CURSO DE PROGRAMACIÓN E INTERFAZ DE MICROCONTROLADORES PIC

Video 02 — Parte 2: El Microcontrolador PIC16F84 y Programación de un PIC · Plataforma Oficial: cursosmecatronica.com

Módulo: Parte 2 — El Microcontrolador PIC16F84 y Programación de un PIC
Enfoque de Ingeniería: Pinout DIP-18, Oscilador XT/HS/RC, Reset MCLR y PWRT 72 ms, Flash 1Kx14, Bancos SFR 0/1, RA4 Open-Drain, HEX e ICSP

```
Flash = 1024 x 14 bits | RAM_GPR = 68 bytes | EEPROM = 64 bytes | T_TMR0 = 4 · PS · (256 - TMR0) / f_osc
```

1. Desarrollo Técnico Diapositiva por Diapositiva

Diapositiva 01: El Microcontrolador PIC16F84 y PIC16F84A: Encapsulado DIP-18 y Pinout

Alimentación VDD y VSS, trece Líneas Bidireccionales de Entrada/Salida, Pines de Reloj y Reinicio Maestro

- **Turno 1 (Orus):** El microcontrolador PIC dieciséis efe ochenta y cuatro, junto con su revisión mejorada terminada en letra A que alcanza hasta veinte megahercios, es el modelo didáctico e industrial clásico por excelencia en encapsulado de dieciocho pines.
- **Turno 2 (Aoede):** Identificar correctamente sus dieciocho terminales a partir de la muesca superior es vital: el pin catorce corresponde a la alimentación positiva ve de de de cinco voltios, mientras que el pin cinco es la referencia de tierra ve ese ese.
- **Turno 3 (Orus):** De los dieciséis pines restantes, trece son líneas digitales bidireccionales agrupadas en dos puertos: cinco líneas del puerto A, etiquetadas erre a cero hasta erre a cuatro, y ocho líneas del puerto B, desde erre be cero hasta erre be siete.
- **Turno 4 (Aoede):** Los últimos tres terminales cumplen funciones vitales de soporte hardware: el pin cuatro es el reinicio maestro activo en nivel bajo, y los pines quince y dieciséis conectan la red externa del oscilador de reloj.

Diapositiva 02: Configuración del Oscilador de Reloj: Modos XT, HS, LP y Red RC

Cristal de Cuarzo de 4 MHz con Capacitores de 22 pF, Resonador Cerámico de Tres Terminales y Oscilador RC

- **Turno 1 (Orus):** Para que el PIC ejecute instrucciones necesita una señal periódica estable en sus pines quince y dieciséis, permitiendo escoger cuatro modos de oscilador según la precisión temporal y el costo del proyecto.
- **Turno 2 (Aoede):** En la gran mayoría de aplicaciones conectamos un cristal de cuarzo de cuatro megahercios en modo estándar X T, acompañado de dos pequeños capacitores cerámicos de entre quince y treinta y tres picofaradios hacia tierra.
- **Turno 3 (Orus):** Si requerimos mayor velocidad, hasta veinte megahercios en la versión A, seleccionamos el modo de alta velocidad H S; o bien empleamos un resonador cerámico de tres patas que ya integra internamente los capacitores de carga.
- **Turno 4 (Aoede):** Para circuitos de muy bajo costo donde la exactitud del tiempo no es crítica existe el modo R C con un resistor a positivo y un capacitor a tierra, aunque su frecuencia deriva notablemente con la temperatura y la tensión.

Diapositiva 03: Circuito de Reinicio Maestro MCLR, Power-On Reset y Power-Up Timer

Resistencia Pull-Up a VDD, Pulsador de Reinicio Manual, Diodo de Protección y Estabilización de 72 ms

- **Turno 1 (Orus):** El pin cuatro corresponde al borrado maestro o reinicio externo activo en bajo: durante la operación normal debe mantenerse firme a cinco voltios mediante una resistencia de diez kiloohmios conectada a la línea positiva.
- **Turno 2 (Aoede):** Si llevamos momentáneamente este pin a cero voltios mediante un pulsador de reinicio, el microcontrolador detiene su actividad de forma inmediata y fuerza el contador de programa al vector inicial cero cero cero cero hexadecimal.
- **Turno 3 (Orus):** Internamente el chip incorpora un circuito de reinicio al encendido llamado Power On Reset, que detecta la subida de la fuente, junto con el temporizador de arranque Power Up Timer de setenta y dos milisegundos.
- **Turno 4 (Aoede):** Esos setenta y dos milisegundos mantienen al procesador en estado de reinicio mientras la fuente de cinco voltios termina de estabilizarse y el cristal de cuarzo alcanza oscilaciones limpias, evitando arranques erráticos.

Diapositiva 04: Memoria de Programa Flash y Pila Hardware Circular de Ocho Niveles

1024 Palabras de 14 Bits, Vector de Reset 0x0000, Vector de Interrupción 0x0004 y Control de Subrutinas

- **Turno 1 (Orus):** La memoria Flash de programa del PIC dieciséis efe ochenta y cuatro almacena hasta mil veinticuatro instrucciones de catorce bits cada una, direccionadas linealmente desde la localidad cero hasta la tres efe efe hexadecimal.
- **Turno 2 (Aoede):** Dos direcciones físicas están reservadas por hardware: la posición cero es el vector de reinicio donde comienza todo programa tras encender el equipo, y la posición cuatro es el vector único de atención a interrupciones.
- **Turno 3 (Orus):** Asociada al contador de programa existe una pila dedicada en hardware de ocho niveles de profundidad que no ocupa memoria de datos y sirve para guardar la dirección de retorno al invocar subrutinas o interrupciones.
- **Turno 4 (Aoede):** Como esta pila es circular de ocho entradas, debemos cuidar en nuestro diseño de software no anidar más de ocho llamadas simultáneas a subrutinas para evitar sobrescribir la primera dirección de retorno.

Diapositiva 05: Memoria de Datos RAM: Bancos 0 y 1, Registros Especiales y STATUS

68 Bytes de Propósito General GPR, Selección de Banco con el Bit RP0 de STATUS y Mapeo de Registros SFR

- **Turno 1 (Orus):** La memoria volátil de datos de ocho bits se divide en dos secciones: los registros de función especial, que configuran y gobiernan el hardware interno, y sesenta y ocho registros de propósito general donde residen nuestras variables.
- **Turno 2 (Aoede):** Esta memoria está organizada en dos columnas llamadas Banco cero y Banco uno; para conmutar entre ambos bancos en lenguaje de bajo nivel se modifica el quinto bit, llamado erre pe cero, dentro del registro STATUS.
- **Turno 3 (Orus):** En el Banco cero encontramos los registros operativos de lectura y escritura cotidiana, tales como el temporizador cero y los registros de datos del puerto A en la dirección cinco y del puerto B en la dirección seis.
- **Turno 4 (Aoede):** En las posiciones espejo del Banco uno se alojan sus respectivos registros gemelos de configuración: el registro OPTION y los registros de dirección de datos TRIS A en ochenta y cinco hexadecimal y TRIS B en ochenta y seis.

Diapositiva 06: Arquitectura Eléctrica del Puerto A y del Puerto B: Detalle de RA4

Pines Bidireccionales CMOS, Salida de Drenador Abierto en RA4/T0CKI y Capacidad de Corriente de 25 mA

- **Turno 1 (Orus):** Analicemos las particularidades eléctricas de los dos puertos: cada bit de los registros TRIS actúa como una válvula de dirección donde escribir un uno lógico configura el pin como entrada de alta impedancia, recordando uno igual a Input.
- **Turno 2 (Aoede):** Por el contrario, escribir un cero lógico en un bit del registro TRIS configura dicho terminal como salida baja impedancia, recordando cero igual a Output, capaz de entregar o drenar hasta veinticinco miliamperios.
- **Turno 3 (Orus):** Existe una trampa clásica en el pin erre a cuatro del puerto A que causa dolores de cabeza a quien no lee la hoja de datos: cuando se configura como salida, su etapa interna es de drenador abierto.
- **Turno 4 (Aoede):** Esto implica que el pin erre a cuatro puede conectar cargas perfectamente a tierra cuando escribe un cero, pero jamás entrega cinco voltios por sí solo al escribir un uno a menos que coloquemos una resistencia externa hacia positivo.

Diapositiva 07: Memoria EEPROM de Datos, Temporizador TMR0 y Perro Guardián Watchdog

64 Bytes No Volátiles para Parámetros, Contador de 8 Bits con Preescalador y Supervisión Anticolgado WDT

- **Turno 1 (Orus):** Además de la memoria Flash de programa, el PIC dieciséis efe ochenta y cuatro integra sesenta y cuatro bytes de memoria EEPROM de datos capaz de retener contraseñas, conteos de piezas o consignas hasta por cuarenta años sin energía.
- **Turno 2 (Aoede):** Para medir intervalos o contar eventos externos disponemos del módulo temporizador contador cero de ocho bits, acompañado de un divisor programable o preescalador capaz de dividir la frecuencia desde dos hasta doscientos cincuenta y seis.
- **Turno 3 (Orus):** Por último, el temporizador perro guardián, o Watchdog Timer, posee su propio oscilador independiente en el chip y vigila continuamente que el software no quede atrapado en un bloqueo por ruido eléctrico industrial.
- **Turno 4 (Aoede):** Si el perro guardián está habilitado y nuestro código no lo reinicia periódicamente dentro de unos dieciocho milisegundos nominales, el hardware reinicia automáticamente todo el microcontrolador devolviéndolo a operación segura.

Diapositiva 08: Flujo Completo para Programar un PIC: Del Editor de Texto al Silicio

Código Fuente en Ensamblador o PicBasic Pro, Compilación Cruzada, Ensamblado MPASM y Verificación de Errores

- **Turno 1 (Orus):** ¿Cómo trasladamos nuestras ideas lógicas al interior de la memoria Flash del microcontrolador? El proceso de desarrollo cruzado se realiza en una computadora personal a través de una cadena ordenada de cuatro etapas.
- **Turno 2 (Aoede):** En la primera etapa escribimos el programa fuente en un entorno de edición integrado como MicroCode Studio o MPLAB, utilizando ya sea lenguaje ensamblador de bajo nivel o un lenguaje estructurado como PicBasic Pro.
- **Turno 3 (Orus):** En la segunda etapa ejecutamos el compilador de PicBasic Pro, el cual verifica la sintaxis de nuestras instrucciones, asigna direcciones en la memoria RAM a las variables y traduce el algoritmo a instrucciones equivalentes en ensamblador.
- **Turno 4 (Aoede):** En la tercera etapa el ensamblador convierte esos mnemónicos simbólicos en palabras binarias exactas de catorce bits para la arquitectura del PIC, generando un listado de diagnóstico y el archivo ejecutable final.

Diapositiva 09: El Archivo Estándar Intel HEX y los Bits de Configuración Fuses

Formato de Registros Hexadecimales con Suma de Comprobación y Palabra de Configuración en 0x2007

- **Turno 1 (Orus):** El resultado universal de la compilación es un archivo de texto plano con extensión punto H E X bajo el estándar Intel, donde cada línea codifica la cantidad de bytes, la dirección destino, los códigos máquina y una suma de verificación.
- **Turno 2 (Aoede):** Dentro de este mismo archivo o en el software grabador debemos definir de forma obligatoria la palabra de configuración alojada en la dirección especial dos mil siete hexadecimal, conocida popularmente como los fusibles del PIC.
- **Turno 3 (Orus):** Estos bits de configuración determinan cuatro decisiones hardware antes de arrancar: el tipo de oscilador conectado, la activación o desactivación del perro guardián, la habilitación del temporizador de encendido y la protección de lectura de código.
- **Turno 4 (Aoede):** Un error frecuente en el laboratorio es grabar un programa perfecto dejando seleccionado por accidente el oscilador R C cuando en la protoboard hay un cristal de cuarzo X T, provocando que el microcontrolador permanezca totalmente inerte.

Diapositiva 10: Grabación Física y Programación Serial en Circuito ICSP

Programadores USB tipo PICkit, Tensión de Programación VPP en MCLR y Líneas Seriales PGD en RB7 y PGC en RB6

- **Turno 1 (Orus):** La cuarta etapa del flujo consiste en transferir el archivo hexadecimal hacia la memoria Flash mediante un hardware programador conectado por puerto U S B, como la familia PICkit o grabadores de zócalo de fuerza de inserción nula.
- **Turno 2 (Aoede):** El método profesional más eficiente es la programación serial en circuito, conocida como I C S P, la cual permite reprogramar el microcontrolador directamente montado en su tarjeta utilizando únicamente cinco líneas eléctricas.
- **Turno 3 (Orus):** Para colocar al PIC en modo de grabación, el programador eleva el pin cuatro de reinicio a una tensión especial de programación ve pe pe cercana a trece voltios, además de conectar alimentación positiva y tierra.
- **Turno 4 (Aoede):** A continuación transmite en serie los bits del programa utilizando el pin erre be seis como reloj de programación y el pin erre be siete como línea bidireccional de datos, verificando palabra por palabra la grabación.

2. Código Fuente en PicBasic Pro (picbasic_pro_video_02.pbp)

```
' ================================================================================
' CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
' Video 02 – Parte 2: El Microcontrolador PIC16F84 y Programación de un PIC
' Enfoque: Pinout DIP-18, Oscilador XT/HS/RC, Reset MCLR y PWRT 72 ms, Flash 1Kx14, Bancos SFR 0/1, RA4 Open-Drain, HEX e ICSP
' Ecuaciones Clave: Flash = 1024 x 14 bits  |  RAM_GPR = 68 bytes  |  EEPROM = 64 bytes  |  T_TMR0 = 4 · PS · (256 - TMR0) / f_osc
' Microcontrolador Objetivo: PIC16F84A @ 4.000 MHz (1 ciclo = 1 us)
' ================================================================================
@ DEVICE pic16F84A, XT_OSC, WDT_OFF, PWRT_ON, PROTECT_OFF
Include "modedefs.bas"
DEFINE OSC 4

' --- Declaracion de Pines y Variables ---
EntradaSensor  VAR PORTA.0
SalidaRele     VAR PORTB.0
LedHeartbeat   VAR PORTB.7
TxDiagnostico  VAR PORTA.1
ContadorCiclos VAR WORD
Estado         VAR BYTE

' --- Inicializacion Segura de Puertos (Estado antes que Direccion TRIS) ---
PORTA = %00000000
PORTB = %00000000
OPTION_REG.7 = 0            ' Pull-ups internos activos en PORTB
TRISA = %00000001           ' RA0 entrada, resto salidas
TRISB = %00000000           ' PORTB como salidas digitales
ContadorCiclos = 0
Estado = 0

' --- Lazo Principal Determinista ---
LazoPrincipal:
    TOGGLE LedHeartbeat
    ContadorCiclos = ContadorCiclos + 1
    IF EntradaSensor = 0 THEN
        PAUSE 20            ' Filtro antirrebote 20 ms
        IF EntradaSensor = 0 THEN
            HIGH SalidaRele
            Estado = 1
        ENDIF
    ELSE
        LOW SalidaRele
        Estado = 0
    ENDIF
    SEROUT TxDiagnostico, T2400, ["V02 Ciclo=", #ContadorCiclos, " Est=", #Estado, 13, 10]
    PAUSE 100
    GOTO LazoPrincipal
    END
```

3. Referencia en Ensamblador MPASM (ensamblador_mpasm_video_02.asm)

```
; ==========================================================================
; CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
; Video 02 – Parte 2: El Microcontrolador PIC16F84 y Programación de un PIC – Referencia de Bajo Nivel MPASM
(PIC16F84A)
; ==========================================================================
    LIST        P=16F84A
    #INCLUDE    <P16F84A.INC>
    __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF

    CBLOCK  0x0C
        W_TEMP
        STATUS_TEMP
        CONTADOR
    ENDC

    ORG     0x0000
    GOTO    INICIO

    ORG     0x0004              ; Vector de Interrupcion Hardware
    MOVWF   W_TEMP              ; Salvado de contexto
    SWAPF   STATUS, W
    MOVWF   STATUS_TEMP
    BCF     INTCON, INTF        ; Limpiar bandera RB0/INT
    SWAPF   STATUS_TEMP, W
    MOVWF   STATUS
    SWAPF   W_TEMP, F
    SWAPF   W_TEMP, W
    RETFIE

INICIO:
    BCF     STATUS, RP0         ; Banco 0
    CLRF    PORTA
    CLRF    PORTB
    BSF     STATUS, RP0         ; Banco 1
    MOVLW   B'00000001'
    MOVWF   TRISA               ; RA0 entrada
    CLRF    TRISB               ; PORTB salidas
    BCF     STATUS, RP0         ; Regresar a Banco 0

BUCLE:
    BTFSS   PORTA, 0
    BSF     PORTB, 0
    BTFSC   PORTA, 0
    BCF     PORTB, 0
    GOTO    BUCLE
    END
```

4. Equivalencia en Lenguaje C XC8 (firmware_xc8_video_02.c)

```c
/* ================================================================================
 * CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
 * Video 02 – Parte 2: El Microcontrolador PIC16F84 y Programación de un PIC – Equivalencia en C Embebido (XC8)
 * ================================================================================ */
#include <xc.h>
#define _XTAL_FREQ 4000000UL

#pragma config FOSC = XT, WDTE = OFF, PWRTE = ON, CP = OFF

void main(void) {
    PORTA = 0x00;
    PORTB = 0x00;
    OPTION_REGbits.nRBPU = 0;   /* Pull-ups internos activos en PORTB */
    TRISA = 0b00000001;         /* RA0 como entrada */
    TRISB = 0b00000000;         /* RB0..RB7 como salidas */

    while (1) {
        PORTBbits.RB7 ^= 1;     /* LED Heartbeat */
        if (PORTAbits.RA0 == 0) {
            __delay_ms(20);
            if (PORTAbits.RA0 == 0) {
                PORTBbits.RB0 = 1;
            }
        } else {
            PORTBbits.RB0 = 0;
        }
        __delay_ms(100);
    }
}
```