Alex Elkman
Hardik Patel
EEC 172 - A02
Spring 2016

Lab 1 - Serial Interfacing using SPI and I2C

Objectives:

The purpose of this lab was to use SPI (Serial Peripheral Interface) bus to interface the CC3200 LaunchPad to a color OLED (Organic Light-Emitting Diode) display, and to use the I2C (Inter-Integrated Circuit) bus to communicate with the on-board accelerometer sensor. Using this sensor our objective was to move a sphere around the OLED display. A Saleae logic probe was also used to capture the SPI and I2C waveforms .

Design and Test Procedures:

1-1: In this section of the lab we implemented the SPI demo demo project provided by TI on the two LaunchPads. The only changes made to the project were the setting of the master module and the slave module with the MASTER_MODE as 1 and 0 respectively. The two projects were saved and flashed onto the on-board non-volatile memory using TI Uniflash software. The boards were then connected to the computer and a tera-term window was opened for each pad. The project was then run by typing onto the master tera-term and it propagating through the SPI channels to be seen on the the slave tera-term.

Problems: The main problem we encountered in this section was the fact that the WIndows system in the lab seemingly could not manage both simultaneous UART connections whilst the boards were initiated. The solution we found was using a laptop for one of the connections.

1-2: In this section a video demonstration was transmitted via SPI from the CC3200 board to the Adafruit OLED interface. As a starting point, we made use of TI's spi_demo project and the Adafruit and test files provided to us. We initialized the SPI connection by using the PinMux utility to set the correct GPIO ports on the CC3200. All of the included starter project files were used to initiate the Adafruit OLED display. We implemented the writeData(), writeCommand() functions in the Adafruit_OLED.c file. We added test() which used the premade demo functions testlines(), testfillrects() etc. in the test.c file and all the headers included in main. The program was run successfully.

Problems: The main problem for us was understanding the order of initializing the board, and all the SPI functions that are needed prior to calling the Adafruit initializer function. If any one of them is in the wrong order, it may prevent the OLED from initializing properly because the SPI connection will not get properly established.

1-3: Here we used the logic analyzer to study the SPI waveforms generated by the GPIOs. This was achieved in much the same way we used the analyzer from Lab 0, by connecting the ground connection to the grey cable on the analyzer, and then adding the SPI analyzer feature in the Windows GUI (Graphical User Interface). The waveform we saw was indicative of the ASCII characters being sent over the connection, which can be interpreted by checking the bit values at each rising clock edge. In this case it is the bit sequence “01101100” or ACII “1”.

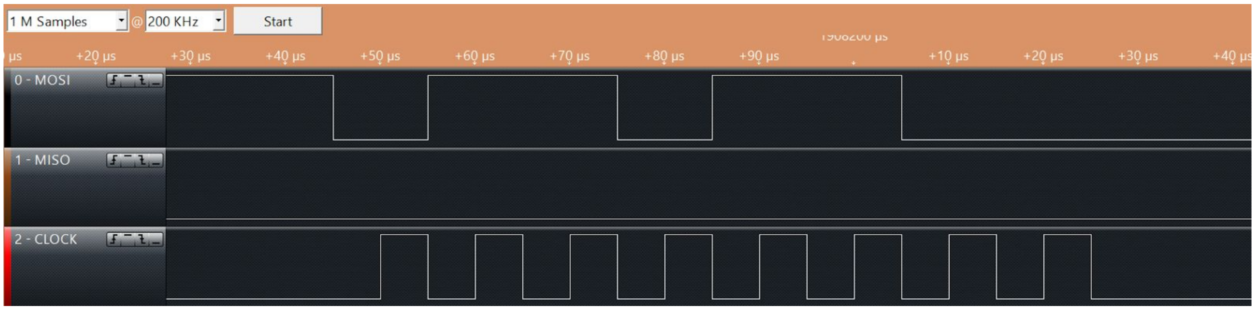

Figure 1: Wave analyzer of SPI_demo showing MOSI and SPI_Clock signals

2-1: For this subpart we use the i2c_demo project provided to read the accelerometer data on tera term, using the provided commands. Setting up the project was simple since it was already done for us.

2-2: Here we analyzed the waveforms generated by the I2c ports (SCL and SDA) from the former part. This was a command used was “readreg 0x18 0x5 1”, and the corresponding I2C signal reflects this. We can interpret the signal in the same method we did with the SPI and SPI_Clock signals, except now we have the first part of the signal corresponding to the “readreg” command, the second corresponding to the hex address “0x18”, the third looking at the register “0x5” for the Y-axis accelerometer data, and “1” for only 1 transmission to read.

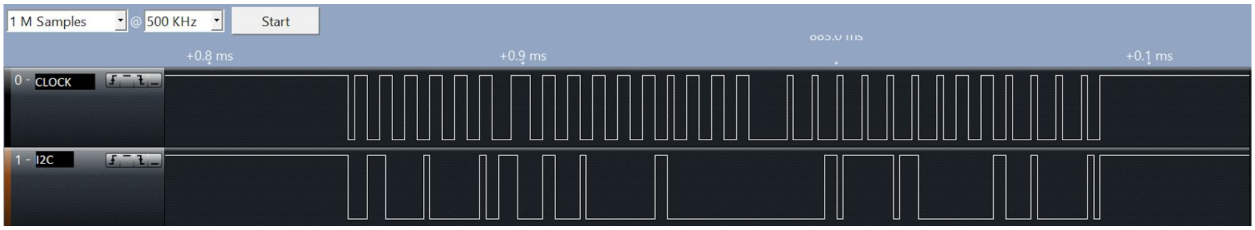

FIgure 2: Wave analyzer of the I2C_demo showing I2C and I2C_Clock signals

2-3: For this application program, we used the accelerometer data from above to move a ball drawn on the OLED display. This was achieved by combining the code from the SPI Adafruit demo to initialize the display and SPI connections, and also the I2C demo discussed above. In an infinite while(1) loop, we simply check the accelerometer data for both the X and Y axes, and pass this info into the create_ball() function from the Adafruit library. We then clear the image after each ball is generated, and the result is a pretty cool visual trick that makes the ball move as you tilt the screen.

Problems: We had some issues modifying the parseReadReg() function because we needed to modify it to check the 0x5 and 0x3 registers for the X and Y axis data. We then passed this data into global variables which were then used by the create_ball() function.

Conclusions:

We applied our new knowledge of I2C and SPI buses for different applications in this lab to make some interesting things happen. We tested using the Saleae Logic Analyzer to heighten our understanding of the material and gain a deeper knowledge of how these embedded system signals really work.

SPI code

```
void writeCommand(unsigned char c) {
//TODO 1
/* Write a function to send a command byte c to the OLED via
*  SPI.
*/
        while(!SPI_INT_TX_EMPTY){}  //Waiting till the buffer is empty
        unsigned long a; //dummy variable
        GPIOPinWrite(GPIOA0_BASE, 0x80, 0); // Sets the DC value to 0
        MAP_SPIDataPut(GSPI_BASE,c); // Puts the data into the base
        MAP_SPIDataGet(GSPI_BASE, &a);; // does a dummy read
        while(!SPI_INT_TX_EMPTY){} //Waiting till the buffer is empty
}
//*******************************************************
void writeData(unsigned char c) {
//TODO 2
/* Write a function to send a data byte c to the OLED via
*  SPI.
*/
        unsigned long a;
        GPIOPinWrite(GPIOA0_BASE, 0x80, 0xFF); //sets DC value to 1
        MAP_SPIDataPut(GSPI_BASE,c);
        MAP_SPIDataGet(GSPI_BASE, &a);
}
///********************************************************
void test() { //calls different functions separated by delays
        fillScreen(BLACK);
        char* hello = "Hello World!";
        delay(9999999);
        fillScreen(BLACK);
        Outstr ((char*)font);
        delay(9999999);
        fillScreen(BLACK);
        Outstr(hello);
        delay(9999999);
        fillScreen(BLACK);
        drawFastVLine(0, 0, 50, BLUE);
        drawFastVLine(5, 0, 50, GREEN);
        drawFastVLine(10, 0, 50, RED);
        drawFastVLine(15, 0, 50, YELLOW);
        drawFastVLine(20, 0, 50, BLACK);
        drawFastVLine(25, 0, 50, MAGENTA);
        drawFastVLine(30, 0, 50, WHITE);
        drawFastVLine(35, 0, 50, CYAN);
        delay(9999999);
        fillScreen(BLACK);
        drawFastHLine(0, 5, 50, BLUE);
        drawFastHLine(0, 10, 50, GREEN);
        drawFastHLine(0, 15, 50, RED);
        drawFastHLine(0, 20, 50, YELLOW);
        drawFastHLine(0, 25, 50, BLACK);
        drawFastHLine(0, 30, 50, MAGENTA);
        drawFastHLine(0, 35, 50, WHITE);
        drawFastHLine(0, 40, 50, CYAN);
        delay(99999999);
        fillScreen(BLACK);
        testlines(BLUE);
        delay(100000);
        testfastlines(BLACK, GREEN);
        delay(100000);
        testdrawrects(RED);
        delay(100000);
        testfillrects(RED, GREEN);
        delay(100000);
        testroundrects();
        delay(100000);
        testtriangles();
        fillScreen(BLACK);
}
//*****************************************************************
void main()
{

  BoardInit(); //Initializes the board
  PinMuxConfig(); //initializes the pinmuxing
  MAP_PRCMPeripheralClkEnable(PRCM_GSPI,PRCM_RUN_MODE_CLK);
  InitTerm();
  ClearTerm();
```

```
    MAP_PRCMPeripheralReset(PRCM_GSPI);
    MAP_SPIReset(GSPI_BASE); //resets SPI
    //MAP_SPIFIFOEnable(GSPI_BASE, SPI_TX_FIFO | SPI_RX_FIFO); //Enables fifo
    MAP_SPIConfigSetExpClk(GSPI_BASE,MAP_PRCMPeripheralClockGet(PRCM_GSPI),
                SPI_IF_BIT_RATE,SPI_MODE_SLAVE,SPI_SUB_MODE_0,
                (SPI_HW_CTRL_CS |
                SPI_4PIN_MODE |
                SPI_TURBO_OFF |
                SPI_CS_ACTIVELOW |
                SPI_WL_8));
    MAP_SPIEnable(GSPI_BASE); //enables SPI
    MAP_SPICSEnable(GSPI_BASE); // Enables chip select
    //MAP_PRCMPeripheralClkEnable(PRCM_GSPI,PRCM_RUN_MODE_CLK);
    Adafruit_Init(); // Initializes the OLED display
    test();
    MAP_SPICSDisable(GSPI_BASE);
}
//**********************************************************************
```

I2C code

```
#include <string.h>
#include <stdlib.h>
#include "hw_types.h"
#include "hw_memmap.h"
#include "hw_common_reg.h"
#include "hw_ints.h"
#include "spi.h"
#include "rom.h"
#include "gpio.h"
#include "rom_map.h"
#include "utils.h"
#include "prcm.h"
#include "uart.h"
#include "i2c_if.h"
#include "interrupt.h"
#include "uart_if.h"
#include "Adafruit_GFX.h"
#include "Adafruit_SSD1351.h"
```

```
#include "glcdfont.h"
#include "pin_mux_config.h"
extern int cursor_x;
extern int cursor_y;
float p = 3.1415926;
//Global variables to read the accelerometer data
char xarray[] = "readreg 0x18 0x3 1";
char yarray[] = "readreg 0x18 0x5 1";
int xaxis, yaxis;
//Global variables end
#define APPLICATION_VERSION     "1.1.1"
// MASTER_MODE = 1 : Application in master mode
// MASTER_MODE = 0 : Application in slave mode
#define MASTER_MODE      1
#define SPI_IF_BIT_RATE  100000
#define TR_BUFF_SIZE     100
#define MASTER_MSG       "This is CC3200 SPI Master Application\n\r"
#define SLAVE_MSG        "This is CC3200 SPI Slave Application\n\r"
#define APPLICATION_VERSION     "1.1.1"
#define APP_NAME                "I2C Demo"
#define UART_PRINT              Report
#define FOREVER                 1
#define CONSOLE                 UARTA0_BASE
#define FAILURE                 -1
#define SUCCESS                 0
#define RETERR_IF_TRUE(condition) {if(condition) return FAILURE;}
#define RET_IF_ERR(Func)          {int iRetVal = (Func); \
                                 if (SUCCESS != iRetVal) \
                                  return  iRetVal;}
#if defined(ccs)
extern void (* const g_pfnVectors[])(void);
#endif
#if defined(ewarm)
extern uVectorEntry __vector_table;
#endif
static void BoardInit(void)
{
/* In case of TI-RTOS vector table is initialize by OS itself */
#ifndef USE_TIRTOS
```

```
 //
 // Set vector table base
 //
#if defined(ccs)
  MAP_IntVTableBaseSet((unsigned long)&g_pfnVectors[0]);
#endif
#if defined(ewarm)
  MAP_IntVTableBaseSet((unsigned long)&__vector_table);
#endif
#endif
  //
  // Enable Processor
  //
  MAP_IntMasterEnable();
  MAP_IntEnable(FAULT_SYSTICK);

  PRCMCC3200MCUInit();
}

void delay(unsigned long ulCount){
        int i;
 do{
  ulCount--;
              for (i=0; i< 65535; i++) ;
        }while(ulCount);
}

int ProcessReadRegCommand(char *pcInpString) //Takes in the global variables (arrays as
{  //parameters) and parses them to get the desired data from the buffer
  unsigned char ucDevAddr, ucRegOffset, ucRdLen;
  unsigned char aucRdDataBuf[256];
  char *pcErrPtr;
  char *temp = pcInpString; // variable holds the input parameter
  //
  // Get the device address
  //
  pcInpString = strtok(NULL, " ");
  //RETERR_IF_TRUE(pcInpString == NULL);
 //  ucDevAddr = (unsigned char)strtoul(pcInpString+2, &pcErrPtr, 16);
```

```
    //
    // Get the register offset address
    //
    pcInpString = strtok(NULL, " ");
   // RETERR_IF_TRUE(pcInpString == NULL);
    ucRegOffset = (unsigned char)strtoul(pcInpString+2, &pcErrPtr, 16);
   // temp = ucDevAddr+ucRegOffset;
    //
    // Get the length of data to be read
    //
    pcInpString = strtok(NULL, " ");
   // RETERR_IF_TRUE(pcInpString == NULL);
    ucRdLen = (unsigned char)strtoul(pcInpString, &pcErrPtr, 10);
    //RETERR_IF_TRUE(ucLen > sizeof(aucDataBuf));
   if(xarray[15] == temp[15]){ // compares the particular bit for a 3 pr a 5
            ucRegOffset = 0x5;} //i.e xaxis of yaxis and assigns proper offset
   if(yarray[15] == temp[15]){
            ucRegOffset = 0x3;}
            ucDevAddr = 0x18; //Hard codes the Base address as 0x18
    I2C_IF_Write(ucDevAddr,&ucRegOffset,1,0);
    I2C_IF_Read(ucDevAddr, &aucRdDataBuf[0], 1); //puts the correct accelorometer data in
the
    if(ucRegOffset == 0x5)                              //in the appropriate global variable
          xaxis = (int)aucRdDataBuf[0];
    if(ucRegOffset == 0x3)
          yaxis = (int)aucRdDataBuf[0];
}
void main()
{
    // Initialize Board configurations
    BoardInit();
    // Muxing UART and SPI lines.
    PinMuxConfig();
    // Enable the SPI module clock
    MAP_PRCMPeripheralClkEnable(PRCM_GSPI,PRCM_RUN_MODE_CLK);
    // Reset the peripheral
    MAP_PRCMPeripheralReset(PRCM_GSPI);
     // Configure SPI interface
     MAP_SPIConfigSetExpClk(GSPI_BASE,MAP_PRCMPeripheralClockGet(PRCM_GSPI),
                SPI_IF_BIT_RATE,SPI_MODE_MASTER,SPI_SUB_MODE_3,
                (SPI_SW_CTRL_CS |
                SPI_4PIN_MODE |
                SPI_TURBO_OFF |
                SPI_CS_ACTIVELOW |
                SPI_WL_8));
        // Enable SPI for communication
        MAP_SPIEnable(GSPI_BASE);
        MAP_SPICSEnable(GSPI_BASE);
        I2CMasterEnable(I2CA0_BASE); // Initializing the I2Cs
        I2CSlaveEnable(I2CA0_BASE);
        Adafruit_Init();
        fillScreen(BLACK); // initializes the screen to black
        int cx = 64, cy = 64; // to start the ball in the center
        drawCircle(cx, cy, 4, BLACK);
        int x2, y2, x1, y1, x3, y3;
        while(1)
        {
                ProcessReadRegCommand(&xarray); // Uses the function to get the data from
                ProcessReadRegCommand(&yarray);                          //sensor
                //printf("%d %d\n", xaxis, yaxis); // to check the correct data
                        // if the ball goes out of bounds, brings it back inside bounds
                        if (xaxis > 64)
                                xaxis = (xaxis - 255)/4;
                        if (yaxis > 64)
                                yaxis = (yaxis - 255)/4;
                        x1 = cx; // Assigns old position values to temporary variables
                        y1 = cy;
                        drawCircle(x1, y1, 4, BLACK); //draws a circle at the old value on
                        // OLED to match with the background

// These if statements maintain the radius of the ball on one side for both x and y directions
                        if (cx + xaxis < 4 )
                                cx = 4;
                        else if (cx + xaxis >123)
                                cx = 123;
                        else
                                cx = ((cx + xaxis) % 128);
                        if (cy + yaxis <4)
```

```
                    cy = 4;
                else if (cy + yaxis > 123)
                    cy = 123;
                else
                    cy = ((cy + yaxis)) % 128;
                drawCircle(cx, cy, 4, BLUE); // Draws a circle at the new position
   }

}
```