2DX4: Microprocessor System
Lab 2

Instructor: Drs.Boursalie, Doyle, Haddara

Lab TAs: Fatemeh Derakshani (derakhsf), Han Zhou (zhouh115)

Junbo Wang – L01 – wangj430

Yichen Lu – L01 – luy191

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Junbo Wang wangj430 400249823]**

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is our own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Yichen Lu luy191 400247938]**

1. Purpose

The objective of this lab is to incorporate digital design methods into the embedded systems. We will use the finite state machine (FSM) approach and the GPIO knowledge in this lab.

2. BackGround

There are two milestones in this lab. Milestone 1 and 2 both need to build the active high circuit which is associated with studio 2. For milestone 1, we write the assembly code to implement a parallel I/O system as a combinational digital lock. In Milestone 2, we use the given sequential lock codes to enable the digital lock by using assembly code to implement this sequential system as a digital lock. The concept of a combinational digital lock and sequential lock are both introduced in studio 2.

3. Method

Milestone 1

The first milestone is connecting an active low circuit and using combinational code to implement the digital lock. We connect the active low circuit by looking through studio 2. The input buttons are connected at PM0, PM1, PM2 on the microcontroller. The load button is connected at PM3 on the microcontroller. First, we edit the address of Port M and Port N. Then, we set up the RCGCGPIO, GPIO_DIR, and GPIO_DEN of Port M and Port N respectively. Also, we need to set the initial state to locked. At last, we create a loop and use ANDing our combinational code with 1111. After the AND operation, we can see if the result is in locked state or unlocked state.

Milestone 2

The milestone 2 is connecting an active high circuit and using sequential code to implement the digital lock. We connect the active high circuit by looking through studio 2. The input button is connected at PM0 on the microcontroller. The load button is connected at PM2 on the microcontroller. First, we edit the address of Port M and Port N. Then, we set up the RCGCGPIO, GPIO_DIR, and GPIO_DEN of Port M and Port N respectively. Also, we need to set the initial state to locked, which is #2_00000001. At last, we create six states and use AND, BNE and BEQ operation codes to implement our sequential code with 0001. After four state operations, we can see if the result is in a locked state or unlocked state. In summary, there are six stages to the entire procedure. If the input is valid, it will go downwards; otherwise, it will lock and then return to the first state. If all of the inputs are right, it will go into an unlocked state for a while before returning to the original state. In this process, we should ensure that the input is genuine, and if it is, we should keep it in the original state and not proceed.

4. Results

Milestone 1

Milestone 2

5. Observation and conclusion

In milestone 1, we use the active low circuit and connect the input buttons that are PM0, PM1, PM2 on the microcontroller. The load button is PM3 on the microcontroller connected. We use the combinational code 0110 to decode the combinational lock. When we run the code, LED D1 is initially on; when we press the button pin1, pin 2, and pin 3 together, the LED D1 will turn off and the LED D2 will turn on.

In milestone 2, we use the active high circuit and just connect PM0 as a input button and PM2 as a load button. We use the sequential code 0, 0, 0, 1 to decode the sequential lock. When we run the code, LED D2 is initially on. After we press the button in the sequence of our code,0001, the LED D2 will turn off, and the LED D1 will turn on.

Code Appendix

Lab2_Milestone1

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


; Name: Junbo Wang;Yichen Lu
; Student Number: 400249823;400247938
; MACID: Wangj430;luy191
; Lab Section: L01
; Description of Code:


; Original: Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;ADDRESS DEFINTIONS

;The EQU directive gives a symbolic name to a numeric constant, a
register-relative value or a program-relative value


SYSCTL_RCGCGPIO_R            EQU 0x400FE608  ;General-Purpose Input/Output Run
Mode Clock Gating Control Register (RCGCGPIO Register)
GPIO_PORTN_DIR_R             EQU 0x40064400  ;GPIO Port N Direction Register
address
GPIO_PORTN_DEN_R             EQU 0x4006451C  ;GPIO Port N Digital Enable
Register address
GPIO_PORTN_DATA_R            EQU 0x400643FC  ;GPIO Port N Data Register address

GPIO_PORTM_DIR_R             EQU 0x40063400  ;GPIO Port M Direction Register
Address (Fill in these addresses)
GPIO_PORTM_DEN_R             EQU 0x4006351C  ;GPIO Port M Direction Register
Address (Fill in these addresses)
GPIO_PORTM_DATA_R            EQU 0x400633FC  ;GPIO Port M Data Register Address
(Fill in these addresses)

COMBINATION EQU 2_0001  ;passward
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Do not alter this section

        AREA    |.text|, CODE, READONLY, ALIGN=2 ;code in flash ROM
        THUMB                                    ;specifies using Thumb
instructions
        EXPORT Start
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;Function PortN_Init
PortN_Init
    ;STEP 1
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0,R0, #0x1000
    STR R0, [R1]
    NOP
    NOP

    ;STEP 5
    LDR R1, =GPIO_PORTN_DIR_R
    LDR R0, [R1]
    ORR R0,R0, #0x3
    STR R0, [R1]

    ;STEP 7
    LDR R1, =GPIO_PORTN_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0x3
    STR R0, [R1]
    BX  LR

PortM_Init
    ;STEP 1
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0,R0, #0x800
    STR R0, [R1]
    NOP
    NOP

    ;STEP 5
    LDR R1, =GPIO_PORTM_DIR_R   ;/*direction*/
    LDR R0, [R1]
    AND R0,R0, #0x00
    STR R0, [R1]

    ;STEP 7
    LDR R1, =GPIO_PORTM_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0xF

    STR R0, [R1]
    BX  LR


State_Init LDR R5,=GPIO_PORTN_DATA_R  ;Locked is the Initial State
           MOV R4,#2_00000010
           STR R4,[R5]
           BX LR

Start
```

```
    BL   PortN_Init
    BL   PortM_Init
    BL   State_Init
    LDR R0, =GPIO_PORTM_DATA_R  ; Inputs set pointer to the input
    LDR R3, =COMBINATION         ;R3 stores our combination

Loop
            LDR R1,[R0]
            AND R2,R1,#2_00001111
            CMP R2,R3
            BEQ Unlocked_State
            BNE Locked_State
Locked_State
    LDR R5,=GPIO_PORTN_DATA_R
    MOV R4,#2_00000010
    STR R4,[R5]
    B Loop

Unlocked_State
    LDR R5, =GPIO_PORTN_DATA_R
    MOV R4,#2_00000001
    STR R4, [R5]
    B Loop
    ALIGN

END
```

Lab2_Milestone2

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


; Milestone 2, Lab 2
; Name: Yichen Lu  Junbo Wang
; Student Number: 400247938 400249823
; Lab Section: L01
; Description of Code: turn on LED D1 if the sequential password is correct
(0001), otherwise turn on LED D2.


; Original: Copyright 2014 by Jonathan W. Valvano, valvano@mail.utexas.edu



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;ADDRESS DEFINTIONS

;The EQU directive gives a symbolic name to a numeric constant, a
register-relative value or a program-relative value


SYSCTL_RCGCGPIO_R            EQU 0x400FE608  ;General-Purpose Input/Output Run
Mode Clock Gating Control Register (RCGCGPIO Register)
GPIO_PORTN_DIR_R             EQU 0x40064400  ;GPIO Port N Direction Register
address
GPIO_PORTN_DEN_R             EQU 0x4006451C  ;GPIO Port N Digital Enable
Register address
GPIO_PORTN_DATA_R            EQU 0x400643FC  ;GPIO Port N Data Register address

GPIO_PORTM_DIR_R             EQU 0x40063400  ;GPIO Port M Direction Register
Address (Fill in these addresses)
GPIO_PORTM_DEN_R             EQU 0x4006351C  ;GPIO Port M Direction Register
Address (Fill in these addresses)
GPIO_PORTM_DATA_R            EQU 0x400633FC  ;GPIO Port M Data Register Address
(Fill in these addresses)

COMBINATION EQU 2_0001
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Do not alter this section

        AREA    |.text|, CODE, READONLY, ALIGN=2 ;code in flash ROM
        THUMB                                    ;specifies using Thumb
instructions
        EXPORT Start

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;Function PortN_Init
```

```
PortN_Init
    ;STEP 1
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0,R0, #0x1000
    STR R0, [R1]
    NOP
    NOP

    ;STEP 5
    LDR R1, =GPIO_PORTN_DIR_R
    LDR R0, [R1]
    ORR R0,R0, #0x3
    STR R0, [R1]

    ;STEP 7
    LDR R1, =GPIO_PORTN_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0x3
    STR R0, [R1]
    BX  LR

PortM_Init
    ;STEP 1
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0,R0, #0x800
    STR R0, [R1]
    NOP
    NOP

    ;STEP 5
    LDR R1, =GPIO_PORTM_DIR_R
    LDR R0, [R1]
    ORR R0,R0, #0x00
    STR R0, [R1]

    ;STEP 7
    LDR R1, =GPIO_PORTM_DEN_R
    LDR R0, [R1]
    ORR R0, R0, #0x05

    STR R0, [R1]
    BX  LR


State_Init LDR R5,=GPIO_PORTN_DATA_R        ;Locked is the Initial State
           MOV R4,#2_00000001
           STR R4,[R5]
           BX LR

Start
    BL  PortN_Init
    BL  PortM_Init
    BL  State_Init                  ; initial state is locked_state
    LDR R3, =COMBINATION                    ;R3 stores our combination
```

```
Step1
    ; get the first digit
    AND R4, R3, #2_00001000
    LSR R4, R4, #0x3

    ; R2 is the input
    LDR R0, = GPIO_PORTM_DATA_R
    LDR R1,[R0]
    AND R2,R1,#2_00000001

    ; keep track of the status of the load button
    MOV R5, R6 ; R5 keep track of the status of the load button in the previous
loop
    LDR R6,[R0]
    AND R6, R6, #0x04; R6 keep track of the status of the load button in the
current loop

    CMP R5, R6 ; if they are of different values, then the user must have
pressed or released the button
    BEQ Step1
    CMPNE R6, #0x0 ; if the button is pressed rather than released, then we
compare the input
    BEQ Step1

    CMPNE R2,R4 ; compare the input
    BEQ Step2
    BNE Locked_State



Step2
    ; get the second digit
    AND R4, R3, #2_00000100
    LSR R4, R4, #0x2

    ; R2 is the input
    LDR R0, = GPIO_PORTM_DATA_R
    LDR R1,[R0]
    AND R2,R1,#2_00000001

    ; keep track of the status of the load button
    MOV R5, R6 ; R5 keep track of the status of the load button in the previous
loop
    LDR R6,[R0]
    AND R6, R6, #0x04 ; R6 keep track of the status of the load button in the
current loop

    CMP R5, R6 ; if they are of different values, then the user must have
pressed or released the button
    BEQ Step2
    CMPNE R6, #0x0 ; if the button is pressed rather than released, then we
compare the input
    BEQ Step2

    CMPNE R2,R4 ; compare the input
    BEQ Step3
    BNE Locked_State



Step3
    ; get the third digit
    AND R4, R3, #2_00000010
    LSR R4, R4, #0x1

    ; R2 is the input
    LDR R0, = GPIO_PORTM_DATA_R
    LDR R1,[R0]
    AND R2,R1,#2_00000001

    ; keep track of the status of the load button
    MOV R5, R6 ; R5 keep track of the status of the load button in the previous
loop
    LDR R6,[R0]
    AND R6, R6, #0x4 ; R6 keep track of the status of the load button in the
current loop

    CMP R5, R6 ; if they are of different values, then the user must have
pressed or released the button
    BEQ Step3
    CMPNE R6, #0x0 ; if the button is pressed rather than released, then we
compare the input
    BEQ Step3

    CMPNE R2,R4 ; compare the input
    BEQ Step4
    BNE Locked_State



Step4
    ; get the fourth digit
    AND R4, R3, #2_00000001

    ; R2 is the input
    LDR R0, = GPIO_PORTM_DATA_R
    LDR R1,[R0]
    AND R2,R1,#2_00000001

    ; keep track of the status of the load button
    MOV R5, R6 ; R5 keep track of the status of the load button in the previous
loop
    LDR R6,[R0]
    AND R6, R6, #0x4 ; R6 keep track of the status of the load button in the
current loop

    CMP R5, R6 ; if they are of different values, then the user must have
pressed or released the button
    BEQ Step4
    CMPNE R6, #0x0 ; if the button is pressed rather than released, then we
compare the input
    BEQ Step4
    CMPNE R2,R4 ; compare the input
    BEQ Unlocked_State
    BNE Locked_State



Locked_State
    LDR R5,=GPIO_PORTN_DATA_R
    MOV R4,#2_00000001                      ; locked_state, LED D2 on
    STR R4,[R5]
    B Step1


Unlocked_State
    LDR R5, =GPIO_PORTN_DATA_R
    MOV R4,#2_00000010                      ; unlocked_state, LED D1 on
    STR R4, [R5]

    ; keep D1 on for some time
    LDR R1, =0x800000
delay
    NOP
    NOP
    SUBS R1,R1,#0x1
    BNE delay

    ; initialization
    MOV R4,#2_00000001                      ; turn LED D2 on
    STR R4,[R5]

    B Step1

    ALIGN
    END
```