FIBONACCI EVALUATION EXERCISE FOR CS2

PAUL COCKSHOTT

Your second exercise is to develop a program that will cause the PIC to evaluate the Fibonacci series to an accuracy of 8 bits.

The Fibonacci series is 1, 1, 2, 3, 5, 8, 13 ....etc

On being initialised the board should display the number 1 in binary on the display LEDs.

Each time the button on the board is pressed the next member of the series should be displayed. The number in the counter should be output as a binary pattern of on and off lights on the 8 LEDs.

You are provided with a skeleton program. This program provides a subroutine that can be called to check if the button has been pressed and which will increment a counter register if it has. You are also provided with 2 sample routines that will switch on LEDs D0 and D1 on the board. You will have to write routines to operate the other LEDs.

The program also contains an initialization sequence that will set up the registers correctly before your program starts.

The exercise will be for 10 marks and will be submitted for marking in the labs during the first week in November. You must demonstrate the program working on your board to your lab supervisor and hand in a report that contains a printing of the listing file produced by running the assembler on your program (has the suffix.lst). You should also provide a one page description of how your program works.

You can get a total of 10 marks for this exercise made up as follows:

Topic	Marks
d0 works	1
d1 works	1
d2..d7 work	3
Your control structure	2
Your data declarations	1
Your documentation	2
total	10

SAMPLE PROGRAM WITH BLANKS TO FILL IN

```
;****************************************************
;Software License Agreement
;
;The software supplied herewith by Microchip Technology
;Incorporated (the "Company") is intended and supplied to you, the
;Companys customer, for use solely and exclusively on Microchip
;products. The software is owned by the Company and/or its supplier,
;and is protected under applicable copyright laws. All rights are
;reserved. Any use in violation of the foregoing restrictions may
;subject the user to criminal sanctions under applicable laws, as
;well as to civil liability for the breach of the terms and
;conditions of this license.
;

;****************************************************
;Filename: counter.asm
```

```
;Author: Reston Condit
;Date: 1/15/03
;Version: 1.00
;Description:
;

;Revision History:
;Modified Paul Cockshott, oct 04, Nov 05
; This firmware implements a fibonacci series that displays its results on the leds
; the counter is toggled by the button
; it was derived from a file provided by Microchip and written by R Condit
;**************************************************
;**************************************************
;Instructions On How To Use This Program
;**************************************************
; Press Switch 1 (SW1) on the PICkit(tm) demonstration board to cycle through
; the 256 LED states.
;**************************************************
list p=12f629 ; list directive to define processor
#include <p12f629.inc> ; processor specific variable definitions
errorlevel -302 ; suppress message 302 from list file
__CONFIG _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_ON & _PWRTE_ON & _INTRC_OSC_NOCLKOUT
; '__CONFIG' directive is used to embed configuration word within .asm file.
; The labels following the directive are located in the respective .inc file.
; See data sheet for additional information on configuration word settings.
;****************** VARIABLE DEFINITIONS *******************
; this reserves a pair of registers starting at
; register 20 Hex
cblock 0x20
STATE_LED ; LED state
STATE_DEBOUNCE ; button debounce state machine counter
; ---------------------------
; declare more variables here if you need them
endc
;****************** DEFINE STATEMENTS *********************
; input and output definitions
#define POT GPIO,2 ; potentiometer (not used in this
; example)
#define SW1 GPIO,3 ; toggle switch
; define input/output designation for LEDs (what TRISIO will equal)
#define TRIS_D0_D1 B'00001111' ; TRISIO setting for D0 and D1
#define TRIS_D2_D3 B'00101011' ; TRISIO setting for D2 and D3
; You will have to insert definitions for the other
; combinations of tristates
; define LED state (what GPIO will equal)
#define D0_ON B'00010000' ; D0 LED
#define D1_ON B'00100000' ; D1 LED
; You will have to insert definitions for the other
; bit patterns needed.
;******************** Start of Program *******************
org 0x000 ; processor reset vector
goto Initialize
;**************************************************
; Initialize
; Initialize Special Function Registers
;**************************************************
org 0x005 ; Start of Programm Memory Vector
Initialize

 bsf STATUS,RP0 ; Bank 1
 movwf OSCCAL ; update register with factory cal
 ; value
 movlw B'00111111' ; Set all I/O pins as inputs
 movwf TRISIO
 movlw B'10000001' ; Weak pullups: disabled
 movwf OPTION_REG ; TMR0 prescaler: 1:64 (TMR0 will
 ; overflow in 10.6ms)
 clrf INTCON ; disable all interrupts, clear all
 ; flags
 bcf STATUS,RP0 ; Bank 0
 clrf GPIO ; clear all outputs
 clrf TMR0 ; clear Timer 0
 clrf STATE_LED ; clear LED state machine counter
 clrf STATE_DEBOUNCE ; clear debounce state machine counter
;****************************************************
; Main Loop
; Implements a state machine that lights up the LEDs on the PICkit board
; sequentially as a counter when SW1 is pressed.
;****************************************************
MainLoop
clrwdt ; clear Watch Dog Timer
; Your main program goes here
;--------------------------
```

Put your very own code here.

```
;-----------------------------------
goto MainLoop
;****************************************************
; Output routines
BitOn0
 ; Turns on D0 LED
 bsf STATUS, RP0 ; Bank 1
 movlw TRIS_D0_D1 ; move predefined value to TRISIO
 movwf TRISIO
 bcf STATUS, RP0 ; Bank 0
 movlw D0_ON ; move predefined value to GPIO
 movwf GPIO
 retlw 0 ; go back to main loop
BitOn1
 ; Turns on D1 LED
 bsf STATUS, RP0 ; Bank 1
 movlw TRIS_D0_D1 ; move predefined value to TRISIO
 movwf TRISIO
 bcf STATUS, RP0 ; Bank 0
 movlw D1_ON ; move predefined value to GPIO
 movwf GPIO
 retlw 0 ; go back to main loop
BitOn2
```

You must write routines for BitOn2, BitOn3 etc.

```
;*************************************
; Fibstep
;  do one step of the fibonacci series
;*************************************
fibstep
;--------
; your code goes here
;****************************************************
; Button_Press
; Looks for button press and implements a button debounce routine.
;****************************************************
Button_Press
 ; Note that what follows is equivalent to a case statement in
 ; a high order language
 movf STATE_DEBOUNCE, w ; Mask out the high order bits of
 andlw B'00000011' ; STATE_DEBOUNCE.
 addwf PCL, f
 goto Debounce_1
 goto Debounce_2
 goto Debounce_3
 goto Debounce_2 ; Send to second state if noise
; corrupts debounce state counter.
Debounce_1
 btfsc SW1 ; Is Switch 1 pushed?
 retlw 0 ; No, then return
 incf STATE_DEBOUNCE, f ; Yes, then increment  state
                     ;machine

 call fibstep            ; increment fibonacci series
 retlw 0
Debounce_2
 btfss SW1 ; Is Switch 1 released?
 retlw 0 ; No, then return
 clrf TMR0 ; Yes, clear Timer0 and Timer0 flag
 bcf INTCON, T0IF
 incf STATE_DEBOUNCE, f ; Increment debounce state machine
 retlw 0
Debounce_3
; Switch must be high for approximately 10 ms before debounce state machine is
; re-initialized.
 btfss INTCON, T0IF ; Has 10.6 ms passed?
 goto Debounce_3a ; No, then check for switch jitter
 clrf STATE_DEBOUNCE ; Yes, then re-initialize state machine
 retlw 0
Debounce_3a
 btfss SW1 ; Is Switch 1 low again (due to switch
 ; jitter)?
 decf STATE_DEBOUNCE, f ; Yes, then go back to debounce state2
 retlw 0 ; No, then return.
end ; directive 'end of program'
```