

# Control reversible para pequeños motores de CD

Algunos motores de CD de 12 o 24 Volts que consumen unos cuantos amperes son muy populares entre los aficionados a la electrónica y al mismo tiempo muy útiles en la industria para mover pequeñas cargas.

Como se construyen con imanes permanentes, para hacerlos girar solamente se requiere de una fuente de CD muy simple o en muchas ocasiones van conectados a baterías.

Este tipo de motores se encuentran en un gran número de aparatos eléctricos por ejemplo en las copadoras para mover el carro que transporta la lámpara, en los automóviles para mover los limpiadores, en los scanner, etc. Además se pueden conseguir fácilmente y son relativamente económicos.

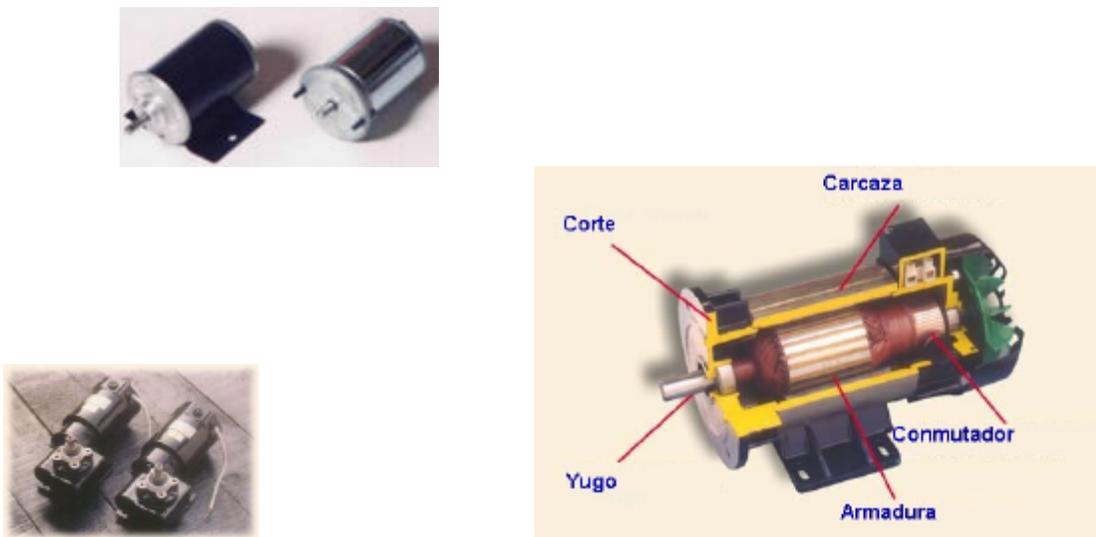


Fig.1 Motores de CD de imanes permanentes

Para invertir el sentido de rotación de este tipo de motores solo se requiere invertir la polaridad en la alimentación. Esta acción puede ser realizada de muy diversas maneras pero en este artículo lo haremos con un par de relevadores. Fig. 2

Observe que cuando se energiza el Relevador RB7 la corriente que alimenta el motor circula de izquierda a derecha y cuando se energiza el relevador RB5 la corriente circula de derecha a izquierda invirtiendo el sentido de rotación. Podemos suponer que RB7 hace girar el motor para atrás y RB5 para adelante.

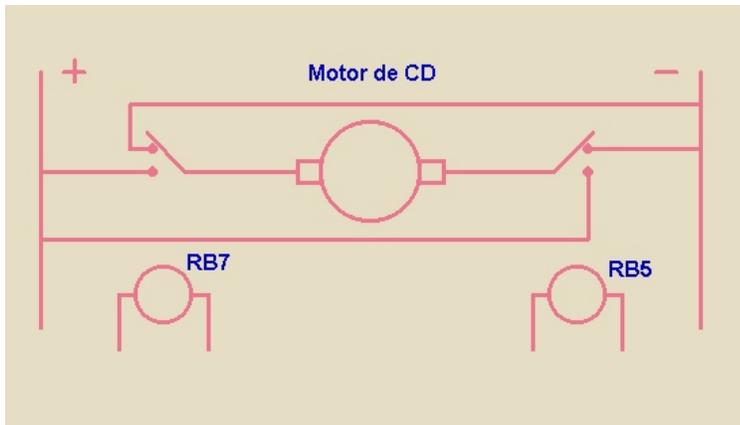


Fig. 2

En este ejercicio vamos a suponer que el motor mueve un tornillo sin fin sobre el que se encuentra montado un carrito, de tal manera que cuando el motor gira para atrás el carrito avanza para la izquierda y cuando el motor gira para adelante, el carrito se desplaza para la derecha. En cada extremo se cuenta con un interruptor de limite, que es activado por el carrito poco antes de llegar al final de la carrera Fig. 3

Se desea que al energizar el control, el carrito (sin importar en donde se encuentre) se desplace hacia el extremo izquierdo, hasta activar el Interruptor de Limite Intlsq. Que ahí permanezca unos 10 seg. Que después se dirija al extremo derecho, hasta activar IntDer para permanecer ahí otros 10 segundos y así sucesivamente.

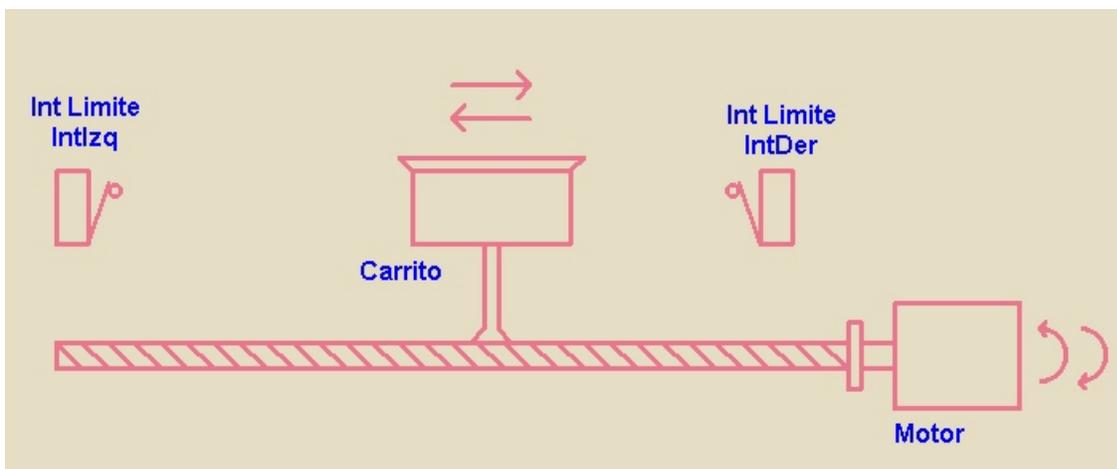


Fig. 3

El módulo de 5 entradas y 3 salidas con relevador (Clave 703) es una buena opción para resolver este ejercicio. Fig. 4

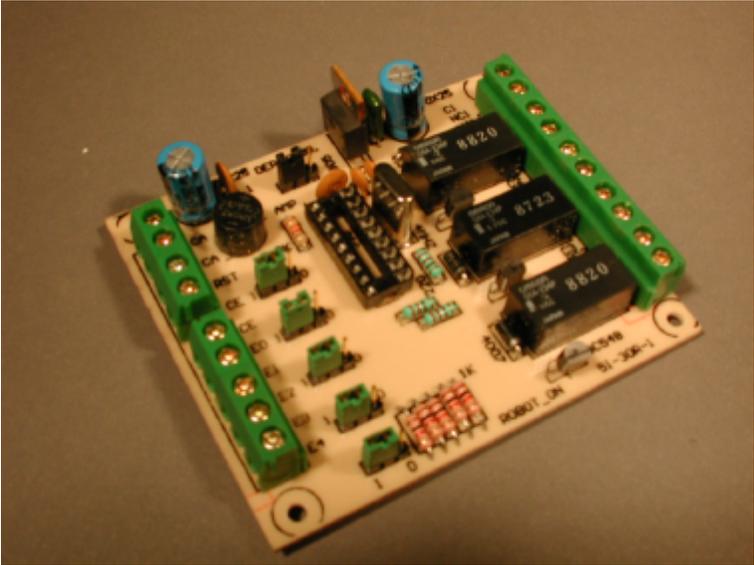


Fig. 4

Como se puede observar en el diagrama esquemático de la Fig. 5 esta tarjeta es muy flexible y puede ser utilizada para esta aplicación y para muchas otras del mismo tipo.

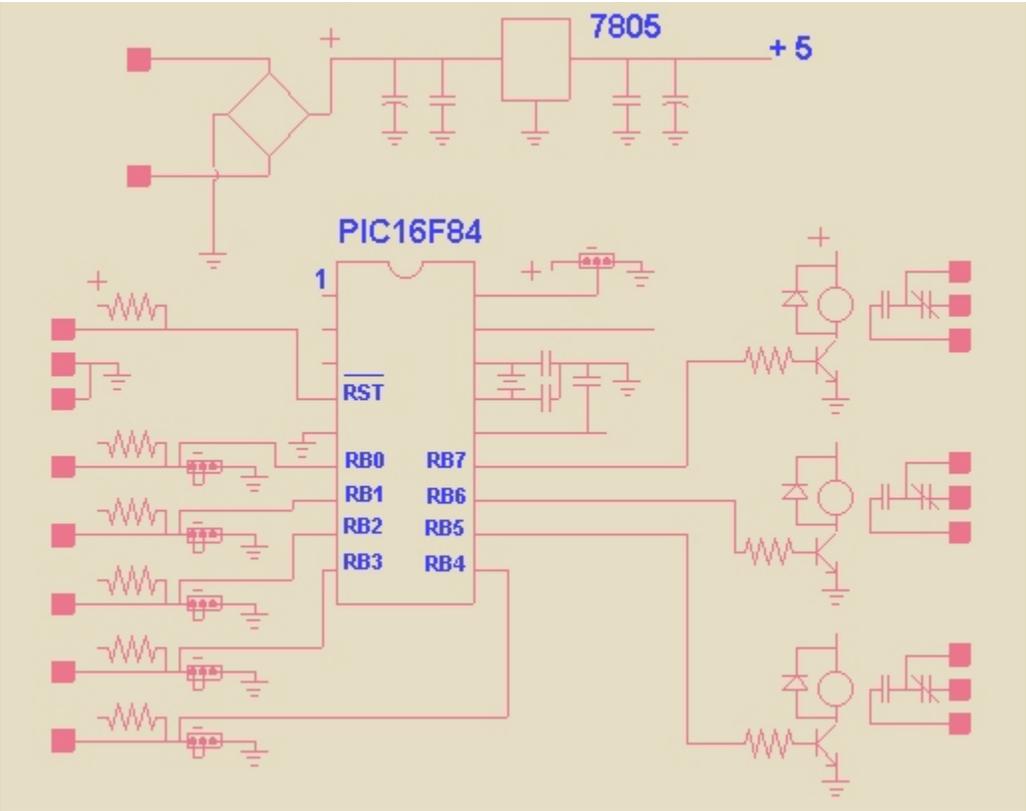


Fig. 5

Los pines RB0 a RB4 están configurados como entradas. Con el puente conectado como se indica en la figura la resistencia mantiene un “uno” en el pin del microcontrolador. Si el borne atornillable se conecta a tierra se envía un “cero”.

Los pines RB5 a RB7 están configurados como salidas. Mediante unos transistores se energizan los relevadores. En las terminales atornillables se proporcionan los contactos UPDT de los rele.

La fuente de alimentación permite que el usuario pueda energizar la tarjeta con AC/DC. Se puede conectar por ejemplo un eliminador o simplemente un transformador de 9 volts.

En este caso los interruptores de limite se conectan a las entradas y los relevadores los utilizaremos para energizar el motor y para invertir su sentido de rotación. El diagrama de alambrado se muestra en la Fig. 6

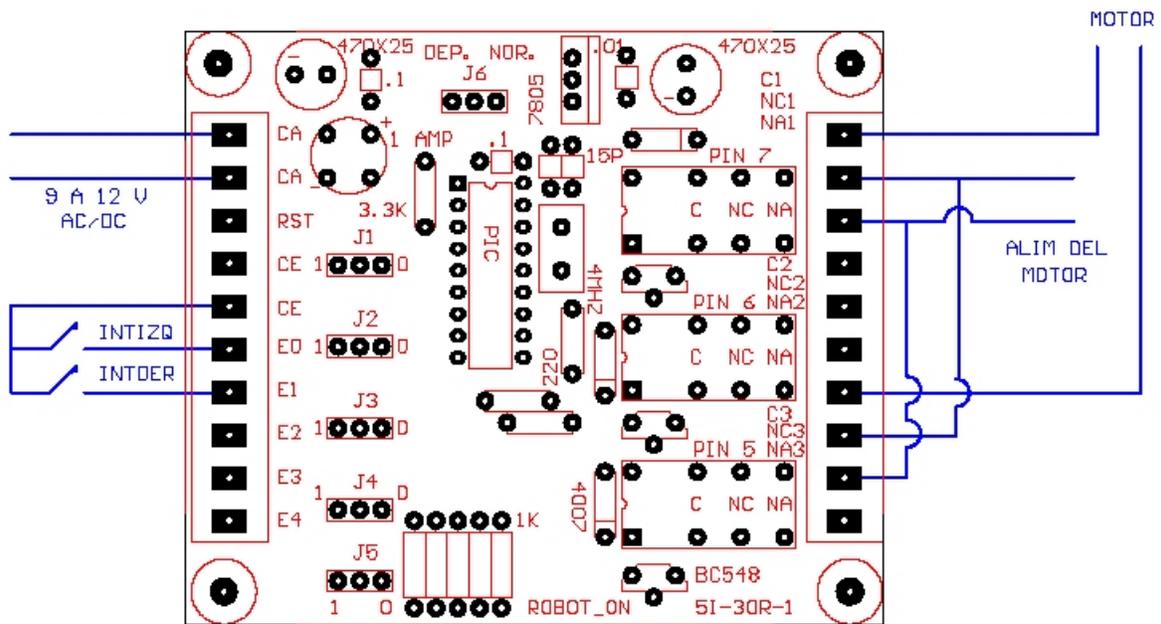


Fig. 6

A continuación se transcribe el programa para esta aplicación.

```

=====CARRITO.asm=====22 de Junio del 2001=====
;PARA SER USADO EN LA TARJETA Modulo de 5 entradas y 3 salidas con relevador
;5i-30r Clave 703
;

```

```

portb      equ    0x06

```

```

ncount    equ    0x0c    ;registro interno de paus_100ms
mcount    equ    0x0d    ;registro externo de paus_100ms
pcount    equ    0x0e    ;registro de npause_100ms

rcount    equ    0x0f    ;registro mas interno de paus_1s
scount    equ    0x10    ;registro medio de paus_1s
tcount    equ    0x11    ;registro externo de paus_1s
ucount    equ    0x12    ;registro de npaus_1s

count1    equ    0x13    ;registro mas interno de paus_1m
count2    equ    0x14    ;registro medio de paus_1m
count3    equ    0x15    ;registro externo de paus_1m
count4    equ    0x16    ;registro mas externo de paus_1m
count5    equ    0x17    ;registro de npaus_1m
veces     equ    0x18

IntIzq    equ    0x0
IntDer    equ    0x1

#define    MotorAdelante    bsf    portb,5
#define    MotorAtras      bsf    portb,7
#define    PararMotor       clrf   portb

;-----
;MACROS
;-----
OutPuerto macro    SalidaGeneral    ;SalidaGeneral b'00000000'
            movlw    SalidaGeneral    ;De izq. a derecha son las salidas 1 a 7
            movwf   portb
            endm

Minutos    macro    min                ; d'1'< min < d'255'
            movlw    min
            movwf   count5
            call    npaus_1m
            endm

Segundos   macro    seg                ; d'1'< seg < d'255'
            movlw    seg
            movwf   ucount
            call    npaus_1s
            endm

Miliseg    macro    miliseg            ; d'1'< miliseg < d'255'
            movlw    miliseg
            movwf   pcount

```

```

    call    npaus_100ms
    endm

Timer    macro    min,seg,miliseg
    if min>0
    Minutos    min
    endif

    if seg>0
    Segundos    seg
    endif

    if miliseg>0
    Miliseg        miliseg
    endif
    endm

Repite    macro    Repeticiones    ;Carga numero de repeticiones en veces
    movlw    Repeticiones    ;1< Repeticiones <255
    movwf    veces
    endm

RepitiendoDesde    macro    Etiqueta
    decfsz    veces,f
    goto    Etiqueta
    endm
;-----

    org    0x000

    movlw    b'00011111'
    tris    portb    ;Define como salida al PuertoB

    clrf    portb    ;Apaga el puerto B

    goto    Programa
;-----
;SUBRUTINAS
;-----
;paus_100ms es una pausa de 100 mili segundos = a .1 seg
paus_100ms    movlw    0x82
    movwf    mcount
loadn    movlw    0xff
    movwf    ncount
decn    decfsz    ncount,f

```

```

goto      decn
decfsz    mcount,f
goto      loadn
return

```

```

;-----
;npause_100ms repite 100ms las veces que contenga el registro pcount
;antes de entrar cargar el registro pcount con el numero deseado
npaus_100ms call      paus_100ms
              decfsz    pcount,f
              goto      npaus_100ms
              return

```

```

;-----
;paus_1s es una pausa de 1 segundo
paus_1s      movlw      0x0a      ;carga
              movwf     tcount    ;tcount
loads        movlw      0x82      ;carga
              movwf     scount    ;scount
loadr        movlw      0xff      ;carga
              movwf     rcount    ;rcount
decr         decfsz    rcount,f    ;decrementa r
              goto      decr      ;again
              decfsz    scount,f   ;decrementa s
              goto      loadr     ;again
              decfsz    tcount     ;decrementa t
              goto      loads
              return

```

```

;-----
;npause_1s repite 1s las veces que contenga el registro ucount
;antes de entrar cargar el registro pcount con el numero deseado
npaus_1s     call      paus_1s
              decfsz    ucount,f
              goto      npaus_1s
              return

```

```

;-----
;paus_1m es una pausa de 1 minuto
paus_1m      movlw      0x3c      ;carga 60 decimal
              movwf     count4    ;count4
load3        movlw      0x0a      ;carga
              movwf     count3    ;count3

load2        movlw      0x82      ;carga 82 se ADELANTA .060
              movwf     count2    ;count2
load1        movlw      0x00      ;carga
              movwf     count1    ;count1

```

```

dec1      decfsz    count1,f    ;decrement 1
          goto     dec1        ;again
          decfsz   count2,f    ;decrement 2
          goto     load1       ;again
          decfsz   count3     ;decrement 3
          goto     load2       ;again
          decfsz   count4     ;decrement 3
          goto     load3       ;again
          return    ;done

```

```

;-----
;npause_1m repite 1m las veces que contenga el registro count5
;antes de entrar cargar el registro count5 con el numero deseado
npaus_1m  call     paus_1m
          decfsz   count5,f
          goto     npaus_1m
          return

```

```

;-----
;-----

```

Programa

MotorAtras

ChecaIntIzqActivado

```

btfsc    portb,IntIzq    ;Salta la siguiente instrucción si..
goto     ChecaIntIzqActivado

```

PararMotor

Segundos d'10' ;Pausa de 10 segundos

MotorAdelante

ChecaIntDerActivado

```

btfsc    portb,IntDer
goto     ChecaIntDerActivado

```

PararMotor

Segundos d'10'

goto Programa

end

Como puede observar este programa es muy parecido al que se presenta en el artículo anterior, por lo que comentaremos solamente las instrucciones nuevas.

### **#DEFINE**

Se usa para sustituir una instrucción por una etiqueta.

Sintaxis

```
#define <nombre > <cadena de caracteres>
```

En el cuerpo del programa, se utiliza <nombre>, tantas veces como se requiera en lugar de la <cadena de caracteres>.

### **Btfsc (Bit test, skip if clear)**

Se utiliza para de detectar si en bit de un registro esta en cero.

Sintaxis

```
btfsc <registro>,bit
```

Quiere decir que se checa el bit y si es cero salta la siguiente instrucción.

En la Fig, 7 se presenta una fotografía del proyecto armado en nuestro laboratorio.

Para editar y compilar este programa es conveniente usar MPLAM y MPASAM respectivamente. Si se desea también se puede bajar el file **carrito.zip** de la dirección de Internet

[www.prodigyweb.net.mx/wgb/articulos](http://www.prodigyweb.net.mx/wgb/articulos)

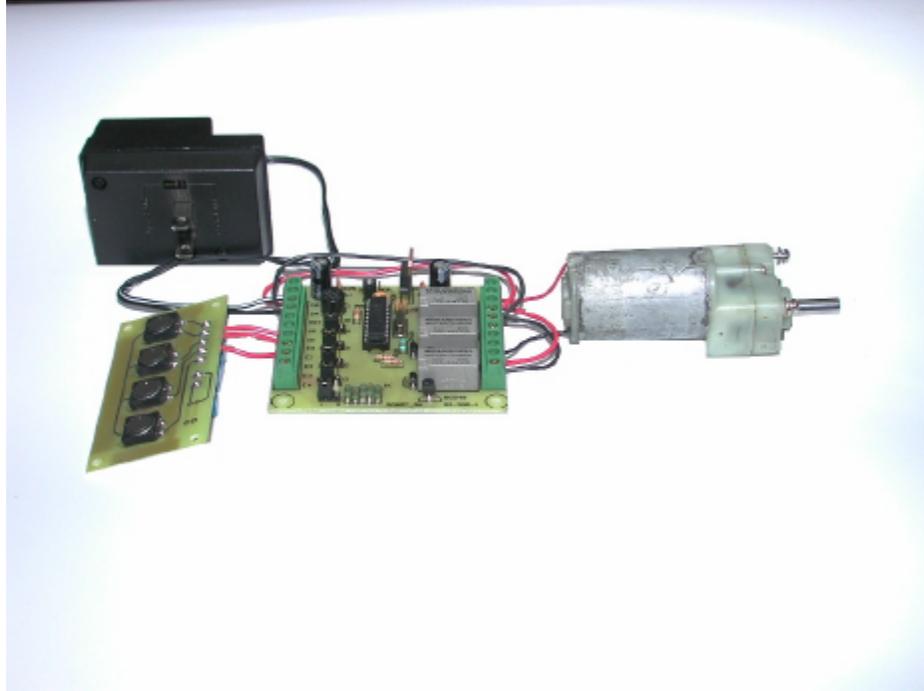


Fig. 7