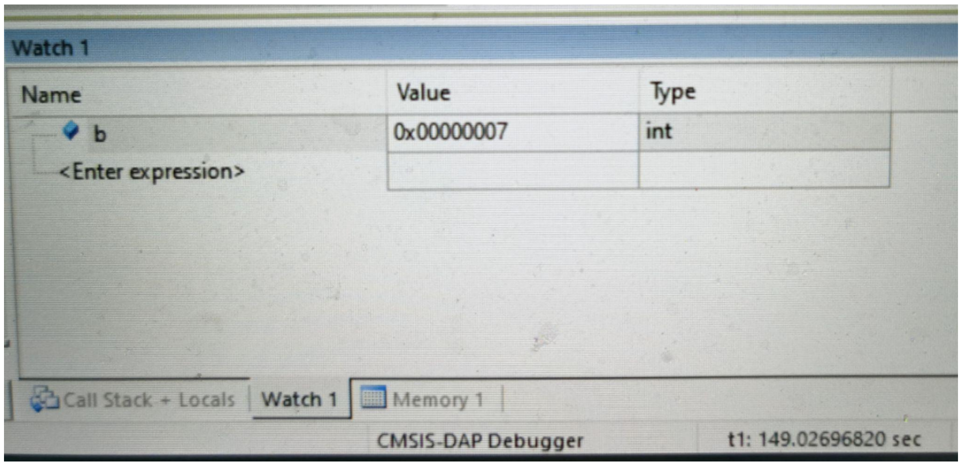
Theme Report - Reason

Junbo Wang - wangj430 - 400249823

We talked about the second theme, called "Reason." For the fourth, fifth, and sixth weeks, we started looking at how the peripheral manipulates the microcontroller, like the keypad. Also, we discovered how the interrupt is triggered for an embedded system.

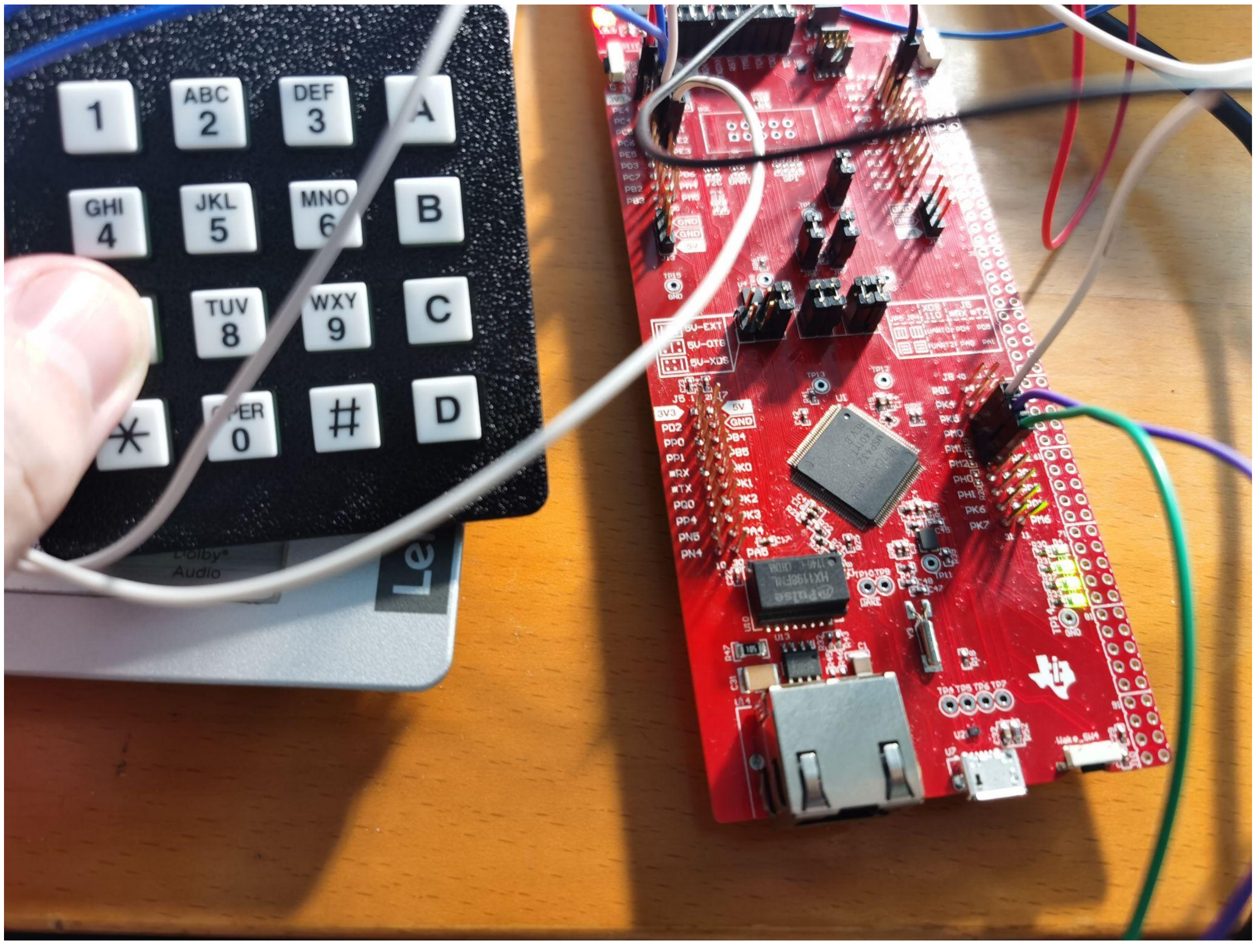
For this theme, "Reason", it is the problem-solving method which tells us how hardware works and how to control the system. For instance, we use the 4*4 keypads in some labs, but why do we need this keypad for the microcontroller. The reasons for that are "As those pins are connected with the switches/keys, we can't use them but only as I/O ports." and "The answer is, using a hex keypad or matrix keypad; we can reduce pin counts, which associate 4x4 matrix keys."**[1]** These answers indicate the theme "Reason", which is introduced in detail in the following paragraphs.

In lab 5 milestone 2 and 3, we used the keypad as input to manipulate the software and the onboard LED on the microcontroller. We ran the code successfully and turned into debugging mode. When we pressed the buttons on the keypad, the values were shown in the table. For instance, when we pressed the button 7, the value of 7 was shown in this table and onboard LEDs D1, D2, and D3 shined, representing the binary value of 7 which is 0111.**[2]** By interfacing peripherals via hardware and using software to decode the keypad input, we get different outputs when we push the input keys on the keyboard. That is the reason for this milestone.


Watch 1

Name	Value	Type
b	0x00000007	int
<Enter expression>		

[2]



[2]

First, we did the initialization for Port M and E, and then set the Port N and F for the related LEDs. Then, we built a variable to reflect the return values from the input key. After that, we wrote several while loops for all of the input keys on the keyboard during the debugging process. For instance, if we want to press 1 on the keypad, we should check 0001 for the value 1 address and turn on the corresponding LED D1 address 0001 in the while loop. When we execute the code and enter debug mode, we can see that the return value is the same as the key we hit. The LED matching to the supplied binary value lights up at the same moment.**[3]**

The theme "Reason" is shown in the milestones. According to the examples above, we observe that different outputs will be shown when we push the input keys on the keyboard. It illustrates that once a unique code for a key is identified, it must be decoded to provide meaning for the key. Also, we set up several while loops in code, this led to identifying the correct push button we pushed and related

LEDs. That concludes the reasons for the milestone results. Therefore, we decode the input key successfully and see the LEDs' display. The theme "Reason" is presented in both peripheral connection and software decoding.

According to the milestones, I understand that the input and output are related. The theme, "Reason," is the problem-solving method or the answer, which is crucial for the process. In the examples above, I can get the correct outputs with a clear reason for the problem and inputs. I know that reasons are important for solving questions. When I grasp the root of the problem, I will be able to solve it correctly and fully comprehend its complexities. If I do the further projects through the microcontroller, I am able to do the debugging or hardware connection with the correct understanding of the problems' reasons.

Reference

[1] circuitdigest.com. "4x4 Matrix Keypad Interfacing with PIC Microcontroller." Available:https://circuitdigest.com/microcontroller-projects/4x4-keypad-interfacing-with-pic16f877a/#:~:text=Why%20we%20need%204x4%20Keypad%3A%20Typically%20we%20use,we%20will%20end%20up%20using%2016%20I%2FO%20ports. , May 28, 2018, [Accessed: March 19,2022].
[2] Junbo Wang & Yichen Lu. "Lab_05_wangj430_luy191". Result, pp3, Mar. 2022
[3] Junbo Wang & Yichen Lu. "Lab_05_wangj430_luy191". Method, pp2-3, Mar. 2022
[4] Junbo Wang & Yichen Lu. "Lab_05_wangj430_luy191". Code Appendix, pp6-11, Mar. 2022

Code Appendix [4]

```
// 2DX4StudioW30E1_Decoding a Button Press
// This program illustrates detecting a single button press.
// This program uses code directly from your course textbook.
//
// This example will be extended for in W21E0 and W21E1.
//
//  Written by Ama Simons
//  January 30, 2020
//  Last Update: January 21, 2020

// Name: Yichen Lu Junbo Wang
// Student id: 400247938 400249823
// Date: Feb, 28th, 2022
```

```
#include <stdint.h>
#include "tm4c1294ncpdt.h"
#include "Systick.h"
#include "PLL.h"

void PortE0E1_Init(void){
    SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R4;                              // activate
the clock for Port E
    while((SYSCTL_PRGPIO_R&SYSCTL_PRGPIO_R4) == 0){};                   // allow time
for clock to stabilize
  GPIO_PORTE_DEN_R = 0b00001111;                                        // Enabled
both as digital outputs
    return;
    }


void PortM0M1_Init(void){
    SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R11;                         //activate the
clock for Port M
    while((SYSCTL_PRGPIO_R&SYSCTL_PRGPIO_R11) == 0){};              //allow time for
clock to stabilize
    GPIO_PORTM_DIR_R = 0b00000000;                                                //
make PM0 an input, PM0 is reading if the button is pressed or not
  GPIO_PORTM_DEN_R = 0b00001111;
    return;
}


//Turns on D2, D1
void PortN0N1_Init(void){
    SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R12;                         //activate the
clock for Port N
    while((SYSCTL_PRGPIO_R&SYSCTL_PRGPIO_R12) == 0){};
    GPIO_PORTN_DIR_R=0b00000011;
    GPIO_PORTN_DEN_R=0b00000011;
    return;
}

//Turns on D3, D4
void PortF0F4_Init(void){
  SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R5;                          //activate the
clock for Port F
    while((SYSCTL_PRGPIO_R&SYSCTL_PRGPIO_R5) == 0){};
    GPIO_PORTF_DIR_R=0b00010001;
    GPIO_PORTF_DEN_R=0b00010001;
    return;
}
```

```
int main(void){
    //PLL_Init();
    //SysTick_Init();
    PortE0E1_Init();
    PortM0M1_Init();
    PortN0N1_Init();
    PortF0F4_Init();
    volatile int a=0b00000000;
    volatile int b=0b00000000;

    while(1){


//Row 1

   GPIO_PORTE_DIR_R  =  0b00000001;            // To drive you use the data
direction register
   GPIO_PORTE_DATA_R =  0b00000000;

    //Checks If Button 3 is pressed - D2 lights up
    //Unique code is:   1010 - In the order of PE1 PE0 PM1 PM0
    while((GPIO_PORTM_DATA_R&0b00000001)==0){
    GPIO_PORTN_DATA_R = 0b00000010;
    a=0b11101110;
    b=0b0001;
    }
    a=0;
    b=0;



    //Checks If Button 4 is pressed - D1 lights up
    //Unique code is: 1001 - In the order of PE1 PE0 PM1 PM0
  while((GPIO_PORTM_DATA_R&0b00000010)==0){
      GPIO_PORTN_DATA_R = 0b00000001;
        a=0b11011110;
        b=0b0010;
    }
    a=0;
    b=0;

    while((GPIO_PORTM_DATA_R&0b00000100)==0){
      GPIO_PORTN_DATA_R = 0b00000011;
        a=0b10111110;
        b=0b0011;
    }
    a=0;
    b=0;

    while((GPIO_PORTM_DATA_R&0b00001000)==0){
      GPIO_PORTN_DATA_R = 0b00000001;
        GPIO_PORTF_DATA_R = 0b00000001;
        a=0b01111110;
        b=0b1010;
    }
    a=0;
    b=0;
    //Row 2

    GPIO_PORTE_DIR_R  =  0b00000010;              //Drive Row 2
    GPIO_PORTE_DATA_R =  0b00000000;
//Data is still as registers


    //Checks if Button 1 is pressed - D3 lights up
    //Unique code is: 0110 - In order of PE1 PE0 PM1 PM0
  while((GPIO_PORTM_DATA_R&0b00000001)==0){
    GPIO_PORTF_DATA_R=0b000010000;
        a=0b11101101;
        b=0b0100;
    }
    a=0;
    b=0;


//Checks if Button 2 is pressed - D4 Lights up
    //Unique Code is: 0101  - In order of PE1 PE0 PM1 PM0
 while((GPIO_PORTM_DATA_R&0b00000010)==0){
        GPIO_PORTF_DATA_R=0b000010000;
      GPIO_PORTN_DATA_R=0b000000010;
        a=0b11011101;
        b=0b0101;
    }
    a=0;
    b=0;

    while((GPIO_PORTM_DATA_R&0b00000100)==0){
        GPIO_PORTF_DATA_R=0b000010000;
      GPIO_PORTN_DATA_R=0b000000001;
        a=0b10111101;
        b=0b0110;
    }
    a=0;
    b=0;
    while((GPIO_PORTM_DATA_R&0b00001000)==0){
        GPIO_PORTF_DATA_R=0b000000001;
      GPIO_PORTN_DATA_R=0b000000011;
        a=0b01111101;
        b=0b1011;
    }
    a=0;
    b=0;

//Row 3
      GPIO_PORTE_DIR_R  =  0b00000100;            //Drive Row 2
    GPIO_PORTE_DATA_R =  0b00000000;

        while((GPIO_PORTM_DATA_R&0b00000001)==0){
            GPIO_PORTF_DATA_R=0b000010000;
            GPIO_PORTN_DATA_R=0b000000011;
            a=0b11101011;
            b=0b0111;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00000010)==0){
            GPIO_PORTF_DATA_R=0b000000001;
            a=0b11011011;
            b=0b1000;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00000100)==0){
            GPIO_PORTF_DATA_R=0b000000001;
            GPIO_PORTN_DATA_R=0b000000010;
            a=0b10111011;
            b=0b1001;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00001000)==0){
            GPIO_PORTF_DATA_R=0b000010001;
            a=0b01111011;
            b=1100;
        }
        a=0;
        b=0;

//Row 4
        GPIO_PORTE_DIR_R  =  0b00001000;              //Drive Row 2
    GPIO_PORTE_DATA_R =  0b00000000;

        while((GPIO_PORTM_DATA_R&0b00000001)==0){
            GPIO_PORTF_DATA_R=0b000010001;
            GPIO_PORTN_DATA_R=0b000000001;
            a=0b11100111;
            b=0b1110;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00000010)==0){
            GPIO_PORTF_DATA_R=0b000000000;
            a=0b11010111;
            b=0b0000;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00000100)==0){
            GPIO_PORTF_DATA_R=0b000010001;
            GPIO_PORTN_DATA_R=0b000000011;
            a=0b10110111;
            b=0b1111;
        }
        a=0;
        b=0;

        while((GPIO_PORTM_DATA_R&0b00001000)==0){
            GPIO_PORTF_DATA_R=0b000010001;
            GPIO_PORTN_DATA_R=0b000000010;
            a=0b01110111;
            b=0b1101;
        }
        a=0;
        b=0;



GPIO_PORTN_DATA_R=0b00000000;
GPIO_PORTF_DATA_R=0b00000000;

}
}
```