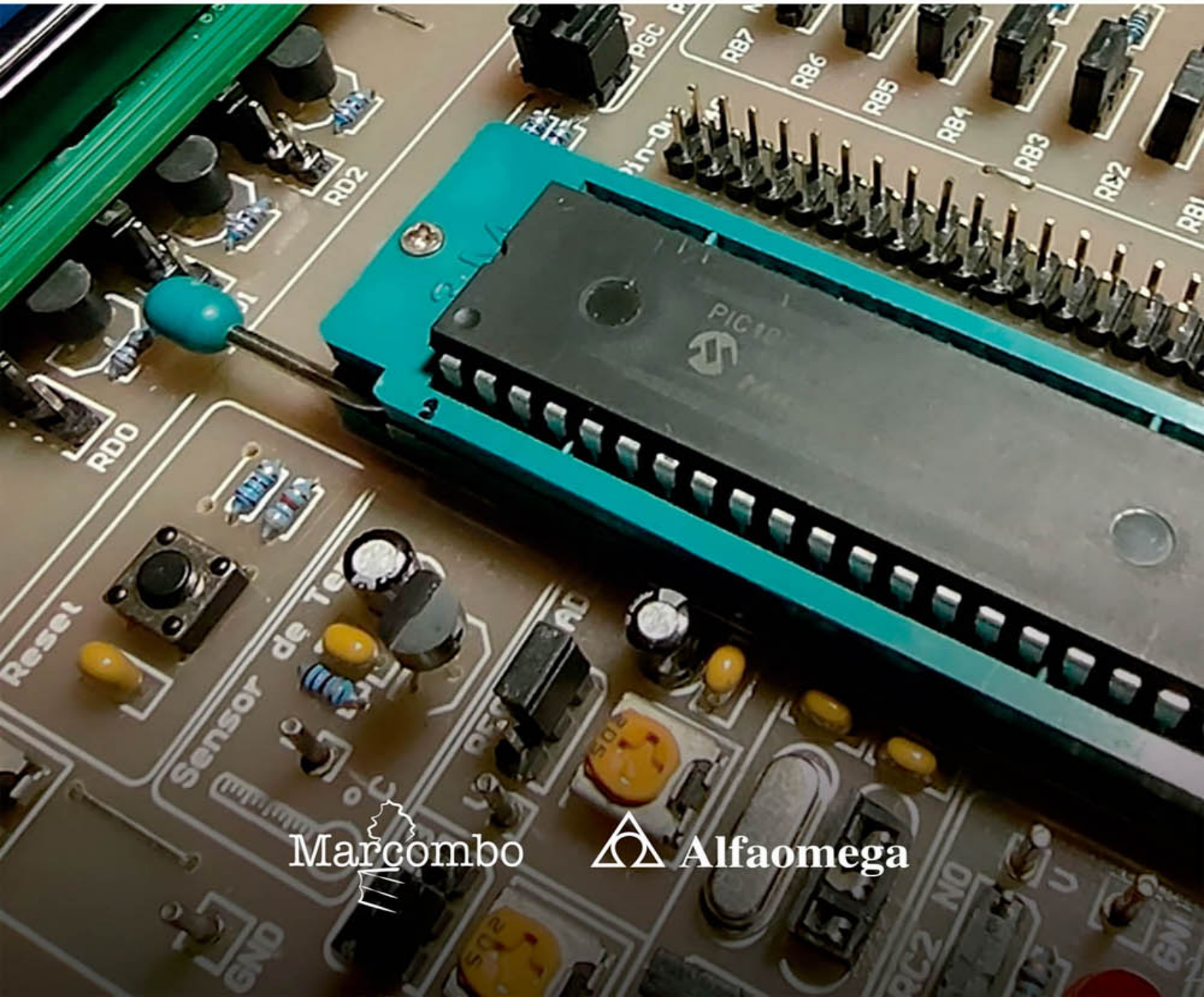
MikroC Pro para PIC y ARM Cortex

PROGRAMACIÓN Y EJEMPLOS

DANIEL SCHMIDT



Marcombo
Alfaomega

MikroC Pro para PIC y ARM Cortex

Programación y ejemplos

Daniel Schmidt

MikroC Pro para PIC y ARM Cortex

Programación y ejemplos

Daniel Schmidt

Marcombo

Alfaomega

MikroC Pro para PIC y ARM Cortex: programación y ejemplos

Daniel Schmidt

Primera edición: 2022
ISBN: 978-987-3832-84-0

Primera edición: MARCOMBO, S.L. 2022

www.marcombo.com

ISBN: 978-84-267-3536-2
Producción del ePub: booqlab

A mi esposa Mabel, compañera inseparable en todos mis caminos.

Contenido

Capítulo 5

Programando ARM con MikroC

Historia de la arquitectura ARM
Qué es Cortex M4
Características heredadas de RISC
Algunas ventajas de RISC
Desventajas de RISC
Bus AMBA
Pipeline
FPU
ARM y Thumb
El sistema de memoria ARM
STM32F407VG Discovery
Características de la placa entrenadora
Shield para Discovery
Qué es MikroBUS
Qué necesito para trabajar con ARM
MikroC Pro para ARM
Configurando el entorno de trabajo
Configurando el reloj y los buses internos
Puedo programar el microcontrolador sin un programador específico
Mi primer programa en MikroC

Capítulo 6

Interrupciones con ARM

Interrupciones para STM32F407
Temporizador del sistema (SysTick)
Pantalla LCD 16 × 2 con STM32
Funcionamiento de la USART
Conversor analógico con STM32F407VG
  Conversor analógico por interrupción
  Midiendo la temperatura del núcleo Cortex

Capítulo 7

Manejo de los canales DMA con MikroC

Capítulo 8

Enlace wifi con ARM y MikroC

Acerca de este libro

El eje central del libro es la programación para PIC de Microchip en ocho bits y en treinta dos bits para la arquitectura de ARM con el compilador MikroC de MikroElektronika.

Se trata de un compilador largamente probado y, sin duda, es una de las herramientas que, junto con los compiladores oficiales de las correspondientes marcas, genera código confiable y muy eficiente en el uso de los recursos de los microcontroladores.

MikroC ofrece gran cantidad de código resuelto, funciones y drivers contenidos en una extensa biblioteca que facilitan el trabajo del programador, al acortar los tiempos de desarrollo y de depuración de código.

Para la mayoría de los programadores de microcontroladores, incorporar a su esquema de trabajo la arquitectura de ARM puede ser un paso complejo, debido a las grandes diferencias que existen con otras arquitecturas, como PIC, Atmel, etc.

Pensando en esto se ha desarrollado el presente trabajo, que pretende hacer más fácil el hecho de aprender a programar tanto ARM con su núcleo Cortex como PIC con el compilador MikroC.

También usaremos Visual TFT, un software creado por MikroElektronika para el desarrollo de interfaces gráficas con pantallas TFT, que genera código para ser compilado directamente por sus compiladores, lo cual acelera y facilita el diseño de interfaces con pantallas táctiles. Si bien considero este trabajo como introductorio a la programación en el lenguaje C, en él encontrará una gran cantidad de ejemplos, rutinas de programación, librerías y textos explicativos sobre una diversidad de temas que pueden ser de utilidad no solo en el proceso de entender cada una de las arquitecturas, sino también para aplicar en desarrollos electrónicos en general.

Todos los ejemplos propuestos están pensados para su realización práctica con electrónica real. Esto no significa que no pueda usar simuladores para corroborar su funcionamiento. Sin embargo, la experiencia nos enseña que los resultados obtenidos en el simulador no siempre coinciden con el comportamiento en el terreno físico.

Recuerde que la simulación se ejecuta en un entorno ideal (la memoria del computador), sin ruidos eléctricos ni interferencias del mundo real.

Acerca del autor

Daniel Schmidt

Schmidt es fundador y director de Firtec Argentina y docente universitario en electrónica digital, programación en C para microcontroladores y control de procesos industriales mediante sistemas electrónicos.

Ha participado en distintos congresos internacionales como expositor y/o moderador, y cuenta con numerosas publicaciones técnicas en distintos campos de la electrónica programable y con reconocimientos y certificaciones internacionales.

A nivel académico, centra su atención en explicar de forma clara y simple lo que se puede hacer —y cómo— con un microcontrolador. Deja de lado el complejo análisis de ingeniería y convierte sus textos en una lectura fácil de seguir y con gran cantidad de ejemplos, que permiten verificar el funcionamiento práctico de los contenidos teóricos.

Esta particular forma de redactar sus libros ha permitido que muchos técnicos electrónicos autodidactas puedan ingresar fácilmente al mundo de la programación para electrónica.

En sus trabajos se encuentran resueltos muchos temas puntuales sobre la adquisición y el manejo de datos con electrónica, lo que acorta los tiempos de aprendizaje de nuevas tecnologías para quienes ya están trabajando con microcontroladores y necesitan actualizar sus conocimientos.

En muchas de sus obras encontrará enlaces para descargar ejemplos, notas técnicas e información adicional de utilidad para hacer la lectura más entretenida y dinámica. Así, sus libros se convierten en verdaderos cursos sobre el tema tratado.

Comentarios del autor

Como casi todos los que sumamos algunos años en electrónica, iniciamos nuestro camino en la programación de la mano de Motorola. Sin duda, eran microcontroladores confiables y poderosos. Pero trabajar con ellos en aquellos lejanos tiempos era todo un desafío: borrado ultravioleta, voltajes de programación altos, procesos lentos y tediosos, el lenguaje era ensamblador, doscientos comandos que alguna vez supimos escribir de corrido sin consultar la tabla de referencia (algo normal para la época), etc.

Cuando los microcontroladores PIC hicieron su aparición, ofrecieron la posibilidad de programación y borrado eléctrico gracias a su memoria EEPROM. Ya no se necesitaba luz ultravioleta ni un complejo hardware de programación; incluso se podía armar el programador con componentes simples que se adquieren en cualquier negocio de electrónica.

Pero, claro, si algo evoluciona rápido es la tecnología, y pronto los ocho bits sonaron a poco para las exigencias crecientes del mercado electrónico. Así llegamos a los nuevos núcleos de 32 bits con arquitectura ARM, que también tratamos en las siguientes páginas.

No se espera que al terminar la lectura de este libro usted sea un experto programador en C, pero sí que tenga una idea clara de lo que es el lenguaje con uno de los compiladores más simple de usar.

Un comentario que siempre hago a mis alumnos es que la evolución es un paso hacia adelante, un cambio superador; y en el campo de la electrónica programable, eso es C.

A partir de este punto, dependerá de usted hasta dónde quiera llegar y cuán largo será el camino.

Gracias por permitirme ser parte de ese camino.

Daniel Schmidt

CAPÍTULO 1
PROGRAMANDO PIC CON MIKROC

LOS PIC Y UN POCO DE SU HISTORIA

Los PIC son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instrument.

El nombre actual no es un acrónimo. En realidad, el nombre completo es PICmicro, aunque generalmente nos referimos a ellos como Peripheral Interface Controller (Controlador de Interfaz Periférico).

El PIC original se diseñó para ser usado con la CPU CP16000. Dicha CPU, en general, tenía prestaciones pobres de entradas y salidas. El PIC de 8 bits se desarrolló, en 1975, para mejorar el rendimiento de la CP16000 mejorando las prestaciones de entradas y salidas.

Para realizar estas tareas, el PIC utilizaba microcódigo simple almacenado en ROM.

Se trataba de un diseño RISC (aunque en esa época no se usaba este término), que ejecutaba una instrucción cada 4 ciclos del

oscilador.

En 1985, la división de microelectrónica de General Instrument se separó como compañía independiente y cambió el nombre a Microchip Technology.

El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable. Hoy en día, multitud de PIC vienen con varios periféricos incluidos (módulos de comunicación serie, UART, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32 000 palabras (una palabra corresponde a una instrucción en lenguaje ensamblador, y puede ser de 12, 14, 16 o 32 bits, dependiendo de la familia específica de PIC).

El PIC usa un juego de instrucciones, cuyo número puede variar desde 35 (para PIC de gama baja) hasta 70 (para los de gama alta). Las instrucciones se clasifican entre: las que realizan operaciones entre el acumulador y una constante, entre el acumulador y una posición de memoria, instrucciones de condicionamiento y de salto/retorno, implementación de interrupciones, y una para pasar a modo de bajo consumo llamada *sleep*.

Microchip proporciona un entorno «oficial» de desarrollo libre, llamado MPLAB, que incluye un simulador software y un ensamblador.

En nuestro caso, estaremos usando MikroC Pro para PIC, que es un IDE más compilador capaz de generar código prácticamente para todos los PIC.

Para transferir el código desde un PC al PIC, normalmente se usa un dispositivo llamado programador. La mayoría de los PIC que Microchip distribuye incorporan ICSP (In Circuit Serial Programming, programación serie incorporada) o LVP (Low Voltage Programming, programación a bajo voltaje), lo que permite programar el PIC directamente en el circuito destino.

Para la ICSP se usan los pines RB6 y RB7 como reloj y datos (en algunos modelos pueden usarse otros pines, como el GP0 y

GP1 o el RA0 y RA1), y el MCLR para activar el modo programación aplicando un voltaje de 13 voltios.

Existen muchos programadores de PIC, desde los más simples, que dejan al software los detalles de comunicaciones, hasta los más complejos, que pueden verificar el dispositivo a diversas tensiones de alimentación e implementan en hardware casi todas las funcionalidades. Muchos de estos programadores complejos incluyen PIC preprogramados como interfaz para enviar las órdenes al PIC que se desea programar.

La arquitectura se dirige a maximizar la velocidad. PIC fue uno de los primeros diseños escalares de CPU, y sigue siendo uno de los más simples y baratos. La arquitectura de Harvard, en que las instrucciones y los datos provienen de diferentes fuentes, simplifica la sincronización y el diseño de microcircuitos.

El conjunto de instrucciones es adecuado para la aplicación de tablas de búsqueda rápida en el espacio de programa. Estas búsquedas tienen una instrucción y dos ciclos de instrucción. Muchas de las funciones se pueden modelar de esta manera.

CÓMO FUNCIONA UN MICROCONTROLADOR PIC

Para entender un poco cómo funcionan las cosas dentro de un microcontrolador, podríamos comenzar definiendo sus partes.

CPU

Dentro de la CPU está la Unidad de Control (UC), que se encarga de que todo suceda de acuerdo con lo previsto. La UC decide cuándo y cómo se llevan a cabo las acciones.

Se lee una instrucción. Desde la memoria de programa, la instrucción pasa al decodificador de instrucciones, y el contador de

programa (PC) se incrementa en uno para leer la siguiente dirección en la memoria de programa.

La Unidad de Aritmética y Lógica (ALU)

Dentro de la CPU, es la encargada de resolver los problemas matemáticos y lógicos. Cuando por ejemplo hay que resolver una operación de suma, la UC le pasa a la ALU las variables, esta lo resuelve y le entrega los resultados a la UC, para que esta disponga de ellos. Entonces, la UC consulta el programa para saber qué debe hacer con el resultado.

I/O Ports

Controlan los pines, que son el vínculo con el exterior. Estos pines se pueden configurar como entradas o salidas, dependiendo de la necesidad.

Memoria de programa

Es la memoria donde reside el código de la aplicación. Si, por ejemplo, hemos programado una alarma, el código de la alarma reside en esta memoria.

La memoria de programa es la que programamos con nuestro programador. Esta memoria es del tipo Flash.

Memoria RAM

La memoria RAM es una memoria de lectura y escritura que es temporal. Es decir, los datos se mantienen el tiempo que haya energía en el sistema; si la memoria se queda sin energía, la información se pierde. En un microcontrolador, es la memoria destinada a guardar las variables temporales de proceso, que son aquellas variables de programa que solo son importantes en ese momento. Todos los microcontroladores tienen memoria RAM (algunos más, otros menos; pero todos disponen de esta memoria). El programa no la usa solamente para guardar los datos generados

en tiempo de ejecución, sino que también radican en la memoria RAM registros propios del microcontrolador, registros de configuración, registros de estado... Se puede decir que la CPU se comunica con nosotros a través de estos registros, que son los llamados SFR (Registro de Funciones Especiales).

Stack de memoria o pila

El stack es memoria RAM donde el controlador guarda datos temporales de uso propio de la CPU. No depende del programador, es manejado por el controlador, y tiene en la serie 18 de PIC 31 niveles de profundidad —lo que mejora notablemente el anidamiento de interrupciones, que se verá más adelante—.

Tradicionalmente, la pila o stack solo se ha utilizado como un espacio de almacenamiento para las direcciones de retorno de subrutinas o rutinas de interrupción, donde las operaciones para meter y sacar datos de la pila están escondidas.

En su mayor parte, los usuarios no tienen acceso directo a la información en la pila.

El microcontrolador PIC18 se aparta un poco de esta tradición. Con el nuevo núcleo PIC18, los usuarios tienen acceso a la pila y se puede modificar el puntero de pila y la pila de datos directamente. Dichos niveles de acceso a la pila permiten algunas posibilidades de programación únicas e interesantes.

La memoria EEPROM

Es una memoria permanente como la ROM, pero a diferencia de esta, se puede escribir en ella, se puede borrar y, desde luego, se puede leer. No todos los microcontroladores tienen EEPROM para guardar variables de programa. Un ejemplo de uso podría ser un sistema de alarma donde el usuario guarda su clave de acceso y no queremos que la clave se pierda si se desconecta la fuente de poder; además, el usuario puede cambiar la clave cuando quiera, ya que puede escribir en ella a voluntad.

El PIC18F4550 dispone de 256 bytes en EEPROM.

La memoria ROM

En ella se encuentran grabadas, entre otras cosas, las instrucciones del microcontrolador. Fueron puestas ahí por Microchip fabricante del PIC (son 77 en el PIC18F4620) y básicamente son la esencia de lo que el micro puede hacer. La ROM es una memoria permanente, no se puede escribir en ella ni borrar, solo se puede leer. Es el propio microcontrolador en funcionamiento quien lee las instrucciones ahí escritas. Esta memoria es transparente al usuario y el programador solo accesible por la CPU.

La memoria Flash

Es una memoria permanente, como la EEPROM. En realidad, es una evolución de esta. Es más rápida que la EEPROM y, si bien es de borrado y de escritura eléctrica como la EEPROM, en la FLASH las cosas son diferentes. Una memoria EEPROM puede ser borrada por palabras, es decir, por bytes; se puede acceder a una EEPROM y alterar solo un byte dentro de la memoria, o bytes en distintas regiones de la memoria. En cambio, la FLASH solo se escribe o se borra por páginas; es decir, está paginada y los datos en ella se procesan por páginas. El tamaño de la página en bytes depende del tipo de memoria (64, 128, 256, etc.). En un microcontrolador, guardaremos nuestro código y nuestro programa en la memoria FLASH; es la memoria de programa. El tamaño de esta memoria determina el tamaño de nuestro programa. Por ejemplo, para el PIC18F452, esta memoria tiene un tamaño de 32K (32 768 bytes); si nuestro programa requiere más memoria, deberemos pensar en un microcontrolador más grande.

Los timers

Son los encargados de generar tiempos, muy útiles cuando tenemos programas que hacen tareas en ciclos que se repiten. Por ejemplo,

el PIC18F452 dispone de un timer de 8 bits y de tres timers de 16 bits.

Los conversores A/D

Son los responsables de hacer mediciones y de convertir variables analógicas físicas en datos digitales. Podemos medir temperatura, peso, etc. y cuantificar estos valores en el mundo digital del microcontrolador. El PIC18F452 dispone de 8 canales analógicos de 10 bits cada uno.

Registros SFR

Un registro no es otra cosa que una posición de memoria que contiene un dato. Los registros son vectores o posiciones de memoria RAM *y* son de vital importancia para el programador (también son fuente de muchos dolores de cabeza). A través de estos registros podemos conocer el estado interno del microcontrolador, qué está haciendo y cómo es su estado general. A través de estos registros, la CPU dialoga con el programador y este puede interactuar con la propia CPU activando bits de estos registros que determinan acciones dentro del controlador.

Es frecuente que este sea uno de los puntos más oscuros que tiene la programación en estos niveles, ya que se requiere un conocimiento fino de la arquitectura interna del controlador para saber interpretar la información de estos registros y enviar hacia la CPU acciones concretas.

El programador también puede crear sus propios registros, lugares de memoria que serán destinados a contener las variables que maneje el programa.

Los registros SFR son propiedad de la CPU y no deben ser confundidos con los registros o posiciones de memoria del usuario.

ARQUITECTURA DE LOS PIC

El alto rendimiento de los microcontroladores PIC puede ser atribuido a las siguientes características:

- Arquitectura Harvard
- Pipelining de instrucciones
- Gran archivo de registros
- Instrucciones de ciclo simple
- Palabras de instrucción simple
- Palabras de instrucción largas
- Set de instrucciones reducido
- Set de instrucciones ortogonal (simétricas)

Básicamente hay dos enfoques para construir microcontroladores:

Arquitectura Von Neumann:

- Búsqueda de instrucciones y datos sobre una memoria de espacio simple.
- Limitado ancho de banda de operación.

Arquitectura Harvard:

- Uso de dos espacios de memoria separados para instrucciones de programa y datos.
- Mejora el ancho de banda de operación.
- Permite diferentes anchos de buses.

Opciones del oscilador

XT	Standard frequency cristal oscillator	100kHz-4MHz
HS	High frequency cristal oscillator	DC-40MHz

HS+PLL	High frequency crystal with 4x PLL	4MHz-10MHz
LP	Low frequency crystal oscillator	5kHz-200kHz
RC	External RC oscillator	DC-4MHz
RCIO	External RC oscillator, OSC2=RA6	DC-4MHz
INTRC	Internal RC oscillator	Various
EC	External Clock, OSC2=fosc/4	DC-40MHz
ECIO	External Clock, OSC2=RA6	DC-40MHz

Tabla 1.1 Posibles configuraciones para el oscilador.

Las opciones de reloj seleccionable proveen mayor flexibilidad para el diseñador:

- Oscilador LP diseñado para absorber menor cantidad de corriente.
- RC o INTRC, provee una solución de coste ultra bajo para el oscilador.
- XT optimizado para osciladores de las frecuencias más usadas.
- HS optimizado para manejar resonadores y cristales de alta frecuencia.

Modo Sleep

El procesador puede ser puesto en modo de baja potencia con la ejecución de una instrucción Sleep:

- El oscilador del sistema se detiene.
- El estado del procesador se mantiene (diseño estático).

- El funcionamiento del WDT continúa, si está habilitado.
- Mínima absorción de corriente, sobre todo debido a las salidas (0.1 – 2.0μA típicos).

Events that wake processor from sleep	
MCLR	Master Clear Pin Asserted (pulled low)
WDT	Watchdog Timer Timeout
INT	INT Pin Interrupt
TMR1	Timer 1 Interrupt (or also TMR3 on PIC18)
ADC	A/D Conversion Complete Interrupt
CMP	Comparator Output Change Interrupt
CCP	Input Capture Event
PORTB	PORTB Interrupto n Change
SSP	Synchronous Serial Port (I2C Mode) Start/Stop Bit Detect Interrupt
PSP	Parallel Slave Port Read or Write

Tabla 1.2 Eventos que interrumpen el modo Sleep.

En la tabla anterior se aprecian las formas de sacar el microcontrolador del estado de Sleep.

ENTORNO DE TRABAJO MIKROC

Se puede descargar directamente desde el sitio oficial de Mikroelektronika, y en su versión libre permite generar código con tamaño limitado. Todo se realiza desde el mismo IDE (ensamblador, enlazador, gestión de proyectos y depurador).

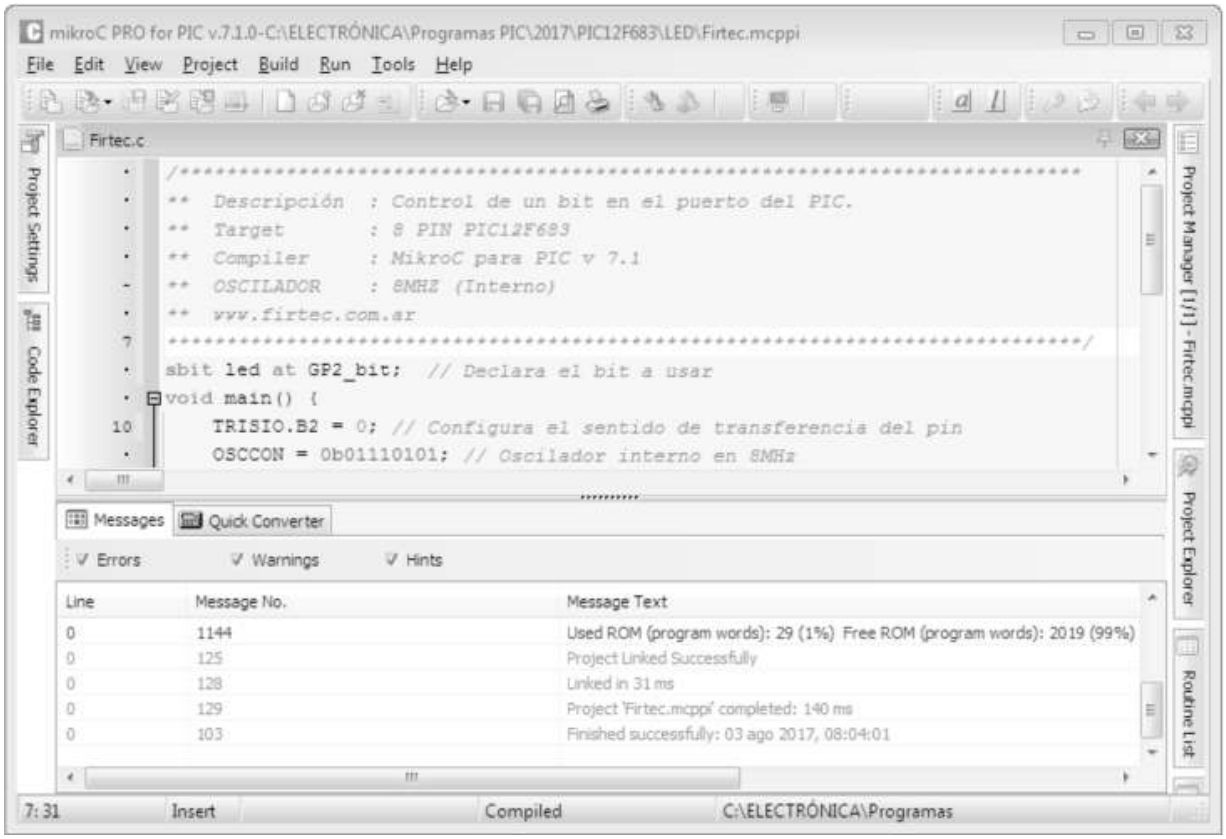


Figura 1.1 Aspecto del IDE para MikroC PIC.

La interfaz gráfica de usuario sirve como un único entorno para escribir, compilar y depurar código para aplicaciones con PIC.

Desde el IDE también se inicia el proceso de programación, siempre que la compilación haya resultado exitosa.

Es simple, intuitivo y cómodo de usar. Desplegando en una serie de pestañas, veremos todas las herramientas de que dispone.

Una particularidad interesante que tiene MikroC es la gran cantidad de bibliotecas que ofrece ya listas para usar, que se

encuentran incorporadas en el propio entorno de trabajo. Al marcar la biblioteca, las funciones de control están listas para usarse.

Ofrece rutinas para el manejo de una pantalla LCD, una comunicación UART, I2C, SPI, incluso bibliotecas para el manejo del ADC o 1-wire, entre muchas más opciones.

Cada biblioteca cuenta con una ayuda que explica en detalle las funciones disponibles y su funcionamiento.

Las bibliotecas se descargan como paquetes de software y se instalan con un instalador especial, una herramienta que también podemos descargar de manera libre desde el sitio de Mikroelektronika.

INSTALADOR DE PAQUETES DE MIKROELEKTRONIKA

El instalador de paquetes es una herramienta que nos permitirá tanto instalar como desinstalar software del entorno de MikroC.

Para instalar un paquete solo debemos descargarlo y, luego, con el instalador de paquetes instalarlo. Una vez instalado, ya lo tendremos en el administrador de librerías.

Para desinstalarlo también se usará el administrador de paquetes.

Hay varias herramientas interesantes que podemos descargar desde el sitio de Mikroelektronika, que funcionan incluso en distintas arquitecturas, ya que hay varios compiladores para distintos lenguajes y arquitecturas.

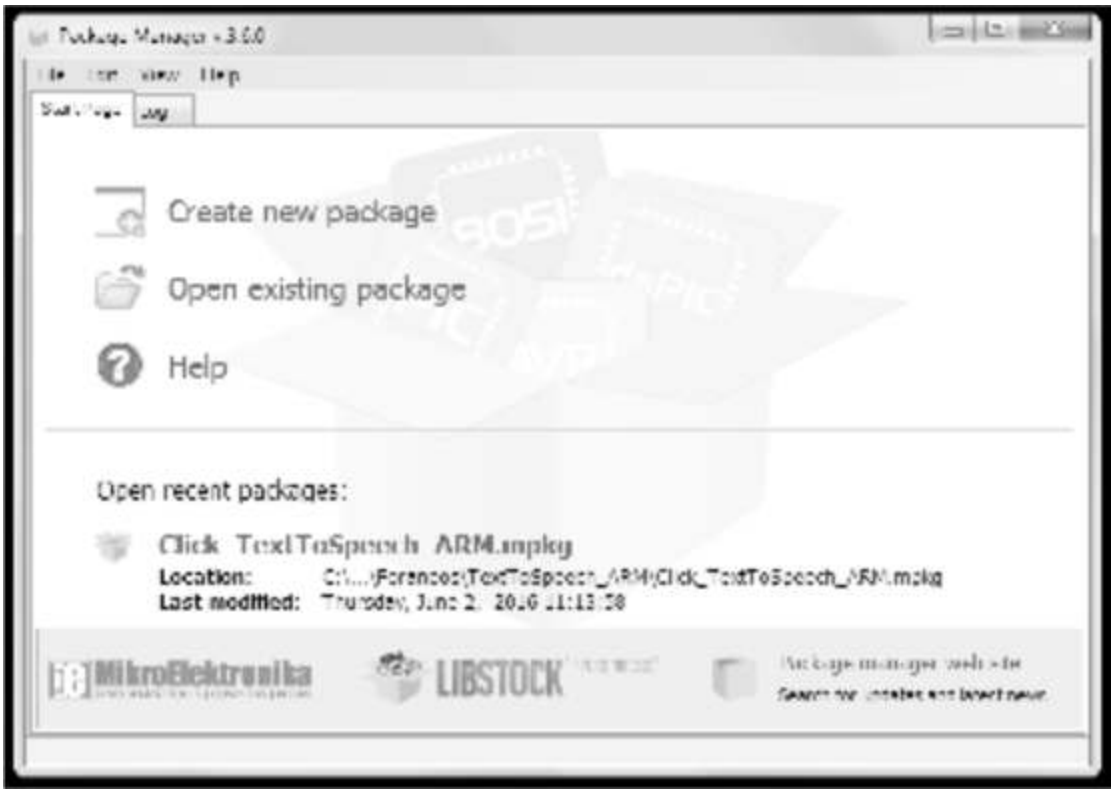


Figura 1.2 Instalador de paquetes.

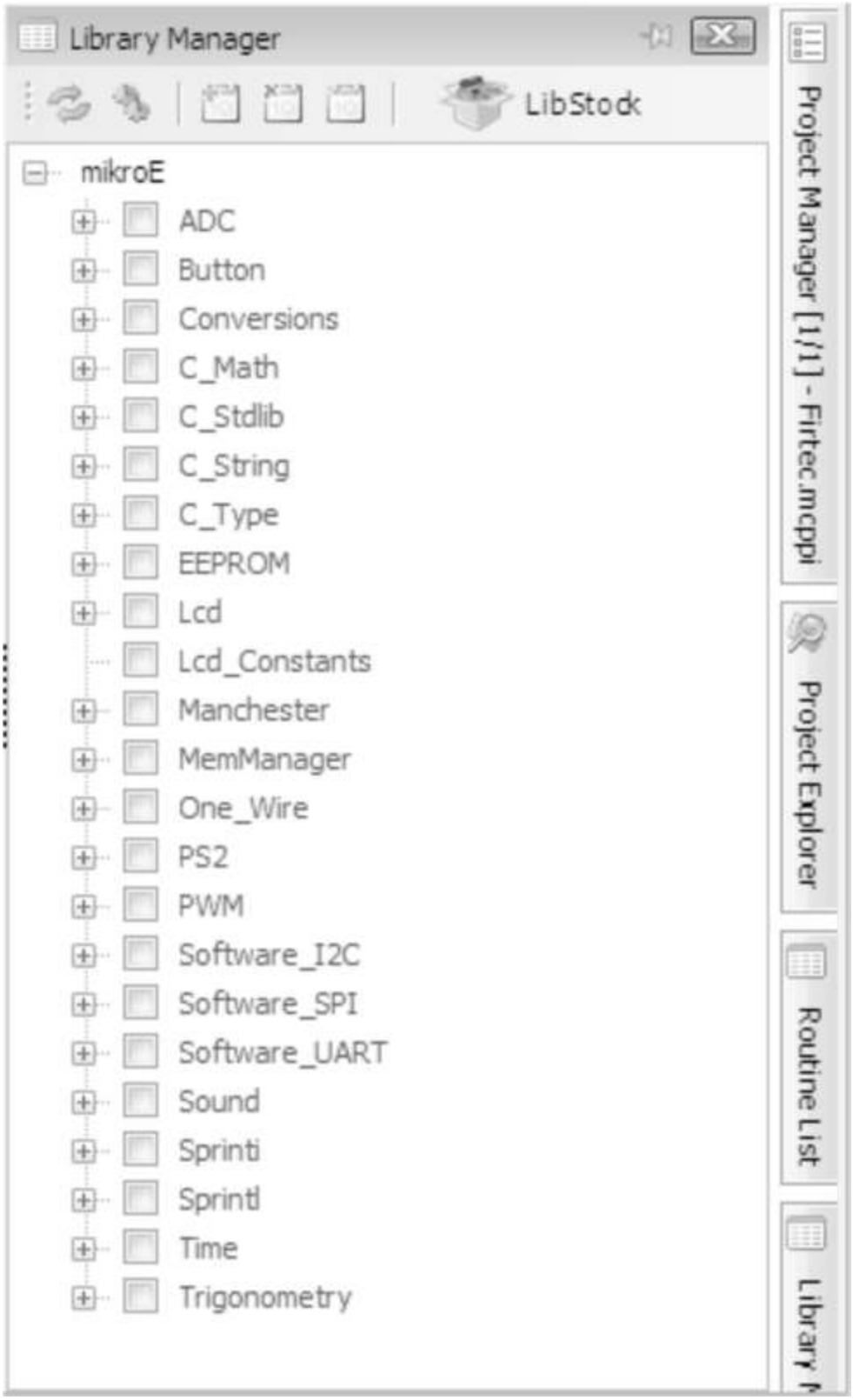


Figura 1.3 Bibliotecas que se instalan por defecto en MikroC.

Al marcar la casilla correspondiente a la biblioteca que vamos a usar, las funciones para el manejo del hardware quedan disponibles para ser implementadas.

PROGRAMACIÓN EN LENGUAJES DE ALTO NIVEL

Por lo general, quienes se inician en la programación en lenguajes de alto nivel (como ANSI-C) vienen del mundo del ensamblador. La primera observación —o crítica— que se escucha es la abstracción del hardware, es decir, el hecho de que con el ensamblador el contacto con el microcontrolador a nivel de su hardware interno es más directo. El programador interactúa directamente con sus registros y bits de configuración. Sin embargo, el ensamblador tiene algunas desventajas:

- Incluso una sola operación en el programa escrito en ensamblador requiere varias instrucciones, lo cual hace que el programa sea muy largo y complicado de seguir y mantener.
- Cada tipo de microcontrolador tiene su propio ensamblador, por lo que un programador tiene que conocerlos todos para poder escribir un programa. Esto dificulta la migración del código a otras arquitecturas.
- Para escribir un programa, un programador tiene que conocer el hardware del microcontrolador, sus registros, el mapa de memoria y los bits de configuración.

Los lenguajes de programación de alto nivel, como C, tienen el propósito de superar las desventajas del ensamblador.

Por ejemplo, varias instrucciones en ensamblador se sustituyen por una función, y el programador no necesita conocer el conjunto

de instrucciones o características del hardware del microcontrolador utilizado.

Es verdad que ya no es posible conocer exactamente cómo se ejecuta cada comando, pero esto ya no importa; lo que sí importa es que el trabajo se cumpla de manera efectiva y eficiente. Sin embargo, siempre se puede insertar en el programa una secuencia escrita en ensamblador.

Otro punto son las limitaciones que existen en las funciones matemáticas para PIC de 8 bits; por ejemplo, multiplicar dos números. Aunque se pueden realizar operaciones complejas dividiendo el problema en un gran número de operaciones más simples. La multiplicación se puede sustituir por una secuencia de adiciones sucesivas: $a \times b = a + a + a + \ldots + a$.

En lenguajes como C, para multiplicar los números a y b basta con escribir a*b, y el compilador encontrará automáticamente la solución a este problema y a otros similares.

LENGUAJE C

C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B. Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas. En la actualidad, es la principal herramienta de programación para electrónica y para el desarrollo de sistemas embebidos con microcontroladores.

Se trata de un lenguaje débilmente tipificado de medio nivel, pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel, pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel trabajando directamente con el hardware.

La primera estandarización del lenguaje C fue en ANSI o ANSI C, y uno de los objetivos de diseño del lenguaje C es que solo sean necesarias unas pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que haga falta un soporte intenso en tiempo de ejecución.

En consecuencia, el lenguaje C está disponible en un amplio abanico de plataformas (seguramente más que cualquier otro lenguaje). Además, a pesar de su naturaleza de bajo nivel, este lenguaje se desarrolló para incentivar la programación independiente del hardware.

Este es uno de los puntos más interesantes que tiene este lenguaje para electrónica, dado que un programa escrito que cumpla los estándares ANSI C puede ser portado a cualquier arquitectura de microcontroladores que lo soporte.

Si bien C se desarrolló originalmente (juntamente con el sistema operativo Unix, con el que ha estado asociado mucho tiempo) por programadores y para programadores, ha alcanzado una popularidad enorme, y se ha usado en contextos muy alejados de la programación de sistemas para la que se diseñó originalmente. Se ha vinculado a la electrónica de manera creciente desde hace ya varios años, y hoy es un lenguaje genérico para la programación de microcontroladores.

En 1978, Ritchie y Brian Kernighan publicaron la primera edición del libro *El lenguaje de programación C*, también conocido como *La biblia de C*. Este libro fue, durante años, la especificación informal del lenguaje. C es un lenguaje compilado, lo que significa que el compilador traduce los archivos fuente que contienen el código C a lenguaje máquina. Todas estas características hicieron que fuera uno de los lenguajes de programación más populares. Una vez compilados los archivos, se genera un archivo hexadecimal que finalmente será el que termine en la memoria de programa del microcontrolador.