CURSO DE PROGRAMACIÓN E INTERFAZ DE MICROCONTROLADORES PIC

Video 03 — Parte 3: PicBasic Pro — Fundamentos de Programación · Plataforma Oficial: cursosmecatronica.com

Módulo: Parte 3 — PicBasic Pro: Fundamentos de Programación
Enfoque de Ingeniería: DEFINE OSC, Variables BIT/BYTE/WORD, Constantes CON, Alias VAR, TRISA/TRISB, HIGH/LOW/PAUSE, Operadores 16-bit, IF, FOR, GOSUB

```
BIT: 0..1 | BYTE: 0..255 | WORD: 0..65535 | Escala: Y = (X */ K_256) | Pila Hardware = 8 niveles
```

1. Desarrollo Técnico Diapositiva por Diapositiva

Diapositiva 01: Introducción a PicBasic Pro: Productividad y Control Directo de Hardware

Comparativa frente al Ensamblador MPASM, Librerías Optimizadas de Periféricos y Eficiencia de Código

- **Turno 1 (Orus):** Programar directamente en lenguaje ensamblador exige gestionar manualmente los cambios de banco de memoria, las sumas de dieciséis bits acarreo por acarreo y cientos de líneas para tareas cotidianas como escribir en una pantalla.
- **Turno 2 (Aoede):** PicBasic Pro resuelve esta complejidad ofreciendo un compilador profesional de alto nivel cuya sintaxis legible multiplica la velocidad de desarrollo sin sacrificar el acceso directo a cada registro interno del microcontrolador.
- **Turno 3 (Orus):** A diferencia de los intérpretes lentos que requieren una memoria especial pregrabada, PicBasic Pro es un compilador nativo real que genera código máquina compacto y veloz para cualquier chip estándar de fábrica.
- **Turno 4 (Aoede):** Además incorpora instrucciones de alto nivel listas para usar en comunicación serial, pantallas de cristal líquido, modulación de pulsos y lectura de botones, insertando en memoria Flash únicamente las rutinas que nuestro programa invoca.

Diapositiva 02: Estructura General de un Programa en PicBasic Pro

Encabezado de Configuración, Directivas DEFINE OSC, Declaraciones, Inicialización de Puertos y Lazo Infinito

- **Turno 1 (Orus):** Un programa bien estructurado en PicBasic Pro se organiza siempre en cuatro bloques secuenciales claros, documentados con comentarios que inician mediante una comilla simple o la palabra reservada R E M sin ocupar un solo bit en el chip.
- **Turno 2 (Aoede):** En el primer bloque colocamos las directivas en mayúsculas DEFINE, comenzando por DEFINE O S C seguido de la frecuencia en megahercios, por ejemplo cuatro, para que todos los retardos y baudios se calculen con exactitud.
- **Turno 3 (Orus):** En el segundo bloque declaramos todas nuestras constantes simbólicas, variables en memoria RAM y nombres descriptivos para los pines físicos de entrada y salida; seguido del tercer bloque que inicializa los registros TRIS y estados seguros.
- **Turno 4 (Aoede):** El cuarto bloque contiene el lazo principal infinito delimitado por una etiqueta de inicio y un salto al final, garantizando que el microcontrolador supervise y controle el sistema ininterrumpidamente mientras reciba energía.

Diapositiva 03: Tipos de Datos y Variables en RAM: BIT, BYTE, WORD y Arreglos

Rangos Enteros sin Signo de 0..1, 0..255 y 0..65535, Consumo de los 68 Bytes de RAM e Indexación

- **Turno 1 (Orus):** Dado que el PIC dieciséis efe ochenta y cuatro cuenta únicamente con sesenta y ocho bytes de memoria RAM de usuario, elegir el tamaño mínimo indispensable para cada variable mediante la palabra clave V A R es una disciplina esencial.
- **Turno 2 (Aoede):** PicBasic Pro maneja tres tamaños enteros sin signo: las variables tipo BIT almacenan cero o uno empaquetando ocho banderas dentro de un solo byte; las variables tipo BYTE ocupan ocho bits y abarcan de cero a doscientos cincuenta y cinco.
- **Turno 3 (Orus):** Cuando necesitamos cuentas mayores, como lecturas de tiempo en milisegundos o cálculos intermedios, declaramos variables tipo WORD de dieciséis bits, capaces de representar números enteros desde cero hasta sesenta y cinco mil quinientos treinta y cinco.
- **Turno 4 (Aoede):** También podemos declarar arreglos indexados indicando entre corchetes la cantidad de elementos, como datos V A R BYTE corchete ocho, accediendo desde el índice cero hasta el índice siete y pudiendo consultar cualquier bit individual con punto cero hasta punto siete.

Diapositiva 04: Constantes CON, Alias de Pines VAR y Sistemas de Numeración

Legibilidad de Hardware sin Costo en RAM y Prefijos Hexadecimal Signo Dólar y Binario Porcentaje

- **Turno 1 (Orus):** Nunca debemos escribir números mágicos ni nombres crípticos de pines dispersos por todo el programa: la directiva C O N asocia un nombre fijo en tiempo de compilación, como LimiteTemp C O N ochenta, sin gastar ni un solo byte de memoria RAM.
- **Turno 2 (Aoede):** De igual modo, la palabra V A R nos permite crear alias semánticos para terminales físicos, por ejemplo Motor Bomba V A R P O R T B punto tres, de suerte que si mañana cambiamos el cableado en la placa solo editamos una línea.
- **Turno 3 (Orus):** En programación de hardware alternamos constantemente entre tres bases numéricas según convenga: escribimos números decimales de forma directa, anteponemos el símbolo dólar para valores hexadecimales como dólar efe e, y usamos el símbolo porcentaje para máscaras binarias.
- **Turno 4 (Aoede):** La notación binaria con porcentaje de ocho dígitos, como porcentaje cero cero cero cero uno uno uno uno, resulta extraordinariamente clara al configurar registros de puertos porque cada uno o cero corresponde visualmente a una patita física del integrado.

Diapositiva 05: Configuración de Puertos con TRISA y TRISB y Acceso a PORTA y PORTB

Escritura Paralela de 8 Bits, Lectura de Entradas y Orden Seguro de Inicialización sin Pulsos Falsos

- **Turno 1 (Orus):** En PicBasic Pro todos los registros de función especial están predefinidos por su nombre oficial sin preocuparnos por cambiar manualmente de banco de memoria: el compilador inserta las instrucciones de cambio de banco de forma transparente.
- **Turno 2 (Aoede):** Al arrancar nuestro programa siempre es una regla de oro de seguridad escribir primero el estado inicial apagado en los registros de datos P O R T A igual a cero y P O R T B igual a cero antes de configurar las direcciones.
- **Turno 3 (Orus):** Inmediatamente después configuramos los registros T R I S A y T R I S B; por ejemplo, T R I S B igual a porcentaje uno uno uno uno cero cero cero cero define los cuatro pines superiores como entradas y los cuatro inferiores como salidas.
- **Turno 4 (Aoede):** Invertir ese orden y habilitar primero las salidas cuando el registro del puerto contiene basura aleatoria del encendido puede disparar durante una fracción de segundo un relé, una válvula o el puente de un motor de potencia.

Diapositiva 06: Comandos Digitales Básicos y Retardos: HIGH, LOW, TOGGLE y PAUSE

Configuración Automática de Dirección en Comandos de Bit y Temporización en Milisegundos y Microsegundos

- **Turno 1 (Orus):** Para manipular líneas individuales PicBasic Pro ofrece cinco comandos directos: HIGH pone un pin en cinco voltios, LOW lo lleva a cero voltios, TOGGLE invierte su estado actual, OUTPUT lo fija como salida e INPUT lo deja en alta impedancia.
- **Turno 2 (Aoede):** Un detalle importante que debemos conocer es que ejecutar HIGH, LOW o TOGGLE sobre un pin no solo cambia su estado lógico, sino que escribe automáticamente un cero en su bit TRIS correspondiente para garantizar que sea salida.
- **Turno 3 (Orus):** Para sincronizar acciones visibles al ojo humano disponemos del comando PAUSE seguido de un valor de hasta sesenta y cinco mil quinientos treinta y cinco milisegundos; por ejemplo, PAUSE quinientos detiene la ejecución exactamente medio segundo.
- **Turno 4 (Aoede):** Cuando requerimos pulsos rápidos de disparo para sensores ultrasónicos o protocolos digitales usamos PAUSEUS, el cual genera retardos precisos en microsegundos con una resolución mínima de veinticuatro microsegundos usando un cristal de cuatro megahercios.

Diapositiva 07: Operadores Aritméticos Enteros de 16 Bits, Módulo y Lógica Bit a Bit

Evaluación Estricta de Izquierda a Derecha, División Entera, Residuo, Desplazamientos y Máscaras Binarias

- **Turno 1 (Orus):** Toda la aritmética interna de PicBasic Pro opera con precisión entera sin signo de dieciséis bits y presenta una particularidad crucial frente a otros lenguajes: evalúa las expresiones estrictamente de izquierda a derecha, salvo que usemos paréntesis explícitos.
- **Turno 2 (Aoede):** Al no existir punto flotante para ahorrar memoria Flash, trabajamos con escalado entero: la barra diagonal entrega el cociente entero truncado, mientras que la doble barra diagonal calcula el residuo o módulo, ideal para separar decenas y unidades.
- **Turno 3 (Orus):** Además contamos con operadores avanzados de escalado como asterisco asterisco y asterisco barra diagonal, que permiten multiplicar una variable de dieciséis bits por una constante fraccionaria utilizando un producto intermedio de treinta y dos bits sin desbordar.
- **Turno 4 (Aoede):** Para manipular registros hardware empleamos desplazamientos a izquierda y derecha junto con los operadores bit a bit ampersand para enmascarar entradas, barra vertical para encender bits y acento circunflejo para invertir estados selectivamente.

Diapositiva 08: Estructuras Condicionales: IF-THEN-ELSE-ENDIF y SELECT CASE

Operadores Relacionales, Conectores Lógicos AND/OR/NOT y Selección Limpia de Múltiples Estados

- **Turno 1 (Orus):** La inteligencia de control surge al evaluar entradas y variables mediante el condicional IF seguido de una comparación como igual, diferente, mayor o menor igual, pudiendo combinar varias condiciones con los operadores lógicos AND y OR.
- **Turno 2 (Aoede):** En su forma compacta de una línea, IF seguido de THEN salta directamente a una etiqueta; pero para escribir código limpio y estructurado siempre preferimos el bloque multilínea IF, THEN, ELSEIF opcional, ELSE y cierre obligatorio con ENDIF.
- **Turno 3 (Orus):** Cuando una variable selectora de modo o una tecla leída puede tomar cinco o seis caminos diferentes, encadenar múltiples condicionales vuelve difícil la lectura, por lo que empleamos la estructura SELECT CASE variable cerrada con END SELECT.
- **Turno 4 (Aoede):** Cada rama CASE admite un valor puntual, una lista separada por comas o incluso un rango continuo usando la palabra IS, finalizando con CASE ELSE para atrapar de forma segura cualquier valor inesperado.

Diapositiva 09: Estructuras Repetitivas: Bucles FOR-NEXT y Bucles WHILE-WEND

Iteración Contada con Incremento o Decremento STEP y Repetición Condicionada por Sensores

- **Turno 1 (Orus):** Para repetir un conjunto exacto de veces una secuencia de barrido, promediar muestras de un sensor o recorrer un arreglo utilizamos el ciclo contado FOR índice igual inicio TO fin cerrado con la instrucción NEXT índice.
- **Turno 2 (Aoede):** Por defecto el contador avanza de uno en uno, pero podemos añadir el modificador STEP con cualquier paso positivo para saltar posiciones, o bien STEP menos uno para realizar conteos regresivos cuidando no pasar por debajo de cero.
- **Turno 3 (Orus):** Cuando el número de repeticiones no se conoce de antemano sino que depende de una señal física externa, como esperar a que un carro alcance un interruptor de límite, empleamos el bucle condicional WHILE condición cerrado con WEND.
- **Turno 4 (Aoede):** En aplicaciones críticas de ingeniería siempre añadimos dentro del bucle WHILE un contador de tiempo máximo de seguridad para que, si el sensor se desconecta, el sistema salga del bucle y active una alarma controlada.

Diapositiva 10: Modularidad con Subrutinas GOSUB-RETURN frente a Saltos GOTO

Uso Eficiente de la Memoria Flash, Límite de Anidamiento de Pila y Prevención de Caída Accidental en Subrutinas

- **Turno 1 (Orus):** Dividir un programa extenso en tareas reutilizables se logra con el par GOSUB etiqueta y RETURN: al ejecutarse GOSUB, la dirección actual se guarda en la pila hardware del PIC, se ejecuta la rutina y RETURN regresa exactamente a la línea siguiente.
- **Turno 2 (Aoede):** En cambio, la instrucción GOTO realiza un salto definitivo sin guardar dirección de retorno, por lo que debemos reservarla únicamente para cerrar el lazo infinito principal del programa y evitar el llamado código espagueti.
- **Turno 3 (Orus):** Una regla crítica de arquitectura en PicBasic Pro es ubicar todas las subrutinas debajo del salto final del lazo principal, asegurando que el flujo normal jamás entre por accidente en una subrutina y ejecute un RETURN huérfano.
- **Turno 4 (Aoede):** Dominados los fundamentos del lenguaje, en la Parte cuatro construiremos ejemplos prácticos completos con actuadores y comunicaciones, y aprenderemos a programar interrupciones hardware en tiempo real.

2. Código Fuente en PicBasic Pro (picbasic_pro_video_03.pbp)

```
' ================================================================================
' CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
' Video 03 – Parte 3: PicBasic Pro – Fundamentos de Programación
' Enfoque: DEFINE OSC, Variables BIT/BYTE/WORD, Constantes CON, Alias VAR, TRISA/TRISB, HIGH/LOW/PAUSE,
Operadores 16-bit, IF, FOR, GOSUB
' Ecuaciones Clave: BIT: 0..1  |  BYTE: 0..255  |  WORD: 0..65535  |  Escala: Y = (X */ K_256)  |  Pila Hardware
= 8 niveles
' 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, ["V03 Ciclo=", #ContadorCiclos, " Est=", #Estado, 13, 10]
    PAUSE 100
    GOTO LazoPrincipal
    END
```

3. Referencia en Ensamblador MPASM (ensamblador_mpasm_video_03.asm)

```
; ==========================================================================
; CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
; Video 03 – Parte 3: PicBasic Pro – Fundamentos de Programación – 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_03.c)

```
/* ============================================================================
 * CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
 * Video 03 – Parte 3: PicBasic Pro – Fundamentos de Programación – 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);
    }
}
```