Tema 4

Ejemplo de un computador real: ATmegaX8PA

Autores originales: Alberto J. Molina, Paulino Ruiz de Clavijo

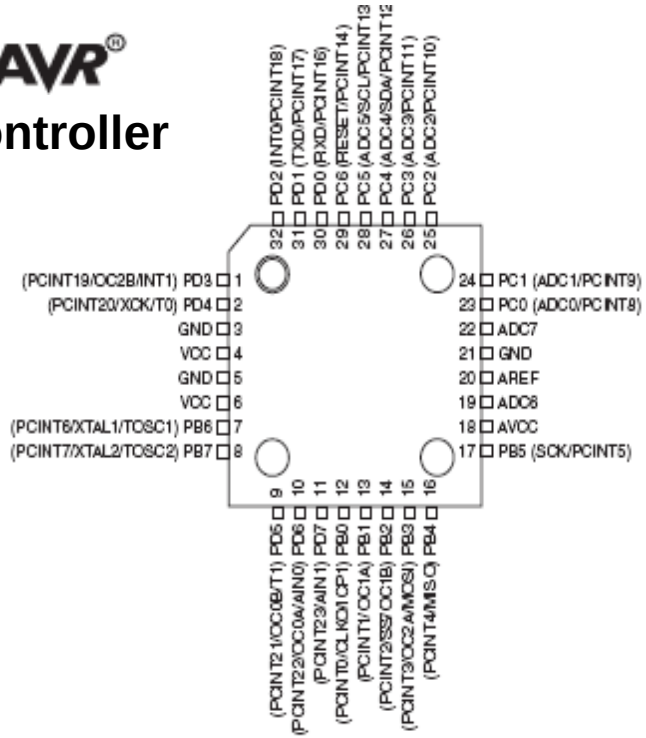
Modificaciones: Pilar Parra e Isabel Gómez

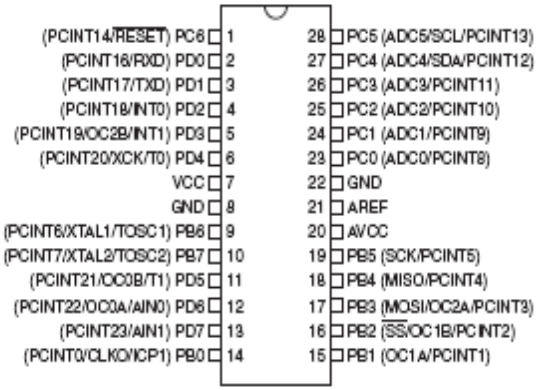
Última adaptación: 17/05/2023

Usted es libre de copiar, distribuir y comunicar públicamente la obra y de hacer obras derivadas siempre que se cite la fuente y se respeten las condiciones de la licencia Attribution-Share alike de Creative Commons.
Texto completo de la licencia: http://creativecommons.org/licenses/by-nc-sa/3.0/es/

Tema 4
Ejemplo de un computador real: ATmegaX8PA

8-bit Microcontroller

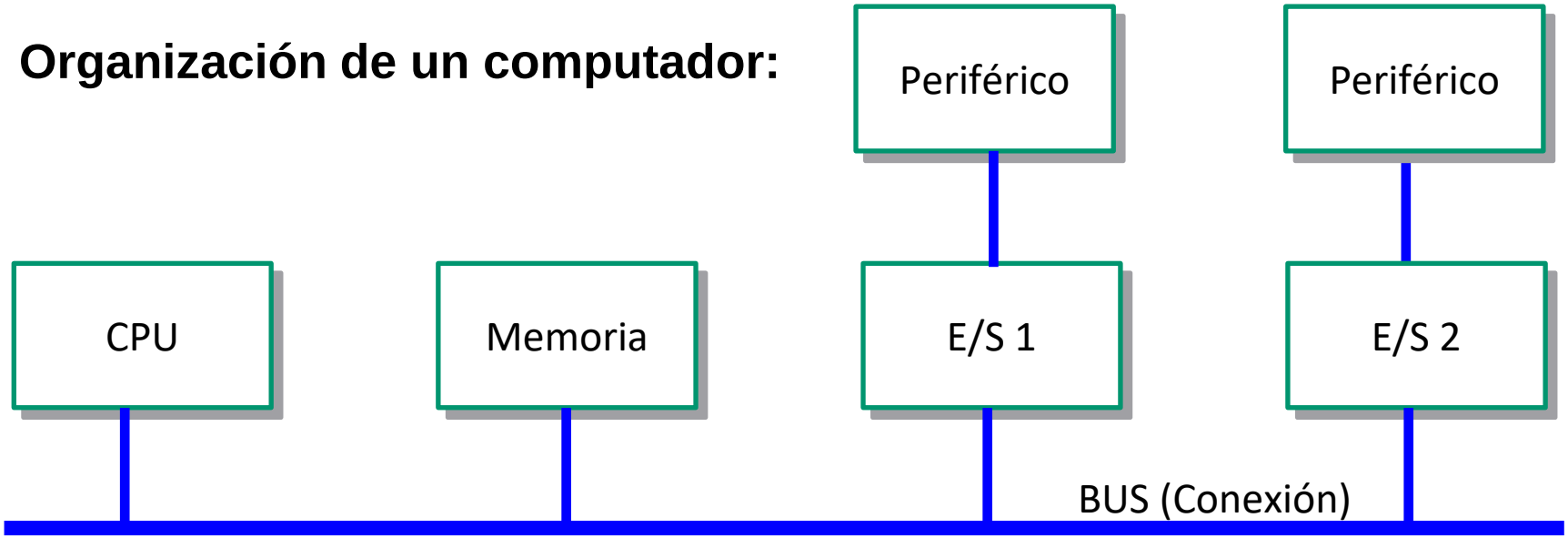




Índice

¿Qué es un microcontrolador?

Organización de un computador:



En un microcontrolador:

- Todos estos componentes están integrados en un chip.
- Constituyen un microcomputador de propósito específico que no posee mucha capacidad

Introducción

- **Alternativas al uso del microcontrolador**

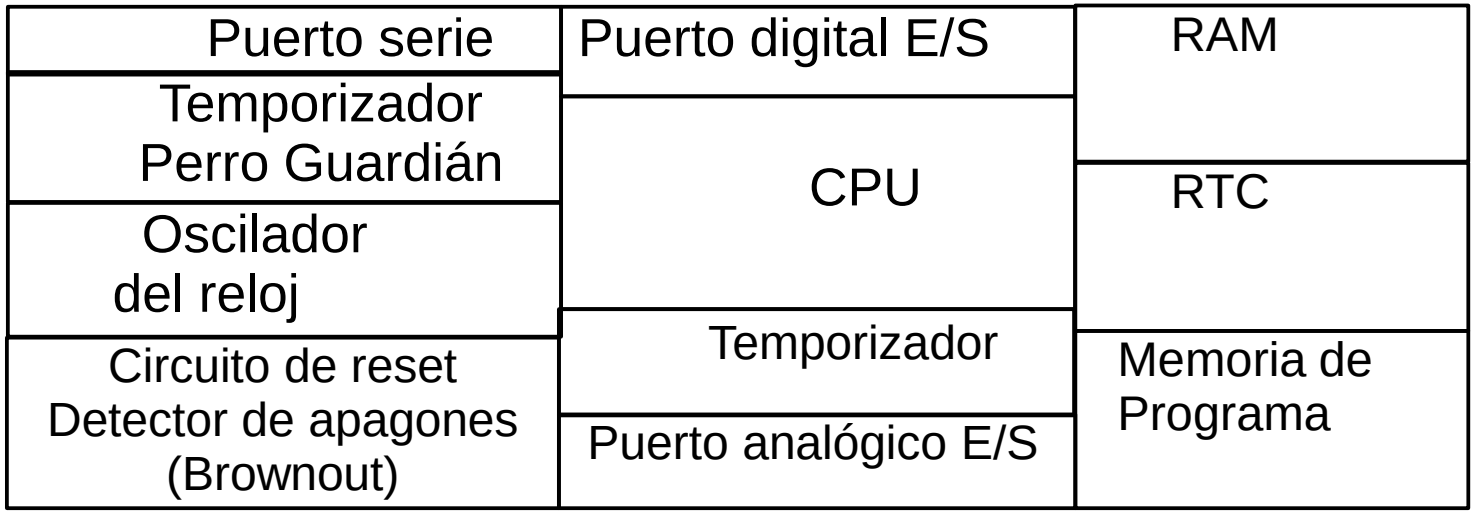
Circuito digital a medida

Circuito digital basado en dispositivos programables

Circuito integrado específico para la aplicación: ASIC

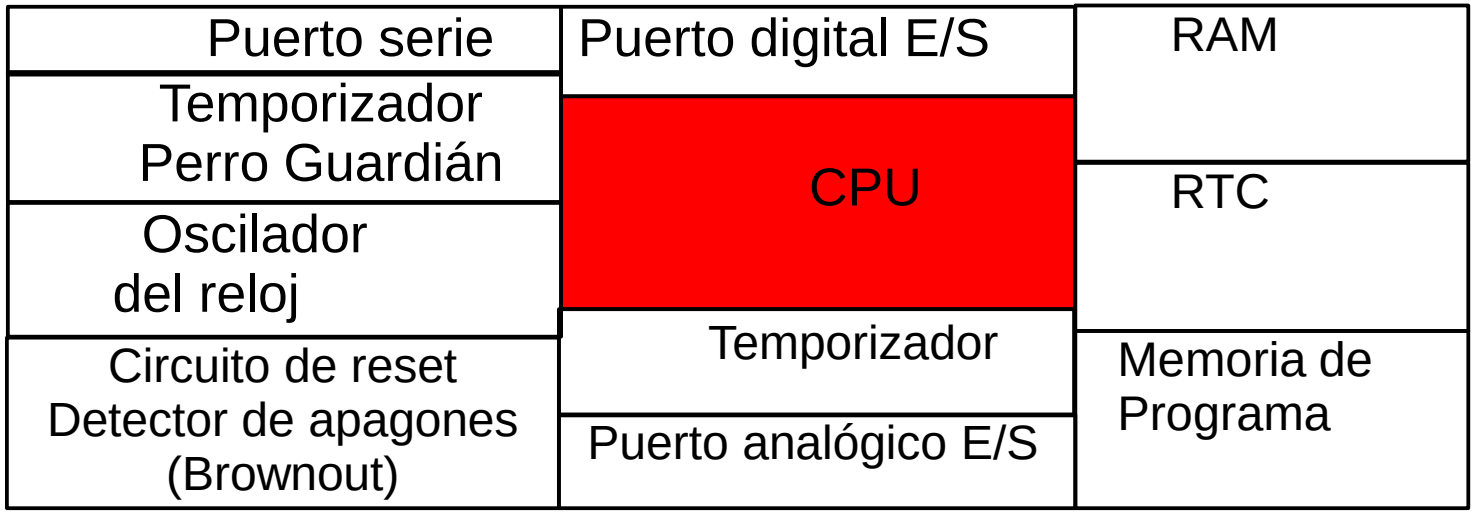
Introducción

- **Componentes de un microcontrolador**



Introducción

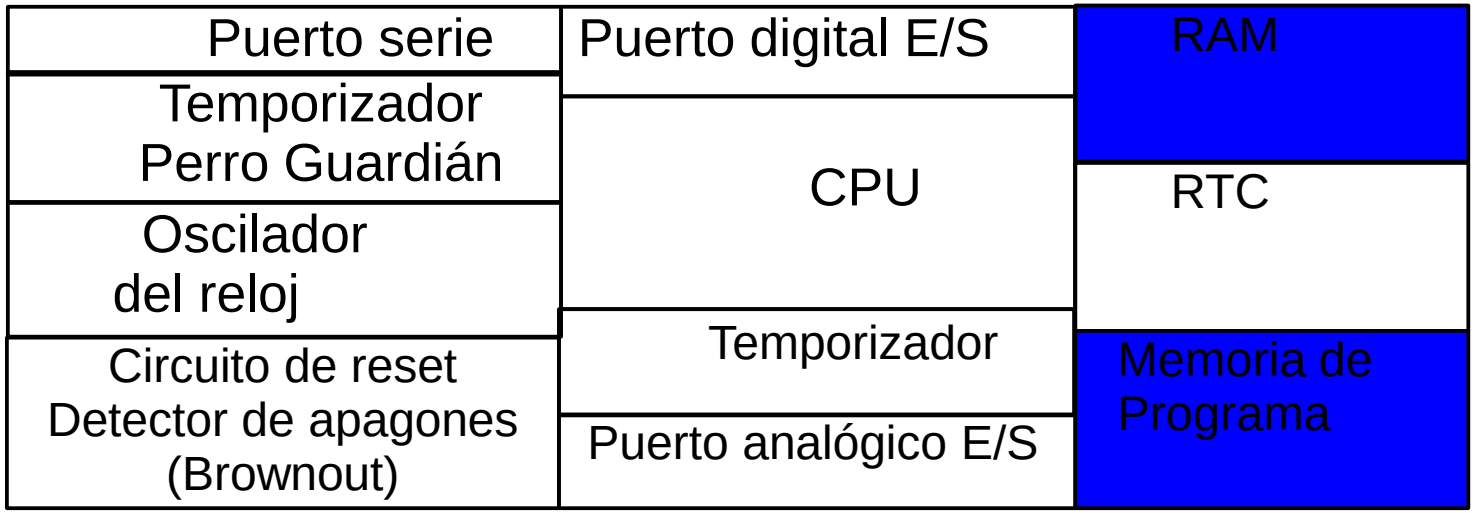
- **Componentes de un microcontrolador**



CPU: corazón del microcontrolador, busca las instrucciones, las decodifica y las ejecuta

Introducción

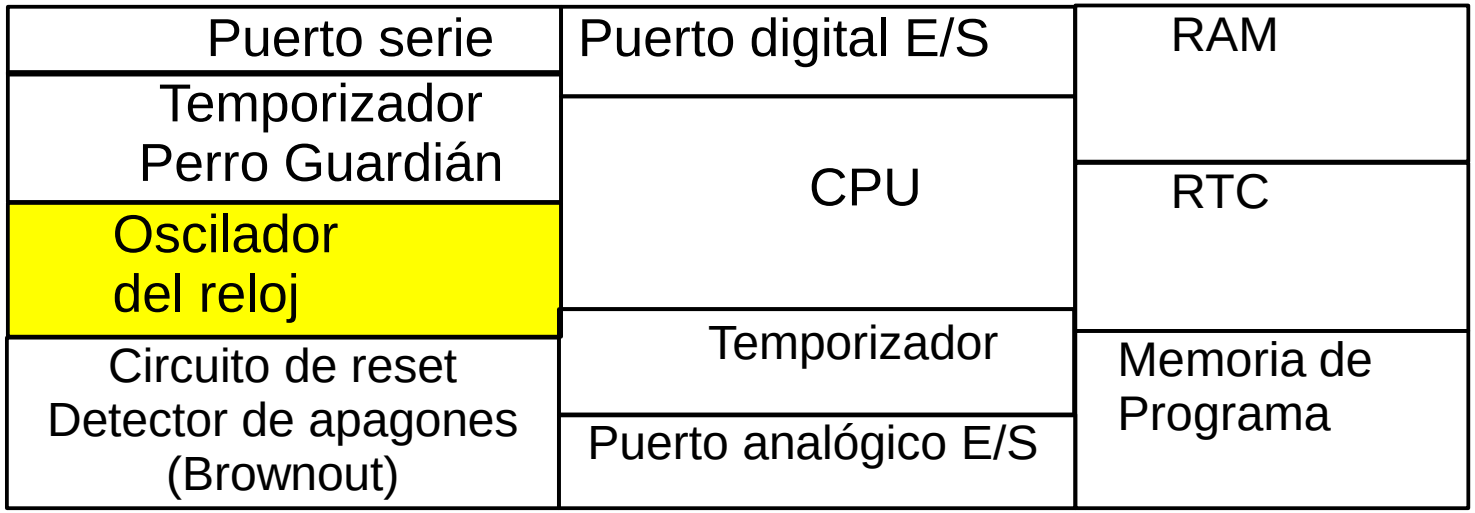
- **Componentes de un microcontrolador**



Memorias:
- La memoria de programa sirve para el almacenamiento de las instrucciones.
- La RAM es una memoria de datos, también se utiliza de Pila.

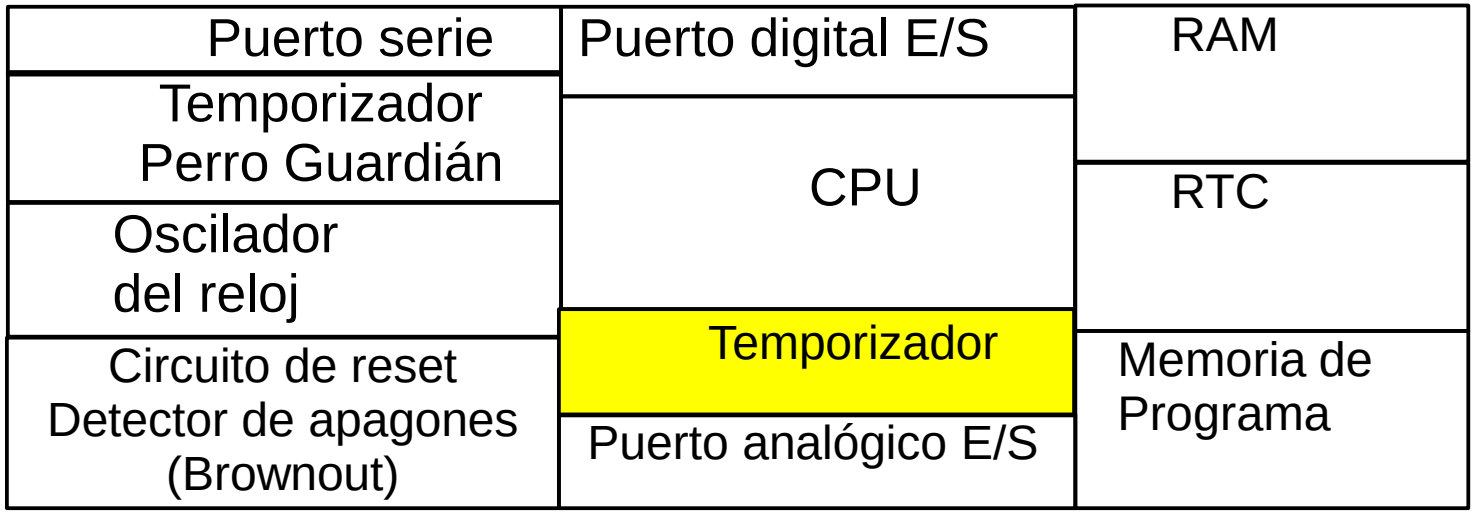
Introducción

- **Componentes de un microcontrolador**



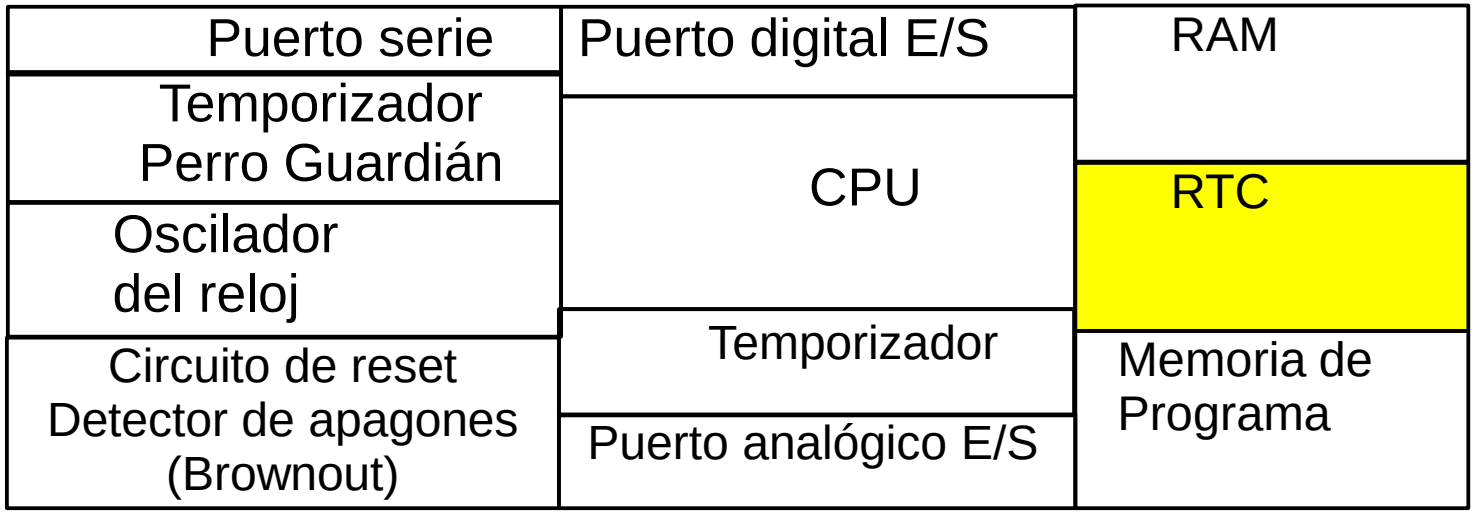
Introducción

- **Componentes de un microcontrolador**



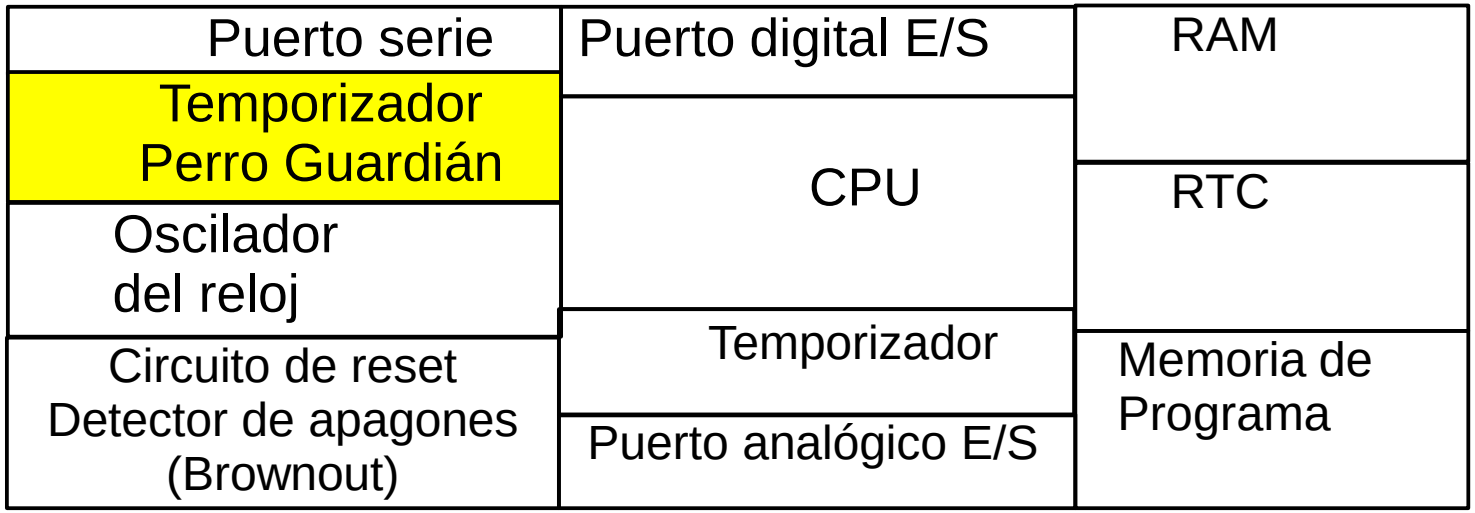
Introducción

- **Componentes de un microcontrolador**



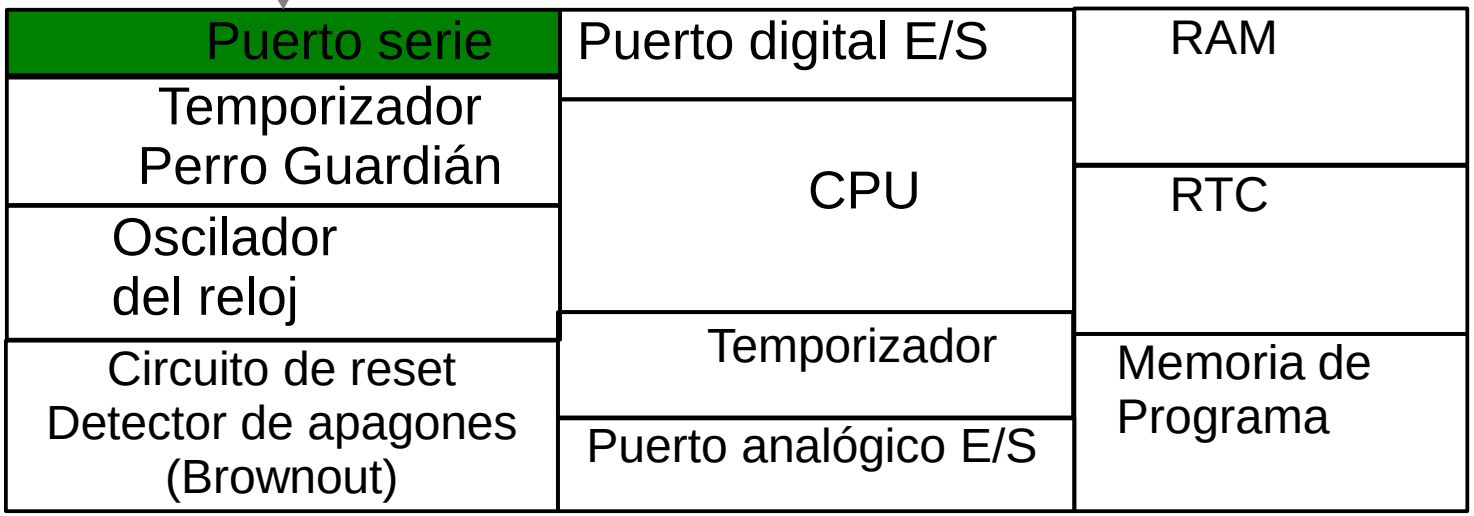
Introducción

- **Componentes de un microcontrolador**



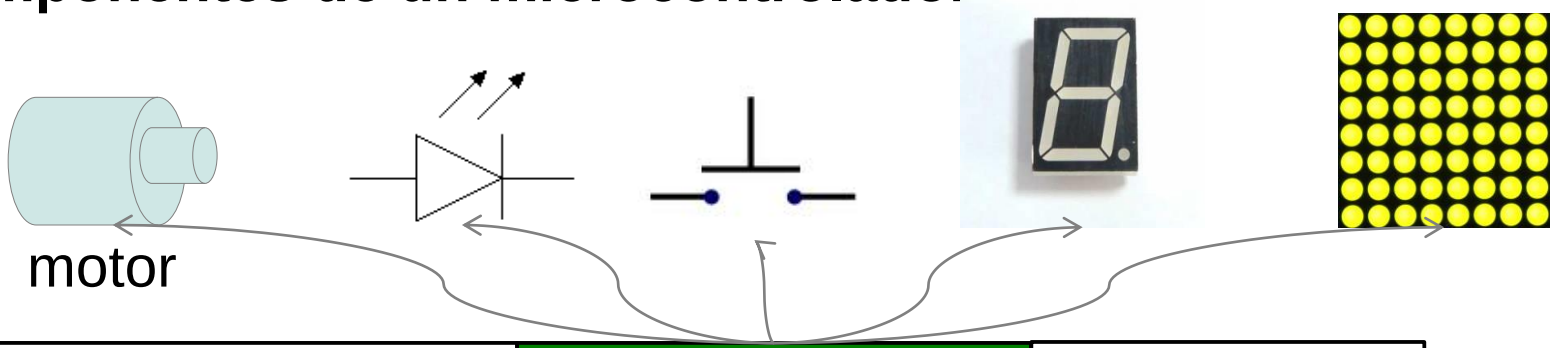
Introducción

- **Componentes de un microcontrolador**



Introducción

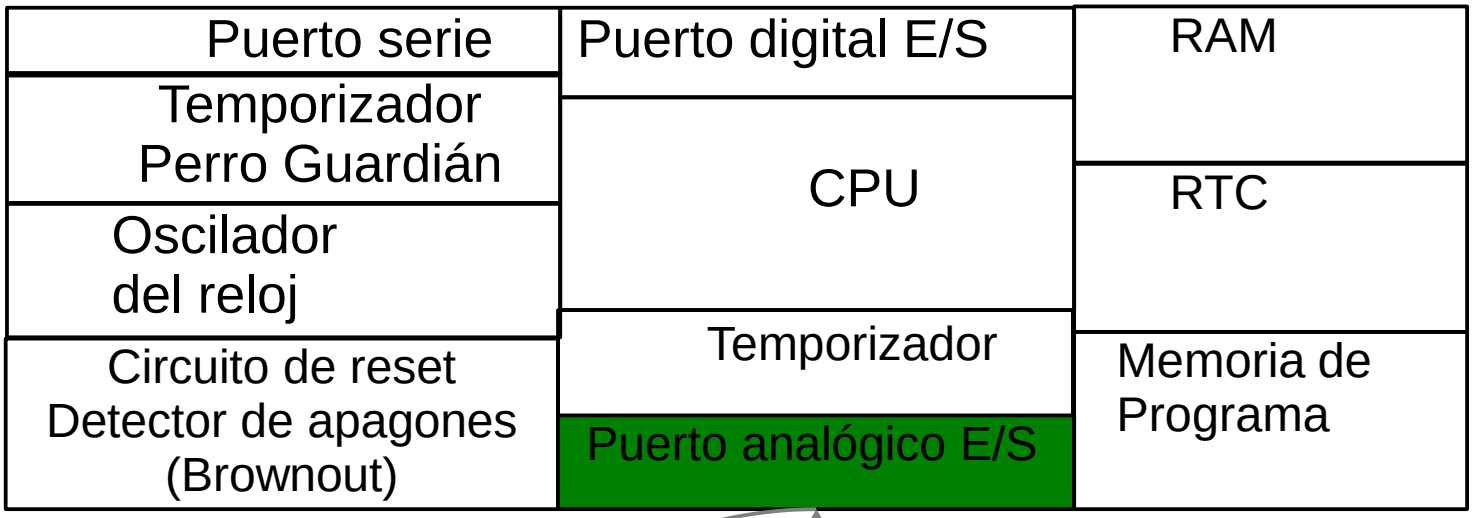
- **Componentes de un microcontrolador**



Puerto serie	Puerto digital E/S	RAM
Temporizador Perro Guardián	CPU	RTC
Oscilador del reloj		
Circuito de reset Detector de apagones (Brownout)	Temporizador	Memoria de Programa
	Puerto analógico E/S	

Introducción

- **Componentes de un microcontrolador**



Principales familias de microcontroladores

- ***PICmicro*** 

- ***AVR*** 

- ***Freescale*** (franquicia de Motorola) 

Aplicaciones µcontroladores

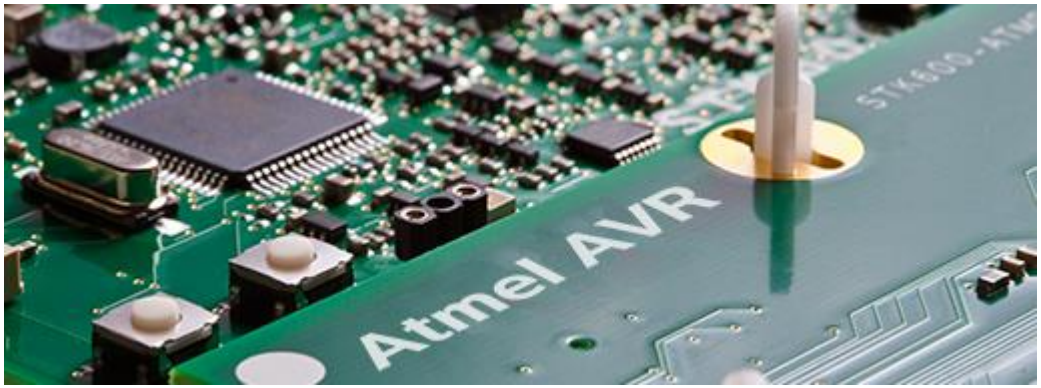
Aplicaciones µcontroladores

Aplicaciones µcontroladores

Índice

Familia AVR ATmega

- AVR ATmega es una familia de **microcontroladores** (MCU o µC) fabricado por Atmel



- Es *una pequeña computadora empotrada en un C.I.* que contiene una CPU sencilla, unidad de reloj, memoria, y puertos de E/S (timers, IO ports, USARTs,..).

- Estudiaremos el **ATmegaX8PA**

ATmegaX8PA
Características

- Arquitectura RISC 8 bits
- Frecuencia de reloj de hasta 20 Mhz @ (4.5-5.5V)
- Hasta 20 Mips (20Mhz)
- Flash EEPROM X=4,8,16,32 Kb ATmegaX8pa
- Data SRAM 512/1K/2K bytes
- Data EEPROM 256/512/1024 bytes
- Versiones bajo consumo: 0-10 Mhz< @(2.7-5.5V) 0-4 Mhz< @(1.8-5.5V)

Familia AVR ATmega

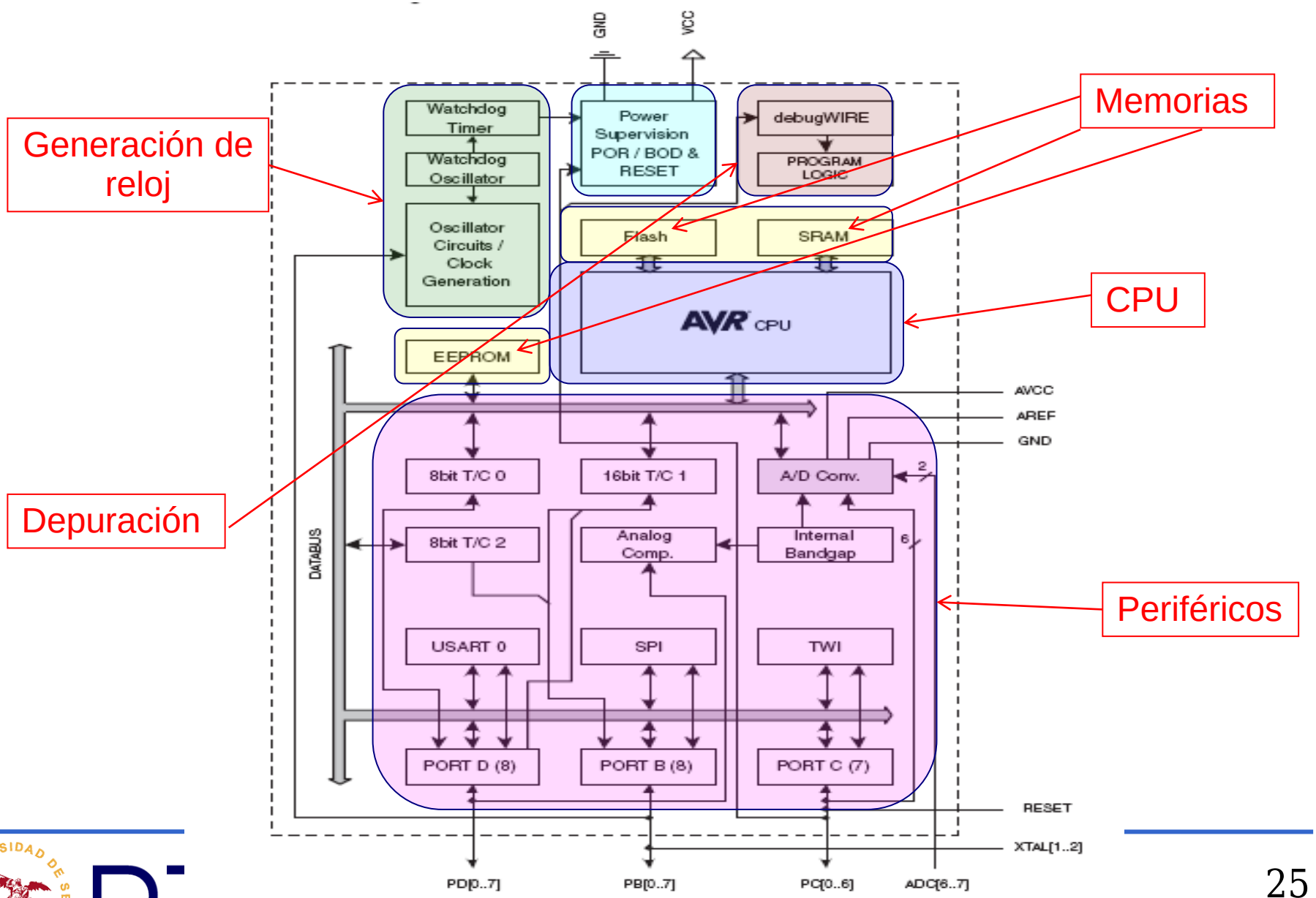
Dispositivos

ATmegaX8PA

Dispositivos	Flash (Programa)	EPROM (datos)	SRAM (datos)	Tamaño vector interrupción
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 palabra
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 palabra
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 palabras
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 palabras

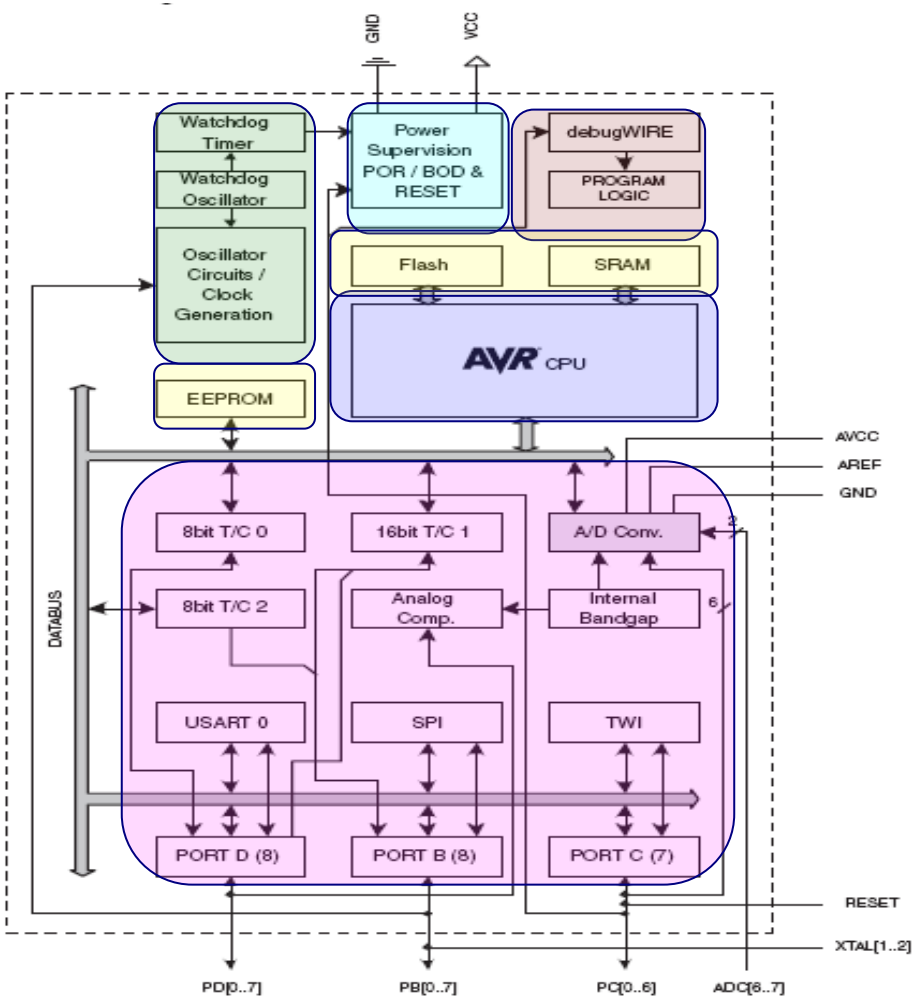
Índice

Arquitectura interna



Arquitectura interna

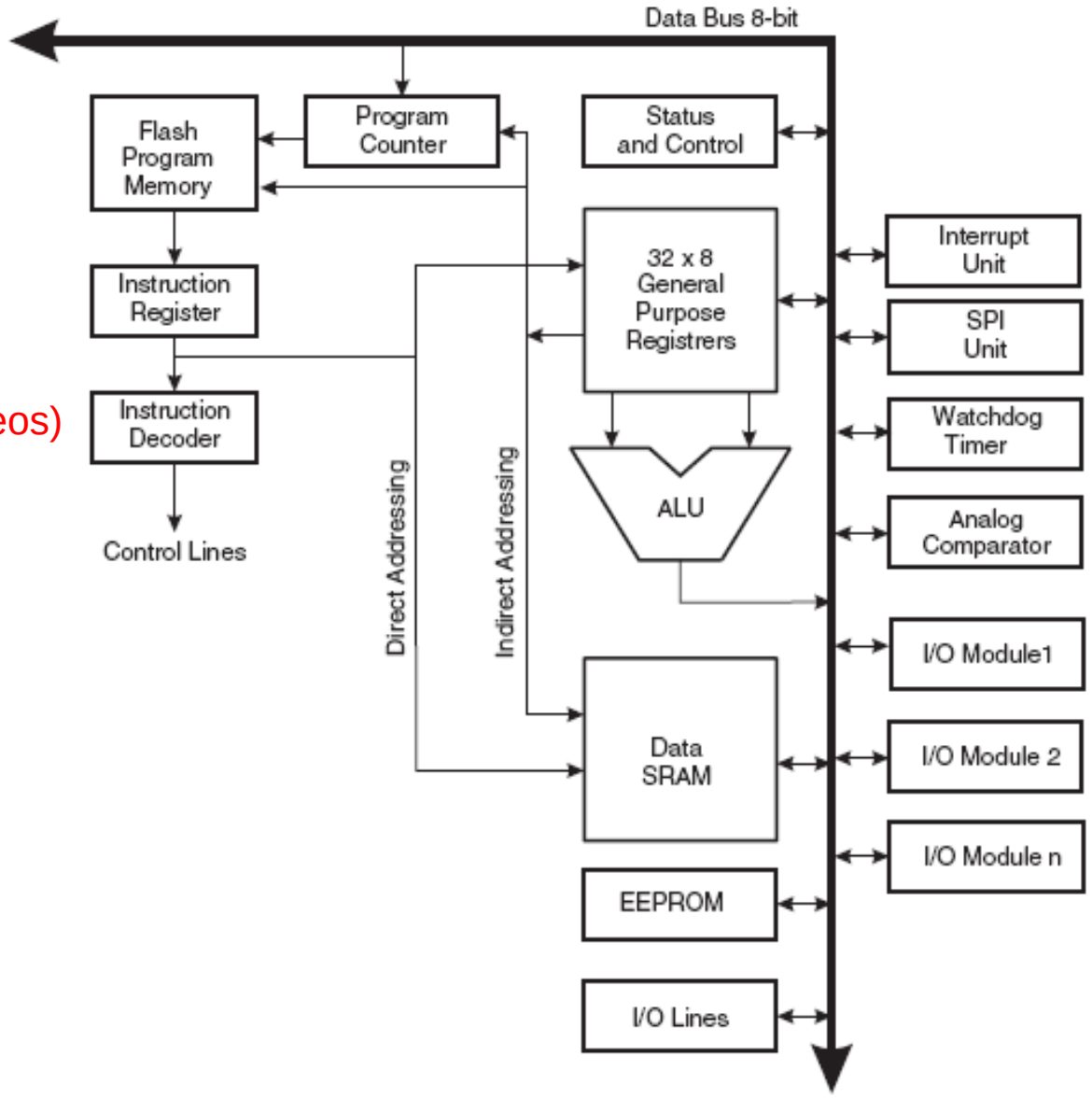
- AVR-CPU
- Memorias (Flash, SRAM)
- Generación de reloj
- Depuración
- Circuito de Reset
- Periféricos y gestión de puertos:
  - CAD, Analog Comp.
  - Ports B,C,D
  - USART, TWI, SPI
  - Timers



Arquitectura interna

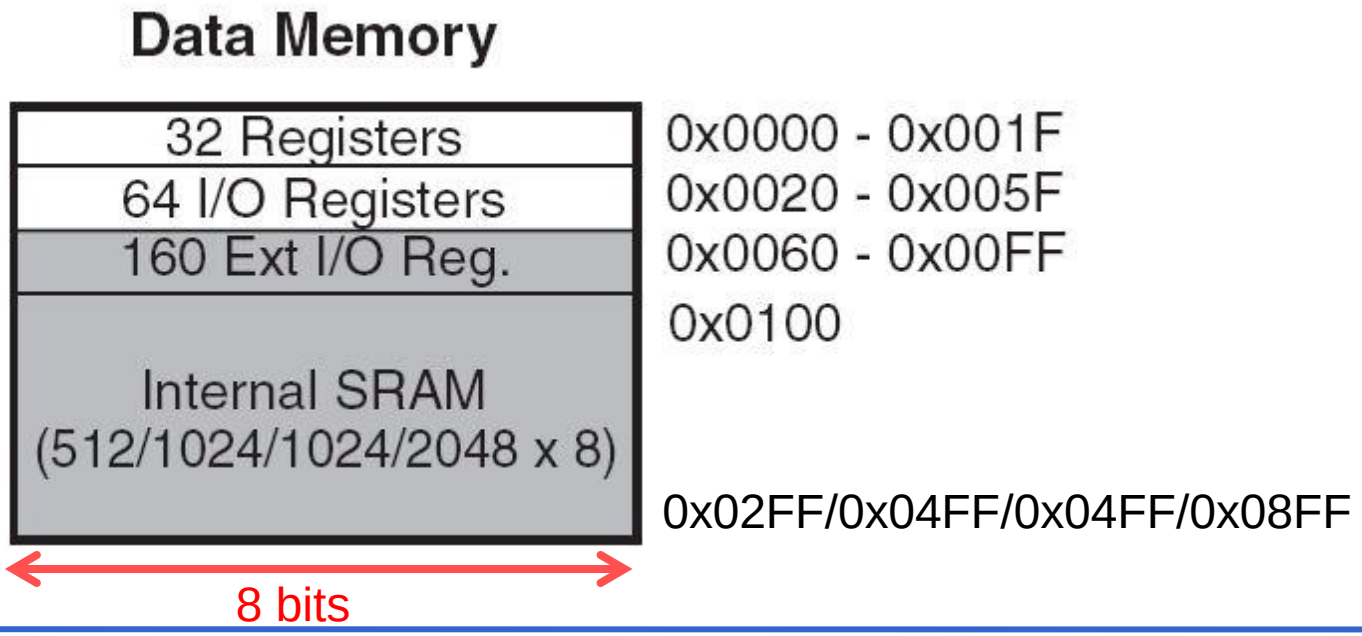
- Arquitectura Harvard
- Pipeline de 1 nivel:

  (Execute y pre-fetch simultáneos)
- Bus de datos de 8 bits
- Banco de registros
- Arquitectura RISC
  — Load/Store
- ALU con multiplicador



Mapa de la memoria de datos

- **32 Registros de propósito general**
- **Registros de E/S**
  - 64 I/O Registers
  - 160 Ext I/O Registers
- **SRAM**

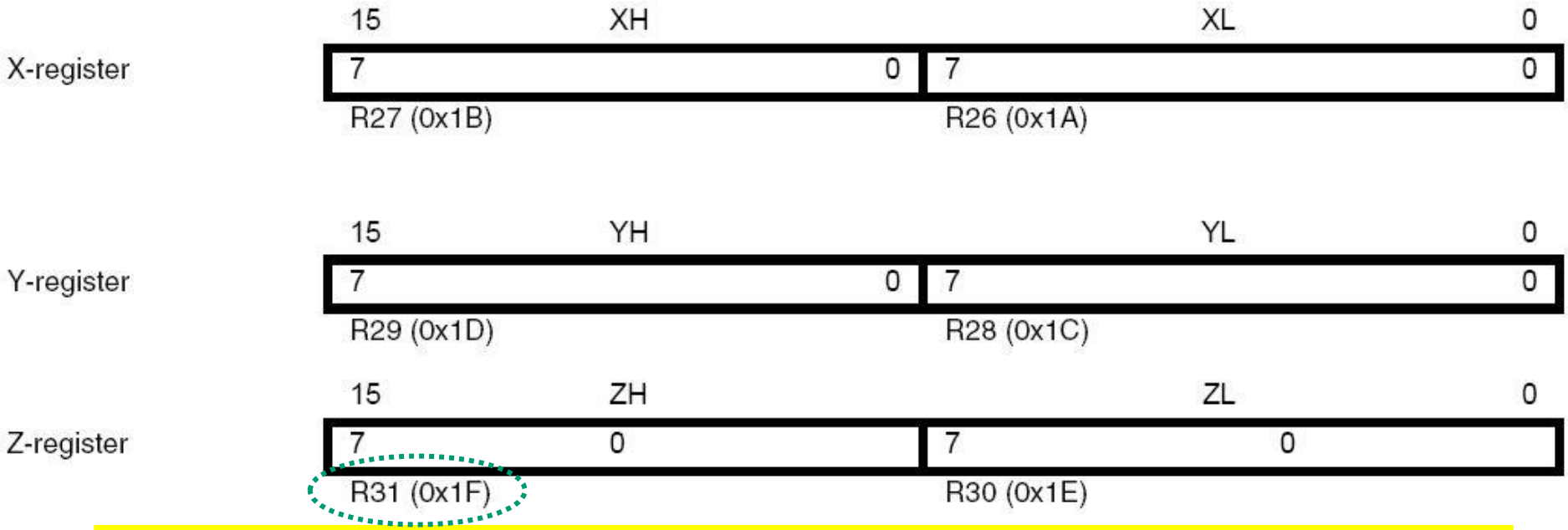


Registros de propósito general

- Ocupan las 32 primeras posiciones dentro del mapa de la memoria de datos
- Las instrucciones con datos de direccionamiento inmediato sólo pueden usar los 16 superiores (**R16 a R31**)
- **X, Y, Z**: registros de 16 bits para modos de direccionamiento indirecto (punteros)

Registros X, Y, Z

- Se comportan como registros de 16 bits cuando se usan (direccionamientos indirectos)



Entre paréntesis aparece la ubicación del registro de propósito general dentro del mapa de la “memoria de datos” (por ejemplo, para R31-> hex(31) = 0x1F

Registro de estado (SREG)

- Ubicado en el área de entrada/salida de la memoria de datos.

  - este registro ocupa la posición 63 dentro de los 64 registros de E/S, en hexadecimal: $3F
  - entre paréntesis aparece $5F (que es 95 en decimal) indicando la posición absoluta del registro SREG dentro del mapa de la “memoria de datos”.

- Sus 8 bits son banderines (flags) que reflejan el resultado de la ejecución de algunas instrucciones (principalmente aritméticas, lógicas, etc.)

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Registro de estado (SREG)

- **Bit C** (Carry)
- **Bit Z** (Zero): Se pone a 1 si los 8 bits del resultado son cero.
- **Bit N** (Negative): Bit 7 del resultado.
- **Bit V** (oVerflow en Ca2)
- **Bit S** (Signo: S = N ⊕ V): Signo correcto en operaciones Ca2
- **Bit H** (Half Carry): Bit de acarreo de la etapa 3 de la ALU (semiacarreo).
- **Bit T** Bit de propósito general, para instrucciones BLD y BST
- **Bit I** (Interrupción): Para permitir (I=1) o deshabilitar (I=0) interrupciones

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Puntero de pila (SP)

- El puntero de pila (Stack Pointer) tiene 16 bits y está formado por dos registros: SPH = SP[15:8] y SPL = SP[7:0]
- SPH y SPL están ubicados en el área de entrada/salida de la memoria de datos.
- SP apunta al área de pila (Valor inicial RAMEND)
- El SP apunta a la dirección siguiente a la cima de la PILA, es decir, a la primera posición disponible para escritura.

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
0x3D (0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

Puntero de pila (SP)

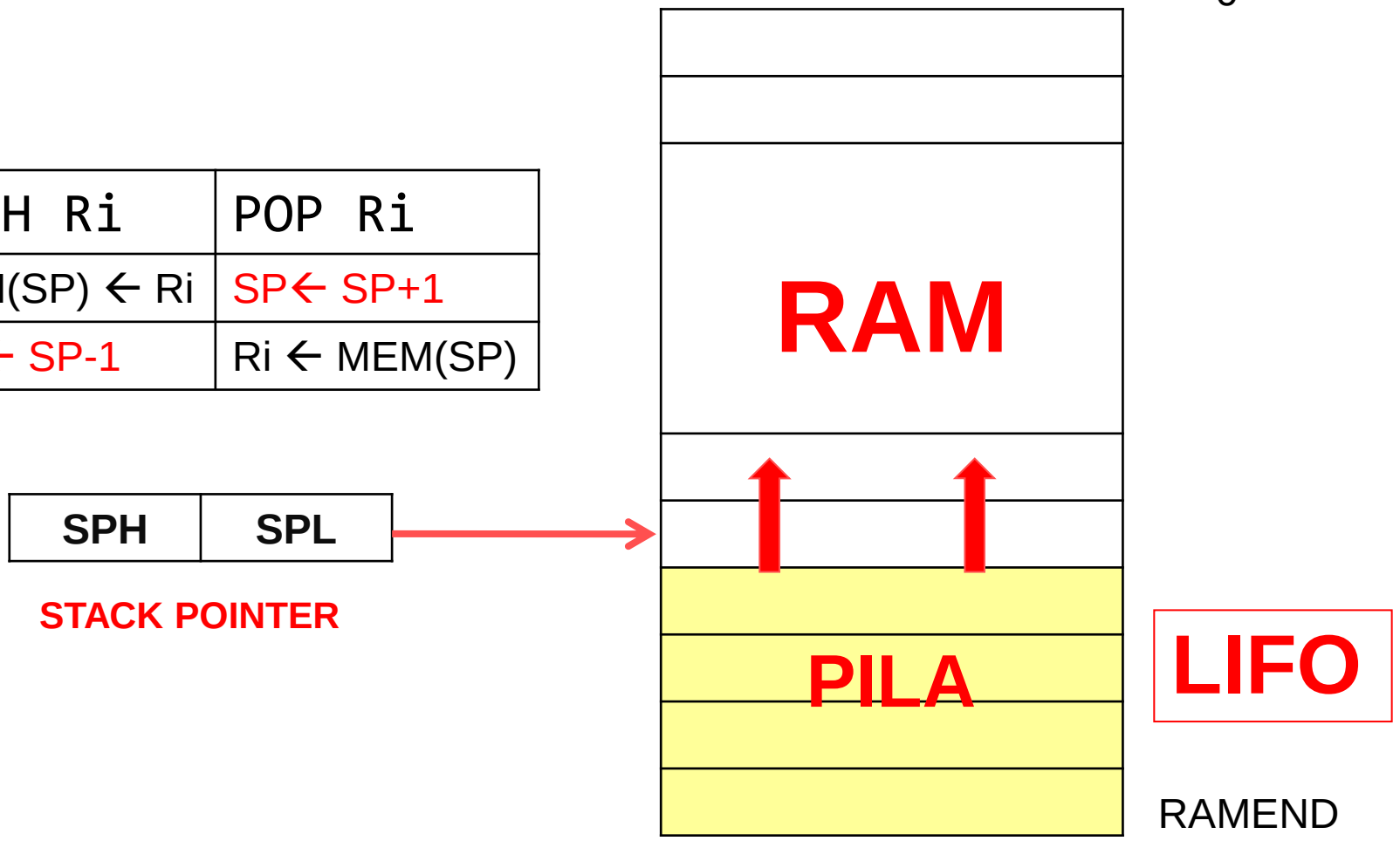
- SP se decrementa (post) una unidad al meter un dato en pila (PUSH) pila y se incrementa (pre) una unidad al sacarlo (POP).

- Las llamadas a subrutinas o rutinas de interrupción (RCALL, etc) hacen que el SP se decremente (post) en dos unidades y para los retornos (RET, RETI), el puntero se incrementa (pre) en dos unidades

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
0x3D (0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	
	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	RAMEND	

Puntero de pila (PUSH/POP)

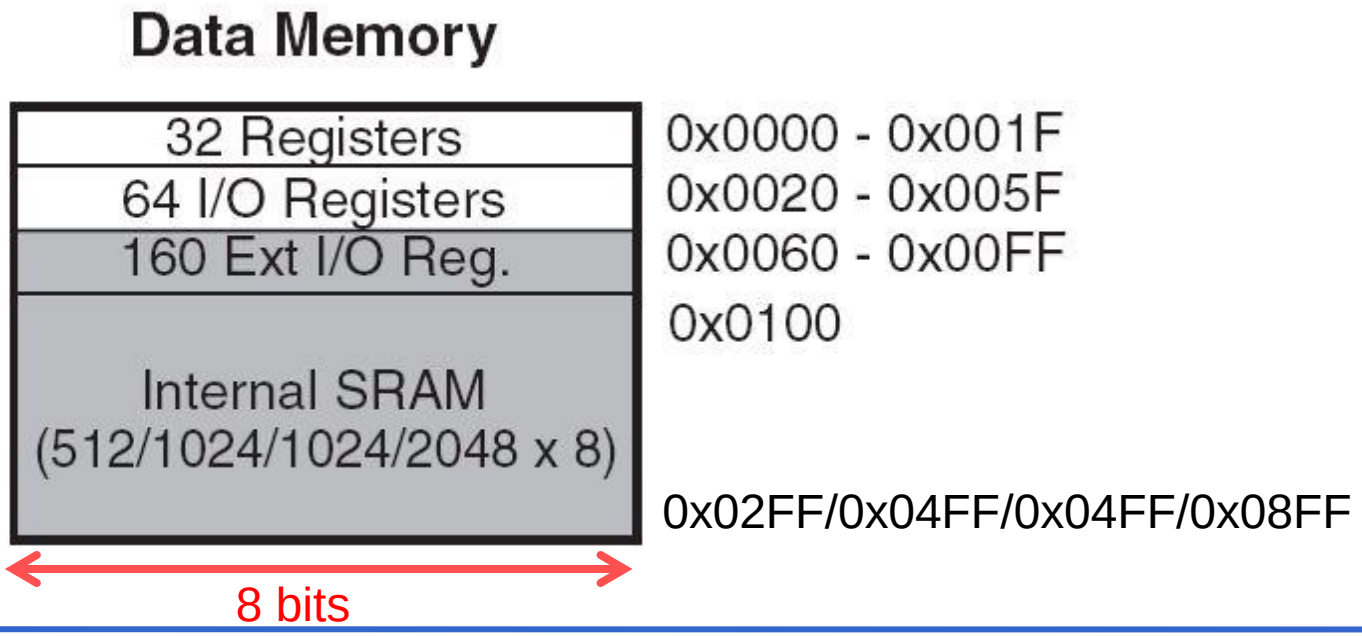
PUSH Ri	POP Ri
MEM(SP) ← Ri	SP← SP+1
SP ← SP-1	Ri ← MEM(SP)



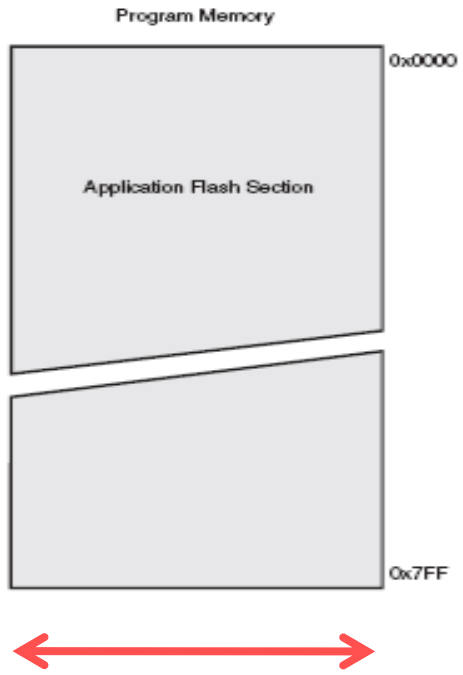
Índice

Mapa de la memoria de datos (rep)

- **32 Registros de propósito general**
- **Registros de E/S**
  - 64 I/O Registers
  - 160 Ext I/O Registers
- **SRAM**

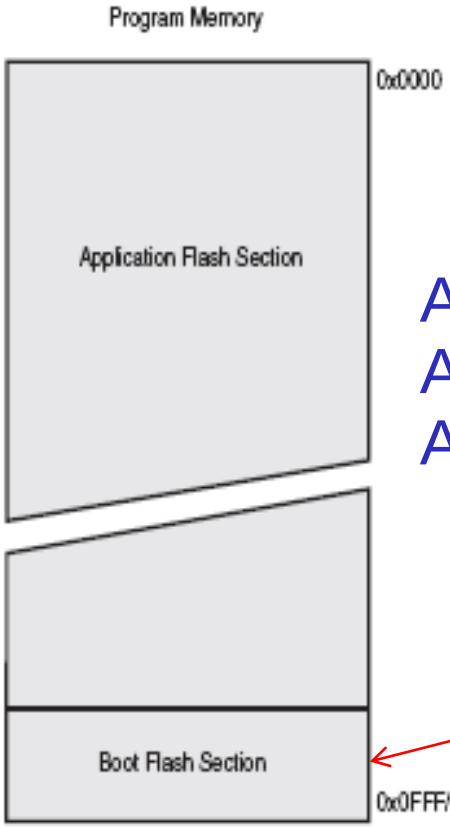


Organización de la memoria de programa



16 bits

Atmega48PA



Atmega88PA
Atmega168PA
Atmega328PA

Sección de arranque

16 bits

Índice

Modos de direccionamiento a la memoria de datos

- **Registro directo**
- **E/S directo**
- **Directo a memoria**
- **Indirectos**
  - Indirecto
  - Indirecto con predecremento
  - Indirecto con postincremento
  - Indirecto con desplazamiento
- **Inmediato**

Modos de direccionamiento de la memoria de datos

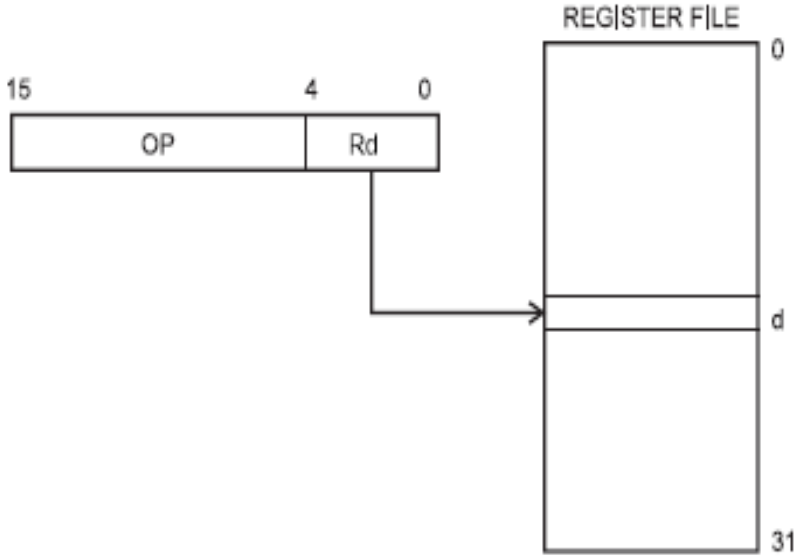
- **32 Registros de propósito general:**
  - todos los modos de direccionamiento, **inmediato sólo de R16 a R31, indirecto (o indexado) solo X,Y,Z**
- **Registros de E/S**
  - 64 I/O Registers:
    - de E/S directo, directo e indirecto
  - 160 Ext I/O Registers
    - directo e indirecto
- **SRAM**
  - directo e indirecto

Direccionamiento de registro directo

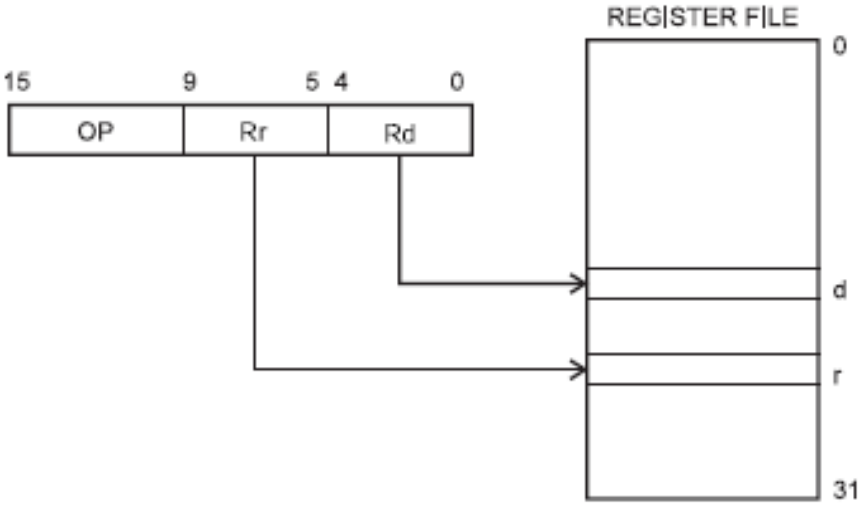
El operando está contenido en un registro de propósito general (R0-R31) y en la instrucción aparece como Ri

Ejemplos:

```
COM R4; Calcula el Ca1 de R4
      R4 ← $FF-R4
```




```
MOV R1,R2; Transfiere R2 a R1
    R1 ← R2
```

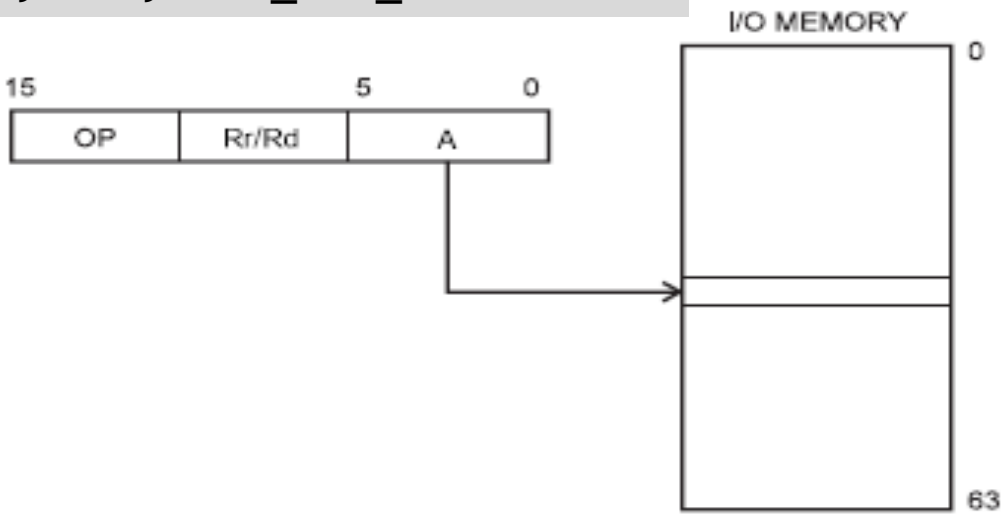


Direccionamiento de E/S directo

El operando es el contenido de un registro de E/S y en la instrucción (IN,OUT) aparece su número P (0 a 63) (en SBI,CBI solo 0 a 31)

Ejemplos:

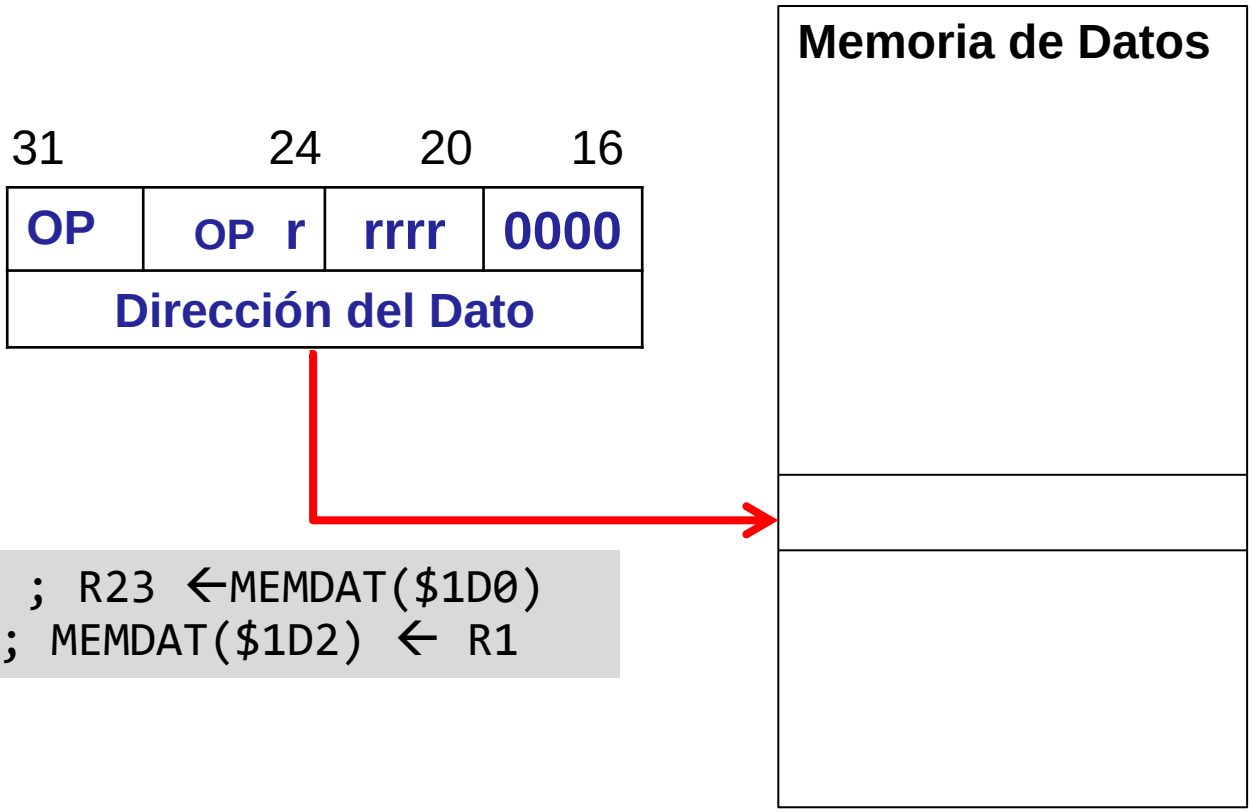
```
IN R1,56   ; R1 ← REG_E/S_56
OUT 21,R10 ; REG_E/S_21 ←R10
```



Direccionamiento directo a memoria

El operando es una palabra de la memoria de datos y en la instrucción aparece su dirección (16 bits).

Ejemplos:

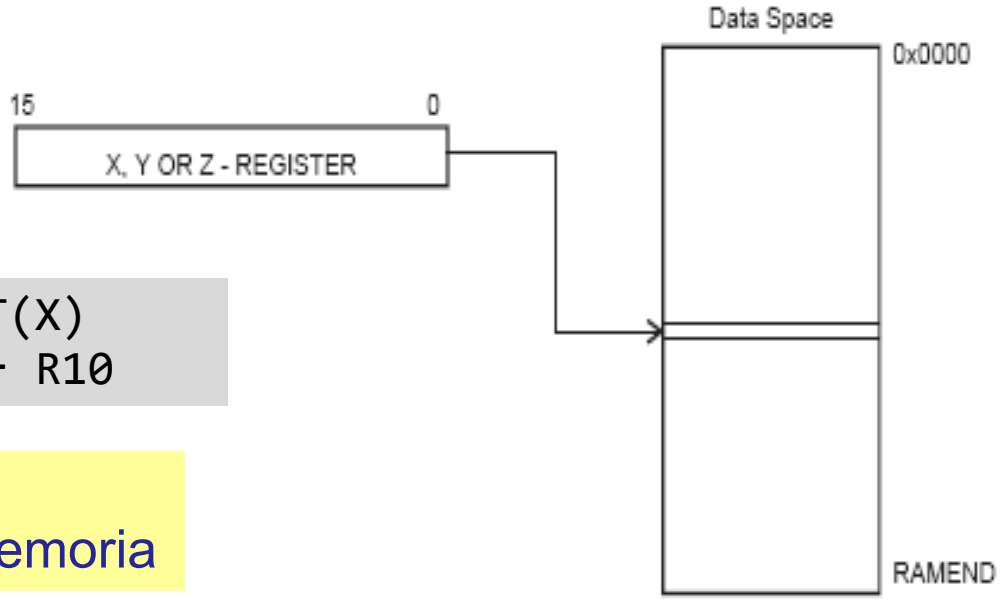



```
LDS R23,$1D0 ; R23 ←MEMDAT($1D0)
STS $1D2,R1 ; MEMDAT($1D2) ← R1
```

Direccionamiento indirecto

El operando es una palabra de la memoria de datos y en la instrucción aparece el registro que contiene su dirección (X,Y o Z).

Ejemplos:



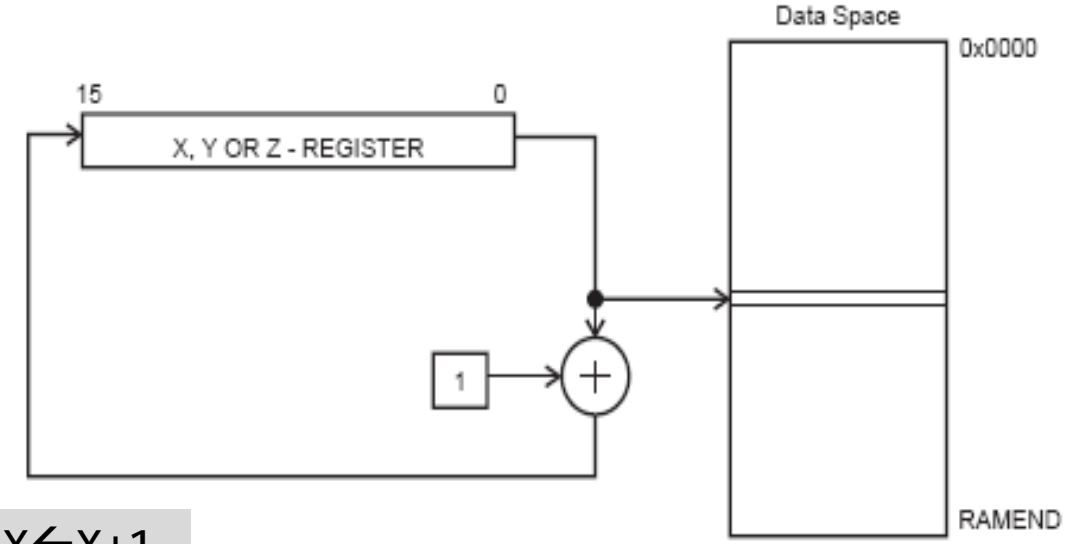

```
LD R1,X     ;R1 ← MEMDAT(X)
ST Z,R10    ;MEMDAT(Z) ← R10
```

Los registros X, Y y Z actúan como “punteros” al dato en memoria

Direccionamiento indirecto con postincremento

El operando es una palabra de la memoria de datos y en la instrucción aparece el registro que contiene su dirección (X,Y o Z), esta dirección se incrementa en uno tras el acceso al dato.

Ejemplos:



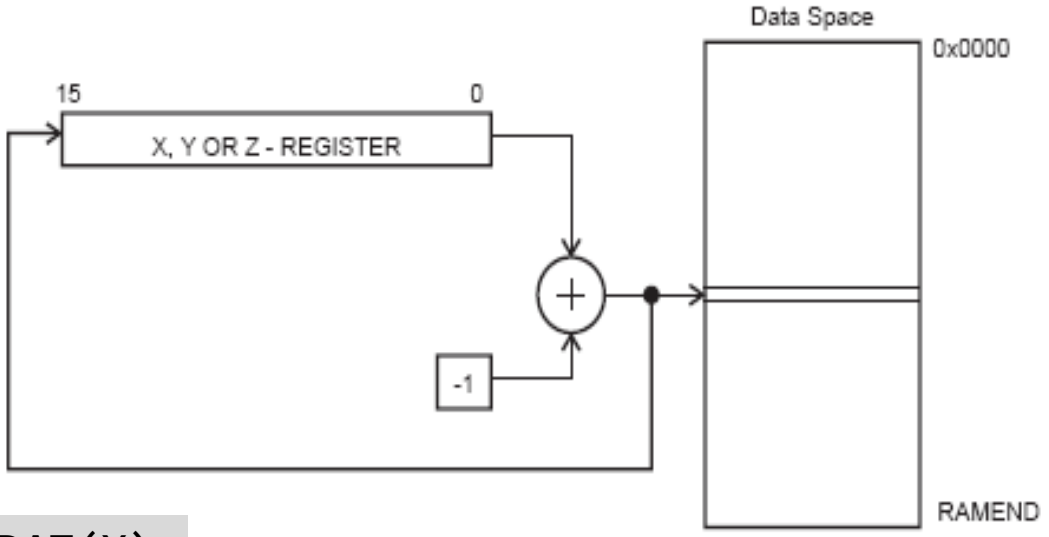

```
LD RO,X+ ; R0←MEMDAT(X),X←X+1
ST Z+,R1 ; MEMDAT(Z)←R1,Z←Z+1
```

Útil para recorrer los datos de una tabla

Direccionamiento indirecto con predecremento

El operando es una palabra de la memoria de datos y en la instrucción aparece el registro que, tras decrementarse, contiene su dirección (X,Y o Z).

Ejemplos:




```
LD RO,-X ; X←X-1, R0←MEMDAT(X)
ST -Z,R1 ; Z←Z-1, MEMDAT(Z)←R1
```

Útil para recorrer los datos de una tabla en orden inverso

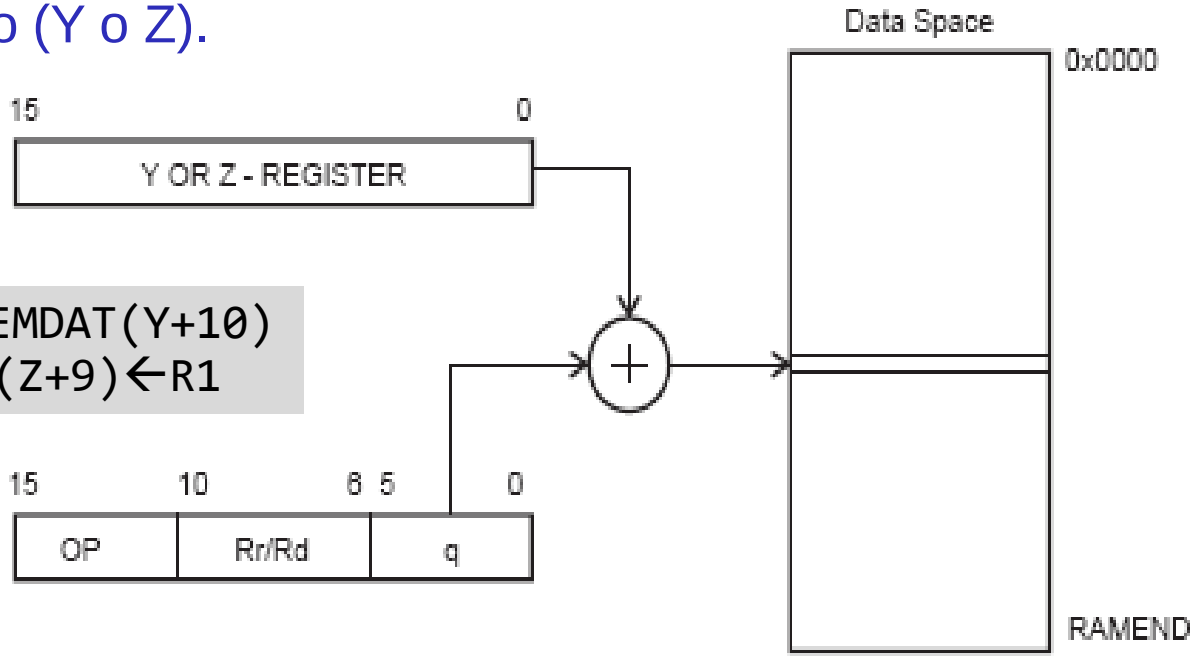
Direccionamiento indirecto con desplazamiento

El operando es una palabra de la memoria de datos y en la instrucción aparece la suma “Y+q” o “Z+q”.

La dirección del dato se obtiene sumando el desplazamiento *q* (6 bits: 0<q<64) a la dirección contenida en el registro Y o Z. El resultado no se actualiza en el registro (Y o Z).

Ejemplos:

```
LDD R0,Y+10; R0 ←MEMDAT(Y+10)
STD Z+9,r1 ; MEMDAT(Z+9)←R1
```



Direccionamiento inmediato

El operando es una constante de 8 bits que está contenido en la propia instrucción.

Puede ser un dato sin signo o un dato con signo expresado en Ca2.

Ejemplos:

```
LDI R16,255    ; R16 ←255
ANDI R25,0X10  ; R25 ← R25 & 0x10
```

```
LDI R16,-1  es lo mismo que LDI R16,255
```

Modos de direccionamiento a la memoria de programa

- Constantes de programa (no las trataremos)
  - Indirecto
  - Indirecto con desplazamiento

- **Instrucciones**
  - **Directo**
  - **Indirecto**
  - **Relativo**

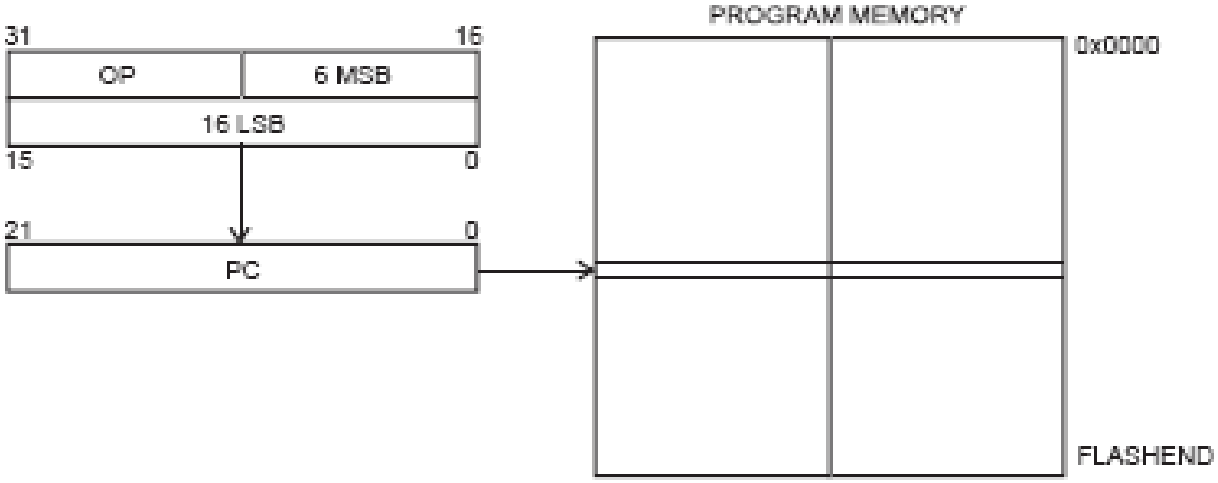
Direccionamiento directo de instrucciones

El programa continúa su ejecución en la dirección de memoria de programa indicada en la instrucción (JMP y CALL lo usan)

Ejemplos:

```
; Salto a la dirección 0x3F0:
    JMP 0x3F0 ; PC← 0x3F0
```

```
; Llamada a la subrutina almacenada
; en 0x500 (retorna con un RET)
    CALL 0x500
```



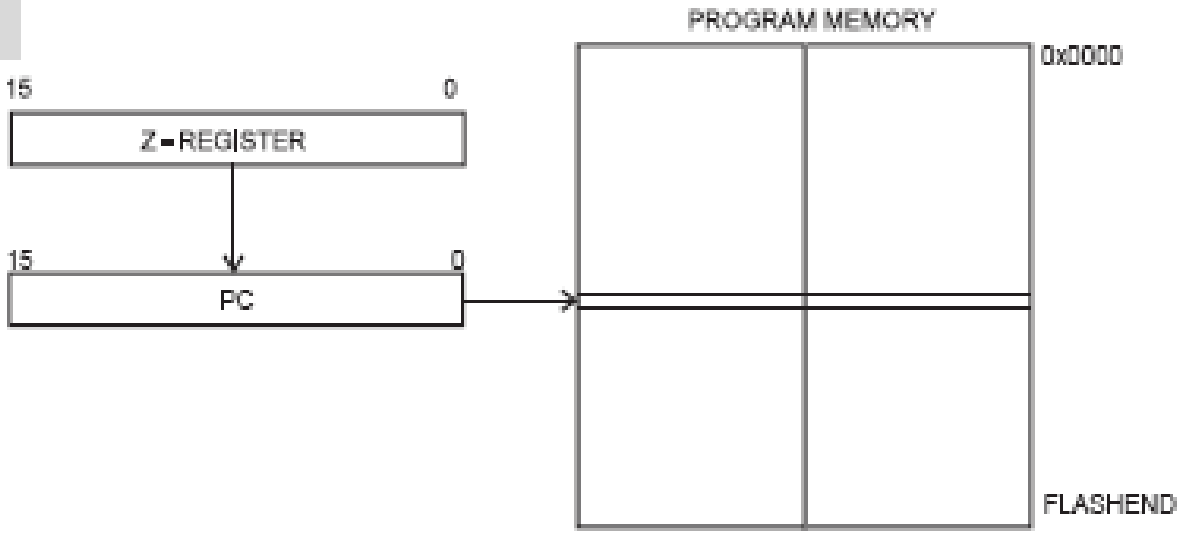
Direccionamiento indirecto de instrucciones

El programa continúa su ejecución en la dirección de memoria almacenada en el registro Z (IJMP e ICALL lo usan)

Ejemplos:

```
; Salto a la dirección
; almacenada en Reg. Z
    IJMP ; PC←Z
```

```
; Llamada a la subrutina almacenada
; en Reg. Z (retorna con un RET)
    ICALL
```



Direccionamiento relativo de instrucciones

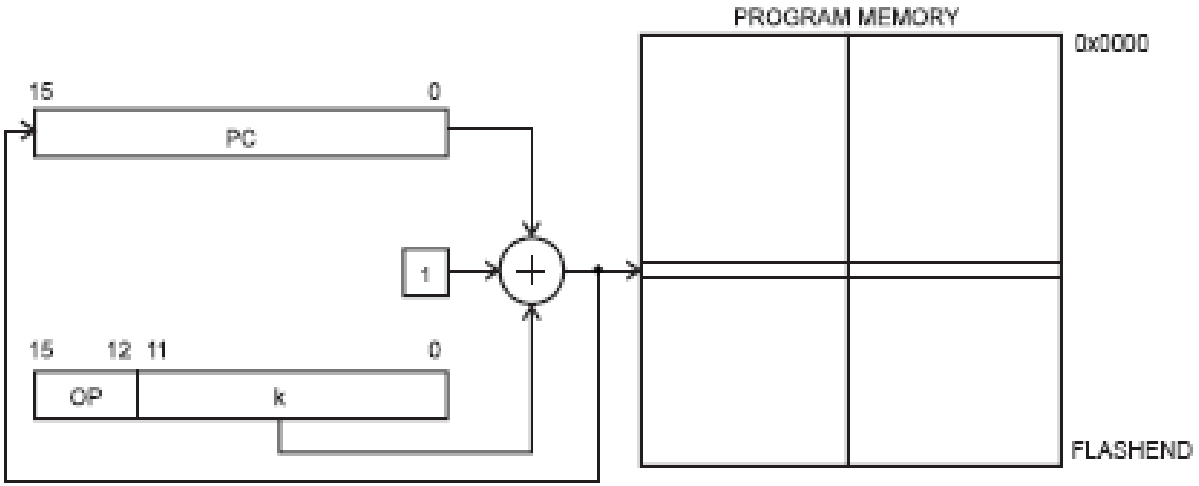
El programa continúa su ejecución en la dirección de memoria PC ← PC+K+1 (RJMP, RCALL, BRxx lo usan).

El programador no tiene por qué conocer el valor de K cuando usa estas instrucciones, este es calculado por el ensamblador.

Rango de valores de K:

RJMP y RCALL:
[-2048,+2047]

BRxx:
[-64,+63]



Índice

Tipos de instrucciones

- Instrucciones de **transferencia de datos**
- Instrucciones **aritmético-lógicas**
- Instrucciones de **salto**
- Instrucciones de **manejo de bits**
- Instrucciones de **control del sistema**

Tipos de instrucciones

- Instrucciones sin operandos
  - Mnemónico
- Instrucciones con un operando
  - Mnemónico operando
- Instrucciones con dos operandos
  - Mnemónico opdestino,opfuente
- Para cada instrucción se presentará la siguiente información

Mnemónico	Operandos	Descripción	Rango	Operación	Banderines	Ciclos de reloj

Instrucciones de transferencia de datos

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
MOV	Rd,Rr	Copiar registro	d,r∈[0,31]	Rd←Rr	Ninguno	1
MOVW	Rd,Rr	Copiar registro W	d,r∈ {0,2,4,...30}	Rd+1:Rd←Rr+1:Rr	Ninguno	1
LDI	Rd,k	Cargar dato inmediato	d∈ [16,31] k∈ [0,255]	Rd← k	Ninguno	1
LDS	Rd,k	Cargar dato desde la memoria	d∈ [0,31] k∈ [0,65535]	Rd← (k)	Ninguno	2
LD	Rd,X Rd,X+ Rd,-X Rd,Y Rd,Y+ Rd,-Y Rd,Z Rd,Z+ Rd,-Z	Carga el registro con un dato indirecto	d∈ [0,31]	Rd← (X) Rd← (X); X←X+1 X←X-1; Rd← (X) Rd← (Y) Rd← (Y); Y←Y+1 Y←Y-1; Rd← (Y) Rd← (Z) Rd← (Z); Z←Z+1 Z←Z-1; Rd← (Z)	Ninguno	2
LDD	Rd,Y+q Rd,Z+q	Carga el registro con un dato indirecto con desplazamiento	d∈ [0,31] q∈ [0,63], q≥0	Rd← (Y+q) Rd← (Z+q)	Ninguno	2

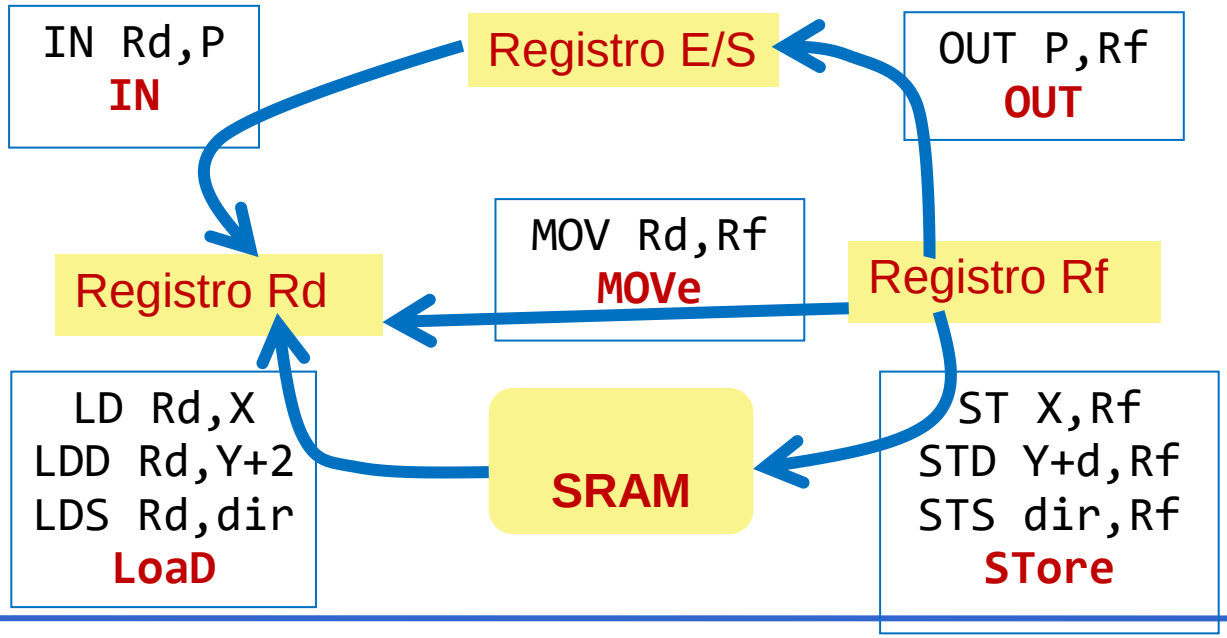
Instrucciones de transferencia de datos

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
STS	k, Rr	Almacenar dato en memoria	r∈[0,31] k∈[0,65535]	(k)← Rr	Ninguno	2
ST	X,Rr X+,Rr -X,Rr Y,Rr Y+,Rr -Y,Rr Z,Rr Z+,Rr -Z,Rr	Almacenar registro en memoria	r∈[0,31]	(X)← Rr (X)← Rr; X←X+1 X←X-1; (X)←Rr (Y)←Rr (Y)← Rr; Y←Y+1 Y←Y-1; (Y)← Rr (Z)← Rr (Z)←Rr; Z←Z+1 Z←Z-1; (Z)←Rr	Ninguno	2
STD	Y+q,Rr Z+q,Rr	Almacenar registro en memoria con indirecto con desplazamiento	r∈[0,31] q∈[0,63], q≥0	(Y+q)←Rr (Z+q)← Rr	Ninguno	2
LPM	Rd,Z Rd,Z+	Carga memoria de programa Z es puntero a byte		R0← (Z) Rd← (Z) Rd← (Z); Z←Z+1	Ninguno	3
SPM		Almacenar en memoria de programa		(Z) ← R1:R0	Ninguno	-

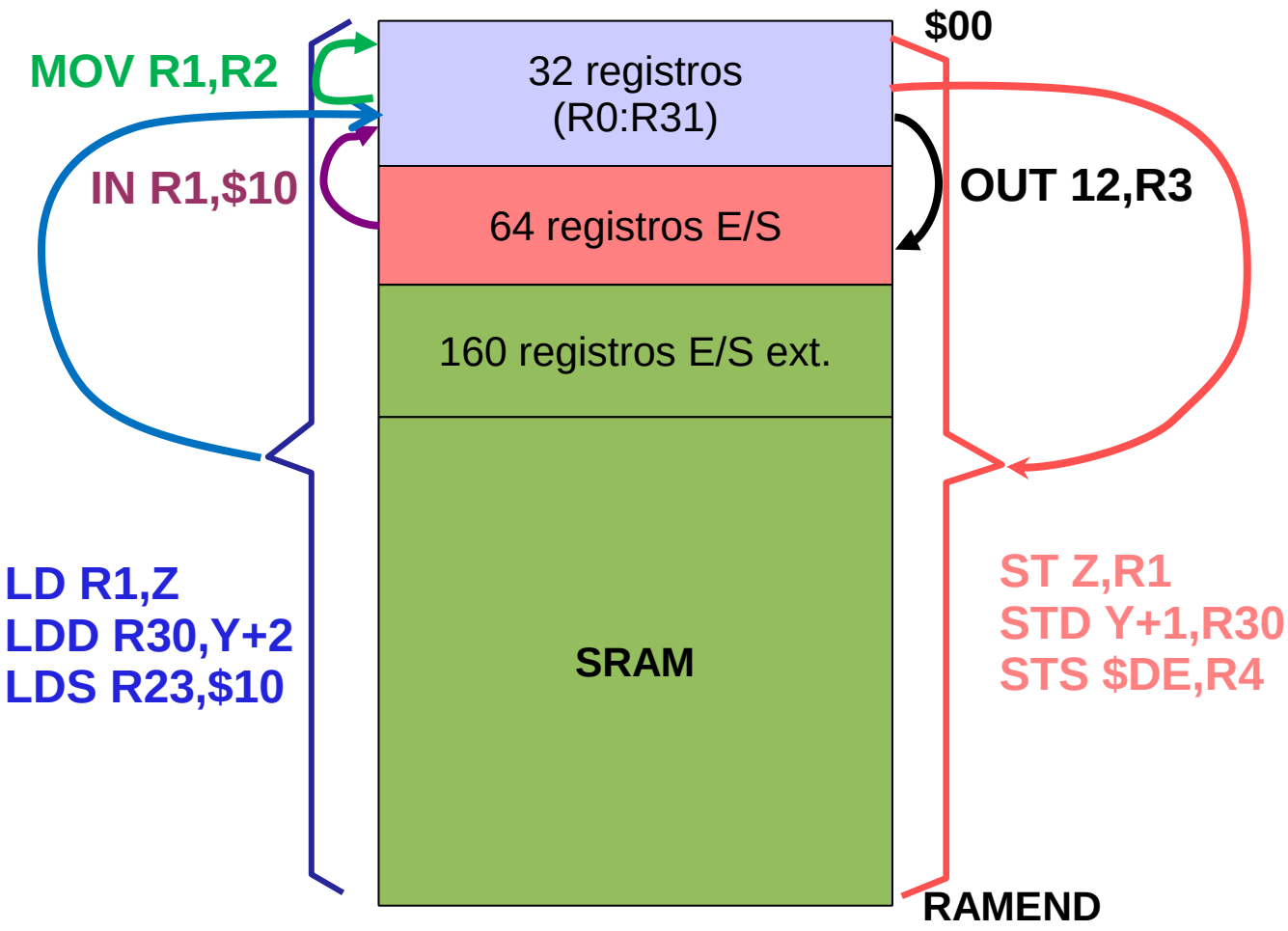
En la instr. LPM, el reg. Z apunta a bytes de la memoria de programa y no a palabras de 16 bits

Instrucciones de transferencia de datos

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
IN	Rd,I/O(A)	Entrada desde registro de I/O	d∈[0,31] A∈ [0,63]	Rd←I/O(A)	Ninguno	1
OUT	I/O(A),Rr	Salida hacia registro de I/O	r∈ [0,31] A∈ [0,63]	I/O(A)← Rr	Ninguno	1
PUSH	Rr	Escritura en pila	r∈ [0,31]	STACK←Rr	Ninguno	2
POP	Rd	Lectura de pila	d∈Î[0,31]	Rd← STACK	Ninguno	2



Instrucciones de transferencia de datos



Instrucciones aritmético-lógicas

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
ADD	Rd,Rr	Suma sin carry	d,r∈[0,31]	Rd← Rd+Rr	Z,N,V,C,H	1
ADC	Rd,Rr	Suma con carry	d,r∈[0,31]	Rd← Rd+Rr+C	Z,N,V,C,H	1
ADIW	Rd,K	Suma inmediato con palabra	d∈{24,26,28,30} K∈[0,63], K≥0	Rd+1:Rd← Rd+1:Rd + K	Z,N,V,C	2
SUB	Rd,Rr	Resta sin carry	d,r∈[0,31]	Rd←Rd-Rr	Z,N,V,C,H	1
SUBI	Rd,K	Resta inmediato	d∈[16,31] K∈[0,255]	Rd←Rd-K	Z,N,V,C,H	1
SBC	Rd,Rr	Resta con carry	d,r∈[0,31]	Rd←Rd-Rr-C	Z,N,V,C,H	1
SBCI	Rd,K	Resta inmediato con carry	d∈[16,31] K∈[0,255]	Rd←Rd-K-C	Z,N,V,C,H	1
SBIW	Rd,K	Resta inmediato con palabra	d∈{24,26,28,30} K∈[0,63], K≥0	Rd+1:Rd← Rd+1:Rd - K	Z,N,V,C	2
COM	Rd	Complemento a 1	d,r∈[0,31]	Rd←$FF-Rd	Z,N,V,C	1
NEG	Rd	Complemento a 2	d,r∈[0,31]	Rd←$00-Rd	Z,N,V,C	1
INC	Rd	Incrementa	d,r∈[0,31]	Rd←Rd+1	Z,N,V	1
DEC	Rd	Decrementa	d,r∈[0,31]	Rd←Rd-1	Z,N,V	1

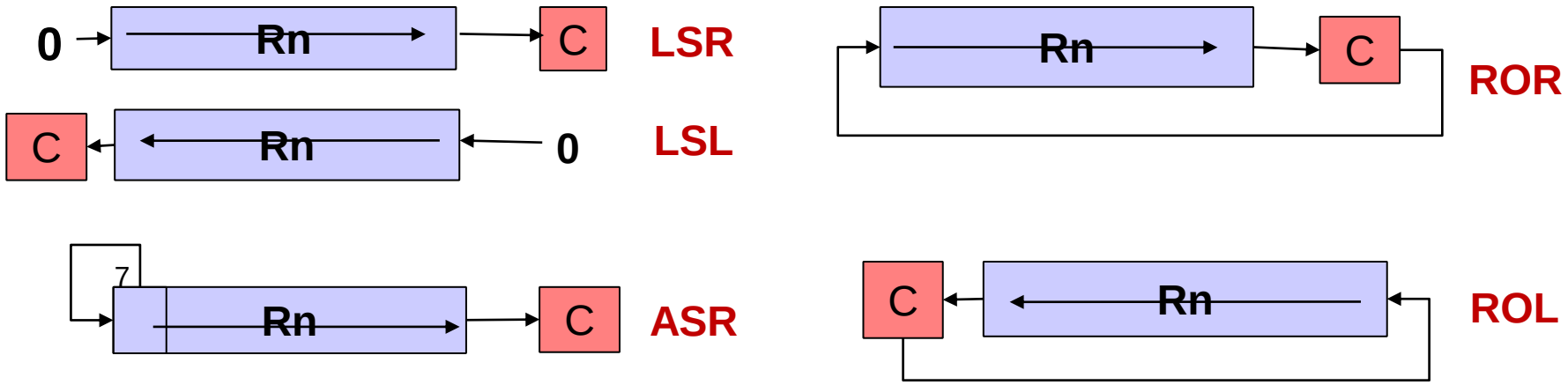
En los casos en que K ∈ [0,63], K es un número sin signo.
En los casos en que K ∈ [0,255], K puede ser un número con signo en Ca2 (rango [-128,+127])

Instrucciones aritmético-lógicas

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
AND	Rd,Rr	And lógica	d,r∈[0,31]	Rd←RdΛRr	Z,N,V	1
ANDI	Rd,K	And lógica con dato inmediato	d∈[16,31] K∈[0,255]	Rd←RdΛK	Z,N,V	1
OR	Rd,Rr	Or lógica	d,r∈[0,31]	Rd←Rd∨Rr	Z,N,V	1
ORI	Rd,K	Or lógica con dato inmediato	d∈[16,31] K∈[0,255]	Rd←Rd∨K	Z,N,V	1
EOR	Rd,Rr	Exclusive or	d,r∈[0,31]	Rd←Rd⊕Rr	Z,N,V	1
CLR	Rd	Poner a cero	d,r∈[0,31]	Rd←0	Z,N,V	1
SER	Rd	Poner todo a 1	d,r∈[16,31]	Rd←$FF	Z,N,V	1
CP	Rd,Rr	Compara	d,r∈[0,31]	Rd-Rr	Z,N,V,C,H	1
CPC	Rd,Rr	Compara con carry	d,r∈[0,31]	Rd-Rr-C	Z,N,V,C,H	1
CPI	Rd,K	Compara inmediato	d∈[16,31] K∈[0,255]	Rd-K	Z,N,V,C,H	1
MUL	Rd,Rr	Multiplica sin signo	d,r∈[0,31]	R1:R0←Rd x Rr	Z,C	2
MULS	Rd,Rr	Multiplica con signo	d,r∈[16,31]	R1:R0←Rd x Rr	Z,C	2
MULSU	Rd,Rr	Multiplica sin signo	d,r∈[16,23]	R1:R0←Rd x Rr (Rd signed Rr unsigned)	Z,C	2

Instrucciones aritmético-lógicas

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
LSL	Rd	Desplazamiento lógico a la izquierda (Logical Shift Left)	d∈[0,31]	Rd(n+1)←Rd(n), Rd(0)←0, C← Rd(7)	Z,C,N,V,H	1
LSR	Rd	Desplazamiento lógico a la derecha (Logical Shift Right)	d∈[0,31]	Rd(n)←Rd(n+1), Rd(7)←0, C← Rd(0)	Z,C,N,V	1
ROL	Rd	Rotación a la izquierda (Rotate Left through Carry)	d∈[0,31]	Rd(n+1)←Rd(n), Rd(0)←C, C← Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotación a la derecha (Rotate Right through Carry)	d∈[0,31]	Rd(n)←Rd(n+1), Rd(7)←C, C← Rd(0)	Z,C,N,V	1
ASR	Rd	Desplazamiento aritmético a la derecha (Arithmetic Shift Right)	d∈[0,31]	Rd(n)←Rd(n+1), Rd(7)←Rd(7), C← Rd(0)	Z,C,N,V	1



Instrucciones de salto

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
RJMP	dir	Salto relativo	K= dir - (PC+1) -2048≤ K <+2047	PC←dir	Ninguno	2
JMP	dir	Salto	0≤dir<4M	PC← dir	Ninguno	3
IJMP		Salto indirecto		PC ← Z	Ninguno	2
RCALL	dir	Llamada a subrutina relativa	K= dir - (PC+1) -2048≤ K<+2047	STACK ← PC+1 PC←dir	Ninguno	3
CALL	dir	Llamada a subrutina	0≤dir<16K	STACK ← PC+2 PC← dir	Ninguno	4
ICALL		Llamada a subrutina indirecta		STACK ← PC+1 PC ← Z	Ninguno	3
RET		Regreso de subrutina		PC← STACK	Ninguno	4
RETI		Regreso de rutina de interrupción		PC← STACK	I	4
CPSE	Rd,Rr	Compara, esquiva si iguales	d,r∈[0,31]	Si Rd=Rr PC← PC+2 (o 3)	Ninguno	1 o 2 o 3
SBRC	Rr,b	Esquiva si el bit b de Rr está a 0	r∈[0,31] b∈[0,7]	Si (Rr(b)=0) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3
SBRS	Rr,b	Esquiva si el bit b de Rr está a 1	r∈[0,31] b∈[0,7]	Si (Rr(b)=1) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3
SBIC	A,b	Esquiva si el bit b del registro I/O está a 0	A∈[0,31] b∈[0,7]	Si (I/O(A,b)=0) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3
SBIS	A,b	Esquiva si el bit b del registro I/O está a 1	A∈[0,31] b∈[0,7]	Si (I/O(A,b)=1) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3

Instrucciones de salto

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
BREQ	dir	Salta si iguales	K= dir - (PC+1) -64≤ K<+63	Si (Z=1) PC← dir	Ninguno	1 o 2
BRNE	dir	Salta si distintos	K= dir - (PC+1) -64≤ K<+63	Si (Z=0) PC← dir	Ninguno	1 o 2
BRCS	dir	Salta si C está a 1	K= dir - (PC+1) -64≤ K<+63	Si (C=1) PC← dir	Ninguno	1 o 2
BRCC	dir	Salta si C está a 0	K= dir - (PC+1) -64≤ K<+63	Si (C=0) PC← dir	Ninguno	1 o 2
BRLO	dir	Salta si menor, datos sin signo (C=0)	K= dir - (PC+1) -64≤ K<+63	Si (C=1) PC← dir	Ninguno	1 o 2
BRSH	dir	Salta si igual o mayor, datos sin signo (C=1)	K= dir - (PC+1) -64≤ K<+63	Si (C=0) PC← dir	Ninguno	1 o 2
BRMI	dir	Salta si negativo (N=1)	K= dir - (PC+1) -64≤ K<+63	Si (N=1) PC← dir	Ninguno	1 o 2
BRPL	dir	Salta si positivo (N=0)	K= dir - (PC+1) -64≤ K<+63	Si (N=0) PC← dir	Ninguno	1 o 2
BRHS	dir	Salta si H está a 1	K= dir - (PC+1) -64≤ K<+63	Si (H=1) PC← dir	Ninguno	1 o 2
BRHC	dir	Salta si H está a 0	K= dir - (PC+1) -64≤ K<+63	Si (H=0) PC← dir	Ninguno	1 o 2
BRTS	dir	Salta si T está a 1	K= dir - (PC+1) -64≤ K<+63	Si (T=1) PC← dir	Ninguno	1 o 2
BRTC	dir	Salta si T está a 0	K= dir - (PC+1) -64≤ K<+63	Si (T=0) PC← dir	Ninguno	1 o 2

Instrucciones de salto

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
BRVS	dir	Salta si V está a 1	K= dir - (PC+1) -64≤ K<+63	Si (V=1) PC← dir	Ninguno	1 o 2
BRVC	dir	Salta si V está a 0	K= dir - (PC+1) -64≤ K<+63	Si (V=0) PC← dir	Ninguno	1 o 2
BRIE	dir	Salta si I está a 1	K= dir - (PC+1) -64≤ K<+63	Si (I=1) PC← dir	Ninguno	1 o 2
BRID	dir	Salta si I está a 0	K= dir - (PC+1) -64≤ K<+63	Si (I=0) PC← dir	Ninguno	1 o 2
BRGE	dir	Salta si mayor o igual, datos con signo (S=0)	K= dir - (PC+1) -64≤ K<+63	Si (N⊕V=0) PC← dir	Ninguno	1 o 2
BRLT	dir	Salta si menor, datos con signo (S=1)	K= dir - (PC+1) -64≤ K<+63	Si (N⊕V=1) PC← dir	Ninguno	1 o 2

Instrucciones de salto

Test (CP Rd,Rr)	Booleana	Mnemonico	Comentario
Rd ≥Rr	(N ⊕ V) = 0	BRGE	Signo
Rd < Rr	(N ⊕ V) = 1	BRLT	Signo
Rd = Rr	Z = 1	BREQ	Signo/Sin signo
Rd ≠ Rr	Z = 0	BRNE	Signo/Sin signo
Rd ≥ Rr	C = 0	BRCC/BRSH	Sin signo
Rd < Rr	C = 1	BRCS/BRLO	Sin signo
Carry	C=1	BRCS	Simple
Sin carry	C=0	BRCC	Simple
Negativo	N=1	BRMI	Simple
Positivo	N=0	BRPL	Simple
Overflow	V=1	BRVS	Simple
Sin overflow	V=0	BRVC	Simple
Cero	Z=1	BREQ	Simple
No cero	Z=0	BRNE	Simple

Instrucciones de manejo de bits y de control del sistema

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
SWAP	Rd	Intercambia nibbles	d∈[0,31]	Rd(3..0)←→Rd(7.4)	Ninguno	1
SBI	A,b	Poner a 1 el bit b del registro de I/O	b∈[0,7] A∈[0,31]	I/O(A,b)←1	Ninguno	2
CBI	A,b	Poner a 0 el bit b del registro de I/O	b∈[0,7] A∈[0,31]	I/O(A,b)←0	Ninguno	2
SEcc	(Ver Nota)	Poner a 1 el bit cc del registro de estado			cc	1
CLcc	(Ver Nota)	Poner a o el bit cc del registro de estado			cc	1

cc= C,N,T,Z,I,V,H,S

Las instrucciones SEcc y CLcc no existen como tal sino que dan lugar a SEC,SEN,SET,…,SEH,SES,CLC,CLN,CLT,…CLH,CLS

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
NOP		Nada			Ninguno	1
BREAK		Para depuración			Ninguno	N/A
WDR		Reinicia el temporizador del perro guardián			Ninguno	1
SLEEP		Dormir			Ninguno	1

Índice

Directivas de ensamblador

- Son comandos al programa que genera el código objeto.

- Se encuentran mezclados con las instrucciones del microcontrolador en el fichero fuente.

- No dan lugar a código máquina, solo ayudan a generarlo adecuadamente.

Veremos solo algunas directivas:

CSEG-Code Segment

Sintaxis: `.CSEG`

DSEG-Data Segment

Sintaxis: `.DSEG`

Estas directivas permiten especificar si las líneas siguientes se refieren a la memoria de programa o a la de datos.

Directivas de ensamblador

BYTE – Reserva bytes para una variable en memoria

Reserva espacio en memoria de datos. Posible solo en DSEG

Sintaxis: label: .BYTE expresión

```
Var1:   .BYTE      1
Tabla: .BYTE     10
```

DEF – Asigna un nombre simbólico a un registro.

Sintaxis: .DEF symbol=register

```
.DEF temp = r16
.DEF ior= r0
```

EQU – Símbolo igual a expresión

Sintaxis: .EQU label = expression

```
.EQU   puertas = 2
```

Directivas de ensamblador

ORG – Establece la dirección inicial de memoria

Sintaxis: .ORG expression

```
          .DSEG         ;comienza el segmento de datos
          .ORG  0X37    ;Dirección $37 de la memoria de datos
                        ;Si no se indica, por defecto es $100
Variable: .BYTE 1
          .CSEG         ;comienza el segmento de código
          .ORG  0x10    ;dirección $10 de la mem. de programa
          mov   r0,r1
```

Constantes y expresiones de ensamblador

Constantes – Pueden darse en diferentes formatos

Decimal (default): 10, 255

Hexadecimal (dos opciones): 0x0a, $0a, 0xff, $ff

Binario: 0b00001010, 0b11111111

Octal (0 como prefijo): 010, 077

Funciones – Pueden darse en diferentes formatos

low(expresion): devuelve el byte inferior de una expresión

high(expresion): devuelve el byte superior de una expresión

Índice

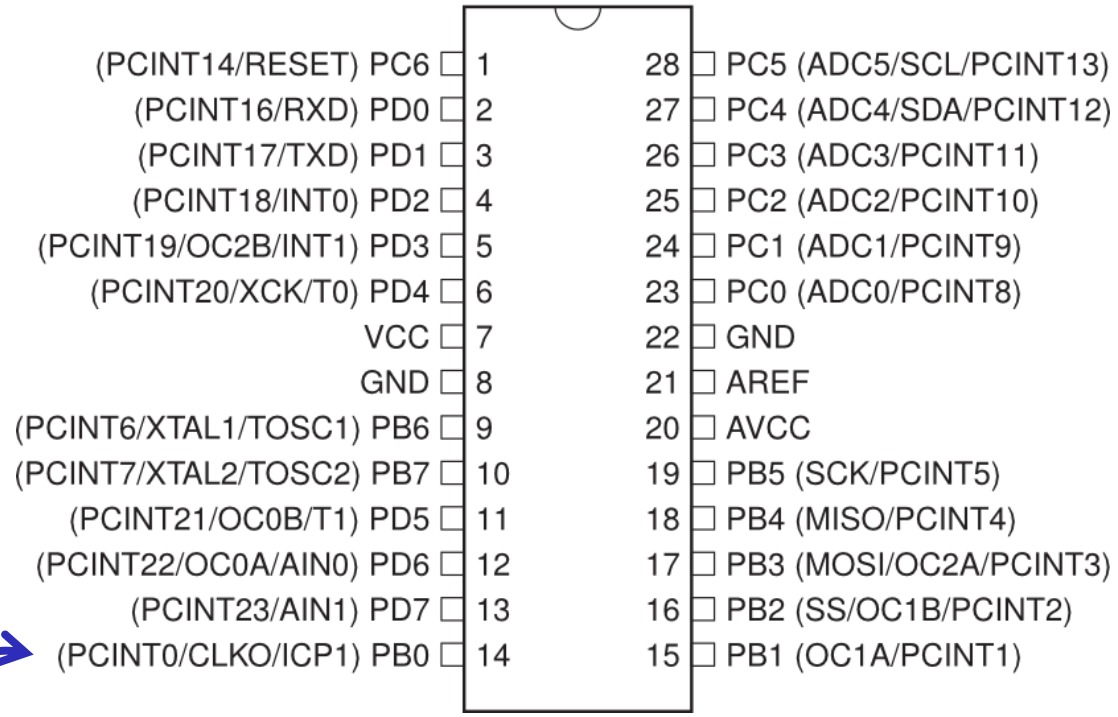
Puertos B,C y D

Permiten la comunicación con el exterior, se pueden utilizar como entrada o como salida.

Hay tres grupos: PB_{7-0}, PC_{6-0} y PD_{7-0}

Son **entradas/salidas** digitales que permiten **leer/escribir** valores lógicos en los pines. Típicamente: "1"=5V, "0" =0V

Cada **pin** puede tener **varias funciones (estudiaremos solo entrada/salida digitales** básicas**)**



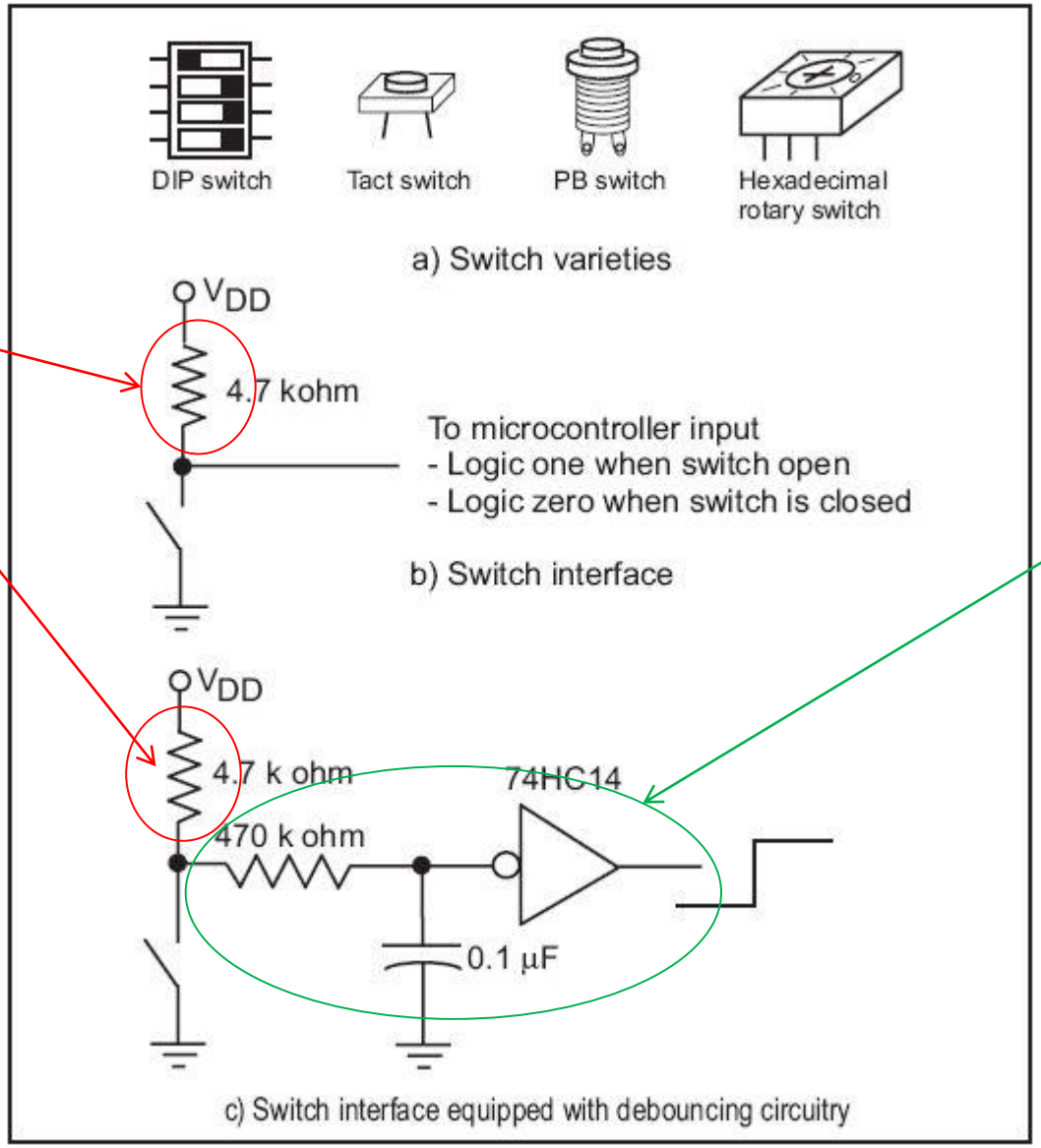
Activar la función alternativa de un PIN **no afecta** al resto del pines del puerto

Funciones alternativas. **Ejemplo**:
PB0: "Capturador de eventos del temporizador" o "Salida de reloj interno" o "Interrupción 0 ante cambio en el PIN"

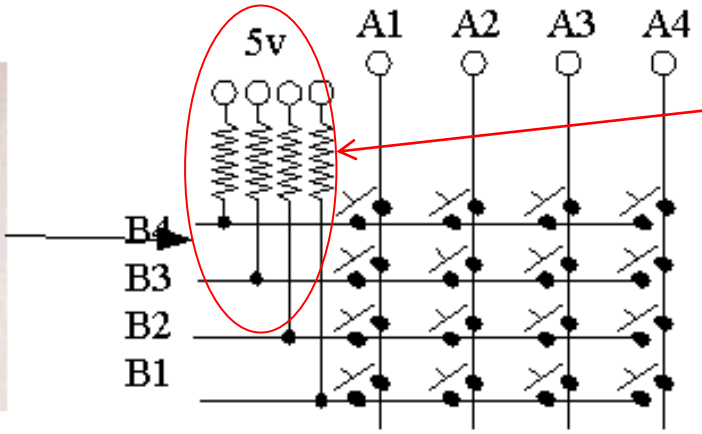
Ejemplos de dispositivos de entrada

Resistencia de pull-up

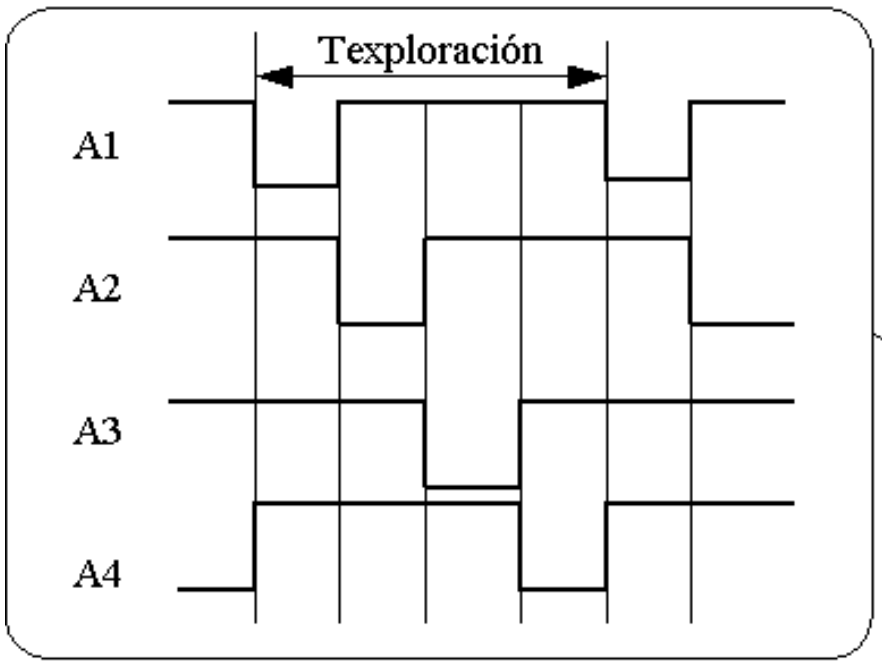
Eliminador de rebotes



Ejemplos de dispositivos de entrada



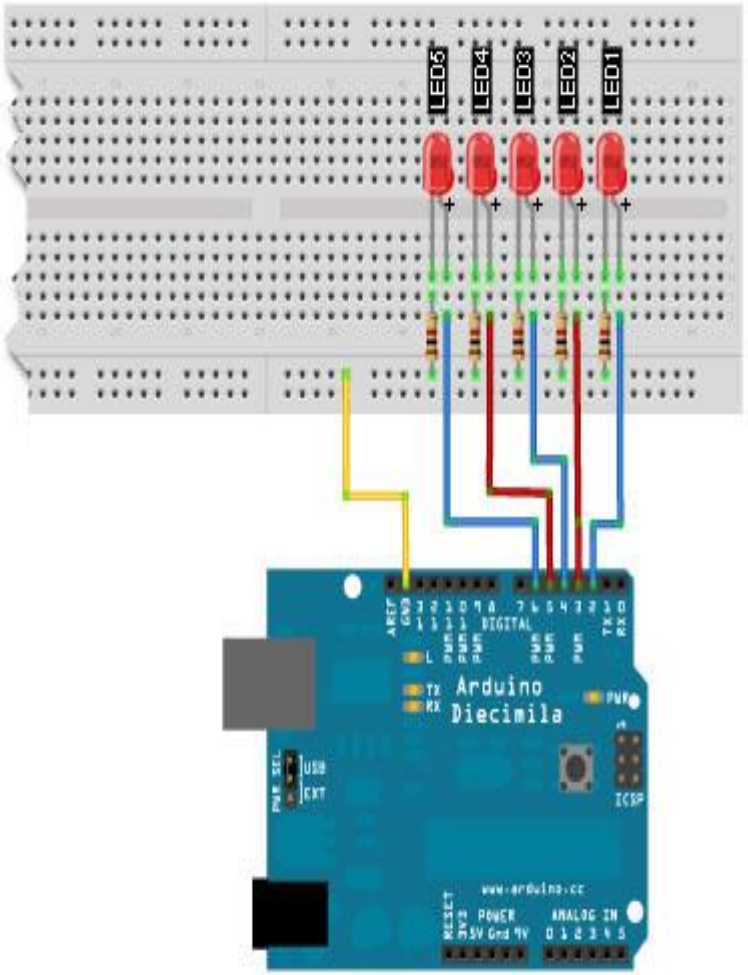
Resistencias de pull-up

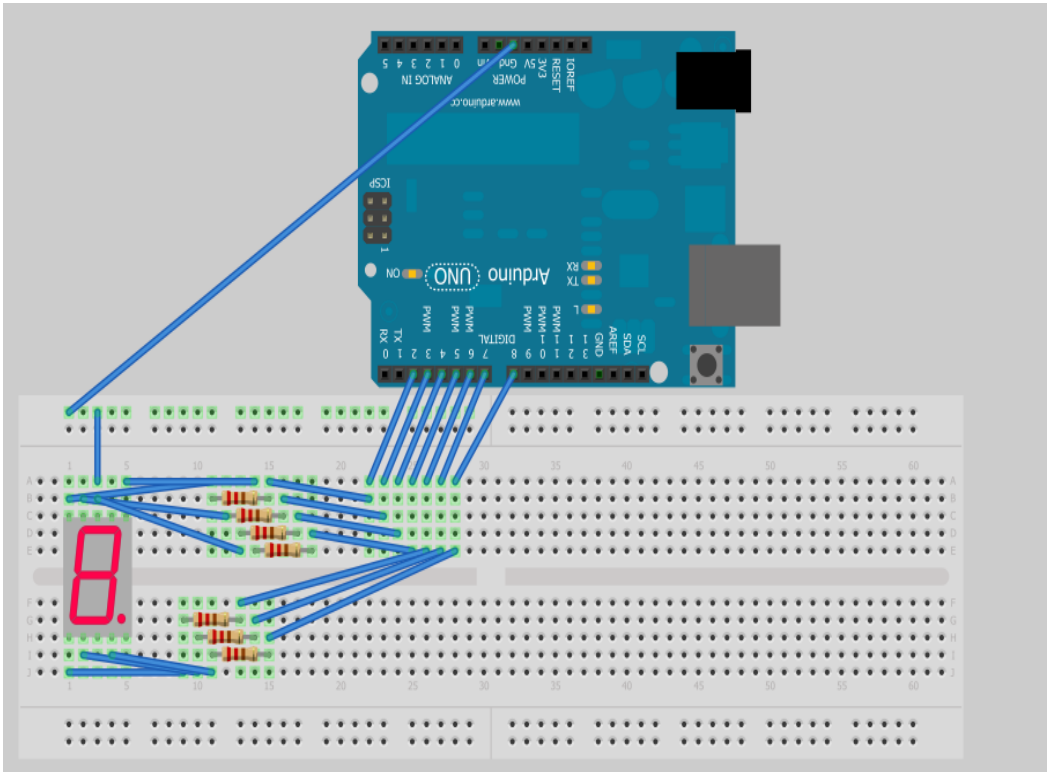


Secuencia de exploración

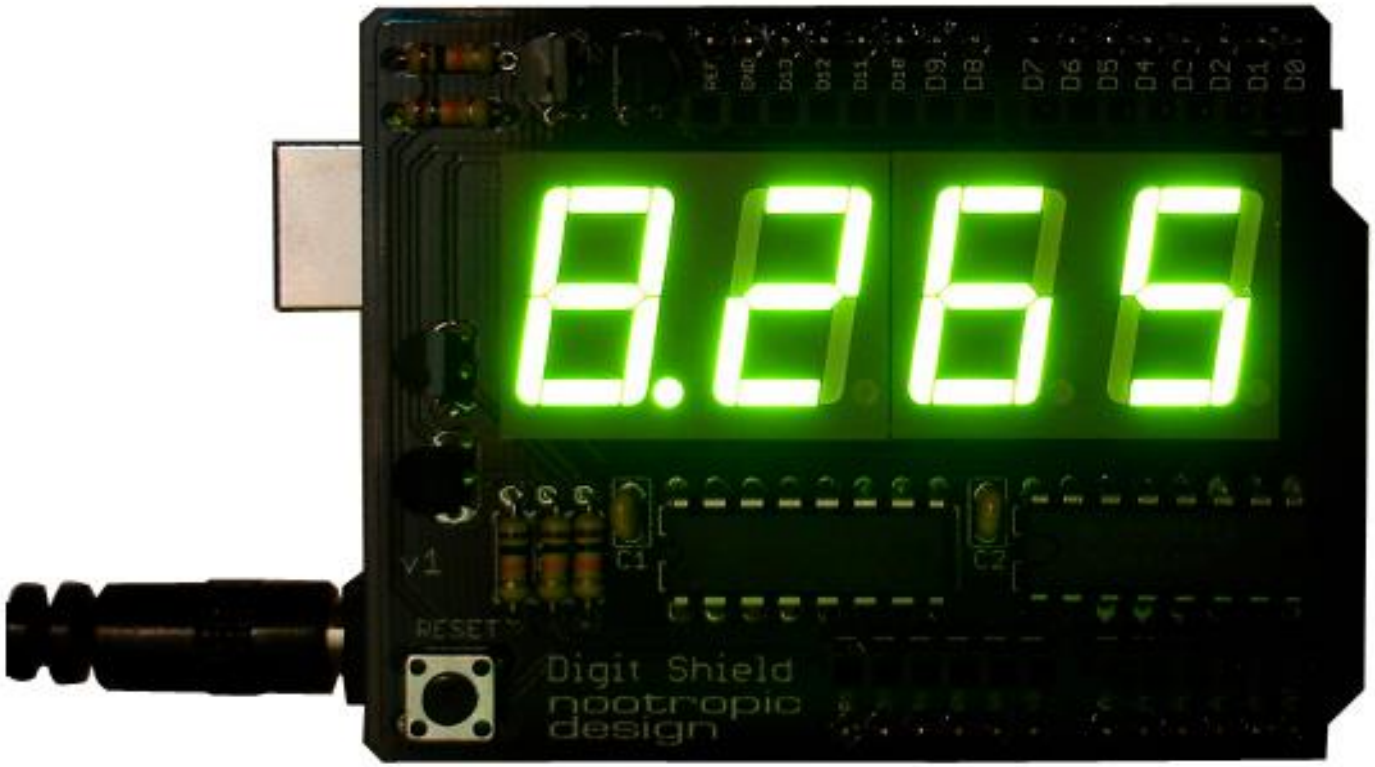
Teclado matricial:
con 8 líneas (B1, B2, B3, B4 de entrada y A1, A2, A3 y A4 de salida), leemos 16 pulsadores.

Ejemplos de dispositivos de salida

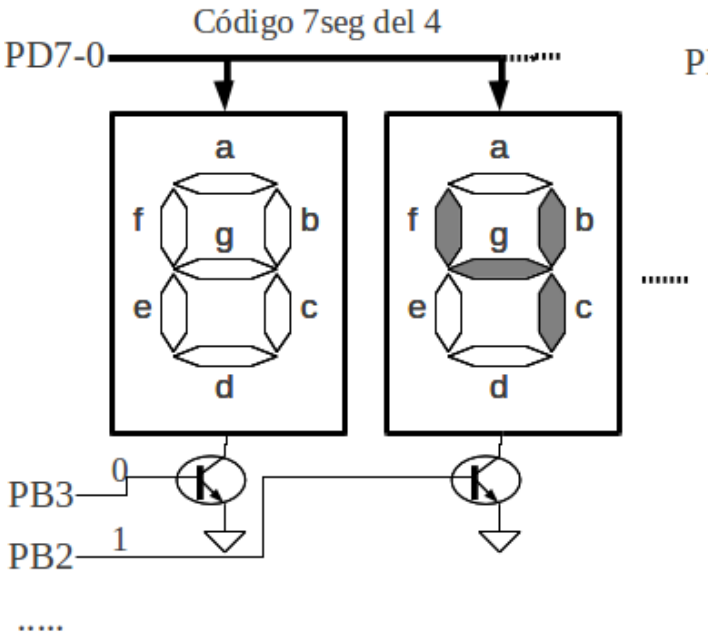


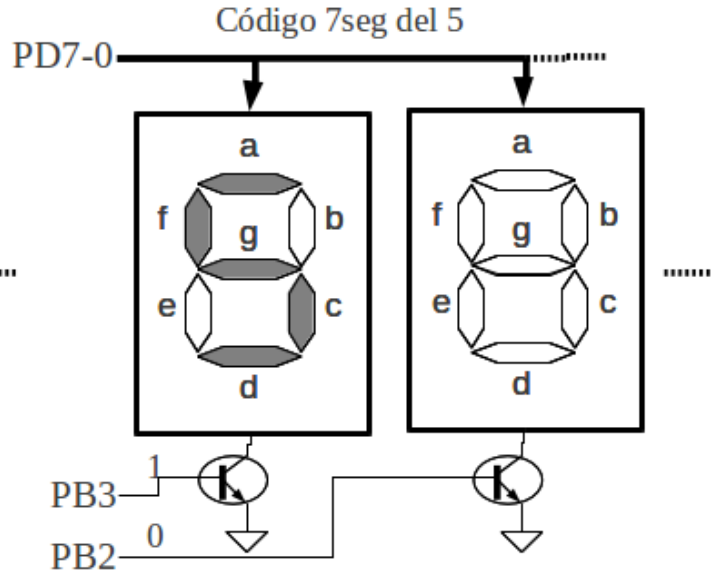


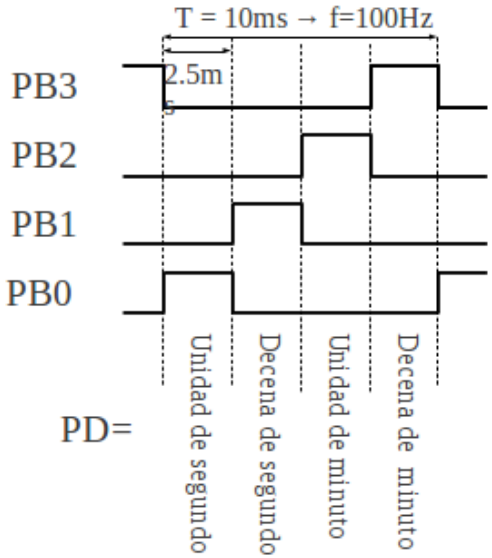
Ejemplos de dispositivos de salida

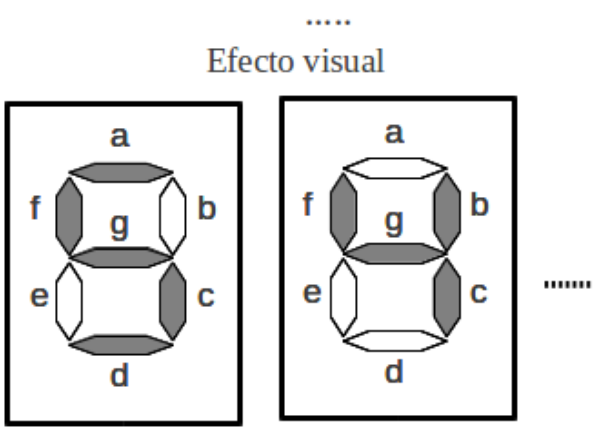


Ejemplos de dispositivos de salida

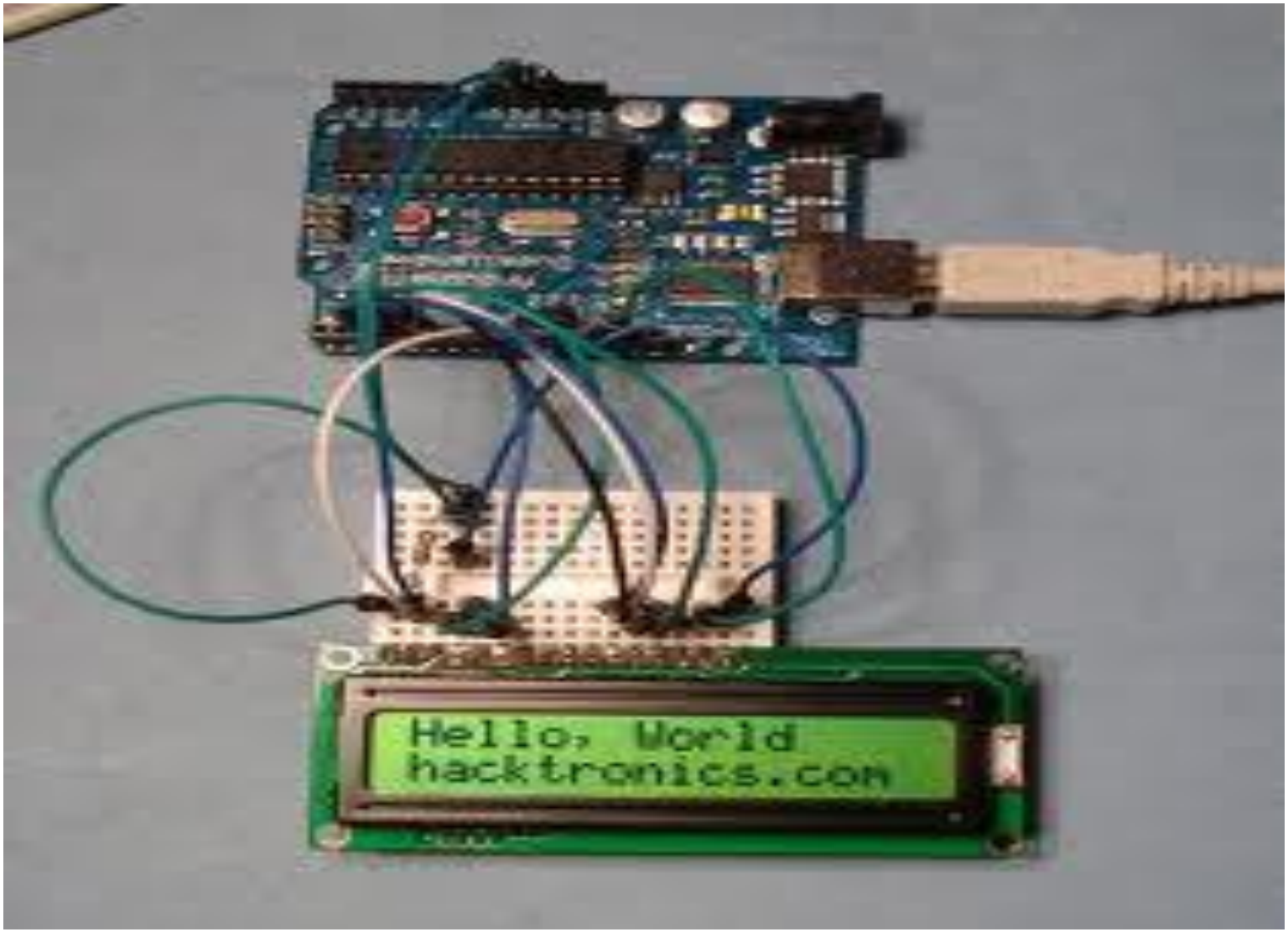








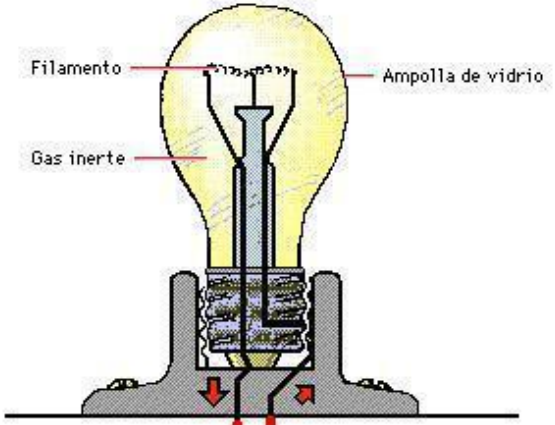
Ejemplos de dispositivos de salida

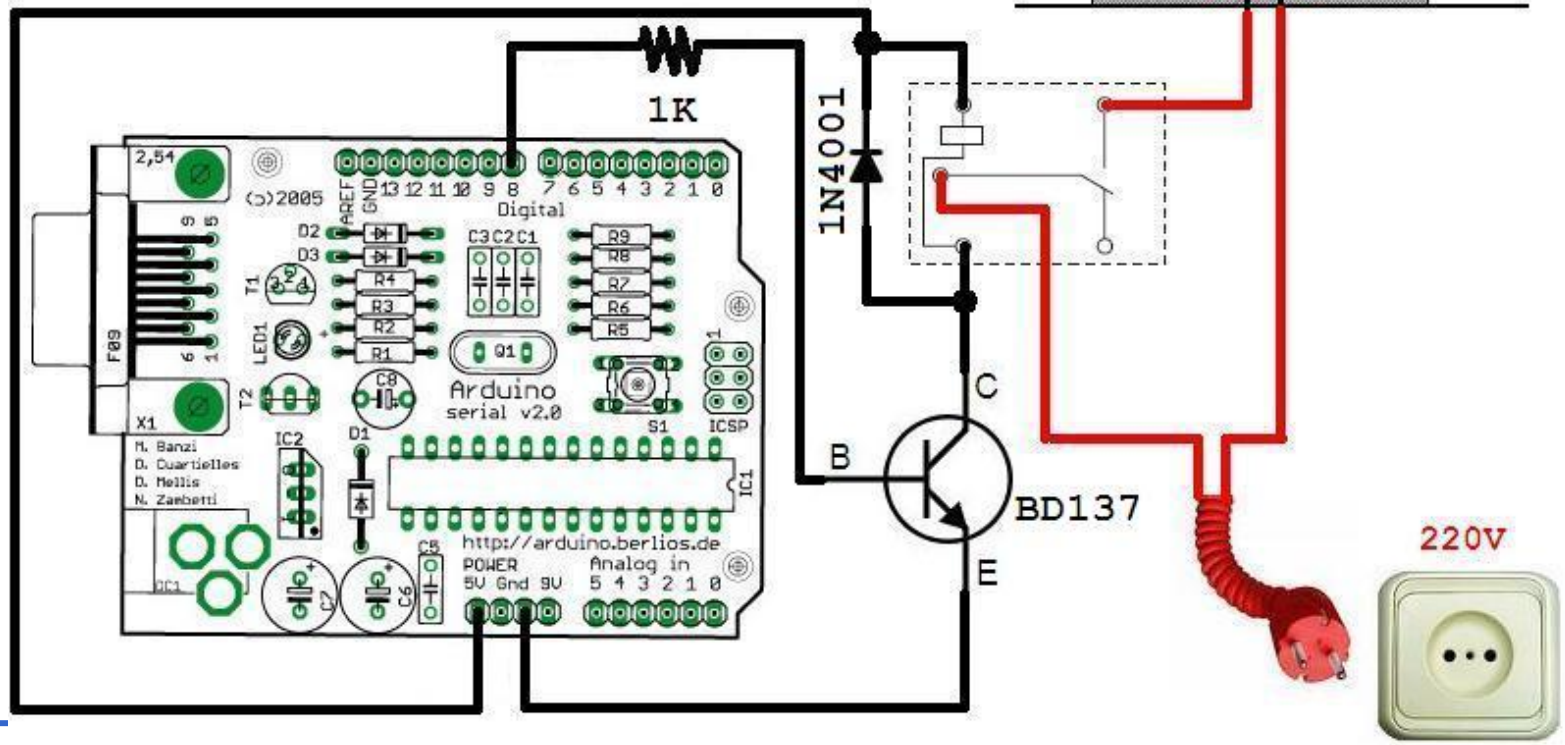


Ejemplos de dispositivos de salida

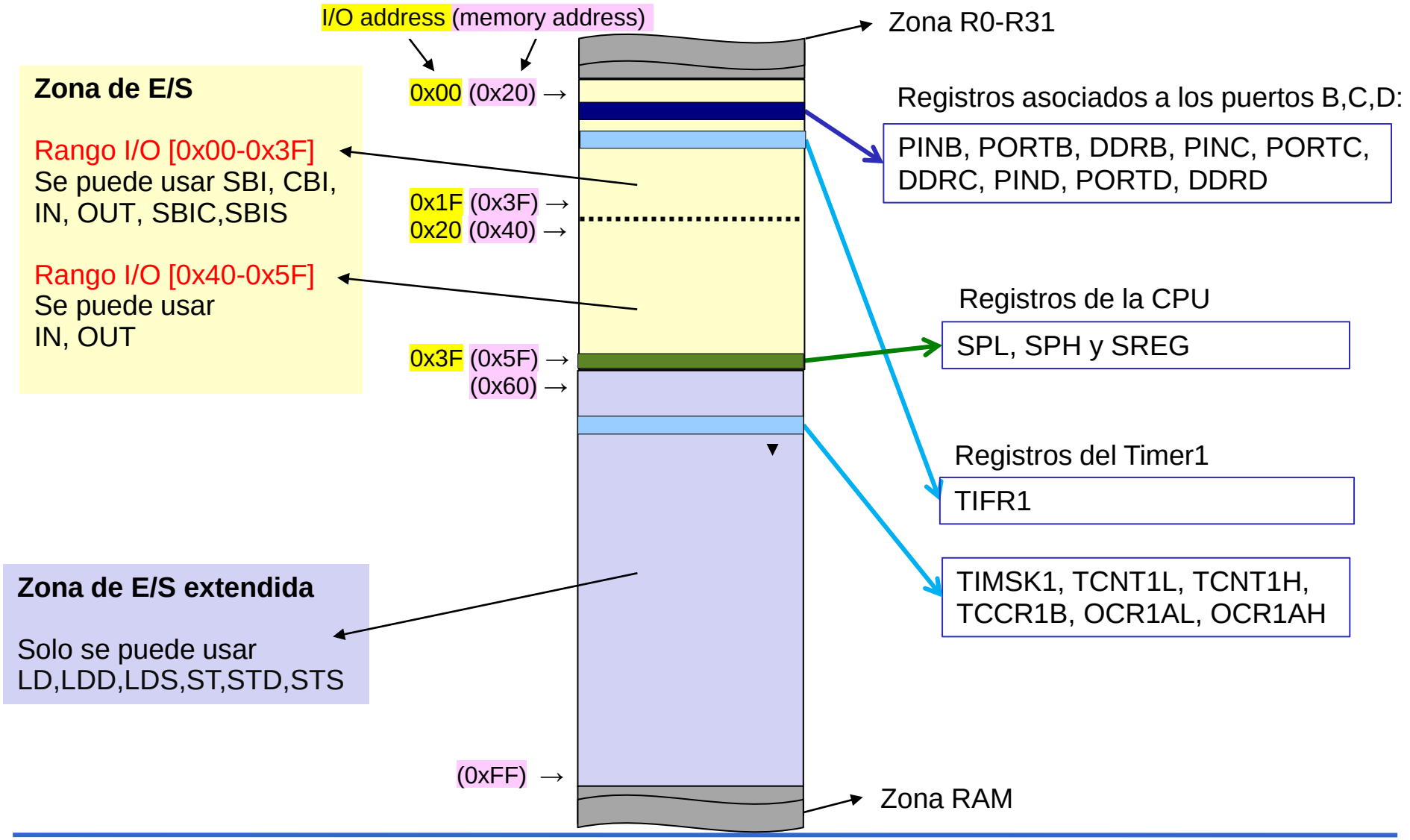
Relé

5V DC
220V AC





Acceso a registros de E/S



Puertos B,C y D

- Los puertos tienen resistencias de pull-up que pueden activarse
- Tienen un circuito de sincronización para leer los valores lógicos.

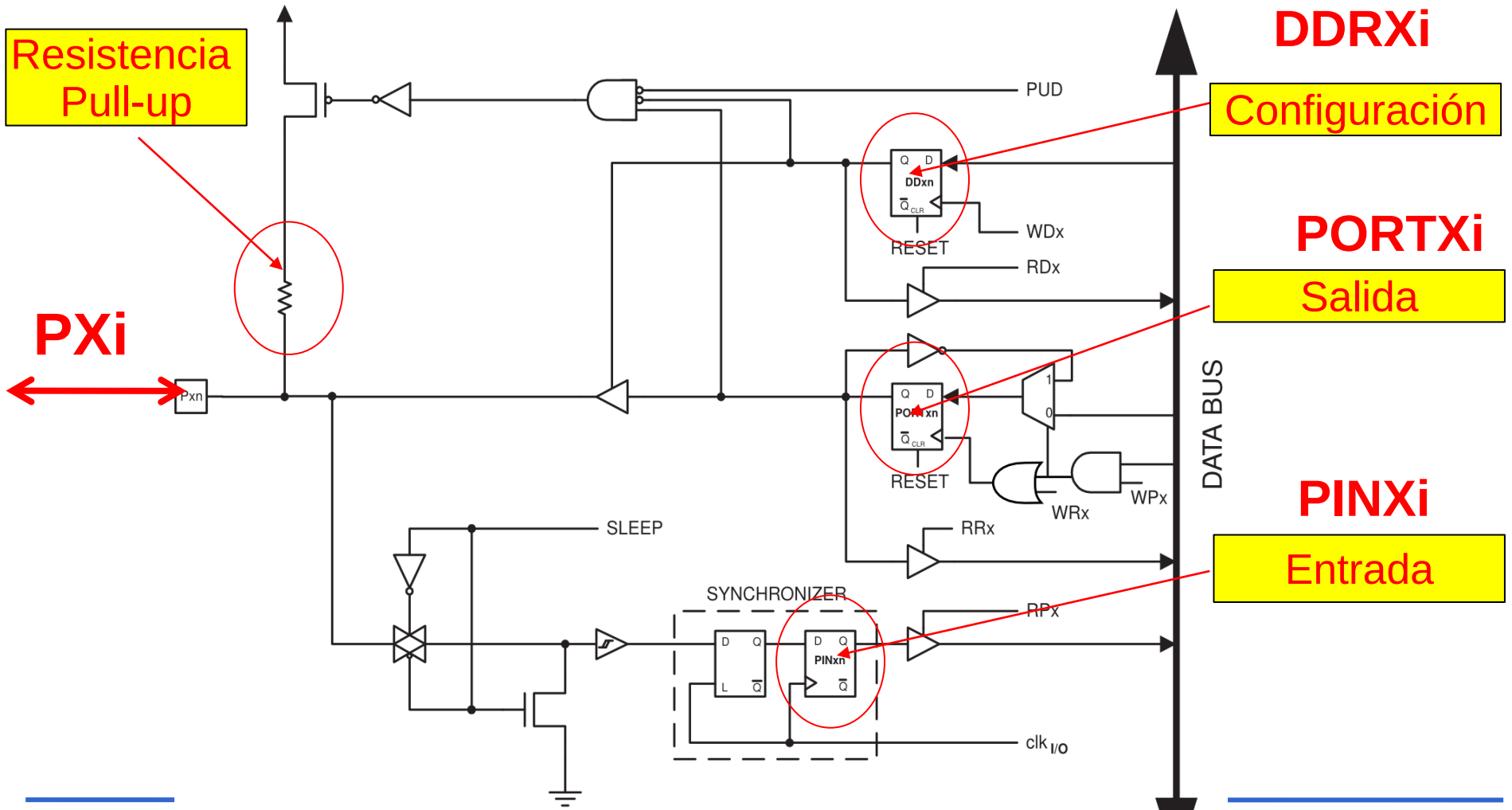
- Para controlar cada uno de los tres puertos hay disponibles 3 registros de E/S (3 para cada puerto). Si X ={B,C,D}:

  $DDRX_{7-0}$: Configura la dirección de cada PIN (entrada o salida)

  $PORTX_{7-0}$: Registro de datos del puerto para escribir en el puerto (salida)

  $PINX_{7-0}$: Permite leer directamente en el PIN independientemente del valor $DDRX_i$

Esquema general para los puertos de E/S digital (B, C o D)



PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
$clk_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

Uso del Puerto B (C y D son similares)

$DDRB_{7-0}$ (R/W): *Data Direction Register B*

El valor del bit DDB**i** indica la dirección del pin PB**i** (“0” para entrada, “1” para salida)

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Este registro ocupa la posición 4 dentro de los 64 registros de E/S. Entre paréntesis aparece $24 (que en decimal es 36 = 4 + 32) indicando la posición absoluta del registro dentro del mapa de la “memoria de datos”.

$PINB_{7-0}$ (R):

Permite la lectura de los valores lógicos de los pines (tanto de entrada como de salida)

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Uso del Puerto B (C y D son similares)

Registro PORTB$_{7-0}$ (R/W):

- DDRBi configurado como salida:

  El valor de PORTBi se muestra en el PIN de salida PBi (salida digital)

- DDRBi configurado como entrada:

  Se activa la resistencia de pull-up del PIN PBi al escribir un “1” en el PORTBi

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

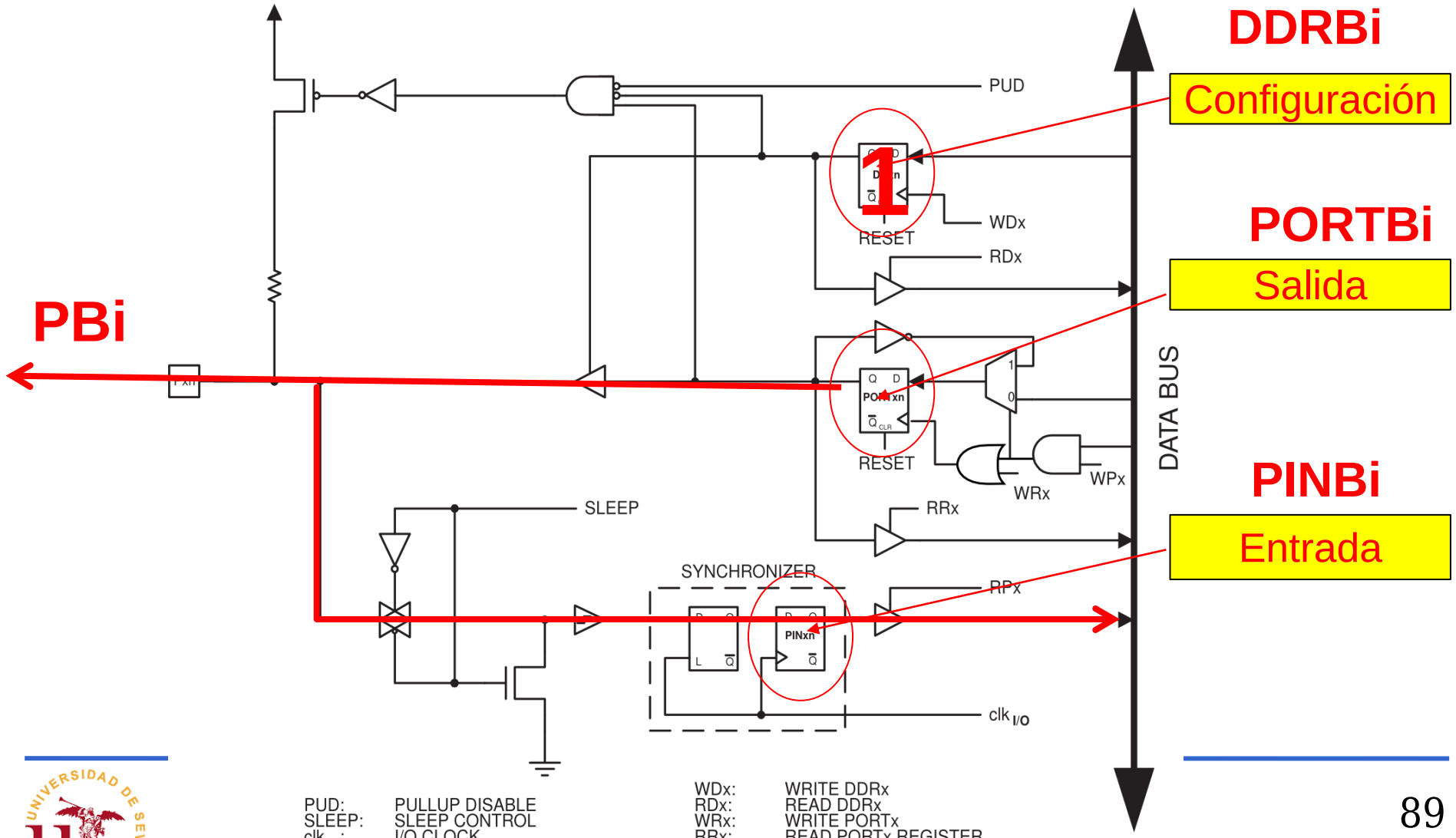
Uso de los puertos B, C y D

Puerto	Registro de E/S	Dirección E/S (memoria)
B	PINB	0x03 (0x23)
	DDRB	0x04 (0x24)
	PORTB	0x05 (0x25)
C	PINC	0x06 (0x26)
	DDRC	0x07 (0x27)
	PORTC	0x08 (0x28)
D	PIND	0x09 (0x29)
	DDRD	0x0a (0x2a)
	PORTD	0x0b (0x2b)

La dirección E/S es utilizada solamente por las IN, OUT, SBI, CBI, SBIC y SBIS.

La dirección entre paréntesis es utilizada por LD, LDD, LDS, ST, STD y STS.

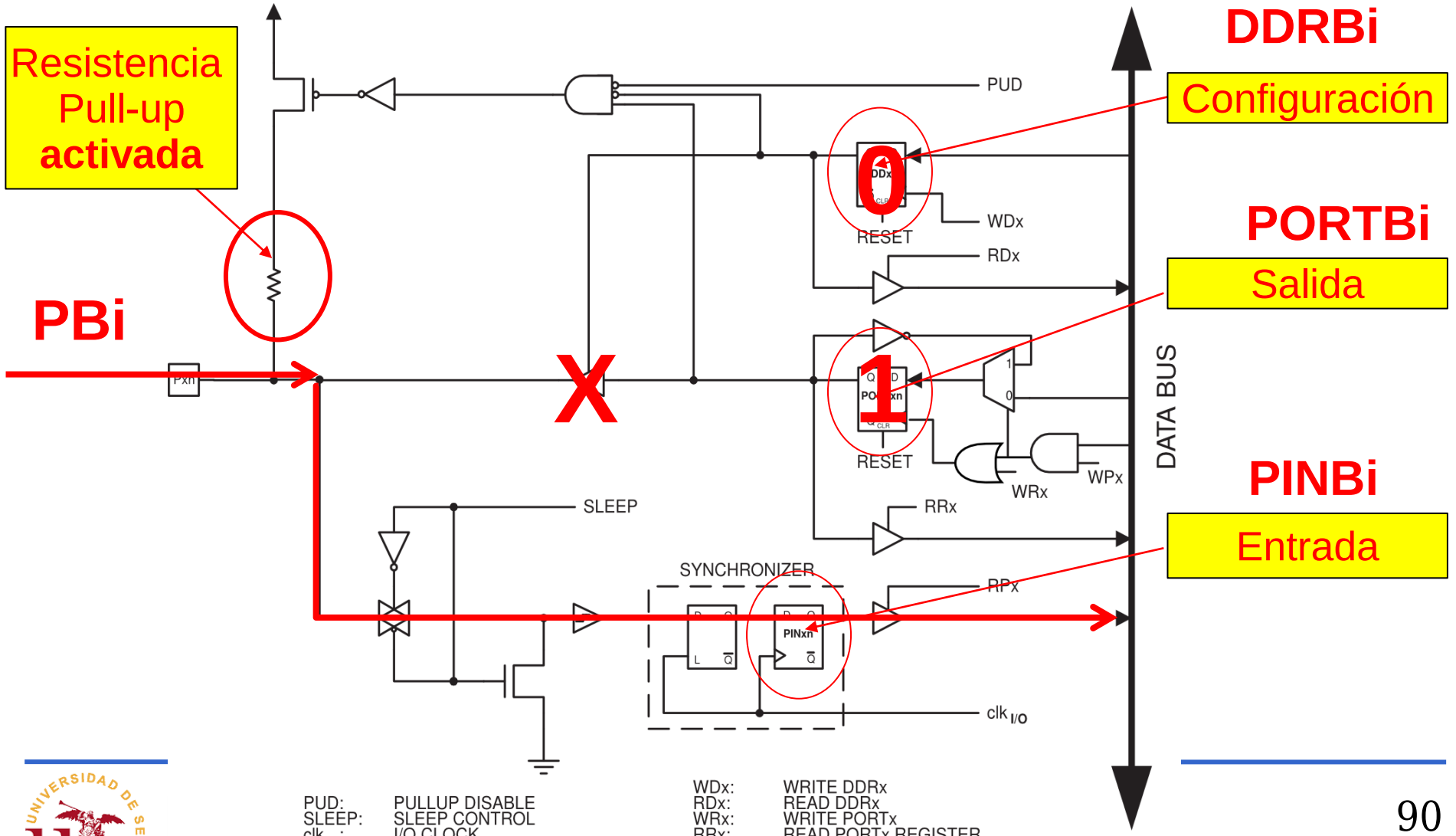
PIN PBi configurado como salida



PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
$clk_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

PIN PBi configurado como entrada



PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
$clk_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WRx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN
WPx: WRITE PINx REGISTER

Configuración y uso de E/S - instrucciones comunes

Sintaxis		Descripción	Rango	Operación	Banderines	Ciclos
IN	Rd,I/O(A)	Entrada desde registro de I/O	d∈[0,31] A∈ [0,63]	Rd←I/O(A)	Ninguno	1
OUT	I/O(A),Rr	Salida hacia registro de I/O	r∈ [0,31] A∈ [0,63]	I/O(A)← Rr	Ninguno	1
SBI	A,b	Poner a 1 el bit b del registro de I/O	b∈[0,7] A∈[0,31]	I/O(A,b)←1	Ninguno	2
CBI	A,b	Poner a 0 el bit b del registro de I/O	b∈[0,7] A∈[0,31]	I/O(A,b)←0	Ninguno	2
SBIC	A,b	Esquiva si el bit b del registro I/O está a 0	A∈[0,31] b∈[0,7]	Si (I/O(A,b)=0) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3
SBIS	A,b	Esquiva si el bit b del registro I/O está a 1	A∈[0,31] b∈[0,7]	Si (I/O(A,b)=1) PC← PC+2 (o 3)	Ninguno	1 o 2 o 3

Ejemplos de uso

Ejemplo 1: Establecer el puerto B como salida y activar el PIN PB3 a '1' y el PIN PB6 a ‘0’

```
    LDI R16,0xFF
    OUT DDRB,R16   ;Puerto entero como salida
    SBI PORTB,3    ;Pone el bit 3 a 1
    CBI PORTB,6    ;Pone el bit 6 a 0
```

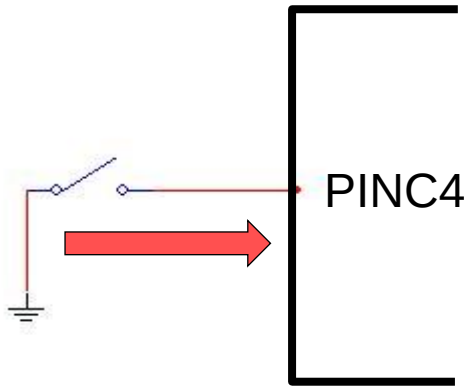
Ejemplo 2: Establecer el PIN PC4 del puerto C como entrada y tomar una decisión en función del valor leído:

```
CBI  DDRC,4    ;PC4 como entrada DDRC4=0
SBIS PINC,4    ;Esquiva una instrucción si PC4=1
JMP  PC4_ES_0  ;Salto si PC4=0
JMP  PC4_ES_1  ;Salto si PC4=1
```

SBIS (SKIP if Bit in I/O register is Set)
SBIC (SKIP if Bit in I/O register is Cleared)

Ejemplos de uso

Ejemplo 3: Configurar PC4 del puerto C como entrada, activar pull-up y esperar pulsación



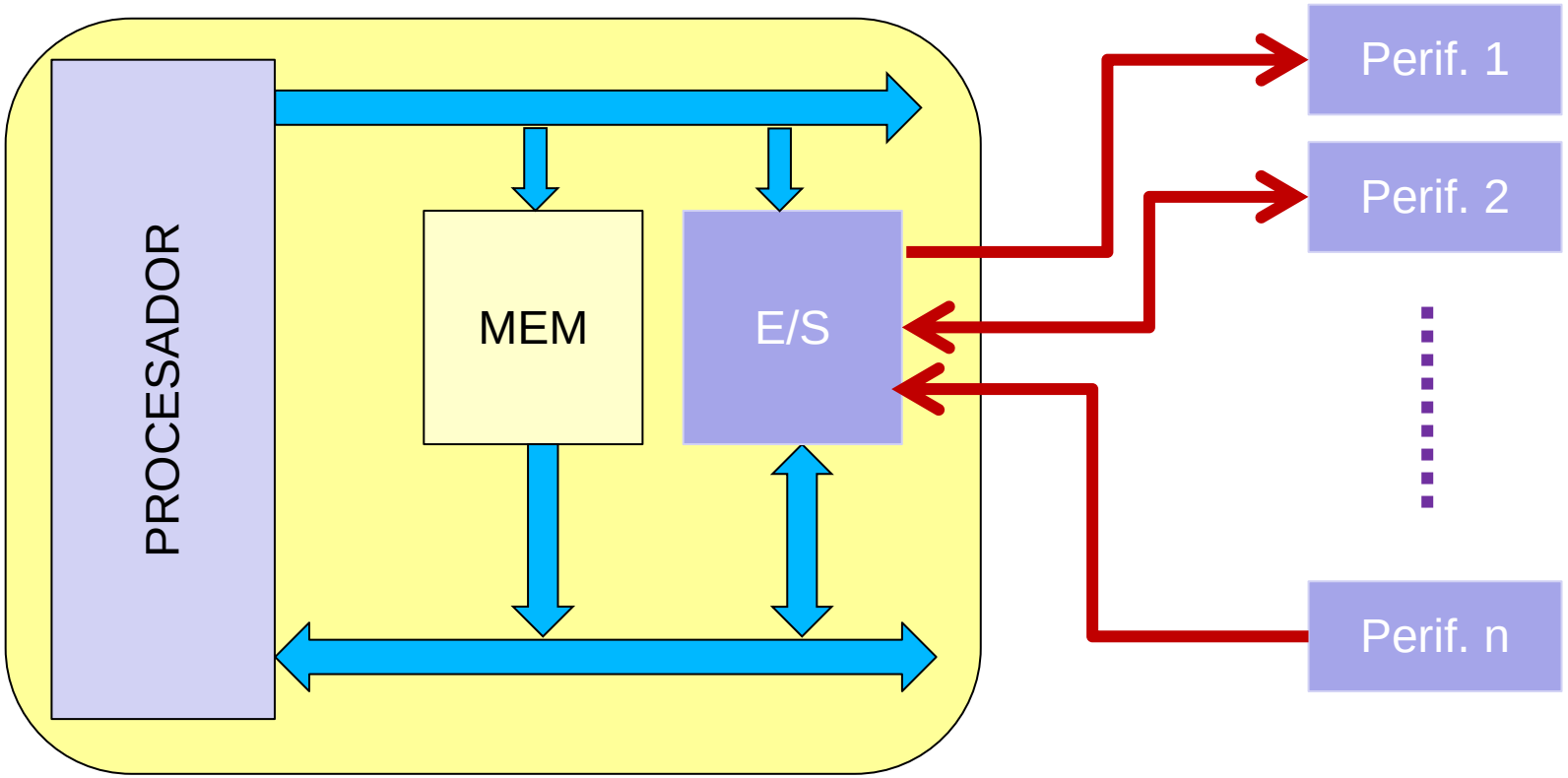
DDRC				**0**				
PORTC				**1**				
PINC				**?**				

```
            CBI  DDRC,4    ;PC4 como entrada DDRC4=0
            SBI  PORTC,4   ;Activo resistencia de pull-up
wait:       SBIC PINC,4    ;Esquiva la siguiente instr.
            rjmp wait      ;si PC4=0
sigue:      ...
```

Índice

Gestión de entrada/salida

Normalmente, un procesador debe atender “en tiempo real” a muchos periféricos para comunicarse con el mundo exterior:



Gestión de entrada/salida

Varias estrategias para el tratamiento de la entrada/salida:

Sondeo periódico (“polling”):

- El procesador sondea periódicamente a todos y cada uno de los periféricos para preguntar si tienen información relevante
- Adecuado cuando hay pocos periféricos y cuando los datos cambian lentamente (por ejemplo, control de temperatura, sensores de contaminación, etc…)

• Interrupciones

- El periférico exige la atención inmediata del procesador
- Por ejemplo, un puerto de comunicaciones.

Interrupciones

Una interrupción es:

un **evento** que requiere la suspensión (interrupción) del programa actual y la ejecución de una rutina concreta (rutina de interrupción), al final de la cual se devuelve el control al programa interrumpido.

Podemos ver una interrupción como un salto a subrutina ocasionado por una señal externa y no por una instrucción del programa.

Interrupciones

Cada interrupción tiene asociado un **vector de interrupción**: un vector que almacena la dirección de la rutina de interrupción.

- Los vectores de interrupción se almacenan en una zona de la memoria llamada "**tabla de vectores de interrupción**"

- Las rutinas de interrupción deben estar ***instaladas*** debidamente según su vector de interrupción.

- La instalación de la rutina de interrupción requiere situar la instrucción jmp o rjmp en la posición adecuada del vector de interrupción.

Modelos Atmega168pa y Atmega328pa requieren instrucción JMP.
Modelos Atmega48pa y Atmega88pa requieren instrucción RJMP.

Tabla de vectores de interrupción (ATmega168pa)

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Notes:
1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 279.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

Ejemplo de inicialización de la tabla de vectores de interrupción

```
;Address    Labels Code        Comments
0x000       jmp RESET          ; Reset Handler
0x002       jmp EXT-INT0       ; IRQ0 Handler
0x004       jmp EXT-INT1       ; IRQ1 Handler
0x006       jmp PCINT0         ; PCINT0 Handler
0x008       jmp PCINT1         ; PCINT1 Handler
0x00A       jmp PCINT2         ; PCINT2 Handler
0x00C       jmp WDT            ; watchdog timer Handler
0x00E       jmp TIM2_COMPA     ; Timer2 Compare A Handler
0x010       jmp TIM2_COMPB     ; Timer2 Compare B Handler
0x012       jmp TIM2_OVF       ; Timer2 Overflow Handler
```

Interrupción de RESET

Provoca que el sistema pase a un **estado inicial** conocido.

- Los registros de E/S toman sus valores por defecto (por ejemplo: PC ←$0000, SP← RAMEND)

- Se busca la instrucción en la posición asociada al vector de RESET (dirección 0 de la memoria de programa):

Causas que pueden generan un RESET:

- Power on RESET (Reset de “encendido del dispositivo”)
- RESET externo
- Reloj “perro guardián" (Watchdog Reset)
- Detector de “apagones” (Brown-out detectors)

Gestión de interrupciones

```
; Así suelen empezar los programas:
; le decimos al ensamblador que lo que viene ahora es código
; para ensamblar en la memoria de programa
                .CSEG
; ahora le decimos que lo que sigue va al comienzo de la memoria
; flash (programa).
                .ORG 0
; la primera instrucción es la que se ejecuta cuando el micro
; detecta un RESET, saltando al programa principal
                JMP main
; luego viene la tabla de vectores de interrupción
; con los saltos a las rutinas de interrupción

                JMP IRQ0_handler
                JMP IRQ1_handler
                . . .
main:           . . .
                {Código programa principal}
```

Gestión de interrupciones

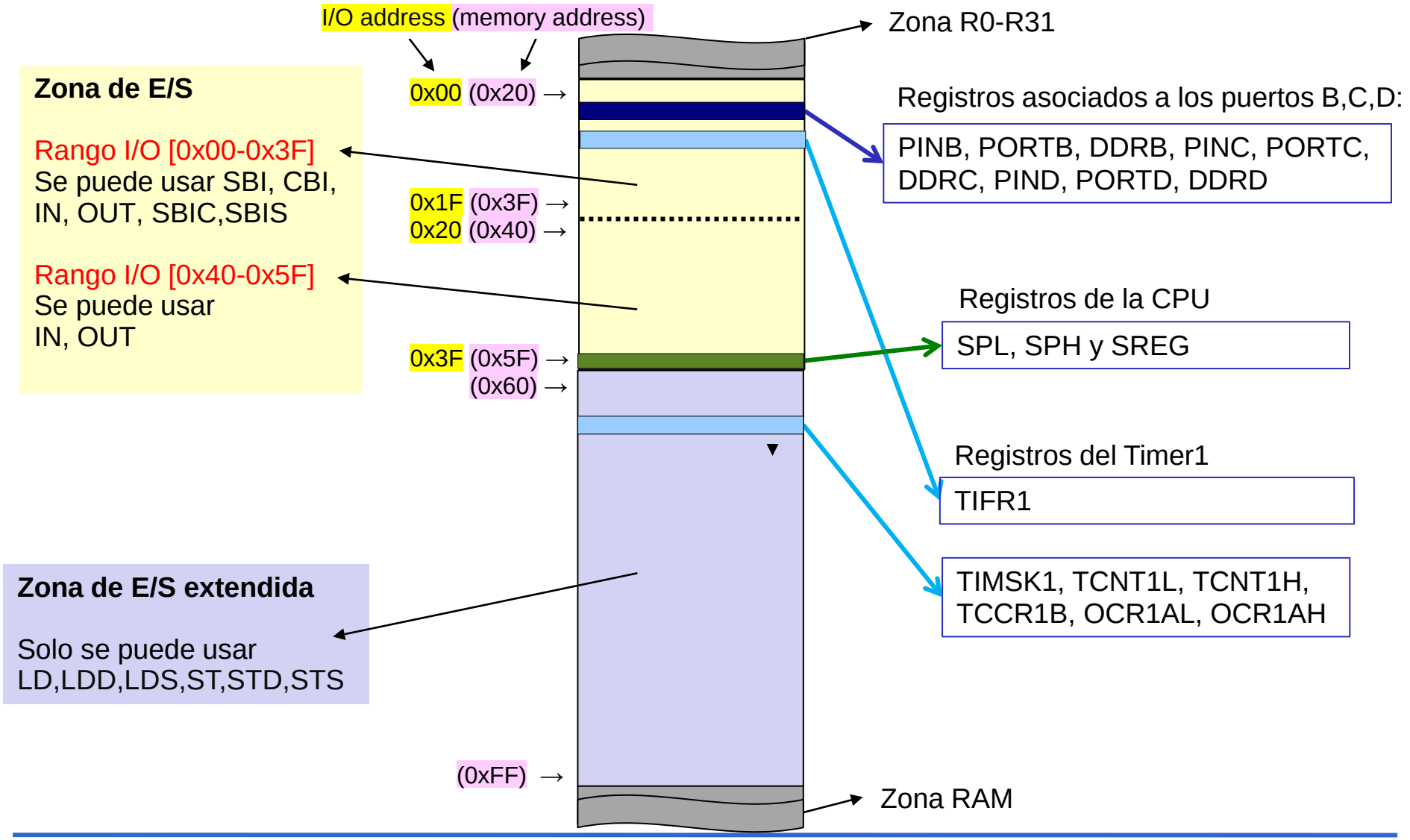
- Para poder procesar interrupciones **I = 1**
- **I = 0 enmascara** todas las interrupciones

- Cuando I =1 y llega una interrupción, el micro:
  - Termina de ejecutar la instrucción en curso
  - Salva en la PILA el PC y se ejecuta la rutina de interrupción correspondiente
  - Durante la ejecución de la interrupción I=0 para evitar interrupciones anidadas
  - La rutina de interrupción termina con RETI (recupera el PC de la PILA y vuelve a hacer I=1)

- La rutina de interrupción debe preservar el registro de estado

Índice

Acceso a registros de E/S



Temporizadores

Hay 3 temporizadores (timers) disponibles.

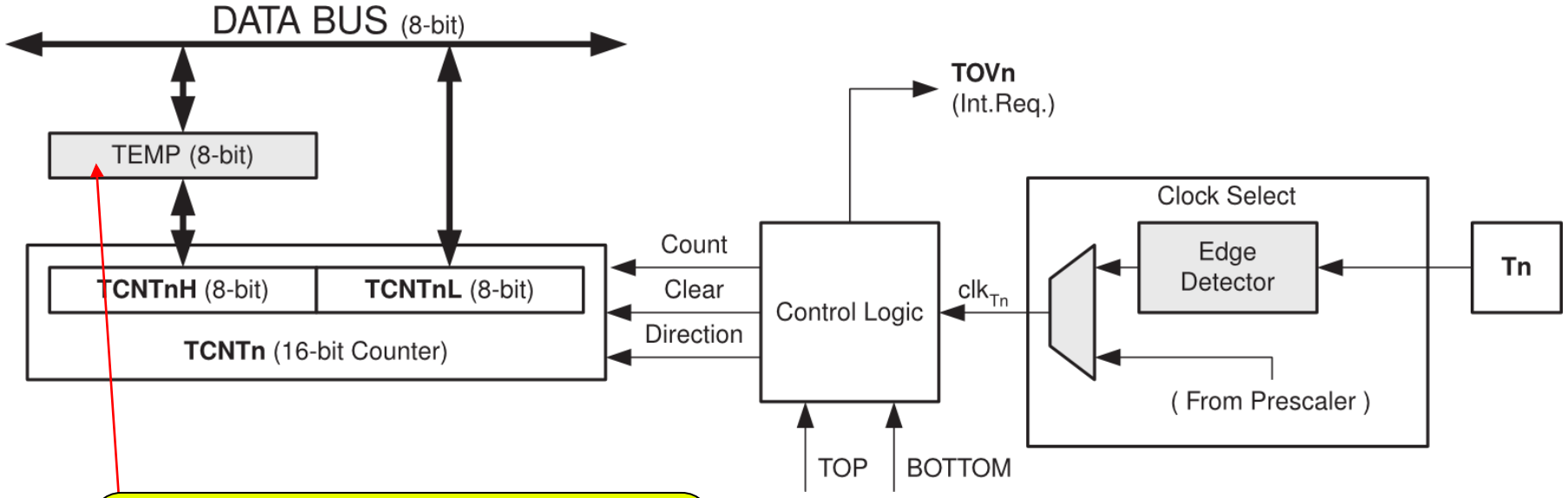
Estudiaremos el Timer 1 (16 bits)

Funcionalidad:

- Generador eventos periódicos
- Contador de eventos
- Generador de señales
- PWM (Pulse Width Modulator)
- Autorreinicio al alcanzar valores programados
- Prescaler (divisor de frecuencias)
- Interrupciones

Temporizadores

Esquema general del temporizador 1 (16 bits)



Registro temporal necesario
al ser el BUS de 8 bits

Temporizadores

De los modos de operación posibles estudiaremos:

Modo normal: Cuenta ascendente de manera indefinida. Después del estado de cuenta $FFFF pasa al $0000

Modo CTC: El contador se pone a cero automáticamente cuando se alcanza el valor (VMAX) establecido en el registro OCR1A

Registros involucrados

Configuración: TCCR1A y TCCR1B

Estado de cuenta: TCNT1H y TCNT1L

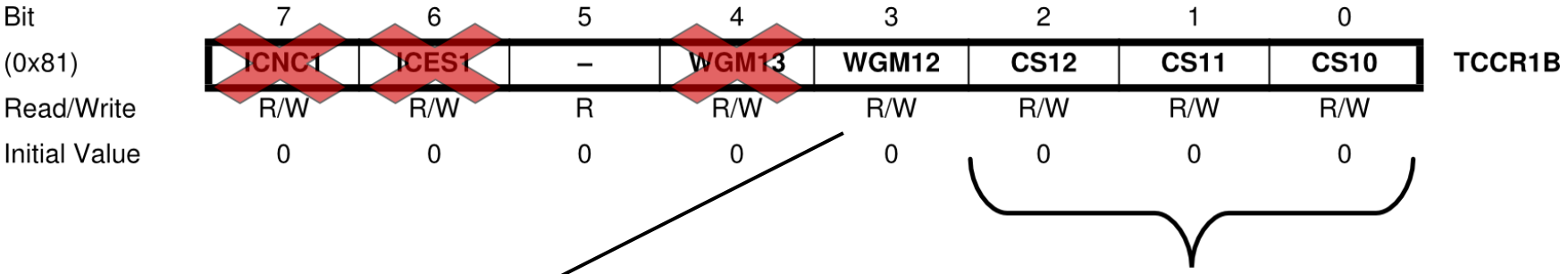
Comparadores: OCR1AH ,OCR1AL, OCR1BH y OCR1BL

Interrupciones: TIFR1 y TIMSK1

Temporizadores

Registro de configuración (TCCR1B)

Solo estudiaremos 4 bits. El resto se pueden quedar sin inicializar, por defecto todos están a cero



WGM12	Modo
0	Normal
1	CTC

CS12	CS11	CS10	Descripción
0	0	0	Temporizador parado
0	0	1	Frecuencia clk/1
0	1	0	Frecuencia clk/8
0	1	1	Frecuencia clk/64
1	0	0	Frecuencia clk/256
1	0	1	Frecuencia clk/1024
1	1	0	Pin T1 en flanco de bajada
1	1	1	Pin T1 en flanco de subida

Temporizadores

Modo normal:	Modo CTC (Clear Timer on Compare Match):
En cada paso por 0x0000 se activa el bit TOV1 (registro TIFR1)	En cada paso por 0x0000 se activa el bit OCF1 (registro TIFR1)
Si el bit TOIE1=1 se produce una interrupción (vector 14: 0x1a)	Si el bit OCIE1A=1 se produce una interrupción (vector 12: 0x16)

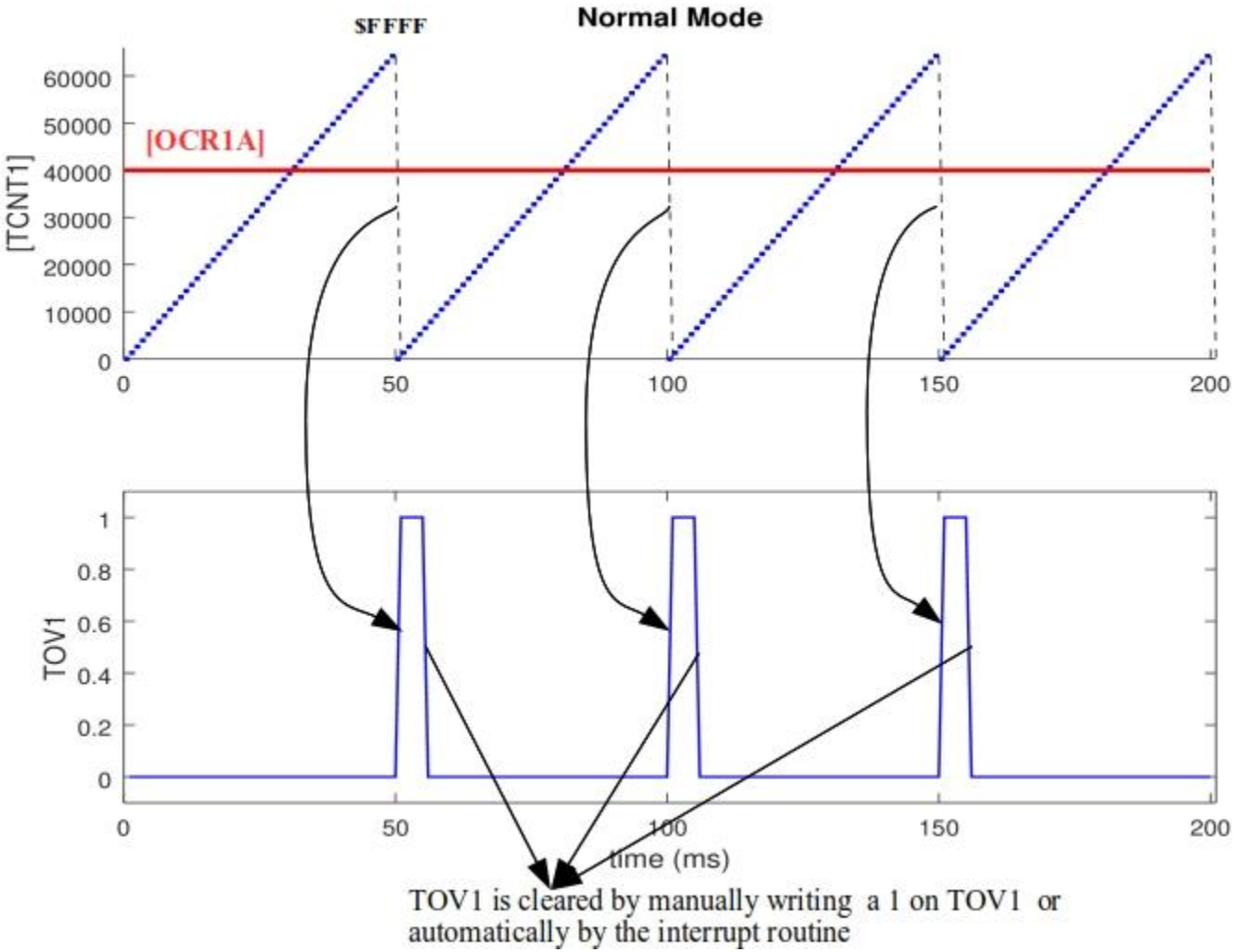
Registro de banderas de interrupciones

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

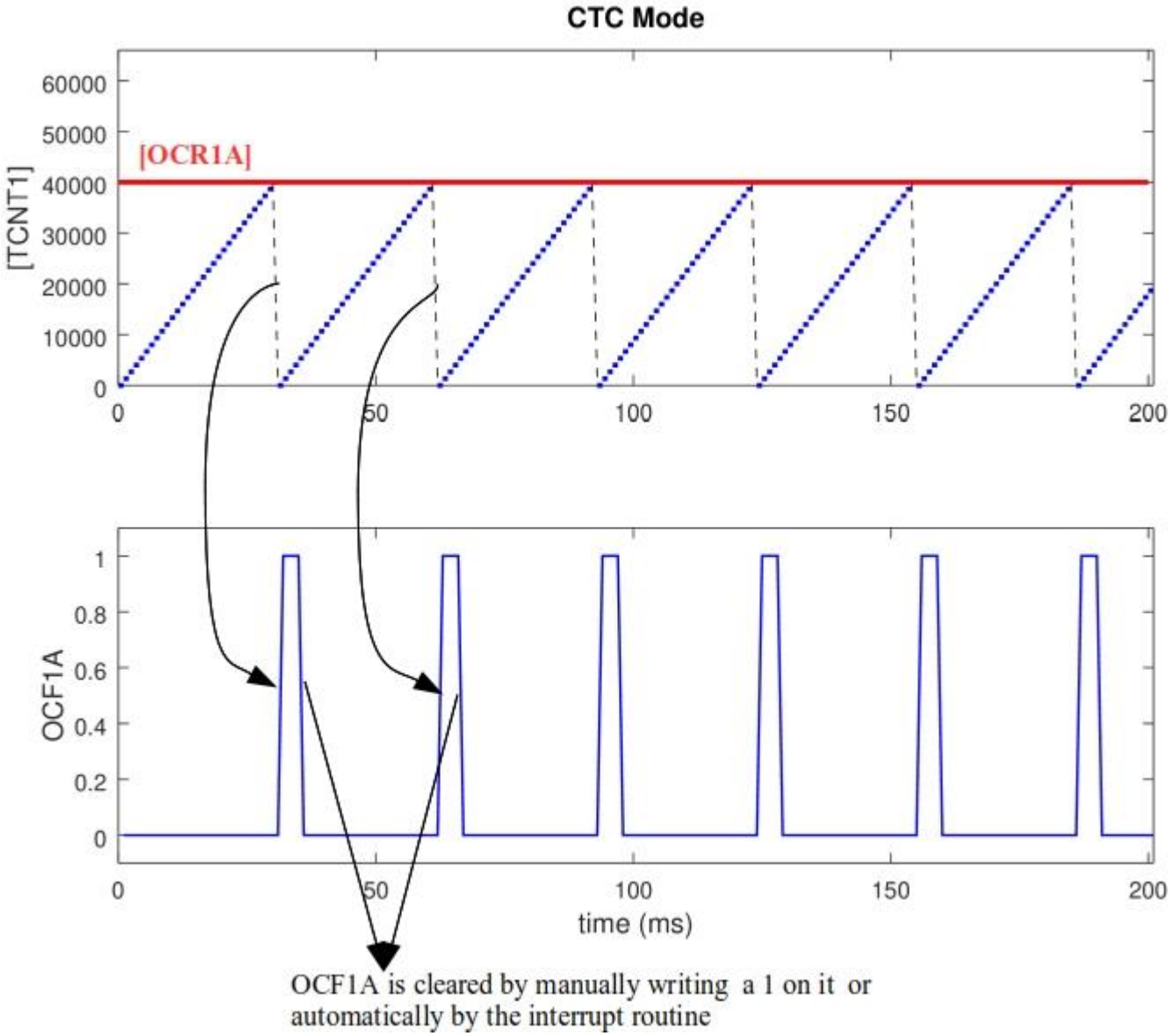
Registro de habilitación de interrupciones

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Temporizadores



Temporizadores



Temporizadores

Importante: El contador es de 16 bits y el bus de 8. La escritura de registros de 16 bits se debe hacer en 2 pasos y en un orden correcto:

1º Parte alta del registro (OCR1AH y TCNT1H)

2º Parte baja del registro (OCR1AL y TCNT1L)

```
LDI R16,0x10
STS OCR1AH,R16  ;Escritura de la parte alta,realmente no
                ;se escribe el registro, se queda en un
                ;registro temporal
LDI R16,0x02
STS OCR1AL,R16  ;Se dispara escritura simultánea de 16
                ;bits: 8 desde un registro temporal y 8
                ;desde el bus del sistema
```

La lectura ha de realizarse en orden inverso.

Temporizadores

Ejemplo:

Con un micro con reloj a 1Mhz conseguir que el contador se reinicie 1 vez por segundo

1º Bajar la frecuencia del reloj con el preescalador: 1Mhz/64=15625Hz

2º Cargar en OCR1A el valor 15625 = $3D09

3º Activar el modo CTC: autoclear cuando el contenido del temporizador sea igual a OCR1A.

Cada vez que se complete el ciclo del contador habrá pasado un segundo

Temporizadores

Solución del ejemplo

```
LDI   R16,0x3D          ;Carga 0x3D09 en OCR1A
STS   OCR1AH,R16        ;Primero parte alta pero OCR1A
                        ;todavía queda inalterado. No se
                        ;puede usar OUT.
LDI   R16,0x09
STS   OCR1AL,R16        ;Tras escribir la parte baja se
                        ;escribe el registro completo
                        ;de 16bits
LDI   R16,0b00001011    ;Activa el modo CLC, bit WGM12=1
                        ;Prescaler CLK/64
STS   TCCR1B,R16        ;A partir de esta instrucción el
                        ;timer está funcionando
```

Herramientas de programación y simulación:

AVR STUDIO (Laboratorio)

Bibliografía

- Instruction Set datasheet.
- Atmegax8pa datasheet.
- Microcontroller projects with the atmel controller.

  Gadre, Dhananjay.
- Atmel AVR microcontroller primer: programming and interfacing.

  Barlett and Pack.
- Embedded Systems Design with the Atmel AVR Microcontroller.

  Steven Barret.