|  |  |
| --- | --- |
|  | Laboratory Assignment Number 3 for ME 218a  Pre-Lab Due by 5:00pm on October 19, 2017 Due by 5:00pm on October 26, 2017 |

Part 0: Pre-Lab

Background:

The pre-lab should be completed after you have read through the entire lab assignment, but before you go into the lab to begin your work there. Completing the pre-lab, which can be done without any special tools or resources, will allow you to be most efficient with your time in the lab. When submitting your lab report, be sure to include a section for your answers to the pre-lab, even if you found no need to adjust them after the submission of the pre-lab.

Report Format Reminder:

1) Use cut and paste to move your schematics into your reports. Do not use screen captures! Limit the size of the area being transferred so that the resulting images fit on the page in the report.  
2) When copying your code into your report, please use the Highlight program to prepare an RTF file using the “bright” color theme. You can then open the RTF file in Word and copy and paste the code into your report and it will maintain the color highlighting.  
3) Be sure to clearly label the question that you are answering (include the question number & quote the question) along with your response.

Assignment:

|  |  |  |
| --- | --- | --- |
| ❑ | 0.1) | Complete the design requested in Part 1.1. |
| ❑ | 0.2) | What lines of C code will you use to do the initialization of Port B in Part 2.2? |
| ❑ | 0.3) | What sequence of C code will you use to implement the toggling in Part 2.4? |
| ❑ | 0.4) | Complete the design requested in Part 2.6. |
| ❑ | 0.5) | Write Pseudo-Code for all of the functions that you will use for Part 3. |

In the report:

Include your answers to the pre-lab section (corrected, if necessary) as part of your final lab report document.

Part 1: Your First Embedded C program and building some debugging hardware

Assignment:

|  |  |  |
| --- | --- | --- |
| ❑ | 1.1) | Work your way through the μVision tutorial. This will be your first Embedded C program. |
|  | | |
| ❑ | 1.4) | Design a circuit using the 74HCT14 to drive a visible red LED. Design for approximately 4mA of current through the LED. Configure the LED such that when the input to the 74HCT14 is high, the LED is ON. After you have completed the schematic(s) in the Circuits folder, commit your changes and push them to Bitbucket. Don’t forget to ignore the extraneous circuits files as you did in Lab 0. |
|  | | |

In the report:

Include your design for part 1.4 and the screen-shot from the end of the tutorial in part 1.1.

Part 2: Your Second Embedded C program

Assignment:

|  |  |  |
| --- | --- | --- |
| ❑ | 2.1) | Using the starter code in the UVBasicTemplate folder, write a simple program that will enable GPIO Port B, initialize bit 0 on Port B to be an output and then go into a loop toggling (set the line high by direct assignment of 1, set the line low by direct assignment of 0, loop) that outputs bits as fast as possible. The loop should terminate when any key is pressed on TeraTerm (use kbhit() to test). Confirm the pulsing of the line using the oscilloscope. Don’t forget to commit your code and push it to the remote repository. |
| 01 // the common headers for I/O, C99 types  02 **#include <stdio.h>**  03 **#include <stdint.h>**  04 **#include <stdbool.h>**  05  06 // the headers to access the GPIO subsystem  07 **#include** "inc/hw\_memmap.h"  08 **#include** "inc/hw\_types.h"  09 **#include** "inc/hw\_gpio.h"  10 **#include** "inc/hw\_sysctl.h"  11  12 // the header to get the timing functions  13 **#include** "ES\_Port.h"  14  15 //header for itself  16 **#include** "Part2Functions.h"  17  18 **#define ALL\_BITS (0xff<<2)**  19  20  21  22 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  23 Function  24 part2Function  25 Parameters  26 None.  27 Returns  28 None.  29 Description  30 Toggles bit 0 on Port B by changing the state of all the bits on Port B  31 Notes  32 None.  33 Author  34 Navjot Singh  35 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  36 **void** part2Function**(void){**  37 printf**(**"Start of part2Function: **\n\r\n**"**);**  38  39 //enable Port B  40 HWREG**(**SYSCTL\_RCGCGPIO**) |=** BIT1HI**;**  41  42 //wait for Port B to be ready  43 **while** **((**HWREG**(**SYSCTL\_PRGPIO**) &** SYSCTL\_PRGPIO\_R1**) !=** SYSCTL\_PRGPIO\_R1**)**  44 **{**  45 **}**  46  47 //Initialize bit 0 on Port B to be a digital bit  48 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DEN**) |=** BIT0HI**;**  49 //Initialize bit 0 on Port B to be an output  50 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DIR**) |=** BIT0HI**;**  51  52 //toggling all bits on Port B  53 **while(!**kbhit**())**  54 **{**  55 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) =** **1;**  56 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) =** **0;**  57 **}**  58  59 **}** | | |
|  | | |
| ❑ | 2.2) | What is the frequency of the pulsing? What is the period of the HI time? What is the period of the LO time? Can you explain why they are different? |
| f = 2.11 MHz HI period = 56ns LO period = 416ns  The low period is much longer because the microcontroller has to loop around the entire function whereas in the high state the microcontroller directly then goes into the low state. | | |
| ❑ | 2.4) | Re-write your code from Part 2.1 so that as you toggle the line, you don’t disturb the state of any of the other lines on Port B. (Ignore for now that fact that since only bit 0 is an output only that pin could change. Pretend for a moment that they all could change if you wrote to them). Don’t forget to commit your code and push it to the remote repository. |
| 01 // the common headers for I/O, C99 types  02 **#include <stdio.h>**  03 **#include <stdint.h>**  04 **#include <stdbool.h>**  05  06 // the headers to access the GPIO subsystem  07 **#include** "inc/hw\_memmap.h"  08 **#include** "inc/hw\_types.h"  09 **#include** "inc/hw\_gpio.h"  10 **#include** "inc/hw\_sysctl.h"  11  12 // the header to get the timing functions  13 **#include** "ES\_Port.h"  14  15  16 //include header for itself  17 **#include** "Part2Functions.h"  18  19 **#define ALL\_BITS (0xff<<2)**  20  21 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  22 Function  23 part2Function  24 Parameters  25 None.  26 Returns  27 None.  28 Description  29 Toggles bit 0 on Port B while not changing the state of the other bits  30 Notes  31 None  32 Author  33 Navjot Singh  34 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  35 **void** part2Function**(void){**  36 printf**(**"Start of part2Function: **\n\r\n**"**);**  37  38 HWREG**(**SYSCTL\_RCGCGPIO**) |=** BIT1HI**;** //enable Port B  39 //wait for Port B to be ready  40 **while** **((**HWREG**(**SYSCTL\_PRGPIO**) &** SYSCTL\_PRGPIO\_R1**) !=** SYSCTL\_PRGPIO\_R1**)**  41 **{**  42 **}**  43  44 //Initialize bit 0 on Port B to be a digital bit  45 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DEN**) |=** BIT0HI**;**  46 //Initialize bit 0 on Port B to be an output  47 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DIR**) |=** BIT0HI**;**  48  49 //toggling only bit 0 on Port B  50 **while(!**kbhit**()){**  51 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) |=** BIT0HI**;**  52 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &=** BIT0LO**;**  53 **}**  54 **}** | | |
| ❑ | 2.5) | What is the frequency of the pulsing now? What is the period of the HI time? What is the period of the LO time? Can you explain why they are different both from one another and from the result in 2.2? |
| f = 1.37 MHz HI period = 120ns LO period = 580ns  The low time is much longer than the high time because of the same reason as in 2.2. From the high state, the microcontroller directly goes into the low state but going back to the high states involves looping around and reentering the function, which takes more computation time. Part 2.5 is in general slower than 2.2 because 2.5 involves masking the bits to only change bit 0, this adds another operation, so increases computation time for the microcontroller. | | |
| ❑ | 2.6) | Design a circuit to connect the 74HCT595 from your lab kit to the Tiva so that you can use 3 lines from the Tiva to control the '595. Connect the serial data in on the '595 to PB0, the shift clock to PB1 and the register clock to PB2. You will need to figure out the static states that you want for the other pins on the '595. |
|  | | |
| ❑ | 2.8) | Design and implement a module that will be the interface to the shift register. The public interface to the module should consist of 3 functions:  void SR\_Init(void); uint8\_t SR\_GetCurrentRegister(void); void SR\_Write(uint8\_t NewValue);  Don’t forget to commit your code and push it to the remote repository. |
| 01 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  02 Module  03 ShiftRegisterWrite.c  04  05 Revision  06 1.0.1  07  08 Description  09 This module acts as the low level interface to a write only shift register.  10  11 Notes  12  13 History  14 When Who What/Why  15 -------------- --- --------  16 10/11/15 19:55 jec first pass  17  18 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  19 // the common headers for C99 types  20 **#include <stdint.h>**  21 **#include <stdbool.h>**  22  23 // the headers to access the GPIO subsystem  24 **#include** "inc/hw\_memmap.h"  25 **#include** "inc/hw\_types.h"  26 **#include** "inc/hw\_gpio.h"  27 **#include** "inc/hw\_sysctl.h"  28  29 // the headers to access the TivaWare Library  30 **#include** "driverlib/sysctl.h"  31 **#include** "driverlib/pin\_map.h"  32 **#include** "driverlib/gpio.h"  33 **#include** "driverlib/timer.h"  34 **#include** "driverlib/interrupt.h"  35  36 **#include** "BITDEFS.H"  37  38 // readability defines  39 **#define DATA GPIO\_PIN\_0**  40  41 **#define SCLK GPIO\_PIN\_1**  42 **#define SCLK\_HI BIT1HI**  43 **#define SCLK\_LO BIT1LO**  44  45 **#define RCLK GPIO\_PIN\_2**  46 **#define RCLK\_LO BIT2LO**  47 **#define RCLK\_HI BIT2HI**  48  49 **#define GET\_MSB\_IN\_LSB(x) ((x & 0x80)>>7)**  50 **#define ALL\_BITS (0xff<<2)**  51  52 // an image of the last 8 bits written to the shift register  53 **static uint8\_t** LocalRegisterImage**=0;**  54  55 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  56 Function  57 SR\_Init  58 Parameters  59 None  60 Returns  61 None.  62 Description  63 Initializes the shift register  64 Notes  65 None.  66 Author  67 Navjot Singh  68 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  69 **void** SR\_Init**(void){**  70  71 // set up port B by enabling the peripheral clock, waiting for the  72 // peripheral to be ready and setting the direction  73 // of PB0, PB1 & PB2 to output  74 HWREG**(**SYSCTL\_RCGCGPIO**) |=** BIT1HI**;** //enable Port B  75  76 //wait for Port B to be ready  77 **while** **((**HWREG**(**SYSCTL\_PRGPIO**) &** SYSCTL\_PRGPIO\_R1**) !=** SYSCTL\_PRGPIO\_R1**)**  78 **{**  79 **}**  80  81 //Initialize bit 0,1,and 2 on Port B to be a digital bit  82 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DEN**) |= (**BIT0HI **|** BIT1HI **|** BIT2HI**);**  83 //Initialize bit 0 on Port B to be an output  84 HWREG**(**GPIO\_PORTB\_BASE**+**GPIO\_O\_DIR**) |= (**BIT0HI **|** BIT1HI **|** BIT2HI**);**  85  86 // start with the data & sclk lines low and the RCLK line high  87 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) |=** BIT2HI**;**  88 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &= (**BIT0LO **&** BIT1LO**);**  89 **}**  90  91 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  92 Function  93 SR\_GetCurrentRegister  94 Parameters  95 None.  96 Returns  97 None.  98 Description  99 retrieves the last uint8\_t that was passed to the shift register  100 Notes  101 None.  102 Author  103 Navjot Singh  104 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  105 **uint8\_t** SR\_GetCurrentRegister**(void){**  106 **return** LocalRegisterImage**;**  107 **}**  108  109 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  110 Function  111 SR\_Write  112 Parameters  113 uint8\_t NewValue  114 Returns  115 None.  116 Description  117 Passes an unsigned 8-bit that will be passed through by the shift register  118 Notes  119 None.  120 Author  121 Navjot Singh  122 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  123 **void** SR\_Write**(uint8\_t** NewValue**){**  124  125 **uint8\_t** BitCounter**;**  126 LocalRegisterImage **=** NewValue**;** // save a local copy  127  128 // lower the register clock  129 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &=** BIT2LO**;**  130  131  132 // this for-loop shifts out the data while pulsing the serial clock  133 **for** **(int** i **=** **0;** i **<** **8;** i**++)**  134 **{**  135  136 // Isolate the MSB of NewValue, put it into the LSB position and output to port  137 **if((**NewValue **&** BIT7HI**) !=** **0)**  138 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) |=** BIT0HI**;**  139 **else**  140 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &=** BIT0LO**;**  141  142 // raise SCLK  143 // lower SCLK  144 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) |=** BIT1HI**;**  145 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &=** BIT1LO**;**  146  147 NewValue **=** NewValue **<<** **1;**  148 **}**  149  150 // raise the register clock to latch the new data  151 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) |=** BIT2HI**;**  152 HWREG**(**GPIO\_PORTB\_BASE**+(**GPIO\_O\_DATA **+** ALL\_BITS**)) &=** BIT2LO**;**  153 **}** | | |
| ShiftRegisterWrite.h  01 **#ifndef SHIFT\_REGISTER\_H**  02 **#define SHIFT\_REGISTER\_H**  03 **void** SR\_Init**(void);**  04 **uint8\_t** SR\_GetCurrentRegister**(void);**  05 **void** SR\_Write**(uint8\_t** NewValue**);**  06 **#endif** | | |
| ❑ | 2.10) | Create a version of the code that you wrote for 2.4 only this time using the shift register functions from 2.8 to output to the shift register outputs rather than writing directly to the Tiva port. Don’t forget to commit your changes as you work. |
| main.c  01 **#include <stdint.h>**  02 **#include <stdbool.h>**  03 **#include <stdio.h>**  04  05 **#include** "ES\_Port.h"  06 **#include** "termio.h"  07 **#include** "ShiftRegisterModule.h"  08  09 **#define clrScrn() printf(**"**\x1b**[2J"**)**  10  11  12 **int** main**(void)**  13 **{**  14 // initialize the timer sub-system and console I/O  15 \_HW\_Timer\_Init**(**ES\_Timer\_RATE\_1mS**);**  16 TERMIO\_Init**();**  17 clrScrn**();**  18  19 puts**(**"**\r**Starting Basic Template **\r**"**);**  20 printf**(**"%s %s**\n**"**,**\_\_TIME\_\_**,** \_\_DATE\_\_**);**  21 printf**(**"**\n\r\n**"**);**  22  23 //arbitrary value chosen to test  24 **int** inputVal **=** **218;**  25 //initialize the shift register  26 SR\_Init**();**  27  28 //keep toggling between inputVal and 0 until a keyboard key is pressed  29 **while(!**kbhit**()){**  30 SR\_Write**(**inputVal**);**  31 SR\_Write**(0);**  32 **}**  33  34  35 // if you fall off the end of your code, then hang around here  36 **for(;;)**  37 **;**  38  39 **}** | | |
|  | | |
| ❑ | 2.11) | What is the frequency of the pulsing now? What is the period of the HI time? What is the period of the LO time? |
| f = 80 kHz HI period = 6.0us LO period = 6.6us | | |
| ❑ | 2.12) | Close the ****Vision IDE. Check in SourceTree if closing the IDE updated any of the configuration files and, if so, commit those final changes and push them up to Bitbucket. Then, take a screen-shot of the SourceTree window (Alt-PrtScr) with the Log/History tab for the Lab3Parts1\_2 repository selected and include it in your report. |
|  | | |
| ❑ | 2.13) | With a browser window opened to your Bitbucket account and the Branches pane of the Lab3 repository opened, take a screen-shot of the window (Alt-PrtScr) and include it in your report. |
|  | | |

In the report:

Include your answers to the questions for Parts 2.2, 2.5 & 2.11, the screenshots from 2.12 and 2.13, the schematic of the design from 2.6, and the listings of the code from part 2.1, 2.4, 2.8 & 2.10. The listings should be processed through the Highlight program and the results pasted into the report document (not in separate files) and formatted in the Courier New font.

Part 3: Controlling An LCD Display

Assignment:

|  |  |  |
| --- | --- | --- |
| ❑ | 3.2) | Using the proto-board, wire up the LCD display connector to the output from the 74HCT595 from 2.8. The LCD should get its power from the Tiva Protection Board.Do not apply power yet! |
| ❑ | 3.17) | After your final commit, take a screen-shot of the SourceTree window (Alt-PrtScr) with the Log/History tab for the Lab3 repository selected. |

In the report:

In the report document, include a schematic of the complete circuit, the screen shot from 3.17, listings of the header files and source code to the shift register, LCD and LCDservice modules. These should be processed through the Highlight program and the results pasted into the report document (not in separate files) and formatted in the Courier New font.

In the Reports folder:

1) A copy of the header files for the modules that you created (shift register, LCD and LCDservice).  
2) A copy of the source files for the modules that you created (shift register, LCD and LCDservice).