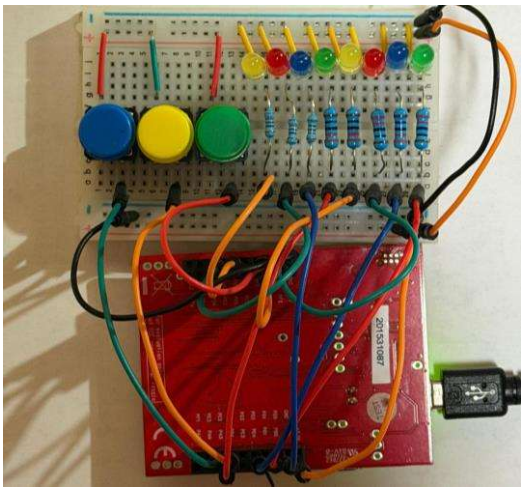
Wayne State University
College of Engineering — Department of Electrical and Computer Engineering

Introduction To Microcomputers (ECE3620)

Festive Christmas Lights

Abid Ahmad

Video: YouTube Link



Instructor: **Mohammed Alawad** **Date:** **11/19/25**

1. Introduction

The purpose of this project is to make a festive light controller on the TM4C microcontroller using ARM assembly. The system will consist of 3 switches, and each of them can start and stop a specific pattern. There will be 8 LEDs of assorted colors in Port B; patterns include right-to-left sweep, left-to-right sweep, and a random pattern. All three patterns will be controlled by 3 switches on Port E. The entire system was designed by following the System Development Life Cycle. Firstly, the constraints and requirements were analyzed (3 inputs/switches, 8 LEDs/outputs, 1 second delay). The system was built using simple assembly and a simple code structure, which includes things like a main loop, a pattern selector, and a frame indexer. The pre-made tables were stored during the coding process, which made it possible to display specific patterns. The specific 0.5-second timing delay was tested, and switch control was tested with robustness to make sure that the system meets all the requirements. One of the key things that was done during circuit building was common grounding. All pins were forced to share the same ground so that the signal

was correct and not distorted. Furthermore, it was ensured that all wires were connected to their designated pins. Wiring circuits to the wrong pins could cause a wrong display of the pattern, and input/output signals could be interpreted wrong. The system was designed using a modular format, which allowed easier debugging and efficient coding. Also, the modular format makes it easy for code blocks to be reused in other projects. Successive refinement was also employed, and each task was broken down into subtasks, and then those subtasks were broken into sub-subtasks, and all those things made system development easier and faster. The reports include a flowchart for logic control, a data-flow graph to show how the data flows, and the call graph to show how software and hardware modules interconnect with each other. A circuit diagram is also included for hardware documentation.

2. System Development Life Cycle

➢ Analyze the Problem

The goal of this project is to build a simple microcontroller system that can drive eight LEDS with three user-accessible switches. The core requirements of this project are to display right-to-left sweep, left-to-right sweep, and a pre-made random pattern for added holiday cheer. The system will loop the chosen pattern smoothly until it is turned off with the same switch that was used to start the pattern. The specifications are as follows: LEDs will be implemented on Port B (PB7-PB0), and switches will be on Port E (PE0-PE3). Both switches and LEDs are active high or positive logic. Each pattern is stored as a byte table in the main module, and they will be loaded sequentially after the user selects a pattern. The delay between frames is about 0.5 seconds, so that we can see the change in color and motion. Other constraints include using the ARM/Thumb, busy wait-delay, and frame/byte tables that will be used to create a pattern. Since each port is limited to 8 pins, multiple ports were used to facilitate everything. One other specific requirement is that the system will need to be implemented using the TM4C123GH6PM microcontroller, and breadboard.

➢ Design

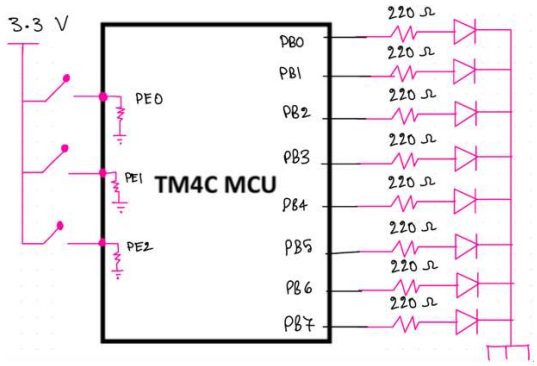
To make code structured and readable, I employed a table-driven design, which makes implementation easy. The main loop uses PE0 to PE2 and masks the lowest 3 bits. Based on the selected bit, the pattern is loaded by loading a base address and its length into registers. A routine call ‘runPattern’ is used with all patterns to display the specific behavior. Upon the pattern selection, the byte/frame tables get loaded into Port B, and as a result, the specific pattern is displayed. While running the pattern, the loop also checks the same switch to satisfy the stop requirement. The selected pattern will run until it is stopped using the same switch that was used to display the specific pattern. The pattern can but turned off at any time during the operation.

➢ Development (The flowchart, data flow diagram, and call graph are included at the end of this document)

For the development process, I used an ARM thumb assembly with multiple routines and labels to facilitate everything smoothly. The code was written in such a way that any person could match an instruction with a step in the flowchart. The initialization at the beginning of the code will

configure all the inputs in Port E and all the outputs in Port B. And all Port configurations were done in accordance with the hardware specification. The 3 patterns are stored in ROM in DCB (right to left, left to right, and Random). The loop will forever keep loading bytes, and it wraps when it reaches the end of the specified length. Furthermore, the 0.5s delay was implemented so that the outputs are human-readable. The delay keeps the pattern moving fast while making it noticeable to the users. To meet the stop requirement of the system, the loop simultaneously checks the input while displaying the outputs. And any non-zero result from that specific switch immediately stops the pattern, turns the LEDs off, and returns to the idle/rest state. The inputs in Port E used internal pull-up resistors, which means 0=not pressed, and 1=pressed. Also, the outputs in Port B were negative logic, and 1 was needed to drive the LED. These things were taken into consideration during the coding and configuration process.

Circuit Diagram (Switches and LEDs are both positive logic)



- **Testing**

The verification step of this project was most robust and intense since it is quite important that the developed system meets all the requirements and constraints. For functionality, pressing SW2 must start a right-to-left sweep, pressing SW2 must start to right sweep, and lastly, SW3 must display a random pattern. In each of the scenarios, pressing the same switch immediately stops the selected pattern. Pressing a different switch should not impact the pattern while it runs, and the time delays between each of the LEDs are about 0.5 seconds. A phone timer was used throughout the testing phase to make sure that the delay was 0.5 seconds, and the same-key stop logic was tested for all three switches. Everything was tested, and it was ensured that everything runs in accordance with the constraints and requirements. The system was also observed during continuous looping, and a correct wraparound plus delay was observed.

- **Deployment**

The developed model and the ready-to-deploy version were designed entirely so that it is easier for users. After loading the application in the microcontroller, pressing SW1 will display a right-to-left sweep. Pressing SW2 will display a left-to-right sweep, and pressing SW3 will display a random pattern. Pressing the same button forces the system to stop the pattern, turn all LEDs off,

and return to the idle state. Everything works correctly, and it follows all the specifications and requirements. The submission package will include all the assembly files so that the project and system can be redeveloped. Also, the flowchart, data-flow graph, and call graph were included for better analysis of the system. The circuit diagram also documents all hardware specifications. A short demo video is also included to show all the patterns and the same-key stop feature.

3. Conclusion

The final and deployable system is fully ready and meets all requirements. Every switch starts and stops a specific pattern, and all features and hardware went through robust testing to ensure they pass the requirements. Each switch will reliably start and stop its assigned pattern, and there will be a 0.5 interval or delay. The system checks the status of the same switch that the started pattern to meet the stop requirement. The system displays patterns in a stable manner, and there are no visible and known timing issues. It may be necessary to hit the switch a little hard so that it turns on. Proper grounding and power sources were provided to ensure the system runs as intended. The modularized design helped develop the system more efficiently, and it allowed subroutines to be reused. Future improvements include adding more switches and patterns so that the system can be used in a wide range of styles.

Wayne State University
College of Engineering — Department of Electrical and Computer Engineering

main_code

```
        AREA    DATA, READWRITE
halfSec  EQU     2666666 ;5333333              ; approximately 0.5s delay at ~16 MHz clock

        AREA    |.text|, CODE, READONLY
        IMPORT  PORTB_Init
                IMPORT  PORTB_Output
        IMPORT  PORTE_Init
                IMPORT  PORTE_Input
        THUMB
        EXPORT  Start

Start
        BL      PORTB_Init            ; initializes output pins of Port B (PB7-PB0) (LEDS)
        BL      PORTE_Init            ; initializes input  pins of Port E (PE2-PE0) (SWITCHES)

main
        ; switches are active-high
        BL      PORTE_Input           ; read the switches on(PE2-PE0)
        ANDS    R1, R0, #0x07         ; mask to only 3 bits; 001? or 010? 100?

        ; branch to appropriate label based on the switch selection
        CMP     R1, #1                ; PE0?
        BEQ     R2L
        CMP     R1, #2                ; PE1?
        BEQ     L2R
        CMP     R1, #4                ; PE2?
        BEQ     RANDOM
        B       main


; pattern selection
R2L
        MOVS    R6, R1 ; save the switch bit to check it is pressed again so that system can turn off
        LDR     R2, =R2L_pattern ; base address of the pattern
        MOVS    R4, #8 ; length of the pattern
        B       runPattern

L2R
        MOVS    R6, R1
        LDR     R2, =L2R_pattern
        MOVS    R4, #8
        B       runPattern

RANDOM
        MOVS    R6, R1
        LDR     R2, =RANDOM_pattern
        MOVS    R4, #16
        B       runPattern

runPattern
        MOVS    R3, #0   ; current counter value; i=0

forLoop
        CMP     R3, R4
        BLT     startloading
        MOVS    R3, #0    ; initializes with zero again so that it can start loading again

startloading
        LDRB    R0, [R2, R3]           ; R0 has the current byte of the pattern
        BL      PORTB_Output
        ; delay
        LDR     R0, =halfSec
        BL      delay
        ; check same key to stop
        BL      PORTE_Input
        ANDS    R5, R0, R6
        CMP     R5, #0  ; it checks the same switch again; if pressed, then stop the light
        BNE     patternDone
        ADDS    R3, R3, #1 ; if not pressed again; keep loading bytes
        B       forLoop

patternDone
        MOVS    R0, #0 ; turn off all lights
        BL      PORTB_Output

wait_and_check ; this is used to prevent stopping and starting the pattern on the same press
        BL      PORTE_Input
        ANDS    R5, R0, R6
        CMP     R5, #0
        BNE     wait_and_check    ; wait until not pressed
        B       main  ; return to rest state

;pre-made patterns
R2L_pattern  DCB 0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80
L2R_pattern  DCB 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01
RANDOM_pattern DCB 0x3C,0x12,0xA9,0x47,0x5E,0x90,0x2D,0x71
                                DCB 0xC3,0x18,0x6F,0x84,0x5A,0xE1,0x0F,0xB2
;------------delay------------
delay
        SUBS    R0, R0, #1
        BNE     delay
        BX      LR

        ALIGN
        END
```

Wayne State University
College of Engineering — Department of Electrical and Computer Engineering

PortE_driver.s

```
                AREA DATA, READWRITE
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_AFSEL_R EQU 0x40024420
GPIO_PORTE_PUR_R   EQU 0x40024510
GPIO_PORTE_DEN_R   EQU 0x4002451C
GPIO_PORTE_LOCK_R  EQU 0x40024520
GPIO_PORTE_CR_R    EQU 0x40024524
GPIO_PORTE_AMSEL_R EQU 0x40024528
GPIO_PORTE_PCTL_R  EQU 0x4002452C
GPIO_PORTE_PDR_R   EQU 0x40024514
GPIO_LOCK_KEY      EQU 0x4C4F434B  ; Unlocks the GPIO_CR register

        AREA    |.text|, CODE, READONLY
        THUMB
        EXPORT  PORTE_Init
        EXPORT  PORTE_Input

;------------PORTE_Init------------
PORTE_Init
    LDR R1, =SYSCTL_RCGCGPIO_R      ; 1) activate clock for Port E
    LDR R0, [R1]
    ORR R0, R0, #0x10
    STR R0, [R1]
    NOP
    NOP                             ; allow time for clock to finish
    LDR R1, =GPIO_PORTE_LOCK_R      ; 2) unlock the lock register
    LDR R0, =0x4C4F434B             ; unlock GPIO Port E Commit Register
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_CR_R        ; 3) enable commit for Port E
    MOV R0, #0xFF                   ; 1 means allow access
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_AMSEL_R     ; 4) disable analog functionality
    MOV R0, #0                      ; 0 means analog is off
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_PCTL_R      ; 5) configure as GPIO
    MOV R0, #0x00000000             ; 0 means configure Port E as GPIO
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_DIR_R       ; 6) set direction register
    MOV R0, #0x00                   ;PE2-PE0 are inputs
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_AFSEL_R     ; 7) regular Port Function
    MOV R0, #0                      ; 0 means disable alternate function
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_PDR_R       ; 8) pull-down resistors for port E
    MOV R0, #0x07                   ; all 3 switches will use internal pull-down resistors
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_PUR_R       ; pull-up resistors for port E are disabled
    MOV R0, #0x00
    STR R0, [R1]
    LDR R1, =GPIO_PORTE_DEN_R       ; 9) enable Port E digital port
    MOV R0, #0xFF                   ; 1 means enable digital I/O
    STR R0, [R1]
    BX  LR

;------------PORTE_Input------------
PORTE_Input
    LDR R1, =GPIO_PORTE_DATA_R ; pointer to Port E data
    LDR R0, [R1]               ; read all of Port E
    AND R0,R0,#0x07            ; just the input pins PE0, PE1 and PE2
    BX  LR                     ; return R0 with inputs
```

Port E was used for inputs, and internal pull-down resistors were used for the switches. The port was configured accordingly.

PortB_driver.s

```
        AREA DATA, READWRITE
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
GPIO_PORTB_DATA_R  EQU 0x400053FC
GPIO_PORTB_DIR_R   EQU 0x40005400
GPIO_PORTB_AFSEL_R EQU 0x40005420
GPIO_PORTB_PUR_R   EQU 0x40005510
GPIO_PORTB_DEN_R   EQU 0x4000551C
GPIO_PORTB_LOCK_R  EQU 0x40005520
GPIO_PORTB_CR_R    EQU 0x40005524
GPIO_PORTB_AMSEL_R EQU 0x40005528
GPIO_PORTB_PCTL_R  EQU 0x4000552C
GPIO_PORTB_PDR_R   EQU 0x40005514
GPIO_LOCK_KEY      EQU 0x4C4F434B  ; Unlocks the GPIO_CR register

        AREA    |.text|, CODE, READONLY
        THUMB
        EXPORT  PORTB_Init
                EXPORT  PORTB_Output

;------------PORTB_Init------------
PORTB_Init
    LDR R1, =SYSCTL_RCGCGPIO_R      ; 1) activate clock for Port B
    LDR R0, [R1]
    ORR R0, R0, #0x02               ; set bit1 to turn on clock
    STR R0, [R1]
    NOP
    NOP                             ; allow time for clock to finish
    LDR R1, =GPIO_PORTB_LOCK_R      ; 2) unlock the lock register
    LDR R0, =0x4C4F434B             ; unlock GPIO Port B Commit Register
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_CR_R        ; 3) enable commit for Port B
    MOV R0, #0xFF                   ; 1 means allow access
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_AMSEL_R     ; 4) disable analog functionality
    MOV R0, #0                      ; 0 means analog is off
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_PCTL_R      ; 5) configure as GPIO
    MOV R0, #0x00000000             ; 0 means configure Port B as GPIO
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_DIR_R       ; 6) set direction register
    MOV R0,#0xFF                   ; PB0-PB7 are outputs
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_AFSEL_R     ; 7) regular Port Bunction
    MOV R0, #0                      ; 0 means disable alternate function
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_PDR_R       ; 8) pull-down resistors for port B
    MOV R0, #0x00
    STR R0, [R1]
    LDR R1, =GPIO_PORTB_PUR_R       ; pull-up resistors for port B
    MOV R0, #0x00
    STR R0, [R1]
        LDR R1, =GPIO_PORTB_DEN_R       ; 9) enable Port B digital port
    MOV R0, #0xFF                   ; 1 means enable digital I/O
    STR R0, [R1]
    BX  LR

;------------PORTB_Output------------
PORTB_Output
    LDR R1, =GPIO_PORTB_DATA_R ; pointer to Port B data
    STR R0, [R1]               ; write to PB7-PB0
    BX  LR

    ALIGN                           ; make sure the end of this section is aligned
    END                             ; end of file
```

Port B was used for outputs, and external resistors were used for the LEDs. The port was configured accordingly.