Resumen Teórico

Final ED3

GPIO

- General Purpose Input Output: la usamos para configurar pines I/O.
- Están mapeados en memoria 0x2009C000 - 0x200A0000 (sale en el gráfico del manual).
- Con “PINSEL” se configura la función del pin. “PINMODE” las resistencias o modo de trabajo interno. “FIODIR” si es entrada o salida y “FIOPIN” lee el estado o escribe sobre el pin. (Como FIOSET y FIOCLR).
- El modo “repetidor” va alternando PULL UP y PULL DOWN según el uso, evita corrientes de fuga.
- Son accesibles por DMA.
- Los registros están el AHB.
- Soportan Bit-Banding (Es una manera de modificar bits mediante los registros, pero de manera que evite interrupciones en el momento que se interpretan las operaciones, usando una palabra completa para modificar un único bit).

Interrupciones por GPIO

- Pueden ser por los pines de los puertos 0 y 2, o por interrupciones externas (EINT).
- Las interrupciones las maneja el NVIC. Es un hardware dedicado a esto especialmente. Está vectorizado porque usa punteros para saltar a rutinas y posiciones auxiliares ante una interrupción. Permite atención múltiple con prioridades.

Interrupción por GPIO (0 & 2)

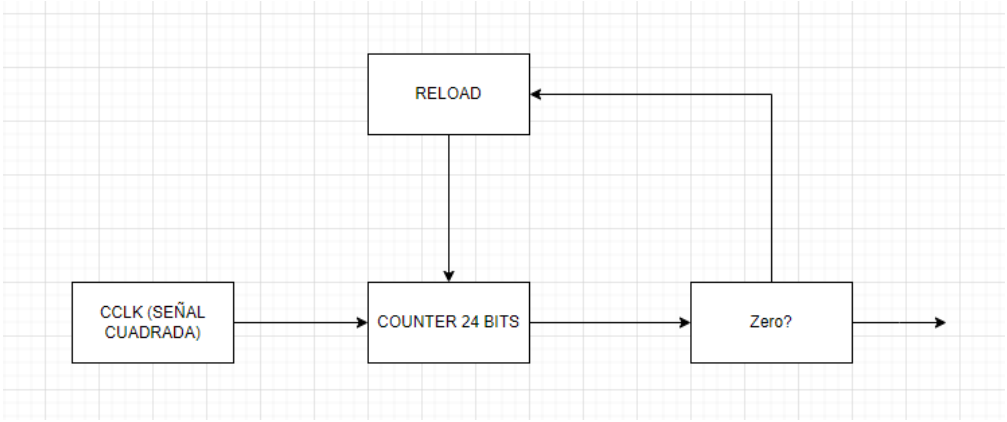
- Cada pin puede interrumpir por cualquier flanco.
- La detección del flanco es asincrónica (tenemos que testearlo en la ISR).
- Puede operar sin clock (Power Down).
- Ambos puertos comparten la posición de EINT3 en el NVIC.

Interrupción externa por pin EINT

- Son 4 pines (0 - 3).
- Es una funcionalidad específica del pin.
- Puede interrumpir por nivel o por flanco.
- Cada una tiene su vector del NVIC.

SysTick Timer

- Es un bloque integrado dentro del core.
- Genera una "interrupción" configurable pero es una excepción porque está integrado al sistema.
- Es un temporizador de 24 bits.



- Cálculos:

$$Reload = Tiempo\ objetivo * \frac{1}{CCLK} - 1$$

- Registros

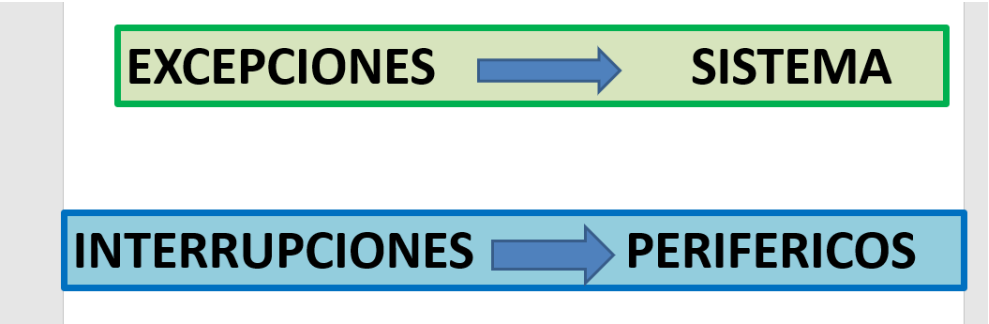
CTRL	Enable's y fuente de reloj
RELOAD	Valor de reload o LOAD
CURRENT	Valor del contador
CALIB	Valor de fábrica que genera interrupciones cada 10 mS.

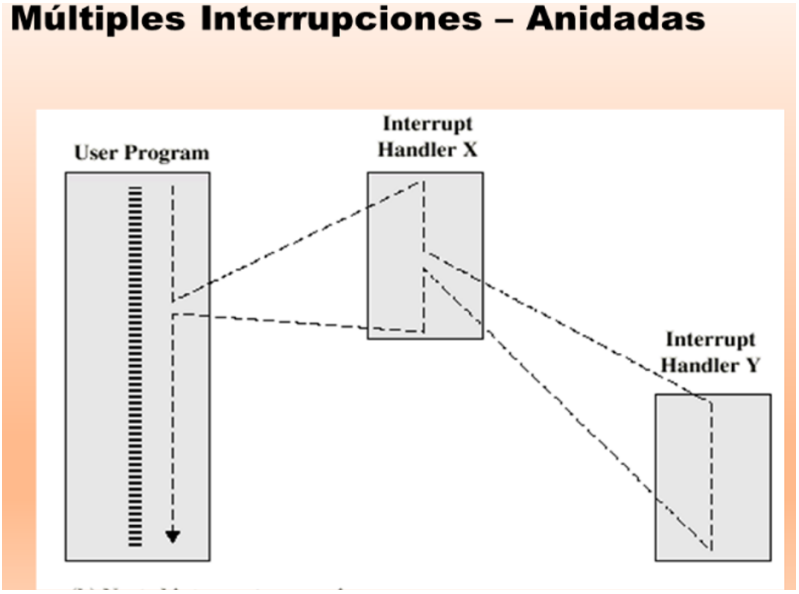
Relojes

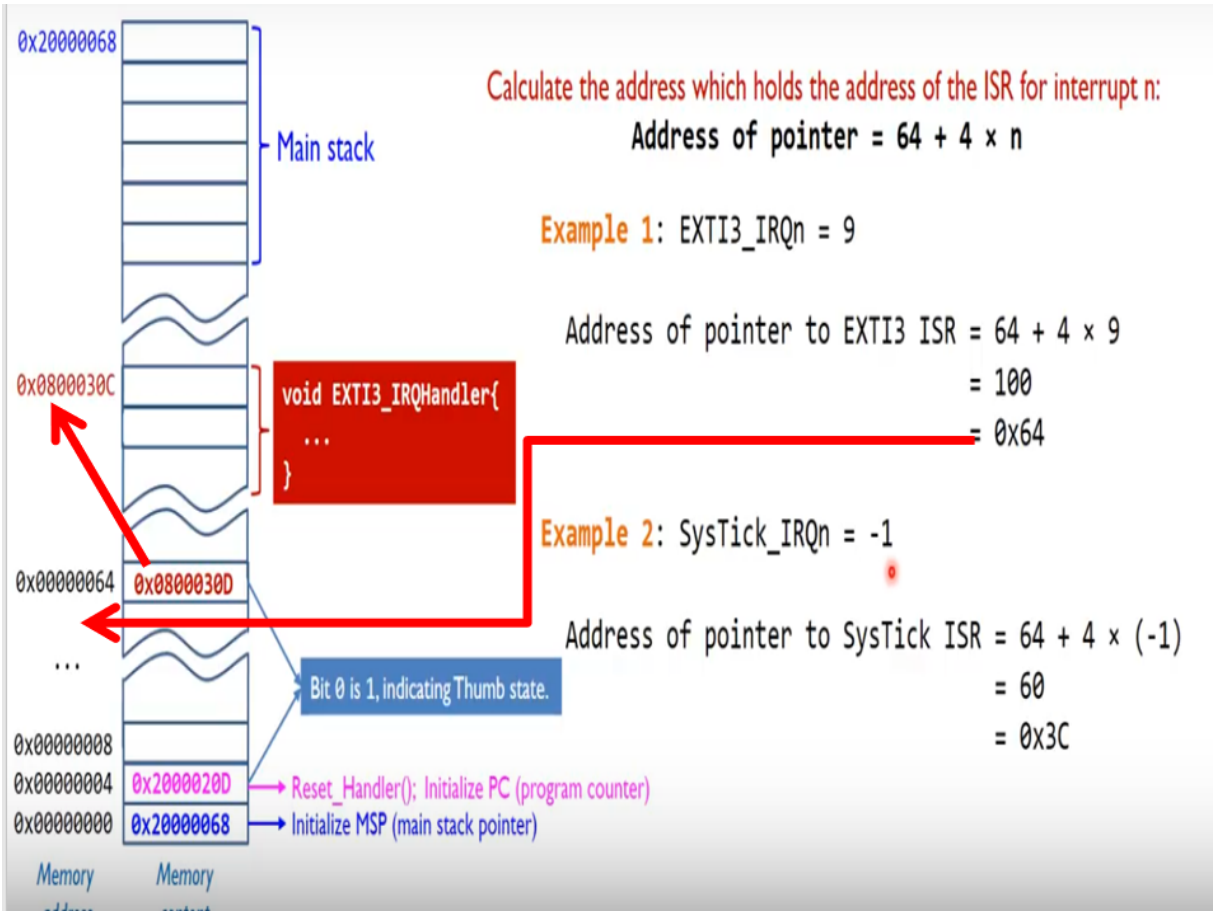
- Oscilador interno RC: Fuente para el WD y/o reloj que excita al PLL. **4MHz**.
- Oscilador Principal: Fuente de CPU con o sin PLL, también excita a USB. **1MHz - 25Mhz.**
- Oscilador RTC: Fuente para RTC (Real Time clock), PLL y CPU. **1MHz.**
- PLL0: Puede usar cualquier oscilador. Frecuencia de entrada: 32KHz ~ 25MHz. Salida hasta el máximo de CPU.
- PLL1: Dedicado al USB. 48MHz.

Manejo de interrupciones (prof. Sanchez)

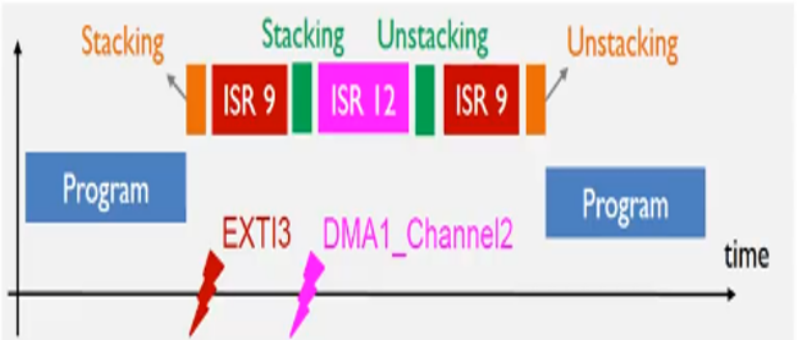
- NVIC = Control de Interrupciones de vectores anidados.
- Baja latencia.
- Controla tanto interrupciones como excepciones.
- Admite 35 interrupciones.
- 32 Niveles de prioridad.
- Tabla reubicable.
- Interrupción no enmascarable.
- Generación de interrupciones por software.
- Anidadas = atención anidada o múltiple de las interrupciones.
- Vectorizadas en flash.
- Las tablas vectoriales se componen por el código de interrupción (8 bits) y la dirección de la ISR.







- Para nuestra placa el bit 0 de direccionamiento siempre debe ser 1.
- Tail Channing: Suprime el delay en los procesos de unstack y stack de las ISR (12 ciclos cada uno) en 6 ciclos totales.



- Se soportan 256 interrupciones. (-16 - -1 excepciones) (0 - 239 interrupciones).
- Algunas interrupciones siempre están activas.
- No hay registros centralizados para habilitar o deshabilitar las interrupciones. Cada una se controla por su correspondiente componente.
- Para interrupciones periféricas existen los registros ISER (set) e ICER (clear).

Timers

- Existen 4 timers de 32 bits.
- Cada uno puede interrumpir o configurar como contador.
- Cada uno tiene su Time Counter y Presacler.
- Post reset, TC =0. Cuando llega al máximo vuelve a 0.
- Existen registros de match que es un valor seteado. Cada vez que incrementa TC, se compara con el valor de match, y en caso de igualdad, puede generar una interrupción o operar sobre el contador con cierta configuración.
- También existen los registros de captura. Se usa para capturar señales de entrada, obteniendo valor de TC, o contar los flancos, etc.
- Registros más usados:

PR	Prescaler
PC	Resolución del timer.
TC	Timer counter
TCR	Enables del timer counter
CTCR	Control de modos
MCR	Configuración match
IR	Flags
EMR	Outputs

- Recordar que también hay que habilitarlos con el PCONP y su fuente con PLCKSEL.
- Cálculos:

  $Tcclk = \frac{1}{cclk}$

  $Tres = PR + 1 \ / \ CCLK$ Es el tiempo que tarda en incrementar el TC.

  $Match = TiempoMatch \ / \ Tres$

- Cada timer tiene dos canales de captura, digamos que “saca una foto” al valor de TC o lo incrementa segun la configuración (Timer o counter).

ADC

- Conversor de aproximaciones sucesivas de 12 bits.
- 8 entradas multiplexadas.

- Modo apagado: es para configurarlo en ahorro de energía cuando no realiza conversiones.
- Los valores de referencia suelen ser gnd y 3.3, a no ser que se configure una externa.
- Frecuencia de operación: 13MHz.
- Una conversión cuesta 65 ciclos de reloj o 64 en modo burst.
- Tasa de conversión: 13MHz / 65 = 200KHz.
- Modo burst: es la capacidad de realizar conversiones continuas de múltiples canales sin intervención del cpu.
- Se puede activar con timers.
- Las interrupciones se pueden hacer por polling del DONE, interrumpir con timer al periodo de muestreo o configurar directamente las flags.

DAC

- 10 bits de resolución.
- Rango máximo de actualización 1MHz (se configura con el bias).
- Salida bufereada.
- Modo apagado.
- La conversión es Kelvin Varley. *Recordar.
- La amplitud de salida es 3.3 x InputValue / 1024
- No se controla con el PCONP, con el pin AOUT.
- El BIAS configura el consumo o lo regula para hacerlo trabajar a distinta frecuencia.

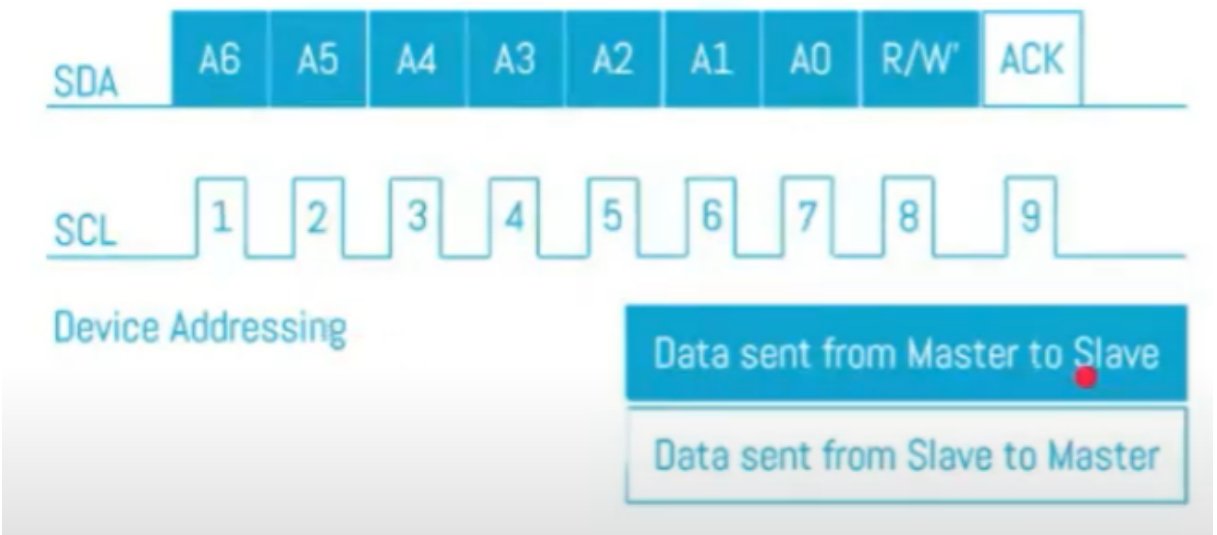
DMA

- La idea principal es acceder a memoria sin pasar por el procesador.
- Los periféricos y SRAM están mapeados (M2M).
- Las transferencias pueden sincronizarse ante algún evento.
- Las regiones de memoria también se configuran.
- Son 8 CHANNELS, y pueden funcionar una transferencia cada uno.
- Las lineas de solicitud son 16.
- El ancho maestro son 32bits.
- Puede interrumpir.
- Admite big y little endian. (en reset little).
- FIFO interna de 4 palabras.
- Existen señales de respuesta y solicitud para activar las transferencias.
- Los canales tienen prioridad FIJA, es decir el 0 tiene mayor prioridad que todos y así sucesivamente. Si se está transmitiendo por el de menor y luego

se activa el mayor, completa la transferencia antes de cambiar. Se llama transferencia delgada.

I2C

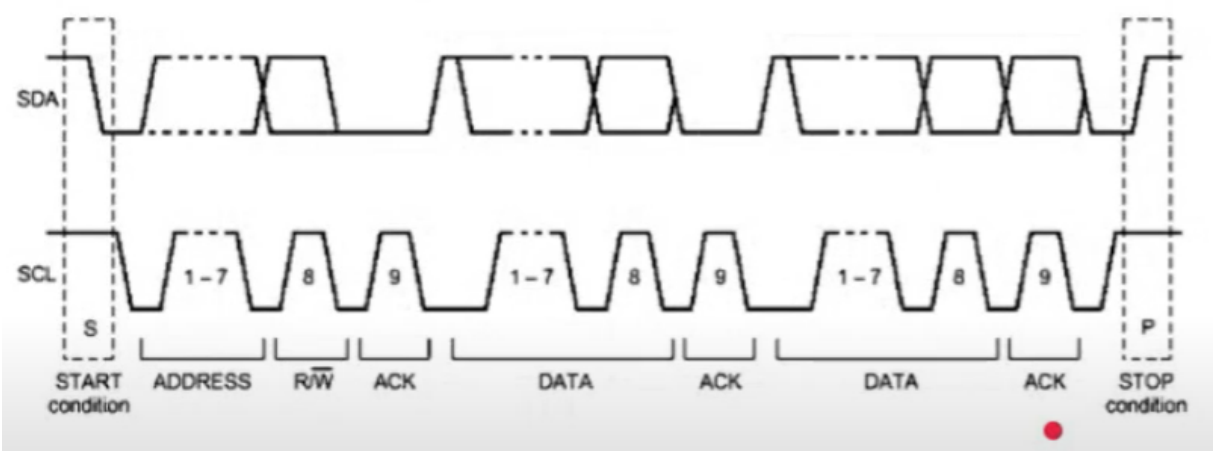
- Sistema de comunicación.
- Diseñado como un protocolo de comunicación entre diferentes Ics a una mother usando un simple bus.
- Es síncrono. (comparten un reloj).
- Se distingue de SPI , porque es más un protocolo y usa solo dos cables para la comunicación.
- 1 cable es para el clock y otro para datos bidireccional.
- Generalmente tiene una frecuencia de transmisión de 100KHz a 400Khz.
- Las lineas son open drain, requieren pullup.
- Para la comunicación entonces, se envian ceros.
- Para que los datos sean validos en la linea SDA, no deben cambiar mientras SCL está en high.
- El cambio se debería hacer cuando SCL esta en low.
- Soporta 3.3 y 5V.
- Se usa maestro esclavo



- La secuencia start y stop generalmente sigue un orden:
- Cuando el master habla a los esclavos, se debe notificar. Esto es la secuencia de start: Cuando SCL está en high y el SDA va de High a low. Y cuando termina, el sda va de low a high y el SCL está todavia en high.
- I2C transfiere 1 byte a la vez (8 bits).
- Cada vez que se completa una transmisión, el receptor envía un ACK con la siguiente secuencia:

- Tx transmite 1B, durante 8CLK y libera SDA para el pulso de ACK. Rx genera un ACK obligatoriamente despues de cada transmisión, generando un low de SDA. (9no clk).
- Si el Rx no recibió el ACK, puede ser que:

1. El esclavo receptor no hace el ACK donde el maestro puede frenar o volver a enviar.
2. El esclavo receptor envia el ACK pero despues tiene un delay para volver a recibir y el maestro toma la decicisón de 1.
3. Si el receptor como maestro comienza la transmisión y genera el ACK, como espera la señal del transmisor esclavo de fin de dato, no genera ningun ACK. En este caso, el transmisor esclavo suelta la linea para que el master vuelva a enviar o se frene.



- **Transmisión Maestro-Esclavo**
- El Maestro envía la secuencia de inicio para comenzar la transacción.
- A continuación, el Maestro envía la dirección del Esclavo de 7 bits y el bit R/W' se pone a cero. porque el Maestro está escribiendo al Esclavo.
- El Esclavo lo reconoce poniendo el bit ACK bajo.
- Una vez que el Esclavo reconoce la dirección, el Maestro puede ahora enviar datos al Esclavo byte por byte. El Esclavo tiene que enviar el bit ACK después de cada byte que recibe.
- Esto continúa hasta que el Esclavo ya no puede recibir datos y NO envía el bit ACK.
- Es entonces cuando el Maestro se da cuenta de que el Esclavo se ha esfumado (ya no acepta más) y entonces PARA la transacción (o reinicia).
- Vemos que la transferencia de datos nunca cambia su dirección. Los datos siempre fluyen de Maestro a Esclavo, lo que hace que la configuración sea bastante fácil.
- Un ejemplo de este caso sería como realizar operaciones de escritura de páginas en un chip EEPROM.

- **Transmisión Esclavo-Maestro**
- El Maestro envía la secuencia de arranque, seguida de la dirección del Esclavo de 7 bits y el bit R/W' puesto a 1.
- Ponemos el bit R/W' a 1 porque el Maestro está leyendo al Esclavo.
- El Esclavo reconoce la dirección, por lo que está listo para enviar los datos ahora.
- El Esclavo sigue enviando datos al Maestro, y el Maestro sigue enviando ACK al Esclavo después de cada byte hasta que ya no puede aceptar más datos.
- Cuando el Maestro tiene ganas de terminar la transacción, no envía el ACK, terminando así con la secuencia STOP.
- Un ejemplo de este caso podría ser un convertidor analógico a digital (ADC) enviando datos al microcontrolador continuamente. El microcontrolador acepta los datos todo el tiempo que quiera, después de lo cual detiene/termina la ejecución.

- **Transmisión Bidireccional**
- El Maestro envía la secuencia de inicio, seguida de la dirección del esclavo de 7 bits y el bit R/W'.
- El Esclavo reconoce la dirección.
- En función del valor del bit R/W', se efectúan operaciones de lectura/escritura (como en los dos casos anteriores).
- Sea cual sea el caso, siempre termina con el receptor no enviando el ACK.
- Hasta ahora, en los dos casos anteriores, hemos visto que el Maestro cerraba la conexión. Pero en este caso, el Maestro intenta un repetido START.
- Y todo el proceso se repite de nuevo, hasta que el Maestro decide STOP.
- Como podemos ver, un cambio en la dirección de la transferencia de datos podría ocurrir dependiendo de los bits R/W' en toda la transacción.
- Un ejemplo de este caso podría ser la realización de una lectura secuencial de un chip EEPROM. Es bidireccional porque la CPU primero escribe la dirección desde donde le gustaría empezar a leer, seguido de la lectura desde el dispositivo. Es como, a menos que le digas al dispositivo desde donde te gustaría empezar a leer, ¿cómo empezaría a enviarte los datos?