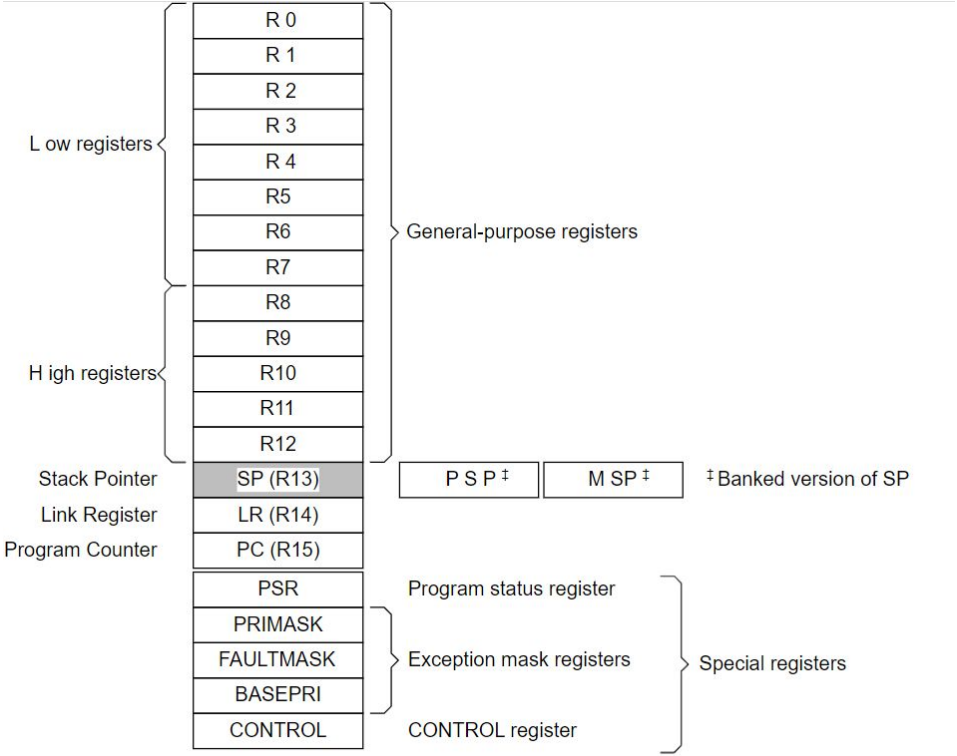
Arquitectura del Microcontrolador II

Temas

- Parte II
  - Manejo de Excepciones y tabla de vectores
  - Sistema de distribución de clock
  - Periféricos de Debug

Repaso

Recordemos: Registros del procesador



Recordemos: Dispositivos bajo un mapa de memoria

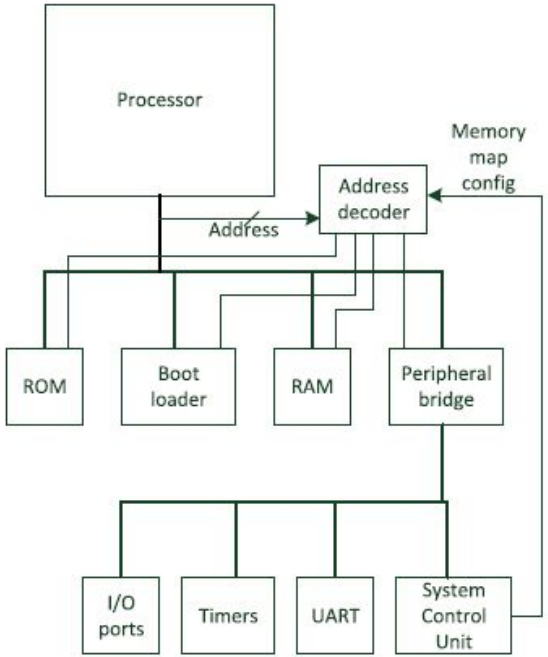


FIGURE 6.19

A simplified memory system with configurable memory map

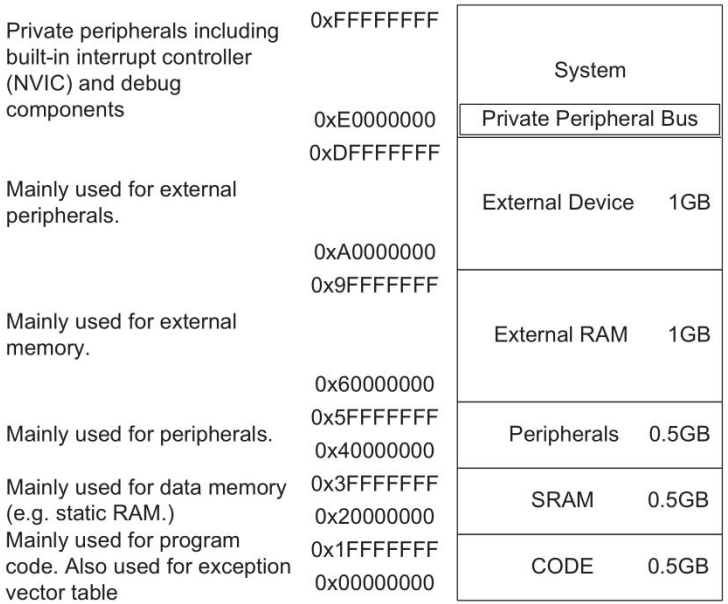


FIGURE 4.18

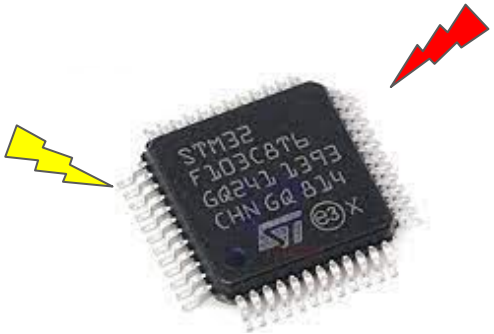
Memory map

Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

Excepciones

Eventos y excepciones

- En un sistema embebido **nuestros programas comúnmente responderán a eventos**, por ejemplo
  - Un ADC que indica que hay una muestra lista
  - Un pulsador conectado al GPIO que fue presionado por el usuario
  - La captura de un byte recibido por un puerto de comunicaciones
- Estos eventos pueden asociarse a **interrupciones** cuando provienen de periféricos o de la detección de **fallas imprevistas**
- Por lo tanto, en los **procesadores Cortex** se tiene el concepto de **excepciones.** Las interrupciones son un caso particular de excepciones.
- Cuando ocurre una excepción se debe
  - Interrumpir rápidamente el flujo de ejecución del procesador y
  - Cambiar de contexto para ejecutar una subrutina que responda al evento

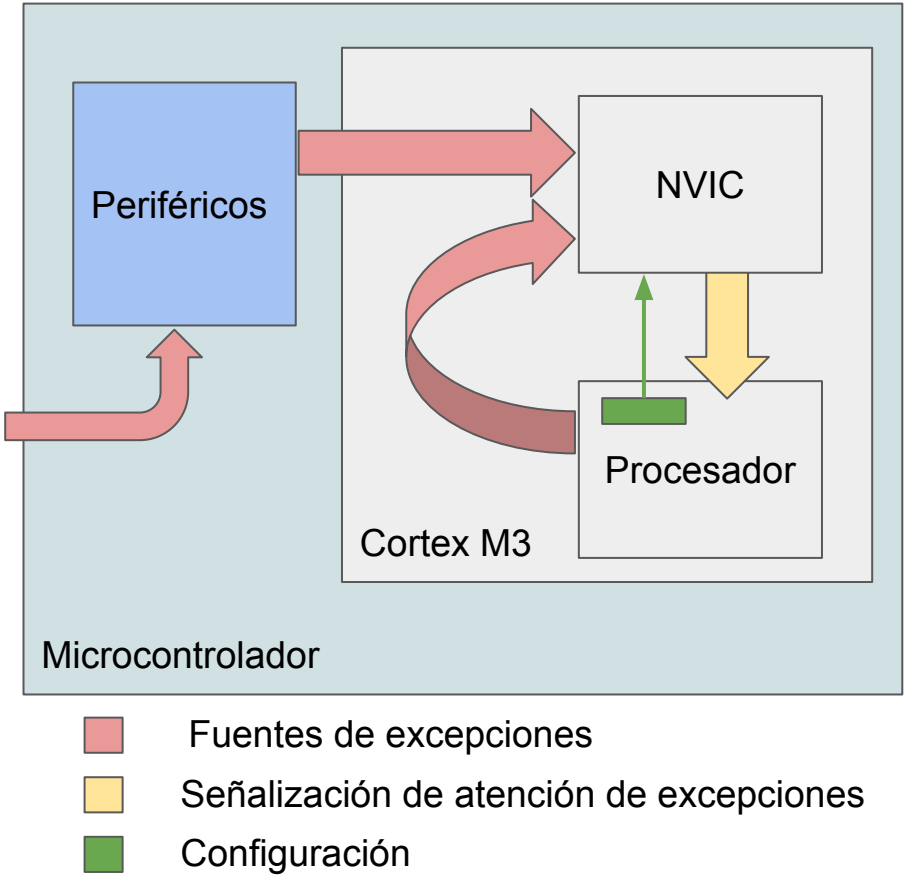


Manejo de excepciones

- El procesador posee los siguientes elementos para administrar todas las posibles excepciones (fallas e interrupciones):
  - **Un elemento de hardware: el NVIC**, es un periférico que detecta las excepciones, las recuerda y ordena para “presentarselas” al procesador
  - **Un elemento de software: la tabla de vectores**. Es simplemente un arreglo de direcciones de memoria, cada una de las cuales apunta a una rutina de atención para cada tipo de excepción
  - Un set de elementos internos: Los **registros especiales** donde mantiene la excepción en ejecución y el control de enmascaramiento, y registros en el segmento **System Control Block** que contienen configuraciones como la prioridad de excepciones y retiene información sobre el disparo de excepciones de falla

Excepciones - NVIC

- Las excepciones tienen prioridad configurable
- Se pueden anidar (la N es de Nested)
  - Para el procesador termina siendo muy parecido a un simple cambio de contexto.
  - Eso permite atender “infinitas” excepciones
  - Implementa la prioridad en la que se ejecutarán
- Se ejecutan desde una tabla (la V de Vector)
  - Hay una tabla configurable en memoria con todas las posibles fuentes de excepción y la dirección de la subrutina que la maneja
  - El NVIC instantáneamente asocia la excepción a su lugar de la tabla y recupera dirección de la subrutina
- El NVIC recuerda, para cada excepción, si está pendiente o no aunque la fuente de excepción haya “bajado la mano” (pulsed vs. level)
- El procesador mantiene la información de la ejecución de excepciones y su enmascaramiento



Tipos de excepciones

- **Reset:** Al encender o activar el reset del microcontrolador
- **NMI:** Non Maskable Interrupt puede ser generada por software o por algún periférico y no puede ser enmascarada (sus fuentes son el watchdog independiente y falla en clock).
- **Hard fault:** Error durante el procesamiento de excepciones o porque no hay un manejador asignado a una Fault de menor prioridad
- **Memory management fault:** Disparada cuando se quiere acceder a una instrucción en zona marcada XN o memoria protegida.
- **Bus fault:** Error en una transacción en memoria.
- **Usage fault:** Instrucción indefinida o estado inválido durante la ejecución de una instrucción, acceso no alineado, error al retornar del manejador de una excepción, división por cero.
- **SVCall:** Supervisor Call usada en OSs
- **PendSV:** También usada en OSs como lógica de interrupción.
- **SysTick:** Disparada por el timer de systema del Cortex M3
- **Interrupt (IRQ):** Asignadas a las interrupciones de periféricos.

Exception number(1)	IRQ number(1)	Exception type	Priority	Vector address or offset(2)
1	-	Reset	-3, the highest	0x00000004
2	-14	NMI	-2	0x00000008
3	-13	Hard fault	-1	0x0000000C
4	-12	Memory management fault	Configurable(3)	0x00000010
5	-11	Bus fault	Configurable(3)	0x00000014
6	-10	Usage fault	Configurable(3)	0x00000018
7-10	-	-	-	Reserved
11	-5	SVCall	Configurable(3)	0x0000002C
12-13	-	-	-	Reserved
14	-2	PendSV	Configurable(3)	0x00000038
15	-1	SysTick	Configurable(3)	0x0000003C
16-83	0-67	Interrupt (IRQ)	Configurable(4)	0x00000040 and above(5)

pm0056 - stm32f10xxx Cortex M3 programming manual, ST

Tabla de vectores

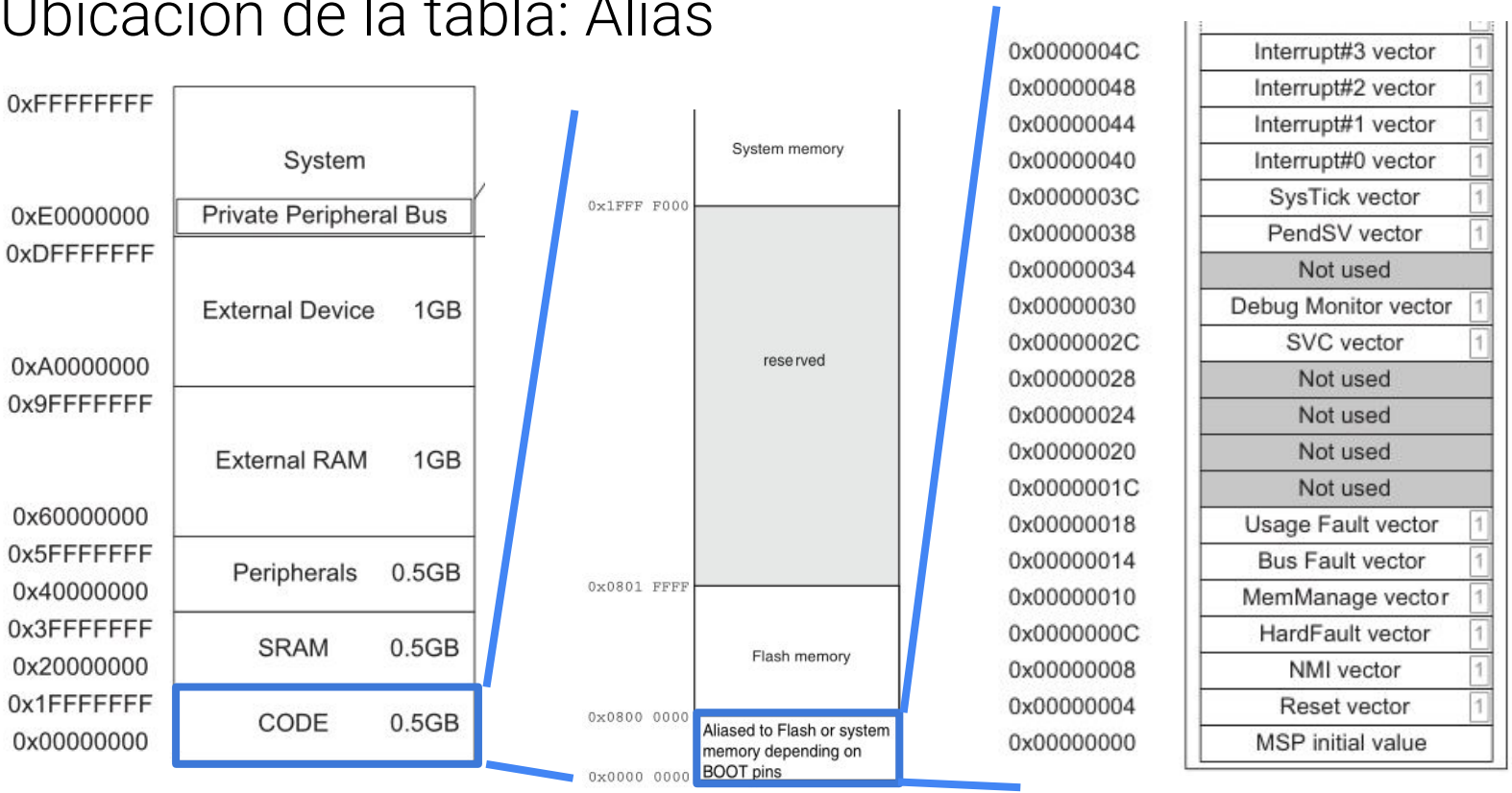
Tabla de vectores

- Cada excepción tiene su espacio en la tabla de vectores donde se indica la **dirección de la subrutina** (función) que la atiende.
- Cuando se produce una excepción el NVIC determina si debe atenderse y de ser así dispara un proceso por el cual se cargará en el PC la dirección correspondiente según la tabla
- Esta tabla tiene importancia también porque **el RESET es una de las excepciones**!
- Debido a esto y para poder cambiar dinámicamente las rutinas de atención se puede querer **relocalizar la posición de la tabla**

0x0000004C	Interrupt#3 vector
0x00000048	Interrupt#2 vector
0x00000044	Interrupt#1 vector
0x00000040	Interrupt#0 vector
0x0000003C	SysTick vector
0x00000038	PendSV vector
0x00000034	Not used
0x00000030	Debug Monitor vector
0x0000002C	SVC vector
0x00000028	Not used
0x00000024	Not used
0x00000020	Not used
0x0000001C	Not used
0x00000018	Usage Fault vector
0x00000014	Bus Fault vector
0x00000010	MemManage vector
0x0000000C	HardFault vector
0x00000008	NMI vector
0x00000004	Reset vector
0x00000000	MSP initial value

pm0056 - stm32f10xxx Cortex M3 programming manual, ST

Ubicación de la tabla: Alias



Datasheet - STM32F103x8 STM32F103xB
Doc ID 13587 Rev 12, ST

Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

Booteo - Posición de la tabla de vectores

- Cuando se prende el dispositivo, automáticamente carga el stack pointer (R13) de la posición 0x00 de la tabla, y el PC (R15) de la posición 0x04 (recordar excepción de RESET).
- Por lo tanto debe saber dónde está la tabla! siempre la busca en la pos. 0
- ST implementó una estrategia para poder “cambiar” la posición de la tabla durante el encendido, según los pines de boot.
- En realidad, lo que hace es generar un alias y el procesador “ve” la dir 0x00000000 en el mismo lugar que la 0x80000000 (para el caso de la Flash) o en el 0x1FFFF000 para System

Table 9. Boot modes

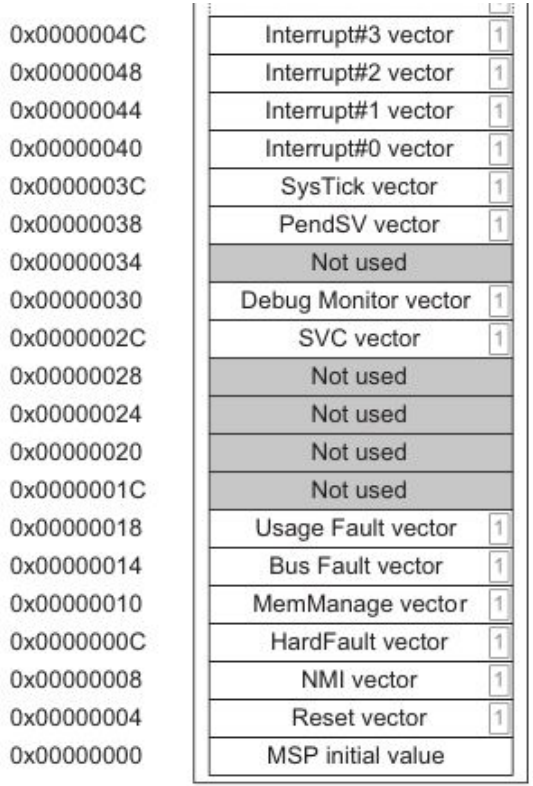
Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as boot space
0	1	System memory	System memory is selected as boot space
1	1	Embedded SRAM	Embedded SRAM is selected as boot space

RM0008 - Reference manual, ST - Tabla 63

Relocalización de la tabla de vectores

- Además de contar con un Alias para ubicar la posición 0x00000000 a la Flash o la ROM de Sistema, se puede asignar un **offset para que la posición de la tabla sea distinta**.
- Esto se hace con el Vector Table Offset Register (**VTOR**) un registro que se ubica en el **System Control Block** (región de memoria de del Cortex M3)
- Luego de un reset, el VTOR vale 0, la posición predeterminada.
- Se puede mover la tabla a una nueva posición escribiendo el **offset en el registro VTOR** para que apunte a la nueva dirección
- La nueva dirección puede estar en RAM o en Flash

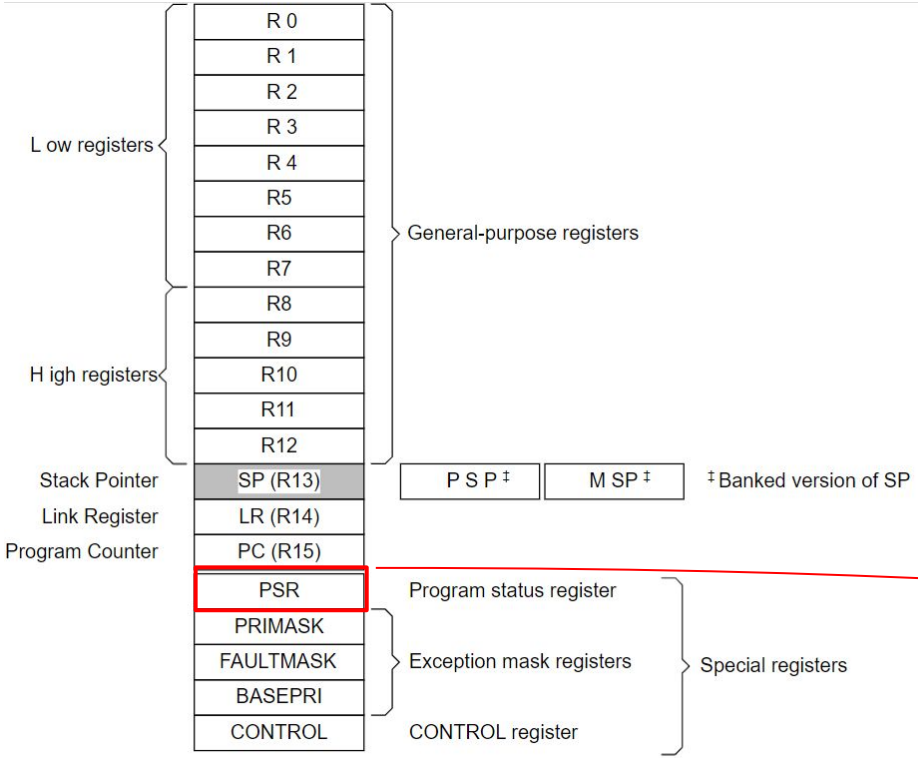
0xE000ED04	ICSR	RW or RO	0x00000000	Interrupt Control and State Register
0xE000ED08	VTOR	RW	0x00000000	Vector Table Offset Register
0xE000ED0C	AIRCR	RW	0x00000000[a]	Application Interrupt and Reset Control Register
0xE000ED10	SCR	RW	0x00000000	System Control Register
0xE000ED14	CCR	RW	0x00000200	Configuration and Control Register.
0xE000ED18	SHPR1	RW	0x00000000	System Handler Priority Register 1



Joseph Yiu - The Definitive Guide to ARM Cortex -M3 and Cortex-M4 Processors

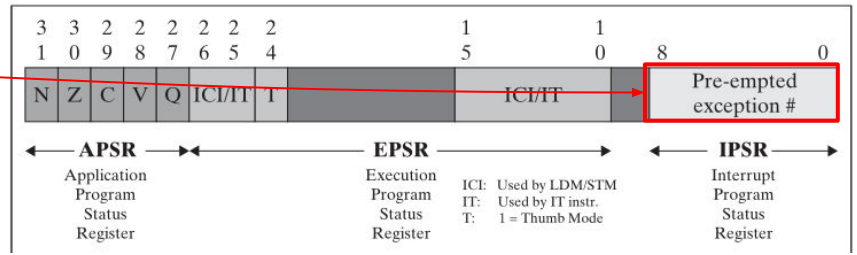
Manejo de Excepciones

Registro especial IPSR



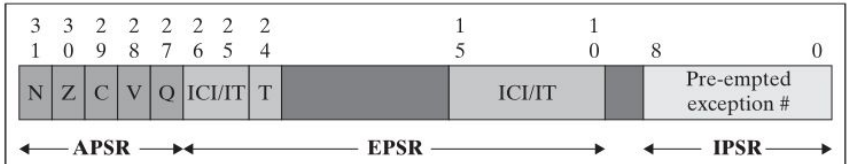
- Vimos que el Program Status Register tiene 3 partes, una de ellas es el IPSR
- Indica el tipo de excepción (es decir, el número) que se está ejecutando en el momento
- Cuando vale 0 significa “Thread mode” o modo normal de “no excepción”. Cualquier otro valor significa que se corre el vector correspondiente a ese número de tipo de excepción

Reg. especial PSR



D.W. Lewis - Fundamentals of Embedded Software with the Arm Cortex M3 (2012)

Número de excepción



Bits	Description
Bits 31:9	Reserved
Bits 8:0	**ISR_NUMBER**: This is the number of the current exception: 0: Thread mode 1: Reserved 2: NMI 3: Hard fault 4: Memory management fault 5: Bus fault 6: Usage fault 7: Reserved .... 10: Reserved 11: SVCall 12: Reserved for Debug 13: Reserved 14: PendSV 15: SysTick 16: IRQ0(1) .... 83: IRQ67(1) see *Exception types on page 32* for more information.

```
HAL_TIM_Base_Start_IT(&htim2);
```

lr	134218559
pc	0x800045e <TIM2_IRQHandl...
xpsr	16777260
primask	0
basepri	0
faultmask	0
control	0
msp	0x20004fd0
psp	0x0

```
Name : xpsr
    Hex:0x100002c
    Decimal:16777260
    Octal:0100000054
    Binary:10000000000000000000101100
    Default:16777260
```

Ejemplo: Al dispararse la interrupción del TIM2 aparece el valor 0x2C en el IPSR
Comparando las tablas vemos que corresponde al # del TIM2

-	-	-	-
-	-3	fixed	Reset
-	-2	fixed	NMI
-	-1	fixed	HardFault
-	0	settable	MemManage
-	1	settable	BusFault
-	2	settable	UsageFault
-	-	-	-
-	3	settable	SVCall
-	4	settable	Debug Monitor
-	-	-	-
-	5	settable	PendSV
-	6	settable	SysTick
0	7	settable	WWDG
1	8	settable	PVD
2	9	settable	TAMPER
3	10	settable	RTC
25	32	settable	TIM1_UP
26	33	settable	TIM1_TRG_COM
27	34	settable	TIM1_CC
28	35	settable	TIM2
29	36	settable	TIM3
30	37	settable	TIM4
31	38	settable	I2C1_EV

pm0056 - stm32f10xxx Cortex M3 programming manual, ST

D.W. Lewis - Fundamentals of Embedded Software with the Arm Cortex M3 (2012)

Excepciones de falla

- Todas las excepciones de falla, derivan en una HardFault si no existe un vector asignado
  - HardFault
  - MemManage
  - Bus Fault
  - Usage Fault
- La información de estas excepciones se almacena en registros del System Control Block para poder **depurar las fallas**
- Además se puede configurar para algunas si producir o no una excepción ante su ocurrencia (e.g. divide by zero)
- La documentación específica está en el Programming manual del Cortex M3 (PM0056)

0xE000ED10	SCR	RW	0x00000000	System Control Register
0xE000ED14	CCR	RW	0x00000200	Configuration and Control Register.
0xE000ED24	SHCSR	RW	0x00000000	System Handler Control and State Register
0xE000ED28	CFSR	RW	0x00000000	Configurable Fault Status Registers
0xE000ED2C	HFSR	RW	0x00000000	HardFault Status register
0xE000ED30	DFSR	RW	0x00000000	Debug Fault Status Register
0xE000ED34	MMFAR	RW	Unknown	MemManage Address Register[b]
0xE000ED38	BFAR	RW	Unknown	BusFault Address Register[b]

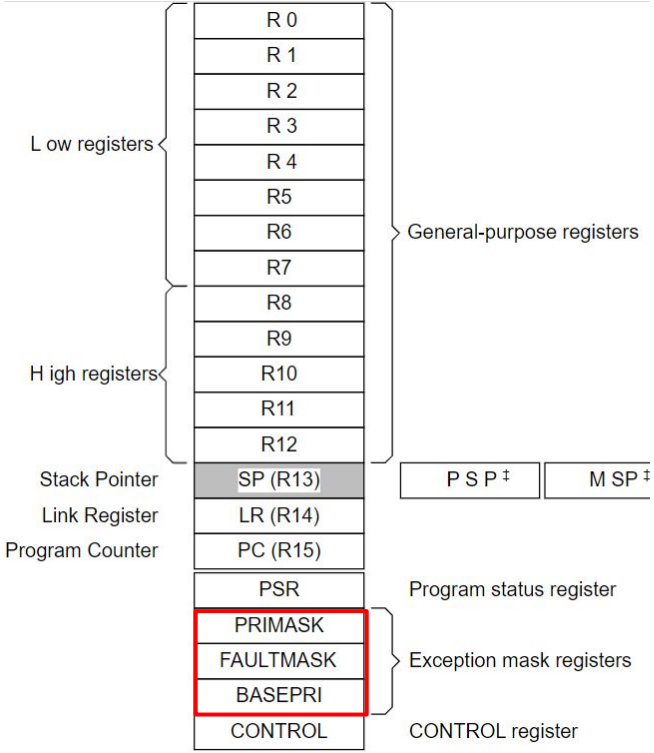
SHCSR Habilita detección de ciertas fallas.
CFSR Indica qué tipo de fallo ocurrió: MemManage (MMFSR), BusFault (BFSR), UsageFault (UFSR).
HFSR Indica si el HardFault fue por escalado de prioridad o por fallo crítico.
DFSR Indica si fue por breakpoint, watchpoint, etc.
BFAR y MMFAR direcciones de fallos de bus o memoria.

Enmascaramiento de excepciones

Habilitadas por defecto
El resto deshabilitadas

Priority	Type of priority	Acronym	Description	Address
-	-	-	Reserved	0x0000_0000
-3	fixed	Reset	Reset	0x0000_0004
-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008
-1	fixed	HardFault	All class of fault	0x0000_000C
0	settable	MemManage	Memory management	0x0000_0010
1	settable	BusFault	Prefetch fault, memory access fault	0x0000_0014
2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018
-	-	-	Reserved	0x0000_001C - 0x0000_002B
3	settable	SVCall	System service call via SWI instruction	0x0000_002C
4	settable	Debug Monitor	Debug Monitor	0x0000_0030
-	-	-	Reserved	0x0000_0034
5	settable	PendSV	Pendable request for system service	0x0000_0038
6	settable	SysTick	System tick timer	0x0000_003C

Registros especiales de enmascaramiento



- Dentro de los registros especiales, encontramos los que sirven para enmascarar interrupciones
- PRIMASK (sólo LSB)
  - es un único bit que sirve para enmascarar todas las excepciones “normales”, es decir aquellas de prioridad configurable exceptuando las de Fault
- FAULTMASK (sólo LSB)
  - Para sorpresa de nadie, enmascara las excepciones incluyendo la de tipo Fault (pero no la HardFault)
- BASEPRI (sólo [7:4])
  - Permite determinar un nivel a partir del cual se enmascaran las excepciones configurables
  - Un valor de 0 significa que BASEPRI está deshabilitado
  - Un valor N != 0 significa que desde el nivel N (inclusive) hasta el último, las interrupciones quedarán enmascaradas, y sólo se permitirán las de mayor prioridad que N

Nivel de prioridad

- El NVIC tiene los registros IPR donde, **para cada tipo de excepción**, se escribe el nivel de prioridad
- En los STM32, cada IPR tiene **sólo 4 bits útiles**
- Es decir que **hay 16 niveles de prioridad** de excepciones, asignables a las interrupciones de la tabla
- Como se utilizan los bits más significativos, los valores van del 0 al 255 en saltos de a 16
  - (Aunque en la librería de software lo vemos con valores del 0 al 16)
- El **número más bajo** es la que tiene **mayor prioridad**
- Este nivel determina si una interrupción puede **apropiarse de la ejecución (“preempt”)** frente a otra de menor nivel

B3.4.8 Interrupt Priority Registers, NVIC_IPR0 - NVC_IPR123

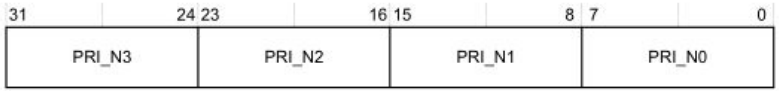


Figure B3-33 NVIC_IPR*n* Register bit assignments

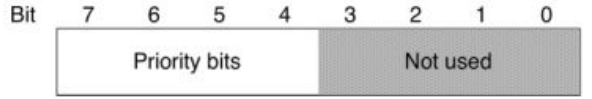


Figure 15: The content of IPR register on an STM32 MCU based on Cortex-M3/4/7 core

RM0008 - Reference manual, ST

Manejo de excepciones - Sub-prioridad o prioridad de grupo

- Es opcional configurar un nivel de **sub-prioridad** con los bits **PRIGROUP** del **registro AIRCR del System Control Block**
- Pueden seleccionarse un subgrupo de los bits del IPR que determinarán una sub-prioridad
- Entonces **los bits más significativos seguirán determinando el nivel de prioridad**, es decir cuál puede apropiarse de la ejecución
- Pero **los menos significativos indicarán el orden de prioridad dentro de ese nivel**, que simplemente indica cuál se ejecuta a continuación, pero no permite la apropiación.

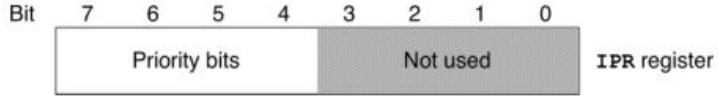
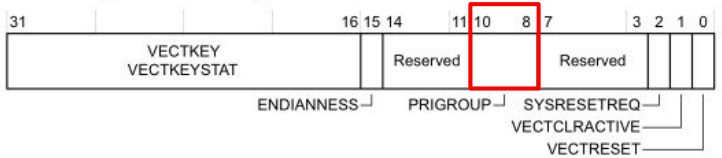


Figure 15: The content of IPR register on an STM32 MCU based on Cortex-M3/4/7 core

B3.2.6 Application Interrupt and Reset Control Register, AIRCR



[10:8]	RW	PRIGROUP	Priority grouping, indicates the binary point position. For information about the use of this field see *Priority grouping* on page B1-636. This field resets to 0b000.

Manejo de excepciones - Ejemplos Sub-prioridad

IPRx

1	1	0	1				

PRIGROUP

0	0

IPRz

1	1	0	0				

En este caso no hay ningún rango asignado a la subprioridad. La IRQ tipo z se aproriará de la ejecución frente a la de tipo x por tener mayor nivel (menor número)

IPRx

1	1	0	1				

PRIGROUP

1	0

IPRz

1	1	0	0				

En este caso los dos últimos bits corresponden a la subprioridad. Ambas IRQ tienen el mismo nivel. Por lo tanto, la de tipo z no se apropiará de la ejecución, pero si están ambas esperando, se ejecutará primero.

Manejo de excepciones - Ejemplos

- El nivel de prioridad determina cual interrupción o excepción puede apropiarse de la ejecución

- Mientras que la subprioridad o “prioridad de grupo” simplemente define cual es la próxima que se ejecutará, dado el mismo nivel

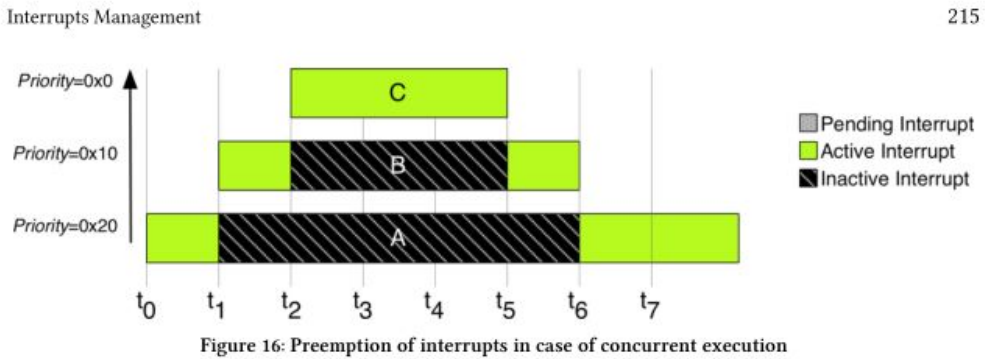
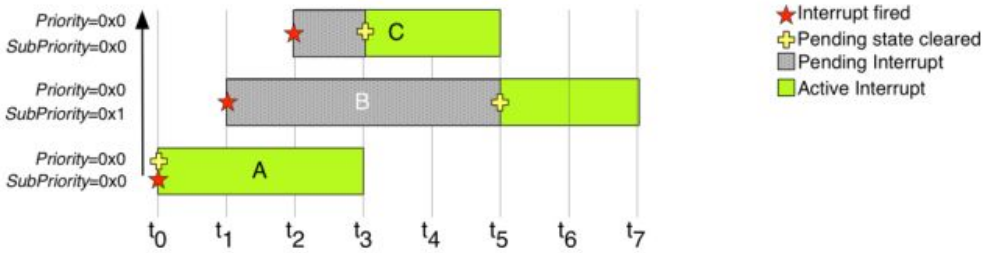


Figure 16: Preemption of interrupts in case of concurrent execution



Carmine Noviello - Mastering STM32 (2016)

Registros NVIC

- Habilitar y deshabilitar interrupciones (ISER, ICER)
- Forzar o borrar interrupciones pendientes al escribir (ISPR, ICPR) e identificar cuales siguen pendientes al leer
- Identificar la excepción activa (IABR) durante la ejecución de la subrutina
- Los ya mencionados registros IPR para almacenar la prioridad

Address	Name	Type	Reset	Description
0xE000E004	ICTR	RO	-	Interrupt Controller Type Register, ICTR
0xE000E100 - 0xE000E11C	NVIC_ISER0 - NVIC_ISER7	RW	0x00000000	Interrupt Set-Enable Registers
0xE000E180 - 0xE000E19C	NVIC_ICER0 - NVIC_ICER7	RW	0x00000000	Interrupt Clear-Enable Registers
0xE000E200 - 0xE000E21C	NVIC_ISPR0 - NVIC_ISPR7	RW	0x00000000	Interrupt Set-Pending Registers
0xE000E280 - 0xE000E29C	NVIC_ICPR0 - NVIC_ICPR7	RW	0x00000000	Interrupt Clear-Pending Registers
0xE000E300 - 0xE000E31C	NVIC_IABR0 - NVIC_IABR7	RO	0x00000000	Interrupt Active Bit Register
0xE000E400 - 0xE000E4EC	NVIC_IPR0 - NVIC_IPR59	RW	0x00000000	Interrupt Priority Register

https://developer.arm.com/documentation/100165/0201/Nested-Vectored-Interrupt-Controller/NVIC-programmers-model/Table-of-NVIC-registers

Interrupciones de periféricos

7	settable	WWDG	Window watchdog interrupt	0x0000_0040
8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044
9	settable	TAMPER	Tamper interrupt	0x0000_0048
10	settable	RTC	RTC global interrupt	0x0000_004C
11	settable	FLASH	Flash global interrupt	0x0000_0050
12	settable	RCC	RCC global interrupt	0x0000_0054
13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C
19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070
20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074

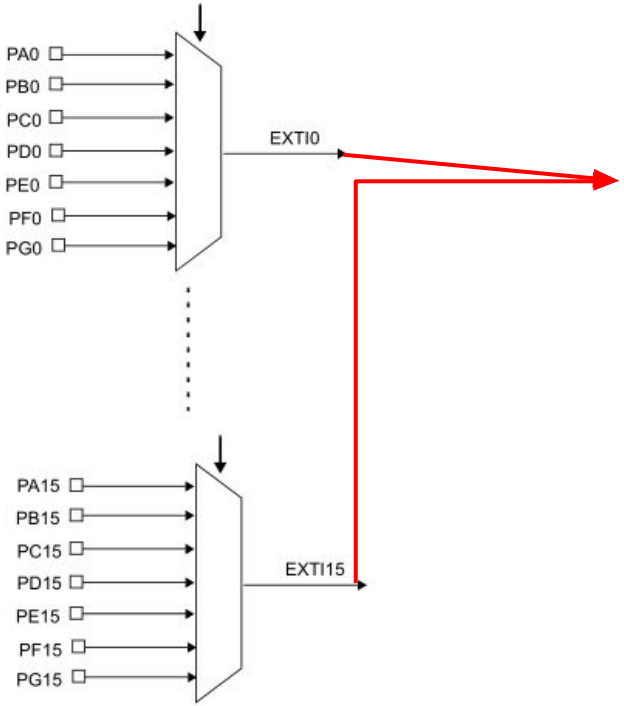
A partir de la 0x40 empiezan las interrupciones de **periféricos**

RM0008 - Reference manual, ST - Tabla 63

Controlador EXTI

- El uC tiene muchas entradas de GPIO y asignar cada posible fuente de interrupción de esos pines en el NVIC no es posible o no conviene para no implementar un NVIC “monstruoso”
- ST decidió concentrar esas interrupciones en un periférico adicional: el External Interrupt/Event Controller **EXTI**
- Concentra interrupciones al GPIO, permite implementar interrupciones por software, y también tiene algunas líneas de otros periféricos
  - Es interesante que detecta pulsos de menor duración que el clock del bus
- La configuración del EXTI varía entre dispositivos, por lo que veremos concretamente las características para el STM32F103c8

Configuración del EXTI



settable	EXTI0	EXTI Line0 interrupt	0x0000_0058
settable	EXTI1	EXTI Line1 interrupt	0x0000_005C
settable	EXTI2	EXTI Line2 interrupt	0x0000_0060
settable	EXTI3	EXTI Line3 interrupt	0x0000_0064
settable	EXTI4	EXTI Line4 interrupt	0x0000_0068
settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C
settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0
settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
settable	USBWakeup	USB wakeup from suspend through EXTI line interrupt	0x0000_00E8
settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044

18 líneas EXTI en total

RM0008 - Reference manual,, ST

Stacking

- Las excepciones se comportan en el Cortex como subrutinas comunes de C
  - Esto permite tener un anidamiento de muchas excepciones
- Cuando ocurre un cambio de contexto, siempre se debe guardar el estado de ejecución para poder volver luego y continuar lo que se dejó “a medias”
- El mecanismo de excepción incluye el disparo del stacking automático, sin requerir instrucciones extra, pero, insume tiempo

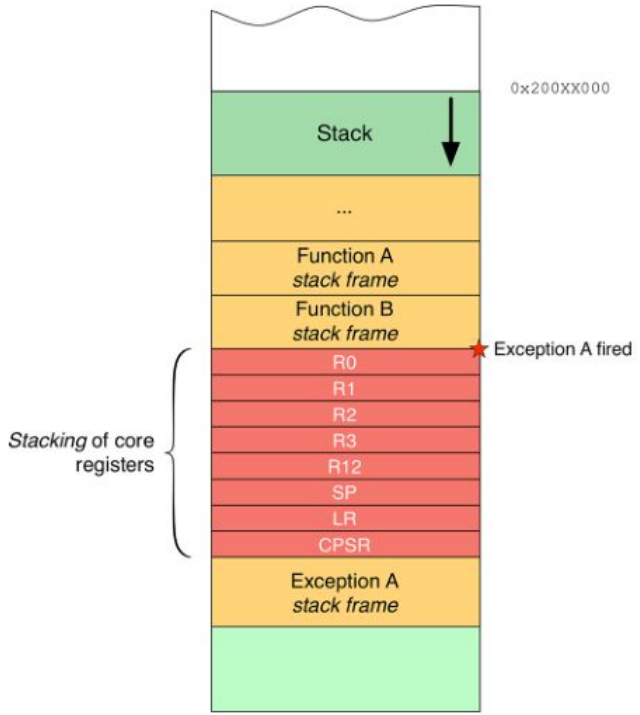


Figure 2: How core registers are stacked by the CPU on exception entrance

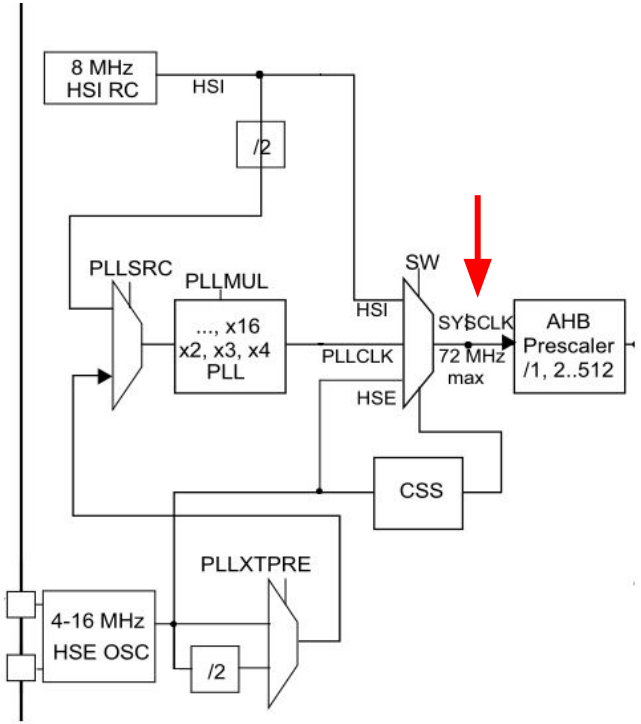
Atención de una excepción

1. Se marca la excepción x como pendiente (NVIC->ISPR [x])
2. Si está activada (NVIC->ISER [SETENA x]) el procesador evalúa si debe ejecutarse
   a. Si está en modo Thread o con menor prioridad que la actual (Registro especial IPSR) se comparan prioridades según NVIC->IPRx teniendo en cuenta SCB->AIRCR [PRIGROUP]
   b. Se verifica que supere los niveles de enmascaramiento (PRIMASK, FAULTMASK, BASEPRI)
3. Si es aceptada, se vuelve activa (NVIC->IABRx se pone en 1).
4. Si no es *tail-chained* o *late-arrival* se realiza el stacking (push automático de R0-R3, R12, LR, PC, xPSR)
5. Se carga la rutina de atención
   a. Si durante el proceso previo llegó una excepción de mayor prioridad se ejecuta directamente la nueva, dejando esta como pendiente (*late-arrival*), aprovechando de esa manera el stacking.
   b. Si no, se recupera la dirección de la rutina de atención de la Tabla de Vectores y se carga en el PC
   c. LR se carga con código especial EXC_RETURN, que cuenta con un espacio para dejar indicado el modo de retorno (privilegiado/no, MSP/PSP).
6. Se ejecuta la rutina de atención.
7. Cuando termina, se realiza EXC_RETURN con la instrucción bx o equivalente
   a. Si hay otra ISR pendiente con suficiente prioridad, se ejecuta esa sin destacking (*tail-chaining*).
   b. Si no hay más excepciones pendientes, se realiza el destacking

Sistema de Clock

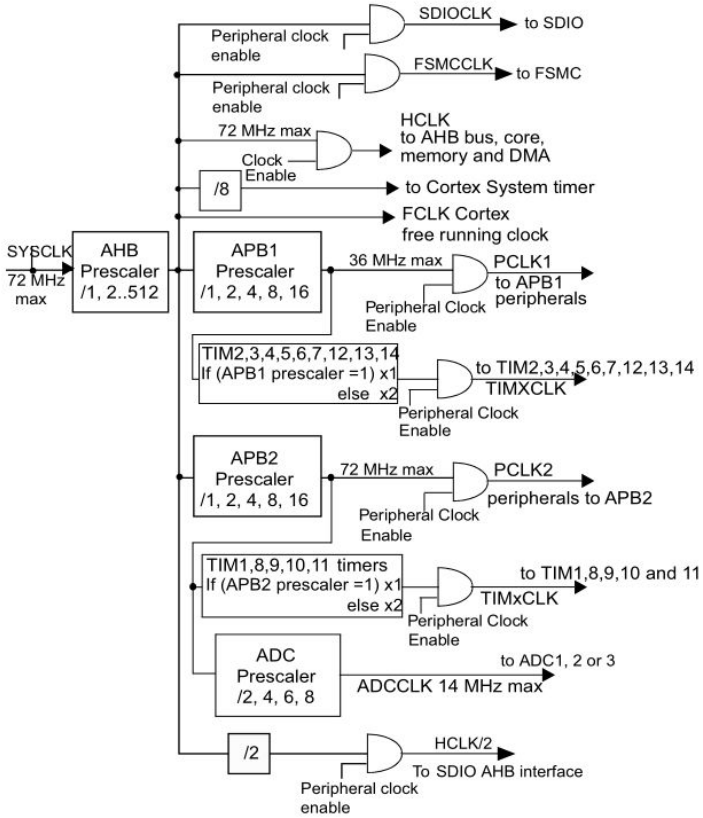
Fuente de clock principal

- Fuente de clock principal
  - De origen interno (HSI): Tiene un oscilador interno de 8 MHz de precisión “media”
  - De origen externo (HSE): Puede conectarse un oscilador o un resonador de cristal o cerámico para obtener un clock de mayor precisión
- A partir de la fuente de clock, intermediando un PLL se obtiene el clock principal del sistema: SYSCLK



Clock de Periféricos

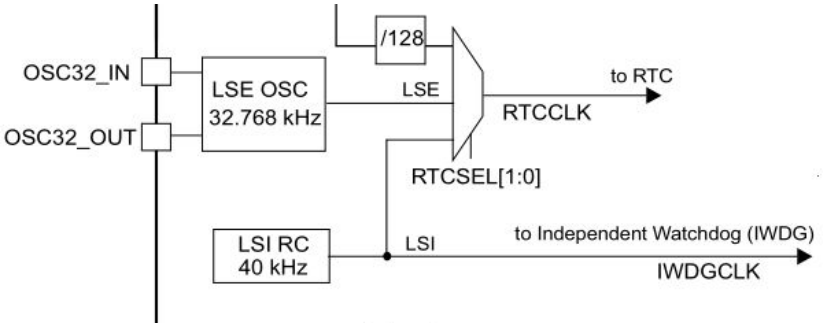
- Del SYSCLK se obtienen los clocks para el resto de los periféricos (excepto algunos que no se muestran en esta imágen)
- Es importante notar que todos los periféricos tienen un “clock enable”
  - Debemos habilitar el clock respectivo para usarlo - Incluso el GPIO!
  - Puede ahorrarse energía deshabilitando lo que no se usa



RM0008 - Reference manual, ST

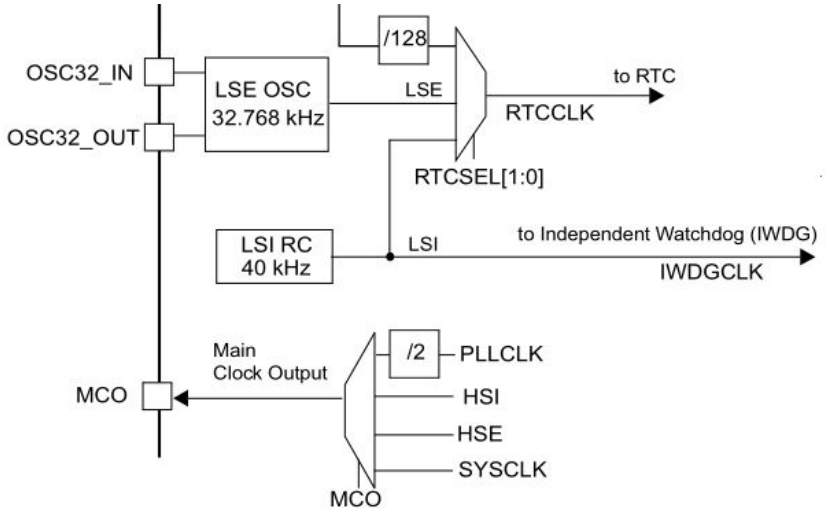
Fuente de clock de tiempo real

- Fuente de clock “de tiempo real” o “lento”
- **No confundir** “tiempo real” en este contexto significa poder contar el tiempo en segundos exactos para medir minutos, horas, etc
- De origen interno: Un oscilador RC de baja precisión, aunque se puede calibrar
- De origen externo: Puede conectarse un resonador de 32.768 kHz o al HSE/128



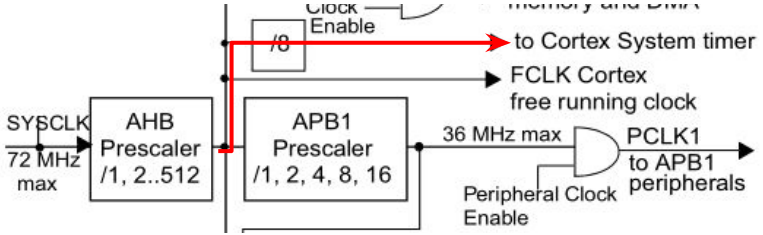
Características de interés

- Clock Security System (CSS)
  - Permite recuperar el microcontrolador ante la pérdida del HSE
  - Se cambia la fuente de clock al HSI y se ejecuta la NMI indefinidamente hasta que se limpia el pending bit del CSS interrupt
- Watchdog independiente
  - La fuente interna LSI que mencionamos para el RTC sirve para alimentar un watchdog que de esta manera será independiente del resto del sistema
- Externalización del clock
  - Puede sacarse la señal de clock de diversas fuentes por un pin



Cortex System Timer (SysTick)

- Desde el clock y a través de un divisor se extrae una señal de clock usada en el Cortex como referencia de un timer interno llamado **SysTick**
- Es un timer capaz de producir una excepción en cada “tick”
- Los registros que lo controlan se ubican en el segmento del System Control Block
- Se incorporó por **portabilidad** de código para microcontroladores con núcleo Cortex
  - La librería HAL de ST lo usa como timer para sus funciones (HAL_Delay)
  - Los sistemas operativos programados para Cortex-M3 lo usan para el tick del sistema operativo (fundamental para esquemas multitarea)



0xE000E010	STCSR	SysTick Control and Status Register
0xE000E014	STRVR	SysTick Reload Value Register
0xE000E018	STCVR	SysTick Current Value Register
0xE000E01C	STCR	SysTick Calibration Value Register

RM0008 - Reference manual, ST

Tasa de clock y ejecución de instrucciones

- La tasa de clock será importante porque dentro del rango admitido determina la tasa de instrucciones; A grandes rasgos será de 1 instrucción por ciclo
- Sin embargo hay factores que disminuyen esta performance:
  - Algunas instrucciones demoran más ciclos
  - La operación del pipeline puede verse perturbada por cambios de contexto, o cuando si la decodificación de una instrucción depende de que se complete la ejecución de la instrucción previa
- A su vez, si bien los pipelines permiten ejecutar accesos a gran velocidad, la memoria flash tiene un límite, y **ambos valores están tecnológicamente en puja**
  - Se introducen wait states para esperar a que las instrucciones o los datos estén disponibles

- 0 wait states, if 0 < SYSCLK ≤ 24 MHz
- 1 wait state, if 24 MHz < SYSCLK ≤ 48 MHz
- 2 wait states, if 48 MHz < SYSCLK ≤ 72 MHz

  - El buffer FLIFT intenta remediar estos retardos
- Finalmente si bien el procesador ve algunas operaciones como “de un ciclo”, el controlador del bus realiza estas operaciones en más de un ciclo, como accesos no alineados, operaciones de bit-band (lee-modifica-escribe)

Depuración

¿Cuáles son los mecanismos de debug?

1. Acceso al modo de debug Halt deteniendo el procesador y accediendo a los registros de control del debug
   a. Permite instrucciones paso a paso
   b. Permite acceder y modificar registros del procesador
   c. Permite insertar “breakpoints” y “watchpoints” que detienen la ejecución
2. Acceso como maestro a la matriz de interconexionado del bus logrando acceder a cualquier posición de memoria
   a. Permite consultar en tiempo real los valores de las memorias con intrusividad mínima
3. Acceso a periféricos específicos que recolectan y envían información con intrusividad 0.
4. Acceso al modo debug por excepción (introduciendo instrucción) disparando rutinas preprogramadas

Cortex-M3 ROM table components

Address	Component	Value
0xE00FF000	SCS	0xFFF0F003
0xE00FF004	DWT	0xFFF02003[a]
0xE00FF008	FPB	0xFFF03003[b]
0xE00FF00C	ITM	0xFFF01003[c]
0xE00FF010	TPIU	0xFFF41003[d]
0xE00FF014	ETM	0xFFF42003[e]
0xE00FF018	End marker	0x00000000
0xE00FFFCC	SYSTEM ACCESS	0x00000001

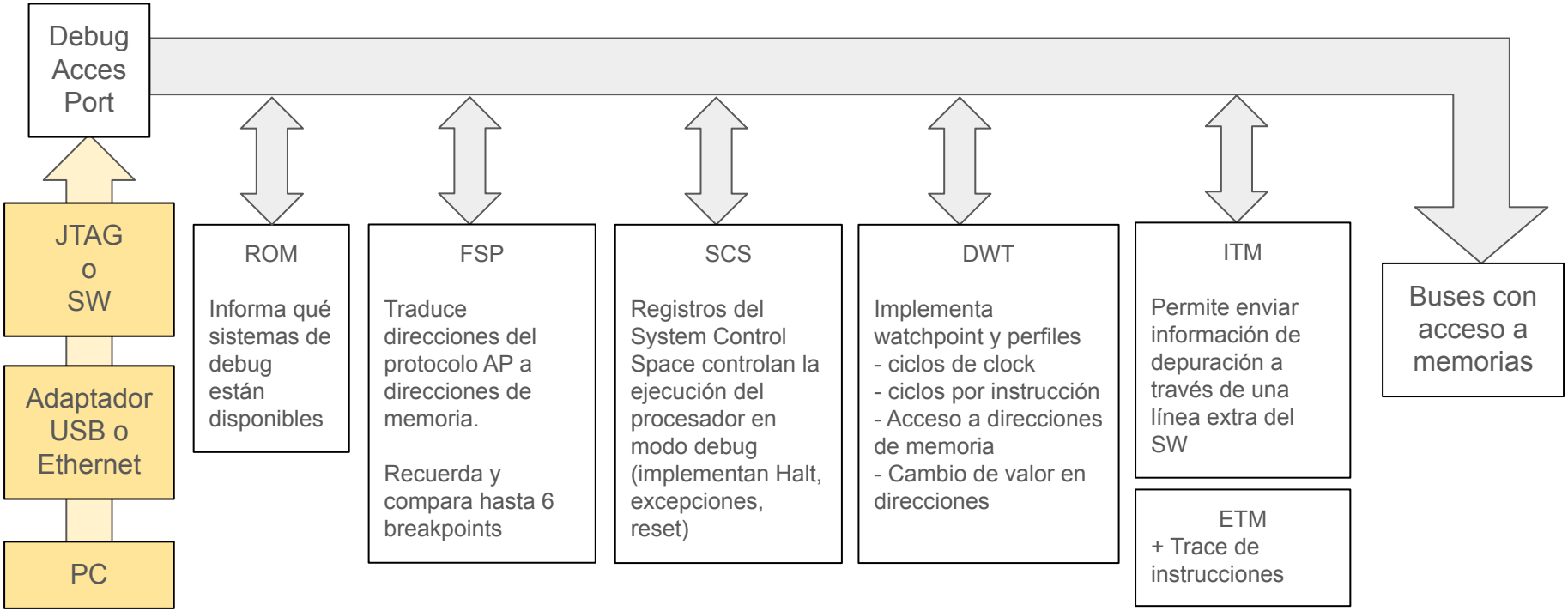
Unidades que proveen facilidades de debug

- System Control Space
- Data Watchpoint and Trace Unit
- Flash Patch and Breakpoint Unit
- Instrumentation Trace Macrocell Unit
- Embedded Trace Macrocell Unit
- Conexión del DAP al AHB

Cortex-M3 ROM table components

Address	Component	Value
0xE00FF000	SCS	0xFFF0F003
0xE00FF004	DWT	0xFFF02003[a]
0xE00FF008	FPB	0xFFF03003[b]
0xE00FF00C	ITM	0xFFF01003[c]
0xE00FF010	TPIU	0xFFF41003[d]
0xE00FF014	ETM	0xFFF42003[e]
0xE00FF018	End marker	0x00000000
0xE00FFFCC	SYSTEM ACCESS	0x00000001

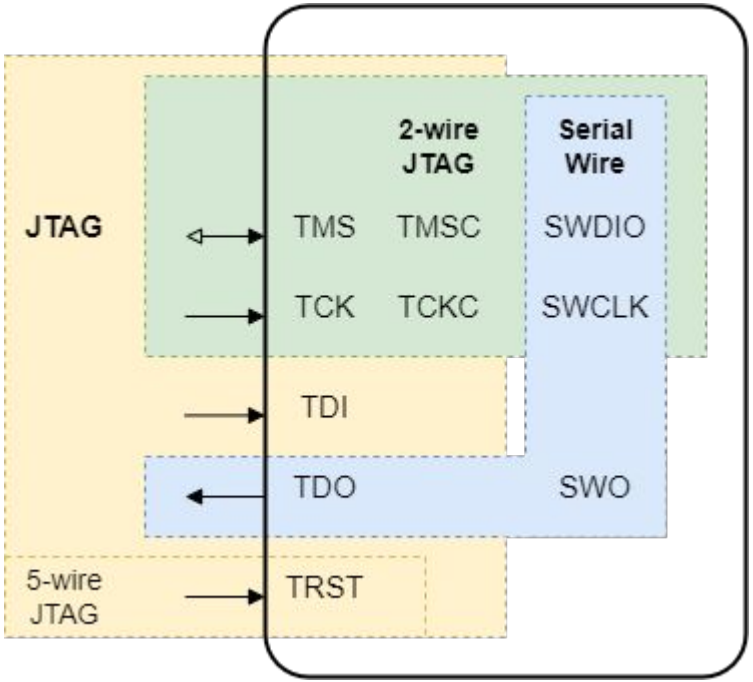
Bloques de debug



Documentación en ARM DDI 0337H y ARMv7-M

Interfaz de debug

- En JTAG
  - TMS es una línea para indicar “modo debug”
  - La comunicación es similar a SPI a través de TCK, TDI, TDO
  - Opcionalmente en el protocolo de 5 líneas una línea TRST permite resetear la lógica del controlador JTAG en el dispositivo
- En Serial Wire
  - Las características de debug se logran a través de las líneas:
    - SWDIO es una línea bidireccional de comunicación
    - SWCLK es el clock
  - Y si se desea usar el Trace, puede hacerse a través de la línea SWO



Interfaz de debug

- Configuración de pines en STM32F103C8
- La interfaz de debug es:
  - una interfaz JTAG estándar que soporta opciones de 5, 4 o 2 pines
  - compartida con una interfaz Serial Wire (de ARM) de 2 pines
- JTAG es un estándar para sistemas embebidos. Casi cualquier SE “moderno” tendrá un puerto JTAG, aunque puede variar el conector. Nació para obtener el “boundary scan” de múltiples dispositivos en daisy-chain y luego se utilizó para distintos objetivos de depuración

Table 219. SWJ debug port pins

SWJ-DP pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Debug assignment	
JTMS/SWDIO	I	JTAG Test Mode Selection	IO	Serial Wire Data Input/Output	PA13
JTCK/SWCLK	I	JTAG Test Clock	I	Serial Wire Clock	PA14
JTDI	I	JTAG Test Data Input	-	-	PA15
JTDO/TRACESWO	O	JTAG Test Data Output	-	TRACESWO if async trace is enabled	PB3
NJTRST	I	JTAG Test nReset	-	-	PB4

RM0008 - Reference manual, ST

Acceso a los bloques

- Utilizamos Breakpoints en el IDE
  - El debugger carga en el FSP las direcciones donde detener la ejecución y el FSP señaliza al SCS para ejecutar el Halt
- Utilizamos Watchpoints en el IDE
  - El debugger carga el DWT y se detiene la ejecución cuando se detecta un cambio de valor
- Utilizamos LiveView en el IDE
  - El debugger programa accesos periódicos a través del bus a las direcciones de memoria de interés para monitorearlas
- Conteo de ciclos
  - Debemos habilitarlo a mano en el SCS y luego acceder al DWT