PIC MICRO ESTUDIO

Entrenador de PIC's para la Gama de 8 Pines (12C5XX, 12F6XX)

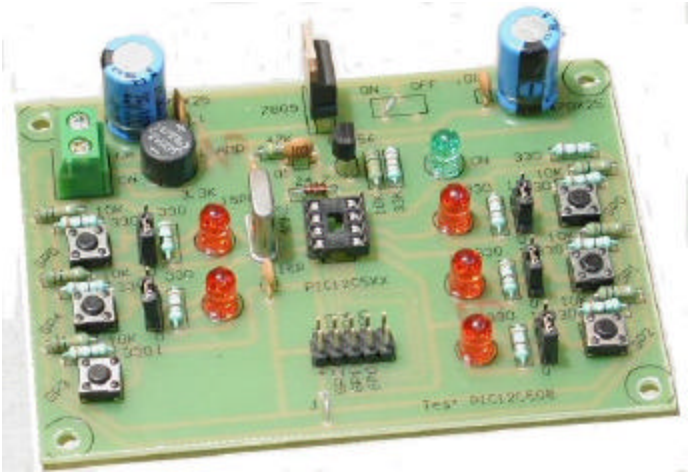
www.electronicaestudio.com

Guía de Uso

Entrenador de PIC's para la Gama de 8 Pines

Aprenda a programar el PIC12C508 con una tarjeta lista para usarse. No malgaste su tiempo alambrando en tarjetas universales, ejecute sus programas en esta tarjeta fácilmente y así compruebe rápidamente sus programas. Por esta razón **Picmicro Estudio** ofrece este módulo con el nombre de **Test12C508 (Clave: 509).**

El cual consta de las siguientes características:



- **Zócalo para montar las siguientes familias**:
  - **12C508**
  - **12C509**
  - **12CR509A**
  - **12CE518**
  - **12C671**
  - **12F675**
  - **12F629**
- **1 cristal de 4 Mhz**
- **Fuente de alimentación**
- **5 Leds indicadores de salidas**
- **6 Botones Pulsadores, para la simulación de entrada de datos**

Descripción del Circuito

Básicamente esta tarjeta consta de las siguientes partes:

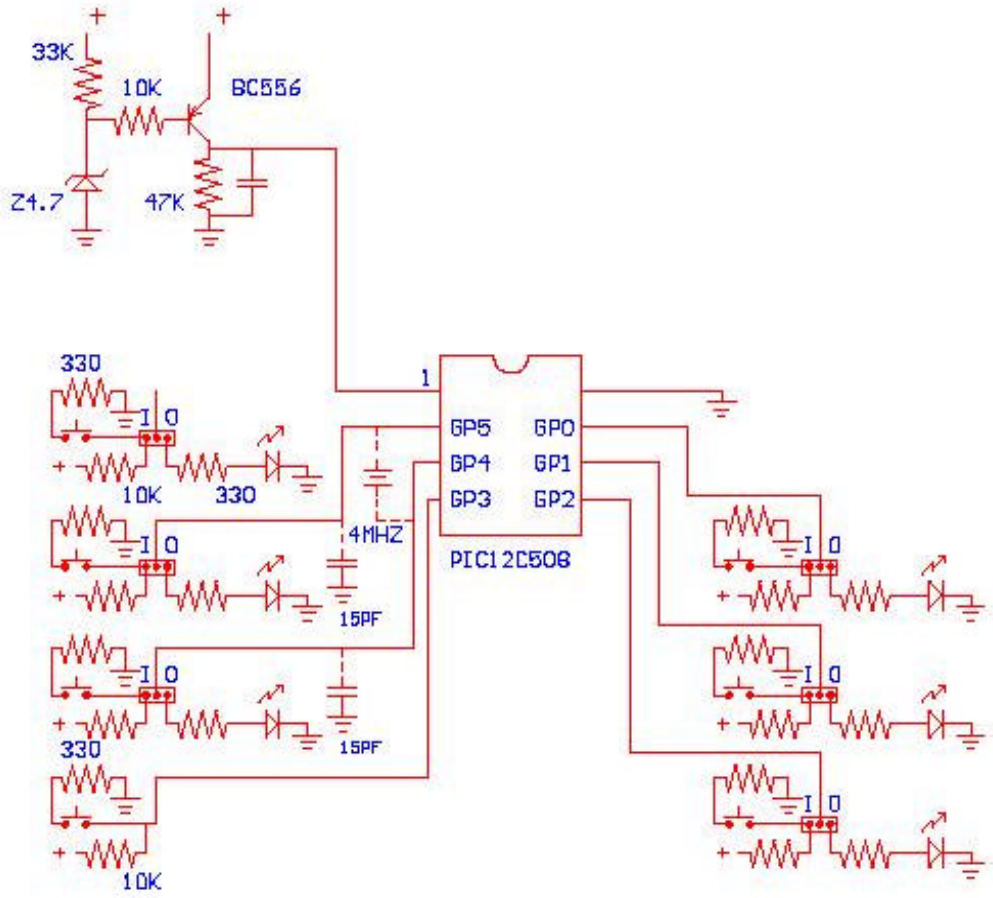
- **Base para el PIC**
  Es donde se debe de insertar el PIC12XXX que se va utilizar. Por favor antes de insertar el PIC apague la tarjeta con el interruptor de encendido.

- **Fuente de Alimentación.**
  Para alimentar la tarjeta se usan los bornes que se encuentran a un costado de la tarjeta. Se puede utilizar una batería, un eliminador universal o un transformador.

- **Botón de Encendido (Remplazado por un Jumper)**
  Para encender la tarjeta se hace uso de un interruptor deslizable y un Led indicador de encendido nos permite verificar que la tarjeta esta lista para funcionar.

- **Reset**
  El botón de **Reset,** tiene como tarea reiniciar el programa que esta ejecutando el microcontrolador. Sin embargo en este circuito se opto por introducir un **Reset** automático, el cual consta de una red RC con un diodo **Zener**, el cual se activa cuando la fuente de la alimentación cae por debajo de Vzener aproximadamente.

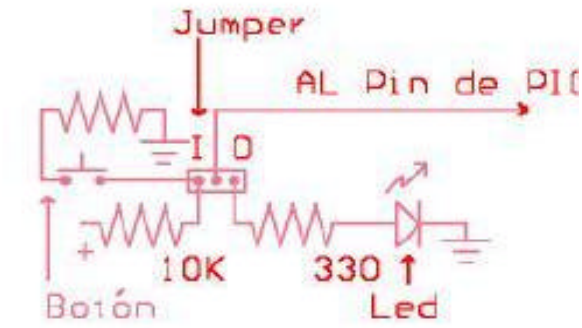
A continuación se describe el diagrama esquemático completo de esta tarjeta el cual también se encuentra en el disco que acompaña este producto (Clave 509).



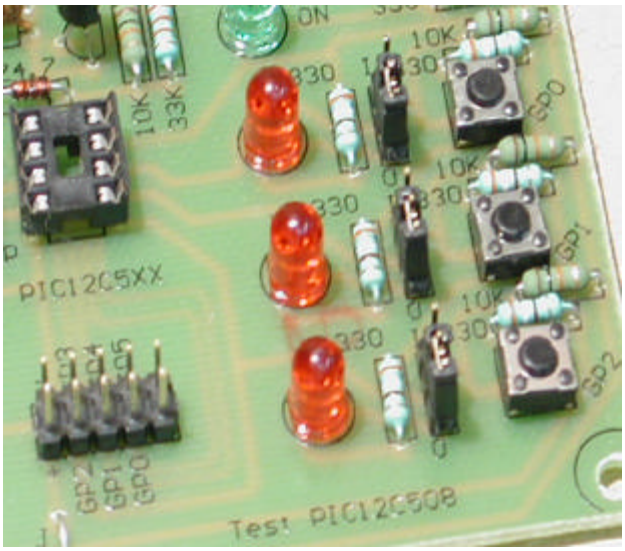
Configuración de las Terminales

Si una terminal se programa como **ENTRADA**, entonces esta podrá detectar un voltaje: “0” o “1”, que a su vez puede ser la apertura o el cierre de un interruptor, un botonazo, o la acción de un sensor.

Si una terminal se programa como **SALIDA,** será capaz de prender un Led, energizar un relevador o un solenoide, etc.



Entradas/Salidas



Cuando se quiere estudiar el comportamiento de un a terminal como salida el jumper se conecta en “O” (Output) y entonces el Led queda disponible para encenderlo o apagarlo.

Cuando lo que se necesita es insertar un pulso a alguna terminal del microcontrolador, el jumper se conecta en “I” (Input). En esta condición cualquier terminal recibe un “UNO” con el botón desactivado y un “CERO” cuando se oprime el botón.

A continuación se procederá a explicar un programa, cuyo objetivo es mostrar una pequeña aplicación con dicha tarjeta.

Código Fuente " primero.asm "

Este programa tiene como objetivo que usuario presione un botón, y al hacer esto se activara un Led, si se presiona por segunda ocasión, se apaga dicho Led y se enciende otro.

Primeramente se procede a declarar el tipo de microcontrolador que se va a utilizar y además se agrega la directiva **#include** la cual contiene el nombre de todos los registros de propósito especifico que contiene el PIC.

```
list        p=12F629
#include <p12F629.inc>
```

Acto seguido se declara los fusibles de configuración, lo cual evita que al momento de programar el microcontrolador, perdamos tiempo en seleccionar estos valores**.**

```
__CONFIG   _CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF &
                  _WDT_OFF & _PWRTE_ON & _INTRC_OSC_NOCLKOUT
```

Se declaran las variables a utilizar, en este caso son:

```
    CBLOCK 0x20
       x1,x2,x3,x4,loops,loops2
    endc
```

Asignamos una etiqueta a la terminal 3 del microcontrolador

```
#define   pulsador              GPIO,3
```

Se define la dirección de origen del programa

```
       org      0x000
       goto     inicio
       org       0x005
```

Se procede a configurar el puerto **GPIO**, en el cual todos los pines actuaran como salidas excepto la terminal 3, la cual funcionara como entrada.

```
inicio   bsf      STATUS,RP0
         movwf    OSCCAL
         movlw    B'000001000'
         movwf    TRISIO
         bcf      STATUS,RP0
```

Una vez hecho esto, se ejecuta una rutina la cual enciende y apaga momentáneamente todos los Leds, el tiempo que tarda en visualizarse depende de la rutina **retardo.**

```
         clrf     GPIO
         movlw    0xff
         movwf    GPIO
         call     retardo
         movlw    0x00
         movwf    GPIO
```

En este unto el programa queda en un bucle, esperando que el usuario presione el botón pulsador, en este caso la terminal 3 del microcontrolador.

```
ciclo
        btfsc    pulsador
         goto    ciclo
        call     retardito
        btfsc    pulsador
        goto     ciclo
```

Cuando se presiona este botón, se enciende el **Pin 0** y se apaga el **Pin 1**

```
        bcf      GPIO,1
        bsf      GPIO,0
       nop
```

Nuevamente el programa queda en un bucle, esperando otra vez que se presione este botón.

Al hacer esto el microcontrolador pone en bajo la terminal 0 y pone en alto la terminal 1

```
ciclo2
        btfsc    pulsador
        goto     ciclo2
        call     retardito
        btfsc    pulsador
        goto     ciclo2
        bcf      GPIO,0
        nop
        bsf      GPIO,1
        goto     ciclo
```

Una rutina de retardo consiste en anidar varios registros, los cuales se van decrementando en una unidad.

```
retardo             ;rutina de retardo de 1 segundo
            movlw       D'6'
            movwf       x3
            movlw       D'24'
            movwf       x2
            movlw       D'168'
            movwf       x1
loop        decfsz      x1,1
            goto        loop
            decfsz      x2,1
            goto        loop
            decfsz      x3,1
            goto        loop
            retlw       0
```

```
Retardito ; retardo de 10ms
        movlw    D'100'
        movwf    loops
top2    movlw    D'110'
        movwf    loops2
top     nop
        nop
        nop
        nop
        nop
        nop
        decfsz   loops2
        goto     top
        decfsz
        goto     top2
        retlw    0
```

El código fuente se encuentra en el disco adjunto a este producto.

Código Fuente “ Ejemplo.asm ”

Se declara el tipo de PIC que se va a utilizar, en este caso es el 12C508

```
LIST P=PIC12C508
 R=HEX

W equ 0
F  equ 1
```

Se procede a declarar la dirección del puerto GPIO y de las variables a utilizar.

```
Port      equ   0x06
Count     equ   0x0c
Ncount    equ   0x0d
Mcount    equ   0x0e
```

A continuación se configura todo el Puerto GPIO como salidas.

```
      Org 0
      movlw   b'11001000'
      OPTION
      movlw    b'00001000’
      tris port
```

Al iniciar el programa se inician con 0 los registros **port, count**.

```
      nop
      nop
      nop
      clrf    port
      clrf    count
```

Por ultimo se activan todas las terminales del puerto del microcontrolador, con su respectiva rutina de retardo para que se posible su visualización

```
Start       movlw   b'00110111'
            movwf   port
            call pause
            call pause
            call pause
            call pause
            call pause
```

Una vez que se encendieron todos los Leds, se procede a apagarlos con la misma rutina de retardo.

```
            movlw   b'00000000'
            movwf   port
            call pause
            call pause
            call pause
            call pause
            call pause

          goto    Start
```

El programa queda en un bucle que repite indefinidamente

```
Pause     movlw 0xff
          movwf mcount
loadn     movlw 0xff
          movwf ncount
decn      decfsz ncount,f
          goto decn
          decfsz  mcount,f
          goto loadn
return
```

El código fuente de este programa se encuentra en el disco adjunto a este producto.