

Sistemas Basados en Microprocesador

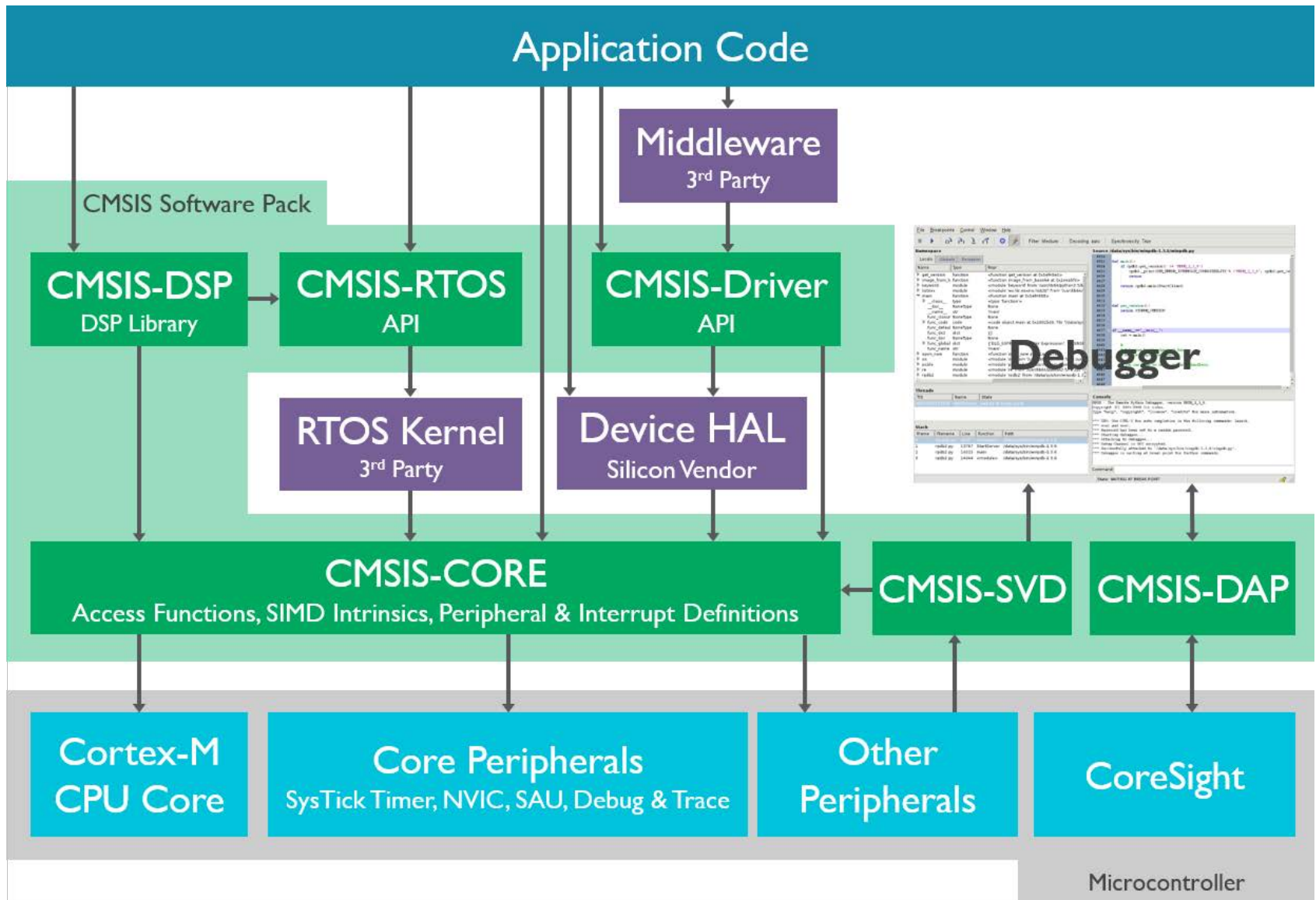
B2 CMSIS DRIVER

CMSIS Driver

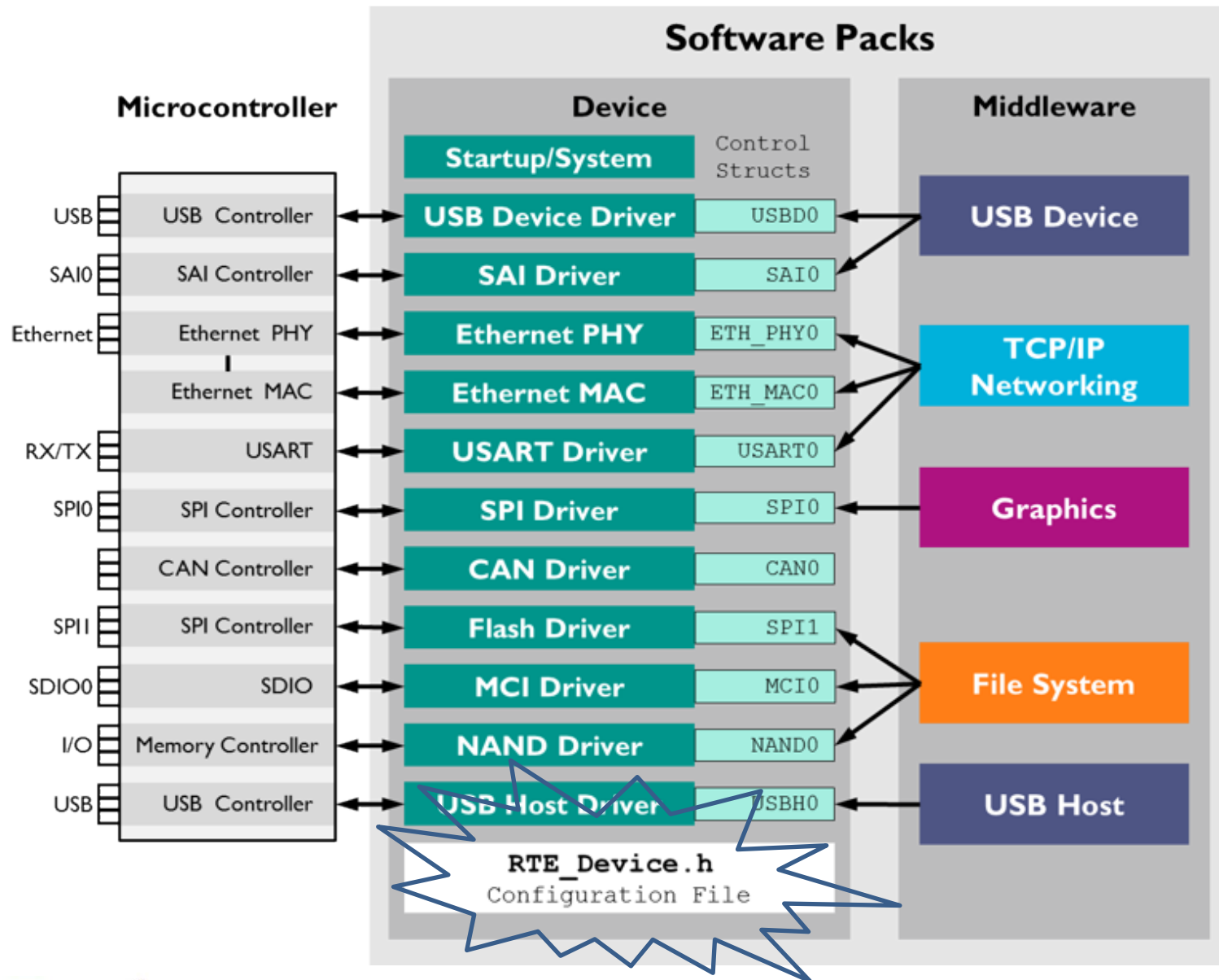
- ¿Qué es un Driver?
 - Elemento software que se encarga de gestionar las funciones realizadas por un hardware
- ¿Qué es una API?
 - API: Application Program Interface
 - Especificación de las funciones que se incluyen en una librería (o en un código) para realizar un conjunto de operaciones
- CMSIS-Driver: Estandarización en el uso y manejo de los periféricos del microcontrolador con una API
- Todos los driver de dispositivos tienen las mismas funciones con objeto de homogeneizar su uso

<http://www.keil.com/pack/doc/CMSIS/Driver/html/index.html>

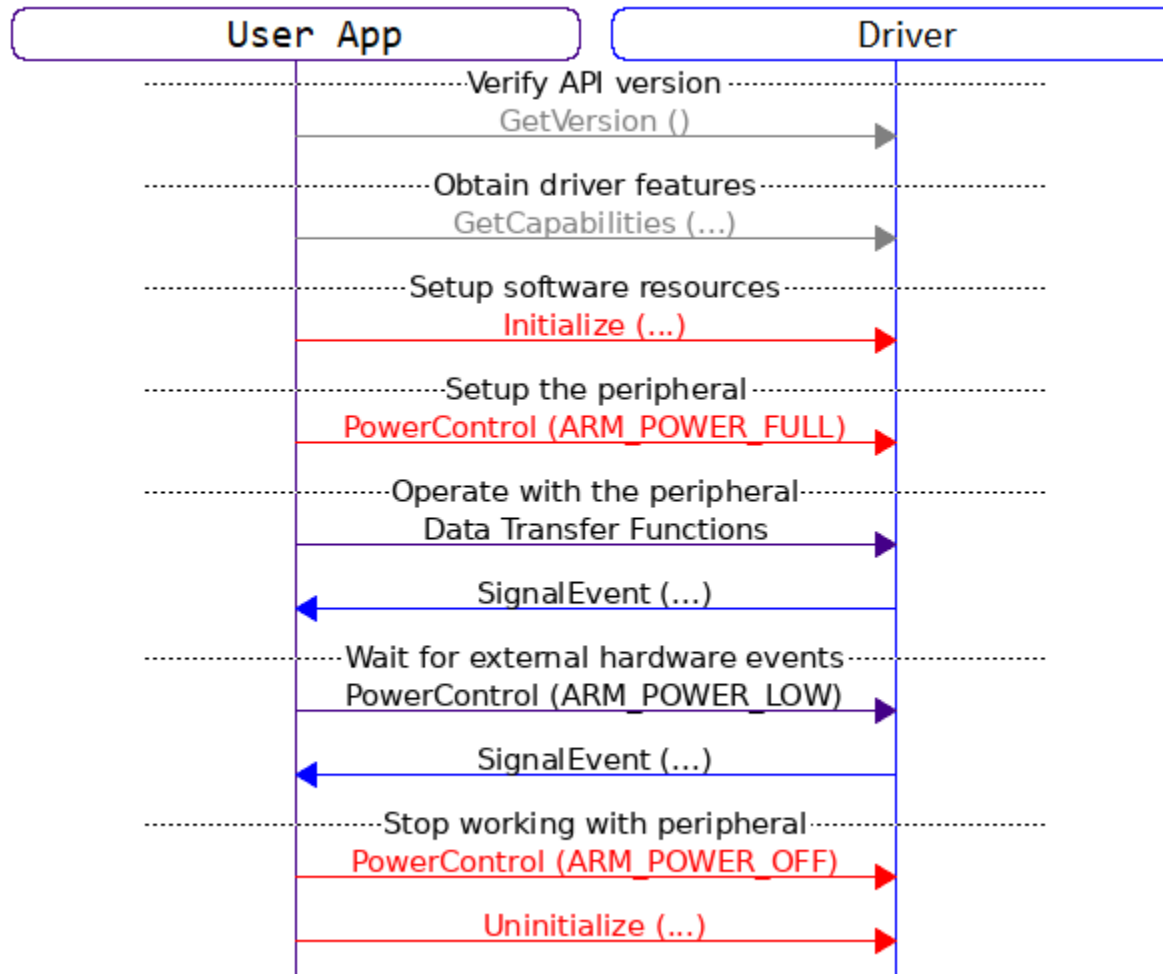
CMSIS



CMSIS-Driver: Periféricos soportados



Secuencia de uso de un driver



Ejemplo de secuencia de operaciones

1. SPIdrv->Initialize
2. SPIdrv->PowerControl (ON)
3. SPIdrv->Control (Parámetros)
4. SPIdrv->Send
5. SPIdrv->Receive (opcional)
6. SPIdrv->PowerControl (OFF)
7. SPIdrv->Uninitialize

API estandarizada por CMSIS-Driver

Función	Significado
GetVersion:	Devuelve la información de la versión del driver
GetCapabilities	Devuelve información acerca de las opciones soportadas por el driver
Initialize	<p>Función que configura los pines a utilizar por el periférico y su CLK, si es necesario. Los pines se configuran usando PULL-UP.</p> <p>Esta función debe ser la primera en ejecutarse (a excepción de las dos anteriores)</p> <p>A esta función se le puede pasar como parámetro el nombre de una función que se ejecutará cuando se produzca un evento asociado con el periférico (callback)</p>
GetStatus	Devuelve información acerca del estado del periférico. Se suele consultar tras una operación de transferencia de información.

API estandarizada por CMSIS-Driver

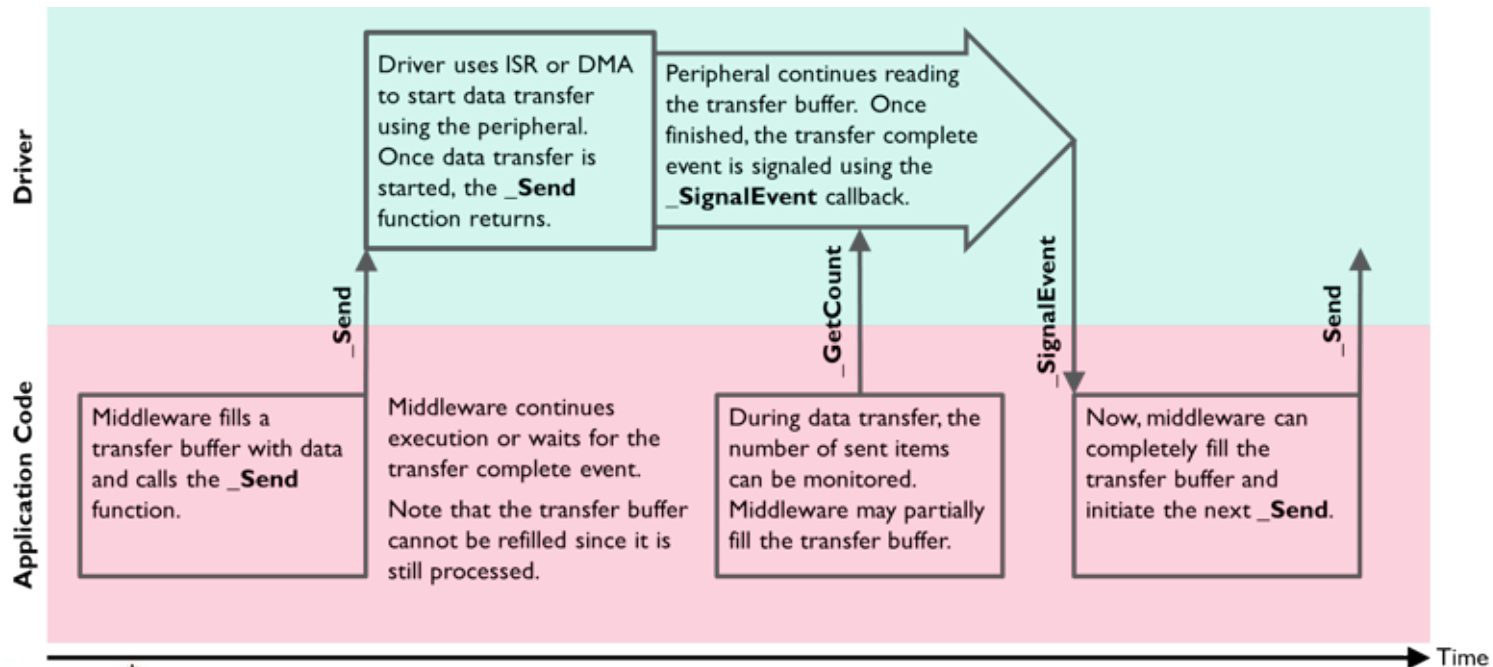
Función	Significado
PowerControl	<p>Función que configura el funcionamiento en términos de consumo del periférico. Hay tres posibles valores para el parámetro de entrada:</p> <ul style="list-style-type: none">•ARM_POWER_FULL: Peripheral is turned on and fully operational. The driver initializes the peripheral registers, interrupts, and (optionally) DMA.•ARM_POWER_LOW: (optional) Peripheral is in low power mode and partially operational; usually, it can detect external events and wake-up.•ARM_POWER_OFF: Peripheral is turned off and not operational (pending operations are terminated). This is the state after device reset
Uninitialize	Última función a utilizar que libera los recursos utilizados
Control	Función que permite configurar el funcionamiento del periférico con valores (<i>define's</i>) que conforman la palabra de configuración

Transferencia en CMSIS-Driver

Función	Significado
Send	Envía datos al periférico
Receive	Recibe datos del periférico
Transfer	Operación combinada de escritura/lectura

Transferencia de información no bloqueante

Utilización del callback/eventos asociados a la instancia del driver



Transferencia en CMSIS-Driver

Función	Significado
Send	Envía datos al periférico
Receive	Recibe datos del periférico
Transfer	Operación combinada de escritura/lectura

Transferencia de información bloqueante:

```
stat = SPIDrv->Send(&buf, sizeof(buf));  
  
do  
{  
    stat = SPIDrv->GetStatus();  
}  
while (stat.busy);
```



Añadiendo Drivers CMSIS (RTE)

Manage Run-Time Environment
Select Software Components for the project

Manage Run-Time Environment

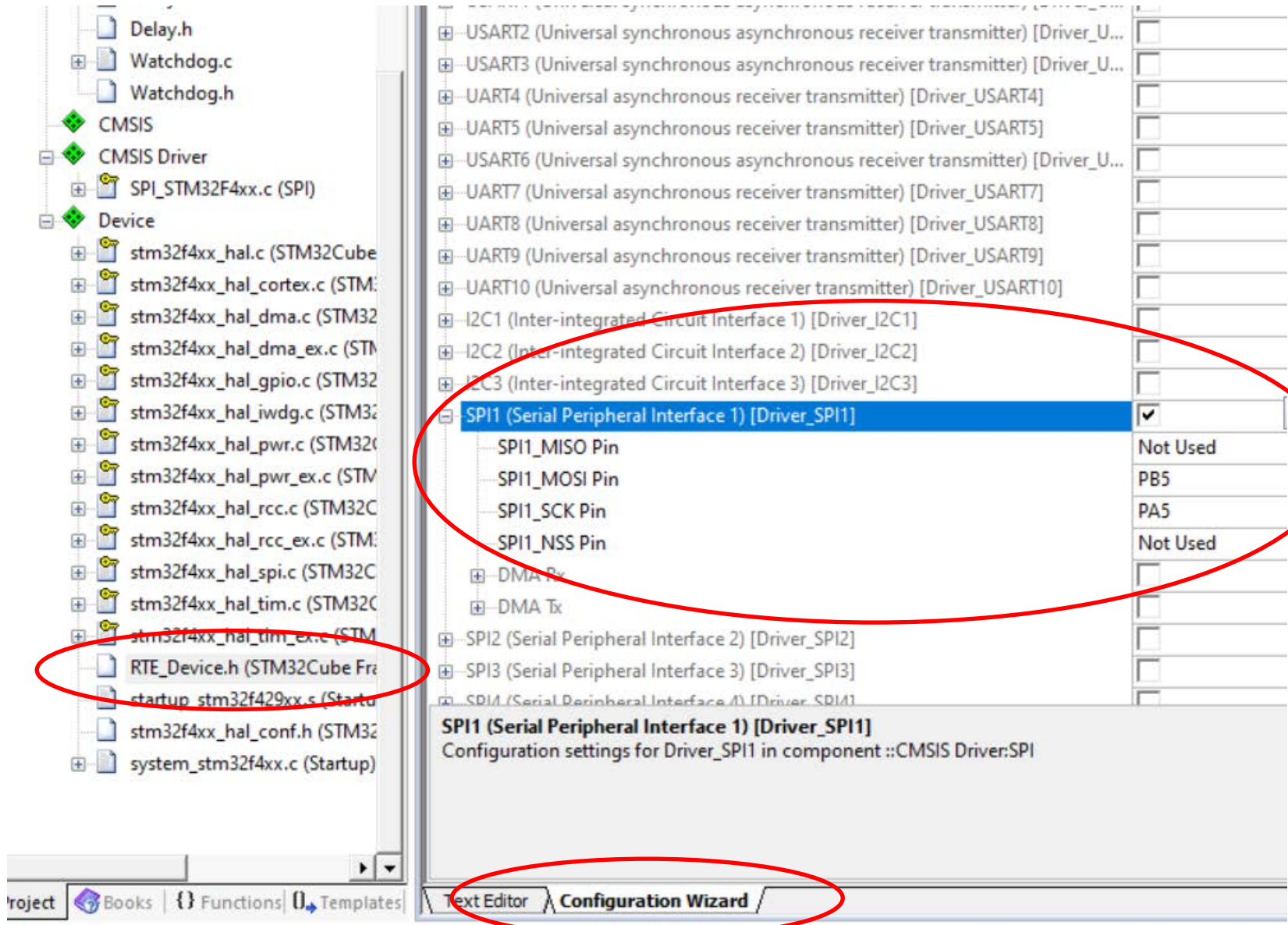
Software Component	Sel.	Variant	Version	Description
Board Support		NUCLEO-F429ZI	1.0.0	STMicroelectronics NUCLEO-F429ZI Development Board
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specificatic
CAN (API)			1.3.0	CAN Driver API for Cortex-M
Ethernet (API)			2.2.0	Ethernet MAC and PHY Driver API for Cortex-M
Ethernet MAC (API)			2.2.0	Ethernet MAC Driver API for Cortex-M
Ethernet PHY (API)			2.2.0	Ethernet PHY Driver API for Cortex-M
Flash (API)			2.3.0	Flash Driver API for Cortex-M
I2C (API)			2.4.0	I2C Driver API for Cortex-M
MCI (API)			2.4.0	MCI Driver API for Cortex-M
NAND (API)			2.4.0	NAND Flash Driver API for Cortex-M
SAI (API)			1.2.0	SAI Driver API for Cortex-M
SPI (API)			2.3.0	SPI Driver API for Cortex-M
Custom	<input type="checkbox"/>		1.0.0	Access to #include Driver_SPI.h file and code template for cu:
Multi-Slave	<input type="checkbox"/>		1.0.1	SPI Master to Multi-Slave Driver Wrapper
SPI	<input checked="" type="checkbox"/>		2.15	SPI Driver for STM32F4 Series
USART (API)			2.4.0	USART Driver API for Cortex-M
USB Device (API)			2.3.0	USB Device Driver API for Cortex-M
USB Host (API)			2.3.0	USB Host Driver API for Cortex-M
VIO (API)			0.1.0	Virtual I/O

Validation Output

Description

Resolve Select Packs Details OK Cancel Help

Configuración del sistema con RTE_Device.h



¿Cómo se maneja el driver desde una aplicación en C?

- CMSIS define esta estructura de datos. Para acceder a ella se deben añadir estas sentencias en el programa:
 - extern [ARM_DRIVER_SPI](#) Driver_SPI<n>;
 - [ARM_DRIVER_SPI](#)* SPIDrv = &Driver_SPI<n>;

```
typedef struct _ARM_DRIVER_SPI {  
    ARM\_DRIVER\_VERSION (*GetVersion) (void);  
    ARM\_SPI\_CAPABILITIES (*GetCapabilities) (void);  
    int32_t (*Initialize) (ARM\_SPI\_SignalEvent\_t cb_event);  
    int32_t (*Uninitialize) (void);  
    int32_t (*PowerControl) (ARM\_POWER\_STATE state);  
    int32_t (*Send) (const void *data, uint32_t num);  
    int32_t (*Receive) (void *data, uint32_t num);  
    int32_t (*Transfer) (const void *data_out, void  
        *data_in, uint32_t num);  
    uint32_t (*GetDataCount) (void);  
    int32_t (*Control) (uint32_t control, uint32_t arg);  
    ARM\_SPI\_STATUS (*GetStatus) (void);  
} const ARM\_DRIVER\_SPI;
```

n debe ser el número del
interface seleccionado en
el fichero de configuración
RTE_device.h