# Harsh Environment Acquisition Terminal (H.E.A.T.) System EVM User's Guide

# User's Guide

Literature Number: SLVU468B
June 2011–Revised August 2012

# Contents

1 Overview .......... 5
2 System Design Guidelines .......... 7
2.1 PCB Details .......... 7
2.1.1 PCB Materials .......... 7
2.1.2 Layout Guidelines and Recommendations .......... 7
2.2 Derating of Integrated Circuits and Passive Components .......... 7
3 Hardware Setup .......... 9
3.1 Power-Up Requirements / Sequence .......... 10
3.2 ADS1278-HT Connection Options .......... 10
3.3 Voltage Reference .......... 11
3.4 SM470R1B1M-HT Connection Options .......... 11
3.5 ADS1278-HT Analog Inputs .......... 11
3.5.1 Channels 1, 2, and 3: Inclination Circuits .......... 11
3.5.2 Channel 4: Simple Amplifier Using the OPA211-HT .......... 11
3.5.3 Channel 5: RTD .......... 12
3.5.4 Channel 6: INA333-HT Instrumentation Amplifier .......... 12
3.5.5 Channel 7: THS4521-HT Differential Amplifier .......... 12
3.5.6 Channel 8: Discrete Differential Amplifier Circuit .......... 12
4 Software Installation .......... 13
4.1 First Time Installation .......... 13
4.2 Upgrading Existing Installation .......... 13
5 Software Application Overview .......... 14
5.1 Connect / Disconnect .......... 14
5.2 Output / Stop .......... 14
5.3 Status Label .......... 15
5.4 Temperature Data .......... 15
5.5 Axis Data (Line Charts) .......... 15
5.6 Four Miscellaneous A/D Channels (Not Visible on GUI) .......... 15
6 Output File .......... 16
7 Calibration .......... 17
8 Firmware Code .......... 18
A Establishing Tool Communication Via Terminal .......... 27
A.1 Interpreting Data Frames .......... 27
B Hexadecimal .......... 28

## List of Figures

1-1. HEATEVM Block Diagram .......... 5
3-1. HEATEVM ARM7 / ADC Communications .......... 9
3-2. Inclination Block Diagram .......... 11
5-1. HEATEVM Demonstration Screen .......... 14

## List of Tables

A-1. Configuration .......... 27
A-2. Data Frame Example Breakdown .......... 27

This document outlines the basic steps and functions that are required to ensure proper operation of the Harsh Environment Acquisition Terminal (H.E.A.T.) system evaluation module (EVM) kit. The HEATEVM is a high-temperature development platform that provides a complete signal-acquisition system for operating temperature environments of 200°C. All of the high-temperature semiconductor devices are qualified and characterized from -55°C to 210°C operation for 1,000 hour operating life, thus eliminating the need for expensive up-screening components or using commercial products who’s temperature range falls outside the datasheet specifications.

# Overview

HEATEVM platform contains a signal chain consisting of an eight channel analog input which conditions, digitizes and processes all of these signals. The EVM has six channels dedicated for temperature, pressure, and accelerometers and two are general purpose channels (i.e. fully differential and single-ended).

Figure 1-1 shows the HEATEVM block diagram. The analog inputs channels are comprised of six channels dedicated for temperature, pressure and accelerometers. The OPA211-HT enables accelerometer inputs and all three axis inputs have a buffer amplifier with two poles in a 20-Hz Butterworth filter. Temperature sensing is performed with the THS4521-HT and OPA2333-HT devices. For pressure sensing, a high impedance bridge instrumentation amplifier is used from the THS4521-HT and the OPA2333-HT devices. This circuit realizes a high gain and is ideal for measuring pressure in harsh-environment applications such as down-hole drilling. HEATEVM also has dedicated INA333-HT inputs for pressure transducer coupling.

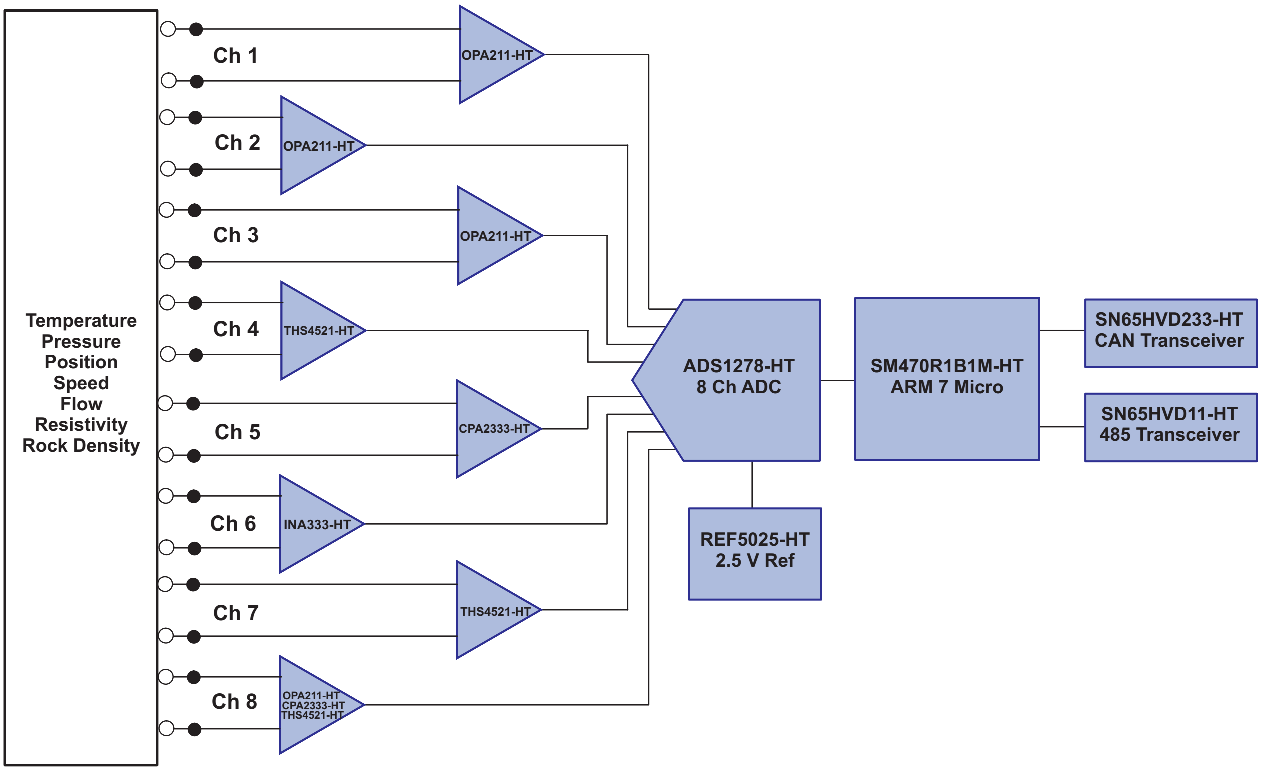


**Figure 1-1. HEATEVM Block Diagram**

The platform is suited for interfacing with external sensors. The user is expected to interface these sensors by means of user generated cabling. The HEATEVM is pre-loaded with basic firmware layer for the ARM7 microcontroller and source-code is provided in this User Manual. The data acquired by the system can be extracted by the HEATEVM graphical user interface (GUI) software. The GUI performs auto scrolling of data, provides a temperature display, illustrates actual minimum and maximum values, and provides data logging capability for all eight channels. This GUI software is optional, and the user can program their own firmware into the ARM7 microcontroller by means of the JTAG port on the HEATEVM. The optional software runs on a PC terminal and proper serial interfacing is required. The optional software is available for downloading through:

http://focus.ti.com/docs/toolsw/folders/print/heatevm.html

The HEATEVM kit includes one EVM board. All power supply requirements/sequencing need to be provided externally and are outlined in the Power-Up Requirements/Sequencing section 3.1 of this manual. Please refer to the HEATEVM schematic, Gerber and BOM files for component use and their locations and references from this user guide. The schematics, gerbers and BOM files are available at:

http://focus.ti.com/docs/toolsw/folders/print/heatevm.html

# System Design Guidelines

## 2.1 PCB Details

The PCB in the HEATEVM is built from P95 polyimide material and has very good mechanical stability up to temperatures in excess of 210°C.

### 2.1.1 PCB Materials

The dielectric constant of the polymide varies between 3.74 and 3.95. With a base layer PCB thickness of 0.093 inches, the 8 layer HEATEVM has a reasonable amount of stiffness to reduce strain on solder joints. For prolonged 210°C elevated testing periods, the polyimide PCB needs to be kept dry and best results are obtained when testing is conducted inside a dry pressure chamber. In general, this is the case with many harsh environment high temperature drilling and logging tools.

The surface copper traces and pads are protected with a nickel-gold finish. The nickel isolates the gold from the copper, while the gold provides for good solder bonding. This type of system prevents the tin found in most solders from forming intermetalic bronze with the copper traces. For short term exposure at elevated temperatures or temperatures below 98°C, the nickel-gold finish may not be required, however, this added margin safeguards against the intermetallic bronze formations.

### 2.1.2 Layout Guidelines and Recommendations

Power and ground plains are suggested for optimizing the performance of the ADS1278. Decoupling capacitance should be physically placed as close as possible to the power pins of the ADS1278 and the SN470R1B1M-HT ARM7 microcontroller to minimize capacitance and gain optimized performance. The location of these decoupling capacitors becomes even more critical in high temperature operations since, in addition to the higher costs, larger valued capacitors become the life limiting component in many high-temperature systems. Most capacitor types have falling capacitance values and increasing IR values as the ambient temperature increases which present trade offs when working with capacitors at elevated temperatures. The HEATEVM uses a nominal number of capacitors and value of capacitance.

To reduce temperature effects, 200°C EIA rated capacitors have been used. The COG dielectric has a flat temperature coefficient over temperature and are also reffered to as NPO (negative-positive-zero), referring to its value change over temperature. These COG/NPO capacitors are used as decoupling capacitors while 200°C rated tantalum capacitors are used to reduce power supply ripple to the circuit board. To reduce ripple on the voltage reference, a 1uF stacked capacitor was used to maintain fast transient response to reduce noise.

## 2.2 Derating of Integrated Circuits and Passive Components

When designing high-temperature circuits, engineers normally derate the electronic components from the manufacturer’s specifications. This is particularly true for the voltage rating of capacitors and the power rating of resistors. As more manufacturers produce electronic components rated for operation at elevated temperatures and engineers gain experience working with these components, the act of derating will become less important.

In any case, the overall rating of an electronic circuit will always be less than the rating of the weakest component in the circuit. Virtually every high temperature application is also a high reliability application. For example, in high reliability applications using all electronic components rated to 200°C and 1,000 operating hours, an overall derating of the electronic circuit could be as high as 50°C.

Despite the advancement of computer generated life prediction models, most manufacturers of high temperature control systems require hard numbers in performing accelerated life testing such as HALT (Highly Accelerated Life Tests) to find weak links in the design and/or manufacturing process. By starting with manufacturer qualified high temperature electronic components, the design engineer will enjoy a higher percentage of successful HALT testing of new products. Perhaps just as important, is knowing that the high temperature electronic component manufacture will continue to support a high temperature rating though design cycles which within the electronic industry are normally much shorter than the high temperature, high reliability product being designed. For example, an aircraft engine control system can be in production for extened product life-cycles between 20 to 30 years while an electronic component may see a new design cycle every 6 to 10 years.

For high-temperature operation, resistors are choosen for their temperature drift and hydrogen sensitivity. The HEATEVM uses low temperature drift, thin film resistors with a temperature drift of 30ppm/°C.

# Hardware Setup

The HEATEVM system is controlled by the SN470R1B1M-HT ARM7 microcontroller in conjunction with ADS1278-HT Octal 24-Bit 128KSPS Analog-to-Digital Converter (ADC).

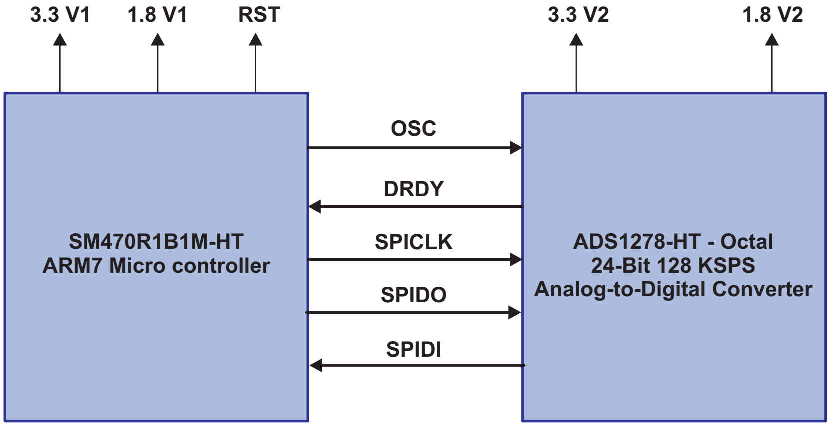


**Figure 3-1. HEATEVM ARM7 / ADC Communications**

The communications between the SM470R1B1M-HT ARM7 Microcontroller and the ADS1278-HT Octal 24-Bit 128KSPS Analog-to-Digital Converter is established by using the oscillate signal. The ARM7 microcontroller reads the ADS1278-HTdata stream using the SPI1 interface on the ARM7. The data ready line (ADSync, Pin 39 of the ADS1278-HT) signals the ARM7 that data is ready for reading on its pin 1 (which is defined as a general I/O pin in software). The three SPI1CLK, SPI1DO, SPI1DI signals from the ARM7 SPI1 are used. The software setup for the SPI1 interface is polarity 1 and phase 0 and the ADS1278-HT data changes on the negative SPI1CLK.

The ADS1278-HT, requires an external clock for operation. In the default case, the ARM7 generates an ADCLK signal from its ECLK, pin 80. ECLK is software programmable and in the source-code included in this manual, ECLK is set to 1.25MHz.

There is no software setup between the ARM7 and the ADS1278-HT as the setup options on the ADS1278-HT set up are defined by the user's choice of format and mode options with the channel enable pins.

The HEATEVM is shipped with preset format pin options on the ADS1278-HT as Format [001] which are set by soldered jumpers on PJ10 and establishes a Time-Division Multiplexed bit stream. With this format, the position of each 24 bit ADC reading is fixed in time such that disabling any channel results in a 000000H reading for that channel and all the ADS1278-HT readings are serial out on pin DOUT1. If the user desires to change the format from this default setting, high-temperature solder should be used.

Each of the eight ADS1278-HT channels are enabled by default. To change the default channel enables, solder jumpers at JP12. Each jumper pair is tied to each PWDN channel enable (pins 45 to 52).

The ADS1278 has 4 mode options which can be set by the user by jumpers on JP11. The default setting is Mode[11] for low speed.

The power supply rails have been separated since the power-up sequence differs between the ADC and ARM7 devices.

## 3.1 Power-Up Requirements / Sequence

The HEATEVM has split power inputs supporting the ARM7 and the ADS1278-HT. Both the ARM7 and the ADS1278-HT have 3.3V IO and 1.8V core voltages. Splitting the supply inputs allows for differing power-up sequences for each chip. All voltages reference a common ground. The GND is tied to a ground plane located on layer 2 of the PCB.

The ARM7 power input and its reset (PORRSTN, pin 68 of the ARM7) is on JP24. The SM470R1B1M-HT, document SPNS155C requires that the 3.3V IO voltage is greater than 1.1V before the 1.8V core voltage is greater than 0.6V. Specific timing requirements for the PORRSTN input are also outlined in SPNS155C.

Pin 1 on JP24 has a square pad

JP24- Pin 1: 3.3V (This is the VIO for the ARM7)

JP24- Pin 2: GND

JP24- Pin 3: PORRSTN (This is the power on or off reset for the ARM7)

JP24- Pin 4: 1.8V (This is the ARM7 core voltage)

JP24- Pin 5: GND The power input for the ADS1278 is on JP2.

The 5V supply needed by the analog input circuits and the ADS1278-HT analog voltage (AVDD) is on JP25. For detailed information on the power-up sequence of the ADS1278-HT, see SBAS447B. In general, core voltage should reach a nominal 1V before the IO voltage reaches 1V. The analog voltage must be last, with a nominal 3 Volts triggering an internal reset to the chip. Note, the power up sequence is different than the ARM7 so tying supply inputs together is not recommended.

Pin 1 on JP2 has a square pad

JP2- Pin 1: 3.3V (This is the VIO for the ADS1278-HT)

JP2- Pin 2: GND

JP2- Pin 3: NC

JP2- Pin 4: 1.8V (This is the core voltage for the ADS1278)

JP2- Pin 5: GND

Pin 1 on JP25 has a square pad

JP25- Pin 1: 5V (Analog voltage)

JP25- Pin 2: GND

## 3.2 ADS1278-HT Connection Options

The ADS1278-HT has a several pin assigned options. To provide the user with maximum flexibility, the following jumper blocks connected to the following pin assignment pins of the ADS1278-HT. In all cases, a jumper installed results in a ‘low’ setting.

- JP10 pins set the AD Format Options
- JP11 pins set the AD Mode Options
- JP12 pins enable the AD Channels

For running the demonstration board at 200°C, some jumpers are hardwired. The user can change these setting by removing or adding a solders jumper. These jumpers are of the same board layout as the other jumper blocks, as such all the hardwired jumpers can be replaced with jumper blocks if so desired.

- JP13 is jumper to set CLKDIV to ‘low’ input.
- JP16 are jumpers to interface the ADS1278 to SM470R1B1M (ARM7). The ADS1278 is connecting to the ARM7 as a SPI device.

## 3.3 Voltage Reference

The REF5025-HT is supplied as the on board voltage reference. The REF5025-HT has a nominal voltage of 2.5 V. The on board voltage reference can be replaced by placing a jumper between pins 2 and 3 of JP5 and applying a new external voltage to JP6 with pin 1 as GND and pin 2 as the positive voltage.

## 3.4 SM470R1B1M-HT Connection Options

The TI ARM7 is in circuit programmable using either the 20-pin JP20 or the 14-pin JP19. Because the demonstration board can be heated to 200°C, a standard gold plated pin header is used in both cases. The user must check the pin orientation against the circuit layout. Looking at the circuit board, with the ARM7 chip toward the user’s left hand, pin 1 on JP20 and JP19 is on the top left side of each pin header. A 1.8-V oscillator is strongly recommended to drive the SM470R1B1M-HT as indicated in the datasheet.

## 3.5 ADS1278-HT Analog Inputs

The ADS1278-HT has eight differential inputs to its eight parallel 24-bit analog to digital converters. The HEATEVM provides input to all eight channels using the analog circuits described below.

### *3.5.1 Channels 1, 2, and 3: Inclination Circuits*

Measuring inclination (or tilt) is a common measurement in the harsh-environment down-hole drilling industry. For example, most drilling tools measure 3-axis inclination to determine what the orientation of the drill bit is. The first 3 channels of the ADS1278-HT are examples of a filtered buffer/amplifier for use in measuring the 3-axis inclination signals.

The HEATEVM inclination signals are filtered in an active two pole Butterworth filter using the using the OPA211-HT. Low Pass filters of 20 Hz to reduce 60-Hz noise picked up by the cable running between the HEATEVM (inside an oven) and the inclination sensor board located outside of the oven.

When the cable is not connected, a precision resistor network provided a fixed DC voltage to the three inclination inputs on the HEATEVM. This allows operation without the inclination circuit.

An RTD circuit can be added to the HEATEVM to provide a differential signal to the THS4521-HT and will need to use the REF5025-HT to bias the circuit. The OPA211-HT can be used to buffer the REF5025-HT to the circuit. The REF5025-HT will also provide the reference voltage to the ADS1278-HT.

Figure 3-2 illustrates the block diagram of the inclination circuits.

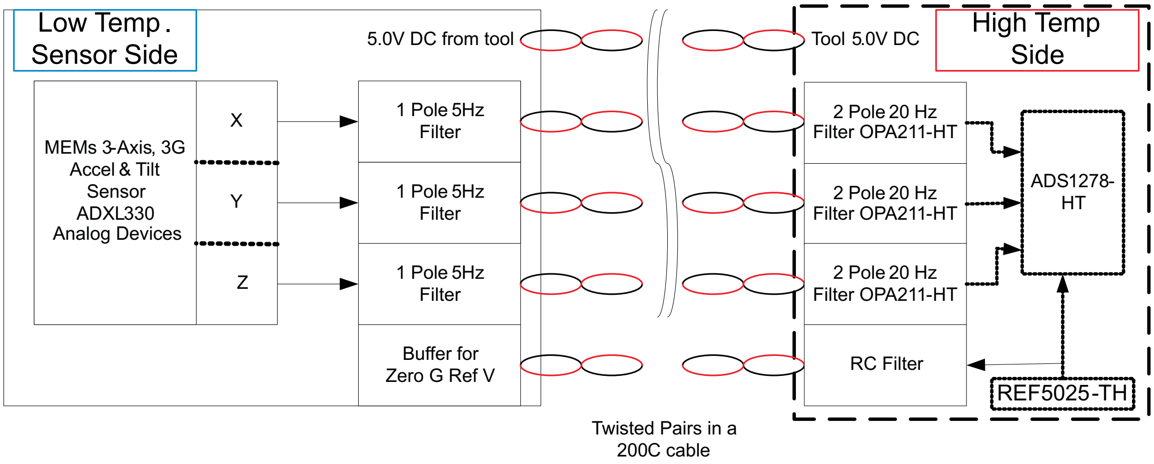

Low Temp . Sensor Side
5.0V DC from tool
Tool 5.0V DC
High Temp Side
MEMs 3-Axis, 3G Accel & Tilt Sensor ADXL330 Analog Devices
X
Y
Z
1 Pole 5Hz Filter
1 Pole 5Hz Filter
1 Pole 5Hz Filter
Buffer for Zero G Ref V
2 Pole 20 Hz Filter OPA211-HT
2 Pole 20 Hz Filter OPA211-HT
2 Pole 20 Hz Filter OPA211-HT
RC Filter
ADS1278-HT
REF5025-TH
Twisted Pairs in a 200C cable

**Figure 3-2. Inclination Block Diagram**

### *3.5.2 Channel 4: Simple Amplifier Using the OPA211-HT*

A simple amplifier circuit using the OPA211-HT with a fixed gain of -10. The user can supply an input signal to pin 7 of JP26. The amplifier has a 1.25-V reference at the +in pin. PT8 is a test point for monitoring the amplifiers output.

### *3.5.3 Channel 5: RTD*

Channel 5 of the ADS1278-HT is setup to measure board temperature using an onboard 1000-Ω RTD located at JP8. An external RTD can be used by removing the onboard RTD and the two jumpers on JP8 and replacing them with the standard 4-wire RTD.

In Well Logging systems, it is typical to detect relative temperature changes of 0.001ºC so the 24-bit accuracy associated with the ADS1278-HT is necessary in order to facilitate this measurement.

The RTD has a nearly flat response of 3.85Ω/ºC starting with a nominal 1000 Ω at 0ºC. At 20ºC room temperature, the RTD is ~1077 Ω and ~1385 Ω at 100ºC. The use of 3.85 Ω per degree will create ~1-2ºC error at 200ºC. Use of a typical second-degree polynomial curve fit will improve this measurement while also correcting for small circuit offsets. However, for demonstration purposes these small errors are negligible for a board mounted sensor and no system calibration unique to each board was performed.

The RTD measurement circuit uses the voltage reference (IC6, REF5025-HT) to create a bias current for a 1000-Ω RTD. The reference voltage is 2.5 V. This voltage is buffered (IC7, OPA2333 “B”) with a resistor divider network of R25 and R26. The resistor divider is used to reduce the reference voltage to 0.5 V. This voltage is used to create a DC bias to the positive input to +INA of IC7.

The “A” amplifier of IC7 (OPA2333-HT) has the 1000-Ω RTD in its feedback. This amplifier holds the voltage across R24 to the bias value of 0.5 V. This sets the current in the feedback RTD resistor to $I_{RDT}$ = 0.5/R24. As R24 is a fixed, high precision resistor equal to 1000 Ω, $I_{RDT}$ = 0.005 A and the voltage across the RTD at 0ºC is 0.5 V. As the temperature increases, the voltage across the RTD increases at 0.0173 V/ºC.

IC8 (THS4521) is used to amplify the differential signal across the RTD. The gain of IC8 is fixed by R19/R18 = R20/R21 = 2. At 0ºC, the differential output is 2 x 0.5 = 1 V. The output is then subject to change over temperature by ~0.0346 V/ºC.

The differential amplifier, IC8 (THS4521-HT) has its VOCM pin either as an open circuit or tied back to the ADS1278, VCOM pin using jumper block JP9. In the case of the RTD circuit, a zero temperature creates a differential voltage of 0.5 V, so an offset is accounted for in the display software

### *3.5.4 Channel 6: INA333-HT Instrumentation Amplifier*

Channel 6 of the ADS1278-HT is setup as a differential channel and is a user illustrated circuit based on the INA333-HT and pins 6 and 5 of JP26. R42 is used to set the gain value of the INA333-HT. The gain is calculated by Gain = 1 + (100k/R42), for R42 = 100 Ω the gain is 1001. A simple RC filter is placed between the amplifier and the ADS1278. Test point PT10 is available to monitor the amplifiers output.

### *3.5.5 Channel 7: THS4521-HT Differential Amplifier*

Channel 6 of the ADS1278-HT is setup as a differential channel and a differential amplifier circuit for user input signals based on the THS4521-HT. Full access to the THS4521 input pins are available at JP26, pins 1-4. The amplifier has a fix gain of 1 set by R47/R48 and R51/R52. This amplifier is operating from the 3.3-V supply.

### *3.5.6 Channel 8: Discrete Differential Amplifier Circuit*

Channel 8 of the ADS1278-HT is setup as a differential channel and is left for the users discretion. It can be used for simulating the measurement of a standard 4-wire bridge type pressure transducer. In this case, the pressure transducer cable would be connected to JP8. IC10 (OPA211-HT) is used to bias the bridge circuit to 2.5 V. The bridge differential voltage is input is buffered by IC11 (OPA2333-HT). The OPA2333-HT is precision amplifier well suited for this type of application. IC14 (THS4521-HT) is used to amplify the buffered signals coming from IC11. The gain of this circuit is ~501 set by the resistor pairs R54/R53 and R56/R55. A small differential RC filter is in between the outputs of IC14 and the ADS1278.

# Software Installation

The software installation for the HEATEVM system is broken into two sections: First Time Installation and Upgrading Existing Installation.

## 4.1 First Time Installation

The software installation has several prerequisites that may or may not be already installed on your PC. If the software is already installed you can safely proceed to the next step.

Prerequisites:

1. Microsoft .NET Framework 3.5 sp1
   http://www.microsoft.com/downloads/en/details.aspx?FamilyId=333325fd-ae52-4e35-b531-508d977d32a6
2. Microsoft Charting
   This can be found in the in the 3rd Party folder of the program files: MSChart.exe

   OR

   Download from:
   http://www.microsoft.com/downloads/en/details.aspx?FamilyId=130F7986-BF49-4FE5-9CA8-910AE6EA442C

Installation:

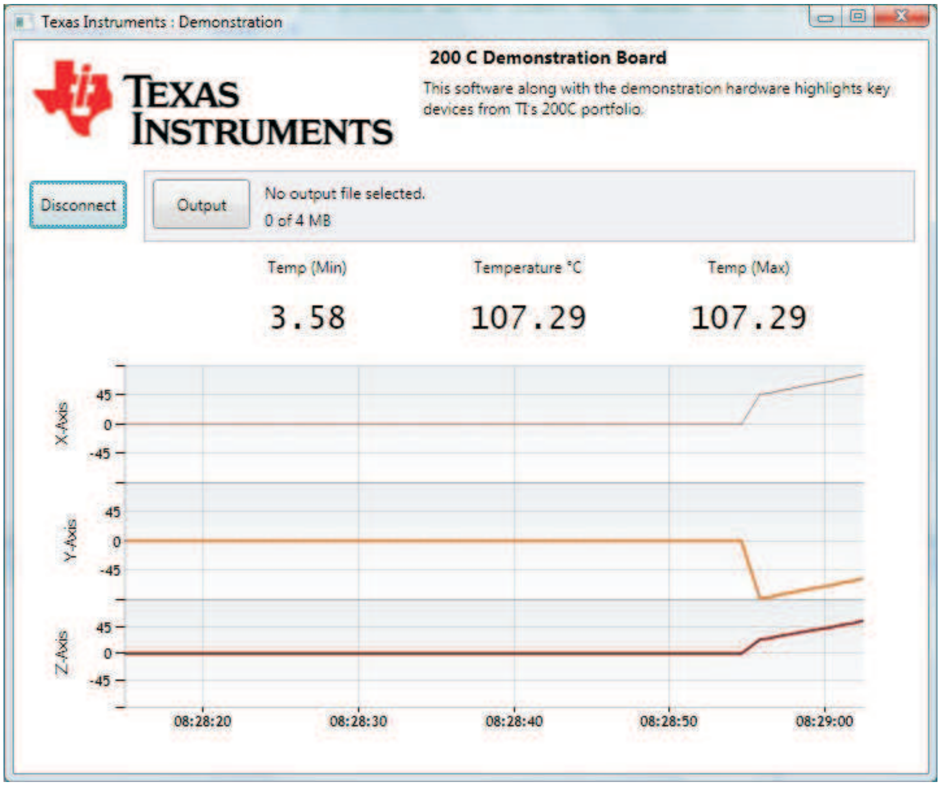
1. Install Microsoft .NET Framework 3.5 sp1 and follow instructions.
2. Install Microsoft Charting and follow instructions.
3. Copy the program folder onto the local computer. The software can run from the desktop or can be placed in Program Files along with other software.

## 4.2 Upgrading Existing Installation

When upgrading to a new version there is only one file that could be important to preserve between versions. It is located in the program folder and is the TIDemo.exe.config file. This file, as you'll read later, possibly contains calibration information regarding your actual hardware. Other files can be simply copied over or removed and replaced with the updated software version.

# Software Application Overview

The HEATEVM demonstration screen (Figure 5-1) illustrates what will be seen on the user terminal interface.


Texas Instruments : Demonstration

200 C Demonstration Board

This software along with the demonstration hardware highlights key devices from TI's 200C portfolio.

No output file selected.
0 of 4 MB

Temp (Min) 3.58

Temperature °C 107.29

Temp (Max) 107.29

**Figure 5-1. HEATEVM Demonstration Screen**

## 5.1 Connect / Disconnect

The upper left most button is labeled “Connect” or “Disconnect” depending on whether or not you're currently connected and receiving data from the electronics.

The HEATEVM uses serial port (COM) communication. This is frequently provided by USB to Serial connectors in recent computers. If the computer has only a single COM port installed the Connect button will automatically try to connect to that port. If there are more than one COM port available a window will pop up allowing you to choose which COM port is connected to the TI Demo hardware.

## 5.2 Output / Stop

This button is labeled “Output” or “Stop” depending on whether or not the real time data is being saved to a file. The output button allows the saving of the real time data into a file. By selecting the output file all data from that point begins to be saved. Data that was received prior to setting an output file is not retroactively saved.

## 5.3 Status Label

The file name label below and to the right of the Output button is a status label, here it shows “0 of 4 MB”. When saving data to an output file this counter will demonstrate how much raw binary data is being saved. This correlates to the future version of a memory demonstration where the 200 C memory will be 4 MB.

It will also report errors if it cannot successfully interpret the data frame being sent from the electronics or if the electronics themselves report an error.

1. Electronics error, if the analog to digital circuit is not responding to the processor it will flash in red: “!ADC communication failure“
2. Communications error, if the software is unable to interpret the data frame: “Cannot parse: [index, value]”

There are log files saved with non-critical exceptions but can be useful in identifying issues.

## 5.4 Temperature Data

Low, max, and current temperatures are displayed here. These are reported from a built-in RTD on the board. See calibration section on how to calibrate the values.

## 5.5 Axis Data (Line Charts)

The initial demonstration of the Demo Board is to have an accelerometer transmitting data from all three axes (X, Y, Z). These values are demonstrated here in degrees. See calibration section for how to calibrate the values.

## 5.6 Four Miscellaneous A/D Channels (Not Visible on GUI)

There are four additional channels in the eight channel A/D converter that are not displayed but are still recorded in the output file. These values do not have a calibration applied to them but the data is accessible via Excel or other comma separated values (csv) software. See the section on the output file.

Chapter 6

# Output File

The output file contains time data, 4 calibrated channels, and 4 non-calibrated channels in a comma separated file. The 4 calibrated correspond to the 4 displayed data in the software: temperature, x-axis, y-axis, and z-axis.

```
# TI 200C Demonstration Output File
# Below is a CSV file of the data seen while running the demonstration
# Recording started: 4/19/2011 8:55:02 AM
Time         Temp (F)  X Axis  Y Axis  Z Axis  AD4      AD6      AD7      AD8
   55:02.70    185.97 -79.27  -34.21   78.29 -8388608 -8388608 -8388608 -8388608
   55:02.90    189.55  -78.2  -33.14   79.36 -8388608 -8388608 -8388608 -8388608
   55:03.20    193.12 -77.13  -32.06   80.44 -8388608 -8388608 -8388608 -8388608
```

The other four channels can be raw data as well if the calibration is changed to pass through the value rather than modifying it. See calibration section.

The four non-calibrated channels cannot be calibrated via the demonstration software.

# Calibration

Calibration applies coefficients and functions to obtain an accurate representation of a sensor's measurement. To do this to the analog to digital converter the coefficients can be entered into a text file prior to running the software.

Within the program's folders there is a file “TIDemo.exe.config”. Open this file with a text editor such as Notepad and you'll see something similar to below.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
       <appSettings>
    <!-- To fake serial data for GUI testing make the value="Fake" below  -->
    <add key="COMPort" value="Fake" />

    <!-- To allow the chart axis to show all possible values (24bit int) rather than
     the axis +/-90 degrees set value="True" below.  -->
    <add key="FullRange" value="False" />

    <!-- Temp Coefficients Match Cal Function  y(x) = a*x+b  -->
    <add key="TempCoeff1" value="0.0000357632473028550992430112654229" />
    <add key="TempCoeff2" value="0" />

    <!-- Axis Coefficients Match Cal Function y(x) = a*x+b  -->
    <add key="AxisCoeff1" value="0.0000107288360595703125" />
    <add key="AxisCoeff2" value="0" />

  </appSettings>
</configuration>
```

The TempCoeff1, and TempCoeff2 correspond to the a and b variables of the point slope function y(x) = a x x + b respectively. When the AD counts (x) are evaluated the result is the RTD's temperature.

The AxisCoeff1, and AxisCoeff2 do the same thing for the axes calibration into degrees.

By changing Coeff1 value to 1 and Coeff2 value to 0 for both Temp and Axis the resulting output will be the raw AD data just like the other 4 channels which have no calibrations applied.

If this is done, or any calibration applied that would result in a range greater than 90, the “FullRange” value can be set to “True” so that the data can still be visible in the line charts.

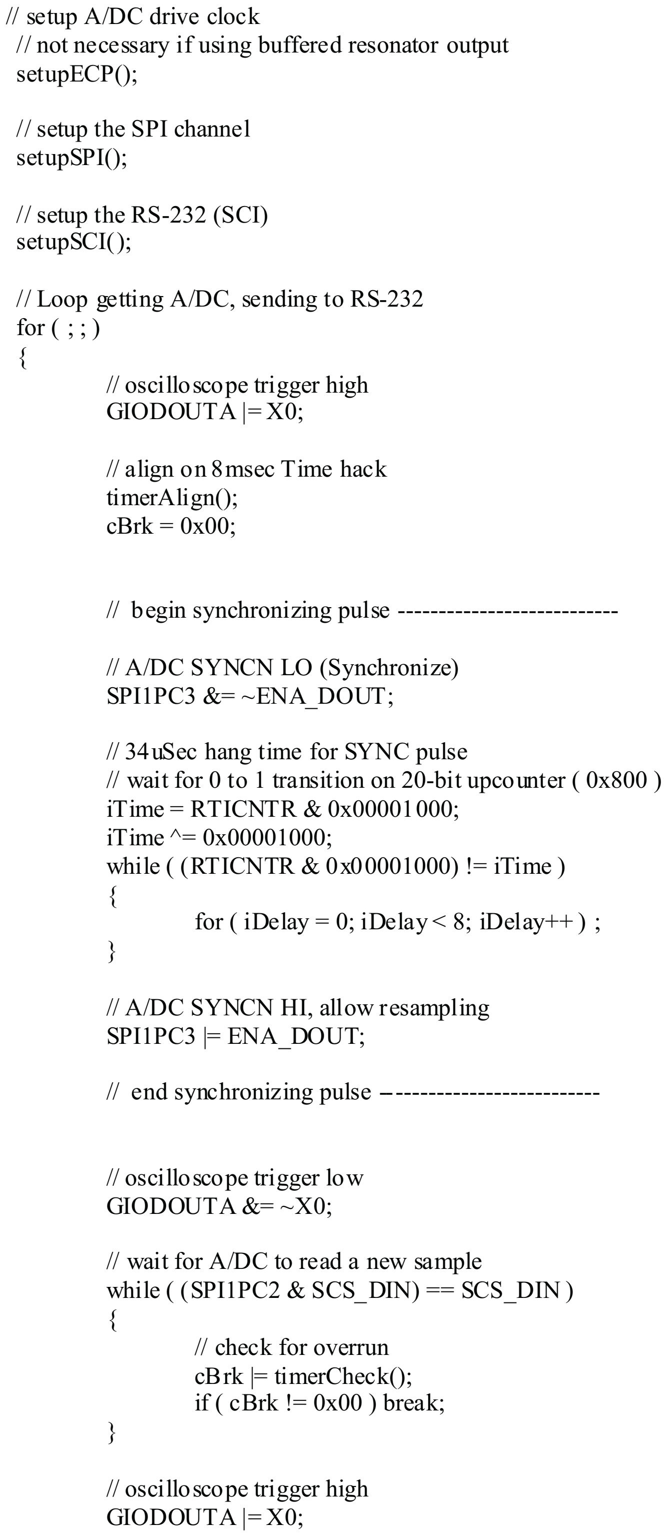
Changes to this file are only interpreted at the application start. So if a change has been made here the software must be closed and re-opened to recognize it.

# Firmware Code

Listed below is a printout of the Firmware code supplied with the HEATEVM. This code is written in C programming language using the IAR Workbench for the SM470R1B1M-HT (ARM7).

```
    /*
     Master file for ARM7 Tool firmware
     Platform:               IAR Workbench, MSWindows
     Target:                 TMS470R1B1M

     Master Oscillator speed 7.500MHz
     Send a 1.25 MHz clock to drive the A/DC conversions
     Send SPI CLK to A/DC SPI system  (SPIch1)
     Read SPI MISO from A/DC
     Send SCI RS-232 data out to retrieval device (COM1)
     New tool version :- no lcd, no Port D

     Febrary 2011
     Alastair Black
     Charles D. Normann
    */

    /*
     Notes for tool characteristics;

     REF OSC from Experimenter Board - 5MHz
     SYSCLK = REF OSC x 8 = 40MHz
     ICLK = SYSCLK / 2 = 20MHz (CLKDIV = 2)
     CLKOUT = ICLK / 16 = 1.25MHz (EPCON, 16)
     RS232 = 57600 (00 01 45)
    */

    #include <intrinsic.h>
    #include "iotms470r1b1m.h"
    #include "tms470r1b1m_bit_definitions.h"
    #include "lcd.h"

    // comment the following line if program for TOOL; decomment for DEVKIT board
    //#define LCD_EXISTS 1

    // comment the following line if no timeout wanted for testing transmit
    //#define NO_TIMEOUT 1

    /*
    // 280 mSec outer loop
    #define TIMER_MAXVAL 0x00800000

    // 17.6 mSec outer loop
    #define TIMER_MAXVAL 0x00080000
    */

    // ?? outer loop
#define TIMER_MAXVAL 0x00400000
```

```
void setupECP(void);                // CLKOUT A/DC oscillator drive setup
void setupSPI(void);                // SPI setup
void setupSCI(void);                // RS-232 setup
char read24SPI( char *buf );
void timerAlign(void);        // wait for 8msec boundary
char timerCheck(void);        // 8msec passed? 0x00: less than    0xFF: more than


void main(void)
{
 char                 szASCbuf[64], *ps;
 char                 cBrk = 0x00;
 int                            iTime, iDelay;
 short                iCtr = 0;

 #ifdef LCD_EXISTS
 // ========================
 int x_pos = 0;                               // lcd display
 int x_pos_old = 0;
 // ========================
 #endif // LCD_EXISTS

 // the following values are based on 7.5MHz oscillator
 // SYSCLK = 8 x fOSC     (60.0MHz)
 GCR &= ~ZPLL_MULT4;
 GCR &= ~ZPLL_CLK_DIV_PRE1;
 GCR &= ~ZPLL_CLK_DIV_PRE2;
 GCR &= ~ZPLL_CLK_DIV_PRE3;
//          GCR = ZPLL_CLK_DIV_PRE_1;
//          GCR = ZPLL_CLK_DIV_PRE_4;

//          PCR = CLKDIV_16;                 // ICLK = SYSCLK / 16   (3.75MHZ)
//          PCR = CLKDIV_8;                  // ICLK = SYSCLK / 8     (7.50MHz)
//          PCR = CLKDIV_4;                  // ICLK = SYSCLK / 4     (15.0MHz)
 PCR = CLKDIV_3;                  // ICLK = SYSCLK / 3     (20.0MHz)
//          PCR = CLKDIV_2;                  // ICLK = SYSCLK / 2     (30.0MHz)

//          PCR |= RTI_CTRL;

 PCR |= PENABLE;                  // Enable peripherals

 #ifdef LCD_EXISTS
 // ========================
 HETDIR  = 0xFFFFFFFF;                // HETx Output direction
 HETDOUT = 0x00000000;

 init_lcd();
 light_on();

 put_cursor( 0,0 );
 put_text( "Ready " );
 // ========================
 #endif // LCD_EXISTS
```



```
// setup A/DC drive clock
 // not necessary if using buffered resonator output
 setupECP();

 // setup the SPI channel
 setupSPI();

 // setup the RS-232 (SCI)
 setupSCI();

 // Loop getting A/DC, sending to RS-232
 for ( ; ; )
 {
         // oscilloscope trigger high
         GIODOUTA |= X0;

         // align on 8msec Time hack
         timerAlign();
         cBrk = 0x00;


         //  begin synchronizing pulse ---------------------------

         // A/DC SYNCN LO (Synchronize)
         SPI1PC3 &= ~ENA_DOUT;

         // 34uSec hang time for SYNC pulse
         // wait for 0 to 1 transition on 20-bit upcounter ( 0x800 )
         iTime = RTICNTR & 0x00001000;
         iTime ^= 0x00001000;
         while ( (RTICNTR & 0x00001000) != iTime )
         {
                 for ( iDelay = 0; iDelay < 8; iDelay++ ) ;
         }

         // A/DC SYNCN HI, allow resampling
         SPI1PC3 |= ENA_DOUT;

         //  end synchronizing pulse –-------------------------


         // oscilloscope trigger low
         GIODOUTA &= ~X0;

         // wait for A/DC to read a new sample
         while ( (SPI1PC2 & SCS_DIN) == SCS_DIN )
         {
                 // check for overrun
                 cBrk |= timerCheck();
                 if ( cBrk != 0x00 ) break;
         }

         // oscilloscope trigger high
         GIODOUTA |= X0;
// read 24 bytes from A/DC
          cBrk |= read24SPI( &szASCbuf[0] );

          // send output string to RS-232
          // 24 x 2 digit chars                          48
          // 8 commas                                                     8
          // [CR]                                                         1
          //                                            total            57
          ps = &szASCbuf[0];
          for ( iCtr = 0; iCtr < 57; iCtr++ )
          {
                  // might be "timeout" string
                  if ( *ps == 0 ) break;

                  // RS-232 out
                  SCI1TXBUF = *ps++;

                  // RS-232 finished transmitting?
                  while ( (SCI1CTL2 & TXRDY) == 0 )
                  {
                          for ( iDelay = 0; iDelay < 8; iDelay++ ) ;
                  }
          }       // 56 Hex ASCII chars plus [CR]

          #ifdef LCD_EXISTS
          // =======================
          szASCbuf[6] = (char) 0x00;
          put_cursor( 0,0 );
          put_text( szASCbuf );
          // =======================
          #endif // LCD_EXISTS

 }        // while forever
} // main()

// CLKOUT A/DC oscillator drive setup
// CLKOUT appears on pin 48 of the 84-pin device
void setupECP(void)
{
 // spnu202b.pdf ECP Ref for TMS470R1
//        ECPCTRL &= ~ECPEN;                        // undo enable

 CLKCNTL |= CLKDIR | CLKSR_ICLK;
//        CLKCNTL = 0x20;

 // with ICLK at 20MHz
//        ECPCTRL = ECPCOS | 0x03;        // ECPCOS is ON, divide by 4 (default 5.0MHz)
//        ECPCTRL = ECPCOS | 0x07;        // ECPCOS is ON, divide by 8 (default 2.5MHz)
 ECPCTRL = ECPCOS | 0x0F;        // ECPCOS is ON, divide by 16 (1.25MHz)
//        ECPCTRL = ECPCOS | 0x0F;        // ECPCOS is ON, divide by 16 (1.25MHz)
//        ECPCTRL = ECPCOS | 0x1F;        // ECPCOS is ON, divide by 32 (633KHz)
//        ECPCTRL = ECPCOS | 0x3F;        // ECPCOS is ON, divide by 64
//        ECPCTRL |= ECPCOS | 0x7F;       // ECPCOS is ON, divide by 128
```

```
ECPCTRL |= ECPEN;                          // enable
}

// SPI setup
// A/DC 1278-HT, DOUT changes on falling SCLK
// Therefore, TMS470 should use "Polarity 1, Phase 0" mode
// SPI:EN -> GPIO In </DRDY>
// SPI:CS -> GPIO Out <scope trigger>
void setupSPI(void)
{
 SPI1CTRL1 = 0 | CHARLEN_8;                                   // 8 bits per xfer

 // SYSCLK = 40MHz, ICLK = 20MHz

 SPI1CTRL1 |= PRESCALE_2;                                     // SPICLK = 5MHz
//          SPI1CTRL1 |= PRESCALE_4;                                  // SPICLK = 5MHz
 SPI1CTRL1 |= PRESCALE_8;                                     // SPICLK = 2.5MHz
//          SPI1CTRL1 |= PRESCALE_16;                                 // SPICLK = 1.25MHz
//          SPI1CTRL1 |= PRESCALE_32;                                 // SPICLK = 0.625MHz
//          SPI1CTRL1 |= PRESCALE_64;                                 // SPICLK = 3125KHz


 // *** swapping SPI:CS and SPI:EN
 // setup CS for A/DC "SYNCN" sync
 SPI1PC1 |= ENA_DIR;
 // default EN is set already for input
 SPI1PC1 &= ~SCS_DIR;


 SPI1CTRL2 |= MASTER;                                         // We are the master
 SPI1CTRL2 |= CLKMOD;                                                 // We drive the clock

//          SPI1CTRL2 |= POLARITY;                                            // CLK Polarity = 1
 SPI1CTRL2 |= PHASE;                                                  // PHASE = 1

 SPI1PC6 |= CLK_FUN;
 SPI1PC6 |= SIMO_FUN;
 SPI1PC6 |= SOMI_FUN;

 //  Sync'ing input from SPIENA 4-pin mode.
//          SPI1CTRL3 |= ENABLE_HIGHZ;

 // enable
 SPI1CTRL2 |= SPIEN;

 // PortA,0   add a FLAG (output) to assert SYNC (Convert)
 GIODIRA = 0 | X0;
//          GIODIRD = 0 | X4;

 // PortD,5   add a READY sensor (input)
//          GIODIRD &= ~X5;
}

// channel 2 RS-232 setup
```

```
void setupSCI(void)
{
 // allow peripheral config changes
 SCI1CTL3 = 0;

 SCI1CCR = 0x07;
 SCI1CTL2 |= TXENA;
 SCI1CTL2 |= TXWAKE;
 SCI1PC3 |= TX_FUNC;

/*
 // ICLK = 25MHz
 // 57600 baud == 0x000145
 SCI1HBAUD = 0x00;
 SCI1MBAUD = 0x01;
 SCI1LBAUD = 0x45;
*/

 // old values from weird clocks
 // 38400 baud = 00 00 1F
 // 19200 baud = 00 00 3F


 // ICLK = 20MHz
 SCI1HBAUD = 0x00;
 SCI1MBAUD = 0x01;
 SCI1LBAUD = 0x58;        // changeme
 // nominal   00 01 45
 // other day 00 01 58
 // double is 00 02 8A

 SCI1CTL3 = 0 | CLOCK | TX_ACTION_ENA;
 SCI1CTL3 |= SW_NRESET;
}

//////////////////
// SPI Read, reformat
//////////////////

// read 24 bytes from SPI
// store in <buf> as HEX ASCII + Comma
// replace last Comma with [CR]
char read24SPI( char *buf )
{
 static char szErr[] = "! A/DC Timeout\r";
 short              iCtr, iComma = 3;
 char               cFlag = 0x00;
 char               cSPI, cHi, cLo, *ps;

 ps = buf;

 for ( iCtr = 0; iCtr < 24; iCtr++ )
 {
```

```
/*
        // wait for A/DC -DRDY
        while ( (SPI1PC2 & SCS_DIN) != SCS_DIN )
        {
                // check for overrun
                cFlag |= timerCheck();
                if ( cFlag != 0x00 ) break;
        }
*/
        // send 8 bits on the SPI channel
        SPI1DAT0 = iCtr;
        // wait for 8 bits received on SPI channel
        while ( (SPI1CTRL3 & RX_INT_FLAG) == 0 )
        {
                cFlag |= timerCheck();
                if ( cFlag != 0x00 ) break;
        }

        // did we timeout?
        if ( cFlag == 0x00 )
        {
                // no, save this byte as HEX ASCII
//                      cSPI = 0 - SPI1BUF;
//                      cSPI = ~SPI1BUF;


/*
                if ( iComma == 3 )
                        cSPI = SPI1BUF ^ 0x80;   // 00 to FF, not 2's complement
                else
*/
                        cSPI = SPI1BUF;                  // no change bytes 2 and 3

                cHi = (cSPI & 0xF0) >> 4 ;
                if ( cHi > 0x09 ) cHi += 0x37;
                else cHi += 0x30;
                cLo = cSPI & 0x0F ;
                if ( cLo > 0x09 ) cLo += 0x37;
                else cLo += 0x30;
                *ps++ = cHi;
                *ps++ = cLo;
                if ( --iComma == 0 )
                {
                        iComma = 3;
                        // on the last character, make EOL, not Comma
                        if ( iCtr == 23 ) *ps++ = 0x0D;
                        else *ps++ = ',';
                        *ps = '\0';
                }
        }
        else
        {
                // yes, timeout asserted, say "!Timeout"
                iCtr = 0;
ps = szErr;
                while ( *ps != '\0' )
                {
                        buf[iCtr++] = *ps++;
                }
                buf[iCtr++] = *ps++;
                break;
        }
 }
 return cFlag;
}
//////////////////
// timer routines
//////////////////

// wait until we are on an 20ms boundary
// uses the RTICNTR top 21 bits upcounter
// assume 7.5MHz oscillator
// PLL multiple of 8 = 60MHz SYSCLK
// 11-bit modulus timer has rate of 60MHz / 2048 = 29296.875 Hz
// 11-bit modulus timer has overflow period of 34uSec
// 21-bit upcounter rate is 34uSec
//   1111 1111 1111 1111 1111 1... .... ....
// 1         34uSec            00 0800
// 2         68uSec            00 1000
// 4         140uSec           00 2000
// 8         280uSec           00 4000
// 10        560uSec           00 8000
// 20        1.1mSec           01 0000
// 40        2.2mSec           02 0000
// 80        4.4mSec           04 0000
// 100       8.8mSec           08 0000
// 200       17.6mSec          10 0000
// ------ full word mask:
// 0004 0000         8ms
// 0008 0000        17ms
// 0010 0000        34ms
// 0020 0000        68ms
// 0040 0000       140ms
// 0080 0000       280ms

void timerAlign(void)
{
 int                iTime, iDelay;

 iTime = RTICNTR;

 // wait for 0 to 1 transition
 while ( (iTime & TIMER_MAXVAL) == 0x00000000 )
 {
        for ( iDelay = 0; iDelay < 8; iDelay++ ) ;
        iTime = RTICNTR;
 }
```

```
// wait for 1 to 0 transition
 while ( (iTime & TIMER_MAXVAL) != 0x00000000 )
 {
        for ( iDelay = 0; iDelay < 8; iDelay++ ) ;
        iTime = RTICNTR;
 }
}

// check for timer 17ms overrun
char timerCheck(void)
{
        int             iTime;
 char   cRetVal = 0x00;

 iTime = RTICNTR;
 if ( (iTime & TIMER_MAXVAL) != 0x00000000 )
 {
        cRetVal = 0xFF ;
 }

#ifdef NO_TIMEOUT
//================
 cRetVal = 0x00;
//================
#endif // NO_TIMEOUT

 return cRetVal;
}
```

Appendix A

SLVU468B–June 2011–Revised August 2012

# Establishing Tool Communication Via Terminal

The tool data can be seen and interpreted directly without using the demonstration software.

Close all software that might be using the same port the electronics are connected to. Start your terminal program with the following settings:

**Table A-1. Configuration**

| | |
|---|---|
| Connect using: | Installed COM Port tool is attached to |
| Bits per second: | 576000 |
| Data bits: | 8 |
| Parity: | None |
| Stop bits: | 1 |
| Flow control: | None |

If the electronics are currently logging information its output should appear almost immediately.

## A.1 Interpreting Data Frames

Data frames are visible while communicating with the tool in a terminal.

Terminal output example:

```
9E58BE,BD8D85,800071,BD07F2,C9D0CE,7FFDE9,80005C,80006C
```

The frame is a series of numeric values in hexadecimal format separated by commas. The values represent A/D counts. Converting these A/D counts into sensor data requires calibration as described above. Accessing the data in this manner removes any limitations on interpretation.

Terminal error output example:

```
!Error message here
```

Any data frame that starts with an exclamation point is an error frame and the cause of the error is contained in the text following it.

**Table A-2. Data Frame Example Breakdown**

| 9E58BE,BD8D85,800071,BD07F2,C9D0CE,7FFDE9,80005C,80006C | |
|---|---|
| 9E58BE | 24-bit channel (x-axis) |
| BD8D85 | 24-bit channel (x-axis) |
| 800071 | 24-bit channel (x-axis) |
| BD07F2 | 24-bit channel (spare) |
| C9D0CE | 24-bit channel (temperature) |
| 7FFDE9 | 24-bit channel (spare) |
| 80005C | 24-bit channel (spare) |
| 80006C | 24-bit channel (spare) |

*Appendix B*

*SLVU468B–June 2011–Revised August 2012*

# Hexadecimal

Hexadecimal is the number format output by the electronics. For quick reference hexadecimal (base 16) is counted as follows with decimal (base 10).

**Decimal (Base 10)**

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | ... | 24 | 25 | 26 | 27 |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | ... | 18 | 19 | 1A | 1B |

**Hexadecimal (Base 16)**