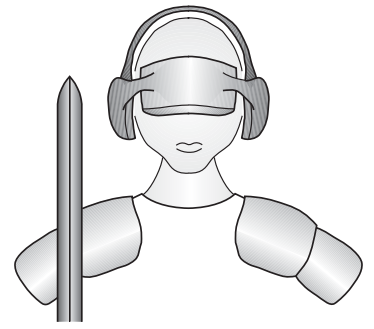


# AN4: Measuring high temperature with IO-Warrior

Applicable for IOW24



Code Mercenaries

## 1. Overview

Measuring physical parameters with IO-Warrior chips can be accomplished relatively simply by adding external A/D converters or specialised sensor interface chips.

This application note shows how to control a MAX6675 by Maxim Integrated Products via the SPI function of IO-Warrior 24. The MAX6675 measures temperatures in the range of 0°C to 1024°C using a type-K thermocouple.

## 2. Theory of operation

The MAX6675 is an extremely simple to use chip. It only needs the SPI connections to the IO-Warrior and a power supply filter capacitor in addition to the thermocouple. No configuration data needs to be sent to the MAX6675.

Data from the MAX6675 is read in a 16 bit package. The SPI of IOW24 always sends and receives data at the same time (like any bidirectional SPI device) while the MAX6675 only transmits data and has no data input. A dummy two byte packet is transmitted via the IOW24 to receive the data from the MAX6675.

### 2.1 Setting up the SPI function

Before data can be read from the MAX6675 it is necessary to enable the SPI function of the IOW24 and set the proper communication parameters. While IOW24 is capable of up to 2MBit/sec data speed on the SPI it is advisable to use the slowest rate of 62.5kBit/sec as this produces less RF noise and allows longer connections between the IOW24 and the MAX6675.

The MAX6675 data sheet shows the SPI clock to idle low (CPOL = 0) and the data bits are shifted

out on the falling edges of the clock (second edge, CPHA = 1).

The complete report to write is therefore:  
\$08 \$01 \$07 \$00 \$00 \$00 \$00 \$00

### 2.2 Reading data packets

Reading and writing of the SPI is a single operation so the data from the MAXIM 6675 is retrieved by writing a report with dummy data to the SPI and then getting back a report with the received data.

The MAX 6675 sends two bytes of data so a report with a payload of two dummy bytes has to be written to the SPI. The dummy bytes may contain any data as the MOSI output through which this data is shifted is not used.

The required report contains a byte with the byte count and flags. None of the three flag bits needs to be set for the MAXIM 6675 since the transfer is done in one report and there are no handshake signals, so only the data count has to be set to 2 bytes.

The resulting report to write is:  
\$09 \$02 \$00 \$00 \$00 \$00 \$00 \$00

Each time the above report is written to the IOW24 it sends back a report containing the two bytes of data payload from the MAX6675:

\$09 \$02 \$yy \$xx \$00 \$00 \$00 \$00

yy is the high byte of the data word and xx the low byte. The lowest 3 bits contain flags so after shifting the value 3 bits to the right you get the temperature in 1/4°C. Value 0 is 0°C and \$0FFF is 1023.75 °C.

Bit 2 of the flags is 1 if no thermocouple is attached to the MAX6675.

# AN4: Measuring high temperature with IO-Warrior

---

## 3 Delphi-Program using HID component

For the program there is only one design decision to be made: How often should the device be read? Temperature measurement is usually not fast due to the thermal inertia of the sensors. Also the A/D conversion in the MAX6675 is relatively slow with a typical conversion time of 0.17sec. Reading the temperature once a second can be considered more than adequate.

So the Delphi sample program consists of a label to show the temperature, a timer to drive the SPI and the HID component to access the IOW24. The chart component is simply some eye candy.

The program handles only one IOW24 so it uses the first one found and expects it to have the MAX6675 connected to it.

The strategy of the OnDeviceChange event of the HID component is simple:

Check if the OnDeviceChange has been caused by the unplug of the IOW24 in use. If so then free this now unusable device object and deactivate the timer to read sensor data.

If no IOW24 is in use then try to find one. After all the OnDeviceChange may have been caused by a plug of an IOW24.

The search is for a HID device with VendorID = \$07C0, ProductID = \$1501 and an output report size of 8 bytes. This designates the special mode interface of an IOW24.

The timer is activated to fire once a second. The timer event in turn writes the dummy bytes to the SPI. As result the OnData event of the HID device object fires.

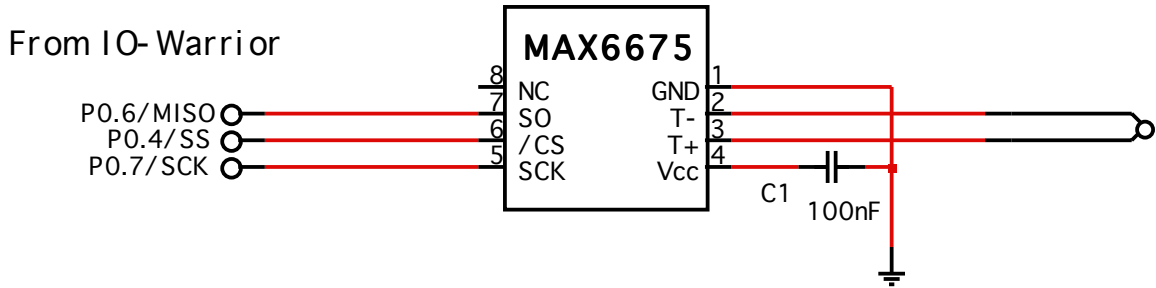
In the OnData event the 12 bit temperature payload gets extracted and displayed on the label. The data is also fed to the chart component which displays the last 60 values.

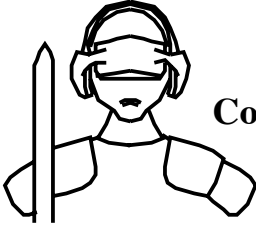
### 3.1 Delphi-Program using iowkit.dll

The difference to the above program is the even simpler strategy. It scans the available IO-Warrior for the first IOW24 available and uses it. The Read/Write is a simple pair of calls.

# AN4: Measuring high temperature with IO-Warrior

Fig. 1: The circuit



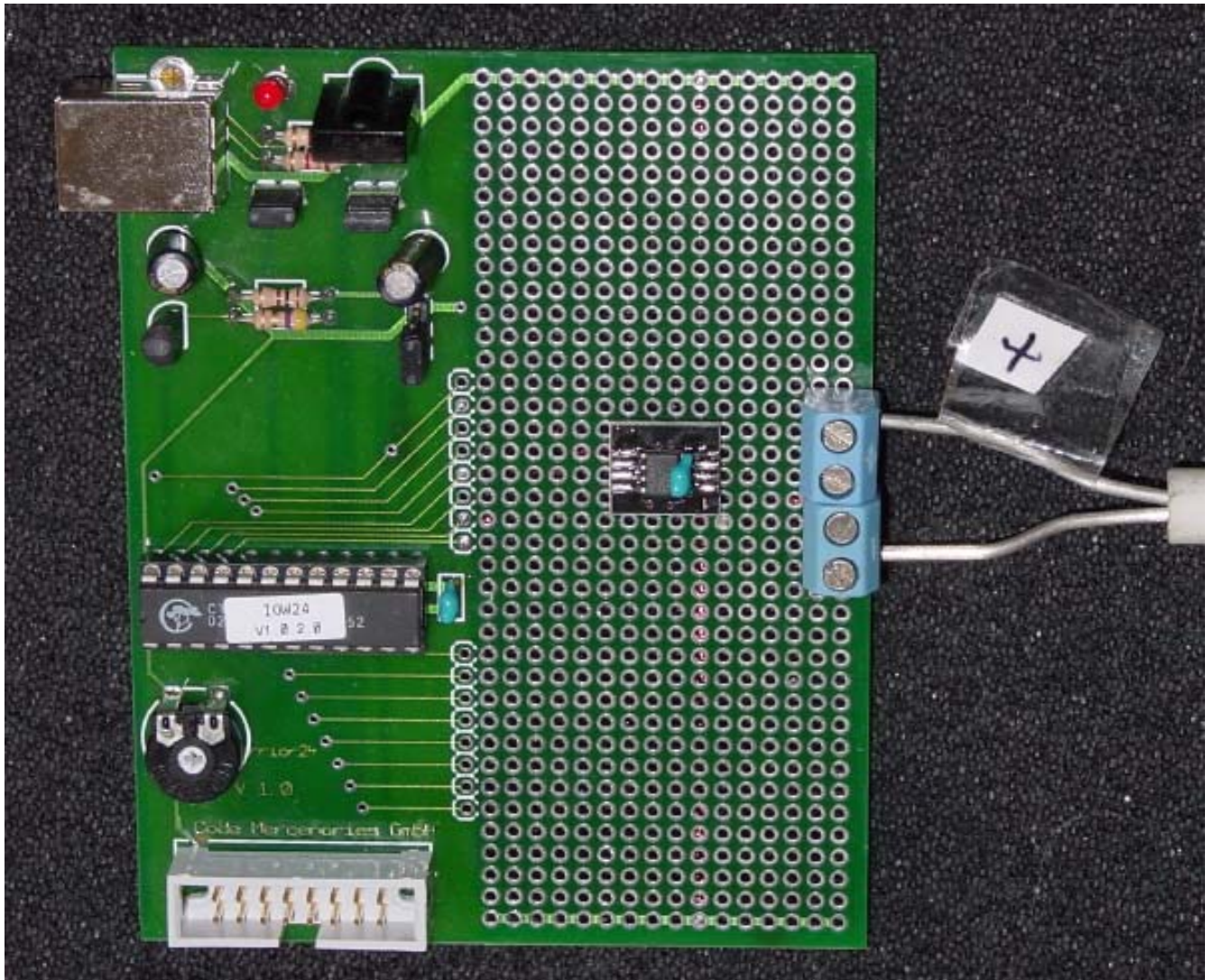
<b>Circuit:</b> IOW24 Thermocouple				 <p>Code Mercenaries</p>
<b>Version:</b> 1.0				
<b>Date:</b> 6.05.2005				
<b>Drawn by:</b>				
<b>Function:</b>				
<b>Page:</b>				
Rev.	Date	By	Change	Sign.

The 100nF capacitor must be placed as close as possible to the power supply pins of the MAX6675.

To enable the sensor detection of the MAX6675 the T- input must be connected to ground.

# AN4: Measuring high temperature with IO-Warrior

Fig. 2: Sample setup



The MAX6675 is available only in the SOIC8 package. To simplify the setup on the starter kit breadboard area we used a SO8 to DIL8 adapter board which is available at most electronic distributors.

# AN4: Measuring high temperature with IO-Warrior

---

## Legal Stuff

This document is ©2005 by Code Mercenaries.

The information contained herein is subject to change without notice. Code Mercenaries makes no claims as to the completeness or correctness of the information contained in this document.

Code Mercenaries assumes no responsibility for the use of any circuitry other than circuitry embodied in a Code Mercenaries product. Nor does it convey or imply any license under patent or other rights.

Code Mercenaries products may not be used in any medical apparatus or other technical products that are critical for the functioning of lifesaving or supporting systems. We define these systems as such that in the case of failure may lead to the death or injury of a person. Incorporation in such a system requires the explicit written permission of the president of Code Mercenaries.

Trademarks used in this document are properties of their respective owners.

Code Mercenaries  
Hard- und Software GmbH  
Karl-Marx-Str. 147a  
12529 Schönefeld OT Grossziethen  
Germany  
Tel: x49-3379-20509-20  
Fax: x49-3379-20509-30  
Mail: support@codemerics.com  
Web: www.codemerics.com

HRB 16007 P  
Geschäftsführer: Guido Körber, Christian Lucht