AN65977

PSoC® 3 and PSoC 5LP - Creating an Interface to a TMP05/TMP06 Digital Temperature Sensor

Author: Rodolfo Lossio
Associated Project: Yes
Associated Part Family: PSoC 3, PSoC 5LP
Software Version: PSoC Creator™ v2.1 SP1

Abstract

AN65977 describes the TMP05 Digital Temperature Sensor Interface Component, which is a building block for thermal management applications. It enables designers using PSoC to quickly and easily interface with multiple Analog Devices' TMP05 or TMP06 digital temperature sensors through a simple, serial 2-wire digital interface.

Contents

Introduction 1
Typical System Architecture 2
Design Considerations when using TMP05 Sensors in Daisy-Chain Mode 2
Getting Started with the TMP05 Digital Temperature Sensor Interface Component 3
Example 1 – Firmware Details 4
Example 2 – Firmware Details 5
Example 3 – Firmware Details 6
Example 4 – Firmware Details 8
Future Projects 11
Conclusion 11

Introduction

Digital temperature sensors are widely used in systems where multiple temperature measurements are required in specific locations on a large board (such as a line card in a switch or router) or in a remote location. The use of digital temperature sensors frees the designer from worrying about digital noise coupling on to sensitive analog signals on the PCB layout because the digital temperature sensor does the analog to digital conversion within its own package, at the location where the temperature measurement is needed.

The most common digital temperature sensors use industry-standard communications interfaces such as I^2C or SPI. These interfaces are well known and need no description here. For applications in which the thermal management controller has no I^2C or SPI interfaces available, an alternative is Analog Devices' TMP05 and TMP06 monolithic temperature sensors that generate a pulse-width modulated (PWM) serial digital output.

The duty cycle of the PWM output varies in direct proportion to the ambient temperature of the devices. The high period (T_H) of the PWM remains static over all temperatures, while the low period (T_L) varies. It offers a high temperature accuracy of ±1 °C from 0 °C to 70 °C with excellent transducer linearity. The digital output of the TMP05 is CMOS/TTL-compatible and, therefore, can be interfaced directly to PSoC. The digital output of the TMP06 is open-drain and requires a pull-up resistor for proper operation, which can be integrated inside PSoC. Throughout the rest of this document, any references to TMP05 also apply to the TMP06 sensor.

The **TMP05 Digital Temperature Sensor Interface** component makes it extremely easy for designers to develop thermal monitoring and management solutions using multiple TMP05 sensors. The component requires just one input and one output pin from PSoC, which enables the designer to implement many other system management functions in the same device.

The TMP05 Digital Temperature Sensor Interface component can be used in thermal management solutions for base stations, telecommunications, and in server and storage applications. Typical applications may include, but are not limited to, areas where remote temperature sensing, environmental control systems, computer thermal monitoring, thermal protection, industrial process control, and power-system monitoring and management are required.

Typical System Architecture

The TMP05 sensor has three modes of operation:

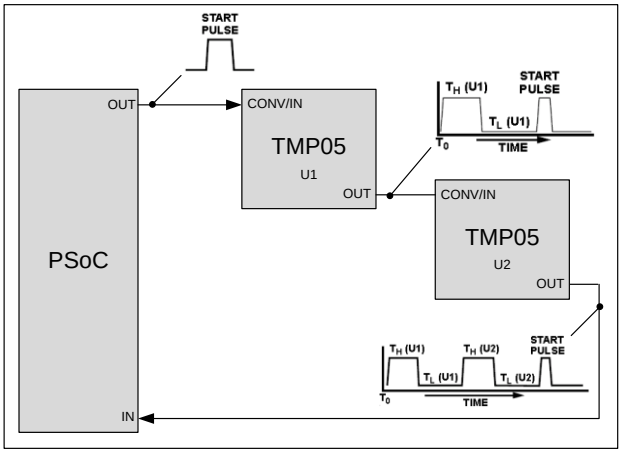
- Continuously converting mode
- Daisy-chained mode
- One-shot mode

A three-state function control input pin (FUNC) sampled at power-up determines the mode in which the device operates. Setting the FUNC pin to a high state allows multiple TMP05s to be connected in daisy-chained mode. In that mode, multiple TMP05 temperature sensors can be connected, which enables PSoC to read all the sensors through the same two-wire interface.

In this configuration, PSoC generates a “Start” pulse to begin a new temperature-to-PWM conversion cycle. The output of the first TMP05 sensor begins with its own PWM output, followed by a new Start pulse that it generated internally. The output of the second TMP05 sensor begins with the PWM output of the previous sensor, followed by its own PWM output and terminated by a new Start pulse that it generated internally. When more sensors are daisy-chained in this fashion, the final return signal to PSoC contains the PWM outputs from all sensors starting with the output of the first sensor, followed by the output from the second sensor and so on until the terminating Start pulse is detected. Temperature-to-PWM conversion then stops until PSoC generates another Start pulse. The system-level connection scheme and expected waveforms are shown in Figure 1.

PSoC is an ideal choice for interfacing to TMP05 sensors because it has the programmable universal digital block resources available to implement the required custom interface logic with minimal firmware. This application note focuses on how to use the TMP05 Digital Temperature Sensor Interface component with PSoC by working through some example projects on the CY8CKIT-001 PSoC Development Kit (DVK). The examples will show how to operate the component and monitor the temperature of multiple TMP05 devices reliably. Supporting other management protocols and higher-level functionality is beyond the scope of this application note.

Figure 1. Multiple TMP05 Sensors Daisy-Chained



Design Considerations when using TMP05 Sensors in Daisy-Chain Mode

Certain timing constraints must be observed when operating TMP05 sensors in daisy-chain mode. PSoC needs to generate the conversion start pulse on the CONV/IN pin of the first sensor. The start pulse lets the first TMP05 part know that it should now start a conversion and output its own temperature. The pulse width of the start pulse should be less than 25 µs but greater than 20 ns. These constraints are handled automatically by the component design and are mentioned in this application note for reference purposes only.

Figure 2 and Figure 3 show the input and output waveforms for the first sensor in the daisy-chain (taken from the TMP05 device datasheet):

Figure 2. TMP05 Start Pulse Waveform

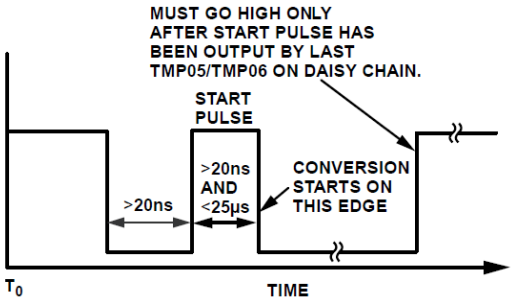
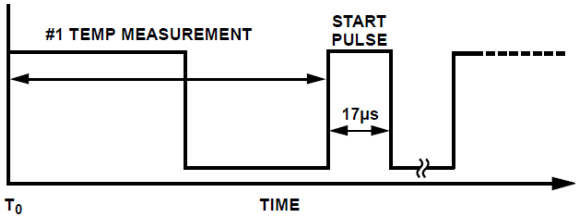


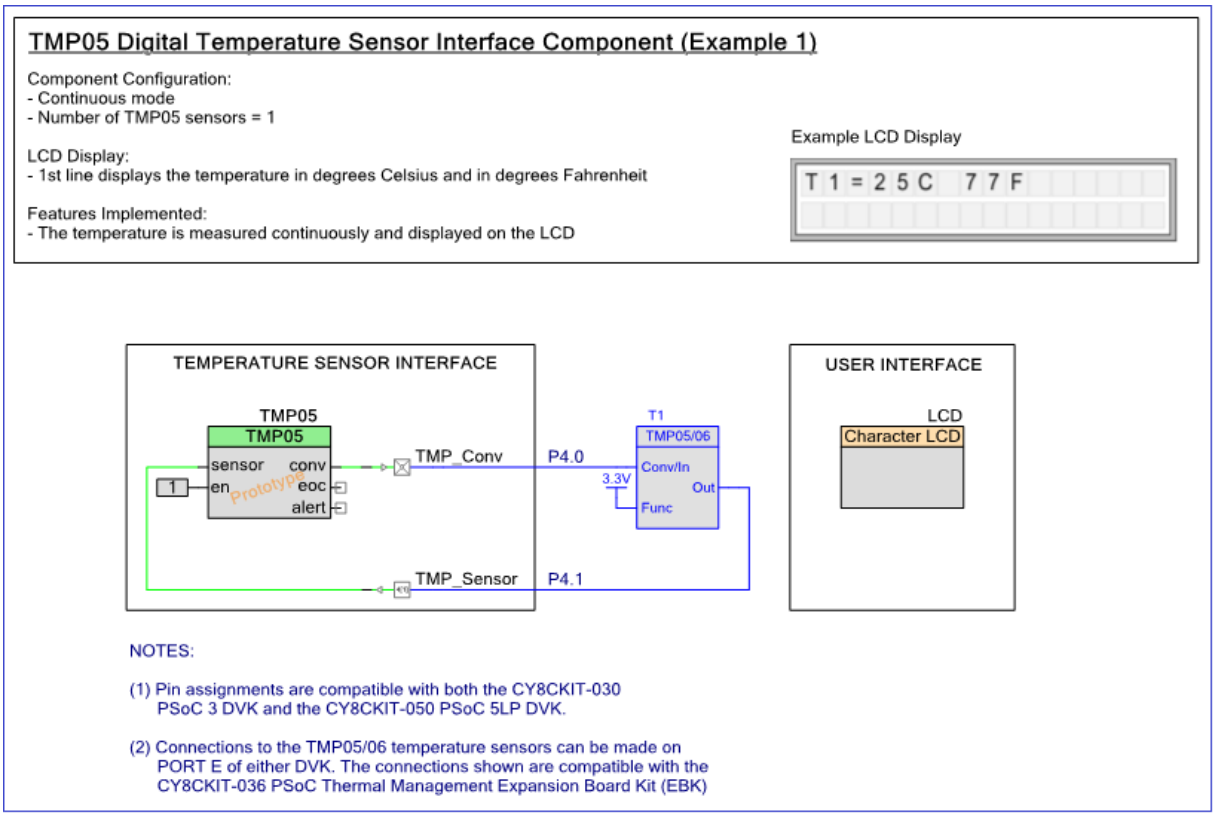
Figure 3. TMP05 PWM Output Waveform



Getting Started with the TMP05 Digital Temperature Sensor Interface Component

Distributed with this application note is a PSoC Creator™ bundled project ZIP file that contains four example design projects and a library project. The library project contains the **TMP05** Creator component.

Figure 4. Top-Level Schematic for TMP05 Example1



1. In the **Workspace Explorer** tab to the left of the screen, expand project **Example1** by clicking on the small '+' icon to the left of it.
2. Double-click on *TopDesign.cysch* to open the top-level design schematic, which shows the components used inside PSoC.

   To the right of the schematic is the PSoC Creator Component Catalog. The tab labeled **Cypress** contains the default library of Cypress-provided content. The Category labeled **Thermal Management** contains the TMP05 component. The provided library project can be re-used by importing into any PSoC Creator design project that requires the TMP05 interface component functionality.

Figure 4 shows the top-level design schematic for Example Project 1.

All of the example projects provided with this application note are designed to run on either the CY8CKIT-030 PSoC 3 Development Kit (DVK) or the CY8CKIT-050 PSoC 5LP DVK, supporting up to four TMP05 or TMP06 temperature sensors. The projects are easily modified to support more TMP05 or TMP06 sensors for further development and testing.

Each example project consists of monitoring the ambient temperature from a specific number of TMP05 sensors. To replicate the environment for each project properly, some wire connections are required on the PSoC DVK. The *Top Design* of each project in PSoC Creator illustrates a circuit shown in blue. You can also refer to Figure 5 for the example using two sensors in daisy-chain.

Double-click the **TMP05** component to open the component customizer as show in Figure 6.

Save the ZIP file to a convenient location on your hard drive and extract the contents to a local folder.

To get started with the example projects, double-click on the *AN65977.cywrk* PSoC Creator workspace file.

Figure 5. Wire Connections for two daisy-chained TMP05 sensors

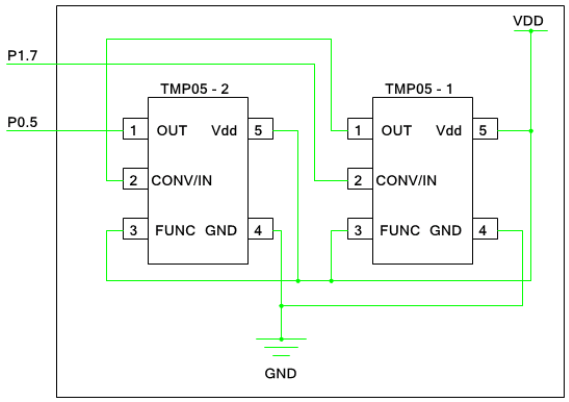
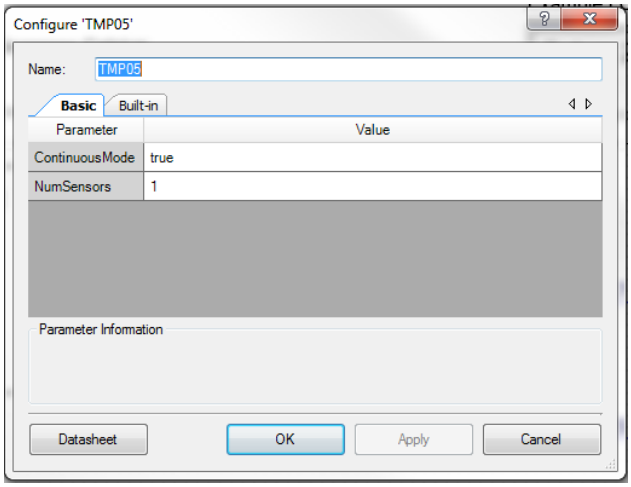
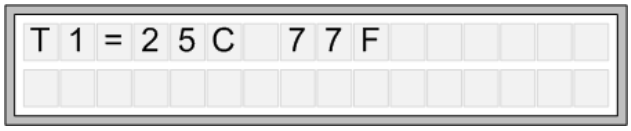


Figure 6. TMP05 Component Customizer



For the first example, the **ContinuousMode** parameter is set to **true** and the **NumSensors** parameter is set to **1**. This will enable continuous monitoring of one TMP05 sensor. Click **OK** to close the customizer.

Program the Example1 project into the DVK by selecting **Debug > Program** from the pull-down menus. If the project is running correctly, the text displayed on the debug LCD should look similar to the following picture.



Note that the temperature is displayed in both degrees Celsius and Fahrenheit.

Example 1 – Firmware Details

In the **Workspace Explorer**, double-click *main.c* to examine the firmware used in this example project. The code is shown in the

Code 1 excerpt. As shown, all lines of code that begin with the prefix **TMP05** reference APIs provided by Cypress for this component. Refer to the component datasheet for a full list of the provided APIs.

In Example1, PSoC is monitoring the temperature of one TMP05 sensor continuously. The monitored temperature is displayed on the LCD.

Congratulations! You have just completed your first **TMP05** temperature sensor interface design with PSoC.

Code 1. Example1 *main.c* Firmware code

```
#include <device.h>

void main()
{
        /* Enable global interrupts to the CPU core */
        CYGlobalIntEnable;

        /* Start the hardware components */
        LCD_Start();
        LCD_ClearDisplay();
        TMP05_Start();

        /* Trigger the first TMP05 sensor measurement */
        TMP05_Trigger();

        /* Display the temperature continuously */
        while (1)
        {
                /* Only display the temperature when end-of-conversion is detected */
                if (TMP05_ConversionStatus() & TMP05_EOC)
                {
                        LCD_Position(0,0);
```

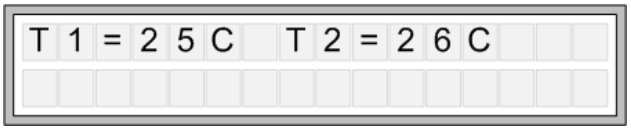
```
                    LCD_PrintString("T1=");
                    LCD_PrintNumber(TMP05_CalcTempC(0));
                    LCD_PrintString("C ");
                }
        }
}
```

Example 2 – Firmware Details

The firmware of Example2 also operates in continuous mode; however, it can sample one or two sensors depending on how the hardware is configured. You should set the number of sensors in the TMP05 component to be equal to the real number of sensors installed in the hardware. This example also supports daisy-chain error detection.

Close *main.c* and *TopDesign.cysch* for Example1. Go back to the Workspace Explorer and right-click on **Project 'Example2'**. Select **Set as Active Project.**

Open *TopDesign.cysch* for Example2 to start working on this project. Program the Example2 project into the DVK by selecting **Debug > Program** from the pull-down menus. If the project is running correctly, the text displayed on the debug LCD should look similar to the following picture.



If a sensing error is detected due to an open circuit anywhere in the daisy-chain, a message is displayed and sampling stops. In that case, the error message displayed on the LCD should look like the following picture:



To generate an error condition, you can disconnect one of the cables connected to P4.0 or P4.1.

The code for Example2 is shown in the Code 2 excerpt.

Code 2. Example2 *main.c* Firmware code

```
#include <device.h>

void displayTemp(void);              /* Display temperatures on the character LCD */
void displayErrorMsg(void);          /* Display error messages */

void main()
{
        uint8 status;

        /* Enable global interrupts to the CPU core */
        CYGlobalIntEnable;

        /* Start the hardware components */
        LCD_Start();
        LCD_ClearDisplay();
        TMP05_Start();

        /* Trigger the first TMP05 sensor measurement */
        TMP05_Trigger();

        /* Display the temperature continuously */
        while (1)
        {
                status = TMP05_ConversionStatus();

                /* Error detected, display error message */
                if (status & TMP05_ERR)
                {
                        displayErrorMsg();

                        /* Stop the component */
                        TMP05_Stop();
                        /* Stop here */
                        while (1);
                }
                /* Display the temperature(s) measured from the sensor(s) */
                else if (status & TMP05_EOC)
                {
```

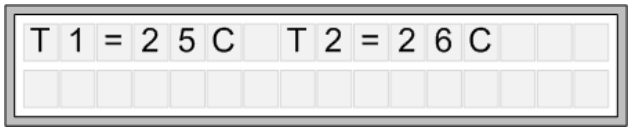
```
                        displayTemp();
                }
        }
}

/* Function to display temperatures on the character LCD */
void displayTemp(void)
{
        uint8 i;
        LCD_Position(0,0);

        /* Display the temperature(s) on the LCD */
        for (i = 0; i < TMP05_NUMSENSORS; i++)
        {
                /* Print the following string on LCD: T1=xxC */
                LCD_PrintString("T");
                LCD_PrintNumber(i+1);
                LCD_PrintString("=");
                LCD_PrintNumber(TMP05_CalcTempC(i));
                LCD_PrintString("C ");
        }
}

/* Function to display error messages */
void displayErrorMsg(void)
{
        LCD_ClearDisplay();
        LCD_Position(0,0);
        LCD_PrintString("Error Detected! ");
        LCD_Position(1,0);
        LCD_PrintString("Reset to Retry..");
}
```

Example 3 – Firmware Details

While the **TMP05** component operates in continuous mode in the first two examples, it may be useful to operate it in the one-shot mode to set a deterministic sensor sample rate or to save power. This example also supports error detection, however, unlike Example2, sampling is not suspended when an error is detected. Instead, the firmware will continuously retry sampling.

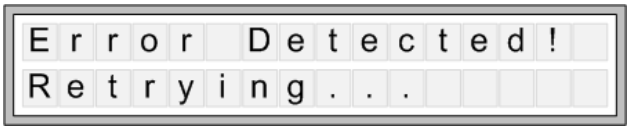
Close *main.c* and *TopDesign.cysch* for Example2. Go back to the Workspace Explorer and right-click on **Project 'Example3'**. Select **Set as Active Project.**

Open *TopDesign.cysch* for Example3 to start working on this project. Program the Example3 project into the DVK by selecting **Debug > Program** from the pull-down menus. If the project is running correctly, the text displayed on the debug LCD should look similar to the following picture:



In this example, a timer is added to control the temperature sensor sample rate. By default, the timer is configured to generate an interrupt every 1 second. This can be changed by changing the **period** parameter of the Timer component placed on the schematic sheet.

If an error is detected, a message is displayed and the firmware tries again to sample from the sensors. The error message displayed on the LCD should look like the following picture:



To generate an error condition, you can disconnect one of the wires connected to P4.0 or P4.1.

The code of Example3 is shown in the Code 3 excerpt.

Code 3. Example3 *main.c* Firmware code

```
#include <device.h>

void displayTemp(void);              /* Display temperatures on the character LCD */
void displayErrorMsg(void);          /* Display error messages */

static uint8 errorFlag;              /* Error condition flag */
static uint8 timerFlag;              /* Timer expired flag */
```

```
/* Interrupt handler for the error scenarios */
CY_ISR(AlertInterruptHandler)
{
        /* Set the error flag for the main loop to process */
        errorFlag = 1;
}

/* Interrupt handler for the temperature sensor sample rate timer */
CY_ISR(TimerInterruptHandler)
{
        /* Clear the timer registers */
        Timer_ReadStatusRegister();

        /* Set the timer flag for the main loop to process */
        timerFlag = 1;
}

void main()
{
        uint8 status;

        /* Clear flags */
        errorFlag = 0;
        timerFlag = 0;

        /* Enable global interrupts to the CPU core */
        CYGlobalIntEnable;

        /* Start the hardware components */
        LCD_Start();
        LCD_ClearDisplay();
        TMP05_Start();
        Timer_Start();

        /* Start and configure the Alert interrupt handler */
        AlertISR_Start();
        AlertISR_Disable();
        AlertISR_SetVector(AlertInterruptHandler);
        AlertISR_Enable();

        /* Start and configure the Timer interrupt handler */
        TimerISR_Start();
        TimerISR_Disable();
        TimerISR_SetVector(TimerInterruptHandler);
        TimerISR_Enable();

        /* Trigger the first TMP05 sensor measurement */
        TMP05_Trigger();

        while (1)
        {
                /* Error handling */
                if (errorFlag)
                {
                        errorFlag = 0;

                        /* Display error message */
                        displayErrorMsg();

                        /* Restart the component */
                        TMP05_Stop();
                        TMP05_Start();

                        /* Trigger again to re-try the measurements */
                        TMP05_Trigger();
                }

                /* Check for new temperature conversion result */
                else
                {
                        /* Read the status of the conversion */
                        status = TMP05_ConversionStatus();
```

```
                        if ((status & TMP05_EOC) && !(status & TMP05_ERR))
                        {
                                /* Display the temperature(s) measured from the sensor(s) */
                                displayTemp();
                        }
                }

                /* Timer expired, initiate a new measurement */
                if (timerFlag)
                {
                        timerFlag = 0;
                        TMP05_Trigger();
                }
        }
}

/* Function to display temperatures on the character LCD */
void displayTemp(void)
{
        uint8 i;

        LCD_ClearDisplay();
        LCD_Position(0,0);

        /* Display the temperature in a specific location on the LCD */
        for (i = 0; i < TMP05_NUMSENSORS; i++)
        {
                /* Print the following string on LCD: T1=xxC */
                LCD_PrintString("T");
                LCD_PrintNumber(i+1);
                LCD_PrintString("=");
                LCD_PrintNumber(TMP05_CalcTempC(i));
                LCD_PrintString("C ");
        }
}

/* Function to display an error message */
void displayErrorMsg(void)
{
        LCD_ClearDisplay();
        LCD_Position(0,0);
        LCD_PrintString("Error Detected! ");
        LCD_Position(1,0);
        LCD_PrintString("Retrying...     ");
}
```

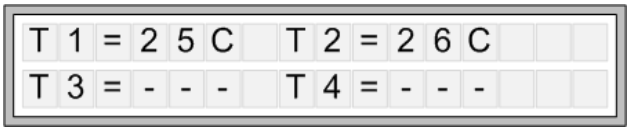
Example 4 – Firmware Details

In this example, the **TMP05** component is configured in one-shot mode. The main difference in this example is the addition of a discovery mechanism that automatically determines the number of sensors in the daisy-chain. This feature is useful in cases where the same PSoC design will be used across several different variations of the same product, but each one supports a different number of TMP05 sensors.

Close *main.c* and *TopDesign.cysch* for Example3. Go back to the Workspace Explorer and right-click on **Project 'Example4'**. Select **Set as Active Project.**

Open *TopDesign.cysch* for Example4 to start working on this project. Program the Example4 project into the DVK by selecting **Debug > Program** from the pull-down menus.

If the project is running correctly, the text displayed on the debug LCD should look similar to the following picture.



This example also uses a 1-second timer to control the temperature sensor sample rate.

If an error is detected, the temperature will be displayed as '- - -'. In the above LCD picture, the firmware only discovered two sensors in the daisy-chain, therefore only the measurements of the first two temperatures are displayed. You could manually change the number of sensors in the daisy-chain of your hardware and test the discovery feature.

The code of Example4 is shown in the Code 4 excerpt.

Code 4. Example4 *main.c* Firmware code

```
#include <device.h>

#define MAX_NUM_SENS   4              /* Maximum number of sensors in the daisy-chain */

void displayTemp(void);               /* Display temperatures on the character LCD */

static uint8 errorFlag;               /* Error condition flag */
static uint8 timerFlag;               /* Timer expired flag */
static uint8 sensorCount;            /* Number of TMP05 sensors connected to PSoC */

/* Interrupt handler for the error scenarios */
CY_ISR(AlertInterruptHandler)
{
       /* Set the error flag for the main loop to process */
       errorFlag = 1;
}

/* Interrupt handler for the temperature sensor sample rate timer */
CY_ISR(TimerInterruptHandler)
{
       /* Clear the timer registers */
       Timer_ReadStatusRegister();

       /* Set the timer flag for the main loop to process */
       timerFlag = 1;
}

void main()
{
       uint8 status;

       /* Clear flags */
       errorFlag = 0;
       timerFlag = 0;

       /* Enable global interrupts to the CPU core */
       CYGlobalIntEnable;

       /* Start the hardware components */
       LCD_Start();
       LCD_ClearDisplay();
       TMP05_Start();

       /* Start and configure the Alert interrupt handler */
       AlertISR_Start();
       AlertISR_Disable();
       AlertISR_SetVector(AlertInterruptHandler);
       AlertISR_Enable();

       /* Discover the number of TMP05 temperature sensors in the daisy-chain */
       sensorCount = TMP05_DiscoverSensors();
       errorFlag = 0;

       /* Trigger the first TMP05 sensor measurement */
       TMP05_Trigger();

       /* Start and configure the Timer interrupt handler */
       TimerISR_Start();
       TimerISR_Disable();
       TimerISR_SetVector(TimerInterruptHandler);
       TimerISR_Enable();
       Timer_Start();

       while (1)
       {
              /* Error handling */
              if (errorFlag)
              {
                     /* Restart the component */
                     TMP05_Stop();
                     TMP05_Start();
```

```
                        /* Clear the temperatures on LCD */
                        sensorCount = 0;
                        displayTemp();

                        /* Re-discover the number of sensors in the daisy-chain */
                        sensorCount = TMP05_DiscoverSensors();
                        errorFlag = 0;

                        /* Trigger again to re-try the measurements */
                        TMP05_Trigger();
                }

                /* Check for new temperature conversion result */
                else
                {
                        /* Read the status of the conversion */
                        status = TMP05_ConversionStatus();

                        if ((status & TMP05_EOC) && !(status & TMP05_ERR))
                        {
                                /* Display the temperature(s) measured from the sensor(s) */
                                displayTemp();
                        }
                }

                /* Timer expired, initiate a new measurement */
                if (timerFlag)
                {
                        timerFlag = 0;
                        TMP05_Trigger();
                }
        }
}

/* Function to display temperatures on the character LCD */
void displayTemp(void)
{
        uint8 i;

        LCD_ClearDisplay();
        LCD_Position(0,0);

        /* Display the temperature in a specific location on the LCD */
        for (i = 0; i < MAX_NUM_SENS; i++)
        {
                /* Print the following string on LCD: T1=xxC */
                LCD_PrintString("T");
                LCD_PrintNumber(i+1);
                LCD_PrintString("=");
                if (i < sensorCount)
                {
                        LCD_PrintNumber(TMP05_CalcTempC(i));
                        LCD_PrintString("C ");
                }
                else
                {
                        LCD_PrintString("--- ");
                }
                if (i == 1)
                {
                        /* Line break */
                        LCD_Position(1,0);
                }
        }
}
```

Future Projects

The PSoC Creator library project provided along with this application note can be shared freely with other designers and re-used in your own Creator design projects. To make use of this feature, copy the **ThermalLibrary.cylib** folder that was provided with the example projects to a convenient location on your hard drive or shared network.

When you begin your next TMP05 design project in Creator, the **TMP05 Interface Component** will not be automatically available to you. To gain access to the TMP05 Digital Temperature Sensor Interface Component, click on **Project > Dependencies** from the pull-down menus. Under **User Dependencies**, click on the folder icon and navigate to the location of the *ThermalLibrary.cyprj* file. Select the **Components** and **Code** checkboxes and click **OK**. The component will now be available in your design project and the **TMP05** component will be available to use in your new project.

If you would like to continue working with the TMP05 component, consider purchasing the PSoC Thermal Management Expansion Board Kit (CY8CKIT-036) shown in Figure 7.

Figure 7. CY8CKIT-036 PSoC Thermal Management Expansion Board Kit (EBK)



Conclusion

The TMP05 Digital Temperature Sensor Interface component enables designers to quickly and easily design thermal monitoring and management solutions providing support of multiple TMP05 sensors in daisy-chain applications.

The unique ability of the PSoC architecture to combine custom digital logic, analog signal chain processing, and an MCU in a single device enables system designers to integrate many external fixed-function ASSPs. This powerful integration capability not only reduces BOM cost, but also results in PCB board layouts that are less congested and more reliable.

Document History

Document Title: PSoC® 3 and PSoC 5LP - Creating an Interface to a TMP05/TMP06 Digital Temperature Sensor

Document Number: 001-65977

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3123189	AMKH	12/29/2010	New Spec – Initial Release
*A	3571727	RLOS	04/05/2012	Updated the document using the new template. Updated the example projects to use the TMP05 Component v1.20. Added support for PSoC 5.
*B	3819560	JK	11/22/2012	Updated the example projects to use the new TMP05 Component v1.30 revised to work with PSoC Creator v2.1 SP1 and PSoC 5LP. Updated the example project pinouts to work with the CY8CKIT-030 PSoC 3 Development Kit (DVK) and CY8CKIT-050 PSoC 5LP DVK.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

PSoC is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2010-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.
Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
Use may be limited by and subject to the applicable Cypress software license agreement.