INTERRUPCIONES

Interrupciones

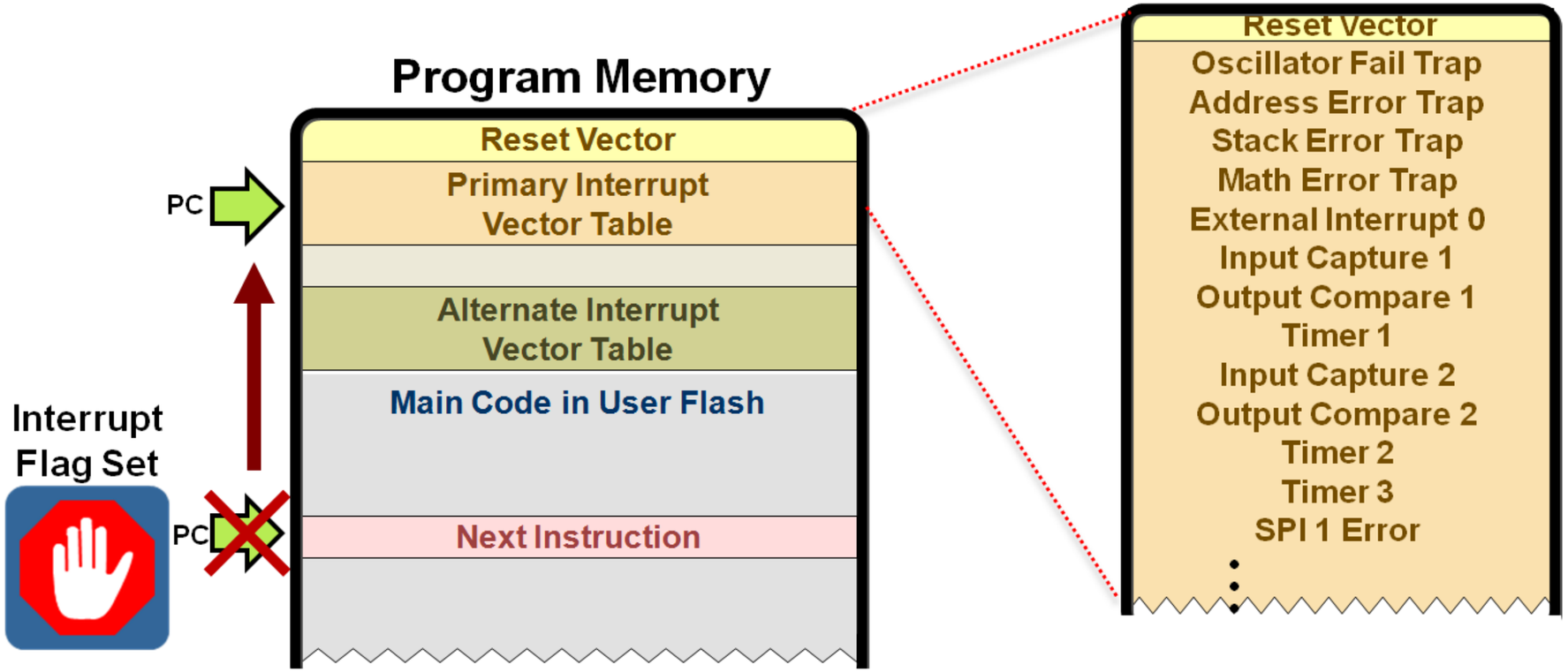
Técnica Polling: La CPU del microcontrolador consulta constantemente el componente/dispositivo de hardware que emite una señal para ser atendida. Por lo general esta técnica degrada el rendimiento del sistema.

Interrupciones: La CPU del microcontrolador no necesita consultar constantemente el componente/dispositivo de hardware ya que el sistema de interrupción le avisará cuando éste tenga una señal para ser atendida por la CPU y atender la petición.

Interrupciones

Exepciones

Las excepciones son eventos asincrónicos impulsados por hardware que hacen que la MCU se desvíe de la ejecución normal del código.

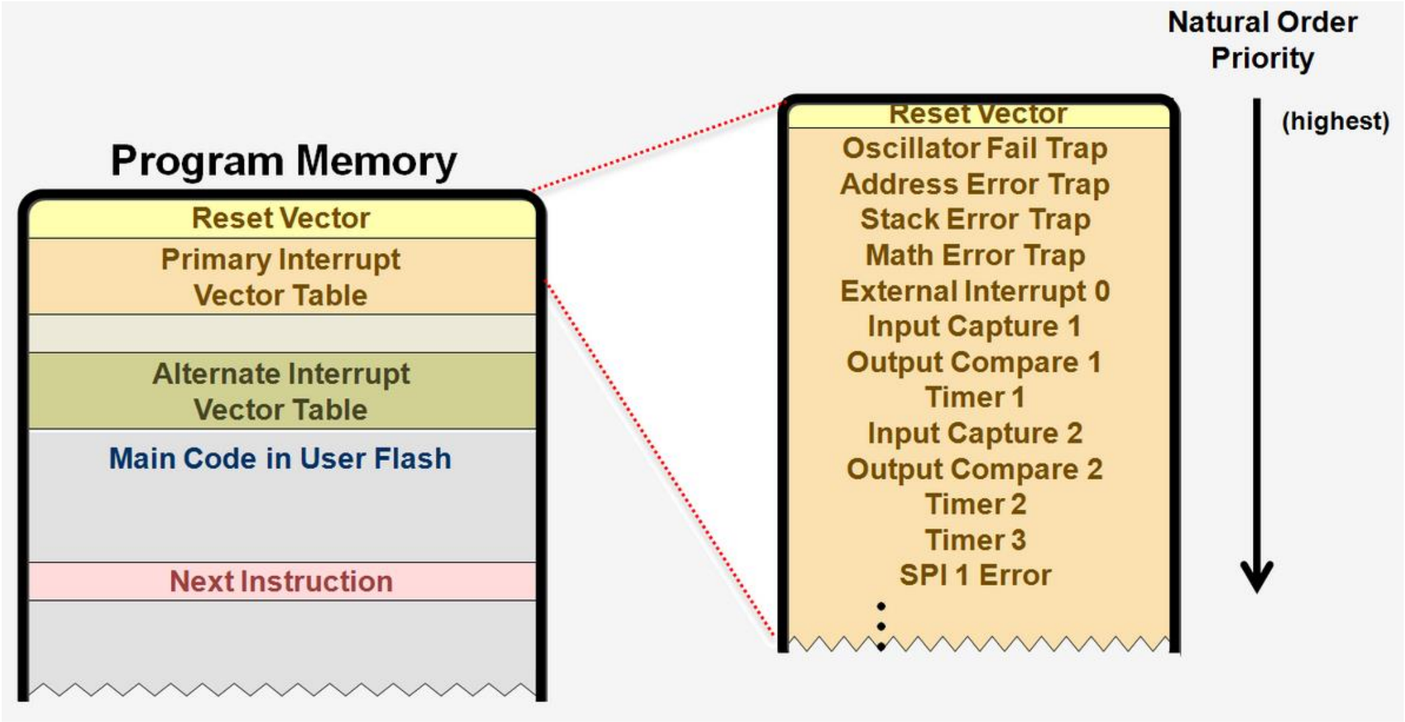


Interrupciones

La Arquitectura de 16 bits que poseen los dsPIC tienen un esquema de excepción vectorial con soporte para hasta ocho fuentes de trampas no enmascarables y hasta 246 fuentes de interrupción. En ambas familias, cada fuente de interrupción puede asignarse a uno de los siete niveles de prioridad.
La latencia de interrupción básica es de cuatro ciclos de instrucción al ingresar y tres ciclos de salida de una Rutina de servicio de interrupción (ISR) .

Esquema de excepción vectorial

Cada fuente de interrupción puede desencadenar la ejecución de una pieza única de código, denominada rutina de servicio de interrupción.
La dirección de inicio de cada ISR (también llamada vector) se almacena en la Tabla de vectores de interrupción primaria (IVT) como se muestra:



Tipos de excepciones

Trampas no enmascarable:

Las trampas se pueden considerar como interrupciones encajables y no enmascarables que se adhieren a una estructura de prioridad fija. Están destinados a detectar determinados problemas de hardware y software.

- Trampa de falla del oscilador
- Trampa de error de pila
- Trampa de error de dirección
- Trampa de error aritmético

Interrupciones periféricas y externas

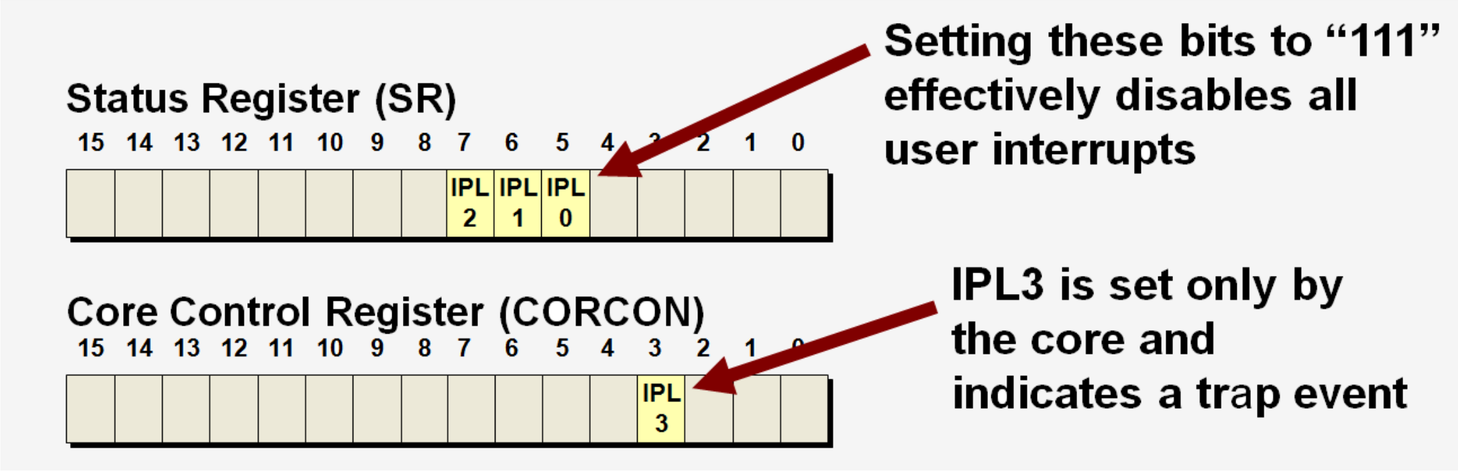
Estas son las solicitudes de interrupción regulares y enmascarables que provienen de una variedad de periféricos MCU implementados:

- Pines de interrupción externos
- Captura de entrada / Comparación de salida
- Interfaces de comunicación (UART / SPI / I²C / USB / Ethernet)
- Puerto maestro paralelo (PMP)
- E / S analógicas (comparador, ADC / DAC)
- etc...

Prioridad de CPU

La CPU de 16 bits puede funcionar en uno de los 16 niveles de prioridad (0-15). Una fuente de interrupción o trampa debe tener un nivel de prioridad establecido mayor que la prioridad actual de la CPU para iniciar un proceso de excepción.

Los niveles de prioridad para las fuentes de interrupciones periféricas y externas se pueden programar en los niveles 0-7, mientras que los niveles de prioridad de la CPU 8-15 están reservados para fuentes de trampa y son fijos .

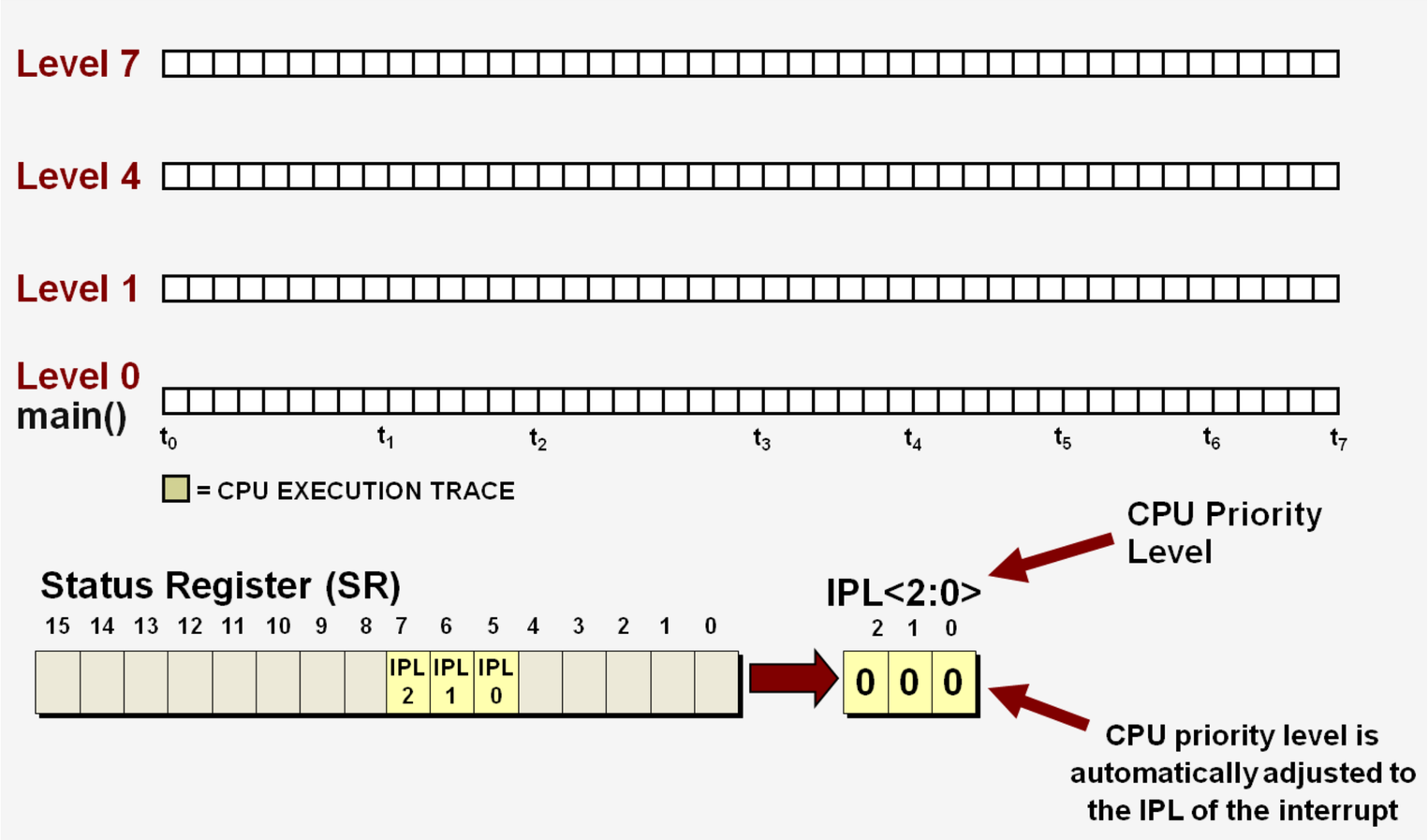


Interrumpir anidamiento

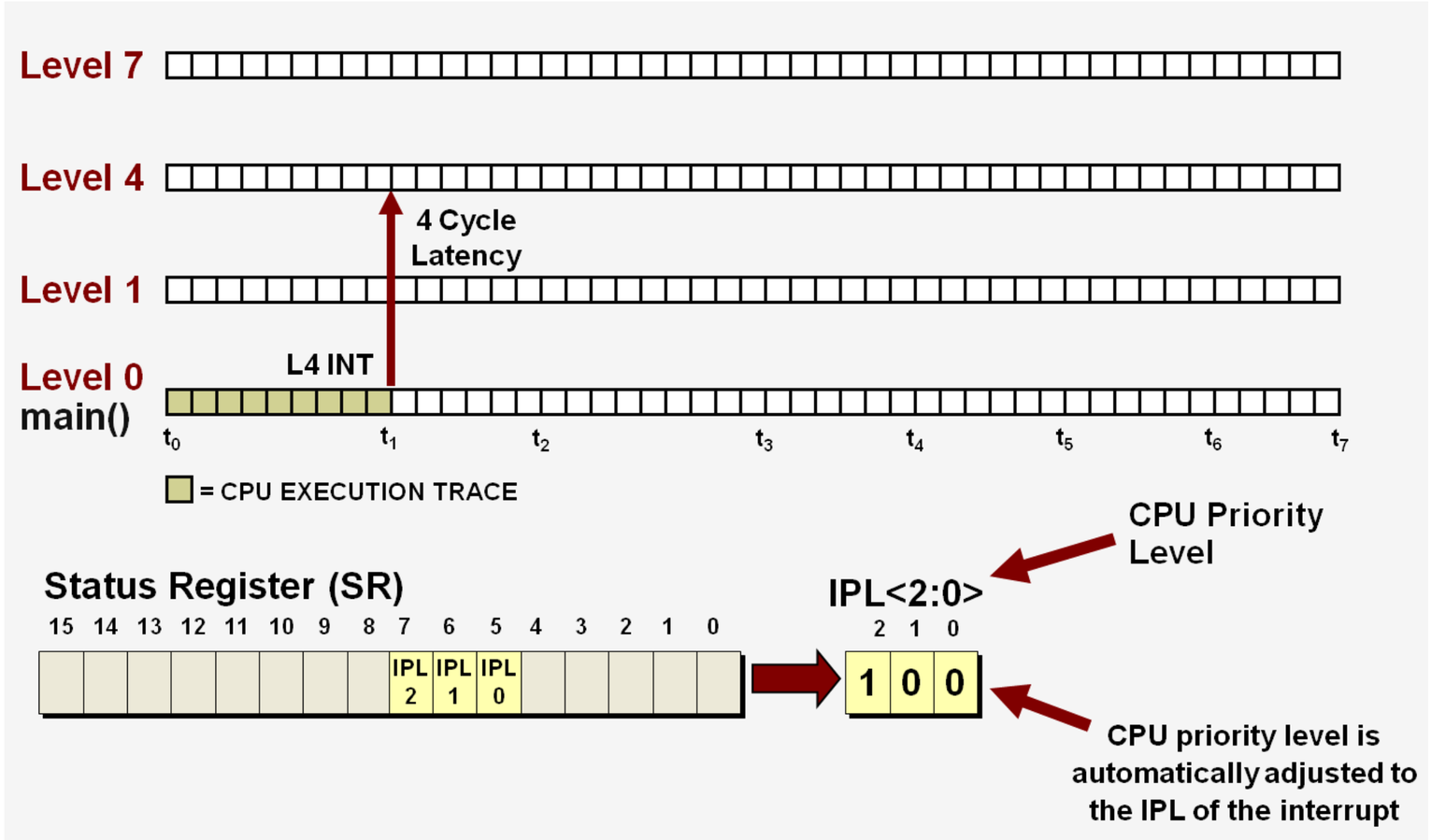
Las interrupciones, por defecto, son encajables . Cualquier ISR que esté en progreso puede ser interrumpido por otra fuente de interrupción que tenga un nivel de prioridad programado más alto.

La siguiente animación demuestra un escenario específico que involucra el subproceso de ejecución main () junto con tres subprocesos ISR preprogramados en diferentes niveles de prioridad con el anidamiento de interrupciones habilitado.

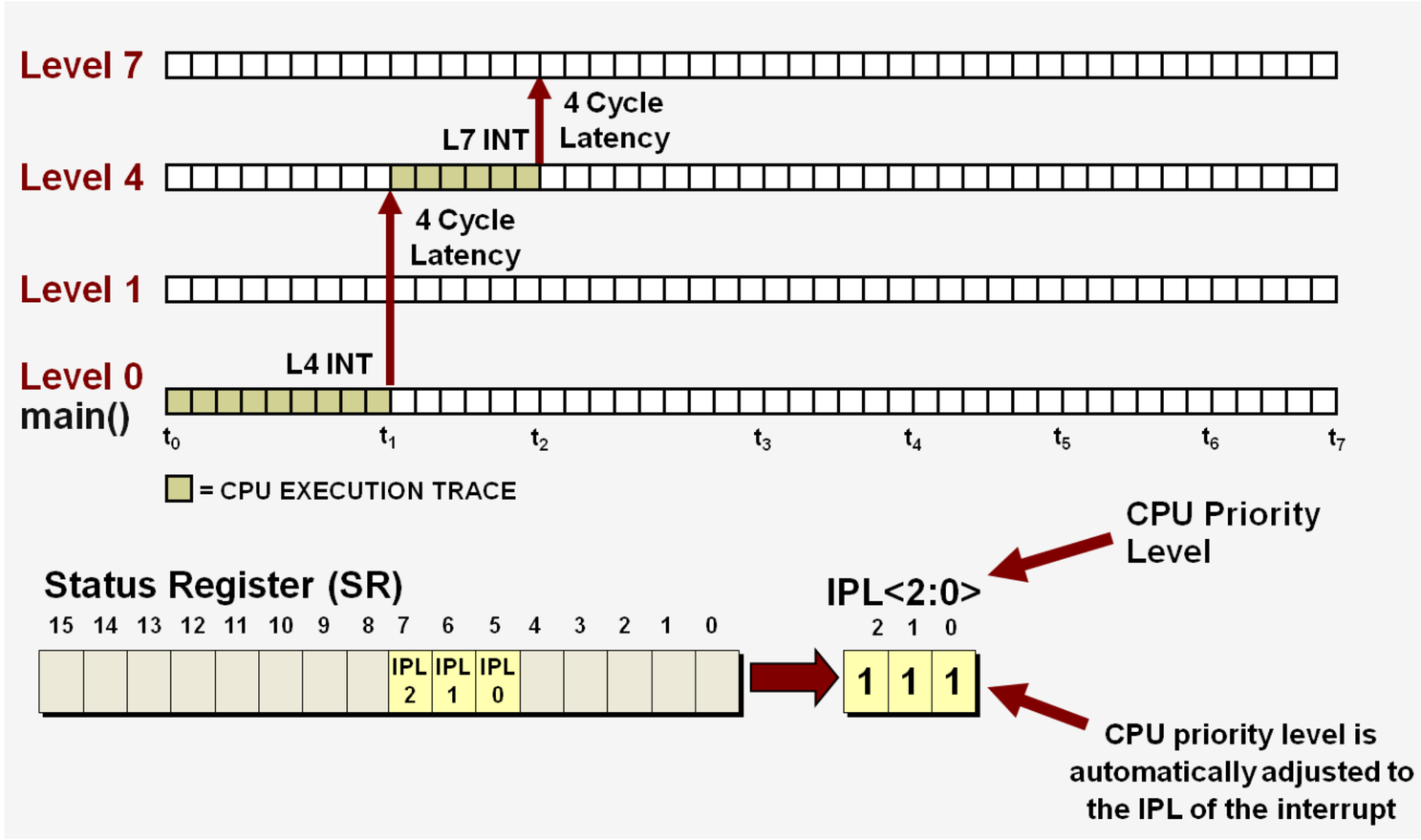
Interrumpir anidamiento



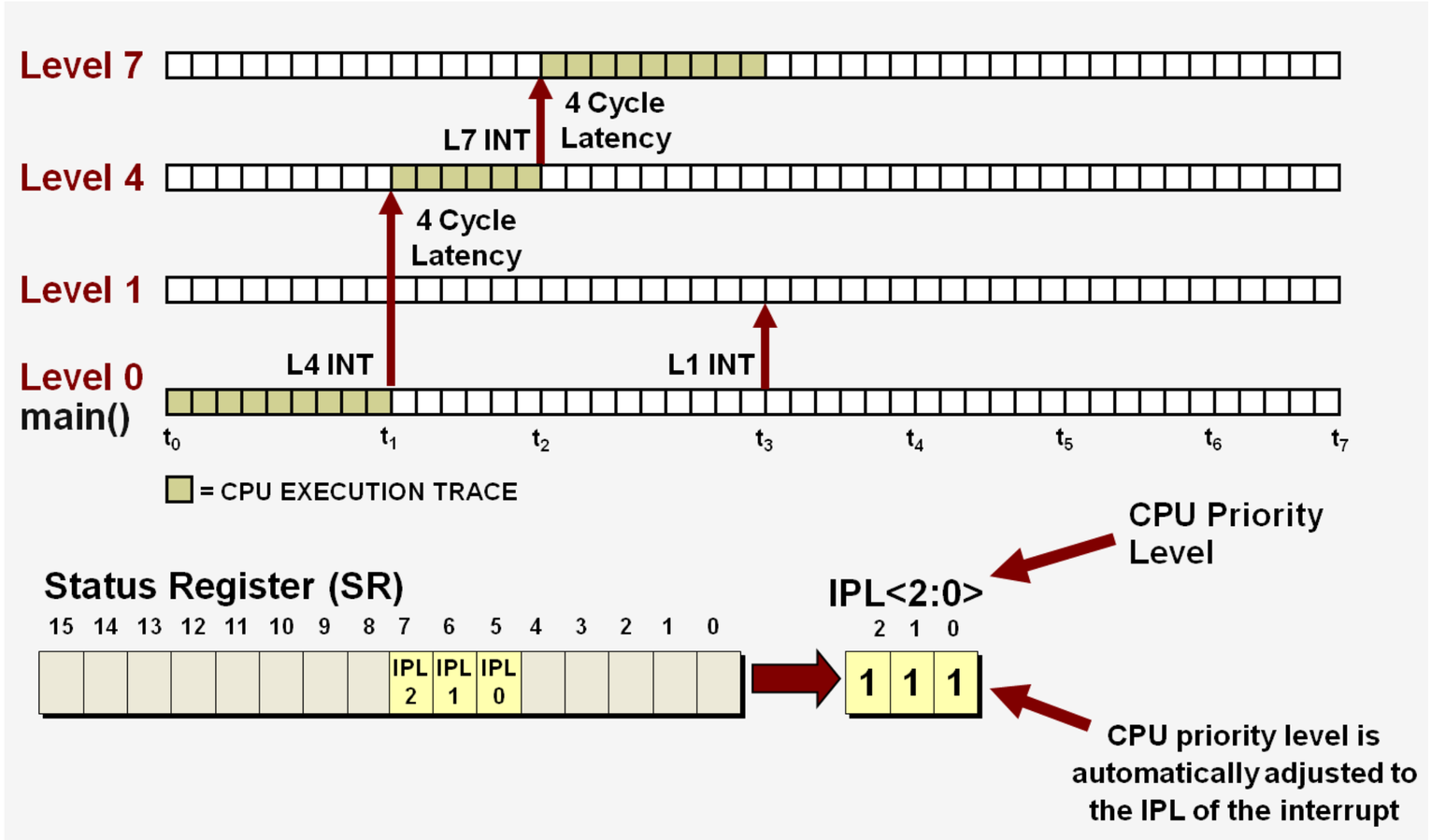
Interrumpir anidamiento



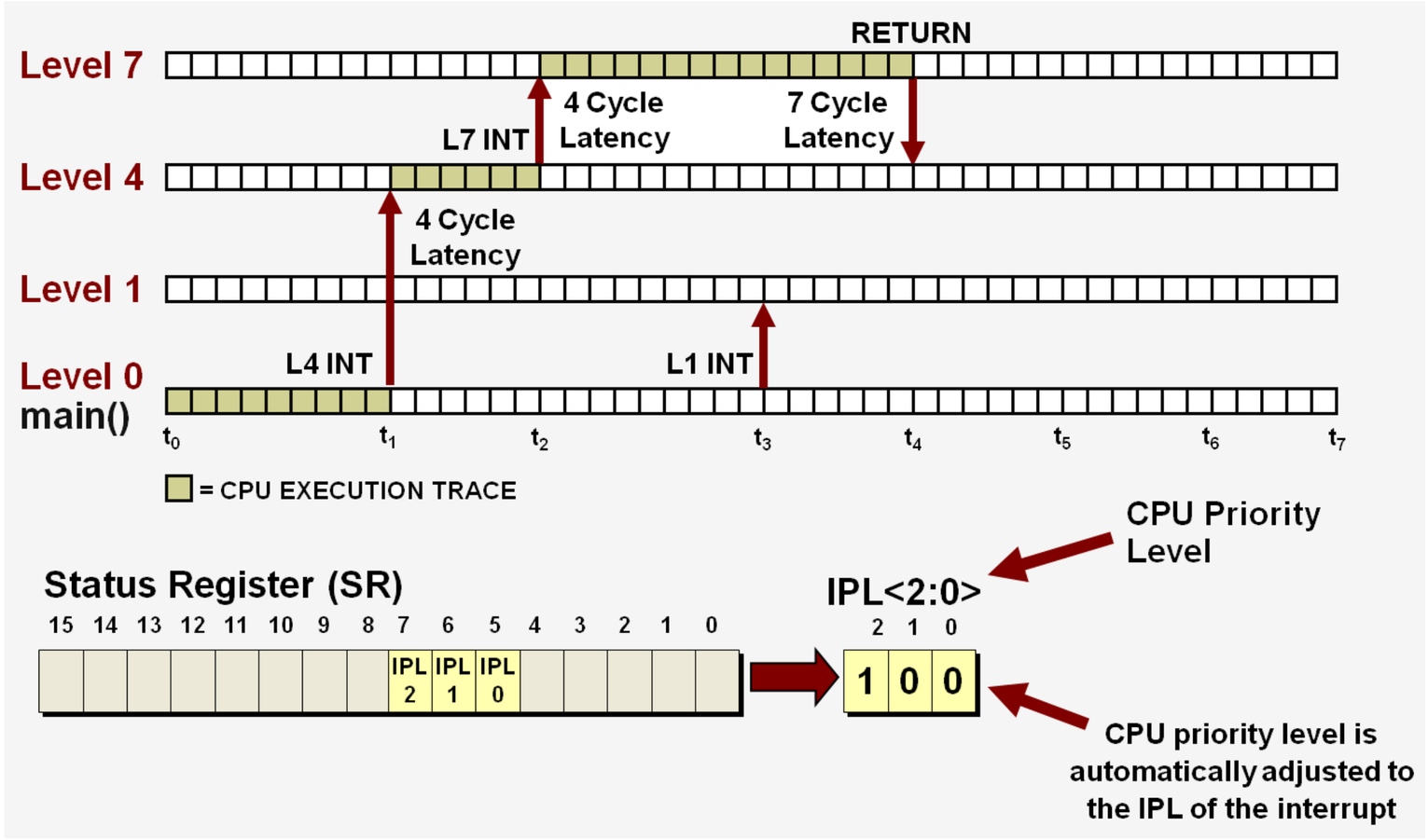
Esquema de excepción vectorial



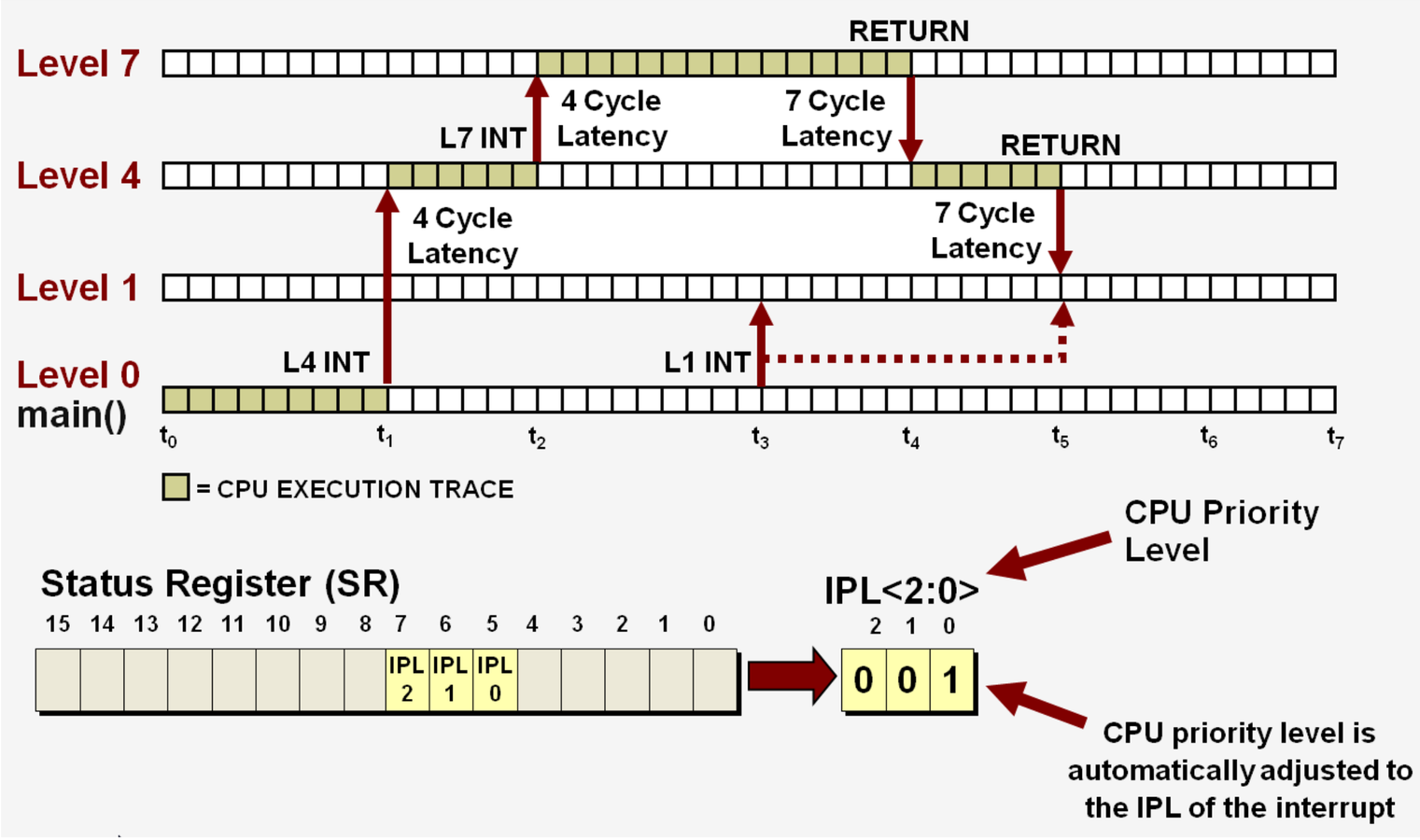
Esquema de excepción vectorial



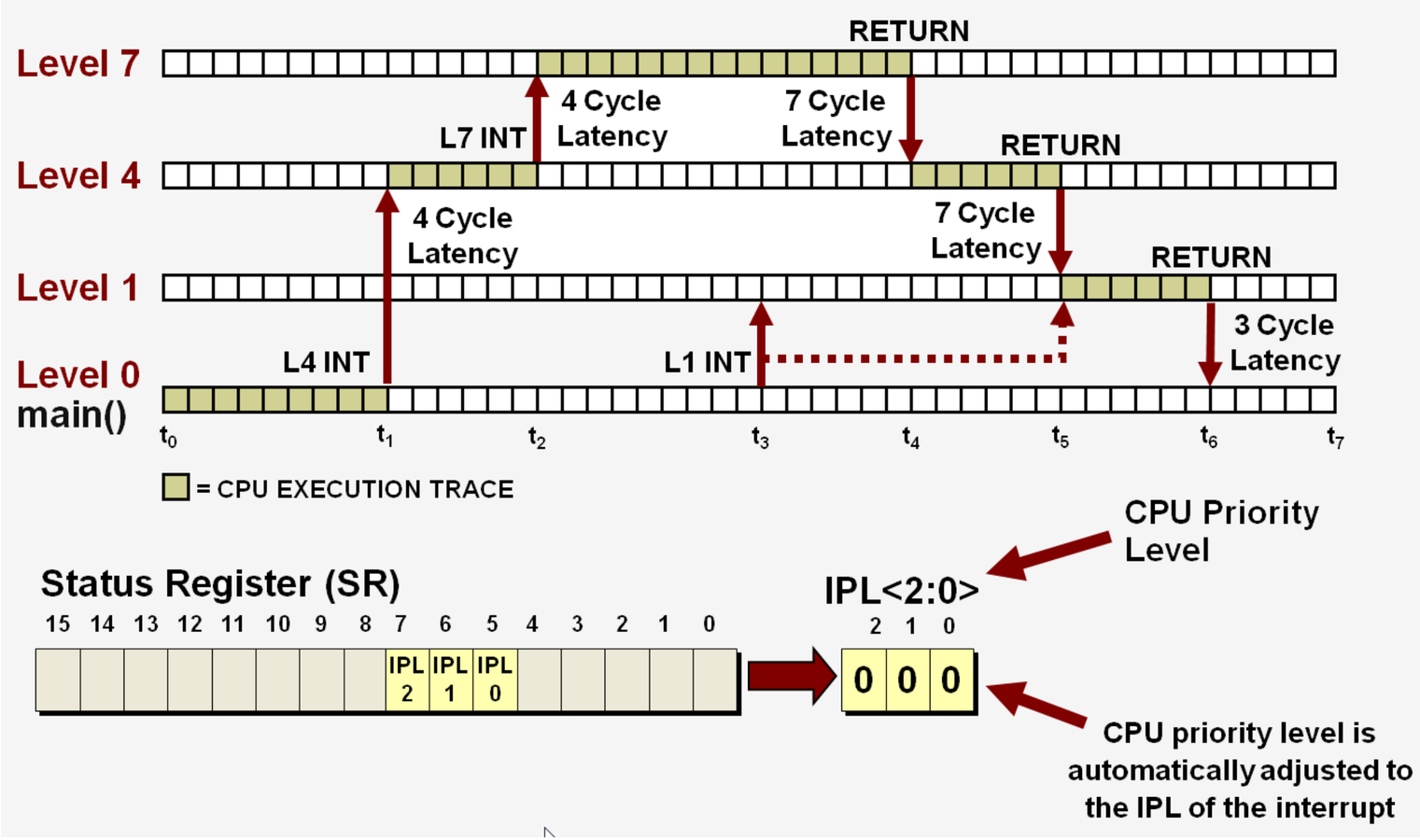
Esquema de excepción vectorial



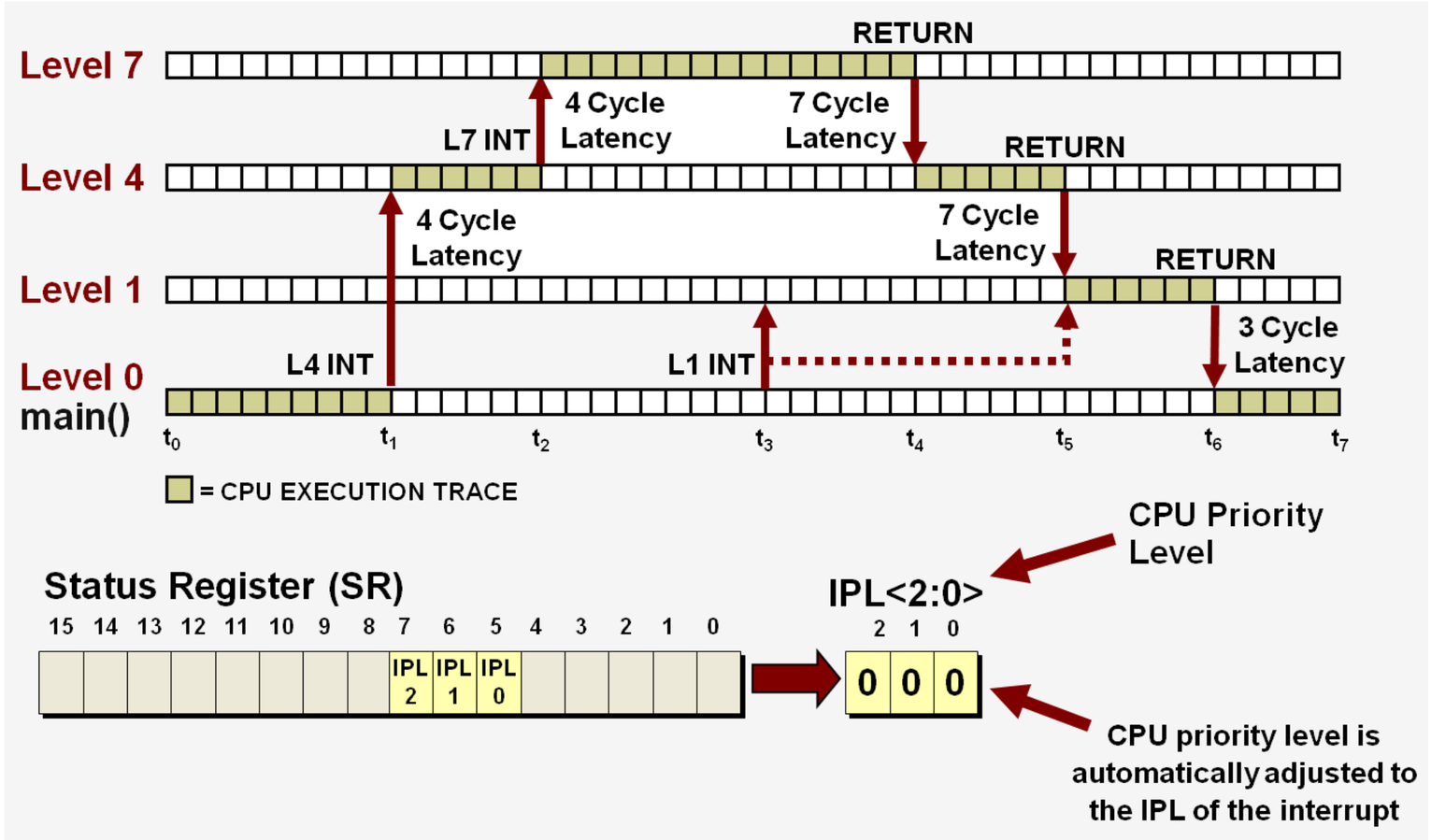
Esquema de excepción vectorial



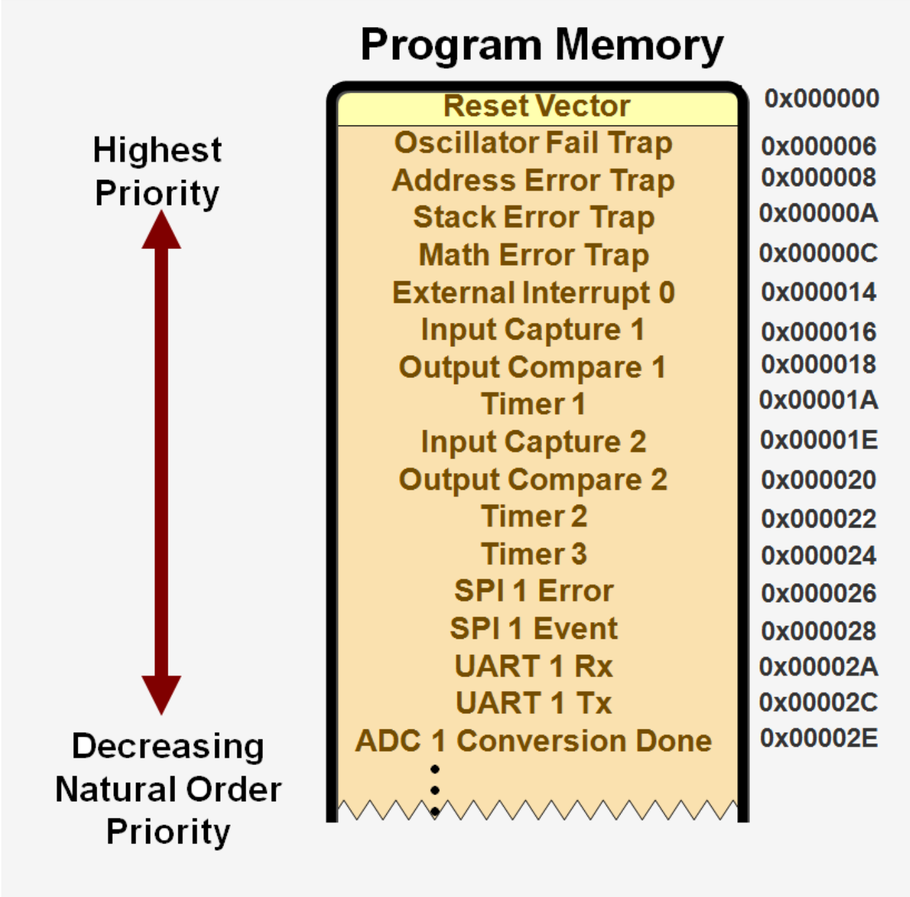
Esquema de excepción vectorial



Esquema de excepción vectorial



Resolución de conflictos de interrupción



Instructor: Godo Sánchez Heredia

Configuración y uso de interrupciones

Hay tres conjuntos de bits de control que deben tenerse en cuenta al trabajar con interrupciones:

IFSX

Indica que se ha producido un evento de interrupción

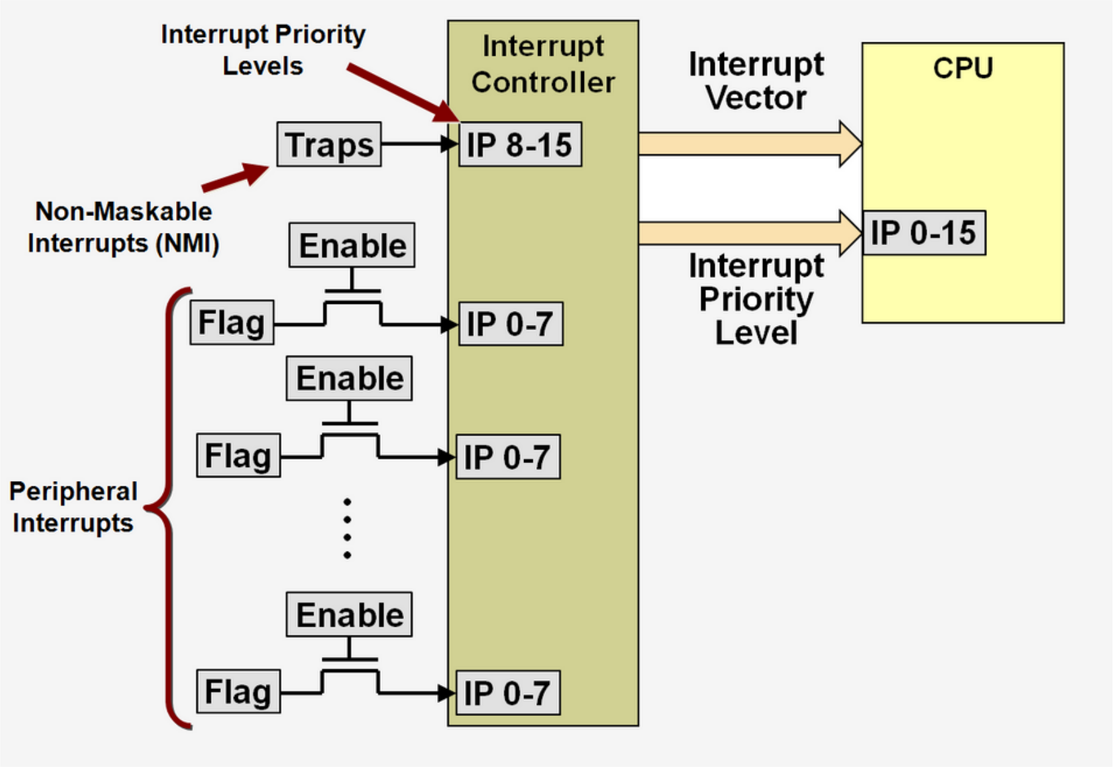
Establecido por el hardware y aprobado por el programador

IECX

Habilitar o deshabilitar fuentes de interrupción individuales

IPCX

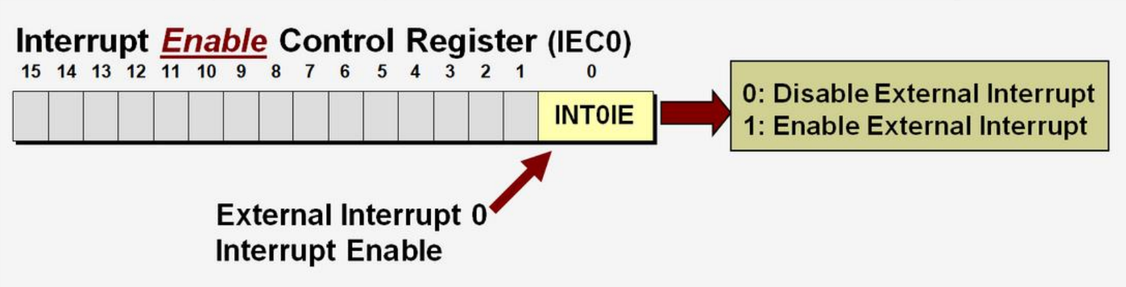
Establecer la prioridad de las fuentes de interrupción individuales



Ejemplo de Configuracion INT0

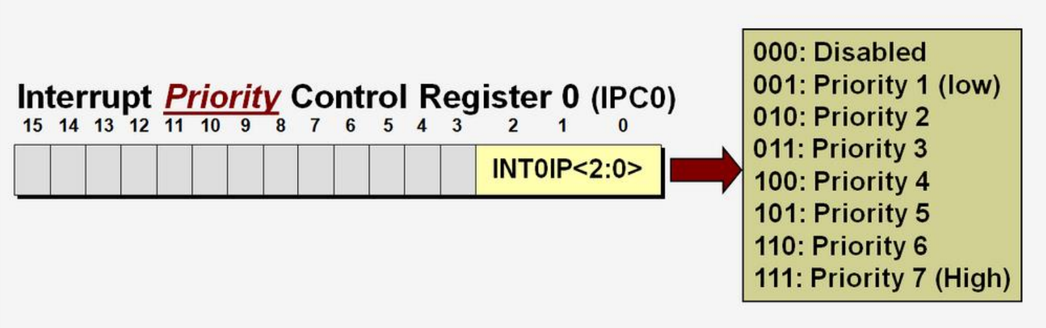
Registro IECx

Para habilitar cualquier interrupción en particular, se deben utilizar registros IECx



Registro IPCx

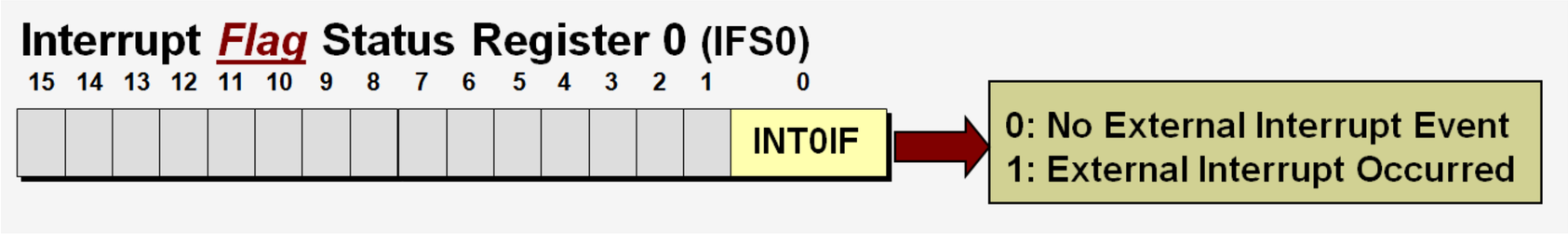
Los registros IPCx se utilizan para establecer la prioridad de cualquier interrupción



Ejemplo de Configuracion INT0

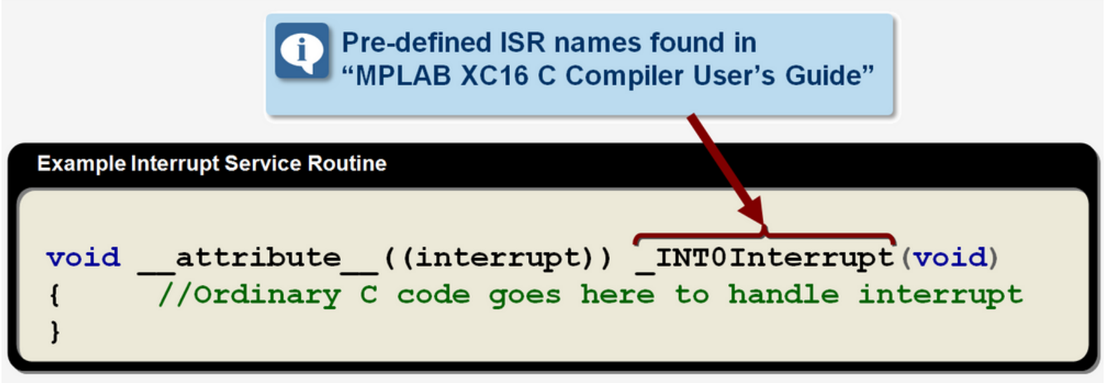
Registro IFSx

Al ocurrir un evento específico, el bit de bandera de interrupción de la interrupción se establecerá en el registro IFSx.



Declaración de una rutina de servicio de interrupción

El compilador MPLAB ® XC16 define un atributo de función especial para declarar una función de rutina de servicio de interrupción:



Extras

Las interrupciones pueden ocurrir en cualquier momento y podrían dañar las variables utilizadas en su programa principal.

Los siguientes registros se guardan y restauran automáticamente desde la pila mediante el hardware de procesamiento de excepciones:

Contador de programas (PCL y PCH)
Byte bajo del registro de estado de la CPU (SRL)
Además, el compilador XC16 generará instrucciones adicionales de ahorro de contexto para guardar y restaurar los siguientes registros de la pila:

Registro de recuento repetido (RCOUNT)
Registros de trabajo utilizados en el ISR (W0-W13)
Registro de la página de visibilidad del espacio del programa (PSVPAG)
Puntero de fotograma anterior (W14)

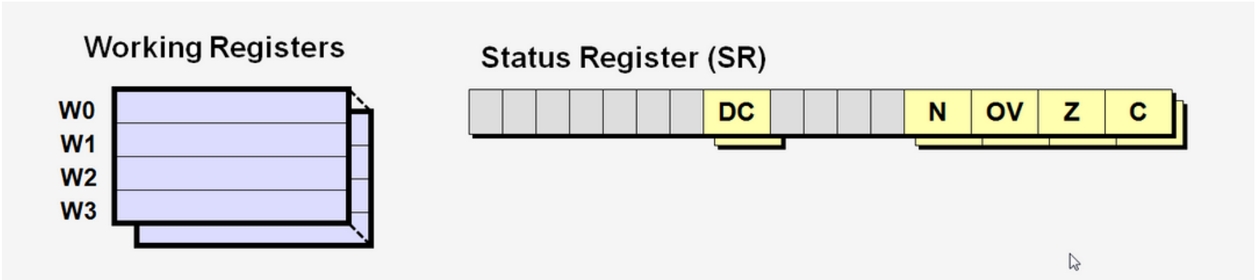
Example

```
void __attribute__((interrupt(save(x, y)))) _INT0Interrupt(void)
```

Extras

Para interrupciones rápidas, use Shadow Registers

Los registros W0-W3 y ciertos registros de estado de la CPU están sombreados, lo que significa que hay un nivel de registros de respaldo dedicados para ellos. El contexto se guarda con una sola instrucción PUSH.S mientras que POP.S restaurará todos estos registros en una sola operación .



Las instrucciones de registro de sombra se pueden generar opcionalmente para su interrupción de mayor prioridad a través del atributo de sombra como se muestr

Example

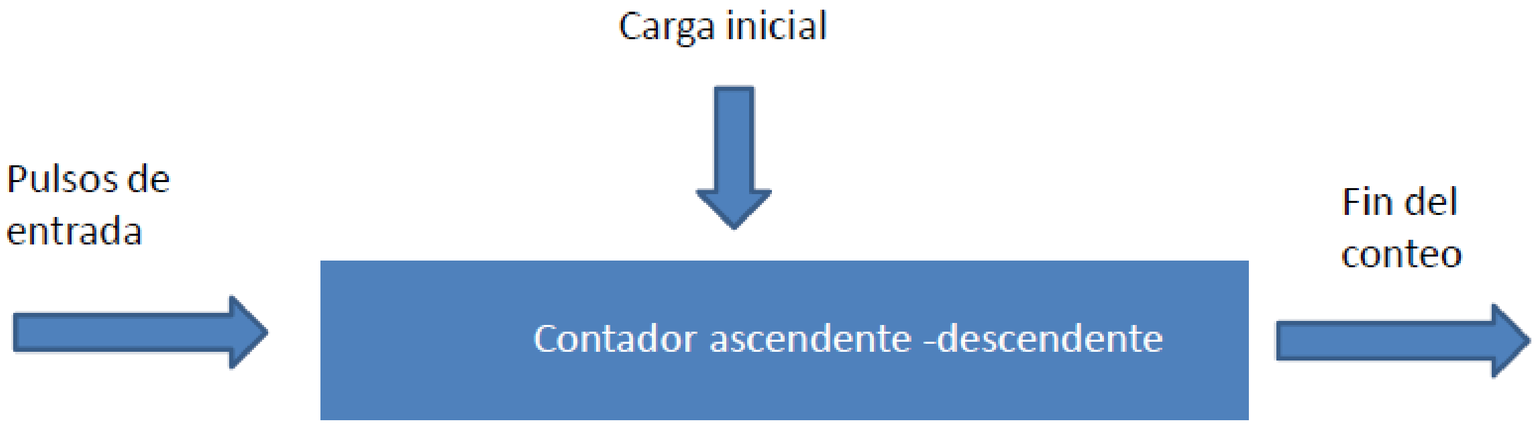
```
void __attribute__((interrupt, shadow)) _INT0Interrupt(void)
```

TIMERS

Timers

¿Qué es un timer?

Un timer es un circuito intern en el microcontrolador que funciona como contador el cual determina un tiempo presico entre el momento que se carga y el instante en el q se produce el desbordamiento



Tipos de Timer

Los dsPIC poseen modulos temporizadores de 16 bits.

TIMER A

Por lo general es el Timer 1 del dsPIC

TIMER B

Aca se encuentran los Timer2, Timer4, Timer6 y Timer8, que se conoce como TimerX

TIMER C

Aca se encuentran los Timer3, Timer5, Timer7 y Timer9), que se conoce como TimerY

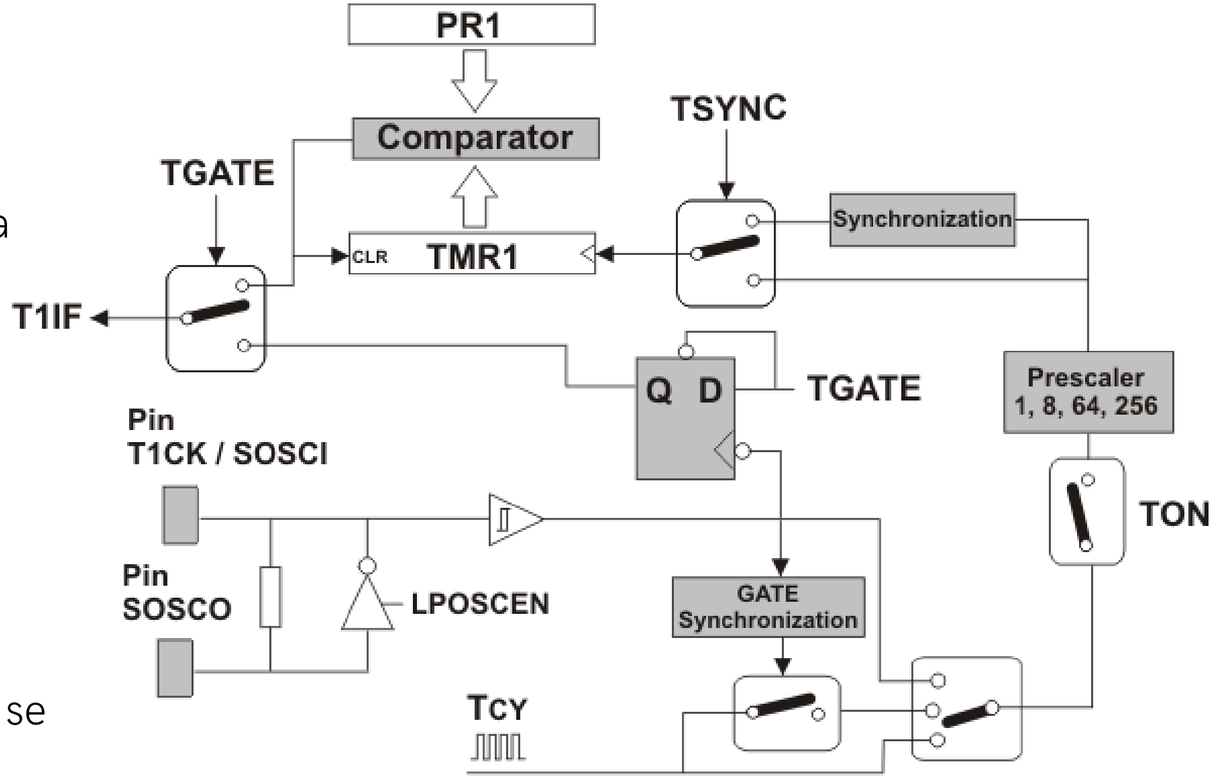
Timer tipo A (Timer 1)

Un temporizador de tipo A tiene las siguientes características únicas sobre otros tipos de temporizadores:

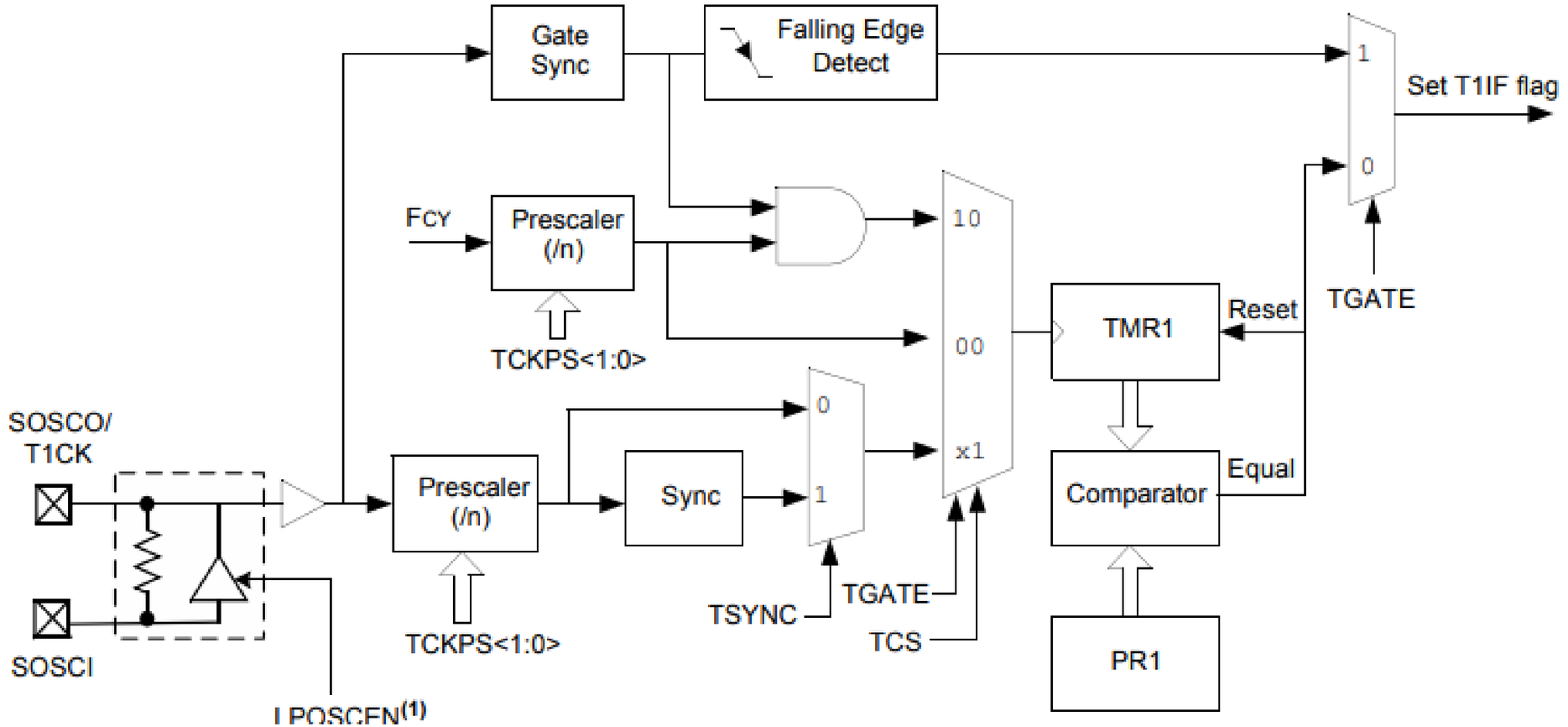
- Se puede operar desde el oscilador de cristal de 32 kHz de baja potencia disponible en el dispositivo
- Se puede operar en modo de contador asíncrono desde una fuente de reloj externa
- Opcionalmente, la entrada de reloj externo (T1CK) se puede sincronizar con el reloj del dispositivo interno

y la sincronización del reloj se realiza después de que el preescalador divida T1CK.

Las características únicas del temporizador Tipo A permiten que se utilice para aplicaciones de reloj en tiempo real (RTC).



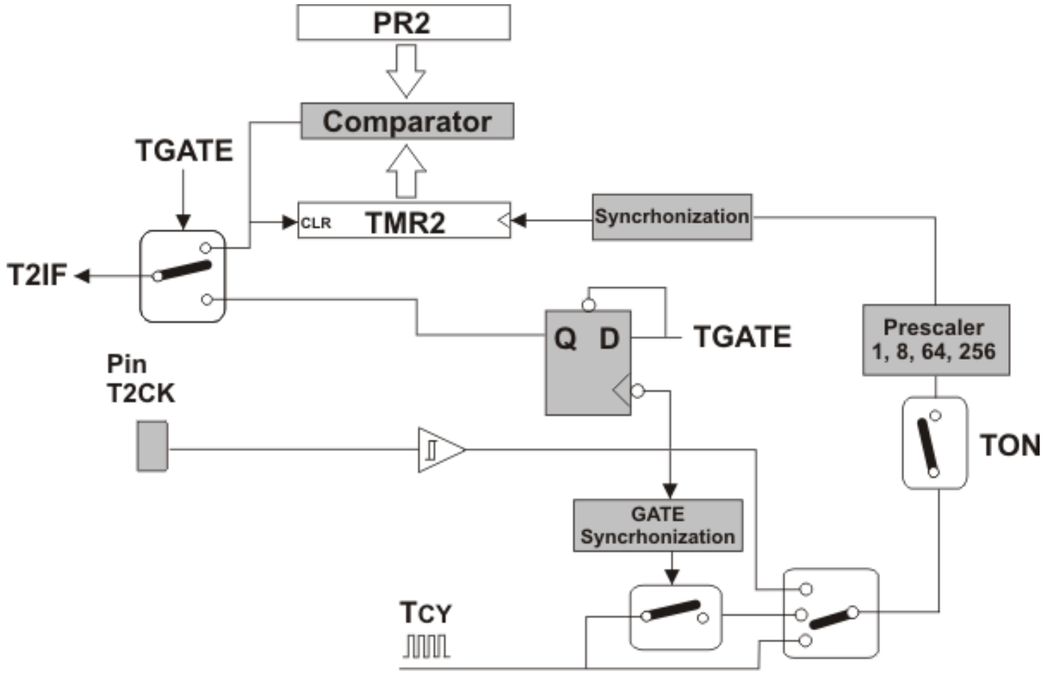
Timer 1


Gate Sync
Falling Edge Detect
1
0
Set T1IF flag
FCY
Prescaler (/n)
TCKPS<1:0>
10
00
TMR1
Reset
TGATE
SOSCO/ T1CK
SOSCI
Prescaler (/n)
TCKPS<1:0>
Sync
0
1
x1
TSYNC
TGATE
TCS
Comparator
Equal
PR1
LPOSCEN(1)

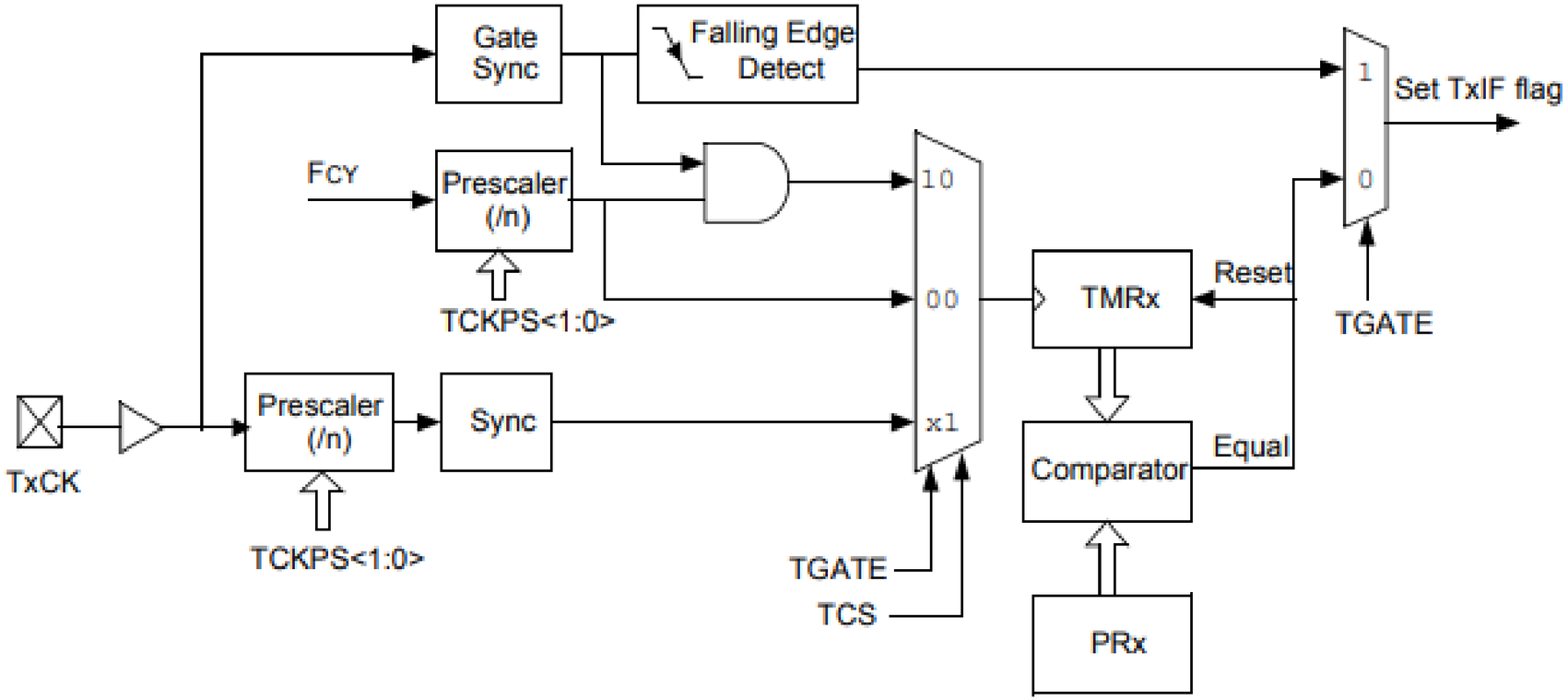
Timer tipo B(Timer2, Timer4, Timer6 y Timer8)

Un temporizador de tipo B consta de las siguientes características específicas:

- Se puede concatenar con un temporizador de tipo C para formar un temporizador de 32 bits
- La entrada de reloj externo (TxCK) siempre está sincronizada con el reloj interno del dispositivo y la sincronización del reloj se realiza después de que el preescalador divida TxCK.



Timer2, Timer4, Timer6 y Timer8

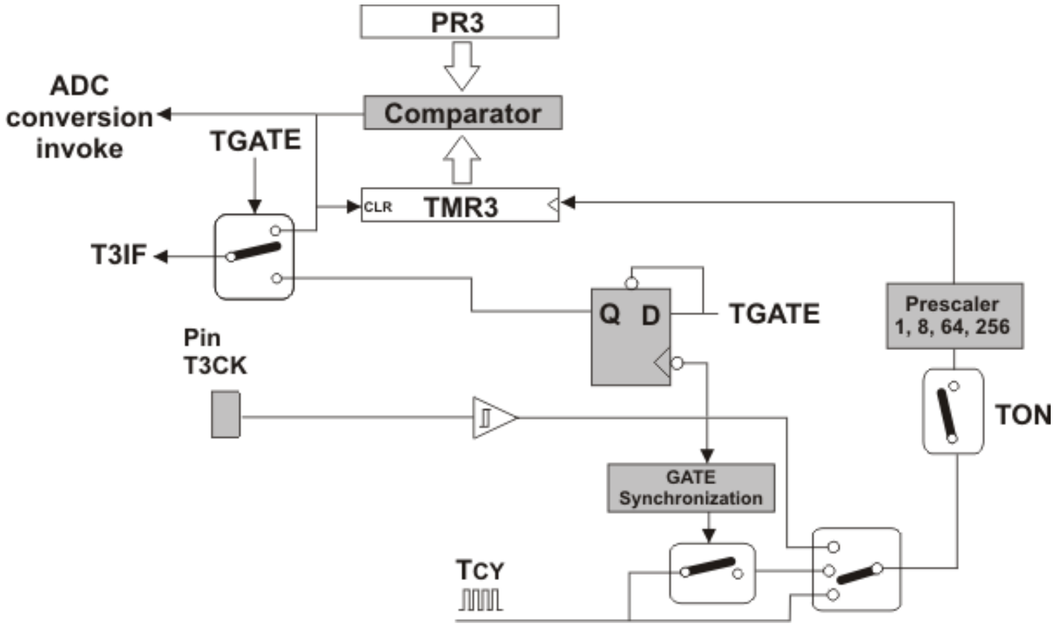


Timer tipo C(Timer3, Timer5, Timer7 y Timer9)

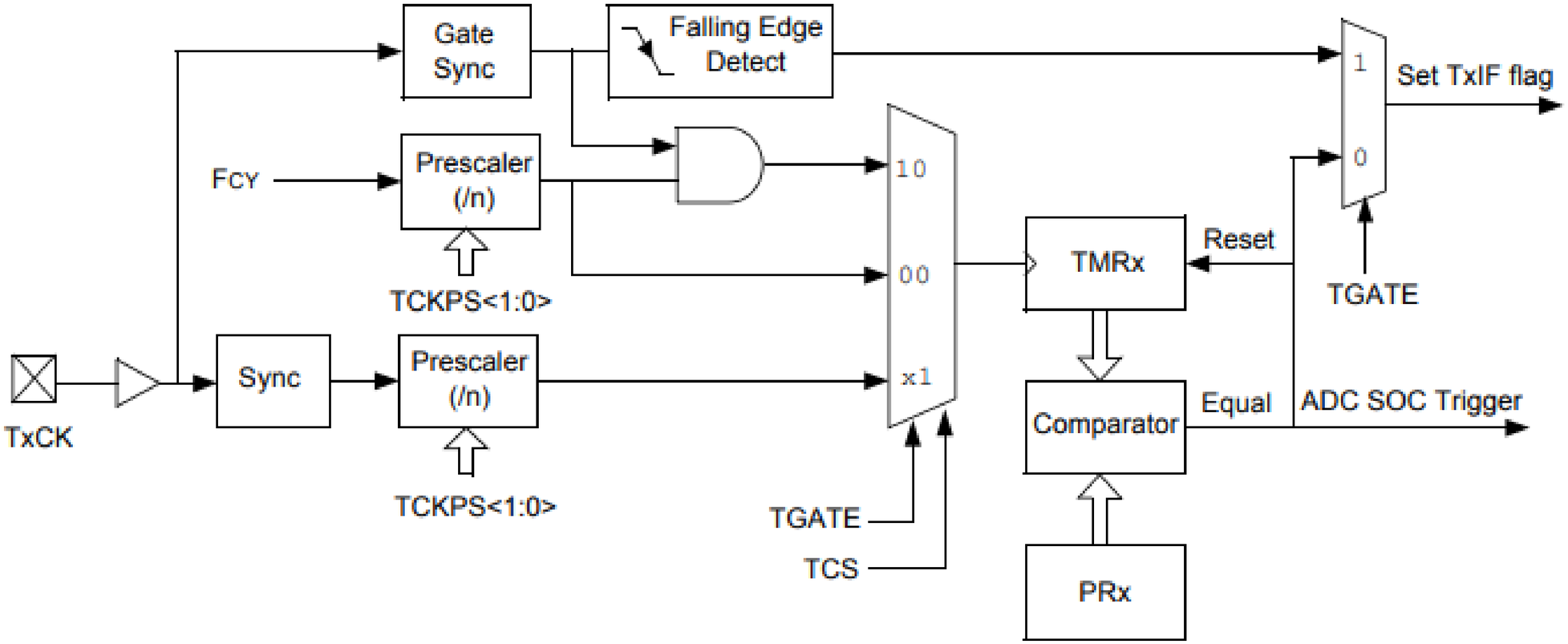
Un temporizador de tipo C tiene siguientes características específicas:

- Se puede concatenar con un temporizador de tipo B para formar un temporizador de 32 bits
- Al menos un temporizador de tipo C tiene la capacidad de activar una conversión de analógico a digital
- La entrada de reloj externo (TyCK) siempre está sincronizada con el reloj interno del dispositivo. los

la sincronización del reloj se realiza mediante TyCK



Timer3, Timer5, Timer7 y Timer9



Registros Asociados

- T1CON: Registro de control de temporizador tipo A

Este registro controla la configuración de un temporizador de tipo A.

- TXCON: Registro de control del temporizador de tipo B (X = 2, 4, 6, 8)

Este registro controla la configuración de un temporizador de tipo B.

- TYCON: Registro de control de temporizador tipo C (Y = 3, 5, 7, 9)

Este registro controla la configuración de un temporizador de tipo C.

Además de los registros anteriores, cada temporizador tiene los siguientes registros asociados de 16 bits:

- PRn: Registro de período del temporizador (n = 1 a 9)

Este es el registro de período del temporizador de 16 bits.

- TMRn: Registro de conteo del temporizador (n = 1 a 9)

Este es el registro de conteo del temporizador de 16 bits

Modos de Operacion

El módulo de temporizador puede funcionar en uno de los siguientes modos:

- Modo de temporizador
- Modo de temporizador cerrado
- Modo contador síncrono
- Modo contador asíncrono (solo temporizador tipo A)

En los modos Timer y Gated Timer, el reloj de entrada se deriva del ciclo de instrucción interno reloj (FCY). En los modos de contador síncrono y asíncrono, el reloj de entrada se deriva de la entrada de reloj externo en el pin TxCK.

Mode	TCS	TGATE	TSYNC
Timer	0	0	x
Gated timer	0	1	x
Synchronous Counter	1	x	1
Asynchronous Counter	1	x	0

Calculo del tiempo

$$PRX = \frac{FCY * Tiempo}{Preescaler} - 1$$

$$Tiempo = \frac{Preescaler * PRX}{FCY} + 1$$

¡MUCHAS GRACIAS!

Telf: 943874659
Correo:
godo.electronica@gmail.com

https://github.com/GodoSanchezH