MANUAL DE CONFIGURACIÓN

dsPIC33FJ32MC204

==========

TABLA DE CONTENIDOS

---

1. 🎯 INTRODUCCIÓN

Este manual explica cómo usar el sistema de configuración modular para el **dsPIC33FJ32MC204**. El sistema permite seleccionar diferentes configuraciones del microcontrolador simplemente descomentando líneas en un archivo.

¿Por qué usar este sistema?

- ☑ **Flexible**: Cambia configuraciones sin modificar código
- ☑ **Documentado**: Cada opción está claramente explicada
- ☑ **Seguro**: Previene configuraciones conflictivas
- ☑ **Reutilizable**: Base para múltiples proyectos
- ☑ **Profesional**: Estructura estándar de la industria

---

2. 🗀 ESTRUCTURA DE ARCHIVOS

```
text

proyecto/
├── config.h              ← ARCHIVO PRINCIPAL (aquí configuras)
├── config_basica.h       ← Configuración predefinida básica
├── config.c              ← Implementación automática
└── main.c                ← Tu aplicación
```

Flujo de trabajo:

1. **Abrir** `config.h`
2. **DESCOMENTAR** las opciones que necesitas
3. **COMENTAR** las opciones que NO necesitas
4. **Compilar** - El sistema aplica automáticamente tu configuración

---

3. ❁ CONFIGURACIONES DISPONIBLES

3.1 OSCILADOR (Clock del Sistema)

¿Qué controla? La fuente y velocidad del reloj principal.

Opciones disponibles:

Opción	Velocidad	Uso Recomendado
`CONFIG_OSC_INTERNO_PLL`	**40 MIPS** (40 MHz)	**DESARROLLO GENERAL** - Máximo rendimiento

Opción	Velocidad	Uso Recomendado
`CONFIG_OSC_INTERNO_SIMPLE`	7.37 MIPS	Bajo consumo, aplicaciones simples
`CONFIG_OSC_EXTERNO_PLL`	Hasta 40 MIPS	Aplicaciones que requieren precisión
`CONFIG_OSC_EXTERNO_SIMPLE`	Frecuencia del cristal	Cuando se necesita clock externo

Ejemplos de uso:

c

```c
// Para máxima velocidad (recomendado):
#define CONFIG_OSC_INTERNO_PLL

// Para ahorrar energía:
// #define CONFIG_OSC_INTERNO_SIMPLE
```

3.2 WATCHDOG TIMER (WDT)

¿Qué controla? Timer que reinicia el sistema si se bloquea.

Opción	Tiempo	Uso Recomendado
`CONFIG_WDT_OFF`	**Deshabilitado**	**DESARROLLO** - Evita reinicios durante debugging
`CONFIG_WDT_ON_NORMAL`	~1ms @ 32kHz	Aplicaciones críticas que no deben bloquearse
`CONFIG_WDT_ON_LONG`	~18 horas	Aplicaciones que corren por días

¿Cuándo usar WDT?

- ☑ **PRODUCCIÓN**: Siempre habilitado
- ☑ **EQUIPOS MÉDICOS/INDUSTRIALES**: Crítico
- ❌ **DESARROLLO**: Deshabilitar para debugging

3.3 BROWN-OUT RESET (BOR)

¿Qué controla? Reinicio automático cuando el voltaje baja.

Opción	Voltaje	Uso Recomendado
`CONFIG_BOR_OFF`	**Deshabilitado**	**DESARROLLO** - Evita reinicios por fluctuaciones
`CONFIG_BOR_27V`	2.7V	Baterías Li-Ion (3.7V nominal)
`CONFIG_BOR_20V`	2.0V	Baterías NiMH (2.4V nominal)
`CONFIG_BOR_42V`	4.2V	Alimentación de 5V con regulador

3.4 MCLR (Master Clear Reset)

¿Qué controla? Pin de reset externo.

Opción	Funcionamiento	Uso Recomendado
`CONFIG_MCLR_ENABLED`	**Pin como RESET**	**GENERAL** - Permite reset externo
`CONFIG_MCLR_DISABLED`	Pin como I/O	Cuando necesitas un pin extra

3.5 PROTECCIÓN DE CÓDIGO

¿Qué controla? Lectura del código desde el exterior.

Opción	Seguridad	Uso Recomendado
`CONFIG_CODE_PROTECT_OFF`	**Sin protección**	**DESARROLLO** - Puedes reprogramar fácilmente
`CONFIG_CODE_PROTECT_ON`	Código protegido	**PRODUCCIÓN** - Protege tu propiedad intelectual

3.6 DEBUG/ICD

¿Qué controla? Pines PGC/PGD para programación y debug.

Opción	PGC/PGD	Uso Recomendado
`CONFIG_DEBUG_OFF`	**Pines como I/O**	**PRODUCCIÓN** - Libera 2 pines
`CONFIG_DEBUG_ON`	Pines para debug	**DESARROLLO** - Para programar y depurar

3.7 PUERTOS I/O

¿Qué controla? Habilitación de puertos para ahorrar energía.

c

```
// Descomenta SOLO los puertos que uses:
// #define CONFIG_PORT_A_ENABLED  // Usa 8 pines
#define CONFIG_PORT_B_ENABLED    // Usa 8 pines
// #define CONFIG_PORT_C_ENABLED  // Usa 8 pines
```

Beneficios:

- ☑ **Menor consumo** (los puertos deshabilitados no consumen)
- ☑ **Menor interferencia** electromagnética

- ☑ **Pines reservados** para futuras expansiones

---

4. 🚀 USO PASO A PASO

PASO 1: Elegir tu configuración

Abre `config.h` y decide qué necesitas:

c

```c
// EJEMPLO: Configuración para desarrollo
#define CONFIG_OSC_INTERNO_PLL    // Máxima velocidad
#define CONFIG_WDT_OFF            // Sin watchdog para debug
#define CONFIG_MCLR_ENABLED       // Reset externo disponible
#define CONFIG_BOR_OFF            // Sin reinicio por bajo voltaje
#define CONFIG_CODE_PROTECT_OFF   // Código reprogramable
#define CONFIG_DEBUG_OFF          // Pines como I/O (usar ICSP)
#define CONFIG_PORT_B_ENABLED     // Solo puerto B activo
```

PASO 2: Incluir en tu proyecto

c

```c
// En tu main.c:
#include "config.h"

int main(void) {
    // Inicializa con tu configuración
    SYSTEM_Initialize();

    // Tu código aquí...

    return 0;
}
```

PASO 3: Compilar

El sistema aplicará automáticamente todas las configuraciones que descomentaste.

---

5. 💡 EJEMPLOS PRÁCTICOS

EJEMPLO 1: DESARROLLO RÁPIDO

c

```c
// config.h - Para prototipado rápido
#define CONFIG_OSC_INTERNO_PLL    // 40 MHz, máximo rendimiento
#define CONFIG_WDT_OFF            // Sin reinicios molestos
#define CONFIG_MCLR_ENABLED       // Reset fácil con botón
#define CONFIG_BOR_OFF            // Evita problemas con fuente de laboratori
o
#define CONFIG_CODE_PROTECT_OFF   // Reprogramar sin restricciones
#define CONFIG_DEBUG_OFF          // Pines libres para I/O
#define CONFIG_PORT_A_ENABLED     // Todos los puertos disponibles
#define CONFIG_PORT_B_ENABLED
#define CONFIG_PORT_C_ENABLED
```

EJEMPLO 2: PRODUCCIÓN INDUSTRIAL

c

```c
// config.h - Para producto final
#define CONFIG_OSC_INTERNO_PLL    // Buen rendimiento
#define CONFIG_WDT_ON_NORMAL      // Protección contra bloqueos
#define CONFIG_MCLR_ENABLED       // Reset externo disponible
#define CONFIG_BOR_27V            // Protección contra bajas de voltaje
#define CONFIG_CODE_PROTECT_ON    // Protege tu código
#define CONFIG_DEBUG_OFF          // Libera pines para la aplicación
#define CONFIG_PORT_B_ENABLED     // Solo puertos necesarios
```

EJEMPLO 3: DISPOSITIVO CON BATERÍA

c

```c
// config.h - Para bajo consumo
```

```
#define CONFIG_OSC_INTERNO_SIMPLE // 7.37 MHz, menos consumo
#define CONFIG_WDT_ON_LONG        // Watchdog muy lento
#define CONFIG_MCLR_DISABLED      // Pin extra como I/O
#define CONFIG_BOR_20V            // Para baterías de 2.4V
#define CONFIG_CODE_PROTECT_ON    // Seguridad en campo
#define CONFIG_DEBUG_OFF          // Mínimo consumo
#define CONFIG_PORT_B_ENABLED     // Solo puerto esencial
```

---

6. 🔧 SOLUCIÓN DE PROBLEMAS

PROBLEMA: No compila después de cambiar configuración

Solución: Verifica que:

1. Solo hay **UNA** opción activa por categoría
2. No hay conflictos (ej: oscilador interno y externo a la vez)
3. Has ejecutado `make clean` antes de recompilar

PROBLEMA: El microcontrolador no arranca

Configuración probablemente incorrecta:

c

```
// Verifica estos ajustes:
#define CONFIG_OSC_INTERNO_PLL    // Para arranque rápido
#define CONFIG_MCLR_ENABLED       // Permite reset
#define CONFIG_BOR_OFF            // Evita reinicios prematuros
```

PROBLEMA: Alto consumo de energía

Posibles causas y soluciones:

Síntoma	Posible Causa	Solución
Consumo > 10mA	Puertos no usados activos	Deshabilitar puertos no usados
Consumo > 50mA	Oscilador muy rápido	Usar `CONFIG_OSC_INTERNO_SIMPLE`
Consumo variable	WDT activo	Usar `CONFIG_WDT_OFF` para testing

PROBLEMA: No puedo reprogramar el micro

Causa probable:

c

```c
#define CONFIG_CODE_PROTECT_ON  // ¡Esto bloquea la reprogramación!
```

Solución:

1. Cambiar a `CONFIG_CODE_PROTECT_OFF`
2. Programar nuevamente
3. Solo activar protección en producción final

---

7. 📊 TABLA DE CONFIGURACIONES RECOMENDADAS

Escenario	Oscilador	WDT	BOR	Código	Debug
Desarrollo	INTERNO_PLL	OFF	OFF	OFF	OFF
Testing	INTERNO_PLL	ON_NORMAL	OFF	OFF	OFF

Escenario	Oscilador	WDT	BOR	Código	Debug
Producción	INTERNO_PLL	ON_NORMAL	27V	ON	OFF
Batería	INTERNO_SIMPLE	ON_LONG	20V	ON	OFF
Industrial	INTERNO_PLL	ON_NORMAL	42V	ON	OFF

---

8. 🎯 MEJORES PRÁCTICAS

1. Durante desarrollo:

c

```c
// Mantener esta configuración:
#define CONFIG_OSC_INTERNO_PLL
#define CONFIG_WDT_OFF
#define CONFIG_CODE_PROTECT_OFF
```

2. Antes de producción:

1. Cambiar `CONFIG_WDT_OFF` → `CONFIG_WDT_ON_NORMAL`
2. Cambiar `CONFIG_CODE_PROTECT_OFF` → `CONFIG_CODE_PROTECT_ON`
3. Activar `CONFIG_BOR_27V` o `CONFIG_BOR_42V` según tu fuente

3. Para depuración avanzada:

c

```c
#define CONFIG_DEBUG_ON  // Permite debugging en circuito
```

```
// Nota: Esto usa los pines PGC/PGD
```

4. Optimización de pines:

c

```
// Descomenta SOLO los puertos que realmente uses
// Esto libera pines y reduce consumo
#define CONFIG_PORT_B_ENABLED  // Si solo usas puerto B
// Los demás puertos quedan deshabilitados
```

---

9. ⚡ CONFIGURACIÓN RÁPIDA (TL;DR)

Para empezar YA (copia y pega en config.h):

c

```
////////////////////////////////////////////////////////////////////////
//
// CONFIGURACIÓN BÁSICA - DESCOMENTAR SOLO UNA OPCIÓN POR BLOQUE
////////////////////////////////////////////////////////////////////////
//

// 1. OSCILADOR (elegir UNO):
#define CONFIG_OSC_INTERNO_PLL     // ← RECOMENDADO: 40 MHz
// #define CONFIG_OSC_INTERNO_SIMPLE // 7.37 MHz, bajo consumo

// 2. WATCHDOG (elegir UNO):
#define CONFIG_WDT_OFF             // ← DESARROLLO: Sin reinicios automáticos
// #define CONFIG_WDT_ON_NORMAL     // PRODUCCIÓN: Protección contra bloqueos

// 3. RESET EXTERNO:
#define CONFIG_MCLR_ENABLED        // ← Pin MCLR funciona como reset
// #define CONFIG_MCLR_DISABLED     // Pin MCLR como I/O normal

// 4. PROTECCIÓN POR BAJO VOLTAJE (BOR):
#define CONFIG_BOR_OFF             // ← DESARROLLO: Sin reinicios por voltaje
// #define CONFIG_BOR_27V           // PRODUCCIÓN: Reinicia si < 2.7V
```

```
// 5. PROTECCIÓN DE CÓDIGO:
#define CONFIG_CODE_PROTECT_OFF     // ← DESARROLLO: Puedes reprogramar
// #define CONFIG_CODE_PROTECT_ON    // PRODUCCIÓN: Código protegido

// 6. DEBUG/ICD:
#define CONFIG_DEBUG_OFF            // ← Pines PGC/PGD como I/O normales
// #define CONFIG_DEBUG_ON           // Pines para programador/debugger

// 7. PUERTOS ACTIVOS (descomentar los que uses):
#define CONFIG_PORT_B_ENABLED       // ← Puertos que necesitas
// #define CONFIG_PORT_A_ENABLED
// #define CONFIG_PORT_C_ENABLED

////////////////////////////////////////////////////////////////////////////
//
// ¡NO MODIFICAR NADA DEBAJO DE ESTA LÍNEA!
////////////////////////////////////////////////////////////////////////////
//
```

---

10. 📞 SOPORTE

Problemas comunes y soluciones:

Q: ¿Por qué mi programa corre más lento de lo esperado?

A: Verifica que tienes `CONFIG_OSC_INTERNO_PLL` activo,
no `CONFIG_OSC_INTERNO_SIMPLE`.

Q: ¿Por qué el micro se reinicia solo?

A: Probablemente tienes `CONFIG_WDT_ON_NORMAL` o `CONFIG_BOR_XXV` activo. Para desarrollo, usa `CONFIG_WDT_OFF` y `CONFIG_BOR_OFF`.

Q: ¿Por qué no puedo usar todos los pines?

A: Solo los puertos descomentados están habilitados.
Descomenta `CONFIG_PORT_A_ENABLED`, etc.

Q: ¿Cómo sé qué configuración está activa?
A: Usa `SYSTEM_PrintConfiguration()` en tu código para ver un resumen.

Para ayuda adicional:

1. Revisa los mensajes de compilación
2. Usa `SYSTEM_PrintConfiguration()` para diagnóstico
3. Verifica que solo una opción por categoría está descomentada

---

📋 RESUMEN FINAL

Para DESARROLLO (lo más fácil):

1. **Descomenta** `CONFIG_OSC_INTERNO_PLL`
2. **Descomenta** `CONFIG_WDT_OFF`
3. **Descomenta** `CONFIG_CODE_PROTECT_OFF`
4. **Descomenta** los puertos que uses
5. **Compila** y programa

Para PRODUCCIÓN (antes de enviar):

1. **Cambia** `CONFIG_WDT_OFF` por `CONFIG_WDT_ON_NORMAL`
2. **Cambia** `CONFIG_CODE_PROTECT_OFF` por `CONFIG_CODE_PROTECT_ON`
3. **Activa** `CONFIG_BOR_27V` o `CONFIG_BOR_42V`
4. **Vuelve a compilar** y programa por última vez

Última actualización: 04/12/2025
Versión del manual: 1.0

Microcontrolador: dsPIC33FJ32MC204

Compilador: XC16 v1.70+