Interrupciones y microcontroladores (y sus periféricos)

Clase 1

Sistemas embebidos para tiempo real

Agenda próximas tres clases

- Primera clase (hoy)
  - Fundamentos de las Interrupciones
  - Periféricos del MSP430G2553
  - Periférico básico: Reloj (Basic Clock Module+)
- Segunda clase
  - Periféricos avanzado: Conversor A/D (ADC10)
  - Problema de datos compartidos (y solución básica)
  - Latencia en las interrupciones
- Tercer clase
  - Soluciones alternativas al problema de los datos compartidos

Objetivos

- Describir el flujo de ejecución de un microcontrolador: contexto principal (main, también llamado “tarea”) y de interrupciones.
- Describir la secuencia de ejecución de interrupciones.
- Utilizar e interpretar manuales de usuario y hojas de datos.
- Familiarizarse con la configuración de periféricos **básicos**.

Actividad colectiva

- Objetivos:
  - Evaluación de conocimientos previos sobre interrupciones, puesta a punto y repaso.
- Responder las siguientes preguntas por escrito
  1. ¿Qué es una ISR? ¿Qué es una IRQ?
  2. ¿Cómo se encuentra la ISR para una IRQ?
  3. ¿Se pude interrumpir en el medio de una instrucción?
  4. Si dos interrupciones ocurren al mismo tiempo: ¿qué pasa?
  5. ¿Se pueden anidar interrupciones? (nested)
  6. ¿Qué pasa con las interrupciones cuando están deshabilidatadas?
  7. ¿Qué pasa si nos olvidamos de rehabilitarlas?
  8. ¿Se pueden escribir ISR en C?
  9. En los microcontroladores: ¿Cómo interrumpen los periféricos integrados?

Actividad en grupo

- Interrupciones en MSP430
  - Actividad:
    - responder las preguntas anteriores para micros de la familia MSP430xF2xxx
  - Grupos:
    - 3 a 4 participantes
  - Tiempo:
    - 5-10 minutos
  - Material: Manual de la familia MSP430xF2xxx, páginas 33-40
  - Puesta en común.

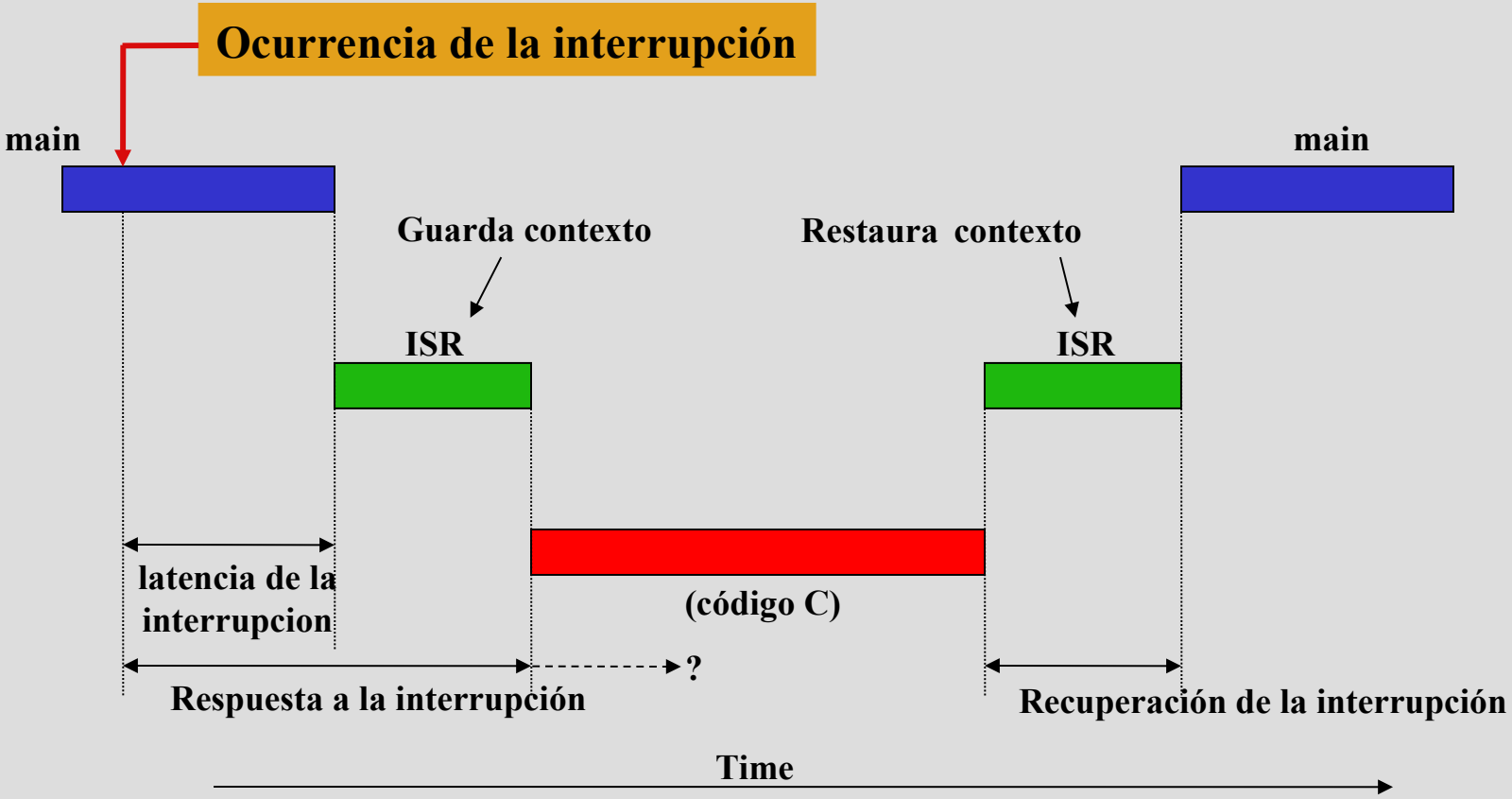
Fundamentos de las Interrupciones

- Escenario en sistemas de tiempo real
  - Procesador ejecutando un programa
  - Ocurre algo que necesita ser atendido
  - Procesador pone en “espera” el hilo principal y lo atiende
- Secuencia de eventos típica
  - Dispositivo que necesita atención se lo indica al μP a través de una señal de hardware.
  - μP guarda información de la tarea en ejecución (PC) y carga en PC la dirección de inicio de la ISR.
  - Se ejecuta la ISR y al final se retorna con RETI
  - Se carga en valor guardado del PC

Fundamentos de las Interrupciones

- Atención:
  - ISR comienza a ejecutarse en el “medio” del programa
  - CPU está en uso (registros con valores, etc.)
  - Al retornar de la ISR debe seguir como si nada hubiera pasado
- Entonces la ISR tiene que:
  - guardar el contexto (estado del procesador: SR, etc)
  - hace lo que tiene que hacer (dar respuesta a la interrupción)
  - restaura el contexto
  - retorna de la subrutina

Interrupciones: diagrama de tiempos



Basado en: J. Archibald, "Real-Time and Embedded Systems", slides: set2.ppt

Alternativas a las interrupciones

- Consulta (Polling):
  - Se toma la iniciativa de chequear (proactivo) en intervalos regulares (idealmente).
  - En aplicaciones simples puede manejar los eventos sin retardos significativos.
  - Usualmente funciona bien para aplicaciones con una sola tarea.
  - El Tiempo de Respuesta (peor caso) está más fácil de determinar.
- Desventajas de polling
  - Cierta dificultad en aplicaciones con múltiples tareas.
  - Usa la CPU ineficientemente
    - Tiempo insumido en el chequeo podría ser usado en procesado
    - No es un problema en cuando el procesado es liviano ya que la CPU no tiene nada que hacer (salvo consumir)

Ventajas de las interrupciones

- Retardos uniformes en respuesta a eventos
- Independiente de la complejidad de la tarea que se está ejecutando
- Las “tareas” pueden dormir hasta que ocurra un evento
  - Esencial en sistemas multitarea basados en prioridades
- Estas ventajas tienen un costo
  - La complejidad del software de sistemas basados en interrupciones es más alta que los sistemas basados en polling

Fuentes de interrupción

- Depende de lo que se conecte a los pines de interrupción
- En microcontroladores: periféricos integrados + pin/es input
- Ejemplos:
  - Llegada de datos de un dispositivo E/S.
  - Terminación de una petición previa a un dispositivo E/S.
  - Cambio en la lectura de un sensor.
  - Acción de un usuario (presionado de un botón).
  - Detección de una falla
    - externa al CPU
    - interna al CPU (exception)
  - Expiración de un temporizador
  - Falla en la alimentación

Cosas que importa saber...

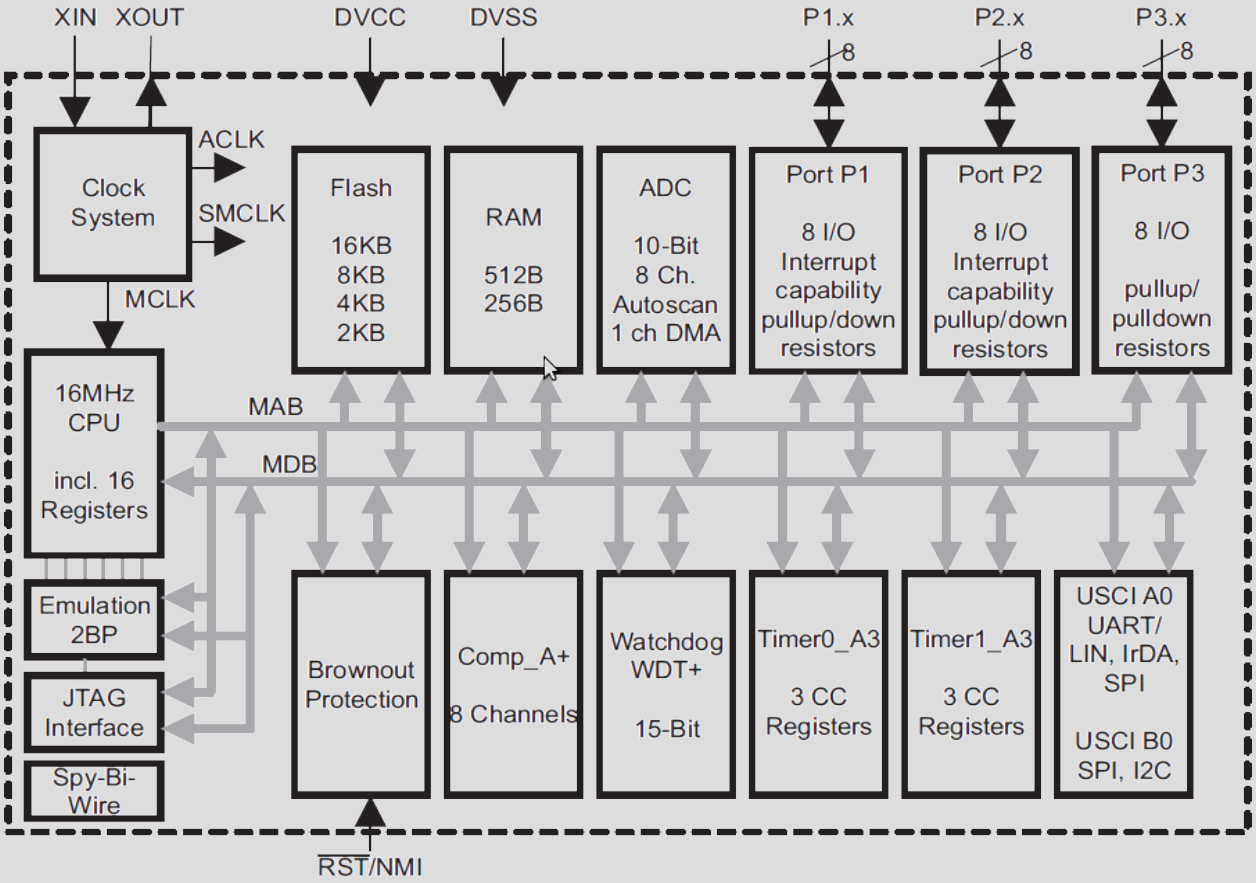
...sobre el microcontrolador para empezar un proyecto de sistemas embebidos:

- Tipo procesador, tamaño palabra, frecuencia máxima
- Arquitectura y mapa de memoria
- Registros (cantidad, uso, etc.)
- Características del PC, SR y SP
- Set de instrucciones
- Modos de direccionamiento.
- Modos de bajo consumo (LPM)
- Interrupciones: ISR, vector de interrupciones, prioridades
- Periféricos disponibles
- Reloj

Periféricos del uC

- Watchdog timer (WDT)
- Puertos digitales de E/S (I/O pin)
- Temporizadores (Timers)
- Conversores A/D (ADC) y D/A (DAC)
- Interfaz/protocolos de comunicación
  - E/S de datos digitales (UART, SPI, I2C)
- DMA
- Especializados (no siempre disponibles)
  - MPU / MMU (Memory Protection Unit / Memory Management Unit)
  - SVS

MSP430G2x53



Fuente: "MSP430G2x53 MSP430G2x13 MIXED SIGNAL MICROC." (file: SLAS735J.pdf) Functional Block Diagram, MSP430G2x53, page 5.

Actividad en grupo

- Estudiar funcionamiento del Reloj (Basic Clock Module+)
  - Responder:
    - ¿Cuántos relojes hay? ¿por qué hay más de uno?
    - ¿Cómo se generan?
    - ¿Qué características tienen? Rangos de frecuencia, necesidad de componentes externos, etc.
    - ¿Están prendidos cuando el uC operan en bajo consumo (LPM)?
  - Grupos:
    - 2 a 4 participantes
  - Tiempo:
    - 10 minutos
  - Material:
    - Manual familia MSP430x2xx y hoja de datos MSP430G2553

Reloj

- 3 relojes
  - ACLK
    - Aux clock
  - MCLK
    - Master clock
  - SMCLK
    - Sub-main clock
- 4 fuentes
  - VLOCLK
  - LFXT1CLK
  - XT2CLK
  - DCOCLK

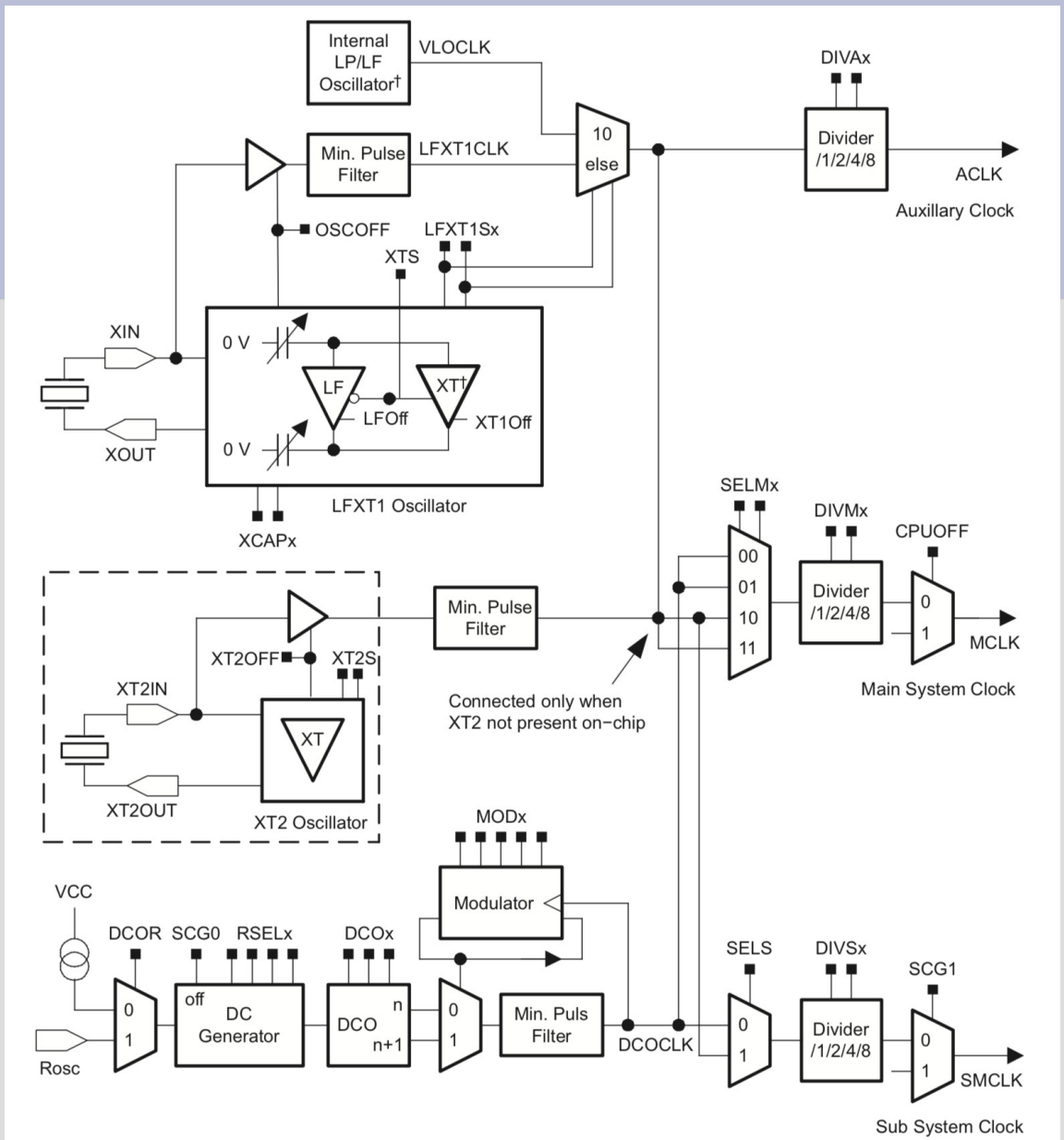


Figure 5-1. Basic Clock Module+ Block Diagram – MSP430F2xx

Reloj (Basic Clock Module+)

- Sin componentes externos:
  - VLOCLK: baja frecuencia (12 kHz)
  - DCOCLK: frecuencia controlada digitalmente
- Requiere componentes externos (por ejemplo un cristal):
  - LFXT1CLK: relojes de 32768 Hz (baja frecuencia) y alta frecuencia
  - XT2CLK: alta frecuencia
- La variedad de relojes permite balancear el trade-off entre consumo y velocidad

Nota: alta frecuencia = 400kHz a 16MHz

Reloj (Basic Clock Module+)

Table 2-2. Operating Modes For Basic Clock System

SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled, SMCLK, ACLK are active
0	1	0	1	LPM1	CPU, MCLK are disabled. DCO and DC generator are disabled if the DCO is not used for SMCLK. ACLK is active.
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO are disabled. DC generator remains enabled. ACLK is active.
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO are disabled. DC generator disabled. ACLK is active.
1	1	1	1	LPM4	CPU and all clocks disabled

Configuración y uso de registros en C (y CCS)

- **Seteo x BIT**
  - **P1SEL |= BIT4;** // BIT4=0x0010
  - P1SEL = P1SEL OR BIT4
  - P1SEL = P1SEL OR 00010000
- **Clear x Bit**
  - **P3DIR &= ~BIT0;** // BIT0=0x0001
  - P3DIR = P3DIR AND (~BIT0)
  - P3DIR = P3DIR AND 11111110
- **Por registro:**
  - **TACTL = TASSEL_1 + MC_1 + TACLR + ID_3;**
- **Uso: test BIT**
  - **if (ADC10CTL1 & ADC10BUSY) {}**

Actividad en grupo

- Configuración de registros del Basic Clock Module+
  - Especificaciones:
    - Configurar el VLOCLK (oscilador interno de bajo consumo de 12 kHz) como ACLK.
    - Dejar incambiados los registros que no se necesitan.
  - Grupos:
    - 2 a 4 participantes
  - Tiempo:
    - 10 minutos
  - Material:
    - Manual familia MSP430x2xx y hoja de datos MSP430G2553

Configuración registros Reloj

- BCSCTL1: no hace falta modificarlo (por defecto está ok) pero podría configurarse:
  - Para hacerlo “evidente” en el código
  - Para evitar bugs (que alguien más lo modifique)

- BCSCTL3: solo hace falta modificar:

```
BCSCTL3 |=  LFXT1S_2; // selecciona VLOCLK
// vale comentario sobre BCSCTL1, asume que el uC viene de un reset
   (BCSCTL3 está inicializado con sus valores x defecto)
```

- DCOCTL y BCSCTL2:
  - no hace falta tocarlos (no juegan con VLOCLK)