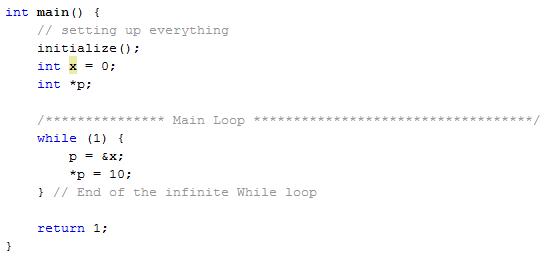
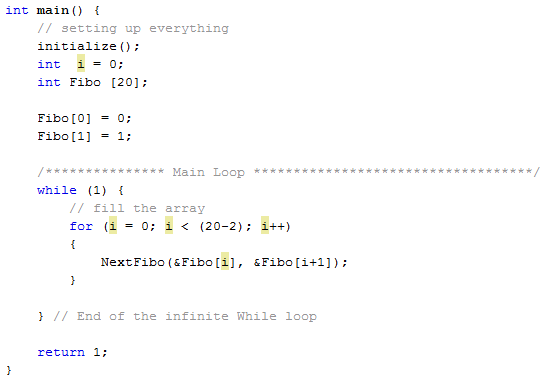
Quang Nguyen

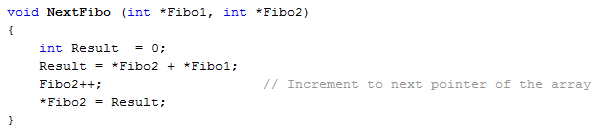
CPE 490 Homework # 3

1. Let **x** be an integer and **p** be an integer pointer. Write a code fragment that will assign the value 10 to x using pointer **p**.

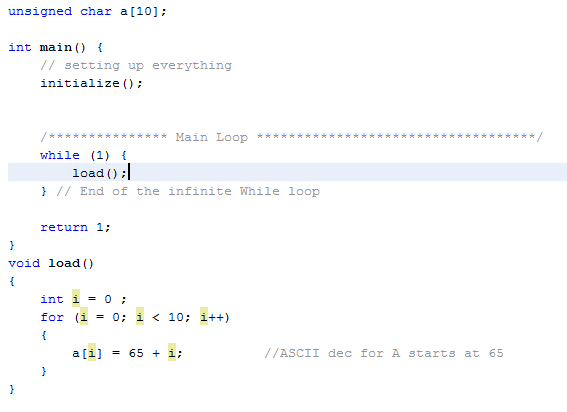


1. Write a C program that will put the first 20 Fibonacci numbers in an integer array **Fibo**. Your program must include a function **NextFibo** that will receive two integers and write the next Fibonacci number directly into that array using a passed pointer. Therefore, **NextFibo** returns a void.





1. Write a function called **load**() that loads a 10-byte character array **a,** with the letters ‘A’ through ‘J’. You must use some type of loop in the function (do not write code that has 10 separate line to load the array).



1. How many bytes of memory will the following arrays need (assume we are writing code with C30 compiler)?

Needs 4 bytes for every 3 bytes of info used. 16 bytes addressable but need that dummy byte.

* 1. char s[80];

80/3 = 27. 27 + 80 = 107 bytes

* 1. char s[80] [10];

(80\*10)/3 = 267 800 + 267 = 1067 bytes

* 1. int n[10];

2 bytes = 1 int

10 \* 2 = 20 20 bytes

* 1. float f[10] [5];

float = 4 bytes

10 \* 5 \* 4 = 200 200 bytes

* 1. char x[10] [9] [8] [7];

same reasoning as a

(10\*9\*8\*7) = 5040 5040/3 = 1680

5040 + 1680 = 6720 6720 bytes

1. You are to write two interrupt routines. Your timing base will be based on a 32 MHZ primary oscillator (just like lab). The first interrupt routine will be when INT1 on pin 18 of the dsPIC33FFJ256GP710A goes from high to low (negative edge). When this occurs the first time you will start timer 1 counting. When this interrupt occurs 4 more times you will record the time it has taken. Realizing that each negative edge signifies a shaft turning once you will calculate the RPM of the shaft and put this value into a global variable. The interrupt will then repeat this calculation on the next 4 counts indefinitely. All calculations can be done in the interrupt.

Next write and interrupt that will have higher priority than the previous interrupt. This interrupt will occur when a new value of RPM is not calculated in .5 seconds since the last new value. When this interrupt activates a global variable called RPMAlarm will be set to a non-zero answer and the previous interrupt will be disabled.

Your homework answer should show both of the interrupt routines, using the proper compiler directives, and the initialization code for the interrupts, and for TIMER1. Please declare the global variables that you will need assigning them a proper data type.

Also answer the following:

1. What is the slowest RPM reading?
2. What is the fastest RPM reading?

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Configuration Bits

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <xc.h>

// FOSCSEL

#pragma config FNOSC = PRIPLL // Oscillator Mode (Primary Oscillator (XT, HS, EC) w/ PLL)

#pragma config IESO = ON // Two-speed Oscillator Start-Up Enable (Start up with FRC, then switch)

// FOSC

#pragma config POSCMD = XT // Primary Oscillator Source (XT Oscillator Mode)

#pragma config OSCIOFNC = OFF // OSC2 Pin Function (OSC2 pin has clock out function)

#pragma config FCKSM = CSDCMD // Clock Switching and Monitor (Both Clock Switching and Fail-Safe Clock Monitor are disabled)

// FWDT

#pragma config FWDTEN = OFF // Watchdog Timer Enable (Watchdog timer enabled/disabled by user software)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Library includes

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <p33FJ256GP710A.h>

#include <libq.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Constant Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Global Variable Declarations

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Function Prototype

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize();

volatile int Int1Counter = 0;

volatile int Timer = 0;

volatile int RPM = 0;

volatile int T6Counter = 0;

volatile int RPMAlarm = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Main Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

// setting up everything

initialize();

while (1) {

;

} // End of the infinite While loop

return 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Initialize Function

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void initialize() {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Setting up for Clock (PLL, M, N1, N2)

\* for 32 MHz and Fcy = 16 MHz

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Fosc = Fin(M/(N1\*N2)) = 8 MHz (32/(2\*4)) = 32 MHz

PLLFBD = 30; // M = 32

// N1 default is 2

// N2 default is 4

// Fcy = Fosc/2 by default

// Setting up Timer 1 module

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Internal Clock = 16 MHz

\* 1/16 MHz = 0.0625 us

\* with prescaler at 1

\* 0.0625 us \* 1 = 0.0625 us

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

T1CONbits.TCS = 0; // Use internal clock

T1CONbits.TCKPS = 0; // Set prescaler to 1

// Set initial value for PR1 to throw an interrupt if the

// prescaller is too smaller and the timer is overflowing

PR1 = 65535;

TMR1 = 0; // Clear TMR1

\_T1IE = 1; // Turn on TMR1 interrupt

// The default priority is 4 so lets not worry about it for now

// Setting up Int1 on

INTCON2bits.INT1EP = 1; // Set INT 1 on negative edge

IEC1bits.INT1IE = 1; // Enable INT1 interrupt

// Setting up TMR6

// Setting it to 32 bits mode and also leave it at default priority

// since it has higher priority anyway

T6CONbits.TCKPS = 2; // prescaler of 64

T6CONbits.T32 = 1; // Make it a 32 bits timer

T6CONbits.TCS = 0;

TMR6 = 0;

PR6 = 125000; // Overflow after 0.5 sec

\_T6IE = 1; // Turn on interrupt

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Interrupt Service Routine

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void \_\_attribute\_\_ ((interrupt, no\_auto\_psv)) \_T1Interrupt (void)

{ //

// Turn off flag //

IFS0bits.T1IF = 0; //

// Code should go in here to indicate that we need to set the prescaler

// higher.

}

// ISR for INT1

void \_\_attribute\_\_ ((interrupt, no\_auto\_psv)) \_INT1Interrup (void)

{ //

// Turn off flag //

IFS1bits.INT1IF = 0; //

if (Int1Counter < 4)

{

Timer = TMR1; // temporary store the time

RPM = Timer \* (0.0000000625) // Convert to second

// I just didn't time to make this nicer

TMR1 = 0;

Int1Counter = 0;

// RPM = 1 Rev / 1 minute = ( 4 Rev / sec ) \* ( 60 sec / 1 minute)

RPM = (240 / RPM);

}

else

Int1Counter++;

}

// ISR for TMR6

void \_\_attribute\_\_ ((interrupt, no\_auto\_psv)) \_T6Interrupt (void)

{ //

// Turn off flag //

IFS2bits.T6IF = 0;

// Set RPMAlarm to some non-zero value

RPMAlarm = 0xFF;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Other Functions

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/