Arquitectura de Microcontroladores

Carrera de Especialización en Sistemas Embebidos

Clase 2: ARMv7-M Core peripherals

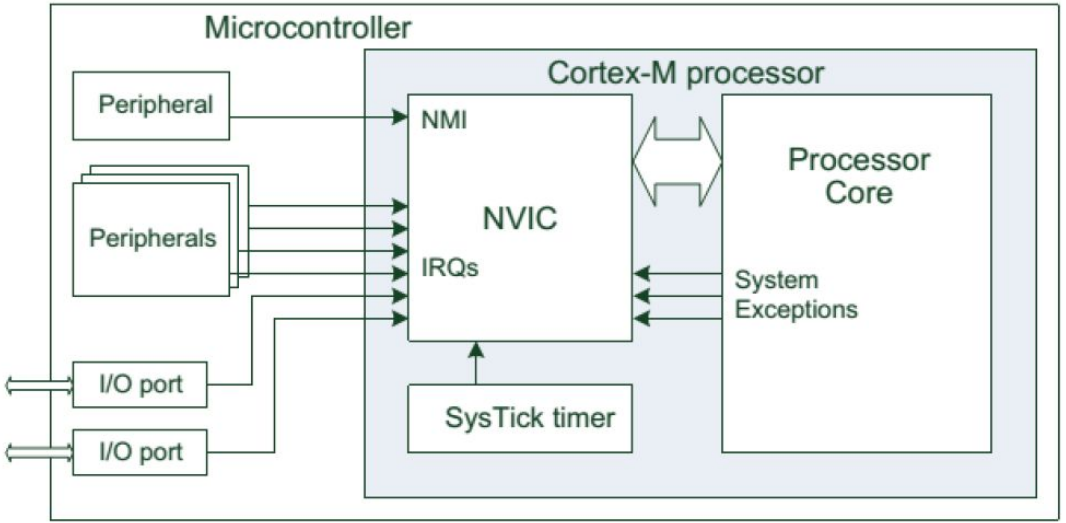
Contenidos de la clase

1. NVIC
2. Excepciones - Mecanismos
3. Floating Point Unit (FPU)
4. Memory Protection Unit (MPU)
5. Systick Timer
6. CMSIS

1. NVIC

Nested Vector Interrupt Controller

Las excepciones de los “core peripherals” tendrán mayor prioridad que las interrupciones externas

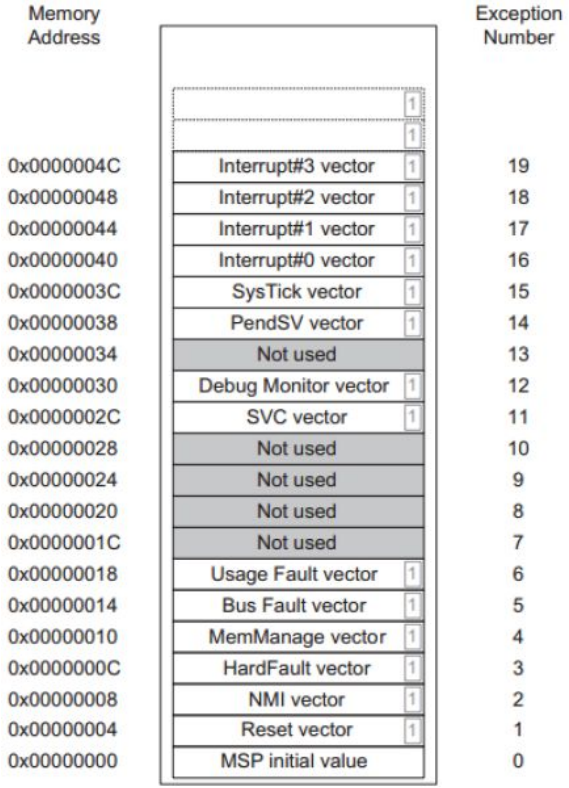


1. NVIC - Excepciones e interrupciones

Table 4.9 Exception Types

Exception Number	CMSIS Interrupt Number	Exception Type	Priority	Function
1	—	Reset	−3 (Highest)	Reset
2	−14	NMI	−2	Non-Maskable interrupt
3	−13	HardFault	−1	All classes of fault, when the corresponding fault handler cannot be activated because it is currently disabled or masked by exception masking
4	−12	MemManage	Settable	Memory Management fault; caused by MPU violation or invalid accesses (such as an instruction fetch from a non-executable region)
5	−11	BusFault	Settable	Error response received from the bus system; caused by an instruction prefetch abort or data access error
6	−10	Usage fault	Settable	Usage fault; typical causes are invalid instructions or invalid state transition attempts (such as trying to switch to ARM state in the Cortex-M3)
7–10	—	—	—	Reserved
11	−5	SVC	Settable	Supervisor Call via SVC instruction
12	−4	Debug monitor	Settable	Debug monitor – for software based debug (often not used)
13	—	—	—	Reserved
14	−2	PendSV	Settable	Pendable request for System Service
15	−1	SYSTICK	Settable	System Tick Timer
16–255	0–239	IRQ	Settable	IRQ input #0–239

1. NVIC - Excepciones e interrupciones



Memory Address		Exception Number
0x0000004C	Interrupt#3 vector	19
0x00000048	Interrupt#2 vector	18
0x00000044	Interrupt#1 vector	17
0x00000040	Interrupt#0 vector	16
0x0000003C	SysTick vector	15
0x00000038	PendSV vector	14
0x00000034	Not used	13
0x00000030	Debug Monitor vector	12
0x0000002C	SVC vector	11
0x00000028	Not used	10
0x00000024	Not used	9
0x00000020	Not used	8
0x0000001C	Not used	7
0x00000018	Usage Fault vector	6
0x00000014	Bus Fault vector	5
0x00000010	MemManage vector	4
0x0000000C	HardFault vector	3
0x00000008	NMI vector	2
0x00000004	Reset vector	1
0x00000000	MSP initial value	0

1. NVIC - Ejemplos

- **NMI: Non-Maskable Interrupts** → reservadas para el control del sistema: fallas en el sistema de clock, inicialización de la flash, etc.
- **HardFault** → Uso indebido de memoria dinámica, incorrecta inicialización de periféricos.
- **MemManage** → Causada por MPU.
- **SVC: SuperVisor Call** → Es una excepción por software que se dispara por la instrucción SVC. Para acceder a funciones del SO o para usar Device Drivers (ejemplo → mandar o recibir datos por UART)

1. NVIC - Ejemplos

- **PendSV** → Excepción llamada en modo Handler. Se usa para hacer el context-switching en un SO.
- **Systick** → Base de tiempo para un SO
- **IRQ** → Configurable para interrupciones externas
  - Se pueden configurar los niveles de prioridad (0 a 239)
  - 0 es la prioridad mayor
  - Mecanismos para atender varias interrupciones…

2. Excepciones

Existen 3 maneras de interactuar con periféricos:

- Por encuesta (Polling)
- Por interrupciones (Interrupt-driven events)
- Por interrupciones con DMA

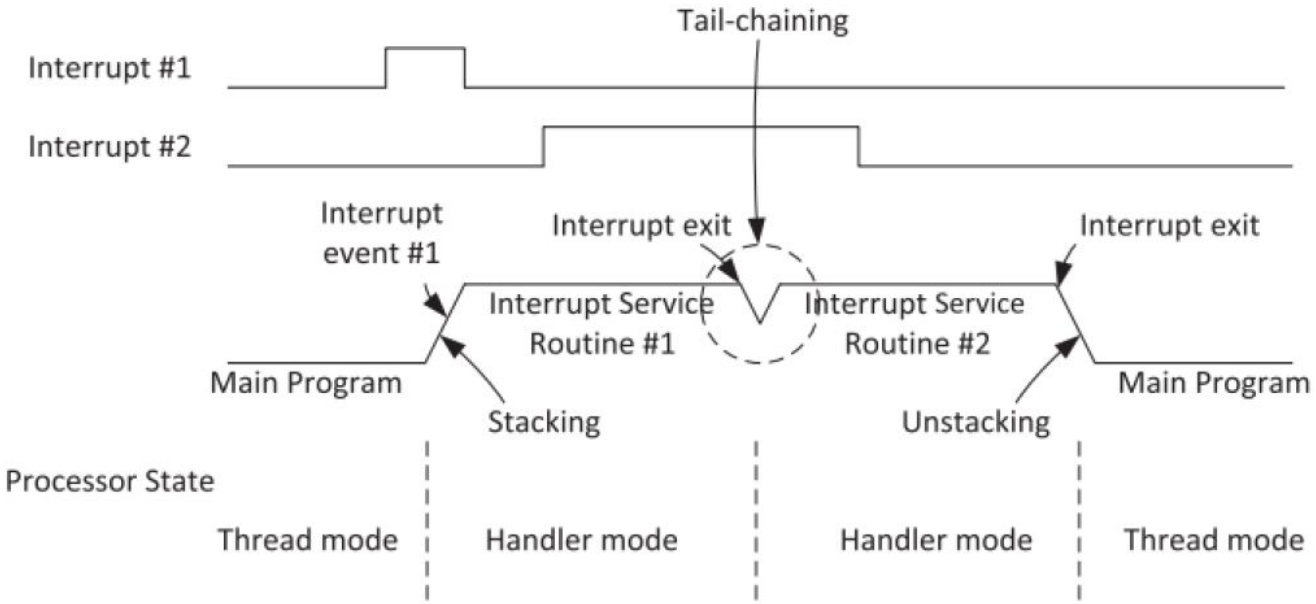
Aumento de complejidad

Los mecanismos que existen en esta arquitectura para atender múltiples interrupciones son 2: **Tail Chaining** y **Late Arrival**

2. Excepciones - Tale Chaining

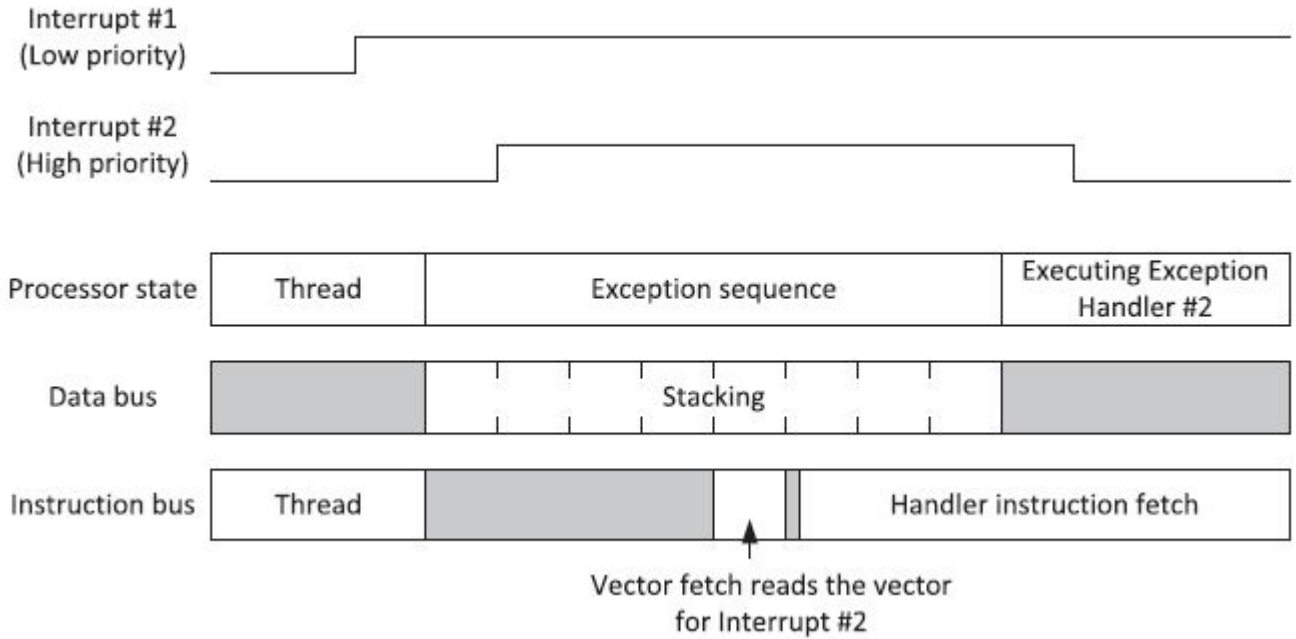
Ocurre cuando en una IRS se provoca otra interrupción de igual o menor prioridad.

No se hace **Context-Switching**



2. Excepciones- Late Arrival

Ocurre cuando **se está haciendo el stacking** para atender una excepción y ocurre otra de mayor prioridad.



¿Qué ocurre cuando se da una excepción de mayor prioridad **después del stacking**?

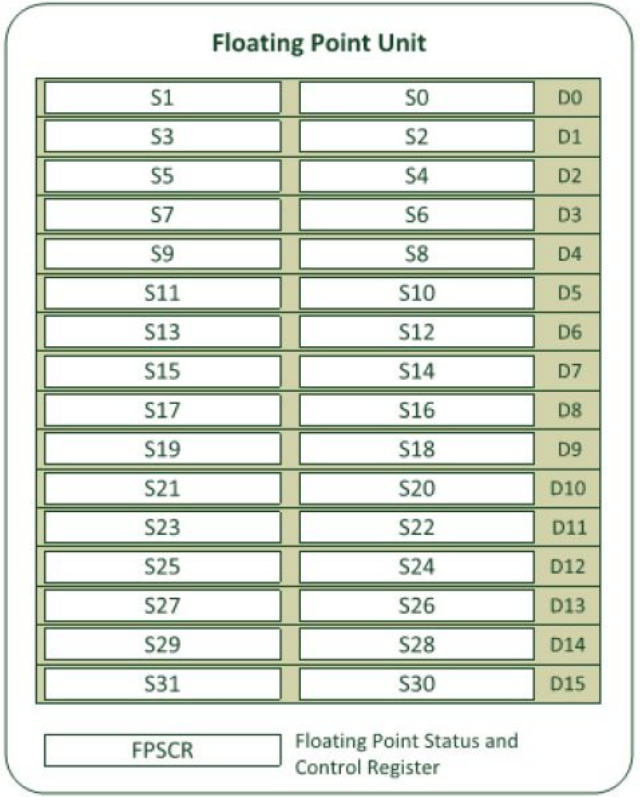
3. Floating Point Unit (FPU)

El Cortex M4F posee una unidad de punto flotante de simple precisión → puede manejar flotantes de 32 bits.

Consta de 32 registros para operar!

Se trata de un **Co-procesador** que se debe activar sino se trabajará mediante software → mucho más lento (igual que sino tendría FPU).

También cuenta con un registro de estado (similar al APSR).



3. Floating Point Unit (FPU)

Formato de un número flotante:

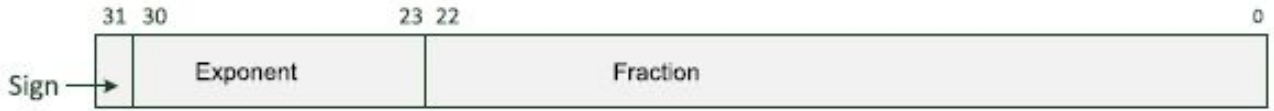


FIGURE 13.1

Single-precision data format

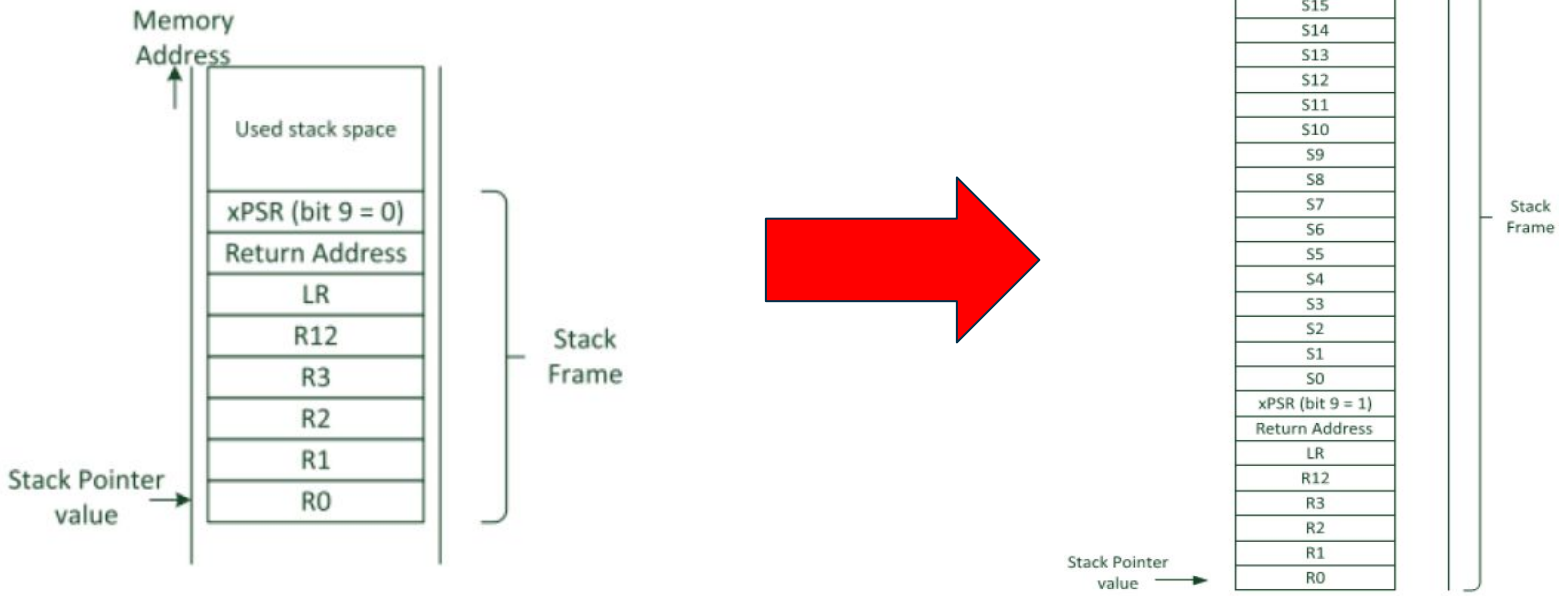
$$\text{Value} = (-1)^{\text{Sign}} \times 2^{(\text{exponent}-127)} \times (1 + (½ * \text{Fracion}[22]) + (¼ * \text{Fraction}[21]) + (1/8 * \text{Fraction}[20]) \ldots (1/(2^{23}) * \text{Fraction}[0]))$$

Conversor de punto flotante on-line (IEEE-754):

https://www.h-schmidt.net/FloatConverter/IEEE754.html

3. Floating Point Unit (FPU) - Stacking

En este caso el stacking involucra más registros…



4. Memory Protection Unit (MPU)

- La función principal de este módulo es proteger el acceso a memoria.
- Se pueden gestionar hasta 8 regiones de memoria
- Otras funciones que posee son:
  - Prevenir que las aplicaciones (tareas) accedan a zonas de memoria de otras aplicaciones o del kernel de un SO
  - Prevenir que las aplicaciones accedan a periféricos sin los permisos adecuados.
  - Evitar que se ejecute código desde zonas no permitidas (ejemplo desde la RAM).

4. Memory Protection Unit (MPU)

En caso de que alguna aplicación intente acceder a una zona de memoria protegida se puede desencadenar una de dos excepciones:

- HardFault
- MemManege (si se habilita)

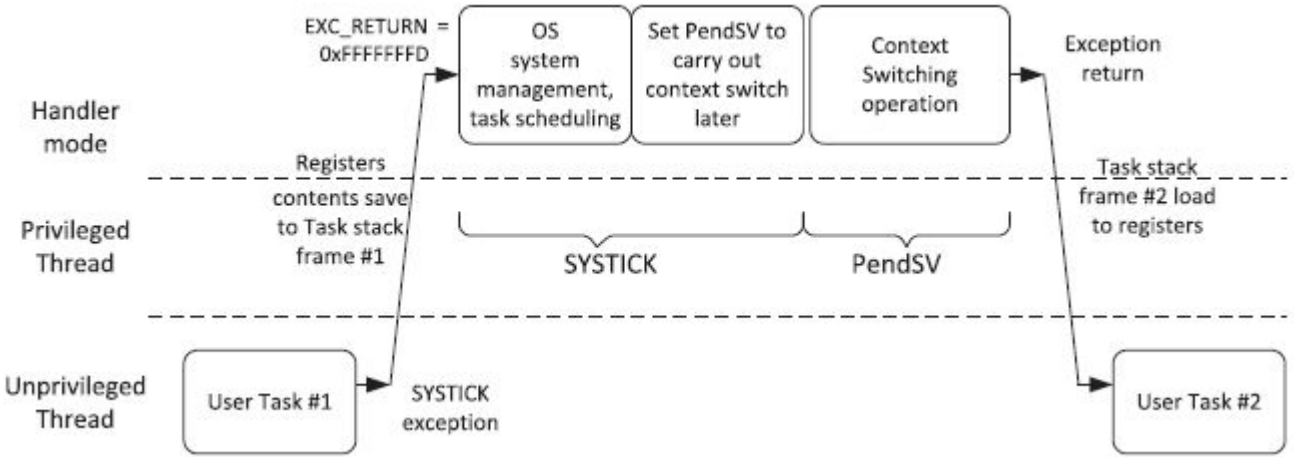
Por defecto la MPU está deshabilitada. Es utilizado por el kernel de un SO para tener control sobre los recursos que se asignan a las diferentes aplicaciones y evitar que el sistema se vuelva inestable

5. Systick Timer

Es un temporizador de 24 bits, que se encuentra integrado en el núcleo del procesador.

- En un RTOS se utiliza como “time-slice” para poder ejecutar el scheduler.
- El hecho de que se encuentre en varias arquitecturas (ARMv6*, ARMv7, ARMv7E, etc.) hace que mejore la portabilidad
- En un RTOS se lo utiliza con la excepción PendSV, lo que permite desacoplar el conteo del Timer con el cambio de contexto del scheduler

5. Systick Timer



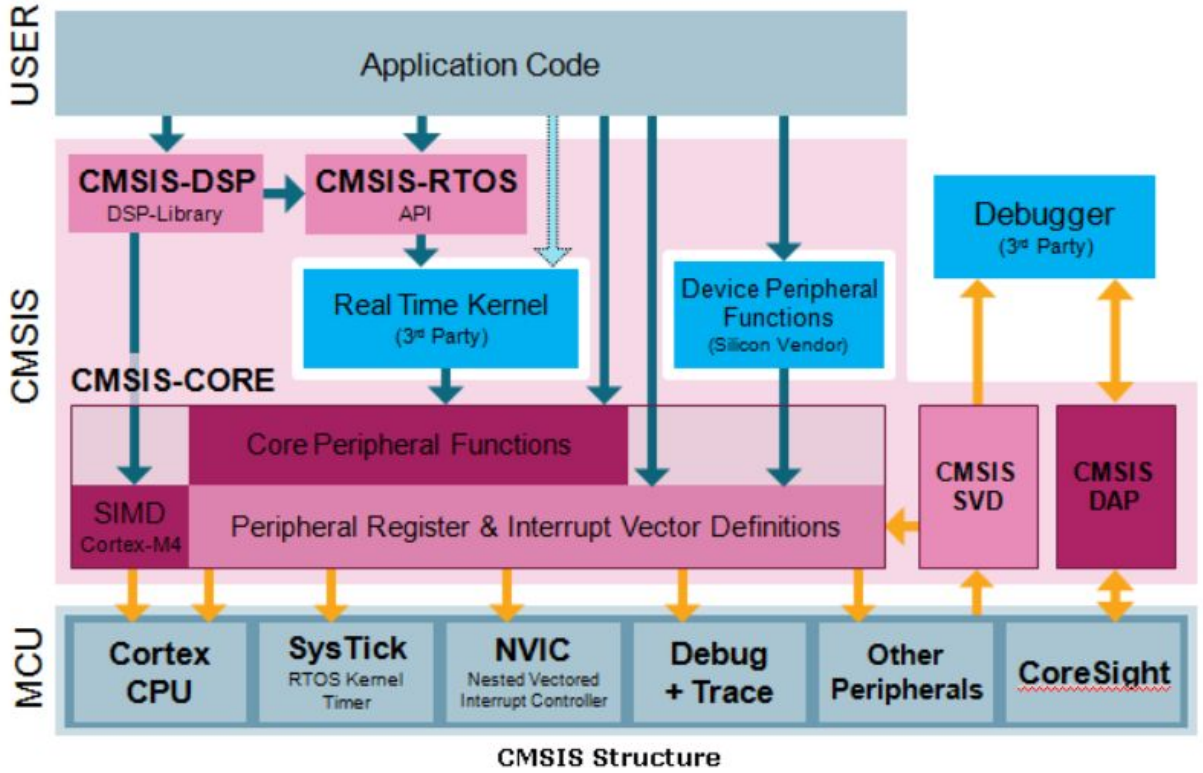
6. CMSIS

CMSIS: *“ARM Cortex Microcontroller Software Interface Standard”*

- Es una capa de abstracción de hardware para los Cortex M independiente del fabricante.
- Es un conjunto de librerías escritas en C, provistas por ARM.
- El objetivo es dar la mayor portabilidad de código posible (entre ARM Cortex M).
- Brinda funciones para interactuar con los “Core peripherals” así como para interactuar con un RTOS.

https://arm-software.github.io/CMSIS_5/Core/html/index.html

6. CMSIS



CMSIS Structure