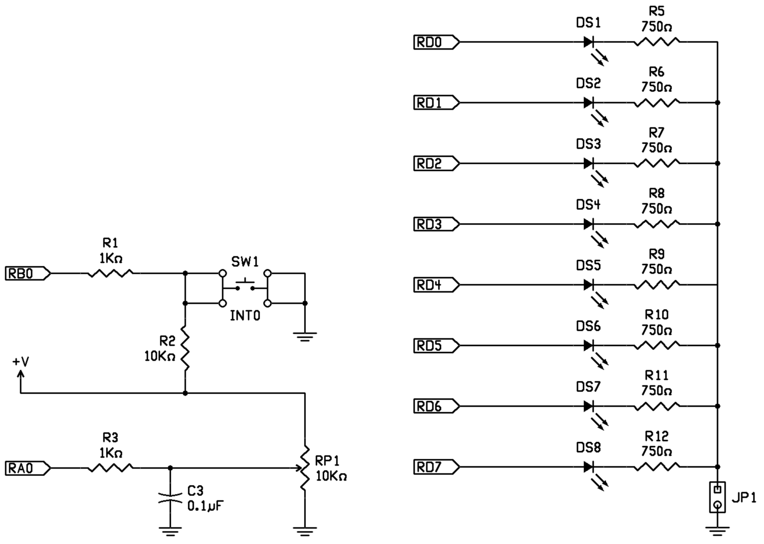
Tricycle Lights
Embedded System Design, Lab 6

Ben Lorenzetti

October 29, 2015

Contents



(a) Potentiometer Input (b) Output LEDs

Figure 1: I/O Circuit Diagrams from the 44–Pin Demo Board User's Manual

1 Objectives and Problem Description

1.1 Tricycle Lights

Blink two LEDs, representing left and right turn signals, depending on the position of a rotary potentiometer, which represents a steering wheel. The following conditions should be met.

1. Use the potentiometer and LEDs on the 44–Pin Demo Board.
2. Use the LED connected to RD7 for the left turn signal, and RD0 for right.
3. When wheel is in neutral position (wiper in middle of pot.), both turn indicator lights should be off.
4. When turned counterclockwise or clockwise from neutral position, the left or right LEDs should blink at a rate proportional to the angular displacement from neutral position.

2 Procedure

2.1 Potentiometer and LEDs on the 44–Pin Demo Board

In this lab, we will need to measure a potentiometer input (the steering wheel) and blink two LEDs for output. The 44–pin demo board for the PIC16F887 has all of this hardware on board. The potentiometer and LEDs are connected to pins as shown in the circuit diagram in figure 1.

According to the specifications, the LED at RD7 should be the left blinker and the LED at RD0 should be the right blinker.

2.2 PIC16F887 Analog to Digital Converter (ADC)

2.3 PIC16F887 Clock Sources

2.4 Implementation Flowchart

2.5 Delay Function

2.6 Linear Mapping

3 Expected Results

4 Experiment and Design Revisions

4.1 Command Line Assembly

My `.asm` source files were assembled on the command line so please do this if they don't compile nicely in the IDE. On Ubuntu, with the default MPLAB installation location, from the directory containting `lfsr.asm`, the commands are:

```
$ cp /opt/microchip/mplabx/v3.10/mpasmx/p16f877.inc ./p16f887.inc
$ /opt/microchip/mplabx/v3.10/mpasmx/mpasmx -p16f887 lfsr.asm
$ more lfsr.ERR
```

5 Observations

Figures/tricycle-lights-demo.jpg

Figure 2: Demonstration of Tricycle Turn Lights

6 Discussion

7 Tricycle Lights Implementation Code

```
; asg.asm
; 3–LFSR Alternating Step Generator on PIC Development Board
; Ben Lorenzetti
; Embedded Systems Design, Fall 2015

#include <p16f887.inc>
        __CONFIG        _CONFIG1, _LVP_OFF & _FCMEN_OFF & _IESO_OFF & _BOR_OFF
            & _CPD_OFF & _CP_OFF & _MCLRE_OFF & _PWRTE_ON & _WDT_OFF &
            _INTRC_OSC_NOCLKOUT
        __CONFIG        _CONFIG2, _WRT_OFF & _BOR21V

;————————————LFSR Bit/Byte Sizes, Tap Locations, and Initial Values
    ————————————;
#define LFSR_SIZE 2              ; LFSR byte size
#define LFSR0_TAP_MASK0 0x02     ; 14−bit LFSR lower byte tap mask
#define LFSR0_TAP_MASK1 0x38     ;   the high byte tap mask
#define LFSR1_TAP_MASK0 0x00     ; 15−bit LFSR lower byte tap mask
#define LFSR1_TAP_MASK1 0x60     ;   the low byte tap mask
#define LFSR2_TAP_MASK0 0x00     ; 16−bit LFSR lower byte tap mask
#define LFSR2_TAP_MASK1 0xB4     ;   the lower byte of tap mask
#define INITIAL_VALUE   1

;————————————————————————Organize Program Memory————————————————————————;
Reset_Vector
        ORG 0
        GOTO Initialize

Interupt_Vector
        ORG 4

;————————————————————Allocate Static Variables————————————————————————;
        cblock  0x20
        OUTER_DELAY
        MIDDLE_DELAY
        INNER_DELAY
        REMAINDER_DELAY
        PROP_SIGNAL
        BIT_INDEX
        ASG
        COUNTER
        LOCAL_LFSR: LFSR_SIZE
        LFSR0: LFSR_SIZE
        LFSR1: LFSR_SIZE
        LFSR2: LFSR_SIZE
        endc

;———————— void rotate_word_left (word*, word_size); ————————————————;
RLF_Word        MACRO   Word_Addr, Word_Size
        local i = 0
        while i < Word_Size
          RLF   (Word_Addr + i), 1
          i += 1
        endw
        ENDM

#define OUTER_MAX_PLUS_1 60
#define MIDDLE_MAX_PLUS_1 40
#define INNER_MAX_PLUS_1 40
#define REMAINDER 1
;-------------------------------- void Pause_1_Second () ----------------------------------------;
Pause_1_Second
        MOVLW   OUTER_MAX_PLUS_1
        MOVWF   OUTER_DELAY
        MOVLW   REMAINDER
        MOVWF   REMAINDER_DELAY
Inner_Loop
        DECFSZ  INNER_DELAY, f
        GOTO    Inner_Loop
        MOVLW   INNER_MAX_PLUS_1
        MOVWF   INNER_DELAY
Middle_Loop
        DECFSZ  MIDDLE_DELAY, f
        GOTO    Inner_Loop
        MOVLW   MIDDLE_MAX_PLUS_1
        MOVWF   MIDDLE_DELAY
Outer_Loop
        DECFSZ  OUTER_DELAY, f
        GOTO    Inner_Loop
Remainder_Loop
        DECFSZ  REMAINDER_DELAY, f
        GOTO    Remainder_Loop
        RETURN

;--------------------Linear XOR Cascade Function------------------------------------------------;
Linear_Xor_Function     MACRO   LFS_Register, Reg_Size, Tap_Mask
        ;------ Pass the function arguments by Copy------------------ ------;
        local i = 0
        while i < LFSR_SIZE
         MOVFW  (LFS_Register + i)      ; copy the LSFR to accumulator
         ANDLW  Tap_Mask#v(i)           ; AND accum. copy with TAP_MASK
         MOVWF  (LOCAL_LFSR + i)        ; move from accum. to local function
            variable
          i += 1
        endw
        ;------------------- Initialize Local Variables --------------------------------;
        CLRF    PROP_SIGNAL             ; initialize XOR signal to 0
        MOVLW   (8 * Reg_Size)          ; start XOR adding at most significat
           bit
        BCF     STATUS, C               ; set initial XOR add input to 0
Xor_Propagation_Loop#v(LFS_Register)
        ;------------------- rotate 1 bit from local copy of LSFR -----------------;
        RLF_Word        LOCAL_LFSR, LFSR_SIZE
        ;------------------- Add Carry Bit to PROP_SIGNAL ----------------------------;
        MOVFW   STATUS          ; get status register for the carry bit
        ANDLW   1               ; keep only the carry bit
        ADDWF   PROP_SIGNAL, 1  ; add carry bit to prop signal
        DECFSZ  BIT_INDEX, 1      ; break loop after all bits propagated
        GOTO    Xor_Propagation_Loop#v(LFS_Register)
;———————Return (1s—Digit of PROP_SIGNAL) via Carry Flag ———————————;
        RRF     PROP_SIGNAL, 1
        ENDM

;——————————————————Initialize Data Memory——————————————————————————;
Initialize
        ;——————————— Initialize I/O
        BANKSEL TRISD             ; select Register Bank 1
        CLRF    TRISD             ; make Port D all output pins
        BANKSEL PORTD             ; back to Register Bank 0
        CLRF    PORTD
        ;——————————Initialize Memory———————————————————————————————;
        MOVLW   INITIAL_VALUE     ; initialize LFSRs to INITIAL_VALUE
        MOVWF   LFSR0
        MOVWF   LFSR1
        MOVWF   LFSR2
        MOVLW   8                 ; initialize COUNTER to 8
        MOVWF   COUNTER
        CLRF    ASG               ; initialize ASG to 0


;——————————————Begin Main Program Loop——————————————————————————————;
Main
        ;———————————————————Cycle LFSR2————————————————————————————;
        Linear_Xor_Function      LFSR2, LFSR_SIZE, LFSR2_TAP_MASK
        RLF_Word          LFSR2, LFSR_SIZE
        ;—————————Test Carry Bit and Cycle Appropriate Register—————;
        BTFSS   STATUS, C         ; skip next instruction if Carry=1
        GOTO    Cycle_LFSR0
        GOTO    Cycle_LFSR1
        ;————————————Rotate Result into ASG—————————————————————————;
Rotate_C_Into_ASG
        RLF     ASG, 1
        ;————————————————Decrement and Test Counter——————————————————;
        DECFSZ  COUNTER, 1        ; decrement counter, skip next if 0
        GOTO    Main
        ;——————————Reset Counter, Update Display, & Delay———————————;
        MOVLW   8
        MOVWF   COUNTER           ; set counter to 8
        MOVFW   ASG
        MOVWF   PORTD             ; write ASG to the LEDs
        CALL    Pause_1_Second    ; pause remainder of 1 second
        ;————————————————End of Main Loop———————————————————————————;
        GOTO    Main

;——————————————Cycling of the two subjected LFSRs——————————————————;
Cycle_LFSR0
        Linear_Xor_Function      LFSR0, LFSR_SIZE, LFSR0_TAP_MASK
        RLF_Word          LFSR0, LFSR_SIZE
        GOTO    Rotate_C_Into_ASG         ; return to main program
Cycle_LFSR1
        Linear_Xor_Function      LFSR1, LFSR_SIZE, LFSR1_TAP_MASK
```

```
RLF_Word          LFSR1, LFSR_SIZE
GOTO     Rotate_C_Into_ASG        ; return to main program

END
```