Facultad de ingeniería

Materia: Laboratorio de Microcomputadoras

Previo 3

Título: Sistema mínimo microcontrolador PIC16F877

Integrantes:

- Martínez Pérez Brian Erik - 319049792
- Nuñez Rodas Abraham Enrique - 114003546
- Vicenteño Maldonado Jennifer Michel - 317207251

Profesor: Moises Melendez Reyes

Grupo: 1

Fecha de Entrega: 2 de marzo de 2025

Semestre: 2025-2

1. Explique qué es un puerto paralelo en los microcontroladores PIC y cómo se configura su función de entrada/salida.

Son un conjunto de pines que pueden funcionar como entradas o salidas de datos de manera simultánea. pueden manejar múltiples bits a la vez, a diferencia de los puertos seriales que envían los datos bit a bit.

Para definir si un puerto funcionará como entrada o salida, se usa el registro “TRISx”:
1: Configura el pin como entrada (Input).
0: Configura el pin como salida (Output).

2. ¿Por qué es necesario cambiarse de banco para configurar los puertos paralelos en el microcontrolador pic?

Por defecto, al encender el microcontrolador, el Banco 0 está activo, y en este banco solo se pueden leer y escribir los valores de los puertos “PORTx”.

Para acceder a ciertos registros de configuración, como los que controlan los puertos paralelos, es necesario cambiar de banco de memoria usando el bit “RP0” en el registro “STATUS”.

3. Describa tres formas de cargar un programa (firmware) a la memoria ROM de un microcontrolador.

Método 1: Programación In-Circuit (ICSP - In-Circuit Serial Programming)
Se programa el microcontrolador directamente dentro del circuito donde está montado, sin necesidad de retirarlo.

Método 2: Programación mediante Bootloader (USB, UART, Serial, WiFi)
Se usa un bootloader, un pequeño programa previamente grabado en la memoria ROM del microcontrolador.

Método 3: Programación con Grabador Externo
Se usa un programador independiente que graba el firmware en la memoria ROM antes de instalar el microcontrolador en el circuito. Es útil en producción en masa, donde se requiere programar múltiples chips antes de montarlos en una placa PCB.

4. Explique el método de bootloader para descargar un programa al microcontrolador.

Un **bootloader** es un pequeño firmware residente en el microcontrolador que permite cargarle un nuevo programa a través de un puerto de comunicación estándar (por ejemplo RS-232) sin usar un grabador especializado. En el caso de microcontroladores PIC con memoria Flash como el **PIC16F877A**, aprovecha la capacidad de **auto-programación** de la Flash (el propio PIC puede escribir en su memoria de programa). En otras palabras, primero se graba un código de bootloader en la MCU (una única vez, usando un programador convencional) y luego, gracias a ese bootloader, el PIC puede reprogramarse a sí mismo recibiendo los datos del nuevo firmware desde un PC por serie. Esto facilita las pruebas y actualizaciones, ya que no hace falta extraer el chip ni usar un grabador hardware tras la primera vez. El bootloader típicamente **no se borra ni sobrescribe** al actualizar el firmware, pues reside en una región protegida de la memoria Flash (p. ej. en el extremo superior del espacio de programa). Por ello, el programa de usuario debe dejar reservada esa zona para el bootloader (por ejemplo, unos 256 bytes superiores) y ajustar los vectores de interrupción si es necesario.

Cuando el PIC se reinicia, el bootloader toma el control antes que el programa principal. Según la configuración, puede verificar una condición para entrar en "modo bootloader" – por ejemplo, comprobar un pin externo (un botón de actualización) o esperar un comando inicial desde la PC por la UART. Una estrategia común es que el bootloader siempre se active brevemente tras reset: por ejemplo, permanece a la escucha en el puerto serie por un corto intervalo (ej. ~0,2 segundos) esperando un mensaje de inicio de carga. Si en ese lapso recibe el comando correcto desde el software host en la PC (p. ej. un carácter especial o secuencia de sincronización), entonces **se queda en modo programación**; de lo contrario, si no hay comunicación, **cede el control** y arranca la aplicación principal existente. Cuando entra en modo bootloader, típicamente envía una señal de "ready" o identificación al PC y luego el PC transmite el nuevo programa. El PIC16F877A usa su módulo USART asíncrono (UART) configurado con los parámetros estándar **8 bits de datos, sin paridad, 1 bit de parada** (formato 8N1) a una velocidad de baudios acordada (por ejemplo 9600, 19200, etc., dependiendo del cristal utilizado). Este UART del PIC se conecta al puerto serial del PC (RS-232) mediante un nivelador de voltaje MAX232, como se explica más adelante. Durante la transferencia, el bootloader recibe los datos del programa en bloques y los va escribiendo en la Flash interna usando las instrucciones de auto-escritura del PIC. El **protocolo de comunicación** puede variar según el bootloader específico: algunos bootloaders sencillos esperan datos en formato de texto (p. ej. líneas Intel HEX) y envían un acuse de recibo por cada línea para que el PC envíe la siguiente; otros implementan protocolos binarios más robustos (como **XModem** en modo CRC de 128 bytes) para enviar el archivo completo con detección de errores. En cualquier caso, por cada bloque de bytes recibidos el bootloader suele contestar con

un **ACK** (acusando buena recepción) o solicitar reenvío si hubo error, asegurando la integridad de la carga. Una vez transferido todo el programa (y verificado si aplica), el bootloader suele reiniciar el microcontrolador o saltar a la dirección de inicio de la aplicación cargada. En adelante, el PIC ejecutará el nuevo firmware de usuario hasta que se repita el proceso

5. El puerto RS232 se usa, entre otras aplicaciones, para la comunicación entre el bootloader y la computadora, explique la función del circuito integrado MAX232.

El **MAX232** es un conversor de niveles lógicos diseñado para interconectar dispositivos TTL/CMOS (como el UART de un microcontrolador a 5 V) con una interfaz **RS-232** estándar (puerto serie de PC). La necesidad de este IC se debe a que la señalización RS-232 utiliza voltajes **positivos y negativos** mucho más altos que los niveles TTL. Por ejemplo, en el protocolo RS-232 un “1” lógico se representa típicamente con -3 V a -15 V, y un “0” lógico con +3 V a +15 V. En cambio, la UART de un PIC opera con lógica TTL: aproximadamente 0 V para “0” y +5 V para “1”. Además, el puerto serial de PC requiere voltajes bipolares (positivo/negativo) y podría dañar directamente un pin del microcontrolador si se conectara sin adaptación, ya que ±12 V exceden el rango de 0-5 V TTL. Por tanto, se inserta el MAX232 como **driver/receptor** de línea serial: este chip **convierte** e **invierte** las señales entre ambos estándares.

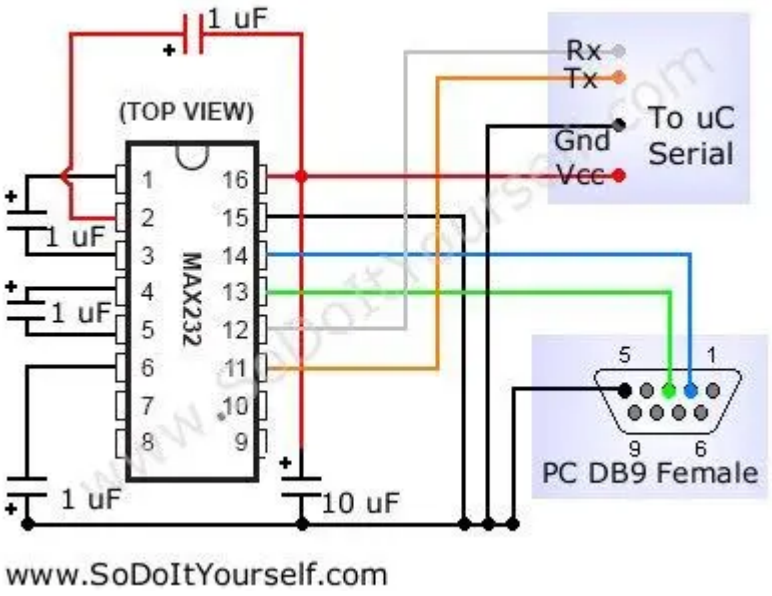
Internamente, el MAX232 contiene dos transmisores (de TTL a RS-232) y dos receptores (de RS-232 a TTL), suficientes para manejar las cuatro líneas más comunes de un puerto serial: TX, RX, CTS, RTS. En una aplicación típica de bootloader se usan solo TX y RX. El MAX232 se alimenta con +5 V y, notablemente, genera internamente las tensiones positivas y negativas necesarias para RS-232 (aprox. ±10 V) a partir de esa única alimentación. Lo logra mediante un **multiplicador/inversor de voltaje** integrado (una **bomba de carga capacitiva**): se conectan cuatro condensadores externos (generalmente 1 µF o 10 µF según la versión) a pines designados (C1+, C1-, C2+, C2-) que el MAX232 conmuta internamente para producir ~+10 V (V_+) y ~-10 V (V_-). Por ejemplo, al cargar y conmutar C1 entre 5 V y 0 V, y transferir carga a C2, se obtiene una tensión duplicada y luego invertida, respectivamente, en los nodos V+ y V-. De este modo, el MAX232 **no requiere fuentes de ±12 V externas**, a diferencia de antiguos conversores (p. ej. MAX202) que sí las necesitaban. Este diseño hace muy sencilla la interfaz RS-232 en sistemas de 5 V. El consumo típico del MAX232 es bajo (~8 mA) y soporta baudios de hasta ~120 kbps sin distorsión.

Internamente, el MAX232 contiene dos transmisores (de TTL a RS-232) y dos receptores (de RS-232 a TTL), suficientes para manejar las cuatro líneas más comunes de un puerto serial: TX, RX, CTS, RTS. En una aplicación típica de

bootloader se usan solo TX y RX. El MAX232 se alimenta con +5 V y, notablemente, genera internamente las tensiones positivas y negativas necesarias para RS-232 (aprox. ±10 V) a partir de esa única alimentación. Lo logra mediante un **multiplicador/inversor de voltaje** integrado (una **bomba de carga capacitiva**): se conectan cuatro condensadores externos (generalmente 1 µF o 10 µF según la versión) a pines designados (C1+, C1-, C2+, C2-) que el MAX232 conmuta internamente para producir ~+10 V (V_+) y ~-10 V (V_-). Por ejemplo, al cargar y conmutar C1 entre 5 V y 0 V, y transferir carga a C2, se obtiene una tensión duplicada y luego invertida, respectivamente, en los nodos V+ y V-. De este modo, el MAX232 **no requiere fuentes de ±12 V externas**, a diferencia de antiguos conversores (p. ej. MAX202) que sí las necesitaban. Este diseño hace muy sencilla la interfaz RS-232 en sistemas de 5 V. El consumo típico del MAX232 es bajo (~8 mA) y soporta baudios de hasta ~120 kbps sin distorsión.

Conversión de niveles: Cuando el MAX232 recibe un nivel desde el lado TTL (por ejemplo, un bit TX del microcontrolador), este lo convierte al nivel RS-232 equivalente invirtiendo la lógica. En concreto, un **‘1’ TTL (≈ +5 V)** lo traduce a un **nivel negativo** en RS-232 (por ej., -10 V en la salida correspondiente), y un **‘0’ TTL (0 V)** lo convierte en un **nivel positivo** de RS-232 (por ej., +10 V). A la inversa, para la recepción, si la línea RS-232 de entrada está en +10 V (señal que en RS-232 representa un ‘0’ lógico), el MAX232 la interpreta y saca un 0 V en su salida TTL; si la entrada RS-232 está en -10 V (indicando un ‘1’ lógico RS-232), el MAX232 entrega ~5 V en la salida TTL. De esta manera se asegura la traducción correcta de bits entre los dos sistemas de lógica inversa. Los receptores RS-232 del MAX232 están diseñados con umbrales de conmutación alrededor de 1,3 V y con histéresis de ~0,5 V para rechazar ruido, y toleran entradas de hasta ±25 V protegiendo al microcontrolador. Asimismo, las salidas RS-232 del MAX232 pueden conducir a ±7,5 V típicos con carga estándar, suficientes para cumplir la especificación RS-232.

Esquema de conexión típico de un MAX232 entre un microcontrolador y un puerto serie RS232. Se muestran los capacitores externos requeridos y las conexiones de Tx, Rx, Vcc=5 V y masa (GND). El DB9 hembra del PC utiliza solo tres pines: 2 (Rx recibe datos desde el Tx del PIC), 3 (Tx envía datos al Rx del PIC) y 5 (GND común).



6. Explique qué es una subrutina de retardo y por qué se requiere en dispositivos de interfaz humana.

Un retardo implica una espera en la ejecución del programa durante un tiempo determinado. Los retardos son necesarios para que el ojo humano pueda percibir ciertas acciones del programa, o incluso para hacer más evidentes los cambios.
Los retardos también pueden ser utilizados como subrutinas o supresores de rebote.

Existen dos formas de realizar retardos:

- **software:**
  Esto sucede cuando el pic se queda en un bucle por un tiempo determinado, esto se puede realizar utilizando uno o varios contadores en decremento y dando fin el retardo cuando la cuenta llega a 0. Con 1 bucle sólo se puede llegar a un retardo de 1ms y en caso de querer un retardo mayor se necesitaría usar bucles anidados.

- **por TMR0:**
  Se utiliza el temporizador del microcontrolador para generar el retardo. Se configura el valor inicial del TMR=, se configura una escala que hace que el contador vaya más lento o más rápido y esperamos a que el temporizador llegue a su límite.

7. Describa brevemente cómo se realiza una subrutina de retardo en lenguaje ensamblador.

A continuación se muestra un ejemplo de funcionamiento de un retardo básico en ensamblador:

```
retardo
        movlw d'100'            ;w←100
        movwf contador          ;conta←w asignamos un valor al contador
        nop                     ;ahora el número que cargamos es más exacto
        decfsz contador, f      ;conta←conta-1- y salta si ya llegó a 0 decremento
        goto $-.1               ;regresa a retardo para seguir incrementando
```

Para calcular el tiempo de retardo, obtenemos la duración de cada instrucción, la multiplicamos por las n repeticiones.

decfsz=1 us, nop=1 us, goto=2 us, con una n=100, entonces tenemos 400 us de retardo.

8. ¿Qué es un sistema mínimo para microcontrolador?

Consiste en el conjunto de componentes de hardware para que el microcontrolador funcione y pueda ejecutar un programa.

El sistema mínimo del PIC16F877 consiste en:

- microcontrolador PIC16F877
- adaptador de niveles, tipo MAX232
- regulador de voltaje, tipo 7805
- diodo, tipo 1N4004
- capacitor cerámico, tipo 0.33 uF
- capacitor cerámico, tipo 15 uF 2 unidades
- capacitor cerámico, tipo 0.1 uF 2 unidades
- capacitor electrolítico, tipo 1 uF 4 unidades
- conector DB9, tipo hembra
- cristal de cuarzo, tipo 20MHz
- resistencia de ½ watt, tipo 10k
- micro boton de 2 terminales, tipo 51
- cable adaptador usb a serie

Referencias

del PIC, 2. 1-La Familia. (s/f). 2.- Descripción General del PIC16F877. Edu.ar. Recuperado el 23 de febrero de 2025, de https://exa.unne.edu.ar/ingenieria/sysistemas/public_html/Archi_pdf/HojaDatos/Microcontrolador es/PIC16F877.pdf

GARCÍA, Rubén, SAVAGE, Jesús, MUNIVE, Carlos. Prácticas de laboratorio de microcomputadoras, México,Universidad Nacional Autónoma de México Facultad de Ingeniería.

Ochoa, Y. (2019, 26 septiembre). Generar retardos-para-pic-en-mplab [Diapositivas]. SlideShare.https://es.slideshare.net/slideshow/generar-retardosparapicenmplab-176473635/176473635

M, Moises. (2020, 02 diciembre).Sistema Mínimo Con El Microcontrolador PIC16F877A. Scribd. https://es.scribd.com/document/486675848/Sistema-minimo-con-el-microcontrolador-PIC16F877A#:~:text=Opciones%20para%20compartir&text=Reportar-,Este%20documento%20presenta%20un%20sistema%20m%C3%ADnimo%20basado%20en%20el%20microcontrolador,sobretensi%C3%B3n%20e%20inversi%C3%B3n%20de%20polaridad

Microchip Technology Inc. (2002). *AN851: A Flash Bootloader for PIC16 and PIC18 Devices* ApplicationNoteApplication Note
ApplicationNote
. Microchip Technology. https://ww1.microchip.com/downloads/en/appnotes/00851b.pdf