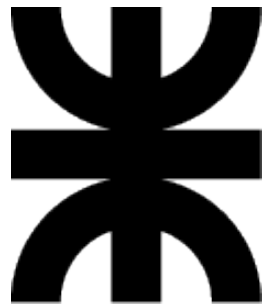
Informática II
Programación del microcontrolador ATmega328
Entradas/salidas digitales

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

Puerto digital de entrada/salida

Para el manejo de los puertos digitales de entrada/salida se utilizan 3 registros: `DDRx`, `PORTx` y `PINx` (`x`: B, C o D).

Puerto digital de entrada/salida

Para el manejo de los puertos digitales de entrada/salida se utilizan 3 registros: `DDRx`, `PORTx` y `PINx` (`x`: B, C o D).

Registros

- `DDRx`: registro de dirección (Data Direction Register) del puerto `x`
- `PORTx`: registro de datos (Data Register) del puerto `x`
- `PINx`: registro de lectura (Input Pin Register) del puerto `x`

Puerto digital de entrada/salida

Para el manejo de los puertos digitales de entrada/salida se utilizan 3 registros: `DDRx`, `PORTx` y `PINx` (`x`: B, C o D).

Registros

- `DDRx`: registro de dirección (Data Direction Register) del puerto `x`
- `PORTx`: registro de datos (Data Register) del puerto `x`
- `PINx`: registro de lectura (Input Pin Register) del puerto `x`

Los registros de manejo de los puertos digitales de entrada/salida se encuentran mapeados en la memoria RAM de μC, o sea que tienen direcciones de memorias fijas.

Puerto digital de entrada/salida

Registros para el manejo de puertos digitales de E/S

Bit	7	6	5	4	3	2	1	0	
	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0	**DDRx**
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	
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0	**PORTx**
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	
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0	**PINx**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Puerto digital de entrada/salida

Registros para el manejo de puertos digitales de E/S

Bit	7	6	5	4	3	2	1	0	
	DDx7	DDx6	DDx5	DDx4	DDx3	DDx2	DDx1	DDx0	**DDRx**
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	
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0	**PORTx**
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	
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0	**PINx**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Los bits `DDxn` del registro `DDRx` seleccionan si el pin correspondiente actuará como entrada o salida:

- Si se escribe un 1 lógico a `DDxn` el pin `Pxn` se configura como salida.
- Si se escribe un 0 lógico a `DDxn`, el pin `Pxn` se configura como entrada.

Puerto digital de entrada/salida

	PINx	DDRx	PORTx
Puerto B	0x23	0x24	0x25
Puerto C	0x26	0x27	0x28
Puerto D	0x29	0x2A	0x2B

Dirección de los registros de manejo de puertos digitales de E/S

Programación de los puertos digitales – salidas (1)

- El código fuente `blink.c` es un ejemplo de manejo de un puerto digital de entrada/salida.
- Se utiliza el bit 5 del puerto B (`PB5`) configurado como salida para encender y apagar un LED conectado al pin correspondiente.
- La placa Arduino UNO incluye un LED conectado a dicho pin

Programación de los puertos digitales – salidas (1)

- El código fuente `blink.c` es un ejemplo de manejo de un puerto digital de entrada/salida.
- Se utiliza el bit 5 del puerto B (`PB5`) configurado como salida para encender y apagar un LED conectado al pin correspondiente.
- La placa Arduino UNO incluye un LED conectado a dicho pin

Configuración del pin `PB5` como salida:

```
DDRB |= _BV(DDB5); /* Equiv. a DDRB = DDRB | _BV(DDB5); */
```

Programación de los puertos digitales – salidas (1)

- El código fuente `blink.c` es un ejemplo de manejo de un puerto digital de entrada/salida.
- Se utiliza el bit 5 del puerto B (`PB5`) configurado como salida para encender y apagar un LED conectado al pin correspondiente.
- La placa Arduino UNO incluye un LED conectado a dicho pin

Configuración del pin `PB5` como salida:

```
DDRB |= _BV(DDB5); /* Equiv. a DDRB = DDRB | _BV(DDB5); */
```

que usa la macro `_BV()` definida como:

```
#define _BV(bit) (1 << (bit))
```

y la constante simbólica `DDB5` que vale 5.

Programación de los puertos digitales – salidas (1)

- El código fuente `blink.c` es un ejemplo de manejo de un puerto digital de entrada/salida.
- Se utiliza el bit 5 del puerto B (`PB5`) configurado como salida para encender y apagar un LED conectado al pin correspondiente.
- La placa Arduino UNO incluye un LED conectado a dicho pin

Configuración del pin `PB5` como salida:

```
DDRB |= _BV(DDB5); /* Equiv. a DDRB = DDRB | _BV(DDB5); */
```

que usa la macro `_BV()` definida como:

```
#define _BV(bit) (1 << (bit))
```

y la constante simbólica `DDB5` que vale 5. Por lo que la operación queda:

```
DDRB = DDRB | (1 << (5)); /* DDRB = DDRB | 0x40 */
```

Programación de los puertos digitales – salidas (1)

- El código fuente `blink.c` es un ejemplo de manejo de un puerto digital de entrada/salida.
- Se utiliza el bit 5 del puerto B (`PB5`) configurado como salida para encender y apagar un LED conectado al pin correspondiente.
- La placa Arduino UNO incluye un LED conectado a dicho pin

Configuración del pin `PB5` como salida:

```
DDRB |= _BV(DDB5); /* Equiv. a DDRB = DDRB | _BV(DDB5); */
```

que usa la macro `_BV()` definida como:

```
#define _BV(bit) (1 << (bit))
```

y la constante simbólica `DDB5` que vale 5. Por lo que la operación queda:

```
DDRB = DDRB | (1 << (5)); /* DDRB = DDRB | 0x40 */
```

que pone a 1 el bit 5 del registro `DDRB`.

Programación de los puertos digitales – salidas (1)

Luego en el bucle principal se enciende y apaga el LED cada 1000ms, poniendo a 1 y a 0 respectivamente el bit 5 del registro de datos del puerto B (`PORTB5`).

Programación de los puertos digitales – salidas (1)

Luego en el bucle principal se enciende y apaga el LED cada 1000ms, poniendo a 1 y a 0 respectivamente el bit 5 del registro de datos del puerto B (`PORTB5`).

El bit se pone a 1 con la línea:

```
PORTB |= _BV(PORTB5);
```

Programación de los puertos digitales – salidas (1)

Luego en el bucle principal se enciende y apaga el LED cada 1000ms, poniendo a 1 y a 0 respectivamente el bit 5 del registro de datos del puerto B (`PORTB5`).

El bit se pone a 1 con la línea:

```
PORTB |= _BV(PORTB5);
```

y se pone a 0 con la línea:

```
PORTB = PORTB & ~(1 << (5)); /* PORTB = PORTB & 0xBF; */
```

utilizando el operador AND y NOT a nivel de bit. La constante simbólica `PORTB5` vale 5.

Programación de los puertos digitales – salidas (1)

Luego en el bucle principal se enciende y apaga el LED cada 1000ms, poniendo a 1 y a 0 respectivamente el bit 5 del registro de datos del puerto B (`PORTB5`).

El bit se pone a 1 con la línea:

```
PORTB |= _BV(PORTB5);
```

y se pone a 0 con la línea:

```
PORTB = PORTB & ~(1 << (5)); /* PORTB = PORTB & 0xBF; */
```

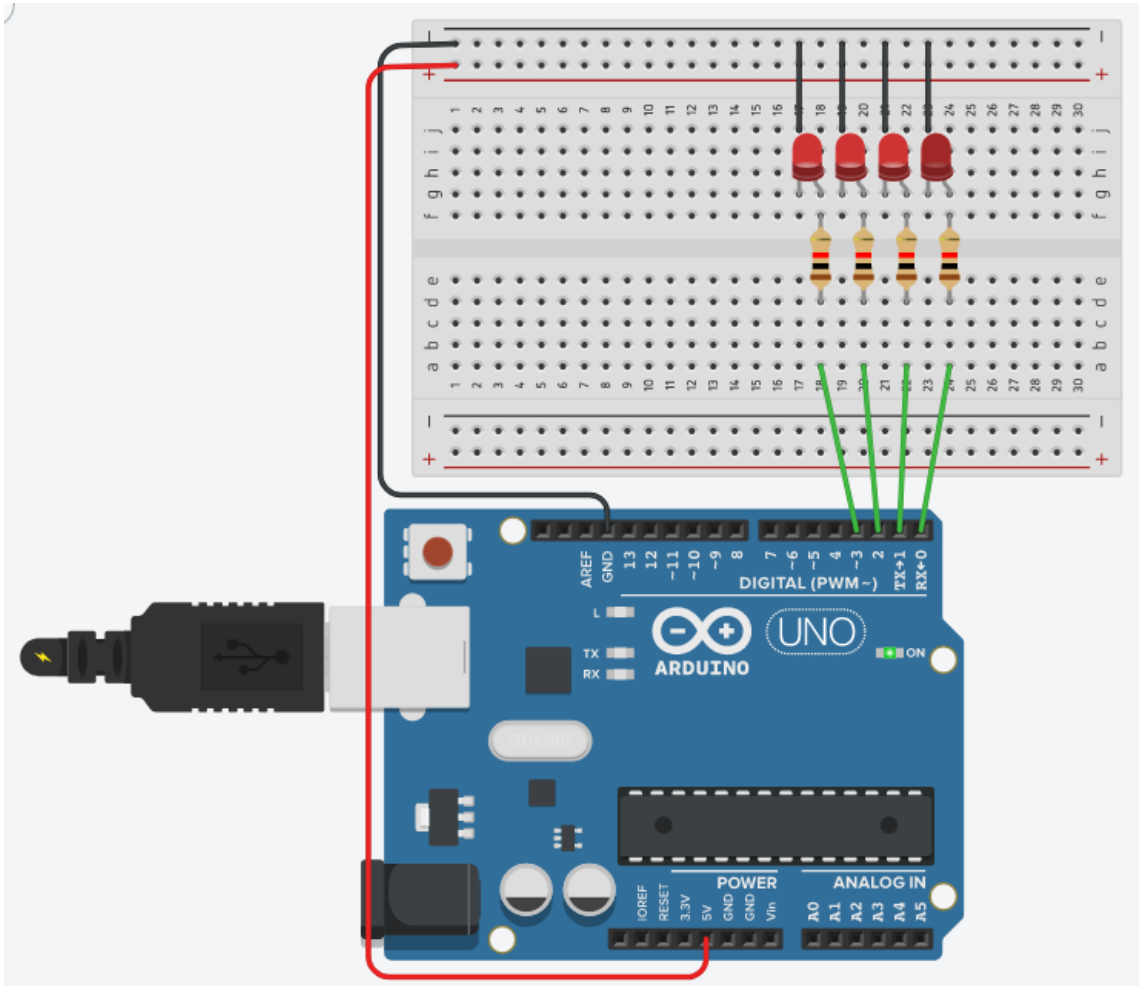
utilizando el operador AND y NOT a nivel de bit. La constante simbólica `PORTB5` vale 5.

Otra forma de hacer lo mismo es usando el operador XOR, reemplazando el bucle principal por:

```
while(1)
{
  PORTB ^= _BV(PORTB5); /* Toggle LED */
  _delay_ms(BLINK_DELAY_MS);
}
```

Programación de los puertos digitales – salidas (2)

Circuito contador de 4 bits



Utiliza los 4 bits menos significativos del pueto D (`PD0` a `PD3`).

Programación de los puertos digitales – salidas (2)

Archivo 'led_hex_counter.c'

```
1  #include <avr/io.h>
2  #include <util/delay.h>
3
4  #define LED_DDR DDRD
5  #define LED_PORT PORTD
6  #define LED_MASK 0x0F
7
8  #define DELAY_MS 500
9
10 int main (void)
11 {
12   unsigned char hex = 0;
13
14   /* Inicializa salidas digitales */
15   LED_DDR |= LED_MASK;
16
17   while(1)
18   {
19     LED_PORT &= ~(LED_MASK); // Apaga los LEDs
20     LED_PORT |= hex; // Muestra valor
21
```

Programación de los puertos digitales – salidas (2)

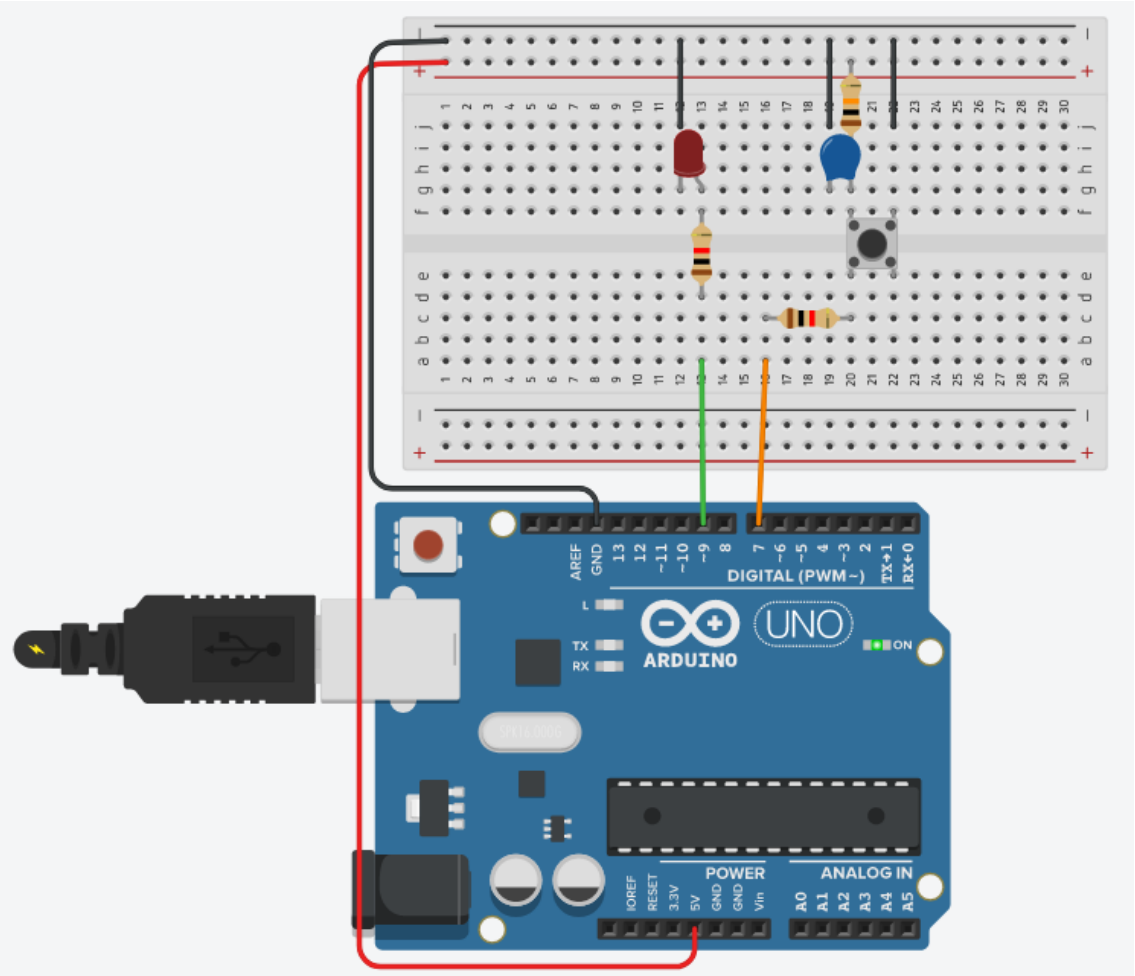
Archivo 'led_hex_counter.c' (cont.)

```
22      if(++hex > 16) // Controla rango máximo
23        hex = 0;
24
25      _delay_ms(DELAY_MS);
26    }
27  return 0;
28  }
```

Los LEDs muestran una cuenta binario de 4 bits, o sea del rango de valores que va desde 0d = 0000b = 0x00 hasta 15d = 1111b = 0x0F.

Programación de los puertos digitales – entradas

Circuito con pulsador y LED



El pulsador se conecta al pin 7 que es el bit 7 del puerto D (`PD7`) el cual debe configurarse como entrada y el LED se conecta al pin 9 que es el bit 1 del puerto B (`PB1`) que debe configurarse como salida.

Programación de los puertos digitales – entradas

Archivo 'led_sw.c'

```
1  #include <avr/io.h>
2  #include <util/delay.h>
3
4  /*
5   * LED: pin 9 --> PB1 (Salida)
6   * Sw: pin 7 --> PD7 (Entrada)
7   */
8  #define SW_MASK _BV(PIND7)
9  #define SW_PRES 0
10
11 int main(void)
12 {
13   uint8_t sw;
14
15   /* Configuración de entrada/salida digitales */
16   DDRB |= _BV(DDB1); /* LED como salida */
17   DDRD &= ~_BV(DDD7); /* Sw como entrada */
18   PORTB &= ~_BV(PORTB1); /* Apaga el LED */
19
```

Programación de los puertos digitales – entradas

Archivo 'led_sw.c' (cont.)

```
20   while(1)
21   {
22     /* Lee el valor del pulsador */
23     sw = (PIND & SW_MASK) >> PIND7;
24
25     if(sw == SW_PRES)
26     {
27       PORTB |= _BV(PORTB1); /* Enciende LED */
28       _delay_ms(1000);
29       PORTB &= ~_BV(PORTB1); /* Apaga LED */
30     }
31     _delay_ms(1);
32   }
33   return 0;
34 }
```

El programa lee el valor de la entrada digital conectada al pulsador y, en caso de estar presionado, encender el LED durante 1 seg. y luego apagarlo.

Programación de los puertos digitales – entradas

La constante simbólica `SW_MASK` se utiliza como máscara para la lectura de la entrada digital (línea 23).

Programación de los puertos digitales – entradas

La constante simbólica `SW_MASK` se utiliza como máscara para la lectura de la entrada digital (línea 23).

La línea que realiza la lectura de la entrada es:

```
sw = (PIND & SW_MASK) >> PIND7;
```

donde se utiliza el valor del registro `PIND` para la lectura del Puerto D, la que se enmascara con `SW_MASK`, o sea que se ponen a cero todos los bits menos aquel cuyo valor interesa (a esto se le denomina máscara).

Programación de los puertos digitales – entradas

La constante simbólica `SW_MASK` se utiliza como máscara para la lectura de la entrada digital (línea 23).

La línea que realiza la lectura de la entrada es:

```
sw = (PIND & SW_MASK) >> PIND7;
```

donde se utiliza el valor del registro `PIND` para la lectura del Puerto D, la que se enmascara con `SW_MASK`, o sea que se ponen a cero todos los bits menos aquel cuyo valor interesa (a esto se le denomina máscara).

Este valor luego se desplaza hacia la derecha para ocupar el bit menos significativo de la variable de 8 bits de nombre `sw`, que se utiliza luego en la estructura de selección `if` para determinar si hay que encender el LED.

Programación de los puertos digitales – entradas

La constante simbólica `SW_MASK` se utiliza como máscara para la lectura de la entrada digital (línea 23).

La línea que realiza la lectura de la entrada es:

```
sw = (PIND & SW_MASK) >> PIND7;
```

donde se utiliza el valor del registro `PIND` para la lectura del Puerto D, la que se enmascara con `SW_MASK`, o sea que se ponen a cero todos los bits menos aquel cuyo valor interesa (a esto se le denomina máscara).

Este valor luego se desplaza hacia la derecha para ocupar el bit menos significativo de la variable de 8 bits de nombre `sw`, que se utiliza luego en la estructura de selección `if` para determinar si hay que encender el LED.

La constante simbólica `SW_PRES` fija el valor que tendrá la entrada digital si el pulsador se encuentra presionado, lo cual dependerá de cómo este armado el circuito.