Workshop 2

Lecture Material Covered : Topic 1, Topic 2
Required Documents: Programmers Register Reference (from the Course Homepage)
Course Reference Notes 2025 (from the Course Homepage)
TM4C1294 Datasheet (from the Course Homepage)

AIM:

The aim of this tutorial is to :

1. Understand the role of bit masking when looking for a change of state of a bit in a register.
2. Understand enabling the GPIO Ports using the SYSCTL registers
3. Understand configuring the GPIO Ports using the GPIO registers
4. Accessing the GPIO registers by using the formalism as defined by the uVision header file <TM4C129.h> (See the Register Reference document)

Setup the Template

- **Download** the *template.zip* file from the course website.
- **Extract** its contents.
- **Open** project file in the extracted template folder.

1 Using a bit mask to determine the state of bits in a variable

Modify the program so that it uses the **systctl** registers to enable GPIO Port B and

wait for it to be ready for use.
Refer to the supplied register reference and Appendix A for more information and Step 1 in Appendix B.

The program requirements are:

- Use the register notation defined in <TM4C129.h> header file, specifically:
  SYSCTL->GPIO
  SYSCTL->PRGPIO

- Write functions to:
  Enable Port B
  Wait until Port B is ready for use (via SYSCTL->GPIO)

- Before enabling Port B, display a status message using ES_printf() that includes the hexadecimal value of the RCGCGPIO and PRGPIO registers.
  (via SYSCTL->RCGCGPIO and SYSCTL->PRGPIO)

- Once Port B is ready after enabling it, use ES_printf() to print a confirmation message that includes the hexadecimal value of the RCGCGPIO and PRGPIO registers.
  (via SYSCTL->RCGCGPIO and SYSCTL->PRGPIO)

Are these register values what you expected?

2 Enabling multiple GPIO Ports Simultaneously

Modify the function from question 2 such that it

- Enables Port Q, Port N, and Port B at the same time and waits for all those ports to be ready and available to use before exiting the function

3 Develop a Program to Toggle a LED on Port N at 2Hz

Write a program so that Port N will be enabled and configured to allow LED D1 to be turned on and off via the GPIO Port N data register. Use the <TM4C129.h> header file

The program requirements are:

- The program must include a main() function that:
  Calls a function to **enable Port N**
  Calls a function to **configure Port N**
  Toggles the LED at **2Hz**
- Include a function named enablePortN() to Enable GPIO Port N. (see Appendix B step 1)
- Include a function named configPortN() to configure Port N Pin to use LED D1. (see Appendix B steps 2 -> 5)

Quiz Questions can be done here

4 Where is de bug hiding? (Debugging practice)

Run the code in Q3 and verify its operation using the peripheral debugging tools in the uVision debugger. (see the supplied video is on the course website)

5 Writing Reuseable Code

Generalise the code written for enabling the port by:

- Creating a function called that will enable the port and wait for the port to be ready before returning. The function header is given below:

```
void enablePort(uint32_t portMask)
```

where

The portMask contains 1's in the bit positions for the port according to the RCGCGPIO and PRGPR register definitions.

The Appendices can be found on the following page.

Appendix A - Using the TM4C1294.h header file.

The <TM4C1294.h> header file is supplied by Texas Instruments and its purpose is to give a simple method of access to the registers of the TM4C1294 microcontroller instead of referring to it by its memory address. Each peripheral is represented as a structure pointer with each member of the structure referring to a specific register within that peripheral. Refer to the lecture notes or datasheet for the use of the registers and the functionality associated with each bit of the registers.

Note: the naming scheme can differ slightly from the datasheet in places.

Usage:

Header File:	#include <TM4C1294.h>	
Syntax:	Peripheral **->** Register	// where peripheral is SYSTCTL, GPIO, etc
Example Write:	SYSCTL->RCGCGPIO = 0x4 ;	// Base is SYSCTL, Register is RCGCGPIO.
Example Read:	y = SYSCTL->PRGPIO ;	// Base is SYSCTL, Register is PRGPIO.

Appendix B - Basic Configuration of the GPIO peripheral.

1) Enable clock for port (SYSCTL->RCGCGPIO register) & Wait for clock to stabilise (SYSCTL->PRGPIO register)
2) Set data direction register (GPIO**X**->DIR register).
3) Set as GPIO port access (GPIO**X**->AFSEL)
4) Disable analog (GPIO**X**->AMSEL).
5) Enable digital (GPIO**X**-> DEN register) where X refers to one of the following GPIO ports

GPIO**A**_AHB	GPIO**B**_AHB	GPIO**C**_AHB	GPIO**D**_AHB	GPIO**E**_AHB	GPIO**F**_AHB	GPIO**G**_AHB	GPIO**H**_AHB	GPIO**J**_AHB
GPIO**K**	GPIO**L**	GPIO**M**	GPIO**N**	GPIO**P**	GPIO**Q**			

Table 1: *The peripheral names for each of the GPIO ports as defined in <TM4C1294.h>.*