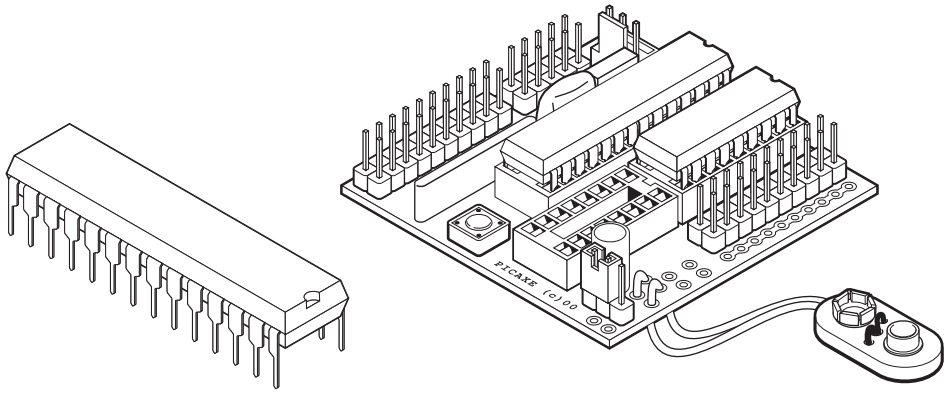
# SISTEMA DE PROGRAMACIÓN DE MICROCONTROLADOR



El "PICAXE" es un sistema de microcontroladores fáciles de programar que explota las características únicas de la nueva generación de microcontroladores de bajo costo con memoria FLASH. Estos microcontroladores pueden ser programados una y otra vez sin la necesidad de un costoso programador.

El poder del sistema PICAXE radica en su sencillez. No necesita de ningún programador, borrador o complejo sistema electrónico – el microcontrolador es programado (con un simple programa en BASIC o un diagrama de flujo) mediante una conexión de tres alambres conectada al puerto serie (USB) del ordenador. El circuito operacional PICAXE utiliza únicamente tres componentes y puede ser ensamblado fácilmente en un tablero experimental para componentes electrónicos, en una placa corriente o en una placa PCB.

El microcontrolador PICAXE-28 provee 22 pines de entrada/salida – 8 salidas digitales, 8 entradas, 4 analógicas y 2 pines de interfase en serie.

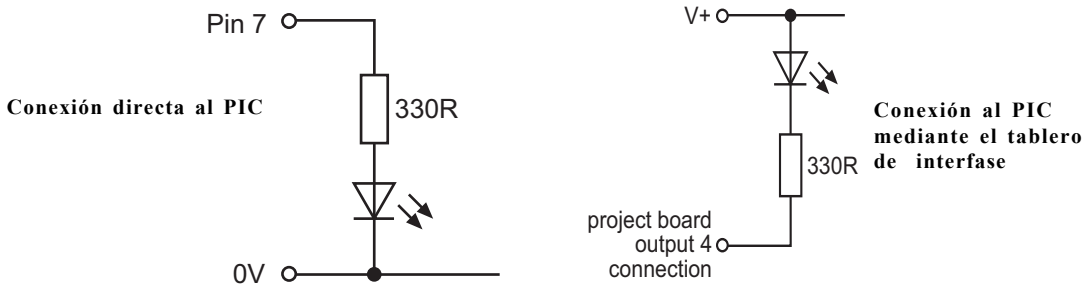
- *bajo costo, circuito de fácil construcción*
- *8 salidas y 5 entradas y 4analógicas*
- *rápida operación de descarga mediante el cable serie*
- *Software "Editor de Programación" gratuito y de fácil uso*
- *lenguaje BASIC simple y fácil de aprender*
- *manuales gratuitos y foro de apoyo en línea*
- *también puede programarse utilizando Organigramas*

El paquete de inicio incluye los siguientes elementos:

- *tablero estándar de interfase*
- *cable de descarga*
- *CDROM con programas y manuales*
- *chip de microcontrolador PICAXE-28A*

## Descargando su primer programa

Este primer programa es muy sencillo y puede ser utilizado para probar su sistema. Requiere la conexión de un LED (Light Emitting Diode – Diodo Emisor de Luz) y una resistencia 330R al pin de salida 7. Si está conectando el LED directamente a un chip PICAXE en un tablero hecho en casa, conecte el LED entre el pin de salida y 0 V (tierra). Si en cambio está utilizando el tablero de interfase (suministrado dentro del paquete de inicio), conecte el LED entre V+ y el pin, ya que la salida es controlada por el chip controlador darlington en el tablero de interfase. (¡Asegúrese que el LED esté conectado correctamente!).




**Nota:** El microcontrolador PICAXE puede conectarse a un ordenador mediante el tablero de proyectos de 28 pines ó mediante un simple circuito hecho en casa. Las instrucciones para la conexión de circuitos se encuentran detalladas más adelante en este libreto.

1. Conecte el cable PICAXE a un puerto serie del ordenador y tome nota a cual de los puertos lo conecta (normalmente COM1 ó COM2).
2. Ejecute el Software "Editor de Programación".
3. En el menú desplegable escoja Ver>Opciones para acceder la pantalla de opciones (esta puede que aparezca automáticamente).
4. Haga clic en "Modo" y seleccione PICAXE-28A.
5. Haga clic en "Puerto Serie" y seleccione el puerto serie al cual el cable PICAXE está conectado.
6. Haga clic en "OK".
7. Escriba el siguiente programa:

```
main:  high 7
       pause 1000
       low 7
       pause 1000
       goto main
```

(No olvide el símbolo de dos puntos (:) directamente después de la etiqueta "main" y los espacios entre los comandos y los números)

8. Asegúrese que el circuito PICAXE esté conectado al cable serie y que las baterías (4.5V) estén conectadas. Asegúrese también que el LED y la resistencia de 330R estén conectados a la salida 7.
9. Seleccione PICAXE>Ejecutar. Una barra de progreso deberá aparecer mientras el programa es descargado. Al terminar la descarga, el programa debe comenzar a ejecutarse automáticamente – el LED de la salida 7 deberá encenderse y apagarse cada segundo.

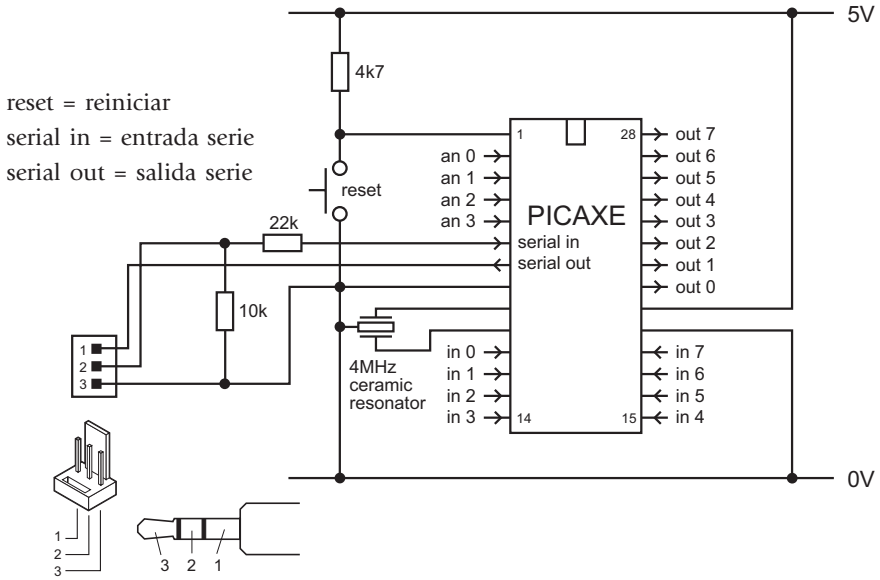
## Revisión de Fallos

Si le aparece un mensaje de error, lea la descripción del mismo. Algunos de los errores más comunes son:

- *No conectar el cable de descarga al circuito PICAXE o al puerto serie del ordenador.*
- *No seleccionar el puerto serie correcto en el software Editor de Programación.*
- *No conectar la batería o utilizar una batería descargada.*

## El Circuito PICAXE-28

La siguiente figura muestra el circuito PICAXE-28 básico




La resistencia de 4k7 se utiliza para activar (high) el pin de reinicio (pin 1) del microcontrolador PICAXE. Si se desea, también se puede conectar un interruptor de reinicio entre el pin de reinicio (pin 1) y 0V. Cuando se presiona este interruptor el microcontrolador PICAXE se "reinicia" a la primera línea del programa.

## El Circuito PICAXE de Interfase del Ordenador

El sistema PICAXE utiliza una muy simple interfase al puerto serie del ordenador. Aunque esta interfase no utiliza verdaderos voltajes RS232, es de muy bajo costo y ha tenido un desempeño confiable en casi todos los ordenadores modernos.

Se recomienda incluir este circuito de interfase en todo PCB diseñado para ser utilizado con el microcontrolador PICAXE. Esto permite que el microcontrolador PICAXE sea reprogramado sin necesidad de removerlo del PCB.

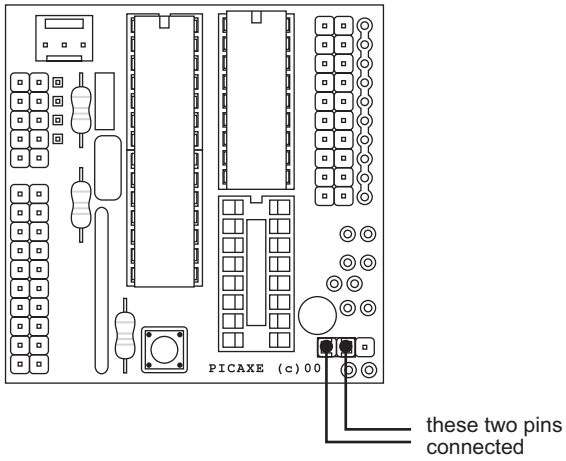
**Nota:**
La mayor parte de los ordenadores modernos tienen dos puertos serie, normalmente denominados COM1 y COM2. El software "Editor de Programación debe ser configurado con el puerto al cual el microcontrolador está conectado – en el menú desplegable seleccione **Ver>Opciones>Puerto Serie** para elegir el puerto serie correspondiente en su ordenador.

Si utiliza un ordenador que posee el antiguo conector de puerto USB, utilice un adaptador USB <> serial (USB010).

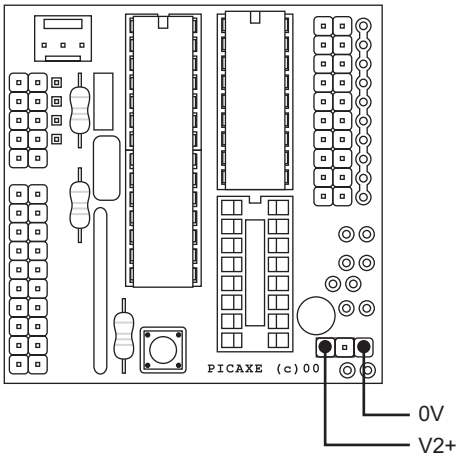
## Las Interfases del PICAXE-28A

**Fuente de Alimentación de los Tableros de Proyectos**

Los tableros de proyectos requieren para operar una fuente de alimentación de 3-5V. Se recomienda que esta potencia sea suministrada mediante paquetes de baterías AA (3xAA = 4.5V) conectados a los terminales. Estos paquetes se encargaran de alimentar tanto al microcontrolador como a los dispositivos de salida.

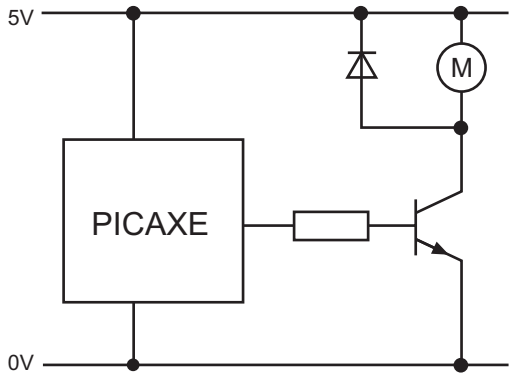



Si se requiere un voltaje mayor (Por ejemplo 12V) para alimentar a las salidas, se pueden utilizar dos fuentes de alimentación separadas. En este caso la segunda fuente de alimentación sólo alimenta a los dispositivos de salida. La fuente de 3-5V se conecta a V1+ y la fuente de 12V a V2+. Al utilizar dos fuentes de alimentación, la resistencia mostrada en la siguiente figura debe sacarse del tablero para separar a ambas fuentes.

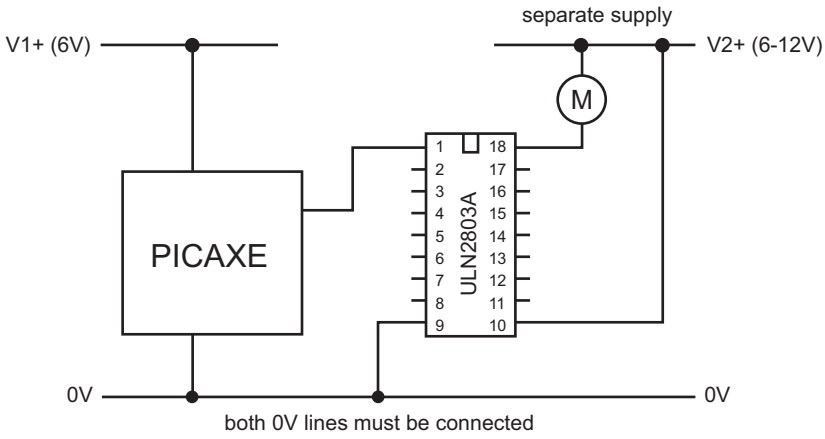


## Salidas Digitales

Cada salida digital se conecta a un transistor como se indica a continuación.

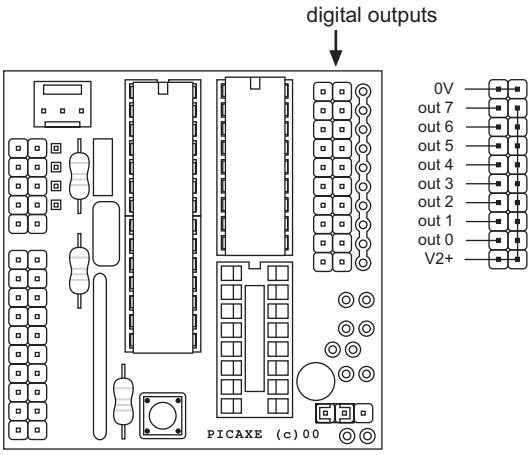



En el tablero, todos los transistores están contenidos en un chip controlador darlington ULN2803A. El circuito equivalente de salida se muestra en la figura abajo.



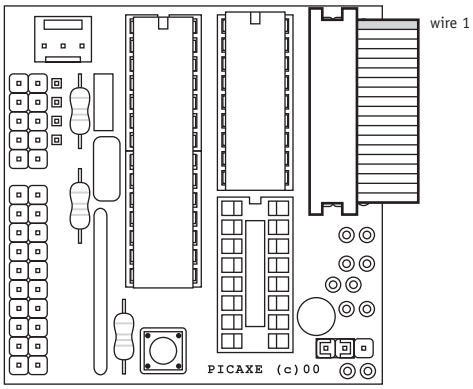

"both 0V lines must be connected" = ambas líneas 0V deben estar conectadas

Salidas Digitales ("digital outputs")




Wire 1 = rojo

| | |
|---|---|
| 1 | 0V |
| 2 | 0V |
| 3 | 7 |
| 4 | V2+ |
| 5 | 6 |
| 6 | V2+ |
| 7 | 5 |
| 8 | V2+ |
| 9 | 4 |
| 10 | V2+ |
| 11 | 3 |
| 12 | V2+ |
| 13 | 2 |
| 14 | V2+ |
| 15 | 1 |
| 16 | V2+ |
| 17 | 0 |
| 18 | V2+ |
| 19 | V2+ |
| 20 | V2+ |

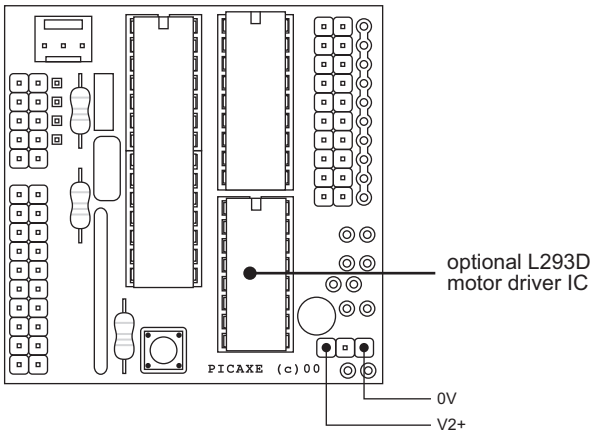


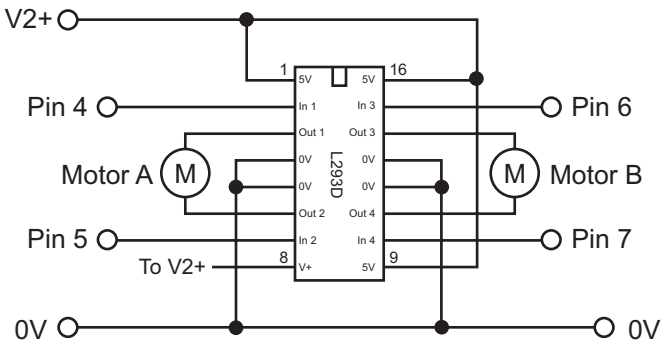
## Controlador de Motor

Opcionalmente se puede agregar un chip controlador de motor L293D (no suministrado) a la interfase de alta potencia. Esto permite un control avance-reversa de dos motores CC, controlados por las salidas 4 y 7. Naturalmente, si sólo se necesita controlar un motor, sólo se utilizaran dos líneas de salida.

| **Pin 4** | **Pin 5** | **Motor A** |
|---|---|---|
| apagado | apagado | apagado |
| encendido | apagado | avance |
| apagado | encendido | reversa |
| encendido | encendido | apagado |

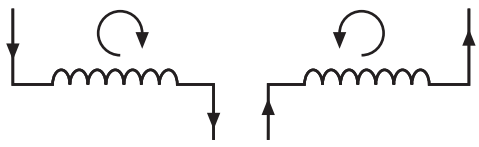
| **Pin 6** | **Pin 7** | **Motor A** |
|---|---|---|
| apagado | apagado | apagado |
| encendido | apagado | avance |
| apagado | encendido | reversa |
| encendido | encendido | apagado |




| | |
|---|---|
| Ambas entradas en *low* | -motor detenido |
| Primera salida en *high*, segunda salida en *low* | -motor en marcha |
| Primera salida en *low*, segunda salida en high | -motor en reversa |
| Ambas entradas en *high* | -motor detenido |

El cambio en el estado de los pines altera la dirección de la corriente a través del motor como se muestra abajo.

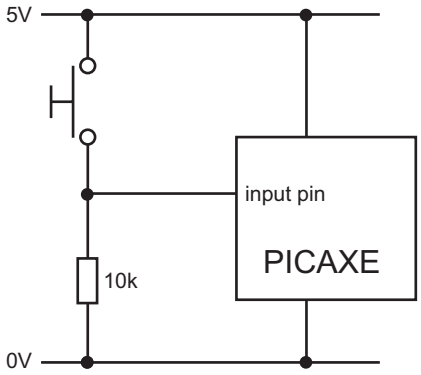


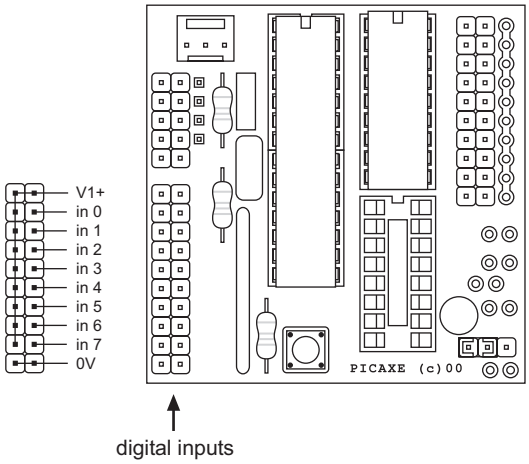
Note que el chip L293D se calienta al ser utilizado continuamente. Para ayudarlo a refrescarse se le puede pegar encima un disipador de calor.

Los motores A y B se conectan a la interfase mediante los terminales incluidos. Es necesario soldar conectores o alambres sobre estos terminales.

## Las entradas digitales

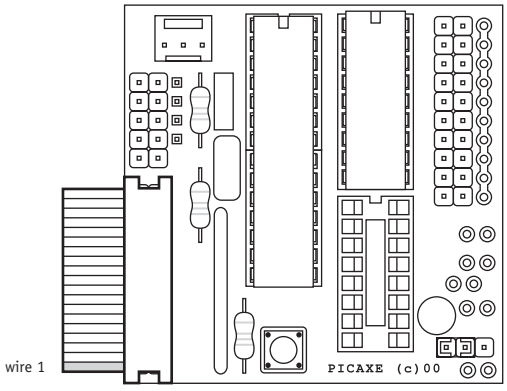
Las entradas digitales ("digital inputs") se conectan entre V1+ y el pin como se muestra abajo



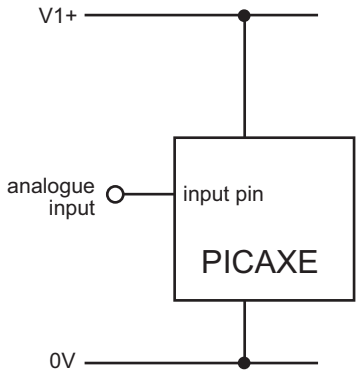

Wire 1 = rojo

| | |
|---|---|
| 20 | V1+ |
| 19 | V1+ |
| 18 | V1+ |
| 17 | 0 |
| 16 | V1+ |
| 15 | 1 |
| 14 | V1+ |
| 13 | 2 |
| 12 | V1+ |
| 11 | 3 |
| 10 | V1+ |
| 9 | 4 |
| 8 | V1+ |
| 7 | 5 |
| 6 | V1+ |
| 5 | 6 |
| 4 | V1+ |
| 3 | 7 |
| 2 | 0V |
| 1 | 0V |



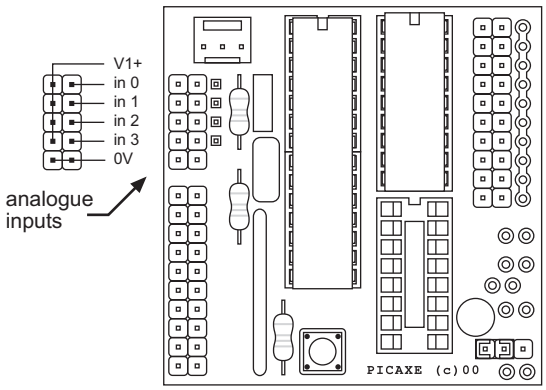
## Canales de Entradas Analógicas

Los canales de entradas analógicas son flotantes (sin conexión propia).
El circuito equivalente de entrada se muestra aquí.




Wire 1 = rojo

| | |
|---|---|
| 10 | V1+ |
| 9 | 0 |
| 8 | V1+ |
| 7 | 1 |
| 6 | V1+ |
| 5 | 2 |
| 4 | V1+ |
| 3 | 3 |
| 2 | 0V |
| 1 | 0V |



## Principios de Programación BASIC

Los siguientes programas se incluyen para dar una breve introducción acerca de algunas de las principales técnicas de programación. Todos los programas pueden probarse conectando un LED (con una resistencia de 330R) a las salidas 6 y 7, un interruptor a la entrada 0, y una resistencia variable al canal analógico 0.

Para mayor información acerca de cada programa vea los archivos de ayuda Comandos BASIC y Tutorial PICAXE-18, dentro del software Editor de Programación.

### Encendiendo y apagando salidas

El siguiente programa enciende y apaga la salida 7 cada segundo. El programa demuestra como utilizar los comandos *high*, *low*, *wait*, *pause* y *goto*.
Al bajar este programa, el LED rojo conectado al pin de salida 7 deberá encenderse y apagarse continuamente.

```
main:                   ' hacer una etiqueta llamada "main"
      high 7            ' encender salida 7
      wait 1            ' esperar  1 segundo
      low 7             ' apagar salida 7
      pause 1000        ' esperar 1000 ms (= 1 segundo)
      goto main         ' regresar el inicio
```

La primera línea sólo crea una etiqueta llamada "main" en el programa. Las etiquetas son utilizadas para indicar posiciones dentro del programa. En este programa la última línea utiliza el comando "*goto main*" para saltar a la primera línea. Esto crea un bucle continuo.

Las etiquetas pueden ser cualquier palabra (con la excepción de palabras claves como por ejemplo "high" ) pero deben empezar con una letra. Cuando la etiqueta es definida por primera vez debe llevar al final el símbolo de dos puntos (:). Esto indica al ordenador que la palabra es una nueva etiqueta.

Se acostumbra dejar espacios (o una tabulación) al inicio de cada línea, aparte de las líneas en donde se definen etiquetas. Esto hace que el programa sea más fácil de leer y comprender. Opcionalmente, para facilitar la comprensión de la operación del programa también se pueden agregar comentarios en las líneas después de un apóstrofe (').

Note que tanto el comando *pause* como el comando *delay* crean tiempos de retardo. Sin embargo, el comando *wait* puede ser utilizado únicamente con segundos enteros mientras que *pause* se puede utilizar para retardos más cortos (el mismo se asigna en milésimas de segundo).

**Detectando Entradas**

El siguiente programa hace parpadear al pin de salida 7 cada vez que un interruptor en el pin de entrada 0 es presionado.

```
main:                          'hacer una etiqueta llamada "main"
      if pin0 = 1 then flash   ' saltar a flash si la entrada
                               ' está encendida
      goto main                ' sino  regresar a inicio

flash:                         ' hacer una etiqueta llamada "flash"
      high 7                   ' encender  salida
      pause 500                ' esperar 0.5 segundos
      low 7                    ' apagar salida 7
      goto main                ' regresar a inicio
```

En este programa las tres primeras líneas forman un bucle continuo. Si la entrada está apagada, el programa regresará al inicio y se ejecutará una y otra vez.

Si el interruptor es presionado, el programa saltará a la etiqueta "flash". El programa luego encenderá la salida 7 por 0.5 segundos antes de regresar el bucle principal.

Note cuidadosamente la ortografía en la línea del comando *if…then* – pin0 es una sola palabra (sin espacios en blanco). Note también que únicamente se debe escribir el nombre de la etiqueta posterior al comando *then* – no se permite ninguna otra palabra aparte de la etiqueta.

**Utilizando el comando *Symbol***

Algunas veces es difícil recordar cuales pines están conectados a cuales dispositivos. El comando *symbol* puede en estos casos ser utilizado al inicio del programa para renombrar a entradas y salidas.

```
symbol rojo = 7                'renombrar salida7 (output4) rojo
symbol verde = 5               'renombrar salida5 (output0) verde

main:                    'hacer una etiqueta llamada "main"
      high rojo          'LED rojo encendido
      wait 2             'esperar 2 segundos
      low rojo           'LED rojo  apagado
      high verde         'LED verde encendido
      wait 1             'esperar 1 segundo
      low verde          'LED verde apagado
      goto main          'regresar al inicio ("main")
```

**Bucles For...Next**

Con frecuencia es útil repetir una parte de un programa varias veces, por ejemplo al encender/apagar sucesivamente un diodo LED. En estos casos se puede utilizar un bucle *for…next*.

```
symbol rojo = 7           'renombrar salida7 como "red" (rojo)
symbol contador = b0      'definir un counter (contador) utilizando la
                          'variable b0

main:                               ' hacer una etiqueta llamada "main"
      for contador = 1 to 25        'iniciar un  bucle for…next
            high rojo               'encender LED rojo
            pause 500               'esperar 0.5 segundos
            low rojo                'apagar LED rojo
            pause 500               'esperar 0.5 segundos
      next contador                 'siguiente b0

      end                           'fin del programa
```

En este programa, toda el código entre los comandos *for* y *next* se repite 25 veces. El número de veces que el código debe ser repetido es almacenado en la variable llamada "counter", la cual en este programa es un símbolo (comando *symbol*) para la variable "b0". El PICAXE tiene 14 variables disponibles, b0 a b13, que pueden utilizarse de esta manera. Una variable es una ubicación en la memoria en donde se pueden almacenar números.

**Utilizando variables**

Las variables se utilizan frecuentemente para almacenar "números" a medida que el programa se ejecuta. Este programa enciende todas las salidas en diferentes combinaciones.

```
main:                       ' hacer una etiqueta llamada "main"
      let b0 = b0 + 1       'sumar un 1 a b0
      let pins = b0         'asignar el número de  pin especificado por b0
      pause 10      0       'esperar 0.1 segundos
      goto main             'regresar al inicio
```

Note que b0 es una variable byte. Esto significa que tolera cualquier número entre 0 y 255. Por consiguiente, el programa de arriba eventualmente "desbordará" en el número más alto y se comportará de la siguiente manera: …253-254-255-0-1-2… Esto es un dato importante de recordar al realizar operaciones matemáticas con variables.

Para mayor información acerca de las capacidades matemáticas del microcontrolador PICAXE vea la sección de Comandos.

### Reading Analogue Input Channels

### Leyendo Canales de Entradas Analógicas

El valor de una entrada analógica puede copiarse fácilmente dentro de una variable utilizando el comando *readadc*. El valor de la variable (valor entre 0 y 255) puede luego ser probado. El siguiente programa enciende un diodo LED si el valor es mayor que 150 y otro diodo LED si el valor es menor de 100. Si el valor está entre 100 y 150, ambos diodos LED permanecen apagados.

```
main:                          'hacer etiqueta llamada "main"
      readadc 0,b0             'leer señal de canal 0 en variable b0
      if b0 > 150 then rojo1   'si b0 >150 ir a "rojo1"
      if b0 < 100 then verde1  'si b0 <100 ir a "verde1"
      low 7                    'sino apagar 7
      low 6                    ' y apagar 6
      goto main                'regresar a inicio

rojo1:                    'hacer etiqueta llamada "rojo1"
      high 7              'encender 7
      low 6               'apagar 6
      goto main           'regresar a inicio

verde1:                   'hacer etiqueta llamada "verde1"
      high 6              'encender 6
      low 7               'apagar 7
      goto main           'regresar a inicio
```

Note que el microcontrolador PICAXE-28A tiene 4 canales analógicos nombrados de 0 a 3

Frecuentemente al utilizar sensores analógicos es necesario calcular el valor de "umbral" necesario para el programa (esto es, los valores 100 y 150 en el programa anterior). El comando *debug* permite ver fácilmente el valor en "tiempo real" de un sensor permitiendo calcular el valor umbral experimentalmente.

```
main:                     'hacer etiqueta llamada "main"
      readadc 0,b0        'leer señal de canal 0 en variable b0
      debug b0            'transmitir valor a la pantalla del ordenador
      pause 100           'pausa corta
      goto main           'regresar a inicio
```

Al ejecutar este programa aparecerá una ventana de debug (depuración) en la pantalla del ordenador indicando el valor de la variable b0. A medida que el sensor es probado con la variable, la ventana va indicando la lectura actual del sensor.

## Sub-Procedimientos

Un sub-procedimiento es un "mini-programa" separado el cual puede ser llamado desde el programa principal. Una vez que el sub-procedimiento ha sido ejecutado, el programa principal continua.

Los sub-procedimientos son frecuentemente utilizados para separar el programa principal en pequeñas secciones para hacerlo más fácil de comprender. Sub-procedimientos que realizan tareas comunes pueden también ser copiados de programa a programa para ahorrar tiempo.

El siguiente programa utiliza dos sub-procedimientos para separar las dos secciones principales del programa ("flash1" y "flash2").

```
symbol rojo = 7                 ' renombrar salida7 rojo
symbol verde = 6                ' renombrar  salida6 verde
symbol  counter = b0            ' definir a la variable "counter" como b0

main:                       'hacer etiqueta llamada "main"
      gosub flash1          'ir al sub-procedimiento "flash1"
      gosub flash2          'ir al sub-procedimiento "flash2"
      goto main             'ir a  "main"

      end                   'fin del programa principal

flash1:                             'hacer un sub-procedimiento llamado "flash1"
      for counter = 1 to 25         'iniciar un bucle for…next
            high  rojo              'encender diodo LED
            pause 50                'esperar 0.05 segundos
            low rojo                'apagar diodo LED
            pause 50                'esperar 0.05 segundos
      next counter          'siguiente counter (b0)
      return                'retornar del sub-procedimiento

flash2:                             'hacer un sub-procedimiento llamado "flash2"
      high verde            'encender LED
      pause 2000            'esperar 2 segundos
      low verde             'apagar LED
      return                'retornar del sub-procedimiento
```

## Los Comandos PICAXE-28A

La siguiente lista es un resumen completo de todos los comandos utilizables con el sistema PICAXE-28. Para detalles de sintaxis y ejemplos de programas vea el archivo de ayuda "Comandos BASIC".

**SALIDA DIGITAL**

| | |
|---|---|
| HIGH | Conmutar un pin de salida en *high* (encendido). |
| LOW | Conmutar un pin de salida en *low* (apagado). |
| TOGGLE | Bascular el estado de un pin de salida. |
| PULSOUT | Generar un impulso en un pin por un tiempo determinado. |

**SALIDA ANALÓGICA**

| | |
|---|---|
| PWM | Proveer un impulso de salida PWM. |
| SOUND | Emitir un sonido. |

**ENTRADA DIGITAL**

| | |
|---|---|
| IF…THEN | Saltar a otra línea del programa si se cumple una determinada condición en la entrada |
| PULSIN | Medir la longitud de un impulso en un pin de entrada. |

**ENTRADA ANALÓGICA**

| | |
|---|---|
| READADC | Leer señal de un canal analógico dentro de una variable. |
| READTEMP | Leer valor de temperatura dentro de una variable. |

**FLUJO DEL PROGRAMA**

| | |
|---|---|
| FOR…NEXT | Crear un bucle for…next. |
| BRANCH | Saltar a dirección especificada por posición indicada. |
| GO TO | Saltar a dirección indicada. |
| GOSUB | Saltar a la sub-rutina especificada. |
| RETURN | Retornar de la sub-rutina. |
| IF…THEN | Comparar y saltar condicionalmente. |

**MANIPULACIÓN DE VARIABLES**

| | |
|---|---|
| {LET} | Realizar matemáticas de variables. |
| LOOKUP | Buscar dato especificado por la posición indicada y almacenarlo en una variable. |
| LOOKDOWN | Encontrar número (0-N) concordante con el objetivo y almacenarlo en una variable. |
| RANDOM | Generar un número seudo-aleatorio. |
| PEEK | Leer desde registros RAM adicionales. |
| POKE | Escribir a registros RAM adicionales. |

**ENTRADA/SALIDA SERIE**

| | |
|---|---|
| SEROUT | Sacar datos en serie del pin de salida. Hasta 2400 baudios. |
| SERIN | Introducir datos en serie al pin de entrada. Hasta 2400 baudios. |

**ACCESO DEL EEPROM INTERNO**

| | |
|---|---|
| EEPROM | Almacenar datos en EEPROM de datos antes de descargar el programa BASIC |
| READ | Leer datos de la ubicación EEPROM especificada dentro de la variable. |
| WRITE | Escribir variable dentro de ubicación EEPROM especificada. |

**REDUCCIÓN DE POTENCIA**

| | |
|---|---|
| NAP | Introducir modo de baja potencia por períodos cortos (hasta 2.3 segundos). |
| SLEEP | Introducir modo de baja potencia por un período (hasta 65535 segundos). |
| END | Apagar y mantenerse apagado hasta reinicio. |

**MISCELANEOS**

| | |
|---|---|
| PAUSE | Esperar hasta 65535 segundos. |
| WAIT | Esperar hasta 65 segundos. |
| SYMBOL | Renombrar variables. |
| DEBUG | Transmitir variables al ordenador para depurarlas. |