Github:

**Ejercicio 1**

Escriba un código que muestre un contador en el puerto A que incremente cada segundo.

Configure el oscilador interno a 4MHz.

**HDCT.c**

/\*

\* File: HDTC.c

\* Author: Dina Rodríguez

\*

\* Created on 1 de abril de 2022, 11:03 PM

\*/

// CONFIG1

#pragma config FOSC = INTRC\_NOCLKOUT// Oscillator Selection bits (INTOSCIO oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)

#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)

#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)

#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)

// CONFIG2

#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#include <stdio.h>

#include "setup.h"

#define \_XTAL\_FREQ 4000000 //configuracion 4MHz

/\*------------------------------------------------------------------------------

\* CONSTANTES

------------------------------------------------------------------------------\*/

#define INCREMENTAR PORTBbits.RB0 // Asignamos un alias a RB0

#define DECREMENTAR PORTBbits.RB1 // Asignamos un alias a RB1

/\*----------VARIABLES----------------------------

// Ejemplos:\*/

uint8\_t var; // Solo declarada

// uint8\_t var2 = 0; // Declarada e inicializada

/\*------------------------------------------------\*/

/\* -----------PROTOTIPO DE FUNCIONES

//void setup(void);

\*/

/\*-------INTERRUPCIONES -----------------\*/

void \_\_interrupt() isr (void){

if(INTCONbits.RBIF){ // Fue interrupci n del PORTB�

if(!INCREMENTAR){ // Verificamos si fue RB0 quien gener la �interrupci n�

//PORTA++; // Incremento del PORTC (INCF PORTC)

}

if(!DECREMENTAR){ // Verificamos si fue RB0 quien gener la �interrupci n�

//PORTA--; // Incremento del PORTC (INCF PORTC)

}

INTCONbits.RBIF = 0; // Limpiamos bandera de interrupci n�

}

if(INTCONbits.T0IF){ //se revisa bandera del timer

resettmr0();

//PORTC++;

}

if(PIR1bits.TMR1IF){

resettmr1();

var++;

if (var%2==0) PORTA++;

}

return;

}

void main(void) {

setup(); // Llamamos a la funci n de configuraciones�

setuptmr0(); // Llamamos a la funci n de configuraciones�

setuptmr1();

configint();

while(1){

}

return;

}

**setup.c**

/\*

\* File: setup.c

\* Author: Dina

\*

\* Created on 4 de abril de 2022, 12:53 AM

\*/

#include <xc.h>

/\*------------------------------------------------------------------------------

\* CONFIGURACION

------------------------------------------------------------------------------\*/

void setup(void){

ANSEL = 0;

ANSELH = 0b00000000; // Usaremos solo I/O digitales

TRISA = 0x00; // PORTA como salida

PORTA = 0; // Limpiamos PORTA

TRISC = 0x00; // PORTC como salida

PORTC = 0; // Limpiamos PORTC

//TRISB = 1; // RB0 como entrada (configurada en decimal)

//TRISB = 0b00000001; // RB0 como entrada (configurada con binario)

TRISBbits.TRISB0 = 1; // RB0 como entrada (configurada con control de bits)

TRISBbits.TRISB1 = 1; // RB1 como entrada (configurada con control de bits)

OPTION\_REGbits.nRBPU = 0; // Habilitamos resistencias de pull-up del PORTB

WPUBbits.WPUB0 = 1; // Habilitamos resistencia de pull-up de RB0

WPUBbits.WPUB1 = 1; // Habilitamos resistencia de pull-up de RB1

INTCONbits.RBIE = 1; // Habilitamos interrupciones del PORTB

IOCBbits.IOCB0 = 1; // Habilitamos interrupci n por cambio de estado para RB0

IOCBbits.IOCB1 = 1; // Habilitamos interrupci n por cambio de estado para RB1

INTCONbits.RBIF = 0; // Limpiamos bandera de interrupci n�

//configuracion del reloj

OSCCONbits.IRCF = 0b0110; // 4MHz

OSCCONbits.SCS = 1; // Oscilador interno

}

// Td = Pre \* TMR1\*Ti

// N = 65536-(Td/Pre\*Ti)

//Ttmr1if = Prescaler \* PR2 \* Postscaler \* (1/(Fosc/4))

//PR2 = Ttmr2if/Prescaler \* Postscaler \* (1/(Fosc/4))

void resettmr0(void){

TMR0 = 60; //para 100 ms

INTCONbits.T0IF=0; // ; Limpiamos bandera de TMR0

}

void resettmr1(void){

TMR1H = 11; //para 500 ms

TMR1L = 184;

PIR1bits.TMR1IF=0; // ; Limpiamos bandera de TMR0

}

void configint(void){

INTCONbits.T0IE = 1; // Habilitamos interrupcion TMR0

INTCONbits.T0IF = 0; // ; Limpiamos bandera de TMR0

PIE1bits.TMR1IE = 1; //tmr1 interrupt

PIR1bits.TMR1IF = 0; //tmr1 interrupt

INTCONbits.GIE = 1; // Habilitamos interrupciones globales

INTCONbits.PEIE = 1; // ENABLE peripherial INTERRUPT

}

void setuptmr0(void){

//OSCCON|=0x70; // SELECT 8MHz Internal Oscillator

OPTION\_REGbits.T0CS=0; // SELECT INTERNAL SOURCE TMR0

OPTION\_REGbits.PSA=0; // PRESCALER ASSIGNED TO TIMER0

OPTION\_REGbits.PS2=1; // PRESCALER ASSIGNED TO TIMER0

OPTION\_REGbits.PS1=1; // PRESCALER ASSIGNED TO TIMER0

OPTION\_REGbits.PS0=1; // PRESCALER ASSIGNED TO TIMER0 256

//OPTION\_REG&=0xF8; // SELECT 1:2 PRESCALER

resettmr0();

}

void setuptmr1(void){

T1CONbits.T1OSCEN = 1; // SELECT INTERNAL SOURCE TMR1

T1CONbits.TMR1CS = 0; // SELECT INTERNAL SOURCE TMR1

T1CONbits.TMR1ON = 1; // SELECT INTERNAL SOURCE TMR1

T1CONbits.T1CKPS = 0b11; //PRESCALER 1:8

resettmr1();

}

**setup.h**

#ifndef SETUP\_H

#define SETUP\_H

extern void setup(void);

extern void setuptmr0(void);

extern void resettmr0(void);

extern void setuptmr1(void);

extern void resettmr1(void);

extern void configint(void);

#endif

**Ejercicio 2**

Escriba un código que configure el puerto A como salida y que recorra los pines del puerto encendiéndolos uno a uno, manteniendo encendido uno a la vez por un segundo. Es decir, enciende RA0 y apaga los demás por un segundo, luego enciende RA1 y apaga los demás por un segundo, así hasta llegar a RA7 e iniciar nuevamente con RA0.

Configure el oscilador interno a 4MHz.

**HDCT.c**

/\*

\* File: HDTC.c

\* Author: Dina Rodríguez

\*

\* Created on 1 de abril de 2022, 11:03 PM

\*/

// CONFIG1

#pragma config FOSC = INTRC\_NOCLKOUT// Oscillator Selection bits (INTOSCIO oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config MCLRE = OFF // RE3/MCLR pin function select bit (RE3/MCLR pin function is digital input, MCLR internally tied to VDD)

#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)

#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is disabled)

#pragma config BOREN = OFF // Brown Out Reset Selection bits (BOR disabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal/External Switchover mode is disabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)

#pragma config LVP = OFF // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)

// CONFIG2

#pragma config BOR4V = BOR40V // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)

#pragma config WRT = OFF // Flash Program Memory Self Write Enable bits (Write protection off)

// #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

#include <stdio.h>

#include "setup.h"

#define \_XTAL\_FREQ 4000000 //configuracion 4MHz

/\*------------------------------------------------------------------------------

\* CONSTANTES

------------------------------------------------------------------------------\*/

#define INCREMENTAR PORTBbits.RB0 // Asignamos un alias a RB0

#define DECREMENTAR PORTBbits.RB1 // Asignamos un alias a RB1

/\*----------VARIABLES----------------------------

// Ejemplos:\*/

uint8\_t var = 0;

uint8\_t var1 = -1; // Solo declarada

// uint8\_t var2 = 0; // Declarada e inicializada

/\*------------------------------------------------\*/

/\* -----------PROTOTIPO DE FUNCIONES \*/

void setport(void);

/\*----------------------\*/

/\*-------INTERRUPCIONES -----------------\*/

void \_\_interrupt() isr (void){

if(INTCONbits.RBIF){ // Fue interrupci n del PORTB�

if(!INCREMENTAR){ // Verificamos si fue RB0 quien gener la �interrupci n�

//PORTA++; // Incremento del PORTC (INCF PORTC)

}

if(!DECREMENTAR){ // Verificamos si fue RB0 quien gener la �interrupci n�

//PORTA--; // Incremento del PORTC (INCF PORTC)

}

INTCONbits.RBIF = 0; // Limpiamos bandera de interrupci n�

}

if(INTCONbits.T0IF){ //se revisa bandera del timer

resettmr0();

//PORTC++;

}

if(PIR1bits.TMR1IF){

resettmr1();

var++;

if (var%2==0) setport();

}

return;

}

void setport(void){

PORTA = 0b00000000;

var1++;

switch (var1){

case 0:

RA0 = 1;

break;

case 1:

RA1 = 1;

break;

case 2:

RA2 = 1;

break;

case 3:

RA3 = 1;

break;

case 4:

RA4 = 1;

break;

case 5:

RA5 = 1;

break;

case 6:

RA6 = 1;

break;

case 7:

RA7 = 1;

break;

default:

var1 = 0;

RA0 = 1;

break;

}

return;

}

void main(void) {

setup(); // Llamamos a la funci n de configuraciones�

setuptmr0(); // Llamamos a la funci n de configuraciones�

setuptmr1();

configint();

while(1){

}

return;

}

**Ejercicio 3**

Defina el siguiente arreglo:

valores = [1, 127, 95, 36, 15, 253, 63]

Usando un ciclo for, recorra el arreglo y muestre cada uno de los valores, durante un segundo, en el puerto C.

**Ejercicio 4**

Defina un arreglo de 10 posiciones y guarde valores entre 0 y 20 (no repetidos) en cada una de las posiciones.

Agregue un contador (entre 0 y 20) y muestre el valor del contador en el puerto a, al presionar RB0 el valor del contador incrementa y al presionar RB1 el contador decrementa.

Si el valor del contador es igual al valor algún valor del arreglo, muestre el puerto c la posición del valor en el arreglo.