Memoria EEPROM (***E****lectrically* ***E****rasable* ***P****rogrammable* ***R****ead-****O****nly* ***M****emory*)

- Las memorias EEPROM son memorias no-volátiles (pasivas). Si se desconectan de la fuente de alimentación no pierden la información que guardan.

- A pesar de lo que indica su nombre, en este tipo de memorias se puede guardar (escribir) información (datos), con la particularidad de que el tiempo que tardan en realizar una operación de escritura es mucho mayor que el tiempo que tardan en realizar una operación de lectura.

- A continuación se realiza un pequeño repaso de algunos conceptos básicos sobre memorias semiconductoras.

Memorias semiconductoras

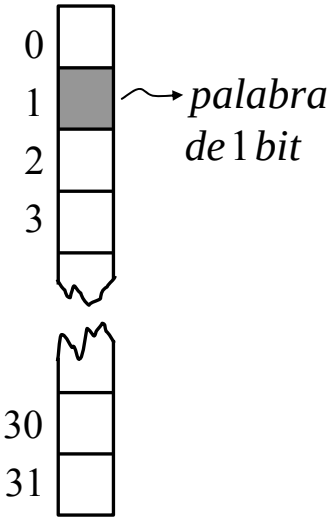
Conceptos básicos:

- A nivel funcional, se puede considerar que una memoria está formada por una *matriz de celdas* en las que se almacena información binaria (1 bit en cada celda)

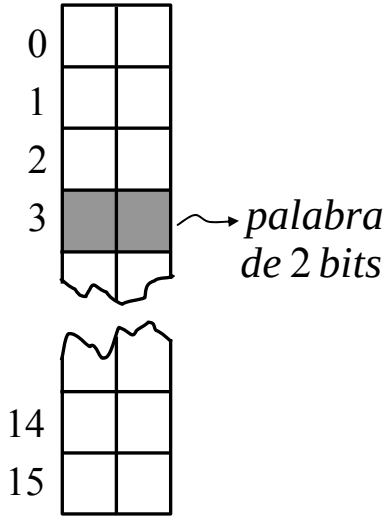
- Las memorias están diseñadas para manejar información en grupos de bits denominados palabras (*words*).

- En cada fila de la *matriz de celdas* se guarda una *palabra* (*word*).

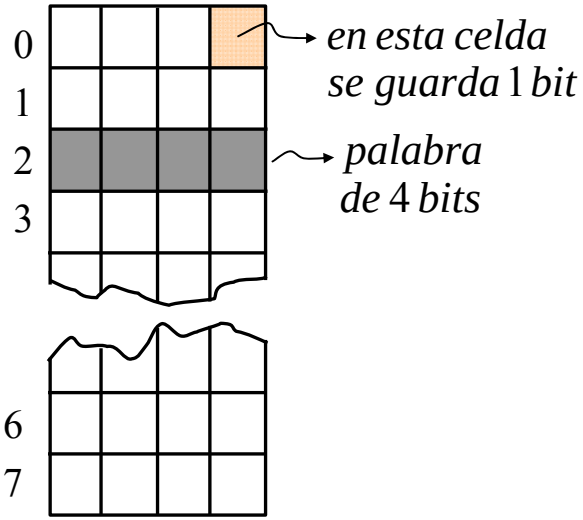
- En función de la memoria que se considere, las *palabras* pueden ser de 1 bit, 4 bits, 8 bits , 9 bits o un múltiplo entero de ocho



32×1



16×2



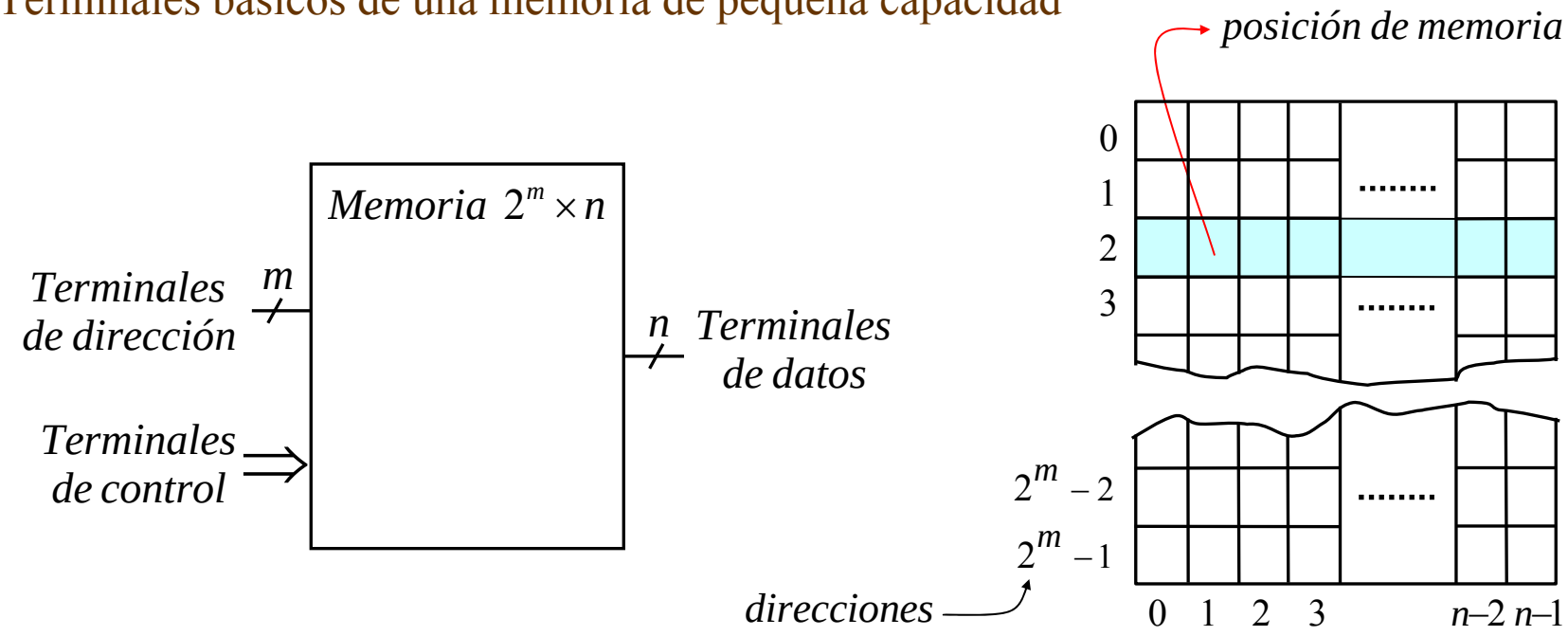
8×4

- Capacidad de una memoria: indica el número total de bits que puede guardar

Los fabricantes acostumbran a indicar la capacidad de las memorias como el producto del número de palabras que pueden guardar por el número de bits que tiene cada palabra. Así, por ejemplo, una memoria ROM 512×4 guarda 512 palabras de 4 bits o, si se prefiere, 2048 bits ó 512 *nibbles*.

La memoria EEPROM del PIC18F452 tiene una capacidad de 256x8 bits

- Terminales básicos de una memoria de pequeña capacidad

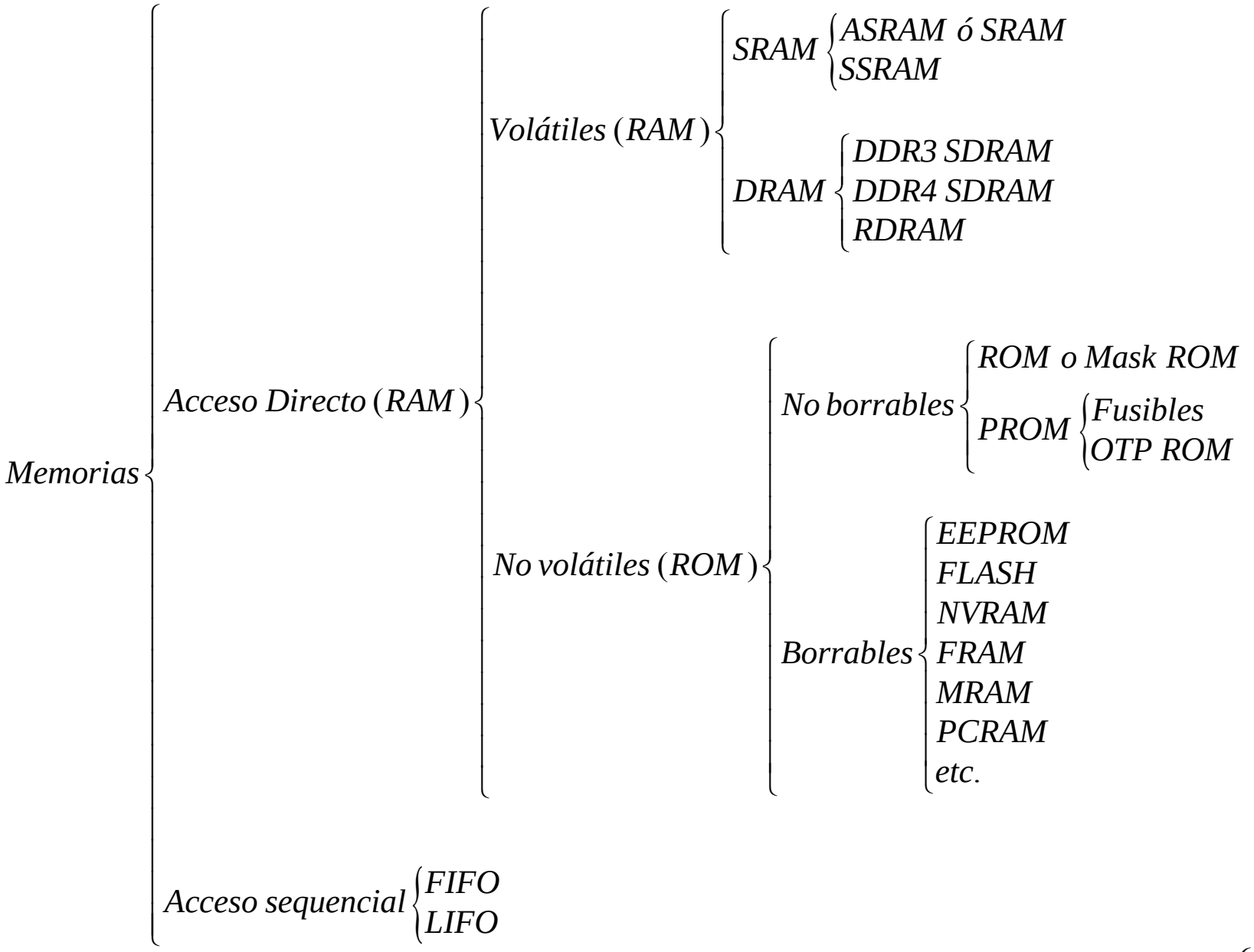


_ La memoria tiene 2^m posiciones (≡ la matriz de celdas tiene 2^m filas).

_ Cada posición tiene asociada una dirección.

_ La posición en la que se lee (*read*) o se guarda (*write*) un dato se determina decodificando la dirección presente en los terminales de dirección.

_ En cada posición (dirección) de la memoria se guardan *n* bits (≡ la matriz de celdas tiene *n* columnas).


Memorias
Acceso Directo (RAM)
Volátiles (RAM)
SRAM
ASRAM ó SRAM
SSRAM
DRAM
DDR3 SDRAM
DDR4 SDRAM
RDRAM
No volátiles (ROM)
No borrables
ROM o Mask ROM
PROM
Fusibles
OTP ROM
Borrables
EEPROM
FLASH
NVRAM
FRAM
MRAM
PCRAM
etc.
Acceso sequencial
FIFO
LIFO

- El PIC18F452 tiene una memoria EEPROM de capacidad 256x8. Lo que implica que puede guardar 256 bytes.

- La memoria EEPROM no aparece en el mapa de memoria del microcontrolador. Se puede acceder a ella a por medio de 4 registros: EECON1, EECON2, EEDATA y EEADR.

- La E^2PROM realiza un borrado de una posición de memoria antes de escribir un dato en ducha posición (*erase-before-write*)

- El registro EEADR guarda la dirección de la posición de memoria a la que se quiere acceder para realizar una operación de lectura o de escritura.

- El registro EEDATA guarda el dato leído en la EEPROM (en una operación de lectura) o el dato a guardar (en el caso de una operación de escritura).

Registro EECON1: controla el acceso a las memorias EEPROM y FLASH

EEPGD	CFGS	–	FREE	WRERR	WREN	WR	RD

EECON1.EEPGD → hay que poner este bit a 0 para acceder a la memoria EEPROM. Para acceder a la memoria Flash hay que ponerlo a 1.

EECON1.CFGS → hay que poner este bit a 0 para acceder a la memoria EEPROM. Para acceder a la memoria Flash hay que ponerlo a 1.

EECON1.B5 no tiene ninguna función asociada

EECON1.FREE (se utiliza con la memoria Flash)

EECON1.WRERR → si este bit se pone a 1 significa que la operación de escritura no se ha realizado correctamente. Y si se mantiene a 0, significa que la operación de escritura se ha realizado correctamente.

EECON1.WREN = 1/0; → habilita/deshabilita las operaciones de escritura en la EEPROM.

EECON1.WR → cuando este bit se pone a 1 se inicia un ciclo de borrado + escritura en una posición de memoria de la EEPROM. Una vez que la operación de escritura ha finalizado, este bit se pone a 0 (por hardware).

EECON1.RD → cuando se pone a 1 este bit, se inicia una operación de lectura en la EEPROM

▪ Operación de *lectura* en la EEPROM: Microchip indica el siguiente orden de programación de una operación de lectura (de 1 byte)

1º) EEADR = *xxxx* ≡ dirección en la que se guarda el byte que se quiere conocer

2º) EECON1.EEPGD = 0;
3º) EECON1.CFGS = 0;
} → para acceder a la memoria EEPROM

4º) EECON1.RD = 1; → se inicia la operación de lectura.

5º) *x* = EEDATA; → se lee el contenido del registro EEDATA. En este registro hay una copia del contenido (1 byte) de la dirección indicada en el registro EEADR. El registro EEDATA conserva el dato leído en la memoria EEPROM hasta que se realice otra operación de lectura o de escritura.

Nota compilador MikroC: *x* = eeprom_read (*dirección*);

· Operación de *escritura* en la EEPROM: Microchip indica el siguiente orden de programación de una operación de escritura (de 1 byte)

1º) EEADR = *dirección*; → en el registro EEADR se guarda la dirección de la EEPROM en la que se guardará el dato (byte)

2º) EEDATA = *dato*; → en el registro EEDATA se guarda el dato que se quiere guardar en la EEPROM

3º) EECON1.EEPGD = 0;

4º) EECON1.CFGS = 0;

5º) EECON1.WREN = 1; → se habilitan las operaciones de escritura en la EEPROM

6º) INTCON.GIE = 0;
EECON2 = 0x55;
EECON2 = 0xAA;
EECON1.WR = 1;
INTCON.GIE = 1;

Cuando finaliza una operación de escritura, el bit (flag) PIR2.EEIF se pone a 1 y el bit EECON1.WR se pone a 0.

Una vez finalizada una operación de escritura hay que poner el bit EECON1.WREN a 0 para deshabilitar las operaciones de escritura.

Nota compilador MikroC: eeprom_write (*dirección*, *dato*);

Para *habilitar las interrupciones por finalización de una operación de escritura* en la memoria EEPROM hay que hacer lo siguiente:

```
PIE2.EEIE = 1:
PIR2.EEIF = 0;
INTCON.PEIE = 1;
INTCON.GIE = 1;
```

Nota: una buena práctica de programación consiste en verificar las operaciones de escritura. Es decir, realizar una operación de lectura en la posición de memoria en la que se acaba de escribir un dato y comprobar que el dato leído coincide con el que se quería guardar.

Nota: una vez iniciada una operación de escritura, se recomienda esperar un mínimo de 20mseg. antes de iniciar otra operación de escritura o de lectura.

Nota: se recomienda deshabilitar las interrupciones (GIE = 0) durante la ejecución de las funciones de librería del compilador de MikroC para acceder a la EEPROM.

TABLE 6-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	T0IF	INTF	RBIF
FA9h	EEADR	EEPROM Address Register							
FA8h	EEDATA	EEPROM Data Register							
FA7h	EECON2	EEPROM Control Register2 (not a physical register)							
FA6h	EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
FA2h	IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
FA1h	PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
FA0h	PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE

Legend: x = unknown, u = unchanged, r = reserved, - = unimplemented, read as '0'.
Shaded cells are not used during FLASH/EEPROM access.

En relación a la práctica de '*control de acceso*' hay que tener en cuenta que el sistema a diseñar debe realizar las siguientes tareas:

- Debe guardar en la memoria EEPROM una clave de, al menos, 5 dígitos

- En el LCD se debe mostrar un mensaje indicando que se introduzca una clave

- El sistema debe comprobar si la clave introducida es correcta o no. En el caso de que sea correcta, el pin RA0 debe ponerse a 1 durante 4 segundos (para temporizar los 4 segundos se puede utilizar la función *delay*). Si la clave no es correcta, el pin RA0 debe mantenerse a 0 y en el LCD debe mostrarse un mensaje indicando que la clave introducida no es correcta.

Ejemplo de uso de las funciones de la biblioteca de funciones del compilador: con las siguientes instrucciones se guarda el valor 23 en la posición (dirección) 150 de la memoria EEPROM y a continuación se lee el valor guardado en dicha posición (dirección). El valor leído se guarda en la variable *aux*.

```
Eeprom_Write(150,23); //dirección, dato (escritura)
delay_ms(20);
aux = Eeprom_Read(150);  // lectura
```