UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

ÁREA ACADÉMICA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

“PROPUESTA DE TECNOLOGÍA PARA EL DESARROLLO DE PRÁCTICAS DEL SISTEMA DE DESARROLLO SIS9S12E“

TESIS

QUE PARA OBTENER EL TITULO DE LICENCIADO EN INGENIERA ELECTRÓNICA Y TELECOMUNICACIONES

Presenta

RODOLFO REYES SERRANO

ASESOR DE TESIS: M. EN C. NORBERTO HERNÁNDEZ ROMERO

PACHUCA DE SOTO, HIDALGO, NOVIEMBRE DE 2007

ÍNDICE

Descripción

Capítulo III Introducción a la programación del sistema de desarrollo SIS9S12E con CodeWarrior.

Capítulo IV Propuesta de tecnología para el desarrollo de prácticas del sistema de desarrollo SIS9S12E.

Conclusiones

Anexos

ÍNDICE DE FIGURAS

Descripción **Página**

ÍNDICE DE TABLAS

Descripción **Página**

RESUMEN

En este trabajo se presenta el desarrollo de un manual de prácticas del sistema de desarrollo SIS9S12E, éste dispositivo pertenece a la familia de microcontroladores MC9s12E de Freescale.

Al desarrollar ésta propuesta de tecnología, basada en la realización del manual de prácticas del sistema de desarrollo sis9s12E se manejaron diversos recursos del microcontrolador como: Memoria, Comunicación, Timers, PWM (Modulación por ancho de pulso), ADC (Convertidor Analógico-Digital), del mismo modo se utilizó DAC (Convertidor Digital-Analógico) y LCD, los programas desarrollados en lenguaje C para usar dichos recursos son sencillos y compactos.

En el proyecto de tesis se desarrollaron prácticas para cada uno de los recursos del microcontrolador, primero se manejaron los puertos de entrada/salida, luego se utilizó el Timer- PWM, posteriormente la comunicación serie para el manejo de los convertidores, y finalmente la combinación de todos los recursos.

La combinación del uso de los recursos permite realizar prácticas como: Generador de Funciones (Senoidales, Cosenoidales, Triangulares, Exponenciales.), Desfasamiento de Funciones, Control de PWM para la manipulación de Servomotores, Selección de señales, manejo y configuración de LCD.

También se manejaron diversos software como: *CodeWarrior* que se utilizó para la programación del microcontrolador en lenguaje C, *LabVIEW* como un interfaz gráfico para la manipulación de PWM, y *Niplesoft* como un programa de visualización de microcontroladores.

La configuración de estos recursos de forma fácil y sencilla fue gracias al uso de beans en el sistema de programación CodeWarrior.

Para la visualización de resultados de algunas prácticas se utilizó un instrumento fundamental como el osciloscopio.

OBJETIVO GENERAL

Proporcionar los fundamentos de la tarjeta de desarrollo Freescale de 16 bits MC9S12Exx y la metodología de programación mediante el sistema de desarrollo integrado CodeWarrior por medio de un manual de prácticas, el cual permitirá tener un amplio conocimiento del manejo de los recursos y aplicaciones en Ingeniería en Electrónica.

HIPÓTESIS

El desarrollo del manual de prácticas del microcontrolador como el sis9s12E permitirá un manejo y entendimiento amplio de recursos de gran importancia en áreas de la ingeniería electrónica, permitiendo también la realización de diversos proyectos en diversas áreas de dicha ingeniería.

JUSTIFICACIÓN

Debido a los grandes recursos que maneja el sistema de desarrollo sis9s12E es utilizado para darle diversas aplicaciones en electrónica y para la vida cotidiana.

La razón para realizar este manual de prácticas es mostrar como se utilizan cada uno de los recursos que maneja el microcontrolador, además que el usuario tenga un rápido entendimiento de los beneficios y aplicaciones que tiene el uso de esta herramienta tan importante en el desarrollo de diversos proyectos.

Otro punto importante para el desarrollo de este manual es entender el manejo combinado de los recursos como lo son: Memoria, Comunicación, Timers, PWM (Modulación por ancho de pulso), ADC (Convertidor Analógico-Digital), del mismo modo el uso del DAC (Convertidor Digital-Analógico) y LCD, todo esto da como resultado el desarrollo de proyectos más avanzados en la área de ingeniería electrónica.

Los desarrolladores de aplicaciones en sistemas embebidos ya no necesitan aprender el Lenguaje ensamblador de cierto microcontrolador y su arquitectura. CodeWarrior es un software que permite la fácil programación de microcontroladores mediante el lenguaje de programación C.

Capítulo I

INTRODUCCIÓN

1.1 Definición de microcontrolador

Un *microcontrolador* es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S, es decir, se trata de un computador completo en un sólo circuito integrado. Aunque sus prestaciones son limitadas, además de dicha integración, su característica principal es su alto nivel de integración.

Un microcontrolador tiene un generador de reloj integrado y una pequeña cantidad de memoria RAM y ROM/EPROM/EEPROM, significando que para hacerlo funcionar, todo lo que se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores analógico/digital, temporizadores, UARTs y buses de interfaz serie especializados. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados.

Los microcontroladores modernos por medio de un lenguaje de programación integrado como CodeWarrior, realiza la transformación de un lenguaje C o Basic a un lenguaje ensamblador.[1]

1.2 Definición de microprocesador

Es la Unidad Central de Procesamiento (CPU), parte fundamental de un microcontrolador. En el interior de este componente electrónico existen millones de transistores integrados.

[1] Angulo Usategui, José María y Angulo Martínez Ignacio, Microcontroladores PIC. Diseño práctico de aplicaciones, McGraw-Hill, pp 15-37.

El microprocesador está compuesto por: registros, la Unidad de control, la Unidad aritmético-lógica, y dependiendo del procesador, una unidad en coma flotante [2].

1.2.1 Funcionamiento

El microprocesador secciona en varias fases de ejecución (la realización de cada instrucción):

- Prelectura de la instrucción desde la memoria principal.
- Ordenamiento de los datos necesarios para la realización de la operación.
- Decodificación de la instrucción, es decir, determinar qué instrucción es y por tanto qué se debe hacer.
- Ejecución.
- Escritura de los resultados en la memoria principal o en los registros.

Cada una de estas fases se realiza en uno o varios ciclos de CPU, dependiendo de la estructura del procesador**,** y concretamente de su grado de supersegmentación. La duración de estos ciclos viene determinada por la frecuencia de reloj, y nunca podrá ser inferior al tiempo requerido para realizar la tarea individual (realizada en un solo ciclo) de mayor coste temporal. El microprocesador dispone de un oscilador de cuarzo capaz de generar pulsos a un ritmo constante, de modo que genera varios ciclos (o pulsos) en un segundo.

1.2.2 Características del Microcontrolador

- **Velocidad**: Se maneja frecuencias de GigaHertz (GHz.), o de MegaHertz (MHz.). Lo que supone miles de millones o millones, respectivamente, de ciclos por segundo.
- **Bus de direcciones.**

[2] MacKenzie, I. Scott., The 8051 microcontroller / I. Scott MacKenzie., 3rd ed., Upper Saddler River, N.J. : Prentice Hall, pp 37-45.

- **Bus de Datos:** El tamaño del bus de datos va a depender del volumen de procesamiento por segundo que requiera realizar el microprocesador. Este puede ir de 32 bits hasta 128 o 256 bits.
- **Zócalo:** Es una matriz de pequeños agujeros ubicados en una placa madre es la base donde encajan, sin dificultad, los pines de un microprocesador. Esta matriz permite la conexión entre el microprocesador y el resto del equipo[3].

1.3 Antecedentes del Microcontrolador MC9S12E.

La familia de microcontroladores MC9S12E pertenece a la familia de Freescale (antes Motorota), los primeros microcontroladores desarrollados por Motorola son los 68HC08, 68HC11 y el 68HC12.

1.3.1 Microcontrolador Zilog Z80.

El *Zilog Z80* es un microprocesador de 8 bits cuya arquitectura se encuentra a medio camino entre la organización de acumulador y de registros de propósito general. Si consideramos al Z80 como procesador de arquitectura de registros generales, se sitúa dentro del tipo de registro-memoria.

Fue lanzado al mercado en julio de 1976 por la compañía Zilog, y se popularizó en los años 80 a través de ordenadores como el Amstrad CPC, el Sinclair ZX-Spectrum o los ordenadores de sistema MSX. Es uno de los procesadores de más éxito del mercado, del cual se han producido infinidad de versiones clónicas, y sigue siendo usado de forma extensiva en la actualidad en multitud de dispositivos empotrados.

El Z80 estaba diseñado para ser compatible a nivel de código con el Intel 8080, de forma que la mayoría de los programas para el 8080 pudieran funcionar en él, especialmente el sistema operativo CP/M (programa de control para monitores).

[3] Angulo Usategui, José María y Angulo Martínez Ignacio, Microcontroladores PIC. Diseño práctico de aplicaciones, McGraw-Hill, pp 15-37.

A comienzos de los años 80 el Z80 o versiones clónicas del mismo fueron usadas en multitud de ordenadores domésticos, como la gama MSX, el Radio ShackTRS-80, el Sinclair ZX80, ZX81 y ZX Spectrum. También fue usado en el Osborne 1, el Kaypro y otra gran cantidad de ordenadores empresariales que dominaban el mercado por aquella época y que usaban el sistema operativo CP/M.

A mediados de los años 80 el Z80 fue usado en el Tatung Einstein y la familia de ordenadores domésticos y empresariales Amstrad CPC y Amstrad PCW. El Z80 también fue usado en los ordenadores Tiki 100, que se empleaban en los colegios de Europa.

1.3.2 Microcontrolador 68HC08.

El *68HC08* es una familia de microcontroladores de Freescale de 8 bits y arquitectura Von Neumann, con un solo bloque de memoria. La arquitectura de Von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control). Es conocida también simplemente por HC08.

Los HC08 son microcontroladores de propósito general, cada miembro de esta familia cuenta con diferentes periféricos internos, pero con una CPU común que permite migrar aplicaciones entre ellos, facilitando con ello el diseño.

Entre los periféricos internos con estos microcontroladores están los conversores analógicos-digital, módulo de control de tiempos y sistemas de comunicación como SPI, USB o SSCI entre otros.

Freescale creo una mejora a esta familia de microcontroladores la HCS08 que ofrece mejoras en algunas instrucciones y agrega nuevas, además en esta

mejora los microcontroladores pueden ser programados por puerto dedicado, que mejora su versatilidad y amplia su gama de aplicaciones.[4]

1.3.3 Microcontrolador 68HC11.

La familia *Motorola 68HC11* (abreviado HC11 o 6811) es una familia de microcontroladores de Motorola, derivada del microprocesador Motorola 6800. Los microcontroladores 68HC11 son más potentes y costosos que los de las familias anteriores y se utilizan en múltiples dispositivos empotrados.

Siguen la arquitectura Von-Newman, en la que la memoria del programa, de datos y de entrada/salida se direcciona en un único mapa de memoria.

Internamente, el conjunto de instrucciones de la familia 68HC11 es compatible con la del 6801 y el 6809, con el añadido de un registro Y (que puede ser empleado por las mismas instrucciones que el registro X). La familia 68HC11 emplea instrucciones de longitud variable y se considera que emplea una arquitectura CISC (Computadora de Sistema de Instrucción Compleja).

Tienen dos acumuladores de ocho bits (A y B), cuenta con un acumulador virtual D, que no es más que la unión de A y B (16 bits), dos registros índice de 16 bits (X e Y), un registro de banderas, un puntero de pila y un contador de programa.

Los 68HC11 tienen cinco puertos externos (A, B, C, D y E), cada uno de ocho bits excepto el D, que es generalmente de seis bits.

El puerto A se emplea en captura de eventos, salida comparada, acumulador de pulsos y otras funciones de reloj; el puerto D para E/S serie y el puerto E como conversor analógico-digital.

La familia 68HC11 puede funcionar tanto con memoria interna o externa. En caso de emplear memoria externa, los puertos B y C funcionan como bus de datos y direcciones respectivamente.

[4] http://www.motorola.com/mx/products

El 68HC11 consta de un convertidor analógico/digital, el cuál recibe una tensión comprendida entre 0 y 5 voltios; y devuelve, por medio de registros internos, valores de 8 bits (del 0 al 255), que son proporcionales a la entrada. Estos valores pueden ser utilizados en el programa para hacer cálculos, por ejemplo.

Si el modelo de HC11 es de 52 pines, el puerto tiene ocho entradas al convertidor, mientras que si es el modelo de 48, este puerto sólo tiene cuatro entradas.[5]

1.3.4 Microcontrolador 68HC12.

Este es el primer y realmente completo sistema de desarrollo para microcontroladores *68HC12*, accesible e integrado en un único paquete. El sistema está comprendido por una tarjeta de desarrollo, y software compatible con los sistemas operativos WIN95/98/ME/NT/ 2000/XP. El software incluye editor, assemble y Compilador 'C'.

- Motorola 68HC12 CPU
- 32K RAM
- 32K EPROM
- 8 Canales A/D
- 1 Canal D/A
- 2 x 16LCD Display
- 4 entradas digitales
- 4 salidas digitales
- Puerto serie
- Win 95/98/ME/NT/2000/XP Compatible Manager Software

Software Drivers para Keypad, LCD, A/D, D/A, Switch sensor y todo el hardware incluido en la tarjeta de desarrollo.

El monitor de EPROM es útil para dos funciones principales: comunicarse con un CPU por medio de interface serie, operando también como una EPROM de bufeo, llevando al microcontrolador a un estado inicial predeterminado después que ha terminado de la secuencia de alimentación.

[5] http://www.motorola.com/mx/products

Los 32k bytes de RAM de la tarjeta principal son utilizados para guardar el programa de usuario y sus variables. También posee una inteface serie para conectarse a una computadora personal. Esta interface permite al usuario permite bajar comandos y programas a la memoria RAM.

El conector de expansión es muy importante, por que le permitirá conectar la tarjeta a hardware experto. Todas las señales del microcontrolador estarán disponibles.

Lo más importante de un sistema de desarrollo es el software. El micro manager software es compatible con los sistemas operativos Windows 95/98/ME/NT/2000/XP y esta diseñado para ser utilizado fácilmente. Los menús pull-down, dialog box y el help son características que lo convierten en un software fácil y rápido de usar. El usuario puede crear y editar código fuente, ensamblarlo, compilarlo y bajarlo a la tarjeta de desarrollo.[6]

1.4 Aportaciones de la Tesis

En este trabajo se realizaron diversas prácticas para el manejo del sistema de desarrollo SIS9S12E, aportando una metodología para el uso del sistema de programación CodeWarrior y una metodología para la programación de recursos mediante Beans. Estas metodologías nos permitirán un rápido manejo y entendimiento de todos los recursos que maneja el microcontrolador.

1.5 Contenido de la Tesis

En el capítulo dos se presentan las características y especificaciones del microcontrolador SIS9S12E, las cuales son: descripción del sistema de reloj, modo de operación, convertidor analógico digital, bloque de descripción del PWM, descripción de la interfase de comunicación serie y características eléctricas. Estas sirven como guía para la configuración y programación de todos los dispositivo que se manejan, y también nos presenta los esquemáticos de microprocesador (CPU), fuentes (alimentación y de interfaces), y conectores.

[6] http://www.motorola.com/mx/products

En el tercer capítulo se hace una descripción de los programas que se utilizaron para el desarrollo del manual de prácticas (CodeWarrior), además del lenguaje de programación que se utilizó basado en Processor Expert.

En el capítulo cuatro se presentan las prácticas desarrolladas dentro de este trabajo, dándole uso a la mayoría de los recursos que contienen el sistema de desarrollo, permitiendo realizar prácticas como: Manejo de puertos, uso del los convertidores (analógico/digital y digital/analógico), uso del PWM para el control de servomotores, control del microcontrolador por medio de un software de monitoreo, generación de funciones (seno, coseno, triangular, dientes de sierra, exponencial, etc.), así como el manejo del puerto de salida LCD.

En las conclusiones del trabajo, se dan a conocer las aportaciones en diversas áreas de la carrera de Ingeniería Electrónica y Telecomunicaciones para trabajos futuros.

Por último se hace referencia a la bibliografía que se manejó para la elaboración de este proyecto, además del apéndice de las herramientas utilizadas.

Capítulo II

ESPECIFICACIONES Y CARACTERÍSTICAS DEL SISTEMA DE DESARROLLO SIS9S12E.

2.1 Características generales

La Figura 2.1 muestra el sistema de desarrollo SIS9S12E que se basa en la familia de microcontroladores MC9s12E64CPV de FREESCALE, es una familia de bajo costo de propósito general, la cual tienen periféricos estándar, CPU: HCS12, Memoria: 64K de flash, 4K de RAM, Comunicación:3 puertos SCI, 1 SPI, 1 Interface bus IIC, Timers: 3 (TIM) de cuatro canales cada uno (Entrada de captura, salida de comparación, acumulador de pulsos) PWM:6 canales de 15 bits con módulo y de protección por falla (PMF), ADC:16 canales del convertidor analógico/digital de 10 bits DAC: 2 convertidores de digital/analógico de 8 bits, 16 I/O con la característica de despertar (Wake-up) del modo de operación STOP o WAIT PLL: permite variar la frecuencia de operación interna hasta 25 Mhz. Regulador interno 2.5V que permite su operación con fuente externa de 3.3 a 5.5V.[13]

Figura 2.1 Sistema de desarrollo SIS9S12E.

[13] http://www.racom.com.mx/desarrollo.php.pdf

2.2 Diagrama a bloques de la familia MC9S12E.

Los recursos de la tarjeta de desarrollo pueden ser observados en la Figura 2.2 que muestra el diagrama a bloques del microcontrolador MC9S12E.[14]

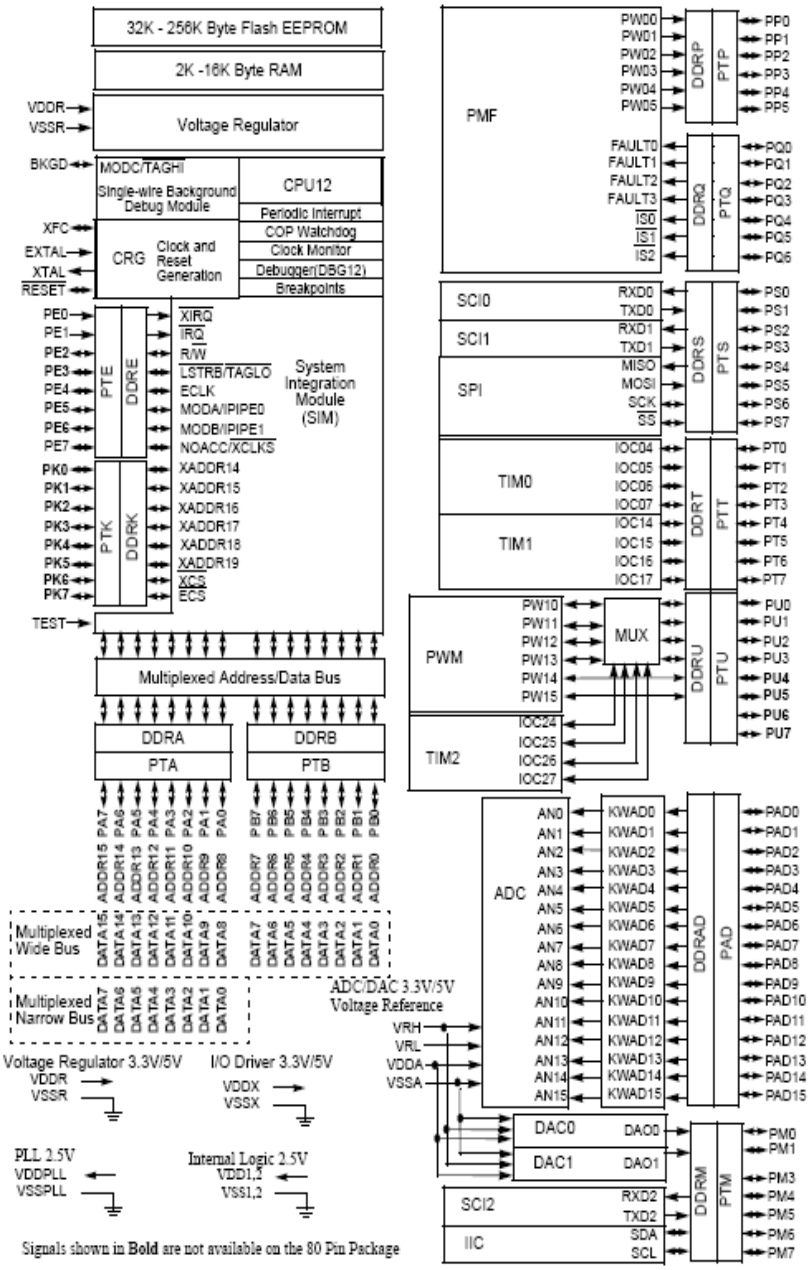


Figura 2.2 Diagrama a Bloques de MC9S12E.

[14] http://www.metrowerks.com

2.3 Esquema del sistema de desarrollo SIS9S12E.

La Figura 2.3 muestra la asignación de pines de salida utilizados por el microcontrolador (CPU) HC12. Estos pines están conectados a los diversos recursos de hardware que utiliza el sistema de desarrollo SIS9S12E como: los botones, el potenciómetro, display, leds, etc.[15]

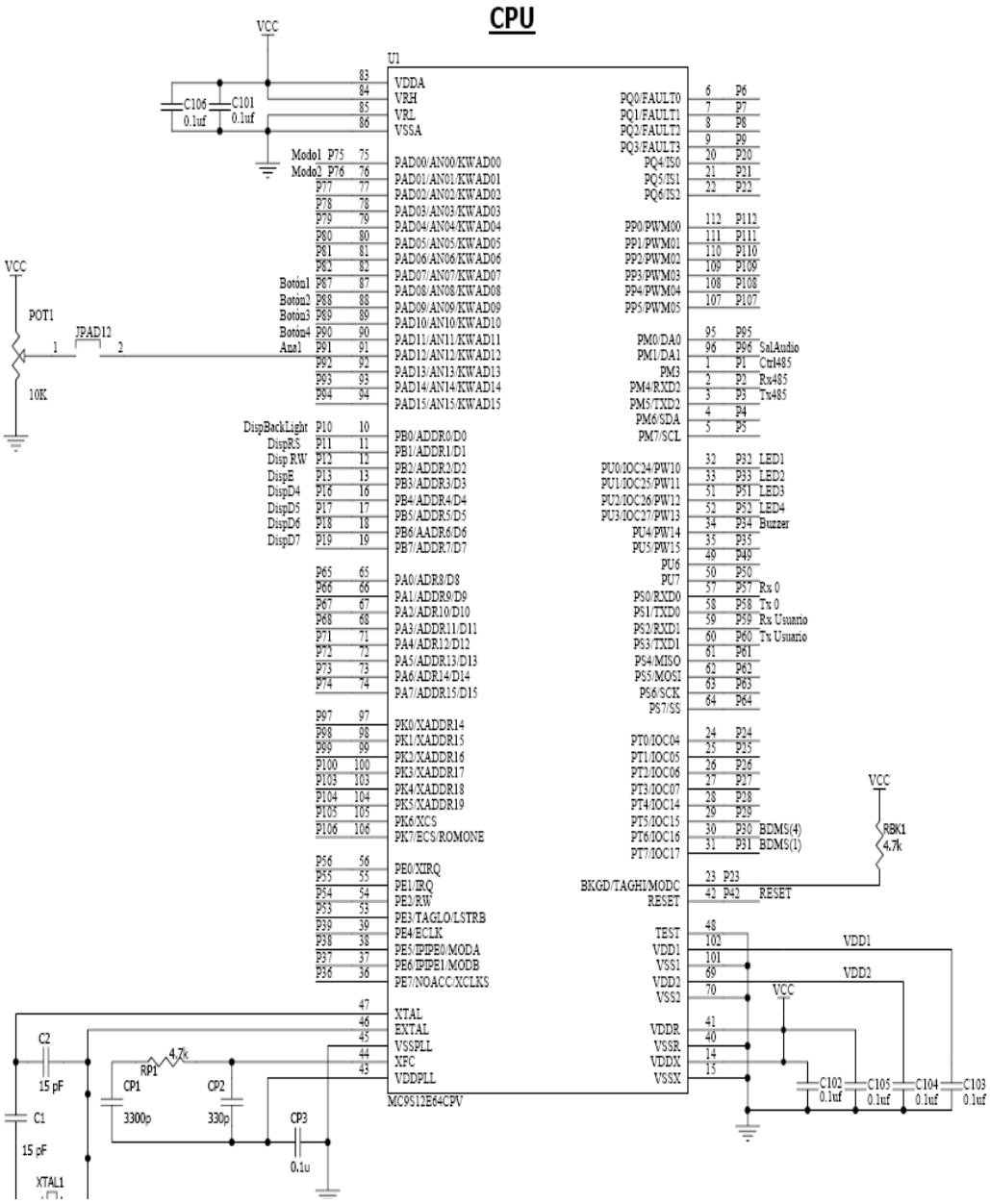


Figura 2.3 Esquema del CPU utilizado en el sistema SIS9S12E.

15

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC9S12E64&nodeId=016246844976630029.pdf

2.4 Asignación de pines de salida de microcontrolador MC9S12E.

El sistema de desarrollo SIS9S12E utiliza el microcontrolador MC9S12E como parte elemental de su funcionamiento. Este es el encargado de la administración y manejo de todos los recursos del sistema de desarrollo, en la Figura 2.4 muestra el esquema de pines de salida del microcontrolador MC9S12E.[16]

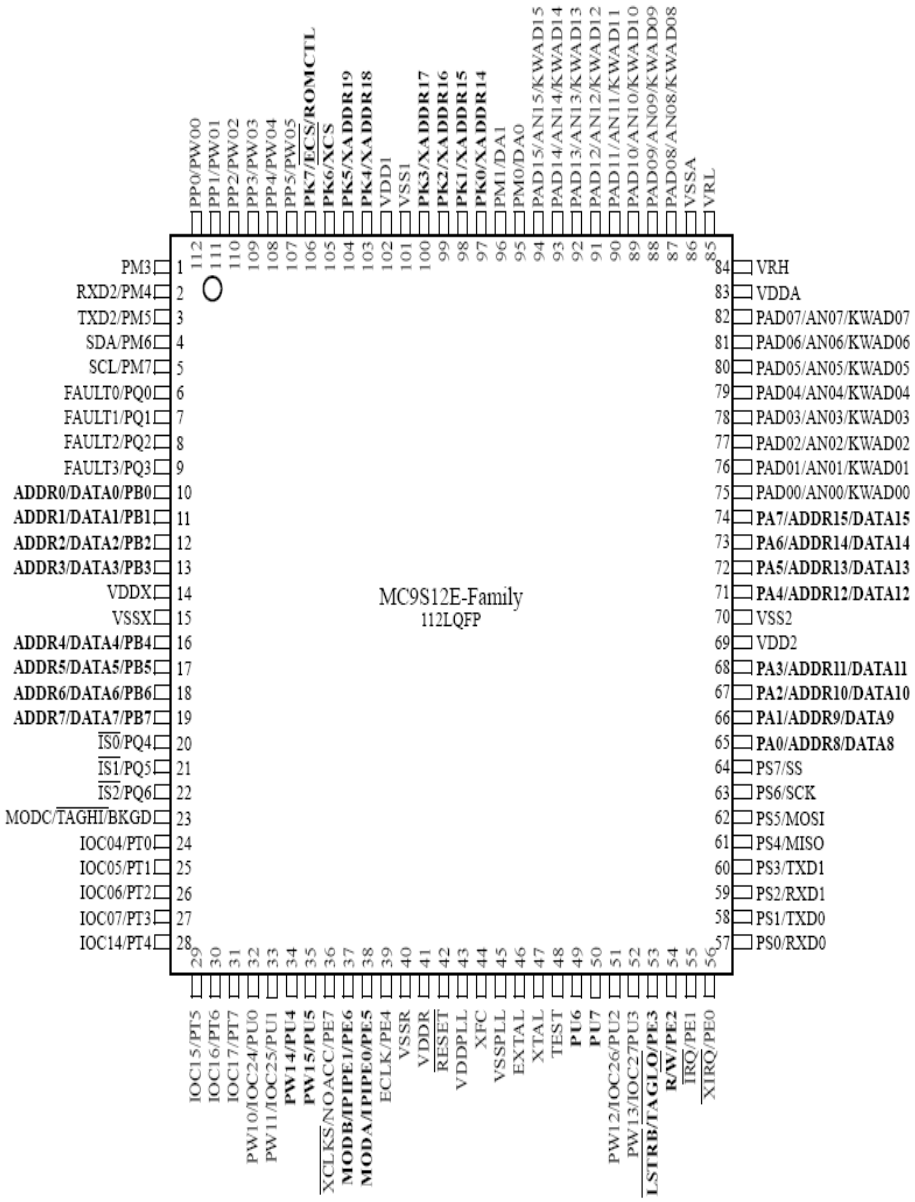


Figura 2.4 Asignaciones de pines en 112 LQFP.

[16] http://www.metrowerks.com

2.4.1 Características de hardware de la tarjeta de desarrollo

- Programación por puerto serial RS232 DB9.
- 4 Leds para el usuario.
- Botones (tipo push bottom).
-1 potenciómetro para pruebas con el convertidor AD.
- Conector para Display LCD estandar.
- Voltaje de alimentación de 7 a 12 Vcd con jack o terminales.
- No requiere fuente especial para programación.
- Led de encendido.
- Puerto RS232 para monitor (SCI0).
- Salida en pines para conector externo RS232 (Puerto SCI1).
- Opción para colocar puerto RS485. (SCI2).
- Salida a pines de DAC0.
- Dimensiones tarjeta: 13x 10 cm.[17]

La Figura 2.5 muestra el esquemático del sistema SIS9S12E.

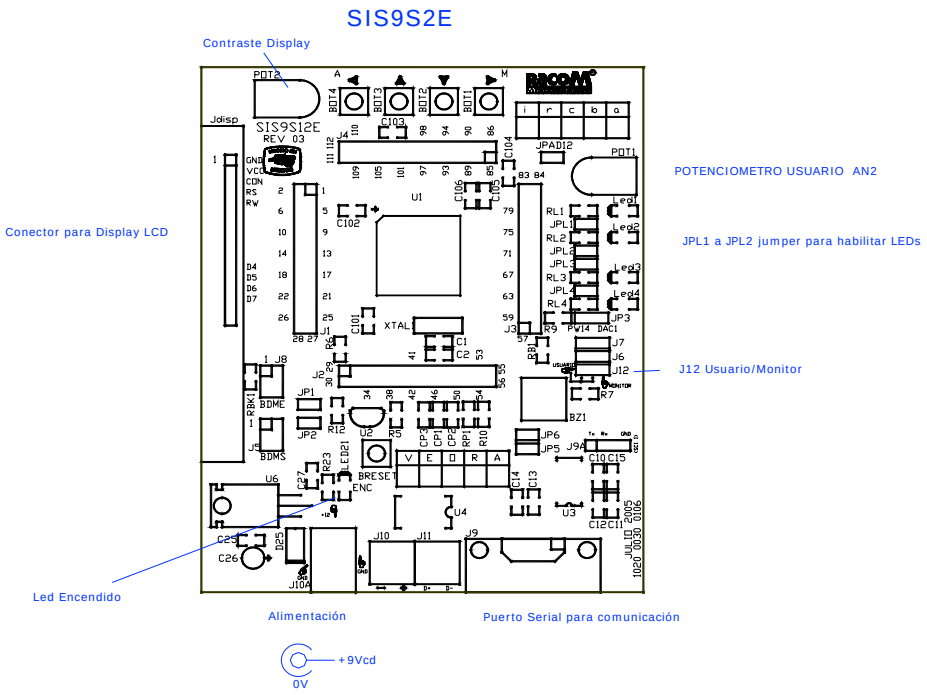


Figura 2.5 Esquemático del Sistema SIS9S12E.

[17] http://www.freescale.com/webapp/sps/site/homepage.jsp.pdf

2.4.2 Características generales de programación.

- El programa monitor ocupa 2K de flash.
- Se programa con Code Warrior CW12V3.1
- En ensamblador, en C o ambos.
- Permite licencia especial hasta 32K en C
- Se entrega con plantillas para el sistema de desarrollo y programa Demo.

2.4.3 Requerimientos de hardware

- 200MHz Pentium II o AMD-K6 o superior.
- 128 MB de RAM, CD-ROM Drive.
- Puerto Serial 9 pines.
- Sistema Operativo: Windows 98/2000/XP/NT.
- Espacio Disco de 232MB compacto Completo: 344MB.

2.4.4 Requerimientos de Software

CodeWarrior Development Studio for FreeScale 68HC(s)2 Microcontrollers.

2.5 Identificación de Nomenclatura

La Figura 2.6 muestra la nomenclatura utilizada en los microcontroladores de la familia MCS9S12E, permitiendo un reconocimiento más rápido del microcontrolador. Esto nos ayudará a buscar las especificaciones de los microcontroladores.

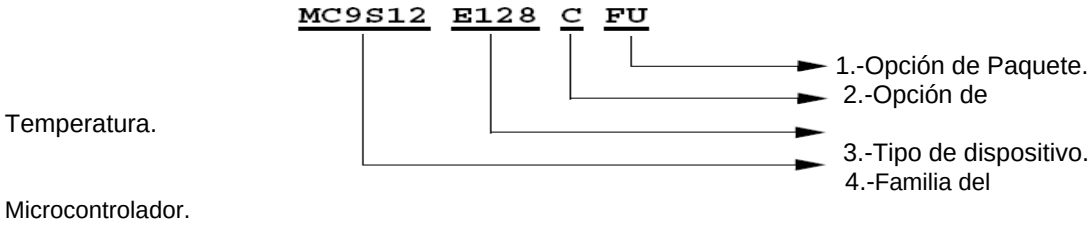


Figura 2.6 Orden de la parte de número de codificación.

1.- Opción de paquete.

```
FC = 64QFN
FU = 80QFP
PV = 112LQFP
```

2.- Opción de temperatura.

```
C = -40°C to  85°C
V = -40°C to 105°C
M = -40°C to 125°C
```

3.- Tipo de dispositivo.

4.- Familia del controlador.

2.6 Esquema de recursos del sistema de desarrollo SIS9S12E[18]

2.6.1 Esquema de interfase RS232. J9 (DB9 RS232) Puerto SCI0 y J9A (RS232 Opcional) Puerto SCI1.

La Figura 2.7 muestra el esquema de comunicación RS232.

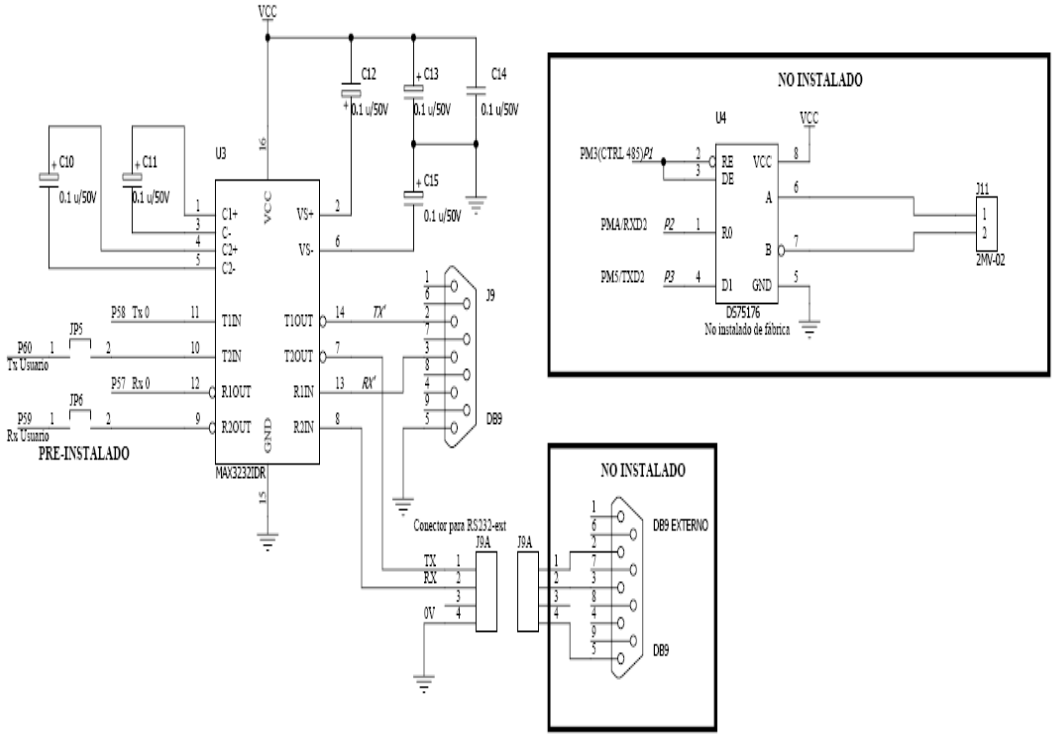


Figura 2.7 Esquema de interfase RS232.

[18] http://www.freescale.com/webapp/sps/site/homepage.jsp.pdf

2.6.2 Esquema de la fuente de alimentación.

La Figura 2.7. muestra el esquema de la fuente de alimentación del sistema MC9S12E.

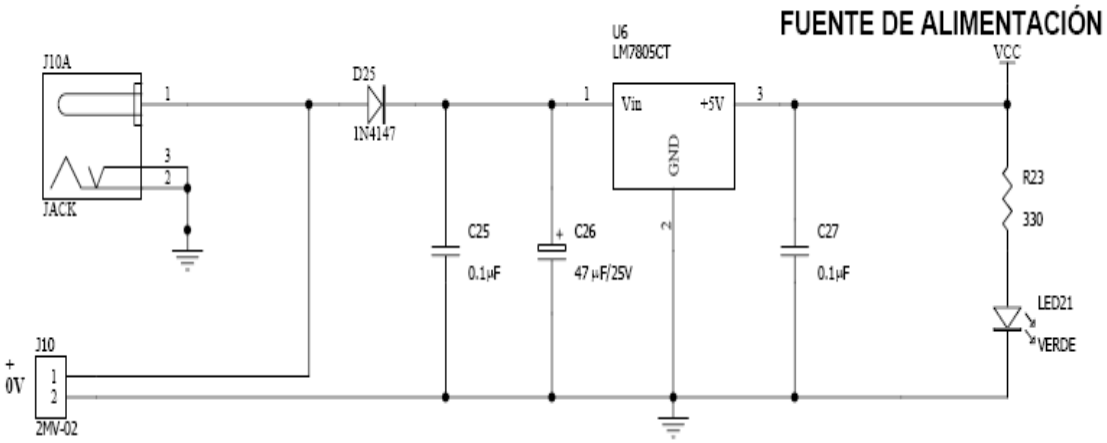


Figura 2.7.1 Esquema de fuente de alimentación.

2.6.3 Esquema del sistema de modo Usuario/Monitor.

Este modo permitirá cargar el microcontrolador y utilizar el programa realizado en Code Warrior. Para el uso monitor/Usuario (Figura 2.8) se debe de tomar en cuenta las siguientes características:

- Se deben poner la frecuencia del bus a 24 MHz.
- Con cristal de 8 MHz.
- No utilizar el COP, ni atender el COP, ya que lo utiliza el serial-monitor

El puerto SCI0 se usa de forma exclusiva para el monitor:

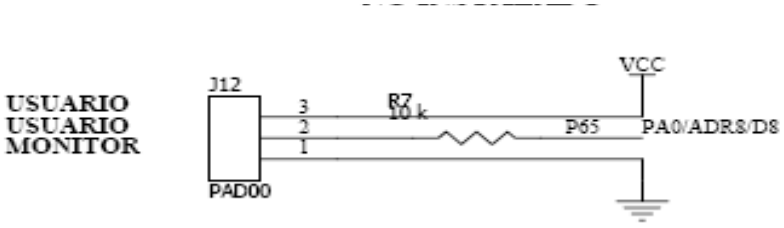


Figura 2.8 Sistema de modo Usuario/Monitor.

2.6.4 Esquema de botón Reset.

Permite resetear el microcontrolador, así como borrar el programa que anteriormente había sido cargado (Figura 2.9).

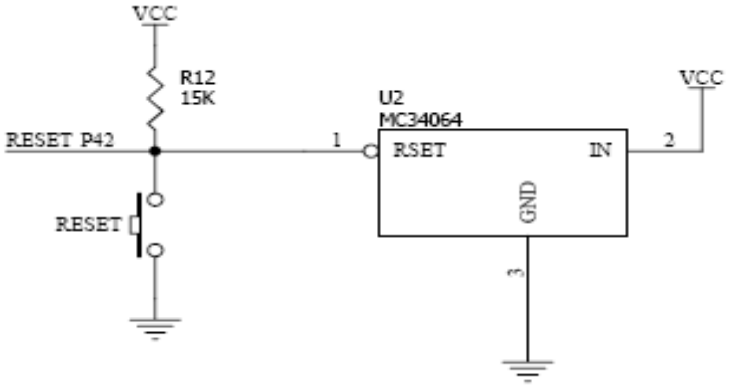


Figura 2.9 Esquema de Botón Reset.

2.6.5 Esquema de pines de Conectores.

La figura 2.10 muestra los conectores con el que trabaja el sistema que se ha desarrollado (SIS9S12E). Manejo un total de 112 conectores (pines).

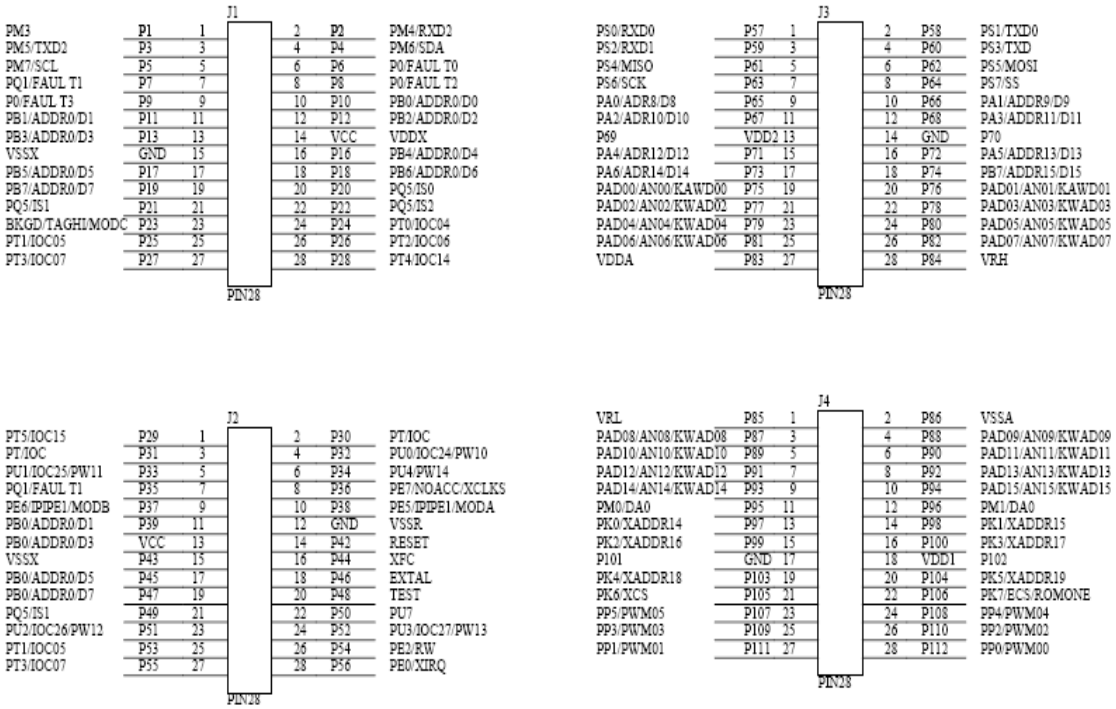


Figura 2.10 Conectores del Sistema SIS9S12E (MC9S12E).

2.6.6 Conectores de los LEDs.

La Tabla 2.1 y la Figura 2.11 muestran el puerto U tiene cuatro pines configurados como salida (Led):

Nombre Led	Jumper para habilitar los Leds	Pin Micro	Puerto
Led1	**JPL1**	32	PU0
Led2	JPL2	33	PU1
Led3	JPL3	51	PU2
Led4	JPL4	52	PU3

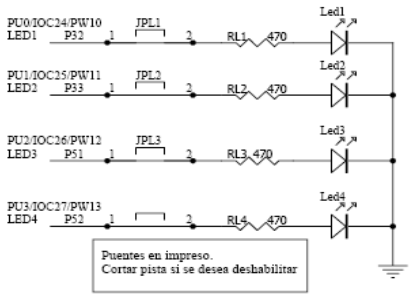


Tabla 2.1 Características del puerto U. del Led.

Figura 2.11 Esquema

2.6.7 Esquemas de conexión de los botones (puerto de entrada) y display

La Tabla 2.2 muestra la lista de los botones que utiliza el sistema de desarrollo SIS9S12E conjuntamente con sus pines de salida y el puerto que utiliza.

Nombre	Pin Micro	Puerto
Bot1	87	PAD08
Bot2	88	PAD09
Bot3	89	PAD10
Bot4	90	PAD11

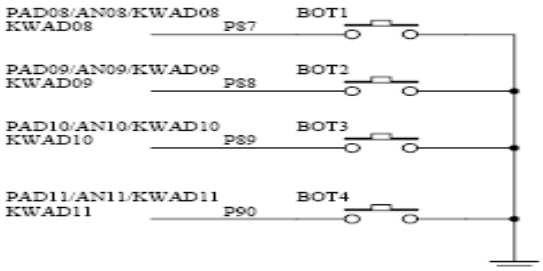


Tabla 2.2 Características del puerto PAD.

Figura 2.12 Esquema de Botón.

La Figura 2.12 muestra el esquema de conexiones que utilizan los botones.

2.6.8 Esquema de conexión de display.

En la Tabla 2. y la Figura 2.13 se pueden observar los pines de conexión que son necesarios para conectar el LCD.

Nombre	JDISP	Señal		Pin Micro
POT1	1	GND		
	2	VCC		
	3	Contraste/POT2		
	4	RS	11	PB1
	5	RW	12	PB2
	6	E	13	PB3
	7	-		
	8	-		
	9	-		
	10	-		
	11	D4	16	PB4
	12	D5	17	PB5
	13	D6	18	PB6
	14	D7	19	PB7
	15	-		
	16	-		

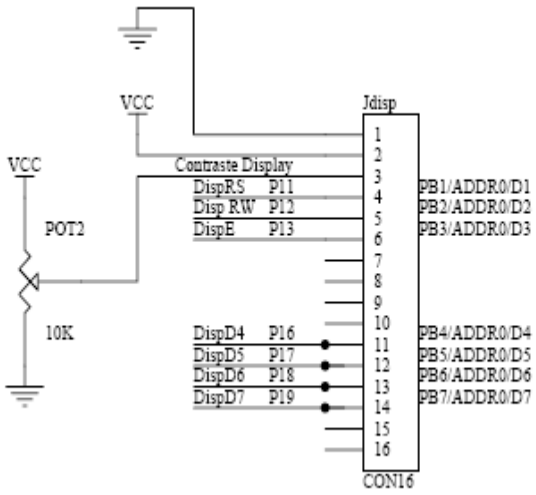


Tabla 2.3 Características de display. **Figura 2.13 Esquema de Display.**

2.6.9 Lista de codificación de microcontrolador basado en paquete y temperatura.

La Tabla 2.4 muestra una lista de microcontroladores de la familia MCS9S12 clasificados en base a la temperatura que trabaja, el paquete de pines y su descripción.

Número	Temperatura	Paquete	Descripción
MC9S12E256CFU	-40°C, 85°C	80QFP	MC9S12E256
MC9S12E256CPV	-40°C, 85°C	112LQFP	MC9S12E256
MC9S12E256MFU	-40°C, 125°C	80QFP	MC9S12E256
MC9S12E256MPV	-40°C, 125°C	112LQFP	MC9S12E256
MC9S12E128CFU	-40°C, 85°C	80QFP	MC9S12E128
MC9S12E128CPV	-40°C, 85°C	112LQFP	MC9S12E128
MC9S12E128MFU	-40°C, 125°C	80QFP	MC9S12E128
MC9S12E128MPV	-40°C, 125°C	112LQFP	MC9S12E128
MC9S12E64CFU	-40°C, 85°C	80QFP	MC9S12E64
MC9S12E64CPV	-40°C, 85°C	112LQFP	MC9S12E64
MC9S12E64MFU	-40°C, 125°C	80QFP	MC9S12E64
MC9S12E64MPV	-40°C, 125°C	112LQFP	MC9S12E64
MC9S12E32CFU	-40°C, 85°C	80QFP	MC9S12E32
MC9S12E32MFU	-40°C, 125°C	80QFP	MC9S12E32

Tabla 2.4 Lista de codificación del microcontrolador.

2.7 Rangos y características del uso de HCS12.[19]

2.7.1 16 bits HCS12 – (HCS12 CPU).

- Compatible con la instrucción que pone M68HC11.
- Interrumpe la colocación y el modelo del programador idéntico a M68HC11.
- Coleta de instrucción.
- Realzada la dirección incluida en un índice.
- Control de modulo de mapeo (MMC).
- Control de interrupción (INT).
- Modulo de eliminación de errores (BDM).
- Depurador (DBG12) incluyendo límites de facturación y cambio de flujo remoto del buffer.
- El Interfaz multiplexado externo del bus (MEBI).

2.7.2 Interrupción de las entradas por Wake up

- Hasta 16 bits de puerto disponibles para activar la función de interrupción con la filtración digital.

2.7.3 Opciones de memoria.

- 32 kilobyte, 64 kilobyte, 128 kilobyte o 256 kilobyte de memoria Flash EEPROM.
- 2 kilobyte, 4 kilobyte, 8 kilobyte o 16 kilobyte de memoria RAM.

2.7.4 Dos Convertidores digitales a análogo de 1 canal (DAC).

- La resolución de 8 bits.

2.7.5 Convertidor analógico a digital (ADC).

- Módulo de 16 canales con la resolución de 10 bits.

---

19 http://www.freescale.com/webapp/sps/site/homepage.jsp.pdf

2.7.6 Tres temporizadores de 4 canales (TIM).

- Captura de entrada programable o la salida comparan canales
- Modo PWM simple.
- Puesta a cero del Módulo del contador.
- Conteo de eventos Externos.
- Acumulación de Tiempo Gated.

2.7.7 6 canales de PWM (PWM)

- Período y ciclo de trabajo programables.
- 6 canales de 8 bits o 3 canales de 16 bits.
- Control separado de cada anchura de pulso y ciclo de trabajo.
- Salidas Alineadas por centro o izquierdas.
- Reloj Programable selecciona la lógica con una amplio rango de frecuencias.
- Entrada de parada rápida de emergencia.
- 6 canales de modulador de ancho de pulso con protección determinada.
- Tres contadores independientes de 15 bits con modo sincrónico.
- Operación de canal complementario.
- Borde y centro alinearon señales de PWM
- Inserción de tiempo muerto programable.
- Integra recarga tarifas de 1 a 16.
- Cuatro protecciones a pines de entrada.
- Tres pines de entrada en sentido corriente.

2.7.8 Interfaces seriales:

- Tres interfaces de comunicación asincrónicos serial(SCI).
- Interfaz sincrónico serial periférico (SPI).
- Inter-IC bús (IIC).

2.7.9 Reloj y generador Reset (poner a cero) (CRG).

- Windowed PILLAN watchdog (perro guardián).
- Interrupción en tiempo real.
- Monitor de Reloj.
- Multiplicador de frecuencia de reloj de lazo cerrado por fase.
- Modo propio de cronometro en la ausencia de reloj externo.
- Baja energía de 0.5 a 16 Mhz para el oscilador cristal referente al reloj.

2.7.10 Frecuencia de operaciones

- 50MHz equivalente con 25MHz Velocidad del Bús.

2.7.11 Regulador interno de 2.5 volts:

- Rango de voltaje de Entrada de 3.135 V a 5.5 V
- Capacidad de baja energía.
- Incluye circuito Reset de bajo voltaje.
- Incluye circuito de interrupción de bajo voltaje.

2.7.12 112 pines LQFP O 80 pines QFP.

- Hasta 90 líneas de entrada/salida con 5V la entrada y la capacidad de manejo (de unidad de disco) (112 paquete de pines)
- Hasta dos dedican 5V de entrada en algunas líneas (IRQ y XIRQ).
- 16 entradas de convertidor 3.3V/5V analógico/digital[7].

2.8 Modo de operación. [20]

Modos de usuario (los modos extendidos están sólo disponibles en la versión de paquetes de 112 pines).

2.8.1 Modo normal.

- Modo normal de chip sencillo y Modo expandido estrecho.
- Modo normal amplio expandido y Modo de emulación expandido amplio.

---

[20] http://www.racom.com.mx/desarrollo.php.pdf

2.8.2 Modo especial de operación:

- Modo especial de solo chip con modo activo de ajuste de fondo.
- Modo especial de prueba (solo lo uso Motorola).
- Modo especial periférico.

2.8.3 Modo de bajo energía:

- Modo stop (alto).
- Modo pseudos stop (pseudos alto).
- Modo espera.

2.9 Descripción de los pines del sistema MC9S12[21]

2.9.1 EXTAL y XTAL.- Pines del oscilador.

EXTAL y XTAL son los pines del reloj externo y el controlador (driver) de cristal. Sobre resetear todos los relojes de los dispositivos son derivados de la frecuencia de entrada EXTAL. XTAL es la salida del cristal.

2.9.2 RESET.- Pin externo de reseteo.

El reset es un control de señal activo bajo bidireccional que actúa como una entrada para inicializar el MCU a un conocido estado star-up. Esto también actúa como un open-drain de salida que sirve para indicar que un error interno ha sido detectado en monitor del reloj o en el circuito de perro guardián COP.

2.9.3 TEST- Pin test (prueba).

El pin test es reservado para la prueba y debe estar conectado a VSS en todas las formas de uso.

21

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC9S12E64&nodeId=016246844976630029.pdf

2.9.4 XFC – Pin PLL Loop Filter

El pin dedicado suele crear el filtro de lazo PLL.

2.9.5 PA.- Puerto A de entrada-salida.

El puerto A son pines de entrada o salida de propósito general. Son usados como modo de operación MCU, se escogen durante el reseteo, estos pines son usados para la multiplexación de la dirección externa y el bus de datos [7].

2.9.6 PB.- Puerto B de entrada-salida.

El puerto B son pines de entrada o salida de propósito general. Son usados como modo de operación MCU así como para la multiplexación de la dirección externa y el bus de datos.

2.9.7 PE7 / NOACC / XCLKS — Port E I/O Pin 7.

PE7 es un pin de entrada o salida de propósito general. Usado durante el modo operación MCU, la señal NOACC, cuando se habilita, se utiliza principalmente para indicar que la corriente del ciclo del bus es un ciclo libre. Esta señal indicará el momento que el CPU no éste usando el bus. El XCLKS es una señal de entrada que se utiliza sí controla un cristal en combinación con el oscilador Colpitts.

2.9.8 Esquema de conexión del oscilador Colpitts (PE7=1, PE7=0) [12].

La Figura 2.15 y 2.16 muestra el esquema de conexión del oscilador Colpitts, usado por el microcontrolador.

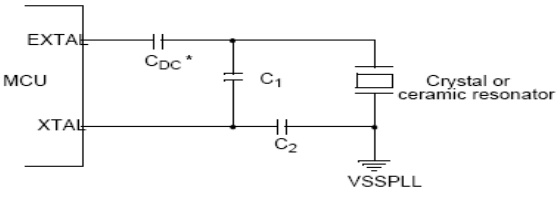


Figura 2.15 Oscilador Colpitts PE7=1.

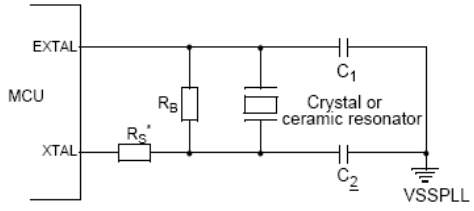


Figura 2.16 Oscilador Colpitts PE7=0.

2.9.9 PE6 / MODB / IPIPE1 — Port E I/O Pin 6.

PE6 es un pin entrada o salida de propósito general. Es usado como modo de operación MCU se escoge durante el reseteo. El estado de este pin esta apegado el bit MODB ala orilla del RESET, comparte la serie de instrucciones que resetea la señal IPIE1.

2.9.10 PE5 / MODA / IPIPE0 — Port E I/O Pin 5.

PE5 es un pin entrada o salida de propósito general. Es usado como modo de operación MCU este pin es escogido durante el reseteo. El estado de este pin esta apegado el bit MODB ala orilla del RESET, se comparte con la serie de instrucciones que resetea la señal IPIE1.

2.9.11 PE4- Clock.

PE4 es un pin de fines generales de la entrada y salida, se utiliza en modo normal del chip PE4, configurando con un activo pull-up éste puede ser apagado por el clearing PUPEE en el registro de PUCR. En todos los modos excepto en modo normal del chip, el pin PE4 se configura inicialmente como la conexión de salida del bús interno clock(ECLK). ECLK se utiliza como referencia de la sincronización y demultiplexa la dirección y los datos en modos ampliados.

La frecuencia de ECLK es igual a la mitad de la frecuencia del cristal fuera de reajuste. La función de salida de ECLK depende de los ajustes del chip de NECLK en el registro PERA, del chip de IVIS en el modo registro y del pedazo de ESTR en el registro EBICTL. Todos los relojes, incluyendo el ECLK, se paran cuando el MCU está en modo PARADA (Stop). Es posible configurar el MCU para interconectar, para retardar memoria externa.

2.9.12 Port E I/O Pin 3 / Low-Byte Strobe (LSTRB).

PE3 se puede utilizar como uso general en todos los modos y es una entrada con un activo pull-up fuera de reajuste. El pull-up puede ser apagado por clearing PUPEE en el registro PUCR. El PE3 logra también configurarse como estroboscopio (LSTRB).

LSTRB es autorizado fijando la parte de LSTRE en el registro PERA, ampliando los modos estrechos de par en par y de la emulación y cuando se permite el marcar con etiqueta del BDM, la función de LSTRB se multiplexan con la función de TAGLO. Cuando se permite una lógica cero en el pin de TAGLO en el borde que cae de ECLK marcará el octeto con etiqueta bajo de una palabra de la instrucción que es leída en la coleta de la instrucción. PE3 no está disponible en la versión del paquete de 80 pines.

2.9.13 Port E I/O Pin 2 / Read/Write.

PE2 se puede utilizar como I/O de uso general en todos los modos y es configurado como una entrada con un activo pull-up fuera de reajuste. Si se requiere la función de lectura/grabación debe ser permitida fijando la parte de RDWE en el registro PERA. El pin PE2 no está disponible en la versión del paquete de 80 pines. En el caso del microcontrolador MC9S12E si estará disponible.

Capítulo III

INTRODUCCIÓN A LA PROGRAMACIÓN DEL SISTEMA DE DESARROLLO SIS9S12E CON CODEWARRIOR

3.1 Introducción.

En este capítulo se muestran las características del software utilizado en el desarrollo del manual de prácticas. Por ejemplo el uso de un sistema de desarrollo avanzado como CodeWarrior, el cual nos permitirá un fácil manejo de los recursos del microcontrolador.

3.2 CODE WARRIOR. [30]

3.2.1 Introducción a CodeWarrior.

Code Warrior es un sistema de desarrollo integrado (IDE) muy potente que facilita el diseño de sistemas dedicados utilizando lenguaje de nivel medio (ANSI C). El ambiente de trabajo de Code Warrior está diseñado para trabajar con proyectos teniendo la facilidad de utilizar módulos depurados anteriormente (librerías) por lo que se agiliza la construcción de un sistema. Ofrece las siguientes ventajas:

- Uso de lenguaje de nivel medio (ANSI C).
- Portabilidad de código entre diferentes fabricantes (teniendo el compilador especifico).
- De mantenimiento sencillo.
- Depuración abarcando la mayoría de los casos.

[30] Code Warrior IDE Quichstart manual.

CodeWarrior tiene la capacidad de manipular proyectos, edición de archivos, compilación optimizada, simulación y ejecución del código generado. A cada proyecto se le asigna una carpeta que contiene la mayoría de los archivos fuente que componen el sistema. Cada archivo es recomendable que contenga las funciones, declaraciones de variables, etc., de un módulo completo para que este pueda ser reutilizado en los proyectos que contengan un módulo similar. Por ejemplo: manejo de un LCD, temporizadores, DAC, etc. Este modelo de programación nos permite generar librerías en las cuales el tiempo de depuración solo es la primer vez y, cuando es requerido (el módulo) por otro sistema equivale a un # <include>.

3.2.2 Características de CodeWarrior.

El software de CodeWarrior provee un intuitivo interfaz de uso gráfico (GUI) estas características principales:

- Administrador del Proyecto.
- Editor de Código Fuente.
- Buscardor o navegador.

3.2.2.1 Administrador de proyecto.

Se puede utilizar al administrador de proyecto de CodeWarrior IDE para recolectar archivos y opciones del programa en un solo archivo de proyecto. También se puede utilizar al administrador de proyecto para especificar los compiladores y los linkers que IDE debe utilizar para crear su aplicación.

3.2.2.2 Editor de código fuente.

El editor de código fuente de CodeWarrior IDE sirve para corregir código y el texto fuente. El editor del código fuente permite:

- Edita, busca, y remplaza texto en un archivo o varios archivos.
- Parte una ventana del editor en diversas partes.

- Acciona el interruptor entre un archivo fuente y su archivo de interfaz.
- Abre un archivo de interfaz referido por un archivo fuente.
- Fija los marcadores en las localizaciones arbitrarias en un archivo de texto.
- Salta a cualquier rutina en cualquier archivo inmediatamente.
- Modifica la exhibición del código fuente para requerimientos particulares.

3.2.2.3 Buscador o navegador.

Permite utilizar los comandos en cualquier opinión y control del código fuente en la ventana del buscador para el acceso rápido, intuitivo a las variables, las rutinas, la enumeración, y las definiciones de clases, y de otros elementos del código fuente.

3.2.3 Funciones de CodeWarrior.

El software tiene estas funciones principales:

- Compilador altamente óptimo de C/C++.
- Macro ensamblador de gran alcance.
- SmartLinker, que liga solamente los objetos que realmente tienen referencia.
- Decodificador para descifrar el objeto y archivos completos.
- Libmaker para generar bibliotecas.
- Depuración multipropósito, que permite:
  - simulación y depuración de errores de aplicaciones.
  - simulación y depuración de aplicaciones en tiempo real.
  - simulación y/o depuración de errores de aplicaciones.
- La depuración de errores en diversos lenguajes: ensamblador, C, y C++.

- Simulación en tiempo real.
- Simulación de un diseño del hardware (tal como un tablero, un procesador, o un chip de I/O).

3.2.3.1 Instalación de CodeWarrior IDE.

En esta parte se explica cómo instalar el CodeWarrior IDE. También explica los directorios que el instalador de CodeWarrior crea, y tips y trucos para usar el CodeWarrior IDE con más eficacia.

Requerimientos del sistema.

- Procesador de Pentium o mayor (recomendado: Procesador de la clase del Pentium de Intel o microprocesador de la clase de AMD-K6).
- Por lo menos 128 megabytes del RAM.
- Aproximadamente 120 megabytes de espacio de disco duro libre para la instalación mínima.
- Aproximadamente 450 megabytes de espacio de disco duro libre para la instalación completa.
- Microsoft Windows 9X, Windows NT 4.0 con el service pack 3.0 del servicio, Windows 2000, o un sistema operativo más actual.
- Unidad de CD-ROM para instalar el software.

3.2.3.2 Instalación del software.

Para instalar el software de CodeWarrior:

1) Instalar el software CodeWarrior

A. - Cargar el CD-ROM en la unidad de CD-ROM.

Si el CD Auto play esta activo Windows los carga de forma automática.

Si el CD Auto play no es activo, corre install.exe en el CD-ROM.

B. - Development Studio for HC(s)12 Special Edition. Se encuentra en: *software\CW_3_1\setup.exe*.

C.- Hacer clic en Instalar.

2) Instalar el service pack CW12_V3_1_PE_V2.95_SP

Se encuentra en software\service pack *HC(s)12*\ *CW12_V3_1_PE_V2.95_SP.exe*

3) Obtener la licencia

A).- En inicio*programa**metrowerks**cw3_1**register code warrior*

B).- Se llena la forma y se envía a metrowerks que posteriormente hará llegar la licencia por correo.

license.dat Licencia de evaluación completa pero solo durante 30 días
special.dat Licencia especial permanente, permite compilar en C hasta 32 K.

La cual se deberá copiar en el directorio como *license.dat*, en el caso de la *special.dat*, se cambiará el nombre a license.dat.

C:\Archivos de programa\Metrowerks\CodeWarrior CW12_V3.1

4) Plantillas de RACOM

Se copia la carpeta *racom*, que se encuentra en la carpeta código en

C:\Archivos de programa\Metrowerks\CodeWarrior CW12_V3.1\Stationery

En el comienzo del proceso de la instalación. En los cuadros de diálogo que aparecen, se puede modificar la instalación para requisitos particulares para satisfacer sus necesidades.

3.2.3.3 Tips y trucos para el uso de CodeWarrior.

Aquí se muestran algunas tips y trucos para el uso el CodeWarrior IDE:

Si no se puede ocupar depuración en modo simulador, compruebe los ajustes en *Build Extras Preference Panel*. Compruebe también si la *Debugger* esta habilitado en la entrada del menú *Proyect* **.**

Si se borra la carpeta de los datos del proyecto y los ajustes relacionados con el proyecto. Uno de los ajustes se determina si la depuración y el simulador están habilitados. Por esa razón, no es recomendable borrar la carpeta de los datos del directorio del proyecto.

Si no se puede agregar un archivo a un proyecto, el problema puede ser que la extensión de archivo no está definida en el panel File Mappings.

Preferentemente se debe utilizar el panel *File Mappings Preference* de modo que el CodeWarrior IDE reconozca la extensión de archivo.

Si el software de CodeWarrior no funciona, comprobar si se tiene otra versión del CodeWarrior IDE instalada en la computadora. Si se tiene una versión más antigua del CodeWarrior IDE, o una versión para una diferente arquitectura, puede haber un conflicto entre las dos versiones. Para fijar el problema, se corre el archivo *regservers.bat.* El archivo corrige las trayectorias

definidas en el registro de Windows. Se puede encontrar el archivo *regservers.bat* en el compartimiento del directorio de instalación.

3.2.4 Menú principal y barra de tarea de CodeWarrior.

En la Figura 3.1 muestra el menú principal de CodeWarrior.

Figura 3.1 Menú Principal de CodeWarrior.

La barra de tareas es localizada bajo la barra de menú principal. Esta barra contiene los iconos que representan muchas de las órdenes de menú.

Recomendación: Colocar el cursor sobre un icono o el objeto de pantalla de GUI, un tooltip aparecerá y brevemente describe el objeto bajo el cursor.

3.2.4.1 Build Targets (Construcción de objetivos)

Cada proyecto contiene uno o varios *Build Targets*. Cada Build Targets en un proyecto es una colección de archivos que el IDE suele crear (construyen) como un archivo de salida. El proyecto Build Targets puede compartir unos o todos sus archivos.

3.2.4.2 Personalización y configuración de Objetivos.

Cada Build Targets de un proyecto tiene sus propias opciones que instruyen el IDE sobre como construir el archivo de salida. Las opciones controlan algunas cosas como la optimización de código, eliminación de fallos, advertencias de compilador.

También se puede configurar un Build Targets para depender de otro Build Targets en el proyecto.

Esto quiere decir que usted puede construir el software que combina los archivos de salida para objetivos de plataforma diferentes en un archivo de salida.

3.2.5 Project Windows (Ventana de Proyecto).

La ventana del proyecto contiene la información sobre los archivos y de Build Targets de un archivo de proyecto. En la Figura 3.2 se muestra la Ventana de Proyectos.

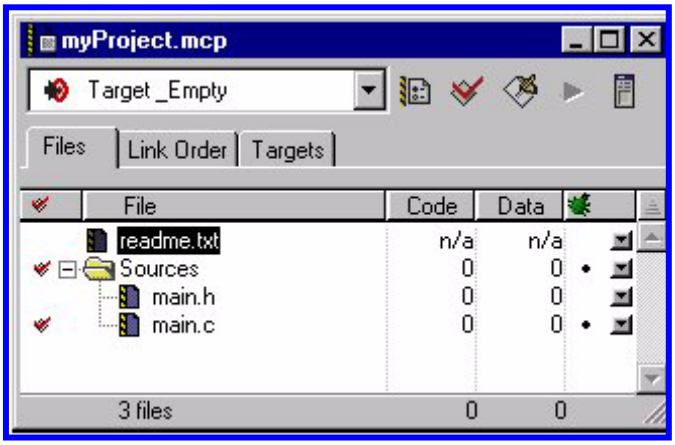


Figura 3.2 Ventana de Project Windows.

La ventana de proyecto tiene tres vistas (opiniones) diferentes:

- Archivos
- La Orden de la instrucción.
- Objetivos

Para seleccionar una opinión, pulse la etiqueta correspondiente.

La vista de archivos

La vista de archivos de la ventana de proyecto muestra una lista de todos los archivos en un proyecto. Usted puede organizar los artículos en esta opción en grupos jerárquicos. Es decir usted puede crear carpetas y arreglar los archivos en un sentido que usted lo desee.

Crear un grupo

1. De la barra de menú principal, seleccione el Proyecto> Crean el Grupo... Aparece el cuadro de diálogo de Crear Grupo.

2. Escribir un nombre para el nuevo grupo en el cuadro de diálogo de Crear Grupo.

3. Clic Ok. Un nuevo grupo (la carpeta) aparece en la opinión Files (Archivos) de la ventana de Project (proyecto).

Mover uno o varios archivos o grupos en la opinión de Files de la ventana Project:

1. Seleccionar los archivos o grupos para ser movidos.

2. Arrastrar los archivos seleccionados o grupos a su nueva ubicación en la ventana de proyecto (Project Windows). Una barra de subrayar indica donde los archivos seleccionados serán movidos cuando usted libera el botón de ratón.

3. Cuando la barra está en el archivo deseado o la posición de grupo, deshabilitar el botón de ratón. El IDE mueve los archivos seleccionados o grupos a la nueva posición.

3.2.6 Crear un proyecto.

1.- Desde el menú principal seleccionar File>New.

Aparecerá una ventana Nueva (Figura 3.3):

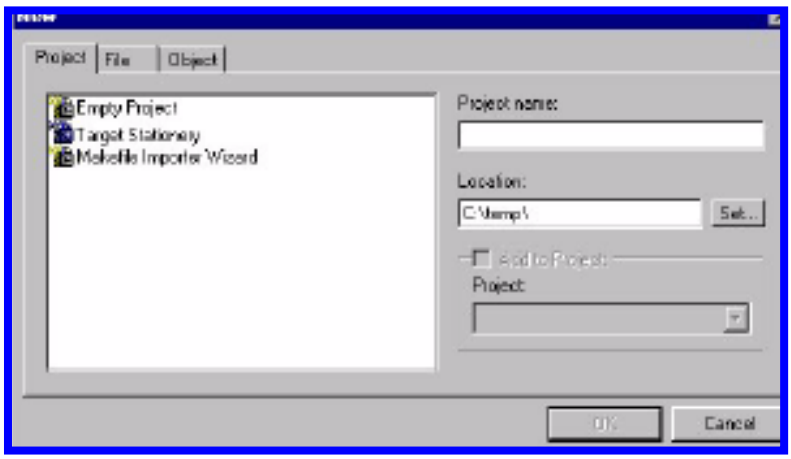


Figura3.3 Ventana New.

2. Click en la etiqueta de proyecto para mostrar el panel de proyecto (la Figura 3.3).

3. Seleccionar Target Stationary en el que usted quiere crear el nuevo proyecto.

4. En Project Name, escribir el nombre de tu proyecto.

5. Hacer Click en New Project.

3.2.7 Construir un proyecto.

1.- Hacer Click en Files en la ventana de Proyecto (Project Windows).

Aparece el panel Files. El panel Files muestra los nombres de todos los archivos de tu proyecto. En la Figura 3.4 se muestra el panel de archivos de la ventana de Proyecto.

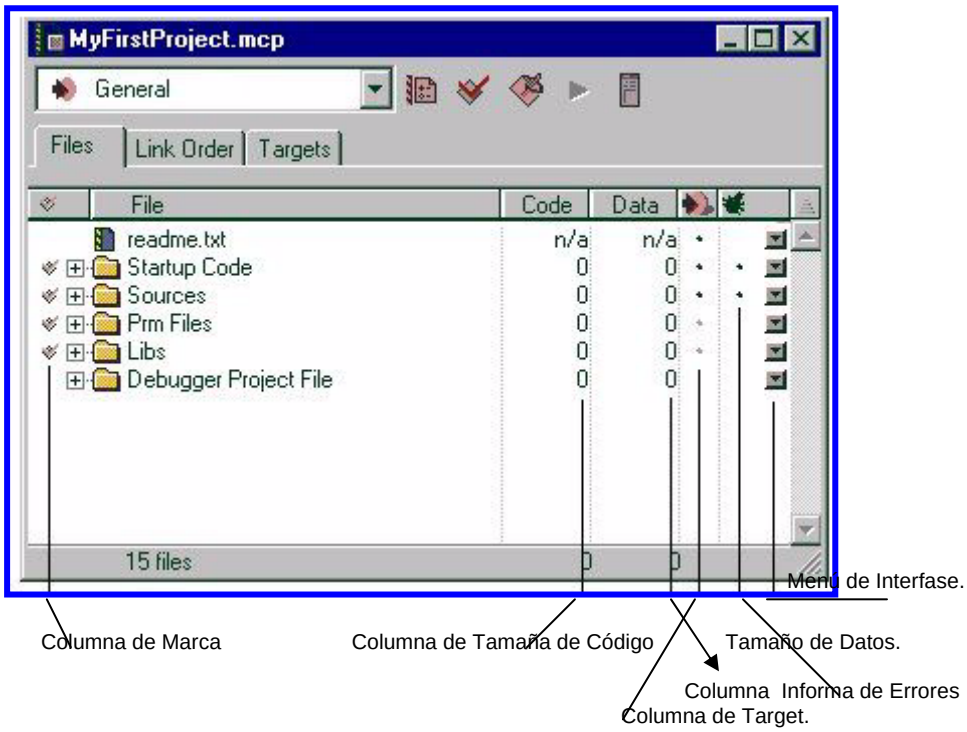


Figura 3.4 Panel Files de la ventana Project.

3.2.8 Botón Make.

Al utilizar este botón (Figura 3.5) se muestra las ventanas de errores y advertencias generadas en el código del proyecto (Figura 3.6).

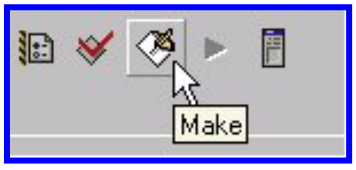


Figura 3.5 Botón Make.

Este botón nos permitirá realizar la compilación del programa generado en Code Warrior. En la siguiente figura se muestra la ventana de advertencias que aparecerá en caso de que el código realizado tenga algún error.

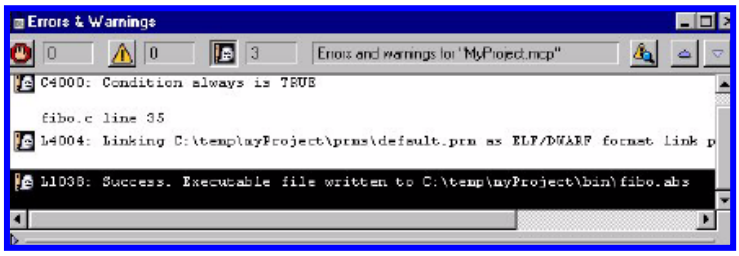


Figura 3.6 Ventana de Errores y Advertencias.

3.3 USO DE SIMULADOR/DEPURADOR[31]

3.3.1 ¿Que es Simulador/Depurador?.

El simulador/depurador es un miembro de las herramientas para desarrollo de sistemas embebidos. Son herramientas multipropósito que pueden usarse en sistemas embebidos y en el control de la industria mundial.

Algunos propósitos típicos son:

- Simulación y depuración de una aplicación embebida.
- Simulación y depuración de aplicaciones embebida en tiempo real.
- Simulación y/o depuración de una aplicación embebida.
- Multilenguaje de depuración: Ensamblador, C y C++.
- Simulación en tiempo real.
- Simulación de un diseño de hardware.

3.3.2 ¿Que es una aplicación de simulación y depuración?

Una aplicación de Simulador/Depurador contiene un mecanismo y componentes de depuración enfocados a la realización de diversas tareas (por ejemplo a simulación o depuración de una aplicación). El mecanismo de Simulación /Depuración es la parte central de un sistema. Monitorea y coordina las tareas de los componentes. Cada componente del Simulador/Depurador

31 http://www.ciao.es/Code_Warrior_273990_3.PDF

tiene su propio funcionamiento (por ejemplo: depuración del código fuente, simulación de I/O, etc.).

El uso de la Simulación/Depuración se puede adaptar a necesidades específicas. Se puede integrar o quitar componentes de Simulación/Depuración (Figura 3.7) de una forma muy sencilla. También se puede elegir automáticamente una configuración.

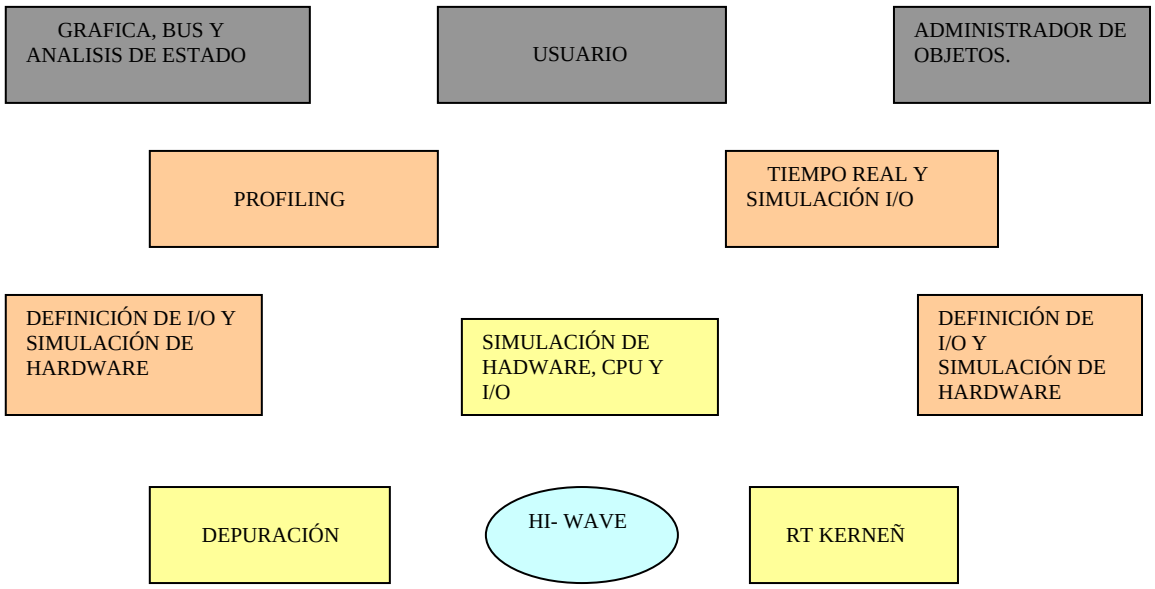


Figura 3.7 Ejemplo de una aplicación de Simulador/Depurador

3.3.3 Simulación/Depuración

La ventana principal maneja una diversidad de los diferentes componentes de esta (el menú Simulador/Depurador). Las ventanas de los componentes se organizan de la siguiente manera:

- De acuerdo a la estructura de la ventana.
- Auto estructura, las ventanas de los componentes se vuelven a clasificar automáticamente según su tamaño cuando la ventana principal se auto clasifica.

- Traslapado.
- Icono (ventanas que se pueden minimizar).

3.3.4 Programas de aplicación

Después de la instalación, todos los programas ejecutables se ponen en el subdirectorio “prog”, por ejemplo si usted instalo el software en la dirección C:\Metrowerks' de la PC, todos los archivos del programa están situados en C:\Metrowerks\PROG.

La lista siguiente proporciona una descripción detallada de los archivos usados en la Depuración en C/C++.

hiwave.exe Archivo ejecutable de Depuración.

hibase.dll Función principal de depuración dll.

elfload.dll Debugger loader dll

*.wnd Componente de depuración.

*.tgt Tarjeta de archivo de Depuración.

*.cpu Archivos de contenido del CPU.

3.3.5 Inicialización de Depuración.

Esta sección explica cómo encender la depuración desde el IDE o desde una línea de comando

3.3.6 Inicialización de la Depuración desde IDE.

Usted puede iniciar la depuración desde el IDE oprimiendo el botón de Depuración de la ventana del proyecto.

El simulador y el depurador usan el archivo completo que se ha creado para utilizar el comando *Make*.

Simular y depurar los proyectos:

1.- Click en el botón de *Debug* (Figura 3.8) en la ventana de proyecto (Project window). El IDE comienza a depurar.

Figura 3.8 Botón Debug.

Al hacer click en el botón *Debug* aparecerá el menú principal de depuración. El IDE lee todas las aplicaciones de depuración.

2.- Coloque el indicador del mause sobre una declaración C en la ventana del Simulador / Depurador.

3.- Click en el botón derecho del mause. Se desplegará un Menú (Figura 3.9).

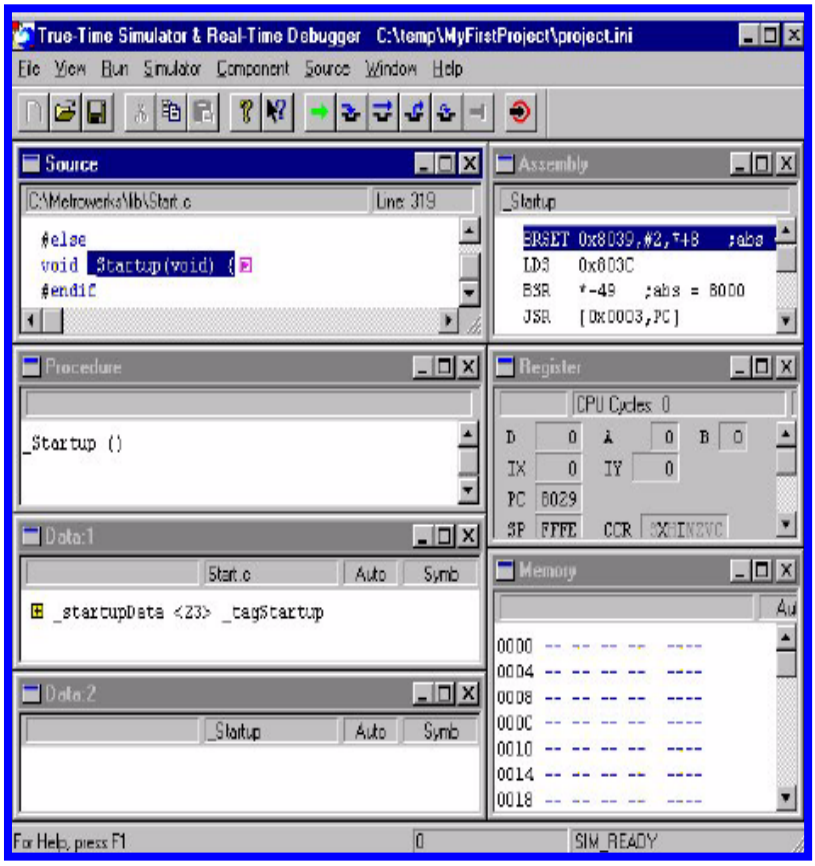


Figura 3.9 Ventana principal de depuración (Debugger).

4.- Seleccionar Breakpoin del menú desplegado.

5.- Hacer click en Run en la barra de tareas.

3.3.7 Barra de menú principal del Simulador/Depurador.

Esta barra de menú, mostrada en la Figura 3.10 se asocia a la función principal de la aplicación de depuración, de la tarjeta, y de las ventanas seleccionadas.

Figura 3.10 Barra de menú principal de Depuración

3.3.8 Simulador/Depurador Simulador/Barra de herramientas de Depuración.

Esta barra de herramientas (Figura 3.11) es definida automáticamente. La mayoría de los comandos del menú tiene un icono relacionado con la barra de herramientas del depurador.

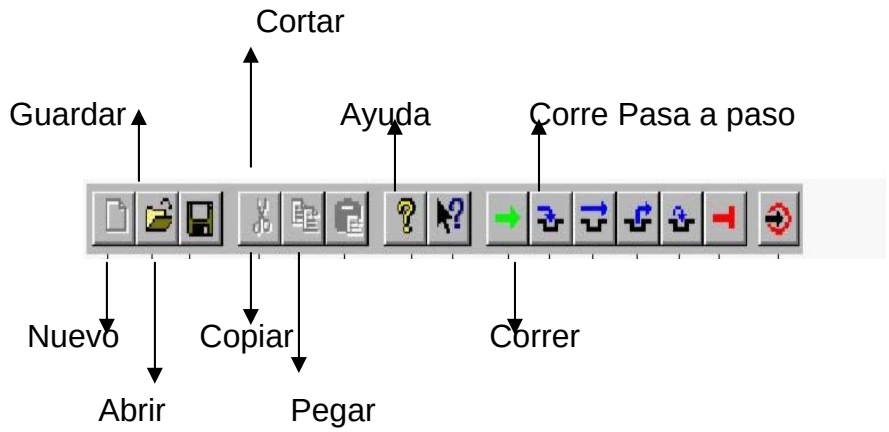


Figura 3.11 Barra de herramientas de la ventana de Depuración.

3.3.9 HI-WAVE.

La facilidad de la configuración de memoria es una parte integral del desarrollo del HI-WAVE con sus posibles configuraciones de la tarjeta. La memoria se divide en bloques. Un manager de memoria maneja la lista de los bloques de esta. La facilidad de la configuración de memoria también ofrece un cierto grado de automatización, pero no restringe la flexibilidad del ajuste del manual. La facilidad de configuración de memoria puede dejar especificar tipos y características de los bloques de memoria, por ejemplo RAM, ROM y así sucesivamente. La facilidad de la configuración de memoria utiliza un formato del archivo binario para leer y para fijar la configuración de la tarjeta.

3.3.10 Inicialización de la depuración con una línea de comando

Se puede inicializar la depuración *HI-WAVE* desde (DOS) con la línea de comando.

La sintaxis del comando se muestra a continuación:

HIWAVE.EXE [<AbsFileName> {-<options>}]

Donde AbsFileName es el nombre de la aplicación para cargar la depuración. Las opciones se pueden introducir por el mínimo carácter.

Las opciones son:

- **T**=<tiempo> modo prueba. La depuración terminará después de un tiempo específico. El valor dado es de 300 seg., por ejemplo:

c:\Metrowerks\prog\hiwave.exe *-T=10*

La depuración terminará después de 10 segundos.

Target=<nombre de tarjeta> pone las especificaciones de tarjeta, por ejemplo:

C:\Metrowerks\prog\hiwave.exe
c:\Metrowerks\demo\hc12\sim\fibo.abs -w *-Target=sim*

Inicia la depuración, fija la tarjeta de depuración, y carga el archivo fibo.ads.

W: modo espera- espera incluso cuando el <exeName> es especificado.

Instante = %currentTergetName: define un nombre instantáneo para el arreglo. Cuando el arreglo es definido, este mismo será utilizado, ejemplo:

c:\Metrowerks\prog\hiwave.exe *-Instance=%currentTargetName*

Ahora si se procura encender la depuración otra vez, al instante de la depuración se trae al primero plano.

- **Prod**: especifica la dirección del proyecto y/o el archivo del proyecto que se utiliza en el inicio.

Prod = <nombre del archivo> ejemplo:

c:\Metrowerks\prog\hiwave.exe *-Prod=c:\demoproject\test.pjt*

- Nodefaults: no cargará la automáticamente.

c:\Metrowerks\prog\hiwave.exe *-nodefaults*

- Cmd: especifica un comando para ser ejecutado a la inicialización: **-cmd** = "" {caracteres}, por ejemplo:

c:\Metrowerks\prog\hiwave.exe -cmd="open recorder"

- C: especifica en archivo de comando que es ejecutado al inicializar.**-c** <cmdFile>

c:\Metrowerks\prog\hiwave.exe -c c:\temp\mycommandfile.txt

-ENVpath: "-Env" <Environment Variable> "=" <Variable Setting>, esta opción fija una variable de entorno. Esta variable de entorno se puede utilizar para sobrescribir variables de entorno de sistema por ejemplo:

c:\Metrowerks\prog\hiwave.exe -EnvOBJPATH=c:\sources\obj

3.3.11 Componente de la línea de comando.

El componente del comando mostrado en la Figura 3.12 interpreta y ejecuta todos los comandos y funciones de Simulator/Debugger. La entrada del comando ocurre siempre en la línea pasada del componente del comando. Los caracteres pueden ser introducidos o pegados en la línea de edición.

Figura 3.12 Ventana de la línea de comando.

3.3.12 Lista de comandos disponibles.

Los comandos utilizados se clasifican en varios grupos, según sus acciones o tarjetas específicas. Sin embargo, estos grupos no tienen ninguna importancia en el uso de estos comandos. Una lista de todos los comandos en su grupo respectivo se da abajo:

3.3.13 Comandos Kernel.

Los comandos Kernel son los comandos que se pueden utilizar para construir programas de comando. Pueden ser utilizados solamente en un archivo de comando de depuración, puesto que la línea de comando puede aceptar solamente un comando a la vez. Es posible construir programas de gran alcance combinando comandos de Kernel con comandos base, comandos comunes y comandos específicos de componentes. La Tabla 3.1 contiene todos los comandos disponibles del Kernel.

Sintaxis de Comando	Descripción
A	Afecta un valor
AT	Fija un retraso para la ejecución del comando y ejecuta un archivo de comando
CALL fileName**[;C][;NL]**	Ejecuta un archivo de comando
DEFINE symbol [=] expression	Define un símbolo de usuario
ELSE	Otra operación asociada con el comando **IF**
ELSEIF condition	Otra operación condicional asociada con el comando **IF**
ENDFOCUS	Reajusta el foco actual (refiera al comando del **FOCUS**.)
ENDFOR	Salida a un lazo **FOR**
ENDIF	Sale un condicional **IF**
ENDWHILE	Sale un lazo **WHILE**
FOCUS component	Pone un alerta en un componente especifico
FOR [variable =]range ["," step]	Instrucción **FOR**
FPRINTF (fileName,format,parameters**)**	Instrucción **FPRINTF**
GOTO label	Incondicional de archive de comando
GOTOIF condition Label	Condicional de archive de comando
IF condition	Condicional ejecución
PAUSETEST	Despliega un cuadro de mensaje
PRINTF ("Text:," value]**)**	Instrucción **PRINTF**
REPEAT	**Instrucción REPEAT**
RETURN	Regresa desde un comando CALL
TESTBOX	Despliega un cuadro de mensaje de una secuencia
UNDEF symbol | *	No define un símbolo

Tabla 3.1 Comandos de Kerler.

3.3.14 Comandos Base.

Son utilizados para monitorear las tarjetas de ejecución de Simulator/Debugger. Los archivos de la entrada-salida de la tarjeta, la tarjeta de control de ejecución, edición de la dirección de memoria, manejo del punto de ruptura y administración de los archivos de CPU son otras funciones que desempeñan estos comandos. Los comandos base pueden ser ejecutados independientemente de los componentes que están abiertos. La Tabla 3.2 contiene todos los comandos de base disponibles.

Sintaxis de comando	Descripción
BC address|*	Elimina un punto de ruptura
BS address|function [P|T[state]]	Activa un punto de ruptura
CD [path]	Cambia el directorio de trabajo actual
CR [fileName][;A]	Abre un archive de registro
DASM	Remonta
DB [address|**range]**	Muestra los bytes de memoria
DL [address|**range]**	Muestra los bytes de memoria como palabra
DW [address|**range]**	Muestra los bytes en forma de palabra
G [address]	Comienza la ejecución del uso cargado actualmente.
GO [address]	Comienza la ejecución del uso cargado actualmente.
LF [fileName][;A]	Abre un archivo largo
LOG type [=] state {[,] type [=] state}	Habilita o inhabilita el registro de un tipo de información especificado
MEM	Muestra el mapa de memoria
MS range list	Coloca bytes de memoria
NOCR	Cierra archivos de memoria
NOLF	Cierra un largo archivo
P [address]	single assembly steps into program
RESTART	Reinicia la ejecución cargada
RD [list|*]	Muestra el contenido del registro
RS register[=]value{,register[=]value}	Activa un registro
S	Para la ejecución de una aplicación
STEPINTO	Activa la siguiente instrucción de la aplicación cargada
STEPOUT	Ejecuta un programa fuera de una función llamada
STEPOVER	Avanza a la siguiente instrucción de la aplicación cargada
STOP	Detiene ejecuciones que están cargas
SAVEBP on|off	Guarda puntos de ruptura
T [address][,count]	Remonta instrucciones de programa a una dirección especifica
WB range list	Escribe bytes
WL range lis	Escribe largas palabras
WW range list	Escribe palabras

Tabla 3.2 Comandos Base.

Ejemplos del uso de comandos para Simulador/Depurador:

```
1.-   in>a counter=8
```

La variable **counter** tiene el valor de 8.

```
2.-   in>A day1 = "monday_8U"
```

La variable **day1** tiene un nuevo valor **monday_8U**.

```
3.-  in>A value = "3.3"
```

Los comandos mostrados anteriormente solo son algunos ejemplos que utiliza la línea de comandos para el manejo de la depuración de una aplicación.

3.4 Interface GDI.[32]

Una característica avanzada del depurador de Metrowerks para el desarrollo sistemas embebidos es la capacidad de cargar diversos interfaces en la tarjeta.

El depurador Metrowerks 8/16 bits (Metrowerks CodeWarrior) puede ser conectado al hardware del HC12 y HCS12 usando el Instrumento de Interface de Depuración Generico (GDI). Un GDI DLL puede ser cargado vía el interface de la tarjeta GDI.TGT. El GDI DLL debe ser sujeto a las especificaciones del “Depurador Metrowerks de 8/16 bits conectado al instrumento de depuración usando el protocolo de interface GDI”.

Con este interface, se puede descargar un programa desde el Metroswerks CodeWarrior HC12 o HCS12, a un sistema externo de la tarjeta basado en un Motorota HC12 o HCS12, que ejecutará el programa. También se tendrá la regeneración de un sistema real del depurador del Metrowerks.

---

32 Code Warrior IDE Quichstart manual.

El debugger supervisará completamente el sistema de la tarjeta del MCU es decir la ejecución del CPU. Se puede leer y escribir en la memoria interna y externa cuando el MCU esta en modo base. Se tiene el control completo sobre el estado del CPU con la posibilidad de detener la ejecución, para proceder en modo de una simple secuencia y en la secuencia de punto de ruptura en el código.

3.4.1 Conexión de hardware.

El interface GDI de la tarjeta esta definido por un instrumento de interface de depuración genérica. No hay un cable específico para esta interface. De hecho el hardware de depuración esta limitado enteramente al cargar el GDI DLL para la tarjeta GDI.

3.4.2 Monitor Serial HCS12 en GDI.

El monitor serial HCS12 hcs12serialmon.dll esta sujeto a la interface de la tarjeta GDI y recargable vía el interface de la tarjeta GDI en CodeWarrior. Estas especificaciones de implementaciones de comunicación GDI DLL son descritas en la aplicación de Motorota HCS12 Serial Monitor.

Cuando la depuración se esta realizando la hcs12serialmon.dll dentro de la interface de la tarjeta GDI, puede comunicar y depurar el hardware corriendo el Monitor Serial HCS12.

3.5 Motorola HCS12 Serial Monitor.[33]

Esta aplicación describe el programa monitor de 2 kbyte para la serie de microcontroladores del HCS12. Este programa sirve como soporte de 23 comandos primitivos de depuración que permiten la programación y depuración FLASH/EEPROM a través del interface de comunicación serie RS232 a una computadora personal. El monitor permite el uso de comandos para resetear la

33 http://www.motorola.com/mx/products

tarjeta del microcontrolador, leer o modificar la memoria (incluyendo FLASH/EEPROM), lee o modifica los registros del CPU con simples instrucciones como Go, HALT o TRACE. Para permitir que un usuario especifique la dirección de cada interrupción en una rutina, este monitor redirecciona los vectores de interrupción a un porción autoprotegida de la memoria FLASH justo antes de la protección del programa monitor. Este programa es pensado para ser genérico, así que su uso debe ejecutarse en cualquier derivado del HCS12. Al usar este programa monitor provee a los usuarios un método para evaluar microcontroladores programados (MC9S12E), programación y depuración de sus aplicaciones (HCS12) usando solamente un cable de serial de entrada y salida y un software para la computadora personal. El monitor no usa otra memoria RAM con excepción de la su propio uso. Porque el monitor usa al COP watchdog para una función de reseteo. Este ambiente de desarrollo asume que el usuario resetea a el monitor cuando realizan la depuración de aplicaciones. Si el código del usuario toma el control directamente desde el reseteo, entonces una interrupción SCI0 para entrar en el monitor, el monitor podría no funcionar por el SCI0 y los registros de inicialización de memoria podrían no estar inicializados pues lo estarían con un reset en el monitor.

Si la frecuencia falla el monitor no funcionará. Con el uso de las definiciones la velocidad del microcontrolador será a 24 MHz. Si el usuario modifica la velocidad del microcontrolador el programa monitor podría tener una falta.

3.5.1 Configuración de memoria.

El monitor puede trabajar con todos los miembros de la familia disponibles del HCS12. La suma de un nuevo dispositivo requiere crear algunas definiciones de memoria para el nuevo dispositivo. La definición de memoria se almacena en el archivo S12SerMonxrx.def. del cual se debe tener conocimiento de:

- El espacio de registro esta situado en la dirección $0000-$03FF.
- La memoria FLASH esta en cualquier dirección mayor a la $4000
- La memoria EEPROM esta limitada al espacio disponible en los registros de la RAM.
- Los dispositivos externos aunados al interfaz del bus externo multiplexado no están soportados.

3.5.2 Uso del puerto serial.

Para trabajar en monitor con el uso del puerto serial, se utiliza la interface serial SCI0. El monitor debe tener uso exclusivo de esta interfaz. El código del usuario no debe poner en ejecución la comunicación en el canal serial. El monitor acomoda la comunicaciones seriales RS232 con SCI0 a 115.2 Kbaud. Para los sistemas que requieren el uso del SCI0, se utiliza un sistema de desarrollo BDM que permite una sofisticada depuración. Alternativamente el programa monitor puede modificarse por el usuario para comprobar el estado de las interrupciones usadas para entrar en el monitor y ajustar la reubicación. Esta modificación permitirá al usuario el uso del puerto SCI, aunque la depuración de la rutina SCI0 provocará el uso imposible del monitor.

Durante la inicialización después de cualquier reseteo, se transmite una pausa larga antes de que ocurra otra comunicación de SCI. Esta pausa es cercana a 30 bit tiempo para asegurarse de que la computadora puede reconocer la pausa. Para establecer la comunicación con el monitor, el host debe enviar un retorno ($0D) en la velocidad correcta, si la velocidad del host es incorrecta o si el host detecta otro carácter continuará esperando antes de imprimir la primera secuencia del aviso.

En modo de carga, el monitor tendrá acceso completo del puerto SCI0. En modo RUN, el monitor comprobará para saber si hay un vector válido de la interrupción del usuario SCI0 y saltará a este vector si es válido.

3.5.3 Comandos del monitor

El monitor reconoce 23 comandos binarios primitivos que permiten al usuario una herramienta de desarrollo para crear un completamente equipado depurador de programa.

Estos comandos utilizan los códigos de comando 8-bit, seguidos opcionalmente por la dirección binaria, el control, y la información de datos, dependiendo del comando específico. En las descripciones siguientes del comando, se describe la sintaxis del comando. Cada comando comienza con un código de comando binario. Una raya vertical (/) que se utiliza para separar las partes del comando, pero estos caracteres de la raya vertical no se envía como caracteres seriales en comandos. Las partes subrayadas del comando se transmiten desde el host de la PC a la tarjeta del MCU mientras que las porciones que no se subrayan se transmiten de la tarjeta de MCU al host de la PC. Los primeros dos caracteres en cada secuencia del comando son el 1-byte del código de comando y se muestran en valor hexadecimal y en literal como A1.

En la siguiente Tabla 3.3 se muestra una lista de comandos del Monitor serial, los cuales están encargados del manejo de la depuración vía línea de comando.

Sintaxis de comando	Descripción
AAAA A	Dirección de 16 bits.
CCR	El contenido del registro de 8 bits del código.
D	El contenido del registro D de 16 bits, conformados de los acumulados A y B.
EADR	La dirección de 16 bits para un borrado o bloque de comando.
IDID	La división ID de 2 kbyte desde el registro PARTID.
IX	El contenido del registro indicado X de 16 bits.
IY	El contenido del registro indicado Y de 16 bits.
NN	El número de bytes para el bloque de lectura y comandos de escritura.
PC	El contenido del programa de usuario de 16 bits.
RD	Un byte de datos de lectura.
RDW	Una palabra de datos de lectura.
RDB(AAAA) / RDB(AAAA+1) /.../ RDB(AAAA+NN)	Una serie de 8 bits de datos de lectura.
SADR	El inicio de direccionamiento de 16 bits para un borrado o bloque de comando.
SP	16 bits para el usuario SP.
WD	Un byte de dato de escritura.
WW	Una palabra de dato de escritura.
WB(AAAA) / WB(AAAA+1) /.../ WB(AAAA+NN)	Una serie de 8 bits de datos de escritura.

Tabla 3.3 Comandos del Monitor Serial.

3.6 Processor Expert para Code Warrior.[34]

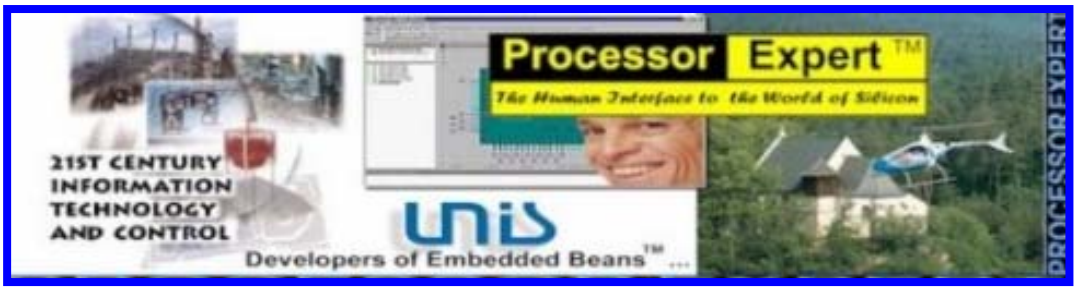


Figura 3.13 Processor Expert.

El hardware y el diseño del software han progresado tanto como las nuevas tecnologías, siempre con el avance diario, pero sus correlaciones e interdependencia se han descuidado sobre todo. En una parte, vemos a menudo una nueva arquitectura de hardware pero el diseño del software es demasiado costoso para tal arquitectura. Por otra parte, la automatización de casi todos los dispositivos mecánicos sobre todo en el mundo moderno conduce al uso de los sistemas informáticos encajados en situaciones donde está en consideración el costo, los sistemas informáticos con un eficaz software puede reducir perceptiblemente el coste del diseño total.

3.6.1 Características principales del Processor Expert

Processor Expert se diseña para el desarrollo rápido del uso de una amplia gama de microcontroladores y de sistemas del microprocesador.

Processor expert se basa en los componentes llamados *Embedded Bean.*

Embedded Bean encapsula la funcionalidad de los elementos básicos de sistemas aunados como los periférico de la base del CPU, dentro de los periféricos del chip del CPU, FPGA, los periférico independientes, los dispositivos virtuales, y los algoritmos puros del software y estas cambian las instalaciones a las características, los métodos y los eventos.

[34] Ayuda de Processor Expert.

Processor Expert sugiere, conecta y genera el conductor para el hardware, los periféricos o los algoritmos encajados del sistema. Esto permite que el usuario se concentre en la parte creativa del proceso del diseño del conjunto.

Processor Expert permite un verdadero estilo descendente para el diseño de la aplicación. El usuario comienza el diseño directamente definiendo el comportamiento de la aplicación.

Processor Expert trabaja con una biblioteca extensa de Beans para apoyar de microprocesadores, periférico y dispositivos virtuales.

Un usuario puede crear sus propias Bean usando la herramienta externa del *Bean Wizard*.

Las fuentes se pueden generar con varios lenguajes de programación incluyendo el ensamblador, ANSI-C, MODULA-2 y JAVA.

3.6.2 Acerca de Processor Expert.

La tarea principal del Processor Expert es manejar la CPU y otros recursos de hardware y permitir el prototipo y diseño virtual.

La generación de código desde Beans, la capacidad de mantener el uso y el código generado, y una estructura basada en un evento reduce perceptiblemente el esfuerzo de programación en comparación con las herramientas clásicas.

Un Bean es la encapsulación esencial de funciones. Por ejemplo el Bean de TimerInt encapsula todos los recursos de la CPU que proporcionen la sincronización y el hardware de interrupciones del CPU.

Se encontrarán muchos componentes que llamados Embedded Bean en el Processor Expert en la Ventana de Bean Selector. Estos componentes fueron seleccionados para cubrir la funcionalidad de los más comúnmente requeridos usos del microcontrolador, como manejar las operaciones de los bits de los puertos, las interrupciones externas, y los modos del contador de tiempo hasta las comunicaciones seriales asynchronous/synchronous y el uso del convertidor de analógico a digital, etc.

Un Bean proporciona el interfaz que permite al usuario controlar el tiempo de diseño para el uso del *Bean Inspector*. El Bean Inspector tiene varias páginas para ver las características de los Beans, sus métodos y sus acontecimientos.

Fijando características, un usuario define el comportamiento futuro del Bean en tiempo de rutina.

El usuario puede habilitar o inhabilitar aspectos de los métodos del Bean en código de fuente.

Los eventos, si están utilizados, se pueden habilitar por la interrupción del recurso de hardware (Timer.) o por una razón pura del software (desbordamiento.) en una aplicación de rutina en uso. También se puede habilitar o inhabilitar interrupciones usando métodos del Bean y definir la prioridad para la ocurrencia del acontecimiento y para ejecutar su rutina del servicio de la interrupción (ISR). El hardware ISR proporcionado por el Bean maneja la razón de la interrupción. Si el vector de la interrupción es compartido por dos (o más) recursos entonces este ISR proporciona la identificación del recurso. Entonces el usuario esta notificando el uso de eventos que maneja el código.

La creación de una aplicación en Processor Expert en cualquier microcontrolador de Freescale se hace rápidamente. Primero elija y presione

Bean CPU, agregue otros, modifique sus características, defina los eventos y elija el diseño del código. Processor Expert genera todo el código de Beans según sus ajustes.

Otros Beans pueden ayudar a incluir muy rápidamente cuadros, archivos y sonidos.

Los otros Beans pueden ser obtenidos de www.processorexpert.com o ser creadas de fuentes existentes. También otros Beans pueden incorporar Bean ya existentes. Pueden heredar sus características, métodos, y acontecimientos.

Si se desea compartir un Bean con otros reveladores. Por ejemplo un Bean que puede conducir una exhibición de segmento del LED. Porque se utiliza a menudo para diversas configuraciones de hardware en diversos pines de la CPU entonces debe ser portable e independiente de los recursos de la CPU. Las tareas y de algoritmos se pueden poner en el Bean.

Tales Beans se llaman Beans del software (interruptor). Los Beans del interruptor pueden ser Beans puros del interruptor o pueden heredar incluso Bean múltiples que encapsulan diversos recursos. La ventaja es independencia en una capa física, una portabilidad y una parte del código una vez escrito y probado.

Para ejemplificar elegimos simplemente como los beans padres (BitIO, Beans de BitsIO o de ByteIO y de TimerInt de la biblioteca de Beans). El Bean nuevo de la exhibición de LED proporcionará las características de un tipo de la referencia del Bean para este. En tiempo del diseño esto permite el nuevo acceso del Bean a las características de sus padres y define los pines de la conexión física o los recursos del Timer. Además, el Bean nuevo tendrá sus propias características y métodos. Los métodos y los eventos se pueden construir usando los métodos de las Beans del padre. Los beans padres son beans nuevos que tienen recursos que han sido tomados de otros beans.

Al elegir Processor Expert en el menú de herramienta del Bean Wizard se podrá definir los rangos. Además solo se podrá introducir el código de métodos y de eventos, guardar nuevo Bean e instalarlo en la paleta de Bean o compartirlo con otros.

Para la información adicional sobre Processor Expert y los Beans que las bibliotecas satisfacen, en la opción Ayudar o en sitio Web de www.processorexpert.com.

Como parte fundamental en el siguiente capítulo (IV) se explicará detalladamente otros aspectos importantes del uso de CodeWarrior para el desarrollo de diversas prácticas. Además se explicará las funciones utilizadas por el sistema que estamos ocupando (sis9s12E), así como también las especificaciones de funcionamiento de este sistema.

Capítulo IV

PROPUESTA DE TECNOLOGÍA PARA EL DESARROLLO DE PRÁCTICAS DEL SISTEMA SIS9S12E.

Propuesta:

Las herramientas de desarrollo de Freescale y Metrowerks CodeWarrior utilizadas en la realización del proyecto ayudan a acelerar el diseño de sistemas con MCU de alta funcionalidad y eficiencia de costos.

El microcontrolador HCS12 usado en el sistema de desarrollo SIS9S12E es una solución de 16 bits diseñado para proveer la flexibilidad que necesita el diseñador de sistemas electrónicos para mejoras económicas de software por medio de programabilidad y reprogramabilidad en la aplicación.

Con el uso de CodeWarrior para HCS12 se ofrece simulación, programación Flash, inicialización automática y generación de código de controlador con la herramienta de diseño de aplicación rápida de Processor Expert, un ensamblador, enlazador, y un compilador C limitado de tamaño de código y un depurador de nivel de fuente C. El Ambiente de Desarrollo Integrado (IDE) de CodeWarrior brinda una interfaz con el usuario gráfica intuitiva que es común en todos los microcontroladores de Motorola para una migración fácil. Los alumnos de ingeniería en electrónica pueden acelerar el tiempo para generar una aplicación al crear, compilar, enlazar, ensamblar y depurar todo de un sólo IDE. La preparación inicial y los pasos de proceso múltiples se pueden gestionar con un sencillo click.

El uso del sistema de desarrollo SIS9S12E es utilizado en el Centro de Investigaciones Avanzadas de Ingeniería Industrial de la Universidad Autónoma del Estado de Hidalgo (UAEH). Con este sistema se busca involucrar al alumno

del Instituto de Ciencias Básicas e Ingeniería (ICBI), a un sistema de microcontroladores de la familia de Motorota (Freescale).

El sistema SIS9S12E permitirá al alumno de esta área, manejar diversos dispositivos electrónicos, que permitirán el desarrollo de proyectos futuros, facilitando así el proceso de trabajo.

Con la simulación de software, los alumnos de ingeniería de la UAEH pueden empezar a diseñar inmediatamente, sin esperar al hardware o sin requerir una tarjeta de evaluación o emulador. También está diseñado para verificar el diseño en base de los recursos y opiniones de temporización reales de MCU para ayudar a captar problemas posibles antes que los diseñadores empiecen a depurar.

El simulador y potente depurador incluidos les dan a los usuarios los recursos que necesitan para desarrollar aplicaciones, mientras que el software de programación Flash permite la programación del código del prototipo.

Por mencionar unas propuestas de aplicación en el área de ingeniería electrónica, se puede señalar, muestreo de señal, discretización de una señal, sensores de velocidad, entre otros. Además está dirigido a aplicaciones tales como instrumentación, manejo de energía, robótica, control industrial y sistemas de seguridad.

El microcontrolador HCS12 que es utilizado en este sistema de desarrollo (SIS9S12E-HCS12) ha generado grandes ganancias para motorota en diseños automotrices, por ello la importancia de implementar el manejo de este microcontrolador en los alumnos de ingeniería.[40]

[40] http://www.motorola.com/mx/products

Anteriormente se han mencionado los beneficios que puede traer el uso de este microcontrolador, por ello es importante este Manual de prácticas del Sistema de Desarrollo SIS9S12E, en el que se da una amplia explicación del uso y configuración de cada bean y de sus funciones utilizadas en CodeWarrior, está metodología llevará de la mano al usuario para poder realizar cada una de las prácticas que se desarrollaron, además permitirá que el alumno tenga conocimiento del manejo de todos los recursos que se utilizaron en el desarrollo del proyecto.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

INSTRUCCIONES DE ALIMENTACIÓN Y CONEXIÓN DEL MICROCONTROLADOR A LA PC.

Sistema de Desarrollo SIS9S12E

MATERIAL:

- Microcontrolador SIS9S12E.
- Cable de Puerto Serie(macho –hembra)
- Instalación del programa Metrowersks CodeWarrior IDE.
- Fuente de 12 V en CD.

INTRUCCIONES.-

Una vez instalado el software se realiza la conexión de sistema SIS9S12E como se muestra en la Figura 3.14

SIS9S12E

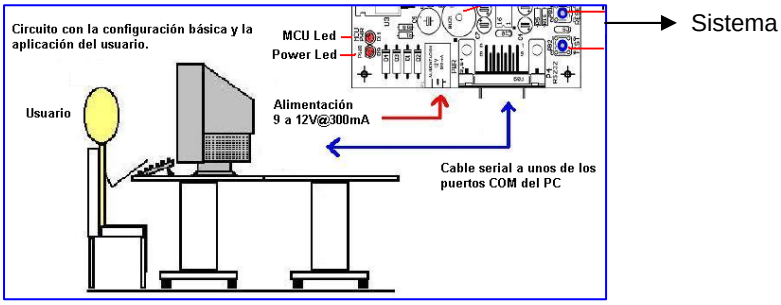


Figura 3.14 Esquema general de configuración.

1.- Conectar el cable de Puerto Serie al microcontrolador. Tomar en cuenta la configuración (macho-hembra). En la Figura 3.15 se muestra el diagrama de pines del interface RS232

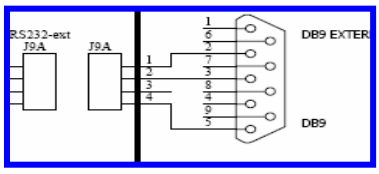


Figura 3.15 Diagrama de pines de la interface RS232.

2.- Conectar el cable de Puerto Serie al CPU de la computadora.

3.- Conectar la fuente regulada de 12 V al microcontrolador.

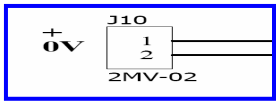


Figura 3.16 Conexión de la fuente de Alimentación.

En la Figura 3.16 se mostró el diagrama de conexión de la fuente de alimentación.

La siguiente Figura 3.17 muestra las conexiones del sistema de desarrollo SIS9S12E.

Figura 3.17 Conexiones del sistema SIS9S12E.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 1

NOMBRE: Manejo de los puertos como entrada-salida del sistema de desarrollo SIS9S12E.

.

OBJETIVO: Que el usuario aprenda a configurar los puertos del microcontrolador como I/O digitales.

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v en C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior* IDE en el menú Inicio.

4.- En el menú elegir *File-New* (Figura 4.1).

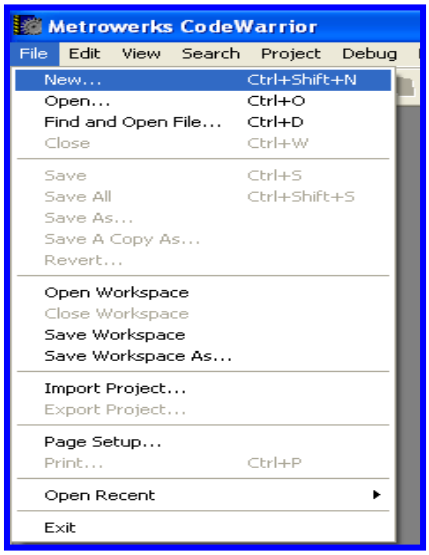


Figura 4.1 Menú File-New

5.-Seleccionar la opción la opción CH(S)12 *New Project Wizard*, proporcionándole nombre del proyecto y la dirección para guardar el proyecto. Elegimos *Aceptar*. La Figura 3.2 muestra la ventana de Nuevo Proyecto.

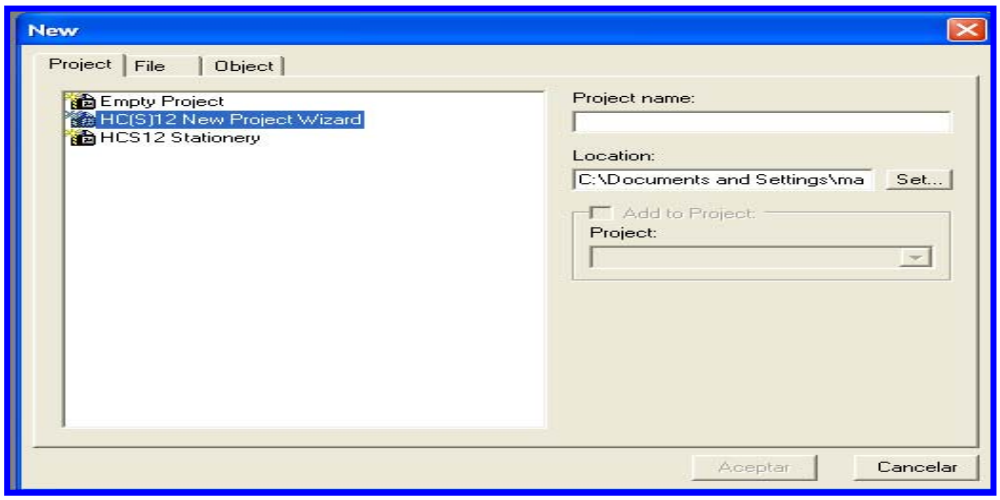


Figura 4.2 Ventana de Nuevo Proyecto.

6.- Elegimos el modelo del microcontrolador que usamos, que este caso sería: MC9S12E64. Y elegimos *Siguiente*. En la Figura 4.3 se realiza la selección del modelo del microcontrolador que vamos a utilizar.

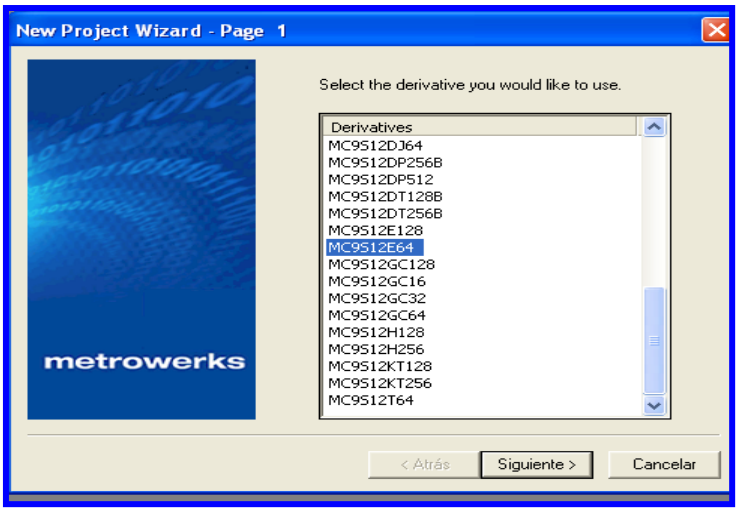


Figura 4.3 Ventana para seleccionar la familia de microcontrolador.

7.- Seleccionamos el lenguaje que utilizaremos para programar el microcontrolador, para el manejo del *MC9S12E64* se utilizará el lenguaje C. Elegimos *Siguiente*. En la Figura 4.4 se realizará la elección del lenguaje de programación que se utilizará.

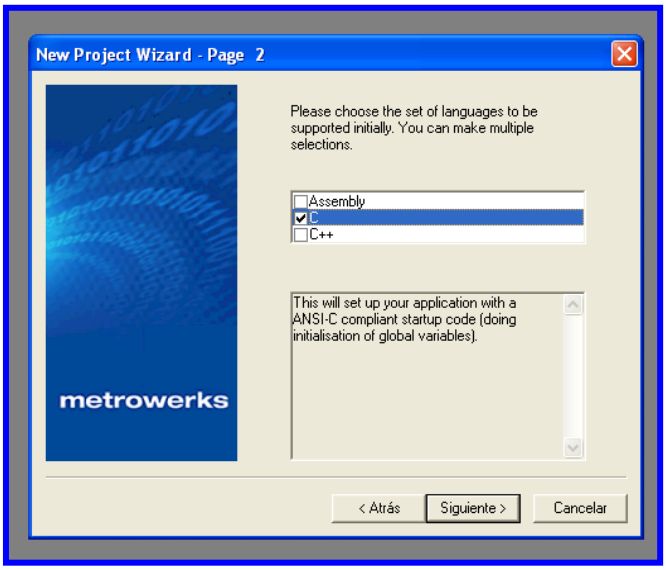


Figura 4.4 Ventana de selección de lenguaje de programación.

8.- Elegiremos el uso del *Processor Expert*, colocando en la opción Yes (Figura 4.5).

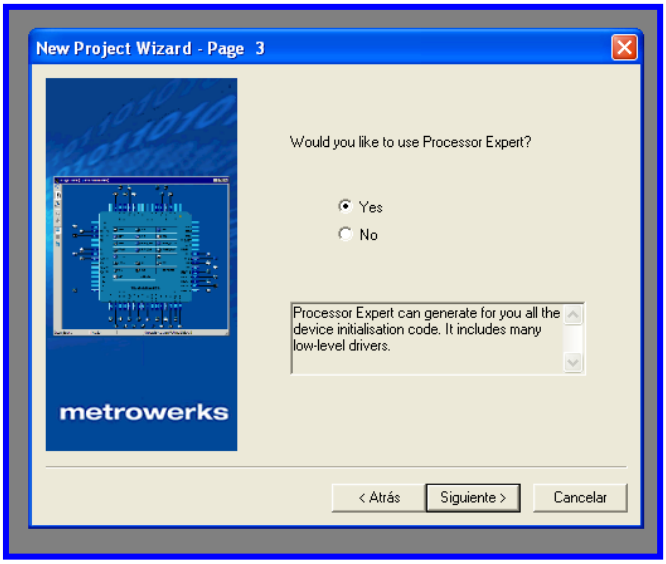


Figura 4.5 Ventana se selección de Processor Expert.

9.- Seleccionaremos la opción si queremos crear un proyecto para *PC-Int* (Figura 4.6). En este caso elegiremos la opción No. Hacer clic en Siguiente.

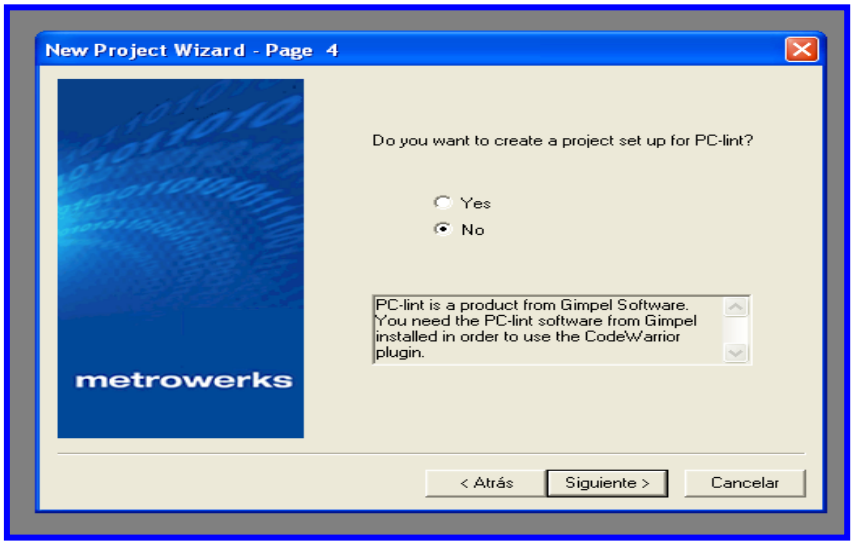
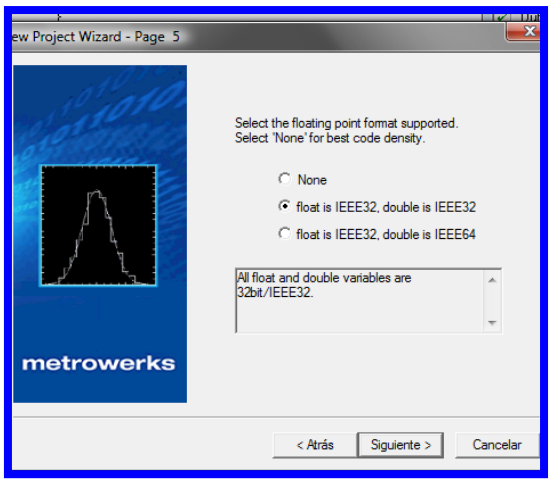


Figura 4.6 Ventana de PC-int

10.- Al momento de configurar o cargar el programa para utilizar el tipo de variable seleccionaremos la siguiente: (utilizaremos una variable flotatente). La Figura 4.7 muestra la ventana para la selección de tipo de variable.



Es importante seleccionar el tipo de variable que vamos a utilizar ya que nos permitirá utilizar diversas variables al momento de realizar nuestro código de programación. La flotante seleccionada es la IEEE32 de 32 bit.

Figura 4.7 Selección de tipo de punto flotante.

11.- Seleccionamos las conexiones que se quieran utilizar. Se pueden elegir conexiones múltiples, en este caso elegiremos *Metrowerks Full Chip Simulator y Motorota Serial Monitor Hardware Debugging.* Hacer clic en Finalizar. En la Figura 4.8 se muestra la ventana de selección de conexión.

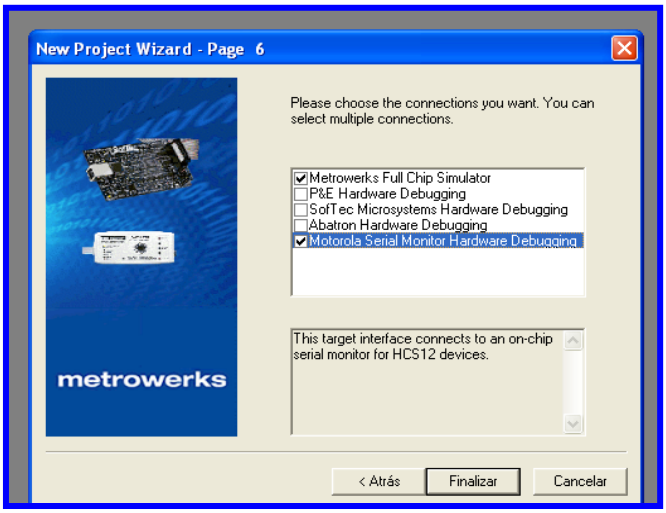


Figura 4.8 Ventana de selección de conexiones.

12.-En esta parte podremos observar toda la pantalla completa en donde se trabajará para la manipulación del microcontrolador MC9S12E64 (Figura 4.9).

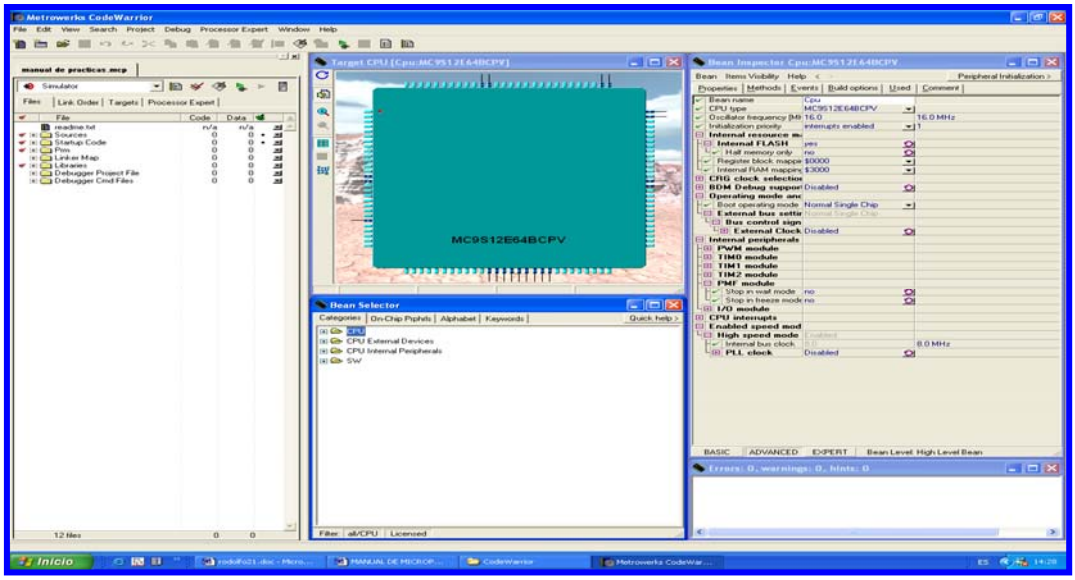


Figura 4.9 Ventana Principal de Metrowerks Codewarrior.

Aquí podremos observar diferentes ventanas:

a.- En la ventana Target CPU[Cpu: MC9s12E64BCPV] podremos observar los pines del microprocesador que utilizamos.

b.- En la ventana Bean Selector: Podremos elegir que dispositivos del microcontrolador queremos manipular.

c.- En la ventana Bean Inspector CPU[Cpu: MC9s12E64BCPV] se podrá observar todas las especificaciones de los dispositivos seleccionados que se van a utilizar.

d.- En la última ventana nos mostrará el nombre del proyecto. También podremos observar los dispositivos dados de alta, así como el programa que desarrollemos.

13.- Para realizar la práctica del Manejo de los puertos como entrada-salida del microcontrolador utilizaremos los dispositivos *Cpu Internal Peripherals* del *Bean Selector*, con los puertos de entrada-salida *port I/O.* En la Figura 4.10 se muestra la ventana de Bean Selector.

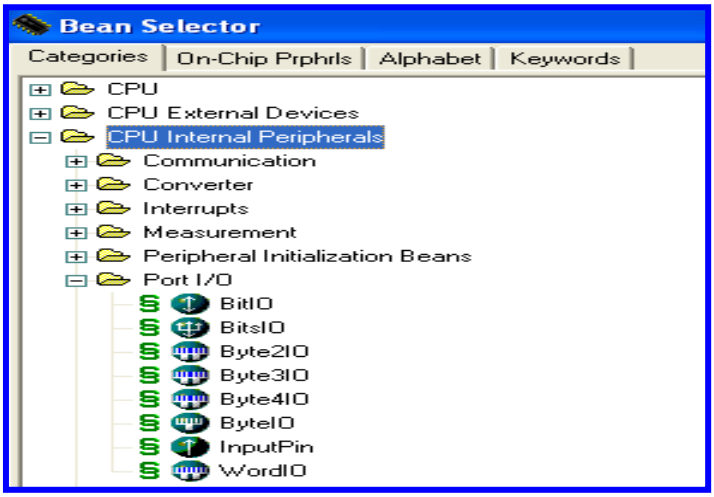


Figura 4.10 Ventana Bean Selector.

14.- Para la manipulación de un puerto de un bit como salida/entrada daremos doble clic en el icono BitIO .

De esta forma daremos de alta un solo bit de salida/entrada, en el caso de que quisiéramos dar de alta otro bits de salida/entrada se realizará de la misma forma.

15.-Configuramos el BitIO como un dispositivo de salida de la siguiente manera:

Para el uso de LEDs como dispositivos de salida será necesario dar de alta el puerto U:

1.-PU0_/IOC24/PW10

2.-PU1_/IOC25/PW11

3.-PU2_/IOC26/PW12

4.-PU3_/IOC27/PW13

Los Bits BitIO tendrán la siguiente configuración para ser utilizados en el puerto U como dispositivos de salida. En la Figura 4.11 se muestra la ventana de configuración del bean de un bit.

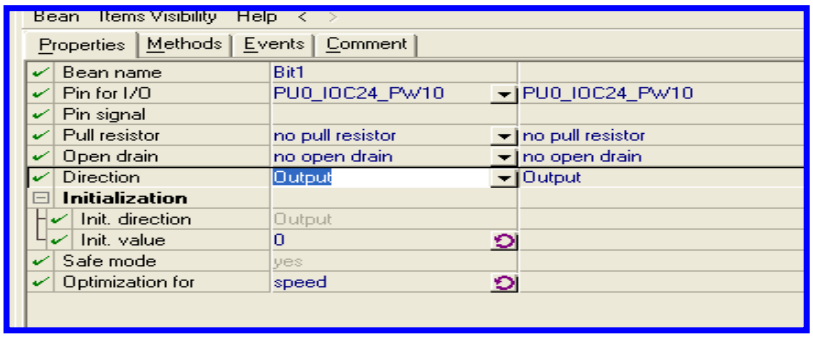


Figura 4.11 Ventana de configuración del bean Bit.

16.- Como siguiente paso cargaremos todos los dispositivos que seleccionamos:

- En el menú principal (Figura 4.12) seleccionamos *Proccesor Expert-Generate Code ´nombre.mcp´*

Figura 4.12 Menú de Processor Expert.

17.- En el siguiente menú (Figura 4.13) podremos manipular nuestros dispositivos cargados y desarrollar nuestro programa con el cual manipulamos el microcontrolador. Hacer doble clic en *Files-User Models- nombre.c.* como se muestra en la siguiente imagen.

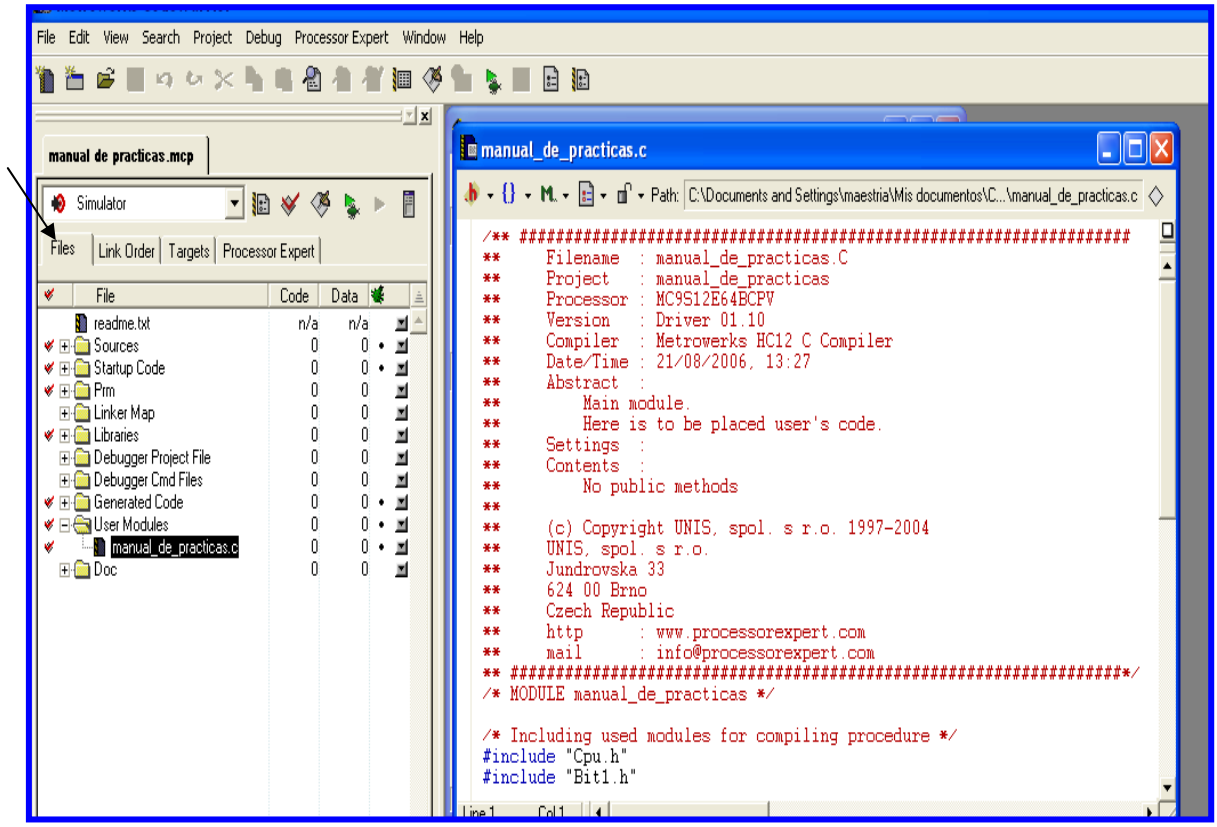


Figura 4.13 Ventana de desarrollo de código de programación.

18.- Hacer doble clic en *Processor Expert-Beans*, al realizar esta operación nos desplegará los dispositivos cargados (Bean), y al abrir cada dispositivo nos aparecerá las funciones que pueden utilizar. En la Figura 4.14 muestra las funciones del bean de un bit.

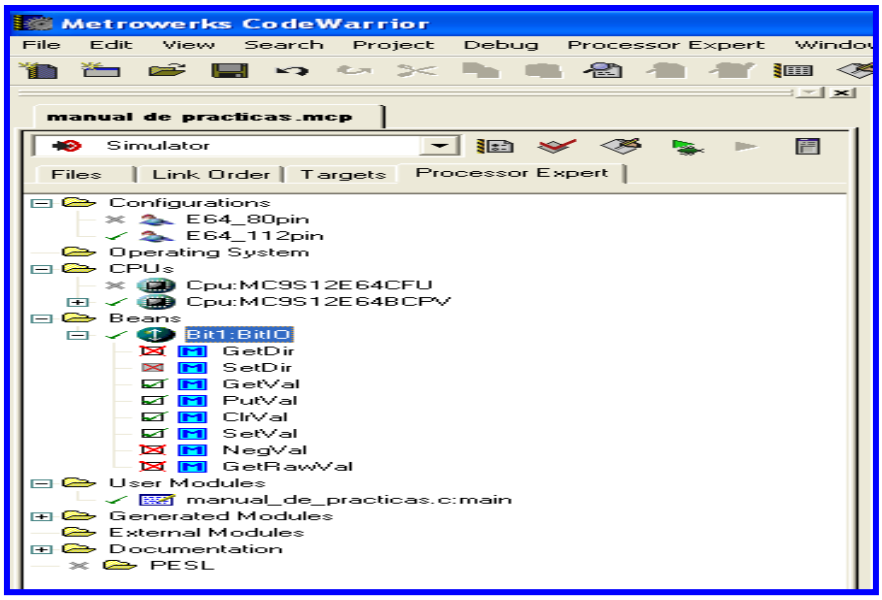


Figura 4.14 Funciones del Bean Bit.

Todas las funciones desplegadas por cada Bean nos permitirán una programación más sencilla.

El nombre de los Beans puede ser cambiado para un mejor manejo. Botón izquierdo. Ver la figura 4.15.

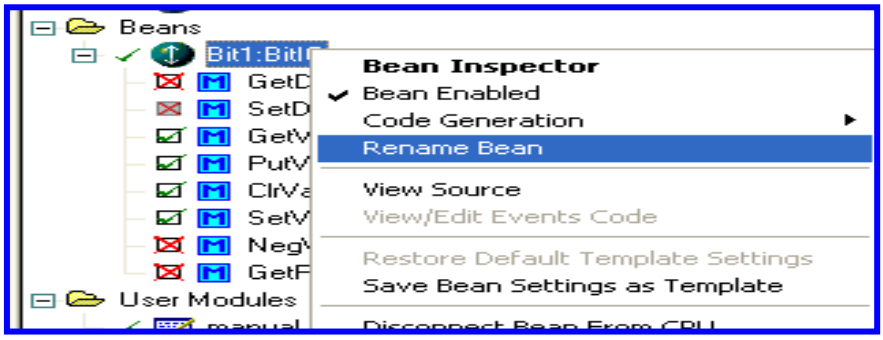
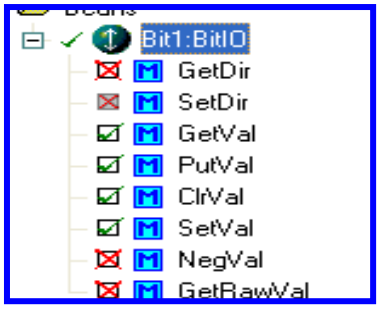


Figura 4.15 Renombrar el Bean Bit.

19.- Realizar el desarrollo del programa. En este caso el objetivo es la manipulación del puerto U, el cual estará configurado como salida.

EJERCICIO NO.1

Realizar un programa el cual permita encender el primer bit del puerto U (LED 1). Se utilizara la siguiente función del Bean BitIO . La Configuración del Bean se realizará de la misma forma como se explico anteriormente.



SetVal - Coloca un 1 en un bit de salida. Esta función se puede jalar haciendo clic sobre él y pegándolo en la pantalla donde se desarrolla el código.

Código del Programa.- La Figura 4.16 muestra la ventana del código de programación para encender un led.

```
/* Including used modules for compilling procedure */
#include "Cpu.h"
#include "Led1.h"
#include "Boton1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.


  /*Write your code here*/

        for(;;)

{

         LED1_SetVal()  ;
}
```

Figura 4.16 Código de programa (ejercicio 1).

20.- En el menú principal (Figura 4.17) seleccionamos *File-Save* para guardar el programa realizado, en el caso de que programa sea modificado este tiene que ser guardado nuevamente.

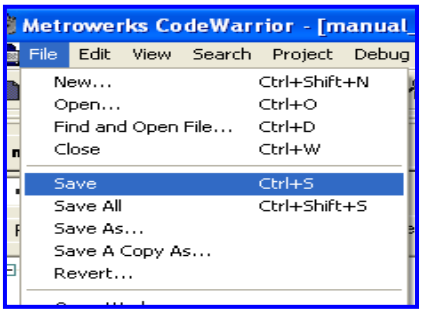


Figura 4.17 Opción guardar.

21.- Realizar la *compilación* del programa desarrollado. Hacer clic en el icono *Make* del menú (Figura 4.18).

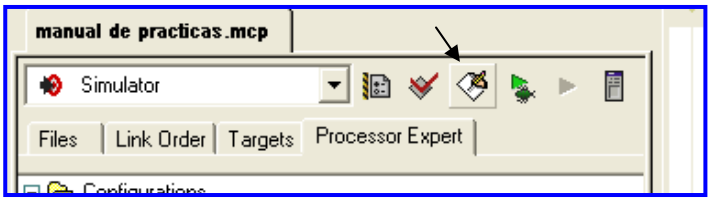


Figura 4.18 Icono Make.

Al realizar la compilación del programa detectará si hay algún error de programación o también alguna advertencia

22.- Elegir la forma que se desea visualizar el programa realizado. En forma de simulación o cargando el programa al microcontrolador. Elegiremos modo *Monitor*.

Figura 4.19 Selección de Monitor.

23.-Para ejecutar el programa ya sea en forma de simulación o en forma de monitor, hacer clic en el icono *Debug* del siguiente menú (Figura 4.20).

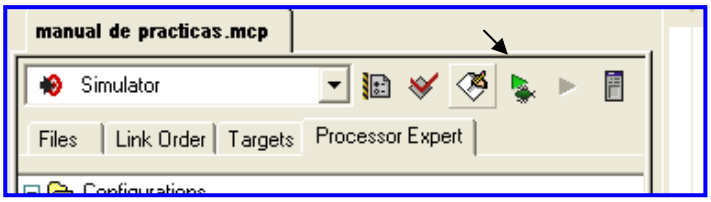


Figura 4.20 Icono Debug.

Al hacer clic en el icono anteriormente mencionado el programa podrá ser visualizado en el microcontrolador,

24.- En este punto se muestra como será cargado el sistema de desarrollo sis9s12E. Para que el microcontrolador pueda ser cargado con el programa realizado anteriormente debe de encontrarse en forma MONITOR, y para poder ver el resultado que mostrará el microcontrolador deberá estar en forma de USUARIO, y presionar el botón RESET del microcontrolador.

Como se muestra en la Figura 4.21.-

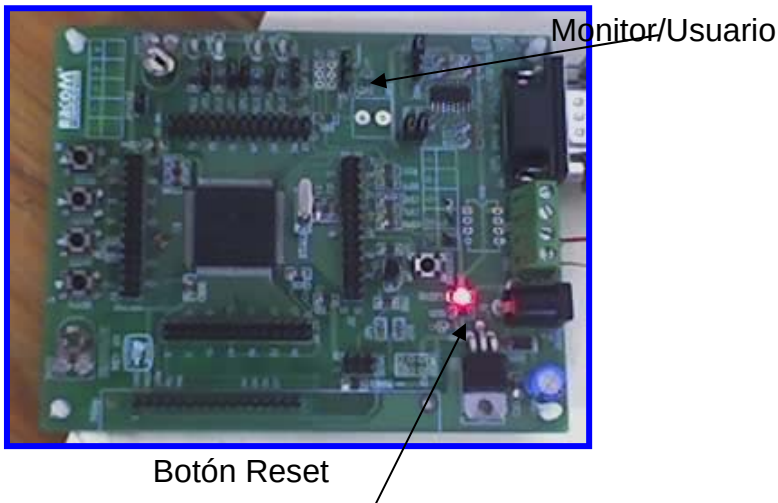


Figura 4.21 Sistema en modo Monitor/Usuario.

La figura 4.22 muestra el diagrama eléctrico de la manipulación de puerto U.

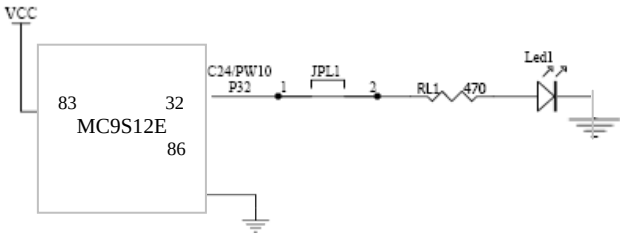


Figura 4.22 Diagrama eléctrico del manejo del puerto U.

EJERCICIO NO. 2

Realizar un programa que manipule los bits de salida del puerto U (4 bits).

Para poder manipular todas las salidas del puerto bit a bit tendríamos que dar de alta y manejar todas las salidas. Dar de alta cuatro BitIO . En la Figura 4.23 muestra cuatro beans configurados como salida (led).

Figura 4.23 Imagen de cuatro Bits.

De esta forma serán configurados todos los bits (salida). En el siguiente menú (Figura 4.24) se observa la configuración de un bit como salida.

Figura 4.24 Ventana de configuración del Bean LED1 (ejercicio 2).

PUERTO U nos servirá como puerto de salida.

En la Fígura 4.24.1 se muestra el diagrama eléctrico de la manipulación del puerto U.

1.-PU0_/IOC24/PW10

2.-PU1_/IOC25/PW11

3.-PU2_/IOC26/PW12

4.-PU3_/IOC27/PW13

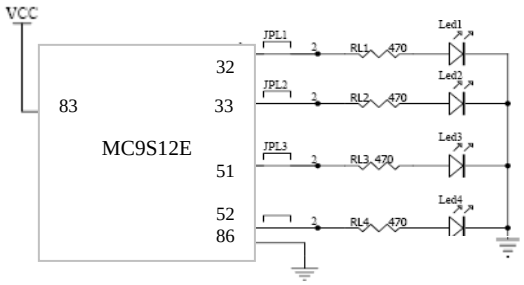


Figura 4.24.1 Diagrama eléctrico del puerto U.

Desarrollando el programa.-

Funciones utilizadas para generar el código del programa.-

- SetVal - Coloca un 1 en un bit de salida.

Código del programa.-

```
/* Including used modules for compilling procedure */
#include "Cpu.h"
#include "Led1.h"
#include "Boton1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /*Write your code here*/

       for(;;)

{
        LED1_SetVal()  ;
        LED2_SetVal()  ;
        LED3_SetVal()  ;
        LED4_SetVal()  ;
}
```

Figura 4.25 Código de programa (ejercicio 2).

En la Figura 4.25 mostró el código de programa para encender cuatro pines de salida (leds) del puerto U.

EJERCICIO NO.3

Realizar un programa que manipule los cuatro LEDs del puertos, los cuales tendrán que prender uno a la vez (primero el uno, después dos, etc.).

Para poder manejar los cuatro LEDs independientemente y cumplir con el objetivo señalado será necesario darle un retrazo a cada LED como se muestra.

Retrazo.-

```
LED1_SetVal()          ;
for(i=0;i <= 31000; i++)
   ;
LED1_ClrVal();
for(i=0;i <= 31000; i++)
     ;
```

La configuración de los Bean será la misma. Configurados como salida (Puerto U).

La Figura 4.26 muestra la ventana de configuración del bean LED2.

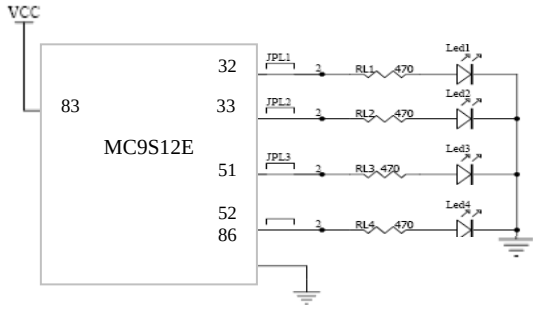


Figura 4.26 Diagrama eléctrico del puerto U.

Código del programa.- En la siguiente figura 4.27 se muestra el código del programa para encender y apagar cuatro leds.

```
/* Including used modules for compilling procedure */
#include "Cpu.h"
#include "Led1.h"
#include "Boton1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
  int i;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /*Write your code here*/

        for(;;)

{
   led1_SetVal();
  for(i=0;i <= 31000; i++);

  led1_ClrVal();
  for(i=0;i <= 31000; i++) ;

  led2_SetVal();
  for(i=0;i <= 31000; i++);

  led2_ClrVal();
  for(i=0;i <= 31000; i++) ;

  led3_SetVal();
  for(i=0;i <= 31000; i++);

  led3_ClrVal();
  for(i=0;i <= 31000; i++) ;

  led4_SetVal();
  for(i=0;i <= 31000; i++);
}
```

Figura 4.27 Código de programa (ejercicio 3).

EJERCICIO NO. 4

Realizar un programa el cual mande un valor específico al puerto U configurado como salida.

Para poder mandar un número específico al puerto de salida, es necesario configurar un Bean de 8 bits (byte) del puerto U, ya que los números serán representados en los Leds de forma de un número binario.

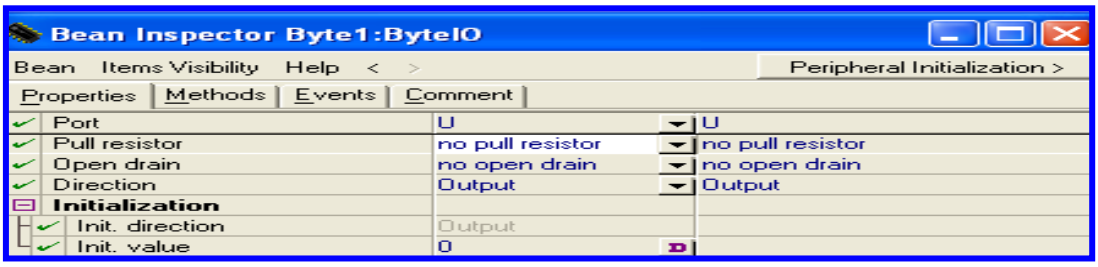


Figura 4.28 Configuración del Bean (ejercicio 4).

Desarrollo del programa.- Coloca un numero siete en el puerto U. El número podrá ser visualizado en el puerto U en forma binario.

Funciones utilizadas en la realización del código del programa:

PutVal **.-** Pone un valor específico en el puerto de salida.

Código del programa.-

```
#include "PE_Const.h"
#include "IO_Map.h"


void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /* Write your code here */

   for(;;)
{

Byte1_PutVal(7);




  }
```

Figura 4.29 Código de programa (ejercicio 4).

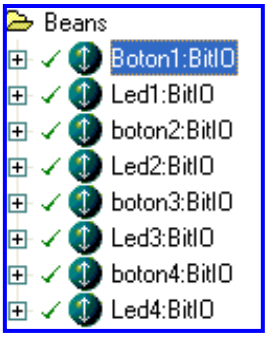
La Figura 4.29 mostró el código del programa para enviar un siete al puerto U representado en forma binaria.

Como anteriormente se menciono para que el programa sea observado en el microcontrolador se realizaran los pasos 22 al 24.

EJERCICIO NO. 5

CONFIGURACIÓN DE LOS PUERTOS DE ENTRADA-SALIDA

Realizar un programa en el cual un botón configurado como salida manipule un led de salida.



En la figura 4.30 se muestran los **bean** que se usarán para la manipulación del puerto de salida. Cada botón manipulará el encendido y apagado de cada led del puerto.

Figura 4.30 Bean como Botón (puerto de entrada).

En este caso tendremos que dar de alta cuatro Bean Bit como entrada y cuatro como salida. (8 Bean como se muestra en la Figura 4.30)

Configuración de los botones como entradas. En el siguiente menú (Figura 4.31) podemos observar la configuración del bean como botón,

Figura 4.31 Configuración de un Bit como un puerto de entrada (Boton1).

Puerto de entrada.- Para los botones (entradas) utilizaremos el puerto PAD. En la siguiente Figura 4.31.1 se muestra el diagrama eléctrico del manejo del puerto PAD.

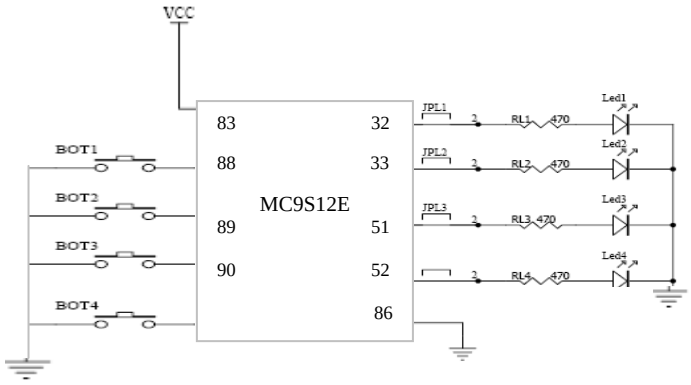


Figura 4.31.1 Diagrama eléctrico del manejo del puerto PAD.

La configuración de los Leds como salida será la misma. Utilizando el *Puerto U*.

Funciones utilizadas.

PutVal **.-** Pone un valor en el puerto de salida.

GetVal.-Retoma un valor de entrada-salida. Si el valor es de entrada lee el valor y lo regresa.

Código del programa.- En la Figura 4.32 observaremos el código generado para enviar 1 al puerto U manipulados por los botones.

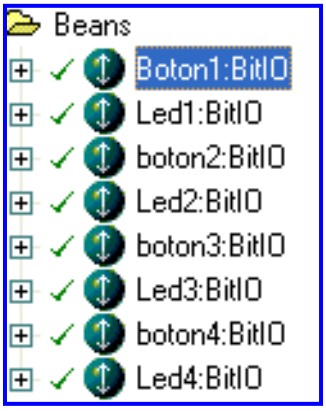
```
/* Include shared modules, which are used for wh
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
int i;
void main(void)

{
  /*** Processor Expert internal initialization.
  PE_low_level_init();
  /*** End of Processor Expert internal initiali

  for(;;){
    Led1_PutVal(Boton1_GetVal());
    Led2_PutVal(boton2_GetVal());
    Led3_PutVal(!boton3_GetVal());
    Led4_PutVal(!boton4_GetVal());
  }
```

Figura 4.32 Código de programa (ejercicio 5).

Los dos primeros puertos de salida (LEDS) aparecerán encendidos y podrán ser apagados al presionar los dos primeros botones; mientras que los dos últimos aparecerán apagados y podrán ser encendidos con los otros dos botones dados de alta.

EJERCICIO NO. 6

Realizar un programa en el cual un botón configurado como salida mande a una rutina, en este caso la rutina tendrá que ser manipulando el puerto U.



En la Figura 4.33 se muestran los **bean** que se utilizarán para el desarrollo del programa. Cada botón configurado como entrada tendrá que mandar a diferente rutina.
En la Figura 4.34 se muestra la configuración del bean como Botón 1.

Figura 4.33 Cuatro Bean configurados como entrada y cuatro como salida.

Configuración de los botones como entradas.-

La Figura 4.34 muestra la ventana de configuración del Boton1.

Figura 4.34 Configuración de Boton1.

La Figura 4.35 muestra el diagrama eléctrico de manejo de los recursos del microcontrolador necesarios para realizar esta práctica.

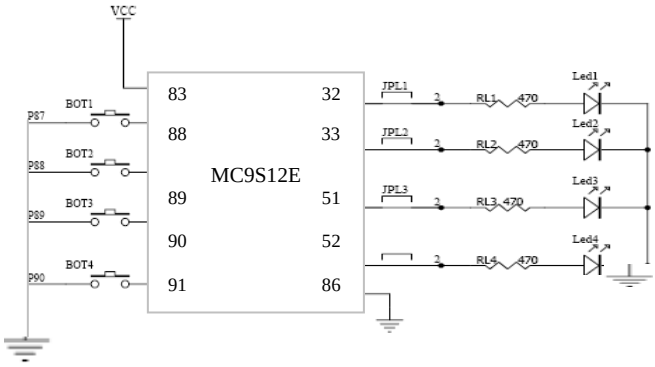


Figura 4.35 Diagrama eléctrico del manejo de los leds por medio de botones.

Desarrollo del programa.-

Funciones utilizadas:

PutVal **.-** Pone un valor específico en el puerto de salida.

GetVal.-Retoma un valor de entrada-salida. Si el valor es de entrada lee el valor y lo regresa.

Código del programa.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "led1.h"
#include "led2.h"
#include "led3.h"
#include "led4.h"
#include "boton1.h"
#include "boton2.h"
#include "boton3.h"
#include "boton4.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
  int i,a,b,c,d;

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

for(;;) {

 a= boton1_GetVal();
 b= boton2_GetVal();
 c= boton3_GetVal();
 d= boton4_GetVal();


if(a==0) {

rut1:
 b= boton2_GetVal();
 c= boton3_GetVal();
```

```
        d= boton4_GetVal();

        led1_SetVal();
        for(i=0;i <= 31000; i++);
        led1_ClrVal();
        for(i=0;i <= 31000; i++) ;
        led2_SetVal();
        for(i=0;i <= 31000; i++);
       led2_ClrVal();
        for(i=0;i <= 31000; i++) ;
        led3_SetVal();
        for(i=0;i <= 31000; i++);
       led3_ClrVal();
       for(i=0;i <= 31000; i++) ;

        led4_SetVal();
        for(i=0;i <= 31000; i++);
        led4_ClrVal();
        for(i=0;i <= 31000; i++) ;

        if(b==0) goto rut2;
        if(c==0) goto rut3;
        if(d==0) goto rut4;
          goto rut1;

       }


       if(b==0) {

rut2:
        a= boton1_GetVal();
        c= boton3_GetVal();
        d= boton4_GetVal();

         led4_SetVal();
         for(i=0;i <= 31000; i++);
         led4_ClrVal();
         for(i=0;i <= 31000; i++) ;
         led3_SetVal();
         for(i=0;i <= 31000; i++);
         led3_ClrVal();
         for(i=0;i <= 31000; i++) ;
         led2_SetVal();
         for(i=0;i <= 31000; i++);
         led2_ClrVal();
         for(i=0;i <= 31000; i++) ;
         led1_SetVal();
         for(i=0;i <= 31000; i++);
        led1_ClrVal();
         for(i=0;i <= 31000; i++) ;

       if(a==0) goto rut1;
       if(c==0) goto rut3;
       if(d==0) goto rut4;

       goto rut2;

       }


          if(c==0) {
 rut3:

       a= boton1_GetVal();
       b= boton2_GetVal();
       d= boton4_GetVal();

         led4_SetVal();
         for(i=0;i <= 31000; i++);
         led4_ClrVal();
        for(i=0;i <= 31000; i++) ;
         led1_SetVal();
         for(i=0;i <= 31000; i++);
```

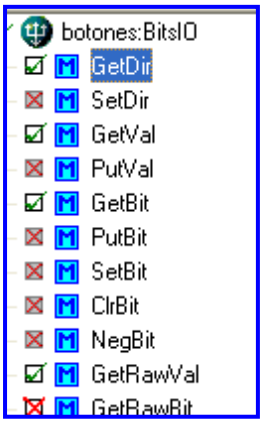
```
        led1_ClrVal();
        for(i=0;i <= 31000; i++) ;
        led3_SetVal();
        for(i=0;i <= 31000; i++);
        led3_ClrVal();
       for(i=0;i <= 31000; i++) ;
        led2_SetVal();
        for(i=0;i <= 31000; i++);
        led2_ClrVal();
        for(i=0;i <= 31000; i++) ;

      if(b==0) goto rut2;
         if(a==0) goto rut1;
         if(d==0) goto rut4;

          goto rut3;

         }


if(d==0) {

          rut4:
       a= boton1_GetVal();
       b= boton2_GetVal();
       c= boton3_GetVal();

        led2_SetVal();
        for(i=0;i <= 31000; i++);
        led2_ClrVal();
        for(i=0;i <= 31000; i++) ;
        led3_SetVal();
        for(i=0;i <= 31000; i++);
        led3_ClrVal();
       for(i=0;i <= 31000; i++) ;
        led1_SetVal();
        for(i=0;i <= 31000; i++);
        led1_ClrVal();
        for(i=0;i <= 31000; i++) ;
        led4_SetVal();
        for(i=0;i <= 31000; i++);
        led4_ClrVal();
        for(i=0;i <= 31000; i++) ;

        if(b==0) goto rut2;
        if(c==0) goto rut3;
        if(a==0) goto rut1;
         goto rut4;

       }

        }

      /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
       for(;;){}
       /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
```

EJERCICIO.7

MANIPULACIÓN DE TRES BITS ENTRADA-SALIDA

Realizar un programa que manipule tres bits de salida del puerto U.

Para el manejo de estos 3 bits de salida utilizaremos el siguiente bean de (1-8) bits.



El Bean botones:BitsIO es un Multi-bits de I/O.(1-8bits)

Este bean (Figura 4.36) también lo encontraremos como un bean del Port I/O. Permitirá la manipulación de tres bits a la vez.

En esta ocasión lo utilizaremos para el manejo del puerto U.

Figura 4.36 Funciones del Bean de 3 bits.

La configuración del bean como salida se puede observar en la Figura 4.36.1.

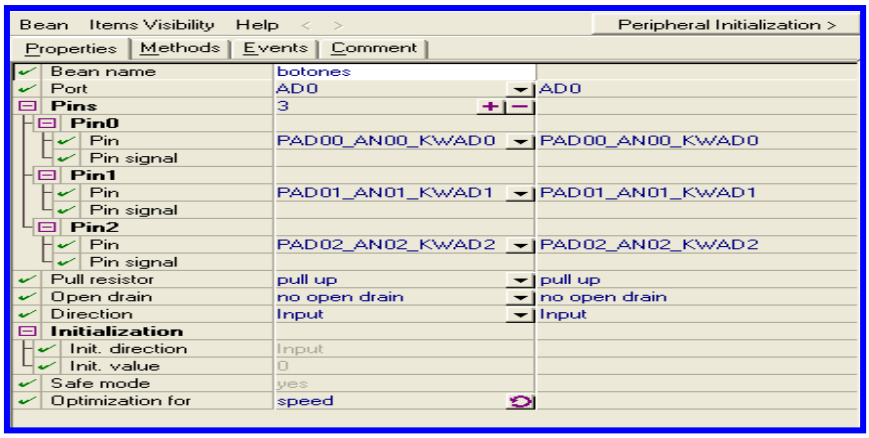


Figura 4.36.1 Configuración del Bean de tres bits como salida.

Este bean utilizará hasta 8 pin (bits) de salida dependiendo el número de ellos que se deseen utilizar. En este caso solo se requiere la manipulación de 3 de ellos.

En la Figura 4.37 muestra el código generado para enviar un 3 al puerto U.

Código del programa.-

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Bits1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! *
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    *

  /* Write your code here */

  for(;;){


  Bits1_PutVal(3);

  }



  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Figura 4.37 Código de programa para enviar un valor utilizando un bean de 3 bits.

La Figura 4.37.1 muestra el diagrama eléctrico de la manipulación de 3 pines del puerto U para enviar un 3 en valor binario.

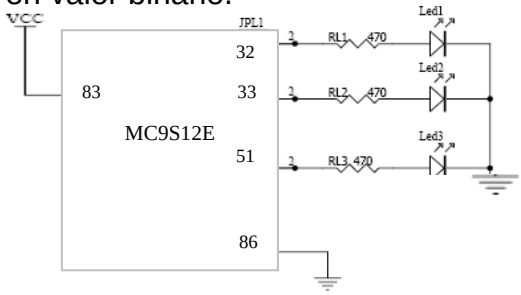


Figura 4.37.1 Diagrama eléctrico para el envió de un valor al puerto U.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 2

NOMBRE: Manejo del *Timer – PWM* de sistema de desarrollo SIS9S12E.

OBJETIVO: Que el usuario aprenda a manejar y a configurar el *Timer-PWM* del microcontrolador.

EJERCICIO NO. 1

Realizar un programa que permita la manipulación del PWM con el objeto de manejar paulatinamente el encendido y apagado de un Led, a través de dos botones.

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v de C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana Bean Selector (Figura 4.38).- *CPU Internal Peripherals-Timer-PWM.*

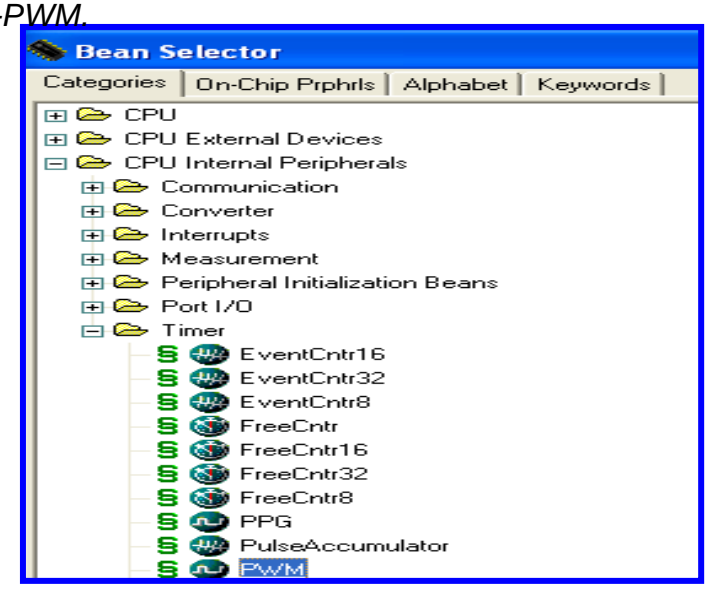


Figura 4.38 Bean Selector (selección del PWM).

6.- Dar de alta los botones para la manipulación del *PWM*.- La configuración de los botos serán configurados como *Bean* de entrada.

Figura 4.39 Bean PWM.

7.- La configuración del *PWM* (Figura 4.40) se muestra a continuación.-

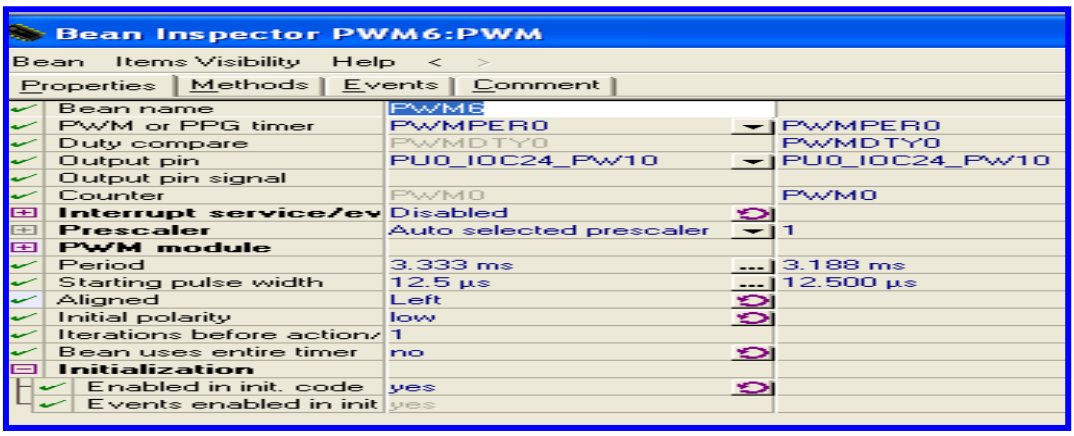


Figura 4.40 Configuración del PWM.

Al configurar el PWM de la forma anterior habilitamos un bit de salida (Led) que en este caso será el primer led del puerto U *(PU0_/IOC24/PW10).*

8**.-** Para la configuración del *Timer-PWM* un aspecto importante de considerar es la configuración de su *Period y Starting pulse width.*

En la siguiente ventana (Figura 4.41) se configura el Periodo del PWM.

Se configura en 3.333 ms para una frecuencia de 300Hz.

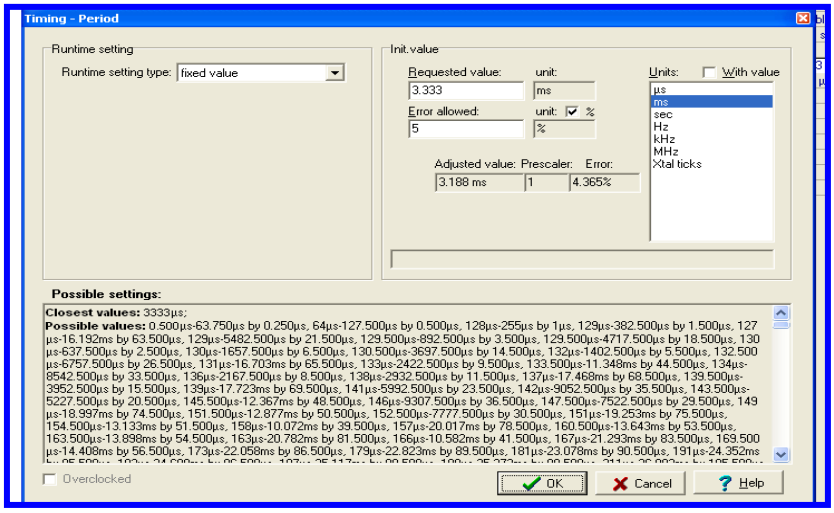


Figura 4.41 Configuración de Periodo.

En la siguiente Figura 4.41.1 se muestra el diagrama de los recursos utilizados en esta práctica.

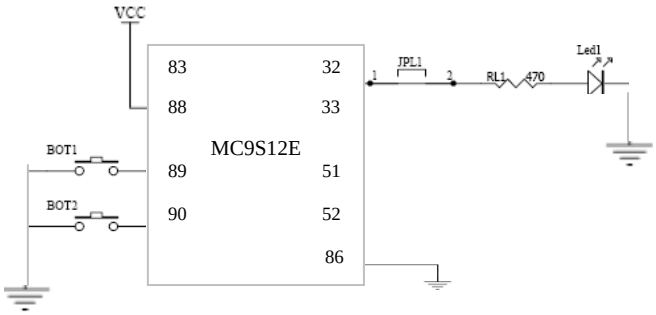


Figura 4.41.1 Diagrama eléctrico de práctica 2.

Starting pulse width.-

En la siguiente ventana (Figura 4.42) se realizará la configuración del Starting Pulse.

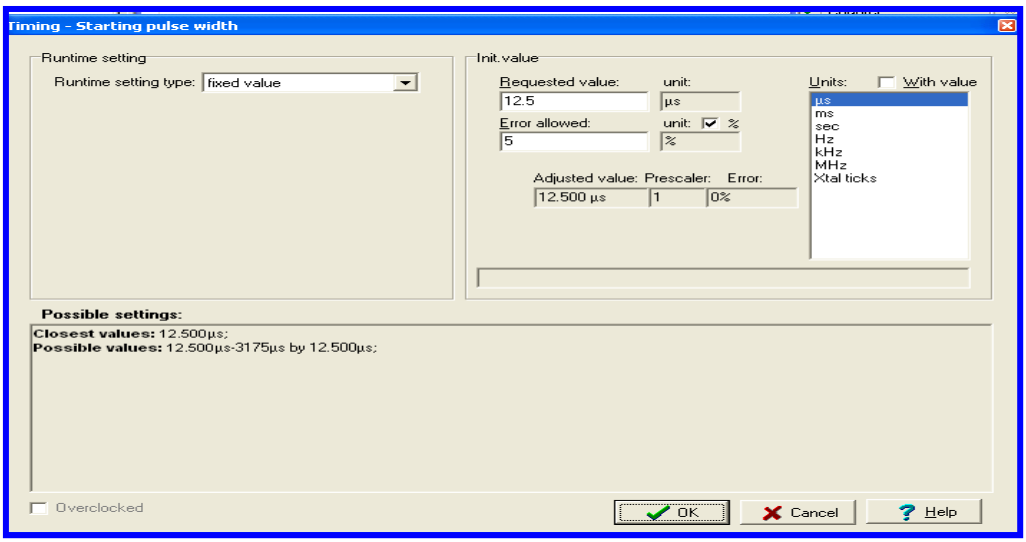


Figura 4.42 Configuración de starting pulse with.

9.- Dar de alta dos beans de un bit configurados como botones. Estos beans nos permitirá la manipulación del PWM.

La configuración de estos botones se realizará de la siguiente manera.

La siguiente Figura 4.42.1 muestra la configuración de los botones.

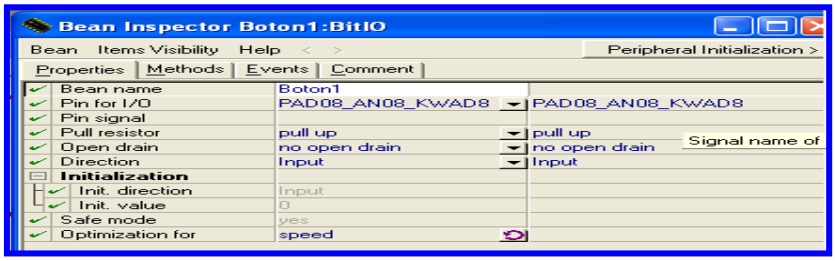


Figura 4.42.1 Configuración de botón 1.

La configuración del segundo botón utilizará *PAD09/AN09/KWAD09.*

10**.-** Realizar el desarrollo del programa.

Funciones utilizadas para realizar el código.-

GetVal.-Retoma un valor de entrada-salida. Si el valor es de entrada lee el valor y lo regresa.

SetRatio8.- Es una duración de radio. El radio esta expresado como numero asignado de 8 bits. Del 0 al FF proporcionalmente 0-100 %.

Código de programa.- La Figura 4.43 muestra el código generado para manipular el Timer-PWM.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "boton1.h"
#include "boton2.h"
#include "PWM6.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
int valor,i;

void main(void)
{

 valor= 0;
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
   for(;;){

   if(boton1_GetVal()==0)
   valor--;


   if(boton2_GetVal()==0)
   valor++;

   PWM6_SetRatio8(valor) ;

   for (i=0;i<10000;i++)
          ;
     |
   }
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Figura 4.43 Código de programa Timer-PWM.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BASICAS E INGENIERIA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 3

NOMBRE: Manejo del Convertidor Analógico- Digital (ADC) con el PWM**.**

OBJETIVO: Que el usuario aprenda a manejar y a configurar el ***Convertidor Analógico- Digital*** manipulado con una interrupción por Timer.

EJERCICIO NO. 1

Realizar un programa que permita convertir una señal analógica en una señal digital, tal conversión será representada en un Led del puerto U.

METODOLOGÍA:

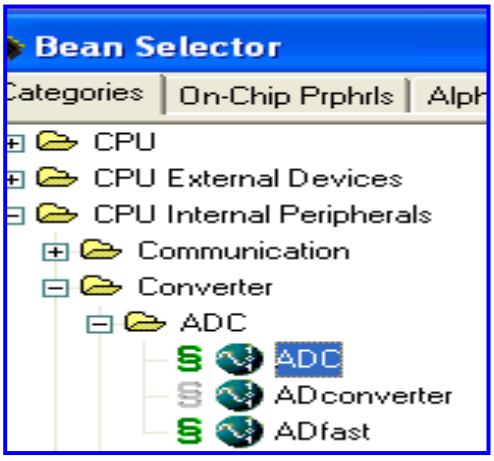
1.- Conectar el microcontrolador a una fuente regulada de 12 v de C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals- Converter-ADC.* Hacer docle clic en el Bean para dar de alta el ADC.

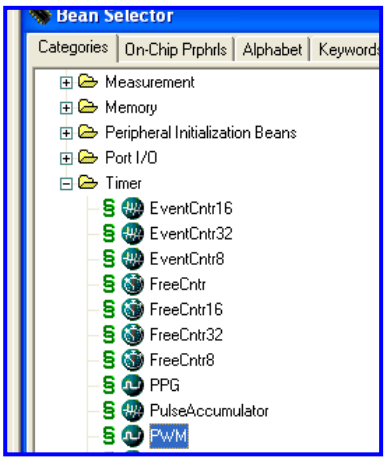


Al dar de alta este Bean (Figura 4.44) podremos realizar la conversión de una señal analógica a una señal digital, la cual podrá ser observada en los Leds (puerto U).

Figura 4.44 Bean ADC.

6**.-** Elegir en la ventana de Bean Selector- *CPU Internal Peripherals- Timer-PWM.*

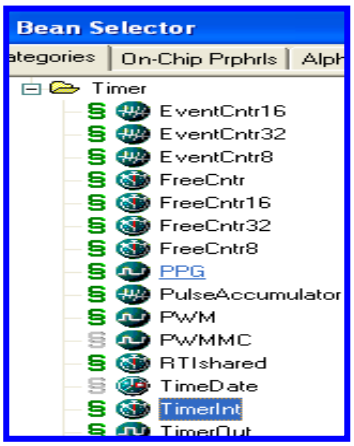
Hacer doble clic en el *Bean del PWM* para darlo de alta.



Al dar de alta el Bean del PWM (Figura 4.45) nos permitirá la manipulación del convertidor analógico-digital ADC.

Figura 4.45 Selección del Bean PWM.

7.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals- Timer- TimerInt.*



Este Bean (Figura 4.46) implementa un generador de interrupción periódica. Cuando el Bean y sus eventos son cargados, el evento **OnInterrupt** es llamado periódicamente con un periodo específico.

Figura 4.46 Selección del TimerInt.

8.- El Bean del convertidor  será configurado de la siguiente manera.-

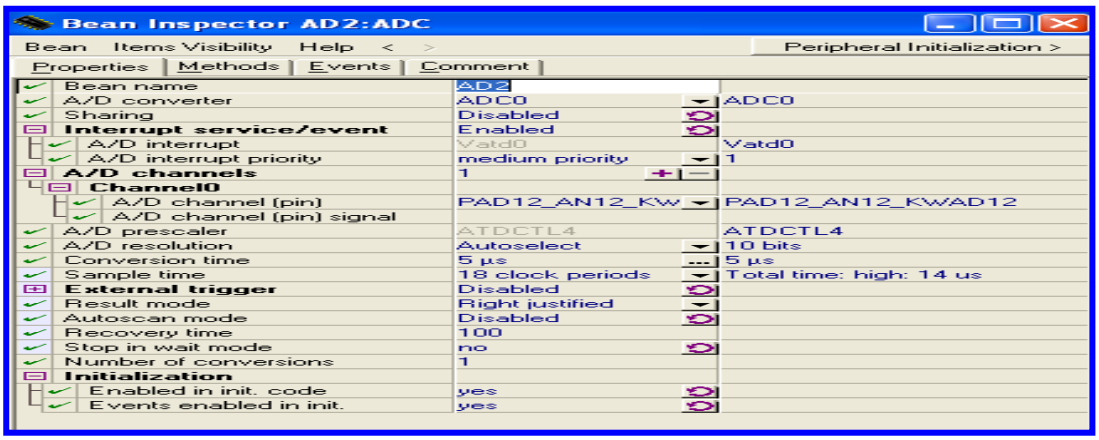
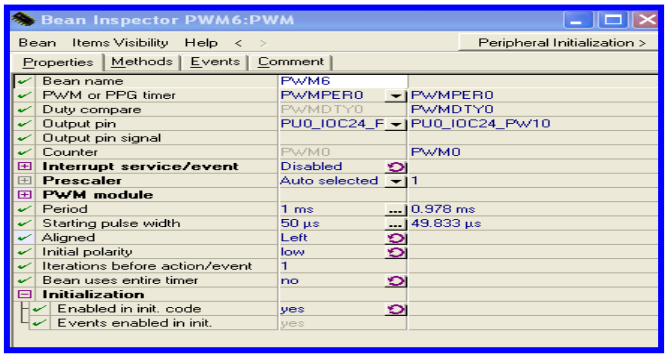


Figura 4.47 Configuración del Bean ADC.

En la Figura 4.47 se mostró la configuración del bean del convertidor analógico –digital.

Un aspecto importante que debemos considerar el configurar el convertidor analógico- digital es el canal que utilizará, en este caso el microcontrolador utiliza el pin PAD12_AN12_KW.

9.- El PWM se configura de la siguiente manera.-



Al configurar de esta forma el PWM se establece el pin de salida donde se mostrará la conversión de las señales. En este caso el pin de salida que se utilizará será PU0-IOC24_PW10. (Figura 4.48)

Figura 4.48 Configuración del PWM.

10.- El Bean de la interrupción TimerInt se configurará de la siguiente manera.-

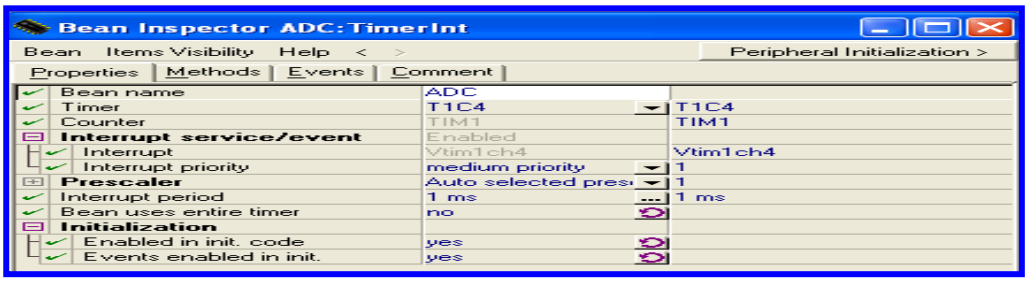


Figura 4.49 Configuración del ADC.

Para la interrupción se especificará el periodo de está.

La figura 4.49 nos mostró la configuración de la interrupción del bean TimerInt.

11.- Realizar el código para el convertidor analógico- digital.-

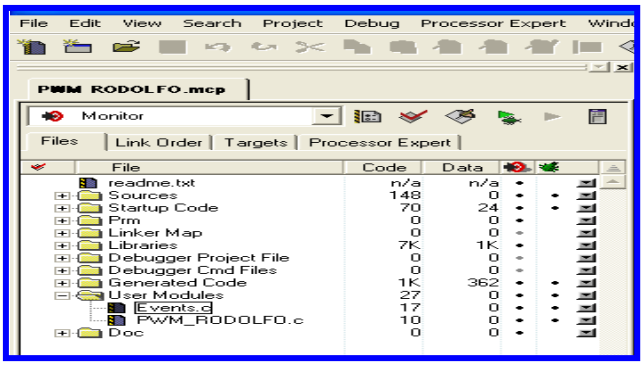


Figura 4.50 Selección de eventos para el código.

El código para manejar el convertidor se realizará en dos partes:
1.- La primera es en **Files-User Modules- Eventos**(solo sirve para configurar el convertidor- Figura 4.50).

2.- La segunda también se realizará de forma normal en **Files- User Modules- 'nombre.c'-** Figura 4.50)

12.- Se realiza el código del programa para la manipulación del ADC. *Files- User Modules- 'nombre.c'*

```
  /* Write your code here ... */
}

/*
** ===================================================================
**     Event       :  ADC_OnInterrupt (module Events)
**
**     From bean   :  ADC [TimerInt]
**     Description :
**         When a timer interrupt occurs this event is called (
**         when the bean is enabled - "Enable" and the events a
**         enabled - "EnableEvent").
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
void ADC_OnInterrupt(void)


{

  byte resultado;

  unsigned char valor_ad;

  resultado=AD2_GetValue8(&valor_ad);

  PWM6_SetRatio8(valor_ad) ;


   }


  /* Write your code here ... */
```

Figura 4.51 Código de programa del ADC.

13.- Inicializar el convertidor.- Se realiza una pequeña inicialización en en donde realizamos nuestro código normalmente. *Files- User Modules- events.*

La Figura 4.52 muestra el código con del que se inicializa el uso del *convertidor analógico-digital.*

```
/* MODULE PWM_RODOLFO */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "AD2.h"
#include "ADC.h"
#include "PWM6.h"
#include "TI2.h"
#include "PWM7.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***


AD2_Start();


  /* Write your code here */

  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END PWM_RODOLFO */
/*
** ###################################################################
**
**     This file was created by UNIS Processor Expert 2.95 [03.62]
**     for the Freescale HCS12 series of microcontrollers.
**
** ###################################################################
*/
```

Figura 4.52 Programa para configuración del ADC.

14.- Para ser observada la conversión de la señal analógica a la señal digital en el microcontrolador es necesario el uso del potenciómetro POT1, esto para realizar la variación de la señal analógica. Como se muestra en la siguiente imagen (Figura 4.53). El canal donde se encuentra el potenciómetro fue configurado en el programa.

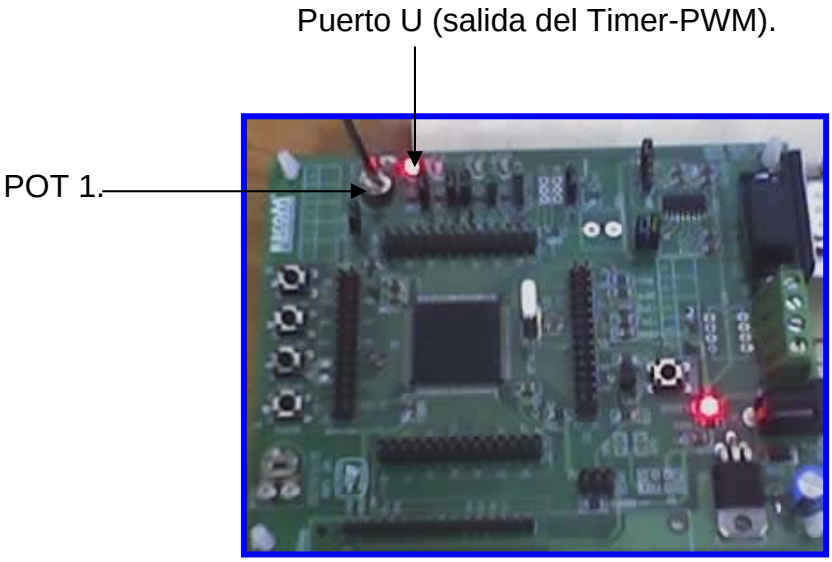


Figura 4.53 Conversión de una señal analógica a digital manipulada por PWM, observada en el puerto U.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 4.-

NOMBRE: Manejo del convertidor analógico-digital (DAC) utilizando el puerto serie de comunicación.

OBJETIVO: Que el usuario aprenda a manejar y a configurar el puerto serie, para ser utilizado como transmisor de la conversión de una señal analógica a una señal digital por la interface de comunicación RS232.

EJERCICIO NO. 1

Realizar un programa que convierta una señal analógica a una digital, la cual será visualizada por medio de un programa Visual de microcontroladores Niplesoft por el interface de comunicación RS232, esta señal digital podrá ser manipulada por un potenciómetro del microcontrolador.

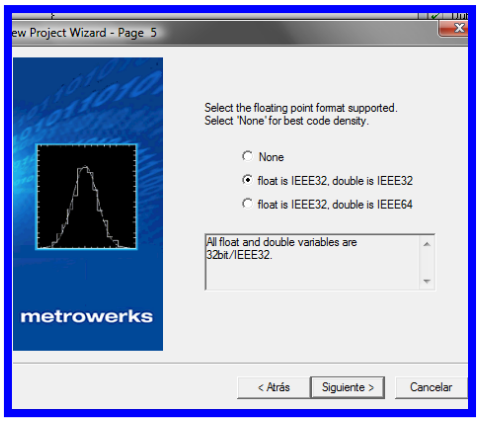
METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v de C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- En esta ocasión al momento de configurar o cargar el programa para utilizar el tipo de variable seleccionaremos la siguiente.- (utilizaremos una variable flotante de 32 bits). En la Figura 4.54 podremos observar la ventana para seleccionar el tipo de variable que utilizaremos.

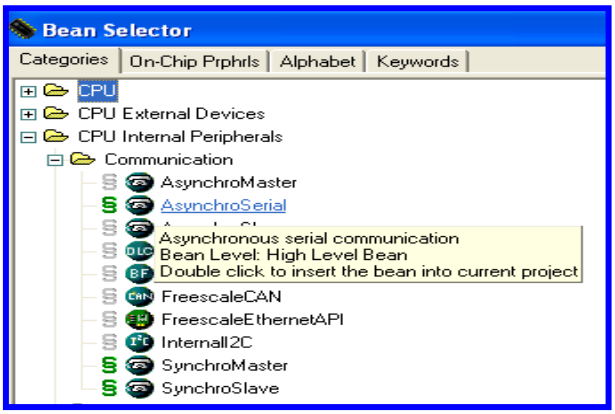


Es importante seleccionar el tipo de variable que vamos a utilizar ya que nos permitirá utilizar diversas variables al momento de realizar nuestro código de programación.

Figura 4.54 Selección de tipo de variable flotante.

5.- Crear un nuevo archivo.

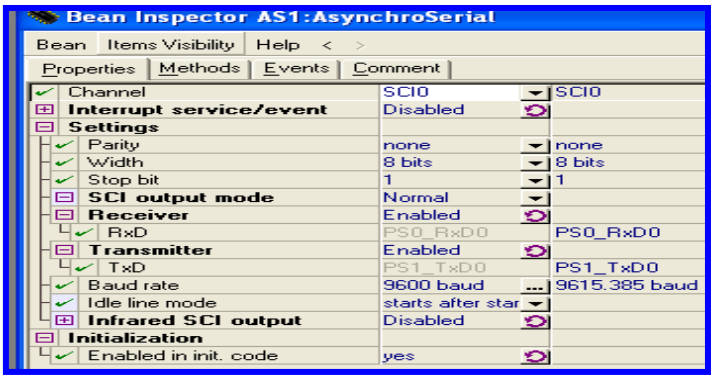
6.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals- Conmunication- AsynchroSerial.*



Este Bean (Figura 4.55) nos permitirá la comunicación por la interface
RS232.

Figura 4.55 Bean AsynchroSerial.

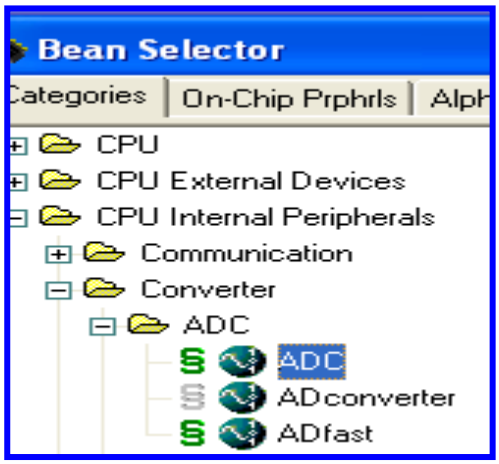
7.- La configuración del Bean AsynchroSerial se realizará de la siguiente manera (Figura 4.55).-



En esta parte tendremos la opción de elegir el canal por donde podremos ver la salida de la conversión de la señal. Si elegimos el canal SCIO utilizaremos el interface RS232 y si utilizamos el canal SCI1 podremos ver la señal conectando un osciloscopio.

Figura 5.56 Configuración del Bean AsynchroSerial.

8.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converter-ADC.* Hacer doble clic en el Bean para dar de alta el ADC.

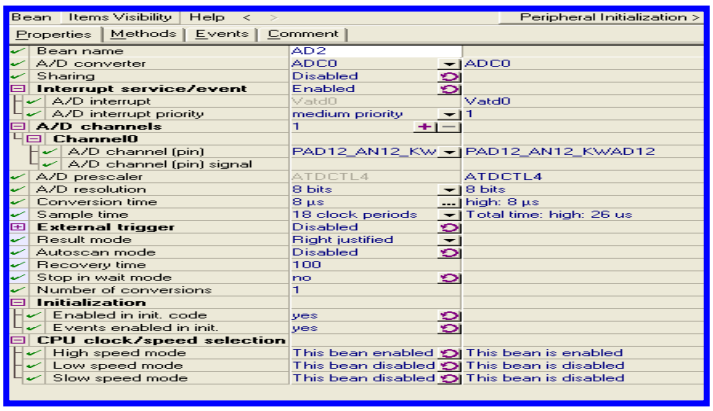


Al dar de alta este Bean podremos realizar la conversión de una señal analógica a una señal digital, la cual podrá ser observada en el programa visual de microcontroladores Niplesoft utilizando la interface RS232 o con un osciloscopio en el microcontrolador.

Figura 4.57 Bean ADC.

En la Figura 4.57 mostró el bean del convertidor analógico-digital el cual será utilizado para convertir la señal analógica en digital para poder ser enviada por la interface RS232.

9.- El Bean del convertidor analógico-digital se configurará de la siguiente manera. La Figura 4.58 muestra la configuración del bean.



Un aspecto importante a considerar para la configuración de este, es el uso que se hará para la comunicación serial (8 bits). También considerar el tiempo de conversión.

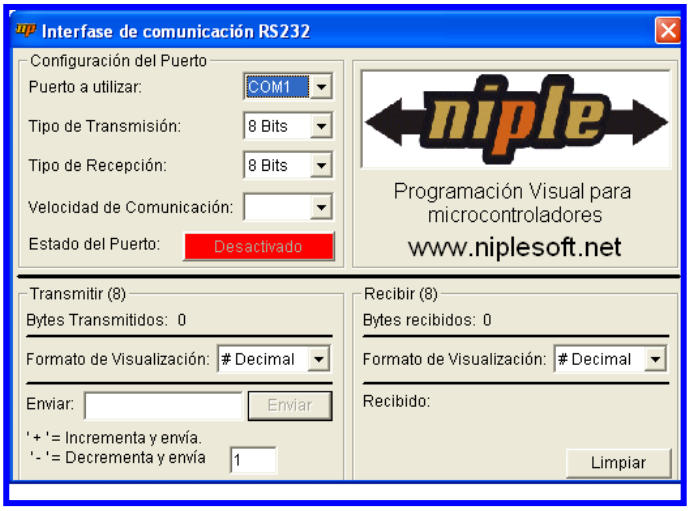
Figura 4.58 Configuración del ADC.

10.- Realizar el código del programa. La Figura 4.58 muestra el código de programa para enviar bits por medio de la interface RS232.

```
/* MODULE tesis_3 */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "Events.h"
#include "AD2.h"
#include "AS1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

byte valor;


void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

  AD2_Start();
   for(;;) {

   AD2_GetValue8(&valor);


AS1_SendChar(valor);
   }



  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

Figura 5.59 Código de Programa de Comunicación Serie.

10.- La conversión de la señal se podrá observar en Niplesoft.- *Hacer clic en Inicio- Todos los programas- Interface de Comunicación RS232.*



De esta forma configuraremos este programa para poder realizar la comunicación con el microcontrolador.
Al variar el Potenciómetro del convertidor analógico- digital del microcontrolador, en la ventana del programa de Niplesoft (Figura 4.60) veremos la variación de los bits, los cuales serán recibidos en el programa (Niplesoft)

Figura 5.60 Ventana del programa Niplesoft.

Niplesofs es un programa que permite el monitoreo de microcontroladores el cual tiene como características principales:

- Control automático de la estructura lógica del programa.
- Facilidad en la interpretación del programa.
- Control automático de cambios de banco de registros y cambios de página de programa.
- Asignación de valores a bits y a registros de 8 y 16 bits.
- Manejo de memorias EEPROM y RAM I2C (a 8 y 16 bits).
- Manejo de Reloj de tiempo Real por I2C.
- Manejo de conversores D/A por I2C.
- Comunicación Serial RS232 y redes RS485.
- Conversión A/D a 8 y 10 bits. Múltiples mediciones A/D con cálculo automático de promedio.

Para descargar este programa ingresar a la página Web www.Niplesoft.net

11.- Para la manipulación del envío de bits de 0-255 por el puerto serie se utilizara el potenciómetro PAD12_AN12_KW12AD (Figura 4.61).

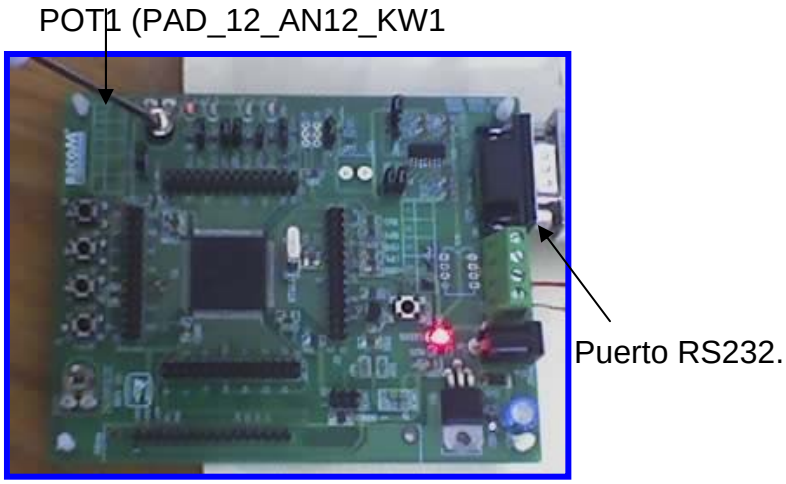


Figura 4.61 Manipulación de envío de bits por medio del potenciómetro.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 5

NOMBRE: Convertidor Digital – Analógico (DAC).

OBJETIVO: Que el usuario aprenda a utilizar el convertidor digital-analógico.

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v de C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- **CPU** *Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor.

Este Bean (Figura 4.62) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.62 Bean DAC.

6.- La configuración del Bean se realizará de la siguiente manera.-

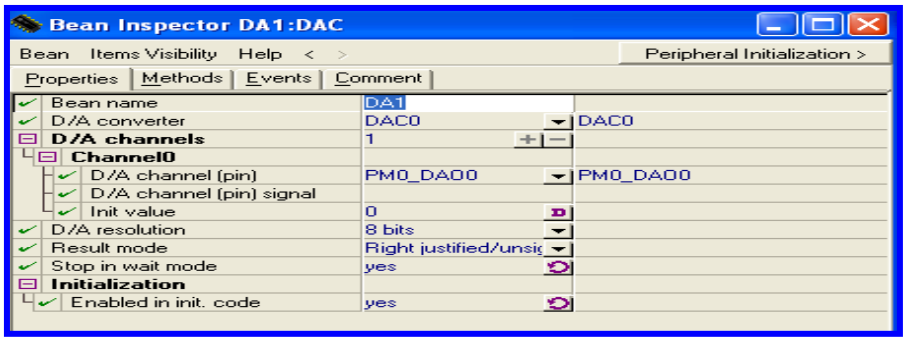


Figura 4.63 Configuración del DAC.

En la Figura 4.63 se mostró la configuración del DAC dado de alta. Además en esta parte podremos observar el número de canales, el canal específicamente de salida, así también la resolución que se utilizará que en este caso será de 8 bits.

Pines del DAC del microcontrolador MC9s12E.

Los pines (Figura 4.64) que se conectaran en el microcontrolador para ser observados en el osciloscopio serán: el pin 95 (que es el pin del DAC) y el pin 101 que es tierra.

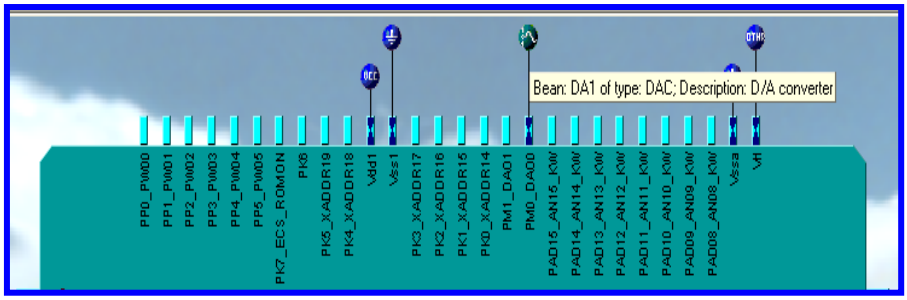


Figura 4.64 Mapa de pines del DAC (95 y 101).

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este con figuración se puede realizar en la ventana siguiente (Figura 4.65) en el icono Methods.

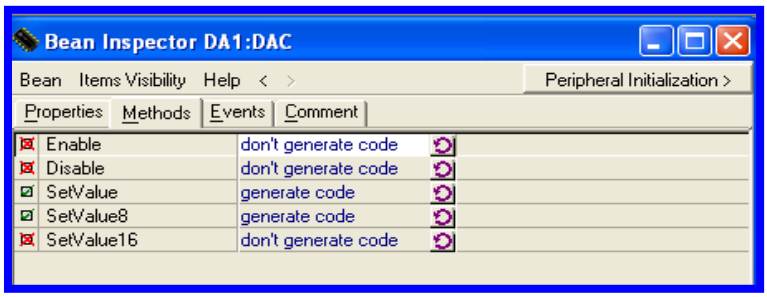


Figura 4.65 Habilitación de funciones del DAC.

8.- Realizar el código del programa (Figura 4.66).-

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "DA1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
unsigned char valor;

void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  valor = 255;
  for(;;){

  DA1_SetValue8(&valor);
          valor--;



  }
```

Figura 4.66 Código de programa DAC.

9.-Conexiones del Microprocesador.-

En la Figura 4.67 se muestran los pines de salida que utiliza el DAC en el sistema de desarrollo sis9s12E.

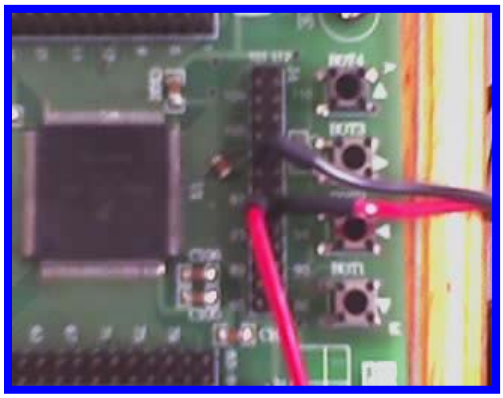


Figura 4.67 Pines de Salida del DAC.

10.- Señal generada por medio del Convertidor Digital-Analógico (DAC) observada por medio de un osciloscopio (Figura 4.68).

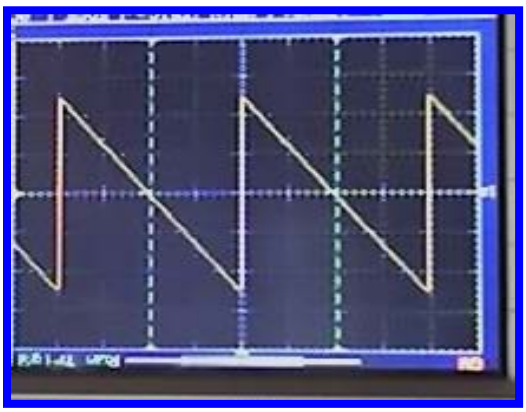


Figura 4.68 Señal Generada por el DAC.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 6

NOMBRE: Generador de Funciones: Dientes de Sierra.

OBJETIVO: Que el usuario realice un código el cual genere una función de diente de sierra, la cual será enviada al DAC y observada en su puerto de salida(en este caso utilizaremos el puerto de salida del DAC).

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 voltios de corriente continua.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor.

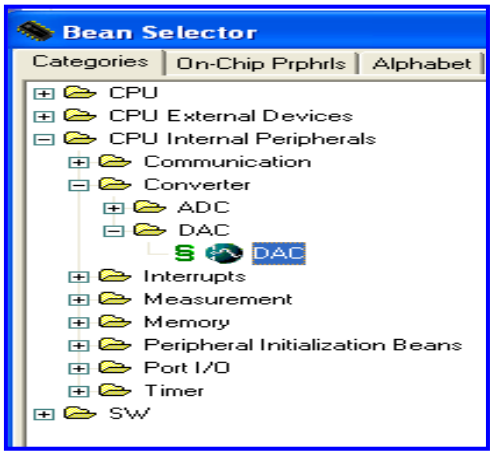


Figura 4.69 Bean DAC (practica 6).

6.- La configuración del Bean se realizará de la siguiente manera (Figura 4.70).-

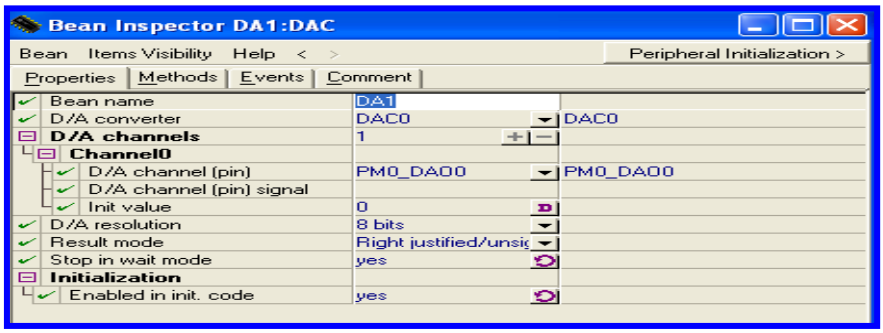


Figura 4.70 Ventana de configuración del DAC.

En esta parte podremos observar el número de canales, el canal específicamente de salida, así también la resolución que se utilizará que en este caso será de 8 bits.

La configuración de la ventana anterior permitirá generar una señal analógica, el número y canal utilizado es importante al momento de visualizar la señal obtenida en este caso su utilizará el canal PM0_DAC0.

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. En la Figura 4.71 muestra las funciones que pueden ser dadas de alta.

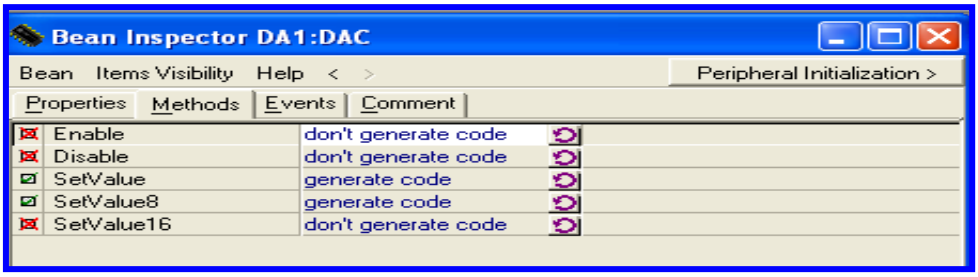


Figura 4.71 Habilitación de funciones de DAC.

8.- Realizar el código.-

En este código asignaremos una variable de valor absoluto para que solo pueda tomar valores positivos. En la Figura 4.72 muestra el código para generar una señal dientes de sierra.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "DA1.h"
#include "Byte1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
 unsigned char valor;
void main(void)

{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /* Write your code here */
      valor=0;
  for(;;)
  {


    DA1_SetValue8(&valor);

              valor++;


  }


  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE
```

Figura 4.72 Código de Programa para generar una señal dientes de sierra.

9.- Para poder observar la señal generada por el programa desarrollado será necesario el uso de osciloscopio. El cual sus puntas están conectadas a la salida del PM0_DAC0 (pin 95 y 101) como se muestra en la Figura 4.73.

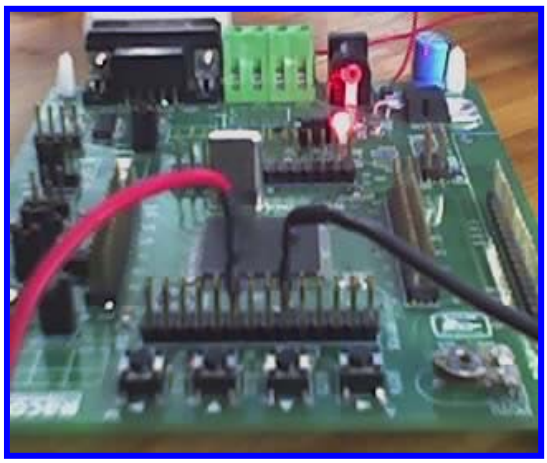


Figura 4.73 Salida del convertidor Digital- Analógico.

Señal generada de Dientes de Sierra (Figura 4.74). Es observada por medio del uso del osciloscopio.

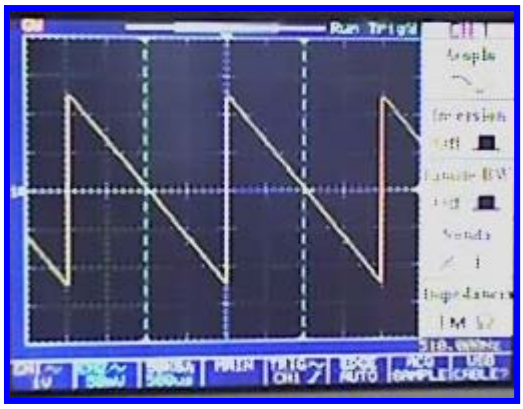


Figura 4.74 Señal de Dientes de Sierra.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 7

NOMBRE: Generador de Funciones (Señal triangular).

OBJETIVO: Que el usuario realice un código el cual genere una función triangular, la cual será enviada al DAC y observada en su puerto de salida (en este caso utilizaremos el puerto de salida del DAC).

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 voltios de corriente continua.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC (Figura 4.75) para dar de alta al convertidor.

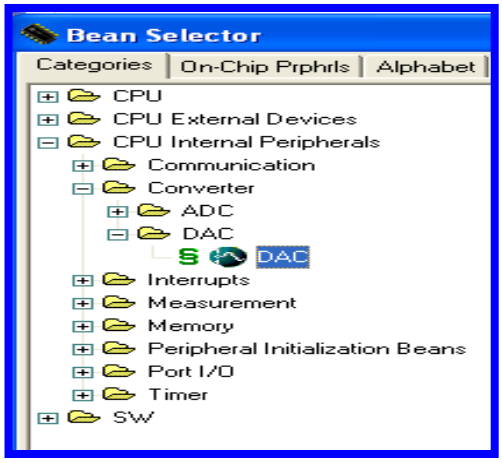


Figura 4.75 Bean DAC (Señal triangular)

6.- La configuración del Bean se realizará como se muestra en la Figura 4.76.

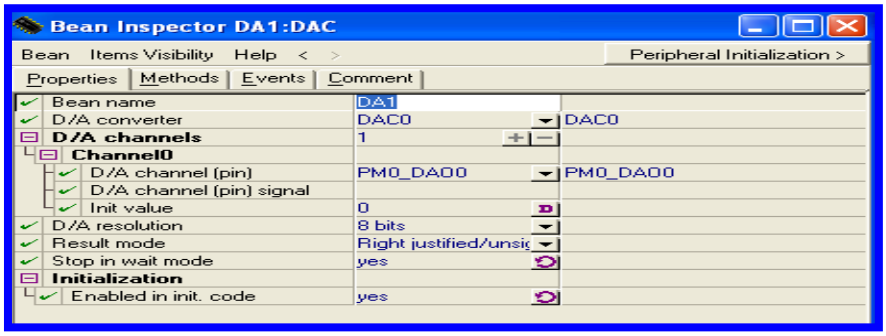
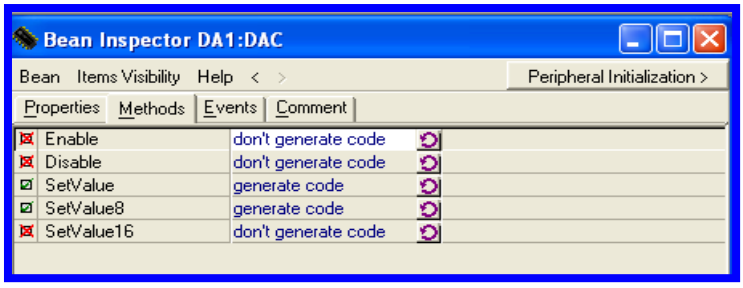


Figura 4.76 Configuración del Bean DAC (Señal triangular).

En la Figura 4.76 podremos ver la configuración que debe tener el bean DAC para poder generar una función triangular.

Además en esta parte podremos observar el número de canales, el canal específicamente de salida, así también la resolución que se utilizará que en este caso será de 8 bits.

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este con figuración se puede realizar en la ventana anterior en el botón Methods.



4.77 Habilitaciones de funciones del DAC (Señal Triangular).

8.- Realizar el código del programa. Este podrá ser observado en la Figura 4.78.

```
#include "Cpu.h"
#include "DA1.h"
/* Include shared modules, which are used for whole p
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
 unsigned char valor;
 unsigned char a;
 int i;
void main(void)
{
  /*** Processor Expert internal initialization. DON'
  PE_low_level_init();
  /*** End of Processor Expert internal initializatic

  /* Write your code here */

 for(;;)
  {
  for(valor=0;valor<255;valor++){

  DA1_SetValue8(&valor) ;

      }


  for(valor=255;valor>0;valor--) {

    DA1_SetValue8(&valor);
        }


      }
```

Figura 4.78 Código de Programa para generar una señal Triangular.

8.- La señal triangular generada por medio de la programación de esta práctica podrá ser observada por medio del osciloscopio, esta señal generada se podrá observar en la Figura 4.79:

Figura 4.79 Señal Triangular.

En el osciloscopio podrá ser observada el periodo, frecuencia, amplitud, etc. de la señal generada en la práctica. Todas estas características de la señal podrán ser modificadas al alterar el código del programa y las características del bean (DAC).

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 8

NOMBRE: Generador de Funciones (Señal seno).

OBJETIVO: Que el usuario realice un código el cual genere una función seno, la cual será enviada al DAC y observada en su puerto de salida (en este caso los puertos de salida que utilizaremos serán el puerto U y el puerto de salida del DAC).

METODOLOGÍA:

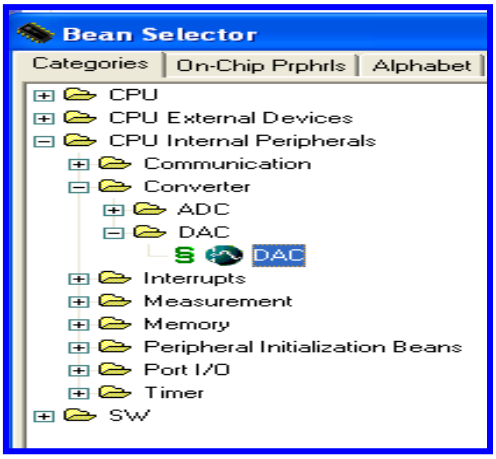
1.- Conectar el microcontrolador a una fuente regulada de 12 v de CD.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor.



Este Bean DAC (Figura 4.80) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.80 Bean DAC (seno).

6.- La configuración del Bean se realizará de la siguiente manera.-

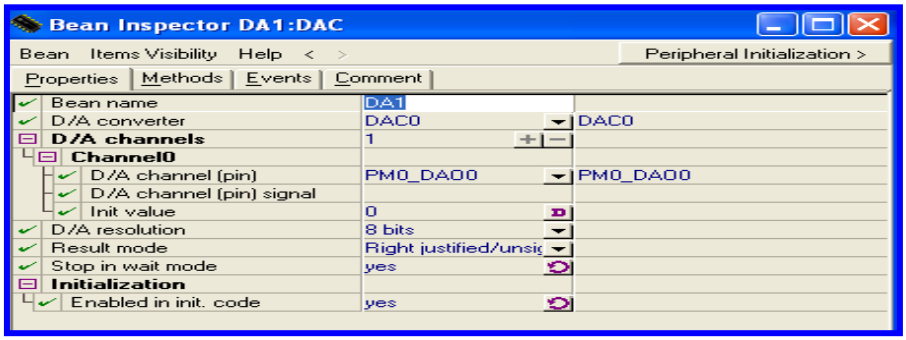


Figura 4.81 Configuración del DAC (seno).

La Figura 4.81 mostró la configuración del Convertidor digital-analógico utilizado para generar una señal Seno. En la configuración dada podemos observar características importantes que se deben tomar en cuenta, por ejemplo el DAC que se utilizará (DAC0 o DAC1), el puerto de salida del DAC, la resolución del convertidor, etc.

7.- Mapa de bits para la configuración y conexión del DAC.

En la Figura 4.82 podremos observar el número de pin que maneja el Convertidor Digital- Analógico, en el mapa podremos observar que hay dos pines de DACs con diferentes pines: pin No. 95 y pin No. 96.

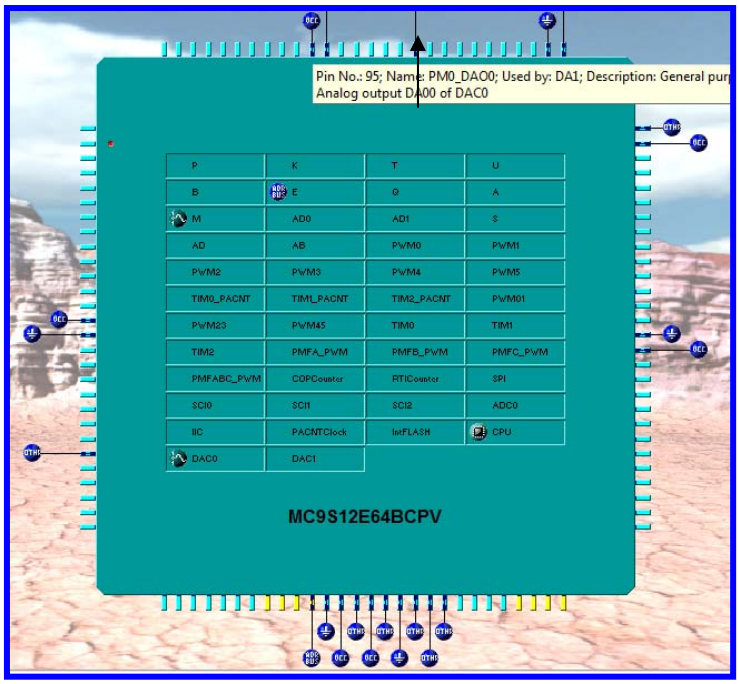


Figura 4.82 Mapa de pines de MC9S12E64 (seno).

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este con figuración se puede realizar en la ventana anterior en el botón Methods.

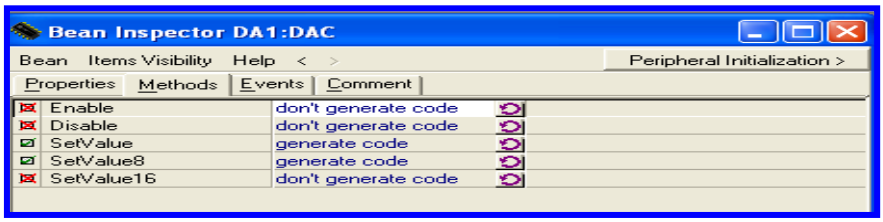


Figura 4.83 Habilitación de Funciones del DAC (Seno).

8.- Realizar el código que permita generar una señal seno utilizando el convertidor Digital-Analógico. En la siguiente imagen (Figura 4.84) se muestra generado para obtener una señal seno.

```
**     Mail       : Info@ProcessorExpert.com
** ###################################################################*/
/* MODULE coseno2 */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "DA1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include  <math.h>
int i;
unsigned char m;
float x,n;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

  for(;;)

  {
    x=0 ;
   for(i=0;i<255;i++){
     n = sin(x);

    m=((n+1)/2)*255;



    DA1_SetValue8(&m);

    x=x+.02464;
   }
  }
```

Figura 4.84 Código de Programa para generar una señal seno.

9.- Después de realizar el código se cargará en el microcontrolador permitiendo generar la señal seno por medio del DAC dado de alta. Esta señal podrá ser observada conectando las puntas del osciloscopio en la salida del DAC (PM0- Pin No. 95 y 101 de GND (tierra)).

La señal seno generada en esta práctica se observa en la Figura 4.85.

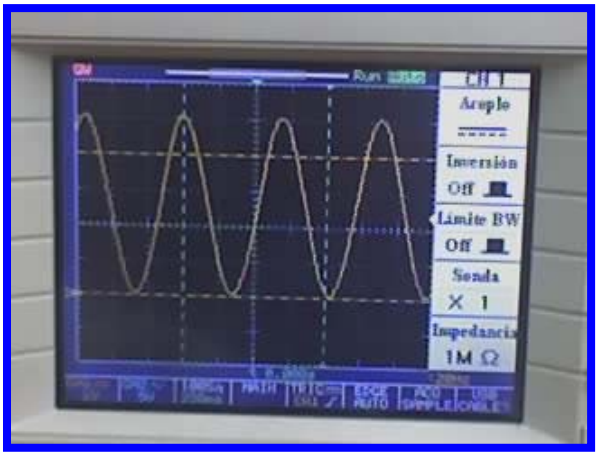
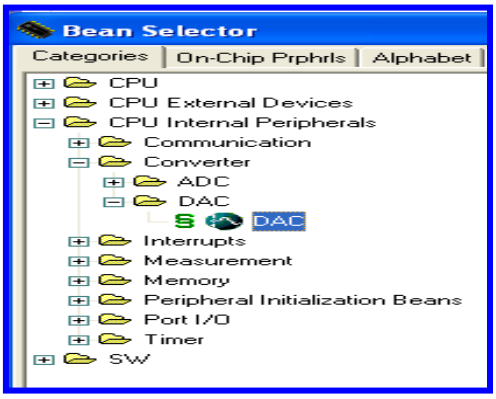


Figura 4.85 Señal Seno observada en el osciloscopio.

EJERCICIO NO. 2

Que el usuario realice un código el cual genere una función seno y un coseno, las cuales serán enviadas al DAC0 y DAC1 y observadas en su puerto de salida (en este caso los puertos de salida de los DACs).

1.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor. En esta práctica se utilizaran dos DAC.



Este Bean (Figura 4.86) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.86 Bean DAC (Seno desfasada).

2.-Para poder generar dos funciones seno tendremos que dar de alta dos convertidores Digitales- Analógicos. Figura 4.87

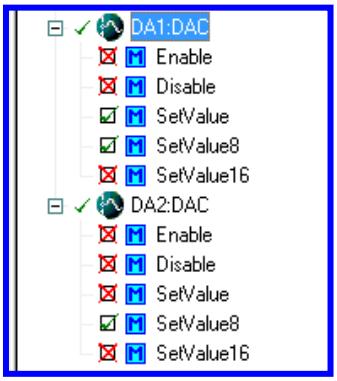


Figura 4.87 Bean de DACs (Seno desfasada).

En esta práctica se usa dos DACs para generar dos señales diferentes, la primera señal nos servirá de referencia para la segunda señal generada.

El DAC1 y DAC2 tendrá la misma configuración usando dos canales de salida, para el DAC1 utilizará el PM0_DAC0 y el DAC2 utilizará el PM1_DAC1.

3.- La configuración del Bean se realizará de la siguiente manera. Esta configuración se utilizará para los dos DAC.

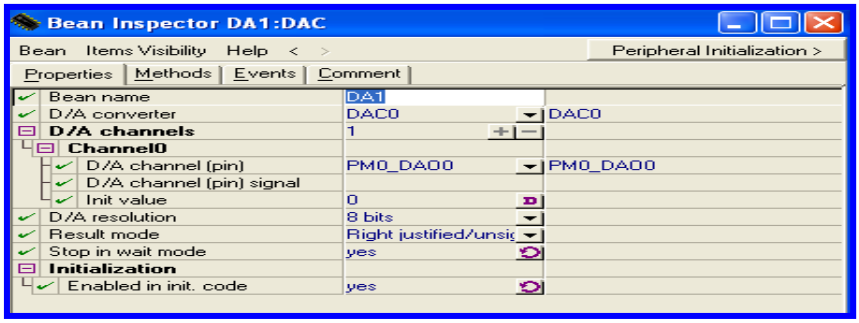


Figura 4.88 Configuración del DAC0.

La configuración de DAC0 muestra que se utilizará el canal PM0_DAO0 para el primer convertidor, y para el segundo convertidor dado de alta utilizará el canal PM1_DAO1.

4.- Visualizar los pines de conexión de los convertidores (DAC0 y DAC1). El convertidor DAC0 utilizará el pin no.95 y el convertidor DAC1 utilizará el pin no. 96. Los dos convertidores utilizarán el mismo pin no. 101.

La Figura 4.89 muestra los pines de los dos convertidores.

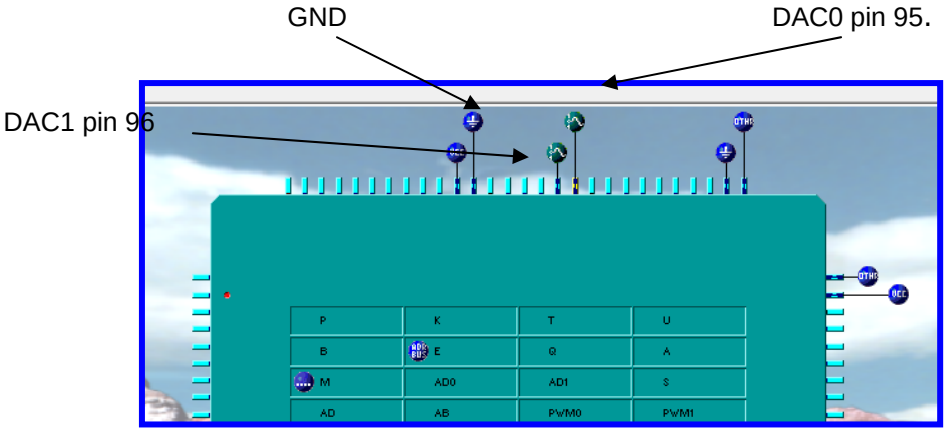


Figura 4.89 Mapa de pines (señal desfasada).

5.- Realizar el código del programa para generar una función seno y una función coseno enviadas a las salidas de los DACs.

Código del programa.-

```
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
 unsigned char valor;
 unsigned char a;
 float y,x,n;
 int i,a;
 unsigned char z,m;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!!
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

  /* Write your code here */

     for(::){

      x=0;

      for(i=0:i<255:i++){

      y= sin(x);

      z=((y+1)/2)*255;

      n= cos(x);

      m=((n+1)/2)*255;

      Byte1_PutVal(z);

      for(a=0:a<50:a++);

      DA2_SetValue(&m);

      DA1_SetValue(&z);

      x=x+.02464;

      }

     }

  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
```

Figura 4.90 Código para generar una señal seno y una coseno.

6.- Después de realizar el código se cargará en el microcontrolador permitiendo generar la señal seno y coseno por medio del DAC0 y DAC1 dados de alta. Estas señales podrá ser observada conectando las puntas del osciloscopio en la salida del DAC0 y DAC1 (PM0- Pin No. 95 - PM1- pin 96 y pin 101 de GND (tierra)).

Las señales seno y coseno generadas en esta práctica se observan en la Figura 4.91.

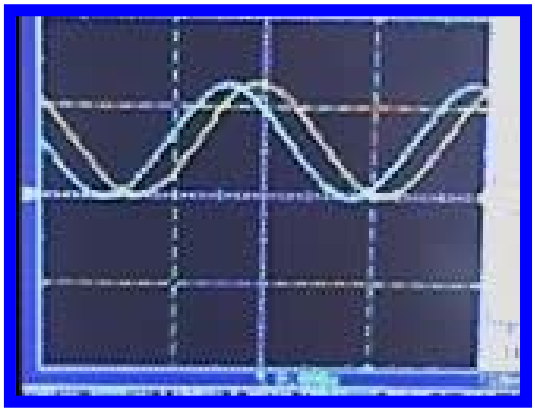


Figura 4.91 Señal Seno y coseno.

Salida del DAC0 y DAC1 se muestran en la Figura 4.92.

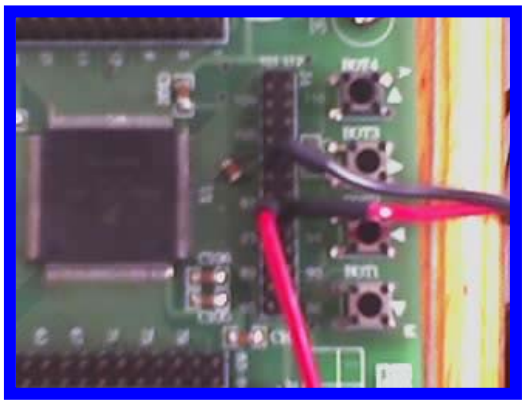


Figura 4.92 Conexión DAC0 y DAC1

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BASICAS E INGENIERIA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRCTICA NO. 9

NOMBRE: Generador de Funciones (Función Exponencial).

OBJETIVO: Que el usuario realice un código el cual genere una función exponencial, la cual será enviada al DAC y observada en su puerto de salida (en este caso el puerto de salida será el DAC).

METODOLOGÍA:

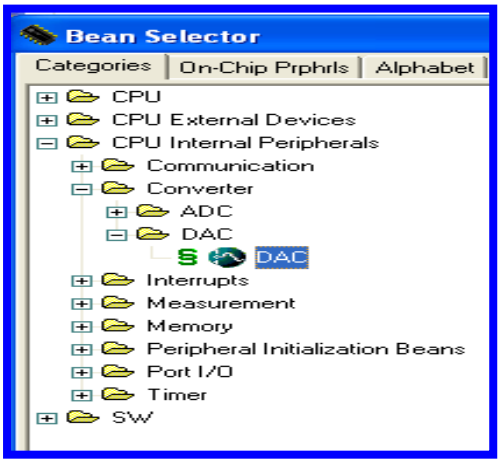
1.- Conectar el microcontrolador a una fuente regulada de 12 v de CD.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor.



Este Bean (Figura 4.93) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.93 Bean DAC (exponencial).

6.- La configuración del Bean se realizará de la siguiente manera (Figura 4.94):

Figura 4.94 Configuración de DAC (exponencial).

En esta parte podremos observar el número de canales, el canal específicamente de salida, así también la resolución que se utilizará que en este caso será de 8 bits.

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este con figuración (Figura 4.95) se puede realizar en la ventana anterior en el botón Methods.

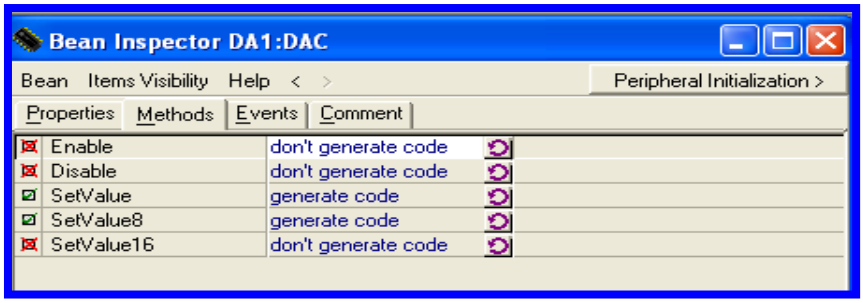


Figura 4.95 Habilitación de funciones del DAC para generar una función exponencial.

8.- En la siguiente ventana (Figura 4.96) podremos observar las funciones habilitadas en el DAC, estas permitirán realizar el código que genera la señal exponencial.

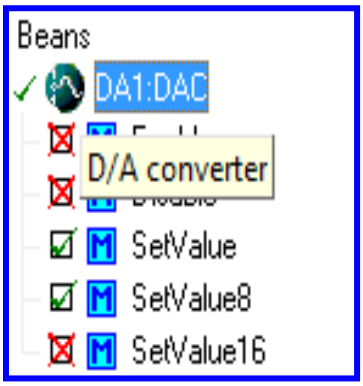


Figura 4.96 Funciones habilitadas del DAC.

9.- Realizar el código en lenguaje C que permitirá generar la señal exponencial.

```
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include <math.h>
float a;
int x;
unsigned char y;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***

  /* Write your code here */


  for(;;){

      a=0;
    for(x=0;x<255;x++){

     y=exp(a);


      DA1_SetValue8(&y);

         a=a+.021730445;



    }

  }


  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END Exponencial_rodolfo_DAC */
/*
** ###################################################################
```

Figura 4.97 Código de programa para generar una señal exponencial.

En la Figura 4.97 mostró el código necesario para generar una señal exponencial.

10.- Después de realizar el código se cargará en el microcontrolador permitiendo generar la señal exponencial por medio del DAC dado de alta. Esta señal podrá ser observada conectando las puntas del osciloscopio en la salida del DAC (PM0- Pin No. 95 y 101 de GND (tierra)).

La señal exponencial generada en esta práctica se observa en la Figura 4.97.1

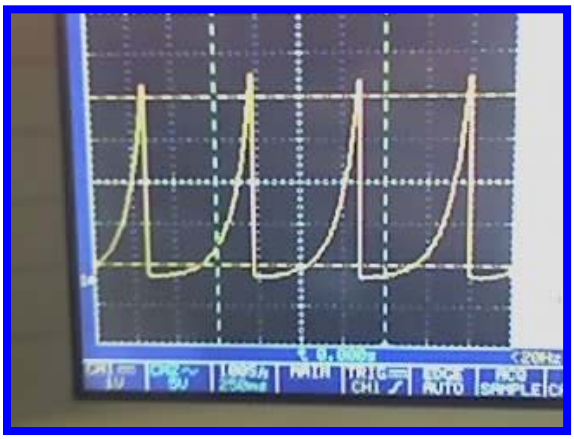


Figura 4.97.1 Función Exponencial observada por medio del osciloscopio.

Al generar todas estas funciones nos permitirán mas adelante darle diversos usos, por ejemplo nos permitirá sustituir en algunas ocasiones un instrumento tan importante como un Generador de Funciones.

EJERCICIO NO. 2

Que el usuario realice un código el cual genere una función exponencial y exponencial negativa, la cual será enviada al DAC y observada en su puerto de salida (en este caso el puerto de salida será la del DAC). Esta señal podrá ser observada conectando un osciloscopio.

1.- Para realizar esta práctica se dará de alta un DAC, la configuración de este será la misma que en el ejercicio anterior así como también sus funciones utilizadas.

En la Figura 4.98 se muestra el bean del DAC de la ventana de Bean Selector.

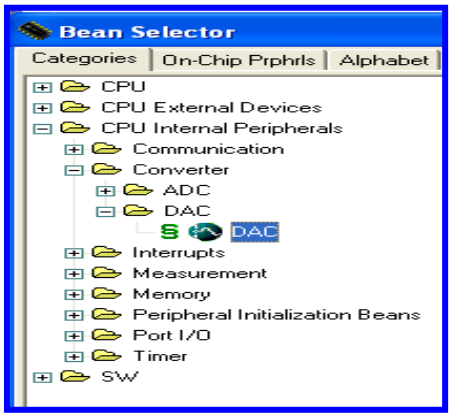


Figura 4.98 Bean DAC (doble exponencial).

2.- Realizar el código de programa que permita generar una función exponencial y una función exponencial invertida. Esta podrá ser observada en la salida del DAC0.

La salida del DAC0 será la misma que en el ejercicio anterior (práctica para generar una señal exponencial).

Nota: Tomar en cuenta la configuración utilizada en los bean del DAC utilizados.

Código del programa.- en la Figura 4.99 muestra el código de programa para generar una señal exponencial y una exponencial invertida.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "DA1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include <math.h>
int x;
float a;
unsigned char y;


void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS (
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.

/* Write your code here */
  for(;;){

    a=0;
    for(x=0;x<255;x++)
    {

     y=exp(a);


      DA1_SetValue8(&y);

         a=a+.021730445;
    }


       a=5.541263475;
     for(x=255;x>0;x--)
     {

          y=exp(a);

         DA1_SetValue8(&y);

         a=a-.021730445;
     }

        }

  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END Doble_Exp_Rodolfo */
```

Figura 4.99 Código para generar una señal doble exponencial.

3.- Después de realizar el código se cargará en el microcontrolador permitiendo generar la señal exponencial por medio del DAC dado de alta. Esta señal podrá ser observada conectando las puntas del osciloscopio en la salida del DAC (PM0- Pin No. 95 y 101 de GND (tierra)).

La señal exponencial generada en esta práctica se observa en la Figura 4.100.

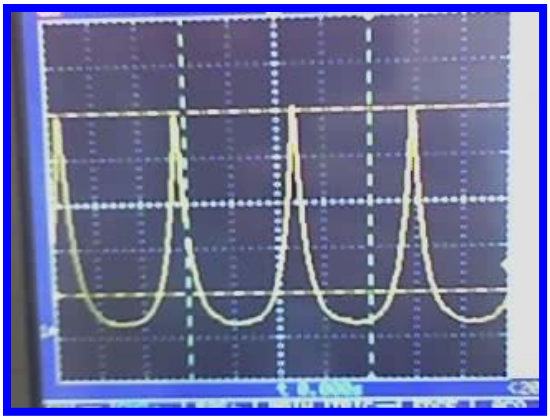


Figura 4.100 Señal doble Exponencial.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 10

NOMBRE: Desfasamiento de una señal Seno.

OBJETIVO: Que el usuario realice un código el cual genere dos funciones seno con diferente fase, las cuales será enviada al DAC y observada en su puerto de salida (en este caso el puerto de salida será la del DAC). Esta señal podrá ser observada conectando un osciloscopio.

METODOLOGÍA:

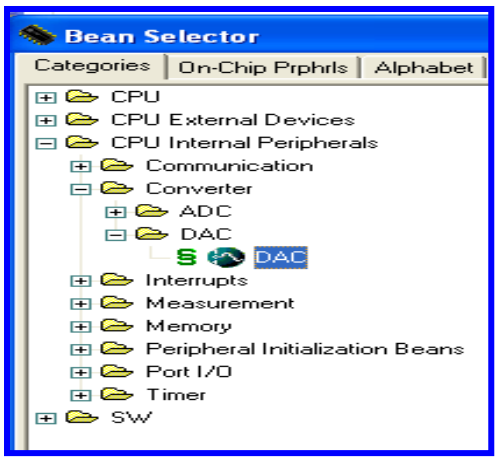
1.- Conectar el microcontrolador a una fuente regulada de 12 v de CD.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor. En esta práctica se utilizaran dos DAC.



Este Bean (Figura 4.101) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.101 Bean DAC (Seno desfasada).

6.-Para poder generar dos funciones seno tendremos que dar de alta dos convertidores Digitales- Analógicos. Figura 4.102

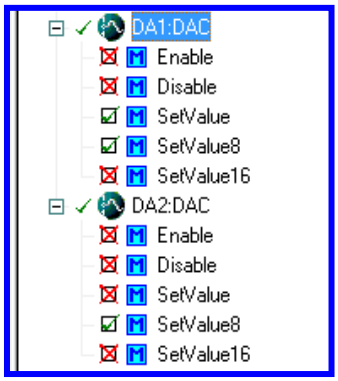


Figura 4.102 Bean de DACs (Seno desfasada).

En esta práctica tuvimos que usar dos DAC para generar dos señales diferentes, la primera señal nos servirá de referencia para la segunda señal generada.

6.- La configuración del Bean se realizará de la siguiente manera. Esta configuración se utilizará para los dos DAC.

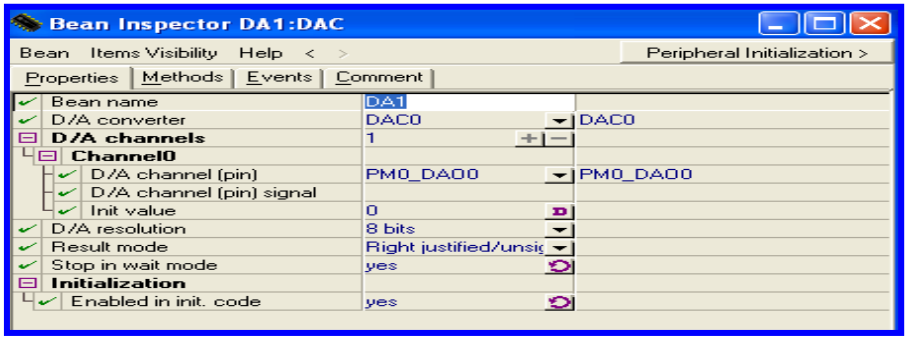


Figura 4.103 Configuración del DAC0.

La configuración de DAC0 muestra que se utilizará el canal PM0_DAO0 para el primer convertidor, y para el segundo convertidor dado de alta utilizará el canal PM1_DAO1.

7.- Visualizar los pines de conexión de los convertidores (DAC0 y DAC1). El convertidor DAC0 utilizará el pin no.95 y el convertidor DAC1 utilizará el pin no. 96. Los dos convertidores utilizarán el mismo pin no. 101.

La Figura 4.104 muestra los pines de los dos convertidores.

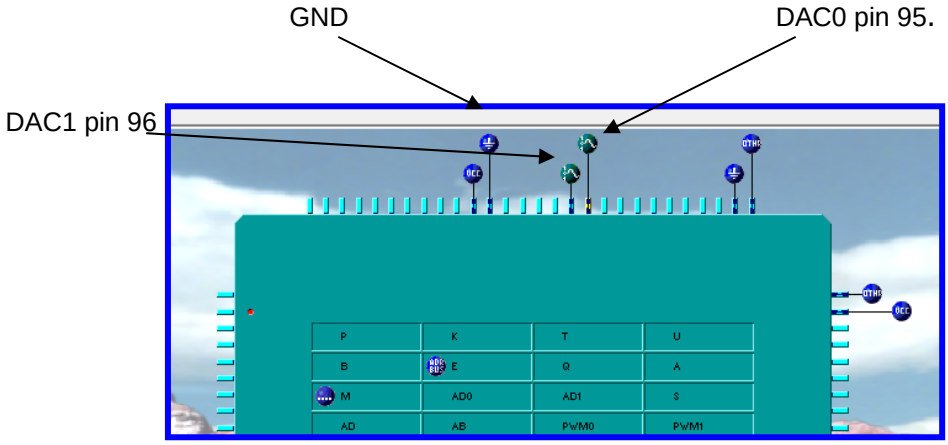


Figura 4.104 Mapa de pines (señal desfasada).

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este configuración (Figura 4.105) se puede realizar en la ventana anterior en el botón Methods.

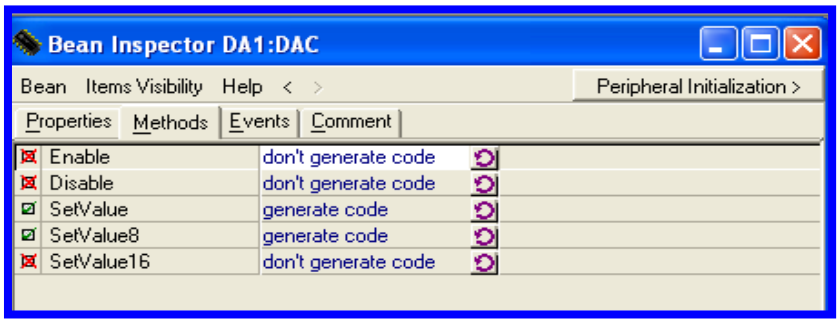


Figura 4.105 Habilitación de funciones del DAC (señal desfasada).

8.- Realizar el código del programa. En este código se generan dos funciones seno desfasadas 90°. Cada señal generada será observada en la salida de cada Convertidor digital-analógico. En codigo se observa en la Figura 4.106.

```
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include <math.h>
float y,x,n;
int i,a;
unsigned char z,m;
void main(void)
{
  /*** Processor Expert internal initialization
  PE_low_level_init();
  /*** End of Processor Expert internal initial

  /* Write your code here */


   for(;;){
   x=0 ;
   for(i=0;i<255;i++){
   y = sin(x);
    z=((y+1)/2)*255;
   n = sin(x + 1.5708);
    m=((n+1)/2)*255;

      Byte1_PutVal(z);
  for(a=0;a<50;a++);

    DA2_SetValue8(&m);
    DA1_SetValue8(&z);
    x=x+.02464;
   }

   }
```

Sen(0°)

Sen (90°)

Figura 4.106 Código de programa para generar dos señales senos desfasados.

La Figura 4.107 muestra los pines de los convertidores del microcontrolador (DAC0 y DAC1).

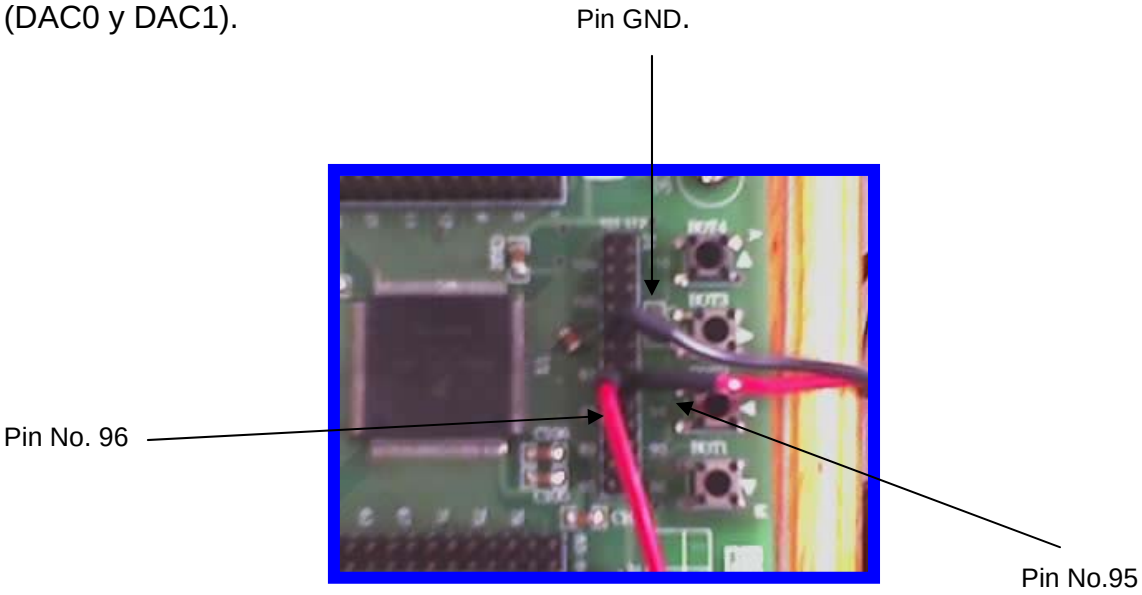


Figura 4.107 Conexiones de DAC0 y DAC1.

La Figura 4.108 muestra las dos señales seno obtenidas.

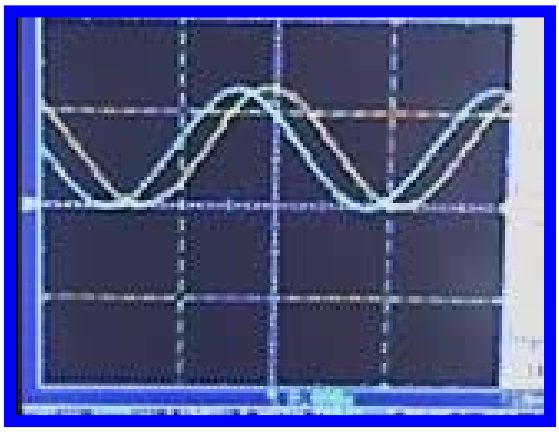


Figura 4.108 Señales seno desfasadas 90°.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 11

NOMBRE: Selección de una señal seno desfasada 45-360°.

OBJETIVO: Realizar el código que genere rutinas para la selección de funciones seno con diferente fase (de 45° a 360° de desfasamiento), los cuales podrán ser seleccionadas por medio de un menú de botones, cada par de señales seleccionada será enviadas al DAC0 - DAC1 y podrán ser observadas en su puerto de salida (en este caso el puerto de salida será la del DAC0 y DAC1). Esta señal podrá ser observada conectando las puntas del osciloscopio en las salidas de los DACs dados de alta.

METODOLOGÍA:

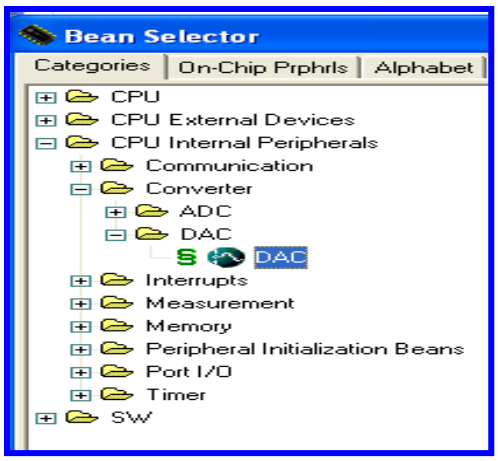
1.- Conectar el microcontrolador a una fuente regulada de 12 v de CD.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor. En esta práctica se utilizaran dos DAC (DAC0 y DAC1).



Este Bean (Figura 4.109) nos servirá para convertir una señal digital en una señal analógica. Habilitar dos Bean DAC (DAC0 y DAC1).

Figura 4.109 Bean DAC (Selección de señales).

6.-Para poder generar dos funciones seno tendremos que dar de alta dos convertidores Digitales- Analógicos. Figura 4.110.

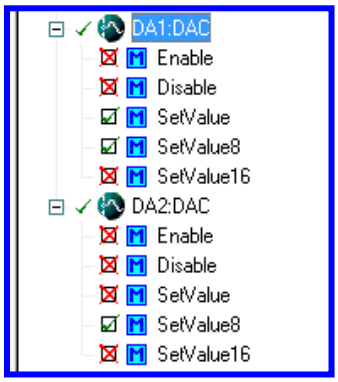


Figura 4.110 Bean de DACs (Selección de señales).

Estos convertidores digitales – analógicos (DAC) se utilizarán para generar las dos señales que serán seleccionadas por medio de cuatro botones.

6.- La configuración del Bean se realizará de la siguiente manera. Esta configuración se utilizará para los dos DAC (DAC0 y DAC1).

Figura 4.111 Configuración del DAC0.

La configuración de DAC0 (Figura 4.111) muestra que se ocupan el canal PM0_DAO0 para el primer convertidor, y para el segundo convertidor dado de alta utilizará el canal PM1_DAO1.

7.- Visualizar los pines de conexión de los convertidores (DAC0 y DAC1). El convertidor DAC0 utilizará el pin no.95 y el convertidor DAC1 utilizará el pin no. 96. Los dos convertidores utilizarán el mismo pin no. 101.

La Figura 4.111 muestra los pines de los dos convertidores.

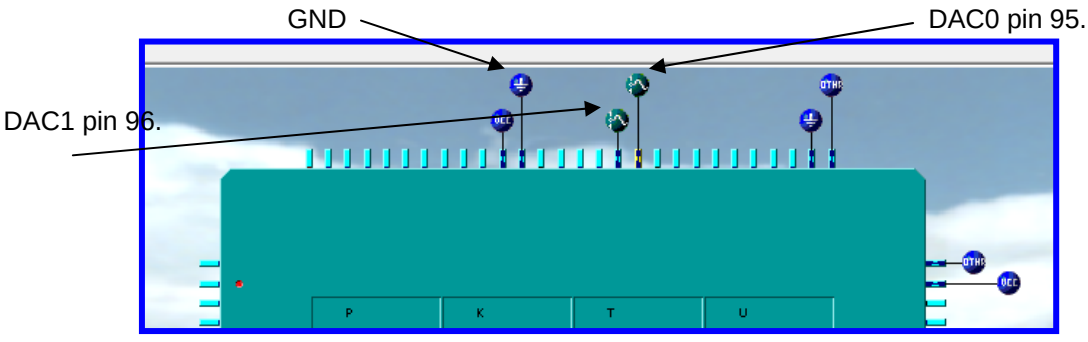


Figura 4.112 Mapa de pines (Selección de señales).

7.- Una parte importante para considerar es dar de alta la función que nos permita una conversión utilizando 8 bits. Este configuración (Figura 4.113) se puede realizar en la ventana anterior en el botón Methods.

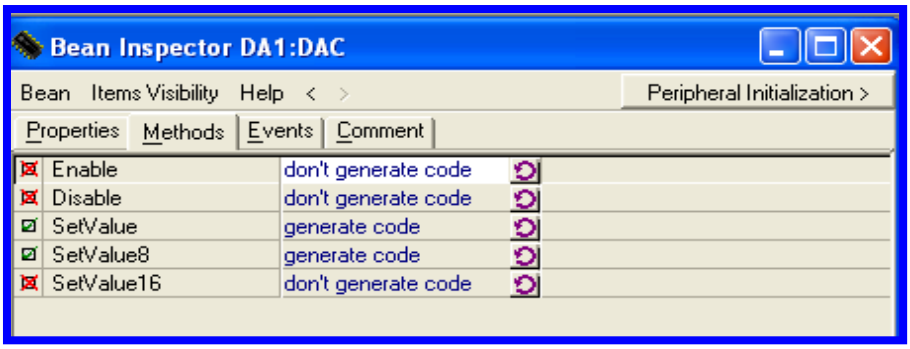


Figura 4.113 Habilitación de funciones del DAC (Selección de señales).

8.- Dar de alta cuatro Bean Bit1. Estos servirán como un menú de selección de las señales generadas, por lo tanto se utilizan como un Bean de entrada. Figura 4.114.

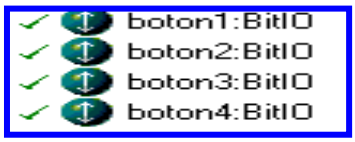


Figura 4.114 Botones de entrada.

8.- Configuración de los Bean de entrada (Botón). Esta se realizará utilizando el puerto AD con los pines 8 al 11. Se muestra en la Figura 4.115.

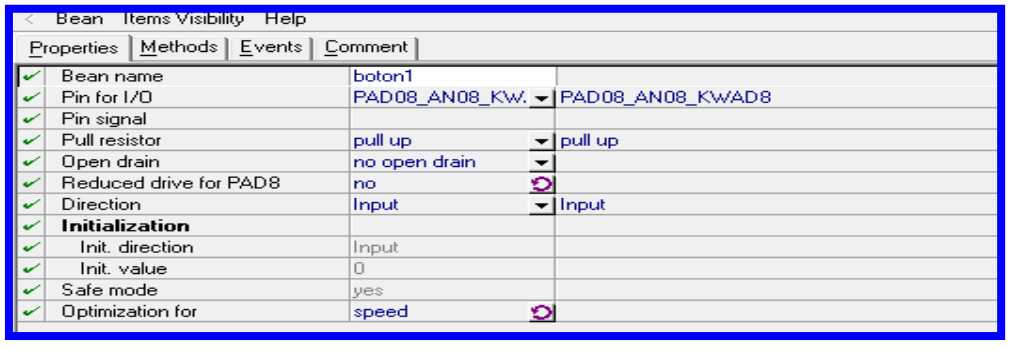


Figura 4.115 Configuración de botón PAD_08_AN08_KWAD8.

8.- Realizar el código del programa. En este código se generan dos pares de funciones senos desfasados, las cuales serán seleccionadas oprimiendo un botón. En el primer par se generaran una señal seno y otra desfasa 90°, en la segunda se generará otra desfasada 180°, en la tercera se genera una desfasada 270° y el la ultima sección se genera una desfasada 360° Cada señal generada será observada en la salida de cada Convertidor digital-analógico.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "DA1.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
 unsigned char valor;
 unsigned char a;
 int a,b,c,d,i,j;
float y,x,n;
unsigned char z,m;
void main(void)
       {
/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
 /*** End of Processor Expert internal initialization.                    ***/

 /* Write your code here */
       for(;;)   {
a = boton1_GetVal();
b = boton2_GetVal();
c = boton3_GetVal();
d = boton4_GetVal();

       while (a==0){

rut1:
b = boton2_GetVal();
c = boton3_GetVal();
d = boton4_GetVal();

 x=0;
```

```
for(i=0;i<255;i++) {
y= sin(x);
z= ((y+1)/2)*255;
n= sin(x+ .7854);
m= ((n+1)/2)*255;

DA1_SetValue(&z);
DA2_SetValue(&m);

x= x+ .02464;

If (b==0) goto rut2;
If (c==0) goto rut3;
If (d==0) goto rut4;

}
        while (b==0)   {

 rut1:
a = boton1_GetVal();
c = boton3_GetVal();
d = boton4_GetVal();

 x=0;
 for(i=0;i<255;i++) {

y= sin(x);
z= ((y+1)/2)*255;
n= sin(x+ 1.5708);
m= ((n+1)/2)*255;

DA1_SetValue(&z);
DA2_SetValue(&m);

x= x+ .02464;

If (a==0) goto rut2;
If (c==0) goto rut3;
If (d==0) goto rut4;

}
  while (a==0)   {

rut 3:
a = boton1_GetVal();


 b = boton2_GetVal();
d = boton4_GetVal();
```

```
 x=0;
 for(i=0;i<255;i++) {

y= sin(x);
z= ((y+1)/2)*255;
n= sin(x+ 3-1416);
m= ((n+1)/2)*255;

DA1_SetValue(&z);
DA2_SetValue(&m);

X= x+ .02464;

If (a==0) goto rut2;
If (b==0) goto rut3;
If (d==0) goto rut4;

}
        while (a==0)  {

rut4:
a = boton1_GetVal();
b = boton2_GetVal();
c = boton3_GetVal();

                    x=0;
for(i=0;i<255;i++)    {

y= sin(x);
z= ((y+1)/2)*255;
n= sin(x+ 6.2852);
m= ((n+1)/2)*255;

DA1_SetValue(&z);
DA2_SetValue(&m);

X= x+ .02464;

If (b==0) goto rut2;
If (c==0) goto rut3;
If (d==0) goto rut4;
}
/*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
```

La Figura 4.116 muestra los botones de selección de los pares de las señales generadas.

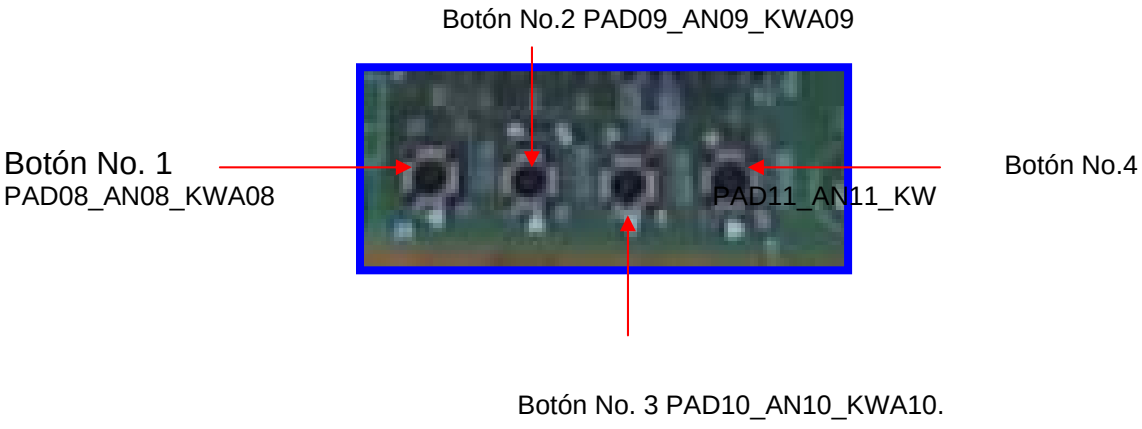


Figura 4.116 Botones de selección.

El botón número 1 permitirá entrar a la primera rutina.
El botón número 2 permitirá entrar a la segunda rutina.
El botón número 3 permitirá entrar a la tercera rutina.
El botón número 4 permitirá entrar a la cuarta rutina.

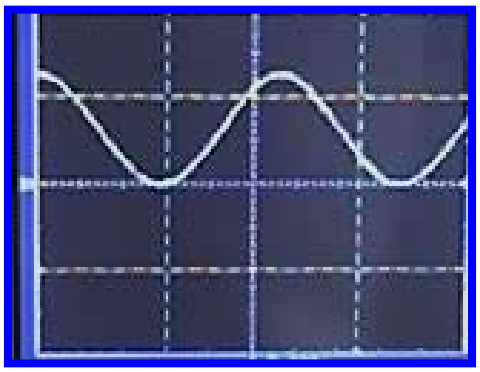
En la siguiente tabla observaremos el menú de selección de rutinas:

Rutina 1	Rutina 2	Rutina 3	Rutina 4
Sen(0°)	Sen(0°)	Sen(0°)	Sen(0°)
Sen(45°)	Sen(90°)	Sen(180°)	Sen(360°)

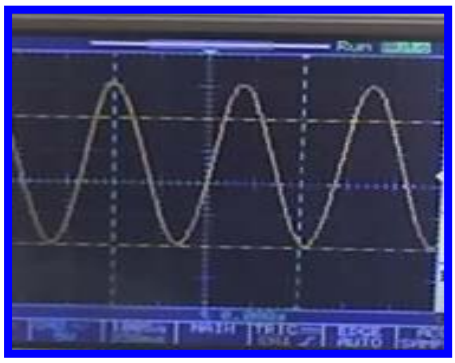
Tabla 4.1 Rutinas de señales.

En las siguientes cuatro figuras se muestran las señales seleccionadas por los botones del puerto AD anteriormente configurados.

En cada una de las rutinas aparecerá la señal seno y la señal desfasada seleccionada.

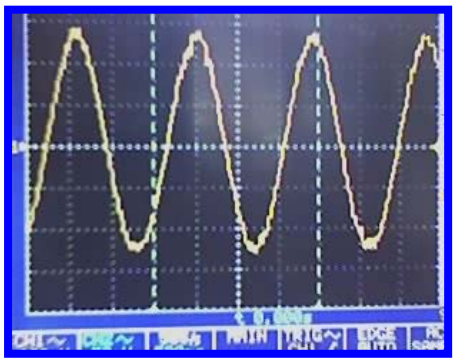


Rutina 1 Sen (90°)



Rutina 3 sen (180°)

Rutina 2 sen (45°)



Rutina 4 sen (360°)

Figura 4.117 Señales seno desfasadas.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 12

NOMBRE: Desfasamiento de una señal con el uso del potenciómetro.

OBJETIVO: Que el usuario realice un código el cual genere una función seno que sea desfasada de 0 a 180° por medio del puerto PAD12_AN12_KWA12 (potenciómetro).

METODOLOGÍA:

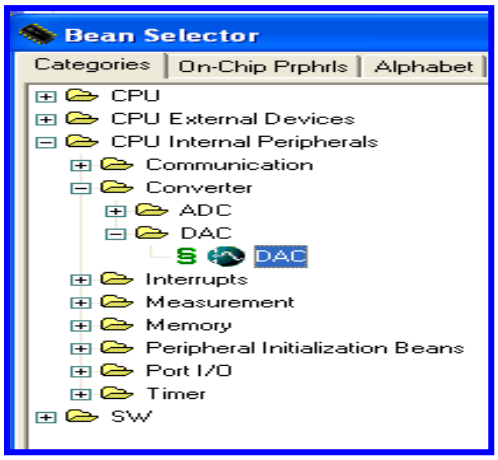
1.- Conectar el microcontrolador a una fuente regulada de 12 voltios de corriente continua.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals-Converte-DAC.* Hacer doble clic en el Bean DAC para dar de alta al convertidor.



Este Bean (Figura 4.118) nos servirá para convertir una señal digital en una señal analógica.

Figura 4.118 Bean DAC (Seno desfasada).

La configuración de DAC será la misma que en las prácticas anteriores.

6.- Dar de alta el un Bean ADC (convertidor analógico-digital).Elegir la ventana Bean Selector *-CPU Internal Peripherals- Converte-DAC.* Hacer doble clic en el Bean ADC para dar de alta al convertidor.

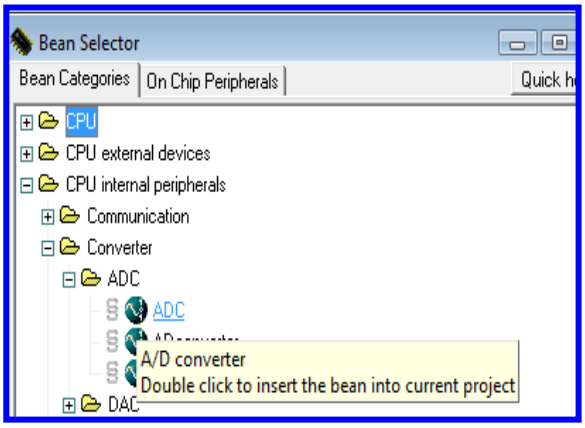


Figura 4.119 Bean ADC.

En la Figura 4.119 muestra el bean del ADC utilizado en esta práctica.

El uso del convertidor analógico- digital nos permitirá el manejo del puerto PAD12_AN12_KWA12.

7.- Configuración del ADC. La configuración del ADC se realizará de la siguiente manera (Figura 4.120):

Nota.-Es de gran importancia realizar una configuración adecuada de esta herramienta.

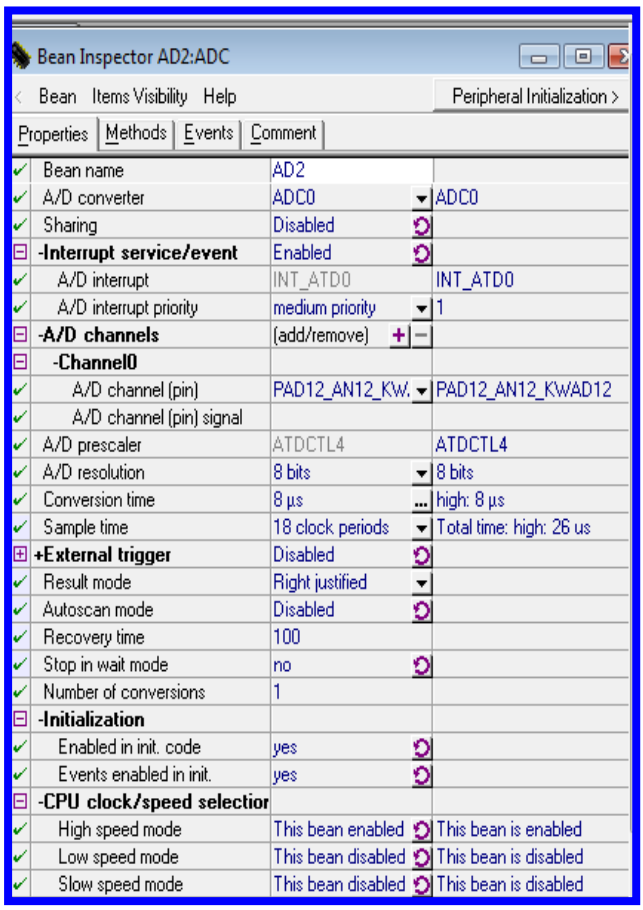


Figura 4.120 Configuración del ADC.

El uso del ADC es primordial en esta práctica ya que es el Bean que nos permite el manejo del POT del microcontrolador. Su funcionamiento radica en convertir la señal analógica dada por el POT1 en una señal digital, la cual será manipulada y enviada al DAC para ser regenerada en una señal continua y poder ser observada por medio del osciloscopio.

8.- Realizar el código del programa. En este código se generan una función seno la cual será desfasada de 0-180° por medio del puerto PAD12_AN12_KWA12.

```
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include  <math.h>
int i;
unsigned char m,z;
float x,n,p,y;
byte valor;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

  AD2_Start();

  for(;;)

  {
    x=0 ;
   for(i=0;i<255;i++){

   AD2_GetValue8(&valor);

   p=((valor+1)/2)*255;

     n = sin(x+p);

    m=((n+1)/2)*255;

      y= sin(x);

      z=((y+1)/2)*255;

      DA1_SetValue8(&m);

    DA2_SetValue8(&z);


    x=x+.02464;
   }
  }
```

Figura 4.121 Código de Programa para generar una señal desfasada por medio del puerto PAD.

En esta práctica se tomo como señal base una función seno, pero este programa puede ser utilizado para desfasar cualquier señal.

9.- Puerto PAD12_AN12_KWA12. En la Figura 4.122 se muestra el puerto AD en el microcontrolador el cual nos permitirá el desfasamiento de la señal generada.

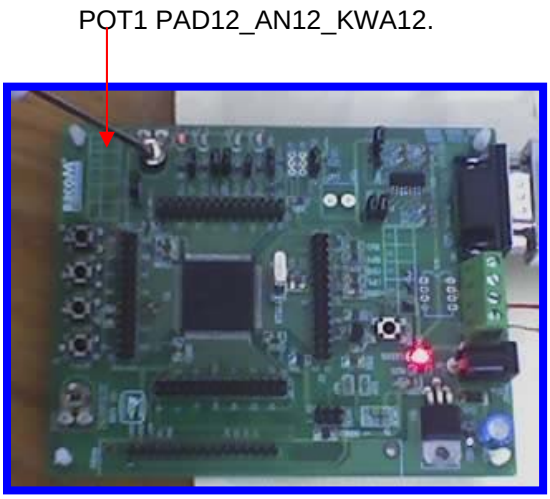


Figura 4.122 PDA12_AN12_KWA12.

10.- Señal generada. En la Figura 4.123 muestra un ejemplo de la señal generada y desfasada 90° por medio del puerto PAD12_AN12_KWA12.

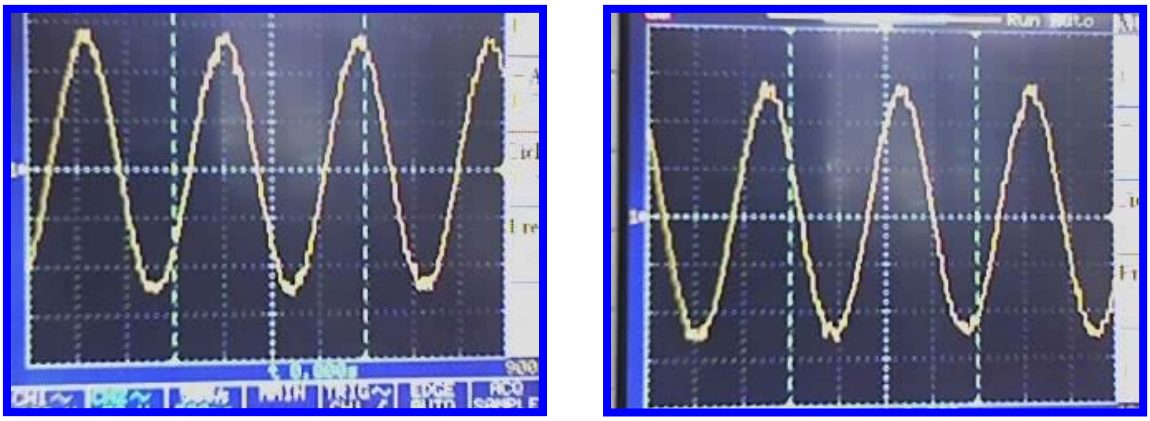


Figura 4.123 Señal sen (0°) y señal sen (90°).

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

MANUAL DE PRÁCTICAS

Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 13

NOMBRE: Control de PWM para manipular un Servo robot.

OBJETIVO: Que el usuario realice un código el cual controle un servo robot de cuatro grados de libertad utilizando cuatro PWM, y controlados por medio de una programación grafica LabVIEW utilizando la comunicación serial para la manipulación de los servomotores.

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v de CD.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.-Elegir en la ventana Bean Selector (Figura 4.124).- *CPU Internal Peripherals-Timer-PWM*. Habilitar cuatro beans *PWM*.

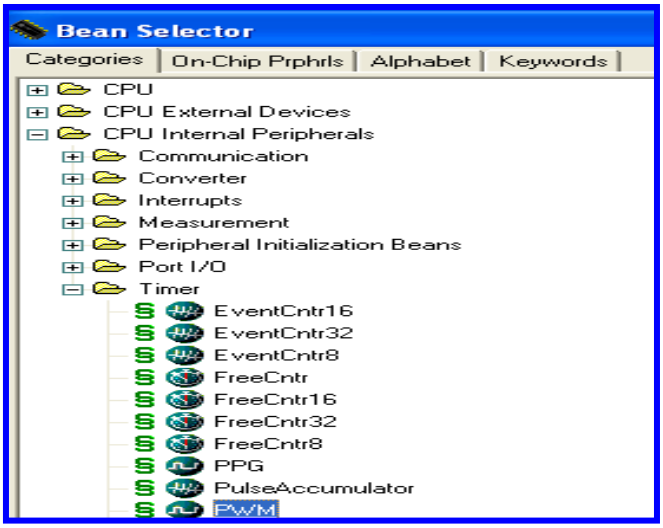


Figura 4.124 Selección PWM (Control de PWM).

6.- Realizar la configuración de los *PWM*. Esta configuración (Figura 4.125) se utilizará para los cuatro PWM que se dieron de alta.

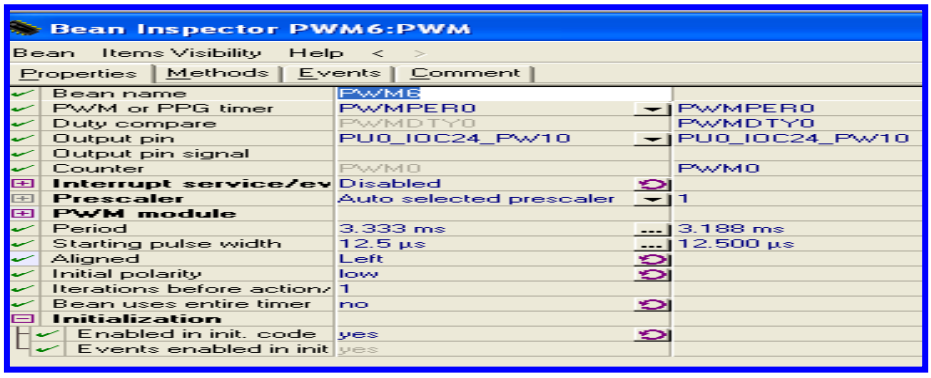


Figura 4.125 Configuración del PWM (control de servo robot).

Un punto importante de mencionar es los pines de salida de los PWM:
Los PWM utilizarán el puerto U como salida.

El primer PWM utilizará el pin PU0_IOC24_PW10 como pin de salida.
El segundo PWM utilizará el pin PU1_IOC25_PW11 como pin de salida.
El tercer PWM utilizará el pin PU2_IOC26_PW12 como pin de salida.
El cuarto PWM utilizará el pin PU3_IOC27_PW12 como pin de salida.

7.-Para la configuración del *Timer-PWM* un aspecto importante de considerar es la configuración de su *Period y Starting pulse width.*

En la siguiente ventana (Figura 4.126) se realiza la configuración del *Timer-PWM*.

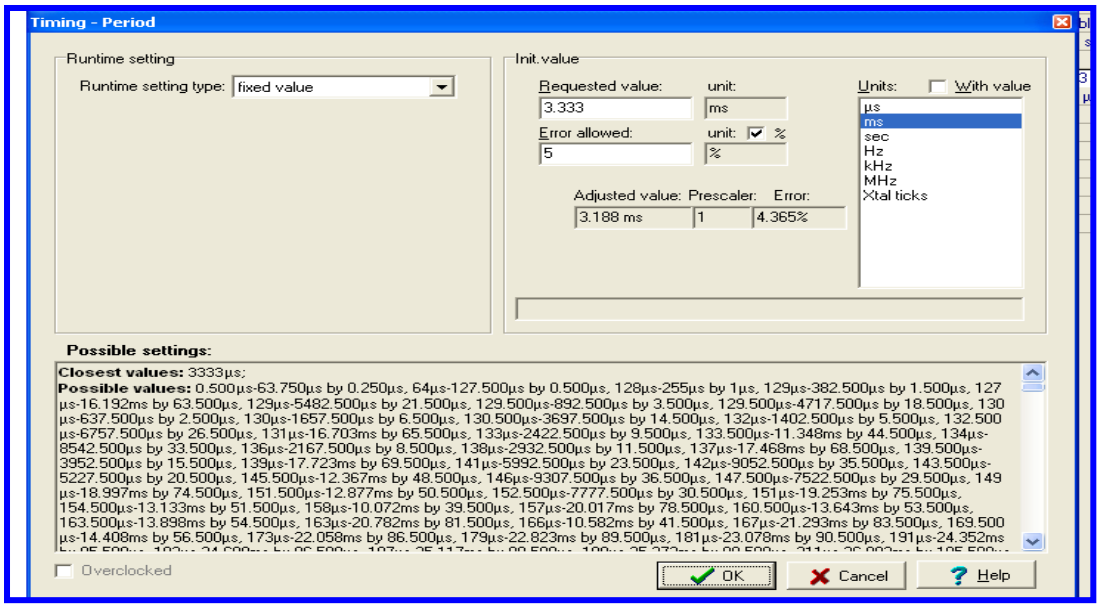


Figura 4.126 Configuración Periodo (Control de servo robot).

En la configuración del Period del PWM podremos elegir unidad en la que queramos trabajar y el valor al que trabajé.

Starting pulse width.-

En la Figura 4.127 muestra la configuración del Starting del PWM.

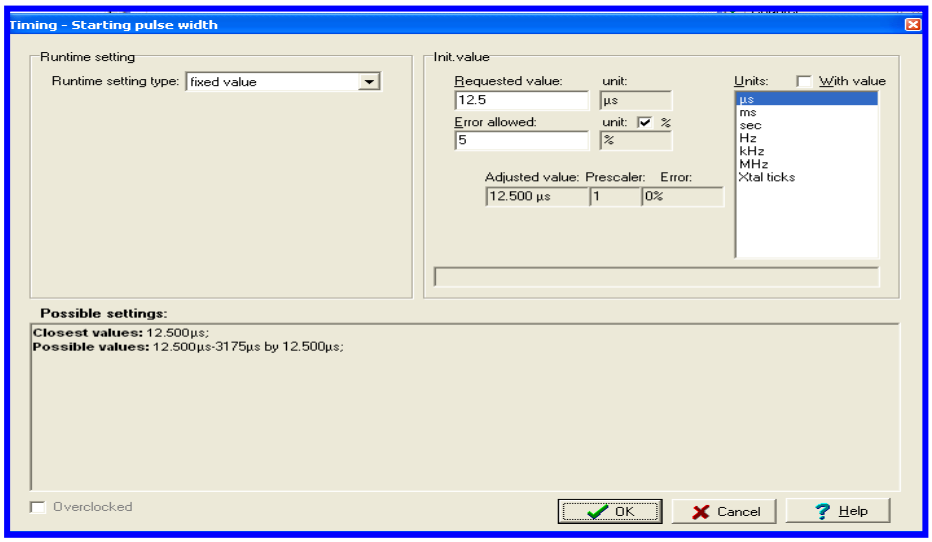
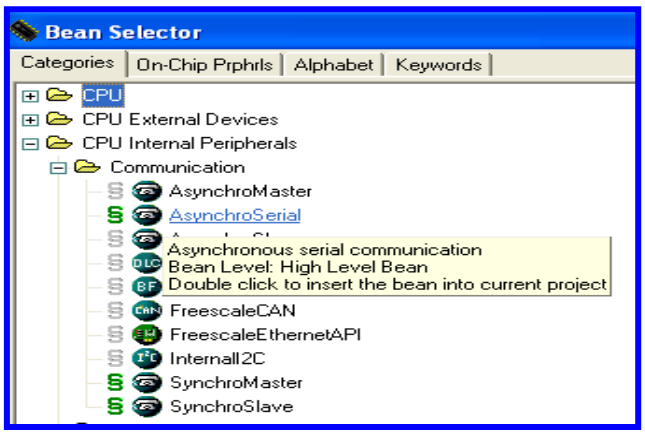


Figura 4.127 Configuración de starting pulse with (control de servo robot).

8.- Elegir en la ventana de Bean Selector- *CPU Internal Peripherals- Conmunication- AsynchroSerial.*

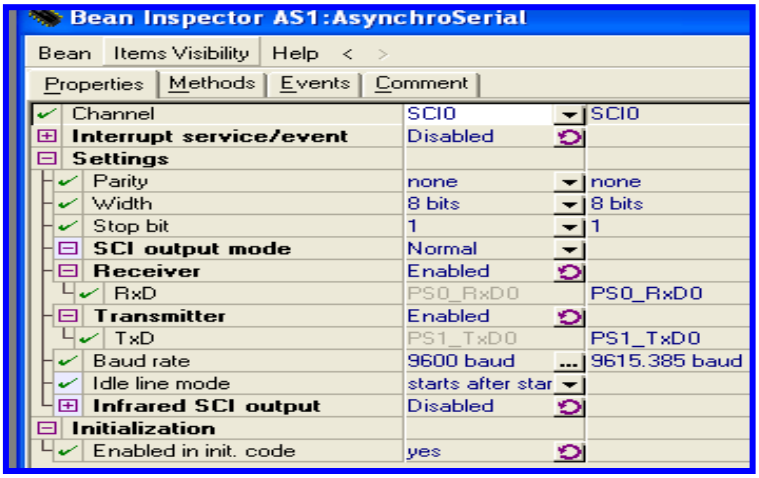
En la siguiente Figura 4.128 muestra la habilitación del bean de comunicación AsynchroSerial.



Este Bean nos permitirá la comunicación por la interface RS232.

Figura 4.128 Bean AsynchroSerial (Control servo robot).

8.- La configuración del Bean AsynchroSerial se realizará de la siguiente manera (Figura 4.129).-



En esta parte tendremos la opción de elegir el canal por donde podremos ver la salida de la conversión de la señal. Si elegimos el canal SCIO utilizaremos el interface RS232 y si utilizamos el canal SCI1 podremos ver la señal conectando un osciloscopio.

Figura 4.129 Configuración del Bean AsynchroSerial (control de servo robot).

9.- Funciones habilitadas para realizar el control del servo robot. En la siguiente Figura 4.130 muestra los Bean de la Comunicación Serial y PWM.

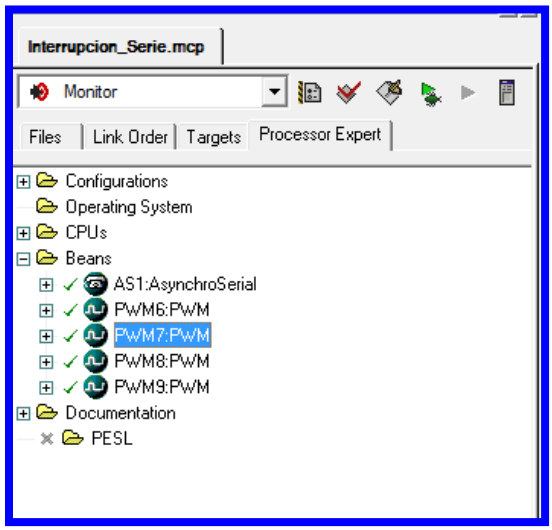


Figura 4.130 Bean habilitados para el control de servo motores.

9.-Realizar el código de programación. Este programa permitirá la manipulación de PWM por medio del interfaz grafico de LabVIEW, este control se realizará por medio de la comunicación serial.

```
/* MODULE Interrupcion_Serie */

/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "AS1.h"
#include "PWM6.h"
#include "PWM7.h"
#include "PWM8.h"
#include "PWM9.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
byte motor;
byte valor;
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

PWM6_SetRatio8(0);
PWM7_SetRatio8(0);
PWM8_SetRatio8(0);
PWM9_SetRatio8(0);
valor=255;

for(;;){

 AS1_RecvChar(&motor);
  switch(motor){
  case(1):{
  while (valor==255){
  AS1_RecvChar(&valor);
     }

  AS1_SendChar(valor);

PWM6_SetRatio8(valor);
motor=0;
valor=255;
break;
```

```
   }

 case(2):{
while(valor==255){
AS1_RecvChar(&valor);
     }

AS1_SendChar(valor);
PWM7_SetRatio8(valor);
motor=0;
valor=255;
break;
   }

    case(3):{
   while(valor==255){
   AS1_RecvChar(&valor);
     }

AS1_SendChar(valor);
PWM8_SetRatio8(valor);
motor=0;
valor=255;
break;

   }

 case(4):{
while(valor==255){
AS1_RecvChar(&valor);
     }

AS1_SendChar(valor);
PWM9_SetRatio8(valor);
motor=0;
valor=255;
break;

   }

default:{
motor=0;

valor=255;
break;

   }
 }
 }
```

```
  /* Write your code here */

  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
```

10.- Realizar el programa en LabVIEW que permita controlar cuatro PWM del microcontrolador por medio de la comunicación serial.

LabVIEW constituye un revolucionario sistema de programación gráfica para aplicaciones que involucren adquisición, control, análisis y presentación de datos.

Además LabVIEW es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el *lenguaje C* o *BASIC*.

LabVIEW también proporciona potentes herramientas que facilitan la depuración de los programas.

Los programas desarrollados mediante LabVIEW se denominan *Instrumentos Virtuales* (*VIs*), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los *VIs* tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros *VIs*

Este instrumento nos permitirá realizar un programa para la manipulación de los PWM del sistema de desarrollo.

La Figura 4.131 muestra la ventana del Panel Frontal de LabVIEW.

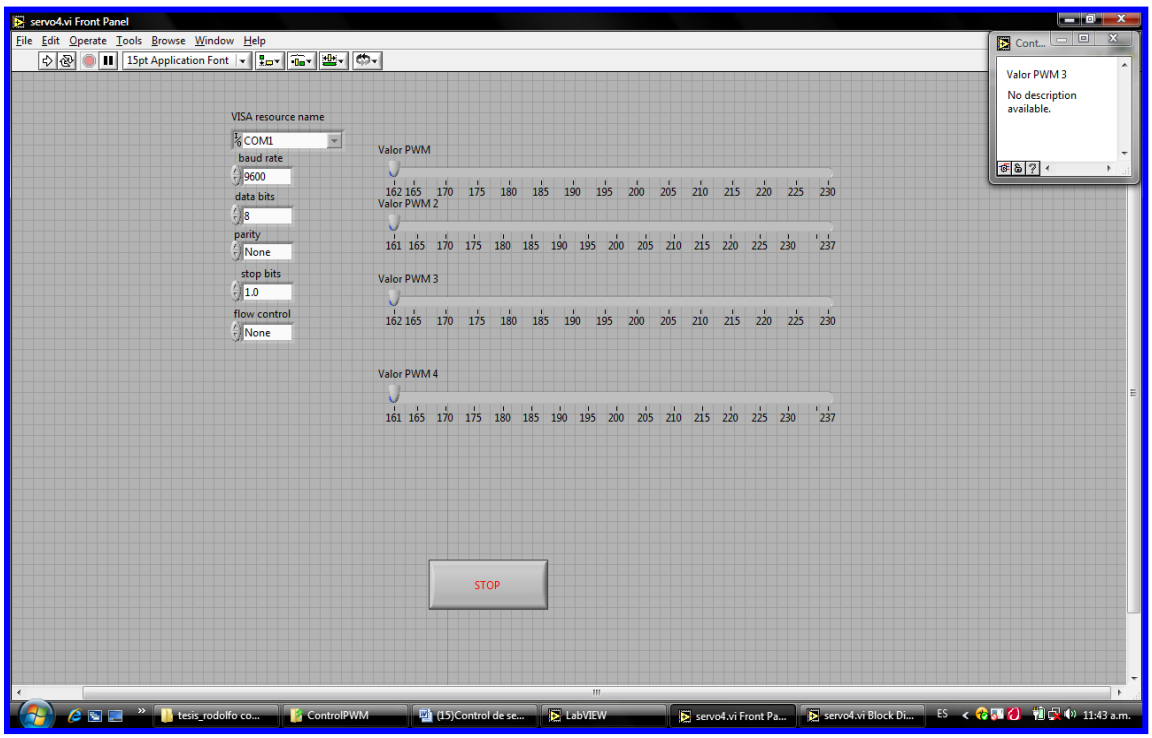


Figura 4.131 Panel Frontal del programa desarrollado para controlar el PWM.

Este programa grafico nos permitirá la manipulación de los PWM utilizando la comunicación serial.

También nos permite utilizar diversas funciones de LabVIEW como: VISA write, Time Daley, Convert form Dynamic Data, Byte Array to String,VISA Close, Simple Error Handler.

Todas estas funciones nos permitirán realizar la comunicación serial con el microcontrolador y el control de los recursos del mismo.

En la Figura 4.132 muestra el panel frontal de las funciones utilizadas.

Funciones utilizadas para desarrollar:

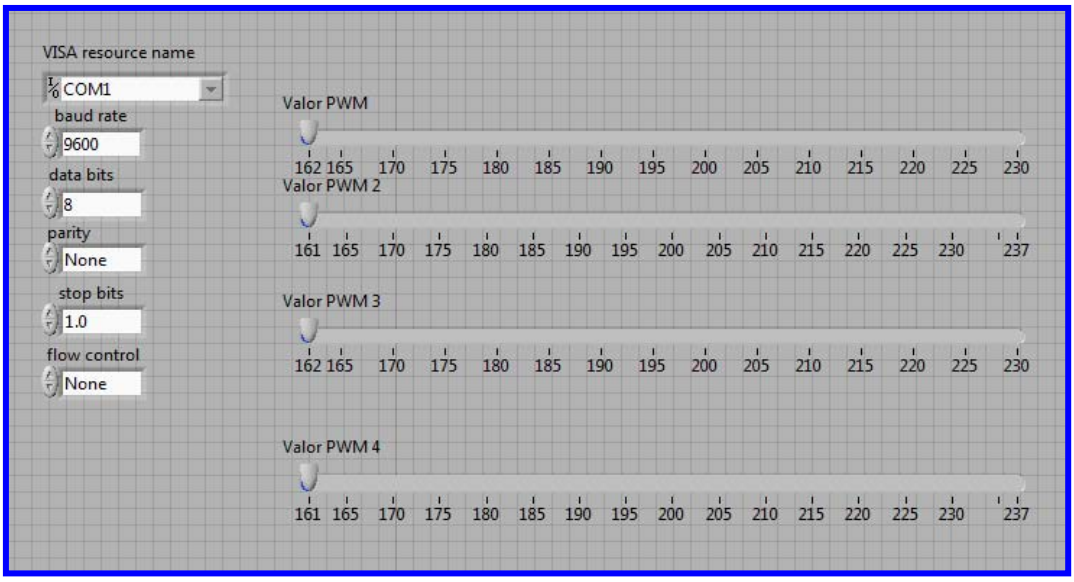


Figura 4.132 Panel Frontal de las funciones utilizadas.

11.- En esta parte se muestra el Diagrama a Bloques del programa desarrollado en LabVIEW. En el diagrama a bloques se realizan las conexiones de las funciones utilizadas. En la figura 4.133 se muestra el diagrama a bloques del programa desarrollado.

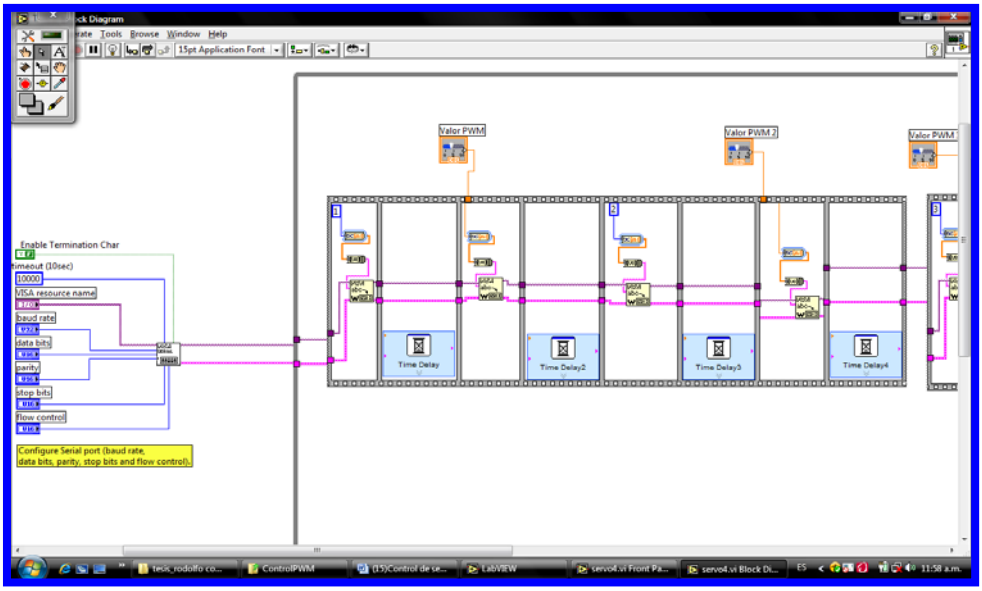
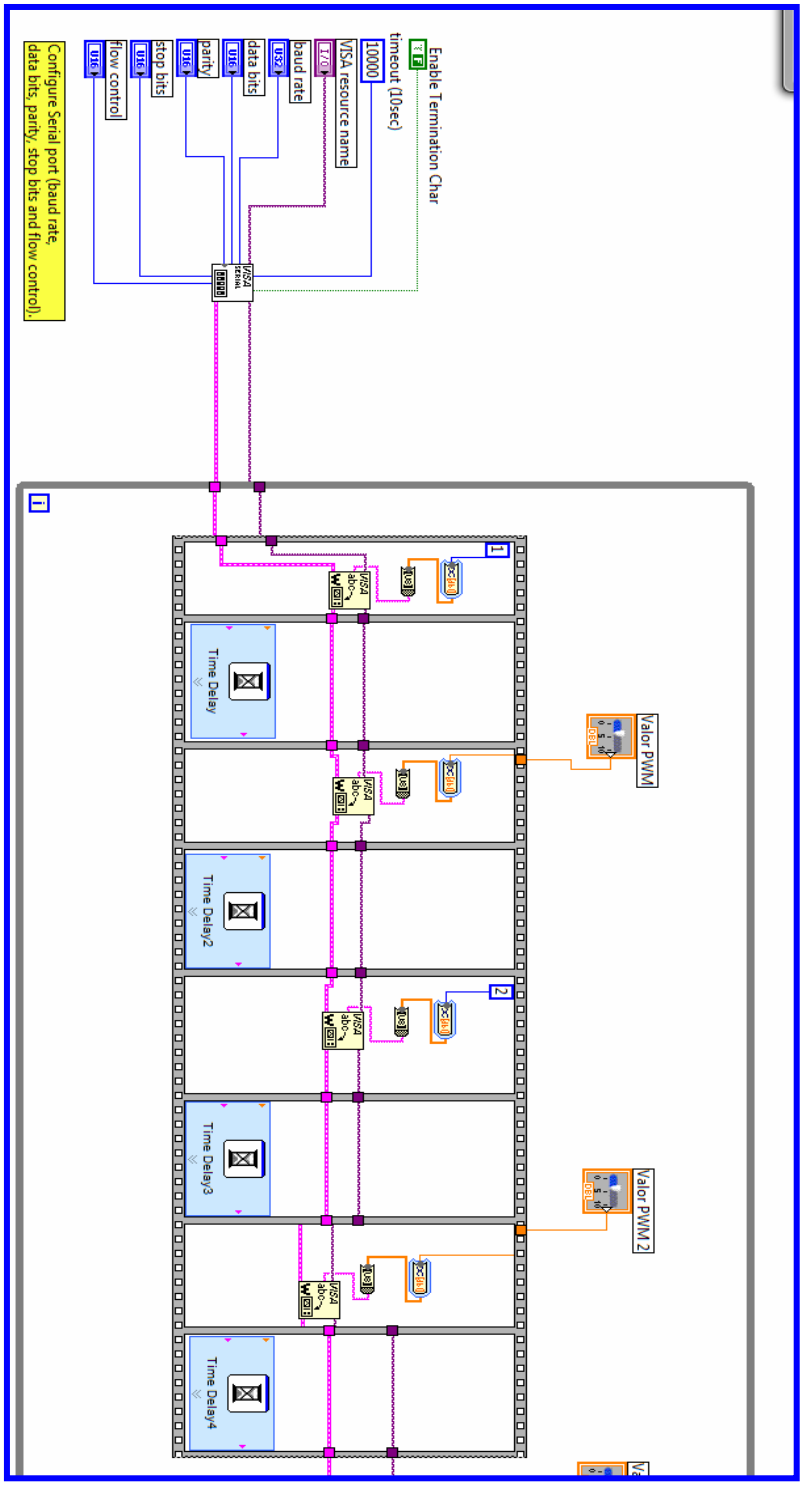


Figura 4.133 Diagrama a Bloques.

12.- Realizar las conexiones de las herramientas en el Diagrama a Bloques.

En la siguiente Figura 4.134 podemos observar las conexiones hechas en el diagrama a bloques en LabView para realizar la comunicación con el microcontrolador y manipular los PWM.



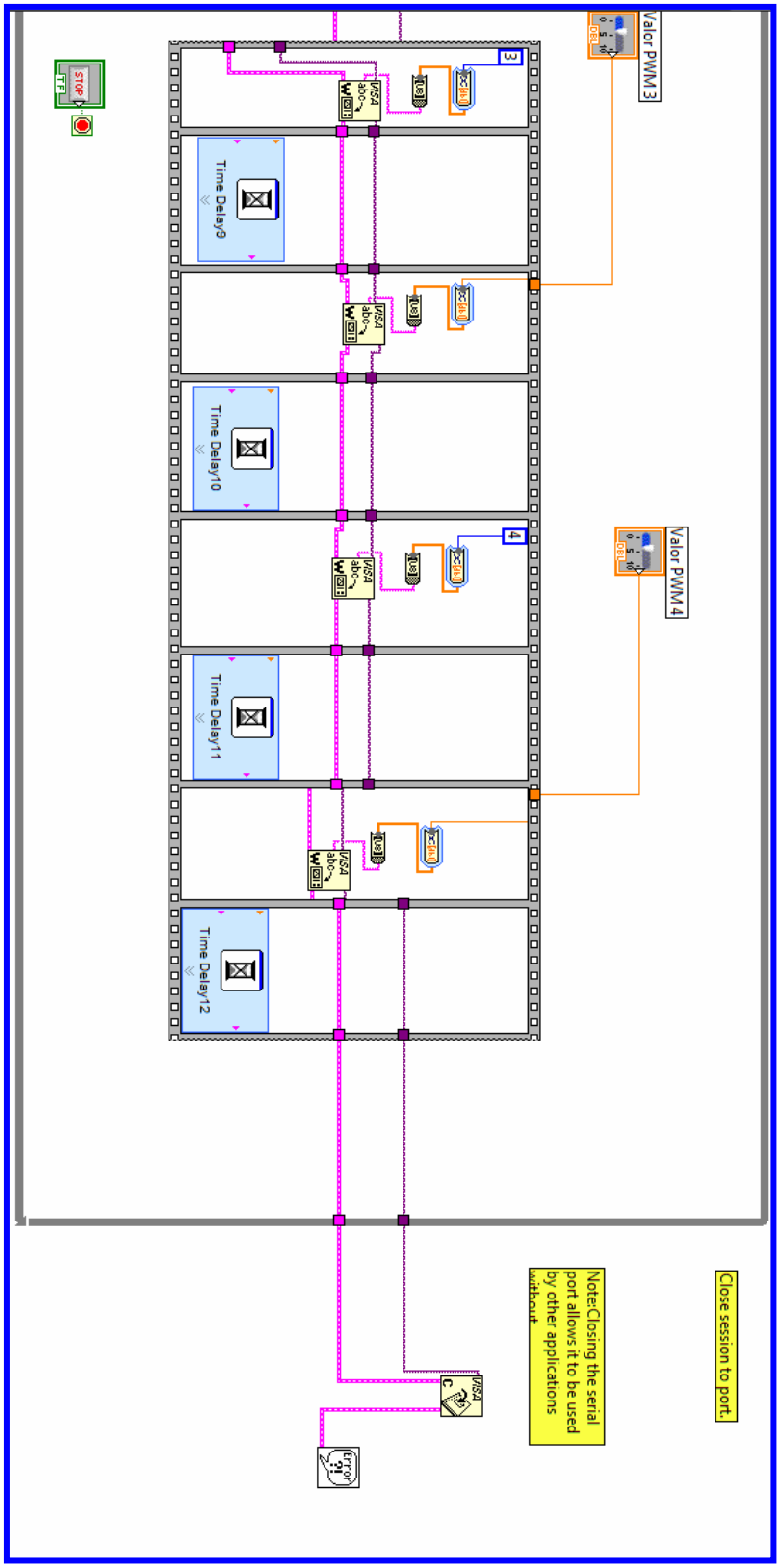


Figura 4.134 Diagrama a bloques de las funciones utilizadas.

12.- En esta parte se muestra el servo motor que es manipulado en esta práctica. La salida de cada PWM será conectada a cada entrada de control de servo motor.

En la siguiente Figura 4.135 muestra el servo motor de 5v de corriente directa:

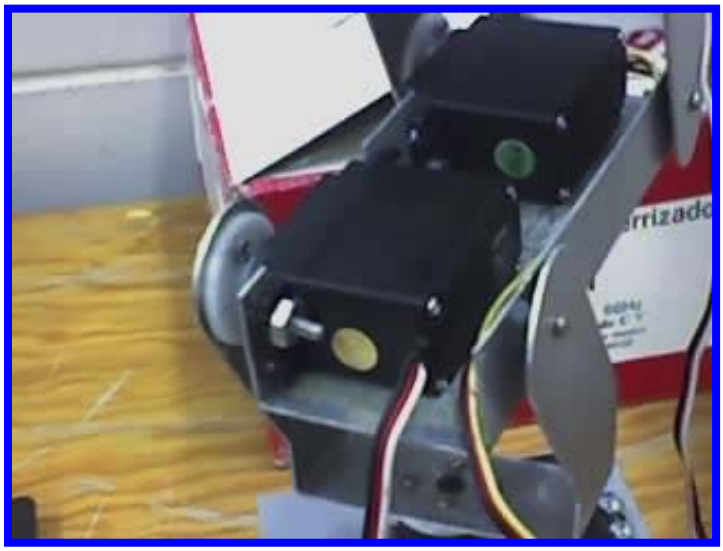


Figura 4.135 Servo Motor.

13.- En esta parte se muestra el servo robot manipulado en esta práctica.

Figura 4.136 Servo Robot con 4 grados de libertad.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PRÁCTICA NO. 14

NOMBRE: Manejo del LCD.

OBJETIVO: Que el usuario realice un código el cual inicialice el puerto de salida LCD y despliegue un mensaje de bienvenida.

METODOLOGÍA:

1.- Conectar el microcontrolador a una fuente regulada de 12 v de C.D.

2.- Realizar una conexión de los puertos seriales del microcontrolador y el CPU de la PC.

3.- Ejecutar el programa *Metrowersks CodeWarrior IDE* en el menú Inicio.

4.- Crear un nuevo archivo.

5.- Para poder inicializar el LCD es necesario dar de alta tres beans de un bit para controlar una señal de *Enable*, una para indicar *lectura/escritura* y otro para seleccionar el *registro*.

6.- Elegir en la ventana Bean Selector (Figura 4.137).- *CPU Internal Peripherals-Port I/O-BitIO. Habilitamos tres beans de un bit.*

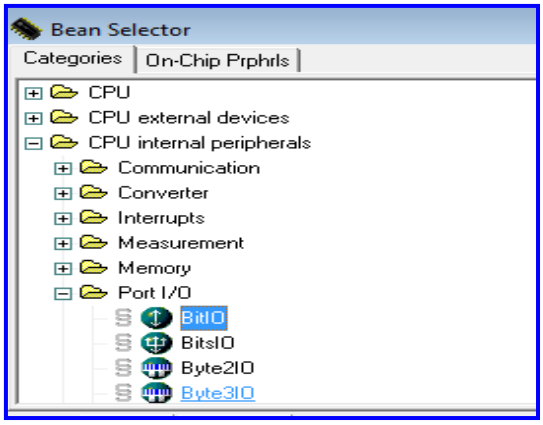


Figura 4.137 Habilitación del bean de un bit.

7.- Habilitar tres beans de un bit configurados como entrada. En la Figura 4.139 muestra la configuración del primer bean de un bit.

Los pines de conexión que utiliza el display es el puerto B. Este diagrama puede ser observado en la Figura 4.138. El primer bean se renombro como RS.

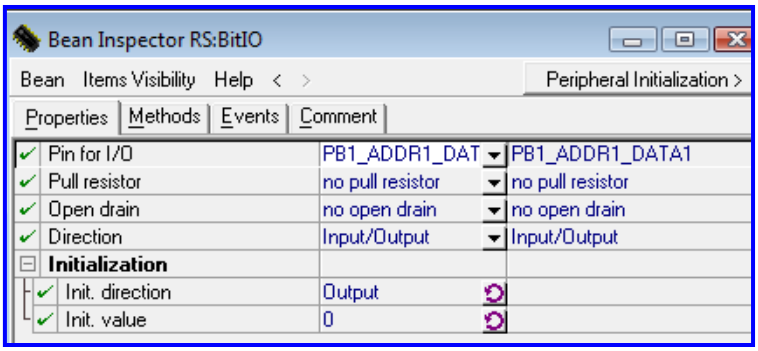


Figura 4.138 Configuración del bean de un bit.

El segundo y tercer bean se configuraron de la misma forma pero utilizando el segundo y tercer pin del puerto B (PB2_ADDR1_DATA1 y PB3_ADDR2_DATA2).

8.- Dar de alta un bean de 1-8 bits en la ventana de Bean Selector. En la Figura 4.139 se muestra la habilitación del bean de 1-8 bits.

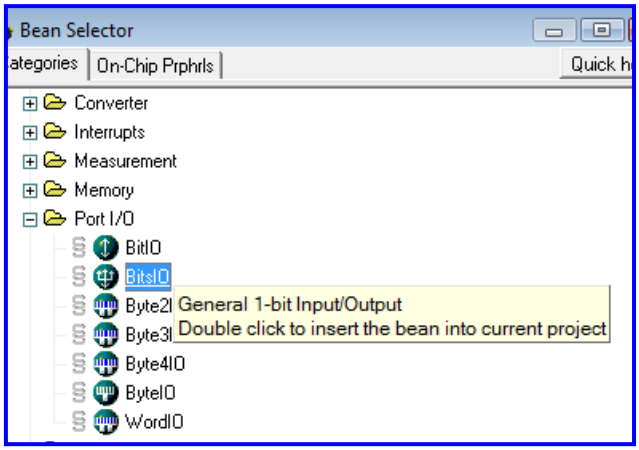


Figura 4.1139 Habilitación del bean de 1-8 bits.

Este bean se renombrará como *Datos.* Y funcionará como entrada de la señal de los datos de cuatro bits que utiliza el display.

9.- Configurar el bean  como se muestra en la Figura 4.140.

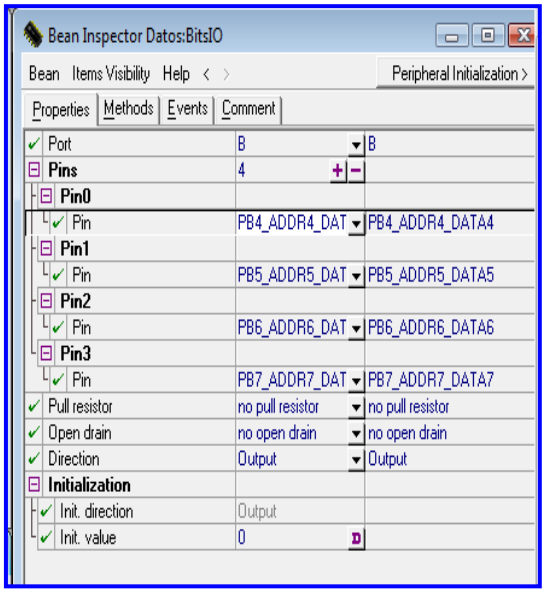


Figura 4.140 Configuración del bean Datos.

10.- Ver las funciones necesarias para realizar el código para inicializar y utilizar el LCD. En la Figura 4.141 se muestra los beans utilizados y los nombres que se les dio.

Figura 4.141 Beans utilizados para el uso del LCD.

11.- Realizar el código del programa que permita inicializar el LCD, manejar un retardo, escribir un comando, escribir un dato, desplegar un carácter, etc.

Además de configurar todas las funciones mencionadas anteriormente desplegaremos un mensaje de bienvenida en el display “microcontroladores”.

```
/* Including used modules for compiling procedure */
#include "Cpu.h"
#include "RS.h"
#include "RW.h"
#include "Enable.h"
#include "Datos.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

#define BORRAR_DISP             0x01
#define DISP_RETURN_HOME   0x02
#define RENGLON_DOS            0xC0


void retardo(unsigned int mseg)
{
  unsigned int i,j;
                    for(i=1; i<mseg;i++)
                     for(j=1;j<4159;j++);
}

void ESC_LCD(unsigned short val)
{
   unsigned int i;
   Datos_PutVal(val);
   Enable_SetVal();
   for(i=0;i<300;i++);
   Enable_ClrVal();

}

void LCD_INI()
{
```

```
    Enable_ClrVal(); //desactivado para transferencia
    RW_ClrVal();     //escritura
    RS_ClrVal();     //seleccionar registro de control


    retardo(100);
    ESC_LCD(0x2);               //Bus de datos a 4 bits
    retardo(100);


    ESC_LCD(0x2);  //Se confirma transferencia a 4 bits
    retardo(100);
    ESC_LCD(0x8);

    retardo(100);


    ESC_LCD(0x0);               //Incrementar contador de direcciones
    retardo(100);
    ESC_LCD(0x6);               //Display quieto
    retardo(100);

    ESC_LCD(0x0);     //Display ON
    retardo(100);
    ESC_LCD(0xE);               //Cursor ON

    ESC_LCD(0x00);     //         Borrar display
    retardo(30);
    ESC_LCD(0x01);              //
    retardo(100);


}
void ESCRIBIR_COMANDO(unsigned short val)
{
    unsigned short aux;
    Enable_ClrVal(); //desactivado para transferencia
    RW_ClrVal();     //escritura
    RS_ClrVal();     //seleccionar registro de control

    aux = val>>4;

    ESC_LCD(aux);   //envia parte alta del valor
    aux = val & 0x0F;

    ESC_LCD(aux);  //envia la parte baja

    retardo(100);



}
void ESCRIBIR_DATO(unsigned short val)
{
     unsigned short aux;
     RW_ClrVal();     //escritura
     RW_SetVal();     //Seleccionar registro de datos

     aux = val>>4;

     ESC_LCD(aux);   //envia parte alta del valor
     aux = val & 0x0F;

     ESC_LCD(aux);  //envia la parte baja

     RS_ClrVal();   //regresa
     retardo(100);

}
void DESPLIEGA_L1(char *cad)
{
  unsigned int i=0;
  char ch;
```

```
  ch = cad[i];

  ESCRIBIR_COMANDO(DISP_RETURN_HOME);

  RS_SetVal();  //configura registro de datos
  while(i < 16 && ch !=0)
  {
     ESCRIBIR_DATO(ch);
                        i++;
     ch = cad[i];

  }

  RS_ClrVal();

}

void DESPLIEGA_L1_CONT(char *cad)
{
  unsigned int i=0;
  char ch;
  ch = cad[i];

  RS_SetVal();   //configura registro de datos
  while(i < 16 && ch !=0)
  {
     ESCRIBIR_DATO(ch);
                        i++;
     ch = cad[i];

  }

  RS_ClrVal();


}

void DESPLIEGA_L2(char *cad)
{
  unsigned int i=0;
  char ch;
  ch = cad[i];

  // Poner el curso en la posición uno
  ESCRIBIR_COMANDO(DISP_RETURN_HOME);

  ESCRIBIR_COMANDO(RENGLON_DOS);

  //Poner el cursor en el renglon dos

  RS_SetVal(); //configura registro de datos
  while(i < 16 && ch !=0)
  {

                        ESCRIBIR_DATO(ch);
                        i++;
     ch = cad[i];

  }
   RS_ClrVal();

}

void main(void)

{
  unsigned int dato;
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
```

```
LCD_INI();

DESPLIEGA_L1 ( “microcontrolador”);

 /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
 for(;;){}
 /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
```

12.- Realizar las conexiones del LCD en el microcontrolador sis9s12E.

En la siguiente Figura 4.142 se muestra el LCD conectado en el microcontrolador sis9s12E.

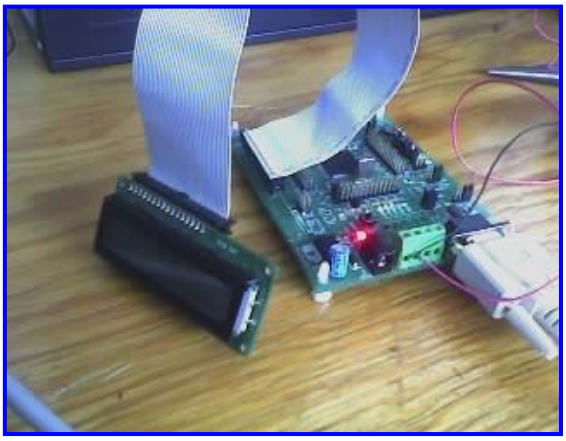


Figura 4.142 Pines de conexión del LCD.

UNIVERSIDAD AUTONOMA DEL ESTADO DE HIDALGO
INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
MANUAL DE PRÁCTICAS
Sistema de Desarrollo sis9s12E

PROPUESTA DE APLICACIÓN FINAL

El uso de un recurso tan potente como el sistema de desarrollo (SIS9S12E) y su fácil manejo mediante un compilador como CodeWarrior permitirá darle diversas aplicaciones en áreas de ingeniería. En el Centro de Investigaciones Avanzadas de Ingeniería Industrial (CIAII) se desarrollan proyectos que tienen como base el uso de este sistema, por ejemplo el Control de velocidad con sintonización ajustable de un PID digital a través de un teclado matricial e interfaz visual usando el sistema SIS9S12E.

NOMBRE: Propuesta de aplicación final: "Control de velocidad con sintonización ajustable de un PID digital a través de un teclado matricial e interfaz visual usando el sistema de desarrollo SIS9S12E para un conjunto par motor-generador de CD"[42].

OBJETIVO: Implementar y analizar el comportamiento de un control de velocidad usando un PID digital variando las constantes en un conjunto par motor-generador de CD con el uso del sistema de desarrollo SIS9S12E.

Objetivos específicos

- Realizar la programación para implementar un teclado matricial para ajustar las variables.
- Programar el microcontrolador para manejar un LCD de 2x16 líneas.

[42] David Domínguez López "Control de velocidad con sintonización ajustable de un PID digital a través de un teclado matricial visual usando el sistema de desarrollo SIS9S12E para un conjunto par motor-generador de CD", Tesis para obtener el grado de Licenciado en Ingeniería Electrónica y Telecomunicaciones. Universidad Autónoma del Estado de Hidalgo, Pachuca Hidalgo, 2007.

- Adquirir la señal del sensor de velocidad.
- Configurar una salida para conseguir la señal PWM para alimentar el motor.
- Implementar y programar una ecuación en diferencias en el microcontrolador.

Otro objetivo de esta propuesta es la de fomentar y hacer interesante la materia de control, ya que el alumno podrá sintonizar un PID digital por cualquier método e introducir el valor de las constantes y observar el comportamiento de la respuesta, así como introducir un microcontrolador de gran capacidad usando el Codewarrior que es un lenguaje de programación en c y compilador.

En esta propuesta se intentará desarrollar un control proporcional integral y derivativo, en el cual se modifican dichas constantes a demás del tiempo discreto a través de un teclado matricial conectado al mismo sistema de desarrollo SIS9S12E, los datos que se ingresan en el microcontrolador son mostrados en un visualizador de cristal liquido así como la velocidad del motor cuando este ya esta funcionando.

METODOLOGIA.

Revisión bibliográfica: se revisara la bibliografía referente al manejo del display convertidores analógico-digital y digital-analógico y manejo de la modulación por ancho de pulso.

Especificaciones: debido a que se requiere controlar una planta específica, se debe tener en cuenta las características de esta, como son velocidad máxima del motor, voltajes y corrientes.

Diseño de un diagrama a bloques: para el manejo del display de cristal líquido, el teclado, las acciones de control y las señales a convertir.

Se manejara cada dispositivo por separado, probando el correcto funcionamiento de cada bloque.

Se crea un Nuevo proyecto en Code Warrior para el manejo del LCD, teclado, y el uso de los convertidores de señal (DAC y ADC).

Se programara una ecuación en diferencias en el microcontrolador MC9S12E para controlar el motor.

Se harán las pruebas pertinentes para determinar la fiabilidad del sistema.

CONCLUSIONES

En este proyecto se hizo una propuesta de un manual de prácticas del sistema de desarrollo sis9s12E de la familia de Motorota MC9s12E.

En el manual se manejaron diversos recursos del microcontrolador como el Timer, DAC, ADC, PWM, puertos de entrada- salida, etc. Además se muestran las especificaciones y características que maneja este sistema, con el objeto de facilitar el manejo de los recursos anteriormente mencionados.

La elaboración de esta tesis permite aprender el uso de diversos recursos; como el manejo de un lenguaje de programación (lenguaje C), un lenguaje de programación gráfico (Lab View), el manejo de un sistema de monitoreo de microcontroladores (Niplesoft) y como parte esencial, la manipulación del microcontrolador por medio de un sistema de desarrollo integrado como CodeWarrior.

Un punto importante al realizar este trabajo fue mostrar los mapas de especificaciones que maneja este sistema de desarrollo SIS9S12E, y también se explica de forma amplia y especifica los recursos de software que se manejaron, estos puntos mencionados facilitaron el uso del sistema y el desarrollo del manual de prácticas que es el objetivo principal de dicho proyecto.

El manejo de la gran parte de los recursos del microcontrolador SIS9S12E nos permitió alcanzar diversos objetivos planteados como:

- La manipulación de los puertos de entrada y salida del microcontrolador.
- El manejo del Timer que se presenta como recurso el microcontrolador.

- La manipulación de los convertidores que maneja el sistema: ADC y DAC.
- La utilización del recurso de modulación por ancho de pulso PWM.
- Que se tenga un manejo óptimo del puerto de salida LCD.

Por otra parte, se da el uso combinado de estos recursos manipulados para desarrollar diversas aplicaciones como generar funciones, el control de servomotores, la selección de señales y el desfasamiento de estas.

ANEXOS

1.- Metodología Desarrollada.

Para la realización del proyecto de tesis se siguieron las siguientes etapas:

- Se identificaron las características necesarias para el manejo de los recursos que utilizó el sistema.
- Se da la identificación y manipulación de los puertos de entrada-salida.
- Se realizaron diversas prácticas con el uso de los dispositivos de entrada-salida.
- Se manipuló el Timer-PWM, permitiendo desarrollar la segunda práctica.
- Se manipuló el convertidor analógico- digital (ADC), y se desarrolla la tercera práctica.
- Se identificaron las especificaciones y se manipuló la comunicación serie. Permitiendo realizar la cuarta práctica y el manejo de un sistema de monitoreo de microcontroladores como Niplesoft.
- Se une el manejo de los recursos ya desarrollados para generar una función de dientes de sierra, seno, coseno, exponencial, doble exponencial o exponencial invertida, así como el desfasamiento de señales por medio del potenciómetro.
- Se aprende la utilización del lenguaje de programación grafico LabView y se desarrolla la práctica quince: Control de un servo robot.
- Se da la identificación de las especificaciones del manejo del LCD y se realiza la última práctica.

2.- Aportaciones

- Traducción de especificaciones del sistema de desarrollo sis9s12E perteneciente a la familia de Motorota Mc9s12E.
- Traducción de características y especificaciones del sistema de desarrollo integrado Code Warrior.

- Desarrollo de prácticas que nos permitirán aprender de forma rápida y sencilla el manejo de recursos como PWM, DAC, ADC, Comunicación Serial, sistemas de entrada salida, etc.
- Desarrollo de un conjunto de programas que nos permitió generar diversas señales (triangular, dientes de sierra, seno, coseno, exponencial, etc.).
- Explicación sobre el manejo de un sistema de monitoreo de microcontroladores Niplesotf.
- Desarrollo del control de un servo robot de cuatro grados de libertad con el uso de los recursos del microcontrolador.
- Desarrollo de programas que permitieron el uso de LCD.

3.- Desarrollo de trabajos futuros

Al desarrollar este manual permitirá al usuario un rápido entendimiento y manejo de los recursos del microcontrolador lo cual permitirá el desarrollo de trabajos futuros.

Para trabajos futuros se propone lo siguiente:

- Discretización de señales con el uso del convertidor analógico digital.
- Desarrollo de un generador de funciones con el uso del microcontrolador.
- Diseño de un sistema difuso de velocidad en un conjunto de motor-generador de CD implementado con el sistema de desarrollo sis9s12E.
- Desarrollo de un carro inteligente utilizando microcontroladores.
- Desarrollo de velocímetros con el uso de los recursos del microcontrolador.

BIBLIOGRAFÍA

1.- Angulo Usategui, José María y Angulo Martínez Ignacio, Microcontroladores PIC. Diseño práctico de aplicaciones, McGraw-Hill, pp 15-37.

2.- MacKenzie, I. Scott., The 8051 microcontroller / I. Scott MacKenzie., 3rd ed., Upper Saddler River, N.J. : Prentice Hall, pp 37-45.

3.- David Domínguez López "Control de velocidad con sintonización ajustable de un PID digital a través de un teclado matricial visual usando el sistema de desarrollo SIS9S12E para un conjunto par motor- generador de CD", Tesis para obtener el grado de Licenciado en Ingeniería Electrónica y Telecomunicaciones. Universidad Autónoma del Estado de Hidalgo, Pachuca Hidalgo, 2007.

4.- David Rodríguez González, "Realización de un conjunto de prácticas basadas en el microcontrolador PIC16F84", en la Especialidad Electrónica Industrial, I.T.T.

5. - Code Warrior IDE Quichstart manual.

6.- Antonio Manuel Lázaro, LabVIEW Programación Gráfica para el Control de Instrumentación. Sexta edición, Thomson, pp 11-57.

7.- Ayuda de LabVIEW.

8.- LabVIEW Bookshelf National Instruments 2000-2004.

9.- Hoja de datos del sistema MC9S12E.
http://www.freescale.com/webapp/sps/site/homepage.jsp.pdf

10.- Hoja de datos de Code Warrior.
http://www.ciao.es/Code_Warrior_273990_3.PDF

11.- Hoja de datos del manual de usuario del sistema sis9s12E.
http://www.racom.com.mx/desarrollo.php.pdf

12.- Hoja de esquemáticos SIS9S12E.
http://www.metrowerks.com

13.- Hoja de datos del manual de Motorota (MC9S12E).
http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC9S12E64&nodeId=016246844976630029.pdf

14.- Hoja de datos de sis9s12E.
http://www.racom.com.mx/admin/uploads/pdfs/desarrollolsis9s12E10_oct_2005.pdf

15.- Hoja de datos del sistema de monitoreo de microcontroladores Niplesoft.
http://www.niplesoft.net/

16.- Hoja de datos de Processor Expert.
http://www.processorexpert.com/

17.- Ayuda de Processor Expert.

18.- Hoja de datos del Motorota.
http://www.motorola.com/mx/products

Apéndice A

MAPA DE MEMORIA

A.1 MAPA DE LA DIVISIÓN DE MEMORIA[43]

En la Figura A.1 muestra el mapa de registro de los dispositivos de la familia MC9S12E, este mapa de registros es utilizado en dispositivos como el sistema de desarrollo sis9s12E.

Dirección	Modulo	Tamaño
$000 - $017	CORE (Ports A, B, E, Modes, Inits, Test)	24
$018	Reserved	1
$019	Voltage Regulator (VREG)	1
$01A - $01B	Device ID register (PARTID)	2
$01C - $01F	CORE (MEMSIZ, IRQ, HPRIO)	4
$020 - $02F	CORE (DBG)	16
$030 - $033	CORE (PPAGE, Port K)	4
$034 - $03F	Clock and Reset Generator (PLL, RTI, COP)	12
$040 - $06F	Standard Timer 16-bit 4 channels (TIM0)	48
$070 - $07F	Reserved	16
$080 - $0AF	Analog to Digital Converter 10-bit 16 channels (ATD)	48
$0B0 - $0C7	Reserved	24
$0C8 - $0CF	Serial Communications Interface 0 (SCI0)	8
$0D0 - $0D7	Serial Communications Interface 1 (SCI1)	8
$0D8 - $0DF	Serial Peripheral Interface (SPI)	8
$0E0 - $0E7	Inter IC Bus	8
$0E8 - $0EF	Serial Communications Interface 2 (SCI2)	8
$0F0 - $0F3	Digital to Analog Converter 8-bit 1-channel (DAC0)	4
$0F4 - $0F7	Digital to Analog Converter 8-bit 1-channel (DAC1)	4
$0F8 - $0FF	Reserved	8
$100- $10F	Flash Control Register	16
$110 - $13F	Reserved	48
$140 - $16F	Standard Timer 16-bit 4 channels (TIM1)	48
$170 - $17F	Reserved	16
$180 - $1AF	Standard Timer 16-bit 4 channels (TIM2)	48
$1B0 - $1DF	Reserved	48
$1E0 - $1FF	Pulse Width Modulator 8-bit 6 channels (PWM)	32
$200 - $23F	Pulse Width Modulator with Fault 15-bit 6 channels (PMF)	64
$240 - $27F	Port Integration Module (PIM)	64
$280 - $3FF	Reserved	384

Figura A.1 Mapa de Memoria.

[43] http://www.metrowerks.com

En las siguientes figuras se muestra el mapa completo de memoria de dispositivos con Flash y RAM. En este caso se muestra el mapa de memoria de sistema de desarrollo sis9s12E de la familia MC9S12E, el cual maneja una memoria RAM de 8K Bytes.

Este mapa de memoria es diferente para cada sistema de desarrollo de la familia MC9S12E, hay sistemas que manejan memoria de 2k bytes de memoria RAM y otros sistemas maneja hasta 16k Bytes de RAM [10].

La Figura A.2 muestra un mapa de memoria de 8K bytes de memoria.

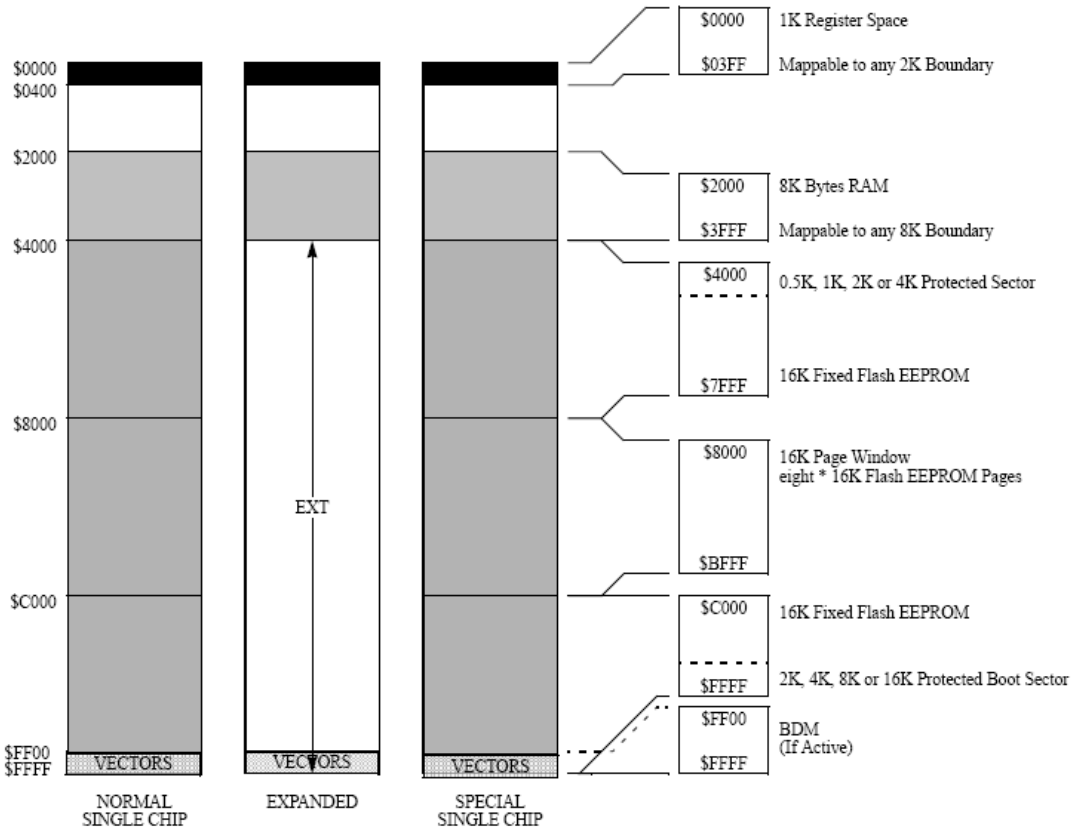


Figura A.2 Mapa de usuario de configuración de memoria (8k Bytes de RAM).

$0000 - $03FF: Espacio de Registro

$0000 - $3FFF: 16K RAM (solo 15k de RAM visible $0400-$3FFF).

La Figura A.2 muestra un mapa útil, no es un mapa que debemos no tomar en cuenta.

A.2 Propiedades de la Señal.

Pin Name Function 1	Pin Name Function 2	Pin Name Function 3	Power Domain	Internal Pull Resistor		Description
				CTRL	Reset State	
EXTAL	—	—	VDDPLL	NA	NA	Oscillator pins
XTAL	—	—	VDDPLL	NA	NA	
XFC	—	—	VDDPLL	NA	NA	PLL loop filter pin
RESET	—	—	VDDX	None	None	External reset pin
BKGD	MODC	TAGHI	VDDX	Up	Up	Background debug, mode pin, tag signal high
TEST	VPP	—	NA	NA	NA	Test pin only
PAD[15:0]	AN[15:0]	KWAD[15:0}	VDDX	PERAD/ PPSAD	Disabled	Port AD I/O Pins, ATD inputs, keypad Wake-up
PA[7:0]	ADDR[15:8]/ DATA[15:8]	—	VDDX	PUCR	Disabled	Port A I/O pin, multiplexed address/data
PB[7:0]	ADDR[7:0]/ DATA[7:0]	—	VDDX	PUCR	Disabled	Port B I/O pin, multiplexed address/data
PE7	NOACC	$\overline{\text{XCLKS}}$	VDDX	Input	Input	Port E I/O pin, access, clock select
PE6	IPIPE1	MODB	VDDX	While RESET is low: Down		Port E I/O pin, pipe status, mode selection
PE5	IPIPE0	MODA	VDDX	While RESET is low: Down		Port E I/O pin, pipe status, mode selection
PE4	ECLK	—	VDDX	PUCR	Mode Dep[1]	Port E I/O pin, bus clock output
PE3	$\overline{\text{LSTRB}}$	$\overline{\text{TAGLO}}$	VDDX	PUCR	Mode Dep(1)	Port E I/O pin, low strobe, tag signal low
PE2	R/$\overline{\text{W}}$	—	VDDX	PUCR	Mode Dep(1)	Port E I/O pin, R/$\overline{\text{W}}$ in expanded modes
PE1	$\overline{\text{IRQ}}$	—	VDDX	PUCR	Up	Port E input, external interrupt pin
PE0	$\overline{\text{XIRQ}}$	—	VDDX	PUCR	Up	Port E input, non-maskable interrupt pin
PK[7]	ECS	ROMCTL	VDDX	PUCR	Up	Port K I/O Pin, Emulation Chip Select
PK[6]	XCS	—	VDDX	PUCR	Up	Port K I/O Pin, External Chip Select
PK[5:0]	XADDR[19:14]	—	VDDX	PUCR	Up	Port K I/O Pins, Extended Addresses
PM7	SCL	—	VDDX	PERM/ PPSM	Up	Port M I/O Pin, IIC SCL signal
PM6	SDA	—	VDDX	PERM/ PPSM	Up	Port M I/O Pin, IIC SDA signal
PM5	TXD2	—	VDDX	PERM/ PPSM	Up	Port M I/O Pin, SCI2 transmit signal
PM4	RXD2	—	VDDX	PERM/ PPSM	Up	Port M I/O Pin, SCI2 receive signal
PM3	—	—	VDDX	PERM/ PPSM	Disabled	Port M I/O Pin, IIC SDA signal
PM1	DAO1	—	VDDX	PERM/ PPSM	Disabled	Port M I/O Pin, DAC1 output
PM0	DAO0	—	VDDX	PERM/ PPSM	Disabled	Port M I/O Pin, DAC0 output
PP[5:0]	PW0[5:0]	—	VDDX	PERP/ PPSP	Disabled	Port P I/O Pins, PWM output

Figura A.3 Propiedades de Señal.

La Figura A.3 muestran las propiedades de la señal.

Pin Name Function 1	Pin Name Function 2	Pin Name Function 3	Power Domain	Internal Pull Resistor		Description
				CTRL	Reset State	
PQ[6:4]	$\overline{IS}$[6:4]	—	VDDX	PERQ/ PPSQ	Disabled	Port Q I/O Pins, $\overline{IS}$[6:4] input
PQ[3:0]	FAULT[3:0]	—	VDDX	PERQ/ PPSQ	Disabled	Port Q I/O Pins, Fault[3:0] input
PS7	$\overline{SS}$	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SPI SS signal
PS6	SCK	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SPI SCK signal
PS5	MOSI	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SPI MOSI signal
PS4	MISO	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SPI MISO signal
PS3	TXD1	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SCI1 transmit signal
PS2	RXD1	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SCI1 receive signal
PS1	TXD0	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SCI0 transmit signal
PS0	RXD0	—	VDDX	PERS/ PPSS	Up	Port S I/O Pin, SCI0 receive signal
PT[7:4]	IOC1[7:4]	—	VDDX	PERT/ PPST	Disabled	Port T I/O Pins, timer (TIM1)
PT[3:0]	IOC0[7:4]	—	VDDX	PERT/ PPST	Disabled	Port T I/O Pins, timer (TIM0)
PU[7:6]	—	—	VDDX	PERU/ PPSU	Disabled	Port U I/O Pins
PU[5:4]	PW1[5:4]	—	VDDX	PERU/ PPSU	Disabled	Port U I/O Pins, PWM outputs
PU[3:0]	IOC2[7:4]	PW1[3:0]	VDDX	PERU/ PPSU	Disabled	Port U I/O Pins, timer (TIM2), PWM outputs

Figura A.4 Propiedades de Señal.

En la Figura A.3 mostró una tabla de las propiedades de la señal que maneja el sistema de desarrollo sis9s12e.

El buffer de salida del puerto E permite el control de la señal puesto que es determinado por el registro PEAR y es de modo dependiente.

Por ejemplo, en especial el modo prueba RDWE= LSTRE=1 que habilita el buffer de salida PE e incapacita el pull-ups.

Las señales mostradas en cuadro anterior no están disponibles en el paquete de 80 pines.

Si los pines del puerto no son relacionados con los de afuera en el paquete escogido el usuario debería inicializar los registros para hacerlos entradas con la resistencia pull para evitar el exceso consumo de corriente.

Apéndice B

INTERFACE RS232

B.1 CANAL DE COMUNICACIÓN RS232[45].

El uso del interface RS232 es una parte fundamental para el manejo del sistema SIS9S12E ya que nos permite la comunicación entre la PC y la tarjeta.

La comunicación serie consiste básicamente en enviar datos de un transmisor (DTE) a un receptor (DCE) de manera sincronizada.

La tarjeta del sistema de desarrollo posee un conector DB9 para la comunicación serie.

En la Figura B.1, se puede ver la distribución de los pines del conector:

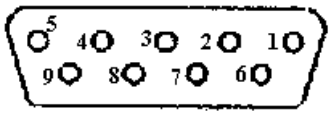


Figura B.1. Conector DB9 hembra

Cada pin del conector, tiene un número asociado que corresponde a:

Pin 1 (DCD): Detector de Portadora de Datos. Es utilizada por el DCE para indicar al DTE que está listo para transmitir datos.
Pin 2 (RxD): Recepción de Datos. Transmite datos del DCE al DTE.
Pin 3 (TxD): ransmisión de Datos. Transmite datos del DTE al DCE
Pin 4 (DTR): Terminal de Datos Listo. Para comunicar al DCE que el DTE está encendido y listo para funcionar.

[45] David Rodríguez González, “Realización de un conjunto de prácticas basadas en el microcontrolador PIC16F84”, en la Especialidad Electrónica Industrial, I.T.T.

Pin 5 (GND): Circuito común. Punto de referencia para todos los voltajes de la conexión.

Pin 6 (DSR): Dispositivo de Datos Listo. Comunica al DTE que el DCE está encendido y listo para funcionar.

Pin 7 (RTS): Petición de Envío. Sus usos varían ampliamente. Va del DTE al DCE.

Pin 8 (CTS): Dispuesto para Enviar. Sus usos varían ampliamente. Va del DCE al DTE.

Pin 9 (RING): No se suele utilizar.

En la siguiente Tabla B.1 se muestran las señales RS-232 más comunes según los pines asignados:

Señal		DE-9 (TIA-574)	EIA/TIA 561	Yost	RJ-50	MMJ
Common Ground	G	5	4	4,5	6	3,4
Transmitted Data	TD	3	6	3	8	2
Received Data	RD	2	5	6	9	5
Data Terminal Ready	DTR	4	3	2	7	1
Data Set Ready	DSR	6	1	7	5	6
Request To Send	RTS	7	8	1	4	-
Clear To Send	CTS	8	7	8	3	-
Carrier Detect	DCD	1	2	7	10	-
Ring Indicator	RI	9	1	-	2	-

Tabla B.1 Señales de la interface RS232.

La interfaz RS-232 está diseñada para distancias cortas, de unos 15 metros o menos, y para velocidades de comunicación bajas, de no más de 20 Kb. A pesar de ello, muchas veces se utiliza a mayores velocidades con un resultado aceptable. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En un canal simplex los datos siempre viajarán en una dirección, por ejemplo desde DCE a DTE. En un canal half duplex, los datos pueden viajar en una u otra dirección, pero sólo durante

un determinado periodo de tiempo; luego la línea debe ser conmutada antes que los datos puedan viajar en la otra dirección. En un canal full duplex, los datos pueden viajar en ambos sentidos simultáneamente.

Apéndice C

RECURSOS ESPECIALES DEL MICROCONTROLADOR MC9S12E64[47]

B.1 Temporizadores o “Timers”.

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores).

Para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las salidas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

B.2 Perro guardián o “Watchdog”

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

B.3 Estado de reposo ó de bajo consumo

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. Para ahorrar energía, (factor clave en los aparatos portátiles), el microcontrolador dispone de una instrucción especial, que le pasa al estado de reposo o de bajo consumo,

[47] http://www.motorola.com/mx/products

en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se “congelan” sus circuitos asociados, quedando sumido en un profundo “sueño” el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

B.4 Conversor A/D (ADC)

El microcontrolador incorpora el uso del Conversor A/D (Analógico/Digital) que pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del ADC diversas señales analógicas desde sus salidas.

B.5 Conversor D/A (DAC)

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por unos de los pines de la cápsula del microcontrolador.

B.6 Modulador de anchura de impulsos o PWM

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

B.7 Puertas de E/S digitales.

El microcontrolador destina algunas de sus pines a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertas.

Las líneas digitales de las Puertas pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

B.8 Puertas de comunicación.

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos.

Glosario

Compilador.- Un compilador acepta programas escritos en un lenguaje de alto nivel y los traduce a otro lenguaje, generando un programa equivalente independiente, que puede ejecutarse tantas veces como se quiera. Este proceso de traducción se conoce como compilación.

Comunicación SPI.- Comunicación serial sincrónica. Es un estándar establecido por Motorota que utiliza un bus de 4 líneas para interconectar dispositivos periféricos de alta y media velocidad.

Embedded Bean.- Es un componente que se puede utilizar en el Processor Expert.

Embedded Bean encapsula las funciones de los elementos básicos de sistemas encajados como los periféricos de la base del CPU, del chip del CPU, los periféricos independientes, los dispositivos virtuales y los algoritmos puros del software y envuelva estas instalaciones a las características, a los métodos, y a los eventos (como objetos en OOP). Las Beans pueden apoyar varios lenguajes (ASM, AnsiC, Modula y otros) y el código se genera en el lenguaje seleccionado.

Bean Inspector.- Ventana con todos los parámetros del Bean seleccionada: características, métodos, eventos.

Bean CPU.- Bean que encapsula la base del CPU. Los Beans del CPU se exhiben en el Bean Inspector, donde los parámetros del CPU se pueden modificar y mucho más.

Driver.- Beans driver. Los drivers son la base del proceso de generación del código del Processor Expert. El Processor Expert utiliza el driver para generar el código para conducir el periférico interno o externo que considera ajustes del Bean. El Bean puede encapsular uno o más drivers.

Events.- Una función de evento es utilizada para procesar un evento (ejemplo., una interrupción, un desbordamiento del almacenador intermediario) por código user-written apropiada del evento, que se supone (aunque no lo hace tuvo que necesariamente) para contener el código del usuario.

External User Module.- Código de fuente externo unido al proyecto del PE. El módulo externo del usuario puede consistir en dos archivos: puesto en práctica e interfaz.

Internal Peripherals.- Dispositivos internos de la CPU (puertos, timer, convertidores DE ANALÓGICO A DIGITAL etc. controlados generalmente por la base de CPU usando los registros especiales).

ISR.- Rutina del servicio de la interrupción - código, se llama que cuando ocurre una interrupción.

Memoria flash.- La memoria flash es un tipo de EEPROM que permite se borren o escriban múltiples localizaciones de memoria en una operación de programación. Las EEPROM normales, sólo permiten que se borre o escriba una localización cada vez.

Méthods. - Funciones o subrutinas accesibles del usuario. El usuario puede seleccionar que quiera que sean generadas y que no. Los métodos seleccionados serán generados durante el proceso de generación del código en los módulos del Bean.

Módule.- módulo del código de fuente. Podría ser generado por el Procesor Expert (módulos del Bean, módulo de la CPU, eventos.) o ser creado por el usuario y ser incluido en el proyecto (módulo del usuario).

OOP.- La programación orientada al objeto (OOP) fue inventada para solucionar ciertos problemas de la modulación y de la reutilidad que ocurren

cuando los lenguajes de programación tradicionales tales como C se utilizan para escribir usos.

PESL.- (Biblioteca del Processor Expert) se dedica a los programadores de la energía, que son familiares con la arquitectura del CPU, cada pin y cada registro. PESL proporciona las macros para tener acceso a los periféricos directamente, así que PESL se debe utilizar solamente en algunos casos especiales. Usted puede encontrar más información en la documentación de PESL.

Popup Menú.- Se exhibe este menú cuando el botón de ratón derecho se presiona en un cierto objeto gráfico.

Properties.- Parámetros del Bean. Los ajustes de la característica definen que los periféricos internos serán utilizados por el Bean y también la inicialización y el comportamiento del Bean en el tiempo de rutina.

Puerto serial RS232.- Un **puerto serie** es una interfaz de comunicaciones entre ordenadores y periféricos en donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits a la vez. Entre el puerto serie y el puerto paralelo, existe la misma diferencia que entre una carretera tradicional de un sólo carril por sentido y una autovía con varios carriles por sentido.

Sistema operativo CP/M.- *Programa de control para monitores.* fue el primer sistema operativo que podía ejecutarse en PCs de diferentes fabricantes.

Target CPU.- El derivado del CPU usado en un proyecto dado.

Template.- La plantilla del Bean es un bean con parámetros preestablecidos. Plantilla definida por el usuario del bean. La plantilla definida por el usario del bean es un bean con los parámetros preestablecidos

ahorrados bajo nombre seleccionado. También el nombre del autor y de la descripción corta se puede agregar a la plantilla.

User Module.- Módulo del código de fuente creado o modificado por el usuario. (Módulo principal, módulo del acontecimiento o módulo externo del usuario).

UARTs.- Las UARTs (**U**niversal **A**syncronous **R**eceiver **T**ransmitter) son circuitos integrados de la placa serie del PC. Su propósito es convertir los datos a bits, enviarlos a la línea serie, y después reconstruir los datos en el otro terminal. Las UARTs tratan los datos en bloques del tamaño de un byte, que además es, convenientemente, el tamaño de los caracteres ASCII.