if you use them. It is recommended to use them even if at 125 kbit it is possible that your bus will work anyway.

1.4.3 Daisy chain connector

The daisy chain connector is a pin-header that can be used as an easy way to daisy chain several modules in a cabinet or similar. You just connect the modules together with a simple cable. The pin-out is:

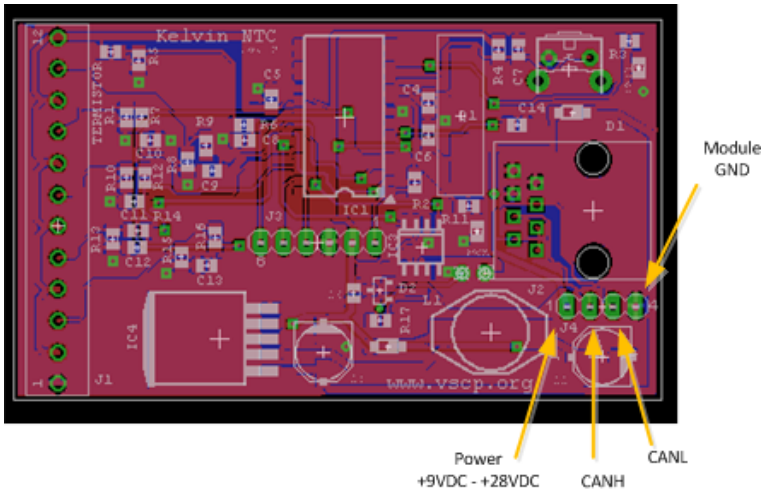


Figure 1.7: Daisy chain connector

Use a twisted cable for CANH/CANL and don't have a cable between mocules that is less then 30 centimeters.

1.4.4 Power the module

You normally power the module through the RJ45 connector over the CAN4VSCP bus. See 1.4.0.1 for a description of which pins to use for power and ground. The voltage range is +9VDC - 28VDC. The current need depend on how many modules you want to power.

An alternative way to power the module is through the daisy chain connector described above. Just connect +9V - 28V to it's pin 1 and ground to pin 4. Needless to say you can't have power supplied by the CAN4VSCP bus at the same time.

1.4.5 Best practice

Even if it is possible to have cable lengths up to 500 meters it is better to stay at shorter distances to have some margins. Here are some guidelines for a reliable set up

- Total cable length of 300m, stub cables count double their length in that total.
- Nodes count as 6m in that total.
- Never less than 30cm of cable between nodes, nor between a stub connection and a node.
- No more than 50 nodes connected.

Following this best practice you will have a setup that will work reliable even in harsh environments.

1.5 Installing the module

Connect the module to the CAN4VSCP bus. The red led on the module should light up indicating that the device is powered. If this is the first time you start up the module the green lamp next to the initializing button will start to blink. This means that the module is trying to negotiate a nickname address with the rest of the modules on the bus. When it found a free nickname the green led will light steady. If the green led does not start to blink press the initialization button until it does. Now your module is ready to use.

1.5.1 Termination block pin-out

The individual positions for the twelve position termination block is numbered from the left (looking into it) as in the figure below.

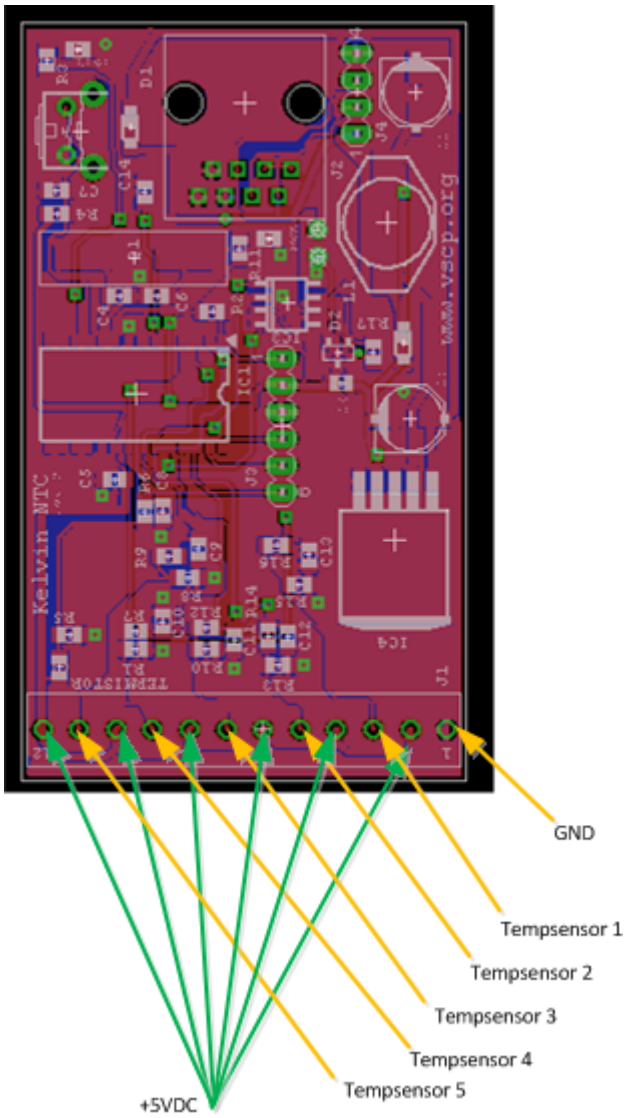


Figure 1.8: Pin-out for termination block

Position	Description
1	Ground
2	+5VDC
3	Temperature sensor 1
4	+5VDC
5	Temperature sensor 2
6	+5VDC
7	Temperature sensor 3
8	+5VDC
9	Temperature sensor 4
10	+5VDC
11	Temperature sensor 5
12	+5VDC

Table 1.3: Pin-out for termination block

1.5.2 Connecting external sensors

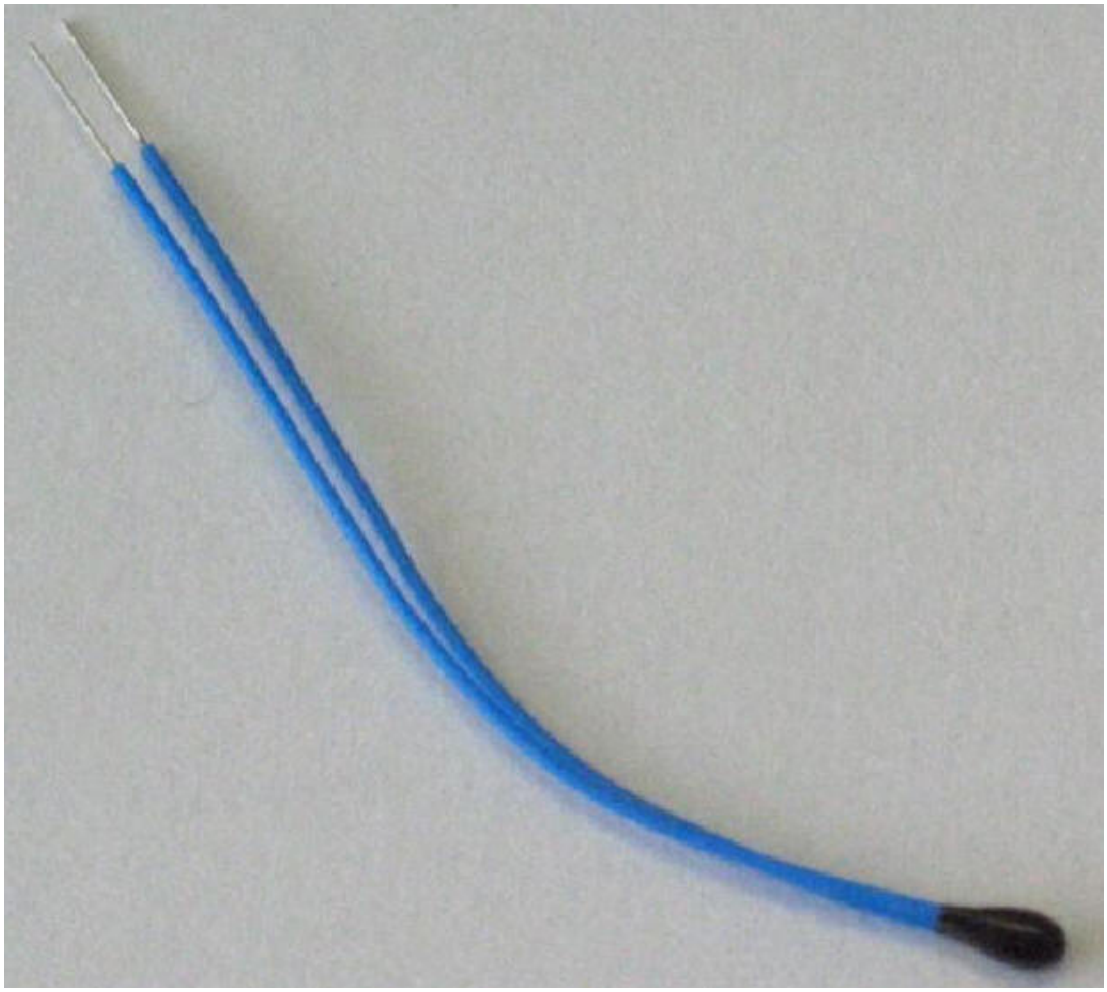


Figure 1.9: Thermistor

Five external NTC sensor can be connected to the Kelvin NTC10KA module. It is recommended to use NTC thermistors with 10K @ 25 řC types for best result. There are sensors available for all kinds of temperature measurement needs. You need the B value from the sensors datasheet to be able to use it with the Kelvin NTC10KA module.

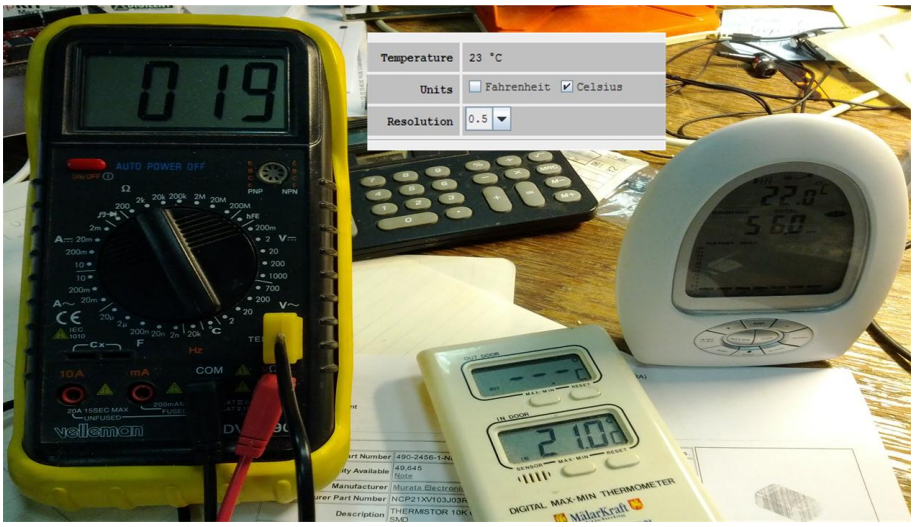
The main advantage for a NTC sensor is that it is low cost. You can get it from anywhere and you can get it in any type and dimension. NTC resistor is also not very sensitive to cable length. Using a CAT-5e cable (4 pairs with AWG 24) we will experience something like 9.38 ohms/100m which will not effect a

10K NTC thermistor that much.

Connect the thermistor to the Kelvin NTC10KA by connecting it to +5VDC and to one of the temperature inputs. A thermistor is not subject to polarity considerations so you can connect it in any way you want. You need to setup the B value for the sensor to get correct temperature measurements.

1.6 Measure Temperature

If you want to measure a temperature, for a rrom for example you will soon realize that temperature measurements ha smore to it then just hook up a sensor and measure the temperature. Look at this picture



Each of the sensors shows a different temperature even when the sensor are on the same desk. If you walk around in the room things will be even more complex. Three sensors of the same type but on different height will show different results.

The readings from the sensors of the Kelvin NTC module is given with two decimals. Does that mean we have 0.01řC precision? Unfortuantely no! The thermistor has error and the analog reading circuitry has error.

We can approximate the expected error by first taking into account the thermistor resistance error. The thermistor is correct to 1%, which means that at 25řC it can read 10,100 to 9900 ohms. At around 25řC a difference of 450 ohms represents 1řC so 1% error means about +-0.25řC (you may be able to calibrate this away by determining the resistance of the thermistor in a 0řC ice bath and removing any offset). You can also spring for a 0.1% thermistor which will reduce the possible resistance error down to +-0.03řC

Then there is the error of the ADC, for every bit that it is wrong the resistance (around 25řC) can be off by about 50 ohms. This isn't too bad, and is a smaller error than the thermistor error itself +-(0.1řC) but there is no way to

calibrate it 'away' - a higher precision ADC (12-16 bits instead of 10) will give you more precise readings

In general, we think thermistors are higher precision than thermocouples, or most low cost digital sensors, but you will not get better than +-0.1řC accuracy on an Arduino with a 1% thermistor and we would suggest assuming no better than +-0.5řC.

You will in addition to this notice that the on-oard sensor will show a temperature that is higher then what you should expect, The same is if you connect a sensor directly to the board. This is due to the fact that the electronics on the board generate heat. This heat flows very well through the copper of the board giving faulty readings. The solution is to take away a degree or two from the reading.

1.7 Updating firmware

There is two ways you can use to update the firmware of the module. You can program the device using the programming socket on board or you can use VSCP Works to remotely program the device.

1.7.1 Update firmware using the ICP connector

The firmware of a circuit equipped with a Microchip microprocessor usually can be programmed in circuit. That is when it is mounted on a printed circuit board. This is also true for the Paris relay module which have the programming connector on-board (J3). If you have a programmer for Microchip processors (Real ICE, ICD-2, ICD-3, PICKIT-2, PICKIT3 or other) you can program your own firmware or the latest official firmware into the module using MPLAB or similar tools. You can always find a link to the latest firmware on the Kelvin NTC10KA module home page (http://www.auto.grodansparadis.comkelvin_ntc/kelvin_ntc.html).

1.7.2 Update firmware with VSCP Works

When a module is installed in a remote location or if you don't have a Microchip programmer you can program the module using the built in boot-loader. This can be done with VSCP Works a program that can be run on the Windows or the Linux platform and can perform and can perform different maintenance, configuration and status checks of VSCP modules. If you have not installed the VSCP & Friends package it is time to do so now. You can always find the latest version on the VSCP projects download page (http://vscp.org/downloads.php).

The boot loader process using VSCP Works is described in section 16.4 of the VSCP specification. The Paris relay module uses the *PIC1* boot loader.

1.8 Configure the module?

You configure a VSCP module by writing content into the modules registers. You can do this manually or with the wizard available in VSCP works. Using the wizard is absolutely the easiest way to use.

1.8.1 Zone/sub-zone

You should always plan your overall structure. The zone and the sub-zone registers found in the first two register positions can help you here. Think of a zone as a house, floor plan or similar and sub-zone as a room or a location. Note that this is not an address. It's a way to group functionality together.

1.8.2 Enter sensor data.

The Kelvin NTC10KA module uses the B constant to calculate the temperature for the sensor you connect to it. You enter this value in registers 38-39 (see 1.9.6). You find this value in the datasheet for you sensor.

The formula used to calculate the temperature from a thermistors resistance is give as

$\frac{1}{T}$= a + b ln(R) + c $\ln^3$(R)

where

T = Degrees kelvin.
R = Resistance of thermistor.
a,b.c = Curve-fitting constants.

This the Steinhart Hart equation. We van get a simple formula by setting

a = $(\frac{1}{T})$ $(\frac{1}{B})$ ln(R_0)
b = $\frac{1}{B}$
c = 0

and get

T = $\frac{B}{ln(\frac{R}{r\infty})}$

where

T = Degrees kelvin.
r_∞ = $R_0 e^{-\frac{B}{T_0}}$
B = curve fitting parameter

R_0 is thermistor resistance at T_0. This is usually taken as the resistance at 25 C. Note that the Kelvin temperature should be used in the formula.

The later formula is used for the sensors on the Kelvin NTC10KA board. When you enter the B value you should use a 25 - x value. Depending on the range you need in your setup you choose x. Typically 100, 85 or 50.

1.8.3 Internal temperature sensor

1.8.3.1 The thermistors

Many people unfairly regard thermistors as inaccurate sensors. This may have been true in the past, when thermistors had 5% tolerances at best. For extreme accuracy the RTD is still the best choice, but modern thermistors are not far behind. Thermistors with 0.1 řC accuracy are now widely available and at very reasonable costs. They have a fast response time and a greater output per řC than RTDs.

As with RTDs, thermistors also exploit the fact that a material's resistance changes with temperature. However, the majority of thermistors employ a metallic oxide and have a negative temperature coefficient (NTC). Thermistors provide relatively high accuracy (0.1 to 1.5 řC) but only operate over a limited temperature range: -100 to 300 řC. Furthermore, no single thermistor will cover this range and a lack of standards means it is often necessary to buy the sensor and measuring equipment together. The thermistor's response is non-linear and, as with RTDs, we must avoid providing too large an excitation current through the thermistor because of self-heating.

Connection to instruments is a simple 2-wire configuration, as — unlike RTDs — we do not need to compensate for lead resistances: this is small compared to the thermistor's resistance (typically between 1 and 100K).

Thermistors, because of their high sensitivity, are ideal for detecting small changes in temperature - especially when it is the change and not the absolute value that is important.

1.8.3.2 Data for the internal sensor

The internal temperature sensor used on the Kelvin NTC10KA is the TDKB57421V2103J062. Data for this sensor is as follows

Description	Value
Time constant	10 s
$B_{25/100}$- value	4000K
Temperature range	-55...+125 °C
Resistance tolerance	±5 %
Resistance @ 25 °C	10 Kohm
Max effect	210 @ 25 °C mW
$B_{25/50}$- value	3940
$B_{25/85}$- value	3980
$B_{25/100}$- value	4000K ±3%
Max error ($B_{25/100}$)	2.8 (%/C)

The resistance table looks like this

T (°C)	R_T/R_{25}	(% K)
-55.0	96.158	7.4
-50.0	66.892	7.1
-45.0	47.127	6.9
-40.0	33.606	6.6
-35.0	24.243	6.4
-30.0	17.681	6.2
-25.0	13.032	6.0
-20.0	9.702	5.8
-15.0	7.2923	5.6
-10.0	5.5314	5.4
-5.0	4.2325	5.3
0.0	3.2657	5.1
5.0	2.54	4.9
10.0	1.9907	4.8
15.0	1.5716	4.7
20.0	1.2494	4.5
25.0	1.0000	4.4
30.0	0.80552	4.3
35.0	0.65288	4.1
40.0	0.53229	4.0
45.0	0.43645	3.9
50.0	0.35981	3.8
55.0	0.29819	3.7
60.0	0.24837	3.6
65.0	0.20787	3.5
70.0	0.17479	3.4
75.0	0.14763	3.3
80.0	0.12523	3.2
85.0	0.10667	3.2
90.0	0.091227	3.1
95.0	0.078319	3.0
100.0	0.067488	2.9
105.0	0.058363	2.9
110.0	0.050647	2.8
115.0	0.044098	2.7
120.0	0.03852	2.7
125.0	0.033752	2.6

1.8.3.3 External sensors

When you choose an external sensor to connect to the Kelvin NTC10KA you get the best result if you choose one with a resistance of 10K at 25 degrees Celsius. You can easily find thermistors of any type, prize and size in this range. In our web shop http://shop.grodansparadis.com you can find two types at the

time this document is written. One with accuracy around ±0.2°C and one with accuracy +-0.6°C.

For the length of the cable 100 meters should be OK for most types. If you use CAT5e cable you will have a resistance of ~8.42 ohms per 100 meters. This will add something around 1% to the temperature error reading at high temperatures. For room and environment temperature measurements (-25 – +25) 1000 meters would give the same error.

1.8.4 Temperature readings

You can get the temperature from a sensor by reading the temperature registers (see 1.9.3) or listening for temperature events from the module. The later must be setup for all but for sensor 0 which by default will send out it's temperature with 30 seconds interval.

1.8.4.1 Read temperature registers.

The temperature registers is located in pairs (Most significant byte followed my the least significant byte) at register positions 8-19 (see 1.9.3) and they can be read to get the current temperature for each sensor. The temperature is stored as a 16-bit two complement of a normalized temperature. This means you have to divide the read temperature by 100 to get the temperature value.

To get the temperature and calculate the temperature for a specific sensor.

1. Read MSB register. Read LSB register.
2. Calculate the 16-bit twos complement as MSB reading * 256 + LSB reading.
3. If the most significant bit is **not** set (Equal or less then 32767) this is a positive temperature. Divide by 100 and you have the temperature.
4. If the most significant bit **is** set (Greater than 32767) this is a negative temperature. Now invert the result (the bits are inverted; 0 becomes 1, and 1 becomes 0) and add one to the result. Dive by 100 and you have the temperature.

1.8.4.2 Temperature events.

To set up temperature events you set the interval you want to get them in the interval registers (see 1.9.4). This value is in seconds. Setting to a zero value disable temperature measurement events. After setting the register you will get temperature events on the bus with the interval you choose. The format for the events are described here 1.11.1.

The measurement format can be strange at first but it is really simple when you understand it. We take some examples

Example: A negative temperature in Celsius from sensor 1 The data part of the event will be

Byte	Description
Byte 0	130 (0x82)
Byte 1	2 (0x02)
Byte 2	255 (0xF0)
Byte 3	216 (0x60)

1. Byte 0: Normalized integer, Celsius, Sensor 1
2. Byte 1: Decimal point should be shifted two steps to the left.
3. Byte 2/3: Bit 16 is a one meaning it's a negative number. Temperature is a two complement number. 0xF060 = 0b1111000001100000. Invert which give 0b0000111110011111 add one which give 0b0000111110100000 = 4000. Divide by 100 gives 40.00 The read temperature is -40.00 °C.

Example: A positive temperature in Celsius from sensor 1 The data part of the event will be

Byte	Description
Byte 0	130 (0x82)
Byte 1	2 (0x02)
Byte 2	46 (0x2E)
Byte 3	224 (0xE0)

1. Byte 0: Normalized integer, Celsius, Sensor 1
2. Byte 1: Decimal point should be shifted two steps to the left.
3. Byte 2/3: Bit 16 is zero meaning it's a positive number. 0x2EE0 = 12000. Divide by 100 gives 120.00. The temperature is 120.00 °C.

Example: A negative temperature in Fahrenheit from sensor 5 The data part of the event will be

Byte	Description
Byte 0	249 (0xF9)
Byte 1	2 (0x02)
Byte 2	255 (0xFF)
Byte 3	118 (0x76)

1. Byte 0: Normalized integer, Fahrenheit, Sensor 5

2. Byte 1: Decimal point should be shifted two steps to the left.

3. Byte 2/3: Bit 16 is a one meaning it's a negative number. Temperature is a two complement number. 0xFF76 = 0b1111111101110110. Invert which give 0b0000000010001001 add one which give 0b0000000010001010 = 138. Divide by 100 gives 1.38 The read temperature is -1.38 °F.

1.8.5 Alarm events

The module can send alarm events if the temperature for a sensor goes below or above a certain temperature. When an alarm occurs a bit is set in the alarm register which is located in register position 128. You can always read this register to see if the module have sent out and alarm events. When you read the register the alarm bits will be cleared.

The alarm register bits will not be cleared when an alarm condition s cleared. This is because it should be possible to detect that an alarm condition have occurred even later.

For low alarms to be sent bit 3 in the temperature sensor control register must be set. For high alarms to be sent bit 4 in the temperature sensor control register must be set. See 1.9.2.

1.8.5.1 Low temperature alarm

You set the low alarm point in registers 50-58 (see 1.9.7). An alarm is sent when the temperature goes below this point and the low alarm bit is set in the alarm register. The hysteresis register at 110-115 (1.9.12) controls when a low alarm condition should be considered to be cleared. The default is five degrees. This means that if you have a low alarm point set to 0 degrees and use the default for hysteresis the alarm condition is over when the temperature raise above low alarm point + 5 = 0 + 5 = 5 degrees. Now if the temperature goes below the low set point again a new alarm event will be sent.

1.8.5.2 High temperature alarm

You set the high alarm point in registers 62-73 (see 1.9.8). An alarm is sent when the temperature goes above this point and the high alarm bit is set in the alarm register. The hysteresis register at 110-115 (1.9.12) controls when a high alarm condition should be considered to be cleared. The default is five degrees. This means that if you have a high alarm point set to 20 degrees and use the default for hysteresis the alarm condition is over when the temperature goes below the high alarm point - 5 = 20 - 5 = 15 degrees. Now if the temperature goes above the set point again a new alarm event will be sent.

1.8.6 TurnOn/TurnOff events instead of alarm events.

The module can send TurnOff/TurnOn events if the temperature for a sensor goes below or above a certain temperature and bit 5 is set (see 1.9.2). The

alarm register at position 128 position works the same way as for alarm events as described above.

1.8.6.1 Example: Using the events to build a thermostat.

You can use the Kelvin NTC10KA together with the Paris relay module (see http://auto.grodansparadis.com/paris/paris.html) or some other VSCP module to build a device that controls the temperature in a room. Suppose that you control a relay with the Paris relay module which in turn controls a heat source. If ON the temperature in the room will increase. If OFF the temperature in the room will decrease.

You program the decision matrix on the Paris relay module to turn on the heater when a TurnOn event is received (set up a zone/sub-zone for the room) and turn it off when a TurnOff event is received. You now program your Kelvin NTC10KA module so that it sends TurnOn/TunrOff events instead of alarm (bit 5 set in the control register for the sensor you want to use (see 1.9.2). Clear bit 6 in the control register to get a TurnOn at the low set point and a TurnOff at the high set point. If you used a cooler instead of a heater you could have bit 6 set to have the opposite behavior.

Set the temperature that you want to have as your low point in the low set point register (see 1.9.7). Set the high temperature in the high temperature set point register (see 1.9.8). To control a room temperature a low value at 20 degrees and a high value at 22 degrees can be OK. Now set the hysteresis value (see 1.9.12). For the 22/20 setting 1 degree might be OK. Now When the temp goes below 20 degrees a TurnOn event will be sent starting the heater. The temperature will raise in the room and at 20 + hysteresis = 20 + 1 the low temperature alarm condition is over (an new TurnOn would have been sent if the temperature went below 20 again) and as the temperature raise above 22 a TurnOff will be sent and the heater will be tuned off. Now the temperature will go down. And at 22 - hysteresis = 22 -1 the high temperature alarms condition is over. This way the temperature will be held between 20-22 degrees.

Note that with a to high hysteresis things can go strange. In the above example imagine a hysteresis set to 10. The heater will be started when the temperature goes below 20 degrees. The alarm condition will not be cleared until 20 + hysteresis = 20 + 10 = 30 degrees is reached but as the high set point is at 22 this will never be reached. At 22 degrees a TurnOff event will be sent turning of the heater. Now when the temperature goes below 20 degrees no TurnOn event will be sent because the low alarm condition is still set resulting in that the heater never goes on again. To prevent a total lockup a write to the sensors low/high set point or hysteresis reset both alarm conditions for a sensor.

1.8.7 Max/Min temperature readings

In registers 86-97 you can get the min temperature ever read by each sensor. In register 98-109 you can get the max temperature ever read by each sensor. Each value is stored in pairs. Most significant byte in the first byte of the pair

and least significant byte in the second. The number is the current temperature * 100.

To reset the min/max value for a sensor write any value to one of it's register.

1.9 Registers

All VSCP modules have a set of 8-bit registers defined. Some of them (register 128-255) are predefined and the information in them are the same for all VSCP modules. See the VSCP specification for a description of their content (http://sourceforge.net/projects/m2m/files/VSCP%20Specification/). The lower 128 register positions are used for module specific registers. It is normally here you find registers with which you configure your module. You can also find registers where you typically can read status information such as measurement data from the module.

Below is a description of the registers on the Kelvin NTC10KA module.

1.9.1 Zone registers

- **Register 0**(0x00) - Zone.
- **Register 1**(0x01) - Sub-zone.

1.9.2 Control registers

- **Register 2**(0x02) - Control byte for sensor 0.
- **Register 3**(0x03) - Control byte for sensor 1.
- **Register 4**(0x04) - Control byte for sensor 2.
- **Register 5**(0x05) - Control byte for sensor 3.
- **Register 6**(0x06) - Control byte for sensor 4.
- **Register 7**(0x07) - Control byte for sensor 5.

  Bit 0,1 - Temperature unit

  00 - Kelvin.

  01 - Celsius (default).

  10 - Fahrenheit.

  Bit 2 - Reserved.

  Bit 3 - Enable low alarm if set to one.

  Bit 4 - Enable high alarm if set to one.

  Bit 5 - Send TurnOn/TurnOff events instead of Alarm if set to one.

Bit 6 - Controls when TurnOn/TurnOff events will be sent if activated.

0 - TurnOn event is sent when low temperature is reached as set in low temperature alarm register (see 1.9.7). TurnOff event is sent when high temperature is reached as set in high temperature alarm register (see 1.9.8).

1 - TurnOff event is sent when low temperature is reached as set in low temperature alarm register (see 1.9.7). TurnOn event is sent when high temperature is reached as set in high temperature alarm register (see 1.9.8).

The hysteresis value (see 1.9.12) will effect TurnOn/TurnOff events The temperature must go above or below the hysteresis value before a new event will be sent out.

Bit 7 - If this bit is set Alarm events will be sent continuous with a one second interval until the alarm register is read or the temperature goes to a non alarm state (with hysteresis taken into account see 1.8.5.1). TurnOn/TurnOff events will not be affected by a cleared alarm register and will be sent until this bit is cleared or the temperature goes to a non alarm state (with hysteresis taken into account see 1.8.6).

1.9.3 Temperature registers

- **Register 8**(0x08) - Temperature sensor 0 MSB (on board sensor).
- **Register 9**(0x09) - Temperature sensor 0 LSB (on board sensor).
- **Register 10**(0x0A) - Temperature sensor 1 MSB
- **Register 11**(0x0B) - Temperature sensor 1 LSB
- **Register 12**(0x0C) - Temperature sensor 2 MSB
- **Register 13**(0x0D) - Temperature sensor 2 LSB
- **Register 14**(0x0E) - Temperature sensor 3 MSB
- **Register 15**(0x0F) - Temperature sensor 3 LSB
- **Register 16**(0x10) - Temperature sensor 4 MSB
- **Register 17**(0x11) - Temperature sensor 4 LSB
- **Register 18**(0x12) - Temperature sensor 5 MSB
- **Register 19**(0x13) - Temperature sensor 5 LSB

The value is stored as a twos complement number as temperature * 100.

1.9.4 Report interval registers

- **Register** 20(0x14) - Report interval for sensor 0 in seconds (Set to zero for no report).
- **Register** 21(0x15) - Report interval for sensor 1 in seconds (Set to zero for no report).
- **Register** 22(0x16) - Report interval for sensor 2 in seconds (Set to zero for no report).
- **Register** 23(0x17) - Report interval for sensor 3 in seconds (Set to zero for no report).
- **Register** 24(0x18) - Report interval for sensor 4 in seconds (Set to zero for no report).
- **Register** 25(0x19) - Report interval for sensor 5 in seconds (Set to zero for no report).

1.9.5 Reserved registers

- **Register** 26(0x1A) - **Register** 37(0x25) - Reserved.

1.9.6 B constant registers

- **Register 38**(0x26) - B constant for sensor 0 MSB.
- **Register 39**(0x27) - B constant for sensor 0 LSB.
- **Register 40**(0x28) - B constant for sensor 1 MSB.
- **Register 41**(0x29) - B constant for sensor 1 LSB.
- **Register 42**(0x2A) - B constant for sensor 2 MSB.
- **Register 43**(0x2B) - B constant for sensor 2 LSB.
- **Register 44**(0x2C) - B constant for sensor 3 MSB.
- **Register 45**(0x2D) - B constant for sensor 3 LSB.
- **Register 46**(0x2E) - B constant for sensor 4 MSB.
- **Register 47**(0x2F) - B constant for sensor 4 LSB.
- **Register 48**(0x30) - B constant for sensor 5 MSB.
- **Register 49**(0x31) - B constant for sensor 5 LSB.

1.9.7 Low alarm registers

- **Register 50**(0x32) - Low alarm set point for sensor 0 MSB.
- **Register 51**(0x33) - Low alarm set point for sensor 0 LSB.
- **Register 52**(0x34) - Low alarm set point for sensor 1 MSB.
- **Register 53**(0x35) - Low alarm set point for sensor 1 LSB.
- **Register 54**(0x36) - Low alarm set point for sensor 2 MSB.
- **Register 55**(0x37) - Low alarm set point for sensor 2 LSB.
- **Register 56**(0x38) - Low alarm set point for sensor 3 MSB.
- **Register 57**(0x39) - Low alarm set point for sensor 3 LSB.
- **Register 58**(0x3A) - Low alarm set point for sensor 4 MSB.
- **Register 59**(0x3B) - Low alarm set point for sensor 4 LSB.
- **Register 60**(0x3C) - Low alarm set point for sensor 5 MSB.
- **Register 61**(0x3D) - Low alarm set point for sensor 5 LSB.

Writing any of these registers will reset high/low alarm conditions for that sensor.

The value is stored as a twos complement number as temperature * 100.

1.9.8 High alarm registers

- **Register 62**(0x3E) - High alarm set point for sensor 0 MSB.
- **Register 63**(0x3F) - High alarm set point for sensor 0 LSB.
- **Register 64**(0x40) - High alarm set point for sensor 1 MSB.
- **Register 65**(0x41) - High alarm set point for sensor 1 LSB.
- **Register 66**(0x42) - High alarm set point for sensor 2 MSB.
- **Register 67**(0x43) - High alarm set point for sensor 2 LSB.
- **Register 68**(0x44) - High alarm set point for sensor 3 MSB.
- **Register 69**(0x45) - High alarm set point for sensor 3 LSB.
- **Register 70**(0x46) - High alarm set point for sensor 4 MSB.
- **Register 71**(0x47) - High alarm set point for sensor 4 LSB.
- **Register 72**(0x48) - High alarm set point for sensor 5 MSB.

- **Register 73**(0x49) - High alarm set point for sensor 5 LSB.

Writing any of these registers will reset high/low alarm conditions for that sensor.

The value is stored as a twos complement number as temperature * 100.

1.9.9 Sensor zone information registers

- **Register 74**(0x4A) - Zone for sensor 0 alarms.
- **Register 75**(0x4B) - Sub-zone for sensor 0 alarms.
- **Register 76**(0x4C) - Zone for sensor 1 alarms.
- **Register 77**(0x4D) - Sub-zone for sensor 1 alarms.
- **Register 78**(0x4E) - Zone for sensor 2 alarms.
- **Register 79**(0x4F) - Sub-zone for sensor 2 alarms.
- **Register 80**(0x50) - Zone for sensor 3 alarms.
- **Register 81**(0x51) - Sub-zone for sensor 3 alarms.
- **Register 82**(0x52) - Zone for sensor 4 alarms.
- **Register 83**(0x53) - Sub-zone for sensor 4 alarms.
- **Register 84**(0x54) - Zone for sensor 4 alarms..
- **Register 85**(0x55) - Sub-zone for sensor 4 alarms.

1.9.10 Sensor absolute low temperature registers

- **Register 86**(0x56) - Absolute low for sensor 0 MSB (Write to reset).
- **Register 87**(0x57) - Absolute low for sensor 0 LSB (Write to reset).
- **Register 88**(0x58) - Absolute low for sensor 1 MSB (Write to reset).
- **Register 89**(0x59) - Absolute low for sensor 1 LSB (Write to reset).
- **Register 90**(0x5A) - Absolute low for sensor 2 MSB (Write to reset).
- **Register 91**(0x5B) - Absolute low for sensor 2 LSB (Write to reset).
- **Register 92**(0x5C) - Absolute low for sensor 3 MSB (Write to reset).
- **Register 93**(0x5D) - Absolute low for sensor 3 LSB (Write to reset).
- **Register 94**(0x5E) - Absolute low for sensor 4 MSB (Write to reset).
- **Register 95**(0x5F) - Absolute low for sensor 4 LSB (Write to reset).

- **Register 96**(0x60) - Absolute low for sensor 5 MSB (Write to reset).
- **Register 97**(0x61) - Absolute low for sensor 5 LSB (Write to reset).

The value is stored as a twos complement number as temperature * 100.

1.9.11 Sensor absolute high temperature registers

- **Register 98**(0x62) - Absolute high for sensor 0 MSB (Write to reset).
- **Register 99**(0x63) - Absolute high for sensor 0 LSB (Write to reset).
- **Register 100**(0x64) - Absolute high for sensor 1 MSB (Write to reset).
- **Register 101**(0x65) - Absolute high for sensor 1 LSB (Write to reset).
- **Register 102**(0x66) - Absolute high for sensor 2 MSB (Write to reset).
- **Register 103**(0x67) - Absolute high for sensor 2 LSB (Write to reset).
- **Register 104**(0x68) - Absolute high for sensor 3 MSB (Write to reset).
- **Register 105**(0x69) - Absolute high for sensor 3 LSB (Write to reset).
- **Register 106**(0x6A) - Absolute high for sensor 4 MSB (Write to reset).
- **Register 107**(0x6B) - Absolute high for sensor 4 LSB (Write to reset).
- **Register 108**(0x6C) - Absolute high for sensor 5 MSB (Write to reset).
- **Register 109**(0x6D) - Absolute high for sensor 5 LSB (Write to reset).

The value is stored as a twos complement number as temperature * 100.

1.9.12 Sensor hysteresis registers

- **Register 110**(0x6E) - Hysteresis in degrees Celsius for sensor 0 (default is 5 degrees).
- **Register 111**(0x6F) - Hysteresis in degrees Celsius for sensor 1 (default is 5 degrees).
- **Register 112**(0x70) - Hysteresis in degrees Celsius for sensor 2 (default is 5 degrees).
- **Register 113**(0x71) - Hysteresis in degrees Celsiusin for sensor 3 (default is 5 degrees).
- **Register 114**(0x72) - Hysteresis in degrees Celsius for sensor 4 (default is 5 degrees).
- **Register 115**(0x73) - Hysteresis in degrees Celsius for sensor 5 (default is 5 degrees).

1.9.13 Sensor calibration registers

The value in the calibration register is added to the measured sensor value before it's reported. The calibration value is stored as a two's complement temperature * 100 to allow for both positive and negative numbers. So 24.23 degrees is stored as 2423 and so on.

- **Register 116**(0x74) - Index into calibration values (0-15).
- **Register 117**(0x75) - Sensor calibration values. Tvelve values stored MSB byte first.

To read a value first set the index in register 116 and then read the value in register 117. Sensor values are stored in MSB, LSB order with sensor0, sensor1, sensor2, sensor3,sensor4, sensor5.

To write a calibration value first set the index in register 116 and then write the value in register 117.

1.10 Alarm register

- **Bit 0** - Low Alarm from one or more sensor.
- **Bit 1** - High Alarm from one or more sensor.
- **Bit 2** - Reserved.
- **Bit 3** - Reserved.
- **Bit 4** - Reserved.
- **Bit 5** - Reserved.
- **Bit 6** - Reserved.
- **Bit 7** - Reserved.

Read the register to clear alarm bits. Alarm bits will be set until this register is cleared even if the actual alarm condition is not active anymore.

1.11 Events

1.11.1 Temperature event

If enabled this event is sent periodically for each sensor. The temperature is reported as a normalized integer value by this event. The event frequency is set in register 20-25 (default is one event per 30 seconds) and must be set to a non zero value for it to be sent.

Temperature can be reported in one of three units. Kelvin, Celsius (default) or Fahrenheit as set in register 2-7.

Class: 10(0x0a) Type: 6(0x06)

1.11.1.1 Data

Byte	Description
0	Format. See description below.
1	Always 2 (0x02). Decimal point should be shifted two steps to the left = divide with hundred.
2	MSB of normalized integer. Two complement number.
3	LSB of normalized integer. Two complement number.

Table 1.4: Temperature event format

Format is the measurement data coding format described in part 8 of the VSCP specification.

The normalized integer is stored as a two complement 16-bit integer. To convert

1. Calculate the 16-bit twos complement as MSB byte * 256 + LSB byte.
2. If the most significant bit is **not** set (Equal or less then 32767) this is a positive temperature. Divide by 100 and you have the temperature.
3. If the most significant bit **is** set (Greater than 32767) this is a negative temperature. Now invert the result (the bits are inverted; 0 becomes 1, and 1 becomes 0) and add one to the result. Dive by 100 and you have the temperature.

Bits	Description	
5,6,7	4 (0b100) - Normalized integer format.	
3,4	0 (0b00) - Kelvin. 1 (0b01) – Celsius. 2 (0b10) – Fahrenheit.	
0,1,2	Sensor Index:	0 (0b000) - Sensor 0. 1 (0b001) - Sensor 1. 2 (0b010) - Sensor 2. 3 (0b011) - Sensor 3. 4 (0b100) - Sensor 4. 5 (0b101) - Sensor 5.

Table 1.5: Temperature event format description.

To make it simpler to interpret data the three tables below list the format bytes for Kelvin, Celsius and Fahrenheit temperature presentation.

Sensor	Format byte
Sensor 0	129 (0x81)
Sensor 1	130(0x82)
Sensor 2	131(0x83)
Sensor 3	132(0x84)
Sensor 4	133(0x85)
Sensor 5	134(0x86)

Table 1.6: Format bytes for Kelvin presentation

Sensor	Format byte
Sensor 0	136(0x88)
Sensor 1	137(0x89)
Sensor 2	138(0x8A)
Sensor 3	139(0x8B)
Sensor 4	140(0x8C)
Sensor 5	141(0x8D)

Table 1.7: Format bytes for Celsius presentation

Sensor	Format byte
Sensor 0	244 (0xF4)
Sensor 1	245 (0xF5)
Sensor 2	246 (0xF6)
Sensor 3	247 (0xF7)
Sensor 4	248 (0xF8)
Sensor 5	249 (0xF9)

Table 1.8: Format bytes for Fahrenheit presentation

1.11.2 Alarm Event

If enabled the event is sent when a temperature sensor goes below a low alarm set point (see 1.9.7) or a high alarm set point (see 1.9.8). The hysteresis registers is used so after an alarm event has been sent a new alarm event is not sent until the temperature goes below/above the value set in the hysteresis register. A read of the alarm register will reset the alarm status and alarm events will not be sent out again until the the temperature changed with the hysteresis amount.

Class: 0x001 Type: 0x02

1.11.2.1 Data

Byte	Description
0	Index
1	Zone
2	Sub-zone

Table 1.9: Temperature event format

Index is 0 for Sensor 0. 1 for Sensor 1, 2 for Sensor 2. 3 for Sensor 3. 4 for Sensor 4. 5 for Sensor 5.

Zone and sub-zone is the modules setting (see 1.9.1).

1.11.3 TurnOn Event

If enabled the event (Bit 5 in control register must be set (see 1.9.2)) is sent when the temperature goes below or above the low(see 1.9.7)/high(see 1.9.8) alarm setpoints. Settings in the control register bit 6 (see 1.9.2) decides which of the TurnOn/TurnOff event that is sent. The hysteresis setting (see 1.9.12) tells how much a temperature must raise or fall below the alarm set point before a new event will be sent.

Class: 30 (0x1E) Type: 5 (0x05)

1.11.3.1 Data

Byte	Description
0	Index
1	Zone
2	Sub-zone

Table 1.10: Temperature event format

Index is 0 for Sensor 0. 1 for Sensor 1, 2 for Sensor 2. 3 for Sensor 3. 4 for Sensor 4. 5 for Sensor 5.

Zone information is fetched from sensor zone register (see 1.9.9) if it contains a value that is not zero. If it has a zero value the module zone will be used (see 1.9.1) instead.

Sub-zone information is fetched from sensor sub-zone register (see 1.9.9) if it contains a value that is not zero. If it has a zero value the module sub-zone will be used (see 1.9.1) instead..

1.11.4 TurnOff Event

If enabled the event (Bit 5 in control register must be set (see 1.9.2)) is sent when the temperature goes below or above the low(see 1.9.7)/high(see 1.9.8) alarm setpoints. Settings in the control register bit 6 (see 1.9.2) decides which of the TurnOn/TurnOff event that is sent. The hysteresis setting (see 1.9.12) tells how much a temperature must raise or fall below the alarm set point before a new event will be sent.

Class: 30 (0x1E) Type: 6 (0x06)

1.11.4.1 Data

Byte	Description
0	Index
1	Zone
2	Sub-zone

Table 1.11: Temperature event format

Index is 0 for Sensor 0. 1 for Sensor 1, 2 for Sensor 2. 3 for Sensor 3. 4 for Sensor 4. 5 for Sensor 5.

Zone information is fetched from sensor zone register (see 1.9.9) if it contains a value that is not zero. If it has a zero value the module zone will be used (see 1.9.1) instead.

Sub-zone information is fetched from sensor sub-zone register (see 1.9.9) if it contains a value that is not zero. If it has a zero value the module sub-zone will be used (see 1.9.1) instead.

1.11.5 Sync Event (Incoming event).

If a SYNC event is received by the module it will check the zone/subzone parameters of the event and send out temperature measurement events for all sensors that match. This can be a handy feature to use of one want syncronized data from several sources.

Class: 30 (0x1E) Type: 26 (0x1A)

1.11.5.1 Data

Byte	Description
0	Index
1	Zone
2	Sub-zone

Table 1.12: Temperature event format

Index is not used.

1.12 Where can I find the source code?

Most VSCP modules from Grodans Paradis AB is Open hardware/Open source meaning that both the hardware information as well as the source code is available. This means that you can modify the source code and /or the hardware to your specific needs if you want.

1.13 Appendix A - Mandatory VSCP registers.

<table>
<tr><th>Address</th><th>Access Mode</th><th>Description</th></tr>
<tr><td>0Œ00 – 0x7f</td><td>—</td><td>Device specific. Unimplemented registers should return zero.</td></tr>
<tr><td>128/0Œ80</td><td>Read Only</td><td>Alarm status register content (!= 0 indicates alarm). Condition is reset by a read operation. The bits represent different alarm conditions.</td></tr>
<tr><td>129/0Œ81</td><td>Read Only</td><td>VSCP Major version number this device is constructed for.</td></tr>
<tr><td>130/0Œ82</td><td>Read Only</td><td>VSCP Minor version number this device is constructed for.</td></tr>
<tr><td>131/0Œ83</td><td>Read/Write</td><td>Node control flags
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>7</td><td>Start-up control</td></tr>
<tr><td>6</td><td>Start-up control</td></tr>
<tr><td>5</td><td>r/w control of registers below 0Œ80. (1 means write enabled)</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>3</td><td>Reserved</td></tr>
<tr><td>2</td><td>Reserved</td></tr>
<tr><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>132/0Œ84</td><td>Read/Write</td><td>User ID 0 – Client settable node id byte 0.</td></tr>
<tr><td>133/0Œ85</td><td>Read/Write</td><td>User ID 1 – Client settable node id byte 1.</td></tr>
<tr><td>134/0Œ86</td><td>Read/Write</td><td>User ID 2 – Client settable node id byte 2.</td></tr>
<tr><td>135/0Œ87</td><td>Read/Write</td><td>User ID 3 – Client settable node id byte 3.</td></tr>
<tr><td>136/0Œ88</td><td>Read/Write</td><td>User ID 4 – Client settable node id byte 4.</td></tr>
<tr><td>137/0Œ89</td><td>Read only</td><td>Manufacturer device ID byte 0.</td></tr>
<tr><td>138/0x8a</td><td>Read only</td><td>Manufacturer device ID byte 1.</td></tr>
<tr><td>139/0x8b</td><td>Read only</td><td>Manufacturer device ID byte 2.</td></tr>
<tr><td>140/0x8c</td><td>Read only</td><td>Manufacturer device ID byte 3.</td></tr>
<tr><td>141/0x8d</td><td>Read only</td><td>Manufacturer sub device ID byte 0.</td></tr>
<tr><td>142/0x8e</td><td>Read only</td><td>Manufacturer sub device ID byte 1.</td></tr>
<tr><td>143/0x8f</td><td>Read only</td><td>Manufacturer sub device ID byte 2.</td></tr>
<tr><td>144/0Œ90</td><td>Read only</td><td>Manufacturer sub device ID byte 3.</td></tr>
<tr><td>145/0Œ91</td><td>Read only</td><td>Nickname id for node if assigned or 0xff if no nickname id assigned.</td></tr>
<tr><td>146/0Œ92</td><td>Read/Write</td><td>Page select register MSB</td></tr>
<tr><td>147/0Œ93</td><td>Read/Write</td><td>Page Select register LSB</td></tr>
<tr><td>148/0Œ94</td><td>Read Only</td><td>Firmware major version number.</td></tr>
<tr><td>149/0Œ95</td><td>Read Only</td><td>Firmware minor version number.</td></tr>
<tr><td>150/0Œ96</td><td>Read Only</td><td>Firmware sub minor version number.</td></tr>
<tr><td>151/0Œ97</td><td>Read Only</td><td>Boot loader algorithm used. 0Xff for no boot loader support. Codes for algorithms are specified here VSCP_event_class_000 for Type = 12</td></tr>
<tr><td>152/0Œ98</td><td>Read Only</td><td>Buffer size. The value here gives an indication for clients that want to talk to this node if it can support the larger mid level Level I control events which has the full GUID. If set to 0 the default size should used. That is 8 bytes for Level I and 512-25 for Level II.</td></tr>
<tr><td>153/0Œ99</td><td>Read Only</td><td>Number of register pages used. If not implemented one page is assumed.</td></tr>
<tr><td>154/0x9A-207/0xcf</td><td>—</td><td>Reserved for future use. Return zero.</td></tr>
<tr><td>208/0xd0-223/0xdf</td><td>Read Only</td><td>128-bit (16-byte) globally unique ID (GUID) identifier for the device. This identifier uniquely identifies the device throughout the world and can give additional information on where driver and driver information can be found for the device. MSB for the identifier is stored first (in 0xd0).</td></tr>
<tr><td>224/0xe0-255/0xff</td><td>Read Only</td><td>Module Description File URL. A zero terminates the ASCII string if not exactly 32 bytes long. The URL points to a file that gives further information about where drivers for different environments are located. Can be returned as a zero string for devices with low memory. It is recommended that unimplemented registers read as oxff. For a node with an embedded MDF return a zero string. The CLASS1.PROTOCOL, Type=34/35 can then be used to get the information if available.</td></tr>
</table>

Table 1.13: VSCP mandatory registers

Index