Experiment 1B (2.5%)
Using GPIO to Interface to a Numeric Keypad

Time: 3 hrs Weight: 2.5%

Expectations

- You have watched the lectures for Topic 1 and Topic 2.
- That you will get sections marked off as they are completed.
- You will need to answer questions at the end to demonstrate understanding of what you have done.

Aim
To build an intelligent interface for a 12 key numeric keypad to the Tiva microcontroller board through GPIO ports and to develop experience in Embedded C programming and debugging using the Keil μVision software.

Note: In this part, the Tiva will be connected to your PC via a USB cable. Some precautions to avoid damaging the device are as follows.

When the Tiva has power on, you can:
- Press reset button
- Touch / release buttons on the board or connected to the board
- Connect / disconnect a voltmeter to the system
- Plug and unplug the USB cable

When the Tiva has power on, you can NOT:
- Plug / unplug the board into / from a bread-board or peripherals board
- Connect / disconnect cables or wires to external devices
- Insert / remove wires, resistors, or chips from the circuit connected to the board.

Introduction

This experiment aims to further develop your familiarity with Embedded C programming and understanding of the operation of the microcontroller by developing a program to enable the Tiva microcontroller to interface with 12 key numeric keypad via General Purpose input/output (GPIO) ports.

The keypad is arranged in a 4 x 3 pattern. See Appendix A to find out how to connect to the device and find keypresses.

Preliminary: 2 marks

Preliminary tasks must be done before your lab session and submitted at the start of the laboratory.

Use the nomenclature as given in the <TM4C129.h> header file. Refer to the supplied Register Reference

1) List the registers and required values needed to be to enable and setup **Port K Pins 0, 1, 2** to be a **digital output** with **2mA current drive**. (Refer to Appendix C).

Register using <TM4C129.h> format (peripheral->register) Eg `SYSCTL->RCGCGPIO`	C code to configure / read register (include the bit mask) `SYSCTL->RCGCGPIO |= (1<<0);`

2) List the registers and required values needed to be to enable and setup **Port E Pins 0, 1, 2, 3** to be a **digital input** with a **Pull-Down resistor**. (Refer to Appendix C)

Register using <TM4C129.h> format (peripheral->register)	C code to configure/ read register (include the bit mask)

Experiment

1. Download the template.zip from the course website
2. Extract the template
3. Copy the folder and rename the folder to Part your are attempting. Eg copy template to P1 for Part 1.
4. Repeat step 3 for each new part. This keeps the code from the previous Part.

Part 1: GPIO PORT K Setup for output 2 marks

Write a function called **enablePKE()** to enable Ports K & E.

Check that the function works as expected by using the debugger (SYSCTL). See the tutorial video about debugging peripherals using uVision. The debugging demonstration video can be found on the course website

Show the running program to your demonstrator and show its correct operation with the debugger.

Part 2: Configure Port K 2 marks

Write a function called **configuePK()** to configure Port K.

The function must set pins PK0, PK1, PK2 to be digital output with 2mA maximum current drive.

Test this program by toggling pins PK[0:2] with a suitable 500ms delay by writing to the Data Register and using the supplied **msDelay()** function. You can check that the pins are working by viewing the waveforms for each pin on the oscilloscope.

Show the running program to your demonstrator and show its correct operation on the oscilloscope.

Part 3: Configure Port E 2 marks

Write a function called **configurePE()** to configure Port E.

The function must set pins PE0, PE1, PE2, PE3 to be digital input with a Pull Down Resistor.

Test this by continually reading the Data register from Port E and printing out the contents of PE[0:3] using ES_printf(). To show only the lower 4 bits of the port, AND the data with 0xF before printing it out.

Connect a wire to the 3.3V pin and connect the other end to PE0. Note what the PE0 shows in the printout when the wire is connected and when the wire is disconnected from the PE0 pin. Verify that all the other pins work by using the same method. (This illustrates the effect of the pull-down resistors on a pin when it is floating)

Show the running program to your demonstrator and show what happens to the print out when the 3.3V wire is connected and disconnected from the pins.

Part 4: Keypad Setup 1 mark

Wire up the Keypad to the ports according to Appendix A (a). Appendix A shows the pinout for the Keypad and the corresponding port pins to use. Eg PE0 is connected to pin 2 on the keypad.

Write a function called **displayPEK()** that continually prints out the Port K data register and the Port E data register. Note that port K will correspond to the columns and Port E will correspond to the rows. If you have it connected properly then the row and the column should both be 1 when the a single key is pressed.

Test this by setting PK0 to a **1** and examining the output for a key press. Compare the output of Port E to the rows and columns shown in Appendix A. This will show you which key was pressed. Eg if PE0 shows a **1** when the key was pressed then this corresponds to the One key.

Show the running program to your demonstrator and show what happens when you press a key on the keypad

Part 5: This is the key! 3 marks

For column 0 (PK0) write the code to make PK0 output a high. Then read the lower nibble of Port E (mask using 0xF) and print it out. Verify that the "147*" column is showing the right row values when the keys are pressed. Modify your code to detect which key in the column has been pressed and print out that key's character.

S**how the running program to your demonstrator**

Part 6: Putting it all together. 3 marks

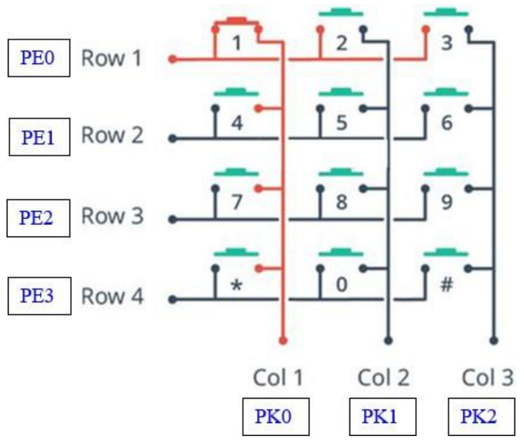
Expand Part 5 so that it can check and print out the character for any key pressed on the keypad.

S**how the running program to your demonstrator**

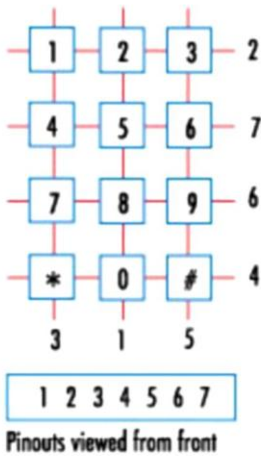
Ensure you have shown your demonstrator the correct operation of each completed part of this lab, and have completed the online submission before leaving the laboratory. You will not be able to complete this later.

Appendix

Appendix A: Matrix Keypad



(a)



(b)

Keypad Pin #	Tiva Pin
1	PK1
2	PE0
3	PK0
4	PE3
5	PK2
6	PE2
7	PE1

The block diagram for the keypad keys can be seen in the (a) and the pinouts for the keypad board can be seen in (b). The key labels are shown on both diagrams.

(a) Shows the Rows and columns and the Port Pins that they should be connected to in this laboratory.
(b) Shows which pins in the keypad header the rows and columns are connected to.

<u>Theory</u>

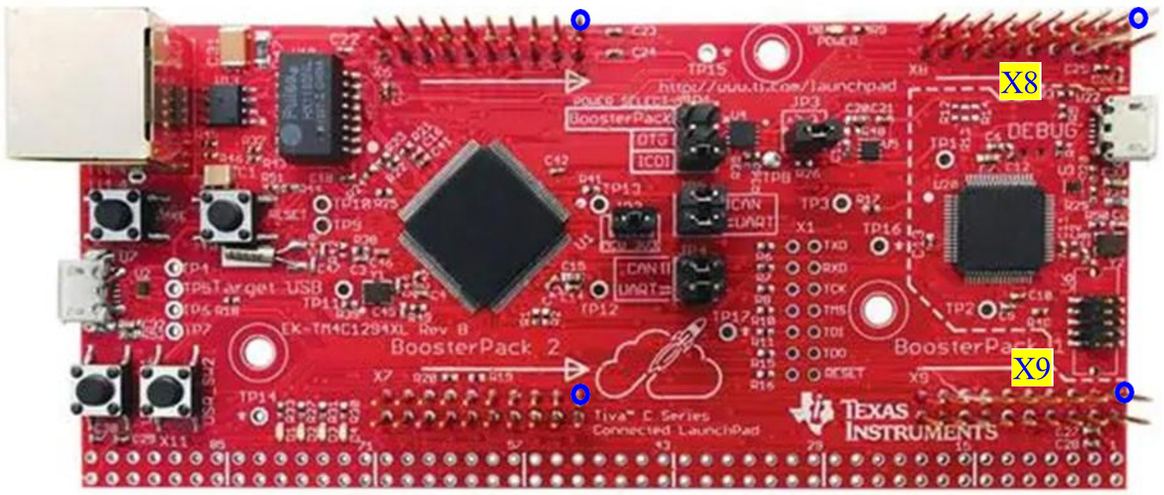
This keypad is called a “matrix keypad” and each of the keys forms a bridge or short across two of the corresponding pins. For example, if you were to press the number “1” on the keypad, there would be a connection between col 1 and row 1 (shown as the red line). If col 1 was connected to 3.3V then row 1 will also have 3.3V. The other keys will be floating. This means that it needs to be tied to a voltage using either a Pull-Up or Pull-Down Resistor. In this lab col 1 will supply a High (3.3V) which means that the floating lines should be pulled down to Low (0V).

To use the keypad on a microcontroller, we put a HIGH (3.3V) on each of the columns, one at a time, and test which row pin receives the signal. This is given the sequence listed below:

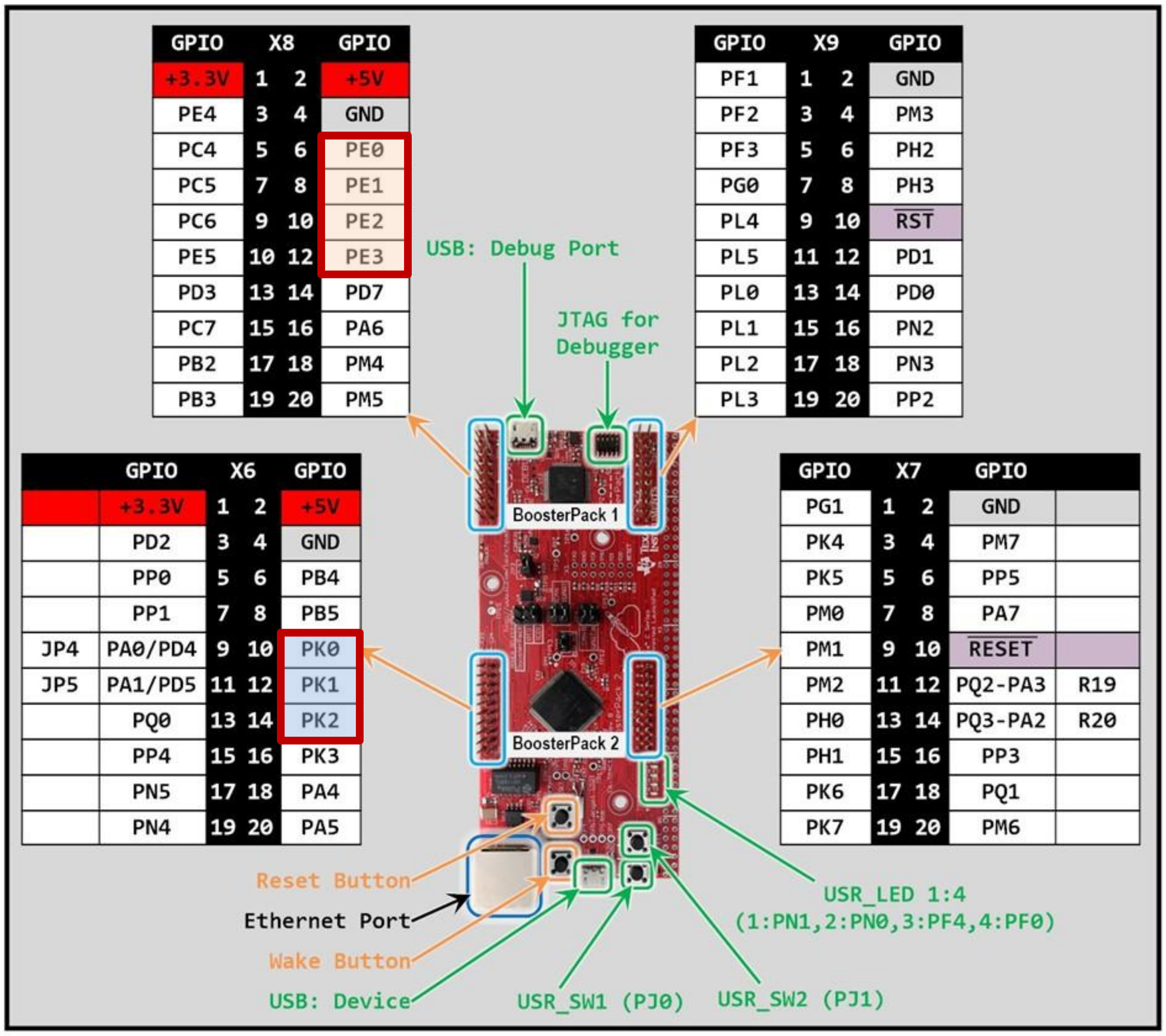
1) Write a High to Col 1
2) Read the rows and check for a High (connection). The other lines will be Low (no connection)
3) The row with the High will be the row of the key that was pressed.
4) Print out which key was pressed.
5) Repeat for Col 2 and Col 3
6) Start the sequence again to keep looking for a keypress

Adapted from the Jaycar SP0770 datasheet

Appendix B: Tiva Board Headers



The board showing the Headers and Header numbers. Pin 1 is shown by the Blue Circle



GPIO	X8		GPIO
+3.3V	1	2	+5V
PE4	3	4	GND
PC4	5	6	PE0
PC5	7	8	PE1
PC6	9	10	PE2
PE5	10	12	PE3
PD3	13	14	PD7
PC7	15	16	PA6
PB2	17	18	PM4
PB3	19	20	PM5

GPIO	X9		GPIO
PF1	1	2	GND
PF2	3	4	PM3
PF3	5	6	PH2
PG0	7	8	PH3
PL4	9	10	$\overline{\text{RST}}$
PL5	11	12	PD1
PL0	13	14	PD0
PL1	15	16	PN2
PL2	17	18	PN3
PL3	19	20	PP2

	GPIO	X6		GPIO
	+3.3V	1	2	+5V
	PD2	3	4	GND
	PP0	5	6	PB4
	PP1	7	8	PB5
JP4	PA0/PD4	9	10	PK0
JP5	PA1/PD5	11	12	PK1
	PQ0	13	14	PK2
	PP4	15	16	PK3
	PN5	17	18	PA4
	PN4	19	20	PA5

GPIO	X7		GPIO	
PG1	1	2	GND	
PK4	3	4	PM7	
PK5	5	6	PP5	
PM0	7	8	PA7	
PM1	9	10	$\overline{\text{RESET}}$	
PM2	11	12	PQ2-PA3	R19
PH0	13	14	PQ3-PA2	R20
PH1	15	16	PP3	
PK6	17	18	PQ1	
PK7	19	20	PM6	

Appendix C: Configuring the GPIO

1) Enable clock for port (SYSCTL_RCGCGPIO register)
   & Wait for clock to stabilise (SYSCTL_PRGPIO register)
2) Set data direction register (GPIO_DIR register).
3) Set as GPIO port access (GPIO_AFSEL).
4) Enable current (4/8mA) (GPIO_DR8R/4R) (if required) on outputs

   OR

   Enable pull-up/pull-down (GPIO_PUR, GPIO_PDR) (if required) on inputs.
5) Enable digital (GPIO_PORT_DEN register) & Disable analog (GPIO_PORT_AMSEL).

Name	Bit Settings
RCGCGPIO	bit # is port #
PRGPIO	bit # is port #

# (1) Uses the same port value for both registers since both operate on the same port eg Port A = 0x1.

# (2-5) Uses the same pin values for all the registers since all operate on the same pin eg 0xFF.

GPIO Register	Purpose	Pin / Bit Settings
DATA	Data port to read/write from/to pins	Output: 0 =0V ; 1 = 3.3V Input: 0V = 0 ; 3.3V = 1
DIR	Data Direction of the pins	0 - input 1 - output
AFSEL	Alternate Function Select	0 - regular port 1 - alternate function given by PCTL
DEN	Digital Mode Enable	0 - disable digital mode 1 - enable digital mode
PCTL	Defines the alternative function	See table of Alternative Functions Given in Datasheet and Course Reference Notes
DR2R	Set pin output drive to 2mA	0 - disable 1 - enable
DR4R	Set pin output drive to 4mA	0 - disable 1 - enable
DR8R	Set pin output drive to 8mA	0 - disable 1 - enable
DR12R	Set pin output drive to 12mA	0 - disable 1 - enable
PDR	Pull Down Resistor (pulls to ground when pin floating)	0 - disable 1 - enable
PUR	Pull Up Resistor (pulls to 3.3V when pin floating)	0 - disable 1 - enable

Note: Each bit in the Register correspond to that Pin in the port
eg bit 0 corresponds to pin 0, bit 1 corresponds to pin 1 etc