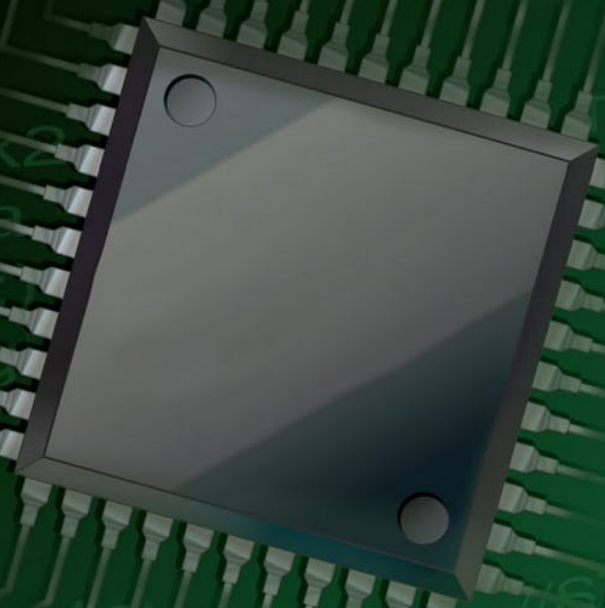
Programación de SISTEMAS EMBEBIDOS en C

Apoyo en la web

Gustavo Galeano



COLECCIÓN 3B

Alfaomega

Datos catalográficos

Galeano, Gustavo

Programación de sistemas embebidos en C, teoría y prácticas aplicadas a cualquier microcontrolador.

Primera Edición

Alfaomega Grupo Editor, S.A. de C.V., México

ISBN: 978-958-682-770-6

Formato: 21 x 24 cm Páginas: 544

Programación de sistemas embebidos en C, teoría y prácticas aplicadas a cualquier microcontrolador
Gustavo Galeano

Primera edición: Alfaomega Grupo Editor, México, junio 2009

Pitágoras 1139, Col. Del Valle, 03100, México D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-958-682-770-6

Nota importante:
La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos, han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. ALFAOMEGA GRUPO EDITOR, S.A. de C.V. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele.

Edición autorizada para venta en todo el mundo.

Impreso en México. Printed in Mexico.

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V. – Pitágoras 1139, Col. Del Valle, México, D.F. – C.P. 03100.
Tel.: (52-55) 5089-7740 – Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S.A. – Carrera 15 No. 64 A 29 – PBX (57-1) 2100122, Bogotá, Colombia,
Fax: (57-1) 6068648 – E-mail: scliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. – General del Canto 370-Providencia, Santiago, Chile
Tel.: (56-2) 235-4248 – Fax: (56-2) 235-5786 – E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. – Paraguay 1307 P.B. "11", Buenos Aires, Argentina,
C.P. 1057 – Tel.: (54-11) 4811-7183 / 8352, E-mail: ventas@alfaomegaeditor.com.ar

Dedico este trabajo a la pareja que más amo y admiro en el mundo:
mis padres, Pedro "Yaglo" y Elvia "Maclo".

Gustavo Adolfo "Poco"

AGRADECIMIENTOS

A todos los ingenieros de Latinoamérica con los cuales he tenido el agrado de trabajar; ellos con sus solicitudes, proyectos y aportes, contribuyeron a que este libro tuviera gran cantidad de experiencias propias, que sin duda, le ahorrarán tiempo y esfuerzo a los nuevos diseñadores.

A los profesores de las instituciones educativas por tomarse el tiempo de examinar el contenido, proponer ejemplos prácticos y plantear nuevos temas; con su ayuda el texto llenará los requisitos de los pénsum actuales y las expectativas de los estudiantes.

A la empresa Freescale Semiconductors, en especial a Gloria Ruiz, Rogelio Muñoz, Marco Fernández y Armando Molano quienes, desde su sector, han contribuido con su valiosa información y disponibilidad para ayudarnos.

Al ingeniero Julián Camargo por el tiempo y las valiosas sugerencias en la corrección técnica del texto, a todas las personas del grupo editorial Alfaomega, a los diseñadores gráficos, los correctores, los ingenieros del material en la web, los encargados de mercadeo, y en especial a Luis Javier Buitrago, quien lideró el proyecto desde su concepción.

A toda mi familia, amigos y seres queridos, quienes siguieron paso a paso la evolución del proyecto y me dieron ánimo para terminarlo.

A todas las personas que pusieron la confianza en las ideas y soluciones aquí plasmadas, para dejar por escrito un legado a las futuras generaciones de programadores latinoamericanos.

MENSAJE DEL EDITOR

Los conocimientos son esenciales en el desempeño profesional. Sin ellos es imposible lograr las habilidades para competir laboralmente. La universidad o las instituciones de formación para el trabajo ofrecen la oportunidad de adquirir conocimientos que serán aprovechados más adelante en beneficio propio y de la sociedad. El avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos. Cuando se toma la decisión de embarcarse en una vida profesional, se adquiere un compromiso de por vida: mantenerse al día en los conocimientos del área u oficio que se ha decidido desempeñar.

Alfaomega tiene por misión ofrecerlos a estudiantes y profesionales conocimientos actualizados dentro de lineamientos pedagógicos que faciliten su utilización y permitan desarrollar las competencias requeridas para una profesión determinada. Alfaomega espera ser su compañera profesional en este viaje de por vida por el mundo del conocimiento.

Alfaomega hace uso de los medios impresos tradicionales en combinación con las tecnologías de la información y las comunicaciones (IT) para facilitar el aprendizaje. Libros como éste tienen su complemento en una página Web, en donde el alumno y su profesor encontrarán materiales adicionales, información actualizada, pruebas (*test*) de autoevaluación, diapositivas y vínculos con otros sitios Web relacionados.

Esta obra contiene numerosos gráficos, cuadros y otros recursos para despertar el interés del estudiante, y facilitarle la comprensión y apropiación del conocimiento.

Cada capítulo se desarrolla con argumentos presentados en forma sencilla y estructurada claramente hacia los objetivos y metas propuestas. Cada capítulo concluye con diversas actividades pedagógicas para asegurar la asimilación del conocimiento y su extensión y actualización futuras.

Los libros de Alfaomega están diseñados para ser utilizados dentro de los procesos de enseñanza-aprendizaje, y pueden ser usados como textos guía en diversos cursos o como apoyo para reforzar el desarrollo profesional.

Alfaomega espera contribuir así a la formación y el desarrollo de profesionales exitosos para beneficio de la sociedad.

ALFAOMEGA

EL AUTOR

Gustavo Adolfo Galeano Atehortúa es ingeniero electrónico de la Universidad Pontificia Bolivariana de Medellín, certificado por la compañía Freescale Semiconductor para el soporte técnico en los países de la zona Andina, Centroamérica y el Caribe.

Inició como ingeniero de desarrollo de hardware y software en la compañía Celsa s.a., y luego fue uno de los fundadores de la empresa Propuesta Dinámica, con la cual representó a varios fabricantes de semiconductores, entre ellos a Motorola SPS, que más tarde se convertiría en Freescale Semiconductor.

Como docente ha trabajado en las universidades EAFIT, UPB y UDEA de Medellín, teniendo a su cargo asignaturas relacionadas con programación de microcontroladores.

Actualmente se desempeña como Gerente de ventas y soporte técnico de la compañía estadounidense Richardson Electronics LTD., prestando soporte para el desarrollo de aplicaciones embebidas y sistemas inalámbricos.

CONTENIDO

CAPÍTULO 6

Funciones y sentencias de control

CAPÍTULO 7

Librerías estándar en C para sistemas embebidos

CAPÍTULO 10

Sistemas operativos de tiempo real para sistemas embebidos

LISTA DE EJEMPLOS

CONTENIDO PÁGINA WEB

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Capítulo 7

Capítulo 8

Capítulo 9

Capítulo 10

Anexo No. 2

CÓDIGO WEB

Para tener acceso al material de la página Web de apoyo del libro "Programación de sistemas embebidos en C, teoría y prácticas aplicadas a cualquier microcontrolador":

1. Ir a la página
   http://virtual.alfaomega.com.mx
2. Registrarse como usuario del sitio.
3. Ingresar al apartado de inscripción de libros y registrar la siguiente clave de acceso:

4. Para navegar en la plataforma virtual de recursos del libro, usar los nombres de Usuario y Contraseña definidos en el punto número dos.

PRÓLOGO

El avance actual de la microelectrónica implica grandes modificaciones en los sistemas digitales con aplicaciones industriales. La programación en sistemas embebidos es parte cotidiana de nuestras vidas, porque permite el correcto funcionamiento de dispositivos electrónicos basados en microcontroladores como teléfonos, televisores, hornos de microondas, juguetes, computadoras de automóviles, dispositivos industriales, entre muchos otros.

Debido a la infinidad de aplicaciones basadas en los microcontroladores, es de suma relevancia para cualquier ingeniero del área electrónica contar con una adecuada formación en el uso y programación de microcontroladores. Las técnicas de enseñanza evolucionan constantemente con los nuevos desarrollos en el diseño y realización de sistemas digitales y por esto surge la necesitad de adaptar los contenidos de la enseñanza a los nuevos circuitos, con el objetivo de capacitar ingenieros competitivos y que se adapten con facilidad a los estándares y necesidades del mercado.

La presente obra introduce al lector en el uso y programación en lenguaje C de microcontroladores de cualquier marca; el libro presenta una abundante cantidad de ejemplos, basándose en dos de las principales marcas del mercado: los microcontroladores de la empresa Freescale Semiconductors y los de la empresa Microchip Technology Inc. Al ser éste un libro teórico-práctico, resulta atractivo y útil para los estudiantes por la sencillez con la que se abordan los temas y por la gran cantidad de ejemplos que ilustran de manera clara la programación embebida. Esta obra también es de potencial interés para la industria que diseña dispositivos con microcontroladores, porque facilita una rápida capacitación de los ingenieros en el tema de la programación de microcontroladores en lenguaje C.

A lo largo de los 10 capítulos de este libro, el lector encontrará una abundante cantidad de ejemplos prácticos, así como *recetas* útiles que facilitan la programación embebida, las cuales coadyuvan en la optimización de la misma.

Los primeros 4 capítulos esclarecen los conceptos básicos de los microcontroladores, su arquitectura básica, los compiladores para su programación y se muestra una comparativa entre la programación de microcontroladores en lenguaje ensamblador vs. la programación en lenguaje C. El lector se adentrará principalmente en las arquitecturas de los microcontroladores de dos de las principales marcas: Microchip Technology Inc. y Freescale Semiconductors.

De la misma forma, en el tema de los compiladores, el lector abordará principalmente el manejo de los compiladores CCS y Codewarrior, de las mismas empresas; sin embargo, el profundizar un poco más en estas dos marcas no limita al lector al uso de estas dos tecnologías porque, en la práctica y con el conocimiento adquirido a lo largo del libro, el lector podrá abordar sin problema alguno cualquier tecnología de microcontroladores del mercado. Los capítulos 5 al 7, llevarán al lector a adentrarse en la programación de microcontroladores en lenguaje C, y en el 8 el lector conocerá las técnicas de optimización de ahorro de energía de un microcontrolador.

El capítulo 9 presenta una poderosa herramienta de programación gráfica conocida como "Processor Expert™" de la empresa Freescale Semiconductors, la cual, sin duda alguna, es una revolución en el concepto de interfaz Computadora-Programador, ya que el ambiente gráfico sitúa al programador en

un nivel más intuitivo de programación. Por último, el capítulo 10 introduce al lector al fascinante ambiente de los sistemas operativos en tiempo real, los cuales son imprescindibles hoy en día en una gran cantidad de procesos industriales.

Esta obra realizada por el Ingeniero Gustavo Galeano, sin duda alguna refleja su gran conocimiento en el ámbito de los microcontroladores. Con una experiencia de 15 años en el mercado de los semiconductores y con una sólida formación gracias a la práctica multidisciplinaria adquirida como ingeniero de aplicación, Galeano plasma todo su conocimiento y experiencia para beneficio de la comunidad docente e industrial.

Sin duda alguna, este libro es un gran aliado para la enseñanza de la programación en lenguaje C de microcontroladores, con el cual, alumnos de ingeniería y hasta programadores avanzados, encontrarán las bases y una gran cantidad de ejemplos prácticos que les permitirán aprender a programar microcontroladores en lenguaje C, con las limitantes de velocidad y consumo propios de estos dispositivos.

Dr. J. Enrique Chong Quero
Profesor, Investigador y Director
Departamento de Ingeniería Eléctrica y Electrónica
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Abril de 2009

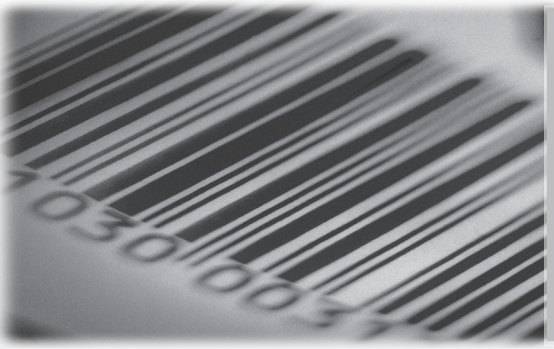
Los conocimientos y habilidades que en este libro se presentan, nos permiten tener acceso al maravilloso mundo de la programación de los sistemas embebidos, mediante ejemplos prácticos, los cuales nos brindan la oportunidad de conocer los aspectos técnicos esenciales en el funcionamiento de los microcontroladores, así como las herramientas de desarrollo que facilitan su implementación.

Un avance adicional que hemos podido aprovechar recientemente, es utilizar el lenguaje de programación C para programar microprocesadores y microcontroladores.

El libro está enfocado a aquellos que deseen ser los alquimistas modernos, quienes permutarán el plomo (en este caso silicio) de los circuitos integrados, en oro y riqueza para la humanidad, tanto social como económica.

Es mi mayor deseo que este texto ayude a potencializar todas estas ideas y que permita el aprendizaje de la tecnología.

Mtro. Waldo Cervantes Solís
Coordinador de Ingeniería Electrónica
Departamento de Ingenierías
Universidad Iberoamericana
Ciudad de México, México
Abril de 2009



CAPÍTULO 1

CONCEPTOS BÁSICOS

El propósito de este capítulo es presentar los conceptos que el programador deberá tener claros, antes de enfrentar la programación en el lenguaje C.

Introducción

El capítulo está orientado a definir los conceptos sobre programación general que servirán durante el completo recorrido del texto; inicialmente se define lo que es un sistema embebido, mostrando el modelo expandido hasta llegar al popular modelo de una sola pastilla (*single chip*).

Se aclara el concepto de compilador con respecto al de interpretador y se muestran los pasos típicos que un compilador tiene que realizar antes de generar el programa ejecutable en el chip, dejando muy claro los tiempos que tardan los procesadores involucrados en el diseño: el que realiza la compilación (tiempo de compilación) y el que ejecuta el código generado por el primero (tiempo de ejecución).

Un poco de historia sobre los orígenes del C y el porqué ha llegado a convertirse en el lenguaje más popular de programación, tanto para programadores de alto nivel en computadoras comerciales como para programadores de microcontroladores de bajo costo. Se muestra de forma general la estructura de un código desarrollado en lenguaje C y se familiariza al programador, que viene del lenguaje de ensamblador, con las palabras reservadas, los operadores y los signos propios de un código en lenguaje C.

Se abordan algunos de los periféricos existentes en los microcontroladores del mercado, con los cuales se realizarán prácticas reales en capítulos posteriores, como son los dispositivos de entrada y salida (I/O), el convertidor analógico a digital (ADC), el perro guardián (COP), el detector de bajo nivel de voltaje (LVI), el temporizador (TIMER), y las comunicaciones seriales asincrónicas (SCI) y sincrónicas (SPI, I2C).

Se dedica un buen espacio a un tema con el que siempre tienen que ver los sistemas embebidos, como lo es el de las interrupciones, parte fundamental y crítica en cualquier diseño basado en microcontroladores. Este concepto será aclarado con un ejemplo de la vida cotidiana y luego puesto en el contexto propio de un procesador.

Al final se describen las herramientas de desarrollo de cuatro de las marcas más populares de fabricantes de microcontroladores como son las de **Microchip**, **Renesas**, **Texas Instruments** y **Freescale** y la posibilidad de tener una herramienta de diseño propia a muy bajo costo, para ejercitar de forma práctica todos los ejemplos del libro.

1.1 ¿QUÉ ES UN SISTEMA EMBEBIDO?

Se conoce como **sistema embebido** a un circuito electrónico computarizado que esta diseñado para cumplir una labor especifica en un producto.

La inteligencia artificial, secuencias y algoritmos de un sistema embebido, están residentes en la memoria de una pequeña computadora denominada microcontrolador.

A diferencia de los sistemas computacionales de oficina y *laptops*, estos sistemas solucionan un problema específico y están dispersos en todos los ambientes posibles de la vida cotidiana. Es común encontrar sistemas embebidos en los vehículos; por ejemplo, controlando el sistema de inyección de combustible, en los sistemas de frenado ABS (*Anti-lock Braking Systems*), en el control de espejos, sistemas de protección contra impacto (*Airbag*), alarmas contra robo, sistemas de ubicación, entre otros. También en los electrodomésticos de uso diario: controlando la temperatura en refrigeradores, estufas, hornos microondas y planchas; el motor de licuadoras, lavadoras de ropa, lavaplatos, aspiradoras y juguetes; en los equipos celulares, agendas de bolsillo, PDA, cajeros automáticos, cámaras fotográficas, reproductores de música (MP3) y video, equipo de gimnasio, equipo médico, y en general, en una gran cantidad de dispositivos de uso diario (*ver Gráfico 1.1*).

Los sistemas embebidos controlan un sinnúmero de objetos cotidianos, como celulares, agendas de bolsillo, cámaras fotográficas o sistemas de inyección de combustible, entre otros.

Se sabe que en general, un consumidor promedio interactúa con alrededor de 400 microcontroladores por día; este número tiende a crecer significativamente para los próximos años, considerando que los procesadores son cada vez mas pequeños, consumen menos energía y el precio es menor gracias a la economía de escala aplicada en su fabricación, aspectos que ayudan a reemplazar en mayor proporción los sistemas lógicos, los equipos electromecánicos y en el futuro, se podrán incorporar en los equipos desechables.

GRÁFICO 1.1

Sistemas Embebidos en todas las áreas[1].

La programación embebida permite desarrollar instrucciones precisas para microcontroladores que cumplen funciones específicas; por ejemplo, controlar el ciclo de trabajo en una lavadora.

La programación embebida, como es el caso de este libro, va siempre orientada a aplicaciones portables o compactas, alimentadas por batería o por una fuente de poder de baja capacidad de corriente (por cuestiones de espacio), menor disipación de calor y muy económica.

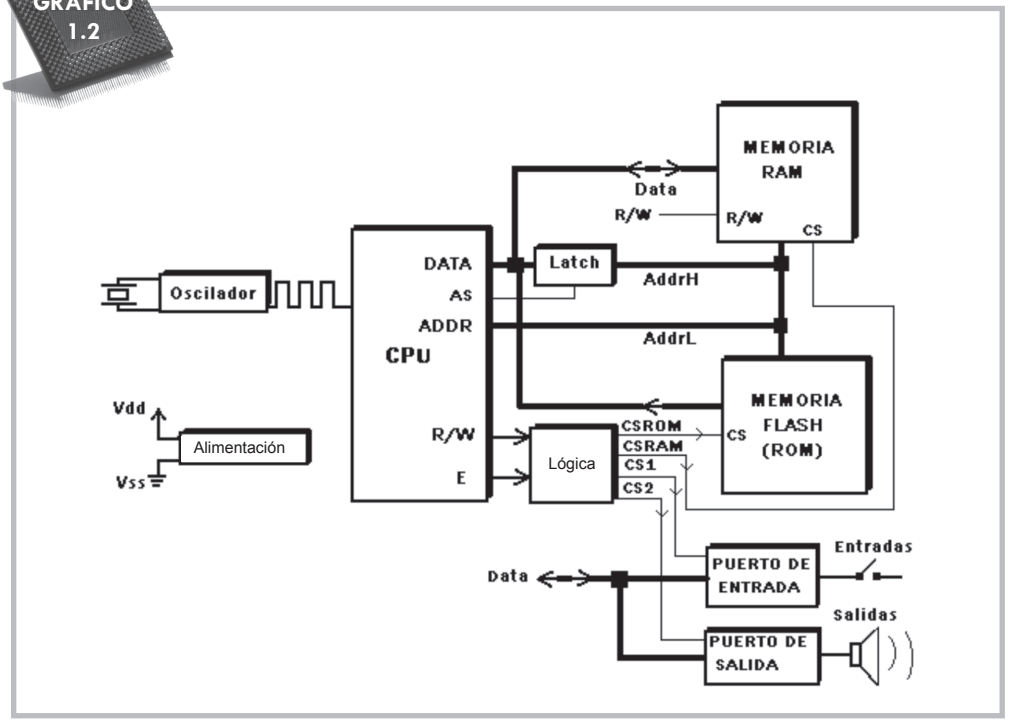
Lo anterior apunta a tener una aplicación final de tamaño reducido y de bajo costo; sin embargo, es propio de los sistemas embebidos su robustez. Esta característica se debe al gran rango de aplicaciones y ambientes que cubren: industriales, de consumo, automotriz, electrodomésticos, donde el equipo final puede estar sometido a situaciones muy exigentes como pueden ser el polvo, la humedad, la vibración, rotaciones de alta velocidad, situaciones extremas de presión y/o temperatura.

El número de aplicaciones y de ambientes soportados por los sistemas embebidos crece cada vez más, y una de las principales razones fue la llegada de los procesadores de una sola pastilla (microcontroladores *single-chip*), en los cuales una gran parte de la electrónica está incorporada y permite reducciones de tamaño, menor consumo y facilidades de producción.

1 Imagen cortesía de Freescale™.

GRÁFICO 1.2

Modelo expandido de un sistema Microprocesador.



Descripción de señales:

AS: Señal de Address Strobe.

R/W: Señal de Read/Write.

E: Señal de Sincronización.

CSROM: Chip Select de Memoria de programa.

CSRAM: Chip Select de Memoria de datos.

Data: Bus de Datos bidireccional.

ADDR: Bus de Direcciones (AddrH: AddrL).

Oscilador: Señal de reloj.

Vdd: Alimentación de voltaje.

Vss: Referencia negativa de voltaje.

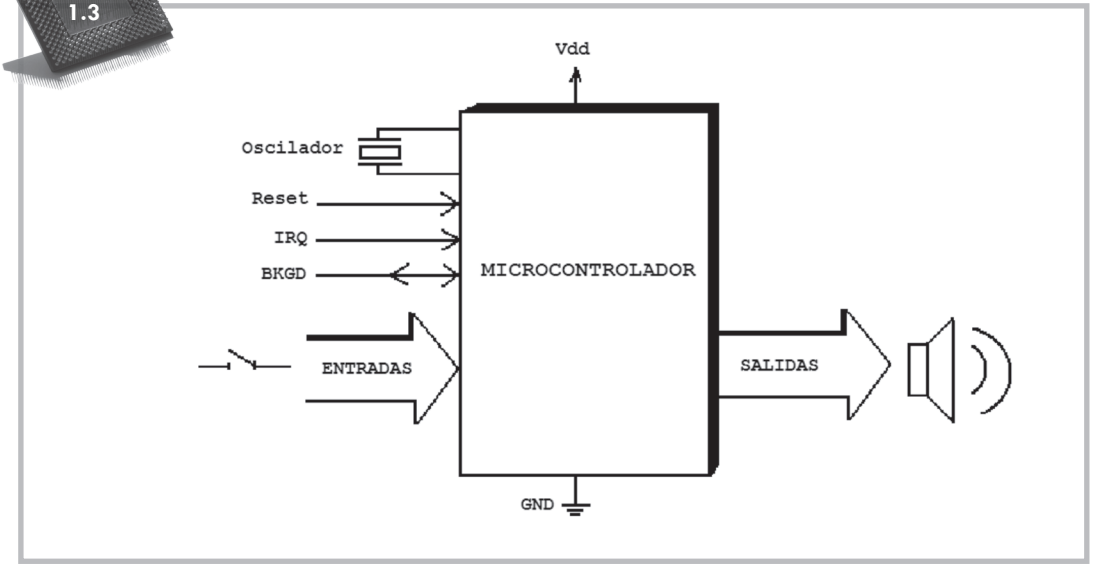
CS1: Chip Select de puerto de Entrada.

CS2: Chip Select de puerto de Salida.

El sistema embebido *single-chip* contiene en usa sola pastilla de silicio, todo el esquema de un sistema expandido convencional con su procesador en el centro, sistema de decodificación, sistema de multiplexaje datos/dirección, memorias de almacenado (en la cual es ejecutado el programa), memoria RAM, además de las señales de sincronización que administran el acceso a las memorias y los ciclos de lectura y escritura (R/W) (*ver Gráfico 1.2* y *Gráfico 1.3*).

GRÁFICO 1.3

Modelo Single-Chip de un Sistema Microcontrolador.



1.2 CONCEPTOS BÁSICOS DE PROGRAMACIÓN EN ALTO NIVEL

1.2.1 ¿Qué es un compilador? ¿Qué es un interpretador?

El programa editado es llevado a la máquina que lo ejecutará finalmente, en este caso un microcontrolador comúnmente llamado "*Target*", de dos formas posibles:

- El programa es convertido a código de máquina al 100% y luego se ejecuta.
- El programa es enviado al "*target*" tal cual fue editado, y éste toma línea por línea y lo va convirtiendo a código de máquina a medida que va pasando por él.

En el primer caso se dice que el código es ***compilado***, y en el segundo caso, que el código es ***interpretado***.

Un archivo compilado se ejecuta directamente en el microcontrolador porque está convertido a lenguaje máquina; en cambio, un archivo interpretado requiere un software incluido en el microcontrolador para su traducción.

Un **compilador** es un programa de software que transforma uno o varios archivos de código fuente y genera un archivo en código de máquina llamado ejecutable; este nuevo archivo es enviado al "*target*" para que lo ejecute.

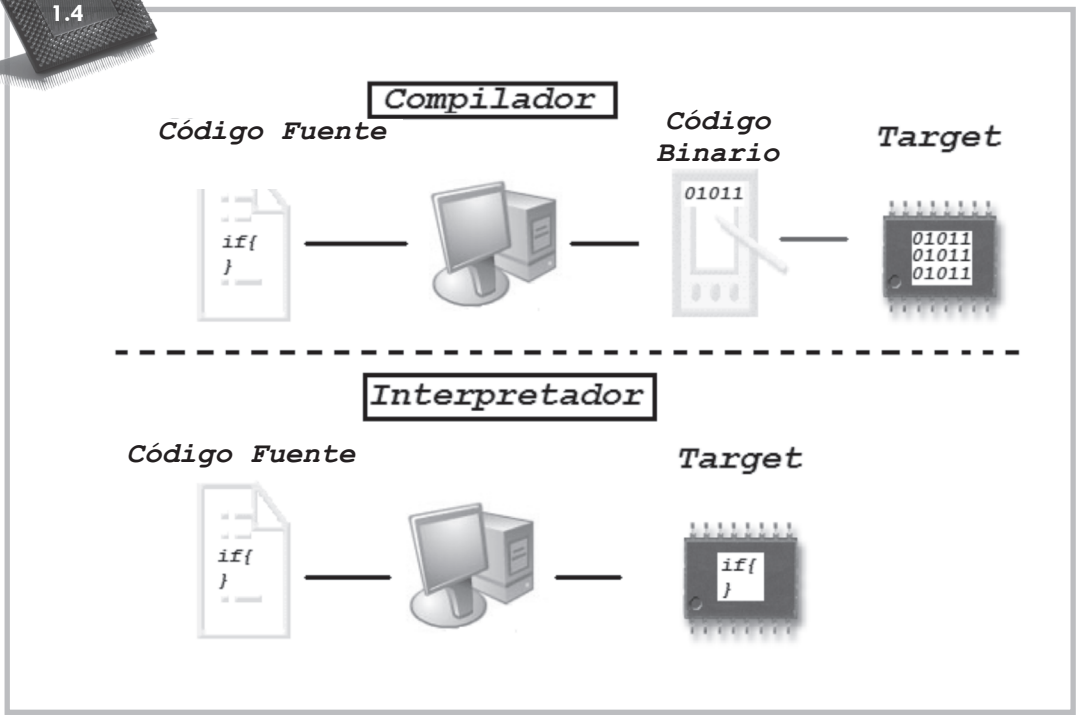
Un **interpretador** es un software que es instalado en el "*target*", el cual está preparado para recibir un archivo fuente editado; una vez el "*target*" recibe la orden de ejecutar inicia el proceso de cambiar línea por línea de programa a su respectivo código de máquina y ejecutar este código (*ver Gráfico 1.4*).

Ambos esquemas tienen ventajas. Para el caso de un compilador el tiempo requerido para convertir el código que ejecutará la máquina se utiliza completamente antes de su ejecución final, y en el caso de compiladores cruzados, este código lo convierte una máquina con mayores recursos que el "*target*", así el código queda listo para ejecutarse ahorrando tiempo a la máquina final, la cual no se tiene que enterar de los pormenores del código. Sin embargo, para hacer la conversión a código de máquina, se deberá conocer con anterioridad la máquina que va a ejecutar el código, y este código quedará limitado a ejecutarse en dicha máquina.

En el caso del interpretador, se tiene el código fuente hasta el último momento, cuando es ejecutado y el "*target*" se encarga de la conversión, de modo que el código no queda limitado a ejecutarse en determinada máquina, sino que quedará libre para que cualquier máquina lo tome, lo convierta y lo ejecute. La gran ventaja en este caso es su amplia portabilidad, lo que independiza el código de la máquina que lo ejecuta; sin embargo, su gran desventaja radica en que la máquina tendrá que invertir parte de su tiempo y recursos en tomar el código y convertirlo a su respectivo código de máquina para luego hacer su ejecución.

Una vez que el programa está compilado las líneas del código fuente dejan de tener sentido en la ejecución del programa. Cuando se usa un intérprete, el código fuente debe estar presente cada vez que se quiere ejecutar el programa.

GRÁFICO 1.4 Ejecución de código en Compilador y en Interpretador.



Receta

Para sistemas embebidos es preferible usar compiladores a interpretadores, debido a las limitaciones en tiempo de ejecución del procesador.

1.2.2 Estructura y pasadas de un compilador

El compilador, como programa de software, seguirá varios pasos para realizar la conversión del código texto original .C y .H a su código ejecutable, en este caso .S19 o .HEX, que será el archivo que se enviará al sistema final *target*".

Los archivos con extensión .C y .H que corresponden al proyecto, son inicialmente pasados por un subprograma llamado el **pre-procesador**, el cual se encarga de crear archivos intermedios con la solución de todos los macros (conversión de texto en números o equivalencias), solucionar la compilación condicional (en caso de existir), e incluir el código que finalmente se compilará (*ver Gráfico 1.5*).

Este archivo temporal se pasa a otro subprograma encargado del **Análisis Semántico**, donde se analiza línea a línea el código, verificando que todos los paréntesis y/o corchetes abiertos sean cerrados, que las palabras reservadas estén correctamente escritas o, lo que es lo mismo, que todas las palabras estén en el léxico del C.

Si este subprograma detecta algún error, suspenderá su entrega al subprograma siguiente y dará como salida un archivo de errores (típicamente de extensión .ERR), el cual es usado para mostrar en el ambiente del editor uno a uno los errores que el compilador encuentra en este paso.

Si por el contrario, no se tienen errores sintácticos, se envía el nuevo código intermedio al **Generador de Código**, el cual tendrá como salida un código en ensamblador (extensión .ASM), equivalente línea a línea al código de entrada entregado.

Este nuevo archivo en lenguaje de máquina .ASM, es pasado por otro subprograma denominado el **Optimizador**, este tomará el archivo y realizará un análisis global al código y dependiendo de las optimizaciones señaladas por el programador, obviará algunas líneas de código, modificando así el código de máquina y generando un nuevo código más corto, más rápido y con la misma funcionalidad del archivo original. Este Optimizador generará a su vez, además del archivo .ASM optimizado, un archivo paralelo de extensión .LST que contiene la equivalencia entre el código ASM y el Código original.

El nuevo archivo ensamblador .ASM pasará por el programa **Ensamblador**, el cual generará el código objeto (extensión .OBJ o .O), que es un código en lenguaje de máquina codificado de los archivos del proyecto.

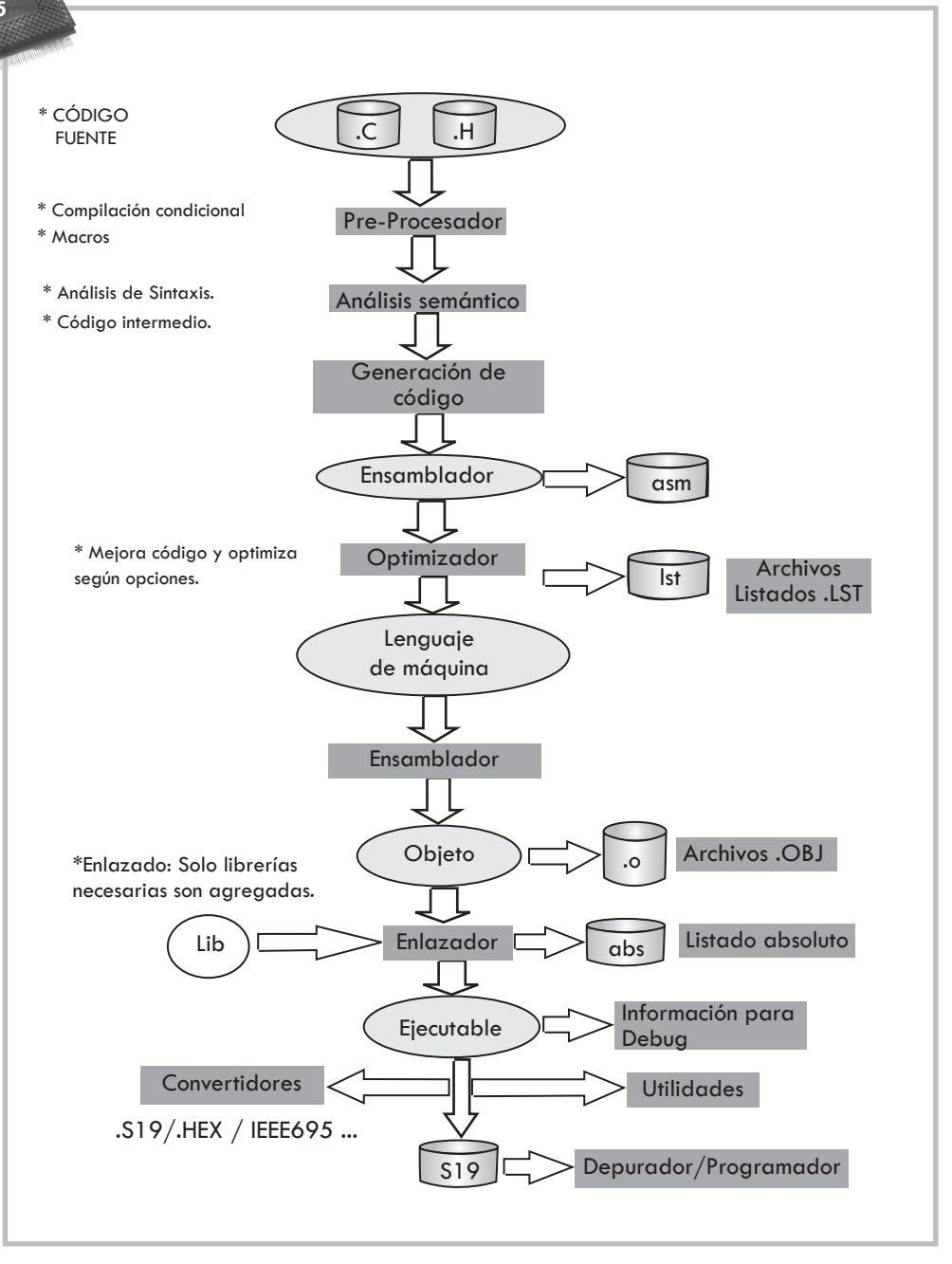
Este archivo es luego enlazado por medio del ***linker*** o **Enlazador**, el que agregará el contenido de las liberarías de C que se invocaron en el proyecto, y de esta forma, generará un código ejecutable posicionado en las direcciones específicas que indica el proyecto, creando el archivo final .S19, .HEX (o ejecutable) y otros archivos adicionales de ayuda para la depuración, sea en tiempo de simulación o en tiempo real.

En una compilación el pre-procesador crea archivos intermedios, los cuales pasan al subprograma de Análisis Semántico y de ahí al Generador de Código; luego, un Optimizador lo analiza y elimina las líneas innecesarias para generar un archivo .ASM en lenguaje máquina.

Todo archivo .ASM optimizado debe pasar por el programa Ensamblador, el cual generará el código objeto .OBJ, que será vinculado a las direcciones específicas indicadas por el proyecto a través del programa Enlazador, originando un archivo final .HEX o ejecutable.

GRÁFICO 1.5

Estructura y pasadas de un compilador estándar.



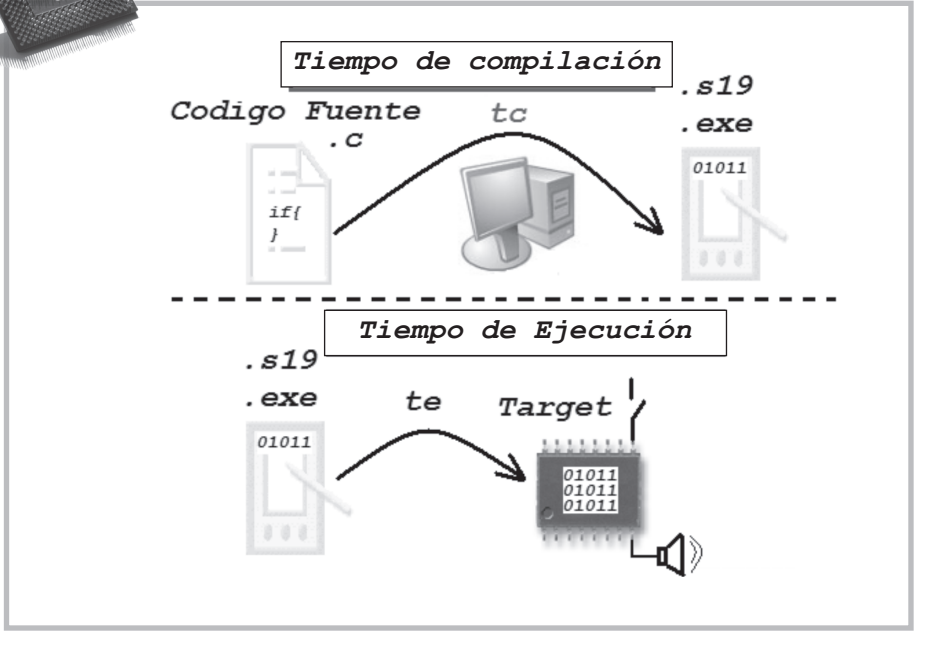
1.2.3 Tiempo de compilación vs. tiempo de ejecución

El **tiempo de compilación** (*tc* en *Gráfico 1.6*), es el tiempo que tarda el compilador instalado en el PC, para realizar la conversión de todas las líneas de código y pasarlo a lenguaje de máquina; esta conversión incluye la solución de todos los símbolos y operaciones que son posibles resolver. Una vez ejecutado, este tiempo no requerirá ser utilizado por el procesador final "*target*".

Para el caso de un interpretador, todo el código deberá ser utilizado en tiempo de ejecución, y el tiempo de compilación será nulo; como sabemos, el código tal como fue digitado es pasado al procesador final para su completa ejecución.

El **tiempo de ejecución** *te* en *Gráfico 1.6* es el tiempo que toma el "*target*" en realizar una operación o resolver alguna expresión, previa evaluación de cada una de las variables involucradas en el proceso.

Tiempo de compilación (tc) y tiempo de ejecución (te).



Cuando se compila un archivo el software debe traducir las líneas del código a lenguaje máquina, lo que tarda dicha operación se conoce como *tiempo de compilación*.