/\*

\* File: hdt.c

\* Author: Dina Rodríguez

\*

\* Created on 1 de abril de 2022, 10:11 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�

}

return;

}

void main(void) {

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

while(1){

}

return;

}

////////////////////////////////setup.c/////////////////////////////////////////////////////////

/\*

\* File: setup.c

\* Author: Dina

\*

\* Created on 2 de abril de 2022, 6:56 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.GIE = 1; // Habilitamos interrupciones globales

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�

}

// Td = Pre \* TMR1\*Ti

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

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

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