Lab 5: I^{2}C

Instructor's Guide

Lab Introduction

This lab introduces a method for digital communication between a microprocessor and peripherals called I^{2}C. There are several other peripheral communication methods available, but I^{2}C is one of the simpler methods. The final project may use this or another communication method, depending on what the student wants to do.

Preparation

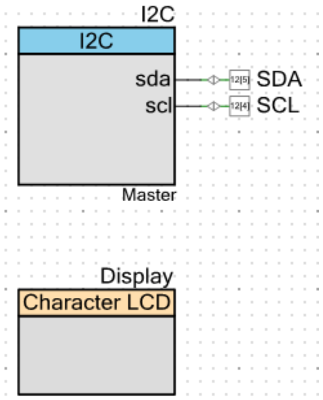
This lab requires an I^{2}C peripheral on a breakout board so it can be easily used in the prototyping area. This lab was written to use the MCP9808 I^{2}C temperature sensor breakout from Adafruit (http://www.adafruit.com/product/1782), but another I^{2}C device could be used if necessary. However, the lab would need to be rewritten to change the device being used.

This breakout requires some assembly: the header pins needed to mount it on a breadboard need to be soldered on. Because soldering is required, it is suggested that the instructor purchases these and solders them before the lab since many Computer Science students likely do not know how to solder. These breakouts are also approximately $5 each, so they are inexpensive.

Instructor Review

IMPORTANT: Ensure that the temperature sensor is wired up exactly the way that the lab indicates it should **before the student powers on the PSoC**. In particular, if the power and ground lines are reversed, the temperature sensor **could be destroyed**.

Here is what the schematic should look like:



Here is a sample firmware solution:

```
#include <project.h>
#include "stdio.h"

asm (".global _printf_float");

#define FALSE 0
#define TRUE 1

//I2C address for the temperature sensor
#define MCP9808_ADDR 0x18

//The expected value of the manufacturer ID register
#define MCP9808_MANUF_ID 0x0054

//The expected value of the upper 8 bits of the device ID register
#define MCP9808_DEVICE_ID_UPPER 0x04

//The addresses for all the registers on the temperature sensor
//  that are read-only
#define MCP9808_REG_UPPER_TEMP         0x02
#define MCP9808_REG_LOWER_TEMP         0x03
#define MCP9808_REG_CRIT_TEMP          0x04
#define MCP9808_REG_AMBIENT_TEMP       0x05
#define MCP9808_REG_MANUF_ID           0x06
#define MCP9808_REG_DEVICE_ID          0x07


//Utility function that reads a 16-bit register from the given device
uint16 I2C_Read16(uint8 addr, uint8 reg)
{
    uint16 ret;

    //Writes the register address to the device
    if(I2C_MasterSendStart(addr, I2C_WRITE_XFER_MODE) != I2C_MSTR_NO_ERROR)
    {
        return 0;
    }
    I2C_MasterWriteByte(reg);

    //Reads the 16-bit register from the device and combine it into a 16-bit
    //  value
    I2C_MasterSendRestart(addr, I2C_READ_XFER_MODE);
    ret = (uint16)I2C_MasterReadByte(I2C_ACK_DATA) << 8;
    ret |= (uint16)I2C_MasterReadByte(I2C_NAK_DATA);

    I2C_MasterSendStop();
    return ret;
}


int main()
{
    //Create and initialize global variables
    uint8 I2C_fail = FALSE;
    int16 read;
    char tstr[16];
    //Initialize I2C and Display
    I2C_Start();
    Display_Start();

    CyGlobalIntEnable;

    //Check to make sure the device is connected on the bus
    //  by checking the manufacturer and device IDs
    read = I2C_Read16(MCP9808_ADDR, MCP9808_REG_MANUF_ID);
    if(read != MCP9808_MANUF_ID)
    {
        I2C_fail = TRUE;
    }
    else
    {
        read = I2C_Read16(MCP9808_ADDR, MCP9808_REG_DEVICE_ID);
        read >>= 8;
        if(read != MCP9808_DEVICE_ID_UPPER)
        {
            I2C_fail = TRUE;
        }
    }

    //Print out an error or an info message
    if(I2C_fail)
    {
        Display_PrintString("Temp Sensor N/C!");
    }
    else
    {
        Display_PrintString("Temperature DegF");
    }

    for(;;)
    {
        //If it seems the device is connected, continually:
        //  read the temperature from the sensor
        //  convert it to degC then degF
        //  print it to the display
        //  sleep 250 ms (time it takes for a new reading)
        if(!I2C_fail)
        {
            read = I2C_Read16(MCP9808_ADDR, MCP9808_REG_AMBIENT_TEMP);

            //Reading is a signed, two's complement 13-bit number.
            //  Extend it to 16-bit
            read <<= 3;
            read >>= 3;

            //Convert to degrees Celsius. Convert from scale 1 = 1/16
            float temperature = read * 0.0625;

            //Convert to degF
            temperature = temperature * 9.0 / 5.0 + 32;

            //Format and print to the display
            sprintf(tstr, "%1.4f  ", temperature);
            Display_Position(1, 0);
            Display_PrintString(tstr);

            //Wait 250ms for the next reading
            CyDelay(250);
        }
    }
}
```