Design Assignment 1 – Embedded 'C' Programming Basics

1 Introduction

You will write code for PSoC 5LP relating to the following topics:

- Data types, signed and unsigned
- Register addressing, PSoC and #defines
- Hexadecimal
- Bitwise operations, register field setting
- Cycle counting

2 Learn About 'C' Integer Types and Sign Extension

Review the 'C' header file <stdint.h>, information is available at:

- http://pubs.opengroup.org/onlinepubs/009695399/basedefs/stdint.h.html

Especially note the types:

```
int32_t     int16_t     int8_t
uint32_t    uint16_t    uint8_t
```

Review the rules for 'C' type conversion between signed and unsigned integer types. A good summary of the rules is found at:

- https://msdn.microsoft.com/en-us/library/xbfs6fd4.aspx
- https://msdn.microsoft.com/en-us/library/e9s326kw.aspx

3 Learn About PSoC Registers

Download the file PSoC5LP Registers TRM_001-82120_0B.pdf from Canvas.

Find the address of the register named NVIC_CPUID_BASE _____

Find the mappings of all of the sub-fields of NVIC_CPUID_BASE, specifically at which bit position does each sub-field start and end. An example of a sub-field of NVIC_CPUID_BASE is ***partno***.

_____

Find the address of the register named PRT6_DR _____

4 Learn About Using 'C' to Read/Write Register Addresses

From http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/1500.html:

Placing C variables at specific addresses to access memory-mapped peripherals

In most ARM embedded systems, peripherals are located at specific addresses in memory. It is often convenient to map a C variable onto each register of a memory-mapped peripheral, and then read/write the register via a pointer. In your code, you will need to consider not only the size and address of the register, but also its alignment in memory.

The simplest way to implement memory-mapped variables is to use pointers to fixed addresses. If the memory is changeable by 'external factors' (for example, by some hardware), it must be labelled as volatile.

Consider a simple example:

```
#define PORTBASE 0x40000000
unsigned int volatile * port = (unsigned int *) PORTBASE;
```

The variable `port` is a constant pointer to a volatile unsigned integer, so we can access the memory-mapped register using:

```
*port = value; /* write to port */
value = *port; /* read from port */
```

The use of `volatile` ensures that the compiler always carries out the memory accesses, rather than optimizing them out (for example, if the access is in a loop).

5 Write 'C' Code to …

Write 'C' code to do the following:

Declare the variables:

- **32-bit unsigned integers**, as globals (outside of main)
  - implementer
  - variant
  - partno
  - revision
  - reg_val

Read the value of the register NVIC_CPUID_BASE into the variable reg_val:

```
// Create a convenient macro to hold the register address
#define NVIC_CPUID_BASE_ADDR ( put the register address here )

// Declare a pointer to a volatile unsigned int
uint32_t volatile * my_reg_ptr;

// Set the pointer to point to the register of interest
my_reg_ptr = ( uint32_t *) NVIC_CPUID_BASE_ADDR;

// Get the value in the register
reg_val = *my_reg_ptr;
```

Use ‘C’ right shift (>>) and bitwise AND (&) operators to extract the values of the fields IMPLEMENTER, VARIANT, PARTNO and REVISION from reg_val, and store these values into their respective variables. NOTE: I want each of these sub-fields from the register to be right-aligned in its variable. For example the field IMPLEMENTER occupies bits [31:24], you might do:

```
implementer = (reg_val >> 24) & 0xFF;
```

Use ‘C’ bitwise OR (|) operator and AND (&) operators to set and then clear bits [3:2] of the register PRT6_DR. NOTE: the idea is to affect only the two bits specified – the rest of the bits in the register must remain unchanged.

- Read the value of the register into reg_val
- Modify the bits of reg_val as directed
- Write the resulting value back to the register

You will find that the PSoC IDE has already created a macro definition for this register:

```
cydevice_trm.h: #define CYREG_PRT6_DR 0x40005160u
```

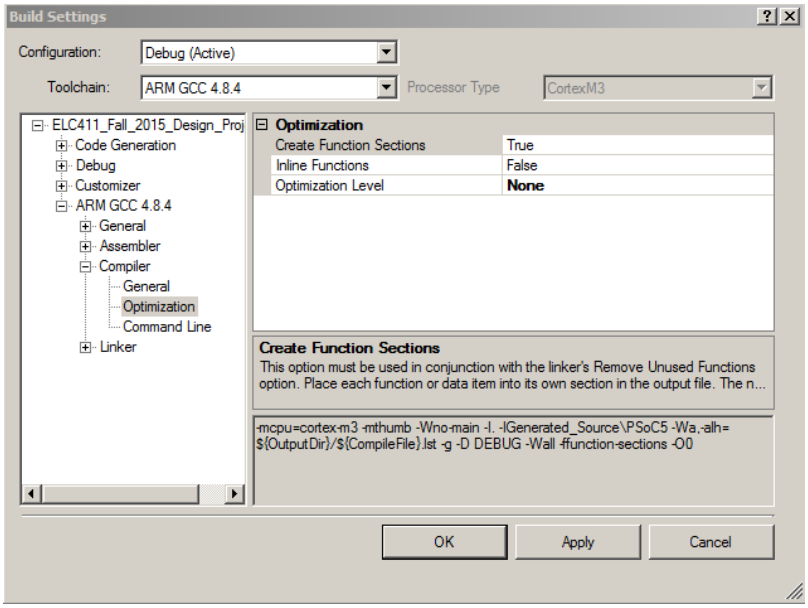
and you should make use of this.

6 CREATE A NEW PROJECT

Do the following:

Action	**Hints**
Get a PSoC 5LP system from the electronics store room, and connect it to your lab PC via USB.	CRITICAL: there are two mini-USB connectors on the board. YOU MUST connect to the one that is most near the edge of the board (not the center)!! The PC should recognize the board.
Use the Start menu to launch Cypress → PSoC Creator 3.3	
Select File → New → Project …	We will build for the CY8C5868AXI-LP035
In the Workspace Explorer window, Source Tab, double click on main.c, to bring up the source file.	
Add your code to main.c	Put your declarations under the comment: /* Place your initialization/startup code here */ Put your other code under the comment: /* Place your application code here. */ All code in this area will run in an infinite loop.
Add two digital outputs pins to your schematic	Go the XYZ.cysch panel for your project. On the right side expand the ‘Ports and Pins’ item. Drag and drop two digital output pins.

	Double click each pin and deselect HW Connection checkbox.
Map your pins to P6[3] and P6[2]	Go the <proj_name>.cydwr panel for your project. Use the pop-up menu in the Port column to select P6[2] and P6[3].
Disable code optimization	In the workspace explorer right click on your project, and select Build Settings … Expand 'Compiler' Click Optimization Set Optimization Level to None, using the pop-up menu
Build and run your project in debug mode	You can use function key F5 to build and run in debug mode.



7 SINGLE STEP THROUGH CODE

Add all of your variables to the Watch window (double click on variable name, right click, Add Watch).

You can execute your 'C' code one line at a time by pressing the F10 function key.

Observe the values of variables as you step through the code. Record the values of:

- implementer
- variant
- partno
- revision
- reg_val

Watch the board as you execute the code that writes values from reg_val back to the register PRT6_DR. What do you observe happening to the LEDs on the board?

8 EXPLORE EXECUTION SPEED

Add two delays using the code:

```
CyDelay(500);
```

before and after the code that clears the bits of PRT6_DR.

Build your code and run at full speed by using the "Debug→Program" option.

9 WRITE A REPORT

Write a report, which will be due by the date announced in class.

The report is to include, but not limited to the following:

a) Introduction – brevity is a virtue, but highlight any features of the lab that were new to you.
b) Software Architecture – very briefly describe the software "architecture" of your project, including as appropriate:
    a. Functions and files that you defined
    b. Macros or any new types or structures that you defined
    c. Input/output, if any
    d. Initialization
    e. Infinite loop
    f. Real-time aspects
c) Procedures & Results
    a. Brief subsection outlining each procedure that led to results.
    b. Present results as table, picture, or other form, as most appropriate
    c. Brief discussion of results including equations and graphs, as appropriate
d) The report must be understandable to another engineer or supervisor not working on this project.
e) A conclusion of your results and discussion of anything you found especially interesting or not expected from your work on this project.

Specific items that I will look for:

1. (2 points) How does 'C' handle the following (give an example where the smaller type is 8 bits, with the most-signficant-bit equal to '1')
    a. converting a smaller unsigned type to a larger signed or unsigned type – what are the bits of the smaller type, and the larger type, and what values are represented?
    b. converting a smaller signed type to a larger signed or unsigned type – what are the bits of the smaller type, and the larger type, and what values are represented?

    c. converting signed and unsigned types of the same size – what are the bits and what values are represented?
2. (1 point) The requested register addresses that you identified
3. (3 points) The values that you observed in the watch window, during single step, along with explanations
4. (3 points) Include your main.c file, strictly formatted using:
    a. spaces, not TAB characters for indenting
    b. 4 spaces for each level of indentation
    c. fully commented

REPORT FORMAT:

- One report per team.
- **Cover sheet with Title, Class, Names, etc.**
- Microsoft Word file or files.
- All source code files checked in to GitHub.

10 GRADING RUBRIC

The total grade will be 10 points, 9 points as indicated above, and 1 point for overall quality.