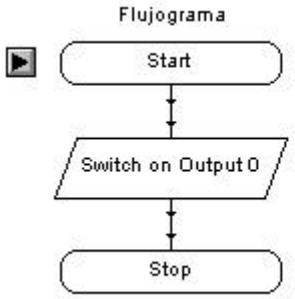
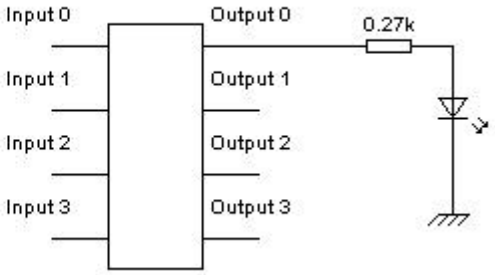
Programación de Microcontroladores

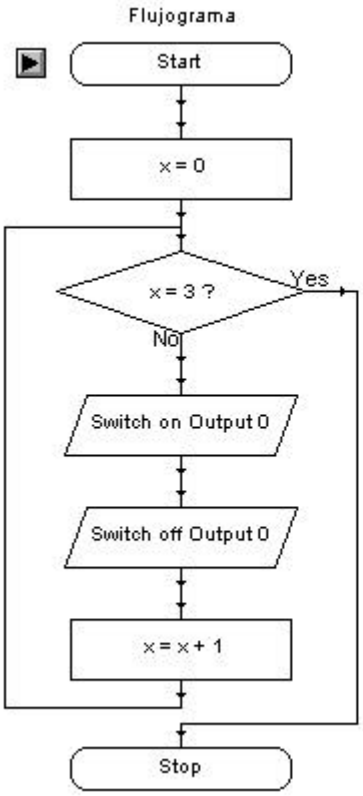
Simulación en Crocodile Technology

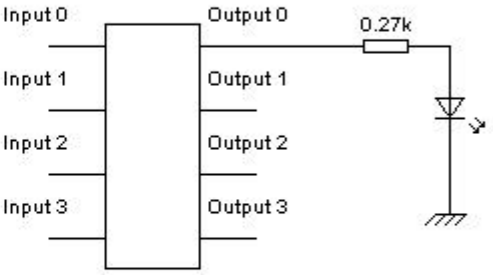
Práctica Nº 1 Encender un led cuando arranque el microcontrolador.



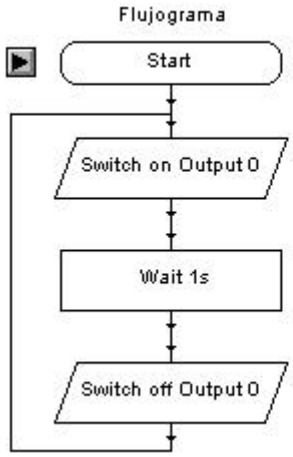


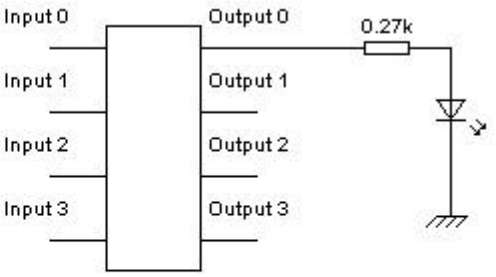
Práctica Nº 2 Encender un led tres veces consecutivas.



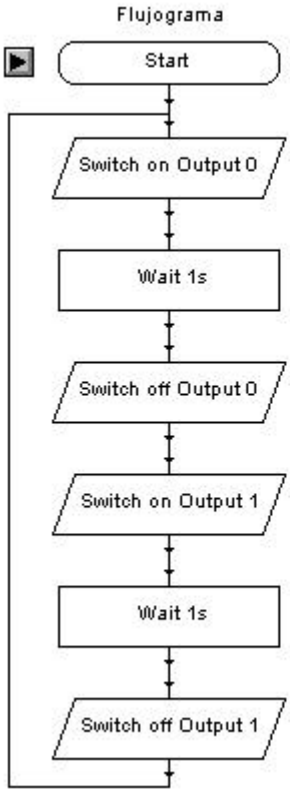


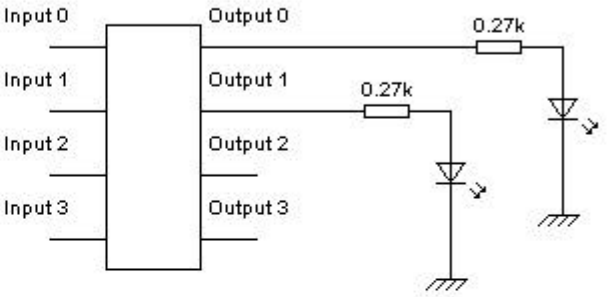
Práctica Nº 3 Hacer destellar el led a razon de 1 segundo aproximadamente.



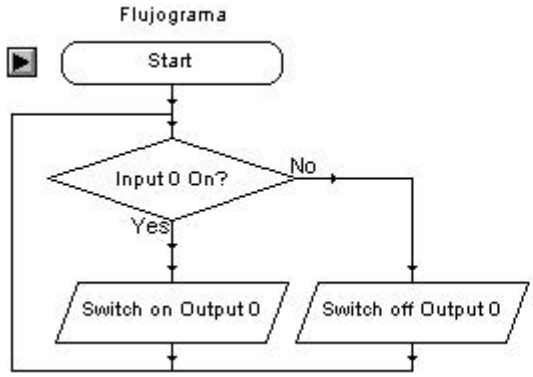


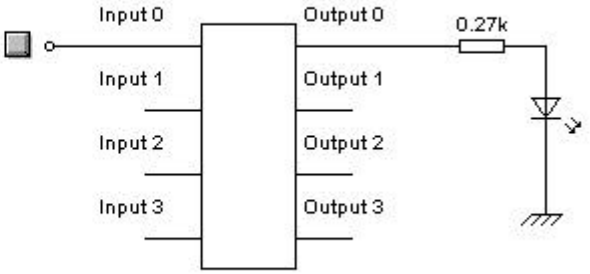
Práctica Nº 4 Hacer destellar alternadamente dos led a razon de 1 segundo aproximadamente.



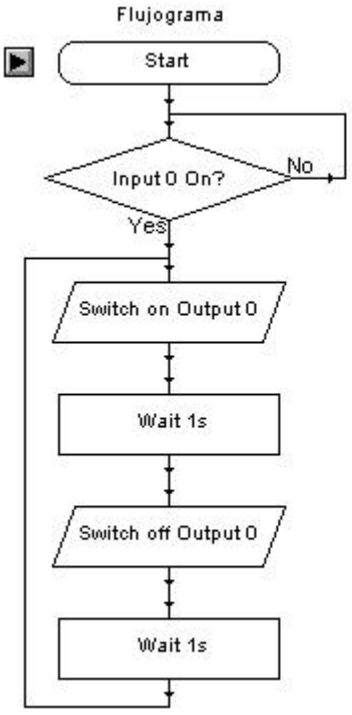


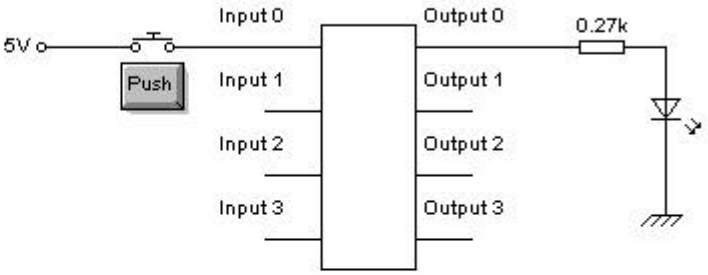
Práctica Nº 5 Encender un led de acuerdo al estado de un interruptor.



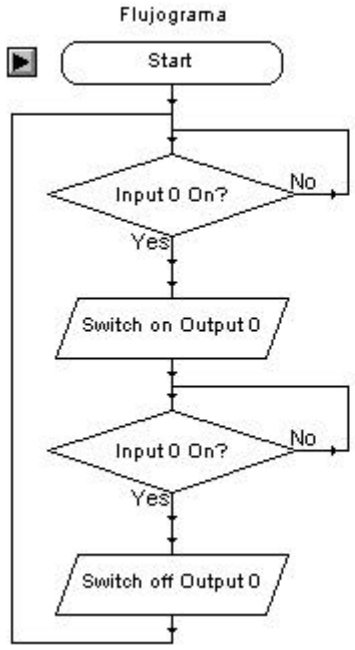


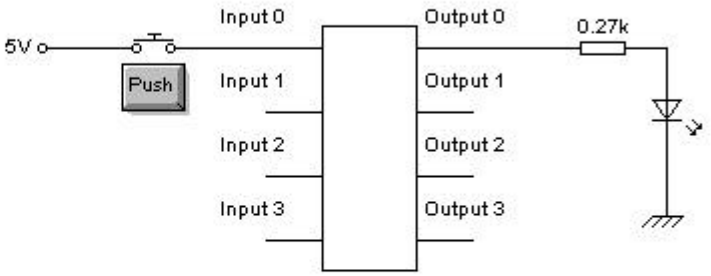
Práctica Nº 6 Detectar cuando se presione un pulsador, encendiendo un led en forma intermitente a intervalos de 1 seg aproximadamente.



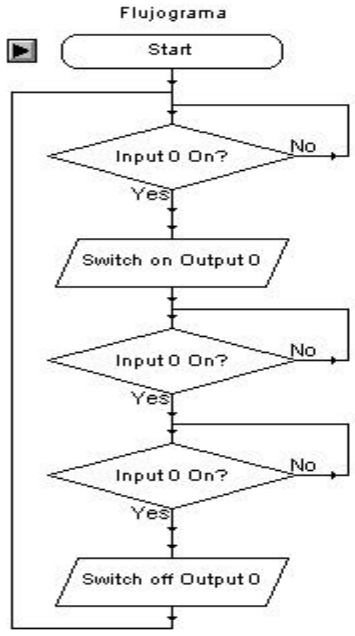


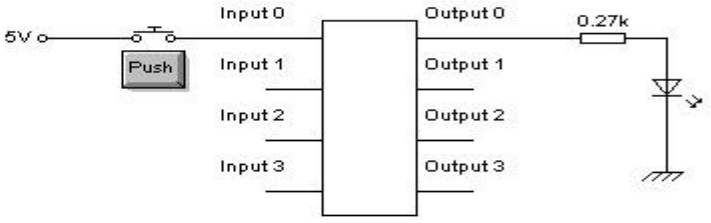
Práctica Nº 7 Detectar cuando se presione un pulsador y activar un led. Cuando se presione de nuevo el pulsador se debera apagar.



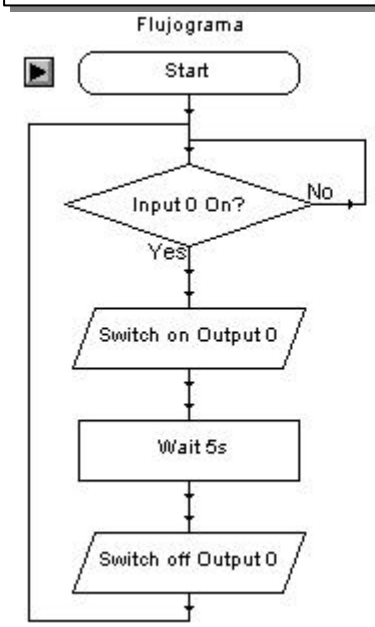


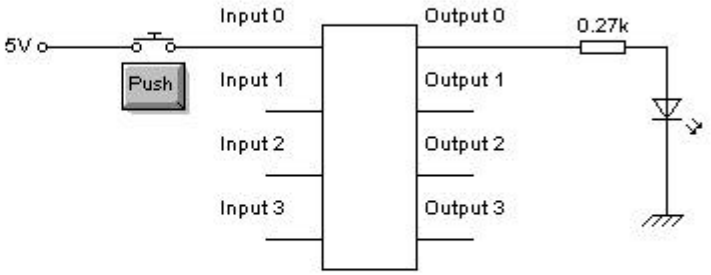
Práctica Nº 8 Encender el led cuando se presione el pulsador y apagarlo cuando se presione dos veces consecutivas.



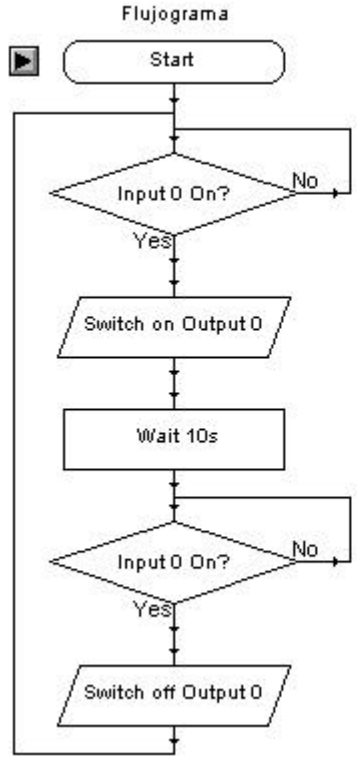


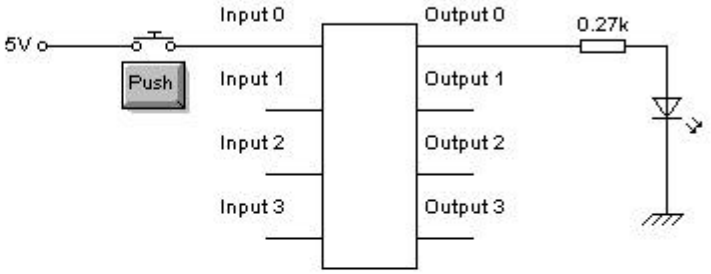
Práctica Nº 9 Detectar cuando se presione un pulsador y activar un led durante un tiempo determinado.



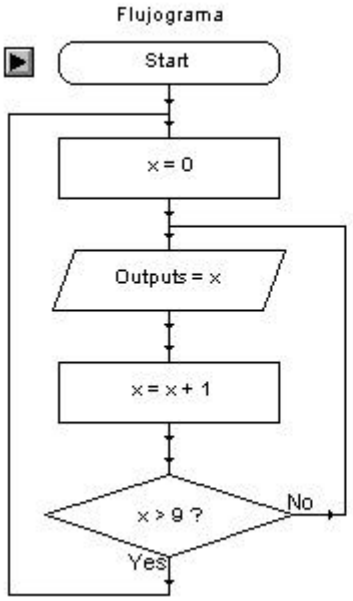


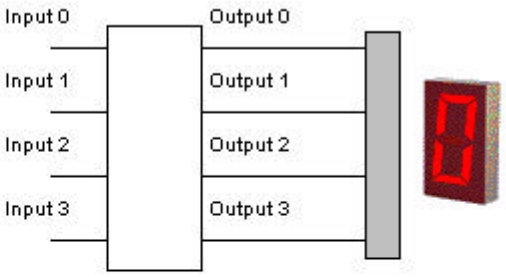
Práctica Nº 10 Encender el led cuando se presione el pulsador y apagarlo unicamente si luego de 10 segundos se oprime nuevamente.



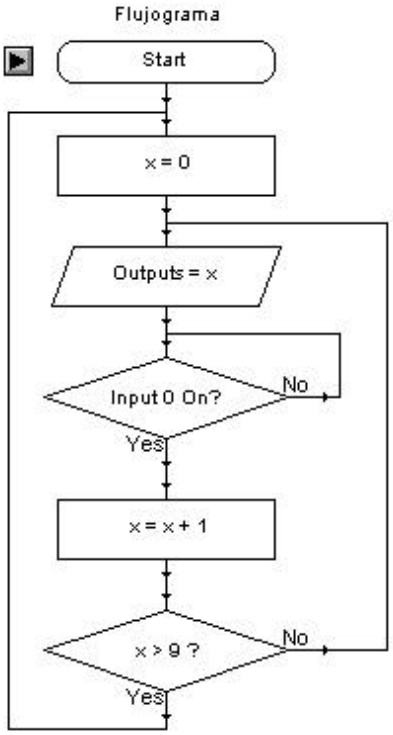


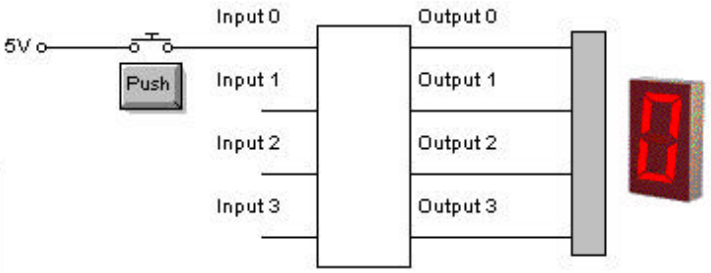
Práctica Nº 11 Visualizar una cuenta ascendente en un display de 7 segmentos, realizando la decodificación por hardware.





Práctica Nº 12 Visualizar una cuenta que se incrementa cada vez que se oprime un pulsador.

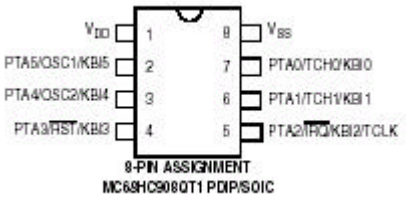




Programación de Microcontroladores

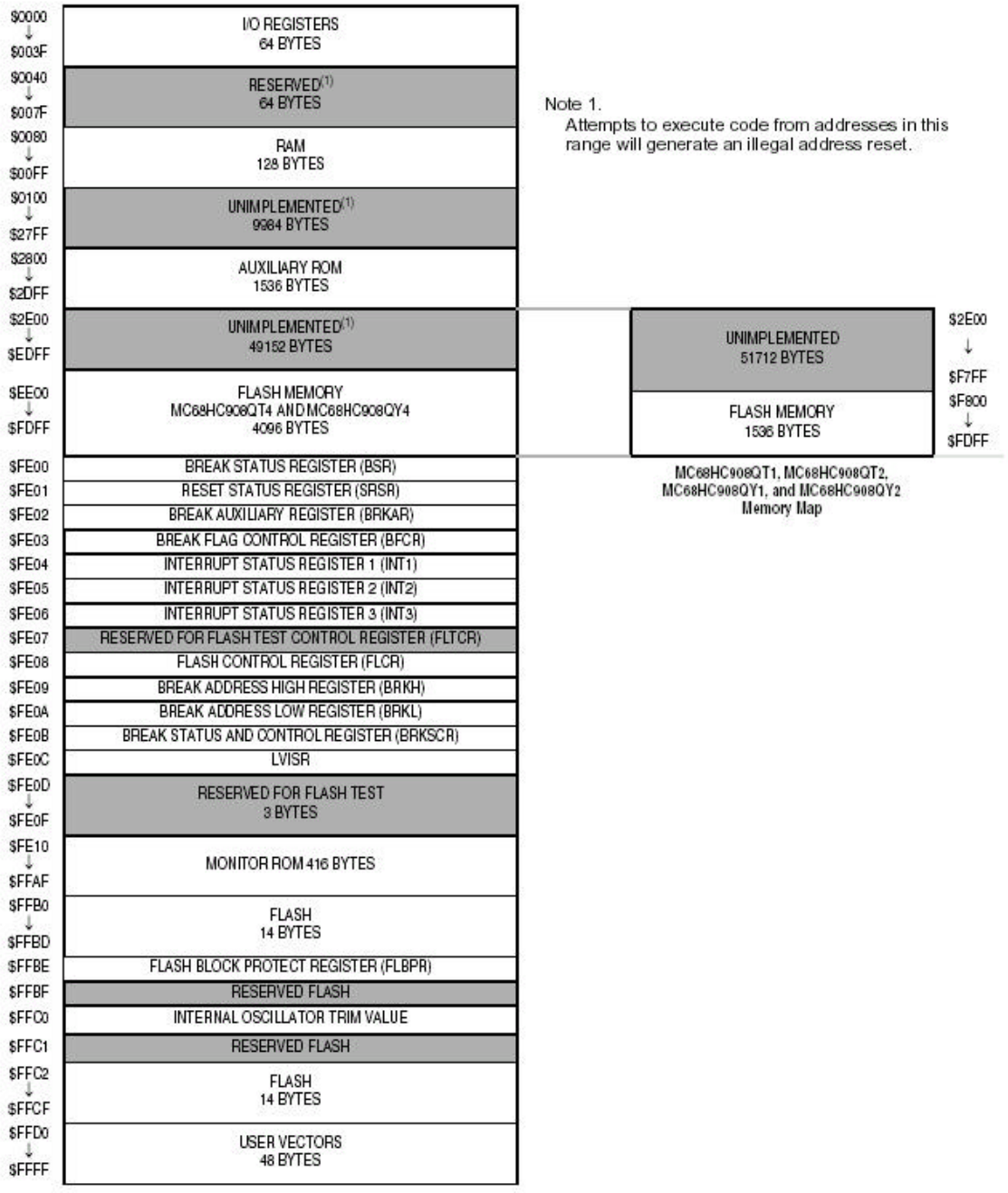
Detalles del micro MC68HC908QT1

Distribución de terminales y asignación de funciones



8-PIN ASSIGNMENT
MC68HC908QT1 PDIP/SOIC

Mapa de memoria

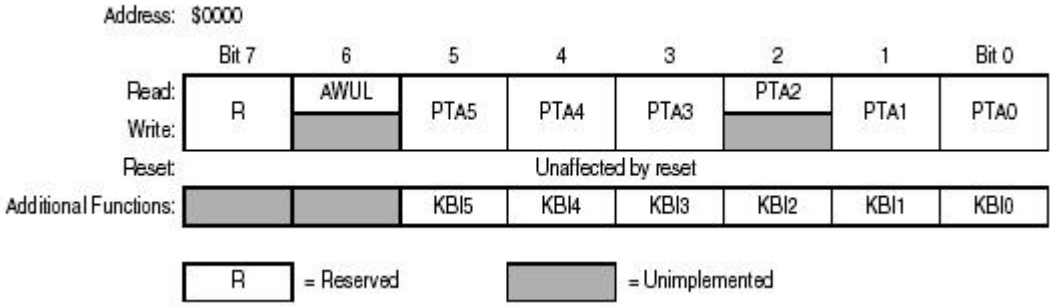


MC68HC908QT1, MC68HC908QT2, MC68HC908QY1, and MC68HC908QY2 Memory Map

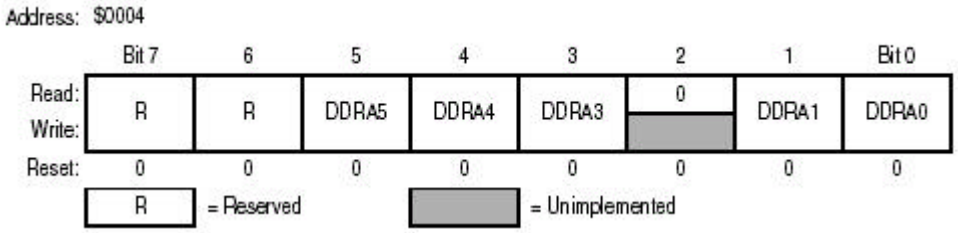
Registros más usados

Registro de datos del Puerto A (PTA)

Contiene un bit por cada uno de los seis pines del puerto A.

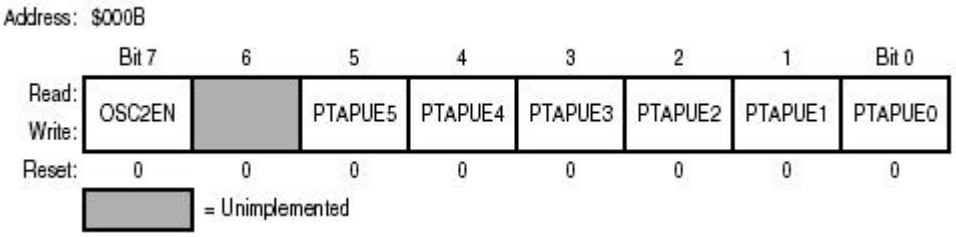


Dirección de datos del Puerto A (DDRA)



SI DDRA es "**1**" configura el bit del puerto correspondiente como **SALIDA**
SI DDRA es "**0**" configura el bit del puerto correspondiente como **ENTRADA**
PTA2 funciona sólo como entrada.

Habilitación de resistencia de pull-up del Puerto A (PTAPUE)



SI PTAPUE es "**1**" habilita una resistencia de pull-up en el pin de entrada
SI PTAPUE es "**0**" deshabilita la resistencia de pull-up en el pin de entrada
Sólo se tienen en cuenta cuando el pin correspondiente se configura como ENTRADA.

En los modelos QY se encuentra un conjunto idéntico de registros para configurar el funcionamiento del Puerto B, que consiste en 8 pines de entrada / salida. Se denominan PTB, DDRB y PTBPUE.

Registros de configuración (CONFIG)

Son usados para la inicialización de varias opciones del microcontrolador. Pueden escribirse UNA SOLA VEZ después de cada reset y se recomienda hacerlo en ese momento.

CONFIG1:

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVI5OR3	SSREC	STOP	COPD
Reset:	0	0	0	0	U	0	0	0
POR:	0	0	0	0	0	0	0	0

U = Unaffected

SI COPD es “1” deshabilita el COP
SI COPD es “0” habilita el COP

CONFIG2:

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IRQPUD	IRQEN	R	OSCOPT1	OSCOPT0	R	R	RSTEN
Reset:	0	0	0	0	0	0	0	U
POR:	0	0	0	0	0	0	0	0

R = Reserved U = Unaffected

SI OSCOPT1: OSCOPT0= “**00**” habilita el OSCILADOR INTERNO (valor por defecto)
Si RSTEN es “1” habilita el pin de RESET en PTA3
Si RSTEN es “0” el pin PTA3 está disponible para su uso normal

Registro de ajuste del oscilador interno (OSCTRIM)

Address: $0038

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TRIM7	TRIM6	TRIM5	TRIM4	TRIM3	TRIM2	TRIM1	TRIM0
Reset:	1	0	0	0	0	0	0	0

El valor de reset fija la frecuencia a 12,8MHz ±25% (frecuencia de bus de 3,2MHz). En la posición FFC0h de la memoria está grabado un valor de ajuste en fábrica. Este valor puede copiarse durante la inicialización y garantiza la variación a ±5%.

Programación de Microcontroladores

Programación en C con ICC08

Práctica Nº 1

```
/*Este programa enciende un led*/

#include <io908qy4.h>   /*Inclusión de archivo de registros*/

void main()
{
/*aquí se ubica la inicialización del micro*/
        PTA = 0b00000000;     /*configuración del puerto A*/
        DDRA  = 0b00000001;
        PTAPUE= 0b00000000;

        asm("cli");     /*deshabilita las interrupciones */
                        /*(en Assembler)*/

        CONFIG1 = 0b00000001; /*deshabilita el timer COP*/
        COPCTL = 0b00000000;
/*fin de inicialización*/

        PTA = 0b00000001;     /*encendido del LED*/

        while(1);       /*espera indefinidamente*/
}
```

Práctica Nº 2

```
/*Este programa enciende un led tres veces consecutivas*/
#include <io908qy4.h>

/*definicion de la variables*/
int x,i;

void main()
{
        PTA = 0x00;     /*los numeros estan en formato hexadecimal*/
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;

        for (x=0 ;x<3 ;x++)
            {       PTA = 0x01;
                    for (i=0 ;i<0x0010 ;i++);/*introduce una espera*/
                    PTA = 0x0;
                    for (i=0 ;i<0x0010 ;i++);/*introduce una espera*/
            }

        while(1);
}
```

Práctica Nº 3

```
/*Este programa hace destellar el led a razon de 1 segundo aproximadamente
*/
#include <io908qy4.h>

/*                      RETARDO
*
*  Descripción    : Rutina Utilitaria de retardo programable
*  Argumentos     : ms = variable de tiempo a retardar en ms
*/

void delay(unsigned int ms)
{                       /* Función de retardo, no retorna nada y acepta
                           una variable de tipo entero sin signo */
while ((ms--) != 0)     /* Repite el ciclo mientras ms sea diferente a
                           cero */
{
      asm ( "DELAY=0x18C  ; Define constante de retardo a 12.800 MHz\n"
            "pshx             ; Salva X en la pila\n"
            "pshh             ; Salva H en la pila\n"
            "ldhx #DELAY      ; Carga constante de bucle fino\n"
"Delay0:     aix #-1          ; Decrementa H:X en 1\n"
            "cphx #0          ; LLegó a cero (0)\n"
            "bne Delay0       ; Si no es igual, salta a Delay0\n"
            "pulh             ; Si es igual, recupera H de la pila\n"
            "pulx             ; Recupera X de la pila\n"
            );
      }
}
/* fin de función RETARDO */

int x;
void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;

        while(1)              /* repite el ciclo indefinidamente*/
        {           PTA = 0x01;
                    delay(1000);
                    PTA = 0x0;
                    delay(1000);
      }
}
```

Práctica Nº 4

```
/*Hacer destellar alternadamente dos led a razon de 1 segundo
aproximadamente*/
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0b00000011;/*configura bits 0 y 1 como salidas*/
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;

        while(1)
        {           PTA = 0x01;
                    delay(1000);
                    PTA = 0x02; /*apaga bit 0 y enciende bit 1*/
                    delay(1000);
      }
}
```

Práctica Nº 5

```
/*Enciende un led de acuerdo al estado de un interruptor*/
#include <io908qy4.h>

void main()
{
        PTA = 0x00;
        DDRA  = 0b01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;

        while(1)
        if (PTA & 0b00000010)/*prueba si hay 1 en el bit 1*/
                    PTA = 0x01;
        else
                    PTA = 0x0;
}
```

Observar la técnica para verificar el estado de un bit del puerto. Se utiliza la función lógica AND

`registro & máscara` (con 1 en el bit a verificar)

Ej: verificar el bit 7

`if (PTA & 0b10000000)`…dará Verdadero si vale 1

`if ( ! (PTA & 0b10000000))`…dará Verdadero si vale 0

Práctica Nº 6

```
/*Detectar cuando se presione un pulsador, encendiendo un led en forma
intermitente a intervalos de 1 seg aproximadamente */
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (!(PTA & 0b00000010));/* espera que el bit 1 sea 1 */
        while(1)
        {           PTA = 0x01;
                    delay(1000);
                    PTA = 0x0;
                    delay(1000);
      }
}
```

La verificación se utiliza para esperar que el bit adopte el valor deseado.
Ej: esperar cambio en el bit 7

```
while (PTA & 0b10000000); espera hasta que sea 0
while (PTA & 0b10000000); espera hasta que sea 1
```

Práctica Nº 7

```
/*Detectar cuando se presione un pulsador y activar un led.
Cuando se presione de nuevo el pulsador se debera apagar. */
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {       while (!(PTA & 0b00000010));
                    PTA = 0x01;
                    delay(500);     /*evita que se detecten dos pulsos
                                     seguidos */
                    while (!(PTA & 0b00000010));
                    PTA = 0x0;
                    delay(500);
            }
}
```

Práctica Nº 8

```
/*Encender el led cuando se presione el pulsador y apagarlo cuando se
presione
dos veces consecutivas. */
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {while (!(PTA & 0b00000010)); /*método antirrebote del
                                          pulsador*/
            delay(100);   /*espera que se estabilice en 1 */
            while (PTA & 0b00000010);     /*actúa al soltar el
                                          pulsador*/
            PTA = 0x01;
            delay(500);
            while (!(PTA & 0b00000010)); /*primera pulsación*/
            delay(100);
            while (PTA & 0b00000010);
            delay(100);
            while (!(PTA & 0b00000010)); /*segunda pulsación*/
            delay(100);
            while (PTA & 0b00000010);
            PTA = 0x0;
            delay(500);
            }
}
```

Práctica Nº 9

```
/*Detectar cuando se presione un pulsador y activar un led durante un
tiempo
determinado. */
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {       while (!(PTA & 0b00000010));
                    delay(100);
                    while (PTA & 0b00000010);
                    PTA = 0x01;
```

```
                delay(5000);
                PTA = 0x0;
                delay(500);
            }
}
```

Práctica Nº 10

```
/*Encender el led cuando se presione el pulsador y apagarlo unicamente si
luego de 10 segundos se oprime nuevamente. */
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

void main()
{
        PTA = 0x00;
        DDRA  = 0x01;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {       while (!(PTA & 0b00000010));
                    delay(100);
                    while (PTA & 0b00000010);
                    PTA = 0x01;
                    delay(10000);
                    while (!(PTA & 0b00000010));
                    delay(100);
                    while (PTA & 0b00000010);
                    PTA = 0x0;
                    delay(500);
            }
}
```

Práctica Nº 11

```
/*Visualizar una cuenta ascendente en un  display de 7 segmentos,
realizando la decodificación por hardware. */
/*Las salidas son PTA5,PTA4,PTA3 y PTA1; recordar que PTA2 solo es entrada;
se reserva PTA0 para un pulsador*/
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

char x;
char aux;
void main()
{
        PTA = 0x00;
        DDRA  = 0b00111010;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {     for (x=0; x<10; x++)
                  {
                        aux = x<<2;/*desplaza 2 bits a la izquierda*/
```

```
                        if (aux & 0b00000100)        /*si bit 2 es 1*/
                              aux = aux | 0b00000010; /*pone 1 en bit 1*/
                        PTA = aux;
                        delay(1000);
                  }
            }
}
```

Para encender un bit se usa la función OR

`registro | máscara` (con 1 en el bit a encender)

Ej: Encender el bit 7

```
PTA | 0b10000000;
```

Para apagarlo se utiliza la función lógica AND

`registro & máscara` (con 0 en el bit a apagar)

Ej: apagar el bit 4

```
PTA & 0b11101111;
```

Práctica Nº 12

```
/*Visualizar una cuenta que se incrementa cada vez que se oprime un
pulsador. */
/*Las salidas son PTA5,PTA4,PTA3 y PTA1; recordar que PTA2 solo es entrada;
se usa PTA0 para el pulsador*/
#include <io908qy4.h>
/*insertar aquí la función delay()de la Práctica 3*/

char x;
char aux;
void main()
{
        PTA = 0x00;
        DDRA  = 0b00111010;
        PTAPUE= 0x00;
        asm("cli");
        CONFIG1 = 0x01;
        COPCTL = 0x00;
        while (1)
            {     for (x=0; x<10; x++)
                  {
                        aux = x<<2;
                        if (aux & 0b00000100)
                              aux = aux | 0b00000010;
                        PTA = aux;
                        while (!(PTA & 0b00000010));
                        delay(100);
                        while (PTA & 0b00000010);
                        delay(1000);
                  }
            }
}
```