Benemérita Universidad Autónoma de Puebla.

Facultad de Ciencias de la Computación.

"Elementos para el desarrollo de Sistemas Embebidos: Un prototipo"

Tesis Profesional que para obtener el título de Licenciado en Ciencias de la Computación.

Presenta:
Oscar Villanueva Celis

Asesor:
M. C. Alfonso Garcés Báez

Puebla, Pue. **Octubre 2008.**

Agradecimientos

A mi familia “Mi padre, Mi madre y mi hermana”, su apoyo fue fundamental para conseguir este objetivo.

Al maestro Alfonso Garcés Báez, por su paciencia y apoyo en el aula y fuera de ella.

A los maestros Marcos González y Consuelo Molina, por sus buenos consejos.

A mis amigos por su apoyo y complicidad en nuestra estancia en la facultad.

A todas las personas que siempre me preguntaban “¿Ya te titulaste?, ¿Para cuándo el titulo?”.

Índice.

Objetivos 1

Introducción 2

Capitulo 1. Antecedentes y Proyecciones 4

1.1. Puntos Tecnológicos a corto y mediano plazo 9
1.2. Mercado Global 10
1.3. Mercado Local 12
1.4. Capacidades Locales 13
1.5. Actores Claves 14
1.6. Objetivos a corto y mediano plazo en el País 14
1.7. Factibilidad de alcanzar los objetivos de desarrollo 18

Capitulo 2. Conceptos Básicos y Definiciones 20

2.1. Definición Sistema Embebido 20
2.2. ¿Para qué sirven los Sistemas Embebidos? 20
2.3. Componentes de un Sistema Embebido 20
2.3.1. Microprocesadores 21
2.3.2. Microcontroladores 21
2.3.3. DSP 22
2.3.4. Memoria 22
2.3.4.1. CMOS RAM 22
2.3.5. Periféricos de E/S 23
2.3.6. Buses 23
2.3.6.1. Bus de Control 23
2.3.6.2. Bus de Direcciones 24
2.3.6.3. Bus de Datos 24
2.4. Componentes Auxiliares 24
2.4.1. Puerto Serie RS232 24
2.4.2. SPI 24
2.4.3. I^2C 25
2.4.4. USB 25
2.4.5. Bluetooth 25
2.4.6. Ethernet 26
2.4.7. GSM 26
2.4.8. GPRS 26
2.4.9. Puertos Digitales 26
2.4.10. Puertos Analógicos 27
2.5. Software Embebido 27

Capitulo 3. Arquitecturas Sistemas Embebidos 30

3.1. Arquitectura Harvard 30
3.2. Arquitectura ARM 32
3.3. Arquitectura MIPS 34
3.4. Arquitectura PowerPC 35
3.5. Arquitectura SH 38
3.6. Arquitectura CompactRIO 42

Capitulo 4. Software Embebido 44

4.1. Características del software embebido 45
4.1.1. Temporalidad 45
4.1.2. Concurrencia 45
4.1.3. Vivacidad 46
4.1.4. Interfaces 46
4.1.5. Heterogeneidad 46
4.1.6. Reactividad 46
4.2. Metodologías de Ingeniería de Software 47
4.2.1. Procedimientos y orientación a objetos y agentes 47
4.2.2. Orientación a objetos en tiempo real 47
4.2.3. Diseño de Hardware 47
4.2.4. Sistemas operativos y lenguajes en tiempo real 47
4.3. Principales necesidades en el desarrollo de software embebido 48
4.3.1. Ingeniería de Computación 48
4.3.2. Ingeniería en Electrónica 48
4.3.3. Ingeniería de Procesos 48
4.3.4. Herramienta de diseño de software embebido 48
4.4. Metodologías para el Desarrollo de un sistema embebido con características hard real-time 49

Capitulo 5. Programando un Microcontrolador 53

5.1. Basic como una herramienta de trabajo 53
5.2. ¿Por qué utilizar Basic? 54
5.3. Desarrollando “Hola Mundo” 54
5.4. Variables, Instrucciones y Declaraciones en Basic 56
5.5. Manipulando un Modulo LCD con Basic. 59

Capitulo 6. Desarrollando un Prototipo 65

6.1. Esquemas de las principales herramientas a utilizar 65
6.2. Analizando las conexiones de las herramientas 66
6.3. El Código y su análisis 67
6.4. La simulación Final 69

Conclusión 72

Bibliografía 73

Apéndice I 75

Objetivos

Objetivo General.

Presentar una introducción al tema de los Sistemas Embebidos, para despertar el interés en la investigación y desarrollo de los mismos, tomando como base el presente trabajo.

Objetivo Particular.

Dar a conocer los principales componentes, características y herramientas de desarrollo de los Sistemas Embebidos, al igual que se expone la situación y proyección de este tipo de tecnología.

Se finalizara con el desarrollo de un prototipo de un Sistema Embebido utilizando herramientas como PIC Simulator IDE y el compilador MikroBasic que serán mencionadas en el desarrollo del presente.

Introducción

Un Sistema Embebido básicamente es un sistema de auto contenido que posee un microcontrolador programable y que desarrollan una o más funciones específicas.

Este tipo de Sistemas se encuentran en cada momento de nuestras vidas, el horno de microondas, el auto, el equipo de audio, la tv. Los Sistemas Embebidos a pesar de no ser muy nombrados están en casi todas partes, de hecho, es difícil encontrar algún dispositivo cuyo funcionamiento no este basado en un sistema embebido.

La falta de información y conocimiento sobre estos sistemas han provocado que existan muy pocas investigaciones y desarrollos de estos sistemas en el país. De ahí el problema que trata de solucionar el presente dando a conocer varios aspectos de estos sistemas.

Lo primero que se aborda en el capítulo 1 es presentar las expectativas que se tienen en el mercado internacional en el desarrollo de estos sistemas, algunos programas que se enfocan al tema en Europa y en México así como las posibilidades que tiene México para incursionar en todos los aspectos relacionados con esta tecnología.

En el capítulo 2 se presentaran de igual manera los componentes de los Sistemas Embebidos haciendo referencia a los SOC (System On Chip) que son componentes que ya vienen armados desde su fabricación con algunos componentes como son los puertos Ethernet, Seriales, USB, etc. Y algunos componentes auxiliares que se pueden integran a estos sistemas.

Existen distintas arquitecturas que son utilizadas en el desarrollo de los sistemas embebidos, en el capítulo 3 se mostraran algunas de las más comunes así como su área de uso, entre las más comúnmente utilizadas se encuentran Harvard, ARM, x86, MIPS, SuperH y PowerPC entre otras. También se encuentran nuevas arquitecturas desarrolladas por empresas que requieren un funcionamiento especial de los sistemas, la más conocida de este tipo de arquitecturas es la CompactRIO.

Las principales características que debe tener el software embebido se presenten en el capítulo 4 de igual manera algunas metodologías para el desarrollo de este.

Después de revisar las características que debe cumplir el software embebido, en el capítulo 5 se da una introducción al lenguaje Basic orientado al ambiente embebido, de igual manera se presentaran los primeros ejemplos, desarrollados en el lenguaje Basic y C, se revisaran algunas de

las principales instrucciones y su sintaxis, finalizando con la demostración de las instrucciones para la manipulación de un modulo LCD.

Finalmente en el capítulo 6 se desarrolla una simulación de un prototipo de sistema embebido, el cual entre sus principales componentes utilizara un microcontrolador con arquitectura Harvard, un conjunto de 8 leds y un modulo LCD. La función que realizara a grandes rasgos será que después de realizar un conteo mostrara un par de letreros en el modulo LCD y realizara un parpadeo de los leds.

Capitulo 1

Antecedentes y proyecciones

Capítulo 1
Antecedentes y Proyecciones.

La falta de información formal sobre los sistemas embebidos motiva la realización de esta investigación en donde se expondrán algunos puntos importantes para el desarrollo de estos como lo son sus componentes, las arquitecturas, algunas metodologías para el desarrollo de software embebido y algunas herramientas para su programación, en especifico para el desarrollo de un prototipo final se programará bajo el lenguaje Basic, utilizando una herramienta de simulación de Microcontroladores la cual será PIC Simulator IDE. Comenzando por presentar las capacidades e iniciativas que se presentan en el mundo para el desarrollo de esta tecnología.

Podemos definir al sistema embebido como un subsistema electrónico de procesamiento, programado para realizar una o pocas funciones para cumplir con un objetivo específico. Generalmente es parte integral de un sistema heterogéneo mayor, que puede incluir partes mecánicas, eléctricas y/o electromecánicas. Por el contrario, un sistema de procesamiento de propósito general, tal cual una computadora personal, puede realizar diferentes tareas dependiendo de la programación.

En la actualidad, los sistemas embebidos forman parte de la vida cotidiana de todos. La mayoría de los consumidores, a los que la palabra computadora les trae la imagen de una PC no tienen información de la gran cantidad de procesadores o microcomputadoras que forman parte importante de la vida diaria de todos. Se despiertan con la alarma del reloj digital, preparan el desayuno en un microondas, conducen automóviles asistidos por computadoras de abordo, realizan estudios médicos utilizando instrumentos biomédicos como los tomógrafos, monitores cardíacos y ecógrafos. El amplio rango de aplicaciones abarca desde entretenimiento y confort hasta vigilancia, seguridad, salud y protección al medio ambiente.

Muchos ejemplos mas que de una larga lista de los sistemas embebidos son la mayoría de los periféricos de la computadora, los teléfonos celulares, routeadores, PDAs, los que manejan las operaciones que se realizan en los cajeros automáticos, los que están contenidos en los discos duros portables, misiles, aviones y distintos tipos de transporte, instrumentos de medición y control, etc.

Existen un importante número de aplicaciones de sistemas embebidos para la reducción del consumo de recursos naturales: calderas inteligentes usan la mínima energía para mantener confortable la temperatura del ambiente; sistemas de riego programables hacen uso del agua en horarios y períodos convenientes. La mayoría de los procesos industriales confían en algún tipo de control computarizado para optimizar el uso de la energía y alcanzar las regulaciones internacionales de protección ambiental ***[i3]***.

Un sistema embebido está compuesto por circuitos integrados programables, memoria flash o ROM, el correspondiente circuito impreso y el software embebido o empotrado como parte esencial del mismo sistema, conocido en inglés como firmware o embedded software. El software embebido se utiliza para controlar los productos electrónicos y usualmente se ejecuta sobre un microprocesador interno, o en un microcontrolador, o en un procesador digital de señal (DSP), o en una compuerta programable en campo (FPGA), o en un controlador lógico programable (PLC) y a veces en una PC de propósitos generales adaptada para fines específicos ***[i7]***.

Este software difiere del software convencional de una computadora de propósitos generales en una serie de características que justifica considerarlo como un nuevo campo de investigación y desarrollo dentro de las tecnologías de la información. Tiene entre otras las siguientes características:

- Tiene una interfaz directa con el hardware del dispositivo y es el intermediario entre el software de alto nivel y las funciones del hardware. Su lenguaje de programación, en la mayoría de los casos es de bajo y mediano nivel.

- Debido a que un sistema embebido está dedicado a una tarea específica, el diseño puede ser optimizado reduciendo los costos, el tamaño del producto y el consumo de potencia, a la vez de incrementar la confiabilidad y eficiencia.

- Los sistemas informáticos incluidos en productos electrónicos que controlan por ejemplo fábricas, tráfico aéreo y la distribución eléctrica se denominan sistemas de tiempo real. Los sistemas de tiempo real se diferencian de los sistemas informáticos de aplicación general en que deben cumplir con requisitos de tiempo que no sólo implica velocidad de respuesta, sino garantía de acción en el instante requerido de acuerdo a lo especificado. El software consiste en un programa que realiza tareas específicas el cual puede incluir un sistema operativo en tiempo real (RTOS).

La naturaleza dedicada en tiempo real del sistema conduce a un mayor grado de dependencia y a una mayor integración con el hardware. Son una combinación de hardware y software en un mismo paquete. Sobre el principio de esta interrelación software-hardware es que se basa, en forma creciente, la producción de equipos electrónicos de nueva tecnología.

Para alcanzar la meta de desarrollar sistemas embebidos eficientes, es necesario emplear sistemas de arquitectura apropiados, hardware de interfaces y dispositivos periféricos, sensores e implementar robustos programas de software para su control. Todos ellos son utilizados en equipos y sistemas electrónicos que requieren el co-diseño integrado de Hardware y Software.

- Generalmente se ejecuta en un hardware limitado tanto en velocidad como en cantidad de memoria.

- En numerosos casos requieren capacidad de auto-prueba, tanto del software como del hardware.

Típicamente, el software debe ser extremadamente confiable, muy eficiente y compacto, y muy preciso en su respuesta al no siempre predecible instante de la transmisión de la información de ingresos y salidas (Interfaces I/O.) Su tolerancia a fallas es muy baja, porque una vez en manos del usuario en la mayoría de los casos es muy difícil o imposible de realizar cambios.

Dentro del Software Embebido se pueden distinguir a su vez diferentes categorías:

El Software original o básico, indispensable para el funcionamiento del dispositivo, el cual constituye su sistema operativo ad-hoc. El lenguaje de programación es Ensamblador, C/C++ o VHDL. Este tipo de Software, de carácter eminentemente tecnológico, está incorporado en el dispositivo desde la salida al mercado de este ***[i1]***.

En general, este Software requiere un elevado esfuerzo de creación inicial, a cargo de mano de obra de muy alta especialización. Usualmente su costo se reparte en la cantidad de equipos entregados al mercado. En la mayor parte de los casos no se modifica a lo largo de su vida activa, siendo reemplazado al aparecer un nuevo modelo del dispositivo.

Un segundo tipo de Software es el requerido por algunos de los equipos o aparatos incluidos en el apartado anterior, pero que por condiciones específicas, como pueden ser requerimientos regulatorios de una determinada comunidad, pero también condiciones existentes o simplemente modalidades o costumbres, se hace necesario adaptar el equipo original al uso específico requerido.

Otro tipo de Software de carácter embebido o tecnológico, generalmente insertado en una PC dedicado a este fin específico, es el dedicado a la gerencia, supervisión y control de sistemas complejos cuyo funcionamiento implica la interrelación de gran cantidad de equipos o aparatos. Este Software es, por ejemplo, el que permite el funcionamiento de las redes de comunicaciones de voz, video, audio y datos en la industria de las comunicaciones, y el funcionamiento y control de los distintos procesos en la industria manufacturera.

Dada la posibilidad, muy frecuente en la práctica de que dichas redes estén conformadas por elementos de muy diversa procedencia y tecnología, la creación y optimización de este Software es particularmente crítica y requiere la mayor especialización y recursos de todo tipo.

Características particulares del Software tecnológico embebido, en especial de las dos últimas categorías descritas, son:

- Requiere de mano de obra sumamente especializada, no sólo de carácter informático y electrónico sino de otra con conocimientos profundos del funcionamiento de los diferentes elementos que conforman el entorno en el que debe funcionar el equipo. Incluye alta capacidad para el diseño y desarrollo.

- No es posible el desarrollo por especialistas en una única disciplina.

Además de los programadores, analistas y otros expertos en informática, en mayor medida que el software de aplicaciones generales se requiere la participación de matemáticos, ingenieros especialistas en telecomunicaciones, electrónica**,** transmisión de datos, mecánicos, de procesos, etc.

Por regla general, el software es más complejo y costoso que el hardware, y por tanto, juega un papel primordial en el desarrollo de los sistemas embebidos. El mercado de software embebido crecerá a una tasa de 16% anual, para pasar de $1,600 millones de dólares en el 2004 a $3,500 millones en el 2009. Por comparación, el mercado de hardware embebido crecerá a 14% anual y el de tarjetas a 10% anual. La tabla 1.1 resume esta información; aquí, "hardware" se refiere a circuitos especializados en aplicaciones embebidas, y "tarjetas" a los circuitos impresos donde se montan los circuitos.

	2004	2009	% de crecimiento
Software	1,641	3,448	16%
Hardware	40,539	78,746	14.2%
Tarjetas	3,693	5,950	10%
Total	45,873	88,144	14%

Tabla 1.1. La Tabla muestra la División del mercado de sistemas embebidos en software, hardware y tarjetas. Cifras en millones de dólares. Fuente: BCC.

Globalmente, este mercado está repartido como se muestra en la tabla 1.2:

	2004	2009	% de crecimiento	% del total en 2009
América	22,012	36,728	10.8%	41.7%
Europa	7,648	14,038	12.9%	15.9%
Japón	7,394	13,160	12.2%	14.9%
Asia	8,819	24,218	22.4%	27.5%
Total	45,873	88,144	14%	

Tabla 1.2. La Tabla muestra la División del mercado de sistemas embebidos por zona geográfica. Cifras en millones de dólares. Fuente: BCC

Es un factor común en la gran mayoría de las predicciones que el mercado asiático es el de mayor crecimiento.

1.1. Puntos tecnológicos a corto y mediano plazo.

- La opinión de los expertos, tanto a nivel nacional como internacional, es que la creación de nuevos componentes microelectrónicos y nanoelectrónicos, microprocesadores multinúcleos, sensores y actuadores tipo MEMs (Microelectromechanical Systems), en general con mayor velocidad de procesamiento , mayor densidad de dispositivos y nueva arquitecturas continuará en la próxima década, dando lugar a la aparición de nuevas aplicaciones y al reemplazo de funciones que hoy se ejecutan en forma mecánica o electromecánica, lo que creará oportunidades para el ingreso de nuevos actores, pero también peligros para aquellos productores que sigan utilizando tecnologías más maduras en sus productos y no se adapten a los cambios.

- Los continuos avances en el dominio de la física, la química, la biología, la mecánica las comunicaciones y la electrónica entre otras disciplinas, permitirán desarrollar productos cada día más complejos como una composición tecnológica de sistemas embebidos los cuales estarán especialmente concebidos para producir resultados significativos en términos de desempeño, calidad y costo en su área específica de aplicación.

- Los sistemas embebidos son y serán un componente crucial en la mayoría de los dispositivos electrónicos incluyendo los productos de consumo diario o los equipos más complejos. Las comunicaciones, los equipos de monitoreo, supervisión y control industrial, equipamientos electro médicos, industria automotriz y del transporte son sólo algunos de los ejemplos de su utilización. En general, una creciente cantidad de equipos de uso diario

serán controlados por software y a su vez estos mismos equipos, crecerán constantemente en prestaciones.

- Por su lado, la propia industria electrónica seguirá cambiando, requiriendo para ello modelos y estructuras de diseño que permitan atender ciclos de vida de producto cada vez más reducidos. Así como el microprocesador y otros elementos programables significaron un cambio profundo en la concepción y el diseño electrónico al integrar Hardware y Software pasando de la lógica cableada a la programada, la creciente utilización de componentes versátiles y reconfigurables como los FPGA significa el cambio hacia un hardware que pueda evolucionar al paso de los requerimientos tecnológicos conservando sus atributos de eficiencia, robustez y costo competitivo. Si a ello se agrega la creación en el país de una mayor capacidad en el diseño de circuitos integrados de aplicación específica y de sensores y actuadores se puede concluir que se crearán oportunidades de generar innovaciones tecnológicas en nichos de mercado.

- A pesar de su enorme inserción en todos los órdenes de la actividad y productiva de la sociedad, el software embebido tiene un atraso relativo respecto del de aplicación general en PCs. y de los grandes sistemas de cómputo donde se alcanzan eficiencias productivas mayores. Esta menor productividad que existe a nivel mundial representa una oportunidad para países como el nuestro, pues todavía se trata de una tarea artesanal, no automatizada, donde la creatividad y la capacidad innovadora son un factor competitivo de gran importancia. Desde el punto de vista del software, resulta imprescindible el desarrollo de metodologías que permitan un tratamiento del sistema en forma modular pero manteniendo una fuerte coherencia con el dominio del problema. De esta manera, cada una de las partes interactúa con las restantes, teniendo el menor solapamiento funcional. Claramente, esta especialización es en si misma el paradigma de los sistemas embebidos.

- Las nuevas arquitecturas de los procesadores generan capacidades que deben ser consideradas y aprovechadas para la eficiencia del desarrollo. Por ejemplo la irrupción del paralelismo de los microprocesadores es un factor que permite incrementar sustancialmente la potencia del cómputo y eficiencia del sistema y por ende debe ser incorporado en el codiseño del sistema embebido.

1.2. Mercado global.

Cada vez más productos electrónicos diferentes de las computadoras, incorporan programas dentro de sí (software embebido). Éstos en general no aparecen en las estadísticas de producción de software y servicios informáticos, sino que quedan incluidos y sin diferenciar en los datos

usuales de la producción de bienes y servicios electrónicos (constituida hoy en forma dominante por las aplicaciones de la microelectrónica), por lo que no es posible distinguir dicho contenido a partir de las estadísticas usuales. Por ello en general se recurre a datos indirectos para visualizar su importancia.

- La Unión Europea a través de Cordis publicó el resultado de estudios en los que se informa que el mercado de sistemas embebidos (constituido por circuitos integrados programables, software e impresos pero donde no está incluido el valor agregado a ellos para lograr el producto final) pasará de una facturación de 46.000 millones de dólares en el año 2004 a una estimada de $ 88.000.millones de dólares en el año 2009, con un crecimiento del 14% anual. Para el mismo período se estimaba un crecimiento del 8% en la producción de PCs y estaciones de trabajo ***[i2]***.

- En el año 2002 se habían vendido más de 6.000 millones de procesadores de todos los tipos (4 a 64 bits, incluyendo DSPs). Esta gran cantidad es, inclusive, un 25% inferior, debido a la crisis económica mundial, al récord de ventas de 8.000 millones alcanzados dos años antes. De estos 6.000 millones, solamente un 2% pasaron a ser el cerebro de computadoras (PCs, Macs, y estaciones de trabajo Unix), mientras que el 98% restante pasó a formar parte de sistemas embebidos, como producto final o bien intermedio.

- Otro ejemplo interesante, sobre todo teniendo en cuenta la importancia que tiene en nuestro estado, es su incidencia en la industria automotriz. En 1990 el costo de la electrónica por auto, en Europa, era de 940 dólares, mientras que en el año 2005 era de 1.720 dólares. En un automóvil moderno no se puede pensar en redes de menos de 30 procesadores controlando sus funcionalidades, cada nodo controlado por su propio software embebido. En el mencionado estudio de la Unión Europea publicado en Cordis sobre el "Futuro de los sistemas embebidos", se informaba que en el valor final de un automóvil en Europa (excluido impuestos) la contribución de la electrónica (en gran proporción sistemas embebidos) pasaba del 20% en el 2004 al 36% en el 2009.

Debido a su creciente importancia, los gobiernos de los países desarrollados realizan esfuerzos especiales para promover su desarrollo, lograr una inserción mayor en este mercado e incrementar su capacidad tecnológica en este sector de creciente valor estratégico.

El programa de la Unión Europea para el desarrollo de la Tecnología de la Sociedad de la Información (IST programme) pone un especial énfasis en el desarrollo de sistemas embebidos creando un grupo de trabajo denominado: "Tecnologías del Software, Sistemas Embebidos y Sistemas Distribuidos" en su estrategia de desarrollar lo que ellos denominan "Ambient Intelligent environment". Este ambiente inteligente (AmI) involucra sistemas directamente bajo el control

humano y también sistemas que controlan el entorno del ser humano que no estén controlados directamente por él. AmI involucrará enormes y complejos sistemas distribuidos que requerirán grandes desarrollos de aplicaciones de software y de la infraestructura para desarrollarlo en forma eficiente.

El programa EUREKA-ITEA es otra iniciativa europea para promover la investigación para el desarrollo de estos tipos de sistemas.

Por otro lado, en los Estados Unidos, programas como "Embedded, Everywhere" o iniciativas como la de la IEEE creando el nuevo foro sobre "Pervasive Computing", o la organización del evento "Embedded Systems Conference" que anualmente se realiza tanto en la costa este como en la costa oeste, y las múltiples publicaciones especializadas confirman la complejidad y la importancia mundial de los Sistemas Embebidos.

1.3. Mercado local.

En las últimas dos décadas el mercado electrónico del país (que también en forma creciente incluye sistemas embebidos) tuvo un crecimiento importante, aunque constituido en su gran mayoría por productos importados.

La producción electrónica en el país en las últimas décadas se vio disminuida, sin embargo en los últimos años a comenzado a recuperarse, con el apoyo del sector empresarial y gubernamental, con el fin de estar a la par con las nuevas tecnologías a nivel global.

El desarrollo de los sistemas embebidos alcanza ya los 200 mdd en México, mientras que en América del Norte es de 10,000 mdd. Se espera que en nuestro país llegue a los 3,000 mdd. En Australia, incluso, han invertido por lo menos 11,000 mdd para desarrollar este tipo de infraestructura que está creciendo en Europa y Asia.

En 2004, señala FUMEC, el mercado mundial de software embebido se estimaba en mil 600 millones de dólares y se calcula que crecerá a 3 mil 500 millones de dólares en 2009 lo que para esta fundación constituye una oportunidad única para México.

Por ello, FUMEC busca establecer un Sistema Nacional de Fortalecimiento a la capacidad de generar software embebido, incorporando a las pequeñas y medianas empresas tecnológicas de México y espera desarrollar proyectos en áreas como la automotriz, telecomunicaciones inalámbricas, dispositivos médicos así como incluir también al sector agroindustrial ***[i4]***.

En el 2006, el INA (Industria Nacional de Autopartes), FUMEC (Fundación México-Estados Unidos para la Ciencia) y diversas instituciones del país han impulsado la integración de un Consejo Nacional de Articulación Productiva y Desarrollo Tecnológico de la Industria Automotriz en México. Considerando que los Sistemas Embebidos son una tecnología habilitadora que impacta a muchas áreas de la cadena productiva de la industria automotriz, INA y FUMEC se han dado a la tarea de conformar un primer Consorcio de Sistemas Embebidos para dicha industria, que implica la formación de recursos humanos especializados en este tipo de tecnología.

El proyecto fue aprobado por Secretaría de Economía, consiste en la instalación de un Laboratorio de Sistemas Embebidos dentro de las instalaciones del CINVESTAV-Guadalajara. En él se ofrecerán programas de diplomado y especialización cuyos contenidos, desarrollados con el apoyo de CADELEC y CANIETE Occidente, se basarán en la identificación de demandas reales de las empresas tractoras y la generación de productos y servicios asociados a las capacidades a desarrollar ***[i5]***.

El LACISER se funda para unir esfuerzos de investigación en sistemas embebidos y de tiempo real de varias universidades en México respondiendo a las necesidades de la industria de sistemas embebidos en el mercado mundial.

En el 2007 se funda el Consorcio Mexicano de Microsistemas (CMM) que tiene entre sus objetivos más importantes impulsar la producción y desarrollo de sistemas embebidos, los cuales controlan los aparatos para que éstos operen de forma continua, en diversas áreas de la industria en el país.

1.4. Capacidades locales.

Según el Consorcio Mexicano de Microsistemas, hace falta en México personal capacitado para crear este tipo de herramientas tecnológicas, pues estimó que sólo hay 2,000 ingenieros capacitados, por lo que se considero urgente ayudar a las universidades a crear programas que formen a la gente.

Es importante ir desarrollando capacidad en las universidades del país para que más empresas mexicanas trabajen en este tipo de sistemas y para que crezcan los grupos de las empresas extranjeras que trabajan en la aplicación de software de sistemas embebidos en México.

El CMM se apoya en el trabajo que ha venido realizando en los últimos años la Fundación México-Estados Unidos para la Ciencia (Fumec), la cual ha trabajado en el desarrollo de microsistemas como son: Sistemas micro electro-mecánicos (MEMS), micro compuertas de campo programable (FPGAS) y sistemas embebidos, los cuales tienen importantes aplicaciones en los sectores de tecnologías de la información y comunicación, salud, energía, alimentos y automotriz ***[i6].***

1.5. Actores claves.

Se encuentran entre los mencionados en el punto anterior, tomando en cuenta la importancia que para la economía va a tener la micro tecnología para los próximos años, el Ejecutivo Federal incluyó dentro del Plan Nacional de Desarrollo 2007 -2012 ya que la intención de elaborar agendas de trabajo para promover el desarrollo de las micro tecnologías dentro de un grupo de tecnologías precursoras junto con mecatrónica y biotecnología.

En el 2007 la FUMEC, informa la estimación de la necesidad de 30 mil especialistas, en el campo de los sistemas embebidos, entonces lo que se está haciendo es trabajar con el Cinvestav de Guadalajara, que es una institución líder en este tema, con otras universidades como la Universidad Autónoma de Querétaro, el Tecnológico de Monterrey.

1.6. Objetivos en el país a corto y mediano plazo en el país.

En nuestro país existen sectores en los cuales ya la electrónica vinculada a los sistemas embebidos, sea producida localmente o importada, tiene un efecto multiplicador importante, que se incrementará. Ello se verifica en aspectos tales como:

- **Agroindustrias**, en la que la certificación de la trazabilidad de los alimentos, en especial la carne, para superar barreras técnicas a la exportación, implica incorporar chips a los productos, lectores y procesadores de la información. Estos chips, de relativamente baja densidad de integración pero gran volumen de mercado, no sólo pueden diseñarse en nuestro país, sino que luego de ser fundidos en alguna “silicon foundries” del exterior, pueden terminarse de fabricar localmente y luego incorporarse a equipos que procesen la información, desarrollados en hardware y software también en el país.

- **Sistemas de telegestión, telesupervisión y telecontrol de servicios**, producción y recursos naturales tales como: alumbrado público, explotación de pozos petrolíferos y gasíferos, redes de distribución de recursos de energía, áreas pesqueras, apoyo a la producción agrícola ganadera, aseguramiento del mantenimiento de la cadena de frío durante el transporte de mercaderías, etc. Estos temas resultan relevantes en un país tan extenso y con zonas de muy baja densidad de población, que debe optimizar el uso de sus recursos.

- **Sistemas como los descritos en el punto anterior**, y por las mismas razones, para su aplicación en temas de seguridad tales como: alerta temprana de emergencias y posibles catástrofes, vigilancia de fronteras, control de rutas y campos, transporte de sustancias peligrosas, etc.

- **Extensión del acceso a la información y a la comunicación a través de Internet a mayor número de hogares**, aprovechando que el 98% de ellos poseen al menos un receptor de televisión, y que su densidad por hogar es y seguirá siendo mucho mayor que la penetración de las PCs, y su uso más familiar para la mayoría de la población. Ello implicará el desarrollo de una red, el software y los dispositivos de adaptación necesarios, que seguramente llegarán con un costo menor que las computadoras, cuestión importante para personas de menores recursos y otras no interesadas en todas sus aplicaciones pero que requieran del correo electrónico y la navegación por la Web. Esto seguramente se potenciará con la introducción en nuestro país, en los próximos años de la Televisión Digital Interactiva, que es ya una realidad en EE.UU., varios países europeos y algunos latinoamericanos.

- **Los tramites digitales gubernamentales**, como forma de contribuir a la transparencia y optimización de la gestión pública, facilitar y agilizar los trámites de la población. No sólo se requerirá del correspondiente software de aplicación de computadoras y de manejo de bases de datos, sino que su extensión a todo el país y a cantidades crecientes de su población implicará la creación de una infraestructura, nuevamente en hardware y software de redes de comunicación, puntos de acceso y nodos de procesamiento de la información.

- **Redes de información para edificios "inteligentes"**, que proveen un óptimo y confiable servicio en este nuevo mercado, permitiendo compartir el intercambio de datos de sistemas computarizados, monitoreo y control de eventos, máquinas hogareñas y accesorios a través de una red local.

- **Sistemas de pago de auto-servicio,** si bien en el país, por ahora, siendo utilizado por usos bancarios y pago de servicios. Su extensión a sistemas de identificación personal con datos clínicos relevantes y otros del portador de la tarjeta posibilitarán medios adicionales del cuidado de su vida ante una eventual emergencia.

- **Electrónica para el automotor**, creando nuevas aplicaciones pero también reemplazando autopartes electromecánicas actuales.

- En la última década se verificó un rápido crecimiento de soluciones móviles embebidas con gran impacto en el consumidor (por ejemplo a través del uso masivo del celular). Seguramente esto continuará en los próximos años pero no centrado en el consumidor final, sino en el campo de su uso en la producción de bienes y servicios y en el transporte, en el que se verificará una mayor demanda de sistemas autónomos y con conectividad inalámbrica.

Si se quiere tener éxito en algunas de las áreas mencionadas anteriormente, un tema clave será el aumento de las capacidades de los grupos de investigación y desarrollo existentes y la mejora continua, tanto en cantidad como en calidad en la formación de los recursos humanos, a nivel profesional.

Un aspecto importante será la ampliación de las capacidades académicas a nivel de doctorado realizados tanto a nivel local como internacional en temas de modelado, sintetizado y verificación de arquitecturas y sistemas modulares, metodologías de software orientadas a sistemas embebidos, herramientas de integración de sistemas complejos, sistemas de tiempo real, arquitecturas reconfigurables, co-diseño hardware/software/sistema y comunicaciones, entre otras áreas de interés. Para ello resulta necesaria la creación de espacios de trabajo multidisciplinar en las universidades e institutos de investigación con grupos provenientes de las ciencias e ingeniería de hardware y software con una infraestructura de equipamiento de avanzada.

En forma prácticamente simultánea, pero con una mayor dinámica tanto en su concreción como en su financiamiento, se deberá lograr un buen número de especialistas tecnológicos con capacidades directamente volcadas a los sectores productivos, ya sea por iniciativa propia o como consecuencia de la transformación y modernización de la industria electrónica existente.

Cabe mencionar las particularidades en el desarrollo de los sistemas embebidos desde el punto de vista de la especialización de los recursos humanos requeridos:

- El diseñador enfrenta el problema que en estos sistemas, muchas veces, no se presenta una tan clara distinción entre hardware y software. El software es en muchos casos una extensión del hardware, componentes del hardware son reemplazados por algoritmos controlados por software. Muchos sistemas embebidos trabajan en tiempo real, el software debe responder a un evento externo en microsegundos. El software y hardware están tan interconectados que el rendimiento de ambos es crucial para tener un sistema útil; a veces, decisiones de programación influencian profundamente la selección del hardware.

- El programador de sistemas embebidos es en parte ingeniero (con conocimientos de hardware), parte analista de sistemas, y parte programador tradicional. Los ingenieros deben ser entrenados para realizar compromisos entre rendimiento, prestaciones y costos de un producto.

- En México, las carreras afines a Electrónica o Mecatronica que hoy cuentan con una buena base en programación son las que brindan la formación básica para este tipo de

desarrollos, aunque se debe destacar hay carreras de Informática en algunas universidades que incluyen el tema del software embebido. Tomándose como modelos sus planes de estudio deben superarse barreras que existen en otras instituciones entre las carreras de electrónica y las de Informática, teniendo en cuenta la convergencia tecnológica cada día más acentuado entre estos dos campos, con vistas a la creación de la especialidad en sistemas embebidos. La Red EICAR, integrados por grupos de electrónica y de informática es un buen ejemplo de esta integración.

- Históricamente, los sistemas embebidos eran programados por diseñadores de hardware, dado que solo ellos conocían los detalles de bits y bytes de su última creación. La creciente complejidad de los sistemas embebidos requiere el correspondiente incremento en la especialización del equipo de diseño. Una nueva clase de ingenieros en firmware ocupa un lugar entre diseñadores de hardware y programadores tradicionales. De cualquier modo, los programadores que desarrollen un código embebido siempre tendrán que tener un conocimiento detallado del software y del hardware que integran un sistema, y de la interacción con sensores, actuadores, etc., es decir, con los dispositivos que son las interfaces con el mundo real.

- Por otro lado se debe tener en cuenta que en el desarrollo de sistemas embebidos además de los ingenieros electrónicos, programadores, analistas y otros expertos en informática se requiere el concurso de matemáticos, ingenieros especialistas en telecomunicaciones, transmisión de datos, mecánicos, de procesos, etc., por lo que la capacitación para trabajar en grupos multidisciplinarios será un elemento importante.

Para tener impacto en el campo de la actividad económica y de esa manera contribuir a mejorar la calidad de vida de nuestra gente se hace necesario contar con un sector industrial fuerte y de alto nivel tecnológico, lo que implica medidas de apoyo por parte de los sectores gubernamentales involucrados que complementen la política de tipo de cambio alto y permitan que este sector de la industria electrónica alcance una masa crítica que incremente su penetración en el mercado tanto local como internacional.

El énfasis deberá estar puesto en el aumento del valor agregado local, por ello deberá realizarse simultáneamente una política que incentive la generación propia de tecnología , desarrolle proveedores de partes y componentes producidos en el país y promueva la formación de empresas que realicen desarrollos de software embebido como proveedoras de empresas integradoras de sistemas embebidos tanto locales como internacionales.

1.7. Factibilidad de alcanzar los objetivos de desarrollo.

En un escenario optimista el gobierno provee un apoyo específico e importante para el desarrollo del sector industrial basado en los hechos mencionados anteriormente, lo que creará una demanda mayor de formación de recursos humanos a todo nivel, de alta con una relación más estrecha entre los grupos de investigación y desarrollo y el sector productor de bienes y servicios. Las instituciones universitarias recibirían el apoyo necesario para satisfacer estas necesidades.

En este caso, las empresas, que actualmente exportan, incrementan en forma importante su presencia en el mercado de productos especiales, con el aporte científico tecnológico de grupos de conocedores que se formen en el país.

En un escenario pesimista, el mercado electrónico continuará creciendo lentamente y seguirá soportando la importación. El país perderá empleos calificados al suplantarse productos fabricados con tecnologías tradicionales por otros realizados en el exterior con tecnología electrónica e informática.

El crecimiento del sector académico es menor y con el objetivo implícito de formar profesionales mayoritariamente para tareas de apoyo técnico, instalación, mantenimiento y de contar con grupos de investigación y desarrollo que mantengan actualizado en el país conocimientos que se generan en el exterior ***[PRO2006]***.

Capitulo 2

Conceptos Básicos y Definiciones

Capítulo 2
Conceptos Básicos y Definiciones.

En este capítulo se expondrán algunos de los conceptos utilizados en el área de Sistemas Embebidos, para que sirven estos sistemas, así como la definición de los componentes que utilizan de una manera predefinida en el hardware embebido y también los componentes auxiliares que pueden ser utilizados por estos sistemas.

2.1. Sistema embebido.

Un sistema embebido, también conocido como sistema integrado, es una combinación de hardware, software y, eventualmente, componentes mecánicos diseñados para realizar una función específica.

2.2. ¿Para qué sirven los sistemas embebidos?

Un sistema embebido está diseñado para realizar funciones que pueden ser riesgosas, repetitivas o que requieran de tiempos de respuesta imposibles de alcanzar para los seres humanos.
Existen sistemas embebidos para realizar los más variados tipos de aplicaciones usando una amplia gama de tecnologías diferentes ***[JOW2003]***.

2.3. Componentes de un sistema embebido.

Un sistema embebido en principio estaría formando por microprocesador, microcontrolador, DSP, etc. Es decir la CPU o unidad que aporta inteligencia al sistema y un software que se ejecute sobre este. Sin embargo este software necesitara sin duda un lugar donde poder guardarse para luego ser ejecutado por el procesador. Esto podría tomar la forma de memoria RAM o ROM, Todo sistema embebido necesitara en alguna medida una cierta cantidad de memoria, la cual puede incluso encontrarse dentro del mismo chip del procesador. Además de esto normalmente un sistema embebido contara con una serie de salidas y entradas necesarias para comunicarse con el mundo exterior.

Debido a que las tareas realizadas por sistemas embebidos son de relativa sencillez, los procesadores comúnmente usados cuentan con registros de 8 o 16 bits.

En su memoria solo reside el programa destinado a gobernar una aplicación determinada.

Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar y todos los recursos complementarios disponibles tiene como única finalidad atender a sus requerimientos

Estas son las únicas características que tienen en común los sistemas embebidos, todo lo demás será totalmente diferente para cada sistema embebido en particular.

Estos recursos no son ampliables (al menos no fácilmente) como en el caso de las computadoras de escritorio, salvo por buses y conectores diseñados para un tipo específico de periféricos (miniPCI) o expansiones de memoria de almacenamiento (compact flash) ***[JOW2003]***.

- **Microprocesadores.**
  La gran mayoría de los sistemas embebidos poseen algún tipo de procesador integrado, también denominados SOC (del inglés System On Chip) Fig. 2.1. Los procesadores SOC poseen integrados controladores DRAM, UART, PCI, Ethernet, etc.

Fig. 2.1 La imagen muestra el Routerboard 532

- **Microcontroladores.**
  Los microcontroladores son sistemas de computadoras completos en un solo chip, normalmente combinando una Unidad Aritmética Lógica, una pequeña cantidad de memoria, contadores de tiempo, puertos serie, entradas/ salidas y un reloj oscilador.

  Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea.

Debido principalmente a su versatilidad y bajo costo. Los MCU son hoy una de las opciones más comunes a la hora de implementar sistemas embebidos. Si bien no poseen la capacidad de procesamiento de una computadora de propósito general. Si cuentan con la suficiente capacidad para el desarrollo de tareas simples y repetitivas. Cuando la tarea es simple, el costo asociado a trabajar con un MCU es mucho menor.

Un MCU también resulta muy conveniente si el proyecto a desarrollar es un proyecto a gran escala, en donde se desarrollarán miles de unidades. Debido a su bajo costo, la opción de utilizar un MCU resulta cada vez más clara.

- **DSP.**
  Procesador Digital de Señales (DSP, sigla en inglés de Digital Signal Processor) es un tipo de microprocesador, increíblemente rápido y poderoso. Un DSP es único porque procesa señales en tiempo real. Esta capacidad de procesamiento en tiempo real hace a los DSP ideales para aplicaciones que no toleran ningún retardo. Por ejemplo, no es fácil conversar a través de un teléfono celular cuando existe un retardo en la línea. Esto lleva a que la señal se corte o a confusión ya que ambos usuarios hablan a la vez. Con los teléfonos celulares actuales, los cuales usan DSP's, es posible hablar normalmente. El DSP dentro del teléfono procesa el sonido (convirtiéndolo de una señal analógica a digital, filtrando, comprimiendo y realizando otras tareas en forma digital) tan rápidamente que uno puede hablar y escuchar sin problemas de retardo ni ninguna molestia que ello implica. O sea, se procesa en tiempo real.

- **Memoria.**
  En ella se encuentra almacenado el código de los programas que el sistema puede ejecutar así como los datos. Su característica principal es que debe tener un acceso de lectura y escritura lo más rápido posible para que el microprocesador no pierda tiempo en tareas que no son meramente de cálculo. Al ser volátil el sistema requiere de un soporte donde se almacenen los datos incluso sin disponer de alimentación o energía.

  - **CMOS-RAM.**
    Es un chip de memoria de lectura y escritura alimentado con una pila donde se almacena el tipo y ubicación de los dispositivos conectados a la placa madre (disco duro, puertos de entrada y salida, etc.). Además contiene un reloj en permanente funcionamiento que ofrece al sistema la fecha y la hora.

- **Periféricos de E/S.**
  Estos subsistemas se interconectan mediante los buses de sistema (formados a su vez por el bus de control, el bus de direcciones y el bus de datos).

  El subsistema de entrada acepta datos del exterior para ser procesados mientras que el subsistema de salida transfiere los resultados hacia el exterior. Lo más habitual es que haya varios subsistemas de entrada y varios de salida. A estos subsistemas se les reconoce habitualmente como periféricos de E/S.

  Los sistemas embebidos se caracterizan normalmente por la necesidad de dispositivos de E/S especiales. Cuando se opta por diseñar el sistema embebido partiendo de una placa con microcomputador también es necesario comprar o diseñar placas de E/S adicionales para cumplir con los requisitos de la aplicación concreta.

- **Buses.**
  Es el conjunto de líneas (cables) de hardware utilizados para la transmisión de datos entre los componentes de un sistema informático. Un bus es en esencia una ruta compartida que conecta diferentes partes del sistema como el procesador, la controladora de unidad de disco, la memoria y los periféricos de entrada, salida, permitiéndoles transmitir información.

  El bus, por lo general supervisado por el microprocesador, se especializa en el transporte de diferentes tipos de información.

  El Bus se refiere al camino que recorren los datos desde una o varias fuentes hacia uno o varios destinos y es una serie de hilos contiguos. En el sentido estricto de la palabra, esta definición sólo se aplica a la interconexión entre el procesador y los periféricos.

  - **Bus de Control.**
    El bus de control (en ocasiones denominado bus de comando) transporta las órdenes y las señales de sincronización que provienen de la unidad de control y viajan hacia los distintos componentes de hardware. Se trata de un bus bidireccional en la medida en que también transmite señales de respuesta del hardware.

- **Bus de Direcciones.**
  El bus de direcciones, (también conocido como bus de memoria) transporta las direcciones de memoria al que el procesador desea acceder, para leer o escribir datos. Se trata de un bus unidireccional.

- **Bus de Datos.**
  Se encarga de mover la información por los distintos componentes de hardware del sistema, tanto de entrada y salida, de igual manera con el Microprocesador.

2.4. Componentes auxiliares.

Como se menciona anteriormente para el desarrollo de alguna aplicación específica se requiere del uso de otros componentes para su funcionamiento óptimo.

La comunicación adquiere gran importancia en los Sistemas Embebidos. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar de cable o inalámbricas. Así un SE normalmente incorporará puertos de comunicaciones del tipo RS232, SPI, I^2C, USB, Bluetooth, Ethernet, GSM, GPRS, etc.

- **Puerto Serie RS232.**
  El RS232 es un estándar de comunicaciones propuesto por la Asociación de Industrias Electrónicas (EIA) y es la última de varias versiones anteriores. Antiguamente se utilizaba para conectar terminales a un ordenador Host. Se envían datos de 7, 8 o 9 bits. La velocidad se mide en baudios (bits/segundo) y sólo son necesarios dos cables, uno de transmisión y otro de recepción.

  Lo más importante del estándar de comunicaciones es la funciones específica de cada pin de entrada y salida de datos porque nos encontramos básicamente con dos tipos de conectores los de 25 pines y los de 9 pines, es probable que se encuentre mas la versión de 9 pines aunque la versión de 25 permite muchas más información en la transferencia de datos.

  Las señales con la que actúa el puerto son digitales (0 - 1) y la tensión a la que trabaja es de 12 Voltios.

- **SPI.**
  El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interface de periféricos serie o bus SPI es un estándar para

controlar casi cualquier electrónica digital que acepte un flujo de bits serie regulado por un reloj

Incluye una línea de reloj, dato entrante, dato saliente y un pin de chip select, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse. De esta forma, este estándar permite multiplexar las líneas de reloj.

- **I²C.**
  I²C es un bus de comunicaciones serie. Su nombre viene de Inter-Integrated Circuit (Circuitos Inter-Integrados). La versión 1.0 data del año 1992 y la versión 2.1 del año 2000, su diseñador es Philips. La velocidad es de 100Kbits por segundo en el modo estándar, aunque también permite velocidades de 3.4 Mbit/s. Es un bus muy usado en la industria, principalmente para comunicar microcontroladores y sus periféricos en Sistemas Embebidos y generalizando más para comunicar circuitos integrados entre sí que normalmente residen en un mismo circuito impreso.

- **USB.**
  USB nace como un estándar de entrada/salida de velocidad media-alta que va a permitir conectar dispositivos que hasta ahora requerían de una tarjeta especial para sacarles todo el rendimiento, lo que ocasionaba un encarecimiento del producto además de ser productos propietarios ya que obligaban a adquirir una tarjeta para cada dispositivo.

  Pero además, USB nos proporciona un único conector para solventar casi todos los problemas de comunicación con el exterior, pudiéndose formar una auténtica red de periféricos de hasta 127 elementos.

  Mediante un par de conectores USB que ya hoy en día son estándar en todas las placas, y en el espacio que hoy ocupa un sólo conector serie de 9 pines nos va a permitir conectar todos los dispositivos que tengamos, desde el teclado al modem, pasando por ratones, impresoras, altavoces, monitores, scanner, cámaras digitales, de video, plotters, etc. sin necesidad de que nuestro PC disponga de un conector dedicado para cada uno de estos elementos, permitiendo ahorrar espacio y dinero.

- **Bluetooth.**
  Si se quiere integrar en la plataforma embebida este sistema de comunicación inalámbrica existen chips que implementan un perfil de UART (Universal Asynchronous Receiver-

Transmitter) a 921,6 Kbps para facilitar una conexión Bluetooth transparente vía RS-232. Tienen un alcance estándar de 10 m y permite prescindir de los molestos cables.

- **Ethernet.**
  Existen integrados simples que se conectan directamente al microprocesador de la placa y se puede disponer así de un puerto de 10/100 Mbps.

- **GSM.**
  Son las siglas de Global System for Mobile communications (Sistema Global para las comunicaciones Móviles), es el sistema de teléfono móvil digital más utilizado y el estándar de facto para teléfonos móviles en Europa.

  Definido originalmente como estándar Europeo abierto para que una red digital de teléfono móvil soporte voz, datos, mensajes de texto y roaming en varios países. El GSM es ahora uno de los estándares digitales inalámbricos 2G más importantes del mundo.

- **GPRS.**
  GPRS (General Packet Radio Service) es una tecnología europea derivada del GSM (Global System for Mobile) y corresponde a lo que se conoce como generación 2.5, correspondiendo GSM a la segunda generación en comunicaciones móviles.

  Con GPRS se consigue beneficiarse de todas las ventajas de la transmisión de datos en movilidad, mejorando la eficiencia, la velocidad y la comodidad de sus comunicaciones. La velocidad máxima teórica de conexión puede llegar a los 115 kbps, 12 veces más que la permitida por la red actual GSM, aunque actualmente en media no supera nunca los 40 Kbps de tasa de transferencia real.

El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos LED, reconocer el estado abierto cerrado de un conmutador o pulsador, etc.

- **Puertos digitales.**
  Las entradas/salidas (E/S) digitales se basan en el principio de todo o nada, es decir o no conducen señal alguna o poseen un nivel mínimo de tensión. Estas E/S se manejan a nivel de bit. La información digital puede tomar dos tipos de valores: “0” o “1”. El bit es igual a

"0" si no hay ningún tipo de señal presente (0 V) y es igual a "1" si se detecta un nivel mínimo de tensión, por ejemplo 3.1 V.

- **Puertos analógicos.**
  Las señales analógicas son las que varían en función del tiempo adquiriendo valores dentro de un intervalo continuo. La información analógica puede tomar infinitos valores y se puede adquirir con distinta resolución en número de bits.

  Las entradas analógicas permiten que se pueda leer y trabajar con señales de tipo analógico, como pueden ser por ejemplo la temperatura, la presión o el caudal. Esta información se obtiene de los sensores, que son unos dispositivos de entrada que captan la señal analógica del exterior y devuelven un valor de tensión que se transforma en información digital.

  Para realizar el procesado de la señal analógica, se precisa de un componente hardware electrónico que realice dicha tarea, junto con un software específico para hacer funcionar el dispositivo.

2.5. Software Embebido.

El software embebido es conocido en inglés como firmware o embedded software. Embebido quiere decir que forma parte esencial del mismo sistema. A diferencia del software de aplicación o de alto nivel, el software embebido tiene las siguientes características:

- Tiene una interfaz directa con el hardware del dispositivo
- Suele estar almacenado en ROM, en la misma tarjeta de hardware
- Es el intermediario entre el software de alto nivel y las funciones del hardware
- Suele tener requerimientos relacionados con el bajo consumo de potencia y el bajo costo de producción
- Generalmente se ejecuta en un hardware muy limitado, tanto en velocidad y recursos como en cantidad de memoria
- Tiene capacidad de auto-prueba, no sólo del software mismo sino también del hardware
- Tiene muy poca tolerancia a fallas porque una vez en manos del usuario, es muy difícil o imposible realizar cambios

- Debe ser robusto y relativamente autónomo; es común esperar que el software corra durante años sin fallar, incluso en lugares sin acceso a personas, como una sonda espacial
- Es común que tenga requerimientos de tiempo real
- Corre sin un sistema operativo propiamente dicho, o bajo uno especializado

Capitulo 3

Arquitecturas Sistemas Embebidos

Capítulo 3
Arquitecturas Sistemas Embebidos.

Entre los Sistemas Embebidos se utilizan varios tipos de arquitecturas, esto dependiendo de cuál será la finalidad del sistema o cuáles son las necesidades de los desarrolladores del sistema.

Para el caso especifico de los futuros ejemplos y el prototipo final se utilizará la arquitectura Harvard ya que se presenta con una mayor facilidad de manejo con los distintos simuladores que existen.

Existen cientos de marcas en el mercado que fabrican soluciones SOC´s que como ya se había mencionado en el capitulo anterior se refiere a un procesador integrado.

Algunos ejemplos de las arquitecturas utilizadas en este tipo de procesadores son:

- Harvard, principalmente utilizado en equipos de sonido y comunicaciones.
- ARM, generalmente utilizados en teléfonos celulares y equipamiento de redes.
- MIPS, arquitectura con implementaciones de 32 y 64 bits utilizado en una gran cantidad de productos populares, televisores Sony de alta definición, access points inalámbricos Linksys y la popular consola de video juegos Sony Play Station 2.

3.1. Arquitectura Harvard.

La arquitectura conocida como Harvard, consiste simplemente en un esquema en el que la Unidad de Control está conectada a dos memorias por intermedio de dos buses separados. Una de las memorias contiene solamente las instrucciones del programa, y es llamada Memoria de Programa. La otra memoria solo almacena los datos y es llamada Memoria de Datos. Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de la memoria de programa pueden diseñarse de manera tal que todas las instrucciones tengan una sola posición de memoria de programa de longitud. Además, como los buses son independientes, el CPU puede estar accediendo a los datos para completar la ejecución de una instrucción, y al mismo tiempo estar leyendo la próxima instrucción a ejecutar. Se puede observar claramente que las principales ventajas de esta arquitectura son:

a) El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa,

b) El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador) ***[TAN2005]***.

Los microcontroladores PIC 16C5X, 16CXX y 17CXX poseen arquitectura Harvard, con una memoria de datos de 8 bits, y una memoria de programa que, según el modelo, puede ser de 12 bits para los 16C5X, 14 bits para los 16CXX *Fig. 3.2.1, Fig. 3.2.2* y 16 bits para los 17CXX.

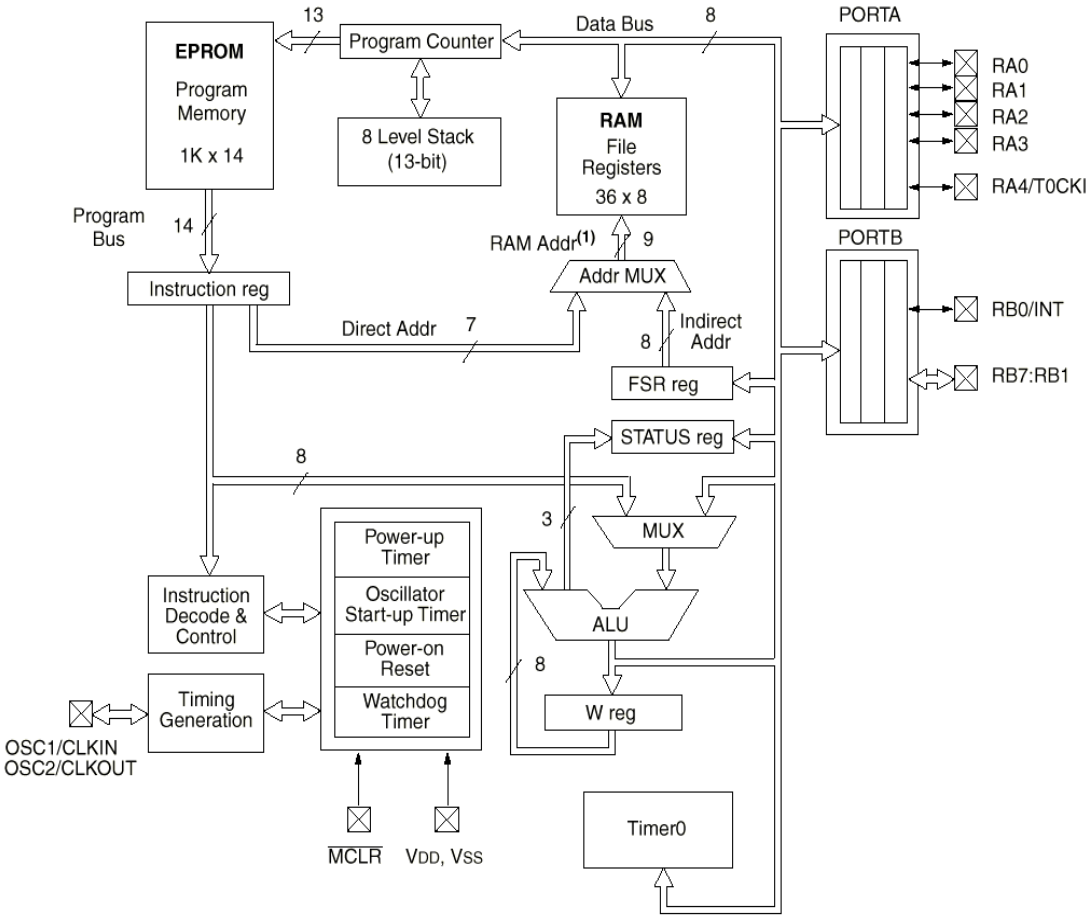


Fig. 3.2.1 La figura muestra el diagrama de bloques del microcontrolador PIC16C61

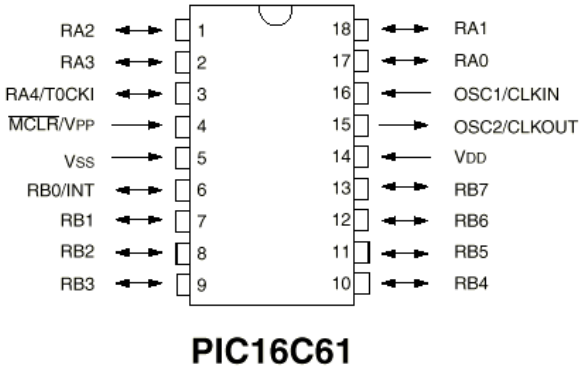


Fig. 3.2.2 La figura muestra el patillaje del microcontrolador PIC16C61

3.2. Arquitectura ARM.

(Advanced RISC Machine) ARM es una arquitectura de procesadores RISC de 32 bits desarrollada por ARM Ltd, que es ampliamente empleada en sistemas integrados.

En la actualidad, ARM Ltd no hace procesadores, solo los diseña y licencia sus diseños a fabricantes (P. ej: Analog Devices, Atmel, Cirrus Logic, Hyundai, Intel, Oki, Plilips, Samsung, Sharp, Lucent, 3Comp, HP, IBM, Sony, etc.).

Por sus características de ahorro energético, ARM domina en el mercado de dispositivos móviles, donde el bajo consumo de energía es un objetivo crítico de sus diseños.

Las características de la arquitectura ARM son las siguientes:

- Registros de 32 bits
- Excepciones vectorizadas
- Número de transistores: > 74,209 implica bajo consumo.
- Frecuencias de operación: 45 – 133 MHz
- Bus de 32 bits para datos e instrucciones.
- Elevado rendimiento: hasta 120 MIPS.
- Elevada densidad de código (Diseñado para trabajar en C)
- Se basa en Arquitectura RISC.
- 37 registros de 32 bits (16 disponibles).
- Registros 0 a 7 disponibles en todo momento
- Memoria caché (dependiendo de la aplicación)
- Estructura del bus tipo Von Neumann (ARM7), tipo Harvard (ARM9).

Los tipos de datos que maneja la arquitectura ARM son de 8/16/32 bits, también tiene la característica que tiene 7 modos de operación: usr, fiq, irq, svc, abt, sys, und *Fig3.3*.

- **User**: Modo de ejecución normal del programa
- **System**: Para ejecutar tareas de sistema operativo privilegiadas
- **Supervisor**: - Modo protegido para el sistema operativo.
- **Abort**: Utilizado para implementar la protección de memoria y/o procesador
  Dos clases de Abort, Abort de prefetch o de datos
- **Undefined**: Soporta emulación de software de instrucciones no soportadas o coprocesadores no implementados.
- **FIQ**: Manejo de la Fast interrupt
- **IRQ**: Manejo de interrupciones de propósito general

Todas las familias de procesadores ARM comparten el mismo conjunto de instrucciones como por ejemplo: AND, EOR, SUB, RSB, ADD, ADC, SBC, RSC, TST, TEQ, CMP, CMN, ORR, MOV, BIC, MVN, (Multiplicaciones) MUL, MLA, MULL, MLAL ***[i8].***

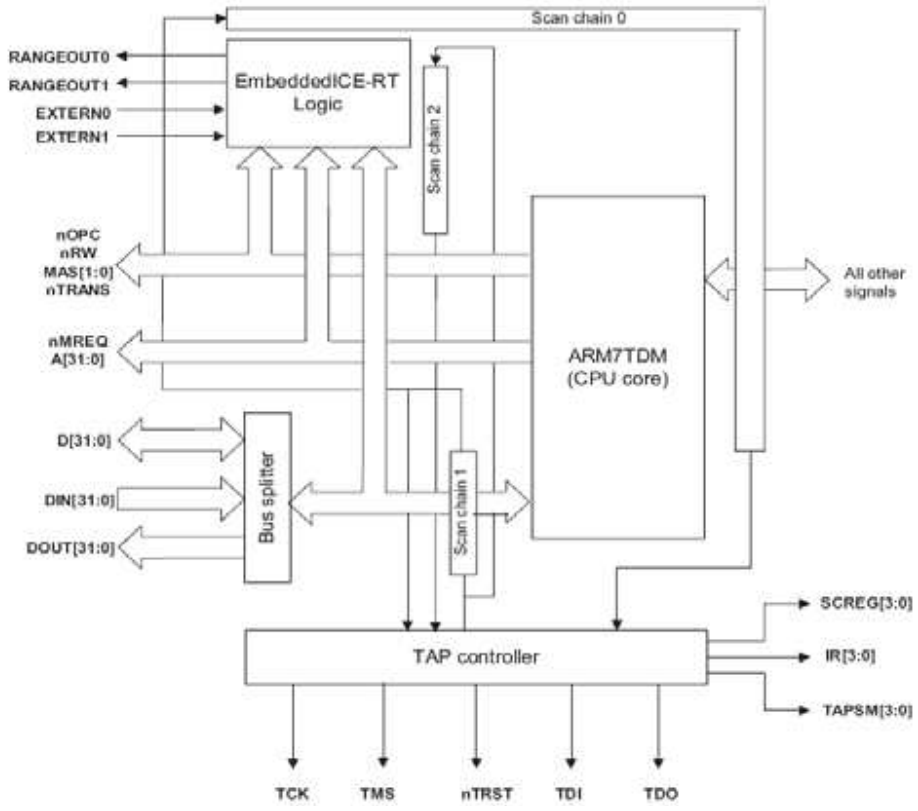


Fig. 3.3 La figura muestra el diagrama de bloques del microcontrolador ARM7TDMI

3.3. Arquitectura MIPS.

(Microprocessor without Interlocked Pipeline Stages) Es una arquitectura de procesadores tipo RISC desarrollada por MIPS Computer Systems Inc. *Fig. 3.4*. Los diseños de MIPS se usan en las estaciones de trabajo de SGI, y tienen mucha implantación en sistemas empotrados, dispositivos que soportan Windows CE, y en los routers de Cisco. La consola Nintendo 64, la Sony PlayStation, la Sony PlayStation 2, y la consola portátil Sony PSP usan procesadores MIPS. A finales de los 90, se estimó que uno de cada tres chips tipo RISC que salieron al mercado estaban basados en MIPS.

Como los diseñadores crearon un conjunto de instrucciones muy limpio, los cursos sobre arquitectura de computadores en las Universidades suelen basarse típicamente en la arquitectura MIPS. El diseño de las CPUs de MIPS, junto con las SPARC, otra arquitectura RISC muy temprana, influyeron mucho en otros diseños de computadores tipo RISC como los DEC Alpha.

Este tipo de microprocesadores tiene 4 modos de funcionamiento: usuario, núcleo (kernel), supervisor y depuración.

- **Modo usuario**: Los programas normales funcionan en este modo, las direcciones de memoria accesibles deben estar por debajo de los 2 GB (bit más significativo de la dirección a 0), Existen instrucciones que no pueden utilizarse (instrucciones privilegiadas).

- **Modo núcleo (kernel):** Es el modo propio del sistema operativo. Los sistemas sencillos y muchos sistemas empotrados funcionan en este modo. Se puede acceder al mapa de memoria completo. Se pueden ejecutar todas las instrucciones del repertorio, incluyendo las instrucciones privilegiadas. Se salta automáticamente al modo núcleo cuando hay una excepción (software o producida por un error) y cuando se acepta una interrupción.

- **Modo supervisor:** implementado como un intento de satisfacer a DEC.

- **Modo depuración (debug):** sólo en ciertas máquinas.

MIPS tiene 32 registros de propósito general con un ancho de 32 bits de cada registro ***[i9]***.

El Direccionamiento se realiza a nivel de octeto: cada octeto tiene su propia dirección de memoria.

MIPS puede funcionar con ordenación de datos big-endian o little-endian.

- Little endian: el octeto menos significativo primero.
- Big endian: el octeto más significativo primero.

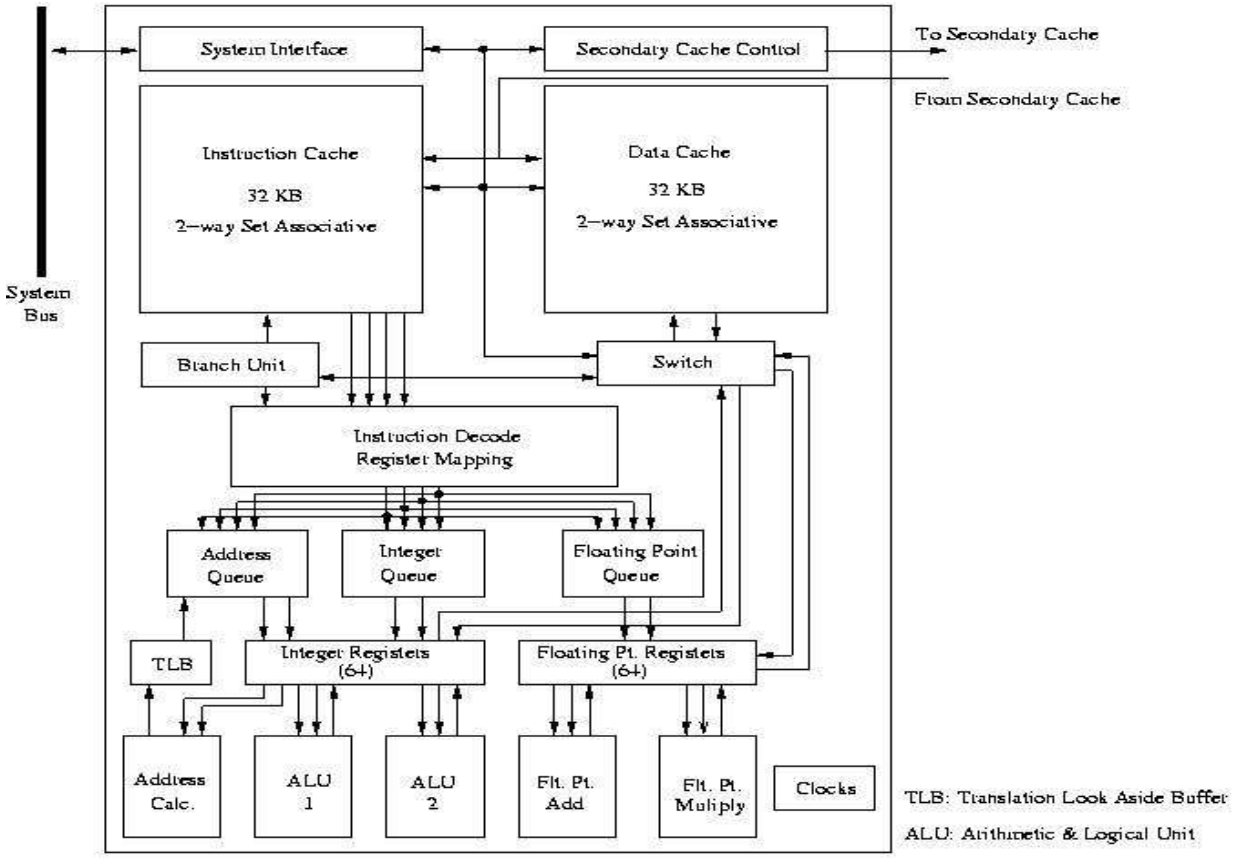


Fig. 3.4. La figura muestra el diagrama de bloques del microprocesador MIPS R16000

3.4. Arquitectura PowerPC

La filosofía RISC fue introducida por John Cocke en los años 70. La primera implementación de un procesador RISC llegaría con el proyecto 801 de IBM, dedicado especialmente a computación embebida. Ofreció un rendimiento bastante malo, por lo que IBM emprendió el Proyecto América, creando el primer procesador de la arquitectura POWER, incluido en el RISC System/6000 a principios de los 90.

En 1991, IBM se dio cuenta de las posibilidades de su diseño y se creó la alianza AIM (Apple, IBM & Motorola, actualmente Freescale), con el objetivo de terminar con el domino del tándem Microsoft-Intel que ofrecía sistemas basados en 80386 y 80486 y tenía al Pentium en desarrollo. Para Motorola supuso una salida al poco éxito que estaba teniendo su serie 88000, por lo que requirió que la nueva solución fuera compatible con el bus del 88000 a nivel de hardware.

El resultado de estos requerimientos fue la arquitectura PowerPC (Performance Optimization With Enhanced RISC Performance Computing).

La primera implementación de un procesador PowerPC llegó en 1993 con el PowerPC 601 de IBM. Los primeros productos tuvieron cierto éxito, Motorola implementó sus propios sistemas y Microsoft portó Windows NT 3.51 a la arquitectura. Aunque en la actualidad sólo versiones de Linux y Mac OS X siguen estando disponibles para PowerPC.

Esta arquitectura ha tenido mucho éxito en entornos embebidos, con representantes como el PowerPC 401 y diversos microcontroladores desde 8 a 32 bits, usados en automóviles, controladores de red e incluso aviones (el nuevo Airbus A380, lleva 25 chips PowerPC y el SO de tiempo real LynxOS).

Se han construido numerosos supercomputadores en torno a procesadores PowerPC, como el MareNostrum, que cuenta con 10.240 PowerPC 970MP a 2,3 GHz.

Además, todas las videoconsolas dominantes tienen corazón PowerPC, con el Broadway de Nintendo Wii, Cell para la PlayStation 3 y Xenon de XBox 360. Y por supuesto, es una excelente solución para estaciones de trabajo y equipos de sobremesa, así como portátiles ***[i10]***.

En la actualidad, la especificación PowerPC está gestionada por el consorcio Power.org, que incluye entre sus miembros a IBM y Freescale.

La ISA (Instruction Set Architecture) de PowerPC se describió originalmente en tres libros [6] que contemplaban las instrucciones de nivel de usuario, el entorno virtual y el entorno operativo.

El primero describe el uso de los registros e instrucciones (todas no privilegiadas) típicamente usados por los programas de aplicación. El segundo describe las instrucciones que permiten manejar el almacenamiento de la CPU (cachés, alineamiento de accesos, etc.). Son instrucciones no privilegiadas, pero que sólo el SO típicamente usa. El último libro hace referencia a las instrucciones privilegiadas para el manejo de la traducción de direcciones, interrupciones, etc.

La última versión de la ISA es la 2.03 *Fig. 3.5*, publicada en noviembre de 2006, cuenta con los tres libros originales, en los que se separa el modo privilegiado en las categorías de servidor y empotrado y un libro adicional de instrucciones de longitud variable. Incluye soporte para virtualización, descripción precisa de las extensiones AltiVec para cálculo vectorial y, lo más importante, fusiona todos los juegos de instrucciones de PowerPC, POWER y los procesadores empotrados en un único ISA.

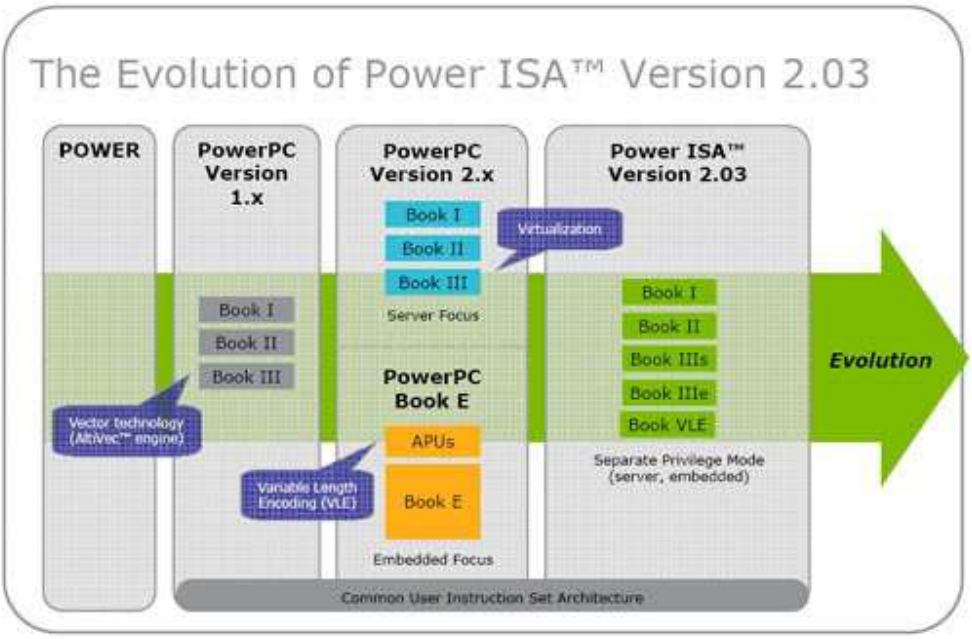


Fig. 3.5 La figura muestra a evolución de la ISA Power

En la especificación básica, las instrucciones tienen un ancho de codificación fijo de 32 bits. Las instrucciones aritméticas y lógicas usan un formato de tres operandos, especificados por el nombre de los registros.

Las CPUs PowerPC *Fig. 3.6*, direccionan al byte y sólo es posible el acceso a memoria mediante las instrucciones LOAD y STORE, que copiarán a los registros del procesador los valores de la memoria principal o los transferirán a ésta.

Para generar las direcciones se sumará el contenido de un registro base a un índice proporcionado en la propia instrucción o en un registro índice. PowerPC permite actualizar el registro base con la última dirección generada para, por ejemplo, procesar elementos consecutivos ***[i11]***.

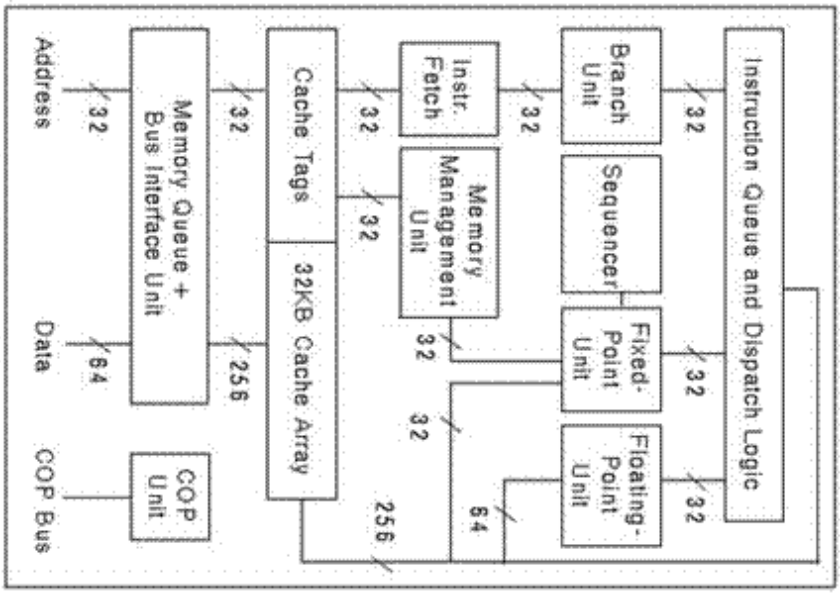


Fig. 3.6 La figura muestra el Diagrama de bloques del PowerPC 601

3.5. Arquitectura SH.

SuperH (o SH) es el nombre de una arquitectura de microcontroladores y microprocesadores. Es fundamentalmente una arquitectura RISC de 32 bits de carga/almacenamiento encontrada en un gran número de sistemas empotrados.

El núcleo de la familia de procesadores SuperH fue desarrollado inicialmente por Hitachi a principios de los 90. Muchos microcontroladores y microprocesadores se basaron en esta arquitectura. Tal vez, el más famoso es el SH7709, usado en los PDA HP Jornada corriendo Windows CE.

Hitachi desarrollo un juego completo de CPUs compatibles hacia arriba en instrucciones. Originalmente, el SH-1 y el SH-2 fueron usados en la Sega Saturn y en la Sega 32X y posteriormente en muchos otros microcontroladores usados en aplicaciones empotradas. Estos núcleos usan un juego de instrucciones de 16 bits, aunque la longitud de los registros y de los buses de datos son de 32 bits, lo que da una excelente densidad del código. Durante su desarrollo la memoria era bastante cara.

Algunos años después, el núcleo SH-3 fue añadido a esta familia de CPUs, extendiendo a los originales principalmente con otro concepto de interrupciones, una MMU y un concepto modificado de caché. El núcleo SH-3 también tuvo una extensión DSP, llamada SH-3-DSP. Con los

buses de datos extendidos para una mayor eficacia en el DSP, los acumuladores especiales y un motor DSP tipo MAC dedicado, este núcleo fue unificando el mundo de los DSP y el de los procesadores RISC. Una derivación fue también usada en el núcleo SH-2 original, llamada SH-DSP.

Para la Sega Dreamcast, Hitachi desarrolló la arquitectura SH-4. Esta fue una extensión masiva de los anteriores núcleos. La ejecución superescalar de instrucciones y una FPU vectorial paralela fueron los puntos más resaltados de esta arquitectura. Este núcleo fue usado también en muchos chipsets para aplicaciones empotradas que requerían unas prestaciones muy altas. Los chips estándar basados en el SH-4 fueron presentados sobre 1998.

Un poco más tarde, Hitachi y STMicroelectronics formaron la compañía de patentes SuperH Inc., que licencia el núcleo SH-4 a otras compañías y desarrolla la arquitectura SH-5, el primer movimiento de SuperH en el área de los 64 bits. SuperH vendió la propiedad intelectual de estos núcleos.

El diseño SH-5 soporta dos modos de operación. El modo SHcompact es equivalente al modo usuario del juego de instrucciones SH-4. El modo SHmedia es muy diferente, usando instrucciones de 32 bits con registros de enteros de 64 bits e instrucciones SIMD. En modo SHmedia, el destino de un salto (jump) es cargado en un registro de salto de manera separada a la propia instrucción de salto. Esto permite al procesador obtener por adelantado instrucciones para una rama sin tener que mirar en el flujo de instrucciones. La combinación de una codificación de instrucciones compactas de 16 bits con una codificación de instrucciones de 32 bits más potente no es exclusiva del SH-5; recientes procesadores ARM tiene un modo Thumb de 16 bits y los procesadores MIPS tiene un modo MIPS-16. Aún así, el SH-5 difiere porque su modo de compatibilidad hacia atrás es de codificación en 16 bits en lugar de 32 bits.

Después de esto, la evolución de la arquitectura SuperH aún continúa. El último paso evolucionario sucedió sobre 2003, cuando los núcleos SH-2 y SH-4 fueron unificados en el núcleo superescalar SH-X, que forma un superconjunto de los juegos de instrucciones de las arquitecturas previas.

Hoy día, los núcleos de CPU SuperH, la arquitectura y los productos son de Renesas Technology, formada por una fusión de los grupos de semiconductores de Hitachi y Mitsubishi.

La familia de núcleos SuperH está formada por:

- SH-1 - usado en microcontroladores para aplicación profundamente empotradas (unidades CD-ROM, electrodomésticos, etc) *Fig. 3.7*.
- SH-2 - usado en microcontroladores con requerimientos mayores de prestaciones, también usado en automóviles como unidad de control del motor o en aplicaciones de red.
- SH-DSP - inicialmente desarrollado para el mercado de telefonía móvil y usado más adelante en muchas aplicaciones de consumo que requieres prestaciones de DSP para compresión JPEG, etc.

- SH-3 - usado para aplicaciones móviles y de mano, fuerte en las aplicaciones Windows CE y el mercado durante años en sistemas de navegación para coche.
- SH-3-DSP - usado principalmente en terminales multimedia y en aplicaciones de red, al igual que en impresoras y en faxes
- SH-4 - usado cuando se necesitan altas prestaciones, como en los terminales multimedia en los coches, en videoconsolas o en set-top-boxes.
- SH-5 - usado en aplicaciones multimedia de gama alta.
- SH-X - núcleo principal usado de varias maneras (con o sin DSP o FPU) en unidades de control de motores, equipos multimedia para coches, set-top boxes o teléfonos móviles.

Los núcleos SuperH son soportados mundialmente por muchos sistemas operativos de tiempo real ***[i12]***.

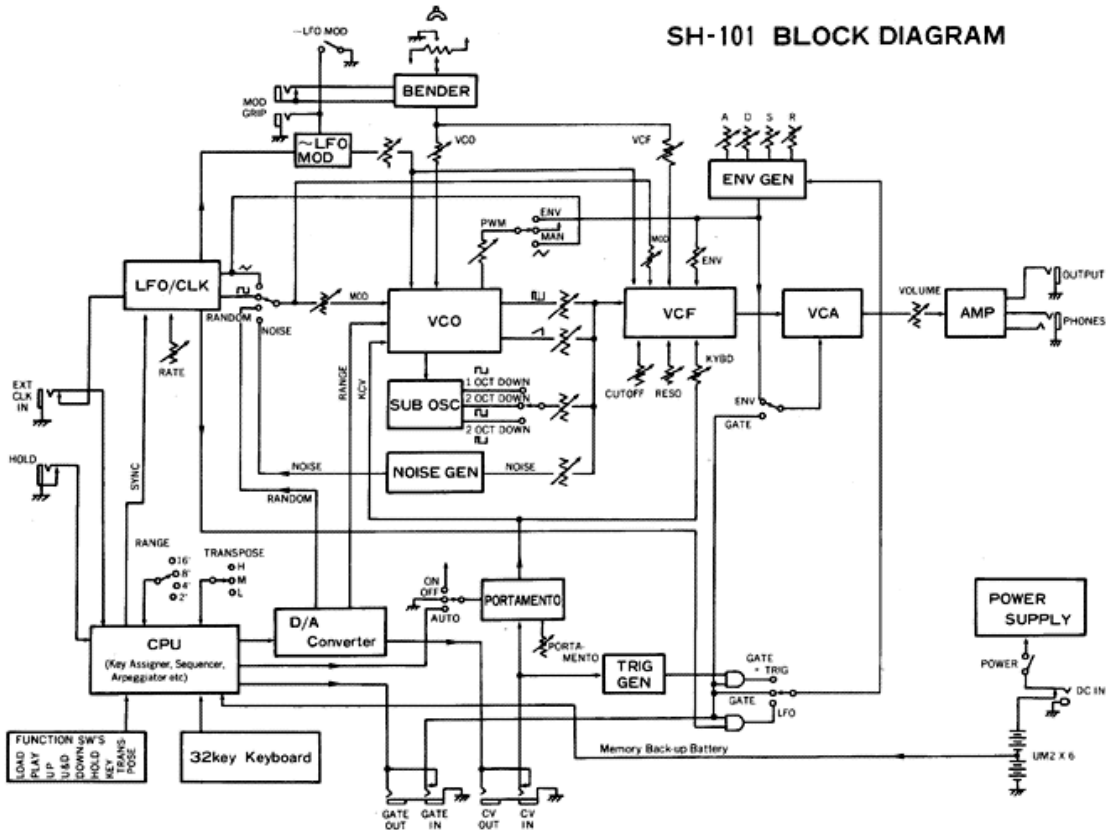


Fig. 3.7 La figura muestra el Diagrama de bloques del Microcontrolador SH-101

Las arquitecturas antes mencionadas son algunas de las más utilizadas en el medio de los sistemas embebidos. *Fig. 3.8*

Existen otro tipo de Arquitecturas como lo son la x86, Coldfire, Softcore que al igual son utilizadas en el desarrollo de los Sistemas Embebidos.

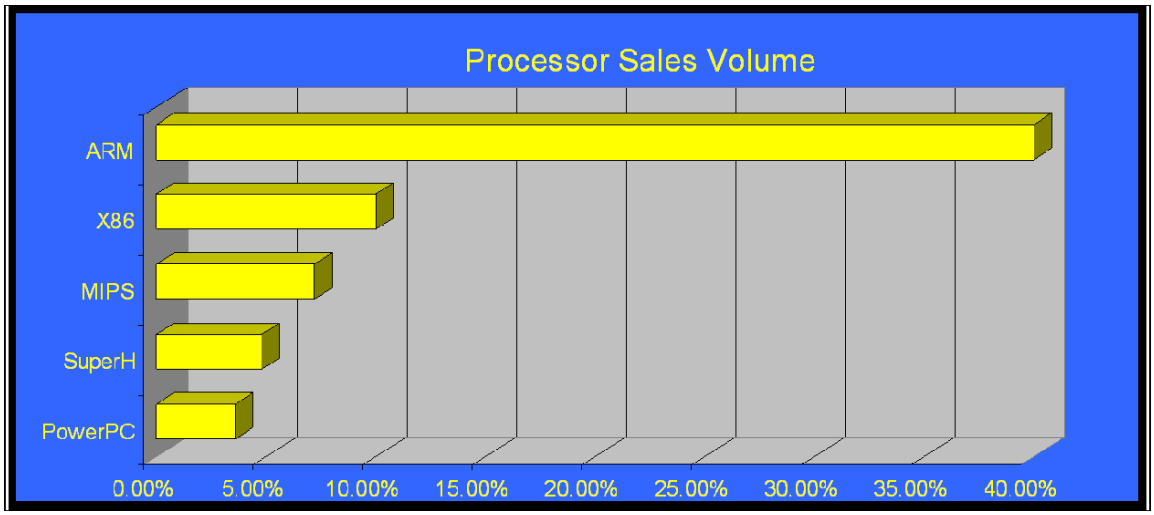


Fig. 3.8 La figura muestra las preferencias en las ventas de microprocesadores para sistemas embebidos, Fuente: Microsoft. (2007)

Estas arquitecturas aunque son de gran confiabilidad a veces en el diseño de sistemas embebidos se requieren acciones en tiempo real así como prestaciones de fiabilidad y durabilidad al igual que las necesidades de rendimiento y funcionalidad, es por esto que algunas empresas optan por construir una solución embebida competitiva para las necesidades del sistema según sea el caso.

La arquitectura más conocida, desarrollada por National Instruments es la CompactRIO que entre sus características se encuentra su bajo costo en comparación con las arquitecturas más comerciales así como también se presenta como un sistema robusto y de altas prestaciones que combina la potencia de procesamiento y la flexibilidad de las FPGAs (Field-Programmable Gate Array) con la fiabilidad de un procesador en tiempo real en un solo paquete fácil de manipular.

NI CompactRIO se basa en la nueva tecnología de Entradas/Salidas reconfigurables (RIO), su funcionalidad básica es proporcionada por una FPGA programable por el usuario. Se puede acceder y configurar la FPGA usando el software de desarrollo gráfico LabVIEW de NI. Normalmente, la programación de una FPGA requiere un conocimiento detallado de la configuración específica del hardware, así como la utilización de un lenguaje de descripción de bajo nivel como VHDL, que tiene una pronunciada curva de aprendizaje. Pero la tecnología NI RIO reduce la complejidad del hardware embebido y de los lenguajes de bajo nivel para proporcionar un acceso sencillo, pero potente, a las FPGAs.

3.6. Arquitectura CompactRIO.

La arquitectura CompactRIO se compone de tres partes principales: el controlador de tiempo real embebido, el chasis embebido reconfigurable que contiene la FPGA y los módulos de E/S intercambiables. La integración del controlador embebido, el chasis que contiene la FPGA y los módulos conectables de E/S permiten la rápida creación de aplicaciones embebidas y prototipos del sistema para las aplicaciones de medida y control eliminando la necesidad de implementar los detalles del hardware de bajo nivel que son requeridos en los sistemas embebidos. Gracias a la conexión directa entre los módulos de E/S y la FPGA se puede integrar perfectamente la sincronización y el disparo entre los módulos de E/S a través de la FPGA y obtener un alto nivel de flexibilidad del sistema.

El controlador embebido de tiempo real CompactRIO incorpora un procesador industrial de Freescale MPC5200 de 400 MHz que ejecuta las aplicaciones de LabVIEW Real-Time de forma determinística y fiable. Se puede elegir entre miles de las funciones incorporadas de LabVIEW para construir un sistema embebido multi-hilo para control, análisis, registro de datos y comunicación en tiempo real. El módulo LabVIEW Real-Time amplía el entorno de desarrollo para proporcionar unas prestaciones deterministas y en tiempo real. Solo hay que desarrollar el código de la aplicación de tiempo real en un ordenador mediante la programación gráfica y luego descargar la aplicación para que se ejecute en el controlador de tiempo real de CompactRIO que contiene un sistema operativo en tiempo real comercial. Para ahorrar tiempo, también se puede integrar el código existente de C/C++ dentro de la aplicación de LabVIEW Real-Time. El controlador de tiempo real de CompactRIO dispone de un puerto Ethernet de 10/100 Mb/s para los programas de comunicación a través de la red (incluyendo el correo electrónico), un servidor web (HTTP), servidores de archivos (FTP) y entradas de alimentación dobles entre 9 y 35VCC.

El chasis reconfigurable es el corazón de los sistemas embebidos de CompactRIO, contiene el núcleo RIO FPGA. El chip RIO FPGA se conecta a los módulos de E/S mediante una topología en estrella, proporcionado un acceso directo a cada módulo para un control preciso y una flexibilidad sin límites de la temporización, disparo y sincronización. La conexión a través un bus local PCI proporciona una interfaz de alto rendimiento entre la RIO FPGA y el procesador en tiempo real. El chasis reconfigurable ofrece las mismas características de construcción metálica robusta que caracteriza a toda la plataforma CompactRIO.

Cada módulo de E/S de la serie C de NI contiene una función de acondicionamiento de señales incorporado y un borne de conexión de presión por tornillo, un borne de conexión de presión por resorte, BNC o conectores D-Sub. Al integrar el conector en la caja de conexiones de los módulos, el sistema CompactRIO reduce significativamente las necesidades de espacio y el coste del cableado en campo. Hay disponibles varios tipos de E/S entre las que se incluyen: entradas para termopares; entradas para acelerómetros; entradas para células de carga y de deformación; entradas analógicas de hasta ± 60V ± 20mA; salidas analógicas hasta ±10V ± 20mA; E/S digitales industriales de 12/24/48V con un suministro de corriente de hasta 1A y E/S digitales de 5V/TTL para encoders, contadores/temporizadores y generadores de pulsos ***[i13]***.

Capitulo 4
Software Embebido

Capítulo 4
Software Embebido.

Algo fundamental en el desarrollo de Software Embebido lo son los lenguajes de programación sobre los que se trabaja, entre los mas conocido están C, Java, VHDL y recientemente incursiona Basic en esta área, a continuación se describirán algunas características que debe cumplir el Software Embebido, así como metodologías que se han utilizado exitosamente en la manufactura de este tipo de tecnología.

El desarrollo de algún programa computacional o software está íntimamente relacionado con el hecho de traducir funciones matemáticas en procedimientos computacionales. Como su nombre lo indica, estos procedimientos lo que hacen es procesar datos de entrada y transformarlos en datos de salida. Para el diseño de tales procedimientos, la ingeniería y ciencias computacionales han desarrollado un gran número de procedimientos y herramientas en las cuales el mundo real es abstraído a representaciones mínimas; incluso es común que éste no sea tomado en cuenta.

En el caso de software embebido tales abstracciones nos llevarían a errores. El software embebido, si bien incluye en gran medida el procesamiento de datos, sus principales características es que está muy orientado a trabajar en un mundo real y su implementación no son las computadoras personales como las conocemos.

Cuando decimos que el software embebido está orientado a trabajar en un mundo real, significa que este software tiene como objetivos interactuar con dispositivos muy diversos, tales como discos duros, procesos y equipos de plantas industriales, tales como motores, válvulas, etcétera, elementos de carros, aviones, elementos de telefonía, impresoras, incluso equipo lúdico o juguetes. Por otro lado, cuando decimos que el software embebido no se implementa en una computadora personal es porque este software se implementa en “pequeñas computadoras” o dispositivos programables, tales como procesadores de señales digitales o microcontroladores, entre otros.

El software embebido consume tiempo de ejecución, consume energía, debe ser robusto, tolerante a fallas, responder con oportunidad, resolver las diversas peticiones de atención, gestionar de forma eficaz los recursos del sistema y mantener niveles de seguridad adecuados para la aplicación a la que está enfocado.

Hoy en día, los programas de software embebido no son escritos por ingenieros en computación, sino por expertos del área de aplicación. Esto trae como consecuencia que muchas de las técnicas de ingeniería de software, que si bien no son completamente adecuadas para el software

embebido, no sean usadas y por tanto la reutilización o documentación de este tipo de software es penosa o inexistente.

4.1. Características del software embebido.

En resumen las principales características del software embebido se pueden clasificar en las siguientes direcciones.

4.1.1. Temporalidad.

El tiempo ha sido constantemente removido de las teorías computacionales. Sin embargo, para el software embebido el considerar el tiempo es primordial. El uso de sistemas en tiempo real y sus técnicas de diseño son las que son imprescindibles en el desarrollo de software embebido.

Aquí hay que tener mucho cuidado con tiempo real. Muchos piensan que un sistema tiempo real es aquel que responde muy rápido, pero no es así. Un sistema tiempo real es aquel que realiza sus funciones matemáticas bien en un tiempo conocido y especificado a priori. Aquí conviene decir que los sistemas tiempo real que asignan tareas temporalmente con políticas EDF, rate monotonic, entre otras, como los propuestos por muchos autores (Barho, Gutiérrez, Sancho) no resultan adecuados porque si bien se sabe que las tareas podrán ser calendarizadas, no se tiene en cuenta la variación con respecto al periodo de ejecución, lo que introduciría armónicos en los sistemas, algo que por supuesto, los expertos en procesos quiere evitar.

Los sistemas que más se utilizan son aquellos que garantizan muy poca deriva temporal, tales como los estudios de programación temporal (scheduling o hard real time), en los cuales las variaciones temporales se toman en cuenta desde el diseño.

4.1.2. Concurrencia.

El software embebido trabaja con el mundo real, donde muchos procesos están trabajando en paralelo. Por tanto el software debe responder a tales características. El uso de las nociones de hilos, semáforos, monitores, citas, paso de mensajes, etc. son ampliamente usados. Sin embargo, resultan insuficientes, aunque se utilicen todas las técnicas para evitar bloqueos y técnicas para descubrir regiones críticas y protegerlas todavía queda el fenómeno de que no se puede conocer cuándo evolucionará el sistema, así que esquemas reactivos con interrupciones por parte del sistema se deberán incluir.

Aquí, las técnicas usadas por los programadores comunes no son adecuadas. Hay que estudiar y extender las técnicas que actualmente se usan.

4.1.3. Vivacidad.

El software embebido, por esencia, no debe ser detenido o bloqueado involuntariamente. Muchos problemas se pueden evitar si se utilizan buenas técnicas y cuidados en el manejo de la concurrencia, pero no es suficiente. Todavía falta estudiar qué se hace en caso de degradación del sistema. Lo que menos quiere ver un desarrollador de estos sistemas, es que cuando su planta tenga alguna falla, el software embebido le incremente por mil este número de fallas. Las técnicas de robustez y tolerancia a faltas deben ser incluidas en este tipo de software.

4.1.4. Interfaces.

Las interfaces son amplias y variadas. Las podemos clasificar en dos grandes grupos. Las interfaces hacia otro tipo de software y las que están orientadas al hardware. Cada una de ellas tiene sus características y debido a la gran diversidad de plataformas que existen, las interfaces suelen variar. Sin embargo, buenas metodologías de diseño de software ayudan a desarrollar buenas interfaces que pueden ser reutilizados.

4.1.5. Heterogeneidad.

La gran diversidad de elementos que confluyen en el desarrollo de software embebido siempre está presente. Por un lado está la gran diversidad de plataformas y elementos que intervienen en el sistema, pero por otro lado están los diferentes estilos de programación y lenguajes utilizados en el desarrollo de un solo sistema.

4.1.6. Reactividad.

El software embebido siempre está trabajando junto con el sistema. Por lo general el diseño de este software se implementa como la solución de sistemas dinámicos (ecuaciones diferenciales o autómatas) que trabajan periódicamente sin interrupción y de manera infinita. Sin embargo la planta no está exenta de eventualidades o disparos, y nuestro software debe ser capaz de reaccionar a estos eventos también. Es por esto que el software embebido debe ser altamente reactivo ***[DAE1999]***.

4.2. Metodologías de Ingeniería de Software.

Existen ya algunas metodologías que se han propuesto en la ingeniería de software, observándose su posible aplicación y limitaciones en el software embebido.

4.2.1. Procedimientos y orientación a objetos y agentes.

Es el primer mecanismo para agrupar datos y procedimientos y abstraerse de su comportamiento. Un objeto bien puede representar un proceso físico o capturar su comportamiento. Aún más, el concepto de agente bien se puede utilizar para representar un proceso físico. Sin embargo, en estas metodologías no está clara la introducción del tiempo ni la seguridad de software requerida por el sistema.

4.2.2. Orientación a objetos en tiempo real.

Actualmente existe el manejo del tiempo en ambientes que manejan objetos (real time CORBA). Desafortunadamente, el tiempo se maneja para un sistema multiusuario en vez de manejarlo hacia el proceso, por lo que resulta en carencias de poca predictibilidad de la ejecución de alguna tarea.

4.2.3. Diseño de Hardware.

Las metodologías orientadas al manejo de hardware digital pueden prestarse al diseño de software embebido. Los mecanismos que tiene VHDL para sincronizar y manejar el tiempo son adecuados. En este caso se tendría que trabajar en la traducción de los objetos que maneja VHDL a los de software embebido y describir claramente los niveles de abstracción a los que se realizan el diseño.

4.2.4. Sistemas operativos y lenguajes en tiempo real.

Existen sistemas operativos en tiempo real tales como RTDOS, MTOS o lenguajes como Ada. En estos sistemas y lenguajes se maneja la concurrencia y el tiempo. Sin embargo, las metodologías que se han desarrollado para ellos no están bien desarrolladas y todavía falta mucho por hacer.

Tal vez el principal problema, es que la descomposición modular implica muchos problemas en el manejo del tiempo, en especial que se puede romper la calendarización por muchas razones como las derivas de tiempo y la inversión de prioridades.

4.3. Principales necesidades en el desarrollo de software embebido.

En el desarrollo de aplicaciones embebidas se requiere personal capacitado en ciertas áreas específicas orientadas hacia el desarrollo de la aplicación, así como también la herramienta de desarrollo necesaria para poder lograr el buen funcionamiento del software.

4.3.1. Ingeniería de computación.

Se debe contar con personal altamente calificado en el desarrollo de programas que tenga conocimientos adecuados de la ingeniería de software y de sus herramientas para especificar programas en la metodología adoptada. Además de contar con sólidas bases teóricas en esta área.

4.3.2. Ingeniería en electrónica.

Se debe contar con personal que conozca diferentes dispositivos electrónicos programables, tales como DSP y microcontroladores, que entienda de su arquitectura y de su programación, así como de la construcción de sistemas con base en ellos.

4.3.4. Ingeniería de procesos.

Se debe contar con personas que conozcan los procesos en todos sus aspectos y a profundidad. No basta con tener descripciones funcionales basadas en reglas que conduzcan sólo a un estilo de programación lógica correctiva, se debe contar con modelos entrada-salida, funcionales y de otros tipos que describan la dinámica interna de los procesos y sistemas. Todo ello con la finalidad de poder generar un buen conjunto de especificaciones que elimine cualquier tipo de ambigüedad del sistema y garantizar de este modo su seguridad.

4.3.5. Herramienta de diseño de software embebido.

Aunque esta es una herramienta que debería manejar toda persona relacionada con los sistemas embebidos, no es así debido a que ha sido muy complicado disponer de una única herramienta que modele todos los aspectos de los sistemas embebidos y reúna, al mismo tiempo, las distintas facetas de desarrollo del software.

Tal vez LabVIEW**,** es lo más parecido a una herramienta con éstas características. Por lo pronto la creación de más y mejores opciones en las herramientas de diseño será un área en mejora continua para ofrecer herramientas más versátiles y robustas ***[JAD2001]***.

4.4. Metodologías para el Desarrollo de un sistema embebido con características hard real-time.

Actualmente existe una metodología que ha sido muy aceptada en el diseño de sistemas embebidos con características hard real-time.

Esta metodología es la llamada metodología V, *Fig. 4.1*.

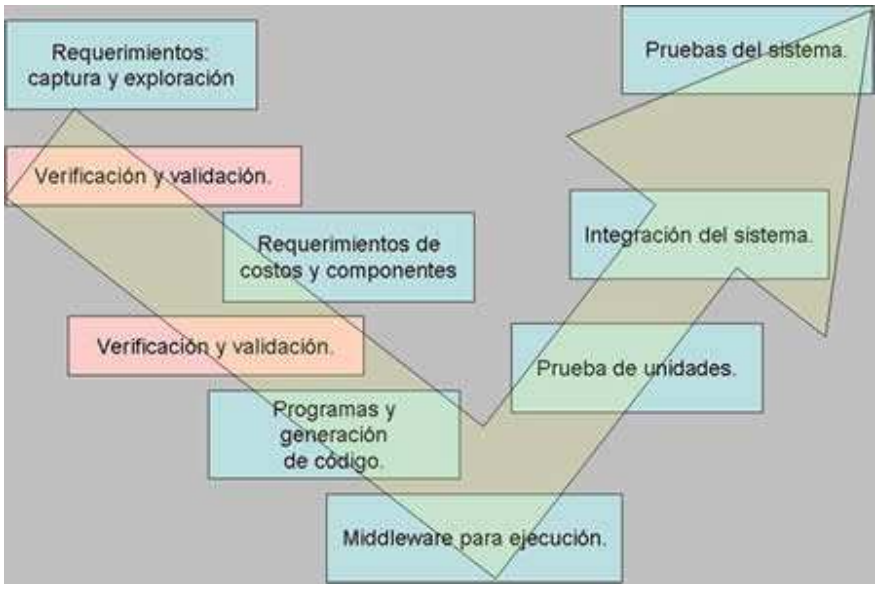


Fig. 4.1 La figura describe la metodología de diseño de sistemas embebidos hard real-time, conocida como ruta V Fuente: Sifakis, Joseph. The ARTIST roadmap for research and development. Springer Verlag. 2005.

En casi todas las áreas de diseño se tiene una metodología bien definida para su desarrollo, sin embargo, las pruebas de verificación y validación aún requieren ser tratadas de forma especial para cada caso de diseño, pues no están desarrolladas técnicas generales para llevarlas a cabo sobre cualquier tipo de sistema.

El diagrama muestra el ciclo de desarrollo típico del software embebido. El proceso inicia con la captura de las especificaciones, y se avanza en el análisis de la información hasta llegar a la creación de diversos módulos o unidades que por si mismos deben ser probados, después se integran estos módulos o unidades para formar el sistema completo y se realizan pruebas nuevamente para ver el funcionamiento de todo el sistema.

Es de gran relevancia aclarar que todas las fases deben usar en su preferencia herramientas formales, esto cumplirá con los más estrictos requerimientos de puestos por la industria.

Los desarrolladores de sistemas embebidos están cada vez más conscientes de los costos asociados con los retrasos, las cancelaciones de los diseños y las fallas de los productos finales para mantenerse en concordancia con las metas iníciales.

Las herramientas de simulación y modelado están ganando relevancia en su incorporación en las metodologías de diseño para solventar estos problemas.

A continuación se muestra un diagrama que sintetiza la metodología de diseño de software para sistemas embebidos de una forma muy detallada donde en la parte superior se inicia con las especificaciones y se termina con la liberación del producto, *Fig. 4.2*.

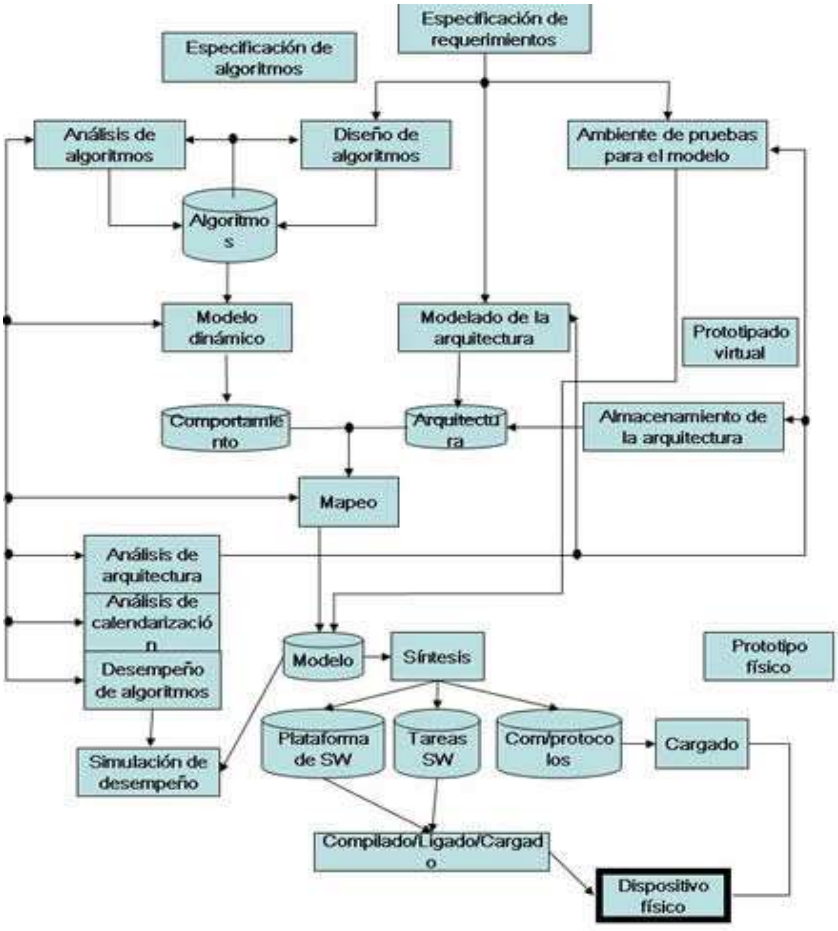


Fig. 4.2 La figura muestra un esquema detallado del desarrollo de software para un sistema embebido Fuente: Sifakis, Joseph. The ARTIST roadmap for research and development. Springer Verlag. 2005.

Finalmente se muestra otro diagrama de la metodología para el desarrollo de software embebido donde se observa el ciclo típico de desarrollo de dicho software. Se observa cómo se parte de la especificación y cómo se divide el trabajo en dos grandes áreas (que si bien progresa por separada cada una de ellas, no significa que estén incomunicadas), el ciclo avanza hasta que se libra el producto tras pruebas de aceptación, *Fig. 4.3*.

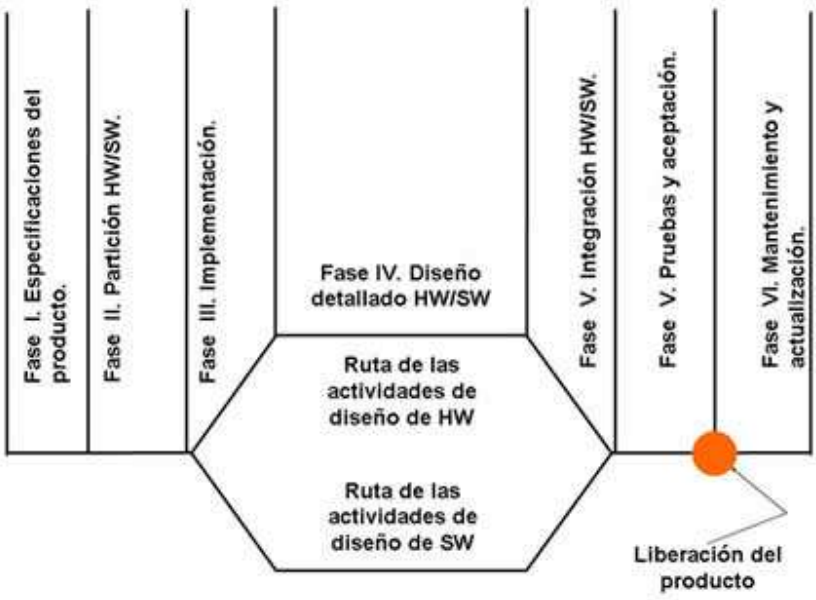


Fig. 4.3 La figura muestra el ciclo típico del desarrollo de software embebido. Fuente: Berger, Arnold. Embedded systems design. CMP Books. 2002.

Capitulo 5

Programando un Microcontrolador

Capitulo 5
Programando un Microcontrolador.

En el desarrollo de Sistemas Embebidos es de mucha importancia tener el control de lo que será el cerebro de nuestro sistema ya sea un Microcontrolador o Microprocesador.

Los lenguajes más comunes en el uso de la programación de un Microcontrolador actualmente son: Basic, C, Ensamblador y Java.

Para los ejemplos futuros, los cuales serán la realización de "Hola Mundo" programados en C y Basic, el prototipo final que serán expuestos, se utilizará el Compilador MikroBasic y MikroC. Apoyados de igual manera por un Simulador de Microcontroladores (PICS), Pic Simulator Ide.

5.1. Basic como una herramienta de trabajo.

Los compiladores así como los lenguajes están orientados totalmente al desarrollo de aplicaciones embebidas, los compiladores mismos ya cuentan con las herramientas necesarias para hacer la carga del programa una vez ya compilado, sabiendo que no tiene errores, a la memoria flash ya sea del PIC o de alguna Memoria Externa.

Tras una exitosa compilación MikroBasic genera los siguientes archivos:

- **Intel HEX (.hex)**
  Estilo Hexadecimal de Intel, este archivo se utiliza para programar el PIC una vez que ya ha sido cargado en el.
- **Librería Binaria Compilada (.mls)**
  Distribución binaria de aplicación que pueden ser incluidos en otros proyectos.
- **Archivos de Lista (. Lst)**
  Descripción del consentimiento fundamentado previo de asignación de memoria: se ocupa de la instrucción, los registros, rutinas, etc.
- **Archivo ensamblador (. asm)**
  Archivo legible ensamblado con nombres simbólicos, extraídos de los archivos de Lista.

Para el desarrollo de algunos ejemplos de código utilizaremos el lenguaje Basic como nuestra herramienta principal de trabajo, pero también tendremos algunos ejemplos desarrollados en C orientado al ámbito de los microcontroladores.

5.2. ¿Por qué utilizar Basic?

La respuesta es sencilla: es legible, fácil de aprender, lenguaje de programación estructurado, con suficiente capacidad y flexibilidad necesarias para programación de microcontroladores.

Si se ha tenido alguna experiencia de programación anterior, se podrá dar cuenta de que escribir programas en mikroBasic es muy fácil.

Uno de los compiladores de lenguaje BASIC para microcontroladores PIC que está atrayendo a muchos usuarios es MikroBasic. Aunque ya lleva un tiempo en el mercado, fue a partir de la versión 5.00 que se ha masificado. Con este tipo de compiladores ya no es necesario aprender el lenguaje ensamblador o conocer a fondo otras herramientas de programación para desarrollar software en proyectos con microcontroladores.

La empresa mikroElectrónica distribuye una serie de compiladores para microcontroladores, entre los que se destacan el mikroC y mikroBasic. Las características más destacadas de estos compiladores, y en particular del que nos ocupará en este capítulo es la inclusión de un IDE (entorno de desarrollo integrado o en inglés Integrated Development Environment) que hace muy cómoda la programación, ya que resalta la sintaxis del lenguaje, proporciona acceso muy rápido a la ayuda incluida, estadísticas sobre el uso de recursos del microcontrolador, entre otras ventajas.

5.3. Desarrollando "Hola Mundo".

Desarrollaremos a continuación nuestro primer programa "Hola Mundo"**,** como en todos los lenguajes de programación, que esto es un equivalente en el ámbito de los Microcontroladores en el encendido de un LED ***[MK2006]***.

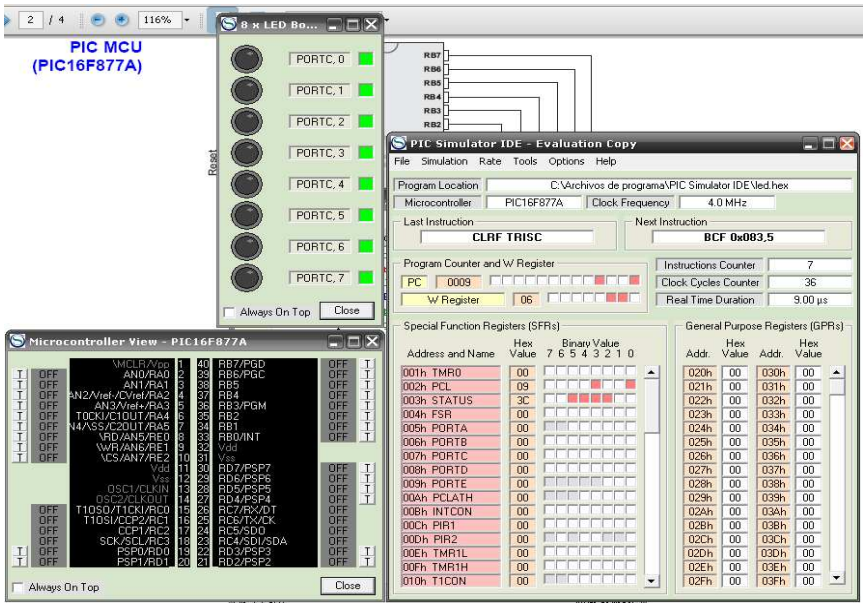
Para el desarrollo de nuestros programas y simulación consiguientes utilizaremos el PIC Microcontrolador PIC16F877A con arquitectura Harvard.

El código quedaría de la siguiente forma:

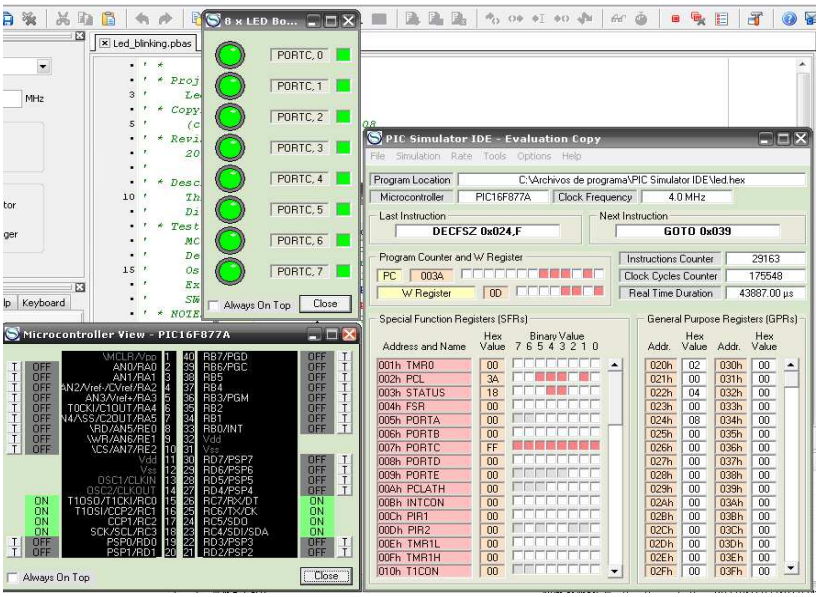
```
AllDigital              'Convierte todos los pines en pines de E/S.
TRISC = 0               'Configura el PuertoC como salida.
      loop:             'Inicializa el bucle infinito
      PORTC = Not PORTC 'Cambia el PuertoC
      WaitMs 500        'Espera 500 milisegundos (Medio Segundo)
Goto loop               'Regresa al bucle
```

La simulación acarrea resultados como los que a continuación se muestran:

Cuando el programa inicia todas las salidas del PuertoC están configuradas como salida y esto hará que pongan a 0 dichas salidas.



Al comenzar el programa y llegar a la instrucción `PORTC = Not PORTC` estamos negando los valores que le fueron asignados al iniciar el programa al PuertoC y pondrá su estado en 1, lo que provocará el encendido de los leds.



Al llegar a la instrucción `Goto loop` el programa comenzará un bucle infinito de repeticiones cambiando en cada una de estas el estado del PuertoC provocando el parpadeo de los leds.

Ahora se desarrollará el mismo programa pero sobre el compilador MikroC, quedando de la siguiente forma ***[MIB1999]***:

```
void main() {
      PORTC = 0;
      TRISC = 0;        // Configura los pines del PuertoC como salida
            while(1) {  // Inicializa bucle infinito
            PORTC = ~PORTC;   // Cambia el puertoC
            Delay_ms(500);    // Espera 500 milisegundos
      }
```

5.4. Variables, Instrucciones y Declaraciones en Basic.

BASIC tiene variables definidas para cada puerto (PORTA, PORTB, etc.) por lo que es muy simple poder interpretar el estado de las entradas de los Microcontroladores.

De igual manera BASIC nos permite el uso de variables con distintos tipos como Bit, Byte, Word y Long.

- **Bit** (un bit de longitud, almacena 0 o 1 únicamente)
- **Byte** (un byte de longitud, almacena números enteros entre 0 y 255)
- **Word** (dos bytes de longitud, almacena números enteros entre 0 y 65,535)
- **Long** (cuatro dos bytes de longitud, almacena números enteros entre 0 y 4,294,967,295)

A diferencia de otros BASIC, la declaración de variables puede ser hecha en cualquier parte del programa, y todas son consideradas globales, es decir, su valor es accesible desde todas las subrutinas y zonas del programa ***[MK2006]***.

El uso de estas variables esta limitado al tamaño de la memoria RAM con la que cuente el microcontrolador, estas variables se declaran mediante la instrucción DIM, quedando de la siguiente forma:

```
DIM A AS BIT
DIM TEMPERATURA AS BYTE
DIM TIEMPO AS WORD
DIM AUX AS LONG
```

Basic también nos permite utilizar instrucciones para realizar toma de decisiones basándose en el estado de una entrada o el valor de una variable, siendo la mas sencilla y frecuentemente utilizada la sentencia IF - THEN - ELSE - ENDIF.

Resulta muy sencillo entender su funcionamiento si traducimos su significado al español. IF - THEN - ELSE – ENDIF significa algo así como “SI ocurre tal cosa ENTONCES realizo esta tarea SINO hago esta otra FIN SI”, como se maneja en otros lenguajes de programación como C y C++.

Existen varias formas de utilizar esta instrucción en BASIC. Veremos los tres casos posibles, comenzando por el más sencillo.

El caso más simple es el siguiente:

```
IF condición THEN instrucción
```

Como vimos, "IF" significa "SI....", y "THEN" significa "LUEGO" o "ENTONCES". Por lo tanto, el caso anterior puede leerse como "SI se cumple la condición, entonces ejecuto la instrucción"

Algunos ejemplos validos de este caso:

```
IF PULSADOR = 1 THEN PORTA.0 = 1
IF B > A THEN LED=ON
IF B = 5 THEN A = 0
IF (A = 0) OR (B = 5) THEN C = 2
IF PORTA.0 THEN PORTB.3 = 0
```

Nuestro segundo caso es cuando luego de evaluar la condición necesitamos ejecutar más de una instrucción. En los ejemplos vistos en el caso anterior siempre se ejecutaba una sola instrucción cuando la condición era cierta. La manera de ejecutar múltiples instrucciones dentro de una estructura IF-THEN implica emplear el ENDIF, con lo que la sintaxis se observaría de la siguiente manera.

```
IF condición THEN
    instrucción 1
    instrucción 2
    ...
   instrucción n
ENDIF
```

Como se podrá observar varia muy poco del primer caso. Se observa que en este caso se van a ejecutar las instrucciones que se encuentren entre el THEN y el ENDIF, esto cada vez que se cumpla la condición.

En el tercer caso se presentan las instrucciones anidadas, con las instrucciones IF-THEN-ELSE-ENDIF, con lo que se pueden tomar decisiones verdaderamente complejas, con forma de "árbol", donde cada condición representa una "rama" diferente.

En este caso debemos tener cuidado al utilizar esta característica ya que debido a las limitaciones en la capacidad de la pila y la cantidad de memoria disponibles en el PIC podría llegar a ocasionarse un desborde provocando que el programa se colapse. Un ejemplo de instrucciones anidadas se observaría de la siguiente manera:

```
IF PORTB.1 = 1 THEN
    IF A = 2 THEN
        A = B + (C * D)
        TOTAL = A * B
    ELSE
        A = 0
    ENDIF
ELSE
    A = 19
ENDIF
```

Como es de notar y de observar Basic permite el uso de etiquetas, mediante la instrucción GOTO, para componer ciclos, el problema que puede tener este tipo de instrucciones, es que este método se utiliza de manera rudimentaria y solo permite que el bucle en cuestión se repetirá un numero infinito de veces, como se pudo observar en la primera simulación realizada.

Sin embargo para resolver problemas donde se necesiten ciclos que requieran un cierto número para detenerse, BASIC también contiene instrucciones dirigidas a resolver este tipo de situaciones como son FOR - TO - STEP – NEXT y WHILE - WEND

FOR - TO - STEP - NEXT

Esta estructura, necesita de una variable tipo Byte o Word para funcionar. En cada iteración del ciclo, la variable va cambiando su valor. Cuando el valor de la variable alcanza o supera el valor prefijado, el ciclo termina. La forma del ciclo quedaría de la siguiente forma:

```
FOR variable = valor_inicial TO valor_final STEP paso
  instruccion1
  instruccion2
  ...
  instruccionn
NEXT variable
```

La cuenta comienza asignando a “variable” el “valor_inicial”, y termina cuando “variable” alcanza el valor “valor_final”. En cada iteración del bucle el valor de la “variable” se incrementa el valor fijado por “paso”.

WHILE - WEND

La otra estructura de control que proporciona Basic es WHILE - WEND. Su propósito también es la construcción de ciclos que se ejecutan un número de veces, se considera como el punto intermedio entre la instrucción GOTO y la instrucción FOR…NEXT. Su estructura es la siguiente:

```
WHILE condición
  instruccion1
  instruccion2
  ...
  instruccionn
WEND
```

Mientras que la condición sea verdadera, el grupo de instrucciones dentro del cuerpo del WHILE-WEND se ejecuta. Las características de la condición son las mismas que para la instrucción IF-THEN-ELSE-ENDIF.

5.5. Manipulando un modulo LCD con Basic.

MikroBasic también nos permite el manejo de displays LCD. En el mercado existe una gran variedad de pantallas de cristal liquido, de un precio accesible, con características comunes en cuanto a la interfaz y programación, gracias que la mayoría utiliza para comunicarse con el “exterior” el mismo chip de la empresa Hitachi, el HD44780, haciendo posible que con unas cuantas instrucciones podamos manejar desde un sencillo display de una línea de 8 caracteres hasta uno de 4 líneas con 80 caracteres.

Estos displays básicamente son una pequeña placa de circuito impreso con un par de integrados (tipo “gota”) pegados en una de sus caras, y la pantalla propiamente dicha en la otra, rodeada de una estructura metálica que la protege. Esta placa casi siempre dispone de agujeros para poder fijar el conjunto a un chasis o gabinete sin grandes complicaciones. Desde el punto de vista eléctrico, hay un conector (a veces solo agujeros metalizados donde soldar los cables) que tiene 14 pines en los que no poseen iluminación propia (backlite) o 16 en los que si la tienen.

Con las instrucciones apropiadas este tipo de circuitos son capaces de representar caracteres, mostrar o esconder un cursor, borrar la pantalla, etc.

Como se mencionó, la gran mayoría de los displays existentes en el mercado respetan la misma distribución de pines. A continuación veremos la función de cada de uno de ellos.

Pines 1,2 y 3: Estos pines están dedicados a la alimentación y contraste del LCD. El pin 1 (VSS) es el que se debe conectar al negativo de la alimentación, y el pin 2 (Vdd/Vcc) es el que va unido al positivo (5 voltios). El pin 3 permite el ajuste del contraste del panel. Se puede unir al pin 1 mediante una resistencia de 220 ohms para obtener un contraste adecuado (pero fijo) o bien utilizar un potenciómetro o preset de 10 Kohm para variar el contraste a gusto.

Pines 4, 5, 6: Estos pines son de alguna manera los que controlan el funcionamiento del display. El pin 4, también llamado RS (Registration Select) es el que le indica al controlador interno del LCD que el valor presente en el bus de datos es un comando (cuando RS=0) o bien un carácter para representar (cuando RS=1).

El pin 5 ("R/W" por "Read/Write" o "Leer/Escribir") permite decidir si queremos enviar datos al display (R/W=0) o bien nos interesa leer lo que el display tiene en su memoria o conocer su estado (R/W=1).

Por ultimo, el pin 6 (E por "Enable" o "habilitado") es el que selecciona el display a utilizar. Es decir, podemos tener varios LCD conectados a un mismo bus de datos (pines 7-14) de control, y mediante E seleccionar cual es el que debe usarse en cada momento.

Pines 7, 8, 9, 10, 11, 12, 13, 14: Estos ocho pines son el "bus de datos" del controlador de la pantalla. Llamados DB0-DB7 son los encargados de recibir (o enviar) los comandos o datos desde o hacia el display. DB0 es el bit de menor peso y DB7 es el más significativo.

Por ultimo, los pines 15 y 16 son los que se utilizan para alimentar el (o los) leds de fondo de la pantalla, que brindan la iluminación. El pin 15 debe ser conectado a 5 voltios y el 16 al negativo. En estas condiciones, la luz de fondo esta encendida a 100% de su brillo. Nuevamente, se puede utilizar un potenciómetro o preset para ajustar el brillo, *Fig. 5.1* ***[i14]***.

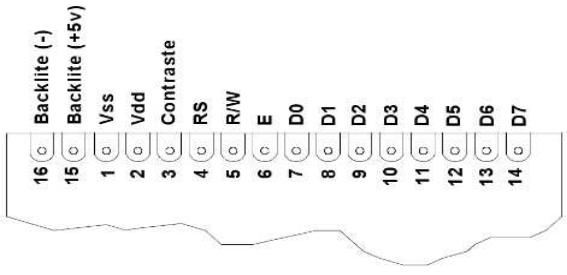


Fig. 5.1 La imagen muestra los pines de un LCD genérico

Estos displays soportan dos modos de trabajo: en uno de ellos reciben en DB0-DB7 los 8 bits del dato, y en el otro, llamado “modo de 4 bits” reciben los datos en dos mitades (nibbles) por los pines DB4-DB7, en dos pasos sucesivos. Si bien esto puede complicar ligeramente la programación en Ensamblador, en Basic es completamente transparente, a la vez que supone un ahorro de 4 pines en el bus de datos, y esto en microcontroladores con pocos pines de I/O es muy útil.

Basic dispone de un juego de instrucciones especiales para manejar displays en modo “8 bits” y en modo “4 bits” que nos evitan toda la complejidad en la programación de estos circuitos. Veremos cuáles son y algunos ejemplos de uso.

El manejo de los LCD se hace mediante el uso de sentencias “DEFINE”, que le dicen al compilador a que pines del microcontrolador hemos conectado cada uno de los pines del LCD. La forma de la instrucción DEFINE es la siguiente:

DEFINE parametro = valor

Donde “parametro” es el nombre del parámetro al que le queremos asignar el “valor”. Los parámetros disponibles para el manejo de LCD alfanuméricos son los siguientes:

- LCD_BITS: Define el número de bits de la interfaz de datos. Se pueden asignar valores de 4 u 8, siendo 4 el valor por defecto.

- LCD_DREG: Define a que puerto del PIC tenemos conectado el port de datos del LCD. Los valores permitidos son PORTA, PORTB, PORTC, etc. Por defecto se asume PORTB.

- LCD_DBIT: Define cual es el primer pin del puerto que usamos para enviar los datos al LCD cuando seleccionamos un bus de 4 bits. Solo puede ser el 0 (para los pines el 0, 1, 2 y 3) o 4 (para usar los pines 4, 5, 6 y 7). Por defecto se asume “4”, y esta instrucción se ignora para LCD_BITS = 8.

- LCD_RSREG: Define a que puerto del PIC tenemos conectado el pin RS del LCD. Los valores permitidos son PORTA, PORTB, PORTC, etc. Por defecto se asume PORTB.

- LCD_RSBIT: Define a que pin del puerto tenemos conectado el pin RS del LCD. Por defecto se asume “3”.

- LCD_EREG: Define a que puerto del PIC tenemos conectado el pin E del LCD. Los valores permitidos son PORTA, PORTB, PORTC, etc. Por defecto se asume PORTB.

- LCD_EBIT: Define a que pin del puerto tenemos conectado el pin E del LCD. Por defecto se asume “2”.

- LCD_RWREG: Define a que puerto del PIC tenemos conectado el pin RW del LCD. Los valores permitidos son 0, PORTA, PORTB, PORTC, etc. Por defecto se asume “0”, que significa “no usamos el pin RW”.

- LCD_RWBIT: Define a que pin del puerto tenemos conectado el pin RW del LCD. Por defecto se asume “0”, que significa “no usamos el pin RW”.

- LCD_COMMANDUS: Define cuantos microsegundos demora la escritura de un comando en el display. Por defecto, este valor es de 5000. La mayoría de los LCD funcionan bien con un valor de 2000, lo que hace más rápidos nuestros programas.

- LCD_DATAUS: Define cuantos microsegundos demora la escritura de un dato en el LCD. Por defecto, este valor es de 100.

- LCD_INITMS: Define cuantos microsegundos demora la inicialización de la electrónica del LCD. Por defecto, este valor es de 100.

También existen una serie de instrucciones que manejan el envió de comandos e instrucciones a un display.

- LCDINIT debe utilizarse antes de enviar cualquier comando o dato al LCD. La forma de esta instrucción es al siguiente:

  LCDINIT n

  Donde “n” es el tipo de cursor que queremos que muestre el display. “0” significa que el cursor estará oculto, “1” significa que el cursor parpadeara, “2” nos mostrara un cursor subrayado, y “3” un cursor subrayado y parpadeando.

- LCDCMDOUT es la instrucción que envía comandos al LCD. Se emplea de la siguiente manera:

  LCDCMDOUT comando

  Donde “comando” es alguno de los siguientes:
  - LcdClear: Borra el contenido del LCD.
  - LcdHome: Lleva el cursor a la primera posición del primer renglón del LCD.
  - LcdLine2Home: Lleva el cursor a la primera posición del segundo renglón del LCD.

- LcdLeft: Mueve el cursor una posición a la izquierda.
- LcdRight: Mueve el cursor una posición a la derecha.
- LcdShiftLeft: Desplaza el contenido del LCD una posición a la izquierda.
- LcdShiftRight: Desplaza el contenido del LCD una posición a la derecha.
- LcdLine1Clear: Borra la primera línea del LCD.
- LcdLine2Clear: Borra la segunda línea del LCD.
- LcdLine1Pos(x): Coloca el cursor en la posición “x” del primer renglón del LCD. “X” puede tener cualquier valor entre 1 y 40.
- LcdLine2Pos(x): Coloca el cursor en la posición “x” del segundo renglón del LCD. “X” puede tener cualquier valor entre 1 y 40.

- LCDOUT envía datos al display. Si son caracteres, simplemente los ponemos entre comillas a continuación del comando. Si se trata de mostrar el contenido de una variable, se escribe la variable (precedida por “#”) a continuación del comando. Si se necesitan imprimir varias variables, se pueden separar por “comas”.

A continuación mostraremos un ejemplo de cómo comenzar a utilizar estas instrucciones:

```
DEFINE LCD_BITS = 8
DEFINE LCD_DREG = PORTB
DEFINE LCD_DBIT = 0
DEFINE LCD_RSREG = PORTD
DEFINE LCD_RSBIT = 1
DEFINE LCD_EREG = PORTD
DEFINE LCD_EBIT = 3
DEFINE LCD_RWREG = PORTD
DEFINE LCD_RWBIT = 2

LCDINIT 0 ‘inicializo el LCD sin cursor.
LCDOUT "FCC-BUAP" ‘Muestra el texto
```

Capitulo 6

Desarrollando un prototipo

Capítulo 6
Desarrollando un Prototipo.

Ya que se han visto algunas definiciones, componentes, arquitecturas, y herramientas de desarrollo, se pasara a la creación de un prototipo en el cual, se manipulará un modulo LCD y un conjunto de 8 Leds, El elemento principal que se utilizará será el microcontrolador PIC16F877A, que cuenta con una arquitectura Harvard, el por que se utiliza esta arquitectura es debido a su gran presencia en los aparatos electrónicos (Estereos, TV´s, DVD´s, Controles Domoticos, etc.) esta es muy utilizada como subsistemas de este tipo de aparatos por su gran capacidad en el monitoreo de los estados de sus compuertas y el tamaño muy pequeño de otros microcontroladores que utilizan esta arquitectura. De igual manera los simuladores de este tipo de tecnología ocupan mucho este tipo de microcontroladores para realizar las simulaciones de trabajo de estos.

El lenguaje que se utilizará para la programación del microcontrolador será Basic, que como ya se ha visto en el capitulo anterior es relativamente sencillo de aprender y esta tomando una gran fuerza en el ámbito de la programación de estos componentes.

Bien, la función que realizará el software será que mediante un botón de encendido este se comience a ejecutar realizando la función de un contador el cual al finalizar su “conteo” muestre algún mensaje ya definido.

Los principales componentes que se utilizarán, como ya se ha mencionado, para el desarrollo de este prototipo son los siguientes: El microcontrolador PIC16F877A, un conjunto de 8 Leds y un modulo LCD de 2 x 16 caracteres.

6.1. Esquemas de las principales herramientas a utilizar.

Los esquemas de estos componentes se muestran a continuación:

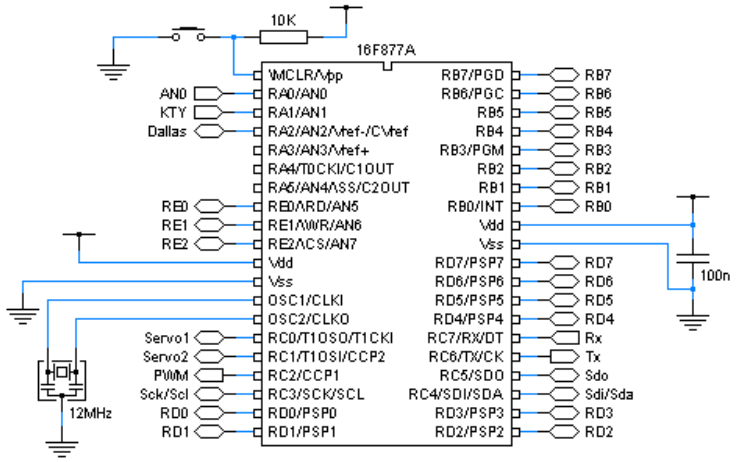


Fig. 6.1 La imagen muestra el esquema de conexión del microcontrolador PIC16F877A con arquitectura Harvard.

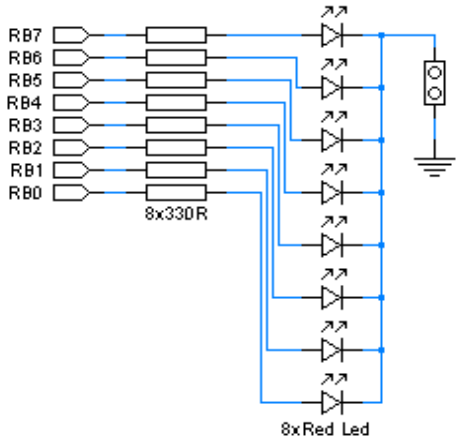


Fig. 6.2 La imagen muestra el esquema de conexión de un conjunto de 8 Leds.

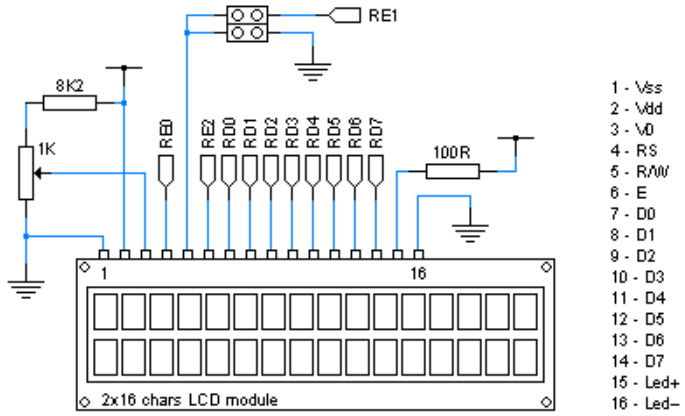


Fig. 6.3 La imagen muestra el esquema de conexión de un Modulo LCD Genérico de 2x16 caracteres.

6.2. Analizando las conexiones de las herramientas.

A continuación observaremos la forma en que físicamente deberán hacerse las conexiones para este proyecto.

El conjunto de 8 Leds, *Fig. 6.2*, se conectaran al Puerto C del microcontrolador, *Fig. 6.1*, esto quiere decir que todas las compuertas serán utilizadas y cada led corresponderá a cada una de ellas.

```
PORTC.0
PORTC.1
PORTC.2
```

```
PORTC.3
PORTC.4
PORTC.5
PORTC.6
PORTC.7
```

El Modulo LCD, *Fig. 6.3*, se conectará a los Puertos B y D del microcontrolador, *Fig. 6.1*, el tamaño del bus de datos (de 8 bits) quedará definido mediante PORTB, el pin PORTD.1 se emplea para el control de la línea RS del LCD, el pin PORTD.3 para la línea E y el control de R/W está a cargo del PIC, mediante el pin PORTD.2, como se muestra a continuación:

```
Define LCD_BITS = 8
Define LCD_DREG = PORTB
Define LCD_DBIT = 0
Define LCD_RSREG = PORTD
Define LCD_RSBIT = 1
Define LCD_EREG = PORTD
Define LCD_EBIT = 3
Define LCD_RWREG = PORTD
Define LCD_RWBIT = 2
```

6.3. El Código y su análisis.

Una vez definidas las conexiones pasamos al código, que en este caso quedaría de la siguiente manera:

```
Define LCD_BITS = 8     'Se definen los puertos que controlaran
Define LCD_DREG = PORTB 'el modulo LCD
Define LCD_DBIT = 0
Define LCD_RSREG = PORTD
Define LCD_RSBIT = 1
Define LCD_EREG = PORTD
Define LCD_EBIT = 3
Define LCD_RWREG = PORTD
Define LCD_RWBIT = 2

Dim a As Word           'Declaración de Variables
a = 10

Lcdinit 3               'Inicio de LCD
WaitMs 500
```

```
loop:                    'Comienza el primer ciclo
Lcdout "Espera..."
Lcdcmdout LcdLine2Home
Lcdout #a
a = a - 1
WaitMs 500
Lcdcmdout LcdClear
If a = 0 Then                  'Condicional del primer ciclo

      Lcdcmdout LcdClear
      loop2:                   'Comienza el ciclo infinito
      TRISC = %00000000
      Lcdout "..FCC - BUAP.."
      WaitMs 500
      PORTC = %11111111
      Lcdcmdout LcdLine2Home
      Lcdout "...PRESENTE..."
      WaitMs 001
      PORTC = %00000000
      Lcdcmdout LcdClear

Goto loop2
Endif
Goto loop
```

Analizaremos de forma general el código para entender su funcionamiento.

Con el primer grupo de instrucciones como ya se explicó anteriormente se controlará el modulo LCD, la instrucción `Dim` nos permite declarar diferentes tipos de variables, en este caso declaramos la variable "a" de tipo Word (dos bytes de longitud, almacena números enteros entre 0 y 65,535). Cabe aclarar que el tamaño de las variables depende de la cantidad de memoria RAM con la que cuente el microcontrolador.

Con la instrucción `Lcdinit 3` inicializará el LCD con el cursor parpadeando, y esperará medio segundo para continuar con las siguientes instrucciones mediante `WaitMs 500.`

Se crea el primer ciclo con `loop` el cual terminará cuando cumpla la condición que se impuso.

`Lcdout` escribirá en el primer renglón del Modulo LCD un mensaje, en este caso será `Espera…`, la siguiente instrucción `Lcdcmdout LcdLine2Home` nos posicionará en el segundo renglón para que ahora nuevamente con `Lcdout` se imprima el contenido de una variable, esto lo puede hacer anteponiendo el símbolo `#` a la variable.

Posteriormente realizamos una resta al contenido de la variable espera medio minuto y limpia la pantalla del Modulo LCD con la instrucción `Lcdcmdout LcdClear`.

La instrucción `IF` será la que tenga el control de cuando terminará el primer ciclo ya que espera a que el valor de la variable a tenga el valor de 0 para poder ejecutar el siguiente grupo de instrucciones, siendo su función presentar en el Modulo LCD la función de un contador de modo inverso que ira del valor de la variable a cero.

Al cumplirse el ciclo pasará a ejecutar el siguiente grupo de instrucciones, donde se ejecutará un ciclo infinito con instrucciones semejantes a las que se realizan dentro del primer ciclo, las instrucciones que son distintas son: `TRISC = %00000000`, `PORTC = %11111111` y `PORTC = %00000000`, La función de TRISC es que convierte todos los pines del Puerto C en salidas El "%" indica que el numero que viene a continuación esta en binario. Se podría haber escrito, por ejemplo TRISC = 0 y hubiera sido lo mismo, también es válido activar como entrada algunos pines, y como salidas otros, haciendo algo parecido a TRISB = %11000111. Con la instrucción `PORTC = %11111111` se activan como entrada todos los pines y con `PORTC = %00000000` nuevamente las activa como salida, su función en este caso es que encenderá y apagará el conjunto de Leds.

6.4. La simulación Final.

La simulación nos da los siguientes resultados.
Realiza el conteo de 10 a 0.

Fig. 6.4, Las imágenes muestran la simulación con el Software PIC Simulator IDE de un contador que en este caso ira de 10 a 0.

El siguiente paso que realizará es imprimir mensajes distintos en el Modulo LCD, encenderá y apagará un conjunto de Leds, *Fig. 6.5*. Una vez que aparezca el primer mensaje encenderá los Leds y cuando termine de aparecer el mensaje del segundo renglón apagará los Leds, *Fig. 6.6*.

Fig. 6.5, La imagen muestra la simulación con el Software PIC Simulator IDE de la escritura de mensajes en un Modulo LCD y la manipulación de un conjunto de Leds.

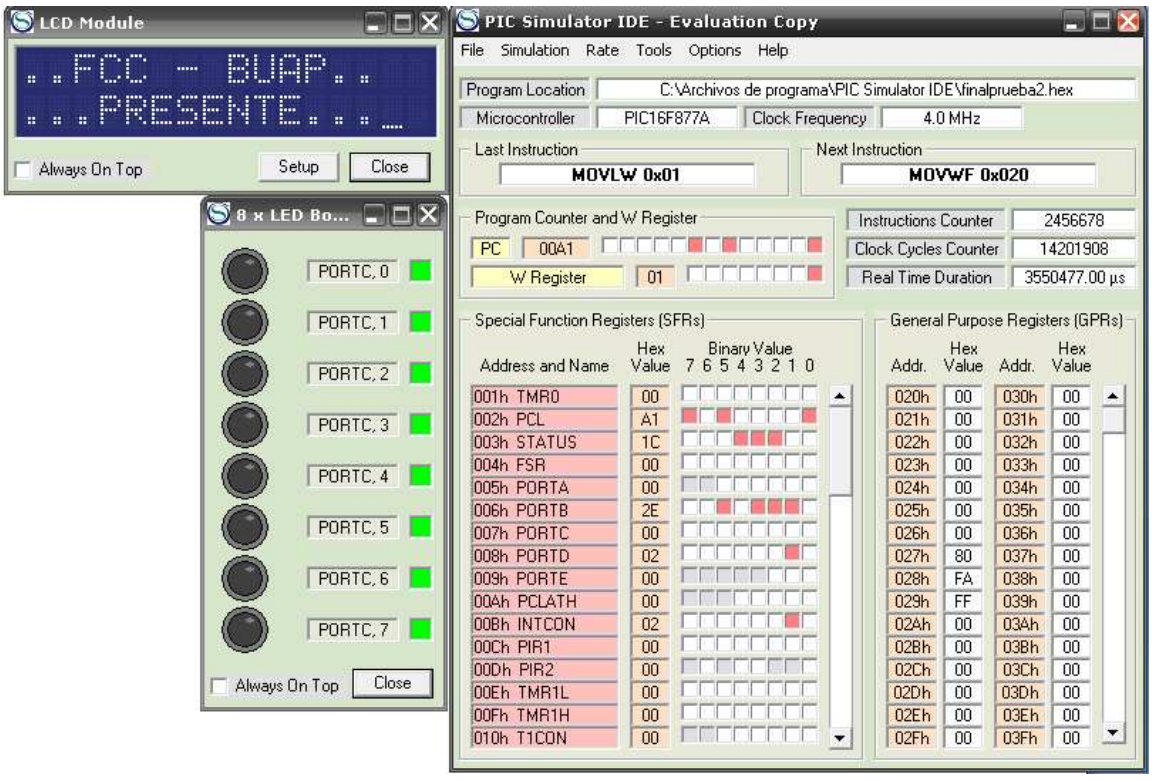


Fig. 6.6, La imagen muestra la simulación con el Software PIC Simulator IDE de la escritura de mensajes en un Modulo LCD y la manipulación de un conjunto de Leds.

El desarrollo de este proyecto tuvo una relativa sencillez debido a las diversas herramientas que se utilizaron ya que si este código hubiese sido escrito en lenguaje ensamblador hubiera sido tres o cuatro veces mayor que el presentado en este capítulo, cabe aclarar que uno de los objetivos que debe cumplir un Sistema Embebido es el ahorro de Memoria. De ahí que lenguajes como Basic y C orientados a Sistemas Embebidos nos brindan una buena opción en el desarrollo de estos.

Conclusiones.

En el desarrollo de este trabajo se trato de despertar el interés en los Sistemas Embebidos, ya que existen cifras que nos dicen que el número de programadores es menor al que realmente se necesita.

A estos días ya existen herramientas que permiten el desarrollo de este tipo de sistemas de una manera más sencilla, que permite el ahorro tanto de memoria en el microcontrolador como el tiempo de desarrollo.

El prototipo realizado esta basado en una de las arquitecturas más utilizada, aunque muy poco conocida, esta es la arquitectura Harvard, la cual es utilizada para el microcontrolador PIC16F877A que es el cerebro del prototipo. El software influyo mucho sobre algunas limitaciones en el diseño de este sistema prototipo ya que las herramientas disponibles en el mercado son demasiado caras y las versiones de prueba tienen muchas limitaciones. Sin embargo con el software PIC Simulator IDE se logro una buena simulación, en este caso de la manipulación de un modulo LCD y un conjunto de 8 Leds. El lenguaje que se utilizo para la programación del microcontrolador fue Basic, ya que actualmente está tomando una gran importancia para el desarrollo de Software Embebido.

Un estudiante de la Facultad de Ciencias de la Computación de la BUAP no tendría problema alguno para desarrollar estos sistemas ya que las disciplinas impartidas son una buena base en la creación de los Sistemas Embebidos.

La importancia que están teniendo en la vida cotidiana y la poca información que existen sobre los Sistemas Embebidos han provocado que exista más necesidad de desarrolladores de estos sistemas principalmente en América donde se pronostica un crecimiento menor en el desarrollo tanto de Hardware y Software para aplicaciones embebidas, en comparación con Asia y Europa.

Trabajos a Futuro.

- Desarrollar una nueva metodología para el desarrollo de Software Embebido.
- Realizar Software Embebido más complejo.
- Ensamblar o adquirir una placa Embebida para realizar pruebas con el software en ella.
- Realizar investigaciones más detalladas sobre las arquitecturas para poder realizar en un futuro el diseño de una de estas.

Bibliografía.

[MK2006] **Manual de Usuario Mickrobasic 5.0**
Autor: MikroElektronica.
2006
http://www.mikroe.com

[JOW2003] **Introducción a los sistemas de microcomputadores empotrados**
Autor: Jonathan W. Valvano
2003
Edit. Thomson Learning Ibero

[PEM2003] **Embedded System Design**
Autor: Peter Marwedel
2003
Edit. Springer

[MIB1999] **Programming Embedded Systems in C and C++**
Autor: Michael Barr
1999
Edit. O'Reilly

[DAE1999] **An embedded software primer**
Autor: David E. Simon
1999
Edit. Addison-Wesley

[PRO2006] **La industria del software en México en el contexto internacional y latinoamericano**
Autor: Prudencio Óscar Mochi Alemán
2006
Edit. UNAM

[TAN2005] **Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers**
Autor: Tammy Noergaard
2005
Edit. Newnes

[JAD2001] **Languages, Compilers, and Tools for Embedded Systems**
Autor: Jack Davidson, Sang Lyul Min
Colaborador: Jack Davidson
2001
Edit. Springer

[ROT2003] Sistemas digitales: principios y aplicaciones
Autor: Ronald J. Tocci, Widmer Neal
2003
Edit. Pearson Educación

[MOM2003] **Diseño digital**
Autor: M. Morris Mano
2003
Edit. Pearson Educación

Referencias de Internet	**Consultado en:**
[i1] http://www.teisa.unican.es/gim/es/tema?id=4	*[Diciembre, 2007]*
[i2] http://cordis.europa.eu/express/	*[Diciembre, 2007]*
[i3] http://www.ikerlan.es/_bin/cast/ifrnoticia.asp?id_noticia=59	*[Enero, 2008]*
[i4] http://www.consultoras.org/frontend/aec/Mexico-Listo-Para-El-Mercado-Mundial-De-TI-vn7186-vst123	*[Febrero, 2008]*
[i5]http://fumec.org.mx/v4/index.php?option=com_content&task=view&id=37&Itemid=88	*[Febrero, 2008]*
[i6] http://www.manufacturaweb.com/nivel2.asp?cve=153_07	*[Febrero, 2008]*
[i7] http://www.netrino.com/Embedded-Systems/Glossary	*[Febrero, 2008]*
[i8] http://www.mechulk.com/docs/arm.pdf	*[Mayo, 2008]*
[i9] http://www.mips.com	*[Mayo, 2008]*
[i10] http://www-03.ibm.com/chips/power/	*[Mayo, 2008]*
[i11] http://www.power.org/news/articles/new_brand/#isa	*[Mayo, 2008]*
[i12] http://www.st.com/stonline/press/news/year2001/c1000p.htm	*[Mayo, 2008]*
[i13] http://www.ni.com/compactrio	*[Mayo, 2008]*
[i14] http://www.mikroe.com	*[Julio, 2008]*

Apéndice I.

LabVIEW.

LabVIEW es una herramienta de software líder en la industria para pruebas de diseño, medida y sistemas de control. Desde su introducción en 1986, ingenieros y científicos alrededor del mundo que han confiado, durante todo el ciclo de diseño, en el desarrollo gráfico de proyectos de NI LabVIEW han logrado mejor calidad, acortar tiempos de mercado y mejorar la eficiencia de la ingeniería y manufactura. Usted puede aumentar la productividad en toda su organización al usar el entorno integrado de LabVIEW para establecer una interfaz con señales de tiempo real, analizar datos para información significativa y compartir resultados. Como que LabVIEW posee la flexibilidad de un lenguaje de programación, combinado con herramientas adicionales diseñadas específicamente para test, medida y control, usted puede crear aplicaciones que van desde la simple monitorización de temperatura hasta la simulación y diseño de sistemas de control. Sin importar de qué proyecto se trate, LabVIEW tiene las herramientas necesarias para que usted tenga éxito rápidamente.

LabVIEW es utilizado en diferentes ámbitos, siendo los más destacados:

- **Análisis automatizado y plataformas de medida:**
  - Test de fabricación
  - Test de validación/medioambiental
  - Test mecánico/estructural
  - Test de fiabilidad en tiempo real
  - Adquisición de datos
  - Test de campo portátil
  - Test de RF y comunicaciones
  - Test en bancos de prueba
  - Adquisición de imagen

- **Medidas industriales y plataformas de control:**
  - Test y control integrado
  - Automatización de máquinas
  - Visión artificial
  - Monitorización de condiciones de máquina
  - Monitorización distribuida y control
  - Monitorización de potencia

- **Diseño embebido y plataformas de prototipaje**
  - Diseño y análisis de sistemas empotrados

  - Diseño de control
  - Diseño de filtros digitales
  - Diseño de circuitos electrónicos
  - Diseño mecánico
  - Diseño de algoritmos

- **Docencia:** LabVIEW es ideal tanto para profesores como para investigadores y estudiantes. Las licencias departamentales y de campus son ideales para implantar la herramienta en los planes de estudio de las universidades.

Fuente: http://www.ni.com/labview