ECE 367 -Experiment #1 Fall 2012

Due at the beginning of lab during week 3 (9/1/2012)

Introduction

The goal of this experiment is the acquaint you with the Technological Arts nanocore12 microcontroller development system, and the procedures used to assemble/download/execute code on this processor. Specifically, you will do what is needed to have the microcontroller independently execute code that blinks some LED's. To do this you will need to install a program on a Windows PC and have access to its serial port.

Required Hardware and Software

1. nanocore starter kit (UIC2) from Technological Arts
2. USB cable from Technological Arts
3. FreeScale CodeWarrior Special Edition: CodeWarrior for HCS12(X) Microcontrollers (Classic) (Rev 5.1) Go here to get it as explained in lab.

Procedure

1. Install the Codewarrior program.
2. Run CodeWarrior IDE and choose Getting Started Tutorial - Absolute Assembly Tutorial
3. Read the Quick Start Guide.

   Then follow the guide instructions to assemble and power the system. Do not load the suggested software.
4. Build the circuit in Figure 1. below.

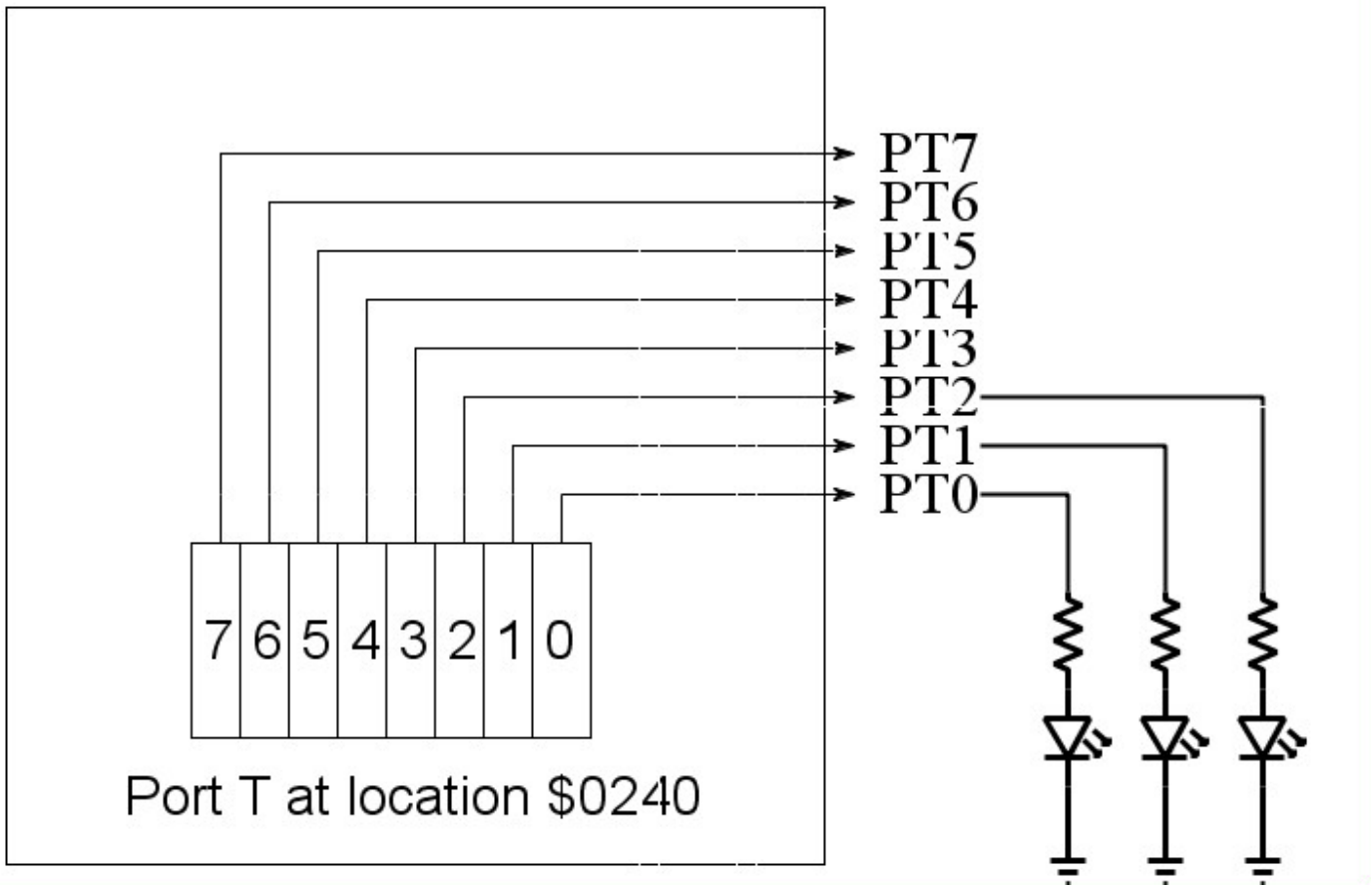

Figure 1.

You are attaching LED's with current limiting resistors to three pins of PortT. The resistors should be 470 Ohms or 1K.

5. Using CodeWarrior, assemble the counter code that is provided for you in the Appendix below. You may copy and paste from between the -- cut here -- symbols. Be sure to delete "pre" and "/pre" html tags.

   Save the code as **exp1.asm**

   By assembling the program, (using the Make option) an executable code having *.s19 extension should have been created>

6. Next, use the Dubug feature to load and test the code. To do this FIRST move the RUN/LOAD switch to LOAD and ground the RESET pin momentarily then run (click on) debug.

7. When the code is successfully downloaded move the switch on the controller from the LOAD position to the RUN position and Ground the RESET pin momentarily. The program should run and the LED's will appear to be a three-bit counter counting at one second intervals.

8. Congratulations! You now have a stand-alone computer that performs a simple function. Inspect the code and change the software delay loop constant

   LDY #0010 to LDY #0005. How does this affect the LED blinking rate?

Experiment 1 -Appendix: Three Bit LED Counter 68HC9S12 assembly language code

-- cut here --

```
; University of Illinois at Chicago, Dept. of Electrical and Computer Engineering
; ECE 367 -Microprocessor-Based Design
; Three Bit Counter
; Version 2, Aug 27, 2011
; By Robert A. Becker

; PAY ATTENTION TO THE ALIGNMENT BELOW
; Labels start in the first column  (left most column = colunm 1)
; OP CODES are at column 9
; COMMENTS follow a ";" symbol
; Blank lines are allowed (Makes the code more readable)

; Define symbolic constants
PortT   EQU     $240            ;Define Register Locations
DDRT    EQU     $242
INITRG  EQU     $11
INITRM  EQU     $10
CLKSEL  EQU     $39
PLLCTL  EQU     $3A
CRGFLG  EQU     $37
SYNR    EQU     $34
REFDV   EQU     $35
COPCTL  EQU     $3C
TSCR1   EQU     $46
TSCR2   EQU     $4D
TIOS    EQU     $40
TCNT    EQU     $44
TC0     EQU     $50
TFLG1   EQU     $4E
;
; The ORG statment below would normally be followed by variable definitions
; There are no variables needed for this project.
; THIS IS THE BEGINNING SETUP CODE
;
        ORG     $3800           ; Beginning of RAM for Variables
;
; The main code begins here. Note the START Label
;
        ORG     $4000           ; Beginning of Flash EEPROM
START   LDS     #$3FCE          ; Top of the Stack
        SEI                     ; Turn Off Interrupts
        MOVB    #$00, INITRG    ; I/O and Control Registers Start at $0000
        MOVB    #$39, INITRM    ; RAM ends at $3FFF
;
; We Need To Set Up The PLL So that the E-Clock = 24MHz
;
        BCLR CLKSEL,$80 ; disengage PLL from system
        BSET PLLCTL,$40 ; turn on PLL
        MOVB #$2,SYNR   ; set PLL multiplier
        MOVB #$0,REFDV  ; set  PLL divider
        NOP             ; No OP
        NOP             ; NO OP
PLP     BRCLR CRGFLG,$08,PLP ; while (!(crg.crgflg.bit.lock==1))
        BSET CLKSEL,$80 ; engage PLL
;
;
;
```

```
        CLI             ; Turn ON Interrupts
;
; End of  setup code. You will always need the above setup code for every experiment
;
        LDAA    #$FF    ; Make PortT Outbound
        STAA    DDRT
        LDAA    #$00    ; Start the count at 0
AG:     STAA    PortT   ; Output the count to PortT
        JSR     DELAY   ; Let's go wait one second
        INCA            ; Increment the count
        BNE     AG      ; Do again unless the count > 255
        STAA    PortT   ; If we get here the output will go blank
        BRA     *       ; Stop the program (Branch here forever)
;
; We use some built-in timer functions to create an accurate delay
;
DELAY   PSHA                    ; Save accumulator A on the stack
        LDY     #10             ; We will repeat this subroutine 10 times
        MOVB    #$90,TSCR1      ; enable TCNT & fast flags clear
        MOVB    #$06,TSCR2      ; configure prescale factor to 64
        MOVB    #$01,TIOS       ; enable OC0
        LDD     TCNT            ; Get current TCNT value
AGAIN   ADDD    #37500          ; start an output compare operation
        STD     TC0             ; with 100 ms time delay
WAIT    BRCLR   TFLG1,$01,WAIT  ; Wait for TCNT to catch up
        LDD     TC0             ; Get the value in TC0
        DBNE    Y,AGAIN         ; 10 X 100ms = 1 sec
        PULA                    ; Pull A
        RTS
;
; End of counter code

; Define Power-On Reset Interrupt Vector

; AGAIN - OP CODES are at column 9

        ORG $FFFE ; $FFFE, $FFFF = Power-On Reset Int. Vector Location
        FDB START ; Specify instruction to execute on power up

; End of Interrupt code

        END        ; (Optional) End of source code
```

-- cut here --

Last modified: Thur Jan 5 01:01:08 2012

ECE 367: Experiment 1

Semester: Fall 2012

Name: Kai Zhao

UIN: 670720413

Due Date: 2012 Sept 11

Lab Section: T11

TA: Chenjie Tang

```
  1: ; University of Illinois at Chicago, Dept. of Electrical and Computer Engineering
  2: ; ECE 367 -Microprocessor-Based Design
  3: ; Three Bit Counter
  4: ; Version 2, Sept 11, 2012
  5: ; By Kai Zhao
  6:
  7: ; PAY ATTENTION TO THE ALIGNMENT BELOW
  8: ; Labels start in the first column  (left most column = colunm 1)
  9: ; OP CODES are at column 9
 10: ; COMMENTS follow a ";" symbol
 11: ; Blank lines are allowed (Makes the code more readable)
 12:
 13: ; Define symbolic constants
 14: PortT  EQU  $240   ;Define Register Locations
 15: DDRT EQU  $242
 16: INITRG EQU  $11
 17: INITRM EQU  $10
 18: CLKSEL EQU  $39
 19: PLLCTL EQU  $3A
 20: CRGFLG EQU  $37
 21: SYNR EQU  $34
 22: REFDV  EQU  $35
 23: COPCTL EQU  $3C
 24: TSCR1  EQU  $46
 25: TSCR2  EQU  $4D
 26: TIOS EQU  $40
 27: TCNT EQU  $44
 28: TC0  EQU  $50
 29: TFLG1  EQU  $4E
 30: ;
 31: ; The ORG statment below would normally be followed by variable definitions
 32: ; There are no variables needed for this project.
 33: ; THIS IS THE BEGINNING SETUP CODE
 34: ;
 35:   ORG  $3800    ; Beginning of RAM for Variables
 36: ;
 37: ; The main code begins here. Note the START Label
 38: ;
 39:   ORG  $4000    ; Beginning of Flash EEPROM
 40: START  LDS  #$3FCE   ; Top of the Stack
 41:   SEI      ; Turn Off Interrupts
 42:         MOVB  #$00, INITRG  ; I/O and Control Registers Start at $0000
 43:   MOVB #$39, INITRM  ; RAM ends at $3FFF
 44: ;
 45: ; We Need To Set Up The PLL So that the E-Clock = 24MHz
 46: ;
 47:   BCLR CLKSEL,$80 ; disengage PLL from system
 48:   BSET PLLCTL,$40 ; turn on PLL
 49:   MOVB #$2,SYNR   ; set PLL multiplier
 50:   MOVB #$0,REFDV  ; set  PLL divider
 51:   NOP    ; No OP
 52:   NOP    ; NO OP
 53: PLP  BRCLR CRGFLG,$08,PLP ; while (!(crg.crgflg.bit.lock==1))
 54:   BSET CLKSEL,$80 ; engage PLL
 55: ;
 56: ;
 57: ;
 58:   CLI    ; Turn ON Interrupts
 59: ;
 60: ; End of  setup code. You will always need the above setup code for every experiment
 61: ;
 62:   LDAA #$FF ; Make PortT Outbound
 63:   STAA DDRT
 64:   LDAA #$00 ; Start the count at 0
 65: AG:  STAA PortT  ; Output the count to PortT
 66:   JSR  DELAY  ; Let's go wait one second
 67:   INCA   ; Increment the count
 68:   BNE  AG; Do again unless the count > 255
 69:   STAA PortT  ; If we get here the output will go blank
 70:   BRA  * ; Stop the program (Branch here forever)
 71: ;
 72: ; We use some built-in timer functions to create an accurate delay
 73: ;
 74: DELAY  PSHA     ; Save accumulator A on the stack
 75:   LDY  #10    ; We will repeat this subroutine 10 times
 76:   MOVB #$90,TSCR1 ; enable TCNT & fast flags clear
 77:   MOVB #$06,TSCR2 ; configure prescale factor to 64
 78:   MOVB #$01,TIOS  ; enable OC0
```

```
 79:    LDD  TCNT   ; Get current TCNT value
 80: AGAIN  ADDD #37500   ; start an output compare operation
 81:    STD  TC0    ; with 100 ms time delay
 82: WAIT BRCLR  TFLG1,$01,WAIT  ; Wait for TCNT to catch up
 83:    LDD  TC0    ; Get the value in TC0
 84:    DBNE Y,AGAIN    ; 10 X 100ms = 1 sec
 85:    PULA     ; Pull A
 86:    RTS
 87: ;
 88: ; End of counter code
 89:
 90: ; Define Power-On Reset Interrupt Vector
 91:
 92: ; AGAIN - OP CODES are at column 9
 93:
 94:         ORG $FFFE ; $FFFE, $FFFF = Power-On Reset Int. Vector Location
 95:         FDB START ; Specify instruction to execute on power up
 96:
 97: ; End of Interrupt code
 98:
 99:         END        ; (Optional) End of source code
```