Notas Técnicas de Uso y Aplicación

NT0001

1.1 COMO EMPEZAR A UTILIZAR EL MICROCONTROLADOR
GUÍA DE CÓMO LEER LAS NOTAS TÉCNICAS Y CONVENCIONES UTILIZADAS

Preparado por: Rangel Alvarado
Estudiante Graduando de Lic. en Ing. Electromecánica
Universidad Tecnológica de Panamá
Panamá, Panamá
e-mail: issaiass@cwpanama.net
web site: http://www.geocities.com/issaiass/

ÏNDICE

1.1.1 Introducción

En este capítulo se introduce el "¿Cómo empezar?" a utilizar los tomos, capítulos, notas de aplicación y todos los documentos generados sobre el microcontrolador Motorola 68HC908. En esta sección especial se lista como se implementó el trabajo, mediante convenciones de código y algunas excepciones para hacerlo legible; los documentos hablan por si mismos con más gráficos que texto para acompañar la aplicación, documento o nota técnica.

Adicionalmente cada nota tiene el código completo de estudio, por eso lo voluminoso de las notas. Si solamente se buscan generalidades, el autor recomienda leer la teoría, el programa principal, las referencias y ejecutar la aplicación. Por otro lado, si se busca total comprensión, el autor recomienda tener en cuenta las salvedades o referencias de las aplicaciones sobre los documentos básicos, pues no hay teoría sin práctica ni práctica sin teoría; todo lo que se explica en los documentos básicos tiene su relevancia en los documentos referenciados.

Cabe destacar, que mientras no se diga lo contrario, todos los programas tienen su núcleo en el microcontrolador 68HC908JL3, pero todos los documentos pueden ser enfocados a los microcontroladores JK3/QT4/QY4/GP32 y cualquier otro microcontrolador de la Familia HC08, que como es de sospechar, tienen el mismo CPU o Unidad Central de Procesamiento. Los programas fueron realizados para el microcontrolador JL3, con cristal externo de 4.9152 MHz.

Por último, destaca el autor, que todo el trabajo realizado no sería posible si no se hubieran hecho "códigos reusables" o rutinas que siempre se utilizan a lo largo de la creación de aplicaciones, estos códigos fueron realizados en lenguajes "Ensamblador" y "C", cuyo énfasis en la tesis es el lenguaje de bajo nivel, o sea "Ensamblador". Para los códigos del lenguaje en alto nivel, consultar el CD que se adjuntó al trabajo de grado.

1.1.2 El Objetivo

Este trabajo tiene como objetivo, ayudar a los programadores de sistemas embebidos o los neófitos en este tipo de tema (como lo fue alguna vez el autor), en empezar, ya sea a nivel universitario, acercando más material de referencia básico para la realización de sus proyectos, cuyo código puede ser extraído y ajustado sin problema ó, a nivel básico, hablando propiamente de la educación media, de transferir conocimientos que se consideran básicos y así agilizar la curva de aprendizaje además de influenciar a estudiantes a seguir las carreras ingenieriles, en este caso, orientadas con la "programación y control".

1.1.3 Orientación del Trabajo

El enfoque del libro es en parte de "Interfase al Mundo Exterior y Códigos Reusables", más sin embargo se hace gran énfasis en la parte básica, pues es a consideración del autor, donde se muestran más fallos; el ¿cómo arrancar? es y será siempre un problema que se necesita resolver y es en donde el autor enfoca su energía. No es fácil escribir para personas que se inicien y no tengan conocimiento alguno de programación y experiencia práctica, pero se trata de hacer el mejor esfuerzo de conseguir el objetivo y orientación.

1.1.4 Lenguajes Utilizados

A pesar de que el autor muestra solo la premisa de utilizar "Ensamblador", el CD adjunto al trabajo provee información que, a criterio del autor se considera invaluable, por las extensas "horas hombre" de trabajo continuo detrás del computador.

Dicho trabajo consta de los códigos portados al lenguaje ANSI C, pues muchos de los compiladores que existen en el mercado tienden a soportar la capacidad del C que conocemos. Aún más, los mismos compiladores soportan híbridos de programación de lenguajes entre bajo y alto nivel. El lenguaje C++ no es estudiado, pues a consideración del autor es ligeramente más complejo entender la OOP (programación orientada a objetos) sin tener un "background" en lenguaje C y conocimientos de las "estructuras".

No solo se portan los códigos de "Ensamblador" a "C", sino que se provee de una maravillosa y amplia gama de rutinas que harán que su trabajo, pasatiempo, tarea o aplicación, sea más fácil de implementar. A futuro, cada vez que una persona lea este trabajo, aunque no se use al 100%, se sabe que los mismos entenderán el concepto que se explica, "es más fácil trabajar con códigos reusables que reinventar

la rueda", después de todo, ¿Microsoft hace su Sistema Operativo Windows siempre desde cero?.

Libros como "*Embedded Systems Building Blocks*" de *Jean J. Labrosse*, demuestran que el paradigma anterior es siempre bueno si se toma uso de la reusabilidad de código.

1.1.5 El Dilema entre C y Ensamblador

A pesar de como se mencionó anteriormente, el uso del lenguaje ensamblador predomina en las notas técnicas, pues no se puede obviar que el uso del lenguaje ensamblador ahorra más tiempo del CPU en realizar operaciones, por el momento ud. no sabrá la diferencia, la única forma es realizar un “hands-on-experience” en cada nota técnica.

Es posible escribir todos los códigos en ensamblador, pero en la opinión personal del autor, si lo que se busca es la aplicación y no el tiempo de respuesta fino, preciso y al centavo, se recomienda utilizar mejor lenguajes de alto nivel, aún así, el lenguaje ensamblador no se debe obviar, pues todo apunta hacia ensamblador.

A criterio del autor, el lenguaje C ha cobrado importancia por:

- Portabilidad de Código
- Mantenimiento del Código
- El código es fácil de entender y de escribir
- Es independiente de la plataforma

El último criterio es el más importante pues de ahí cobra sentido hacer rutinas de uso frecuente, si se tuviese que migrar a otro microcontrolador en ensamblador, el código cambia drásticamente, por el tipo de instrucciones, mientras que el lenguaje ANSI, es un “estándar” y siempre será así. El mismo concepto aplicará entonces para otros dispositivos como VHDL, DSP y PCs. Siempre se busca la “organización”.

1.1.6 Convención de Código

El autor trata de hacer el código consistente, según los conocimientos del mismo. Todas las librerías (*.inc, *.h), ejemplo, la librería *LCD*, sus funciones, todas empezarán con LCD. Así, si una función *escribe una cadena en la fila-columna*, esta puede llamarse, en inglés, *LCDWrMsgXY*.

Es muy difícil mantener un código consistente, más sin embargo, el autor ha hecho todo lo posible por mantener la legibilidad del código. Un libro que se recomienda ampliamente para la estandarización del lenguaje, y que gusta a muchos programadores desde iniciadores hasta avanzados, es el ya sea, en el caso de C, es el libro de los autores *Ritchie & Kernighan*, “*The C Programming Language*”, el código que especifican es consistente y totalmente entendible, más sin embargo, no se recomienda este libro si el usuario es un primerizo en lenguaje C.

Para entender sobre códigos reusables, referirse a la Sección 1.1.8, *Programación Estructurada*.

1.1.7 Notas Técnicas

El núcleo de todo el trabajo, se ha dividido de manera tal enfocada a la parte básica intermedia, lo que se denomina como notas técnicas, que son complementos de información básica como decodificación de teclado, utilidad de LCD, siete segmentos y dispositivos comunes. La siguiente lista da una idea de la división de las notas técnicas

Tabla 5. Notas Técnicas Básicas

Nota Técnica	Título	Descripción
0001	Introducción	¿Cómo empezar a utilizar las notas?
0002	Sistemas Numéricos	Conversión entre sistemas numéricos
0003	Compuertas	Introducción al Álgebra booleana y la lógica secuencial y combinacional
0004	Componentes	Reconocimiento de los componentes de la TD68HC908
0005	Montaje	Procedimiento de Soldadura de la TD68HC908
0006	Software	Introducción al IDE de Programación WinIDE
0007	CPU08	Explicación de la función del CPU y un microcontrolador
0008	Mapa de Memoria	División del Mapa de Memoria del Microcontrolador
0009	Retardos	Cálculo de la subrutina de retardo de Software Delay
0010	Puertos E-S	Utilidad de los Puertos de E-S y activación de LEDs
0011	ADC – 1 Conv	Configuración del ADC y medición del Voltaje
0012	ADC – Conv Cont & Interrupciones	Conversión Continua, Interrupciones y medición de una entrada analógica dada por un potenciómetro

Tabla 6. Notas Técnicas Intermedias

Nota Técnica	Título	Descripción
0101	Timer – Temporizador	Temporizador de 1 segundo
0102	Timer – Output Compare	Tren de Pulsos de 5 milisegundos
0103	Timer – OC Buffered	Tren de Pulsos en modo Buffered
0104	Timer - PWM	Generación de Modulación Por Ancho de Pulso
0105	Timer – PWM Buff.	Generación por PWM tipo Buffer
0106	Timer – Input Capture	Captura de un evento externo de un Puerto de E/S hacia el IC
0107	Tarjeta Universal	Configuraciones de Programación con la TD68HC908
0108	KBI	Generación de un “Toogle” Switch
0109	IRQ1	Generación de una Interrupción externa
0110	TimeBaseModule	Generación de tiempo y Parpadeo de un LED
0111	Registro de Configuraciones	Diferentes estados de los registros de configuraciones para microcontroladores JK3/JL3/QT4/QY4/GP32
0112	AutoWakeUp	Simulación de Alarma y Equipo de Bajo Consumo

Tabla 7. Notas Técnicas Avanzadas

Nota Técnica	Título	Descripción
1001	CGMC	Cristal de 32MHz a base de un PLL y cristal externo de 4 MHz
1002	SCI	Comunicación Serial con la PC
1003	CodeWarrior	Introducción a la Programación en Lenguaje C
1004	LEGO	Interfase a Sensores y Motores LEGO con MC68HC908
1005	SSEG	Contador Ascendente a Base de un Siete Segmentos
1006	SSEGMUXD	Multiplexión de Pantallas con Siete Segmentos
1007	LED	Multiplexión de Pantallas y Control de Segmentos del Siete Segmentos
1008	LCD	Control de una Pantalla de Cristal Líquido
1009	KBD	Técnica de Decodificación de Teclado de 16 Teclas
1010	PSX	Adquisición de Señales de un Pad de Playstation
1011	ComSerial	Comunicación Serie Para Microcontroladores sin SCI

Todas las Notas Técnicas, están respaldadas por la página web, cuyo dominio se encuentra en el servidor gratuito de geocities, http://www.geocities.com/issaiass/. Cualquier duda, referencia o comentario, se puede referir al correo electrónico, issaiass@cwpanama.net, issaiass@yahoo.es ó issaiass@hotmail.com, escritos en orden de prioridad. Cualquier duda, respecto a estos temas, favor enviar un correo.

1.1.8 Programación Estructurada

1.1.8.1 Estructuración de Código

A medida que se procede a avanzar a realizar proyectos, si se es observador, se nota que todos los problemas recaen en lo mismo. Muchas veces se necesita escribir en una pantalla de cristal líquido (LCD), saber que tecla fue presionada para ejecutar una acción (Teclado), controlar motores con engranes reductores pues se necesita de mucho torque en el eje (Servos), aplicar rutinas de conversión de Decimal a ASCII, sensar temperatura (LM335), etc.

Para que el código no envuelva al lector en confusiones debe ser, o tratar de ser consistente, es por esto que se estipulan diversas reglas antes de programar, en este caso ensamblador.

1.1.8.2 El "Back Slash" en los Programas

Cada vez que inicie un documento principal, verá la siguiente línea de código, por lo menos hablando del lenguaje ensamblador.

$include '\FUNCTIONS\RAM.inc'

Todos los códigos de funciones están arrojados en una carpeta del mismo nombre "*FUNCTIONS*', pero lo peculiar es el camino de búsqueda (*PATH*), si se especifica "\", simboliza que en la carpeta donde reside el archivo principal, debe existir una carpeta *FUNCTIONS*, de lo contrario, el compilador, arrojará un error. *RAM.inc*, corresponde a la función incluida como archivo cabecera utilizado.

1.1.8.3 Cabecera de un Programa Principal

Todo programa principal, archivo de funciones, vectores de interrupción, etc.; debe tratar de ser organizado por su archivo cabecera, el cual especifica entre una de las primeras líneas, luego de los comentarios, el *Nombre del Archivo*; primeramente destacando la nota técnica número "n", el nombre, Día-Mes-Año.

```
;*****************************************************************
; ARCHIVO      :  NT1111 – MP3Player –  02 01 08.asm
; PROPÓSITO    :  Reproductor MP3 con microcontrolador.
; NOTAS        :  Ninguna
;
; REFERENCIA:  NT1111 – MP3Player – 02 01 08.doc
;
; LENGUAJE     :  IN-LINE ASSEMBLER
; -----------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   26   05   07   Creado.
;   02   01   08   Modificado.
;*****************************************************************
```

Se deben listar las salvedades o *Notas*, que harán aclaratoria sobre condiciones específicas de operación del documento. Seguidamente el documento de referencia o estudio, material que se deberá generar si se desea preservar viva la información. Se recomienda que se avance de esta manera pues es una de las más ordenadas.

Aspecto importante es especificar el tipo de lenguaje, en este caso ensamblador y llevar una revisión o historial de código de la fecha de creación y la última modificación del código existente.

1.1.8.4 Inclusión de Archivos Cabecera de Mapa de Memoria

Se debe especificar la línea de programación siguiente siempre, pero más importante es que hay en el archivo "*includes.equ*". El archivo "*includes.equ*" contiene la declaración del tipo de microcontrolador usado.

```
;*****************************************
;          Cabecera de Macros, Const. y Memoria
;*****************************************
$include '\MAP\includes.equ'              ; Definiciones de usuario y mapa de memoria
```

1.1.8.5 Includes.equ

Originalmente, el archivo "includes.equ", incluía definiciones del usuario y definiciones del correspondiente microcontrolador en uso, pero a fines de la implementación se eliminó la inclusión de etiquetas varias de usuario, pues, mistificaban el uso del archivo cabecera, aún así, el archivo "*mapjl3*", dentro de la cabecera de la sección 10.3, abarca que se debe realizar.

```
;===========================================
;                  Mapa de Memoria - JL3
;===========================================
$include     '\MAP\mapjl3.equ'  ; Inclusión de Localidades de Memoria Básicas
```

1.1.8.6 Programa principal

El programa principal especifica la iniciación del código, las primeras líneas RSP y BSET, son necesarias para el primerizo, pues causan reinicio de la pila del sistema y desactivación del módulo watchdog. El código puede ir escrito donde avisa el cómentario.

```
;===========================================
; OBJETIVO     :  Inicio de Codif. del Ensam-
;                 blador en Memoria FLASH.
;===========================================
        org FLASH_START                 ; Inicio Mem. FLASH

;===========================================
; OBJETIVO    :
;===========================================
START
        rsp                             ; Inic.Stack = $00ff
        bset BIT0,CONFIG1               ; Desactiva watchdog

        ; El código aquí
```

1.1.8.7 Incluyente de Funciones e Interrupciones

Las funciones son la parte más importante del sistema, con ellas realizamos los programas de manera rápida sin perder tiempo y concentrarnos en la aplicación

```
;===========================================
;            Declaración y definición de funciones
;===========================================
$include '\FUNCTIONS\includes.inc'      ; Incluye Funciones

;===========================================
;          Declaración y Definición de interrupciones
;===========================================
$include '\INTERRUPTS\interrupt.inc'    ; Incluye interrupciones del
                                        ; microcontrolador
```

Igualmente, de la misma manera que se trata de incluir en el mapa de memoria, la memoria del microcontrolador utilizada, ud. debe especificar la función a utilizar dentro de cada uno de los archivos cabecera. Para mayor información, ver el archivo comprimido *NT0026*, el cual *DEBE SER EXTRAÍDO* con sus subcarpetas.

1.1.8.8 Identificadores de Variables y Macros

Para mantener consistencia en el código se propone nombrar los identificadores que no son más que las variables, estructuras de variables, macros, etc. por medio de abreviaturas y mayúsculas, minúsculas.

Por ejemplo, un macro del puerto del microcontrolador será capaz de hacer que el programa sea flexible y cada vez nombrará el contenido del macro donde se ubique

```
LCD_DATA_PORT     equ     PORTA
```

Este macro nombra el puerto de datos de la LCD, así, de esta forma será más fácil cambiar el puerto de datos en la LCD, que ir de línea en línea cambiando el PORTA del microcontrolador.

Las variables deben de tener consistencia y ser congruentes con la librería en uso, así, cuando por ejemplo se llame la librería *LED*, y tengamos la información de los siete segmentos en la memoria, será llamada como *LedDispData*.

A notar, que *los macros están en mayúsculas, mientras que las variables se encuentran en letras minúsculas*. La mayoría del código está de esta manera, si alguna de las dos condiciones fue pasada por alto, el autor le ruega disculpas en la realización del código.

1.1.8.9 Estructuras de Datos

Las estructuras, para ensamblador, son un grupo de variables agrupadas, que se mueven en la memoria dada la ubicación de la primera, por ejemplo:

```
Centenas    equ    $0080        ; Dirección 0x80
Decenas     equ    Centenas+1   ; Dirección 0x81
Unidades    equ    Centenas+2   ; Dirección 0x82
```

Para reubicar las variables, se debe solo mover la primera (centenas), pues las mismas se concatenan y se mueven por la memoria como un solo bloque. Solo se debe tener especial cuidado en que la estructura no caiga encima de alguna variable o el programa borre de manera inesperada estas variables.

1.1.8.10 Funciones

Las funciones son segmentos de código que se repiten y que el programador hace reusable para no tener que compilar el mismo código una y otra vez, así cuando se llame la función, muchas veces, no se copiará el código, sino que hará un salto incondicional a la subrutina directamente.

1.1.8.10.1 Encabezado

Al igual que en el programa principal, las funciones, presentan un encabezado. Si por el caso de ejemplo, fuera una función del "*Timer Interfase Module*" que inicia el temporizador, la misma se puede representar de la siguiente manera:

```
;=========================================
; TIMINIT       :  Inicializa el Módulo TIM
; OBJETIVO      :  Inicializa el TIM y configura
;                  divisor y Módulo
; ENTRADA       :  A  valor del prescalar
;                  HX    contiene el valor del
;                        registro TMOD[H:L]
; SALIDA        :  Ninguna
; REGISTROS
; AFECTADOS     :  TSC, TMOD[H:L], ACCA, SP
;=========================================
```

Debe llevar un nombre que especifica rápidamente en que consiste la rutina; el objetivo de la rutina que es la descripción específica de que realiza, sus argumentos de entrada, que por lo general están asociados a los registros A, H, X y las variables de retorno o banderas. Adicionalmente incluir, para el caso de ensamblador, los registros del microcontrolador afectados para saber las condiciones de salida que alterarían la ejecución de un programa con esta rutina.

1.1.8.10.2 Banderas

Cuando se habla de banderas, son valores booleanos que representan el estado de salida de una función, un valor de retorno, si logró o no su cometido. Por ejemplo, si queremos convertir una cantidad física, al mundo digital, para saber si el resultado es correcto, deberíamos tener una función algo parecida a esta:

```
AGAIN:                  ; Etiqueta de inicio de conversión
        lda #ADC7       ; Canal a tomar muestra PTB7
        jsr ADCMeasure  ; Mide
        beq AGAIN       ; A = 0, Conversión no realizada
        <hacer algo>    ; A = 1, Conversión realizada
```

La bandera, por lo general está definida por algunos casos, o la mayoría, el registro A del CPU y en muy pocas rutinas reusables hechas, por el registro X, también puede darse el caso de una bandera como una variable ubicada en la memoria volátil (RAM).

1.1.8.10.3 Funciones con Más de Tres argumentos

A veces, las funciones que el usuario quiera especificar reclaman por necesidad más de tres argumentos. Una función que escribe en la pantalla LCD, puede ser un buen ejemplo.

```
        clra            ; Fila 0
        clrx            ; Columna 0
        pshx            ; Empuja columna a la pila
        ldhx #MENSAJE   ; Carga Dirección Inicial del Mensaje a Enviar
        jsr LCDWrMsgXY  ; Escribe el mensaje en la Fila-Columna
        ais #1          ; Reposiciona la pila
```

Cada vez que se empuje un valor a la pila, se debe de reposicionar la pila luego de la función, a su posición inicial, pues la pila es una parte de la memoria que crece en sentido inverso, si esta parte se llena, ocurre un desborde por sobreflujo, es mejor no correr riesgos, lo indicado es reposicionar nuevamente la pila para que no afecte tanto la memoria como las direcciones de retorno.

1.1.9 Abreviaturas

Las siguientes abreviaturas son escogidas textualmente del libro "*Embedded Systems Building Blocks*" y refieren al uso de estos acrónimos, mnemónicos, cada vez que escriba una función o parte del programa

Tabla 8. Diccionario de Acrónimos, Abreviaturas y Mnemónicos

Número	Descripción	Mnemónico, Acrónimo, Abreviatura
1	Abajo	Down
2	Acción	Act
3	Actualizar	Update
4	Adición	Add
5	Agenda	Sched
6	Alarma	Alm
7	Alto	Hi
8	Anidamiento	Nesting
9	Anillo	Ring
10	Año	Year
11	Argumentos	Arg
12	Arriba	Up
13	Bajo	Lo
14	Bandera	Flag
15	Barra	Bar
16	Bit	Bit
17	Borde	Edge
18	Borrar	Delete
19	Borrar Pantalla	Cls
20	Buffer	Buf
21	Buffered	Bu
22	Bypass	Bypass
23	Cadena	Str
24	Calcular	Calc
25	Calibración	Cal
26	Cambio	Change
27	Canal	Ch
28	Columna	Col
29	Comando	Cmd
30	Comparar	Cmp

Tabla 9. Diccionario de Acrónimos, Abreviaturas y Mnemónicos. Continuación

Número	Descripción	Mnemónico, Acrónimo, Abreviatura
31	Compensación, corrimiento	Offset
32	Comunicación	Comm
33	Configuración	Cfg
34	Conmutación	Sw
35	Contexto	Ctx
36	Contraseña	Key
37	Control	Ctrl
38	Correo	Mbox
39	Corriendo	Running
40	Corriente (Actual)	Cursor
41	Cuenta	Cnt
42	Cursor	Cursor
43	Decimal	Dec
44	Decodificador	Dec
45	Decodificar	Decode
46	Desbloquear	Unlock
47	Deshabilitar	Disable
48	Desplegar	Disp
49	Detectar/Detección	Det
50	Día	Day
51	Día-de-la-Semana	DOW
52	Dígito	Dig
53	Dividir	Div
54	Divisor	Div
55	El más bajo	Lo
56	En espera	Q
57	Entrada Análoga	AI
58	Entrada Análoga de E/S	AIO
59	Entrada Discreta	DI
60	Entrada(s)	In
61	Entradas	Entries
62	Entrar	Enter
63	Error(es)	Err
64	Escala	Scale
65	Escalamiento	Scaling
66	Escaneo	Scan
67	Escribir	Wr
68	Estadística	Stat
69	Estado	Stat
70	Estampilla-de-Tiempo	TS
71	Estatus	State
72	Eventos	Event

Tabla 10. Diccionario de Acrónimos, Abreviaturas y Mnemónicos. Continuación

Número	Descripción	Mnemónico, Acrónimo, Abreviatura
73	Exponente	Exp
74	Factor de Escala	SF
75	Fecha	Date
76	Fila	Row
77	Formato	Format
78	Fracción	Fract
79	Función	Fnc
80	Ganancia	Gain
81	Grupo	Grp
82	Habilitado	Enable
83	Hexadecimal	Hex
84	Horas	Hr
85	I.D.	Id
86	Imponer	Put
87	Imponer	Set
88	Impresora	Prt
89	Índice	Ix
90	Inicialización	Init
91	Inicio	Start
92	Interrupción	Int
93	Invertir	Inv
94	Itinerario	Sched
95	Leer	Rd
96	Liberar	Free
97	Límite	Lim
98	Limpiar	Clr
99	Lista	List
100	Lista	Tbl
101	Listo	Ready
102	Lleno	Full
103	Manejador	Handler
104	Mantisa	Man
105	Manual	Man
106	Máscara	Msk
107	Matemática	Math
108	Máximo	Max
109	Mensaje	Msg
110	Mes	Month
111	Mínimo	Min
112	Minuto	Min
113	Modo	Mode
114	Múltiple	Mux

Tabla 11. Diccionario de Acrónimos, Abreviaturas y Mnemónicos. Continuación

Número	Descripción	Mnemónico, Acrónimo, Abreviatura
115	Multiplicación	Mul
116	Muy Bajo	Lo
117	Nuevo	New
118	Nuevo Llamado	Rcl
119	Número de	N
120	Ocio	Idle
121	Palabra de Control	CW
122	Pantalla	Scr
123	Parar	Stop
124	Pasar	Pass
125	Pila	Stk
126	Posición	Pos
127	Previo	Prev
128	Prioridad	Prio
129	Puerto	Port
130	Puntero	Ptr
131	Rebote	Debounce
132	Recibiendo	Rx
133	Registro	Reg
134	Reinicio	Reset
135	Reloj	Clk
136	Repetir	Repeat
137	Resumir	Resume
138	Retardo	Delay
139	Rutina de Servicio de Interrupción	ISR
140	Salida	Out
141	Salida Análoga	AO
142	Salida Discreta	DO
143	Salidas	Exit
144	Segmento(s)	Seg
145	Segundos	Seg
146	Seleccionar	Sel
147	Sellado	Lock
148	Semáforo	Sem
149	Siete Segmentos	SSeg
150	Siguiente	Next
151	Simulada	Dummy
152	Sincronización	Sync
153	Sistema Operativo	OS
154	Sobreflujo	Ovf
155	Suspender	Sus
156	Sustracción	Sub
157	Tamaño	Size

Tabla 12. Diccionario de Acrónimos, Abreviaturas y Mnemónicos. Continuación

Número	Descripción	Mnemónico, Acrónimo, Abreviatura
158	Tarea	Task
159	Tarea de Alta Prioridad	HPT
160	Tarea de Baja Prioridad	LPT
161	Teclado	Key, KBD
162	Tecleado	Hit
163	Temporizador	Tmr
164	Tic	Tick
165	Tiempo	Time
166	Tiempo-Real	RT
167	Todo	All
168	Tomar	Get
169	Transmitir	Tx
170	Umbral	Th
171	Unidades Ingenieriles	EU
172	Vacío	Empty
173	Valor	Val
174	Valor Bruto	Raw
175	Vector	Vector
176	Verificar	Chk
177	Viejo	Old

1.1.10 Ambientes de Desarrollo

Los ambientes de Desarrollo utilizados para este trabajo fueron WinIDE, que permite solamente código en Ensamblador y CodeWarrior V3.0 que permite tanto lenguajes Ensamblador, C, C++ o mixtos de estos.

La herramienta WinIDE, se puede bajar gratuitamente del sitio de la Cía PEMicro en: http://www.pemicro.com.

La herramienta CodeWarrior, que hasta la fecha está en la Versión 3.1, permite compilar hasta 16Kb de código residente en su pastilla, esta puede ser bajada de la Cía. Metrowerks en la siguiente dirección: http://www.metrowerks.com/MW/download/request.asp?action=dl&product=HC08DL&submit=Select+%3E%3E.

Adicionalmente, existen otros ambientes de desarrollo para programación de microcontroladores en lenguaje C, como lo son el de la compañía, Cosmic Software y el de la compañía ImageCraft, denominado ICC08:
Cosmic Software - http://www.cosmicsoftware.com/hc08.php
ImageCraft - http://www.imagecraft.com/software/

Como recomendación, si su máquina es de bajo rendimiento es recomendable tener el compilador de ensamblador WinIDE. Sin embargo, la tendencia hacia microcontroladores está orientada a lenguajes de alto nivel como el C, de tal forma, el entorno más completo, es el CodeWarrior de la compañía Metrowerks.

1.1.11 Información para HC08

1.1.11.1 Información Específica de Microcontroladores Motorola

Se ha recopilado una cantidad razonable de sitios web sobre microcontroladores de la familia HC08, de la cual se tiene poca información bibliográfica, a pesar del esfuerzo de Motorola por liberar microcontroladores a bajo costo, el ambiente de desarrollo y los programadores. Dichas páginas tratan de cubrir, en algunas formas que presentan los autores, ciertos tópicos de los HC08 y otros microcontroladores de la misma Cía.

My First HC08 – http://www.geocities.com/issaiass/
Mundomicro – http://www.mundomicro.com.ar/
Ingeniería Inversa – http://www.ingenieria-inversa.com.ar/
Ingeniero Dubatti – http://www.ingdubatti.com.ar/
Buenos Aires Robotics – http://www.bairesrobotics.com.ar/
Electrónica – http://www.elektronikladen.de/hc08/
HC08 – Checoslovaquia http://www.hc08.cz/
RanchBots – http://www.ranchbots.com/HC08/hc08.htm

1.1.11.2 Aprendizaje de Lenguaje C para Microcontroladores

Un punto fuerte para aprender C, es su compatibilidad. Anteriormente los microcontroladores solo se programaban en lenguaje ensamblador, pero, pese a la proliferación del C y el aumento en memoria ROM (FLASH) y RAM en las últimas décadas, este ha sido el lenguaje preferido por ingenieros y “hobbistas”, pues el lenguaje C permite generar una aplicación en un tiempo menor a lo que se realizaría un programa en ensamblador. Un sitio interesante sobre la programación en C, es el siguiente:

Jonathan W. Valvano’s Page – http://www.ece.utexas.edu/~valvano/embed/toc1.htm

1.1.11.3 Proyectos con Microcontroladores de 8 – bits

Siempre se hace uno la pregunta de ¿para que sirve?. Los microcontroladores, son dispositivos que poseen hardware interno que hacen más fácil implementar la solución de un proyecto y se eligen dependiendo de la aplicación. Cada cierto tiempo, las compañías promueven sus productos regalándolos, pero para participar en concursos, de esta manera se realiza su proyección al mercado.

Flash Innovations 2003 – http://www.circuitcellar.com/fi2003/
AVR Design Contest 2004 – http://www.circuitcellar.com/avr2004/
Renesas H8 Design 2004 – http://www.circuitcellar.com/renesas/

Todos los concursos tratan sobre microcontroladores, y da la casualidad son estos de 8 – bits, de esta forma, ud. podrá ver que se puede realizar con una pastilla de este tipo.

1.1.11.4 Compañías de Equipo Electrónico

Algunos materiales, como los microcontroladores Motorola, no se encuentran en el mercado local, y se deben pedir al exterior. De hecho, es más fiable pedir componentes al exterior, pues estas empresas viven de eso y deben tener reservas a mano según las especificaciones de los clientes, a diferencia de las tiendas y mercados locales electrónicos que solo compran por demanda. Dichas compañías son:

Jameco Electronics – http://www.jameco.com/
Digikey – http://www.digikey.com/ (Venden el Microcontrolador)
Mouser Electronics – http://www.mouser.com/

1.1.11.5 Grupos de Yahoo

Además, existen grupos o foros donde el primerizo puede hacer consultas sobre temas básicos, introductorios o avanzados, estos grupos son:

MCU Motorola – http://www.groups.yahoo.com/group/MCUMotorola
68HC05 68HC08 – http://www.groups.yahoo.com/grup/68HC05_08
HC08 – http://www.groups.yahoo.com/group/hc08

De la misma manera, existen diversos grupos para áreas afines como DSP, VHDL, PCs.

1.1.11.6 Respuestas a las Preguntas Más Comunes del 68HC908

1.1.11.6.1 ¿Dónde puedo encontrar mas información sobre el WinIDE y el uso del Assembler del SD 68HC908?

La información del ICS08 (ICS = In–Circuit Simulator) WinIDE (Windows Intergrated Development Environment) y el assembler del 68HC908, se puede encontrar en la manuales técnicos o data del fabricante que acompaña el producto. Los manuales son los siguientes:

Manual de Referencia al instalar WinIDE para QT/QY:
a. ICS08 General Manual

Librería de Documentación en Freescale:
b. MC68HC908JL3E/D Technical Data
c. CPU08RM Central Processor Unit Reference Manual

1.1.11.6.2 ¿Donde puedo encontrar mas informacion sobre el ICS08?

Referencia: Ver (Página / Sección) indicada.

Referencia a – **ICS08 General Manual –** Tópico:

Capítulo 1 – Introducción
- a. Si se olvida el código de seguridad de la memoria Flash, se puede borrar el dispositivo. (1-8/1.6)

Capítulo 3 – WinIDE User Interfase
- a. Botones/Iconos del WinIDE (3-5/3.4.4)
- b. Barra de Herramientas y Barra de Menú del WinIDE (3-8/3.6)

Capítulo 4 – CASM08Z Assembler Interfase
- a. Descripción de los campos de una línea de un Archivo "*.LST" (4-4/4.4.3)
- b. Opciones del Assembler (4-6/4.5)
- c. Operandos y Constantes (4-6/4.5.1)
- d. Comentarios (4-7/4.5.2)
- e. Directiva Include (4-10/4.6.4)
- f. Listado del assembler – Archivo *.LST (4-12/4.7.1)
- g. Etiquetas o "Labels" (4-14/4.7.2)
- h. Asignación de variables o "Equate" (EQU) - (4-15/4.8.1)
- i. Directiva Origen o ORG (4-16/4.8.4.)

Capítulo 5 – ICS08Z In-Circuit Simulator
- a. Ventana de Windows del ICS08 (5-8/Fig 5.2)
- b. Ventana de Memoria (5-13/5.8)
- c. Ventana del CPU (5-17/5.10)
- d. Ventana de la Pila (5-18/5.12)
- e. Barra de Herramientas del ICS08Z (5-26/5.17)
- f. Barra de Menú del ICS08Z (5-28/5.18)
- g. DDRX – Resumen de Comandos del ICS08Z (5-42/Tabla 5.4)
- h. PORTX – Resumen de Comandos del ICS08Z (5-43/Tabla 5.4)

Capítulo 6 – PROG08Z Flash Programmer
- a. Comandos de Programación (6-4/6.3)

Capítulo 7 – ICD08Z In-Circuit Debugger

Capítulo 8 – Debugging Command Set

Apéndice A – S – Record Information
- a. S-Record Content (A-1/A.2)
- b. S-Record Types (A-2/A.3)
- c. S-Record Example (A-4/ A.5)

Apéndice B – Glosario

1.1.11.6.3 ¿Donde puedo encontrar información del MC68HC908?

Referencia: Ver (Página/Sección) indicada.

Referencia b. **MC68HC908JL3E/D Technical Data**

Sección 1 – Descripción General
- a. Resumen de las características del 68HC908 JK1/JK3/JL3 (22/1.3)
- b. Diagrama de Bloque del MCU (24/Fig. 1-1)
- c. Asignación de Pines del MCU (25/Fig. 1-2)
- d. Mapa de Memoria del MCU (28/2.1)
- e. Registros del MCU (30/Fig. 2.2)
- f. Dirección de Vectores del MCU (35/Fig. 2-1)

Sección 6 – Unidad Central de Procesamiento o CPU
a. Registros del CPU (47/ Fig 6-1)

Referencia c. **CPU08RM Central Processor Unit Reference Manual**
a. características Generales del CPU (20/1.3)
b. Modos de Direccionamiento (21/1.6)
c. Registro de Banderas (28/2.3.5)
d. Juego de Instrucciones (89/5.1)
e. Juego de Instrucciones – Ejemplos (189/6.1)