

Project Name: Example_Temperature_LM75

Programming Language: C

Associated Part Families: CY8C29/27/24/22/21xxx, CY8C23x33, CY7C603xx, CY7C64215, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED04D01/02/03/04, CY8CTxx110, CY8CNP102

Software Version: PSoC[®] Designer™ 5.2

Related Hardware: CY3210 PSoC Eval1 Board

Author: Pushek Madaan

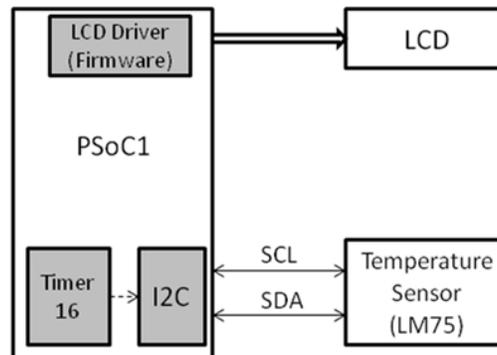
Objective

This Project demonstrates how to interface PSoC[®] 1 to an I²C temperature sensor (LM75) and display the temperature on the LCD.

Overview

This project uses the I2CHW Master user-module to retrieve ambient temperature from the LM75 sensor at regular intervals of time and display it on the LCD after required processing. Timer16 User Module is used to generate interrupt every one second, on which the data is read from the temperature sensor and displayed.

Block Diagram



User Module List and Placement

The following table lists user modules used in this project and the hardware resources occupied by each user module.

User Module	Placement
I2CHW	System Resource
Timer16	DBB00 and DBB01
LCD	Software Implementation

User Module Parameter Settings

The following tables show the user module parameter settings for each of the user modules used in the project.

I2CHW User Module		
Parameter	Value	Comments
Read_Buffer_Types	RAM ONLY	Only RAM data buffer is used.
CPU_Clk_speed_(CY8C27xA)	Not CY8C27xA	See the Notes section at the end of this table.
I ² C Clock	100 K Standard	Sets the I ² C clock as 100 kHz.
I ² C Pin	P1[5]-P1[7]	Selects P1[5] and P1[7] for I ² C communication. P1[5] is SDA and P1[7] is SCL.

Notes

- When the Read_Buffer_Types is set to RAM ONLY, only RAM buffers are transmitted over I²C. To read and transmit data from Flash, set the read buffer type to RAM or FLASH.
- The parameter CPU_Clk_speed is provided as a workaround for a silicon issue that was present in CY8C27x43A (Silicon Rev. A) family of devices. In this family, read or write to the I2C_CFG and I2C_SCR registers occurred with CPU speed less than 6 MHz. If the CPU speed is greater than 6 MHz, it is throttled down to 6 MHz when accessing the I²C registers and restored after the access. This workaround is not required for families other than the CY8C27x43A family.
- The I²C Clock parameter is dependent on the SysClk. The I²C clock setting in the user module is based on a SysClk of 24 MHz. In devices which support slower Sysclk, the I²C clock is reduced by the same proportion. For example, if I²C clock is set to 400 kHz and SysClk is set to 6 MHz, the actual I²C clock is only 100 kHz.

LCD User Module		
Parameter	Value	Comments
LCDPort	Port 2	Use Port 2 to connect LCD.
Bargraph	Disable	Disable the Bargraph feature.

Timer16 User Module		
Parameter	Value	Comments
Clock	VC3	Use the clock for the module as VC3 (10 kHz).
Capture	HIGH	Disable the software capture feature of the timer.
TerminalCountOut	None	Disable the terminal count output.
CompareOut	None	Disable the compare output.
Period	9999	Divide the source clock by 10000 to generate a 1 Hz signal.
CompareValue	5000	Set the compare value for comparing with timer counts.
CompareType	Less Than Or Equal	Sets the logical operation for the comparison.
InterruptType	Terminal Count	Trigger interrupt on terminal count.
ClockSync	Sync to SysClk	Synchronize the clock with SysClk.
TC_PulseWidth	Full Clock	Terminal count should stay HIGH for full clock, not used in this example
Invert Capture	Normal	Make the capture input as Active High, not used in this example

Note

- For more details regarding User Module parameters, please refer to UM datasheet which can be located from Start → All Programs → Cypress → PSoC Designer 5.2 → Documentation → User Module Datasheet → STDUM.

Global Resources

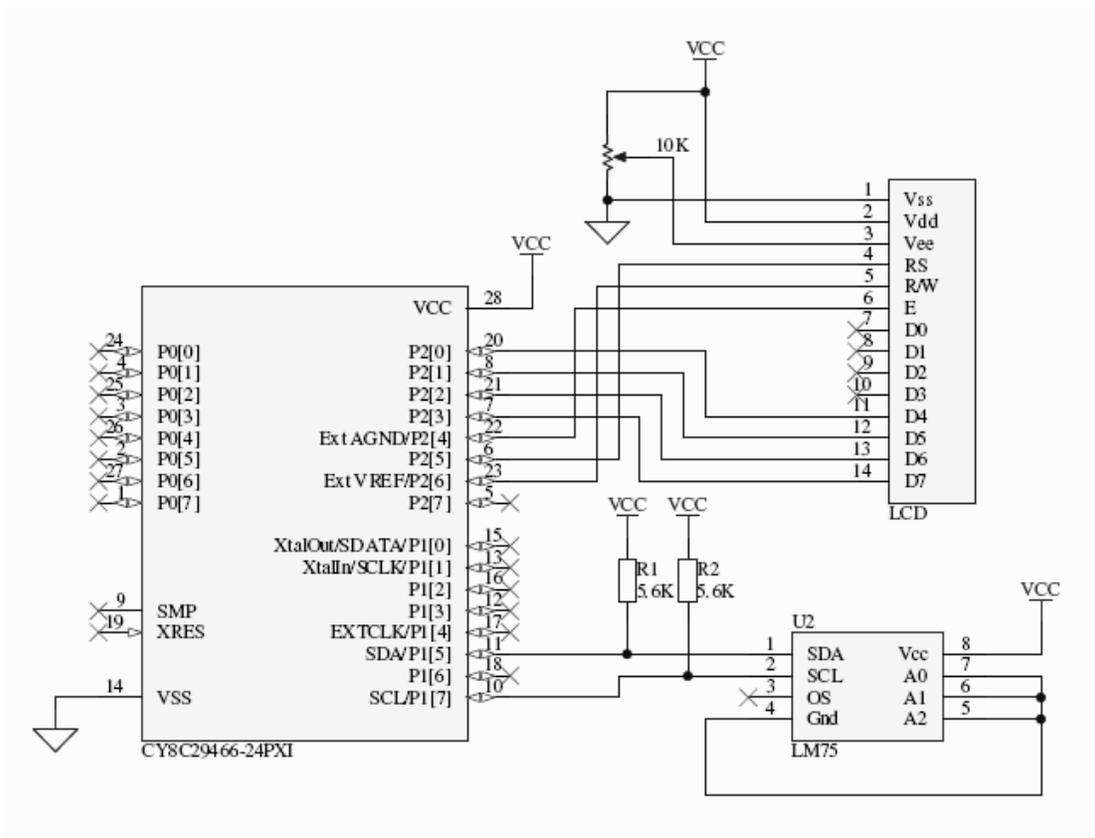
Important Global Resources		
Parameter	Value	Comments
Power Setting [Vcc / SysClk freq]	5.0 V/24 MHz	Selects 5 V operation and 24 MHz SysClk.
CPU_Clock	SysClk/2	Selects 12 MHz as the clock input for the CPU.
VC1 = SysClk/n	10	VC1 output set to 2.4 MHz.
VC3 Source	VC1	Set VC1 as the clock source for VC3.
VC3 Divider	240	Divides VC1 by 240 and generates a 10 kHz output.

Note

- Other parameters are left at their default value.

Hardware Connections

The schematic diagram for the project follows.



U2 (LM75) is a digital temperature sensor, which supports I2C protocol. This device has an integrated Sigma-Delta analog to digital converter and I2C interface. It provides 9-bit digital temperature reading with an accuracy of $\pm 2^\circ\text{C}$. Pins A0, A1, and A2 are used to connect multiple LM75 to a single I²C bus and to hard wire the three least significant bits (LSB) of the device address. In this example, all these lines are tied to GND. R1 and R2 are external pull up resistors as the I²C bus operates in open drain mode. This schematic can be wired using the bread board area of the CY3210 PSoC Eval1 board.

Operation

On reset, all hardware settings from the device configuration are loaded into the device and *main.c* is executed.

The following operations are performed by the firmware.

- Global interrupt is enabled.
- The I2CHW module is configured as Master and its interrupt is enabled.
- OneSecTimer is started and its interrupt is enabled.
- LCD is initialized and welcome message is displayed on Row 0.
- An infinite loop is entered where the following operations are performed:
 - Check if the `bReadTempFlag` is set. This flag is set every second inside the OneSecTimer ISR
 - If `bReadTempFlag` is set:
 1. Clear the flag.
 2. Read the temperature from the LM75 into variable `iTemp` by calling function `ReadTemp`. On power up, the LM75 internally sets its read pointer to the temperature register. Reading directly from LM75 returns the value from the temperature register. `I2CHW_fReadBytes` function is used to read the temperature from LM75.
 3. Convert the temperature into a floating point value and display on LCD. LM75 stores the temperature in bits D7 to D15 in the temperature register. The least significant 7 bits are “Don’t Care” bits. After reading the 16-bit temperature from LM75, the 16-bit value is shifted right by 7 bits to move the 9-bit temperature value to bits D0 to D8. The LSB represents 0.5 °C. Therefore, the 9-bit temperature is multiplied by 0.5 to get the actual temperature. This is converted to ASCII using `ftoa` and printed to LCD using the `LCD_PrString` function.

Timer ISR

The ISR for OneSecTimer is written in C. The ISR function is named as `OneSecTimer_ISR` and is declared as an ISR using the following code in the beginning of *main.c*.

```
#pragma interrupt_handler OneSecTimer_ISR
```

If the function name is identical to the name of the assembly ISR found in *OneSecTimerINT.asm* file in PSoC Designer™, then on interrupt, the control is automatically transferred to the C function (similar methodology is used in this code example). If the function name is different from the default name, then an `ljmp` instruction should be placed either in *boot.tpl* or inside the *OneSecTimerINT.asm* file to the C ISR.

Another advantage of using the identical name of the assembly ISR is user does not have to worry about the PSoC Designer upgrade, as the control will never be transferred to *OneSecTimerINT.asm* file.

For more details about writing an ISR in C, refer to the following Knowledge Base article on the Cypress website: [Implementing an Interrupt Service Routine in C on the PSoC](#).

PSoC is a registered trademark of Cypress Semiconductor Corp. PSoC Designer is a 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
<http://www.cypress.com/>

© Cypress Semiconductor Corporation, 2009-2011. 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.