

Introducción al uso de memorias SD.

Agenda

- Memorias MMC y SD
 - Características Físicas
 - Tipos y Velocidad.
 - Interfaz
 - Memoria SD como dispositivo de bloques.
- Sistemas de archivos.
 - Necesidad de sistema de archivos.
 - Ejemplos de sistemas de archivos conocidos.
- Sistemas de archivos para Embebidos..
 - Uso de FatFs y Petit FS
 - Archivos y Configuración.

Memorias SD/MMC.

- Memorias no volátiles de gran capacidad y bajo costo.
- Son dispositivos de bloques (usualmente de 512 bytes)
- La interfaz con sistemas embebidos suele ser a través de SPI.

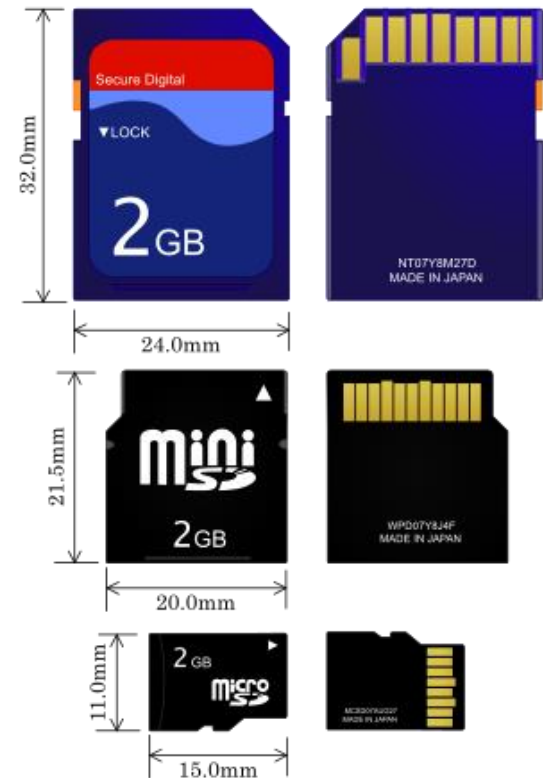


Memorias SD/MMC

- El estándar original fue el de las memorias MMC (MultiMedia Card) que definió la interfaz eléctrica, de datos y mecánica presentado en 1997.
- En 1999 aparece el estándar SD (Secure Digital) como una mejora del MMC.
- El estándar MMC es un estándar abierto (pago) desarrollado por JEDEC (Joint Electron Device Engineering Council) mientras que el estándar SD es propietario y cerrado.
- Todas las memorias SD soportan el modo MMC (sin seguridad) de funcionamiento.

Memorias SD/MMC. Interfaz física

- Las memorias SD se encuentran disponibles en tres formatos físicos:
 - ☐ SD estándar
 - ☐ Mini SD
 - ☐ Micro SD
- Los adaptadores entre tamaños, generalmente son sólo mecánicos



Memorias SD. Familias tecnológicas.

- **SDSC.** Capacidad Estándar. Son las memorias que estaban entre 128MB y 2GB con FAT16 como formato estándar.
- **SDHC.** Alta Capacidad. Son las que van entre 4GB y 32GB con FAT32 por defecto basadas en el estándar SD 2.0. Soporta dispositivos SDSC.
- **SDXC.** Capacidad Extendida. Van entre 64GB y 2TB con exFAT. Basadas en el estándar 3.0. Soportan dispositivos SDHC, SDSC.

Memorias SD. Velocidades.

- Las especificaciones SD 3.0 y SD 4.0 definen los dispositivos:
 - **UHS-I** (SD 3.0) que permiten velocidades de transferencia de datos máximas de 50Mb/s y 104Mb/s (UHS-50 y UHS-104). No hay problema de compatibilidad entre un dispositivo UHS y otro que no lo sea.
 - **UHS-II** (SD 4.0) que permiten velocidades de transferencias de datos máximas de 156Mb/s y 312Mb/s. No hay problema de compatibilidad entre un dispositivo UHS-II en un dispositivo UHS-I o no UHS.

Clase de velocidad.

- La clase de velocidad (Speed Class) es la velocidad mínima garantizada para una memoria SD.










Speed Class (SD Bus)

Class	Minimum Speed
2	2MB/s
4	4MB/s
6	6MB/s
8	8MB/s
10	10MB/s

UHS Speed Class (UHS Bus)

UHS Class	Minimum Speed
1	10MB/s
3	30MB/s

Memorias SD. Compatibilidad.

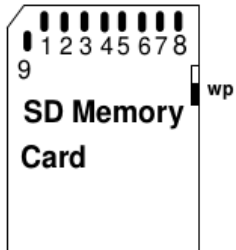
Host device (ex. cameras, video recorders, phones, readers, etc.)	Memory cards supported		
 SDXC host device	 SDXC card 64GB - 2TB	 SDHC card 4GB - 32GB	 SD card 2GB and less
 SDHC host device	 SDHC card 4GB - 32GB	 SD card 2GB and less	
 SD host device	 SD card 2GB and less		

© images by sdcard.org

Memorias SD. Interfaz Eléctrica

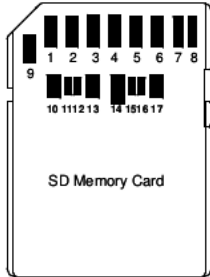
- Las memorias SD tienen 3 modos de funcionamiento:
 - SPI.
 - Un bit SD.
 - Modo SD (4 bits).
- En sistemas con microcontroladores se suele utilizar el modo SD ya que se suele disponer de este controlador por hardware y la tasa de transferencia de datos está limitada por el controlador.

Memoria SD. Interfaz eléctrica.



Pin #	SD Mode			SPI Mode		
	Name	Type ¹	Description	Name	Type ¹	Description
1	CD/DAT3 ²	I/O/PP ³	Card Detect/ Data Line [Bit 3]	CS	I ³	Chip Select (neg true)
2	CMD	I/O/PP	Command/Response	DI	I	Data In
3	VSS1	S	Supply voltage ground	VSS	S	Supply voltage ground
4	VDD	S	Supply voltage	VDD	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	VSS2	S	Supply voltage ground	VSS2	S	Supply voltage ground
7	DAT0	I/O/PP	Data Line [Bit 0]	DO	O/PP	Data Out
8	DAT1 ⁴	I/O/PP	Data Line [Bit 1]	RSV		
9	DAT2 ⁵	I/O/PP	Data Line [Bit 2]	RSV		

Memoria SD UHS-II. Interfaz eléctrica.



Pin #	Name	Type	Description
4	VDD1	Supply voltage	2.7V to 3.6V
7	RCLK+	Differential Signaling: Input	Clock Input
8	RCLK-	Differential Signaling: Input	Clock Input
10	VSS3	Ground	
11	D0+	Differential Signaling: Input (FD) / Bidirectional (HD)	Input in default
12	D0-	Differential Signaling: Input (FD) / Bidirectional (HD)	Input in default
13	VSS4	Ground	
14	VDD2	Supply Voltage 2	1.70V to 1.95V
15	D1-	Differential Signaling: Output (FD) / Bidirectional (HD)	Output in default
16	D1+	Differential Signaling: Output (FD) / Bidirectional (HD)	Output in default
17	VSS5	Ground	

Memorias SD. Tensiones y frecuencia de reloj

Bus Speed Mode*1	Max. Bus Speed [MB/s]	Max. Clock Frequency [MHz]	Signal Voltage [V]	Max. Power*2 [W]		
				SDSC*3	SDHC*4	SDXC*5
FD156*8	156	52	0.4	-	1.80*6	1.80*6
HD312*8	312	52	0.4	-	1.80*6	1.80*6
FD624*8	624	52	0.4	-	1.80*6	1.80*6
SDR104	104	208	1.8	-	1.80*6	1.80*6
SDR50	50	100	1.8	-	1.44	1.44
DDR50	50	50	1.8	-	1.44	1.44
SDR25	25	50	1.8	-	0.72	0.72
SDR12	12.5	25	1.8	-	0.36	0.36/0.54*7
High Speed	25	50	3.3	0.72	0.72	0.72
Default Speed	12.5	25	3.3	0.36	0.36	0.36/0.54*7

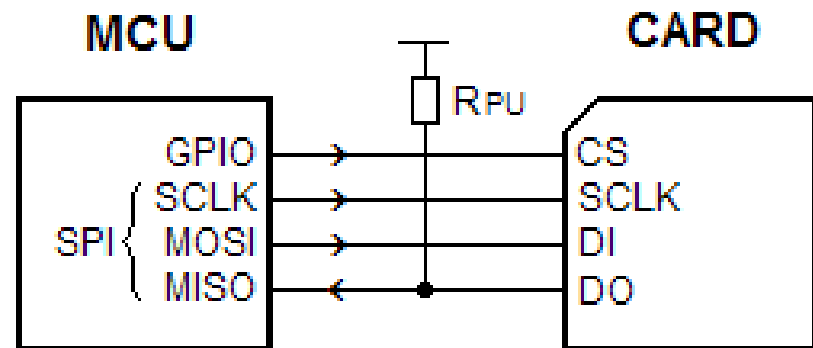
UHS-II I/F

SD I/F

Conectar SD/MMC a microcontrolador

- Para conectar una memoria SD/MMC a un microcontrolador vamos a considerar:

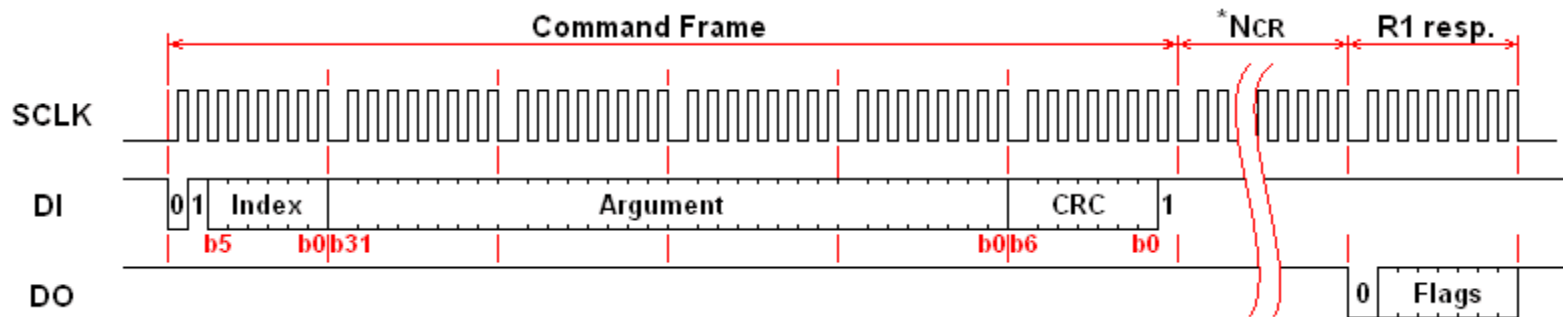
- ☐ Interfaz SPI.
- ☐ Tensiones de 3.3V
- ☐ Frecuencia de reloj máxima de 25MHz



- ☐ RPU es del orden de los 10K Ω
- ☐ El SPI se debe configurar en modo 0 o 3

SD/MMC. Comunicación.

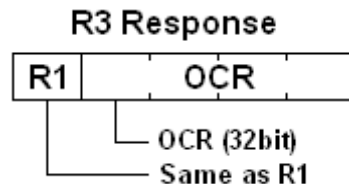
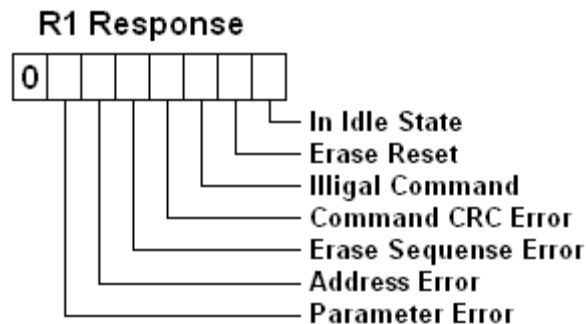
- La comunicación entre el microcontrolador y la memoria está separada en bytes y basada en comandos de varios bytes.



SD/MMC. Comandos.

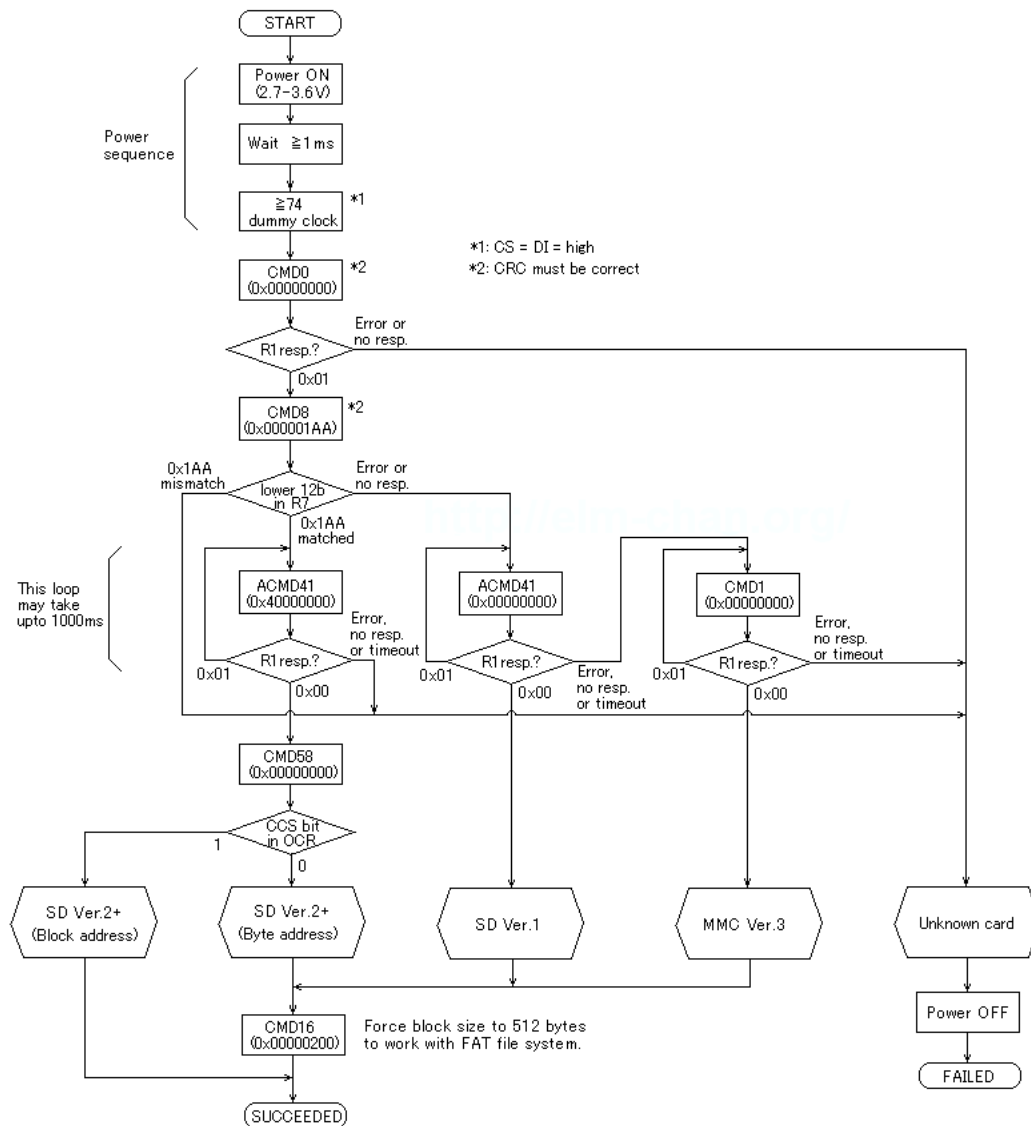
Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.
<p>*1:ACMD<n> means a command sequence of CMD55-CMD<n>.</p> <p>*2: Rsv(0)[31], HCS[30], Rsv(0)[29:0]</p> <p>*3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]</p>					

SD/MMC. Respuestas.



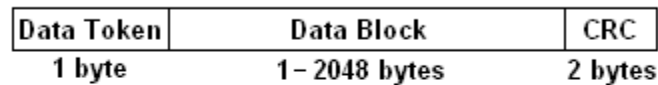
- La mayoría de los comandos responden con R1 (8bits).
- R3 o R7 (32 bits) es la respuesta de CMD58 y CMD8.
- Otros comandos responden con R1b que es igual a R1 pero baja la línea DO hasta que se termine el comando cuando DO sube.

SDC/MMC initialization flow (SPI mode)



SD/MMC Transferencia de datos

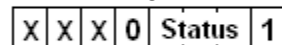
Data Packet



Data Token

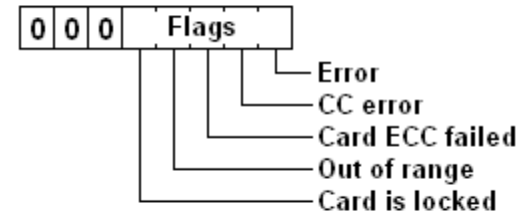


Data Response

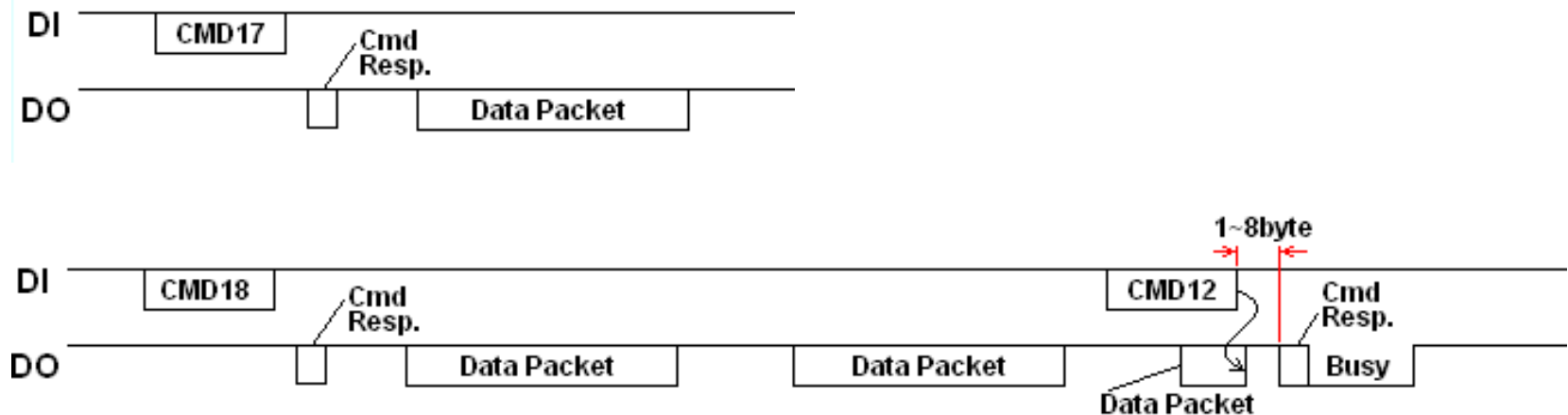


- 0 1 0 — Data accepted
- 1 0 1 — Data rejected due to a CRC error
- 1 1 0 — Data rejected due to a write error

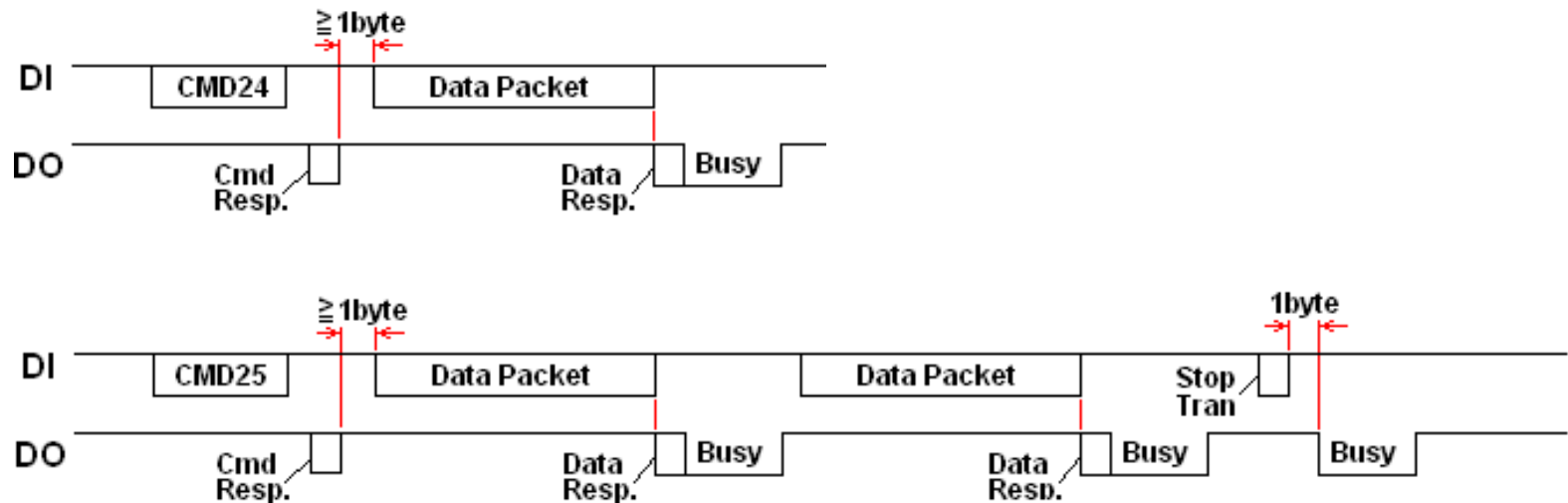
Error Token



SD/MMC. Leer bloques

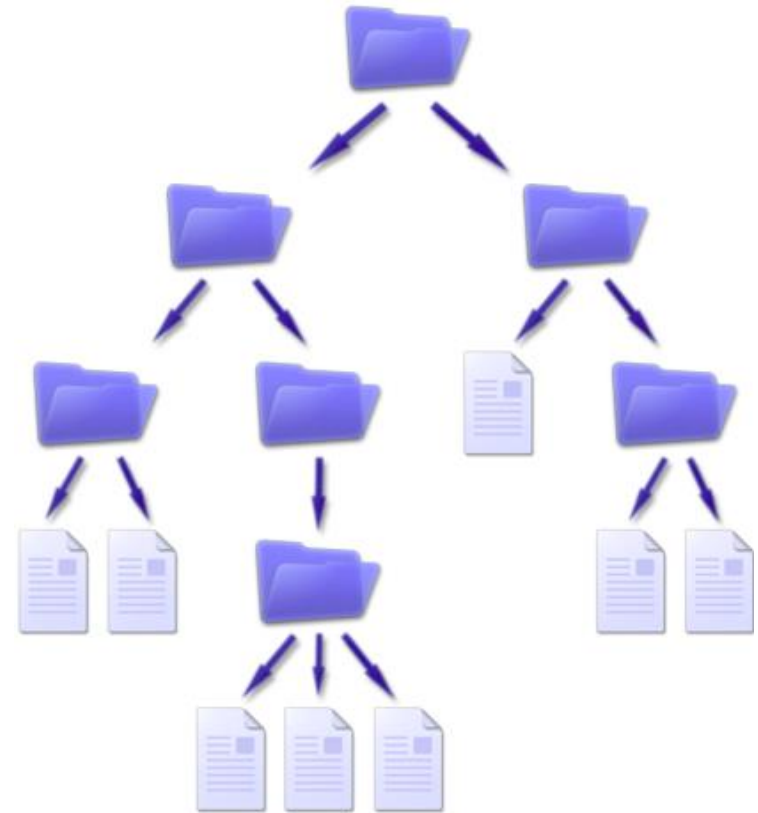


SD/MMC. Escribir bloques



Sistemas de archivos

- Un sistema de archivos es una estructura jerárquica que permite trabajar con diferentes unidades de información (archivos).
- El uso para sistemas embebidos está fuertemente ligado a las memorias SD/MMC para **estandarizar** la información almacenada en la misma.



Sistemas de archivos.

- **EXT4.** Es el sistema de archivos por defecto de Linux. Es un sistema documentado y estable. Es complejo para implementar.
- **F2FS (*Flash-Friendly File System*).** Es un sistema de archivos pensado para memorias de estado sólido. Tiene poco soporte y es relativamente nuevo.
- **FAT. (*File Allocation Table*).** Es un sistema de archivos muy distribuido y es casi un estándar de facto en todas sus versiones (FAT12, FAT16 y FAT32). Es poco fiable, pero muy sencillo de implementar.

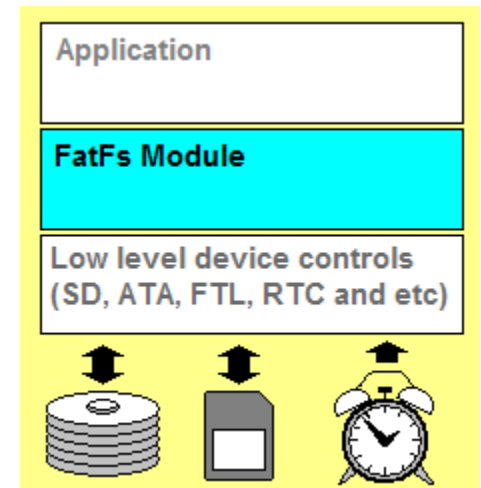
Bibliotecas FatFs para microcontroladores

- ***FatFs***. Es una biblioteca portada a muchos microcontroladores, muy configurable y con mucha funcionalidad. Está escrita en C. Utiliza cerca de 5KB de código y unos 560 bytes de RAM por archivo más otros 600 bytes de RAM por disco. Está orientada a procesadores como los Cortex-M3.
- ***Petit FAT***. Es una biblioteca escrita en C con funcionalidad limitada (un solo archivo abierto por vez en el raíz). Utiliza unos 2KB de código y cerca de 100bytes de RAM para el archivo. Orientada a procesadores de 8-bits pequeños.

FatFs. Requerimientos

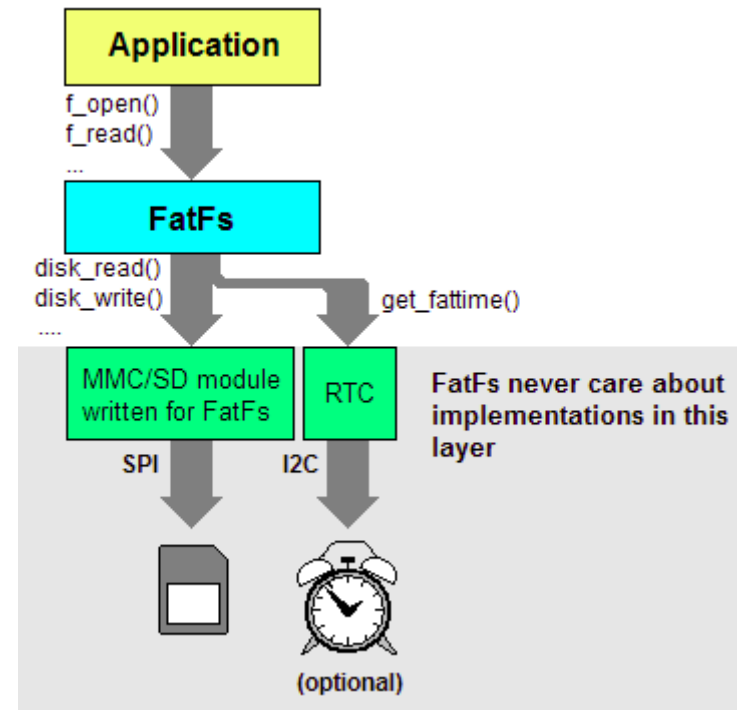
- FatFs provee todo el código para el manejo de un sistema de archivos tipo FAT. No provee las funciones para el acceso al hardware. Para que funcione se deben implementar:

- ☐ ***disk_status***
- ☐ ***disk_initialize***
- ☐ ***disk_read***
- ☐ ***disk_write***
- ☐ ***disk_ioctl***
- ☐ ***get_fattime***

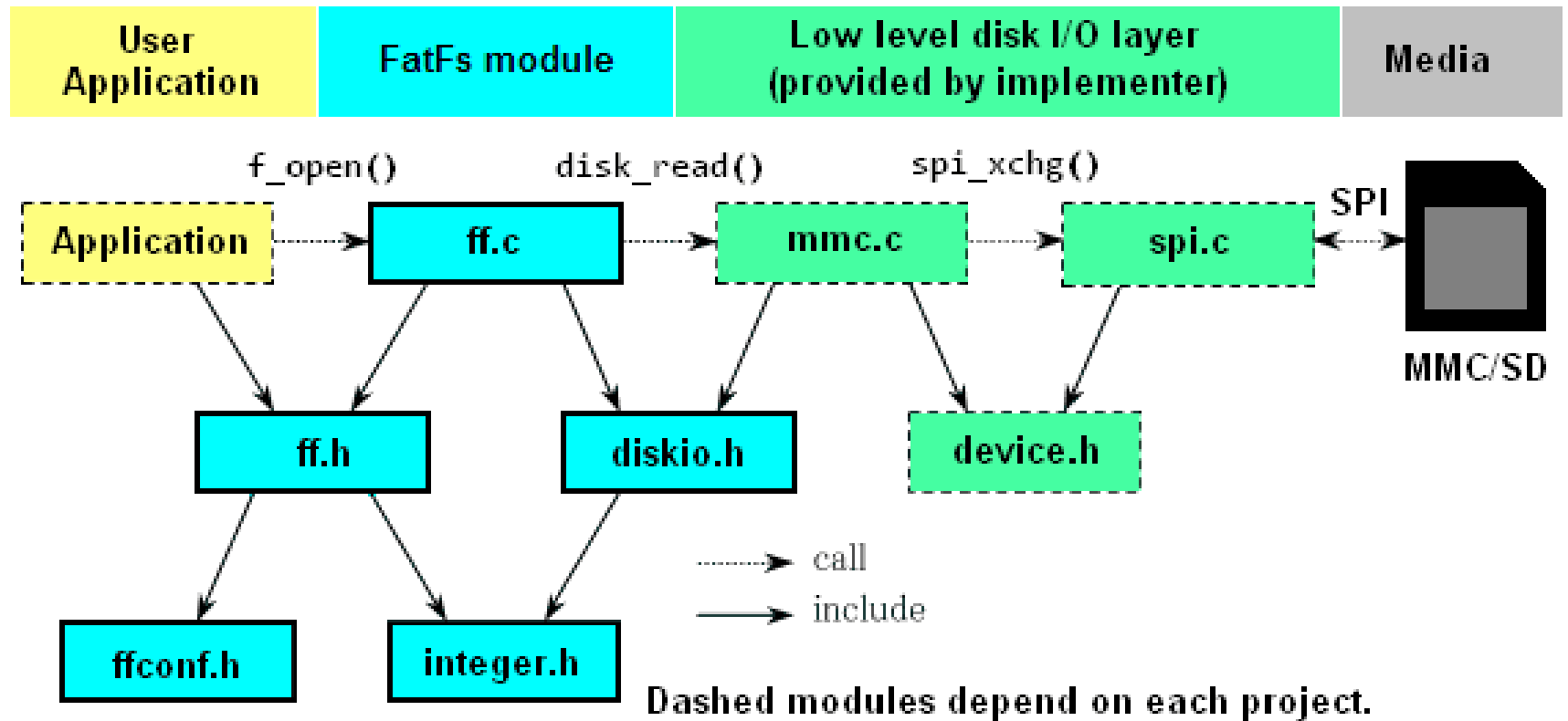


FatFs. Funcionalidad.

- FatFs va a proveer funciones para:
 - Manejo de archivos (lectura, escritura a nivel de bloques y cadenas).
 - Manejo de archivos y directorios (recorrerlos, crearlos y destruirlos, por ejemplo)



FatFs. Organización de archivos



FatFs. Ejemplo de uso

```
int main(void)
{
    UINT nbytes;

    initHardware();

    /* Give a work area to the default drive */
    if (f_mount(&fs, "", 0) != FR_OK) {
        /* If this fails, it means that the function could
         * not register a file system object.
         * Check whether the SD card is correctly connected */
    }

    /* Create/open a file, then write a string and close it */
    if (f_open(&fp, FILENAME, FA_WRITE | FA_CREATE_ALWAYS) == FR_OK) {
        f_write(&fp, "It works!\r\n", 11, &nbytes);

        f_close(&fp);

        if (nbytes == 11) {
            /* Toggle a LED if the write operation was successful */
            Board_LED_Toggle(0);
        }
    }

    while (1) {
        __WFI();
    }

    return 0;
}
```

FatFs. Secciones críticas.

Figure 4. Long critical section

```
f_mount(...);  
  
f_open(...);          //Create file  
  
// any procedure  
do {  
    t = get_adc(...);  
  
    // any procedure  
  
    f_write(...);      // write file  
  
    delay_second(1);  
  
} while (...);  
  
// any procedure  
  
f_close(...);         // close file
```

```
f_mkdir(...);
```

```
f_rename(...);
```

```
f_unlink(...);
```

Figure 5. Minimized critical section

```
f_mount(...);  
  
f_open(...);          //Create file  
f_sync(...);  
  
// any procedure  
do {  
    t = get_adc(...);  
  
    // any procedure  
  
    f_write(...);      // write file  
    f_sync(...);  
    delay_second(1);  
  
} while (...);  
  
// any procedure  
  
f_close(...);         // close file
```

```
f_mkdir(...);
```

```
f_rename(...);
```

```
f_unlink(...);
```

Referencias

- SD Card File System – CookBook. MBED.
<https://developer.mbed.org/cookbook/SD-Card-File-System>
- SD/SDHC/SDXC Specifications and Compatibility. Sandisk
https://kb.sandisk.com/app/answers/detail/a_id/2520/~sd%2Fsdhc%2Fsdxc-specifications-and-compatibility
- Physical Layer Simplified Specification 6.0. SD Association.
<https://www.sdcard.org/downloads/pls/>
- How to Use MMC/SDC. ELM-Chan. http://elm-chan.org/docs/mmc/mmc_e.html
- Filesystem considerations for embedded devices. Tristan Lelong. 2015