WORKSHOP 1

The aim of this tutorial is :

1. Refamiliarise yourself with C programming, in particular bitwise operations.
2. Learn the uVision IDE and be able to compile programs and download them.
3. Create a uvision project for the TM4C1294NCDPT microcontroller
4. Compile, edit, debug, and download and run code on the Tiva board
5. Become familiar with bitwise operators
6. Manipulate bits in variables and registers by setting bits, clearing bits, and merging bits

1 Create a uvison project

Follow the instructions in the provided document on the course website and create an empty uVision project.

2 Your First Embedded Tiva program

Type the program in Listing 1 into the main.c file using the Keil uVision IDE. (Do not copy and paste).

```
#include <stdlib.h> #include
<stdint.h>          #include
<stdbool.h>

#include "TM4C129.h"
#include "ES_Library.h"

int main(void) {

      uint32_t num=0xF;  // number to display in different formats

      ES_Serial(0,"19200,8,N,1") ;

      ES_printf("\nHello World\n") ;

      ES_printf("Hex: 0x%X\nBin: 0b%s\nDec: %i\n\n",num,ES_utob(num),num) ;

      while(true) ;
```

Listing 1 : The main.c for a Hello World program

Note: you will need to add the **ES_Library.h** and **ES_Library.lib** to your project.
(Right click Source Group 1, use *add existing files, and select the above files*).

- Compile the program from Listing 1 and download it. Press the reset button on the tiva board to start the program running on the Tiva board. The program is meant to print “Hello World” and a print out a number in hexadecimal, binary, and decimal formats
- Start TeraTerm and configure it for 19200 baud, 8 bits, No Parity, 1 stop bit. (See the document on the course website). Press reset to see the message on the screen.
- You will not see anything without the TeraTerm program because you the because the board is using a serial communication method and you need a dedicated serial display on the PC to see the serial messages.

3 Number systems (Hex, decimal, types)

1. Compile, Download, and run the program. What was displayed on the screen? Was it what you expected?
2. Change num to be 0xFFFFFFFF and run the program. Note that will give a result you probably did not expect. The decimal should be displayed as 4294967295. Modify the code to fix the problem.

Note: It is very important in embedded programming to be aware of signed and unsigned data types.

4 Bitwise Operators and non-destructive bit manipulation.

- Modify your Question 3 code so that it prints out the binary and the hex only.
- The code should use non-destructive bit manipulation.
- See Appendix A for a summary of bitwise operator

Setting Bits

i. Set num to be 0x00000000. Use the shift operator, **set** bit 7 and print the result.
ii. Set num to be 0x00000000. Use the shift operator, **set** bit 16 and print the result.
iii. Set num to be 0x00000000. Use 3 shift operations to **set** bits 4,5,6 and print the result.
iv. Set num to be 0x00000000. Use 1 shift operator to **set** bits 4,5,6 and print the result.

Clearing bits

i. Set num to be 0xFFFFFFFF. Use the shift operator to **clear** bit 20 and print the result.
ii. Set num to be 0xFFFFFFFF. Use the shift operator to **clear** bit 31 and print the result.
iii. Set num to be 0xFFFFFFFF. Use 3 shift operators to **clear** bits 4,5,6 and print the result.
iv. Set num to be 0xFFFFFFFF. Use 1 shift operator to **clear** bits 4,5,6 and print the result.

Merging bits

i. Set num to be 0x55555555 and run the program. Use shift operators to change the variable num to be 0x55555550 and print the result. Next use shift operators to change num to be 0x5555555A and print the result.
ii. Modify the program again to change num to be 0x555A5555 and print the result.
iii. Modify (ii) so that it does both operation in the one line of code.

5 Create a Resuable set of functions (library)

The setting, clearing, and merging of bits are operations that you will use throughout the course. It would make your life easier and make your code more readable if you convert these operations to functions.

```
setBit(uint32_t *Variable, int bit_position) ;

clearBit(uint32_t *Variable, int bit_position) ;

mergeBits(uint32_t *Variable, int bit_position, int value);
```

Examples of Use

```
setBit(&x, 3) ;   // make bit 3 of variable x become 1

clearBit(&x, 3) ;   // make  bit 3 of variable x become 0

mergeBits(&x, 3,7) ;   // make  bit 3,4,5 of variable x become 111 (7 decimal)
```

6 Reading bit values from a 32 bit variable.

Rewrite the code from Listing 2 to determine and print out value of bit 6 in the variable num.

Eg

1111 0000 has bit 6 set. (0xF0)

0000 1111 has bit 6 cleared. (0x0F)

Appendix A – Bitwise Operators

Symbol	Operator	Syntax Example
&	bitwise AND	result = x **&** y
|	bitwise inclusive OR	result = x **|** y
^	bitwise XOR (exclusive OR)	result = x **^** y
<<	left shift (shifts all bits to the left)	result = x **<<** y
>>	right shift (shifts all bits to the right)	result = x **>>** y
~	bitwise NOT (ones complement)	result = **~**x ;

- In the Syntax Example Column of the table above, **x** and **y** represent a variable or a constant value and **result** represents a variable.
- The result of the bitwise operations will be evaluated to give a value. If you want to use the value later then you must store it in a variable (in the Syntax Example Column result represents any variable)

Common Mistake

- Do not use && or || or ! for the bitwise operations as these are only used for testing expressions in the conditional statement such as if (), while () etc. eg while ((a>0) && (a<5))