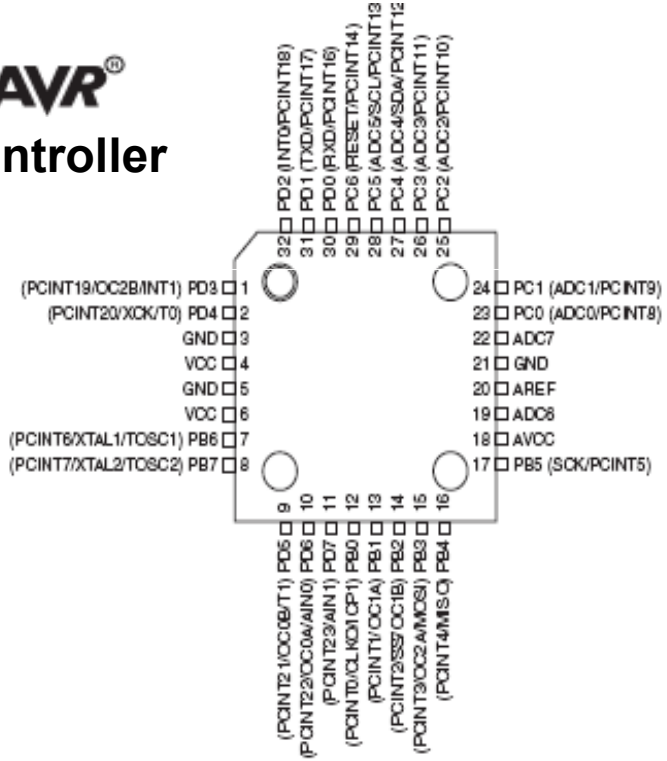
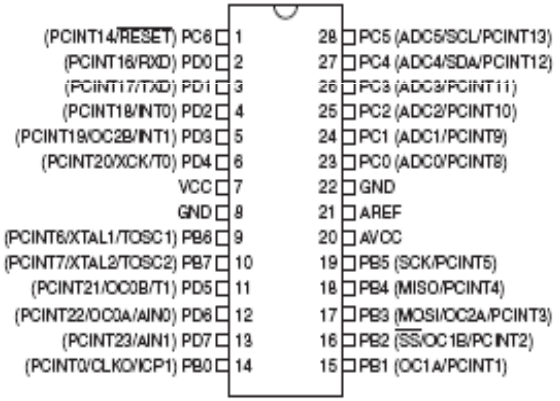
Tema 4: Ejemplo de un computador real: ATmegaX8PA

8-bit AVR® Microcontroller



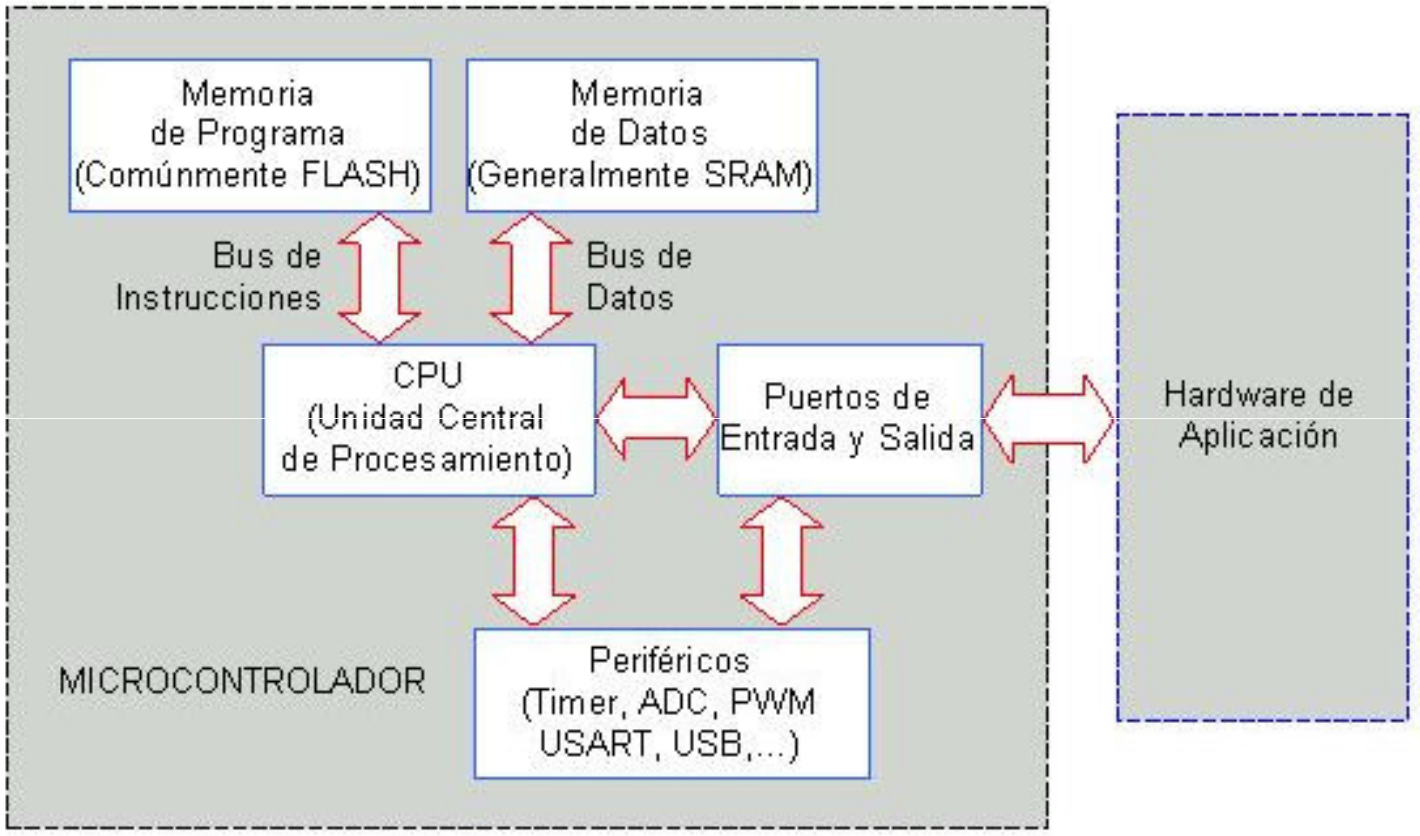


Índice

Índice (ii)

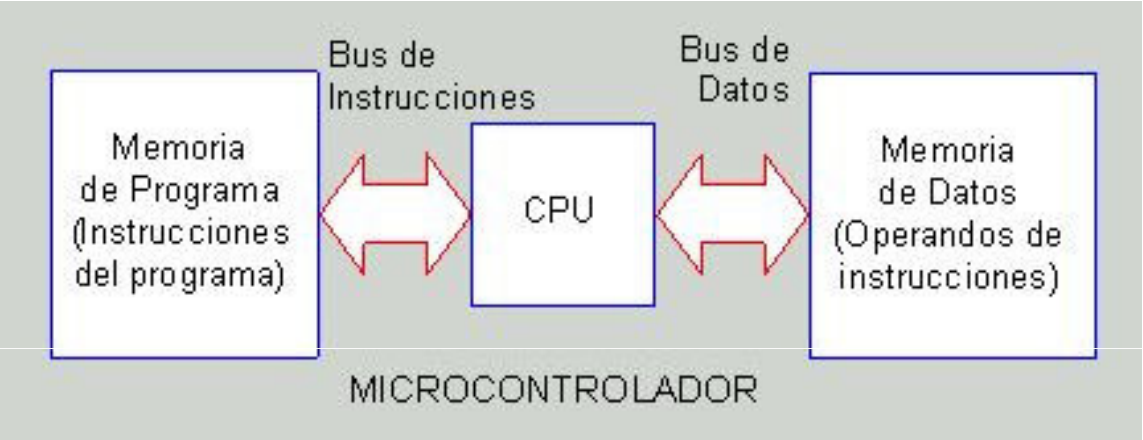
Índice

¿Qué es un microcontrolador?



Es un chip (Circuito integrado C.I.) que contiene una CPU sencilla, unidad de reloj, memoria, y puertos de E/S (timers, I/O ports, puertos de comunicaciones serie,..).

Arquitectura interna



La mayoría de los microcontroladores comerciales tienen **arquitectura Harvard** (Memoria de programa y memoria de datos separadas)

Principales familias de µcontroladores

- ***PICmicro***

- ***AVR***

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

Aplicaciones µcontroladores

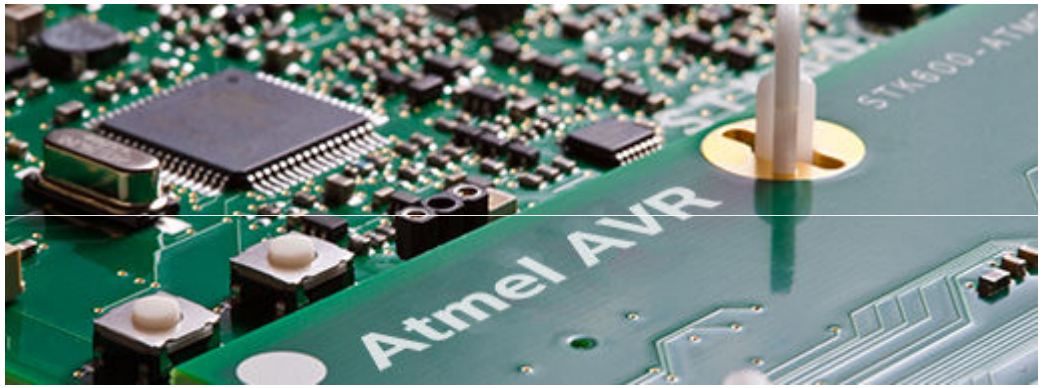
Aplicaciones µcontroladores

Aplicaciones µcontroladores

Índice

Familia AVR ATmega

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



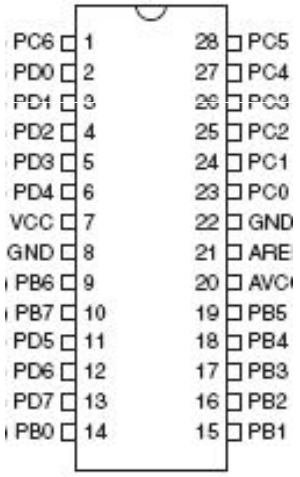
- Un microcontrolador es *una pequeña computadora empotrada en un C.I.* que contiene una CPU sencilla, unidad de reloj, memoria, y puertos de E/S (timers, IO ports, USARTs,..).

- Estudiaremos el **ATmegaX8PA**

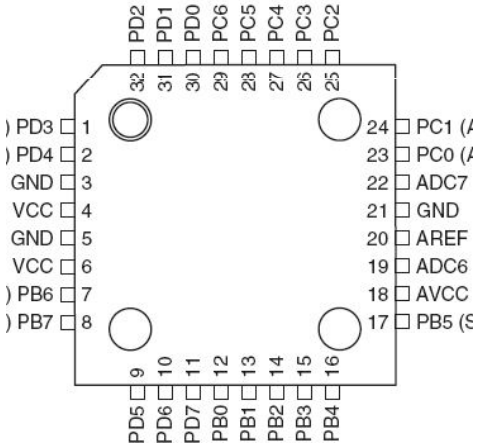
Familia AVR ATmega

Descripción General

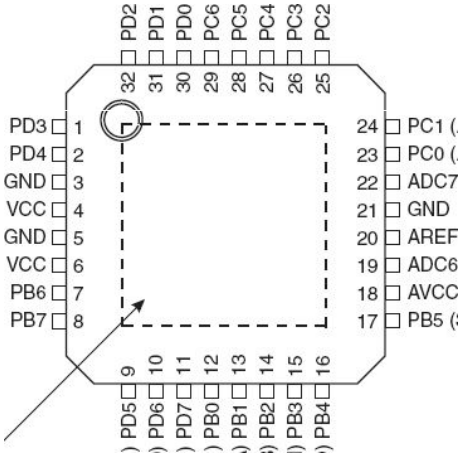
- Diversos periféricos
- Encapsulados: PDIP, TQFP, MLF



PDIP



TQFP



MLF

- 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)	EEPROM (datos)	SRAM (datos)	Tamaño vector interrupción
ATmega48PA	4K Bytes	256 Bytes	512 Bytes	1 instrucción
ATmega88PA	8K Bytes	512 Bytes	1K Bytes	1 instrucción
ATmega168PA	16K Bytes	512 Bytes	1K Bytes	2 instrucciones
ATmega328P	32K Bytes	1K Bytes	2K Bytes	2 instrucciones

ATmegaX8PA

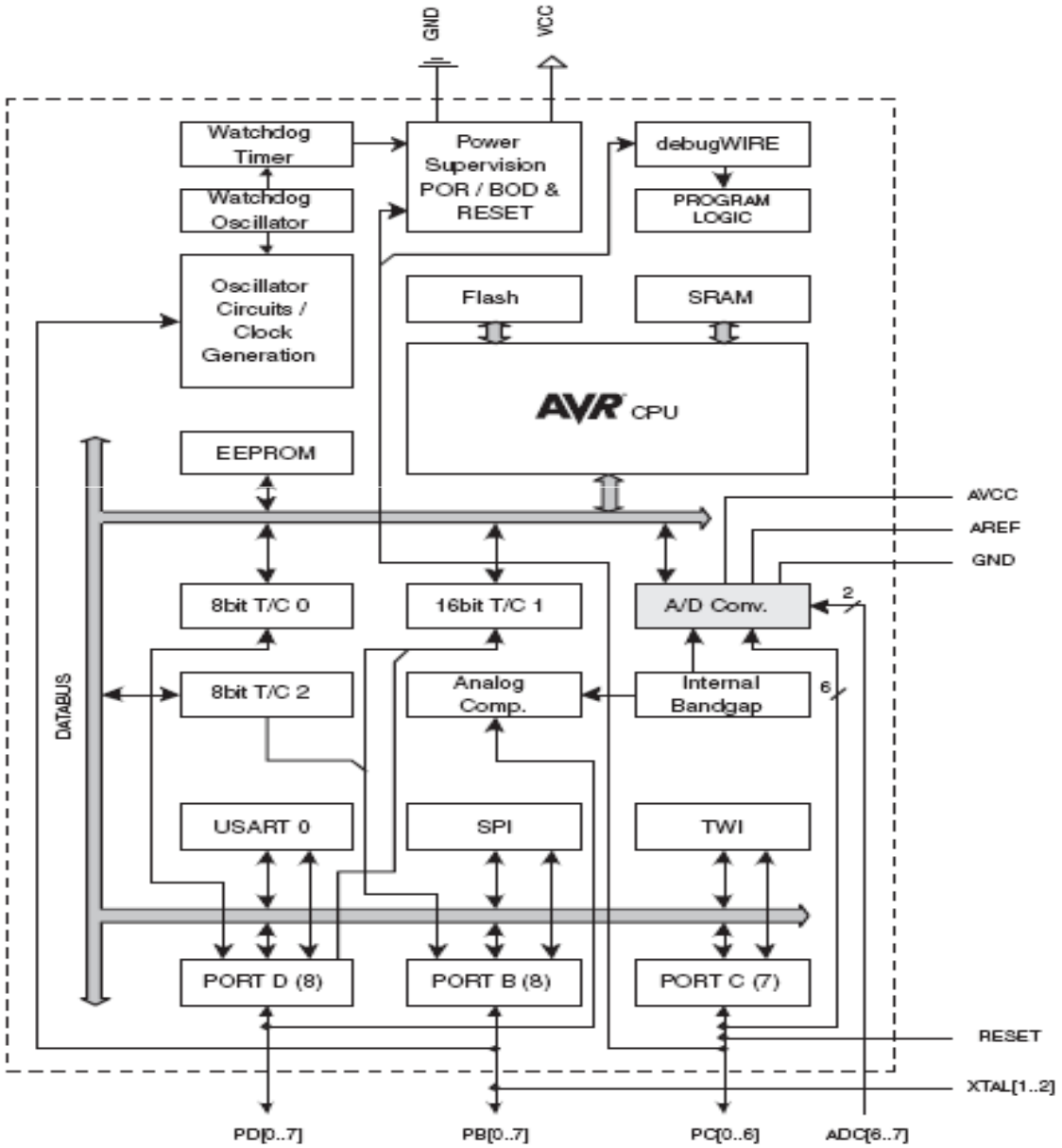
Características

- 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/1K/2K bytes
- Data EEPROM 256/512/512/1K bytes

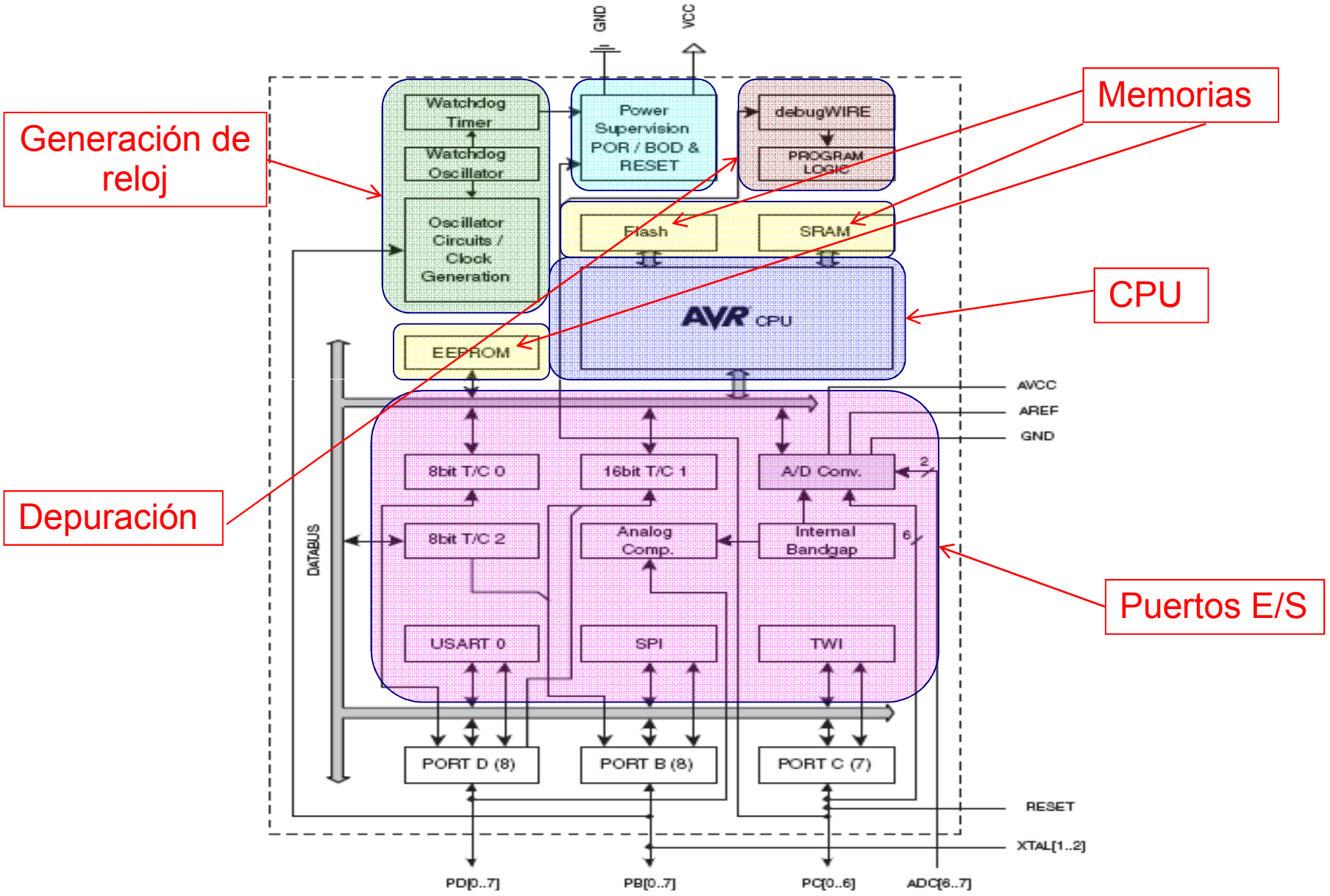
Índice

1. Introducción
2. Descripción general
3. **Arquitectura interna**
4. Organización de memoria
5. Modos de direccionamiento
6. Juego de instrucciones
7. Directivas de ensamblador

Arquitectura interna

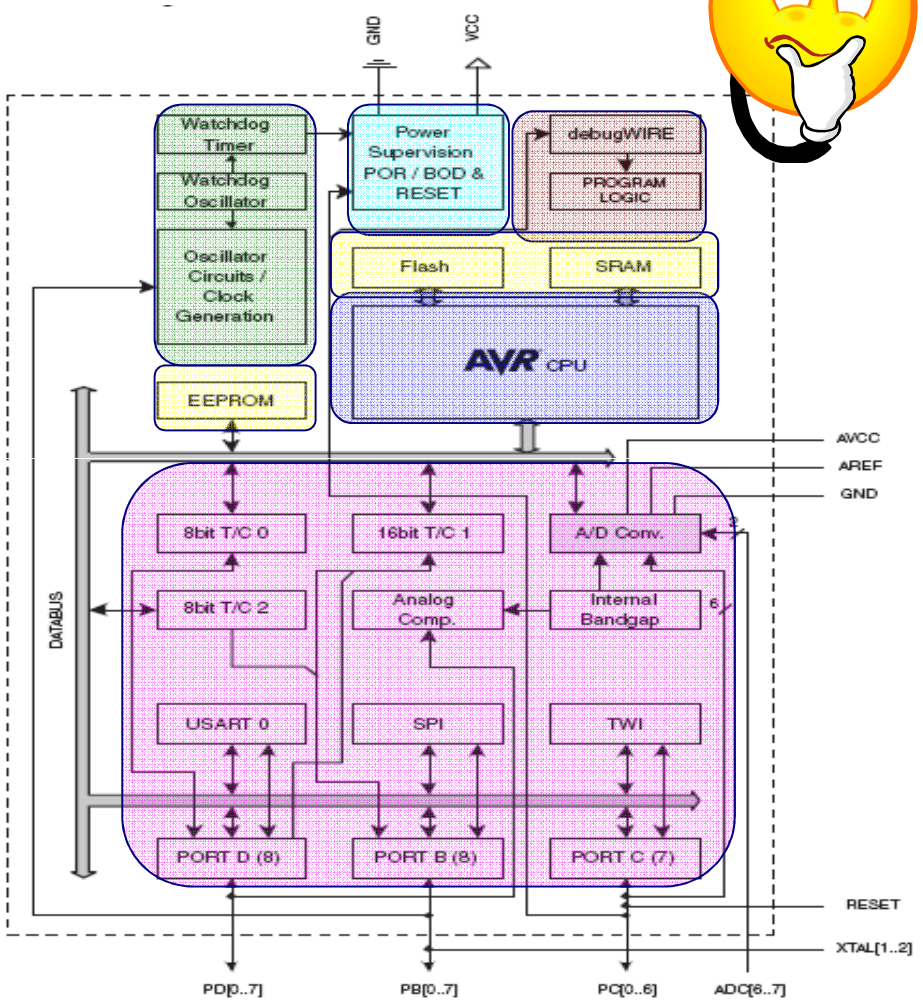


Arquitectura interna



Arquitectura interna

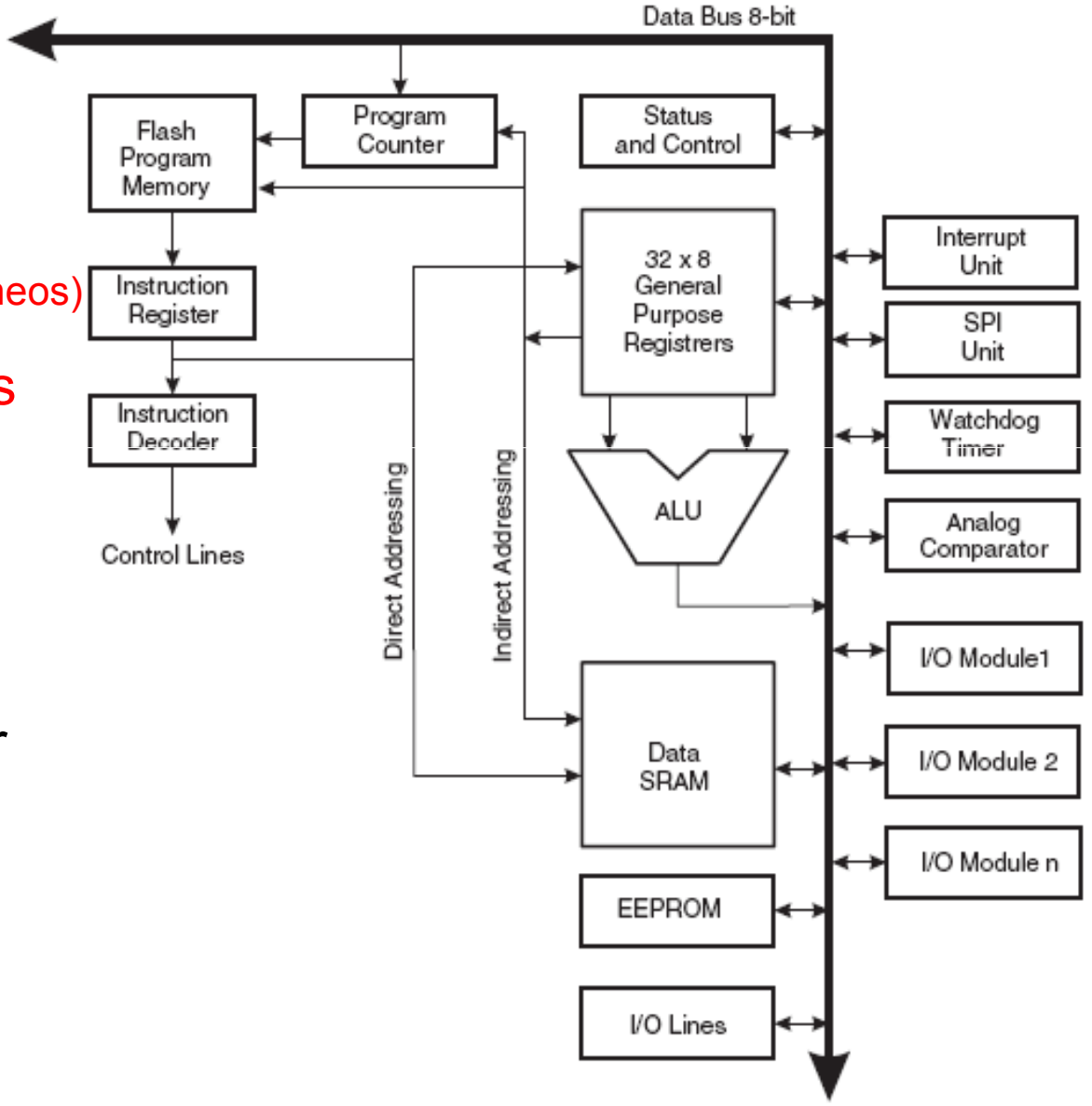
- AVR-CPU
- Memorias (Flash, SRAM)
- Generación de reloj
- Depuración
- Circuito de Reset
- Puertos de E/S:
  - CAD, Analog Comp.
  - Ports B,C,D
  - USART, TWI, SPI
  - Timers



Arquitectura interna CPU AVR

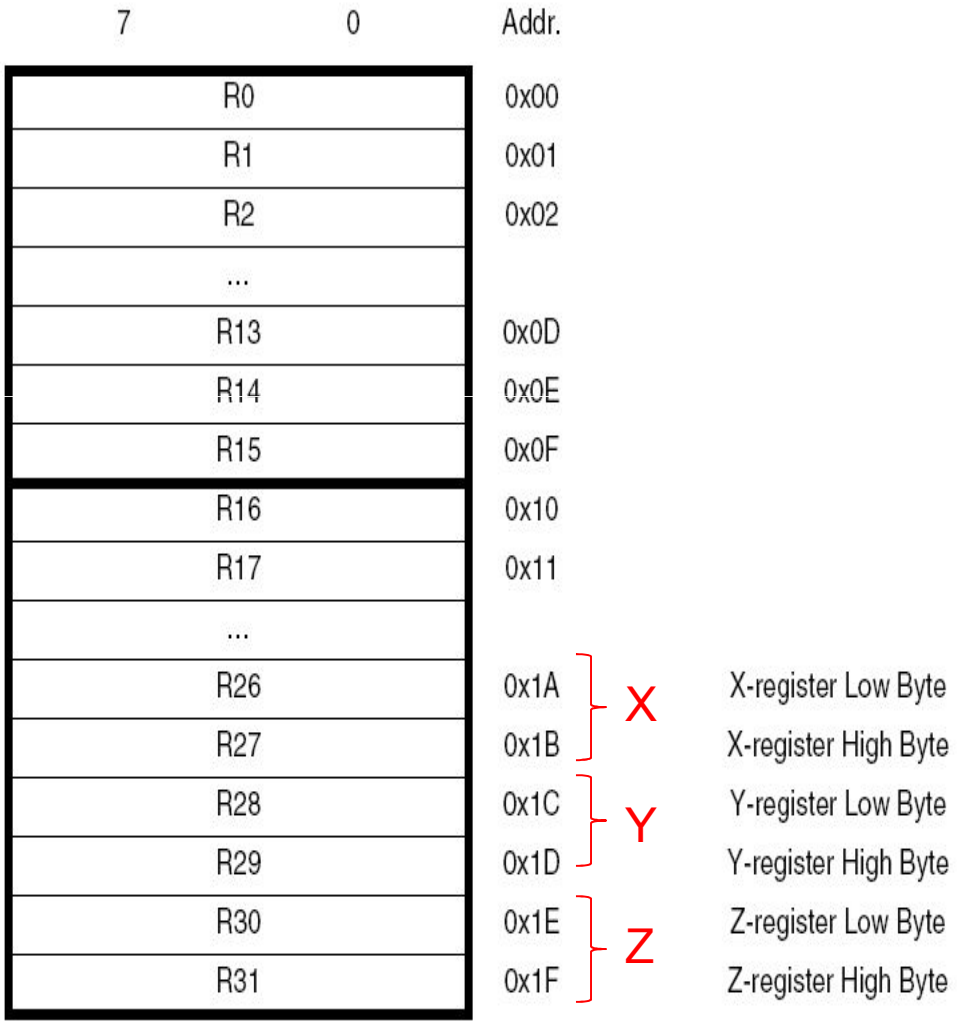
- Arquitectura Harvard
- Pipeline de 1 nivel:

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



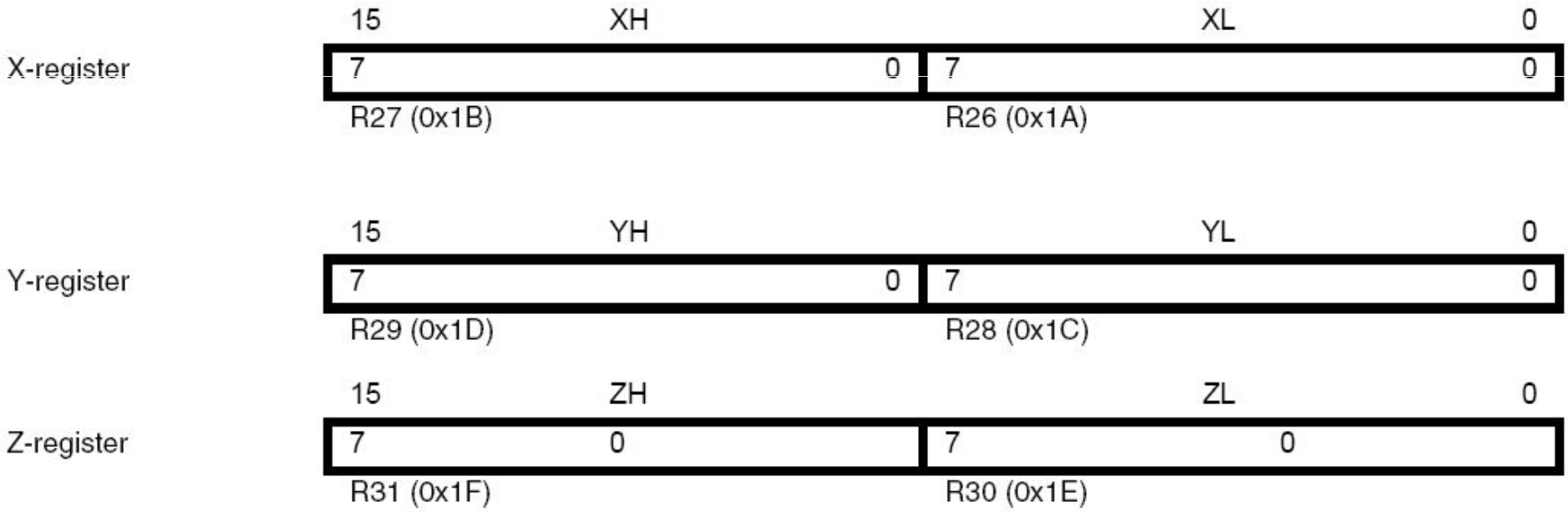
Registros de propósito general (CPU AVR)

- Ocupan las 32 primeras posiciones de la memoria
- 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



Registros X, Y, Z

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



Registro de estado (SR)

- **Bit C** (Carry)
- **Bit Z** (Zero)
- **Bit N** (Negative)
- **Bit V** (oVerflow en Ca2)
- **Bit S** (Signo: S = N $\oplus$ V)
- **Bit H** (Half Carry)
- **Bit T**
- **Bit I** (Interrupción)

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 (cont.)

- Ubicado en el área de entrada/salida de la memoria de datos.
- Banderines (flags) que reflejan el resultado de la ejecución de algunas instrucciones (principalmente aritméticas, lógicas, etc.)
  - **Bit C (Carry)**
  - **Bit Z (Zero)** . Se pone a 1 para resultado cero, 0 en otro caso.
  - **Bit N (Negative).** Bit 7 del resultado.

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 (cont.)

- **Bit V (oVerflow** en Ca2)

  Si A,B son operandos y R resultado, este bit se calcula como:

  - V = A7 B7 R7' + A7' B7' R7 en suma
  - V = A7 B7' R7' + A7' B7 R7 en resta

- **Bit S (Signo)**

  Refleja el signo correcto del resultado en operaciones en Ca2.

  - S = N ⊕ V

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 (cont.)

- **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	

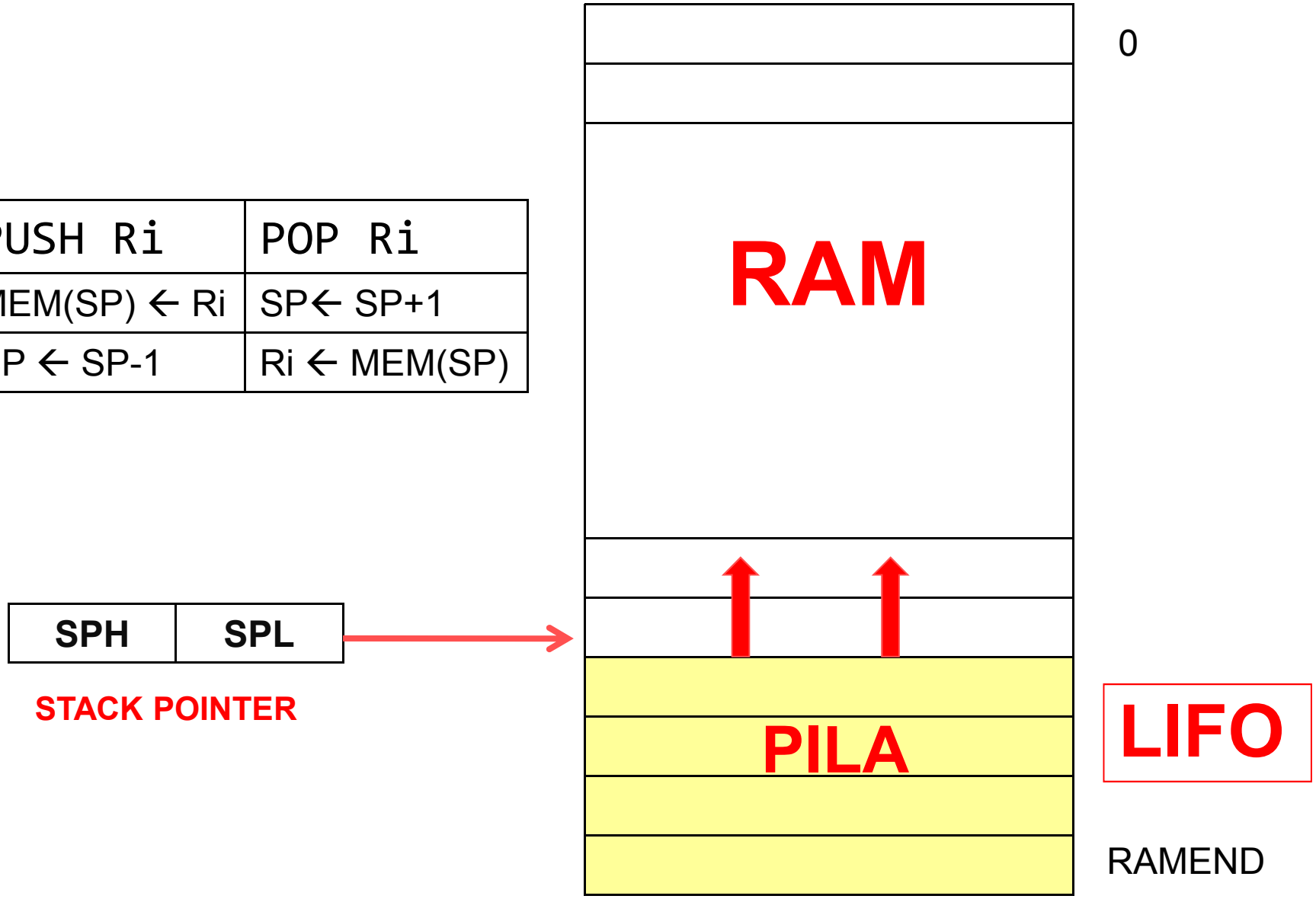
Puntero de Pila (Stack Pointer)

- Se encuentra en la región de memoria asignada a E/S.
- Apunta al área de pila (Valor inicial RAMEND).
  - El SP se decrementa (post) una unidad al meter un dato en pila (PUSH) pila y se incrementa (pre) una unidad al sacarlo (POP). El SP apunta a la dirección siguiente a la cima de la PILA.
  - 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)

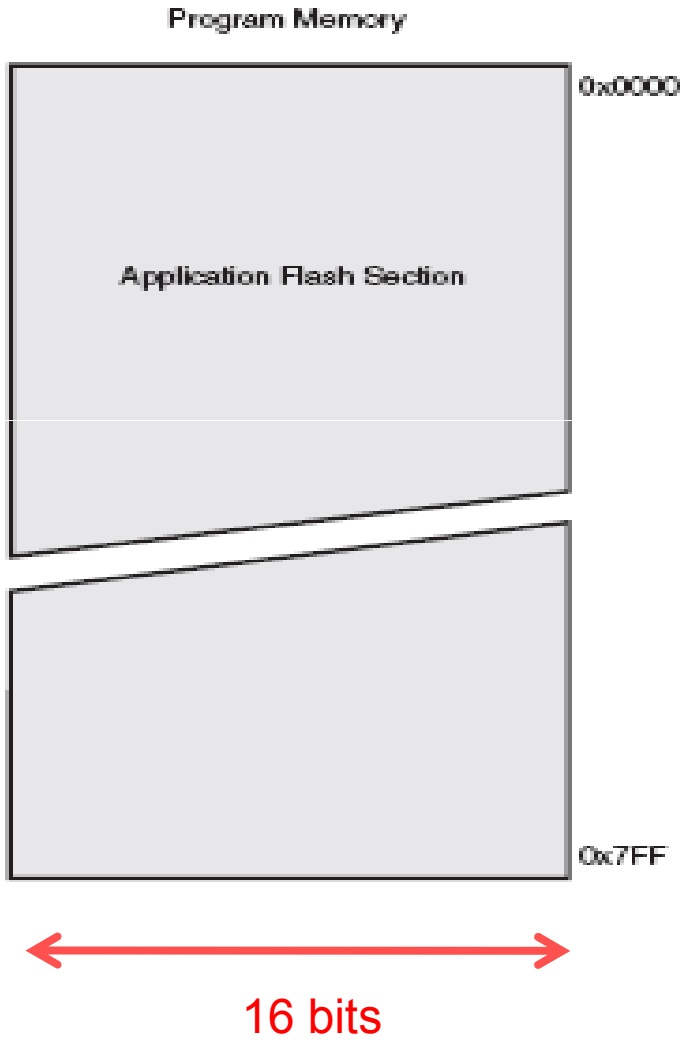
PUSH Ri	POP Ri
MEM(SP) ← Ri	SP← SP+1
SP ← SP-1	Ri ← MEM(SP)



Índice

1. Introducción
2. Descripción general
3. Arquitectura interna
4. **Organización de memoria**
5. Modos de direccionamiento
6. Juego de instrucciones
7. Directivas de ensamblador

Organización de la memoria de Programa (Flash)

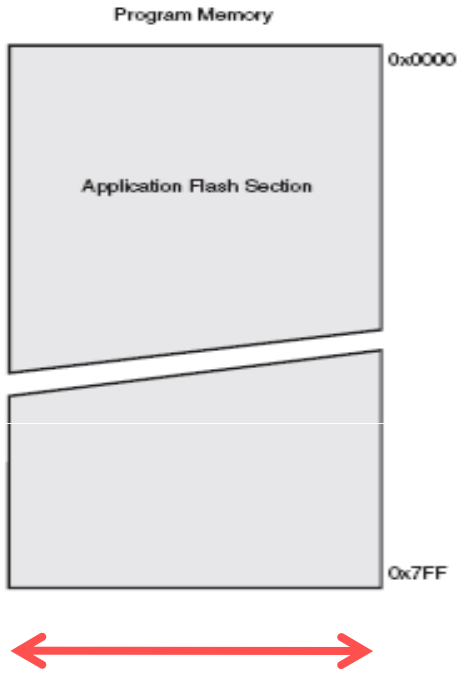


Atmega48PA

- desde: 0x000
- hasta: 0x7FF

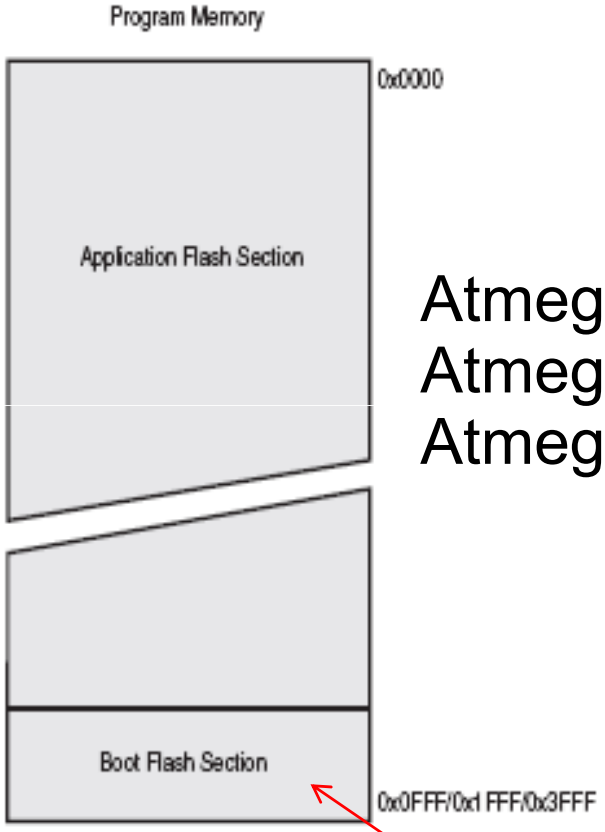
(2K x16 bits: 4K Bytes)

Organización de la memoria de Programa (Flash)



16 bits

Atmega48PA

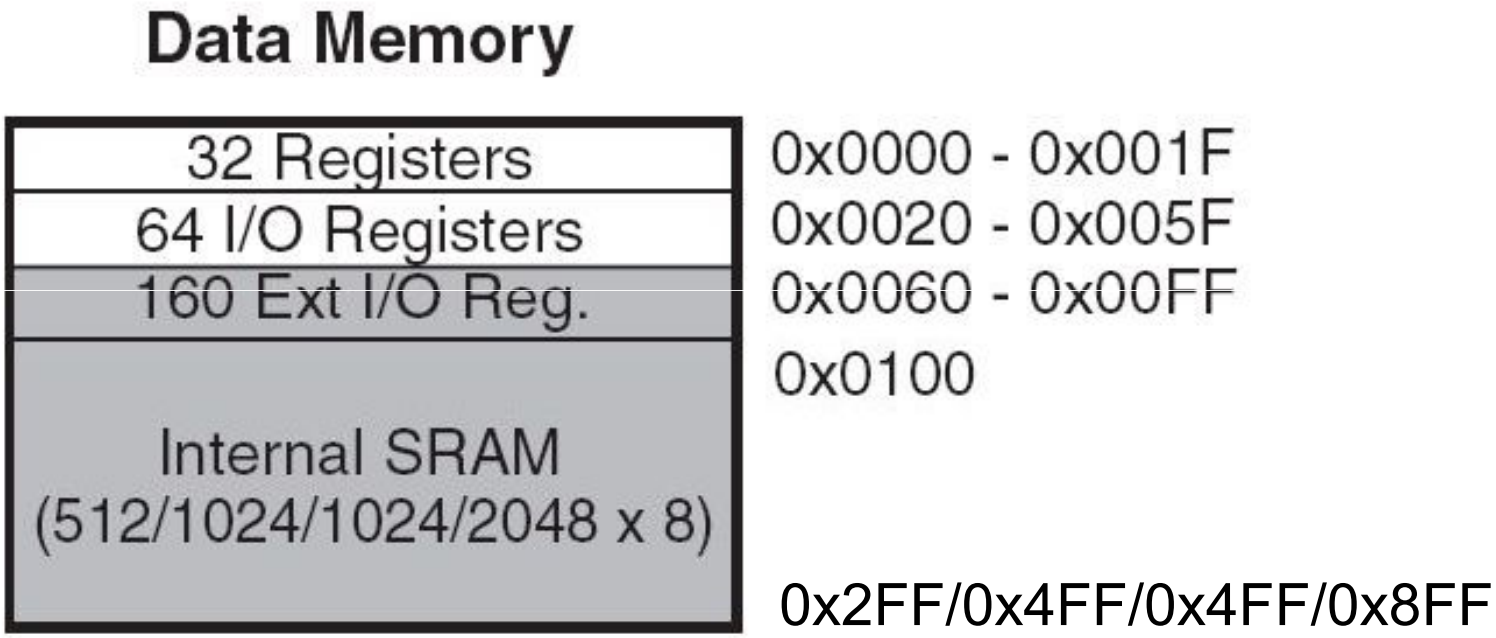


16 bits

Atmega88PA
Atmega168PA
Atmega328PA

Sección de arranque

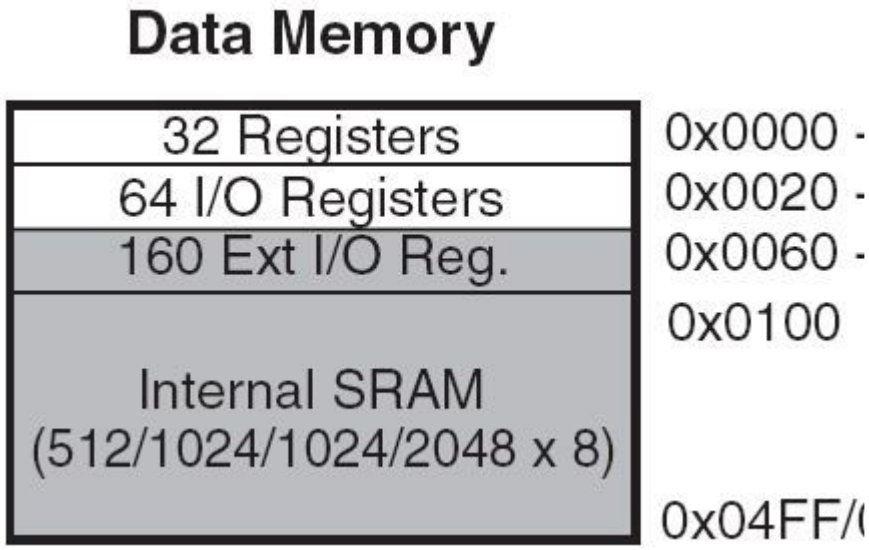
Organización del mapa de memoria de Datos



Organización del mapa de memoria de Datos

- **32 Registros de propósito general** con todos los modos de direccionamiento **(inmediato solo de R16 a R31, indirecto solo X,Y,Z)**
- **Registros de E/S**
  - 64 I/O Registers (instrucciones IN, OUT, modo directo e indirecto)
  - 160 Ext Registers (modos directo e indirecto)
- **SRAM**

  (modos directoe indirecto)



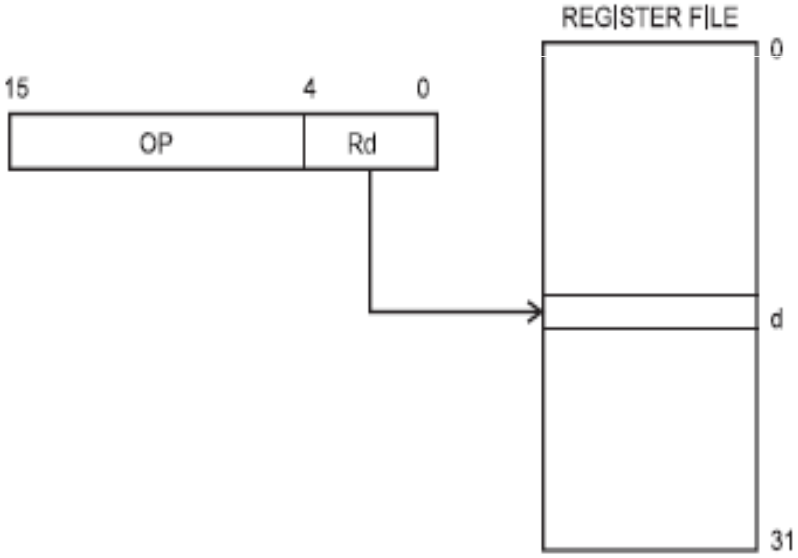
Índice

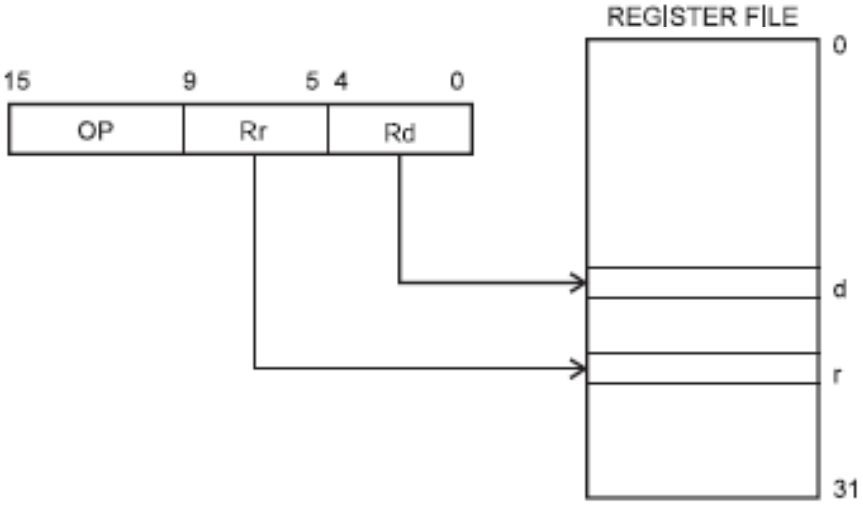
5.1 Modos de direccionamiento de la memoria de Datos

- **Registro directo**
- **Entrada/Salida directo**
- **Directo**
- **Indirectos**
  - Indirecto
  - Indirecto con predecremento
  - Indirecto con postincremento
  - Indirecto con desplazamiento
- **Inmediato**

Direccionamiento de Registro Directo (memoria de Datos, zona de registros pr. g.)

- *La instrucción define el registro o registros cuyo contenido se verá afectado por la propia instrucción.*
  - Ejemplos: COM R4 MOV R1,R2



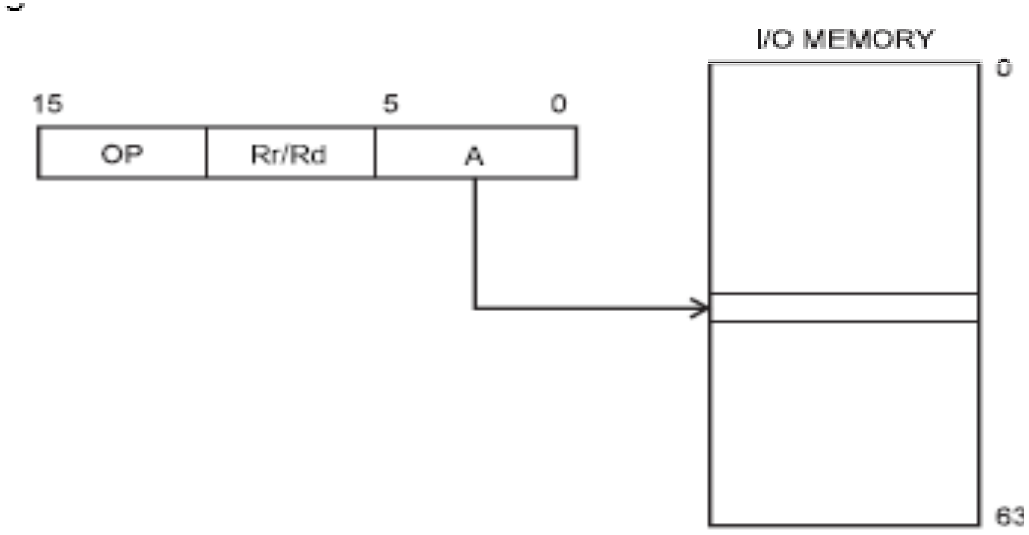


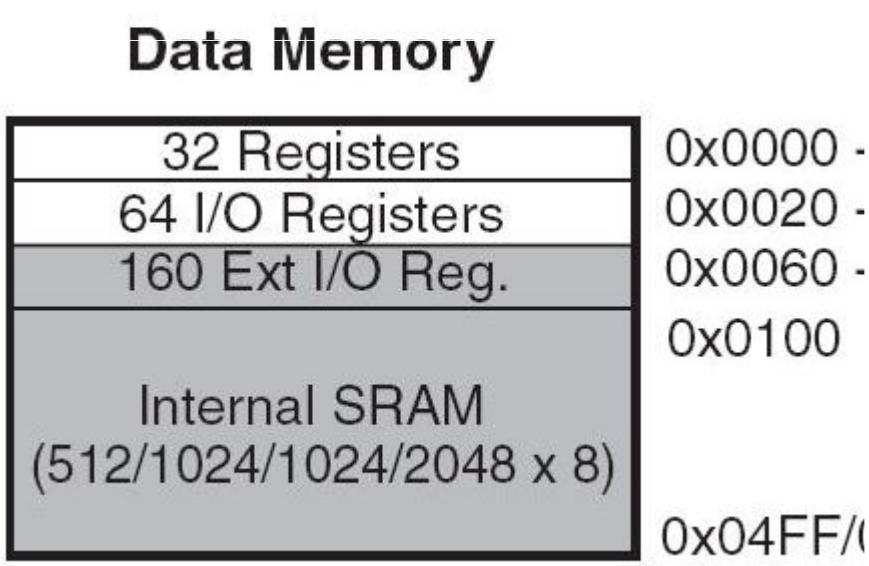

```
; Ej. Calcula el Ca1 de R4
   COM R4 ;R4 ← $FF-R4
```

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

Direccionamiento de E/S Directo (memoria de Datos, zona E/S)

- *La instrucción define el puerto de E/S (0 a 64) y el Registro de propósito general afectados (**Solo instrucciones IN y OUT**).*
  - Ejemplos: IN R1, 56 ; OUT 21, R10

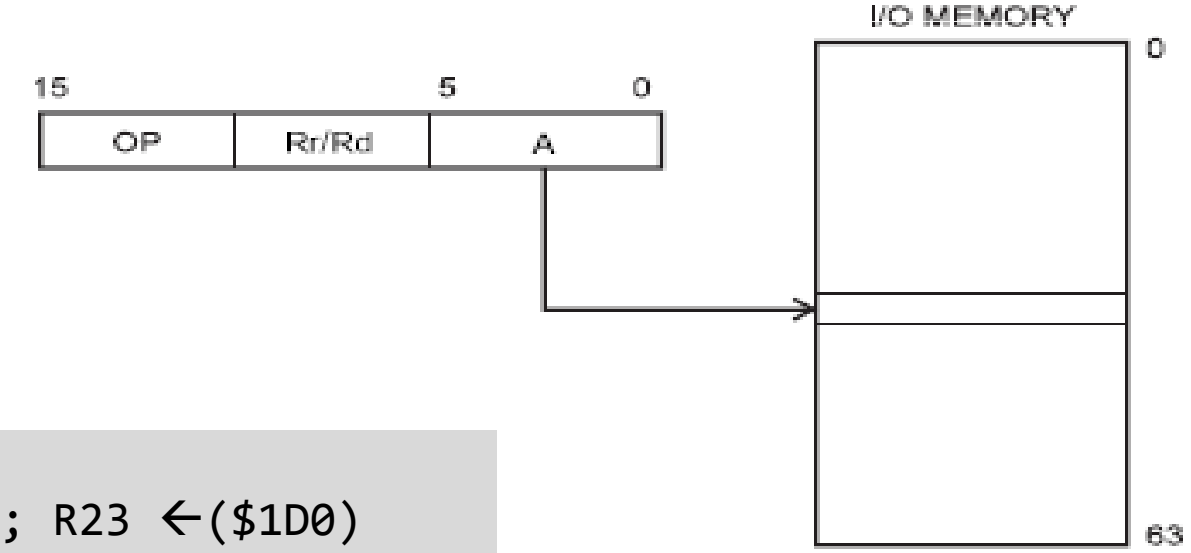





```
; Ejemplos:
    IN R1,56   ;R1← PORT 56
    OUT 21,R10 ;PORT21 ←R10
```

Direccionamiento Directo (*"Straight"*) (memoria de Datos)

- *La instrucción contiene la dirección de memoria (16bits) del operando , así como un campo de 5 bits (Rr/Rf) que identifica el registro destino o fuente.*
- Instrucciones **LDS, STS**.

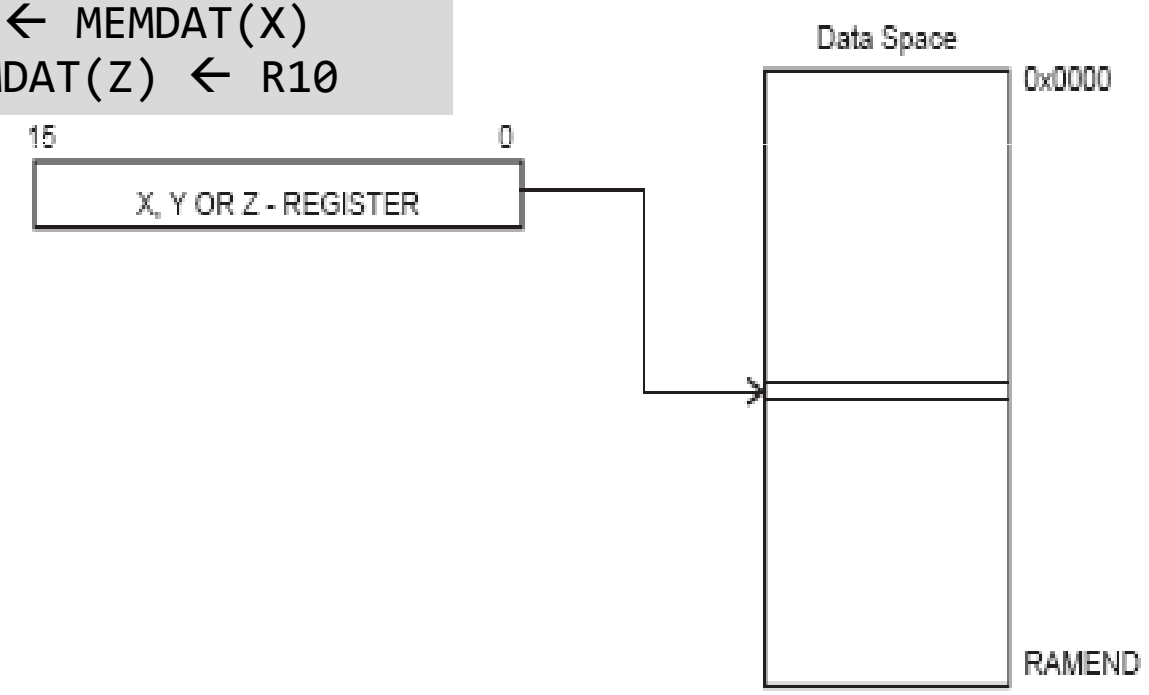



```
; Ejemplos:
    LDS R23,$1D0 ; R23 ←($1D0)
    STS $1D2,R1 ; ($1D2) ← R1
```

Direccionamiento Indirecto (memoria de Datos)

- *La instrucción referencia al registro **X,Y o Z** que contiene la **dirección del operando**.*

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

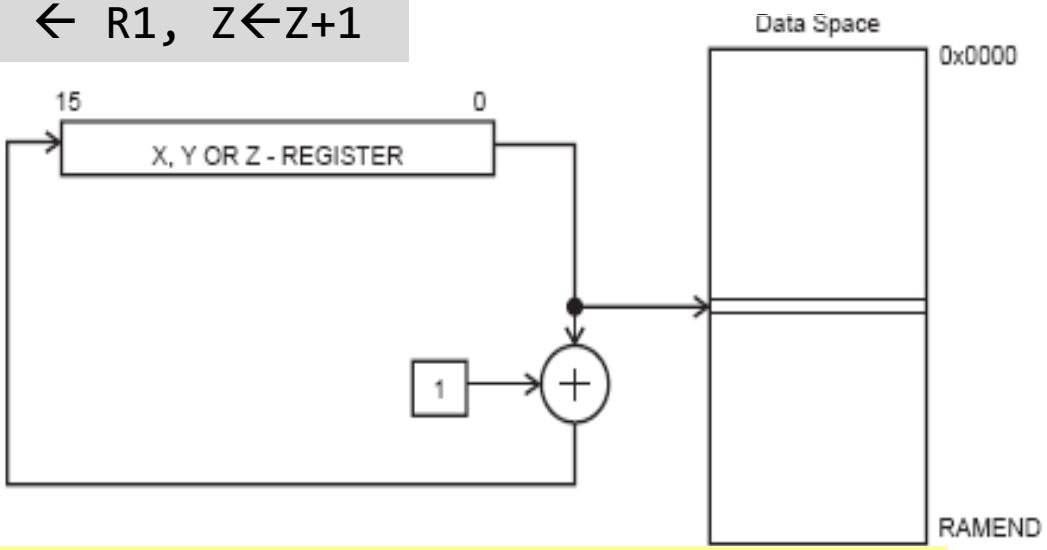


Los Registros X, Y Z actúan como "punteros" al dato en memoria

Direccionamiento Indirecto con postincremento (memoria de Datos)

- *La instrucción referencia al registro X,Y o Z que contiene la dirección del operando, que después se incrementa en una unidad.*

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

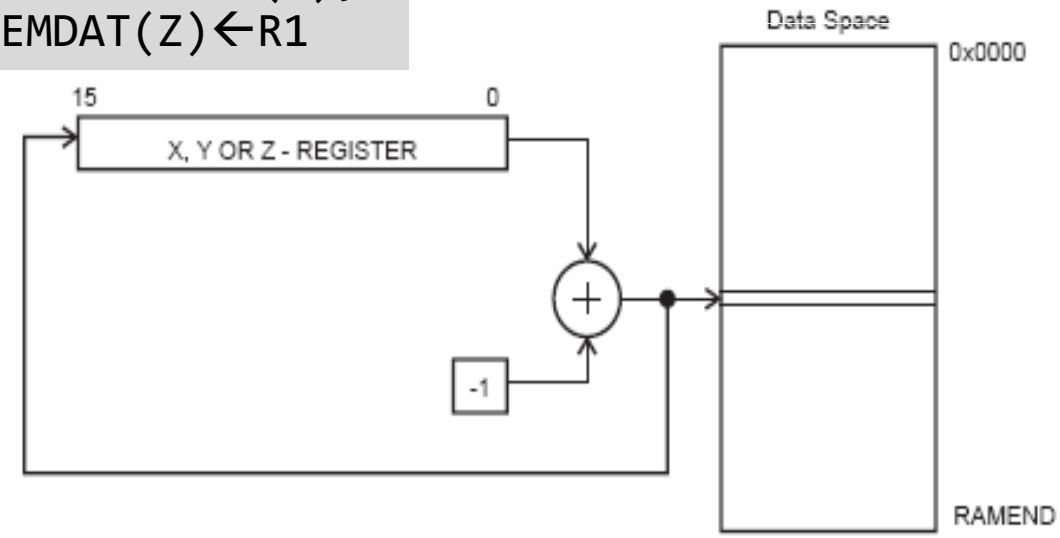


Es muy útil para recorrer los datos de una tabla.
Después de direccionar un dato, el puntero (X, Y ó Z) apunta al Dato siguiente de la tabla

Direccionamiento Indirecto con predecremento (memoria de Datos)

- *La instrucción referencia al registro X,Y o Z que tras decrementarse en una unidad, contiene la dirección del operando.*

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

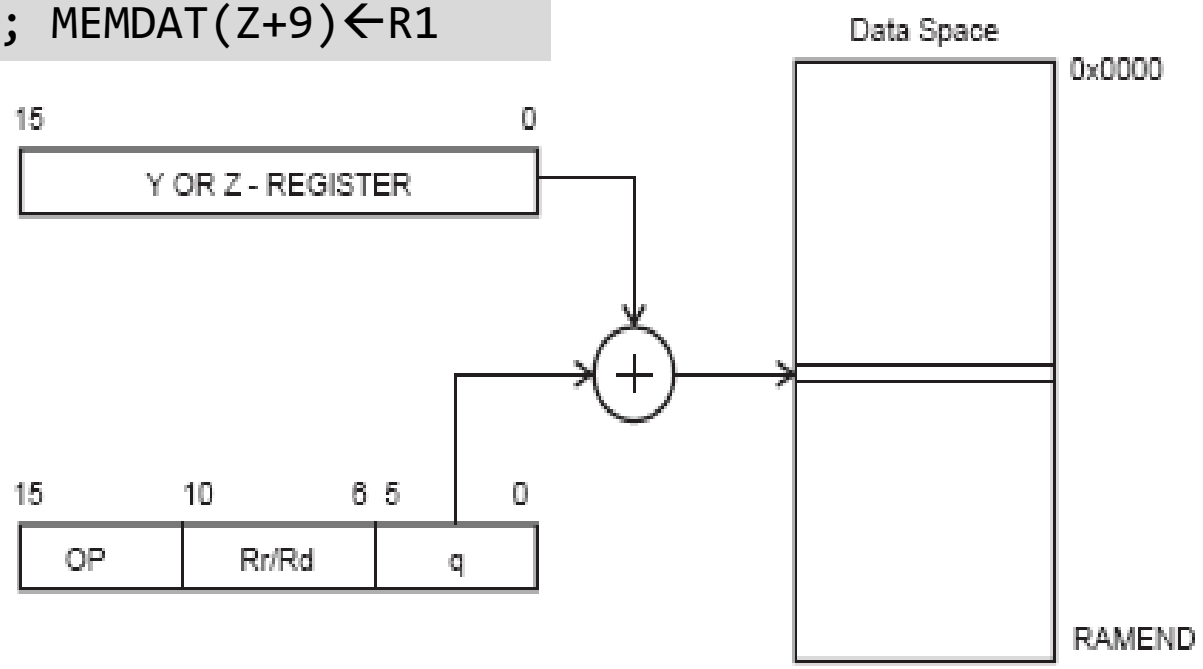


Es muy útil para recorrer los datos de una tabla en orden inverso. Antes de direccionar un dato, el puntero (X, Y ó Z) se decrementa

Direccionamiento Indirecto con desplazamiento (memoria de Datos)

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

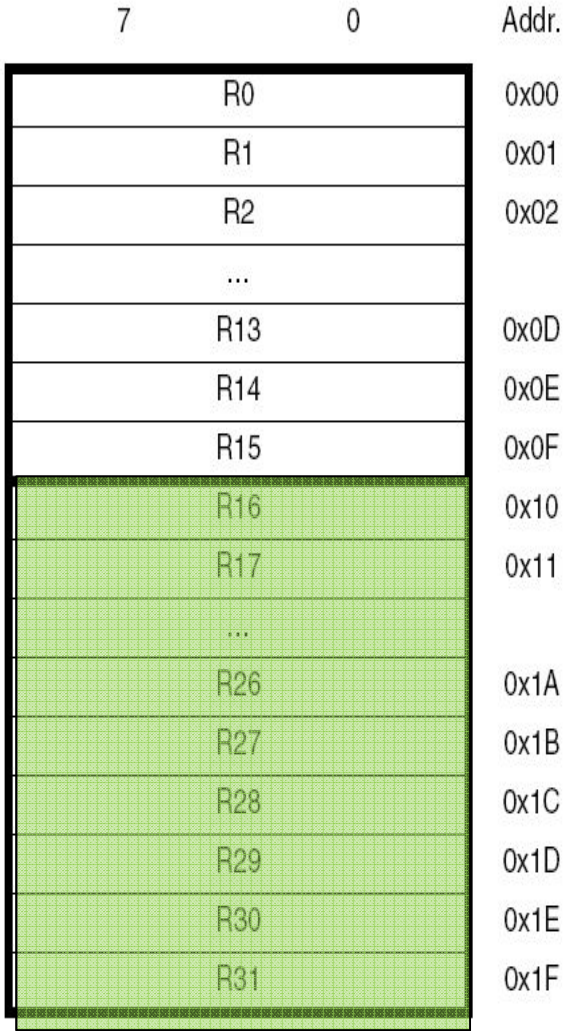
```
; Ejemplos:
    LDD R0,Y+10     ; R0 ←MEMDAT(Y+10)
    STD Z+9,r1      ; MEMDAT(Z+9)←R1
```



Direccionamiento Inmediato (memoria de Datos)

- *El operando es suministrado en la propia instrucción*
  - *En modo inmediato solo pueden usarse como destino los reg. del R16 al R31*

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

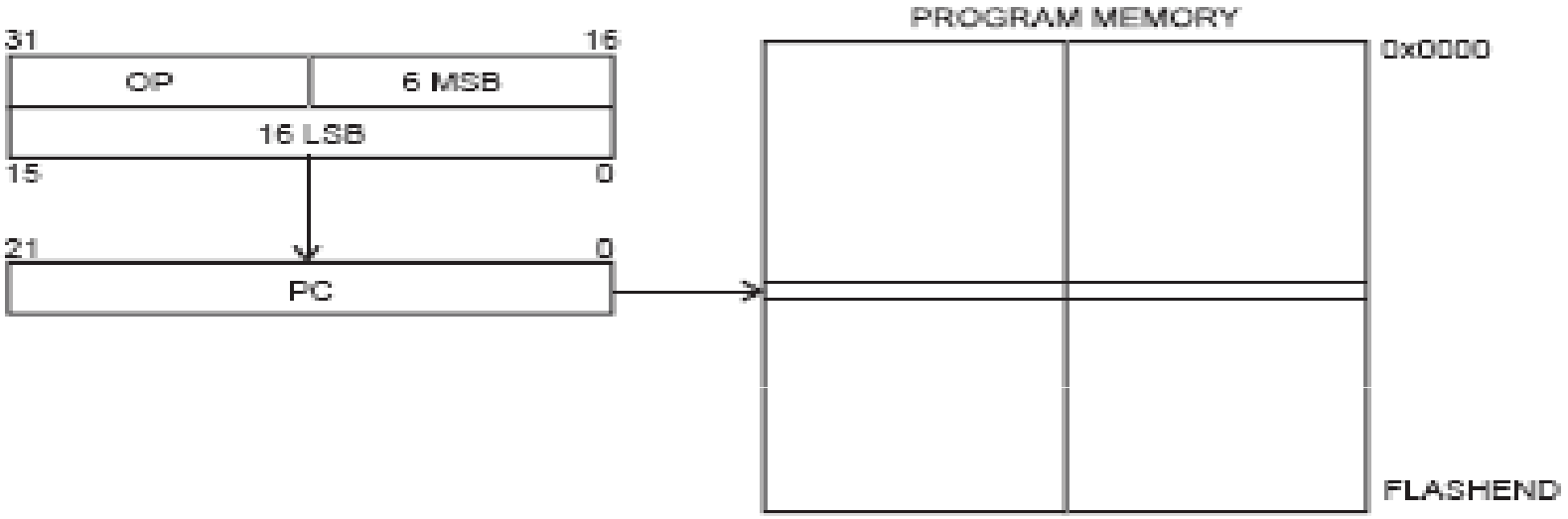


Índice

5.2 Modos de Direccionamiento para la memoria de programa

- Constantes de programa (fuera del alcance de esta asignatura).
  - Indirecto
  - Indirecto con postincremento
- ***Instrucciones***
  - Directo
  - Indirecto
  - Relativo

Direccionamiento Directo de Instrucciones (memoria de programa)

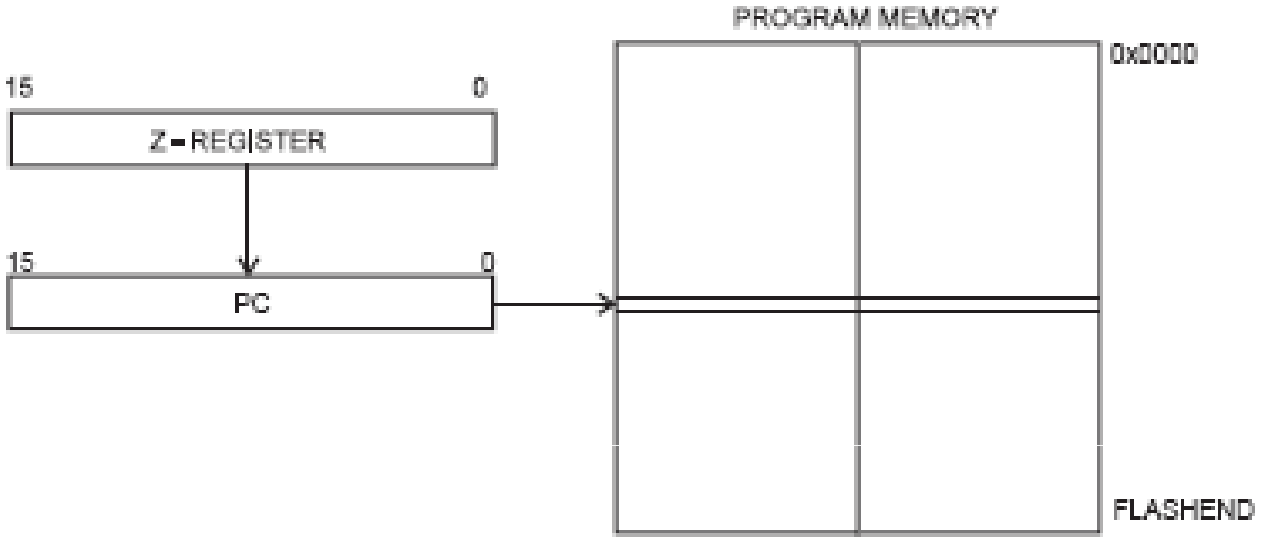


- *El programa continua su ejecución en la dirección de memoria de programa indicada en la instrucción*
- *Instrucciones:* ***JMP*** *y* ***CALL***

```
; 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 (memoria de programa)

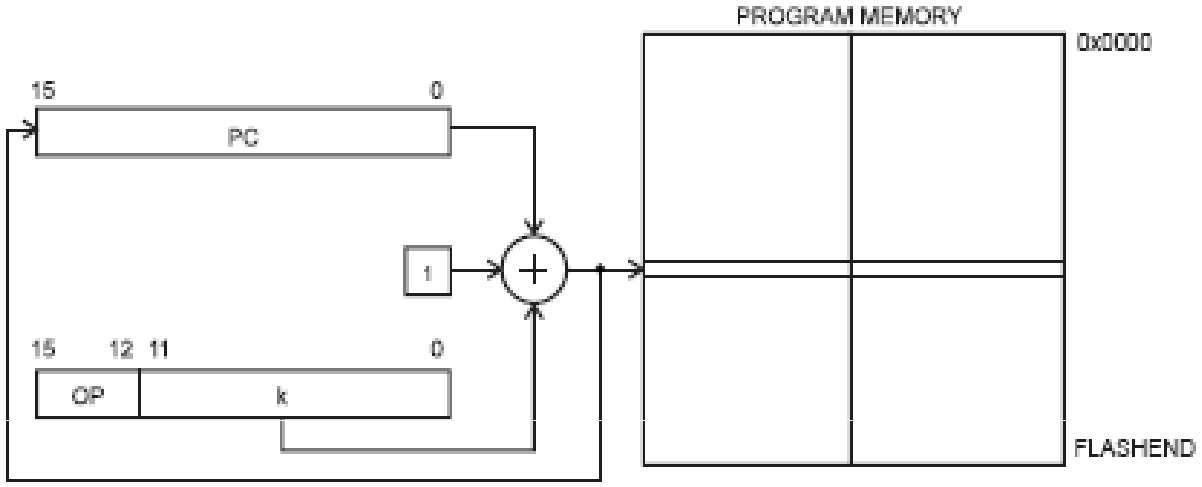


- *El programa continua su ejecución en la dirección de memoria almacenada en el Registro Z*
- *Instrucciones:* ***IJMP*** *y* ***ICALL***

```
; 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 (memoria de programa)



- *El programa continua su ejecución en la dirección de memoria* ***PC ← PC+K+1*** *(k Є [-2048, 2047]*
- *Instrucciones:* ***RJMP*** *(Relative Jump)*

  RCALL *(Relative Call)*

  BRXX *(Branch if condition XX)*

Índice

6. Juego de Instrucciones

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

Juego de Instrucciones AVR

- Instrucciones sin operandos
  - Mnemónico
- Instrucciones con un operando
  - Mnemónico opfuente/opdestino
- 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

MOV	Rd,Rr	Copiar registro	d,r∈[0,31]	Rd←Rr	Ninguno	1
MOVW	Rd,Rr	Copiar registro W	d,r∈[0,2,..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<64K	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]	Rd← (Y+q) Rd← (Z+q)	Ninguno	2

Instrucciones de Transferencia de Datos (ii)

STS	k, Rr	Almacenar dato en memoria	r∈[0,31] k<64K	(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]	(Y+q)←Rr (Z+q)← Rr	Ninguno	2
LPM	Rd,Z Rd,Z+	Carga memoria de programa		R0← (Z) Rd← (Z) Rd← (Z); Z←Z+1	Ninguno	3
SPM		Almacenar en memoria de programa		(Z) ← R1:R0	Ninguno	-

Instrucciones de Transferencia de Datos (iii)

IN	Rd,P	Entrada del puerto	d∈[0,31] P∈[0,63]	Rd←ES(P)	Ninguno	1
OUT	P,Rr	Salida hacia el puerto	r∈[0,31] P∈[0,63]	ES(P)← Rr	Ninguno	1
PUSH	Rr	Empujar en pila	r∈[0,31]	STACK←Rr	Ninguno	2
POP	Rd	Sacar de pila	d∈[0,31]	Rd← STACK	Ninguno	2

Instrucciones de Transferencia de Datos

Instrucciones de Transferencia de Datos

Instrucciones Aritmético-Lógicas

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]	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]	Rd+1:Rd← Rd+1:Rd - K	Z,N,V,C	2
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
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

Instrucciones Aritmético-Lógicas (ii)

CLR	Rd	Poner a cero	d,r∈[0,31]	Rd←0	Z,N,V	1
SER	Rd	Poner todo a 1	d,r∈[0,31]	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∈[0,31]	R1:R0←Rd x Rr	Z,C	2
MULSU	Rd,Rr	Multiplica signo con sin signo	d,r∈[0,31]	R1:R0←Rd x Rr (Rd signed Rr unsigned)	Z,C	2

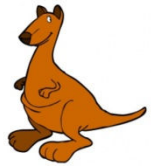
Instrucciones de salto

Permiten saltar a una instrucción en otra zona de memoria de código

4 tipos de instrucciones de salto

JMP: salto incondicional (jump)

BRxx : salto "corto" si se cumple una condición 'xx' (branch)



SKxx : "sáltate" (esquiva) la siguiente instrucción si se cumple la condición (skip)

CALL/RET: llamada/retorno de subrutina

Salto incondicional directo

JMP dir

El programa sigue ejecutándose a partir de la dirección suministrada en la propia instrucción

PC ← dir

```
;Ejemplo de salto incondicional directo
;ponemos una etiqueta y el ensamblador calcula
;la dirección del salto

          mov   r1,r0
          jmp   farplc      ;salto incondicional
          . . . .
farplc:   nop
```

Salto incondicional relativo

RJMP k

La dirección de salto se obtiene sumando una constante con signo (k) al PC

PC ← PC+k+1 $-63 \leq k < 64$

```
;Ejemplo de salto incondicional relativo
;ponemos una etiqueta y el ensamblador calcula
;la constante k
; permite saltos "cortos" de +/- 2K

          mov   r1,r0
          rjmp ok    ;salto incondicional relativo
          . . . .
ok:       nop
```

Salto incondicional indirecto

IJMP

La dirección de salto está en el registro Z

PC ← Z (esto es: PC ←r31:r30)

```
;Ejemplo de salto incondicional indirecto
;
    . . . .
    ldi  r30,0      ;parte baja de la dir. de salto
    ldi  r31,$8     ;parte alta de la dir. de salto
    ijmp            ;salto a la dirección $800
    . . . .
```

Salto condicional relativo

BRxx k

Si se cumple una condición xx, la dirección de salto se obtiene sumando una constante con signo (k) al PC: **PC ← PC+k+1**

La condición de salto xx comprueba el valor de determinados bits del STATUS REGISTER

Ejemplos:

BREQ: Branch if EQual (Z=1)

BRNE: Branch if Not Equal (Z=0)

BRCS: Branch if Carry Set (CY=1)

BRCC: Branch if Carry Clear (CY=0)

BRSH: Branch if Same or Higher (CY=0)

Ejemplo de salto condicional relativo

BRNE Branch if Not Equal(Z=0)

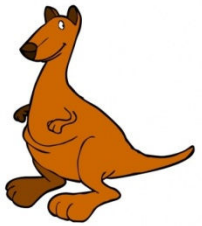
Si Z = 0 entonces PC ←PC+k+1

en otro caso PC ← PC+1 $-63 \leq k < 64$

```
;Ejemplo de salto condicional relativo
;para realizar un bucle
          eor r27,r27     ; borra r27 con una exor
bucle:    . . .
          . . .
          inc r27         ; incrementa r27
          cpi r27,5       ; compara r27 con 5
          brne bucle      ; salta si r27<>5
          nop
```

SKIP condicional

SBxx Rr,b



Esquiva la siguiente instrucción si el bit de un registro cumple una condición

Ejemplos:

SBRC: Skip if Bit in Register is Clear

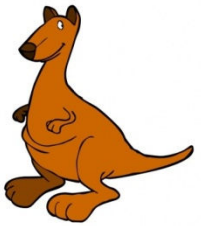
SBRS: Skip if Bit in Register is Set

SBIC: Skip if Bit in I/O Port is Clear

SBIS: Skip if Bit in I/O Port is Set

Ejemplo SKIP condicional

SBRC Rr,b



```
    . . .
    sub r0,r1        ; r0 ← r0-r1
    sbrc r0,7 ;esquiva si el bit 7 de r0 está  a 0
    sub r0,r1 ;sólo se ejecuta si el bit 7 de r0 está a 1
    nop
```

Llamada a subrutina directo

`CALL dir (Long Call to Subrutine)`

Llama a una subrutina almacenada en cualquier posición de la memoria de programa.

PC ← dir

La dirección de retorno se guarda en la pila

PILA ← PC+2

SP ← SP-2 (2bytes)

La subrutina debe terminar con una instrucción `RET`

Ejemplo de llamada a subrutina directo
CALL dir (Long Call to Subrutine)

```
          . . .
          mov r16,r0
          call check ; llamada a subrutina check
          nop        ; continua
          . . .
; nosotros ponemos la etiqueta check y el ensamblador
; calcula la dirección
check:    cpi r16,$42
          breq error
          ret
          . . .
error:    rjmp error      ;bucle infinito
```

Llamada a subrutina relativo

RCALL k (Relative Call to Subrutine)

Llamada a una subrutina ubicada en una posición relativa a la dirección de esa instrucción

PC ← PC+k+1 $-2047 \leq k < +2048$

La dirección de retorno se guarda en la pila

PILA ← PC+1

SP ← SP-2 (2bytes)

La subrutina debe terminar con una instrucción **RET**

Ejemplo de llamada a subrutina relativo
RCALL k (Relative Call to Subrutine)

```
          . . .
          rcall routine    ; llamada a subrutina routine
          . . .
; nosotros ponemos la etiqueta “routine” y el ensamblador
; calcula la dirección
routine:  push r14   ;salva r14 en la pila
          ...
          pop r14    ;restaura r14
          ret
          . . .
```

Resumen de Instrucciones de salto

RJMP	Etiqueta	Salto relativo	-2K< Etiq-PC-1<2K	PC← Etiqueta	Ninguno	2
JMP (1)	Etiqueta	Salto	0<Etiqueta<4M	PC← Etiqueta	Ninguno	2
IJMP		Salto indirecto		PC ← Z	Ninguno	3
RCALL	Etiqueta	Llamada a subrutina relativa	-2K< Etiq-PC-1 <2K	STACK ← PC PC←Etiqueta	Ninguno	3
CALL (1)	Etiqueta	Llamada a subrutina	0<Etiqueta<4M	STACK ← PC PC← Etiqueta	Ninguno	3
ICALL		Llamada a subrutina indirecta		STACK ← PC PC ← Z	Ninguno	4
RET		Regreso de subrutina		PC← STACK	Ninguno	4
RETI		Regreso de interrup.		PC← STACK	I	4
CPSE	Rd,Rr	Compara, esquiva si iguales	d,r∈[0,31]	Si Rd=Rr PC← PC+2 (ó 3)	Ninguno	01/02/03
SBRC	Rr,b	Esquiva si el bit está a cero	r∈[0,31] b∈[0,7]	Si (Rd(b)=0) PC← PC+2 (ó 3)	Ninguno	01/02/03
SBRS	Rr,b	Esquiva si el bit está a uno	r∈[0,31] b∈[0,7]	Si (Rd(b)=1) PC← PC+2 (ó 3)	Ninguno	01/02/03

(1) Sólo disponibles para ATMEGA168PA y ATMEGA328PA

Instrucciones de salto (ii)

SBIC	P,b	Esquiva si el bit del puerto está a 0	P∈[0,31] b∈[0,7]	Si (E/S(P,b)=0) PC← PC+2 (ó 3)	Ninguno	1 o 2 o 3
SBIS	P,b	Esquiva si el bit del puerto está a1	P∈[0,31] b∈[0,7]	Si (E/S(P,b)=1) PC← PC+2 (ó 3)	Ninguno	1 o 2 o 3
BREQ	Etiqueta	Salta si iguales	-64 ≤Etiq-PC-1- <64	Si (Z=1) PC← Etiqueta	Ninguno	1 o 2
BRNE	Etiqueta	Salta si distintos	-64 ≤Etiq-PC-1- <64	Si (Z=0) PC← Etiqueta	Ninguno	1 o 2
BRCS	Etiqueta	Salta si C está a 1	-64 ≤Etiq-PC-1- <64	Si (C=1) PC←Etiqueta	Ninguno	1 o 2
BRCC	Etiqueta	Salta si Cestá a 0	-64 ≤Etiq-PC-1- <64	Si (C=0) PC← Etiqueta	Ninguno	1 o 2
BRSH	Etiqueta	Salta si igual o mayor	-64 ≤Etiq-PC-1- <64	Si (C=1) PC← Etiqueta	Ninguno	1 o 2

Instrucciones de salto (iii)

BRLO	Etiqueta	Salta si menor	-64 ≤Etiq-PC-1- <64	Si (C=0) PC← Etiqueta	Ninguno	1 o 2
BRMI	Etiqueta	Salta si negativo	-64 ≤Etiq-PC-1- <64	Si (N=1) PC← Etiqueta	Ninguno	1 o 2
BRPL	Etiqueta	Salta si positivo	-64 ≤Etiq-PC-1- <64	Si (N=0) PC← Etiqueta	Ninguno	1 o 2
BRHS	Etiqueta	Salta si H está a 1	-64 ≤Etiq-PC-1- <64	Si (H=1) PC← Etiqueta	Ninguno	1 o 2
BRHC	Etiqueta	Salta si H está a 0	-64 ≤Etiq-PC-1- <64	Si (H=0) PC← Etiqueta	Ninguno	1 o 2
BRTS	Etiqueta	Salta si T está a 1	-64 ≤Etiq-PC-1- <64	Si (T=1) PC← Etiqueta	Ninguno	1 o 2
BRTC	Etiqueta	Salta si T está a 0	-64 ≤Etiq-PC-1- <64	Si (T=0) PC← Etiqueta	Ninguno	1 o 2
BRVS	Etiqueta	Salta si V está a 1	-64 ≤Etiq-PC-1- <64	Si (V=1) PC← Etiqueta	Ninguno	1 o 2
BRVC	Etiqueta	Salta si V está a 0	-64 ≤Etiq-PC-1- <64	Si (V=0) PC← Etiqueta	Ninguno	1 o 2
BRIE	Etiqueta	Salta si I está a 1	-64 ≤Etiq-PC-1- <64	Si (I=1) PC← Etiqueta	Ninguno	1 o 2
BRID	Etiqueta	Salta si I está a 0	-64 ≤Etiq-PC-1- <64	Si (I=0) PC← Etiqueta	Ninguno	1 o 2

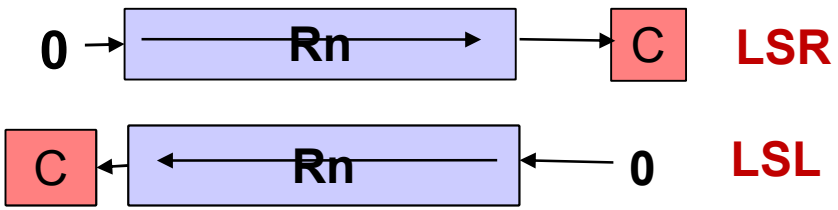
Instrucciones de salto (iv)

BRGE	Etiqueta	Salta si mayor o igual, (signo)	-64 ≤Etiq-PC-1- <64	Si (N⊕V=0) PC← Etiqueta	Ninguno	1 o 2
BRLT	Etiqueta	Salta si menor (signo)	-64 ≤Etiq-PC-1- <64	Si (N⊕V=1) PC← Etiqueta	Ninguno	1 o 2

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

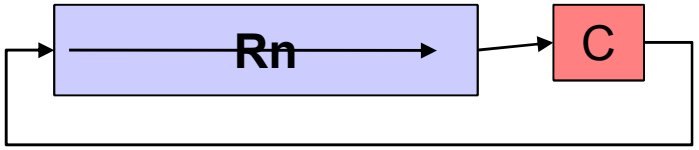
Instrucciones de bit y bit-test

LSL	Rd	Desplazamiento a la izquierda	d∈[0,31]	Rd(n+1) ←Rd(n), Rd(0) ←0, C← Rd(7)	Z,C,N,V,H	1
LSR	Rd	Desplazamiento a la derecha	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	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	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	d∈[0,31]	Rd(n) ←Rd(n+1), Rd(7) ←Rd(7), C← Rd(0)	Z,C,N,V,	1

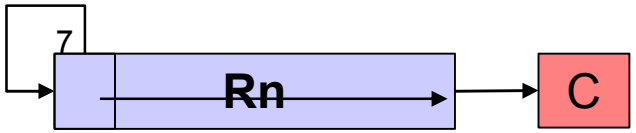


LSR

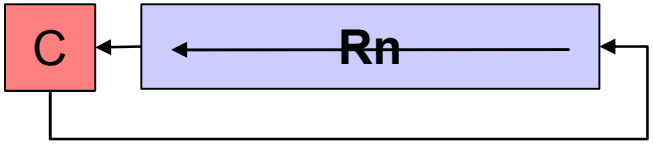
LSL



ROR



ASR



ROL

Instrucciones de bit y bit-test

SWAP	Rd	Intercambia nibbles	d∈[0,31]	Rd(3..0)←→Rd(7.4)	Ninguno	1
SBI	P,b	Poner a 1 el bit b del puerto IO	b∈[0,7] P∈[0,31]	IO(P,b)←1	Ninguno	2
CBI	P,b	Poner a 0 el bit b del puerto IO	b∈[0,7] P∈[0,31]	IO(P,b)←0	Ninguno	2
SEcc	[1]	Poner a 1 el bit cc del registro de estado			cc	1
CLcc	[1]	Poner a o el bit cc del registro de estado			cc	1

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

Instrucciones de control

NOP		Nada			Ninguno	1
BREAK		Para depuración			Ninguno	N/A
WDR		Reinica el temporizador del perro guardián			Ninguno	1
SLEEP		Dormir			Ninguno	1

Índice

Directivas de ensamblador

- *Son comandos al programa que genera el código objeto y que se encuentran mezclados en el fichero fuente con las instrucciones del microcontrolador*.

CSEG-Code Segment

Sintaxis: `.CSEG`

DSEG-Data Segment

Sintaxis: `.DSEG`

BYTE - Reserva bytes a una variable en memoria

Reserva en memoria de datos. Posible sólo en DSEG

Sintaxis: label: .BYTE expresión

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

Directivas de ensamblador (ii)

- *Son comandos al programa que genera el código objeto y que se encuentran mezclados en el fichero fuente con las instrucciones del microcontrolador*.

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 (iii)

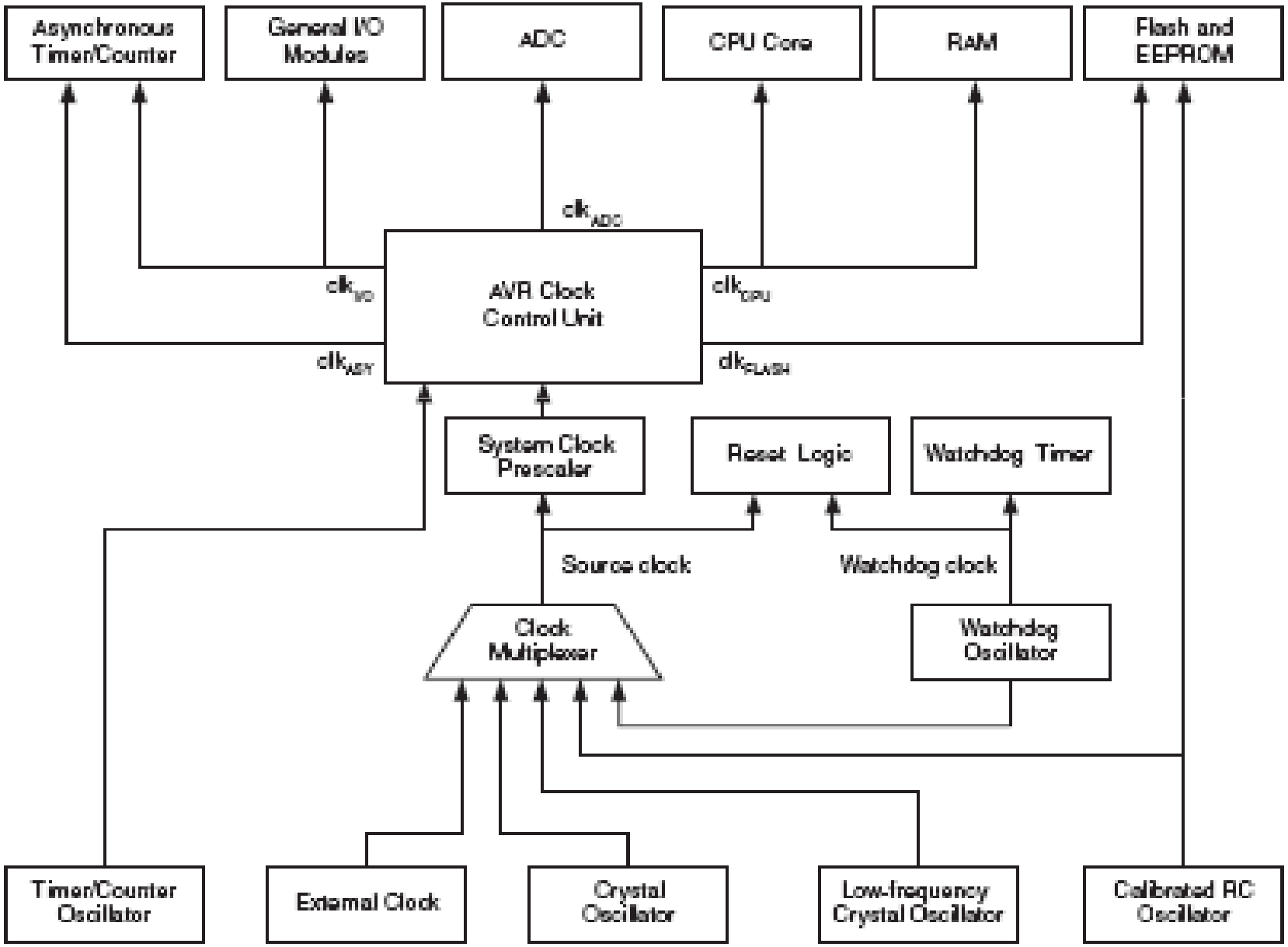
ORG – Establece la dirección inicial de memoria

Sintaxis: .ORG expression

```
.DSEG           ;comienza el segmento de datos
.ORG  0X37       ;Dirección $37 de la memoria de datos
                  ;Si no se indica dirección, por defecto $60
Variable: .BYTE 1
.CSEG           ;comienza el segmento de código
.ORG  0x10      ;dirección $10 de la memoria de programa
mov  r0,r1
```

Índice (ii)

Reloj de sistema y opciones de reloj (i)



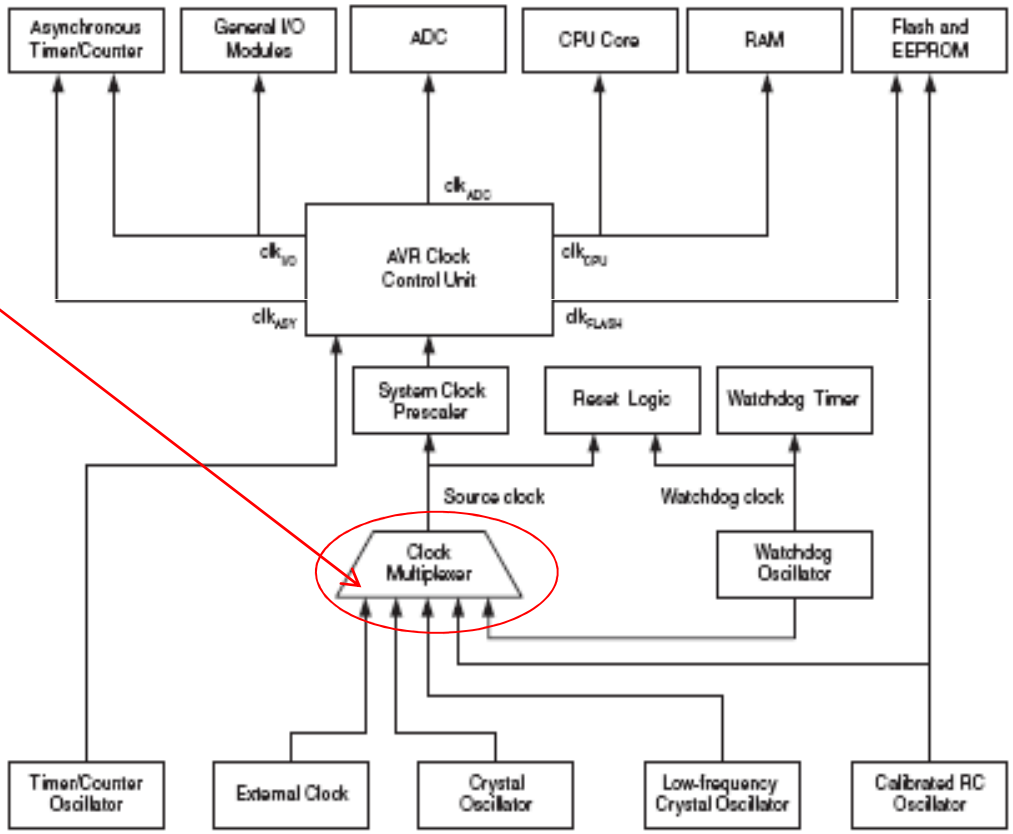
Reloj de sistema y opciones de reloj (ii)

Fuentes

- Externo
- Oscilador de Cristal
  - RC calibrado
  - Watch-Dog

Actúan sobre:

- AVR clock control unit mediante un Prescaler
- Circuito de Reset



Reloj de sistema y opciones de reloj (iii)

- La fuente de reloj se configura mediante la programación de unos ***"fusibles"***
- De fábrica RC a 8MHz con Prescaler de 8.
- ***Secuencia de puesta en marcha del reloj*** (Clock startup sequence)

  - *Esperar un voltaje adecuado de Vcc.*

    - Circuito de RESET y un posterior Timeout llevado a cabo por el Watchdog Timer.

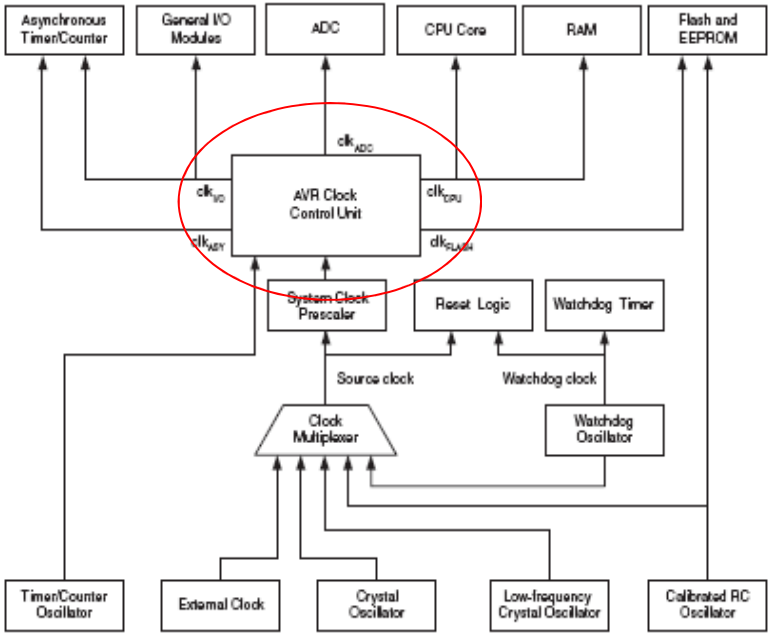
  - *Un número previo de oscilaciones*.

    - Fusibles

Reloj de sistema y opciones de reloj (iv)

Unidad de control y distribución de reloj

- Adapta el consumo al tipo de aplicación Permite apagar aquellos módulos que no sean usados
- Adapta el consumo al tipo de aplicación.
- Modos bajo consumo (instrucción Sleep)



Índice (ii)

Circuito de RESET

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

- Los registros de E/S toman sus valores por defecto.
- Se busca la instrucción en la posición asociada al vector de RESET.

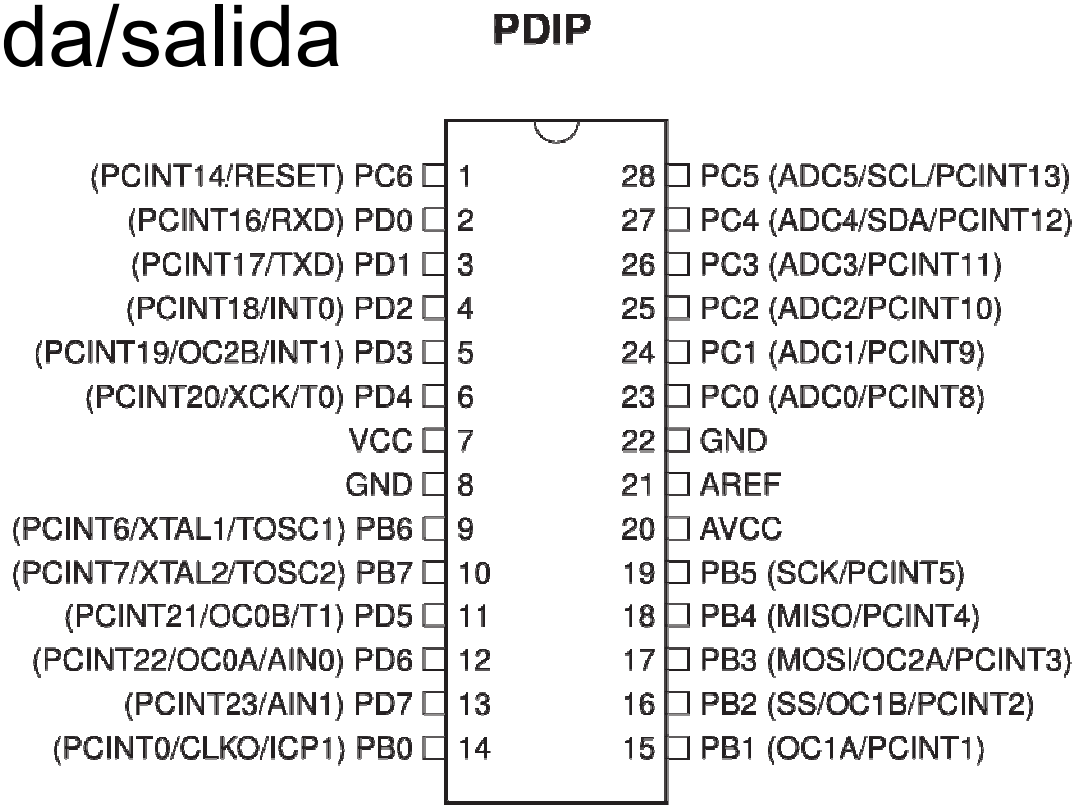
Causas que generan un RESET:

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

Índice (ii)

Entrada/Salida

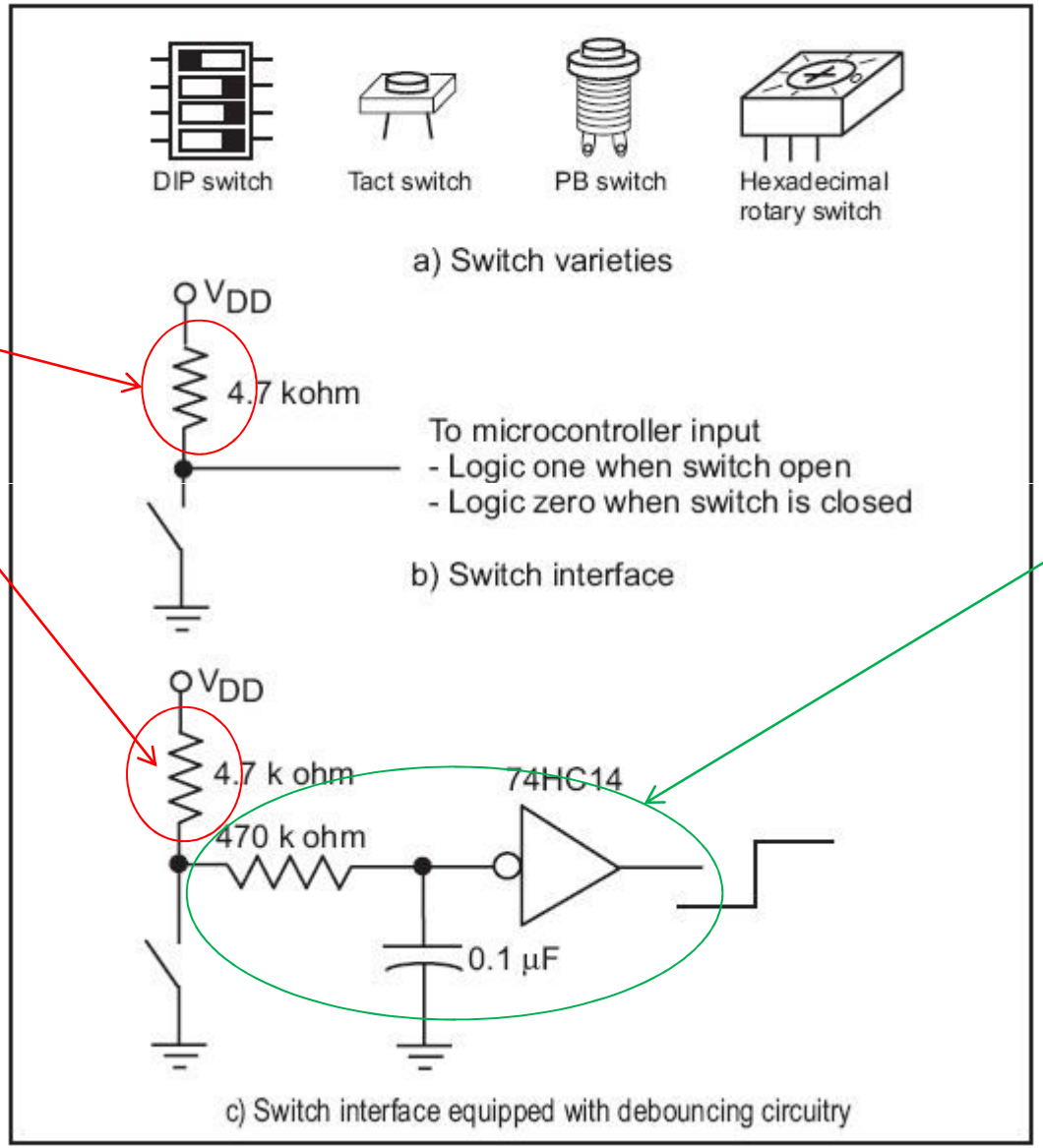
- Puertos de entrada/salida
  - **PB_{7-0}**
  - **PC_{6-0}**
  - **PD_{7-0}**
- Temporizadores
  - **Timer 0**
  - **Timer 1**



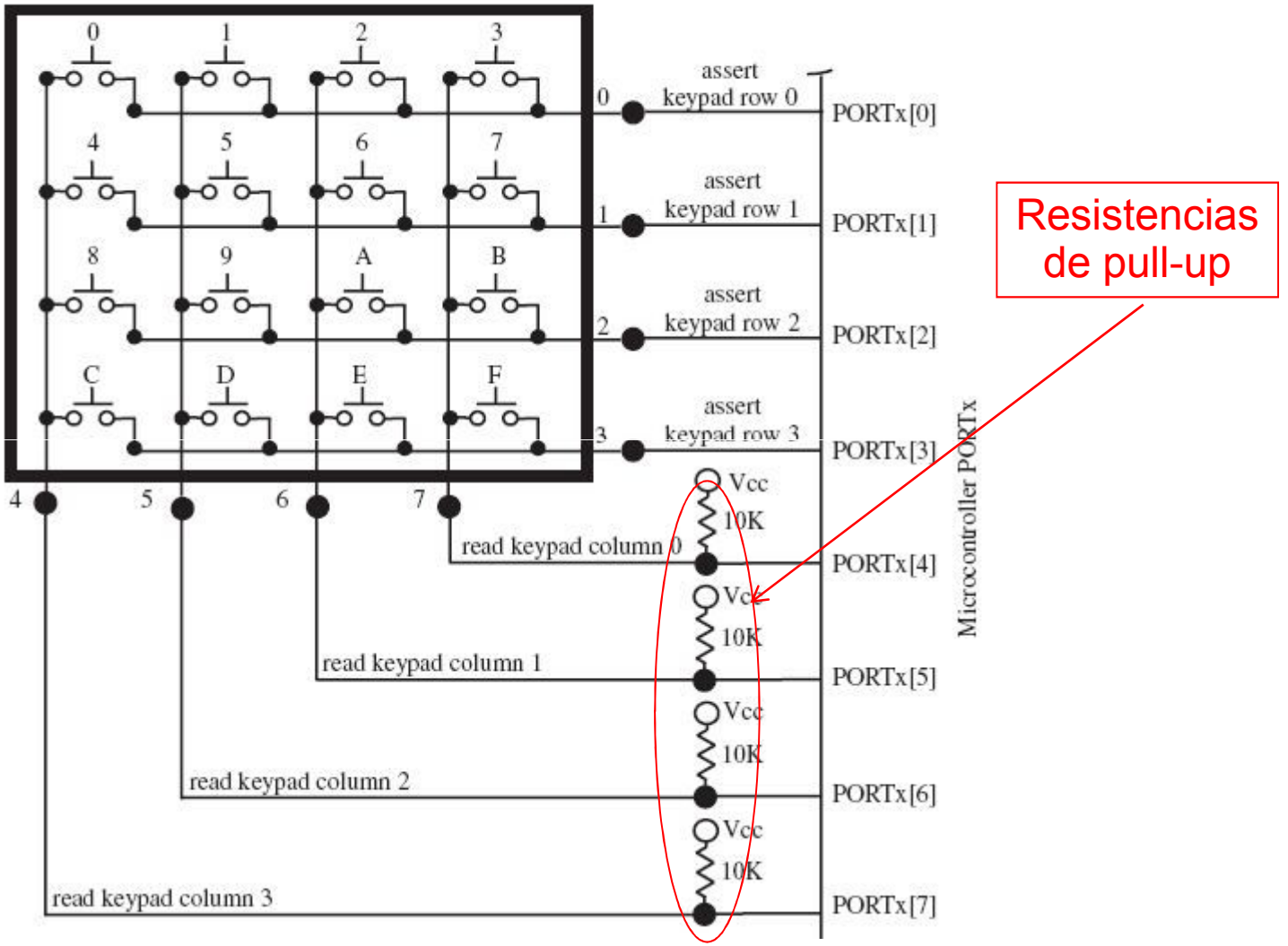
Ejemplos de dispositivos de Entrada

Resistencia de pull-up

Eliminador de rebotes



Ejemplos de dispositivos de Entrada



Teclado matricial: Con 8 líneas de entrada leemos 16 pulsadores

Puertos E/S

- Son ***entradas/salidas digitales*** que permiten ***leer/escribir*** valores lógicos en los pines.
  - Típicamente: "1"=5V, "0" =0V
- 3 puertos de 8 bits**: Puerto B, Puerto C y Puerto D**
- Los puertos tienen funciones alternativas. Ej.
  - PB0: "*Capturador de eventos del temporizador*" o "*Salida de reloj interno*" o "*Interrupción 0 ante cambio en el PIN*"
- Activar la función alternativa de un PIN no afecta al resto del pines del puerto

Acceso a Puertos y Timer1

Puertos B, C y D

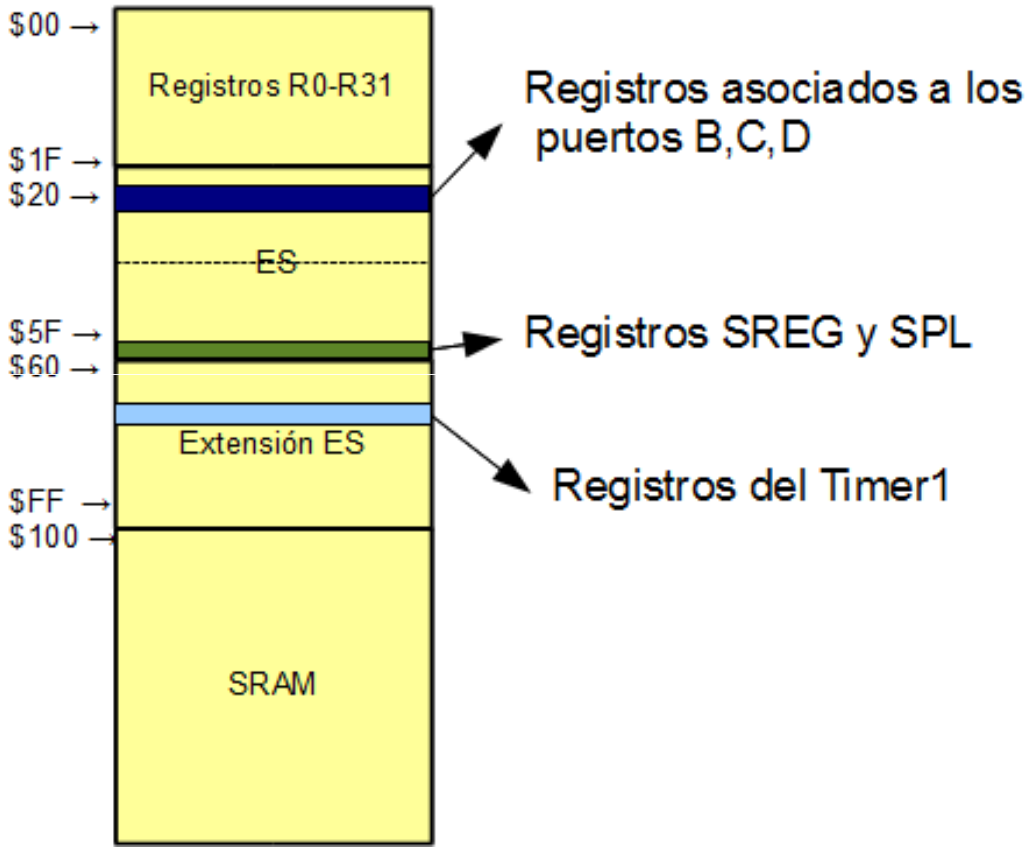
- Zona baja de E/S
- Permite **SBI, CBI, IN, OUT, SBIC,SBIS**,
- modos directos e indirectos

Registros SREG y SPL

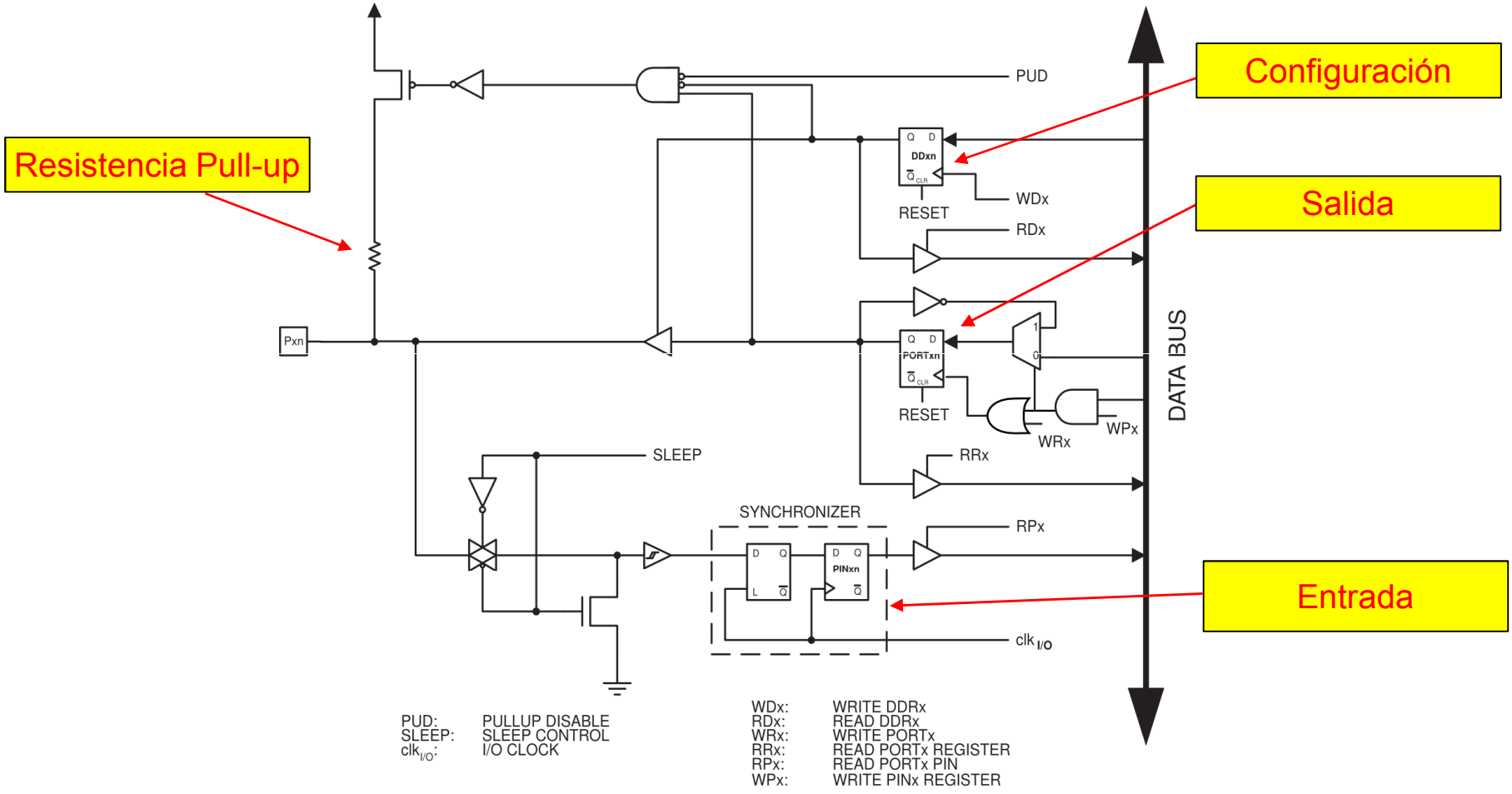
- Zona alta de la ES
- Permite **IN, OUT**
- modos directos e indirectos

Timer1

- Extensión E/S
- Sólo modos directos e indirectos



Esquema general de un PIN de un Puerto E/S



Puertos E/S

Los pines del puertos tienen resistencias de Pull-Up que pueden activarse

Tienen un circuito de sincronización para leer los valores lógicos.

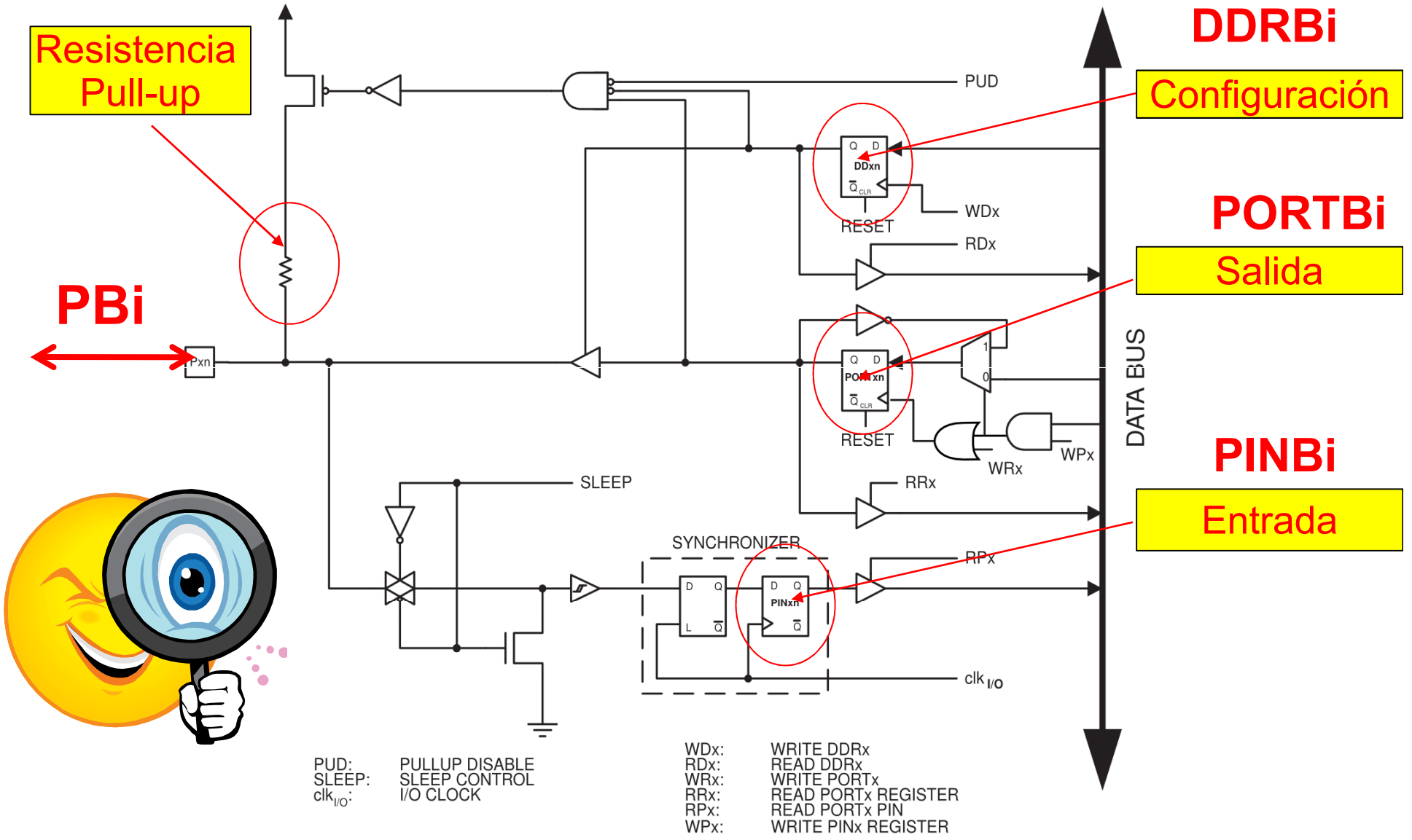
Cada puerto tiene asociado 3 registros: (S={B,C,D})

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

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

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

Esquema general del pin i del Puerto B



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	

$PINB_{7-0}$ (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):

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	

- DDRBi configurado como salida:

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

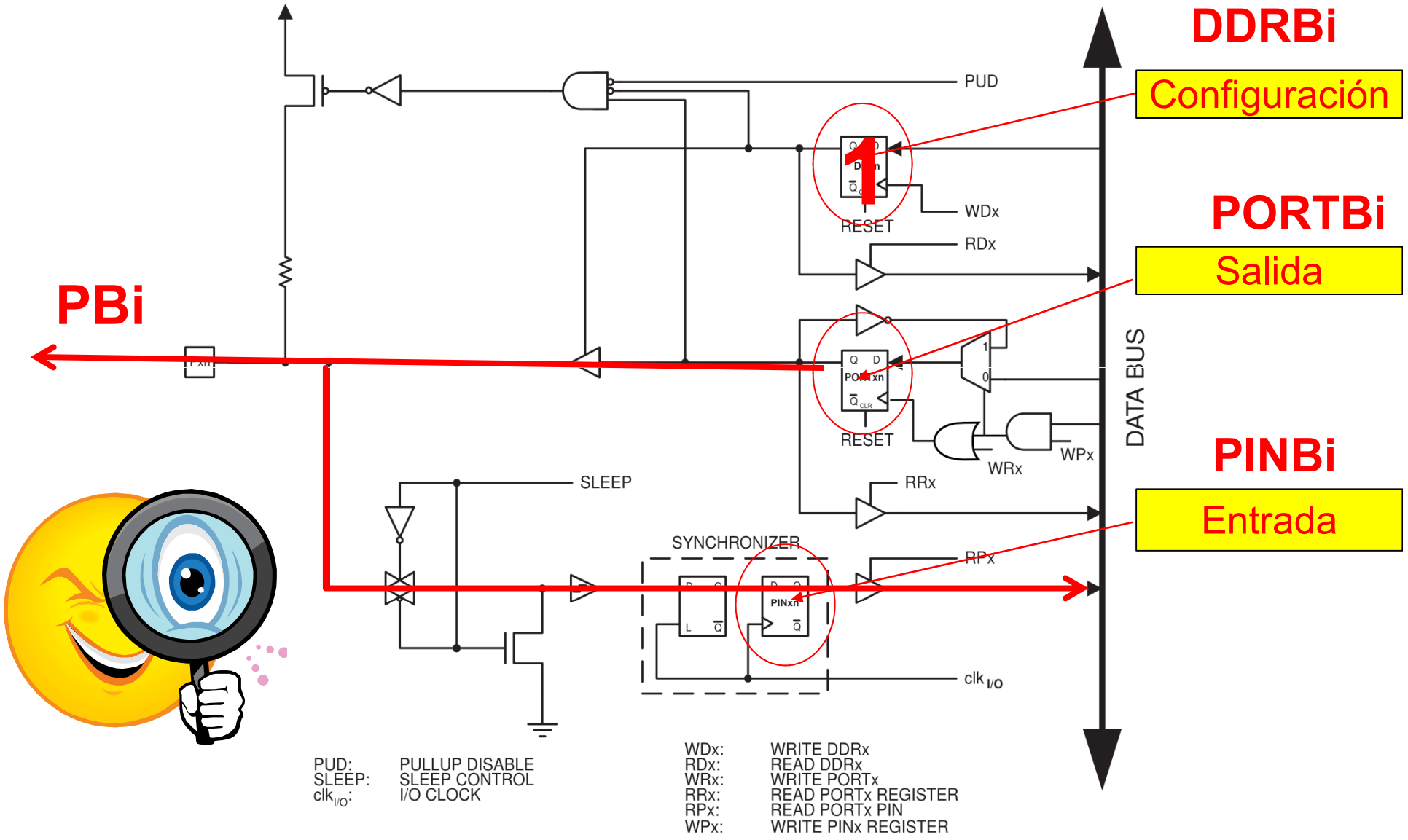
- DDRBi configurado como entrada:

  Se activa la resistencia de pull-up del PIN Pbi al escribir un "1" en el PORTBi

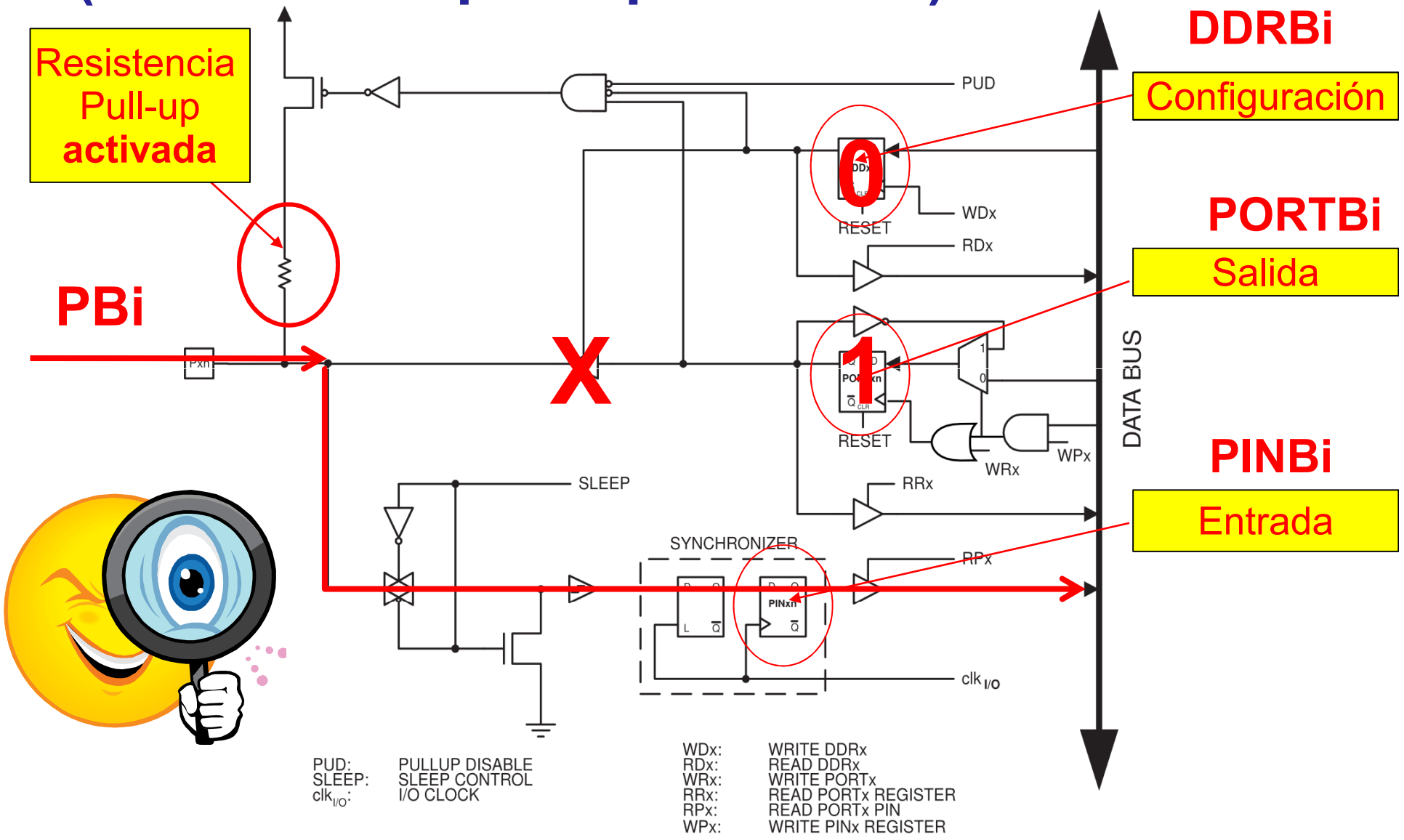
Puertos E/S

Puerto	Registro	Dirección
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)

Ej. PIN PBi configurado como salida



Ej. PIN PBi configurado como entrada (Resistencia pull-up activada)



Puertos E/S

Ejemplo 1: Establecer el puerto B como salida y activar el PINB3 a '1' y el PINB6 a '0'

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

Puertos E/S

Ejemplo 2: Establecer el PINC4 del puerto C como entrada y tomar una decisión en función del valor leído:

```
CBI  DDRC,4    ;PIN4 como entrada DDRC4=0
SBIS PIND,4    ;Esquiva una instrucción si PIND4=1
JMP  PD4_ES_0  ;Salto si PIND4=0
JMP  PD4_ES_1  ;Salto si PIND4=1
```

Ejercicio: Escribir un código equivalente con la instrucción SBIC

Índice (ii)

Interrupciones

- ¿Qué es una interrupción?

  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.

- ¿Qué se necesita para procesar interrupciones?
  - Pila. Almacena la dirección de la instrucción del programa interrumpido.
  - Rutina de interrupción ***instalada*** debidamente según su vector de interrupción.
  - Guardar la información de estado de la CPU.

Interrupciones

- El SP del AtmegaX8pa se inicia automáticamente a la posición más alta de la memoria de datos.
- 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 requiere instrucción JMP.
  - Modelos Atmega48pa y Atmega88pa requiere instrucción RJMP.
- La tabla de vectores de interrupción depende del modelo.

Tabla de verctores 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 incialización 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
ox00A       jmp PCINT2         ; PCINT2 Handler
ox00C       jmp WDT            ; watchdog timer Handler
ox00E       jmp TIM2_COMPA     ; Timer2 Compare A Handler
ox010       jmp TIM2_COMPB     ; Timer2 Compare B Handler
ox012       jmp TIM2_OVF       ; Timer2 Overflow Handler
```

Gestión de interrupciones

- Para poder procesar interrupciones, el bit **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
  - La rutina de interrupción termina con RETI (recupera el PC de la PILA)

- Durante la ejecución de la interrupción, I=0 para evitar interrupciones anidadas

- La rutina de interrupción debe salvar en la pila, al menos, el registro de estado

Ejemplo de rutina de interrupción

```
                .CSEG
                .ORG 0
                JMP main
                JMP IRQ0_handler
                JMP IRQ1_handler
                . . .

Main:           . . .
                {Código programa principal}
                . . .
IRQ0_handler:   IN r16,SREG
                PUSH r16
                {Código interrupción IRQ0}
                POP r16
                OUT SREG,r16
                RETI
IRQ1_handler:  IN r16,SREG
                PUSH r16
                {Código interrupción IRQ1}
                POP r16
                OUT SREG,r16
                RETI
```

Índice (ii)

Temporizadores

Hay 3 timers disponibles.

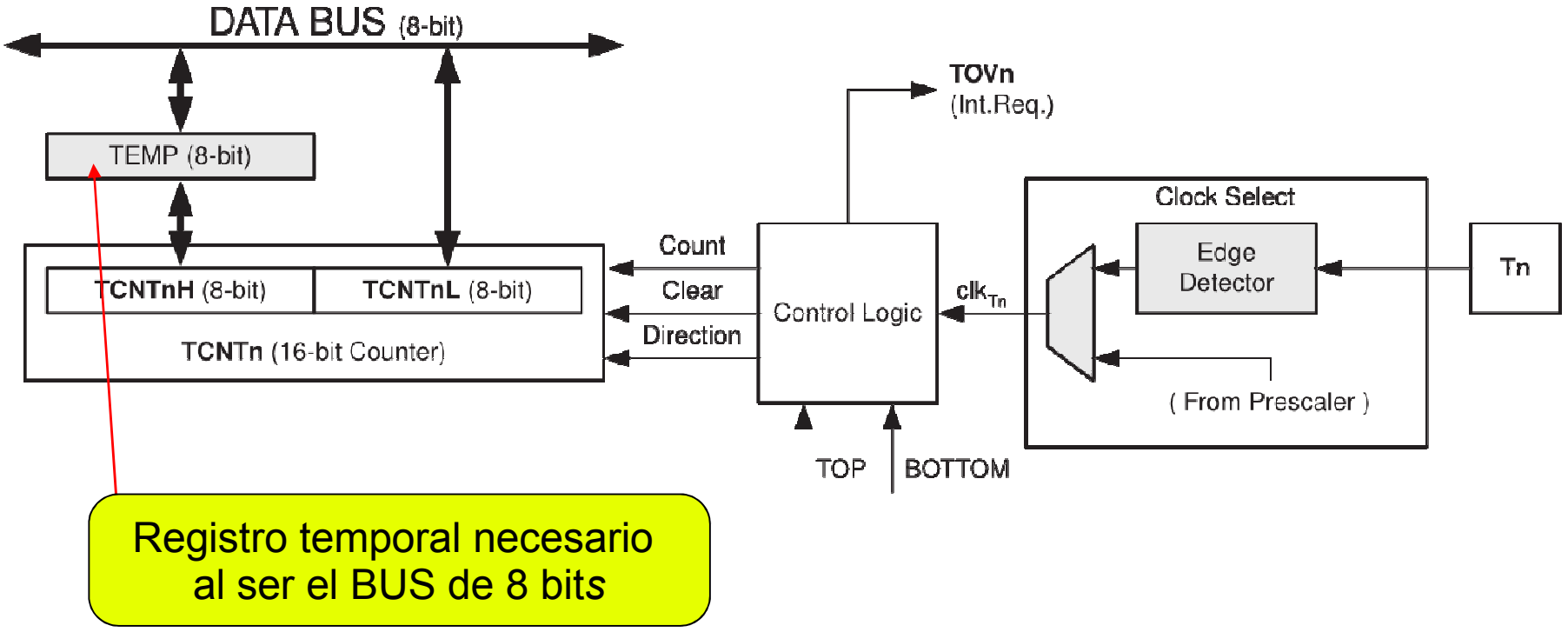
Estudiaremos el Timer 1 (16bits)

Funcionalidad:

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

Temporizadores

Esquema general del temporizador 1 (16 bits)



Temporizadores

De los modos de operación posibles estudiaremos:

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

Puesta a cero al llegar a un valor: El contador se pone a cero automáticamente cuando se alcanza el valor establecido en OCR1A

Registros involucrados

Configuración: TCCR1A y TCCR1B

Estado de cuenta: TCNT1H y TCNT1L

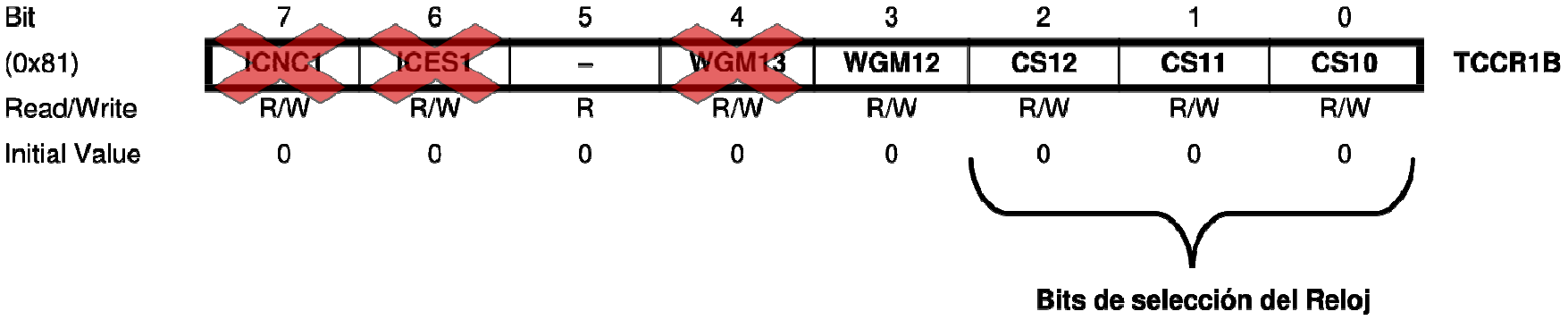
Comparadores: OCR1AH ,OCR1AL, OCR1BH y OCR1BL

Temporizadores

Registro de configuración:

Solo estudiaremos 4 bits el TCCR1B

El resto se pueden quedar sin inicializar, por defecto todos están a cero



Temporizadores

Frecuencia de funcionamiento del temporizador en función de 3 bits (CS12,CS11,CS10)

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

Modos de funcionamiento:

Modo Normal:

El bit WGM12 debe configurarse a '0' (registro TCCR1B).

El contador cuenta desde 0x0000 a 0xFFFF y vuelta a 0x0000, a la frecuencia de reloj configurada.

En cada paso por 0x0000 se activa un bit llamado TOV1

Modo CTC (Clear Timer on Compare Match):

El bit WGM12 debe configurarse a '1' (registro TCCR1B)

El contador cuenta desde 0x0000 hasta que su contenido es igual al almacenado en OCR1A (16 bits). Tras esto se pone automáticamente a 0x0000.

Cuando esto ocurre se activa un bit llamado OCF1

Temporizadores

Registros 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	

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	

Temporizadores

Interrupciones:

Si TOIE1=1 (registro TIMSK1) entonces se produce una interrupción cada vez que el temporizador pasa por 0x0000 y se activa la bandera de interrupción TOV1 (registro TIFR1)

Si OCIE1A=1 (registro TIMSK1) entonces se produce una interrupción cada vez que el temporizador alcanza el valor almacenado en OCR1A se pone a 0x0000 y se activa la bandera de interrupción OCF1A (registro TIFR1)

Si OCIE1B=1 (registro TIMSK1) entonces se produce una interrupción cada vez que el temporizador alcanza el valor almacenado en OCR1B se pone a 0x0000 y se activa la bandera de interrupción OCF1B (registro TIFR1)

Temporizadores

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

1º Parte alta del registro. Por ejemplo OCR1AH

2º Parte baja del registro. Por ejemplo OCR1AL

```
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
```

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 Activa el auto-clear cuando el contenido el temporizador sea igual a OCR1A.

Cada vez que se cicle el contador ha 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
```

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.