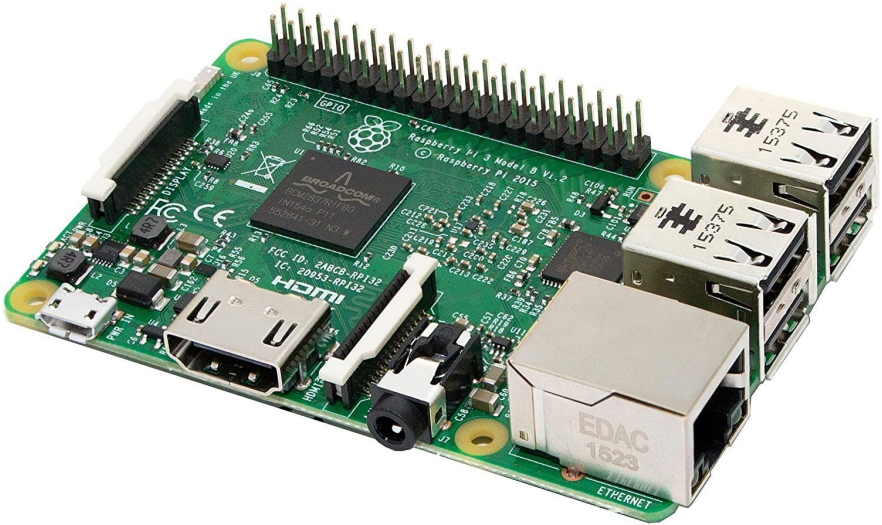
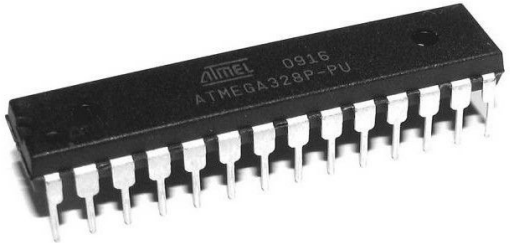
Microcontroladores

AVR

Microprocesadores vs Microcontroladores





- Arquitectura compleja.
- Usada para procesamiento de Datos y ejecución programas de propósito generales
- Tiene sistema operativo. Programación de alto nivel.

- Es como una pequeña computadora en un solo circuito integrado. Contiene CPU, memoria y lo periféricos en la mismo chip.
- Sirve para realizar una tarea de control específica o adquirir datos. Sistemas embebidos.
- No tiene sistema operativo.

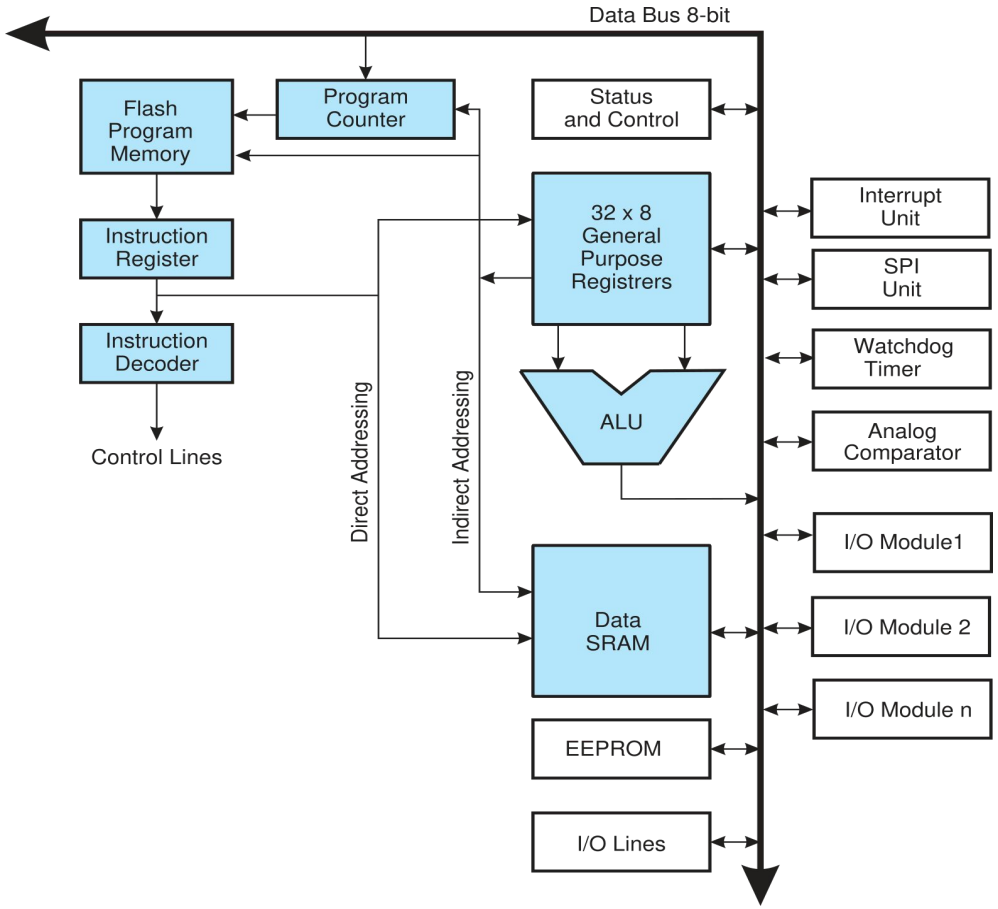
Microcontroladores

Atmega328p:

- Voltaje de operación: 1.8 a 5.5 VDC.
- Arquitectura de CPU: 8 bit AVR.
- Memoria flash: 32 KB.
- Memoria RAM: 2 KB.
- EEPROM: 2 KB.
- Frecuencia de operación: hasta 20 Mhz.

Arquitectura de AVR

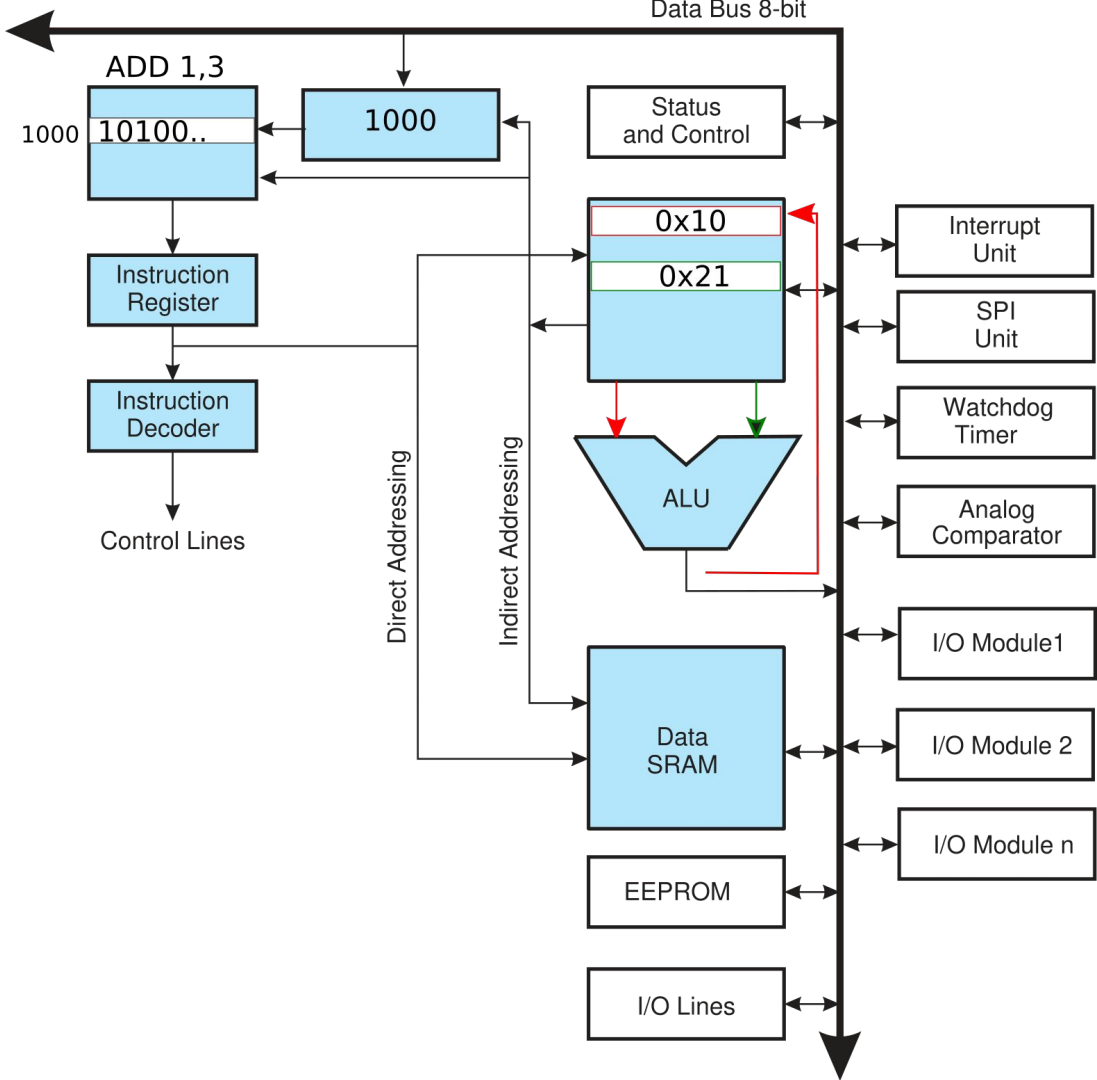
- Arquitectura RISC y harvard
- El bus de datos se comparte con los registros de configuración de los periféricos
- 32 registros de trabajo



Instrucciones aritmética y lógicas

- Suma y resta
- Multiplicación
- Or, And, Xor
- Incremento, decremento

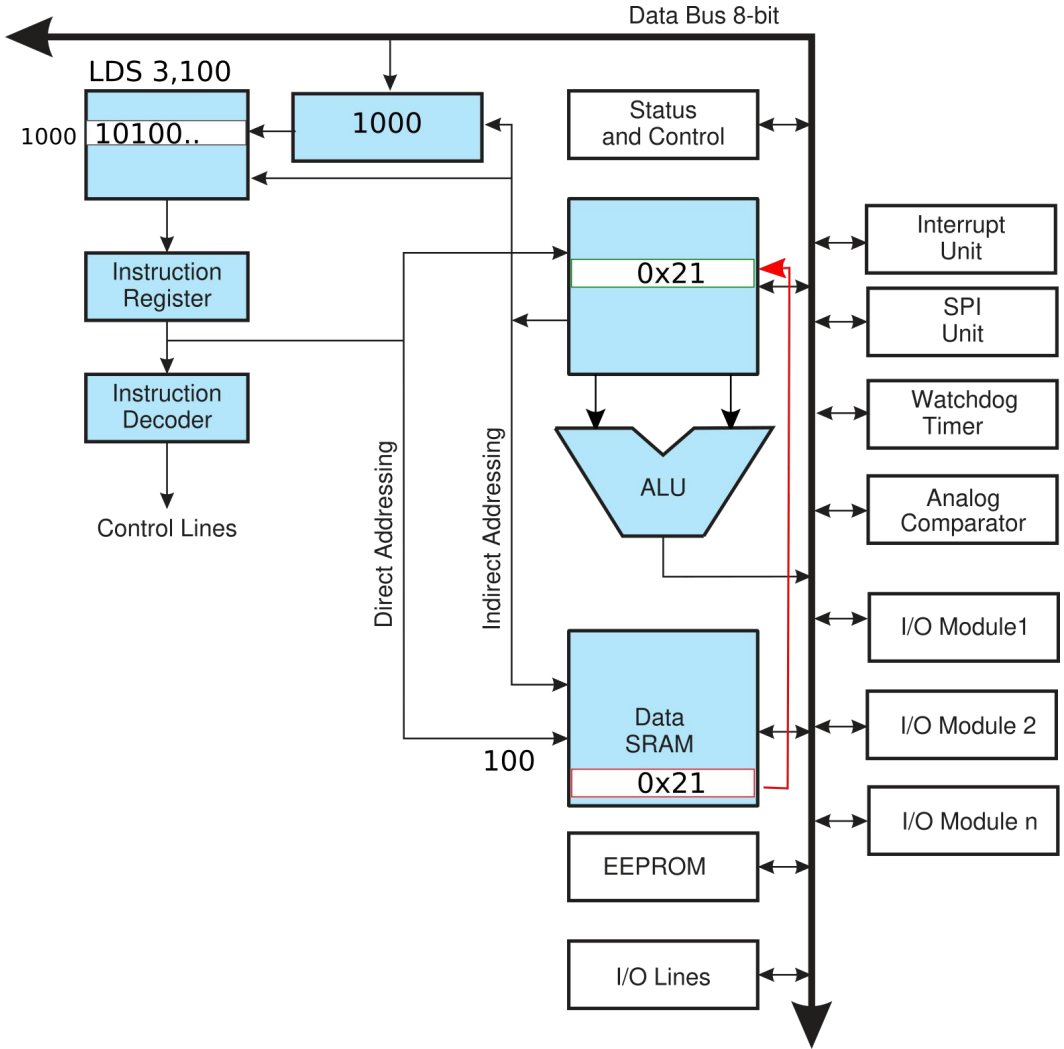
Los operadores pueden ser los registros o ctes



Instrucciones de transferencia de datos

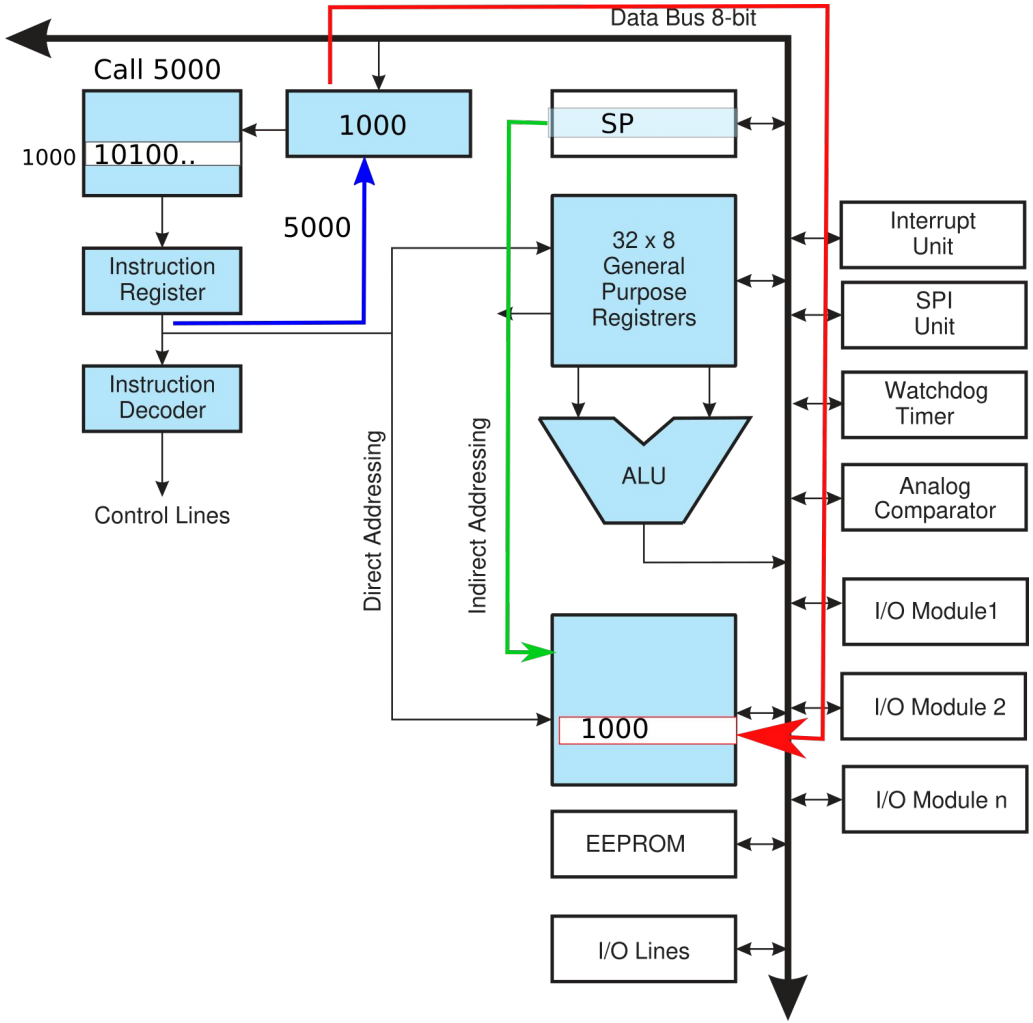
- Mover registro
- Cargar y guardar
- Push, pop LIFO

La memoria se accede de forma directa y en forma indirecta

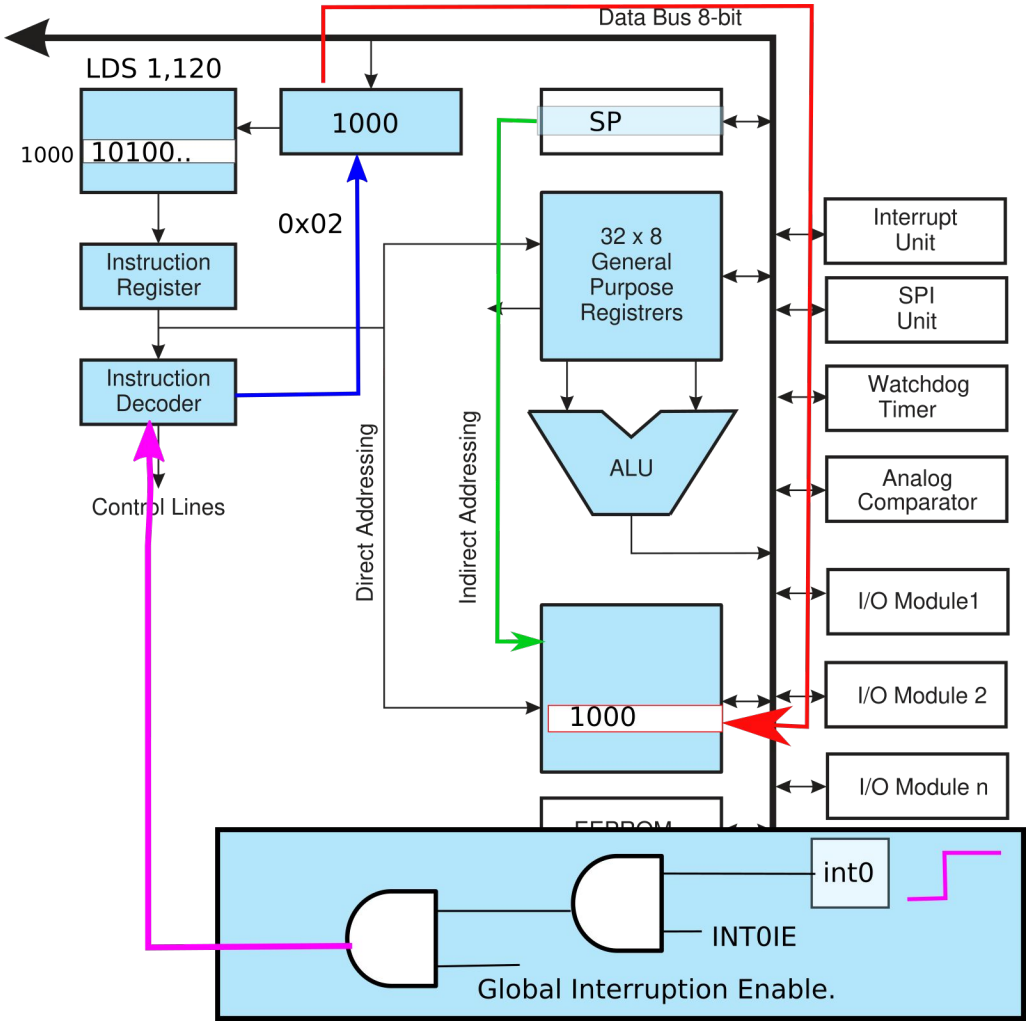


Instrucciones de Flujo de Programa

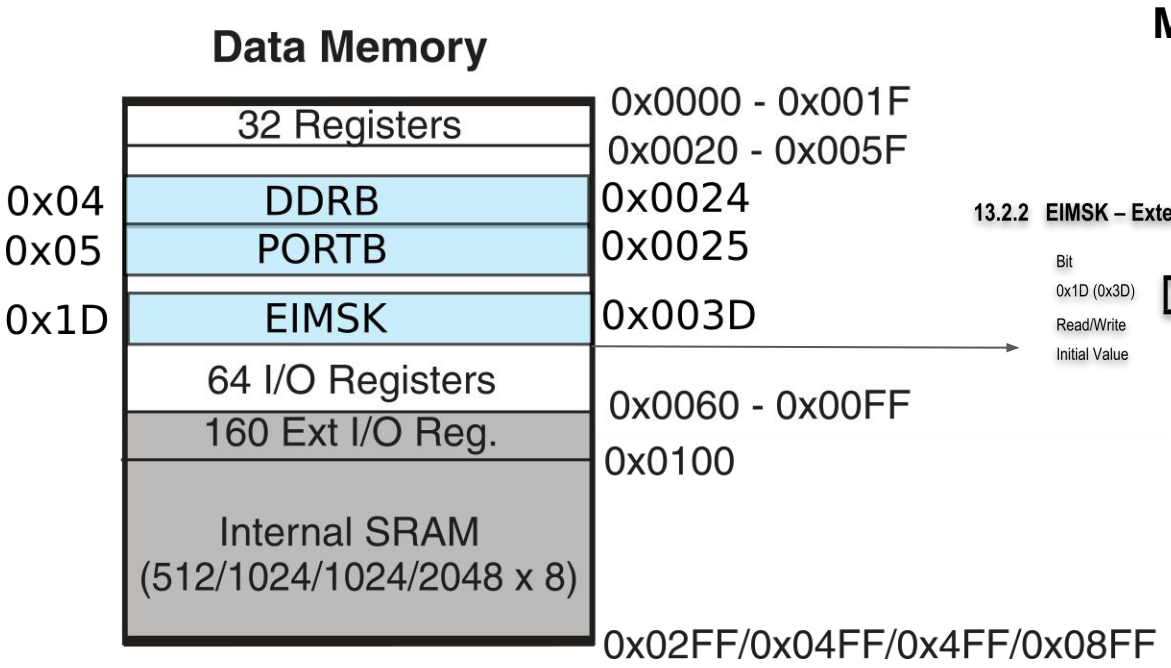
- Saltos, relativo, directo, indirectos
- Llamados a rutina
- Saltos condicionales



Interrupción



Registros en memoria RAM

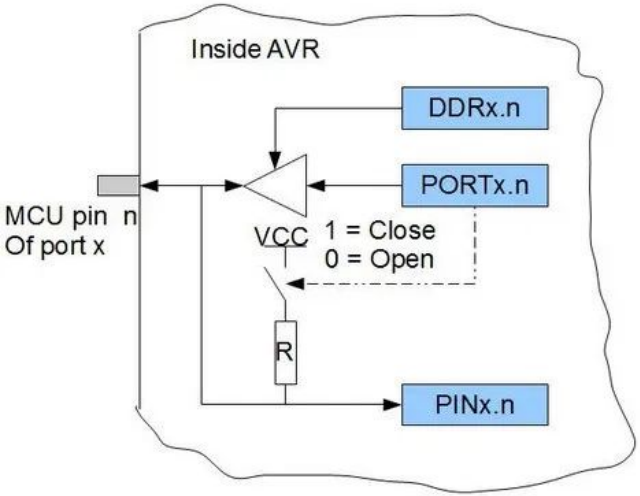


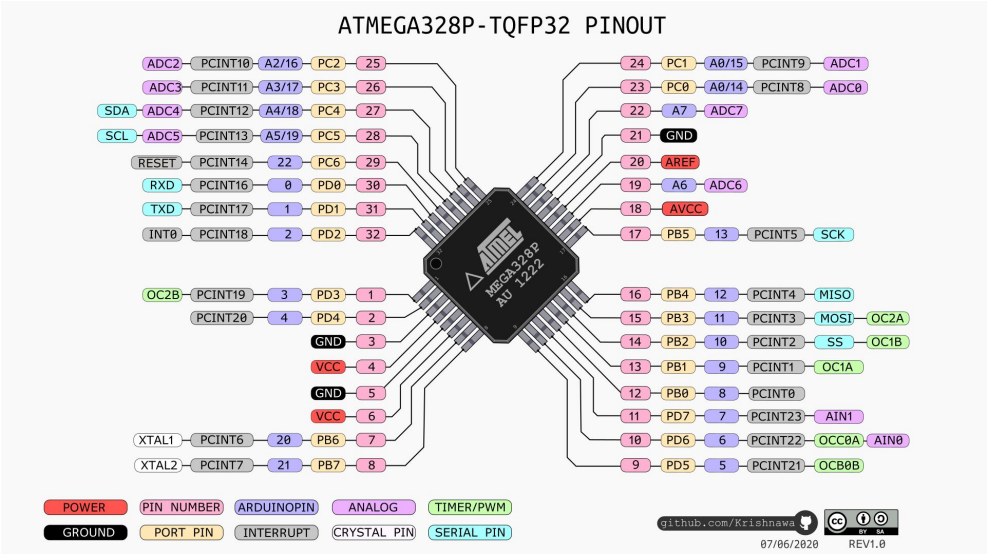
Manual

13.2.2 EIMSK – External Interrupt Mask Register

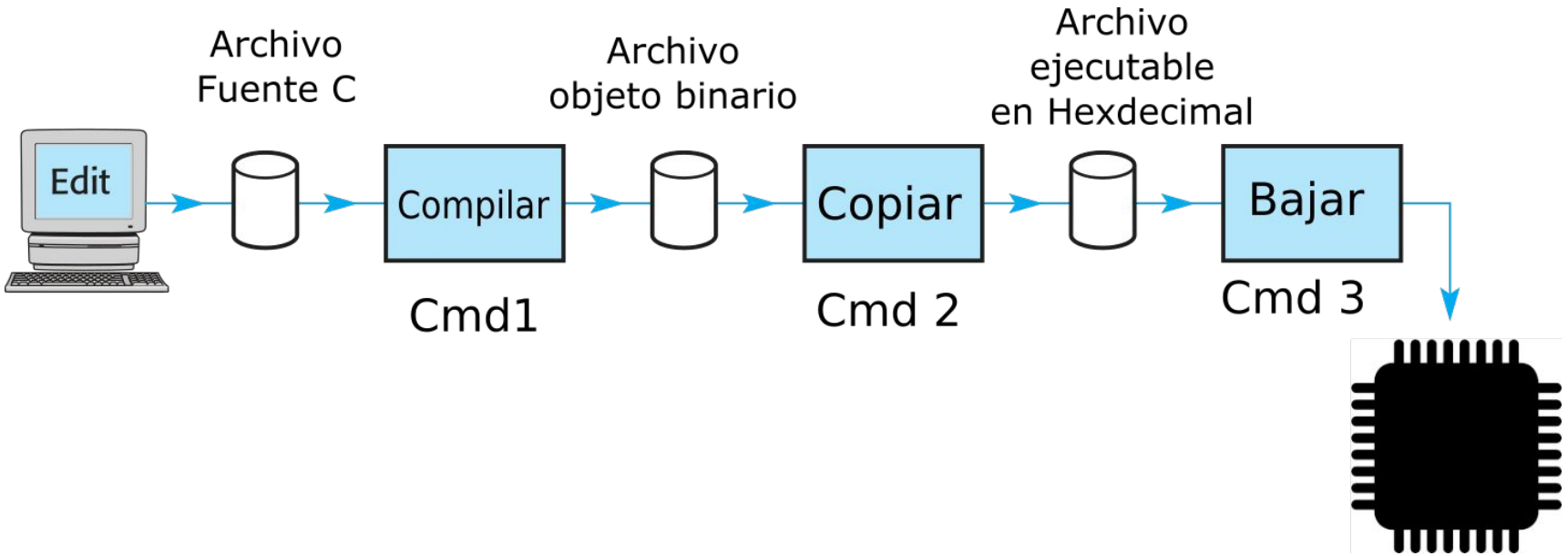
Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Periféricos : Puerto digitales





Programación




```
1.  $ avr-gcc ejemplo.c -o ejemplo.elf -mmcu=atmega328p -Os -Wall -DF_CPU=16000000
2.  $ avr-objcopy -j .text -j .data -O ihex ejemplo.elf ejemplo.hex
3.  $ avrdude -P /dev/ttyUSB0 -c arduino -p m328p -U flash:w:ejemplo.hex -b57600
```

Programación

```
#include <avr/io.h> // Contiene las definiciones de
los registros
#include <util/delay.h> // Contiene la funcion
delay
int main()
{
DDRB = 1<<PORTB5; // Salida el puerto 5

while(1)
{
PORTB ^= 1<<PORTB5; // invierto solo el bit de
puerto 5
_delay_ms(200); // espera 200 milisegundo

}
return 0;
}
```

```
  80:    80 e2          ldi    r24, 0x20    ; 32
  82:    84 b9                 out    0x04, r24
; 4
  84:    90 e2                 ldi    r25, 0x20
; 32
  86:    85 b1                 in    r24, 0x05
; 5
  88:    89 27                 eor    r24, r25
  8a:    85 b9                 out    0x05, r24
; 5
  8c:    2f ef                 ldi    r18, 0xFF
; 255
  8e:    33 ec                 ldi    r19, 0xC3
; 195
  90:    89 e0                 ldi    r24, 0x09
; 9
  92:    21 50                 subi    r18, 0x01
; 1
  94:    30 40                 sbci    r19, 0x00
; 0
  96:    80 40                 sbci    r24, 0x00
; 0
  98:    e1 f7                 brne    .-8
     ; 0x92 <main+0x12>
  9a:    00 c0                 rjmp    .+0
     ; 0x9c <main+0x1c>
```

Programación de Interrupciones

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main()
{
   EIMSK = (1 << INT0);
   DDRB = 1 << PORTB5;
   sei();
   while (1)
       ;
   return 0;
}

ISR(INT0_vect)
{
   PORTB ^= 1 << PORTB5;
   // Rutina de servicio para la interrupción externa en pin 0
}
```

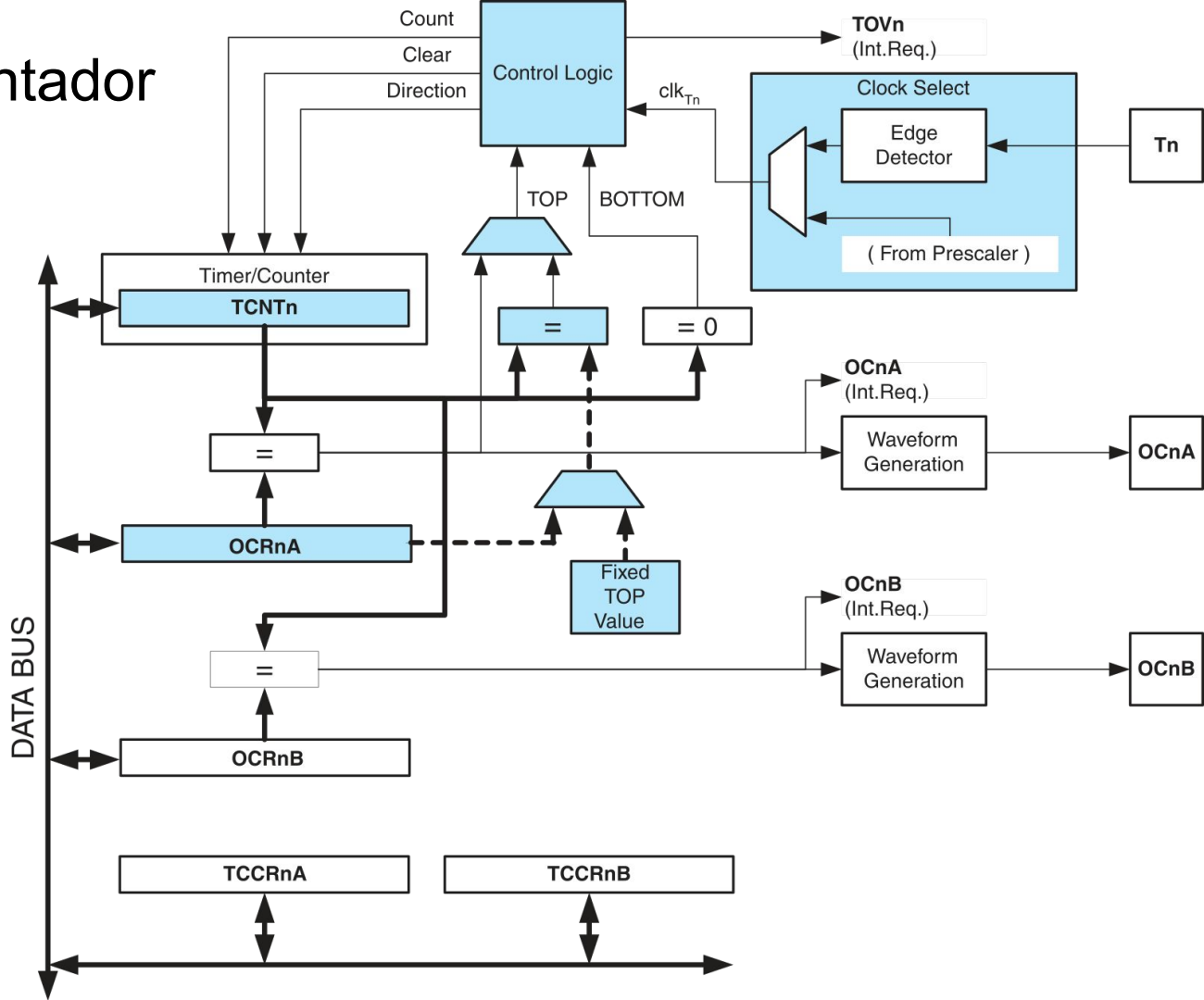
Table 12-6. Reset and Interrupt Vectors in ATmega328 and ATmega328P

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-
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

```
  // Representación numéricas
  a = 10; //Decimal
  a = 0x0A; // Hexadecimal
  a = 0b0001010; //Binaria


  // Operadores bit a bit
  c = a & b; // And bit a bit
  c = a | b; // Or bit a bit
  c = a ^ b; // O-exclusiva bit a bit
  c = ~a; // Negación bit a bit


 // Desplazamientos
 c = a >> b; //Desplazamiento de "b" bits a la derecha
 c = a << b; //Desplazamiento de "b" bits a la izquierda

// Asignaciones = *= /= += %= -= >>= <<= &= |= ^=
a +=b; // Suma "a" y "b" se asigna a "a"
a |=b; // OR bit a bit de "a" y "b" el resultado se asigna a "a"
a &=b; // AND bit a bit de "a" y "b" el resultado se asigna a "a"
```

Temporizador / Contador

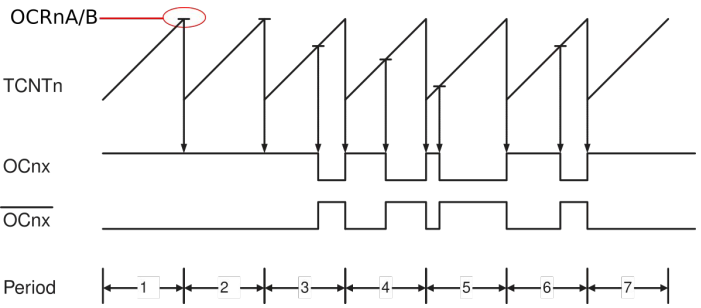
Mode	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag
0	Normal	0xFF	Immediate	MAX
1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	CTC	OCRA	Immediate	MAX
3	Fast PWM	0xFF	BOTTOM	MAX
4	Reserved	–	–	–
5	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	Reserved	–	–	–
7	Fast PWM	OCRA	BOTTOM	TOP

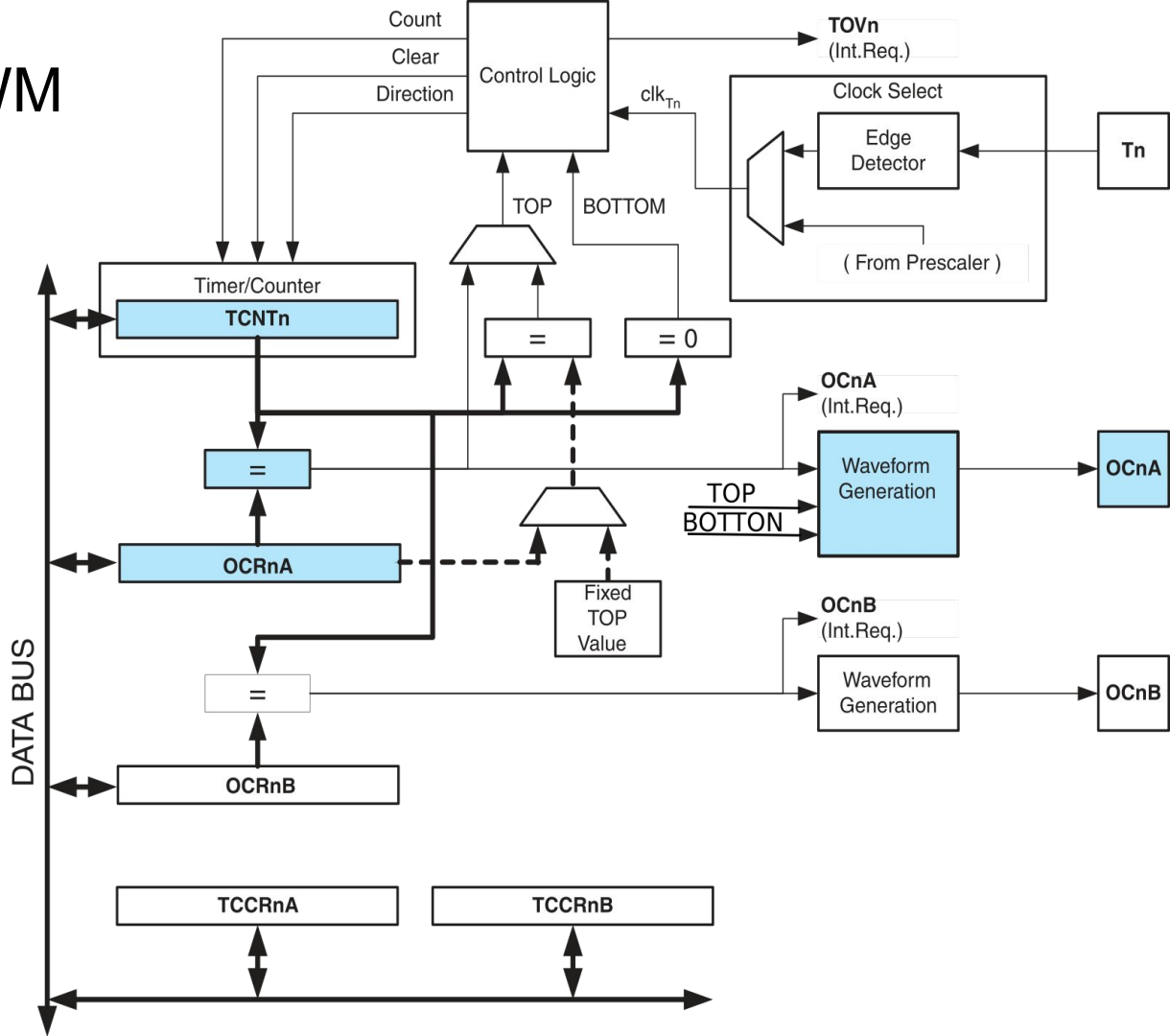


Temporizador con PWM

Modo:

- Desconectado
- Toggle
- Clear, set in Bottom
- Set, clear in Bottom

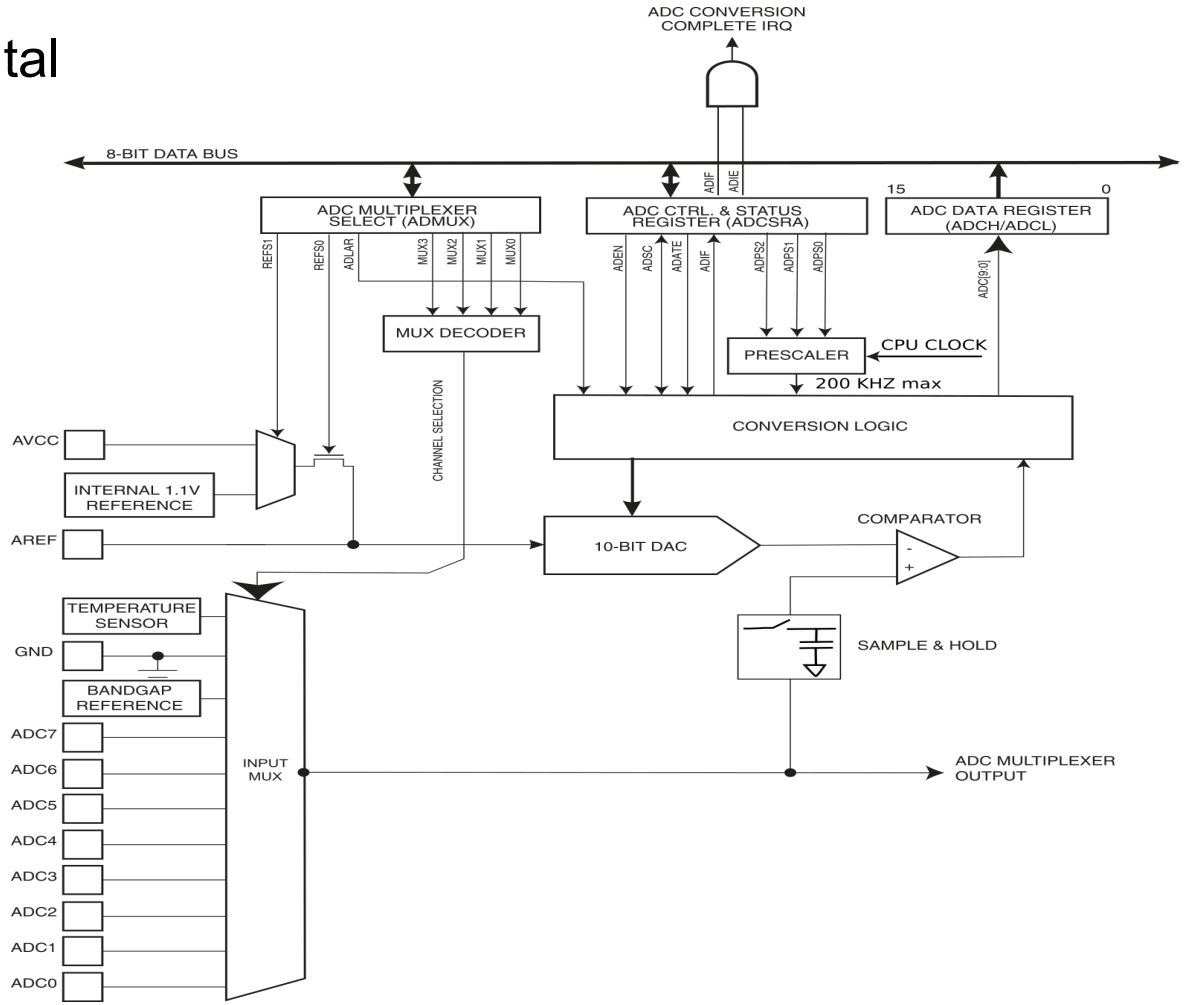




Conversor analógico digital

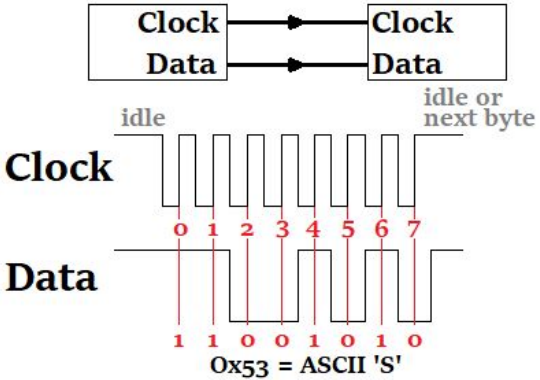
Control de Conversión

- En forma manual (Seteando ADSC)
- Automático, por disparo de timer, finalizado de conversión, interrupción externa, comparador

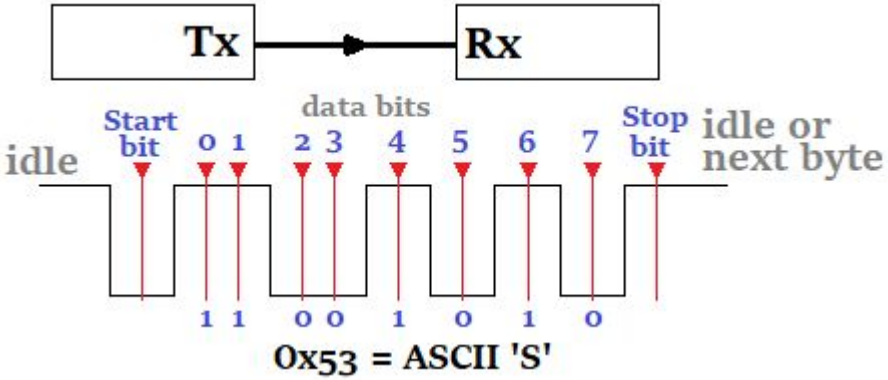


Comunicación Serie

Sincrónica



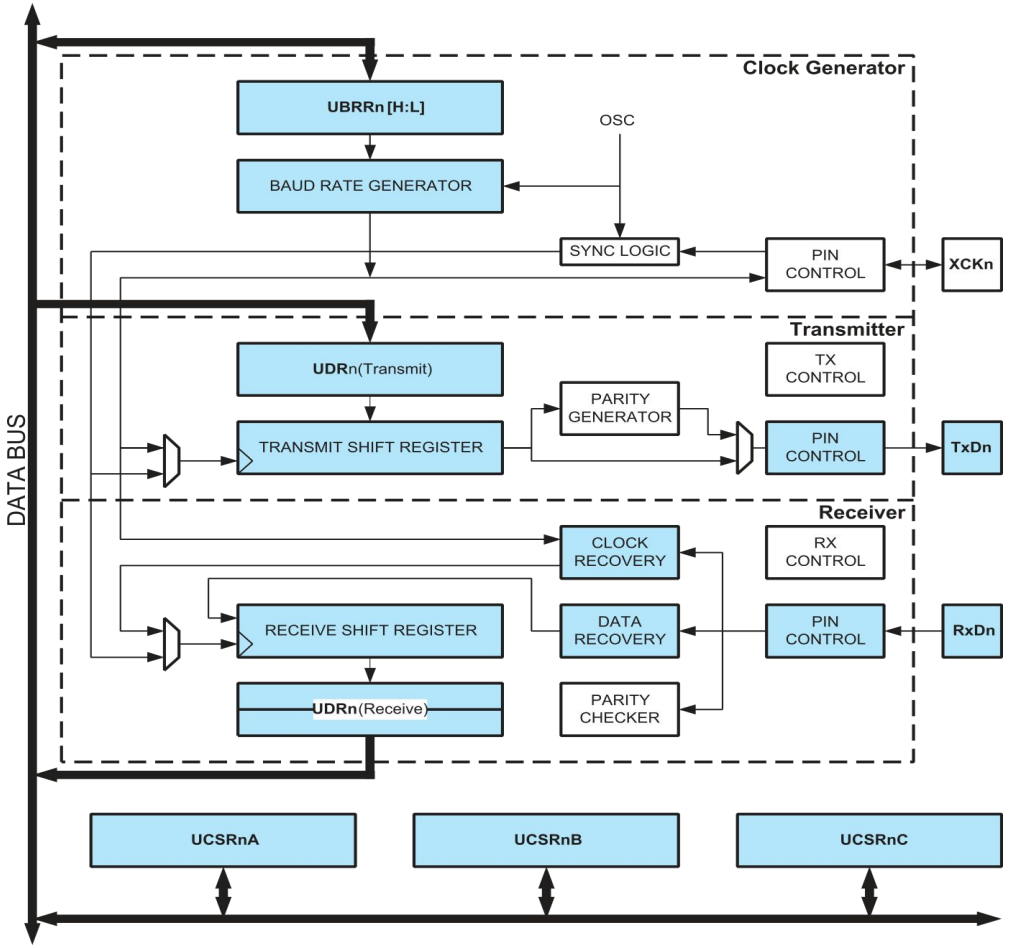
Asincrónica



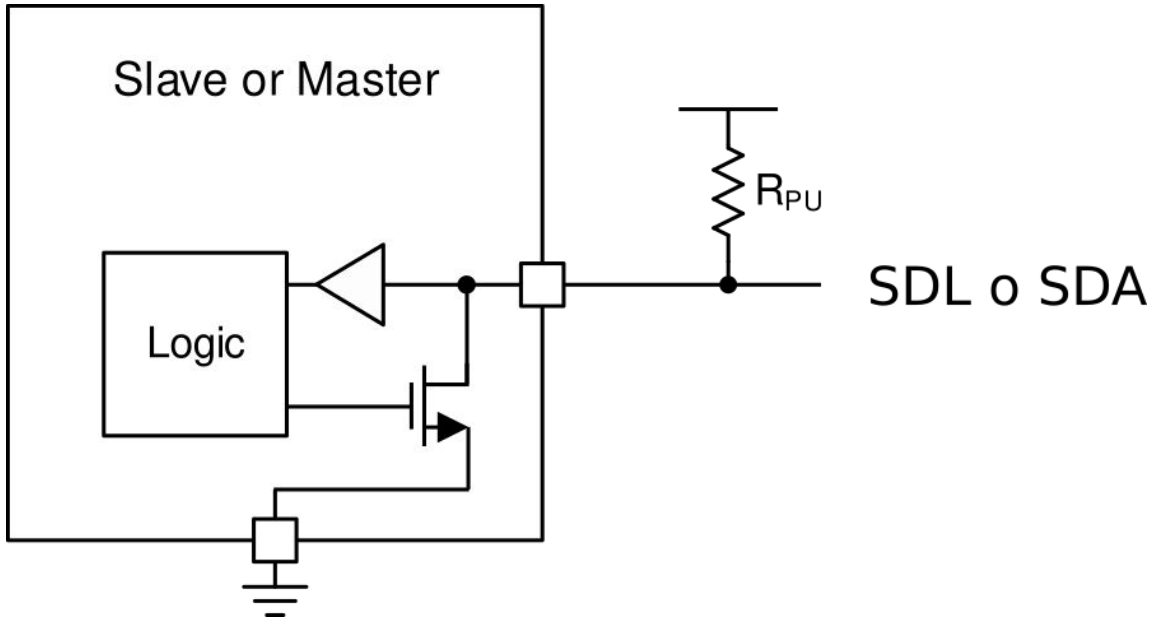
USART

Interrupciones:

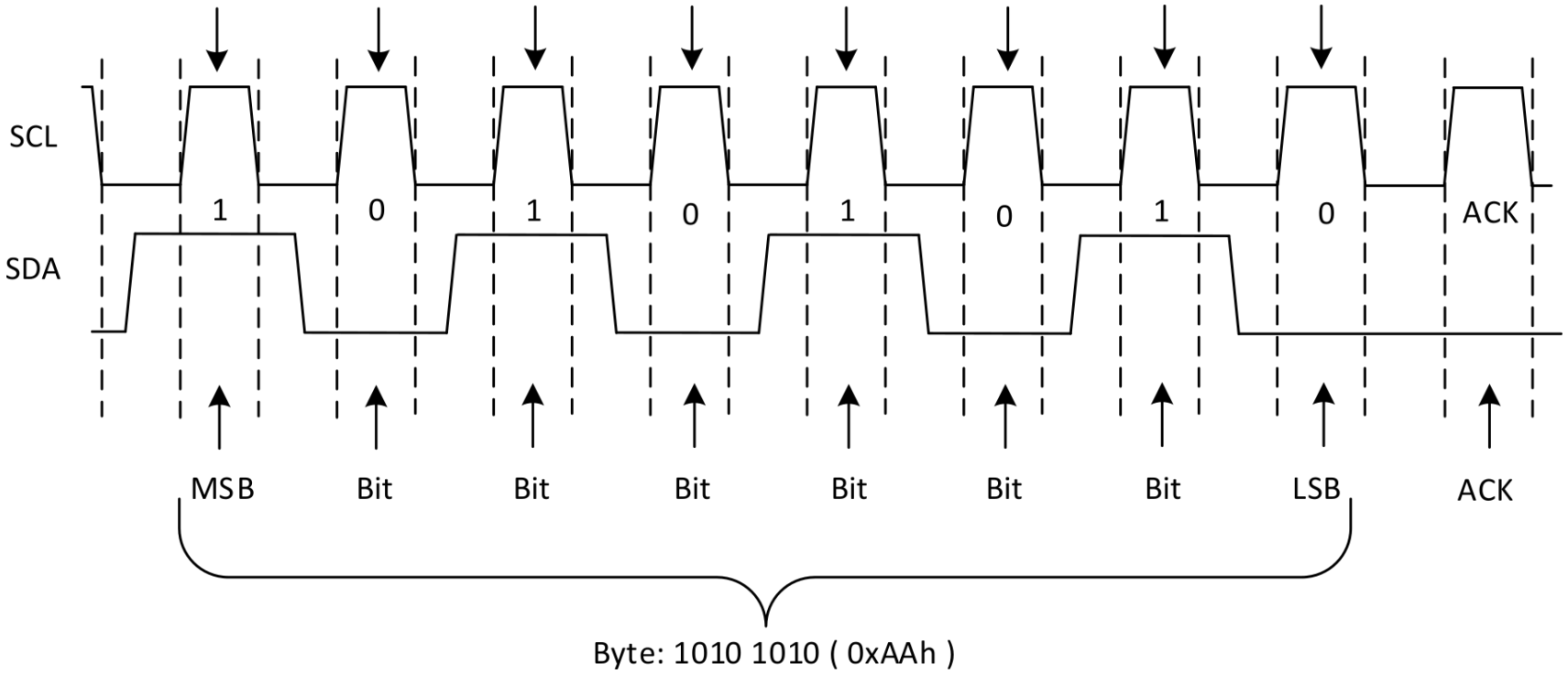
- RXC0; Interrumpe cuando hay dato en UDR0
- TXC0: Cuando terminó de transmitir un UDR0
- UDRE0: Está activo cuando no hay dato que transmitir



Comunicación i2c

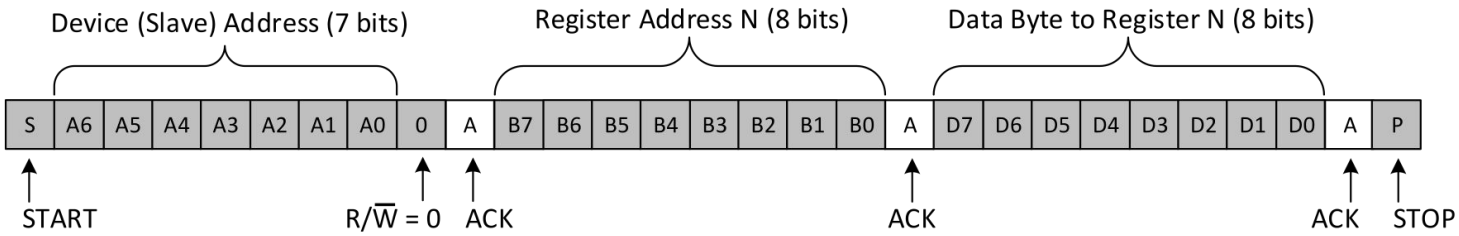


Comunicación I2C - Acknowledge

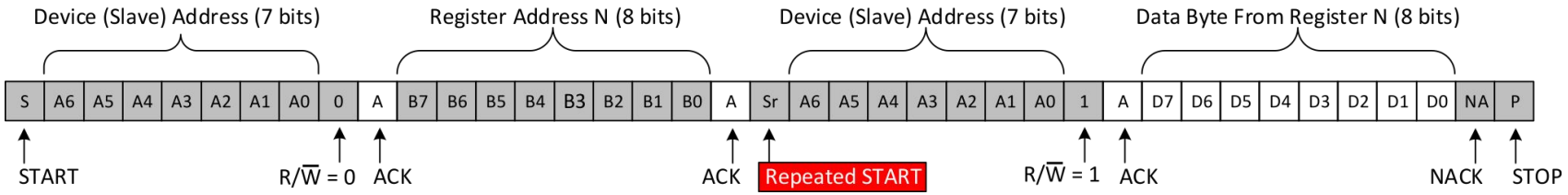


Comunicación i2c

Escritura de un registro esclavo



Lectura de un registro esclavo



En TWCR se tiene los siguiente flag:

- TWINT: Inicializa la operación de comunicación
- TWSTA: Envía el bit de start.
- TWSTO: Envía el bit de stop.
- TWEA: Activa el bit de reconocimiento

En TWSR tenes lo resultados de la operación