

Ministerio del Poder Popular para la Educación Universitaria
Universidad de Carabobo
Facultad de Ciencia y Tecnología

Microcontroladores

Por: Gabriel Isaac Rodríguez Carvallo
C.I. 30172571
Asignación: Arquitectura del Computador
Carrera: Computación

Microcontroladores: Un microcontrolador es un circuito integrado digital que puede ser usado para muy diversos propósitos debido a que es *programable*. Está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos)¹. Este tiene los mismos bloques de funcionamiento básico de una computadora lo que nos permite tratarlo como un pequeño dispositivo de cómputo.

Pueden usarse para muchas aplicaciones como manejo de sensores, controladores, juegos, agendas, avisos lumínicos, relojes, alarmas, robots, entre otros.

Para el diseño de programas es necesario conocer los bloques funcionales básicos del microcontrolador, estos bloques son:

- CPU (Unidad central de procesamiento)
- Memoria ROM (Memoria de solo lectura)
- Memoria RAM (Memoria de acceso aleatorio)
- Líneas de entrada y salida (Periféricos)

Para grabar un programa en un microcontrolador se requieren básicamente tres cosas:

1. Una computadora
2. Un software de programación (con un compilador)
3. Un circuito programador

Familia Intel 8031: Intel 8031 y 80C31 son miembros de la familia Intel MCS-51 de microcontroladores de 8 bits. 8031/80C31 tienen los mismos periféricos integrados que los MCU 8051: 4 puertos de E/S, dos temporizadores/contadores de 16 bits, un oscilador en chip y un puerto serie. Las MCU tienen 128 bytes de RAM interna y, además, pueden utilizar hasta 64 KB de memoria de datos externa. Los microcontroladores no tienen ROM en el chip y deben utilizar una memoria de programa externa.²

Intel 80C31, versión CMOS del microcontrolador NMOS 8031, es totalmente compatible con código objeto y pin con el 8031.

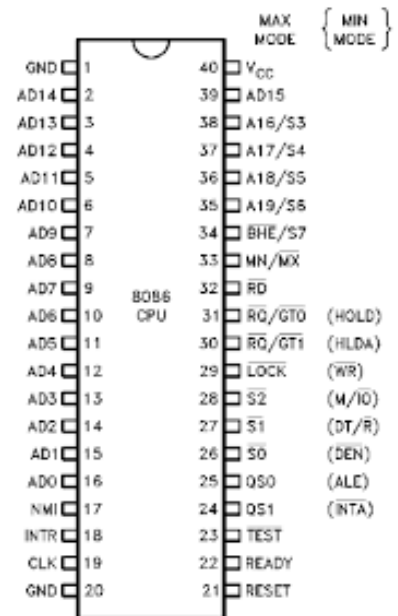
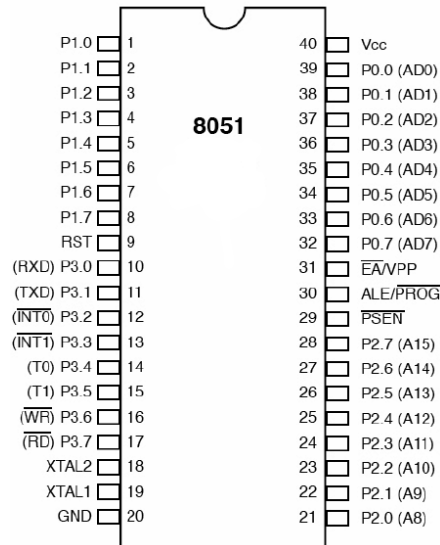
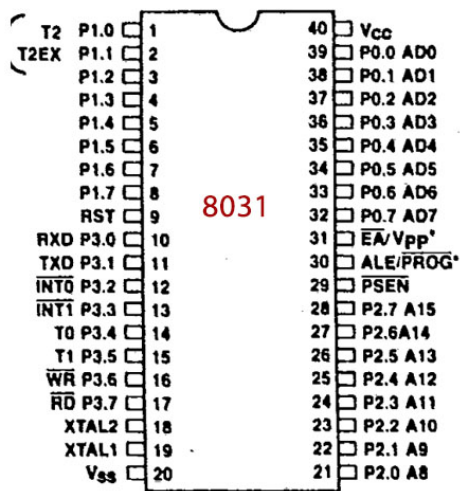
Familia Intel 8051: Intel 8051 y 80C51 son miembros de la familia Intel MCS-51 de microcontroladores de 8 bits. Además del oscilador de reloj en chip, 2 temporizadores de 16 bits, 4 puertos de E/S y un puerto serie, todos los chips 8051 y 80C51 tienen 128 bytes de RAM y 4 KB de ROM enmascarada. Si es necesario, la memoria del 8051 se puede ampliar hasta 128 KB agregando hasta 64 KB de memoria externa de programas y datos.³

Convenciones de nombre generales de la familia Intel MCS-51: 8051 es el nombre original de Intel con 4 KB de ROM y 128 bytes de RAM. Las variantes que comienzan con 87 tienen una EPROM programable por el usuario, a veces borrable por UV. Las variantes con una C como tercer carácter son una especie de CMOS (Que es una pequeña cantidad de memoria que almacena la configuración del BIOS). 8031 y 8032 son versiones sin ROM, con 128 y 256 bytes de RAM. El último dígito puede indicar el tamaño de la memoria, p. 8052 con 8 KB de ROM, 87C54 con 16 KB de EPROM y 87C58 con 32 KB de EPROM, todos con 256 bytes de RAM¹¹.

Familia Intel 8086: El microprocesador Intel 8086 es el primer miembro de la familia de procesadores x86. Anunciado como "código fuente compatible" con los procesadores Intel 8080 e Intel 8085, el 8086 no era código objeto compatible con ellos. El 8086 tiene una arquitectura completa de 16 bits: registros internos de 16 bits, bus de datos de 16 bits y bus de direcciones de 20 bits (1 MB de

memoria física). Debido a que el procesador tiene registros de índice de 16 bits y punteros de memoria, sólo puede direccionar de manera efectiva 64 KB de memoria. Para direccionar la memoria más allá de 64 KB, la CPU utiliza registros de segmento: estos registros especifican ubicaciones de memoria para segmentos de código, pila, datos y datos adicionales de 64 KB. Los segmentos se pueden ubicar en cualquier lugar de la memoria y, si es necesario, los programas de usuario pueden cambiar su posición. Este método de direccionamiento tiene una gran ventaja: es muy fácil escribir código independiente de la memoria cuando el tamaño del código, la pila y los datos es inferior a 64 KB cada uno⁴.

8031 Pinout



Microcontroladores	Intel 8031	Intel 80C31	Intel 8032	Intel 8051	Intel 80C51	Intel 8052
Memoria RAM Interna	128 bytes	128 bytes	256 bytes	128 bytes	128 bytes	256 bytes
Memoria ROM Interna	No tiene	No tiene	No tiene	4 kB	4 kB	8 kB
CMOS	No	Si	No	No	Si	No
Temporizadores	2 de 16 bits	2 de 16 bits	3 de 16 bits	2 de 16 bits	2 de 16 bits	3 de 16 bits

Información del Intel 8051:

Este microcontrolador se puede adquirir en mercadolibre de Venezuela(https://articulo.mercadolibre.com.ve/MLV-518011272-p8031ah-8031-8-bits-microcontrolador-40-pin-8051-intel-sieme-_JM) por 10\$.

Puertos: 32 de las 40 terminales del 8051 funcionan como líneas para puertos de E/S(Entrada y salida). Entre estas, 24 líneas son de propósito dual (26 en el 8032/8052). Cada una de estas líneas puede operar como E/S o como línea de control, o como una parte del bus de direcciones o del bus de datos. Los diseños que requieren de muy poca memoria externa o de otros componentes externos utilizan estos puertos como E/S de propósito general. Las ocho líneas ubicadas en cada puerto se pueden tratar como una sola unidad de interfaz para dispositivos paralelos tales como impresoras,

convertidores digitales-analógicos, entre otros. El puerto serial puede recibir y transmitir datos al mismo tiempo.

- **Puerto 0:** El puerto 0 es un puerto de propósito dual en las terminales 32-39 del circuito integrado 8051. Este se usa como un puerto de E/S de propósito general en diseños que necesitan un mínimo de componentes. Este puerto se puede convertir en un bus de direcciones y datos multiplexados en diseños más complejos que requieran de memoria externa.
- **Puerto 1:** El puerto 1 es un puerto dedicado de E/S en las terminales 1-8. Las terminales, designadas como P1.0, P1.1, P1.2, etc., están disponibles para utilizarse como interfaces para dispositivos externos, en caso de necesitarse. Ninguna de las terminales del puerto 1 tiene otra función asignada, por lo tanto, sólo se utilizan como interfaces para dispositivos externos. Los circuitos integrados 8032/8052 son la excepción, pues utilizan las terminales P1.0 o P1.1 ya sea como líneas de E/S o como entradas externas del tercer temporizador.
- **Puerto 2:** El puerto 2 (terminales 21-28) es un puerto de propósito dual que sirve como E/S de propósito general, o como el byte superior del bus de direcciones en diseños que utilizan memoria externa para código o más de 256 bytes de memoria externa para datos.
- **Puerto 3:** El puerto 3 es un puerto de propósito dual en las terminales 10-17. Se puede utilizar como E/S de propósito general, pero también cumple múltiples funciones ya que sus terminales tienen un propósito alterno relacionado con las características especiales del 8051.

Estructura de los puertos: Si escribimos a la terminal de un puerto se cargan datos en un latch del puerto, el cual controla un transistor de efecto de campo conectado a la terminal. Los puertos 1, 2 y 3 tienen capacidad para controlar cuatro unidades de carga del tipo Schottky TTL de bajo consumo de energía, y el puerto 0, 8 unidades del mismo tipo. Observe que la resistencia de jalón hacia Vcc (pull-up) no está presente en el puerto 0, a menos que se requiera que éste funcione como bus de direcciones/datos externo. Quizas se requiera una resistencia externa de jalón hacia Vcc (pull-up), dependiendo de las características de entrada del dispositivo controlado por la terminal.

Existe capacidad tanto para “leer el latch” como para “leer la terminal”. Las instrucciones que requieren de una operación de lectura-modificación-escritura (CPL P1.5, por ejemplo) leen el latch para evitar que el nivel de voltaje se malinterprete en el caso de que la terminal tenga una carga pesada (por ejemplo, cuando se controla la terminal de base de un transistor). Las instrucciones que envían datos de entrada a un bit de puerto (MOV C,P1.5, por ejemplo) leen la terminal. El latch del puerto debe contener el valor 1, en este caso, de otra manera el controlador del FET está ENCENDIDO y lleva a la salida al nivel bajo. El reinicio de un sistema establece a 1 todos los latches de los puertos, así que se pueden utilizar las terminales de los puertos como entradas sin tener que ponerlos a 1 de manera explícita.

Temporizadores: Un temporizador se compone de una serie de flip-flops de división entre 2, los cuales reciben una señal de entrada como su fuente de reloj. El reloj se aplica al primer flip-flop, el cual divide la frecuencia del reloj entre 2. La salida del primer flip-flop se aplica a la entrada de reloj del segundo flip-flop, que también divide la frecuencia entre 2, y así sucesivamente. Un temporizador con un número n de etapas divide la frecuencia del reloj de entrada entre 2^n , ya que cada etapa sucesiva divide entre 2. La salida de la última etapa se aplica a la entrada de reloj de un flip-flop de desbordamiento del temporizador, o bandera, cuyo estado puede ser verificado mediante el software y su establecimiento en 1 puede generar una interrupción. Podemos considerar al valor binario de las salidas de los flip-flops del temporizador como la “cuenta” del número de pulsos del reloj (o “eventos”) desde que el temporizador se inició. Por ejemplo, un temporizador de 16 bits contaría desde 0000H hasta FFFFH. La bandera de desbordamiento se establecería en 1 en el desbordamiento de FFFFH a 0000H del conteo.

Los temporizadores se utilizan en casi todas las aplicaciones orientadas al control, y los del 8051 no son la excepción. El 8051 tiene dos temporizadores de 16 bits, cada uno con cuatro modos de operación. Un tercer temporizador de 16 bits con tres modos de operación se ha añadido al 8052. Los temporizadores se utilizan para la temporización de intervalos, el conteo de eventos, o la generación de la tasa de transmisión y recepción en baudios para el puerto serial incorporado. Cada temporizador es de 16 bits, por lo que la decimosexta o última etapa divide la frecuencia del reloj de entrada entre $2^{16} = 65,536$.

En las aplicaciones de temporización de intervalos, un temporizador se programa para causar un desbordamiento a intervalos regulares, lo cual establece en 1 la bandera de desbordamiento, misma que se utiliza para sincronizar el programa para que éste realice una acción tal como verificar el estado de las entradas o enviar datos a las salidas. Otras aplicaciones pueden utilizar la capacidad del temporizador de recibir pulsos a intervalos regulares para medir el tiempo transcurrido entre dos condiciones (por ejemplo, medidas de la anchura de un pulso).

El conteo de eventos se utiliza para determinar cuántas veces ocurre un evento, más que para medir el tiempo transcurrido entre varios eventos. Un “evento” es cualquier estímulo externo que provee una transición de 1 a 0 a una terminal en el circuito integrado del 8051.

En esencia los temporizadores permiten ejecutar instrucciones con retrasos especificados.

Interrupción: Se refiere la ocurrencia de una condición (o evento) que ocasiona la suspensión temporal de un programa mientras que otro programa se encarga de servir a dicha condición. Las interrupciones cumplen una función importante en el diseño y la implementación de las aplicaciones con microcontroladores. Las interrupciones permiten que un sistema pueda responder a un evento en forma asíncrona y se encargue del evento mientras se ejecuta otro programa. Un sistema controlado mediante interrupciones nos da la falsa percepción de que está realizando muchas cosas en forma simultánea. Por supuesto que la CPU no tiene la capacidad de ejecutar más de una instrucción al mismo tiempo; pero sí puede suspender temporalmente la ejecución de un programa, ejecutar otro, y después regresar al primer programa. En cierta manera, esto es como una subrutina. La CPU ejecuta otro programa (la subrutina) y después regresa al programa original. La diferencia está en que, en un sistema controlado mediante interrupciones, la interrupción es la respuesta a un “evento” que ocurre de manera asíncrona con el programa principal. En otras palabras, no sabemos cuándo se interrumpirá el programa principal. Una rutina de servicio de interrupción (ISR), también conocida como manejador de interrupciones, es el programa que se encarga de efectuar una interrupción. La ISR se ejecuta en respuesta a la interrupción y, por lo general, realiza una operación de entrada o de salida sobre un dispositivo. El programa principal suspende su ejecución temporalmente y se bifurca a la ISR cuando ocurre una interrupción; la ISR se ejecuta, realiza la operación, y termina con una instrucción de “regreso de la interrupción”; el programa principal continúa desde donde se quedó. Es algo muy común referirnos a que el programa principal se ejecuta a nivel base y que las ISR se ejecutan a nivel de interrupción. También utilizamos los conceptos de primer plano (nivel base) y segundo plano (nivel de interrupción).

Ejemplos de código del microcontrolador 8051:

Usando temporizadores:

Onda cuadrada de 10 kHz 8.2 Escriba un programa que utilice el temporizador 0 para generar una onda cuadrada de 10 kHz en P1.0. Solución

```
#include <REG51.H>                                /* Declaraciones de los SFR */
sbit bitpuerto = P1^0;                             /* Utilizamos la variable bitpuerto para referirnos a P1.0 */
main ()
{
```

```

TMOD = 2;          /* modo de autorrecarga de 8 bits */
TH0 = -50;         /* valor de recarga de -50 en TH0
TR0 = 1;           /* inicia temporizador 0 */

while (1)          /* repite infinitamente */
{
    while (TF0 != 1); /* espera un desbordamiento */
    TF0 = 0;         /* borra bandera de desbordamiento del temporizador */
    bitpuerto = !(bitpuerto); /* dispara P1.0 */
}

```

Usando puertos:

Inicialización del puerto serial 8.5 Escriba una secuencia de instrucciones para inicializar al puerto serial de tal forma que opere como un UART de 8 bits a 2400 baudios. Utilice el temporizador 1 para proporcionar el reloj para la velocidad en baudios. Solución:

```

#include <REG51.H>    /* Declaraciones de los SFR */
main ()
{
    SCON = 0x52;      /* puerto serial, modo 1 */
    TMOD = 0x20;      /* temporizador 1, modo 2 */
    TH1 = -13;        /* conteo de recarga para 2400 baudios */
    TR1 = 1;          /* inicia temporizador 1 */
}

```

PLC

El PLC (Control Lógico Programable) es un equipo comúnmente utilizado por aquellas industrias que buscan dar un salto significativo en la automatización de todos sus procesos. Estos dispositivos se encuentran inmersos en la vida de la sociedad de distintas formas y maneras.⁵

Es una computadora industrial que usa la ingeniería para la automatización de procesos y tiene como finalidad, que las máquinas desarrollen efectivamente todos los sistemas que la componen.

Se pueden adquirir en <https://industriasgsl.com/> a un precios que van desde alrededor de 3200\$ hasta 24500\$ dependiendo del modelo.

Los PLC se programan utilizando un software de aplicación en algun ordenador, este se conecta al PLC mediante cableado USB, Ethernet, RS-232, RS-485 o RS-422. El sistema de programación permite la entrada y edición de la lógica, de hecho, en algunos software, es posible ver y editar el programa en diagramas de bloques de funciones, diagramas de flujo de secuencias y de texto estructurado.

Comúnmente, el sistema brinda funciones para depurar y solucionar problemas presentados en el software del PLC. El software cargará y descargara el programa de PLC, con fines de copia de seguridad y restauración, en ciertos modelos el programa se transfiere por medio de un tablero de programación que escribe el programa en un chip extraíble.⁶

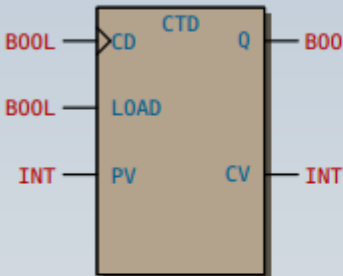
Entre los lenguajes de programación del PLC estan:

- Diagrama de escalera (LD)

- Diagrama de bloques de función (FBD)
- Texto estructurado (ST)
- Secuencial function chart (SFC)
- Lista de instrucciones (IL): también conocido como AWL

Aquí hay un ejemplo de código de PLC usando texto estructurado:

CTD– Contador descendente



CD– *count down* (flanco de subida)

LOAD– *load* (CV:=PV)

PV– *program value*

Q– *cuenta finalizada*


CV– *current value*

Código ST:

```

1 FUNCTION_BLOCK CTD
2   VAR_INPUT
3     CU: BOOL R_EDGE;
4     LOAD: BOOL;
5     PV: INT;
6   END_VAR
7   VAR_OUTPUT
8     Q : BOOL;
9     CV: INT;
10  END_VAR
11 IF LOAD THEN (* Carga del contador. *)
12   CV := PV;
13 ELSIF CU AND (CV>0) THEN
14   CV := CV-1; (* Cuenta descendente. *)
15 END_IF;
16 Q := (CV<=0); (* Cuenta finalizada. *)
17 END_FUNCTION_BLOCK

```



7

Arduino

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

Para poder entender este concepto, primero vas a tener que entender los conceptos de hardware libre y el software libre. El hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público, de manera que cualquiera puede replicarlos. Esto quiere decir que Arduino ofrece las bases para que cualquier otra persona o empresa pueda crear sus propias placas, pudiendo ser diferentes entre ellas pero igualmente funcionales al partir de la misma base.⁸

La compañía de Arduino vende placas que son formadas basicamente por microcontroladores. Estas placas están basadas en un microcontrolador del tipo ATMEL, es decir, un controlador en el que podemos grabar instrucciones para que las ejecute sin necesidad de estar introduciendo los comandos a mano una y otra vez. Para escribir estas instrucciones es necesario usar un software llamado Arduino

IDE, el Entorno de Desarrollo Integrado oficial que cuenta con todo lo necesario para poder dar forma a los programas que creamos para este dispositivo. Además se necesita de una computadora donde instalar dicho entorno.

Aunque Arduino utiliza un lenguaje de programación propio, este está basado en C++. Por lo tanto, comparte las principales ventajas de este lenguaje de programación. Además, en las versiones más recientes del IDE, es posible incluso enviarle las instrucciones directamente en C++ sin tener que traducirlas a su propio lenguaje para programar esta placa. Llegados a este punto no es difícil imaginar que tener ciertos conocimientos en este popular lenguaje de programación que nos lleva acompañando muchos años, en este caso nunca estará de más.

Además de usar C++, el Arduino IDE también soporta otros lenguajes de programación alternativos, como es el caso de C (sin las extensiones de C++), Wiring (una plataforma de prototipado electrónico formada por un lenguaje de programación, un entorno de desarrollo integrado (IDE) y un microcontrolador), así como en Processing (un lenguaje de programación basado en Java, pero enfocado a placas electrónicas).⁹

La estructura básica de programación de Arduino es bastante simple y divide la ejecución en dos partes: setup y loop. Setup() constituye la preparación del programa y loop() es la ejecución. En la función Setup() se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar el pinMode (por ejemplo si un determinado pin digital es de entrada o salida) e inicializar la comunicación serie. La función loop() incluye el código a ser ejecutado continuamente (leyendo las entradas de la placa, salidas, etc.).¹⁰

```
void setup() {  
  pinMode(pin, OUTPUT);    // Establece 'pin' como salida  
}  
void loop() {  
  digitalWrite(pin, HIGH); // Activa 'pin'  
  delay(1000);             // Pausa un segundo  
  digitalWrite(pin, LOW);  // Desactiva 'pin'  
  delay(1000);  
}
```

Como se observa en este bloque de código cada instrucción acaba con ; y los comentarios se indican con //. Al igual que en C se pueden introducir bloques de comentarios con /* ... */.

Un Arduino starter kit (kit para principiantes) se puede adquirir en la pagina oficial de arduino (<https://store-usa.arduino.cc/products/arduino-starter-kit-multi-language?selectedStore=us>) por 82.72\$.

Bibliografía:

1. <https://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/que-es-un-microcontrolador>
2. <https://www.cpu-world.com/CPU/8031/index.html>
3. <https://www.cpu-world.com/CPU/8051/index.html>
4. <https://www.cpu-world.com/CPU/8086/index.html>
5. <https://industriasgsl.com/blogs/automatizacion/que-es-un-plc-y-como-funciona>

6. <https://emacstores.com/como-programar-un-plc/>
7. <https://www.cartagena99.com/recursos/alumnos/apuntes/04-PLC-II.pdf>
8. <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>
9. <https://www.softzone.es/programas/lenguajes/programar-arduino/>
10. http://dfists.ua.es/~jpomares/arduino/page_04.htm
11. <https://en.wikipedia.org/wiki/MCS-51>