



Tarea 3: Cronómetro con minutos y segundos

Integrantes:

- Leyva Mercado Christian Alejandro
- Lugo Saenz Jesus
- Navarrete Zamora Aldo Yael

Fecha de entrega: 14 de mayo de 2023

Grupo: 4

Equipo: 5

Microcomputadoras

▼ Índice

Objetivo

Desarrollo

[Programación del PIC](#)

[Pseudocódigo de la solución](#)

[Diagrama de flujo de la solución](#)

[Código](#)

[Simulación del sistema](#)

[Ejecución física en el PIC](#)

[Análisis de Resultados](#)

[Prueba de revisión del profesor](#)

Conclusiones

Referencias

Objetivo

Diseñar, programar e implementar un sistema utilizando un pic16f877a/pic16f887 que muestre un cronómetro con minutos y segundos; para iniciar la cuenta de tiempo se presionará un botón start/pause, ese mismo botón servirá para pausar y otro botón para reiniciar a cero (reset).

Desarrollo

Programación del PIC

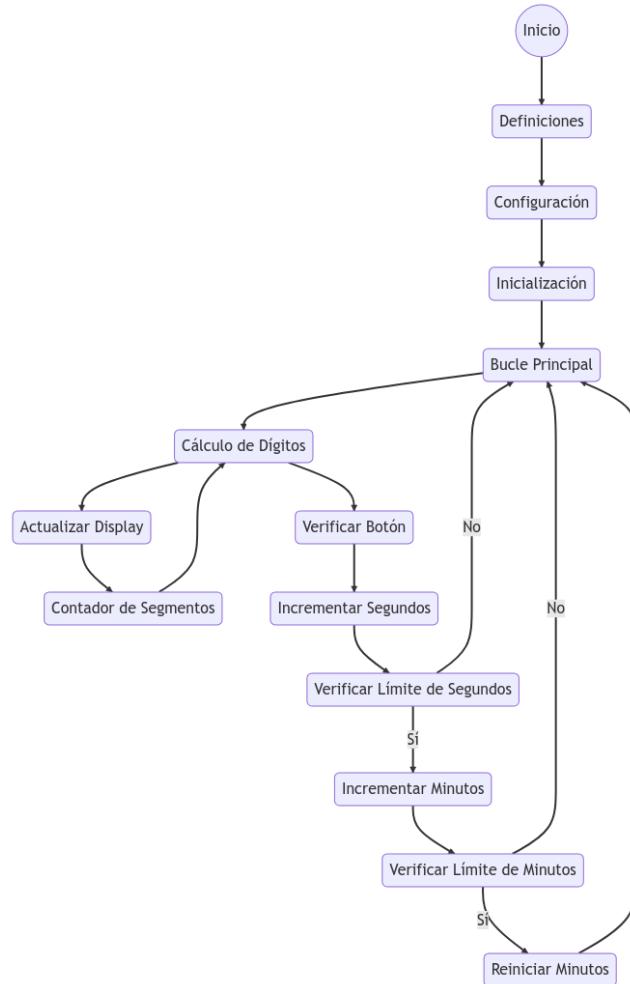
▼ Pseudocódigo de la solución

```
// Incluir la librería "16f887.h".
// Configurar los fusibles del microcontrolador para deshabilitar el watchdog timer (NOWDT), el circuito de protección contra voltajes t
// Establecer la frecuencia del reloj interno en 8MHz mediante la función "delay()".

// Definir los pines correspondientes a los seis displays LED en la placa, donde display_1 es el pin E0, display_2 es el pin E1, display_
// Definir una matriz de caracteres que contiene los valores en hexadecimal correspondientes a los números del 0 al 9 y letras del alfabeto
// Definir cuatro variables para representar las unidades, decenas, centenas y millares de la cuenta regresiva del temporizador.

// Crear una función de interrupción que se ejecuta cada vez que ocurre una interrupción en el registro RTCC. La función cambia el valor
// En la función principal, inicializar las variables y los pines de los displays como salidas.
// Configurar el temporizador interno del microcontrolador con la función "setup_timer_0()" para que se ejecute a una frecuencia de reloj
// Iniciar un bucle infinito que se ejecuta constantemente. Dentro de este bucle, actualizar las variables de las unidades, decenas, centenas y millares
// Esperar 100 milisegundos y luego verificar si se ha presionado el botón B0. Si se ha presionado, incrementar el valor de los segundos
// Si los segundos alcanzan un valor de 60, incrementar los minutos y reiniciar los segundos a cero. Si los minutos alcanzan un valor de
```

▼ Diagrama de flujo de la solución



▼ Código

Una vez que obtuvimos el pseudocódigo implementamos el programa principal que iba a realizar el conteo del cronómetro:

```

#include <16f887.h>
#FUSES NOWDT, NOBROWNOUT, NOLVP
#use delay(internal=8M)

#define display_1 PIN_E0 //Es el display de las decenas de minuto
#define display_2 PIN_E1 //Es el display de las unidades de minuto
#define display_3 PIN_E2 //Es el display de las decenas de segundo
#define display_4 PIN_C0 //Es el display de las unidades de segundo
#define display_5
#define display_6

#define tiempo 10

char seg[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};

char unidades = 0;
char decenas = 0;
char centenas = 0;
char millares = 0;

#INT_RTCC
void interrupciones(void){
    static char contador = 0;
    switch (contador) {
        case 0:

```

```

        output_low(display_4);
        output_high(display_1);
        output_d(seg[millares]);
        break;
    case 1:
        output_low(display_1);
        output_high(display_2);
        output_d(seg[centenas]);
        break;
    case 2:
        output_low(display_2);
        output_high(display_3);
        output_d(seg[decenas]);
        break;
    case 3:
        output_low(display_3);
        output_high(display_4);
        output_d(seg[unidades]);
        break;
    }
    contador++;
    if (contador >= 4) {
        contador = 0;
    }
}

void main(){
    char segundos = 0;
    char minutos = 0;
    set_tris_d(0);
    output_d(0);
    output_low(display_1);
    output_low(display_2);
    output_low(display_3);
    output_low(display_4);

    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_16|RTCC_8_bit);
    enable_interrupts(INT_RTCC);
    enable_interrupts(GLOBAL);

    while(TRUE){
        unidades = segundos % 10;
        decenas = (segundos/10) % 10;
        centenas = minutos % 10;
        millares = (minutos / 10) % 10;

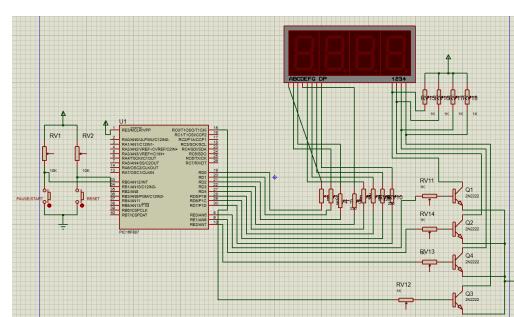
        delay_ms(100);
        if (input(PIN_B0)) {
            segundos++;
        }
        if (segundos == 60) {
            minutos++;
            segundos = 0;
            if (minutos == 60) {
                minutos = 0;
            }
        }
    }
}

```

Simulación del sistema

Después de realizar el programa y compilarlo en PIC C Compiler, se implementó la simulación del circuito en el programa **Proteus** donde se colocaron los siguientes componentes:

- PIC16F887
- Push button (Reset)
- Push button (Start)
- 8 resistencias de 1 kΩ
- 2 resistencias de 10kΩ

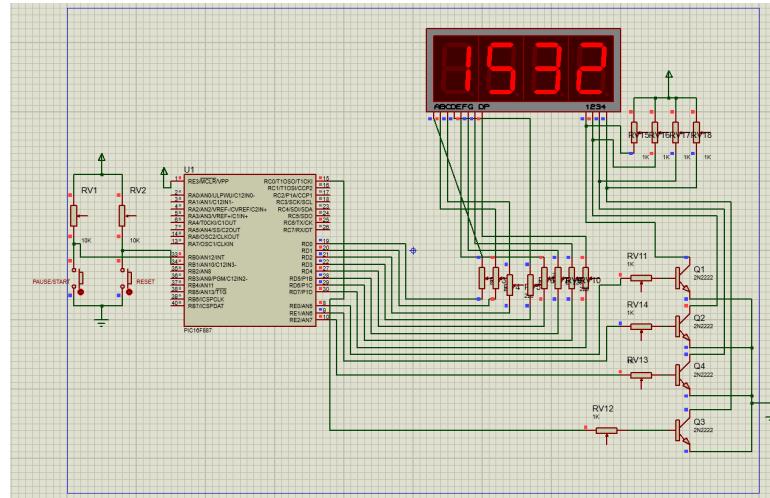
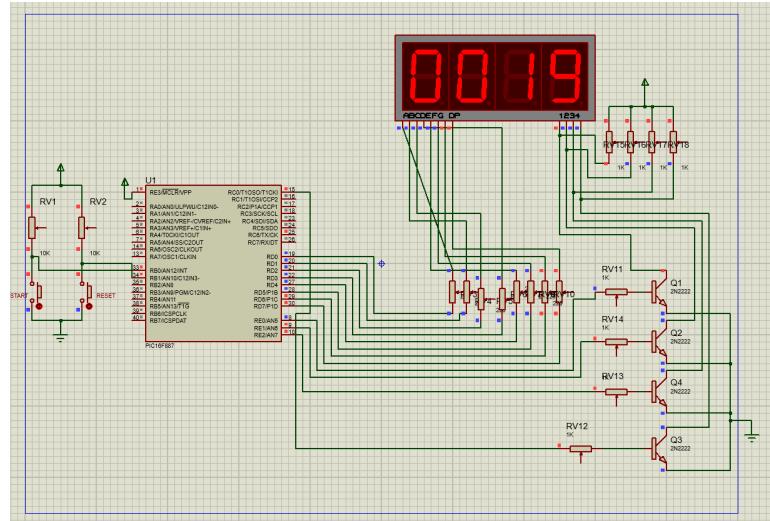


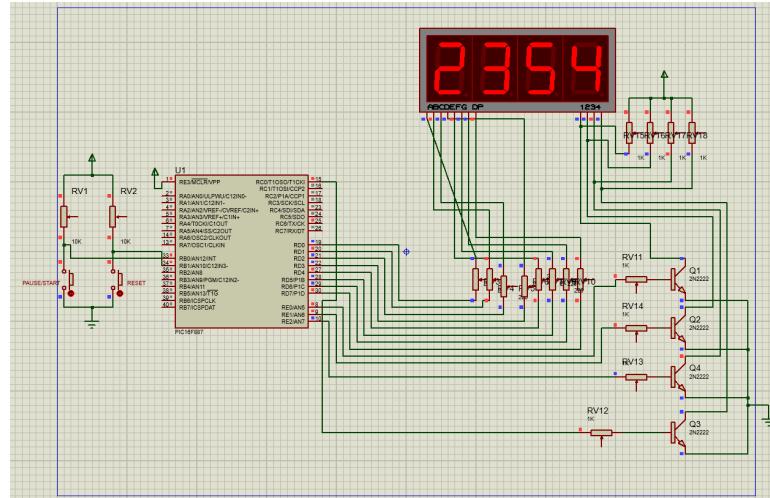
Circuito del sistema. Simulado en el programa Proteus

- 8 resistencias de $220\ \Omega$
- 4 transistores 2N2222
- Display de 7 segmentos de 4 dígitos

Finalmente, se dio doble click en el PIC y se ingreso el archivo **.HEX**, el cual fue generado en el software pic c compiler. El resultado de la ejecución en el microcontrolador es el siguiente:

▼ **Programa principal : Contador.**

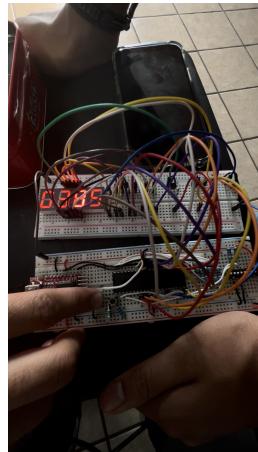




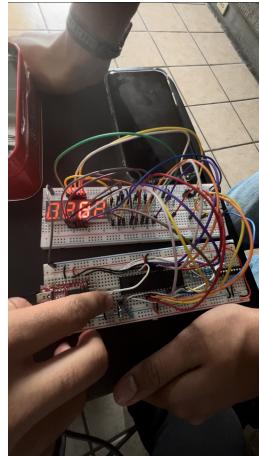
Ejecución física en el PIC

Después de que ensamblemos el programa, lo cargamos en la PIC y este es el resultado de la ejecución en el microcontrolador:

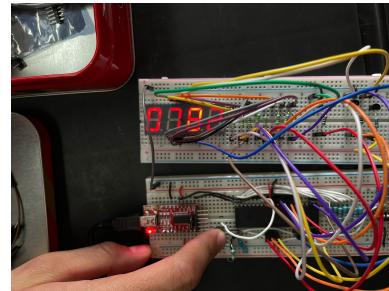
1. La ejecución del código en la tarjeta nos muestra primeramente al circuito empezando a contar desde **00:00**, este contador irá incrementando en 1 con tiempo de retardo definido en el código.



2. Una vez el contador llegue a 60, el flujo del código determina que los minutos aumentan, entonces el contador setea su valor a **01:00**.



- Finalmente, cuando el contador llegue a los 60 minutos, se reiniciará a **00:00** o bien, si se presiona el botón de reset, este regresará a su valor inicial, y cuando se presione el de pausa, el sistema dejará de contar.



Análisis de Resultados

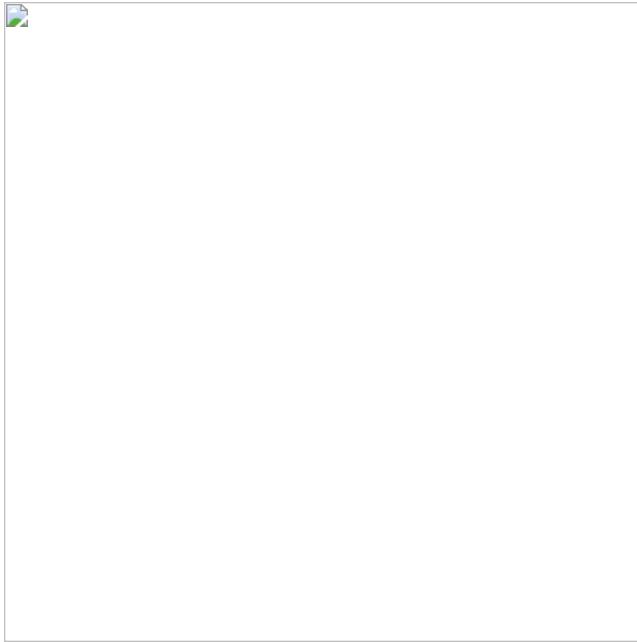
Al ensamblar nuestro programa en C y programar nuestro microcontrolador 16f887, este comienza a contar desde cero a una velocidad acelerada (es decir, no equivalente a un segundo real). Se eligió una velocidad un poco más rápida que la del tiempo real para demostrar el correcto funcionamiento del sistema al llegar a su límite (60:00) de forma rápida (el tiempo de ejecución del programa se puede cambiar de manera sencilla modificando el tiempo de retardo llamado `tiempo`).

El cronómetro muestra de manera correcta el paso del tiempo en los displays de 7 segmentos, sin embargo, tuvimos un problema técnico en la ejecución debido a que el 3er display duplicaba el numero del 4 display, esto sucedió debido a nuestra Protoboard, ya que estaba defectuosa. Sin embargo como se demostró con la simulación, el código funciona de manera correcta.

De igual manera los botones funcionan de manera correcta, el botón RESET reinicia la cuenta a 0, y el botón PAUSE, como su nombre lo indica pausa el contador mientras este se mantenga pulsado. Como se muestra en las imágenes superiores.

En resumen, estamos muy satisfechos con los resultados de nuestra tarea y cumple con todos los requisitos planteados por el profesor.

Prueba de revisión del profesor



Escritorio del profesor revisando nuestro proyecto en físico. El 11 de Mayo de 2023

Conclusiones

- **Leyva Mercado Christian Alejandro**

En conclusión, la tarea nos permitió explorar y comprender en profundidad el manejo de puertos y la programación con interrupciones en microcontroladores con el lenguaje C.

Esto fue muy útil ya que en las prácticas y tareas anteriores, estábamos acostumbrados a programar en ensamblador los PICs, por lo que programar en un nivel de alto nivel como C facilitó en mayor parte la tarea.

En general, con todo y los problemas técnicos que tuvimos a lo largo de la construcción del sistema físico de la tarea, como el problema de la Protoboard defectuosa, considero que la tarea fue un éxito y que nos brindó una valiosa experiencia en el campo de la electrónica y la programación de microcontroladores.

- **Lugo Saenz Jesus**

En conclusión basándonos en que el programa fue hecho en lenguaje C, es que la programación en C permite una alta portabilidad y eficiencia en la implementación de sistemas embebidos como este cronómetro. Además, C es un lenguaje de programación de bajo nivel que permite un mayor control sobre los recursos del microcontrolador y un mejor aprovechamiento de su capacidad de procesamiento. La programación en C también proporciona una mayor flexibilidad en el diseño y la implementación de sistemas, lo que facilita su mantenimiento y actualización en el futuro. En resumen, la elección de C como lenguaje de programación para este sistema de cronómetro permite un alto grado de eficiencia, flexibilidad y control en su implementación.

- **Navarrete Zamora Aldo Yael**

Considero que esta tarea nos demuestra de forma amplia que hay distintas formas de manejar los puertos ya sea mediante lenguaje ensamblador en donde tenemos que estar al tanto de los registros y cada una de las banderas, así como en lenguaje C.

Este último me resulta más familiar y por lo tanto, un poco más comprensible cuando se realizó la programación, el uso de interrupciones como contadores nos mostró como poder aplicar el uso de las mismas, en este caso, se utilizaron las interrupciones externas con un divisor de 16 bits. Creo que se cumple el objetivo de crear el cronómetro contador con éxito, incluso, con el uso de interrupciones.

Referencias

- *Online FlowChart & Diagrams Editor - Mermaid Live Editor.* (s. f.). <https://mermaid.live/>
- Microchip, “**PIC16F87X Data sheet**” **DS30292C, 2001.** Online. Consultado el 10 de mayo del 2023, <https://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>.