|  |  |
| --- | --- |
| TRABAJO PRÁCTICO: MICROCONTROLADORES – Partes 1 a 4 | |
| Asignatura: | Microcontroladores y Electrónica de Potencia |
| Carrera: | Ingeniería Mecatrónica |
| Año: 2024 | UNIDAD 3: MICROCONTROLADORES Y OTROS SISTEMAS EMBEBIDOS |

# Parte 1: Manipulación de bits

*(ver en página de asignatura las diapositivas 27 y 28 de la* [*presentación U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86053) *y leer el* [*apunte*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86056) *de operaciones bit a bit).*

**Objetivo:** Familiarizarse con la manipulación de bits y la notación específica.

**Ej 1.1** Exprese el valor de A en binario y en notación hexadecimal:

A = (1<<3); Rta: A = 0b00001000; A = 0x08;

A = (1<<2)|(1<<7);

A = ~(1<<2);

A = ~((1<<1)|(1<<4));

A = (3<<3);

**Ej 1.2** Exprese en notación de operadores y en notación hexadecimal

A = 0b10010000; Rta: A = (1<<7)|(1<<4); A = 0x90;

A = 0b01010000;

A = 0b11101001;

A = 0b01110111;

A = 0b00100110;

**Ej 1.3** Exprese qué le sucede a la variable **A (a7 a6 a5 a4 a3 a2 a1 a0)**

A |=(1<<2); Rta: A = a7 a6 a5 a4 a3 **1** a1 a0 //*Su bit 1 se pone a 1*

A &=(1<<5);

A &=~(1<<4);

A ^=(1<<1);

A &=~(7<<1);

A &=(7<<1);

#define PULS 2; A |=(1<<PULS);

**Ej 1.4** Manipular los bits solicitados (sin alterar el resto de los bits)

1. Poner en ‘1’ el bit 2 de una variable PuertoB Rta: PuertoB |= (1<<2);
2. Invertir el bit 7 de una variable PuertoA
3. Poner en ‘0’ el bit 2 de una variable PuertoB
4. Verificar con un “if” si el bit 2 de una variable PuertoA es ‘1’ o ‘0’
5. Poner en ‘1’ los bits 1, 4 y 6 de una variable PuertoC
6. Colocar en “110” los bits consecutivos 6,5,4 de una variable PuertoA

# Parte 2: Configuración y uso de GPIO

*Ver diapositivas 30 a 33 de la* [*presentación U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86053) *y la sección 3.2 en el* [*apunte U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86054) *“Programación de Microcontroladores”.*

**Objetivo:** Introducir al manejo de periféricos mediante manipulación de registros. Observar las limitaciones en la respuesta y en la precisión temporal por el uso de *pollings* y *delays*.

**Ej 2.1** Realizar un programa en ATmega328P con salida en PB4. La salida PB4 debe producir una señal cuadrada de 10 Hz permanente.

**Ej 2.2** Realizar un programa en ATmega328P con dos salidas en PB4 y PB2 y entrada en PD3. La salida PB4 debe producir una señal cuadrada de 2 Hz permanente, y la salida en PB2 debe iniciar en ‘0’ e invertirse cada vez que se pulse PD3 (paso de 0 a 1).

*Observar qué sucede si PD3 se oprime muy rápido, o muy lento, proponga cómo subsanar los problemas que se presentan, y concluya qué limitaciones encuentra a las soluciones posibles.*

**Ej 2.3** Realizar un programa en ATmega328P con salidas **PB4** y **PB5**, y entradas **PD2** y **PD4**.

El autómata tendrá dos estados, *Activado* y *Desactivado*. Inicialmente estará *Activado*, y en este estado deberá producir una señal de 0,5 **Hz**, **Duty Cycle** 30% en **PB5**. Al pulsar **PD4** debe pasar a modo *Desactivado*, poniendo ambas salidas en 0. Al pulsar **PD2** debe volver al estado *Activado,* poniendo **PB4** en ‘1’ (en este estado debe volver a generarse la señal de 0,5 Hz/30% en **PB5**).

Representar el funcionamiento (Ejs 2.1, 2.2 y 2.3) mediante diagramas temporales mostrando las señales PD2/PD4/PB4/PB5 según corresponda

Observar qué sucede si PD2 y PD4 se oprimen muy rápido o muy lento, o se mantienen ambos presionados, *proponga cómo subsanar los problemas que se presentan, y concluya qué limitaciones encuentra a las soluciones posibles.*

# Parte 3: Interrupciones de pines.

*Ver diapositivas 33 a 42 de la* [*presentación U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86053) *y las secciones 5 a 5.4 en el* [*apunte U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86054) *“Programación de Microcontroladores” (pasar por alto lo relacionado con microcontroladores PIC).*

**Objetivo:** Introducirel concepto de *interrupción* y de *rutina de servicio* *de interrupción* *ISR*. Aprender a configurar los subsistemas de interrupciones de pines del AVR. Comparar la respuesta obtenida por el uso de interrupciones de pines con respecto al *polling* de pines de entrada.

**Ej 3.1** Realizar el mismo ejemplo 2.2, pero con **PD3** leída por interrupción (INT1, flanco de subida).

**Ej 3.2** Realizar el mismo ejemplo 2.3, pero con **PD2** y **PD4** por interrupción, flanco de subida (PD2 como INT0, PD4 como PCINT20).

**Ej 3.3** Realizar un programa en ATmega328P con tres salidas: **PB1**, **PB2** y **PB3**, y 2 entradas en **PD2**/PCINT18/INT0 y **PD3**/PCINT19/INT1

Cada salida será manejada por una tarea distinta.

La salida **PB1** será manejada en la tarea principal y oscilará permanentemente a 2 Hz.

La salida **PB2** será manejada por una rutina de servicio de interrupción del tipo PCI (Pin Change Interrupt), que producirá 10 ciclos a 10 Hz cuando haya *flanco de subida* en **PD2**.

La salida **PB3** será manejada por una rutina de servicio de interrupción del tipo *External INT,* que producirá 15 ciclos a 10 Hz cuando haya *flanco de subida* en **PD3**. **Esta interrupción deberá tener prioridad sobre la anterior.**

Representar el funcionamiento (Ejs 3.1, 3.2 y 3.3) mediante diagramas temporales mostrando las señales de entrada y las respuestas correspondientes.

# Parte 4: UART.

*(Ver diapositivas 45 a 62 de la* [*presentación U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86053) *y la sección 6 en el* [*apunte U3*](https://aulaabierta.ingenieria.uncuyo.edu.ar/mod/resource/view.php?id=86054) *“Programación de Microcontroladores” (pasar por alto, por ahora, lo relacionado con microcontroladores PIC y ARM).*

**Objetivo:** Programar una UART para establecer una comunicación serie asíncrona en el microcontrolador. Experimentar concepto de *baudrate* y error de *baudrate*, interrupciones de UART, intérprete de comandos, protocolo de comunicaciones, *driver*, *buffer* de comunicación y variables de tipo *volatile*.

**Ej 4.1** Realizar un oscilador con **semiperíodo** ajustable por consignas por UART de tipo “:Txx”, con xx tiempo en **ms** de 1 a 50000, como “cadena decimal”. La recepción de datos no debe detener el funcionamiento del oscilador. Usar **PB4** como salida.

**Ej 4.2** Realizar el siguiente automatismo supervisado:

Control Pulso-Dirección de un driver de motor PaP o servomotor, mediante 2 pines de PORTB.

Recibe consigna de **posición** P(0 a 30000 pulsos) y de **período** T(100 a 30000 µs) como cadenas decimales. Debe aceptar modificación de la consigna durante el movimiento. Usar como salidas **PB2** (pulso) y **PB4** (dirección).

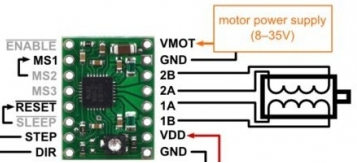
*Si no cuenta con driver y motor, experimentar con leds en las salidas de “Pulso” y “Dirección” generadas por el microcontrolador. No olvidar conectar a través de resistencias de 1k aproximadamente. En este caso* ***utilizar tiempos mayores*** *(T de 10 a 3000 milisegundos) para poder apreciar visualmente el parpadeo del led.*

*Si no cuenta con placa de microcontrolador, comprobar funcionamiento en simulador. El programa que se entrega es con los tiempos del enunciado.*

UART

PC

ATmegaxxx



GND

Solo en A4988

**No** en DRV8825

5V