Universidad Don Bosco
Facultad De Ingeniería
Escuela De Ingeniería Electrónica

Trabajo de Graduación para obtener el grado de Ingeniero en Automatización

Tema:

Diseño e implementación del prototipo de un módulo de aprendizaje para el uso de un microcontrolador H8/3069F (H8/300H Core) fabricado por Renesas y sus guías de desarrollo.

Presentado por:
Rafael Ernesto Martínez Mendoza
Paul Joseph Dennehy Campos

Asesor:
Ingeniero Héctor Rubén Carias

Soyapango, 03 de February de 2007

Universidad Don Bosco
Facultad De Ingeniería

Rector
Ing. Federico Miguel Huguet Rivera

Vicerrector
Padre Víctor Bermúdez

Secretario General
Lic. Mario Rafael Olmos

Decano de la Facultad de Ingeniería
Ing. Ernesto Godofredo Girón

Director de la escuela de Electrónica
Ing. Oscar Giovanni Durán Vizcarra

Soyapango, 03 de February de 2007

Universidad Don Bosco
Facultad De Ingeniería

Subcomité Evaluador del Trabajo de Graduación

Tema:

Diseño e implementación del prototipo de un módulo de aprendizaje para el uso de un microcontrolador H8/3069F (H8/300H Core) fabricado por Renesas y sus guías de desarrollo.

F._____
Ing. Oscar Blanco
JURADO

F._____
Ing. Carlos Bran
JURADO

F._____
Ing. Rubén Carias
ASESOR

F._____
Ing. Néstor Lozano
TUTOR

Índice General

Índice Detallado

Índice de Tablas y Figuras:

Introducción

Caracterizado por un núcleo H8/300H de 16 bits de alto rendimiento y un rico juego de funciones periféricas que incluyen temporizadores de 8 y 16 bits, convertidores análogo-digital y digital-análogo, controladores DMA e interfaces seriales, entre otros, esta serie de microcontroladores (MCU) fabricada por Renesas es ampliamente usada para la comunicación y tratamiento de información en diferentes ramas de la industria. Una de las aplicaciones que ha experimentado un rápido crecimiento en los últimos años es el uso de estos microcontroladores en dispositivos de almacenamiento masivo de información tales como CD-R/RW, DVD ROM/RAM y discos duros (HDD). Hitachi desarrolló previamente los dispositivos H8/3062F, H8/3064F Y H8/3068F los cuales tenían una capacidad de memoria interna de 128, 256 y 384 KBytes respectivamente. Pero la necesidad de programas más grandes para satisfacer el incremento y mejoramiento de las funciones de los dispositivos de almacenamiento masivo antes mencionados demandó un incremento de memoria flash interna a 512 KBytes [1].

A su vez, podemos nombrar la creciente demanda de microcontroladores por parte de *Toyota Motor Corporation* [2] para su utilización de esta misma memoria flash interna en los diferentes sistemas, operados por microcomputadoras, abordo de sus vehículos. Otra de las aplicaciones más recientes de estos microcontroladores es en el área de fotografía digital. *Nikon Corporation* [3] ha seleccionado esta familia debido a su amplio rango de funciones periféricas integradas (temporizadores, convertidores A/D, etc.), así como de su bajo consumo de energía, lo que le permite tener un mayor tiempo de operación, y la disponibilidad de sistemas de desarrollo.

Podemos ver que esta familia de microcontroladores está siendo utilizada en la actualidad en diferentes ramas de la industria por compañías de gran renombre, y su desarrollo está lejos de detenerse.

El microcontrolador H8/3069F, con el cual vamos a trabajar, ha sido desarrollado con una memoria flash interna de 512 KBytes, además de 16 KBytes de RAM lo cual permite almacenar programas más extensos.

Puede operar a gran velocidad, con una frecuencia de hasta 25MHz y una fuente de poder de 5V ó 3.3V. Los circuitos lógicos tales como un núcleo (CPU) trabajan aproximadamente con 1.9V los cuales son generados a partir de las fuentes antes mencionadas, esto hace posible seleccionar la fuente de voltaje de acuerdo con la interfase usada. Por último, pero no menos importante, debemos mencionar la amplia gama de funciones integradas que contiene.

---

[1] [B2] Datos extraídos de *Hitachi Releases 16-Bit Microcontroller with On-Chip Large Flash Memory* (Ver anexos para mayor información).

[2] y [3] [B3] Información obtenida del artículo especial de Renesas *EDGE Vol. 11* del año 2006: *Flash Microcomputers*.

Las características antes mencionadas, así como su versatilidad de programación han hecho de este microcontrolador una poderosa herramienta de desarrollo.

En el presente trabajo de graduación se pretende proporcionar las bases y los conocimientos necesarios para poder trabajar con el microcontrolador antes mencionado. Se abordarán aspectos tanto de hardware como de software, pasando desde la elaboración del sistema mínimo, hasta el manejo de registros, diseño de aplicaciones y programación del microcontrolador.

Todo el trabajo de graduación estará basado en una serie de capítulos que describen cada uno de los subsistemas y además se presentan una serie de guías practicas con las cuales se busca que el estudiante pueda tener una primera experiencia con el tema que se esta tratando.

Dichos capítulos y guías se presentan más adelante en este documento.

Objetivos

GENERAL

Diseñar y desarrollar un sistema didáctico que le permita a otros estudiantes aprender a desarrollar el sistema mínimo del microcontrolador H8/3069F y a utilizar sus diferentes subsistemas básicos (H8/300H Core) por medio de las herramientas de programación facilitadas.

ESPECÍFICOS

- Presentar la información necesaria y los pasos a seguir para el desarrollo del sistema mínimo requerido por el H8/3069F para cualquier aplicación genérica.
- Diseñar e implementar el prototipo de un módulo que facilite la interacción con el microcontrolador, permitiendo aprender a usar sus diferentes subsistemas básicos.
- Elaborar una serie de guías con las cuales el estudiante podrá aprender la configuración básica y el uso de los diferentes subsistemas. Además, se proporcionarán las instrucciones para realizar el sistema mínimo. Dicho sistema podrá interconectarse con la etapa didáctica para poder probarlo y depurar el código de la aplicación.
- Presentar las bases teóricas necesarias para poder trabajar con los subsistemas mencionados.
- Entregar toda la información y equipo necesario para que los estudiantes puedan acumular los conocimientos y habilidades necesarias en una profesión orientada a la práctica[4].

  Notas:
  - En la información proporcionada (guías y capítulos teóricos) se contemplará el funcionamiento del subsistema, la configuración y dirección de los registros, los pasos a seguir para la puesta en marcha y consejos basados en experiencias personales.
  - Se dejará un ejemplo práctico comentado para cada uno de los temas de cada subsistema a tratar.
  - Objetivos académicos más específicos, basados en los objetivos educacionales de la tabla del anexo 3, serán incluidos en cada una de las guías

[4] Según la definición de "Experiencia de Laboratorio" dada por ***A Colloquy on Learning Objectives For Engineering Education Laboratories***

Alcances

- Los subsistemas con los que vamos a trabajar son los mismos que fueron estudiados con relación a los microcontroladores que se abordan a lo largo de la materia Microcontroladores, además de dos subsistemas nuevos. Estos subsistemas son los siguientes:

    1. Sistema mínimo
    2. Uso de los programas: - High-performance Embedded Workshop
       - Flash Development Toolkit
    3. Manejo de puertos (I/O)
    4. Temporizador de 16 bits (aplicación del PWM)
    5. Temporizador de 8 bits
    6. *Timing Pattern Controller* (subsistema único de este microcontrolador, no lo tiene ni PIC ni HC12)
    7. *Watchdog Timer*
    8. Comunicación Serie (Asincrónica y sincrónica)
    9. Convertidores análogo-digital y digital-análogo (este último también es único de este microcontrolador, el DAC no lo tiene PIC ni HC12)
    10. Interrupciones y sus prioridades.

    Nota: El orden de está lista no corresponde al orden de los capítulos.

- El lenguaje de programación a ser usado será el lenguaje de ensamblador (usando programa High-performance Embedded Workshop) de modo que se mantenga la analogía con los microcontroladores ya estudiados. Además, al usar este lenguaje estamos asegurando un estudio más profundo sobre este microcontrolador y a su vez eliminamos el código innecesario introducido por los compiladores en C o C++.
- Se construirá el prototipo del módulo en dos etapas. La primera de ellas consta del sistema mínimo, el cual estará compuesto únicamente por el microcontrolador, el oscilador, circuitos de selección y protección, y las entradas/salidas. La segunda parte estará compuesta por los elementos didácticos (pantalla LCD, LED, conectores, interruptores para entradas digitales…) los cuales facilitarán las tareas de aprendizaje permitiendo que el estudiante se enfoque en la programación. Las dos etapas se interconectarán para funcionar como módulo didáctico, pero se deja la oportunidad de remplazar el sistema mínimo por uno creado por el estudiante para que luego de ser probado, pueda ser utilizado para crear un sistema o una aplicación específica.
- Las bases teóricas dadas no serán copia fiel de la hoja de datos del microcontrolador, más bien será una recopilación, enriquecida con experiencias y conocimientos personales, de los datos que se consideren importantes y necesarios para que el estudiante pueda realizar la aplicación para el subsistema correspondiente y aprender así su funcionamiento.

- Para la realización de los programas y la descarga de éstos al microcontrolador se utilizarán las herramientas antes mencionadas.
- En las guías de laboratorio se reforzarán rápidamente las nociones teóricas necesarias, dadas con anterioridad, y se presentarán los pasos necesarios para la implementación de la aplicación correspondiente.
- Además, se incluirá en las guías el código comentado relacionado con la aplicación a realizarse. Este ejemplo tratará únicamente sobre el subsistema al cual se está haciendo referencia. Solo se combinarán subsistemas cuando sea estrictamente necesario para no desviar la atención del subsistema que corresponde.

  Nota: Una lista de ejemplos ha sido presentada en el anexo 4.

Limitaciones

- El módulo será orientado para que trabaje con los subsistemas antes mencionados. Se deja abierta la posibilidad de trabajar con los demás subsistemas que posee el microcontrolador, tales como el DMA, Smart Card, entre otros, aunque no se presentará información o guías sobre estos.

- Es posible que el estudiante busque implementar su propia aplicación con el sistema mínimo que ha creado y que se apoye de los elementos didácticos para la evaluación y depuración de su sistema. Es por ello que se pensó en hacer el prototipo del módulo en dos impresos diferentes pero de manera a que se acoplen el uno con el otro, de forma que se puedan usar los elementos didácticos (de uno de los impresos) con cualquier sistema mínimo creado con las especificaciones dadas (en el otro impreso).

- Existen otras herramientas de programación y comunicación para el microcontrolador a tratar, pero se emplearán las antes mencionadas debido a que son libres de licencias. De este modo el estudiante podrá descargarlas y hacer uso de ellas sin costo alguno.

Capitulo Uno: Presentación [5]

[5] Para mayor información hacer referencia a la sección 1 de Manual de Hardware [B1].

1.1 Generalidades

El H8/3069F es parte de una serie de microcontroladores basados en un núcleo H8/300H, con arquitectura originaria de Hitachi, el cual contiene un CPU de 32 bits, registros generales de 16 bits y un juego de 62 instrucciones optimizado para trabajar a alta velocidad y compatible con los demás códigos de la serie H8/300, lo cual facilita el intercambio de Software.

El sistema integrado contiene memoria ROM, memoria RAM (16 KBytes), temporizadores de 8 y 16 bits, un controlador de patrón de tiempo programable (TPC: Programmable Timing Pattern Controller), un temporizador WatchDog (WDT: WatchDog Timer), una interfase de comunicación serial (SCI), convertidores A/D y D/A, puertos de entrada/salida, un controlador de acceso directo a memoria, una interfase de comunicación Smart Card y un controlador de bus.

Además, el H8/3069F tiene 512 KBytes de memoria flash, 16 MBytes de memoria direccionable y seis modos de operación para el CPU, lo que permite escoger diferentes anchos de bus y tamaño de memoria direccionable. Estos modos (del 1 al 5 y el 7) incluyen un modo on-chip y 5 modos expandidos.

El microcontrolador a tratar incluye también una versión de F-ZTAT™, junto con la memoria flash integrada, la cual permite cambiar o intercambiar las especificaciones de la aplicación por su facilidad y flexibilidad en la programación. La *F* en F-ZTAT viene de “flash y flexible” y *ZTAT* viene de “zero turn-around time.”
La línea de microcontroladores F-ZTAT pone en juego varias tecnologías flash de Renesas (200 modelos clasificados en 28 categorías según la velocidad del reloj y la capacidad de la memoria) para darle al usuario más opciones a la hora de diseñar, siempre con un gran desempeño y confiabilidad.

En las páginas a seguir detallaremos un poco más las características básicas de este microcontrolador.

1.1.1 Instrucciones

Posee 62 instrucciones básicas dentro de las cuales podemos encontrar instrucciones aritmético-lógicas de 8/16/32 bits, instrucciones para multiplicar y dividir así como instrucciones para manipulación de binarios. A continuación presentamos la tabla 1.1 con un resumen de las instrucciones disponibles:

Function	Instruction	Types
Data transfer	MOV, PUSH*1, POP*1, MOVTPE*2, MOVFPE*2	3
Arithmetic operations	ADD, SUB, ADDX, SUBX, INC, DEC, ADDS, SUBS, DAA, DAS, MULXU, MULXS, DIVXU, DIVXS, CMP, NEG, EXTS, EXTU	18
Logic operations	AND, OR, XOR, NOT	4
Shift operations	SHAL, SHAR, SHLL, SHLR, ROTL, ROTR, ROTXL, ROTXR	8
Bit manipulation	BSET, BCLR, BNOT, BTST, BAND, BIAND, BOR, BIOR, BXOR, BIXOR, BLD, BILD, BST, BIST	14
Branch	Bcc*3, JMP, BSR, JSR, RTS	5
System control	TRAPA, RTE, SLEEP, LDC, STC, ANDC, ORC, XORC, NOP	9
Block data transfer	EEPMOV	1
		Total 62 types

Notes: *1 POP.W Rn is identical to MOV.W @SP+, Rn.
PUSH.W Rn is identical to MOV.W Rn, @–SP.
POP.L ERn is identical to MOV.L @SP+, Rn.
PUSH.L ERn is identical to MOV.L Rn, @–SP.
*2 Not available in the H8/3068F.
*3 Bcc is a generic branching instruction.

Tabla 1. 1: Clasificación de instrucciones

Para poder utilizar estas instrucciones, debemos de seguir un formato específico el cual consiste en una unidad de dos bytes (una palabra). Cada instrucción debe estar compuesta por un Campo de Operación (OP field), un Campo de Registro (R field), una Dirección Efectiva (EA field) y un Campo de Condición (CC field).

- OP Field: Los primeros 4 bits de la instrucción, indica la función de la instrucción, el modo de direccionamiento y la operación a ser efectuada. Ciertas instrucciones poseen dos OP.
- R Field: Especifica un registro general. Los registros de dirección están especificados por 3 bits, los de datos por 3 ó 4 bits.
  Algunas instrucciones tienen dos R Field, otras no tienen ninguno.
- EA Field: 8, 16 ó 32 bits especificando el dato, una dirección absoluta o un desplazamiento. Una dirección de 24 bits es tomada como de 32 bits en donde los primeros 8 bits son cero.
- CC Field: Especifica la condición de salto en caso de una instrucción Bcc.

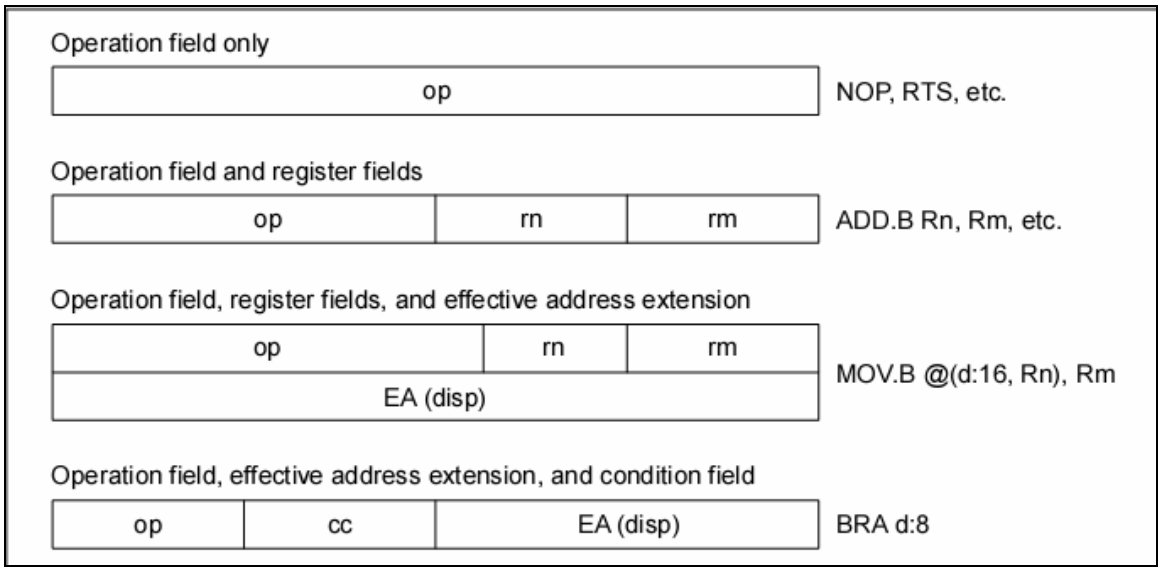


Figura 1. 1: Formato de instrucciones

1.1.2 Espacio direccionable

Tenemos un espacio direccionable el cual se extiende desde H´000000 hasta H´FFFFFF dando un total de 16 MB. Para el microcontrolador en cuestión podemos escoger entre este espacio o uno de 1 MB en donde solo usamos 20 bits de direccionamiento (en lugar de 24), los 4 primeros son ignorados:

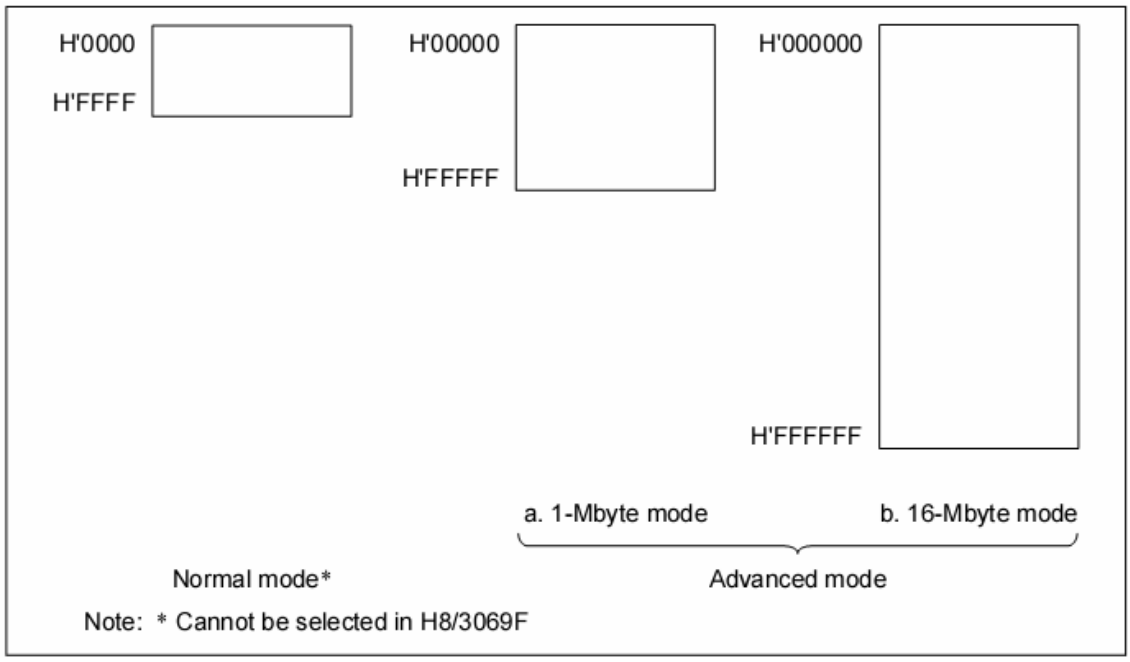


Figura 1. 2: Mapa de memoria generalizado

1.1.3 Registros

Tenemos dos tipos de registros. Primeramente, los registros generales, los cuales pueden ser usados en forma de Byte (RxH o RxL), en forma de palabra (Rx) o en forma de palabra doble (ERx). En el caso del registro ER7, este funciona también como Pila.
La siguiente figura ilustra los diferentes registros generales:

	15 0	7 0	7 0
ER0	E0	R0H	R0L
ER1	E1	R1H	R1L
ER2	E2	R2H	R2L
ER3	E3	R3H	R3L
ER4	E4	R4H	R4L
ER5	E5	R5H	R5L
ER6	E6	R6H	R6L
ER7	E7 (SP)	R7H	R7L

Figura 1. 3: Registros generales

> Nota: En el caso que trabajemos con palabras o palabras dobles los datos deben comenzar en una dirección par, de no ser así la parte alta se completará con ceros.

En la figura 1.4 se muestra el formato de los datos en la memoria y se ilustra claramente la nota anterior.

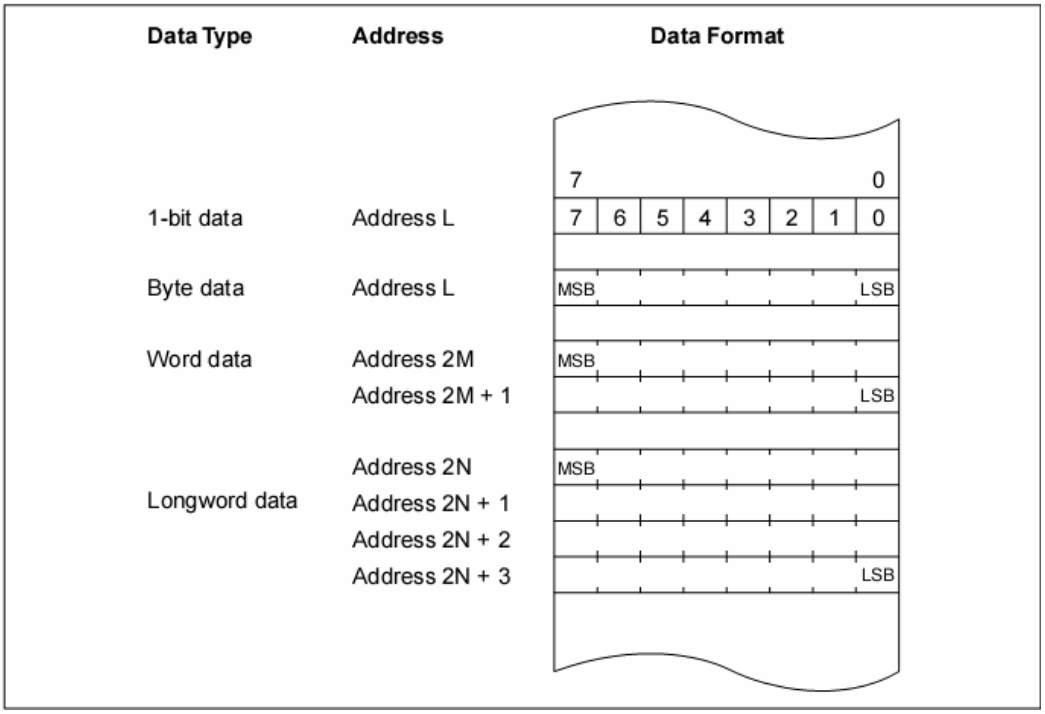


Figura 1. 4: Formato de datos en la memoria

Luego, tenemos el registro de control el cual esta formado por un *Contador de Programa* de 24 bits (3 Bytes), cuyo valor indica la dirección de la próxima instrucción a ser ejecutada, y por el *CCR (Condition Code Register)* de 8 bits en donde encontramos los estados de ciertos parámetros internos del CPU[6]. Tales parámetros están detallados en la figura 1.5.

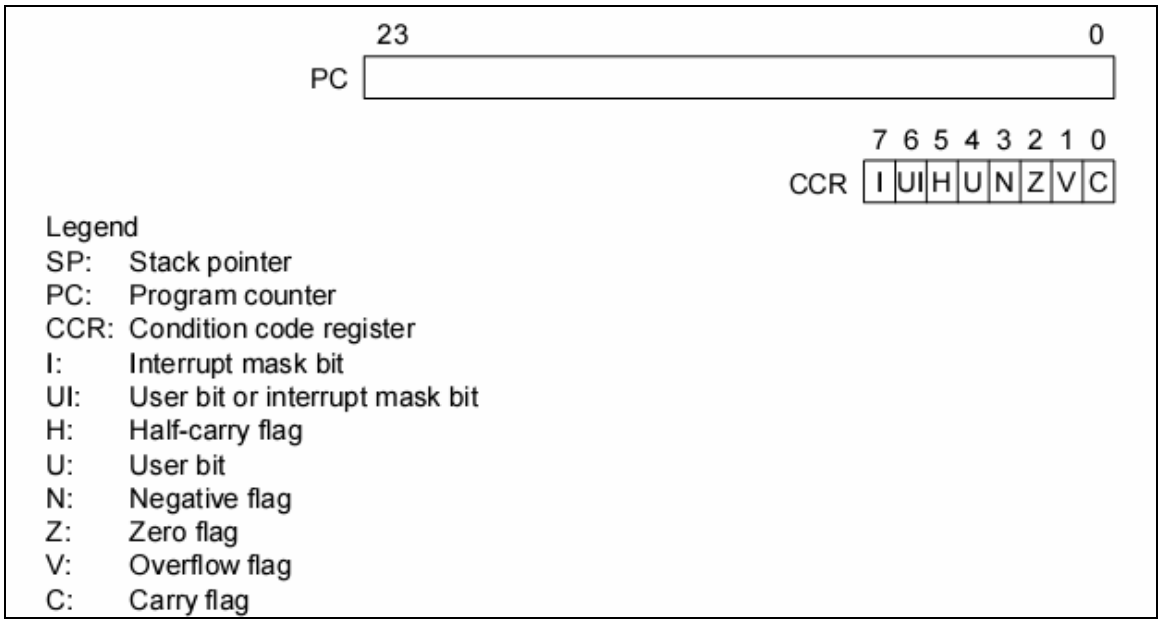


Figura 1. 5: Registros de control

> Nota: La alteración de los estados de las banderas del registro CCR depende de la instrucción que se este usando, para mayor detalle ver el juego de instrucciones proporcionado[7].

En el caso de un RESET el contador de programas es inicializado con el valor de la tabla de vectores de interrupción y el BIT I es puesto en 1. Los demás bits del CCR y los registros generales no son inicializados de modo que tienen valores no determinados. En el caso de ER7 (pila) debemos definir un valor después del RESET mediante una instrucción MOV.L.

1.1.4 Modos de direccionamiento[8]

El Cpu H8/300H soporta los 8 modos de direccionamiento que figuran el la tabla 1.2, cada grupo de instrucciones usa diferentes subconjuntos de estos modos de direccionamiento, algunos ejemplos son mencionados en la tabla 1.3:

[6] La mayoría de las banderas ya son conocidas por el usuario de modo que no serán detalladas. Para mayor información hacer referencia a la sección 1 del Manual de Programación [B4].
[7] Para el juego de instrucciones ver la sección 2.6 del Manual de Hardware del Microcontrolador [B1] o la sección 2 del Manual de Programación [B4].
[8] Para mayor detalle consulte la sección 2.7.1 del manual de Hardware [B1].

No.	Addressing Mode	Symbol
1	Register direct	Rn
2	Register indirect	@ERn
3	Register indirect with displacement	@(d:16, ERn)/@(d:24, ERn)
4	Register indirect with post-increment Register indirect with pre-decrement	@ERn+ @–ERn
5	Absolute address	@aa:8/@aa:16/@aa:24
6	Immediate	#xx:8/#xx:16/#xx:32
7	Program-counter relative	@(d:8, PC)/@(d:16, PC)
8	Memory indirect	@@aa:8

Tabla 1. 2: Modos de direccionamiento

Instrucciones		Modos
Aritmético-Lógicas		Directo e Inmediato
Data Transfer		Todos excepto los modos 7 y 8
BIT Manipulation		Directo, Indirecto, Absoluto e Inmediato

Tabla 1. 3: Clasificación de Modos de direccionamiento por grupo de instrucciones

Detalle:

1. Register Direct—Rn: Este modo de direccionamiento permite especificar un registro de 8, 16 o 32 bits el cual contiene el operando con el cual vamos a trabajar.

2. Register Indirect—@ERn: Con este modo especificamos un registro de 32 bits dentro del cual los 24 bits menos significativos contienen la dirección del operando.

3. Register Indirect with Displacement—@(d:16, ERn) or @(d:24, ERn): Un desplazamiento de 16 ó 24 bits el cual se suma al contenido de un registro de 32 bits especificado. Los 24 bits menos significativos de la suma resultante indica la dirección del operando.

4. Register Indirect with Post-Increment or Pre-Decrement—@ERn+ or @–ERn:

- Register indirect with post-increment—@ERn+

La instrucción especifica un registro de 32 bits en donde los 24 bits menos significativos contienen la dirección del operando. Después que se ha tenido acceso al operando, 1, 2 ó 4 (para byte, palabra o palabra doble respectivamente) es sumado al contenido del registro de 32 bits y almacenado en el mismo para la próxima consulta.

- Register indirect with pre-decrement—@–ERn

En este caso sustraemos la misma cantidad del registro de 32 bits antes del acceso.

5. Absolute Address—@aa:8, @aa:16, or @aa:24:

La instrucción contiene la dirección completa del operando. Podemos tener direcciones absolutas de 8, 16 y 24 bits, los rangos de acceso están dados en la tabla 1.4 siguiente.

Absolute Address	1-Mbyte Modes	16-Mbyte Modes
8 bits (@aa:8)	H'FFF00 to H'FFFFF (1048320 to 1048575)	H'FFFF00 to H'FFFFFF (16776960 to 16777215)
16 bits (@aa:16)	H'00000 to H'07FFF, H'F8000 to H'FFFFF (0 to 32767, 1015808 to 1048575)	H'000000 to H'007FFF, H'FF8000 to H'FFFFFF (0 to 32767, 16744448 to 16777215)
24 bits (@aa:24)	H'00000 to H'FFFFF (0 to 1048575)	H'000000 to H'FFFFFF (0 to 16777215)

Tabla 1. 4: Rango de acceso para direcciones absolutas.

1.1.5 Modos de procesamiento

El CPU de la familia H8/300H tiene 5 estados o modos de procesamiento. En el modo de ejecución, en donde el sistema permanece más tiempo, es donde se ejecutan las instrucciones una por una en secuencia. El sistema solo abandona este estado cuando es llamado a cualquiera de los estados siguientes:

Estado de Excepción:
Es un estado temporal el cual ocurre cuando el CPU altera su proceso normal debido a un Reset, interrupción o alguna otra excepción. El CPU apunta a la dirección de inicio dada por el vector, según la tabla de vectores, y salta a esa dirección para ejecutar su código. En caso que sean interrupciones u otras excepciones, el CPU hace referencia a ER7 (Stack Pointer) y guarda los valores del contador de programa y del CCR.
Tenemos diferentes tipos de excepciones con diferentes prioridades, Estas son presentadas en la siguiente tabla:

Priority	Type of Exception	Detection Timing	Start of Exception Handling
High	Reset	Synchronized with clock	Exception handling starts immediately when $\overline{RES}$ changes from low to high
↑	Interrupt	End of instruction execution or end of exception handling*	When an interrupt is requested, exception handling starts at the end of the current instruction or current exception-handling sequence
	Trap instruction	When TRAPA instruction is executed	Exception handling starts when a trap (TRAPA) instruction is executed
Low			

Note: * Interrupts are not detected at the end of the ANDC, ORC, XORC, and LDC instructions, or immediately after reset exception handling.

Tabla 1. 5: Excepciones y sus prioridades.

Estado de Bus Liberado:
En este estado el bus es liberado a un bus maestro diferente del CPU como respuesta a una señal de petición. Estos maestros de bus pueden ser el controlador DMA, la interfase DRAM y un bus master completamente externo.

Cuando el bus esta liberado, el CPU se detiene y no acepta ninguna otra interrupción, solo las operaciones internas continúan.

Estado de Reset:
Cuando en la entrada RES ponemos un estado bajo, todos los procesos que se estaban ejecutando se detienen y el CPU se pone en estado de Reset. En este estado no se aceptará ninguna interrupción por lo que la mascara I del CCR es puesta en uno. Para salirse de este modo o estado debemos de aplicar un estado alto en la entrada correspondiente.

> Nota: Un desbordamiento en el contador del WatchDog puede poner al CPU en estado de Reset.

Estado de conservación de energía:
Es posible poner al microcontrolador en un estado en el cual el CPU deje de operar y lograr así ahorrar un poco de energía. Ahora, existen tres subestados con características diferentes:

Sleep: Cuando el CPU ejecuta la instrucción SLEEP (con SSBY del registro SYSCR en cero). En este modo, toda operación se detiene (incluyendo el Clock) después de la ejecución de la instrucción pero todos los valores de CCR y de los registros son mantenidos.

SW Standby: El CPU se pone en este modo cuando ejecuta la instrucción SLEEP pero con el bit SSBY activo. Durante este estado todas las operaciones (incluyendo el Clock) se detienen, pero a diferencia del modo anterior, los registros, puertos de salida y la memoria RAM interna mantienen sus valores mientras el voltaje de alimentación sea mantenido. Por otro lado, se detienen y reinician todos los módulos integrados.

HW Standby: Estado el cual se activa cuando tenemos un pulso bajo en la entrada STBY. En este caso el CPU, el Clock y los módulos de soporte integrado se detienen y son reiniciados. La memoria RAM, como es de suponer, mantiene su valor mientras se mantenga el voltaje de alimentación.

1.1.6 Modos de operación

Para el microcontrolador seleccionado existen 6 modos de operación (del 1 al 5 y el 7), la selección de cada uno de estos es hecha mediante a los pines (MD2-MD0). Además de la selección del modo, la combinación del pinado indica la capacidad de direccionamiento y el modo inicial del BUS. La tabla siguiente resume la selección de los modos de operación:

Operating Mode	Mode Pins			Description			
	MD_2	MD_1	MD_0	Address Space	Initial Bus Mode*1	On-Chip ROM	On-Chip RAM
—	0	0	0	—	—	—	—
Mode 1	0	0	1	Expanded mode	8 bits	Disabled	Enabled*2
Mode 2	0	1	0	Expanded mode	16 bits	Disabled	Enabled*2
Mode 3	0	1	1	Expanded mode	8 bits	Disabled	Enabled*2
Mode 4	1	0	0	Expanded mode	16 bits	Disabled	Enabled*2
Mode 5	1	0	1	Expanded mode	8 bits	Enabled	Enabled*2
—	1	1	0	—	—	—	—
Mode 7	1	1	1	Single-chip advanced mode	—	Enabled	Enabled

Notes: *1 In modes 1 to 5, an 8-bit or 16-bit data bus can be selected on a per-area basis by settings made in the area bus width control register (ABWCR). For details see section 6, Bus Controller.

*2 If the RAME bit in SYSCR is cleared to 0, these addresses become external addresses.

Tabla 1. 6: Selección del modo de operación.

Modos de operación del 1 al 4:
Son modos extendidos que permiten el acceso a memoria externa y periféricos, y deshabilitan el acceso a la ROM integrada.

Modos 1 y 2: Tienen un máximo de 1 MBytes de espacio direccionable
Modos 3 y 4: Tienen un máximo de 16 MBytes

Modo de operación 5:
De igual forma que las anteriores, trabaja de forma extendida, pero en este caso podemos tener acceso a la memoria ROM integrada. Además, es necesario decir que, este modo tiene una capacidad de direccionamiento de 16Mbytes.

Modo de operación 7:
Este es el modo "single-chip" en el cual se usa las memorias ROM y RAM integradas, los registros y todas las entradas y salidas. Posee una capacidad de direccionamiento de 1 MByte.

Nota:

- El modo de operación no puede ser cambiado durante la ejecución u operación del sistema.
- El modo actual de operación puede ser visto en el registro MDCR (MoDe Control Register), el cual es solamente de lectura.

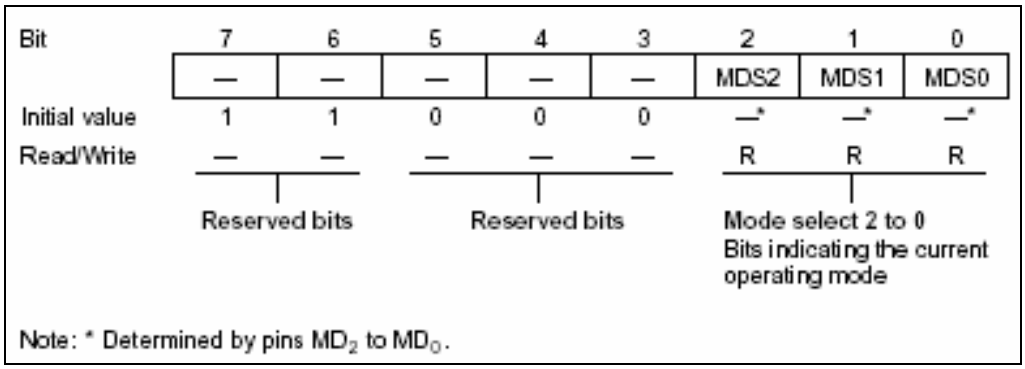


Figura 1. 6: Detalle del registro MDCR

Por otro lado, tenemos también el registro SYSCR (SYStem Control Register) en donde podemos controlar ciertas variables tales como el Standby, la selección de flanco para la entrada NMI (entrada de petición de interrupción) y la habilitación de RAM entre otras.

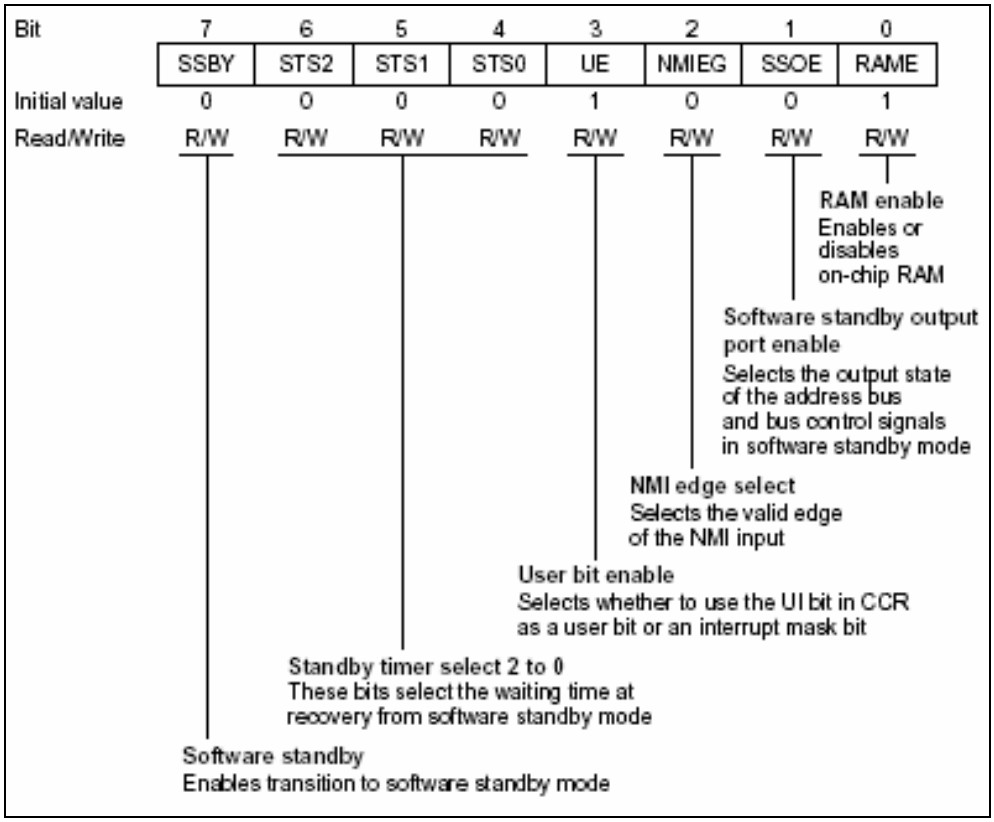


Figura 1. 7: Detalle del registro SYSCR

Es necesario mencionar que de los 6 modos antes expuestos, los dos que serán usados durante el desarrollo de este trabajo serán el modo 5 y el modo 7. Viéndolos de una manera más práctica, el modo 5 se comportaría como un modo de BOOTEO durante el cual podremos cargar el programa. El modo 7 vendría siendo como un modo RUN durante el cual podremos ejecutar el programa antes cargado.

Detalles: **Modo 5**:

Los puertos 1, 2, 5 y parte del puerto A funcionan como pines de direccionamiento (A23-A0) permitiendo direccionar un máximo de 16 MBytes, estos puertos regresan a su función original después de un Reset. Además, permite la descarga del programa a la memoria flash.

Modo 7:

Es este modo el microcontrolador trabaja con todos los registros y puertos disponibles y usa las memorias ROM y RAM internas. A diferencia del modo anterior, este solo puede direccionar 1 MByte.

1.2 Complementos:

1.2.1 Diagrama de Bloques

La figura 1.8 muestra el diagrama interno de bloques.

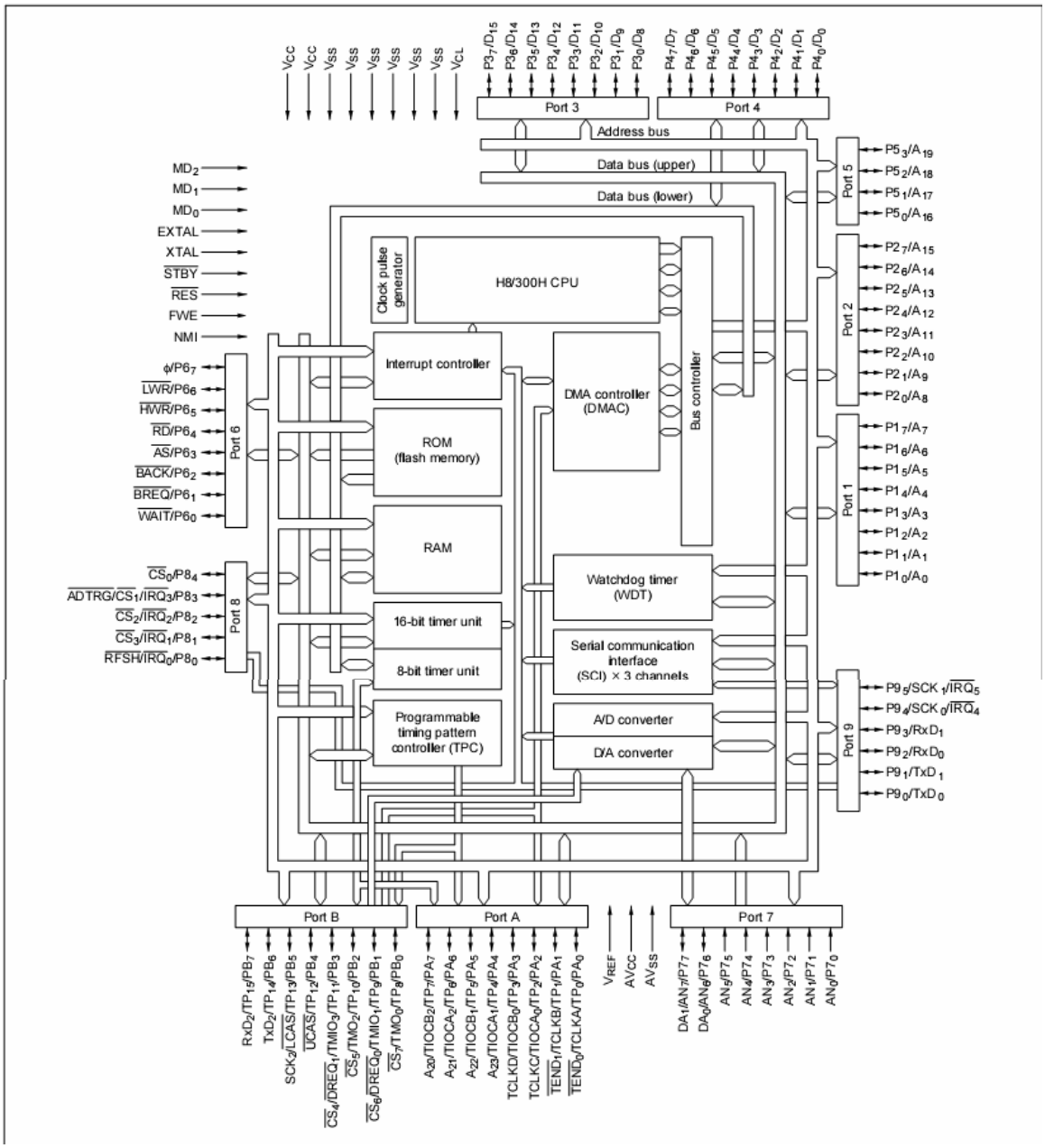


Figura 1. 8: Diagrama de Bloques

1.2.2 Distribución de pines del H8/3069F

La distribución de pines del microcontrolador H8/3069F para los encapsulados FP-100B y TFP-100B se muestra en la figura siguiente:

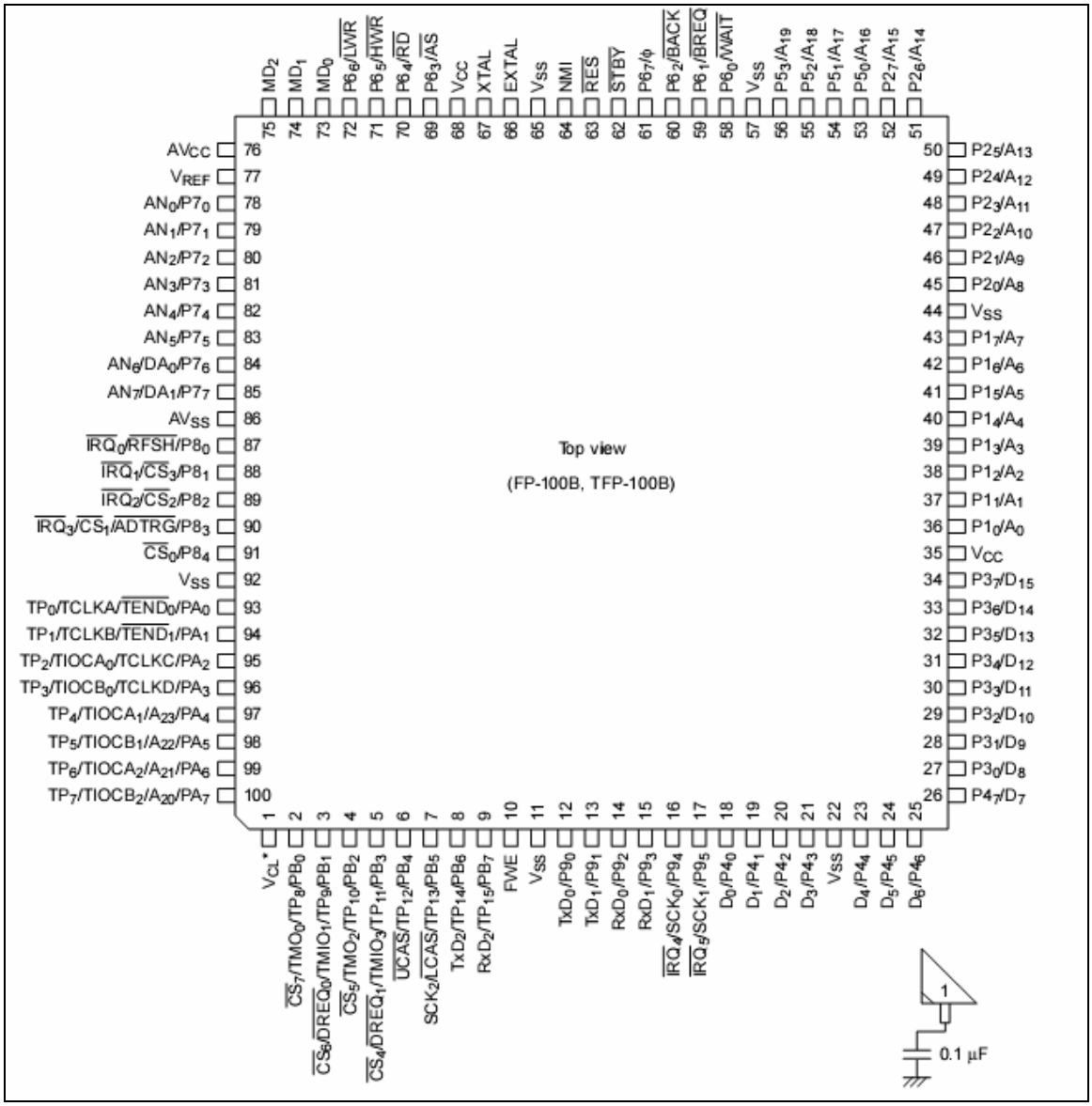


Figura 1. 9: Distribución de pines

1.2.3 Formato para datos en los registros generales

Las figuras 1.10 y 1.11 muestran el formato para los datos que se encuentran dentro de los registros generales.

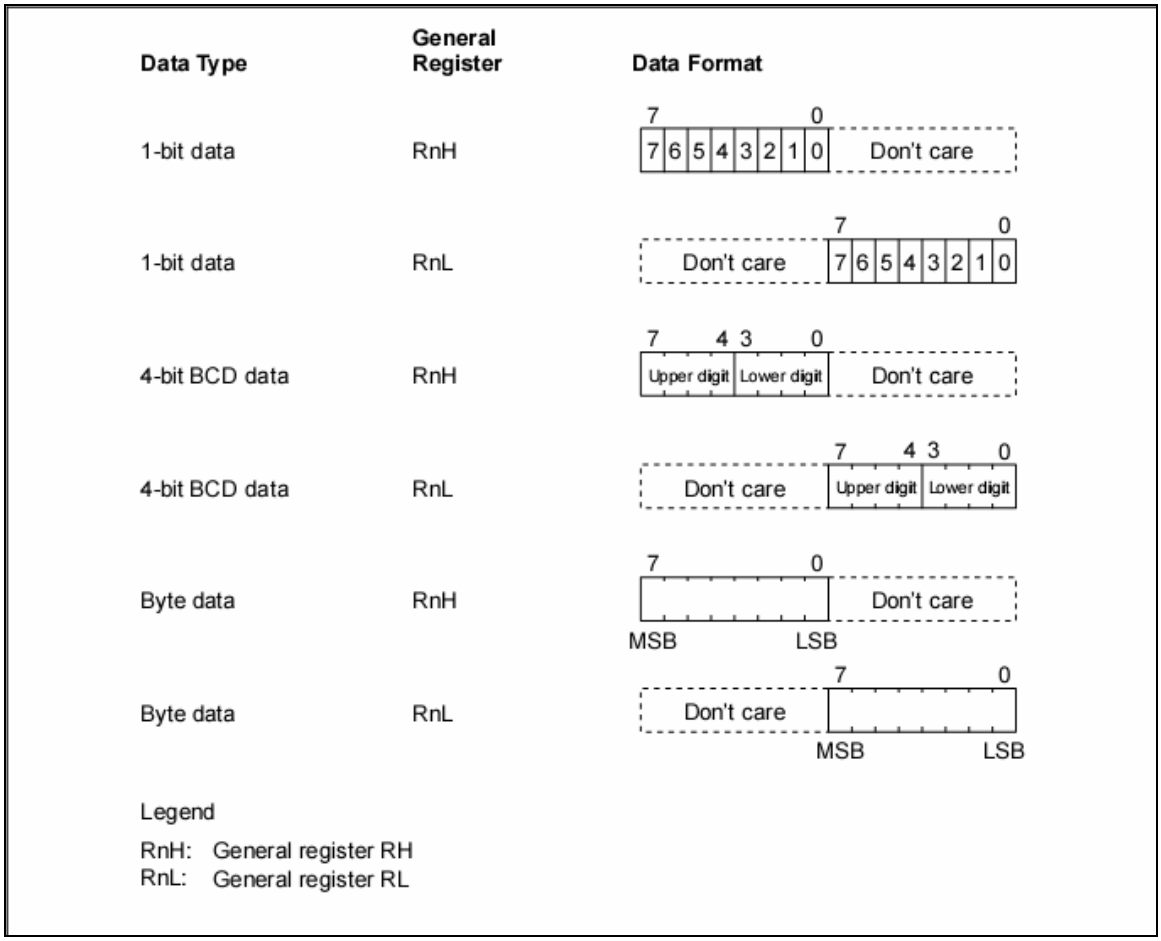


Figura 1. 10: Formato en registros generales

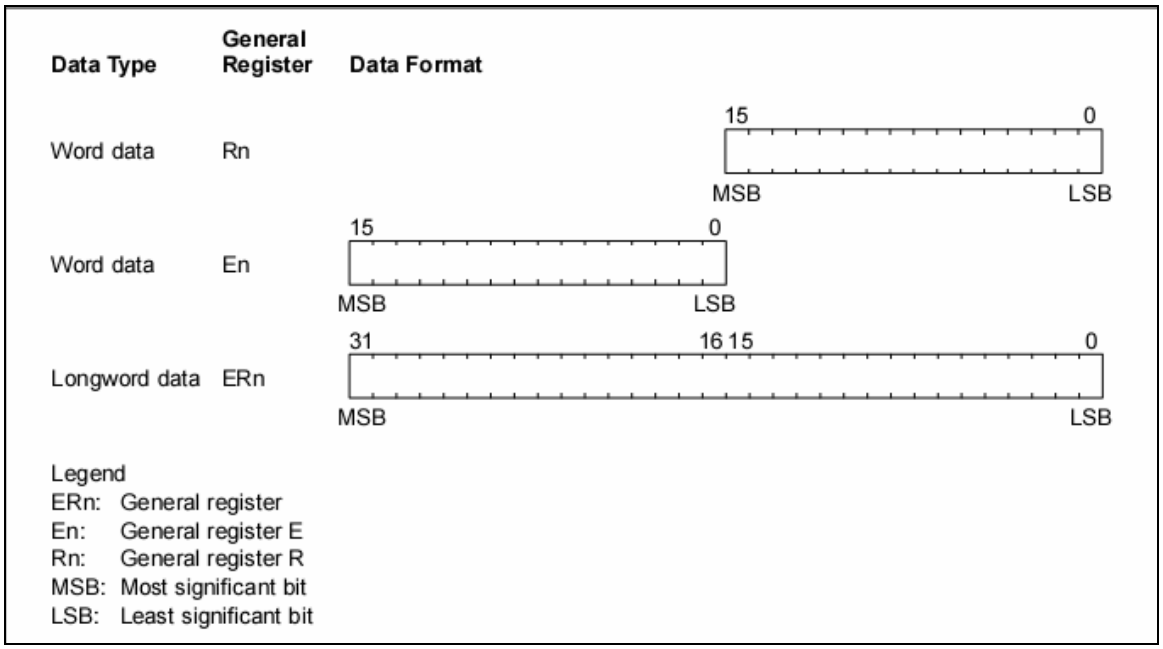


Figura 1. 11: Formato en registros generales

1.2.4 Funciones de los pines para cada modo de operación

La función de los pines de los puertos del 1 al 5, del puerto A y del puerto 6 varía dependiendo del modo de operación que se ha seleccionado. La tabla 1.8 muestra la función para cada uno de los modos de operación.

Port	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 7
Port 1	A_7 to A_0	A_7 to A_0	A_7 to A_0	A_7 to A_0	$P1_7$ to $P1_0$ *2	$P1_7$ to $P1_0$
Port 2	A_{15} to A_8	A_{15} to A_8	A_{15} to A_8	A_{15} to A_8	$P2_7$ to $P2_0$ *2	$P2_7$ to $P2_0$
Port 3	D_{15} to D_8	D_{15} to D_8	D_{15} to D_8	D_{15} to D_8	D_{15} to D_8	$P3_7$ to $P3_0$
Port 4	$P4_7$ to $P4_0$ *1	D_7 to D_0 *1	$P4_7$ to $P4_0$ *1	D_7 to D_0 *1	$P4_7$ to $P4_0$ *1	$P4_7$ to $P4_0$
Port 5	A_{19} to A_{16}	A_{19} to A_{16}	A_{19} to A_{16}	A_{19} to A_{16}	$P5_3$ to $P5_0$ *2	$P5_3$ to $P5_0$
Port 6_7	ϕ *5	ϕ *5	ϕ *5	ϕ *5	ϕ *5	$P6_7$ *5
Port A	PA_7 to PA_4	PA_7 to PA_4	PA_6 to PA_4, A_{20} *3	PA_6 to PA_4, A_{20} *3	PA_7 to PA_4 *4	PA_7 to PA_4

Notes:
*1 Initial state. The bus mode can be switched by settings in ABWCR. These pins function as $P4_7$ to $P4_0$ in 8-bit bus mode, and as D_7 to D_0 in 16-bit bus mode.
*2 Initial state. These pins become address output pins when the corresponding bits in the data direction registers (P1DDR, P2DDR, P5DDR) are set to 1.
*3 Initial state. A_{20} is always an address output pin. PA_6 to PA_4 are switched over to A_{23} to A_{21} output by writing 0 in bits 7 to 5 of BRCR.
*4 Initial state. PA_7 to PA_4 are switched over to A_{23} to A_{20} output by writing 0 in bits 7 to 4 of BRCR.
*5 Initial state. In modes 1 to 5 ϕ_{12} can be set as $P6_7$ by writing 1 to bit 7 in MSTCRH. In mode $P6_7$ can be set to ϕ output by writing 0 to bit 7 in MSTCRH.

Tabla 1. 7: Función de pines en cada modo de operación

1.2.5 Mapa de memoria de los modos de operación 5 y 7

Los mapas de memoria del H8/3069F incluyen áreas reservadas en las cuales el acceso de escritura y/o lectura esta prohibido. Se debe mencionar que no se puede asegurar la operación normal del sistema si las áreas antes mencionadas son usadas.

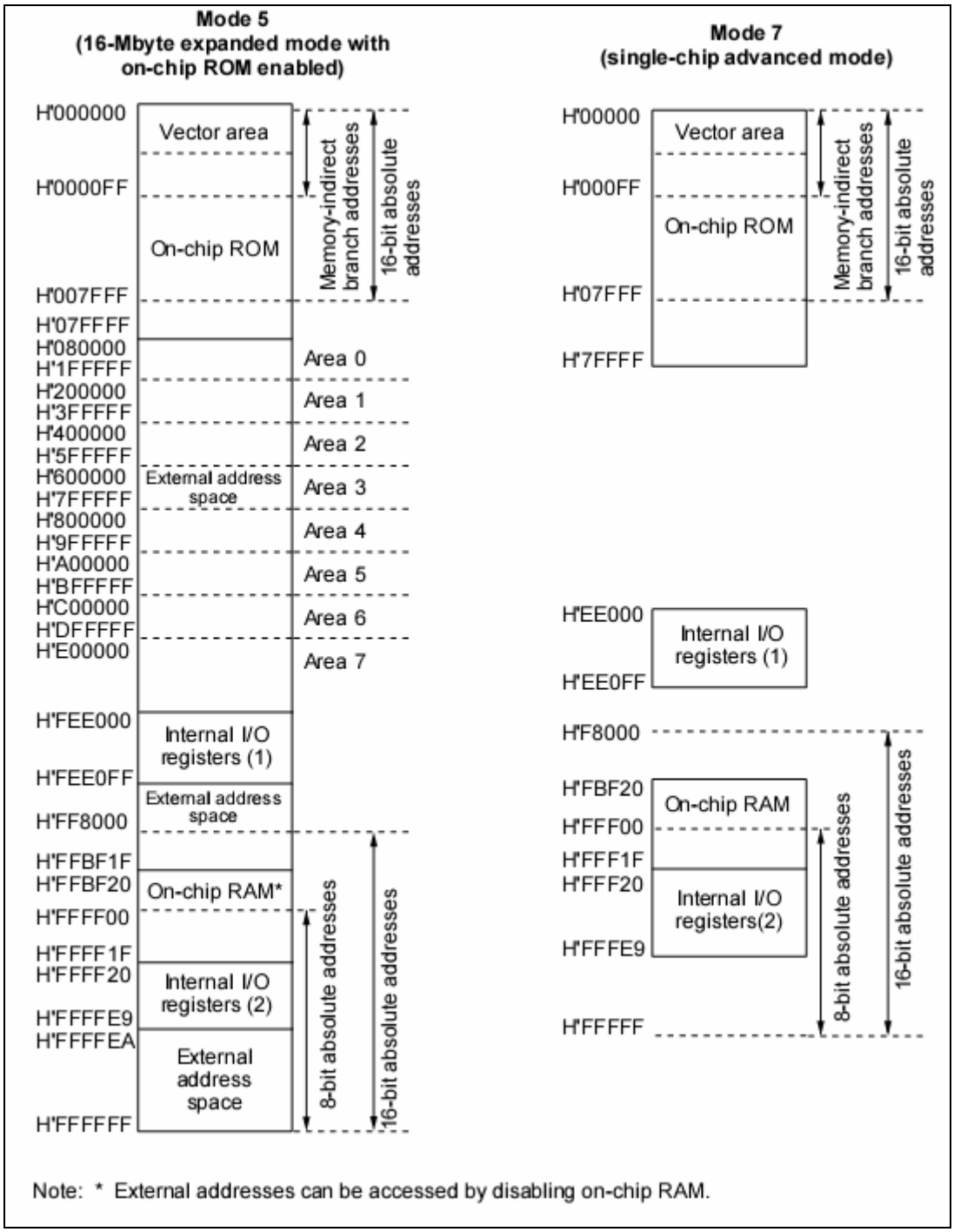


Figura 1. 12: Mapa de memoria de modos 5 y 7.

Capitulo Dos: Sistema Mínimo

2.1 Generalidades

Sistema mínimo debe conocerse como: los elementos necesarios para que un microcontrolador realice la mínima de las funciones y esta conformado por: la alimentación, la fuente reloj, la comunicación serial y elementos extra para generar la operación básica del microcontrolador.

Al contar con todos estos elementos en un circuito, un microcontrolador será capaz de echar a andar un mini sistema operativo, que permitirá que el programa alojado en su memoria se ejecute o, si se solicita la opción de booteo, permitirá la conexión a un programa monitor de aplicación y poder borrar o escribir datos a la memoria del microcontrolador.

Para el caso en estudio el sistema mínimo del microcontrolador H8/3069F es detallado en este capitulo, se describen los pasos a seguir para realización del mismo y se incluye también el diagrama esquemático y el diseño del impreso que tiene el módulo.

2.1.1 Características del LSI.

El integrador a larga escala o LSI es un chip encapsulado en una cubierta estilo QFP que posee 100 pines, la figura del encapsulado se muestra detallada en la figura 2.1

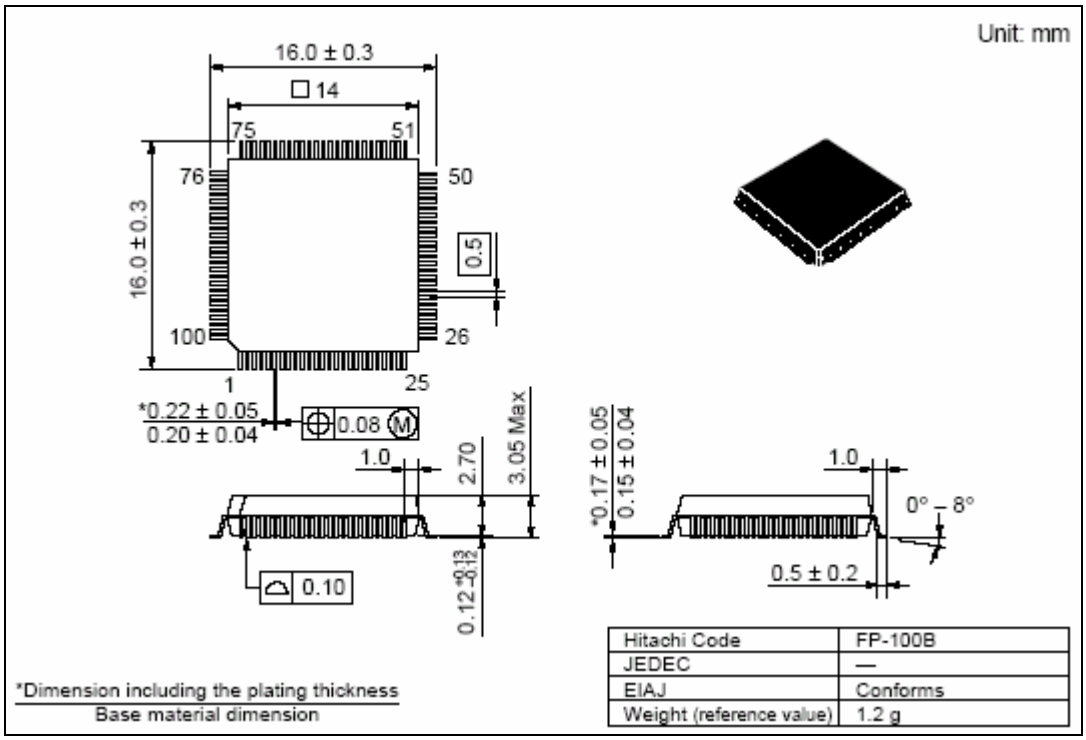


Figura 2. 1: Encapsulado HD64F3069F.

Internamente consta de una memoria flash de 512 KByte, la cual tiene las siguientes características:

- **Dos memorias flash llamadas MATs**: La memoria flash tiene dos espacios de memoria en una sola dirección de almacenamiento, a este arreglo especial se le llama MATs.
- **Modo de booteo:** Este es el modo de programación (o Flashing) y para este se necesita establecer una comunicación SCI con el programa FDT desde una PC o un Host que contenga un programa para controlar el dispositivo. En este modo se programa el arreglo de memoria llamado MATs y la tasa de transferencia es ajustada por el programa en el Host o PC.
- **Modo de ejecución:** en esta modalidad, el microcontrolador ejecuta el programa alojado en la memoria flash interna que posee, la ejecución de un programa solo puede ser interrumpida por un Reset, NMI o Shutdown (fuente de alimentación desconectada).

2.1.2 Configuración de los pines.

La memoria flash es controlada a través de pines, la tabla de la figura 2.2 muestra la configuración de cada pin requerido en el proceso de booteo y en el proceso de ejecución.

Pin Name	Abbreviation	Input/Output	Function
Reset	RES	Input	Reset
Flash programming enable	FWE	Input	Hardware protection when programming flash memory
Mode 2	MD2	Input	Sets operating mode of this LSI
Mode 1	MD1	Input	Sets operating mode of this LSI
Mode 0	MD0	Input	Sets operating mode of this LSI
Non-maskable interrupt	NMI	Input	Sets operating mode of this LSI
Transmit data	TxD1	Output	Serial transmit data output (used in boot mode)
Receive data	RxD1	Input	Serial receive data input (used in boot mode)

Figura 2. 2: Configuración de pines en el modo de booteo.

Para el caso de interés, únicamente se tomará en cuenta el modo de booteo y el modo de ejecución de programa. En el primero, el microcontrolador habilita un puerto de conexión como lo muestra la tabla de la figura 2.2. Este es el puerto de comunicación SCI conocido también como COM1 del microcontrolador.

MD0, MD1, MD2 y FWE:

Entre los pines se encuentran los pines MD0, MD1, MD2 y FWE los cuales seleccionan el modo de operación y se deben de configurar para seleccionar el modo de operación

requerido por el usuario. Para mayor detalle vea la tabla de la figura 2.3 en la cual se muestra como se seleccionan los diversos modos validos para este microcontrolador.

Cuando se selecciona el modo de booteo se debe de tomar en cuenta que ninguna interrupción, como NMI (interrupción no mascarable) o Reset, se ejecute durante el proceso.

Para mayor detalle en el proceso de booteo o de otros modos de operación y manipulación de la memoria flash, lea la sección 18 del manual de hardware del H8/3069F [B1].

	Mode						
Pin	Reset state	On-chip ROM invalid mode*	On-chip ROM valid mode*	User program mode	User boot mode	Boot mode	PROM mode
RES	0	1	1	1	1	1	1
FWE	0/1	0	0	1	1	1	1
MD0	0/1	0/1	1	1	1	1	0
MD1	0/1	0/1	0/1	0/1	0/1	0/1	0
MD2	0/1	0	1	1	0	0	0
NMI	0/1	0/1	0/1	0/1	0	1	0/1

Note: *Modes 1 to 4 are on-chip ROM invalid modes.
Modes 5 and 7 are on-chip ROM valid modes. For details, see section 3, MCU Operating Modes.

Figura 2. 3: Modo de operación según configuración de pines.

Note que el pin FWE (Flash Writing Enable) es usado para proteger la memoria flash interna del microcontrolador por lo tanto es necesario considerarlo en el diseño del diagrama. Su función será que al momento de estar en modo de booteo el pin sea puesto a uno para poder programar la memoria flash, y al momento de que se ejecute un programa, se debe proteger la memoria flash para evitar un posible daño o bien si es requerido por el usuario también se puede acceder a ella a través del modo de ejecución.

El arreglo más sencillo para poder colocar un pin en dos estados diferentes es haciendo el circuito que se muestra en la figura 2.4, el cual muestra un microswitch y una resistencia, que logran realizar esta función.

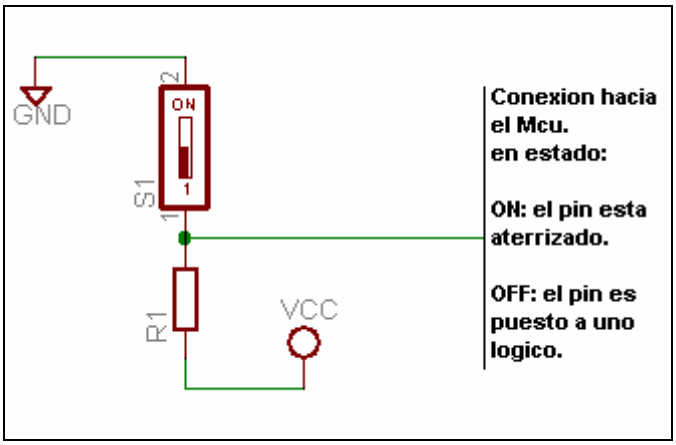


Figura 2. 4: Conexión de pines de configuración hacia el microcontrolador.

Cuando el microswitch esta en estado ON realiza la unión entre los extremos del microswitch provocando la conducción a través de este. Esto genera que el extremo en el cual se conecta el GND tenga prioridad para que se de el flujo de corriente.
Note que la parte del switch donde se coloca Vcc lleva consigo una resistencia en serie, esto genera una oposición al flujo de corriente por lo tanto la manera más sencilla de fluir es a través del extremo de GND, generando un **cero** lógico en el pin.

Por otra parte al conectar el microswitch en estado OFF, la resistencia que se genera entre el punto de unión del microswitch y el pin con GND es infinita, por eso el flujo se producirá del extremo donde se conecta Vcc, obteniendo un **uno** lógico en el pin.

Un arreglo similar al del pin FWE, suele realizarse para los pines de control como MD0, MD1 y MD2: una resistencia y un microswitch para cada pin. De esta manera se logra colocar el modo de operación al dispositivo. Recuerde que mediante estos pines el dispositivo puede ser iniciado en modo de booteo o en modo de ejecución.

NMI y Reset:

Por otro lado, en general el pin NMI esta configurado de manera tal que al energizar el microcontrolador, se genere una interrupción al registrase un cero lógico en su pin de puerto. Para evitar una interrupción en el proceso de booteo, es necesario asegurar que el pin permanezca a uno. Más aun, debemos tener la seguridad que al inicializarse el estado lógico, su valor siempre será de uno. La figura 2.5 muestra un ejemplo básico de un circuito RC el cual retarda la polarización de cualquiera de los dos polos de una fuente de energía.

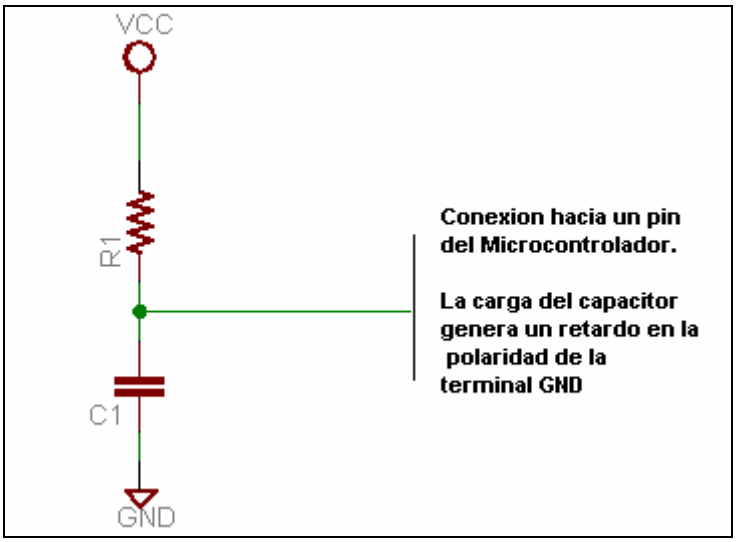


Figura 2. 5: Circuito de retardo en la polarización.

La constante de tiempo 1/RC nos da el tiempo en el cual el capacitor termina de cargarse y de esta manera retardar el tiempo de la polarización hacia tierra.
Este tipo de diseño suele ocuparse con frecuencia para entradas que necesitan estar polarizadas, pero que no únicamente necesitan tener un valor especifico, sino que necesitan un transciende para poder evitar rebotes (o ruido eléctrico) y funcionar de manera correcta.

Además el circuito sirve como filtro pasabajas, así se puede eliminar una señal que haga que el dispositivo realice funciones fuera de control tal y como lo muestra la figura 2.6:

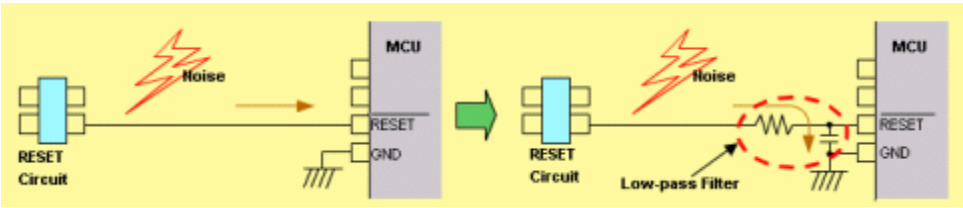


Figura 2. 6: Filtro pasabajas para evitar un reset no deseado.

Generalmente se utilizan para pines como NMI y Reset. Ahora, para aprovechar el recurso de tener un solo circuito que gobierne ambos pines (Reset y NMI) es necesario hacerle ciertas modificaciones, con el fin de ahorrar materiales. La figura 2.7 muestra el circuito utilizado para retardar la polaridad en ambos pines del dispositivo.

El circuito RC consta de unas resistencias y un capacitor, la carga del Reset se realiza a través de este arreglo y tiene retardo para polarizarse con GND. Un switch tipo push-button, es acondicionado para poder generar un Reset en cualquier momento que el usuario lo requiera, el diodo evita que al pulsar el switch se genere una interrupción NMI en vez de un Reset. Cuando se genera una NMI el push-button esta abierto por lo tanto no se genera ningún Reset.

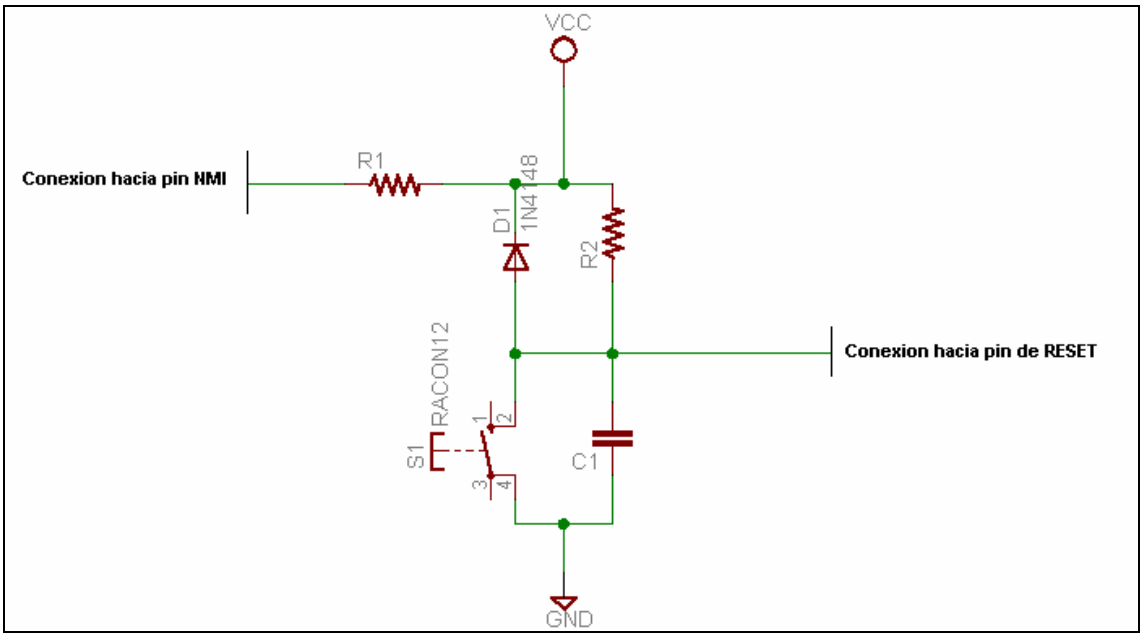


Figura 2. 7: Conexión Reset y NMI. Retardo a la polaridad RC.

Comunicación SCI.

Finalmente, la comunicación SCI debe ser llevada a cabo por un conversor de valores TTL a valores comunes de puerto serial, esto se lleva a cabo mediante un arreglo común de conversión de niveles con un integrado dedicado a este tipo de tratamiento. El dispositivo utilizado es el dispositivo de la serie ADM3202.

La comunicación en el proceso de booteo se lleva a cabo mediante el puerto de comunicación 1 con nomenclatura TxD1 y RxD1. Para recepción y transmisión de datos de manera serial utilizando el protocolo RS232 debe de colocarse un integrado que acopla la salida serial de una PC normal de aproximadamente 15 y –15 Voltios hacia 5 y –5V, los cuales son valores TTL estándar para comunicación serial entre circuitos integrados.

La herramienta que se tiene para este tipo de operación es el integrado con la serie ADM3202AN (de Analog Devices) el cual se detalla mejor en la figura 2.8, la cual muestra la vista física y ciertas características técnicas que el fabricante brinda.

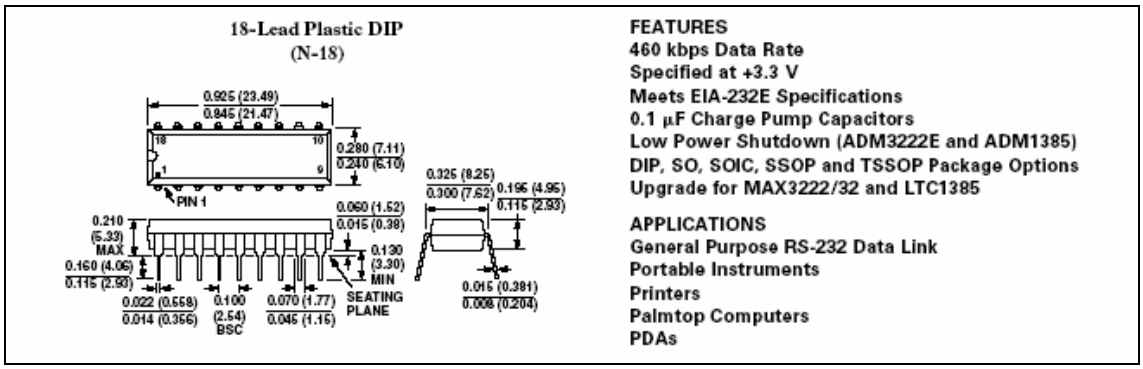


Figura 2. 8: Características del ADM3202.

El fabricante recomienda usar un circuito apropiado y las conexiones respectivas mostradas en la figura 2.9 de manera que la transferencia de datos serial se lleve a cabo de manera exitosa y que pueda usarse como convertidor de medio para la comunicación con un Host o una PC.

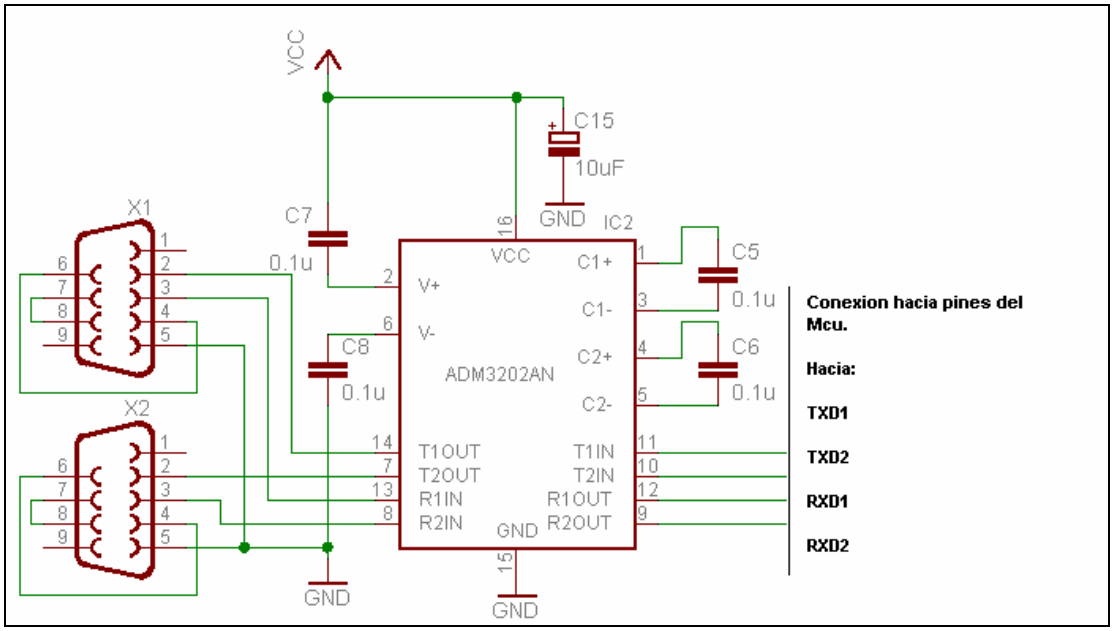


Figura 2. 9: Comunicación serial y conexión PC-Mcu.

Un conector hembra DB9 es colocado para poder hacer la conexión de la manera más práctica posible.

2.2 Diseño del sistema mínimo

Para diseñar el sistema mínimo del microcontrolador es necesario conocer los elementos indispensables para que el dispositivo funcione.

Primero se debe de tomar en cuenta cual es la configuración de los pines para que el microcontrolador funcione. Lo más importante son los pines que alimentan el dispositivo. Luego fijar el valor del oscilador que tendrá el dispositivo, para este caso el microcontrolador trabajara a una velocidad nominal de 25MHz (este dato se especifica en le manual de hardware H8/3069F sección 19) fijamos la alimentación, la cual debe ser de 5V (este parámetro es especificado en el manual de hardware sección 21).

Una vez establecidos los valores del oscilador, la alimentación y la identificación de cada pin físico, se debe diseñar el circuito que controle la operación de estos pines.
En la figura 2.10 se muestra la conexión del oscilador hacia el dispositivo.

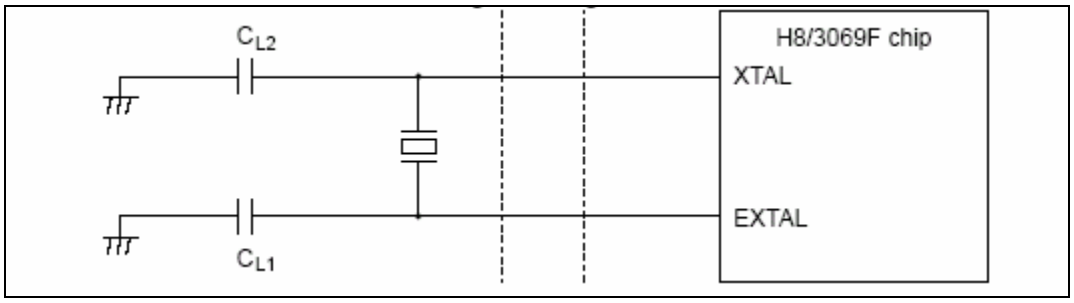


Figura 2. 10: Conexión del cristal oscilador.

El oscilador es un cristal de cuarzo el cual oscila a una frecuencia natural de 25MHz y es necesario colocar también capacitares para evitar los rizos generados por la oscilación del cristal. Con la ayuda de los capacitores se logra estabilizar el reloj y evitar que señales de ruido interfieran en el funcionamiento del subsistema. El fabricante recomienda colocar capacitores de 10 pF para un buen desempeño.

Otro pin importante para el diseño es el pin VCL que sirve como una salida pasabaja interna. La figura 2.11 muestra la conexión que debe de tener el pin.

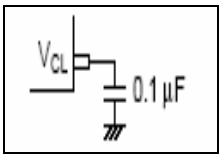


Figura 2. 11: Conexión de pin VCL.

El fabricante recomienda que para la operación correcta del dispositivo se debe conectar un capacitor de 0.1µF entre este pin y tierra (GND).

Diferentes tipos de ruido suelen ser provocados sin querer en un circuito, varias de las señales generadas se muestran en la figura 2.12:

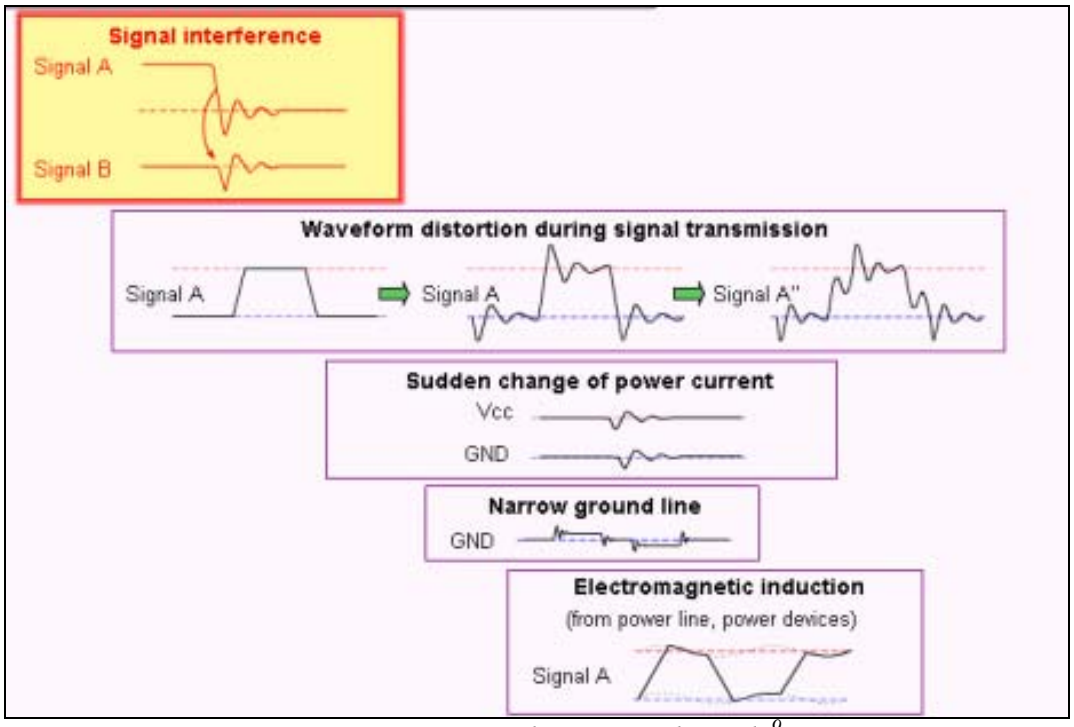


Figura 2. 12: Señales típicas de ruido[9].

La fuente de poder debe estar diseñada para poder soportar voltajes altos sin que se dañe el dispositivo, por esta razón el uso de una fuente sencilla es necesaria.
La fuente consta de un regulador comercial de 5V de la serie L7805, diferentes valores de capacitores son colocados en la fuente, para impedir diferentes señales de ruido que se presenten en la red eléctrica.
La figura 2.13 muestra el diseño de la fuente de poder para el sistema mínimo.

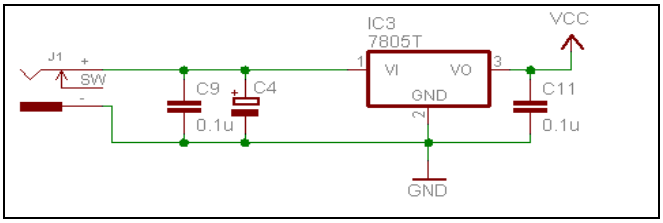


Figura 2. 13: Fuente de poder.

El ruido eléctrico puede ser de dos tipos: ruido por radiación y ruido por recepción.
Una meta común en el diseño con microcontroladores es evitar ruido eléctrico para el buen desempeño de los dispositivos. Según la IEEE ruido es "Señales eléctricas inservibles que producen efectos indeseables en circuitos de control en los cuales ocurre".

Para poder operar correctamente, los microcontroladores deben de estar sincronizados con una señal de reloj la cual fijará su ritmo de trabajo. Estas señales de reloj pueden variar en frecuencia dependiendo de la capacidad del dispositivo y/o de los dispositivos

[9] Para mayor detalle hacer referencia a Renesas Interactive Course [B7].

con los cuales vamos a estar trabajando y tienen varias componentes de frecuencia: entre más alta sea la frecuencia, más componentes habrán.
Ahora, es posible que una señal de ruido (en forma de rizo) afecté una de estas componentes de tal forma que la señal completa de reloj de vea modificada e inclusive se vuelva peligrosa para el sistema.

Por ejemplo, si el rizo decrece de 10nsec/5V a un valor comprendido de 2 a 5nsec/5V, la señal contendrá componentes de frecuencia arriba de los 100MHz.
La gráfica de la figura 2.14 muestra los transciendes provocados por un rizo de estas características.

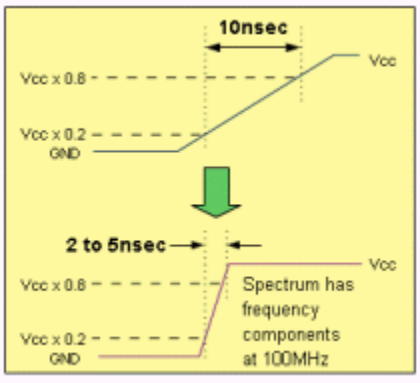


Figura 2. 14: Transciendes de rizo.

La impedancia de los capacitores es importante para el diseño de circuiteria. Es recomendado tomar el dato descrito para la fuente de reloj; vea que esta genera una frecuencia alta y los rizos de este cristal son grandes y las componentes son altas: supongamos un rizo arriba de los 100 MHz, la colocación de un capacitor de 10 pF tendrá como efecto lo siguiente:

$$Z= (2\pi \times 100 \times 10^{6} \times 10 \times 10^{-12})^{-1}$$

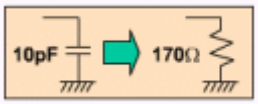


Figura 2. 15: Impedancia del capacitor de 10 pF a 100MHz.

Las aplicaciones de los capacitores en el diseño suelen catalogarse en de acuerdo a la capacitancia que tienen.

Aplicación.	Elección para el diseño.
Capacidad pequeña (0.01 – 0.1uF); instalar entre Vcc y GND de un IC	Capacitor cerámico. Baja capacitancia, pero buena respuesta en frecuencia.
Alta capacitancia (0.1 –10uF); instalar entre Vcc y GND de la fuente	Capacitor de tantalio Pequeña capacitancia menor aun que el electrolítico, pero con mejor respuesta en frecuencia.
Extra alta capacitancia (1 – 100uF); instalar entre las terminales de las líneas de poder.	Electrolíticos de aluminio Respuesta en frecuencia pobre, pero menos caro y de más larga capacitancia.

Tabla 2. 1: Clasificación de los capacitores para el diseño contra el ruido.

Los capacitores ocupados para evitar rizos debido a altas frecuencia no se encuentran detallados en la tabla 2.1 es por que estos capacitores reciben el nombre de Bypass y trabajan en una zona de frecuencia diferente a los capacitores comunes. Esto lo describe mejor la figura 2.16 la cual muestra el diagrama de Bode que es la respuesta en frecuencia para comparar el comportamiento de los capacitores comunes con los Bypass.

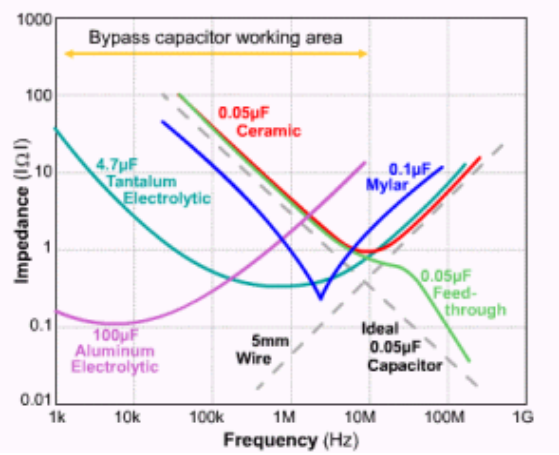


Figura 2. 16: Respuesta en frecuencia de los capacitores bypass.

Note que el capacitor de Bypass generalmente se mantiene en el mismo valor de impedancia. Esto es importante ya que una impedancia alta podría alejar el pin de un valor de voltaje esperado, o bien un cambio brusco en la impedancia también puede generar emisión de ruido como corriente fluctuante.

Por esta razón se recomienda que al oscilador se le conecten capacitores de 10 pF.

Tomando en cuenta todos los pines antes especificados y las necesidades requeridas en cada etapa de conexión, se procede a diseñar el circuito, el resultado es el diagrama mostrado en la figura 2.17:

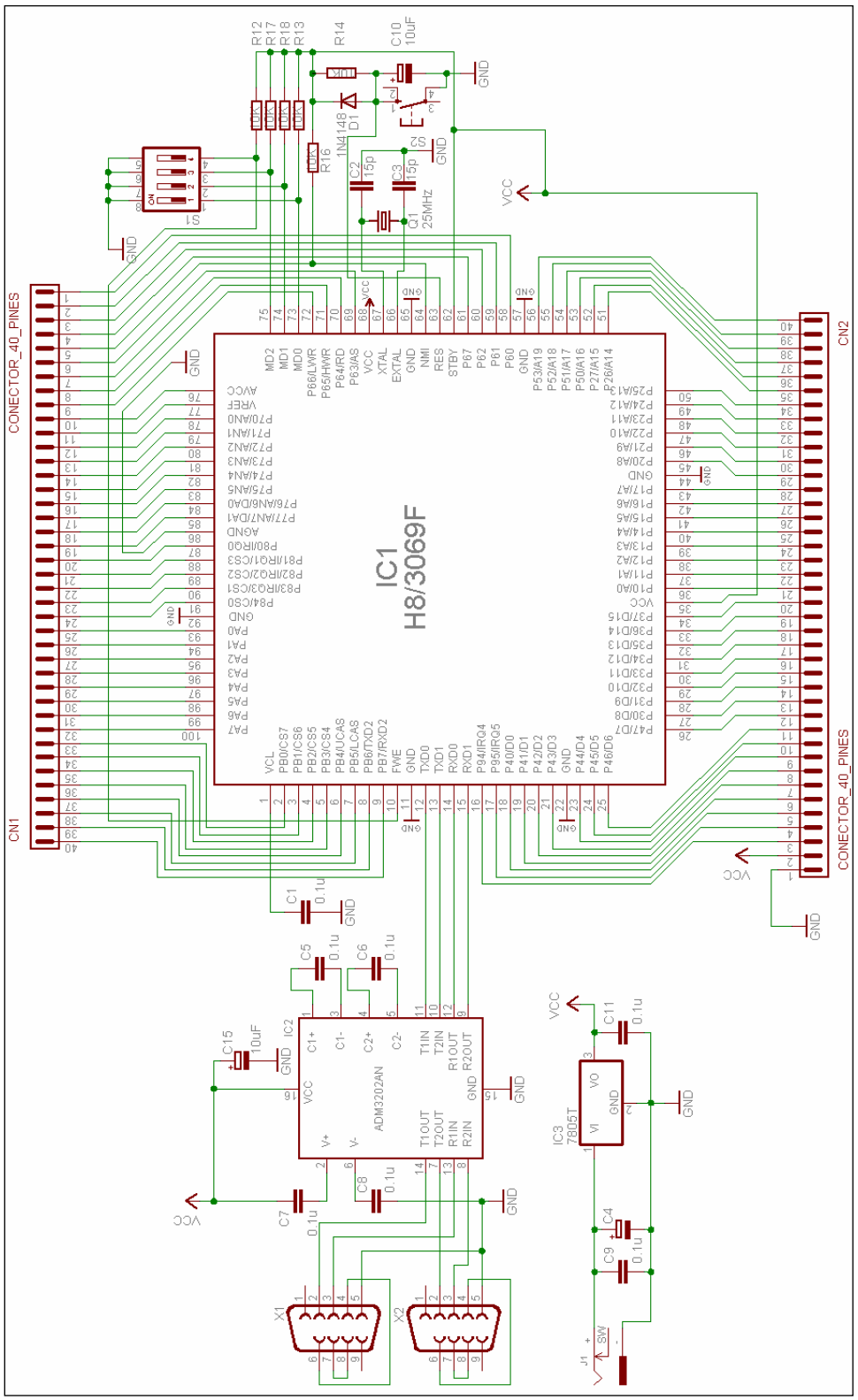


Figura 2. 17: Diagrama esquemático del sistema mínimo.

En el diagrama esquemático se puede observar que todos los pines mencionados con anterioridad son utilizados. Dos conectores de 40 pines son agregados para poder manipular las entradas y salidas físicas del dispositivo.
Todos los puntos de tierra común se aseguran en un solo nodo en el circuito para generar una mayor estabilidad.

Para el diseño de circuiteria se utilizó un programa comercial, el proceso de instalación hacia una PC es omitido, ya que la instalación del mismo suele ser muy fácil y comprensible. Este programa es el EAGLE y para poder diseñar un circuito en este programa basta con tenerlo instalado en una PC normal. Al abrir la pantalla principal aparecerá una ventana como la siguiente.

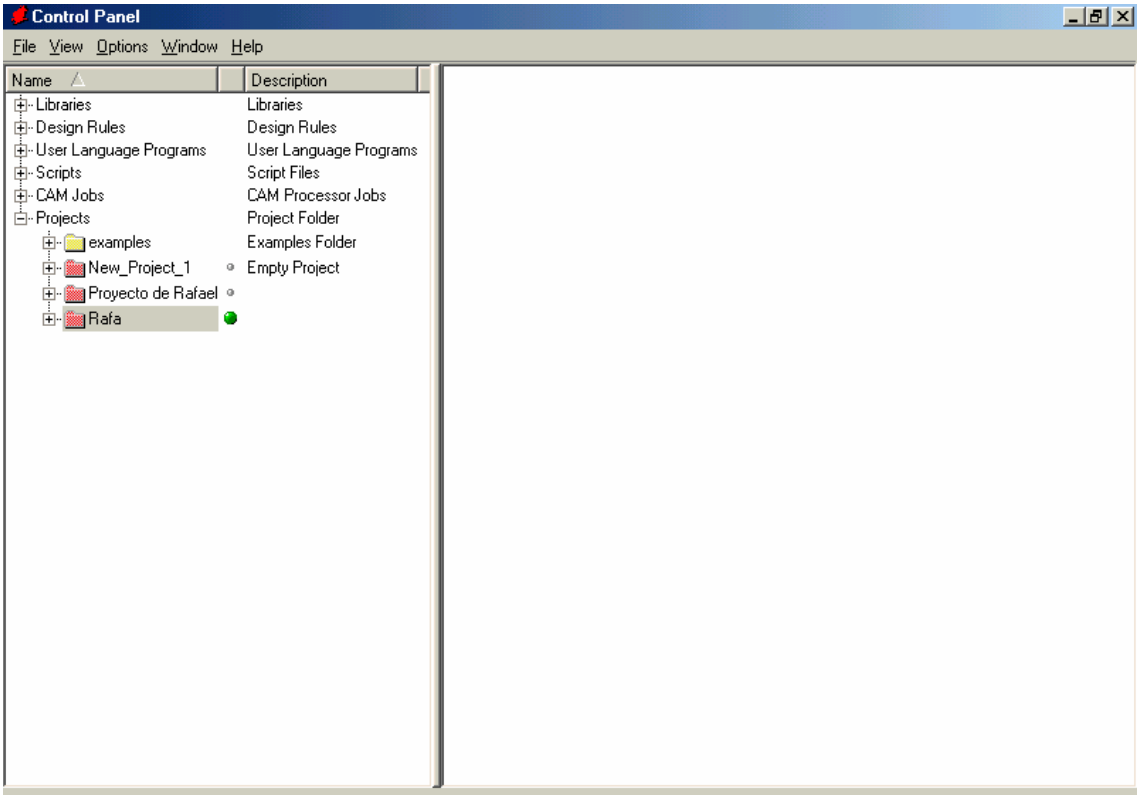


Figura 2. 18: Panel de control de EAGLE.

Para poder generar un proyecto debemos dar clic al menú *File* luego en *New* y por ultimo *Project.* De esta manera se logra crear un proyecto, al cual se le puede agregar un diagrama esquemático que luego puede ser procesado por el programa para crear un impreso tentativo al diseño.

Para poder crear un diagrama esquemático en le proyecto basta con dar un clic derecho sobre la carpeta del proyecto que se quiere modificar, tal y como lo muestra la figura 2.19

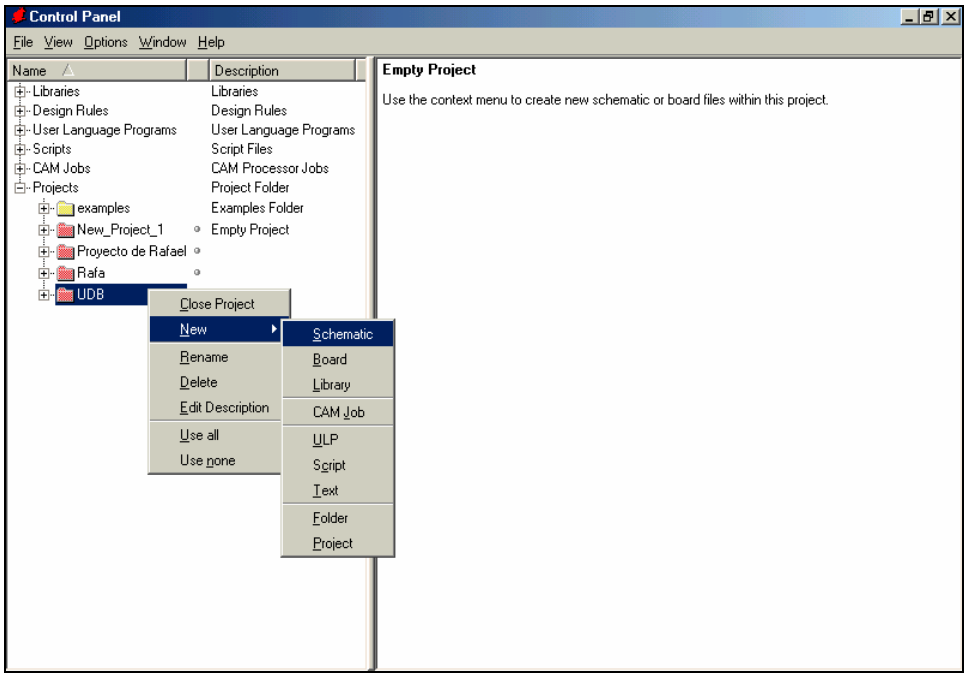


Figura 2. 19: Creación de un diagrama esquemático en un proyecto de EAGLE.

Tomemos por ejemplo que se requiere crear en impreso para los minidips conectados a los pines FWE, MD0, MD1 y MD2 tal y como lo muestra el diagrama de la figura 2.17. Para colocar elementos sobre el editor de esquemáticos en el EAGLE es necesario presionar el botón ADD (ADd Devices), el icono se muestra junto con otros iconos comunes en la figura 2.20

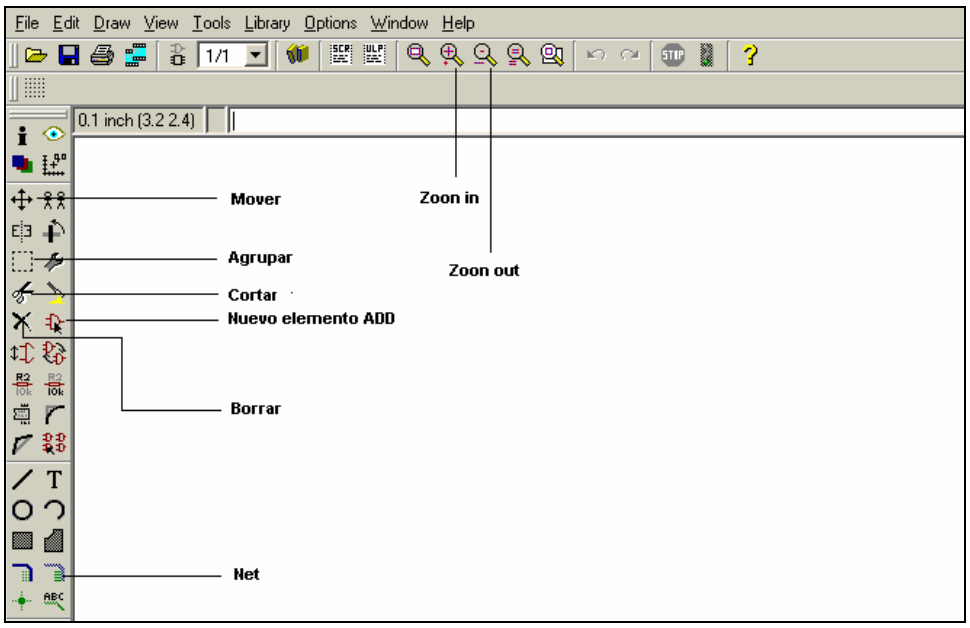


Figura 2. 20: Ubicación de botones más importantes en ventana de dibujo de EAGLE.

Palabras clave como “Resistor”, “Capacitor”, “GND”, “Vcc”, etc. Suelen ser ocupados para llamar elementos comunes en la ventana que se despliega al presionar ADD. Ahora,

para poder dibujar las redes que interconectan elementos dentro de un circuito, lo recomendable es presionar el botón *Net* que se detalla la ubicación en la figura 2.20.

La mayoría de los elementos se encuentran disponibles en las librerías de EAGLE, la única librería que no tiene disponible es la librería del microcontrolador, pero esta puede ser descargada de manera gratuita desde Internet.

Ahora colocando los dispositivos en la ventana de dibujo y haciendo las conexiones necesarias se obtiene un diagrama como el que se describe en la figura 2.21:

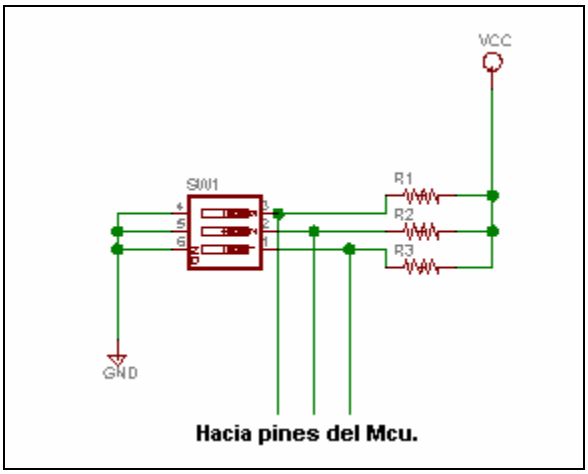


Figura 2. 21: Circuito realizado en EAGLE.

Ahora para generar un impreso alternativo a este diseño de esquemático solo se debe hacer clic en el botón *Board* el cual se presenta en la figura 2.22:

Figura 2. 22: Botón Board del EAGLE.

Al presionarlo el programa desplegará una figura como la que se muestra a continuación:

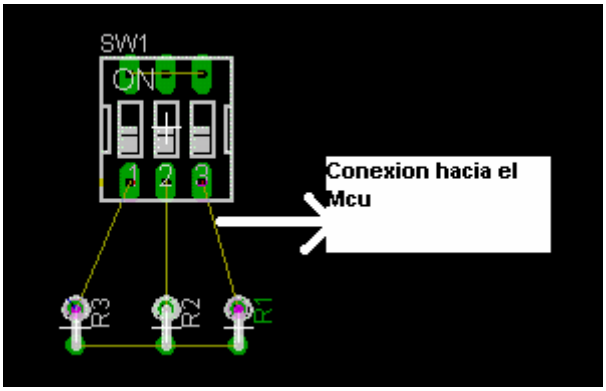


Figura 2. 23: Impreso tentativo del ejemplo.

Cuando el programa realiza el auto impreso, se da la impresión de desorden, más sin embargo el uso de herramientas como mover, zoom in y otras herramientas del espacio de trabajo del EAGLE, permiten personalizar el impreso. A este proceso se le llama "enrutado manual".

Finalmente, con este enrutado, debe diseñarse el impreso de manera tal que no represente mucho problema para el momento de soldar. La anchura de las pistas y los elementos deben estar adecuados a sus dimensiones exactas, y luego se procede con la realización del impreso para el sistema mínimo.

Este diseño de impreso se muestra en la figura 2.24:

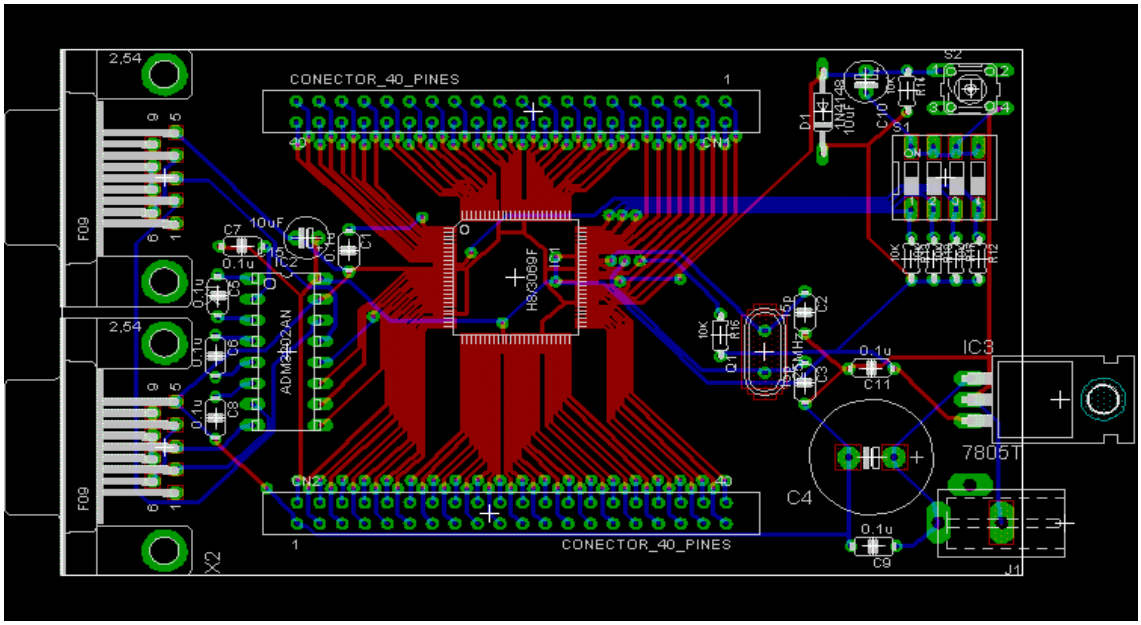


Figura 2. 24: Circuito impreso.

En la figura 2.24 se observa que el color de los dispositivos en blanco, el color de las pistas de arriba es rojo, las pistas de abajo se representan por el azul, y los pads y vias se representan por el color verde.

Para definir los modos de operación del microcontrolador se debe hacer de acuerdo a los minidips. Las figuras 2.25 y 2.26 muestran los dos tipos de configuración a utilizar.

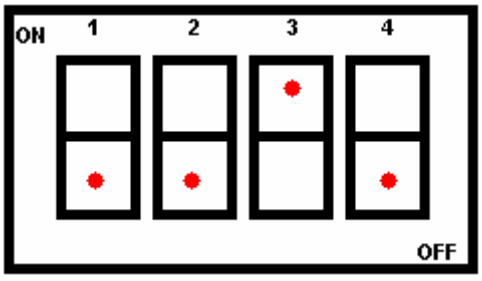


Figura 2. 25: Posición de minidips en modo de booteo.

En la figura 2.25 se especifica la posición para que el microcontrolador al detectar voltaje en sus terminales establezca el modo de booteo. Los puntos rojos representan el estado de cada Switch.

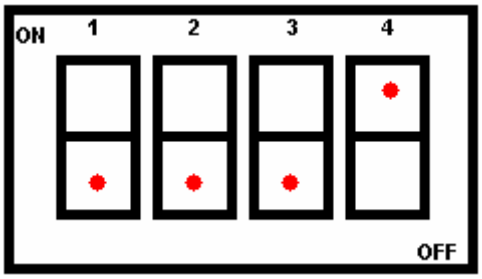


Figura 2. 26: Posición de minidips modo de ejecución.

La figura 2.26 muestra la posición de cada Switch en el minidips para establecer el modo de ejecución. En este modo el programa alojado en la memoria flash del dispositivo es ejecutado por el CPU interno. Recuerde que el microcontrolador H8/3069F no puede ser operado en modo normal, únicamente en modo avanzado y como no será seleccionado para tener memoria externa a menos que el usuario lo requiera debe conectarse en Single chip Advanced Mode. En el cual habilita todos sus puertos como I/O y no como entrada de bus de datos.

La tabla de la figura 2.27 muestra los tipos de configuración que se pueden realizar a través del minidips colocados en los pines de modo del dispositivo.

Operating Mode	Mode Pins			Description			
	MD_2	MD_1	MD_0	Address Space	Initial Bus Mode*1	On-Chip ROM	On-Chip RAM
—	0	0	0	—	—	—	—
Mode 1	0	0	1	Expanded mode	8 bits	Disabled	Enabled*2
Mode 2	0	1	0	Expanded mode	16 bits	Disabled	Enabled*2
Mode 3	0	1	1	Expanded mode	8 bits	Disabled	Enabled*2
Mode 4	1	0	0	Expanded mode	16 bits	Disabled	Enabled*2
Mode 5	1	0	1	Expanded mode	8 bits	Enabled	Enabled*2
—	1	1	0	—	—	—	—
Mode 7	1	1	1	Single-chip advanced mode	—	Enabled	Enabled

Notes: *1 In modes 1 to 5, an 8-bit or 16-bit data bus can be selected on a per-area basis by settings made in the area bus width control register (ABWCR). For details see section 6, Bus Controller.

*2 If the RAME bit in SYSCR is cleared to 0, these addresses become external addresses.

Figura 2. 27: Tipos de configuración según pines de modo.

Al seleccionar este tipo de modo se dispone de un banco de memoria como el representado en la figura x2.28

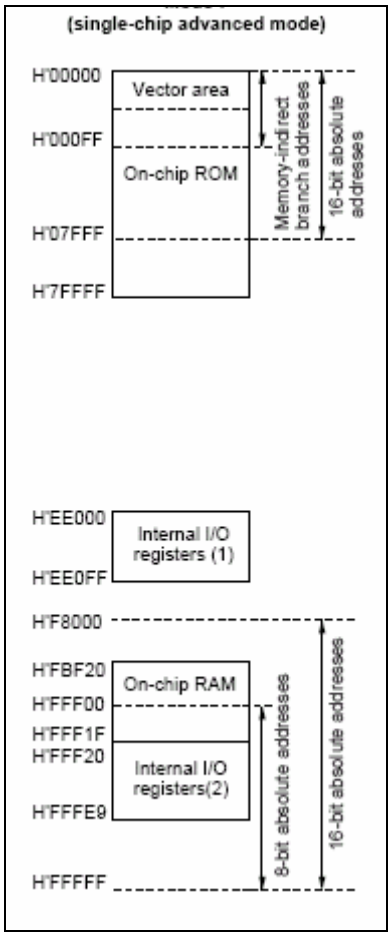


Figura 2. 28: Banco de memoria.

El tipo de banco se puede seleccionar a través del bit EMC (Expantion Memory Map Control) del BCR (Bus Control Register). Como inicialmente el bit esta puesto a uno en el registro BCR, el banco resultante es el de la figura 2.28. Para mejor detalle vea la sección 6.2.5 del manual de hardware del H8/3069F.

2.3 Consideraciones de seguridad de circuitos impresos[10]

Como se menciono anteriormente, existen dos tipos de ruido que pueden afectar un circuito electrónico: ruido de radiación, el cual es emanado del circuito mismo y el ruido de recepción.
Las llamadas incompatibilidades electromagnéticas (EMC) toman en cuenta estos dos tipos de ruido pero con acercamientos diferentes: EMI y EMS en donde se busca reducir

[10] Para mayores detalles hacer referencia a los cursos interactivos de Renesas [B7]: www.renesasinteractive.com

las interferencias electromagnéticas y la susceptibilidad electromagnética respectivamente. En el primer caso (EMI), se busca eliminar el ruido que irradia el circuito y en el segundo se pretende eliminar el ruido externo que podría afectar el sistema.
En este apartado nos vamos a enfocar en la seguridad del sistema y como protegerlo de este ruido externo y sus efectos.

El ruido de fuentes externas puede causar que el microcontrolador trabaje fuera de control, esto podría tener consecuencias solo momentáneas o fatales tanto para el equipo como para el usuario. Es por esto que es necesario, sin importar si los sistemas trabajan con baterías o con alimentación externa, que están protegidos contra estos dañinos EMS.

Separación física:
Una de las primeras medidas que debemos tomar es la reducción de la interconexión entre componentes. Entre más cerca se encuentren estos menos susceptibilidad habrá y menos posibilidad existirá que estos cables o pistas funcionen como antenas.

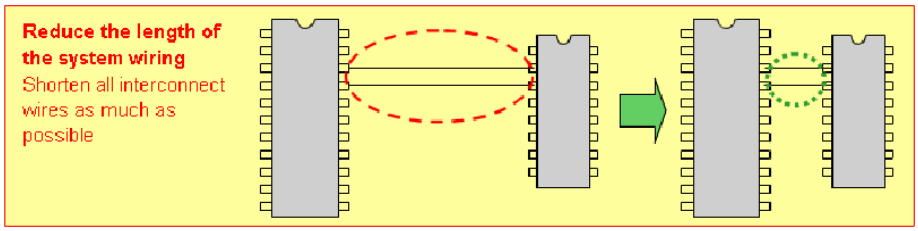


Figura 2. 29: Reducción de la distancia entre componentes.

Con esta misma idea es que se recomienda que se utilicen encapsulados pequeños, estilo QFP, ya que las conexiones internas son más pequeñas:

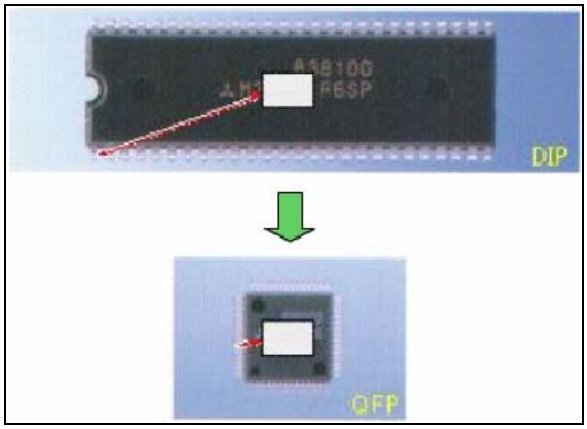


Figura 2. 30: Conexiones internas de los diferentes encapsulados.

Capacitores de Bypass:
La conexión de estos capacitares de Bypass debe ser también lo mas cercano posible al dispositivo y en un lugar en donde las líneas de Vcc y GND sean isométricas o paralelas. Además, si es posible, es mejor si se selecciona un dispositivo con diferentes juegos de

alimentación para poder así poner varios de estos capacitares, siempre siguiendo la misma forma de conexión.
Para un mejor filtrado es posible utilizar un arreglo de varios capacitares de Bypass de diferentes valores y materiales como lo muestra la figura siguiente:

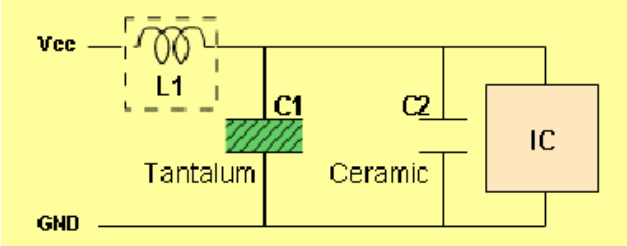


Figura 2. 31: Arreglo para mejor filtrado.

Esta combinación de materiales y valores junto con el inductor permite tener una mejor protección EMS.

Conexiones externas:
Si el microcontrolador será conectado a dispositivos externos y no se requiere una alta tasa de transferencia, es posible reducir la calidad total de cables remplazando el bus paralelo (múltiples cables) por un bus serial (dos cables).

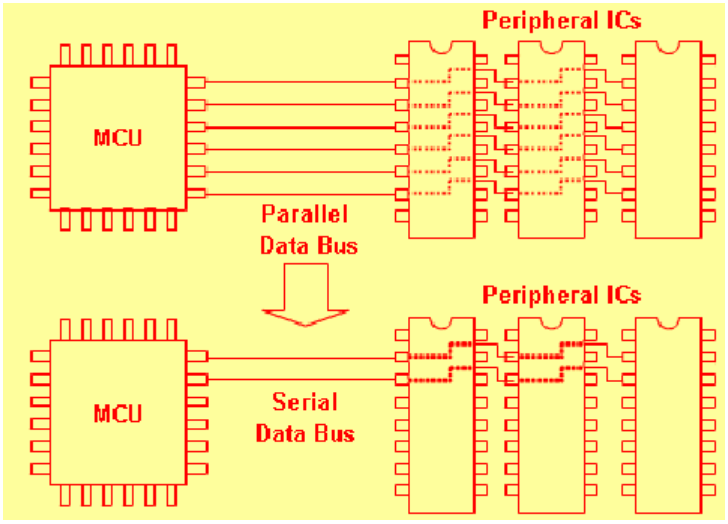


Figura 2. 32: Reemplazo de bus paralelo por bus en serial.

En caso que una tasa de transferencia alta sea necesaria, uso obligatorio de un bus en paralelo, es recomendable poner los dispositivos lo más cerca posible del microcontrolador y procurando que las líneas de conexión sean paralelas la mayor parte del trayecto.

Por otro lado, es preferible usar elementos con este tipo de dispositivos integrados (memoria por ejemplo), así eliminamos el cableado externo asociado.

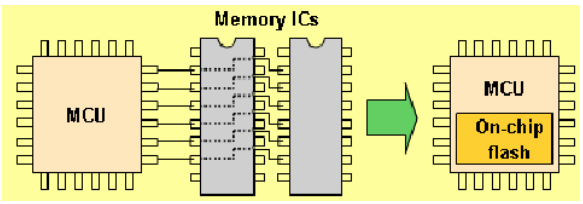


Figura 2. 33: Microcontrolador con dispositivos integrados.

Si estos dispositivos externos vana ser memorias es aun mas recomendable utilizar elementos que tengan buena capacidad, aun cuando esto signifique incurrir en mayores gastos por elemento pero la reducción de ruido es significativa. Inclusive, es mucho mejor utilizar ejemplo dos microcontroladores con memoria integrada que uno solo y agregar más memoria.

Display de LED:
Para el caso que tengamos que montar un display de LED en la tarjeta, es recomendado colocar los resistores a 2cm o menos del microcontrolador para poder así evitar los problemas de ruido que pueden ser capturados por las líneas en el trayecto (las resistencias atenúan el ruido).
Teóricamente los LED deberían de aislar el circuito por el hecho que sus terminales solo están unidas por luz. Sin embargo la cantidad de ruido capturable es considerable debido a las capacitancias parásitos.

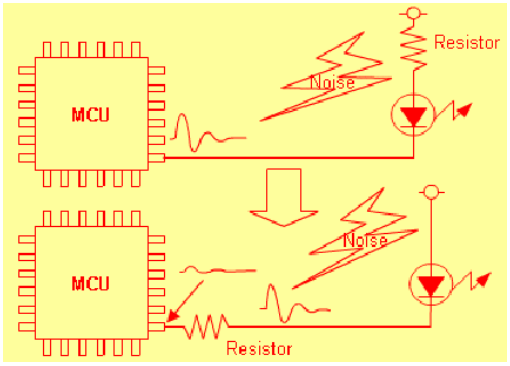


Figura 2. 34: Ubicación de resistores.

Ahora, si estamos conectando el microcontrolador con dispositivos externos y estamos usando resistencias Camping, cuya función es igualar la impedancia del cable para tolerar mejor el ruido, es recomendable colocar estos lo más cerca posible del microcontrolador:

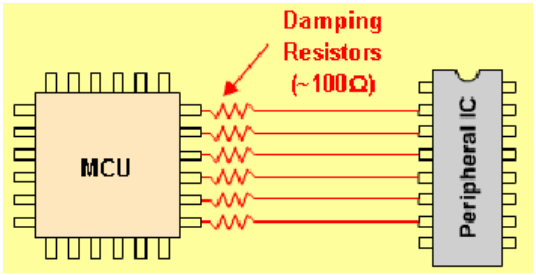


Figura 2. 35: Resistencia Camping.

Conexión con el oscilador:
En cuanto a las conexiones del oscilador, debemos de tomar las siguientes consideraciones:

1. No debemos de interconectar líneas de señales con líneas de reloj:

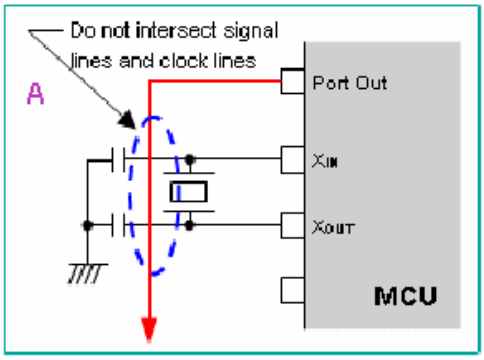


Figura 2. 36: Eliminar cruce de líneas de reloj y señal.

2. Es recomendable no usar la líneas de señales o salidas que se encuentran localizadas a la par de los pines del reloj:

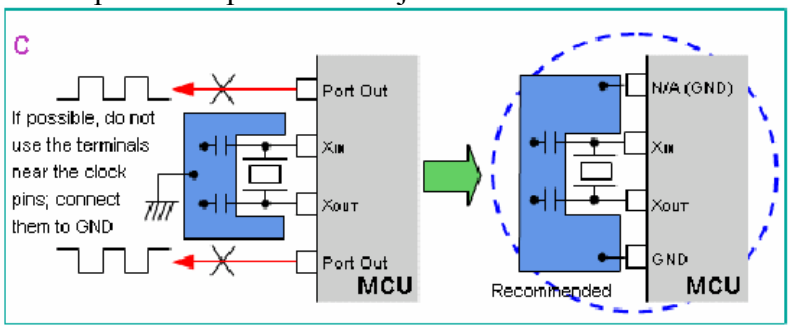


Figura 2. 37: Líneas vecinas inutilizadas.

3. Debemos mantener las líneas que llevan altas corrientes y altas frecuencias alejadas de las líneas de reloj.

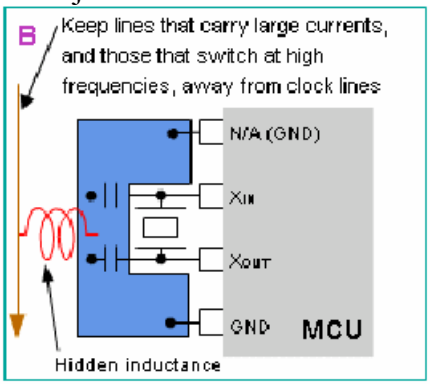


Figura 2. 38: Separar alta frecuencia y alto voltaje de la señal de reloj.

4. debemos procurar tener grandes áreas aterrizadas para separar la dualidad de relojes en caso que tengamos dos diferentes, de este modo evitaremos la interferencia entre las señales.

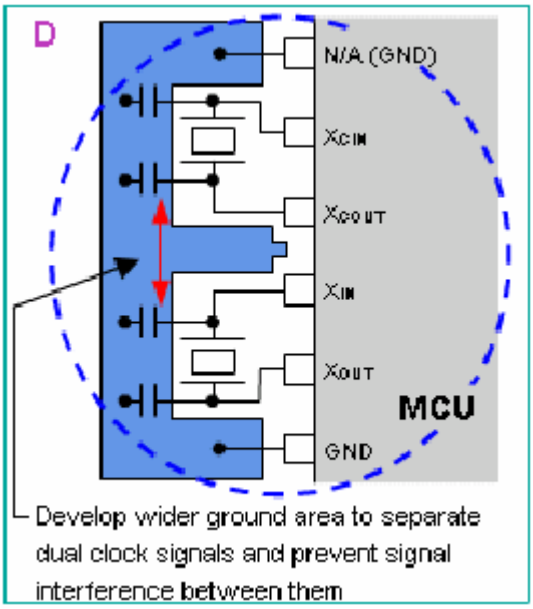


Figura 2. 39: Dualidad de relojes.

2.4 Consideraciones de elaboración de impresos y soldaduras[11]

Un circuito impreso es un modulo en donde todos los componentes electrónicos se encuentran interconectados a través de un conductor impreso en una superficie aislante conocida como substrato. Cada componente es colocado y soldado a la tarjeta mediante un metal llamado estaño.

Podemos tener tres tipos de circuitos impresos: los de una solo cara, los de dos caras y los de múltiples capas. En el primer caso, los dispositivos están colocados en una de las caras y las pistas en la cara opuesta. Para la segunda opción, tenemos que los dispositivos están en una cara pero las pistas están colocadas en ambas. Para la última opción, tenemos que las pistas están separadas por diferentes capas de material aislante. En estas dos ultimas, para poder interconectar las diferentes capas, es necesario hacer puentes a través del material aislante.
En el caso del sistema mínimo desarrollado, tenemos un impreso de dos caras.

Por otro lado, para el montaje de los dispositivos en la tarjeta, tenemos dos métodos diferentes. El tradicional, en el cual es necesario abrir hoyos en la tarjeta para poder introducir los pines de los dispositivos para luego soldarlos, y el método de superficie, en donde los elementos son pegados en la tarjeta y luego son soldados. En ambos caso las soldaduras son las que hacen las conexiones entre las pistas y los dispositivos.
Nuevamente, en nuestro caso, el método implementado fue el tradicional.

[11] Para mayor información hacer referencia al curso de Circuitos impresos: Printed Circuits Board [B8] presentado en los anexos.

No existe un estándar para el diseño de circuitos impresos ya que cada circuito realiza una función en específico y esta diseñado para un espacio determinado. Sin embargo, existen diferentes programas los cuales me facilitan el diseño y la creación de los esquemáticos con los cuales posteriormente se imprimen las pistas en las tarjetas. En nuestro caso se utilizo el programa Eagle para obtener el esquemático de la figura 2.24.

Una vez teniendo dicho esquemático, debemos imprimir su negativo en una lámina transparente de manera que tengamos todas las pistas en negro, para cada una de las caras, y en transparente las partes que representan el cobre que será eliminado en el proceso.
Para poder marcar las pistas de negativo sobre las tarjetas aislantes, las cuales están cubiertas inicialmente con cobre y una película fotosensible, usamos una luz ultravioleta. Esta luz permite la eliminación de la protección del cobre que quedo expuesto; la eliminación de esta protección es realizada con un solvente.

Una vez que tengamos la tarjeta limpia (sin protección en los lugares apropiados) podemos aplicarle una solución ácida la cual eliminará el cobre desprotegido dejando solo las pistas deseadas.

Como resultado tendremos entonces una placa de material aislante con las pistas de cobre deseadas y listas para ser taladrada para luego colocar y soldar los elementos.

El proceso antes explicado es ilustrado en la siguiente figura:

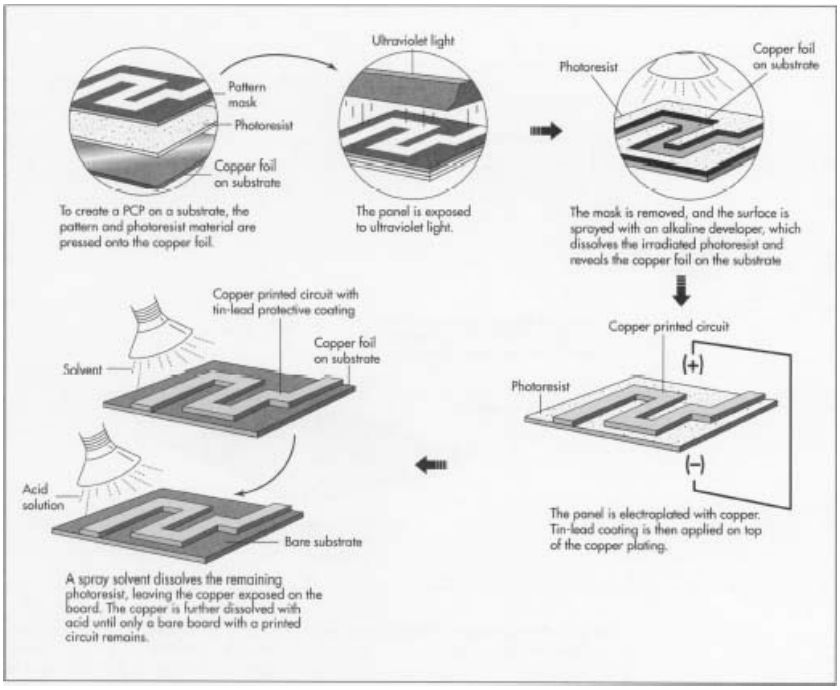


Figura 2. 40: proceso de fabricación de un circuito impreso.

2.5 Consideraciones económicas:

Para la realización de los módulos se llevaron a cabo diferentes etapas: realización del diseño (esquemáticos), quemado de las pistas sobre las tarjetas, colocación de dispositivos, pruebas y reemplazo de elementos dañados.

En los detalles a seguir solo se tomará en cuenta los costos de las últimas tres etapas:

Para el sistema mínimo (precios por tarjeta):

Realización del impreso:	$6	
Conectores RS232	$2	x2
ADM3202	$2	
Capacitores de 0.1 microfaradios	$0.40	x6
Capacitores de 10 microfaradios	$0.50	x3
Resistencias de 10 KiloOhmios	$0.15	x6
Minidips de 4 Switch	$2	x1
Capacitores de 10 picofaradios	$0.70	x2
N4148	$0.35	x1
Capacitores de 1000 picofaradios	$0.80	x1
Conectores de 40 pines (hem.)	$2	x2
Estaño	$1	
H8/3069F y cristal	$20	
Regulador 7805	$1.70	
Mano de obra	8 horas x 3 días	
Total:	**$48.05**	

Para el módulo didáctico (precios por tarjeta):

Realización del impreso:	$12	
Bases para integrado	$0.50	x4
Juego de Jumper	$14	
Operacionales HA17902	$3	
Conectores de 40 pines (mac.)	$2	x2
Conectores de 40 pines (hem.)	$2	x2
Integrado 74245	$3	x3
Pantalla LCD	$8	
Matriz de LED	$2.50	
Resistencias de 150 Ohmios	$0.15	x8
Resistencias de 10 KiloOhmios	$0.15	x4
Regulador 7805	$1.70	
Regulador 7812	$1.80	
Mano de obra	8 horas por 5 días	
Total:	**$63.8**	

Al juntar los gastos de todas las etapas tenemos un total de **$159.90**.

Capitulo Tres: Diseño y Programación

3.1 Generalidades

Las herramientas de diseño, programación/comunicación del microcontrolador, se refieren al tipo de software utilizado para llevar a cabo una acción especifica sobre el microcontrolador (uso de puertos, convertidores A/D, convertidores D/A, temporizadores, etc.). En el presente capítulo se mostrará el uso de los programas utilizados para llevar a cabo dichos procesos, así como también el diagrama de conexión utilizado para la comunicación entre una computadora normal (PC) y el microcontrolador.

Aunque en la actualidad para este tipo de dispositivos existen diversos software libres, sobretodo en Singapur y Japón, en este caso particular se referirá únicamente al software que la empresa fabricante, Renesas, brinda para el público.

Estas herramientas de diseño y programación si bien son de uso público, tienen ciertas limitantes, como por ejemplo para el uso de un programa IDE (Ambiente Integrado de Desarrollo por sus siglas en ingles) se esta limitado a un máximo de 64Kb por archivo de texto y de 30 días de licencia libre para desarrollar aplicaciones. También tenemos el bloqueo de ciertas herramientas que el programa con licencia comprada incluye. Ahora, para el uso propuesto en este documento el programa cumple con las expectativas y permite el desarrollo de aplicaciones.

Para el diseño y la programación del microcontrolador se necesitan únicamente dos herramientas básicas las cuales son el HEW y el FDT cuyos usos se tratan más a fondo en el transcurso del capítulo; pero como una breve descripción se puede adelantar que uno sirve para el diseño de programas con un ambiente IDE (HEW) y el otro para la programación (o *Flashing*) del dispositivo (FDT).

3.2 HEW

El HEW (High Performance Embedded Workshop o Hitachi Embedded Workshop) fue desarrollado con anterioridad por los ingenieros de Hitachi antes de convertirse en Renesas. El HEW es un programa IDE que permite el desarrollo de aplicaciones de texto con comandos de lenguaje ensamblador tanto propios del CPU H8/300H como de las diferentes familias de microcontroladores desarrolladas por esta firma.

> Nota: Haga clic aquí para descargar el programa o en:
> www2.eu.renesas.com/products/mpumcu/tool/hew/support.html

3.2.1 Instalación del programa.

La instalación del programa es sencilla y posee una interfaz amigable que permite la fácil comprensión del proceso. Basta con tener una computadora con un sistema operativo

Window98 o superior y 100MB de espacio disponible para que el programa se ejecute correctamente.
Para el inicio de la instalación se debe de hacer doble clic al archivo ejecutable con el nombre *Setup* que se contiene en la carpeta hew3_1, la cual indica es la versión 3.1 del programa. Al realizar esta acción se abrirá una ventana como la siguiente.

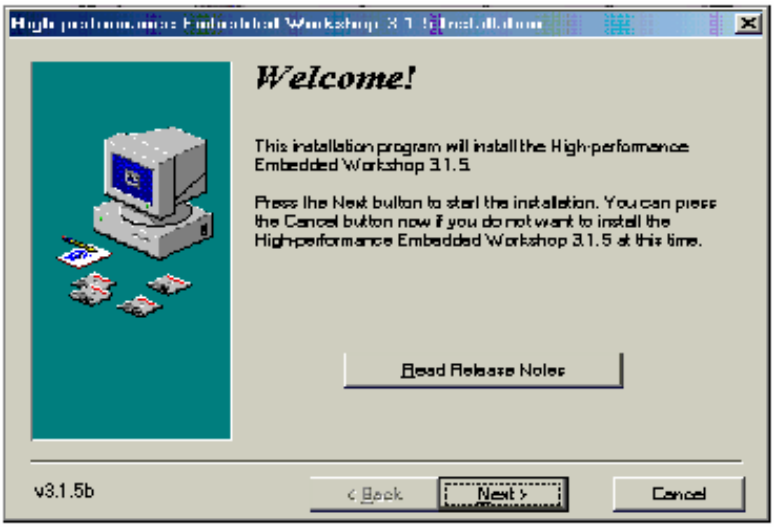


Figura 3.1: Pantalla de inicio de instalación de HEW

Luego presionamos el botón *Next* lo cual indica la continuación del proceso de instalación.

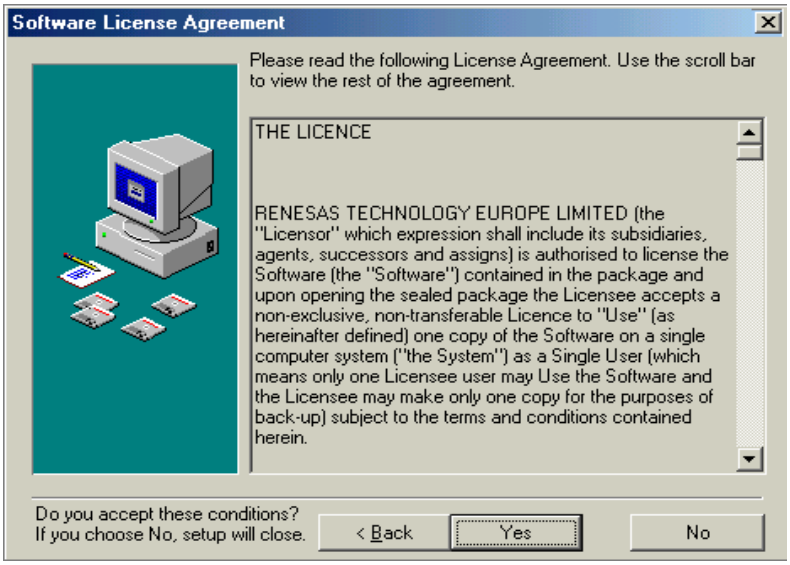


Figura 3.2: Acuerdos de licencia durante el proceso de instalación

Luego de leer los acuerdos de licencia procedemos a aceptar presionando el botón *Yes* para dar el siguiente paso.

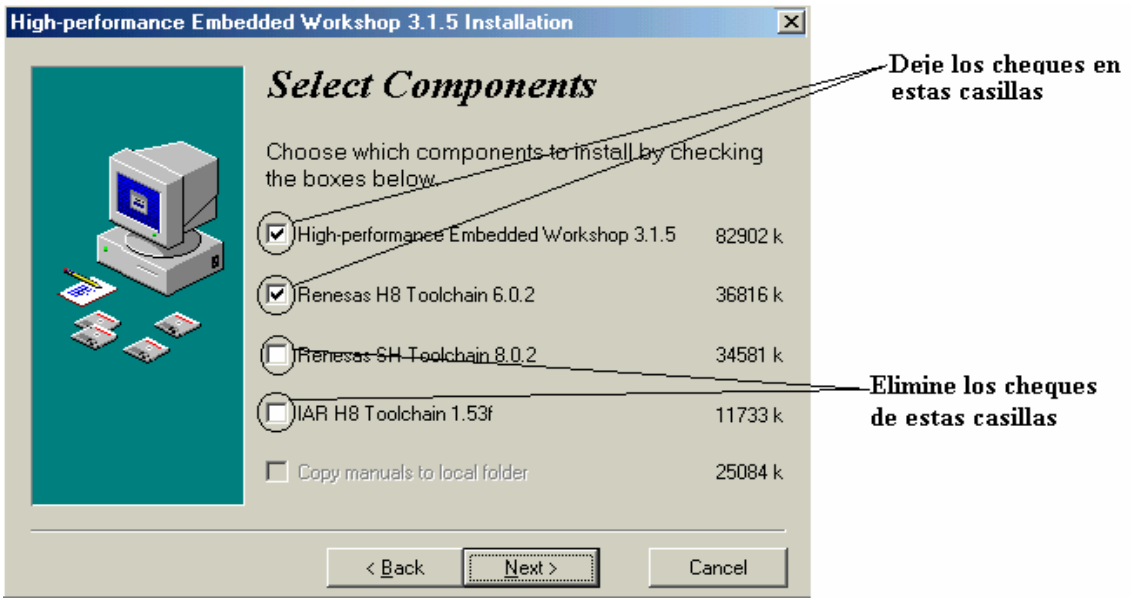


Figura 3.3 Selección de componentes.

En la selección de componentes, se debe de eliminar aquellos de los cuales no se tendrá provecho, en este caso eliminamos los cheques correspondientes a las familias de microcontroladores que no se utilizaran. Se deja únicamente las que soportan el microcontrolador a estudiar.

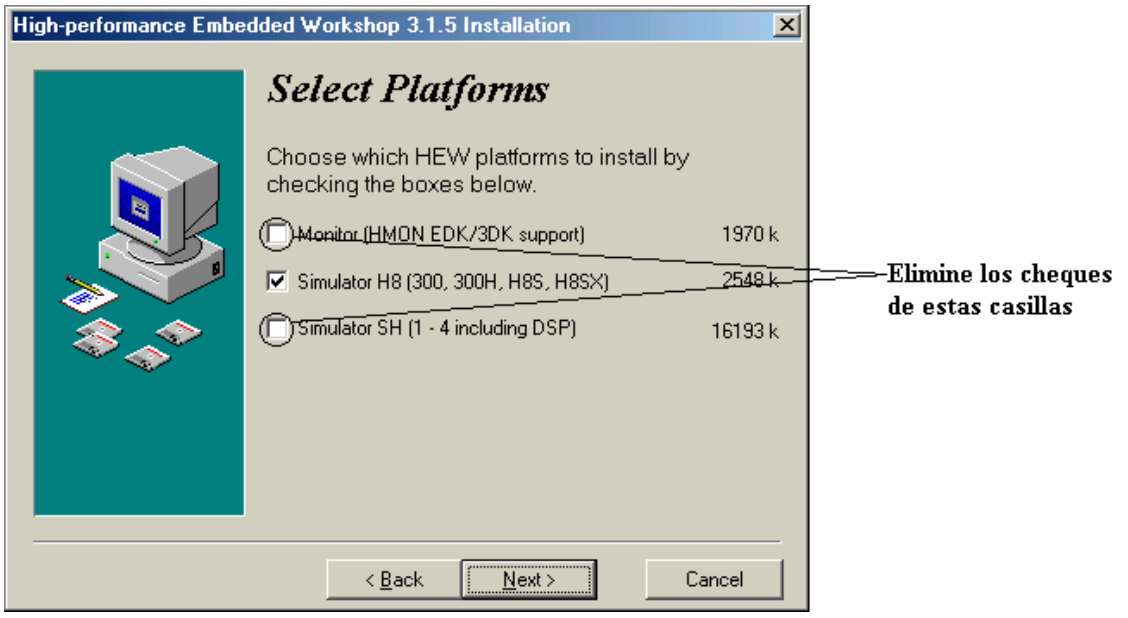


Figura 3. 4: Selección de plataformas.

En la selección de la plataforma de simulación solo se activará aquella que servirá para el CPU a trabajar.

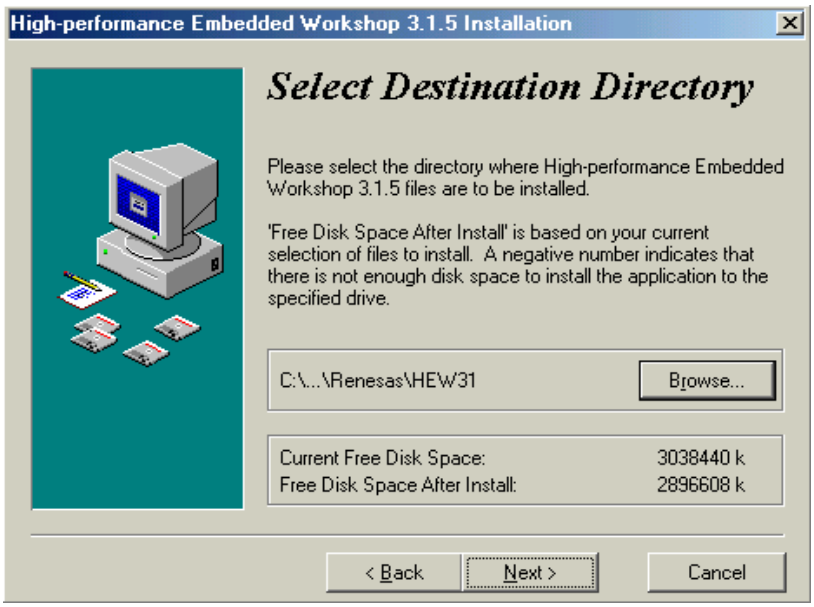


Figura 3. 5Selección del directorio.

La selección del directorio define el destino que contendrá los archivos necesarios para que el programa HEW sea ejecutado en la computadora. Si se quiere una determinada dirección en la cual se desea descargar los archivos del programa, entonces debe de presionarse el botón *Browse* el cual abrirá la siguiente ventana.

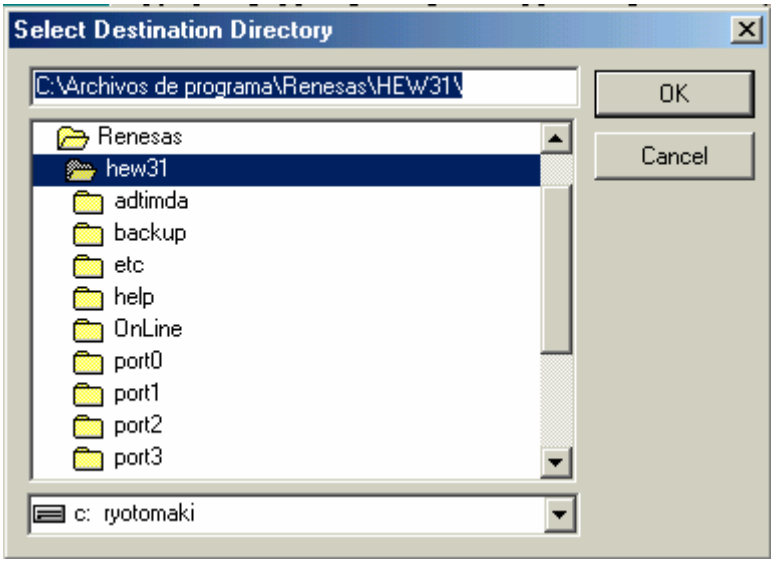


Figura 3. 6: Selección de un directorio definido por el usuario

En esta ventana se puede especificar la ruta destino en donde se deseen descargar los archivos del programa.

Luego de especificar la ruta, ya sea definida por el programa o por el usuario, se procede a seguir con el proceso de instalación pulsando el botón *Next* de la ventana de instalación.

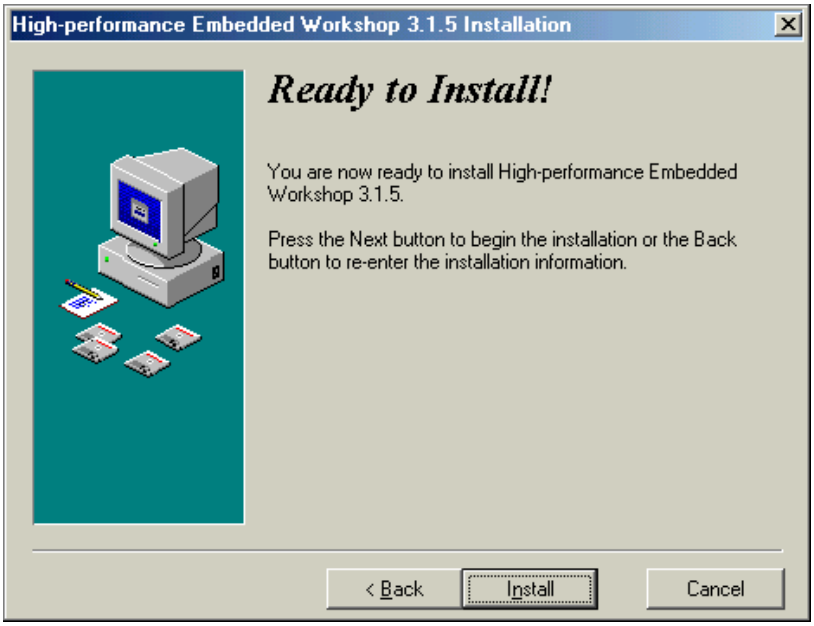


Figura 3. 7: Comienzo de la instalación de archivos en el directorio especificado

Una vez definidos todos los parámetros se inicia el proceso de instalación de archivos, luego de presionar el botón *Install*. Durante dicho proceso se desplegará una pequeña ventana, la cual contiene el porcentaje de la instalación tal como se muestra en la figura siguiente:

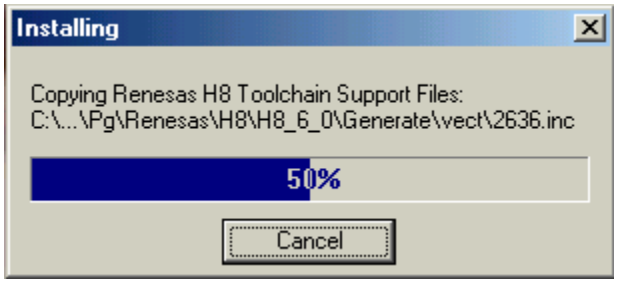


Figura 3. 8: Visualización del proceso de instalación de archivos

Al terminar el proceso de instalación un número denominado código de sitio (Site Code) es dado al usuario para el posterior registro del programa. Al llegar a esta etapa se tiene lista la instalación del programa, con las limitantes antes expuestas, para crear aplicaciones IDE.

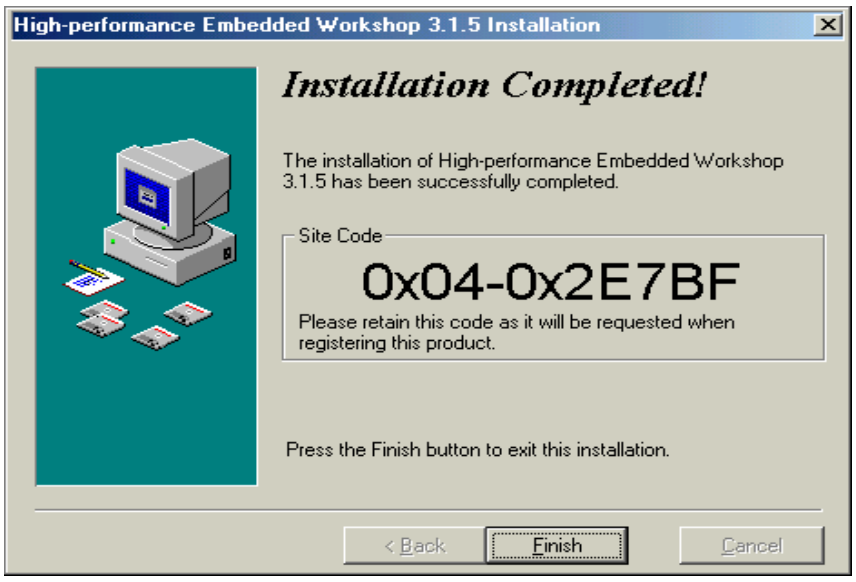


Figura 3. 9: Instalación completada

3.2.2 Uso del HEW.

El HEW permite al usuario el desarrollo de programas en formato de texto para su posterior compilación, es decir que el compilador HEW se encarga de pasar el código ensamblador a código objeto que es el que ejecuta el microcontrolador por el CPU interno.

El HEW consta de cinco componentes los cuales son:

1. **Compilador**: Es utilizado para procesar los programas realizados en C o C++ y transformarlos en lenguaje ensamblador.

2. **Ensamblador**: Toma el código en lenguaje ensamblador y crea un archivo en código objeto.

3. **Enlazador**: Realiza la unión de varios códigos objetos, transformándolos en un solo código que contiene todas las funciones que se realizan en cada archivo.

4. **Conversor de objetos**: convierte el código objeto en un archivo S-record, este es el que contiene la información que se debe transmitir al microcontrolador.

5. **Librerías**: Son archivos que realizan funciones comunes y repetitivas en uno o varios programas, son rutinas que pueden ser llamadas para futuras aplicaciones, estas son realizadas por el usuario.

   Nota: En el estudio del presente capitulo únicamente tomaremos en cuenta las funciones que son la parte del ensamblador y parte del conversor de objetos.

Para comenzar a utilizar el programa debemos situarnos en la pantalla principal del sistema operativo que corre la computadora dar un clic al botón de inicio y seguir la secuencia siguiente.

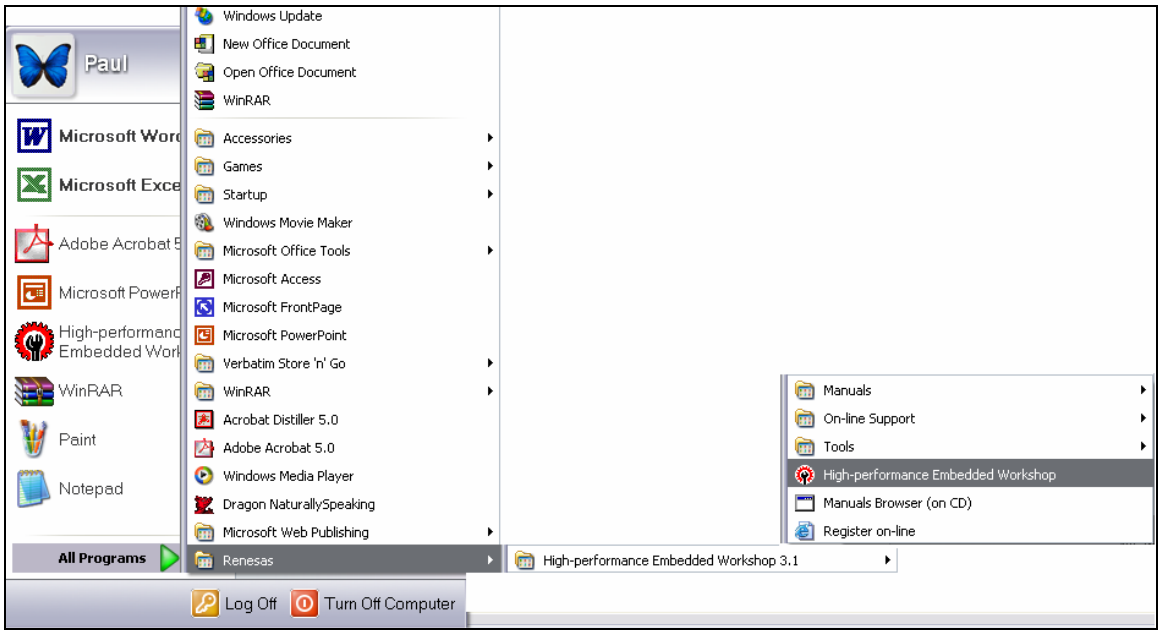


Figura 3. 10: Secuencia de inicio del HEW

Luego de seguir la secuencia e iniciar el programa, aparecerá la ventana de bienvenida como lo describe la siguiente figura.

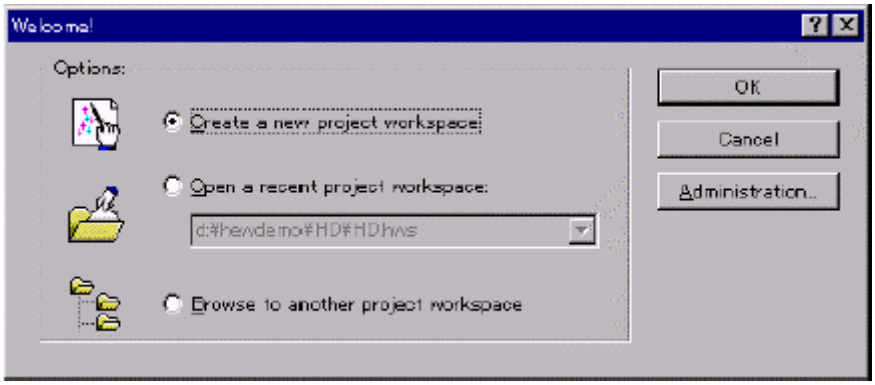


Figura 3. 11: Ventana de bienvenida

En esta ventana se tiene la opción de cargar un espacio de trabajado previamente realizado en la interfaz, o bien crear uno nuevo.

Para crear un nuevo espacio de trabajo se deben realizar los siguientes pasos: Dar clic en el botón OK de la figura 3.11; esta acción desplegará una ventana como la siguiente.

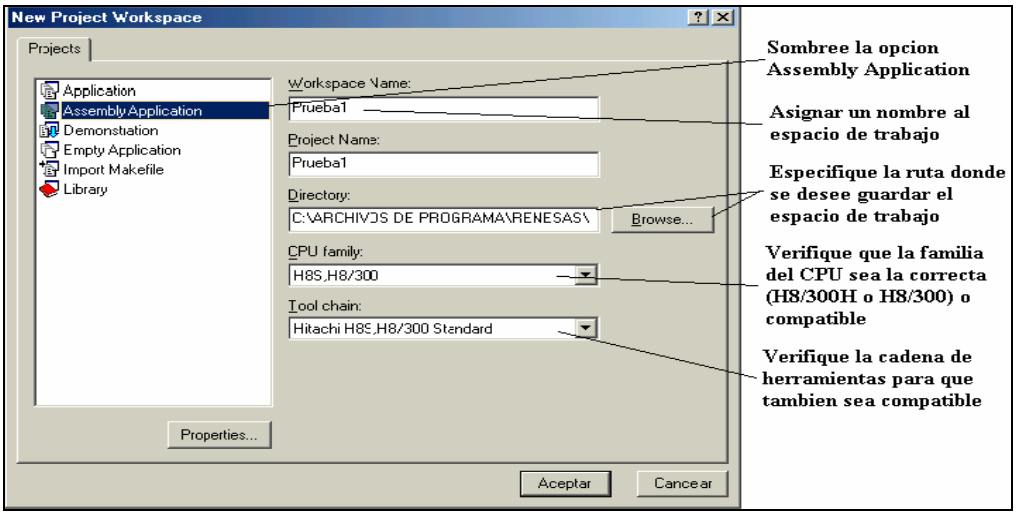


Figura 3. 12: Parámetros del espacio de trabajo

En esta ventana se debe especificar los parámetros principales del espacio de trabajo que se va a utilizar. Se debe especificar el tipo de proyecto a crear. Para nuestro caso, se debe de seleccionar la aplicación de ensamblador (Assembly Application), lo cual indica que se incluirán los componentes necesarios para la creación de archivos en código ensamblador.

Se debe también colocar un nombre al proyecto, asignarle una dirección o ruta en la cual se almacenarán los archivos creados por este espacio de trabajo y finalmente, se debe verificar que tanto la familia del CPU seleccionada como la cadena de herramientas sean compatibles con el microcontrolador a utilizar.
Terminado este proceso presionamos el botón *Aceptar* en la ventana de la figura 3.12, esta acción desplegará una nueva ventana.

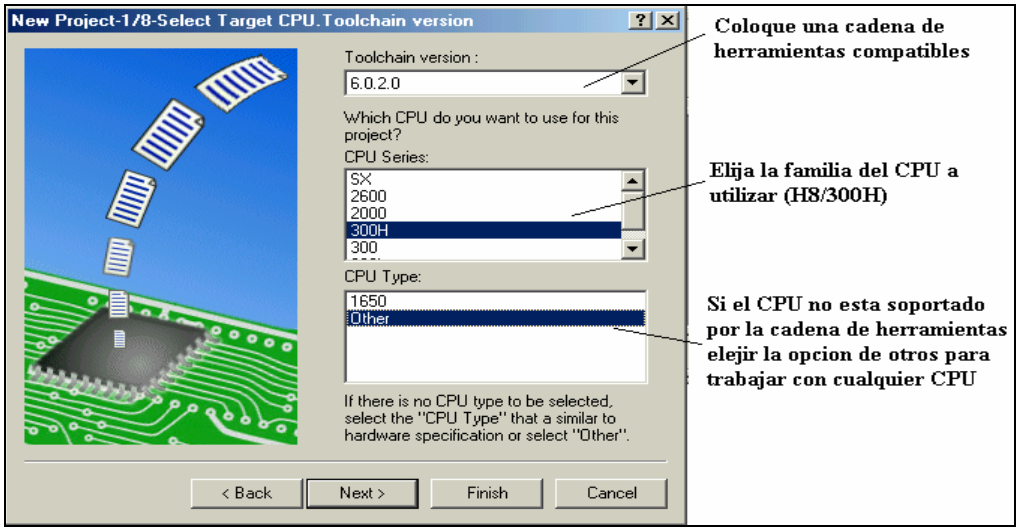


Figura 3. 13: Selección de los parámetros del dispositivo

En esta se especifican los parámetros del dispositivo que se va a utilizar. Las versiones demostrativas del HEW no soportan todos los dispositivos disponibles en el mercado y es necesario comprar la cadena de herramientas que soporte los que vamos a utilizar. Ahora, al no estar soportado algún CPU basta con indicarle al programa que se utilizara el tipo del CPU sin especificar el número.
Teniendo los parámetros del dispositivo procedemos al siguiente paso dando clic al botón *Next*.

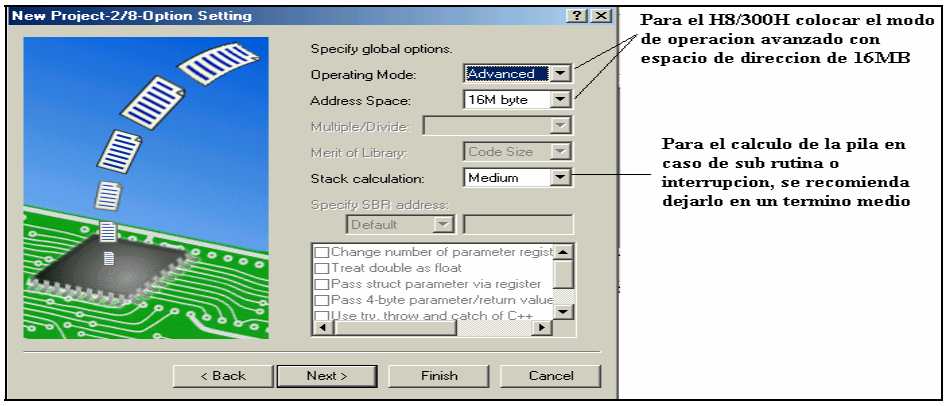


Figura 3. 14: Opciones del espacio de trabajo

En esta ventana se establecen las opciones que tendrá el espacio de trabajo. Ahora, como el CPU H8/300H solo puede ser utilizado en modo expandido entonces la opción de modo expandido es habilitada.
En el caso que se trabaje con otro tipo de CPU que este soportado por la cadena de herramientas, estas opciones serán de gran ayuda al momento de crear la aplicación ya que permitirán al programador crear un código común el cual podrá ser usado en diferentes microcontroladores. Por ejemplo, la delimitación de la pila de datos al momento que se realice una interrupción o una subrutina.

Una vez hecho esto procedemos al siguiente paso.

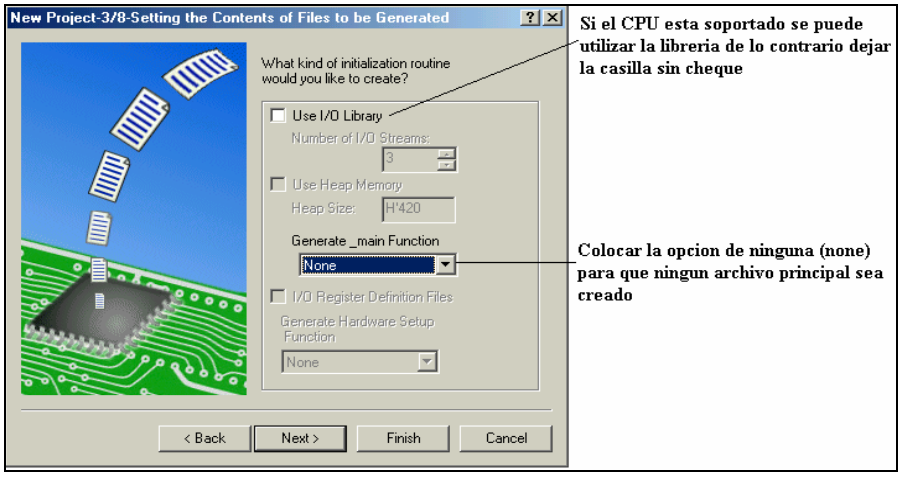


Figura 3. 15: Contenidos de los archivos a crear

Al tenerse una cadena de herramientas que soporte al microcontrolador se puede utilizar la librería de entrada y salida, de esta manera los registros a ser usados no deban ser declarados en el programa.

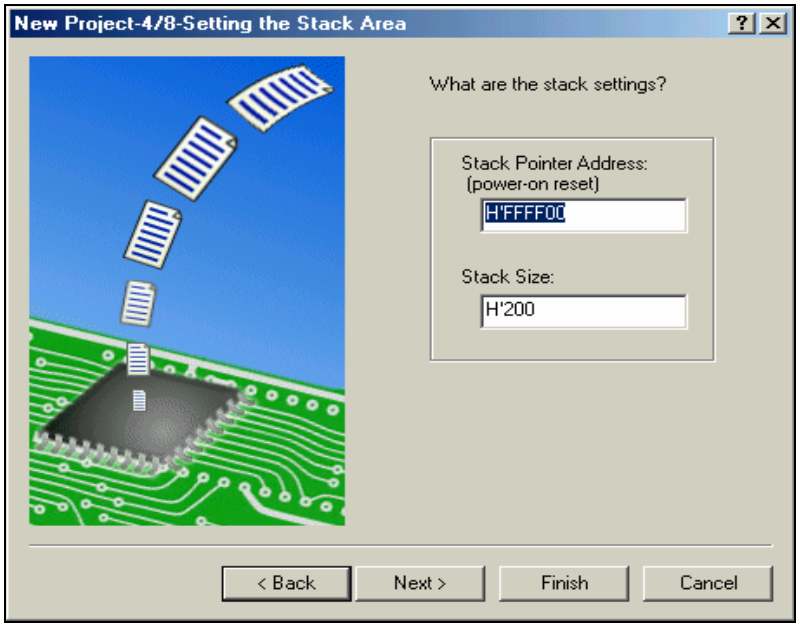


Figura 3. 16: Fijando los valores de la pila de datos

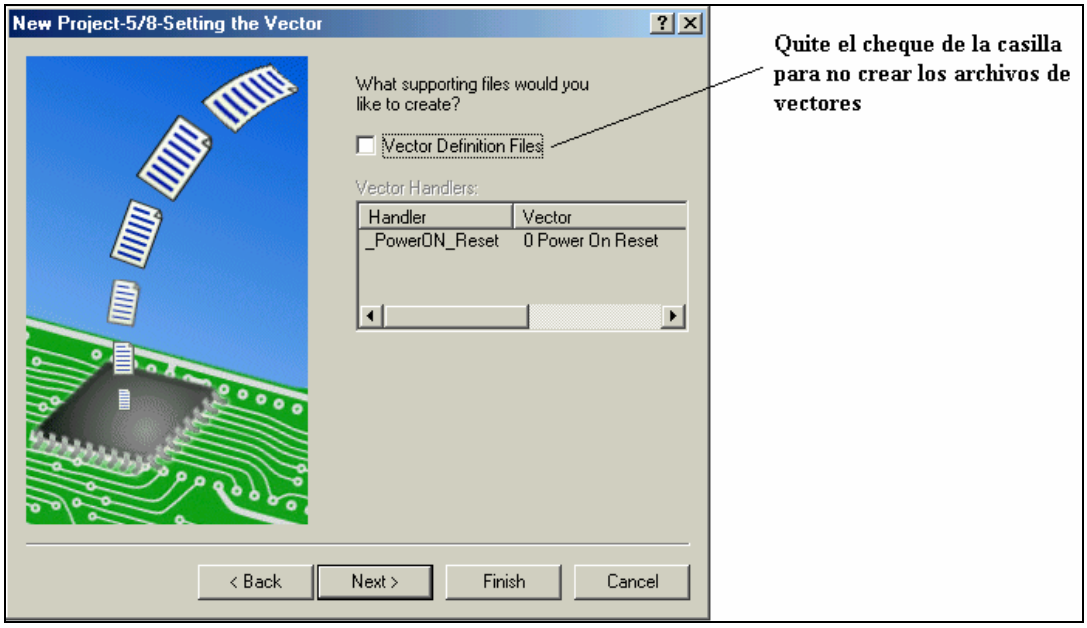


Figura 3. 17: Definición de vectores

Los archivos de vectores son creados para establecer la dirección del PC (contador de programa). Para nuestro caso, no debe de activarse la *definición de vectores* ya que el H8/3069F no es soportado por la versión demostrativa, de modo que para declarar los vectores de interrupción y/o de Reset estos deben colocarse en el código fuente.

Con este último parámetro se procede a la finalización del nuevo *Workspace* presionando el botón de *Finish* en la ventana de definición de vectores.

Por otro lado, en caso que ya se tuviera un programa digitado solo es necesario buscar la opción Open del menú File y buscar el archivo con extensión ".scr" en la dirección correcta para poder abrirlo. Si al momento de abrir un código no se tiene un Workspace creado es posible crear un por defecto o crear uno personalizado como se explicó anteriormente

3.2.3 Escribiendo un programa en lenguaje ensamblador.

En esta parte del capítulo se dan las generalidades para desarrollar programas en lenguaje ensamblador, sólo unas instrucciones básicas son introducidas aquí para ayudar a comprender cómo se ejecuta un programa y como los contenidos de los registros y la memoria cambian en el progreso.

El lenguaje ensamblador es el lenguaje de programación más básico y se relaciona con el lenguaje de máquina. En este las instrucciones son ejecutadas una a una, haciéndolo el lenguaje ideal para comprender la operación del microcontrolador.
Como las instrucciones de máquina son una colección de ceros y unos, esto dificultaría desarrollar programas directamente en este lenguaje. Por esta razón, el lenguaje ensamblador es utilizado para expresar el lenguaje de máquina en un fácil y comprensible lenguaje alfabetizado.

3.2.3.1 Registros internos del CPU.

Antes de desarrollar un programa, es necesario conocer el tipo de registros y funciones que tiene el CPU.
Los registros internos son clasificados en registros de propósito general y registros de control. Los de propósito general son utilizados para calcular datos y direcciones de almacenamiento. Y los de control se utilizan para manipular el contador del programa, lo cual permite controlar el proceso, y controlar y observar las condiciones o la activación de banderas.

Registros de propósito general.

El CPU tiene ocho registros de propósito general, cada uno capaz de almacenar números de 32 bits. Estos registros pueden manipularse de diferentes formas:

- Cuando 32 bits son almacenados, hacemos referencia a estos de la siguiente manera: ER0, ER1, ER2, ER3, ER4, ER5, ER6, ER7

- Cuando 16 bits son almacenados, hacemos referencia a estos de la siguiente manera: E0, E1, E2, E3, E4, E5, E6, E7, R0, R1, R2, R3, R4, R5, R6, R7

- Cuando 8 bits son almacenados, hacemos referencia a estos de la siguiente manera: R0H, R0L, R1H, R1L, R2H, R2L, R3H, R3L, R4H, R4L, R5H, R5L, R6H, R6L, R7H, R7L

Esto es ilustrado mejor en la siguiente figura:

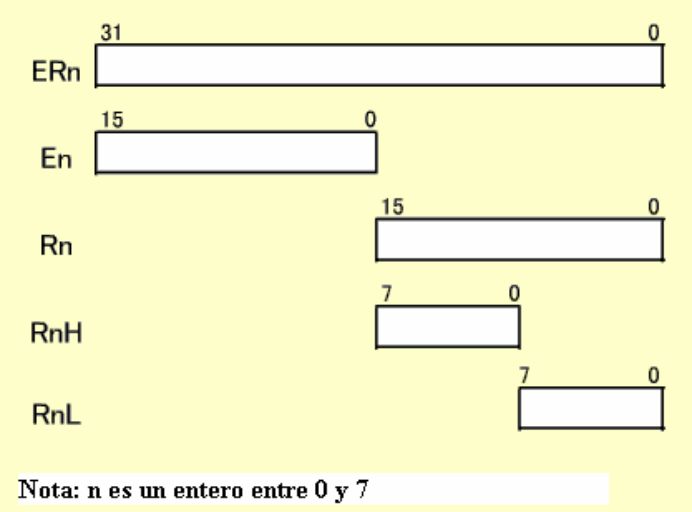


Figura 3. 18: Registros de propósito general.

Ejemplo de cálculo con registros de propósito general.

En este ejemplo, una instrucción de suma es utilizada para mostrar como un registro de propósito general es usado:

ADD.B R0L,R1H es una instrucción para suma de ocho bits. ADD representa "ADDition" (Suma) y B representa "Byte" (8 bits). El contenido del R1H y R0L son sumados y el resultado es almacenado en R1H.

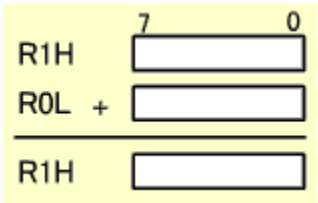


Figura 3. 19: Suma de ocho bits.

Esto no influirá el E1 o R1L. Sólo ocho bits son el resultado obtenido. Por ejemplo, se puede especificar un mismo registros como "ADD.B R1L,R1L". En este caso, R1L dobla su valor.
ADD.W R0,E1 es una instrucción de suma de 16 bits ADD representa "ADDition" y W representa "Word" (16 bits). Los contenidos del E1 y R0 son sumados y el resultado es almacenado en E1.

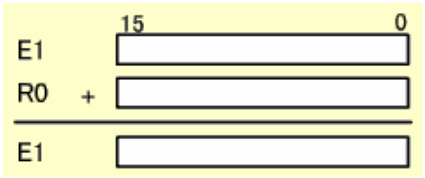


Figura 3. 20: Suma de dieciséis bits.

Note que esto no afectará el valor de R1, por que el resultado obtenido es almacenado en E1 el cual es totalmente independiente de R1 aun cuando pertenezca siempre al registro ER1. además, el resultado obtenido es solo de 16 bits.

ADD.L ER0,ER1 es una instrucción de 32 bits en la cual ADD representa "ADDition" y L representa "Long word" (32 bits). El contenido de los registros generales ER1 y ER0 son sumados y el resultado es almacenado en ER1.

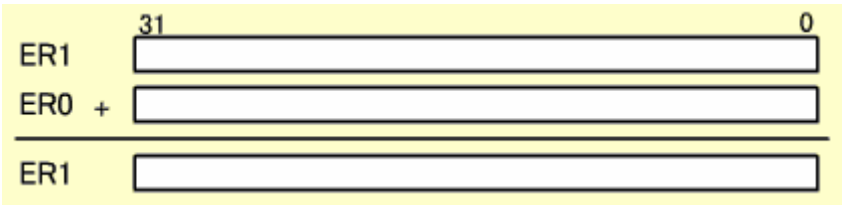


Figura 3. 21: Suma de 32 bits.

PC (Contador del programa)

Guarda la dirección de la instrucción que será ejecutada en el CPU. El dato es automáticamente refrescado cada vez que el CPU lee la instrucción.
En el caso de la arquitectura H8/300H, una instrucción siempre es leída desde una dirección de número par. Esto significa que una dirección de numeración par esta siempre almacenada en el PC.

CCR (Registro de condición de código)

Este es un registro utilizado para el control de las banderas o pruebas de condición. Es un registro de 8 bits en donde cada bit tiene diferente significado (Para detalle vea el capitulo uno).

3.2.4 Desarrollo de un programa en lenguaje ensamblador.

En esta sección se muestra el desarrollo de un programa fuente utilizando los ejemplos anteriores.
Tenemos dos números de 8 bits los cuales están almacenados en las direcciones de memoria H'2000 y H'2001, el objetivo es crear un programa que sume ambos números y que el resultado lo guarde en la dirección H'2002.

Como una suma será ejecutada por byte es preciso entonces utilizar la instrucción *ADD.B R0L,R1L*. Note que otros registros cualesquiera pueden ser utilizados para el ejemplo, no únicamente R0L y R1L.

Para desarrollar la suma usando la instrucción antes mencionada, los datos de las direcciones deben ser introducidos a los registros. Esto se logra con la siguiente instrucción: *MOV.B @H'2000,R0L,* lo cual indica que el valor almacenado en la dirección H'2000 es transferido al registro R0L.
Además usamos *MOV.B @H'2001,R1L, l*o cual indica que el valor de la dirección *H'2001* es transferido al registro R1L.

Ahora, *H'2000* representa la dirección hexadecimal y para acceder a la memoria interna del microcontrolador desde una instrucción se debe utilizar el prefijo "@".

Para ejecutar la suma completa tenemos lo siguiente:

```
MOV.B    @H'2000,R0L
MOV.B    @H'2001,R1L
ADD.B    R0L,R1L
```

A este punto la suma ha sido ejecutada y el resultado es almacenado en R1L, sin embargo aun no esta almacenado en la dirección de memoria *H'2002*, esto debe realizare con la siguiente instrucción:

```
MOV.B    R1L, @H'2002
```

Consecuentemente:

```
MOV.B    @H'2000,R0L
MOV.B    @H'2001,R1L
ADD.B    R0L,R1L
MOV.B    R1L,@H'2002
```

La suma ha sido completada con 4 instrucciones, pero el programa no esta terminado. Después de leer cada instrucción, el CPU automáticamente guarda la dirección de la siguiente instrucción en el PC, ahora después de la última instrucción el CPU no comprenderá cual es la siguiente a ser ejecutada pues no habrán instrucciones existentes. Para prevenir un posible problema es recomendable utilizar las instrucciones de salto de esta manera:

```
MOV.B    @H'2000,R0L
MOV.B    @H'2001,R1L
ADD.B    R0L,R1L
MOV.B    R1L,@H'2002
ABC:     BRA   ABC
```

Utilizando la instrucción BRA (BRanch Always) realizamos un bucle infinito con el cual prevenimos que el CPU caiga en un descontrol después de la operación que ha realizado.

Sin embargo, el programa aun esta incompleto, pues no indica en cual dirección de memoria será localizado. Esto debe realizarse con instrucciones de control en lenguaje ensamblador, las cuales no son ejecutadas.

.SECTION PROG,CODE,LOCATE=H'1000

Cada instrucción de control esta precedida por un "." de esta manera se puede distinguir fácilmente las instrucciones de control y aquellas que serán ejecutadas por el CPU.

".SECTION" indica la sección de la operación de control, *PROG* representa el nombre de la sección, *CODE* hace referencia a que son códigos de instrucción y *LOCATE=H'1000* especifica que las instrucciones ejecutables deben estar localizadas comenzando desde la dirección H'1000.

La instrucción *".CPU"* especifica el tipo de CPU, en el caso aplicado debe corresponder al tipo "300HA" (H8/300H) en el cual "300H" indica el tipo y "A" representa *Avanzado*.

La instrucción *".END"* debe ser colocada siempre al terminar la última línea de código indicando el fin del programa. Finalmente, el programa completo se muestra en la siguiente figura:

```
        .CPU      300HA
        .SECTION  PROG,CODE,LOCATE=H'1000
        MOV.B     @H'2000,R0L
        MOV.B     @H'2001,R1L
        ADD.B     R0L,R1L
        MOV.B     R1L,@H'2002
ABC:    BRA       ABC
        .END
```

Figura 3. 22: Programa simple.

3.2.5 Reglas de programación:

En esta sección del capítulo se detallan algunas de las reglas de programación a tomar en cuenta para el desarrollo de programas en lenguaje ensamblador. Si alguna de ellas no se cumple un error aparecerá en el desarrollo del programa.

3.2.5.1 Configuración de las instrucciones.

Como dijimos anteriormente, una instrucción del tipo ADD.B R0L,R1L esta constituida por "*ADD"* lo cual representa la "ADDition" y *".B"* representa el tamaño en bits que abarca la instrucción, aquí 8 bits (*".W"* representa 16 bits y *".L"* representa 32 bits).

Los conjuntos de letras y números así como: R0L y R1L son colectivamente llamados "operandos", y podemos encontrar que el operando R0L (a la izquierda) es específicamente llamado operando fuente y el operando R1L (a la derecha) es específicamente llamado operando destino.
Siempre en una operación con dos operandos, el resultado o respuesta del cálculo es almacenado en el operando destino.

Es importante hacer notar que algunas instrucciones tienen solo un operando (operando destino) o no tienen ninguno, como por ejemplo INC.B, R0L y RTS.

3.2.5.2 Escritura de líneas.

Reglas:

1. Uno o más espacios (o tabulaciones) deben de ser colocados al inicio de cada línea.
2. Instrucciones y operandos deben estar separados por uno o más espacios (o tabulaciones).
3. Las instrucciones pueden ser escritas tanto en minúsculas como en mayúsculas.
4. Una línea debe terminar en un retorno o *Enter*.

Ejemplos:

	MOV.B	R0L,R1L	Buen ejemplo
	Mov.b	r0l,r1l	Buen ejemplo Es posible mezclar mayusculas y minusculas...
	Mov.b	@h'1000,R1h	Buen ejemplo
MOV.B	R1L,R0L		Mal ejemplo (no hay espacio o Tabulación al inicio).
	MOV.BR0L,R1L		Mal ejemplo (la instrucción y el operando no esta separada por ningún espacio o tabulación).

3.2.5.3 Escritura de viñetas.

Reglas:

1. Escribir un símbolo o una palabra de referencia.
2. Anteponer el símbolo " : " (dos puntos).
3. El resto de la línea se completa con una línea de código, con un *enter* o retorno.
4. Los caracteres habilitados para las viñetas son: "A" a la "Z", "a" a la "z", "0" al "9", "_" y "$". Note que para este caso las mayúsculas y minúsculas son caracteres diferentes.
5. Los primeros caracteres no deben ser valores numéricos.
6. Palabras reservadas como instrucciones o nombres de los registros internos del CPU no pueden ser usados.

Ejemplos:

LOOP:	MOV.B	R0L,R1L	Buen ejemplo
R1:	MOV.B	R0L,R1L	Mal ejemplo (el nombre de un registro interno es utilizado como viñeta).

Ejemplo de viñetas:

Loop:	Minúsculas y mayúsculas pueden ser usadas.
End_of_Loop:	"_" es aceptado como carácter.
DATA1:	Los valores numéricos pueden ser usados, pero no al inicio de la viñeta.

Ejemplo de viñetas no disponibles:

1second	Comienza con un valor numérico.
Second/100	"/" es usado.
Total.Data	"." es usado.
CCR	El mismo nombre de un registro es usado.

3.2.5.4 Inserción de comentarios.

Los comentarios son insertados para hacer los programas más comprensibles y no tiene influencia sobre la operación del programa. Estos pueden ser insertados de dos maneras:
La primera es colocando un ";" al inicio de una línea, esto causa que la línea entera sea tratada como comentario. Todos los caracteres, valores numéricos y símbolos especiales pueden ser usados.

Ejemplo:

```
;*****************************
;*  H8/300H Programa ejemplo       *
;*  22/06/06                                *
;*****************************
```

La segunda manera es insertando el comentario después de la instrucción siempre con ";", de esta manera se puede colocar un comentario a cada instrucción y hacer así el programa mucho mas comprensible. A continuación presentamos el mismo ejemplo de la Figura 3.22 pero comentado:

```
;*************************************
;* H8/300H  Programa ejemplo         *
;* 22/06/06                          *
;*************************************
        .CPU     300HA
        .SECTION PROG,CODE,LOCATE=H'1000
        MOV.B    @H'2000,R0L      ; De @H'2000 a R0L
        MOV.B    @H'2001,R1L      ; De @H'2001 a R1L
        ADD.B    R0L,R1L          ; R1L = R0L + R1L
        MOV.B    R1L,@H'2002      ;De R1L a @H'2002
ABC:    BRA      ABC              ;Loop infinito
        .END
```

Figura 3. 23: Programa comentado.

3.2.5.5 Uso de la instrucción de control .EQU

Si una dirección de memoria es escrita en un programa con notación hexadecimal, es difícil determinar que tipo de dato es incluido. Por eso una dirección del mapa de memoria puede ser expresada como un símbolo, en vez de hacerlo de manera hexadecimal. La instrucción de control *.EQU* es la manera más simple de expresar una dirección en símbolo.

```
DATA1:    .EQU    H'2000
DATA2:    .EQU    H'2001
ANSWER:   .EQU    H'2002
```

En este ejemplo se observa el uso de la instrucción de control *.EQU* lo cual indica que DATA1 hace referencia a H'2000 de modo que podemos escribir la instrucción *MOV.B @H'2000,R0L* así: *MOV.B@DATA1,R0L.*

Este método es utilizado cuando un valor numérico o dirección es fijo y no será cambiado.

En la siguiente figura se muestra el empleo de este método.

```
;*************************************
;* H8/300H  Sample program           *
;* 22/06/06                          *
;*************************************
DATA1:    .EQU      H'2000
DATA2:    .EQU      H'2001
ANSWER:   EQU       H'2002
          .CPU      300HA
          .SECTION  PROG,CODE,LOCATE=H'1000
          MOV.B     @DATA1,R0L        ; R0L = DATA1
          MOV.B     @DATA2,R1L        ; R1L = DATA2
          ADD.B     R0L,R1L           ; R1L = R0L + R1L
          MOV.B     R1L,@ANSWER       ; ANSWER = R1L
ABC:      BRA       ABC               ; Loop infinito
          .END
```

Figura 3. 24: Programa usando ".EQU".

3.2.5.6 Uso de la instrucción de control .RES

La instrucción de control ".RES" es usada para reservar un área para escritura en la memoria RAM. Una dirección de RAM es generalmente especificada no por una instrucción .EQU si no que por una combinación de las instrucciones de control ".RES" y ".SECTION".

La instrucción ".RES" tiene los siguientes beneficios:

1. El comienzo de la dirección de almacenamiento puede ser cambiado o modificando por .SECTION.
2. Regiones de datos pueden ser insertados o borrados fácilmente.

La instrucción de control ".RES" es usada como se demuestra a continuación:

```
          .SECTION WORK,DATA,LOCATE=H'FFBF20
AB:       .RES.L 1      ; Reserva una área de 32 bits para el símbolo AB.
CD:       .RES.B 1      ; Reserva una área de 8 bits para el símbolo CD.
          .RES.B 1      ; Simplemente reserva una área de 8 bits y es usada
                        ; para corregir  el margen de 16 o 32 bits, para reservar en
                        ; la próxima dirección un número par.
EF:       .RES.W 2      ; Reserva 2 áreas de 16 bits para el símbolo EF.
XYZ:      .RES.B 6      ; Reserva 6 áreas de 8 bits para el símbolo XYZ.
```

Debemos mencionar que cada símbolo representa una dirección reservada en RAM.

Modificando el programa del ejemplo de la sección anterior, y utilizando la instrucción de control .RES el programa se escribe de esta manera:

```
;************************************
;* H8/300H Programa ejemplo         *
;* 2006.6.7                         *
;************************************
          .CPU      300HA
          .SECTION  PROG,CODE,LOCATE=H'1000
          MOV.B     @DATA1,ROL          ; ROL = DATA1
          MOV.B     @DATA2,R1L          ; R1L = DATA2
          ADD.B     ROL,R1L             ; R1L = ROL + R1L
          MOV.B     R1L,@ANSWER         ; ANSWER = R1L
ABC:      BRA       ABC                 ; Lazo infinito
          .SECTION  WORK,DATA,LOCATE=H'2000
DATA1:    .RES.B    1
DATA2:    .RES.B    1
ANSWER:   .RES.B    1
          .END
```

Figura 3. 25: Uso de .RES

3.2.5.7 Uso de la instrucción de control .DATA

El programa descrito en ejemplos anteriores en este capítulo, requiere que los valores escritos con los nombres DATA1 y DATA2 sean sumados, pero antes de la operación los valores deben ser introducidos a las direcciones respectivas de memoria. Como el área ocupada por esta operación es una región de memoria RAM (utilizando la instrucción de control .RES), al apagarse el CPU se pierde toda la información almacenada y no se tiene certeza de lo que tendrá al encender nuevamente el CPU.

Contrario a esto, la operación de control .DATA se utiliza para almacenar directamente los valores de constantes en regiones específicas de memoria ROM, lo cual deja sin alterar los valores aun si el CPU se desconecta. Podemos encontrar entonces cierta similitud entre el uso de las instrucciones .DATA y .RES para almacenar datos de carácter constante y datos dinámicos en el programa respectivamente.

Ejemplos:

```
.SECTION     WORK,DATA,LOCATE=H'1100
AB: .DATA.B 10                  ; Reserva 8 bits de área incluyendo el
                                ; valor “10” utilizando el símbolo AB.
CD: .DATA.B H'A6                ; Reserva 8 bits de área incluyendo el
                                ; valor “H’A6” utilizando el símbolo ; CD.
EF:  .DATA.W H'12AB             ; Reserva 16 bits de área incluyendo el
                                ; valor “H’12AB”  Utilizando el
```

```
                                    ; símbolo EF
XYZ:  .DATA.L      40000            ; Reserva 32 bits de área incluyendo el
                                    ; valor “40000”  Utilizando el símbolo
                                    ; XYZ
```

En el caso de que únicamente se quiera reservar espacio en memoria debe hacerse de la siguiente manera.

```
.SECTION WORK,DATA,LOCATE=H'1100
DATA1:      .DATA.L     10000
DATA2:      .DATA.W     1000
ANSWER:     .DATA.B     10
```

Como se puede ver no hay ninguna diferencia de sintaxis en los dos ejemplos, lo único que cambia es la cantidad de espacio reservado en la ROM.

Para el ejemplo, la sesión es localizada en la dirección H’1100; por lo tanto DATA1, DATA2 y ANSWER representan las direcciones H’1100, H’1104 y H’1106 respectivamente.

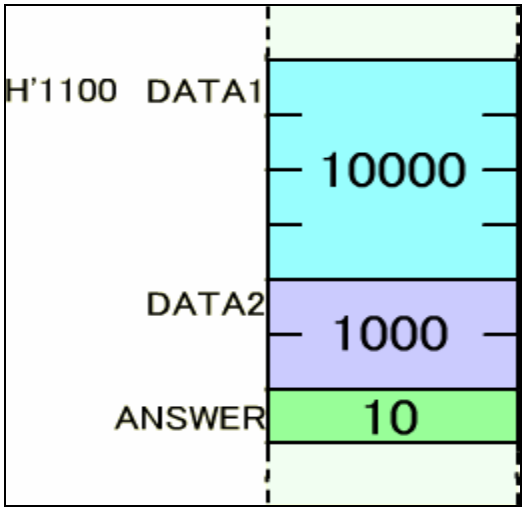


Figura 3. 26: Distribución de memoria utilizando .DATA

También, si se requiere se pueden dejar separados los valores constantes y valores dinámicos del programa tal y como se describe en el siguiente ejemplo:

```
.SECTION    ROM_DATA,DATA,LOCATE=H'1100
DATA1:      .DATA.B     10
DATA2:      .DATA.B     100
.SECTION    RAM_DATA,DATA,LOCATE=H'2000
ANSWER:     .RES.B      1
```

Esto hace que DATA1 este colocado en la dirección H’1100 en donde se escribe el valor “10” (“H’0A” en notación hexadecimal), DATA2 representa la dirección H’1101 adonde

se escribe el valor "100" ("H'64" en notación hexadecimal) y ANSwer representa la dirección H'2000.

Retomando el ejemplo de suma tratado en este capítulo y aplicando la instrucción de control .DATA tenemos lo siguiente:

```
;*************************************
;* H8/300H  Sample program          *
;* 2006.6.7                         *
;*************************************
          .CPU      300HA
                                     ; Program(ROM Area)
          .SECTION  PROG,CODE,LOCATE=H'1000
          MOV.B     @DATA1,ROL       ; ROL = DATA1
          MOV.B     @DATA2,R1L       ; R1L = DATA2
          ADD.B     ROL,R1L          ; R1L = ROL + R1L
          MOV.B     R1L,@ANSWER      ; ANSWER = R1L
ABC:      BRA       ABC              ; Lazo infinito
                                     ;Data(ROM Area)
          .SECTION  ROM_DATA,DATA,LOCATE=H'1100
DATA1:    .DATA.B   10               ; H'0A
DATA2:    .DATA.B   100              ; H'64
                                     ;Data(RAM Area)
          .SECTION  RAM_DATA,DATA,LOCATE=H'2000
ANSWER:   .RES.B    1
          .END
```

Figura 3. 27: Uso de la instrucción de control .DATA

3.2.6 Compilación de programas.

La compilación de programas en lenguaje ensamblador es un factor importante para el diseño, pues es la manera en la que se logra obtener el código de máquina, el cual es el utilizado por el microcontrolador para efectuar la función específica que el programador considere necesaria.

Vamos a retomar el ejemplo de la sección 2.2 en el cual se presentan los pasos a seguir para crear un espacio de trabajo en el HEW (llamado prueba1). Al abrir este espacio de trabajo desde el HEW se puede observar la creación del archivo *"stacksct.src"* el cual especifica los valores que pueden ser almacenados en la pila. El propósito del ejemplo es utilizar un solo archivo donde se hagan estas funciones por lo tanto este debe ser eliminado. Una manera fácil de hacerlo es dando un *click* derecho sobre el archivo desde la pantalla del HEW tal y como se muestra en la siguiente figura:

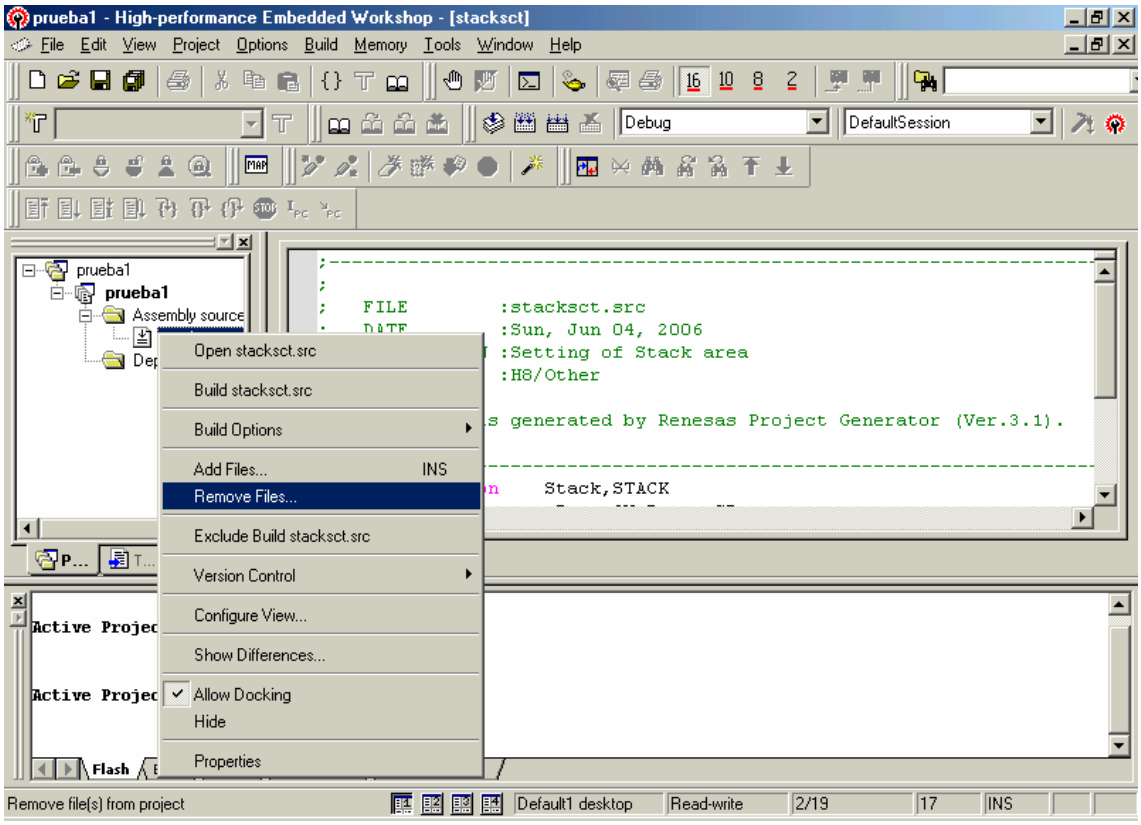


Figura 3. 28: Remover archivos desde HEW.

Otra manera de hacerlo es dar un clic a la viñeta Project y luego elegir la opción *Remove Files* (remover archivos). Al efectuar esto aparecerá una ventana como la siguiente:

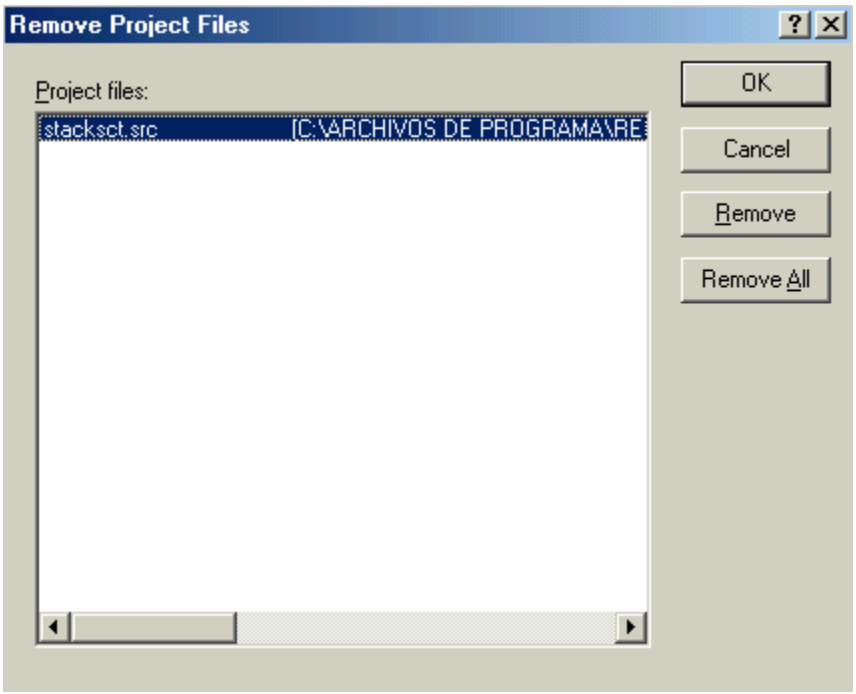


Figura 3. 29: Ventana de remoción de archivos.

Una vez el archivo sea removido procedemos a crear nuestro código fuente en la ventana de edición. La manera mas fácil de hacerlo es dando C*lick* en la viñeta *File* y luego en la opción *New*.

Se escribe el programa y se guarda con la extensión *".src"* y luego, de la misma manera que se removió el archivo *"stacksct.src"*, debe de agregarse el archivo recién creado. Para el caso, el ejemplo creado es el descrito en el capítulo, el cual quedará de la siguiente manera:

```
;*****************************
;* H8/300H Sample program    *
;* 2006. 6. 7                *
;*****************************
            .CPU        300HA       ;ESPECIFICA EL CPU
                                    ;PROGRAMA EN REGION ROM
            .SECTION    PROG,CODE,LOCATE=H'1000
            MOV.B       @DATA1,ROL  ;ROL=DATA1
            MOV.B       @DATA2,R1L  ;R1L=DATA2
            ADD.B       ROL,R1L     ;R1L=ROL+R1L
            MOV.B       R1L,@ANSWER ;ANSWER=R1L
ABC:        BRA         ABC         ;LAZO INFINITO
                                    ;REGION DE DATOS EN ROM
            .SECTION    ROM_DATA,DATA,LOCATE=H'1100
DATA1:      .DATA.B     10          ;H'0A
DATA2:      .DATA.B     100         ;H'64
                                    ;DATA (RAM AREA)
            .SECTION    RAM_DATA,DATA,LOCATE=H'2000
ANSWER:     .RES.B      1
            .END
```

prueba1

Figura 3. 30: Programa editado en HEW.

Una vez hecho esto hay que verificar errores sintácticos, para ello es necesario presionar el botón *Build File*, que se muestra en la siguiente figura.

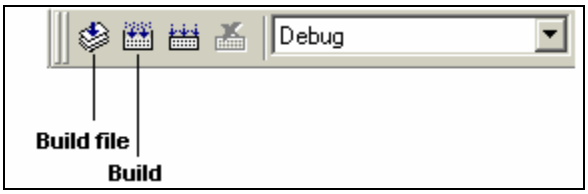


Figura 3. 31: Barra Estándar.

Al presionar este botón el compilador hace una revisión de errores o advertencias sintácticas y crea los archivos necesarios para crear posteriormente el archivo en lenguaje de máquina. El pop up menú que se encuentra a la derecha de la figura sirve para elegir el tipo de la memoria destino en donde se almacenará el programa. Por ejemplo, la selección de la opción *Debug*, que es la que aparece en la figura, sirve para crear aplicaciones que directamente se guardarán en la memoria no volátil del microcontrolador. La elección puede hacerse también como *Release*, la cual también

esta elegible en el pop up menú, y es utilizada para la creación de archivos máquina pero con el propósito de ser almacenados en una región de memoria volátil.

Para verificar si el compilador creo exitosamente el archivo en código máquina y ver si existe un error o advertencia sintáctica es necesario revisar el cuadro inferior de la pantalla editora. Si existe o no una advertencia o error el cuadro lo dirá y brindará la información suficiente para corregir el programa, en caso de ser necesario hacerlo.

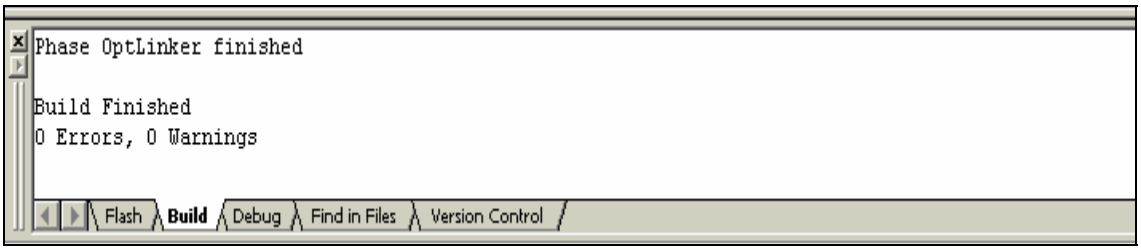


Figura 3. 32: Detección de errores y advertencias.

3.3 FDT (Flash Development Toolkit)

Es la herramienta de aplicación necesaria para poder transferir los archivos en código máquina desde la computadora al microcontrolador. Una vez completado el proceso de compilación y ya construido el archivo en lenguaje de máquina, con el HEW, se puede transferir el archivo que ejecutará el microcontrolador con este programa.

> Nota: Haga clic aquí para descargar el programa o vea el enlace siguiente: http://www.renesas.com/fmwk.jsp?cnt=/download_search_results.jsp&fp=/support/downloads/download_results&layerId=1050

3.3.1 Instalación del FDT.

Para instalar el programa debe darse doble clic al archivo ejecutable con el nombre FDT3_1 proporcionado en este manual y aparecerá una ventana como la siguiente:

Figura 3. 33: Instalación del FDT.

Para seguir con el proceso debe darse clic en el botón *Next*, al hacerlo se abrirá la ventana de acuerdos de licencia, al aceptar los acuerdos se abrirá una ventana como la que sigue:

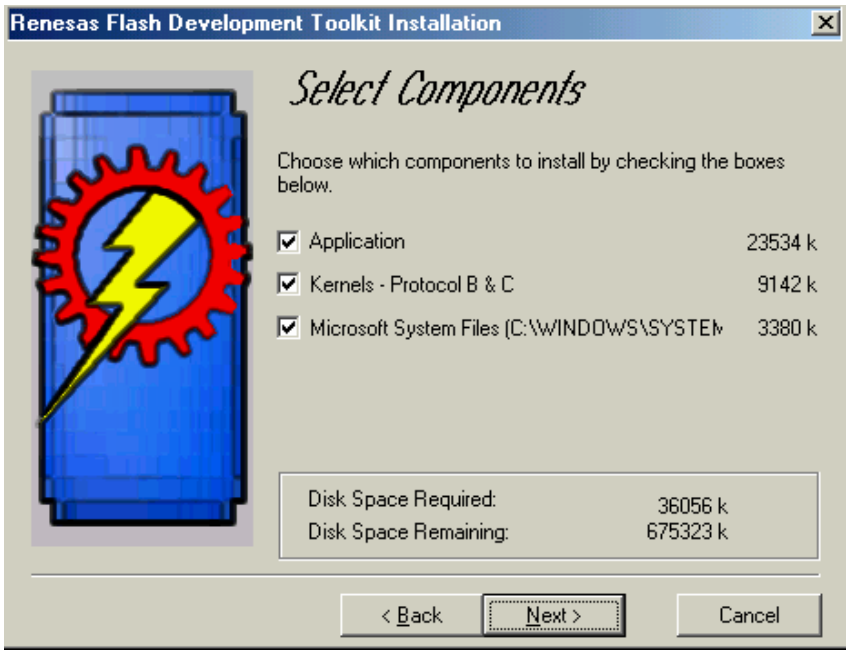


Figura 3. 34: Selección de componentes del FDT.

La selección de los componentes del FDT debe estar completa para la utilización.
Luego aparecerá la ventana de información adicional y a continuación, una ventana de selección como la siguiente:

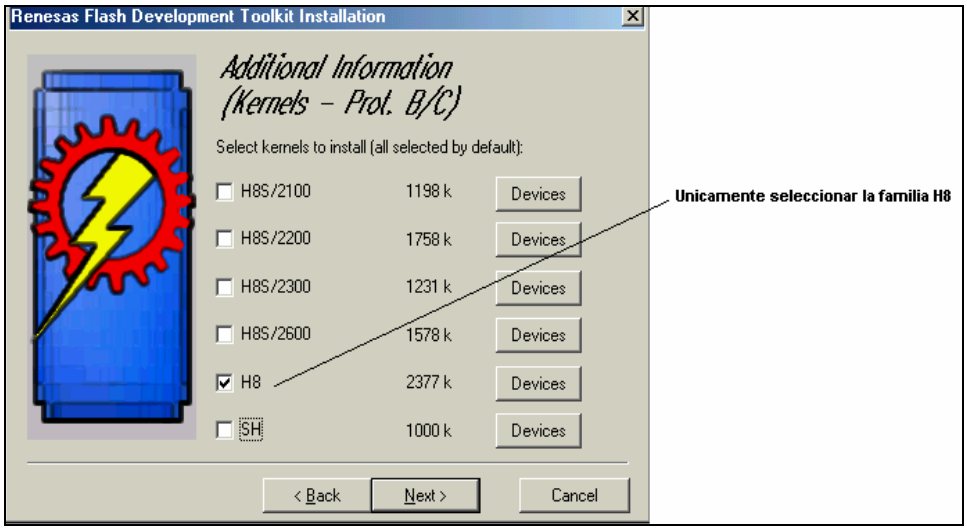


Figura 3. 35 Información adicional del FDT.

En esta parte de la instalación debe seleccionarse únicamente el tipo de microcontrolador (familia) a utilizar para ahorro de espacio en la unidad de almacenamiento.

Luego se lanza la ventana del directorio destino de la aplicación y una vez seleccionado el directorio, se procede instalar completamente el programa.

3.3.2 Uso del FDT.

Cuando se ejecuta el FDT este abre una ventana para la selección del espacio de trabajo que se desea, para el caso particular del microcontrolador a estudiar es suficiente hacer uno solo, pero para aplicaciones mas específicas pueden crearse más.

Para crear un espacio de trabajo desde el FDT es necesario hacer lo siguiente:

1. Dar clic en la viñeta file luego *New* y se lanzará una ventana como la siguiente.

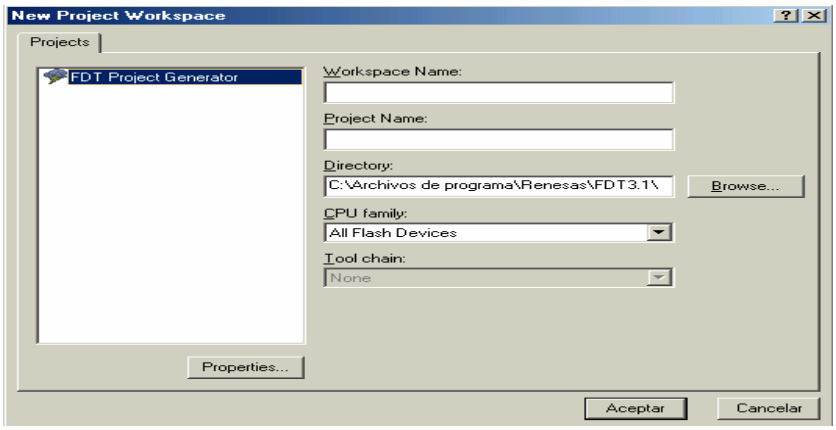


Figura 3. 36: Nuevo espacio de trabajo.

Para crearlo es necesario colocarle un nombre y luego dar clic al botón aceptar. Una vez dado el nombre debe de seleccionarse el microcontrolador a utilizar.

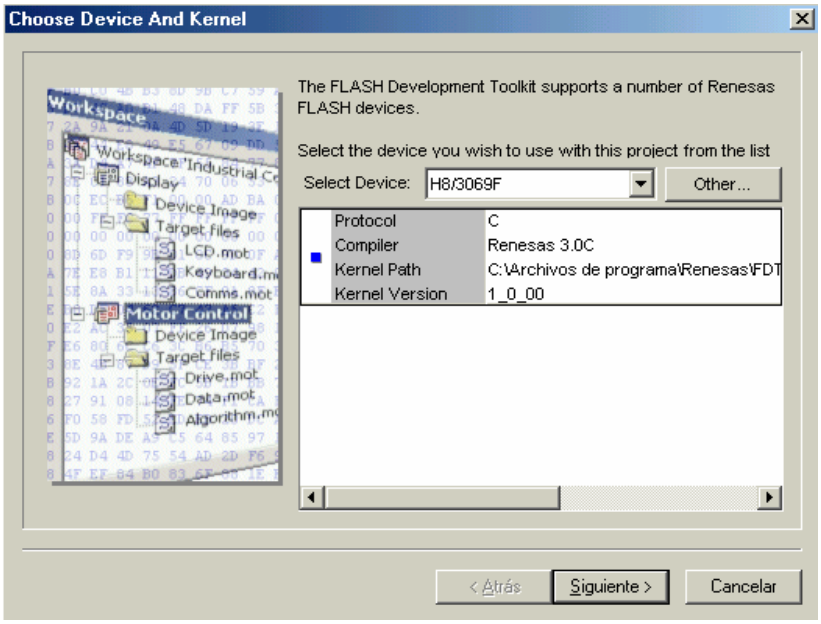


Figura 3. 37: Selección del dispositivo.

Luego de la selección del dispositivo debe seleccionarse el puerto de comunicación en la computadora, el reloj que tendrá el sistema mínimo del microcontrolador y el tipo de modo en el que arrancará el microcontrolador (BOOT), se recomienda dejar la velocidad de comunicación que ya esta establecida en la aplicación.

3.3.3 Descargando un programa al MCU

Para descargar un programa al microcontrolador es necesario colocar el medio físico de comunicación desde la computadora hasta el microcontrolador, tal y como se muestra en la siguiente figura:

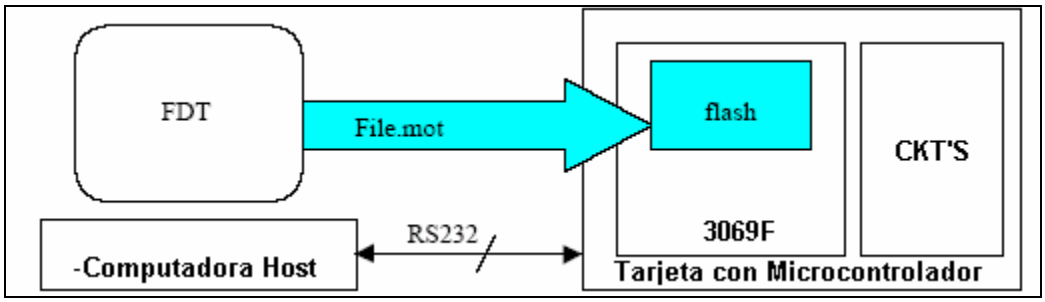


Figura 3. 38: Descargando un programa en la memoria flash.

El puerto físico de la computadora debe ser el COM1 o el configurado en el proceso de creación del espacio de trabajo descrito anteriormente y el tipo de conector es DB9 macho (la tarjeta tendrá y conector hembra de modo que el cable a utilizar sea del tipo comercial).

Retomando el ejemplo desarrollado en el HEW, se ha obtenido el archivo en lenguaje máquina con la extensión *".mot"* el cual deberá ser cargado en la ventana principal del FDT presionando el botón Cargar S-Record.

Figura 3. 39: Botón de carga de archivos en FDT.

Luego de dar clic aparecerá una ventana de exploración para buscar el archivo.

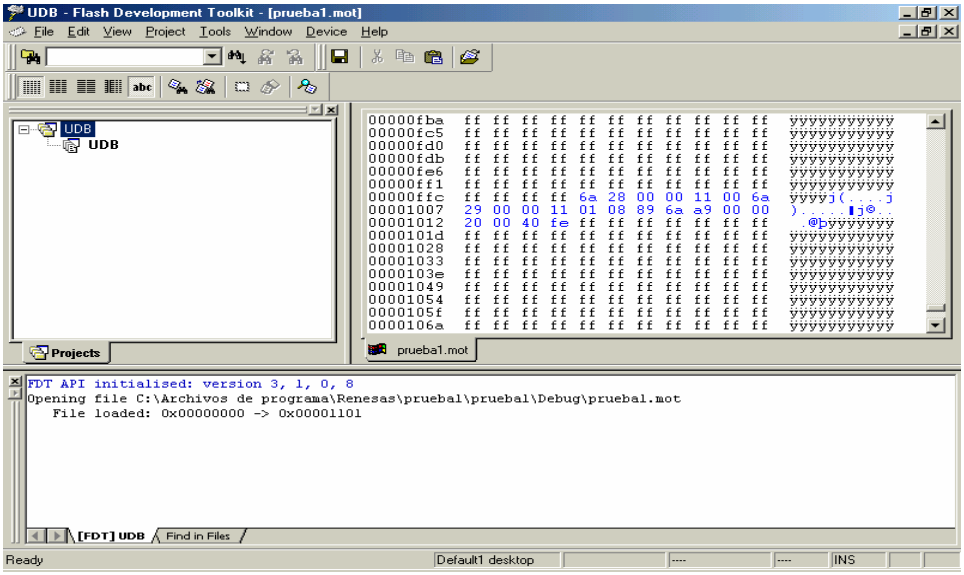


Figura 3. 40: Archivo ".mot" en FDT.

Para establecer la comunicación con el microcontrolador es necesario seleccionar la opción *Conectar al Dispositivo* en la pestaña *Device*, y que la configuración del microcontrolador sea BOOT (ver configuración de jumpers en la tableta).
Cuando la interconexión esta realizada aparecerá un mensaje en la ventana inferior del FDT y para transferir el programa al microcontrolador, es necesario seleccionar la opción *Download Active File* desde la pestaña *Device*.

Por último deberá colocarse el dispositivo en modo RUN (Ver configuración de jumpers en la tableta) y presionar el botón de reset en la tableta que contiene el microcontrolador, de esta manera se iniciará el programa creado.

Capitulo Cuatro: Puertos [12]

[12] Para mayor información hacer referencia a la sección 8 del Manual de Hardware [B1].

4.1 Generalidades

El microcontrolador H8/3069F tiene 10 puertos de entrada/salida cuyos pines están multiplexados para diferentes funciones según se muestra más adelante en la tabla 4.5. Para controlar cada uno de estos puertos tenemos que manipular los registros de dirección (Data Direction Register: DDR) y de datos (Data Register: DR). En el primero de ellos indicamos si el puerto en cuestión será usado como entrada o salida y en el segundo tendremos los datos de entrada o de salida dependiendo como se ha configurado el puerto. Además de estos dos registros, tenemos, para los puertos 2, 4 y 5, un registro (Pull-up Control Register: PCR) en donde podemos desactivar o activar los transistores pull-up de entrada.

A continuación presentamos ciertos detalles eléctricos de los puertos:

Puertos	Detalles
1 a 6 y 8	soportan cargas TTL y cargas capacitivas de 90pF
9, A y B	soportan cargas TTL y cargas capacitivas de 30pF
1 a 6 y 8 a B	Pueden manejar pares Darlington
1, 2 y 5	Pueden manejar LED (10mA)
Pines	
$P8_2$-$P8_0$	Posen circuitos de entrada Schmitt-trigger
PA_7-PA_0	Posen circuitos de entrada Schmitt-trigger

Tabla 4. 1: Características eléctricas de los puertos

Para detallar un poco más el uso de los registros tomaremos el puerto 1 como ejemplo. Para cada uno de los puertos tenemos una dirección diferente para los registros DDR y DR, en este caso tenemos lo siguiente:

				Initial Value	
Address*	Name	Abbreviation	R/W	Modes 1 to 4	Modes 5 and 7
H'EE000	Port 1 data direction register	P1DDR	W	H'FF	H'00
H'FFFD0	Port 1 data register	P1DR	R/W	H'00	H'00
Note: * Lower 20 bits of the address in advanced mode.					

Tabla 4. 2: Dirección de los registros del puerto 1.

El puerto 1 es un puerto de entrada/salida y la configuración de su función debe establecerse en el registro P1DDR. En este podemos configurar cada uno de los pines correspondientes como entrada o como salida colocando un cero o un uno respectivamente en cada uno de los bits del registro.

Bit		7	6	5	4	3	2	1	0
		$P1_7DDR$	$P1_6DDR$	$P1_5DDR$	$P1_4DDR$	$P1_3DDR$	$P1_2DDR$	$P1_1DDR$	$P1_0DDR$
Modes 1 to 4	Initial value	1	1	1	1	1	1	1	1
	Read/Write	—	—	—	—	—	—	—	—
Modes 5 and 7	Initial value	0	0	0	0	0	0	0	0
	Read/Write	W	W	W	W	W	W	W	W

Port 1 data direction 7 to 0
These bits select input or output for port 1 pins

Tabla 4. 3: Registro de direccionamiento del puerto 1.

Podemos ver que según sea el modo de operaciones así tenemos diferentes valores iniciales. Es de recordar que durante este trabajo solamente vamos a usar los modos cinco y siete.

> Nota: El registro DDR, para los modos de operación empleados, solo es de escritura. De modo que al querer leer los datos que allí hay, solo obtendremos unos.

Por otro lado tenemos el registro de datos P1DR en el cual tenemos el valor de salida o entrada según sea el caso.

Bit	7	6	5	4	3	2	1	0
	$P1_7$	$P1_6$	$P1_5$	$P1_4$	$P1_3$	$P1_2$	$P1_1$	$P1_0$
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Port 1 data 7 to 0
These bits store data for port 1 pins

Tabla 4. 4: Registro de datos del puerto 1.

Cuando el puerto 1 funciona como salida, el valor de el P1DR es el valor que será escrito, pero solo para los pines que estén configurados como salida (P1DDR=1). Si tenemos algunos bits configurados como entrada, al momento de hacer la lectura, el valor leído será almacenado únicamente en los bits así configurados.

En caso de trabajar con cualquier otro puerto los registros a modificar y configurar son los anteriores: PxDDR y PxDR; la ubicación de estos registros depende del puerto al que estemos haciendo referencia.

Cada uno de los puertos tiene funciones diferentes según sea el modo de operaciones que se ha seleccionado y en muchas ocasiones los pines de los puertos han sido multiplexados con diferentes subsistemas tal como se muestra en las tablas siguientes:

Port	Description	Pins	Expanded Modes					Single-Chip Modes
			Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 7
Port 1	• 8-bit I/O port • Can drive LEDs	$P1_7$ to $P1_0$/ A_7 to A_0	Address output pins (A_7 to A_0)				Address output (A_7 to A_0) and generic input DDR = 0: generic input DDR = 1: address output	Generic input/output
Port 2	• 8-bit I/O port • Built-in input pull-up transistors • Can drive LEDs	$P2_7$ to $P2_0$/ A_{15} to A_8	Address output pins (A_{15} to A_8)				Address output (A_{15} to A_8) and generic input DDR = 0: generic input DDR = 1: address output	Generic input/output
Port 3	• 8-bit I/O port	$P3_7$ to $P3_0$/ D_{15} to D_8	Data input/output (D_{15} to D_8)					Generic input/output
Port 4	• 8-bit I/O port • Built-in input pull-up transistors	$P4_7$ to $P4_0$/ D_7 to D_0	Data input/output (D_7 to D_0) and 8-bit generic input/output 8-bit bus mode: generic input/output 16-bit bus mode: data input/output					Generic input/output
Port 5	• 4-bit I/O port • Built-in input pull-up transistors • Can drive LEDs	$P5_3$ to $P5_0$/ A_{19} to A_{16}	Address output (A_{19} to A_{16})				Address output (A_{19} to A_{16}) and 4-bit generic input DDR = 0: generic input DDR = 1: address output	Generic input/output
Port 6	• 7-bit I/O port and 1-bit input port	$P6_7$/ϕ	Clock output (ϕ) and generic input					
		$P6_6$/$\overline{LWR}$ $P6_5$/$\overline{HWR}$ $P6_4$/$\overline{RD}$ $P6_3$/$\overline{AS}$	Bus control signal output ($\overline{LWR}$, $\overline{HWR}$, $\overline{RD}$, $\overline{AS}$)					Generic input/output
		$P6_2$/$\overline{BACK}$ $P6_1$/$\overline{BREQ}$ $P6_0$/$\overline{WAIT}$	Bus control signal input/output ($\overline{BACK}$, $\overline{BREQ}$, $\overline{WAIT}$) and 3-bit generic input/output					
Port 7	• 8-bit input port	$P7_7$/AN_7/DA_1 $P7_6$/AN_6/DA_0	Analog input (AN_7, AN_6) to A/D converter, analog output (DA_1, DA_0) from D/A converter, and generic input					
		$P7_5$ to $P7_0$/ AN_5 to AN_0	Analog input (AN_5 to AN_0) to A/D converter, and generic input					
Port 8	• 5-bit I/O port • $P8_2$ to $P8_0$ have Schmitt inputs	$P8_4$/$\overline{CS_0}$	DDR = 0: generic input DDR = 1 (reset value): $\overline{CS_0}$ output				DDR = 0 (reset value): generic input DDR = 1: $\overline{CS_0}$ output	Generic input/output
		$P8_3$/$\overline{IRQ_3}$/ $\overline{CS_1}$/$\overline{ADTRG}$	$\overline{IRQ_3}$ input, $\overline{CS_1}$ output, external trigger input ($\overline{ADTRG}$) to A/D converter, and generic input DDR = 0 (after reset): generic input DDR = 1: $\overline{CS_1}$ output					$\overline{IRQ_3}$ input, external trigger input ($\overline{ADTRG}$) to A/D converter, and generic input/output
		$P8_2$/$\overline{IRQ_2}$/$\overline{CS_2}$ $P8_1$/$\overline{IRQ_1}$/$\overline{CS_3}$	$\overline{IRQ_2}$ and $\overline{IRQ_1}$ input, $\overline{CS_2}$ and $\overline{CS_3}$ output, and generic input* DDR = 0 (reset value): generic input DDR = 1: $\overline{CS_2}$ and $\overline{CS_3}$ output					$\overline{IRQ_2}$ and $\overline{IRQ_1}$ input and generic input/output
		$P8_0$/$\overline{IRQ_0}$ /$\overline{RFSH}$	$\overline{IRQ_0}$ input, $\overline{RFSH}$ output, and generic input/output					$\overline{IRQ_0}$ input and generic input/output

Tabla 4. 5: Funciones de los puertos.

Port	Description	Pins	Expanded Modes					Single-Chip Modes
			Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 7
Port 9	• 8-bit I/O port	$P9_7/\overline{IRQ_5}$ /SCK_1 $P9_6/\overline{IRQ_4}$ /SCK_0 $P9_5/RxD_1$ $P9_4/RxD_0$ $P9_3/TxD_1$ $P9_2/TxD_0$	Input and output (SCK_1, SCK_0, RxD_1, RxD_0, TxD_1, TxD_0) for serial communication interfaces 1 and 0 (SCI1/0), $\overline{IRQ_5}$ and $\overline{IRQ_4}$ input, and 8-bit generic input/output					
Port A	• 8-bit I/O port • Schmitt inputs	PA_7/TP_7/ $TIOCB_2/A_{20}$	Output (TP_7) from programmable timing pattern controller (TPC), input or output ($TIOCB_2$) for 16-bit timer and generic input/output		Address output (A_{06})		Address output (A_{06}), TPC output (TP_7), input or output ($TIOCB_2$) for 16-bit timer, and generic input/output	TPC output (TP_7), 16-bit timer input or output ($TIOCB_2$), and generic input/output
		PA_6/TP_6/ $TIOCA_2/A_{21}$ PA_5/TP_5/ $TIOCB_1/A_{22}$ PA_4/TP_4/ $TIOCA_1/A_{23}$	TPC output (TP_6 to TP_4), 16-bit timer input and output ($TIOCA_2$, $TIOCB_1$, $TIOCA_1$), and generic input/output		TPC output (TP_6 to TP_4),16-bit timer input and output ($TIOCA_2$, $TIOCB_1$, $TIOCA_1$), address output (A_{21} to A_{23}), and generic input/output			TPC output (TP_6 to TP_4), 16-bit timer input and output ($TIOCA_2$, $TIOCB_1$, $TIOCA_1$) and generic input/output
		PA_3/TP_3/ $TIOCB_0$/ TCLKD PA_2/TP_2/ $TIOCA_0$/ TCLKC PA_1/TP_1/ TCLKB /$\overline{TEND_1}$ PA_0/TP_0/ TCLKA /$\overline{TEND_0}$	TPC output (TP_3 to TP_0), 16-bit timer input and output ($TIOCB_0$, $TIOCA_0$, TCLKD, TCLKC, TCLKB, TCLKA), 8-bit timer input (TCLKD, TCLKC, TCLKB, TCLKA), output ($\overline{TEND_1}$, $\overline{TEND_0}$) from DMA controller (DMAC), and generic input/output					
Port B	• 8-bit I/O port	PB_7/TP_{15}/ RXD_2 PB_6/TP_{14}/ TXD_2 PB_5/TP_{13}/ $SCK_2/\overline{LCAS}$ PB_4/TP_{12}/ $\overline{UCAS}$	TPC output (TP_{15} to TP_{12}), SCI2 input and output (SCK_2, RxD_2, TxD_2), DRAM interface output ($\overline{LCAS}$, $\overline{UCAS}$), and generic input/output					TPC output (TP_{15} to TP_{12}), SCI2 input and output (SCK_2, RxD_2, TxD_2), and generic input/output
		PB_3/TP_{11}/ $TMIO_3$/ $\overline{DREQ_1}/\overline{CS_4}$ PB_2/TP_{10}/ $TMO_2/\overline{CS_5}$ PB_1/TP_9/ $TMIO_1$/ $\overline{DREQ_0}/\overline{CS_6}$ PB_0/TP_8/ $TMO_0/\overline{CS_7}$	TPC output (TP_{11} to TP_8), 8-bit timer input and output ($TMIO_3$, TMO_2, $TMIO_1$, TMO_0), DMAC input ($\overline{DREQ_1}$, $\overline{DREQ_0}$), $\overline{CS_7}$ to $\overline{CS_4}$ output, and generic input/output					TPC output (TP_{11} to TP_8), 8-bit timer input and output ($TMIO_3$, TMO_2, $TMIO_1$, TMO_0), DMAC input ($\overline{DREQ_1}$, $\overline{DREQ_0}$), and generic input/output

Tabla 4. 6: Funciones de los puertos.

Capitulo Cinco: Temporizador de 16 bits [13]

[13] Para mayor información hacer referencia a la sección 9 del Manual de Hardware [B1].

5.1 Generalidades

El microcontrolador H8/3069F posee 3 canales de temporizadores/contadores de 16 bits en un módulo integrado lo que le permite procesar hasta 6 pulsos de salida o 6 pulsos de entrada. Además, existe la posibilidad es escoger una de las 8 fuentes (cuatro internas y cuatro externas) para el Clock de los tres canales y es posible seleccionar, para cada uno de los canales, uno de los 5 modos de operación siguientes:

- Forma de onda de salida al tener una comparación exitosa:
  Podemos escoger entre una salida de 0, una salida de 1 o un Toggle cuando tengamos una comparación exitosa. Para el canal 2 solo podremos tener las dos primeras opciones.

- Función de captura de entrada:
  Es posible identificar un evento mediante un flanco positivo, un flanco negativo o por cualquiera de los dos.

- Limpieza del contador:
  Es posible reiniciar el contador al tener una comparación exitosa o una captura de entrada.

- Sincronización;
  Dos o más contadores pueden ser iniciados o reiniciados al mismo tiempo dependiendo de un mismo evento.

- PWM:
  Es posible proveer una salida de PWM con un ciclo de trabajo determinado por el usuario. Además, con la sincronización es posible tener una salida de PWM trifásica.

Estos temporizadores poseen 6 registros (dos por canal) en donde están alojados los valores de Output Compare e Input Capture, además tenemos 9 fuentes de interrupciones (3 por canal), un conteo de fase en el canal 2, un acceso de alta velocidad a los contadores mediante un bus de 16 bits y un Switcheo de salida para el TPC (visto en el Capitulo 11).

Notas: - Las señales de los canales 0, 1 y 2 pueden ser usados como salidas para el TPC.
- Los contadores pueden ser inicializados con cualquier valor.

Esta información así como las funciones de los temporizadores de 16 bits se muestran en la tabla 5.1 y en las figuras 5.1, 5.2 y 5.3 se muestran los diagramas de bloque general y especifico de los tres temporizadores que contiene este Mcu.

Item		Channel 0	Channel 1	Channel 2
Clock sources		Internal clocks: ϕ, ϕ/2, ϕ/4, ϕ/8 External clocks: TCLKA, TCLKB, TCLKC, TCLKD, selectable independently		
General registers (output compare/input capture registers)		GRA0, GRB0	GRA1, GRB1	GRA2, GRB2
Input/output pins		$TIOCA_0$, $TIOCB_0$	$TIOCA_1$, $TIOCB_1$	$TIOCA_2$, $TIOCB_2$
Counter clearing function		GRA0/GRB0 compare match or input capture	GRA1/GRB1 compare match or input capture	GRA2/GRB2 compare match or input capture
Initial output value setting function		Available	Available	Available
Compare match output	0	Available	Available	Available
	1	Available	Available	Available
	Toggle	Available	Available	Not available
Input capture function		Available	Available	Available
Synchronization		Available	Available	Available
PWM mode		Available	Available	Available
Phase counting mode		Not available	Not available	Available
Interrupt sources		Three sources • Compare match/input capture A0 • Compare match/input capture B0 • Overflow	Three sources • Compare match/input capture A1 • Compare match/input capture B1 • Overflow	Three sources • Compare match/input capture A2 • Compare match/input capture B2 • Overflow

Tabla 5. 1 Funciones de los temporizadores de 16 bits.

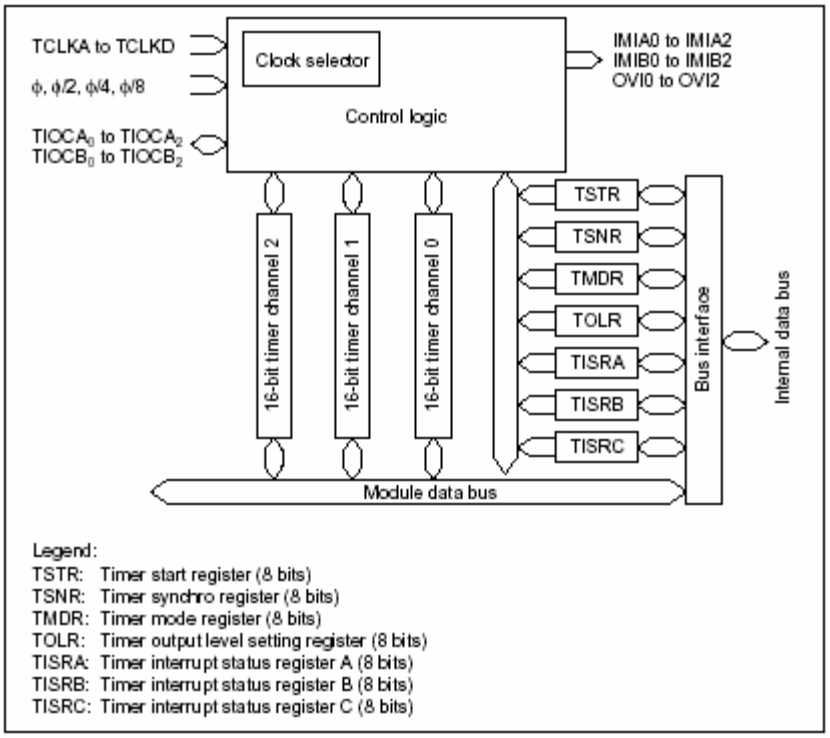


Figura 5. 1: Diagramas de bloques generales

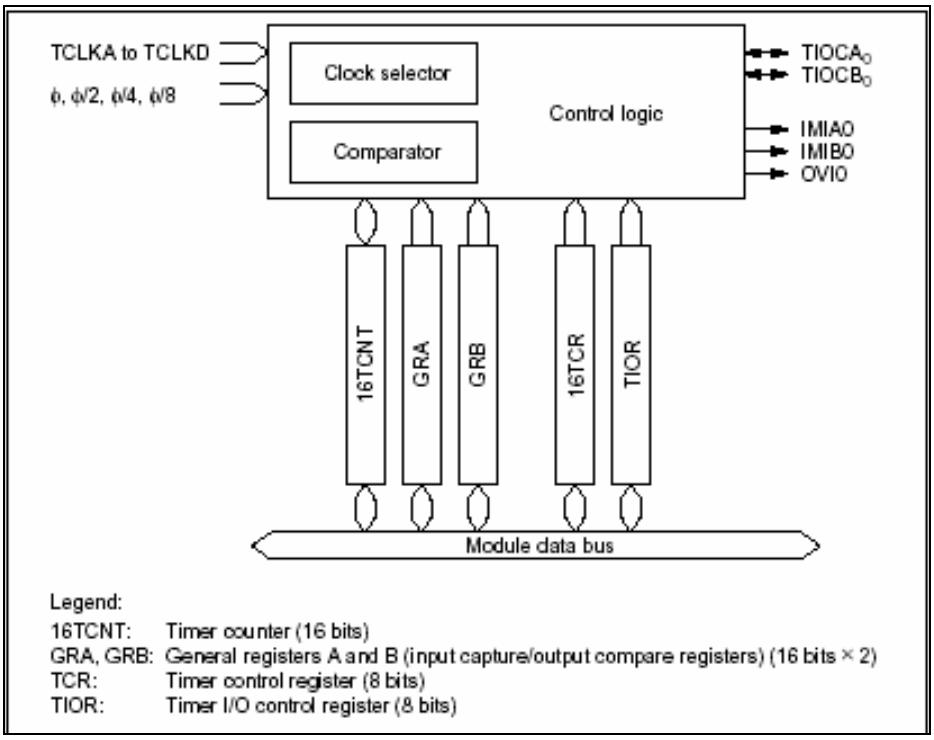


Figura 5. 2: Diagrama especifico de los canales 0 y 1.

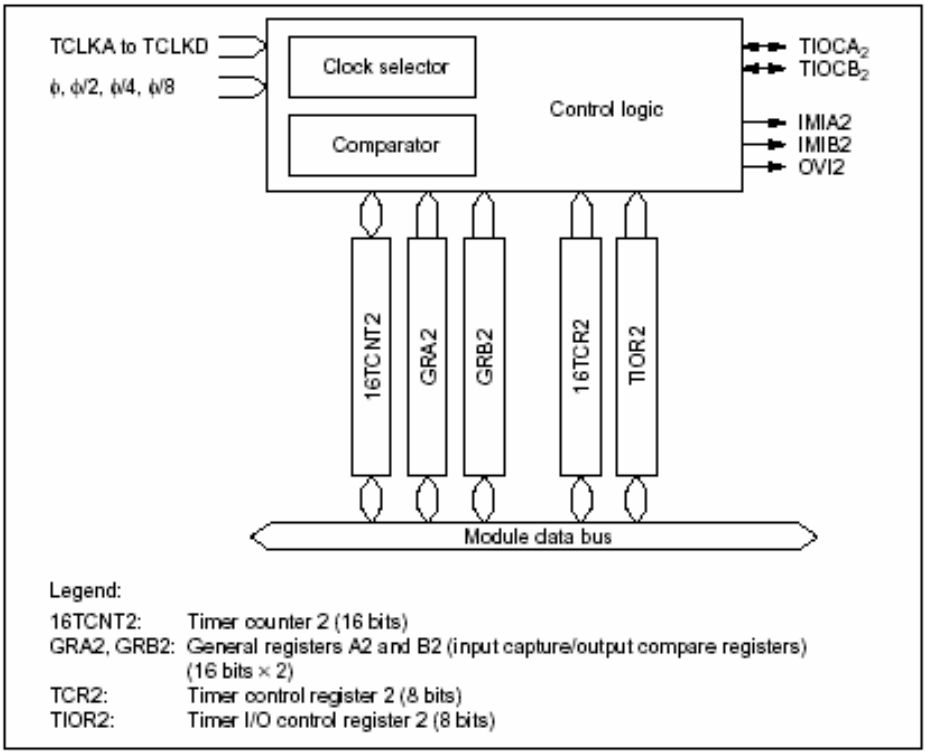


Figura 5. 3: Diagrama especifico de los canales 2

5.2 Registros

Para poder manejar estos contadores de 16 bits, es necesario configurar ciertos parámetros con la ayuda de los registros comunes y de los registros específicos de cada unos de los canales. En la siguiente tabla se muestran cada uno de los registros disponibles, así como su dirección dentro del banco de memoria y si el registro es común o especifico.

Channel	Address*1	Name	Abbre-viation	R/W	Initial Value
Common	H'FFF60	Timer start register	TSTR	R/W	H'F8
	H'FFF61	Timer synchro register	TSNC	R/W	H'F8
	H'FFF62	Timer mode register	TMDR	R/W	H'98
	H'FFF63	Timer output level setting register	TOLR	W	H'C0
	H'FFF64	Timer interrupt status register A	TISRA	R/(W)*2	H'88
	H'FFF65	Timer interrupt status register B	TISRB	R/(W)*2	H'88
	H'FFF66	Timer interrupt status register C	TISRC	R/(W)*2	H'88
0	H'FFF68	Timer control register 0	16TCR0	R/W	H'80
	H'FFF69	Timer I/O control register 0	TIOR0	R/W	H'88
	H'FFF6A	Timer counter 0H	16TCNT0H	R/W	H'00
	H'FFF6B	Timer counter 0L	16TCNT0L	R/W	H'00
	H'FFF6C	General register A0H	GRA0H	R/W	H'FF
	H'FFF6D	General register A0L	GRA0L	R/W	H'FF
	H'FFF6E	General register B0H	GRB0H	R/W	H'FF
	H'FFF6F	General register B0L	GRB0L	R/W	H'FF
1	H'FFF70	Timer control register 1	16TCR1	R/W	H'80
	H'FFF71	Timer I/O control register 1	TIOR1	R/W	H'88
	H'FFF72	Timer counter 1H	16TCNT1H	R/W	H'00
	H'FFF73	Timer counter 1L	16TCNT1L	R/W	H'00
	H'FFF74	General register A1H	GRA1H	R/W	H'FF
	H'FFF75	General register A1L	GRA1L	R/W	H'FF
	H'FFF76	General register B1H	GRB1H	R/W	H'FF
	H'FFF77	General register B1L	GRB1L	R/W	H'FF
2	H'FFF78	Timer control register 2	16TCR2	R/W	H'80
	H'FFF79	Timer I/O control register 2	TIOR2	R/W	H'88
	H'FFF7A	Timer counter 2H	16TCNT2H	R/W	H'00
	H'FFF7B	Timer counter 2L	16TCNT2L	R/W	H'00
	H'FFF7C	General register A2H	GRA2H	R/W	H'FF
	H'FFF7D	General register A2L	GRA2L	R/W	H'FF
	H'FFF7E	General register B2H	GRB2H	R/W	H'FF
	H'FFF7F	General register B2L	GRB2L	R/W	H'FF

Notes: *1 The lower 20 bits of the address in advanced mode are indicated.
*2 Only 0 can be written in bits 3 to 0, to clear the flags.

Tabla 5. 2: Registros y sus direcciones.

5.3 Descripción:

5.3.1 Registros Comunes:

Timer Start Register (TSTR)

Este es un registro de lectura/escritura de 8 bits, de los cuales solo los tres menos significativos son funcionales (los demás están reservados y tienen valor inicial de uno). Estos bits indican si cada uno de los canales esta detenido (bit en cero) o esta contando (bits en uno). Para hacer que un determinado canal comience el conteo o lo detenga es necesario escribir el valor correspondiente (uno para Start y cero para Stop) en la posición adecuada.

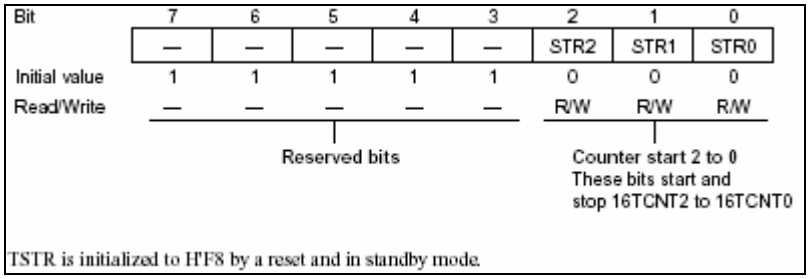


Figura 5. 4 Timer Start Register (TSTR)

Timer Synchro Register (TSNC)

Este registro permite indicar si los canales operaran de manera independiente (poniendo el bits en cero) o de manera sincronizada (poniendo el bit en uno). De igual forma que el registro anterior, los tres bits menos significativos son los bits funcionales y los otros cinco están inicializados con el valor de uno y están reservados.

Un canal, al trabajar de manera sincronizada, tiene la posibilidad de ser activado o desactivado junto con un evento en otro canal.

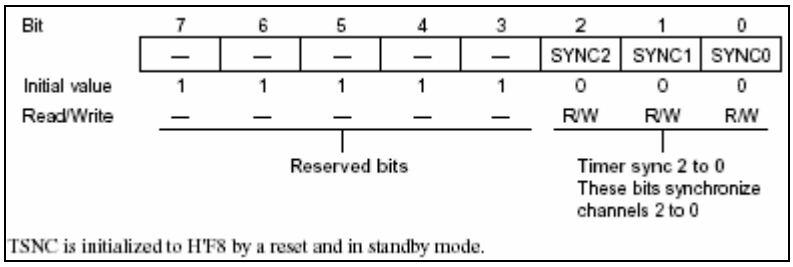


Figura 5. 5: Timer Synchro Register (TSNC)

Timer Mode Register (TMDR)

Registro de lectura/escritura el cual nos permite seleccionar el modo de trabajo PWM para cada uno de los canales de forma independiente. Además, para el canal 2, permite seleccionar el modo de conteo de fase (bit 6 en uno) o modo normal (bit 6 en cero) y establecer la condición de la bandera de Overflow (OVF).

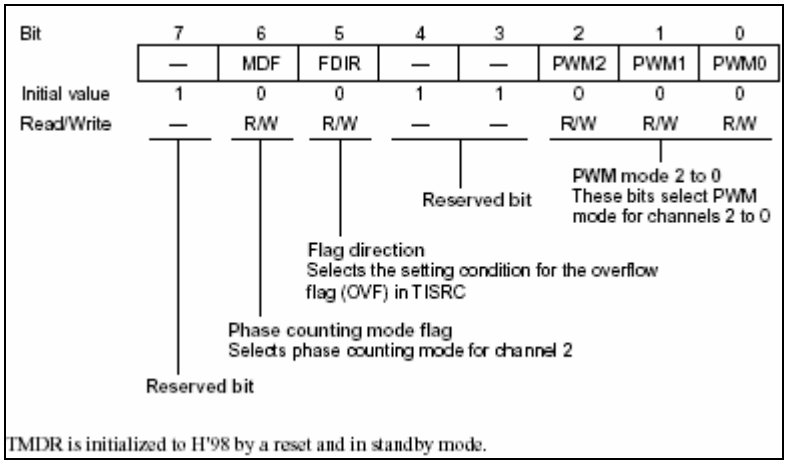


Figura 5. 6: Timer Mode Register (TMDR)

♦ Conteo de fase (MDF):

Cuando el modo de conteo de fase es seleccionado, el canal 2 funciona entonces como un contador UP/DOWN como lo indica la tabla 5.3 y los pines TCLKA y TCLKB se convierten en entradas de Clock.

Counting Direction	Down-Counting				Up-Counting			
TCLKA pin	↑	High	↓	Low	Low	↑	High	↓
TCLKB pin	Low	↑	High	↓	↑	High	↓	Low

Tabla 5. 3: Conteo hacia arriba o hacia abajo.

Nota: La activación del modo de conteo de fase desactiva la selección del flanco de activación (CKEG1-CKEG0) y la selección de Clock (TPSC2-TPSC0) para el canal 2.

♦ Overflow (FDIR):

Indica la condición de la bandera de Overflow para el canal 2. El bits OVF es puesto en uno cuando tenemos un desbordamiento del contador (FDIR en uno) o puede ser puesto en uno por un Overflow o un Underflow (si FDIR en cero).

Nota: Un Underflow ocurre solo cuando el canal 2 esta operando en modo de conteo de fase, es decir, cuando opera como contador UP/DOWN.

♦ Modo PWM (PWM2-PWM0)

Si el bit esta en cero el canal respectivo opera de manera normal, al estar en uno opera como PWM en donde el pin TIOCAx se convierte en la salida de PWM. Dicha salida se pone en uno en una comparación exitosa con GRAx y se pone en cero en comparación exitosa con GRBx.

Timer Interrupt Status Register A (TISRA):

Con este registro podemos habilitar (IMIEAx en uno) o deshabitar (IMIEAx en cero) la petición de una interrupción por la bandera IMFAx y ver el estado de la bandera (IMFAx) que indica un evento de comparación exitosa entre el valor del contador y el valor determinado en GRAx, o cuando la entrada capturada es transferida a GRAx (en estos dos últimos casos, la bandera se pone en uno).

Para limpiar la bandera debemos de escribir un cero en IMFAx después de haber leído ese bit.

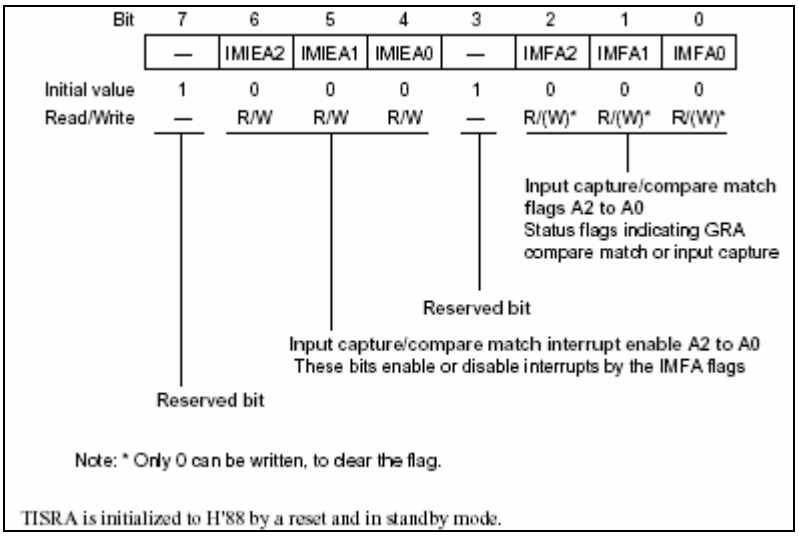


Figura 5. 7: Timer Interrupt Status Register A (TISRA).

Timer Interrupt Status Register B (TISRB):

Este registro tiene la misma función que el anterior con la excepción que las comparaciones o transferencias de la entrada se hacen con GRBx. La habilitación o no de la interrupción (IMIEBx) por medio de la bandera IMFBx es similar al caso anterior.

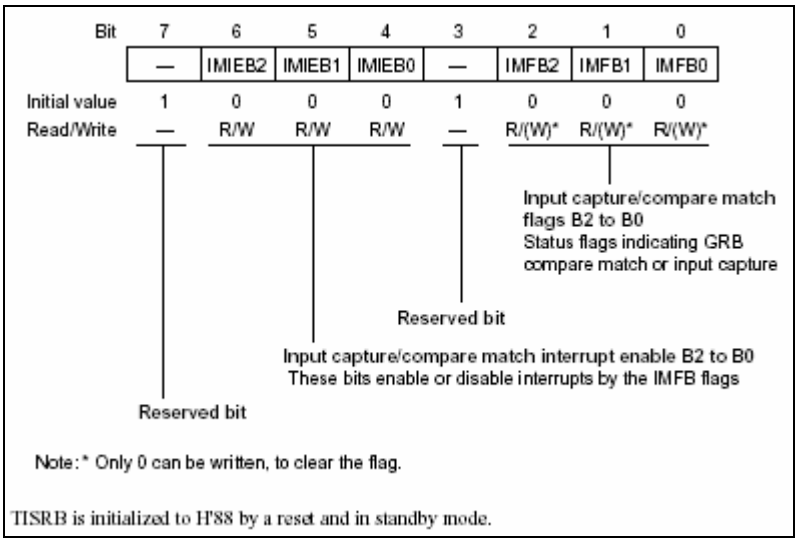


Figura 5. 8: Timer Interrupt Status Register B (TISRB).

Timer Interrupt Status Register C (TISRC).

Con este registro podemos ver si el contador x ha desbordado o no y activar o desactivar la petición de interrupción por Overflow. Con los bits del 6 al 4 deshabitamos (OVIx en cero) o habilitamos (OVIx en uno) la petición de interrupción y en los bit del 2 al 0 tenemos las banderas de Overflow las cuales se activan cuando pasamos de H'FFFF a H'0000 (Overflow) o de H'0000 a H'FFFF (Underflow). Para desactivarla tenemos que leerla primero y luego escribir un cero en ese Bit.

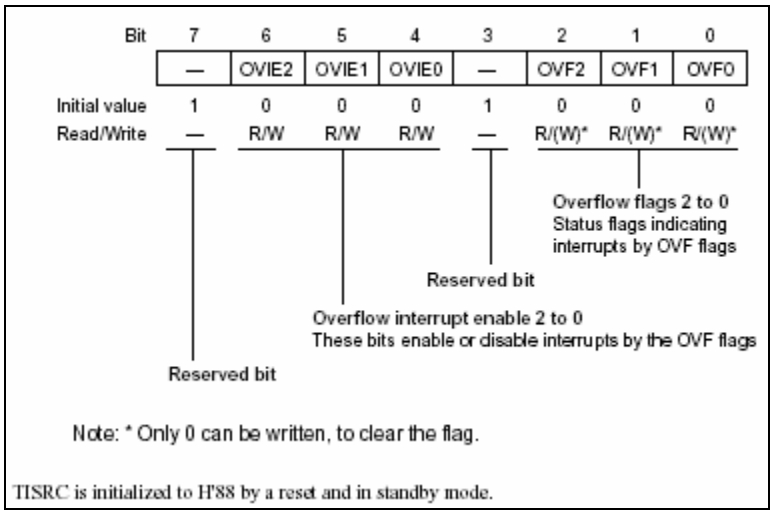


Figura 5. 9: Timer Interrupt Status Register C (TISRC).

Timer Output Level Setting Register (TOLR).

Este registro de escritura únicamente permite seleccionar el nivel de salida de los eventos de los temporizadores, canales 0 a 2. Al poner un cero en estos bits, tendremos un nivel de cero en la salida correspondiente y al poner un uno, tendremos un uno.

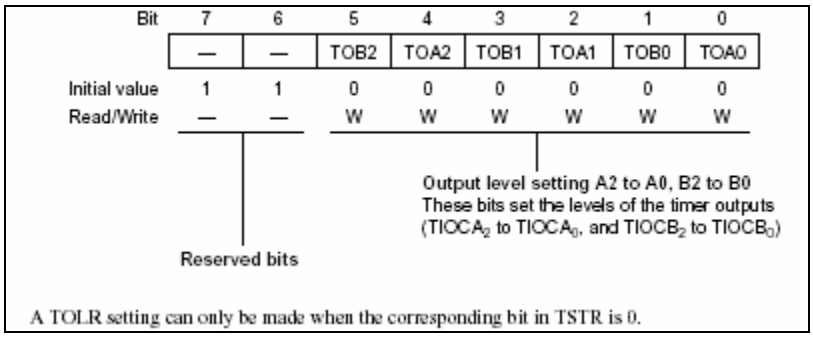


Figura 5. 10: Timer Output Level Setting Register (TOLR).

5.3.2 Registros Específicos:

Timer Counters (16TCNTx)

Tenemos tres registros de 16 bits separados en seis registros (parte alta y parte baja) de 8 bits, dos para cada canal. Cada registro es de lectura/escritura y cuenta los pulsos que provienen del reloj seleccionado con TPSC2-TPSC0 del 16TCR.
Los tres canales funcional como contadores ascendentes en cualquier modo y el canal dos funciona como contador ascendente/descendente cuando esta en modo de conteo de fase.

Los contadores, los cuales son inicializados con cero, pueden ser puestos a cero H'0000 al tener una comparación exitosa del valor que tienen con el valor almacenado en GRAx o GRBx, o cuando tienen un nuevo valor de captura puesto en GRA/Bx. Por otro lado, cada vez que tengamos un Overflow o un Underflow la bandera de OVFx se activara correspondiente a cada canal.

> Nota: Se tomara en cuenta el Underflow dependiendo de la configuración de FDIR.

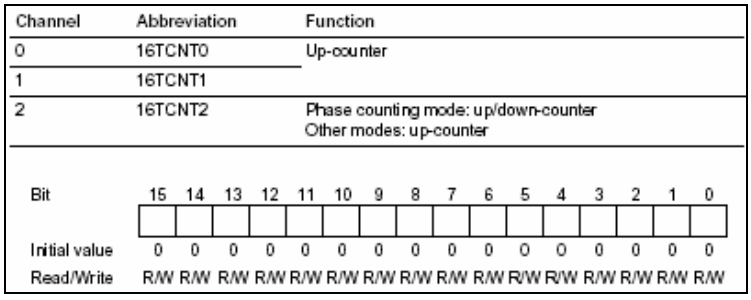


Figura 5. 11: Timer Counters (16TCNTx).

Registros generales (GRA y GRB):

Tenemos seis registros generales, dos para cada canal (GRAx y GRBx) los cuales están a su vez separados en dos registros de 8 bits. Estos registros son de lectura/escritura y pueden funcionar para comparaciones o para capturas de eventos dependiendo de la configuración del TIOR (la configuración del TIOR es ignorada en modo de PWM).

Si se esta trabajando con comparaciones, el valor del registro es comparado en todo momento con el valor del contador. Cuando la comparación es exitosa las banderas de IMFA o IMFB se ponen en 1 en los registros TISRA o TISRB.

Si se esta trabajando como Input Capture, la detección de una señal externa determina la captura del valor del contador en el registro general designado. Las banderas IMFA o IMFB se ponen en 1 en los registros TISRA o TISRB al mismo tiempo de la captura.

> Nota: Los flancos de los eventos de salidas y los flancos de detección de eventos pueden ser configurados en TIOR. Estos registros son inicializados como de comparación y con un valor de H'FFFF.

Channel	Abbreviation	Function
0	GRA0, GRB0	Output compare/input capture register
1	GRA1, GRB1	
2	GRA2, GRB2	

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figura 5. 12: Registros generales (GRA y GRB).

Timer Control Registers (16TCRx)

Existen tres registros como este, uno para cada uno de los canales, desde el cual podemos seleccionar las diferentes opciones para la fuente de limpieza del contador, flancos a tomar en cuenta para el conteo y la fuente de reloj para el temporizador:

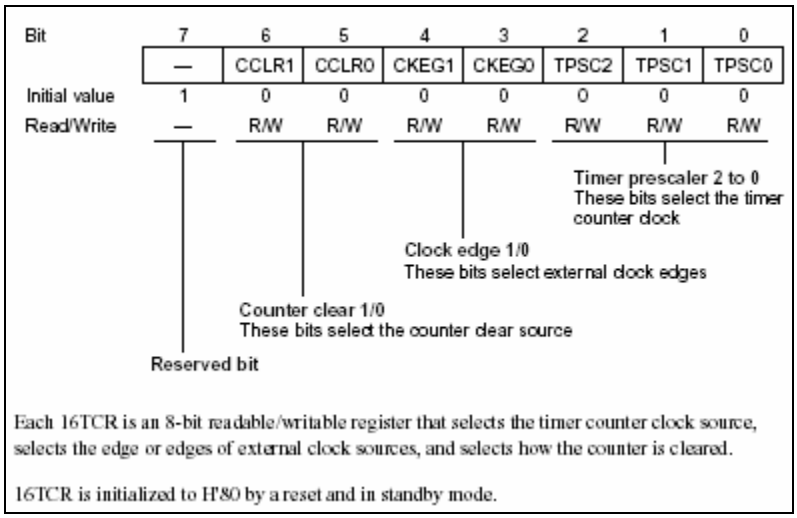


Figura 5. 13: Timer Control Registers (16TCRx)

Bit 6	Bit 5	
0	0	El Contador no es limpiado
0	1	Limpieza automática con un evento de captura o comparación en GRA
1	0	Limpieza automática con un evento de captura o comparación en GRB
1	1	El contador se limpia de manera sincronizada

Bit 4	Bit 3	
0	0	Toma en cuenta flanco ascendentes
0	1	Toma en cuenta flanco descendentes
1	-	Toma en cuenta lo dos Flancos

Bit 2	Bit 1	Bit 0	
0	0	0	Clock int.
0	0	1	Clock interno / 2
0	1	0	Clock interno / 4
0	1	1	Clock interno / 8
1	0	0	Clock externo A
1	0	1	Clock externo B
1	1	0	Clock externo C
1	1	1	Clock externo D

Notas:

- Cuando TPSC2 esta en cero estamos seleccionando un Clock interno, pero además determinamos que solo se tomaran en cuenta los flanco negativos.
- Con TPSC2 en uno tenemos acceso a los Clock externos y a seleccionar el flanco de acción.

- Si el canal 2 esta funcionando como contador de fase la selección del reloj y del flanco de acción son ignorados.

Timer I/O Control Registers (TIORx)

Del mismo modo que para el registro anterior, vamos atener un registro TIOR para cada canal desde el cual podemos controlar los registros generales GRA y GRB. Desde este registro de lectura/escritura podemos seleccionar si el registro general funcionará para entrada de captura o para comparación y dependiendo de eso selecciona a la vez el flanco que va a aceptar como evento o la salida que pondrá en caso de comparación exitosa.

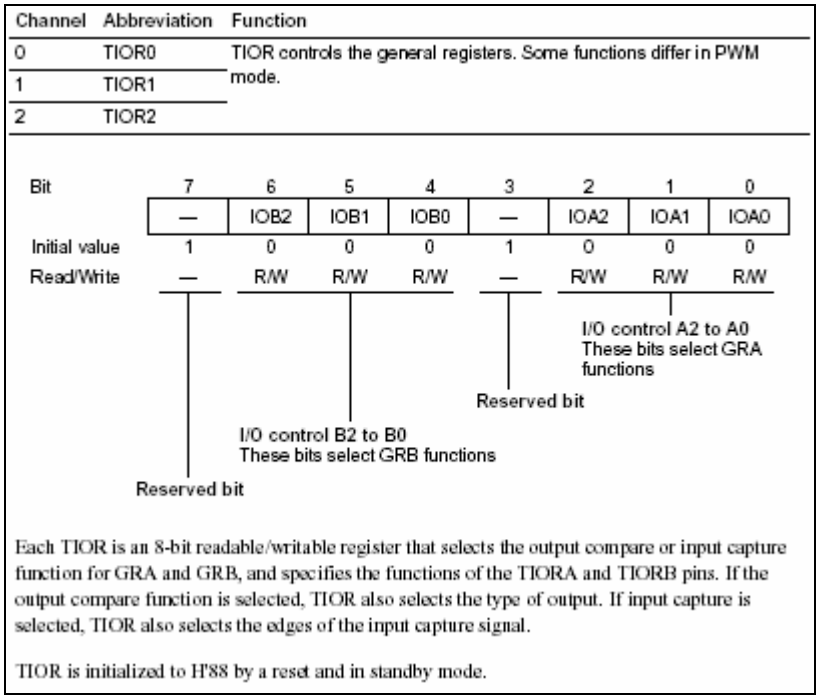


Figura 5. 14: Timer I/O Control Registers (TIOR)

Bit 6	Bit 5	Bit 4	
0	0	0	Sin salida en comparación exitosa con GRB
0	0	1	CERO de salida en comparación exitosa con GRB
0	1	0	UNO de salida en comparación exitosa con GRB
0	1	1	Toggle en la salida
1	0	0	GRB captura solo flancos positivos
1	0	1	GRB captura solo flancos negativos
1	1	0	Captura ambos flancos
1	1	1	

Para los Bits 2, 1 y 0 tenemos lo mismo pero para el registro general GRA.

Nota:
-El canal 2 no puede funcionar con Toggle en la salida, de modo que si se selecciona esa modalidad (011 en los bits 6-5-4 o 2-1-0) la salida que se pondrá será de 1 al haber una comparación exitosa.

5.4 Operación

Los temporizadores de 16 Bits pueden ser usados de diferentes formas y en diferentes modos de operación según el canal que se este utilizando y como se haya configurado:

Modo normal:

Tenemos tres canales de temporización y cada uno de ellos con su timer y sus registros. Este timer puede funcionar de las formas siguientes.

Conteo libre:
Al poner el STR de TSTR en 1 el conteo comenzará desde H´0000 hasta H´FFFF. Al llegar a este punto la bandera OVF se pone en uno y en contador vuelve a comenzar. El uso de la bandera para controlar una interrupción queda a responsabilidad del usuario.

Conteo periódico:
Un valor deberá ser programado en cualquiera de los registros generales y el temporizador se deberá configurar como "Compare Match." Al poner 1 en STR el conteo comenzara en H´0000 hasta el valor configurado en los registros. En ese momento el sistema pone 1 el IMFA o IMFB dependiendo que registro obtuviera la comparación exitosa y además se reinicia el contador si esa opción ha sido configurada. Si la petición de interrupción fue activada en IMIEA o IMIEB entonces la interrupción tomará efecto.
Al tener una comparación exitosa podemos dar una salida determinada dependiendo de la configuración de los registros. Salidas de 1, 0 o Toggle pueden ser seleccionadas, menos para la canal dos en donde el Toggle esta deshabilitado.

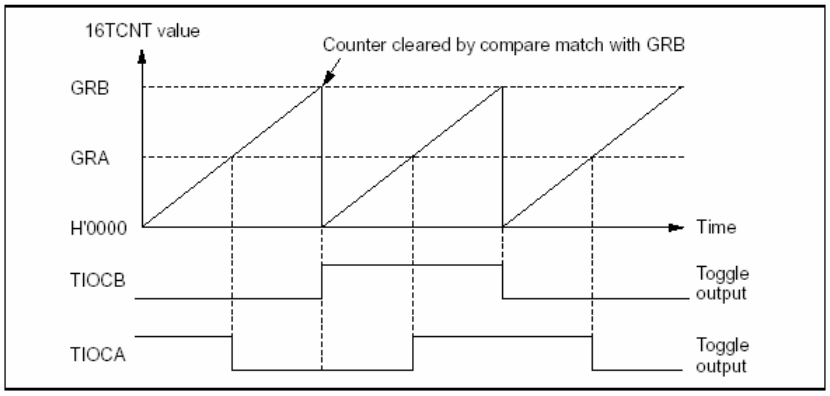


Figura 5. 15: Salidas Toggle.

Contador de eventos externos:
Seleccionamos el canal y configuramos que tipo de entrada va a ser reconocida como un evento. Después de poner 1 en STR el conteo inicia y al tener un evento de entrada, el valor del contador se almacena en los registros. La bandera de evento se activa y de estar propiamente configurado podría dar origen a una interrupción, al mismo tiempo el contador se reinicia. Al tener un nuevo evento, el valor anterior se elimina y el nuevo es tomado.

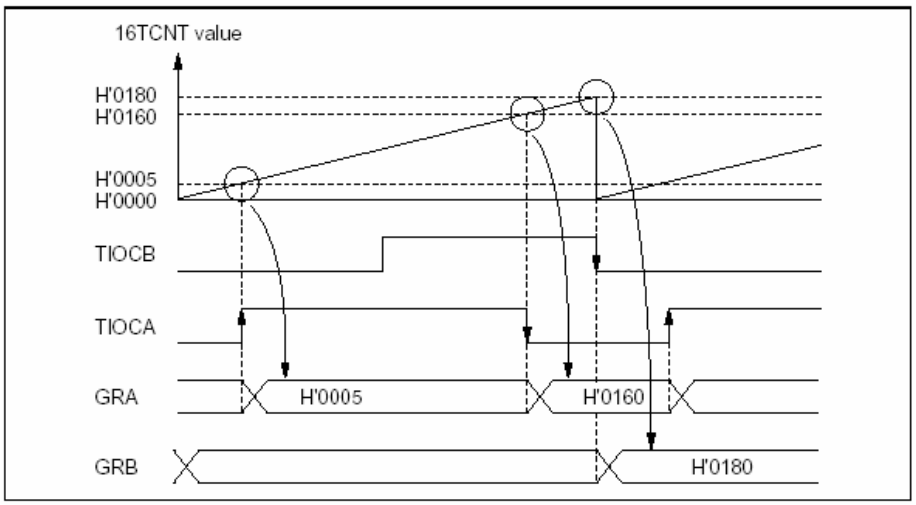


Figura 5. 16: Ejemplo de Captura de eventos.

Modo PWM:

Los registros funcionan como pares, el GRA podría especificar cuando la salida pasara a uno y el GRB especificaría cuando la salida pasara a cero (o viceversa). Además, debemos seleccionar para cualquiera de los dos la configuración de Compare Match para que el TIOCA sea la salida del PWM. Esto aplica para todos los canales.

Casos en los cuales el Duty Cicle será de 0%:

- Los dos registros poseen el mismo valor.
- Si GRB pone a cero y GRA pone a uno y el GRA es mayor que GRB.
- Caso contrario: Si GRA pone a cero y GRB pone a uno y el GRB es mayor que GRA.

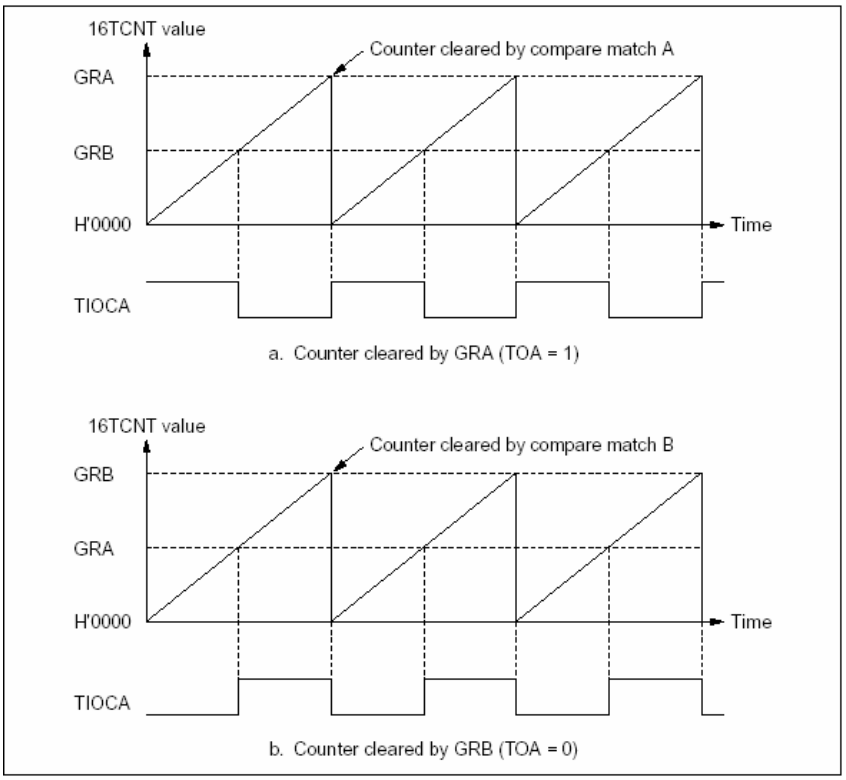


Figura 5. 17: Ejemplo de Modo PWM

Modo de Sincronía:

Dos o más canales pueden ser puestos en sincronía para ser escritos, borrados o para que comiencen al mismo momento. Además, permite tener una asociación de los registros con una sola base de tiempo, la aplicación más común en este caso seria un sistema trifásico.

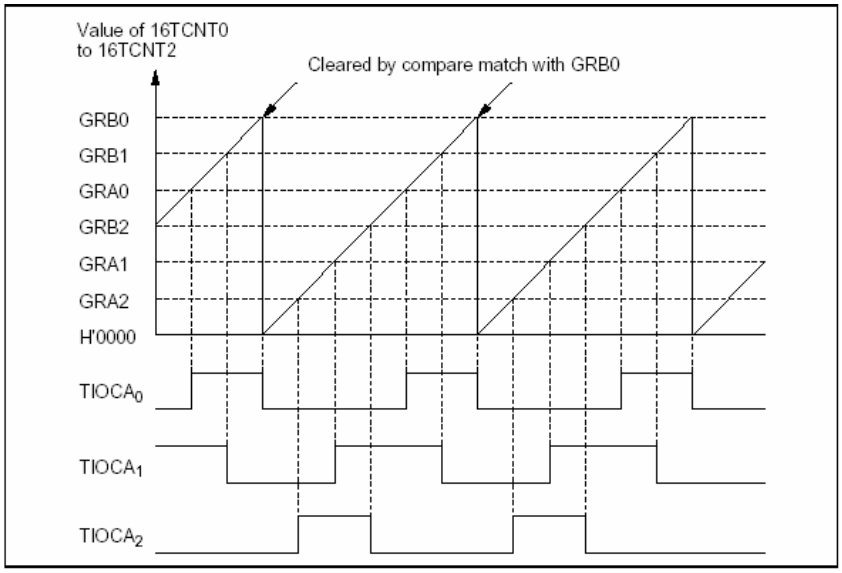


Figura 5. 18: Ejemplo de sincronía trifásica usando PWM

Modo de conteo de Fase:

En conteo de fase, solo disponible para canal 2, los pines TLCKA y TCLKB funcionan como entradas de reloj externo. Cuando se detecta una diferencia de fase entre estos dos un conteo ascendente o descendente se da.

Counting Direction	Up-Counting				Down-Counting			
TCLKB pin	↑	High	↓	Low	HIgh	↓	Low	↑
TCLKA pin	Low	↑	High	↓	↓	Low	↑	HIgh

Figura 5. 19: Conteo según fase.

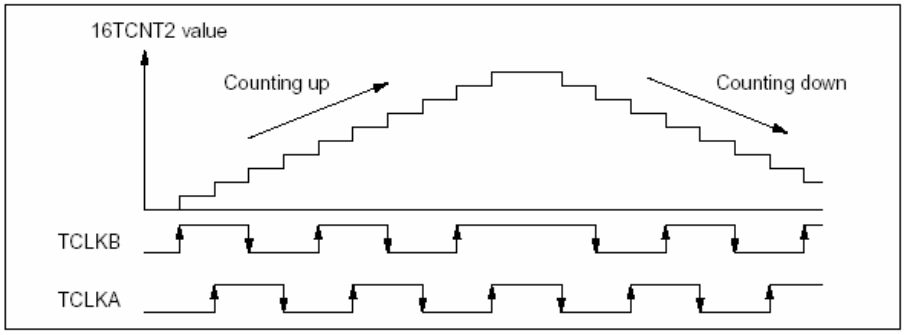


Figura 5. 20: Ejemplo de conteo de fase.

5.5 Interrupciones

Tenemos dos tipos de interrupciones posibles:

- Primero, las que son provocadas por las funciones de Output Compare e Input Capture. IMFA o IMFB es puesto a 1 cuando tenemos una comparación exitosa entre el valor del contador y el valor del registro, en caso de Output compare, o cuando se detecta un evento valido en caso de Input Capture. La puesta a uno es acompañado por el resto del procedimiento especificado anteriormente según sea el caso.

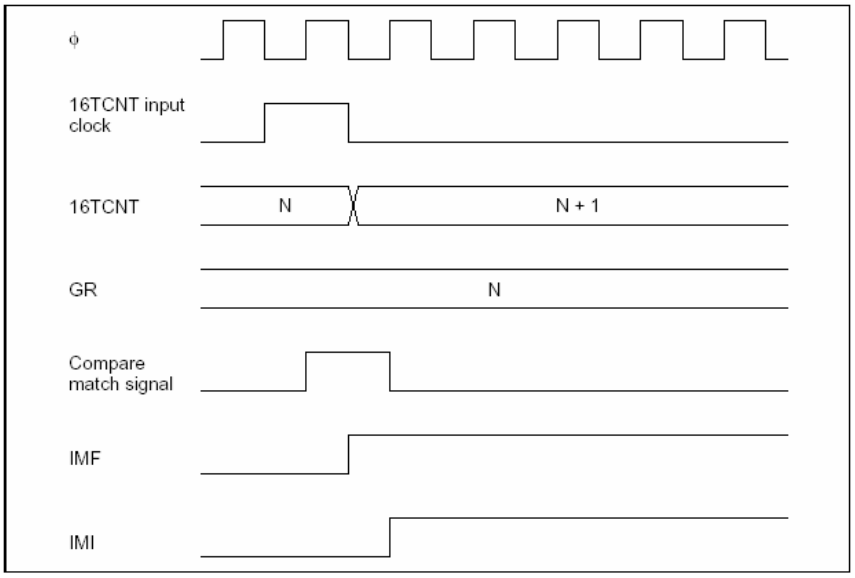


Figura 5. 21: Interrupción por Output Compare

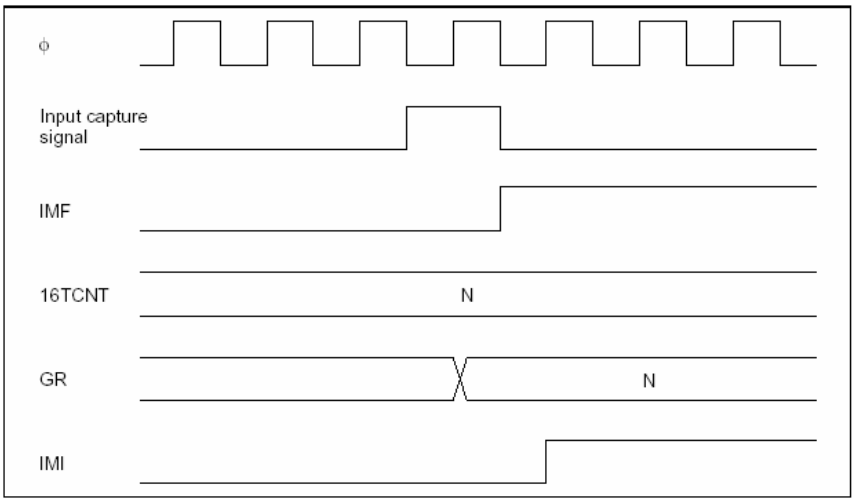


Figura 5. 22: Interrupción por Input Capture

- Segundo, las que son provocadas por Overflow o Underflow en el contador: OVF es puesto en 1 cuando el contador pasa de H´FFFF a H´0000 (Overflow) o cuando pasa de H´0000 a H´FFFF (Underflow).

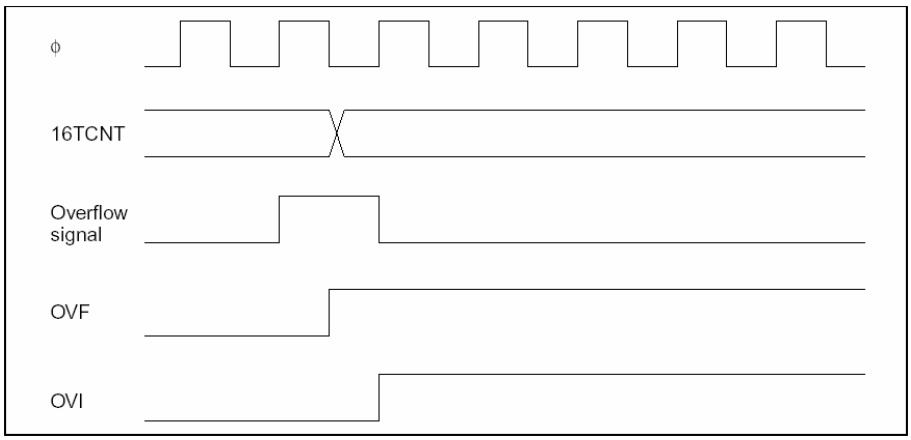


Figura 5. 23: Ejemplo de interrupción de Overflow

Nota: la limpieza de estas banderas de interrupción se da cuando se lee la bandera correspondiente y luego se escribe un cero en ella.

Podemos ver que cada uno de los canales puede generar 3 tipos de interrupciones diferentes las cuales responden a vectores totalmente independientes, de modo que podemos tener 9 interrupciones en total, siempre y cuando tengamos habilitada la respuesta de las interrupciones (IMIEAx, IMIEBx y OVIEx).
La prioridad de la interrupción se muestra a continuación:

Channel	Interrupt Source	Description	Priority*
0	IMIA0	Compare match/input capture A0	High
	IMIB0	Compare match/input capture B0	
	OVI0	Overflow 0	
1	IMIA1	Compare match/input capture A1	
	IMIB1	Compare match/input capture B1	
	OVI1	Overflow 1	
2	IMIA2	Compare match/input capture A2	
	IMIB2	Compare match/input capture B2	
	OVI2	Overflow 2	Low

Note: * The priority immediately after a reset is indicated. Inter-channel priorities can be changed by settings in IPRA.

Figura 5. 24: Prioridad de cada interrupción.

Capitulo Seis: Temporizador de 8 bits [14]

[14] Para mayor información hacer referencia a la sección 10 del Manual de Hardware [B1].

6.1 Generalidades

El microcontrolador H8/3069F tiene internamente un módulo de conteo de 8 bits con cuatro canales (TMR0, TMR1, TMR2 y TMR3). Cada canal tiene un registro de conteo (8TCNT) y dos registros de constantes temporales (TCORA y TCORB) los cuales son constantemente comparados con el 8TCNT con el fin de detectar comparaciones exitosas. Los contadores pueden ser usados como contadores multi-funcionales en una variedad de aplicaciones, como por ejemplo, la generación de una señal rectangular con un tiempo de ciclos arbitrarios.

Características.

- Selección de cuatro fuentes de reloj: los contadores pueden ser manejados por cualquiera de las tres señales internas o por una entrada externa de reloj (lo cual permitiría al contador ser usado como contador de eventos).
- Selección de tres maneras de limpiar los contadores: los contadores pueden ser limpiados por una comparación exitosa con cualquiera de los dos registros, o por una entrada de comparación en captura de eventos en el registro B.
- La salida del temporizador es controlada por dos señales de comparación independientes, habilitando el temporizador para generar salidas con forma de onda con duración de ciclos arbitrarios o PWM.
- El convertidor análogo-digital puede ser disparado por una comparación exitosa.
- Los canales 0-1 y 2-3 pueden ser operados como la parte alta y baja de un temporizador de 16 bits (en modalidad de 16 bits).
- El Canal 1 (3) puede contar los eventos de comparación del canal 0 (2).
- La entrada de comparación de eventos esta disponible para eventos de 8 y 16 bits.
- Consta con doce llamadas a interrupción: cuatro por fuentes de comparación, cuatro de entrada de captura de eventos y cuatro de desbordamiento. Ahora, algunas de estas comparten el mismo vector de interrupciones.

El contador o temporizador de 8 bits esta dividido en dos grupos de dos canales cada uno: el grupo 0 comprende los canales 0 y 1, y el grupo 1 comprende los canales 2 y 3. La figura 6.1 muestra el diagrama en bloques del temporizador de ocho bits para el grupo 0.

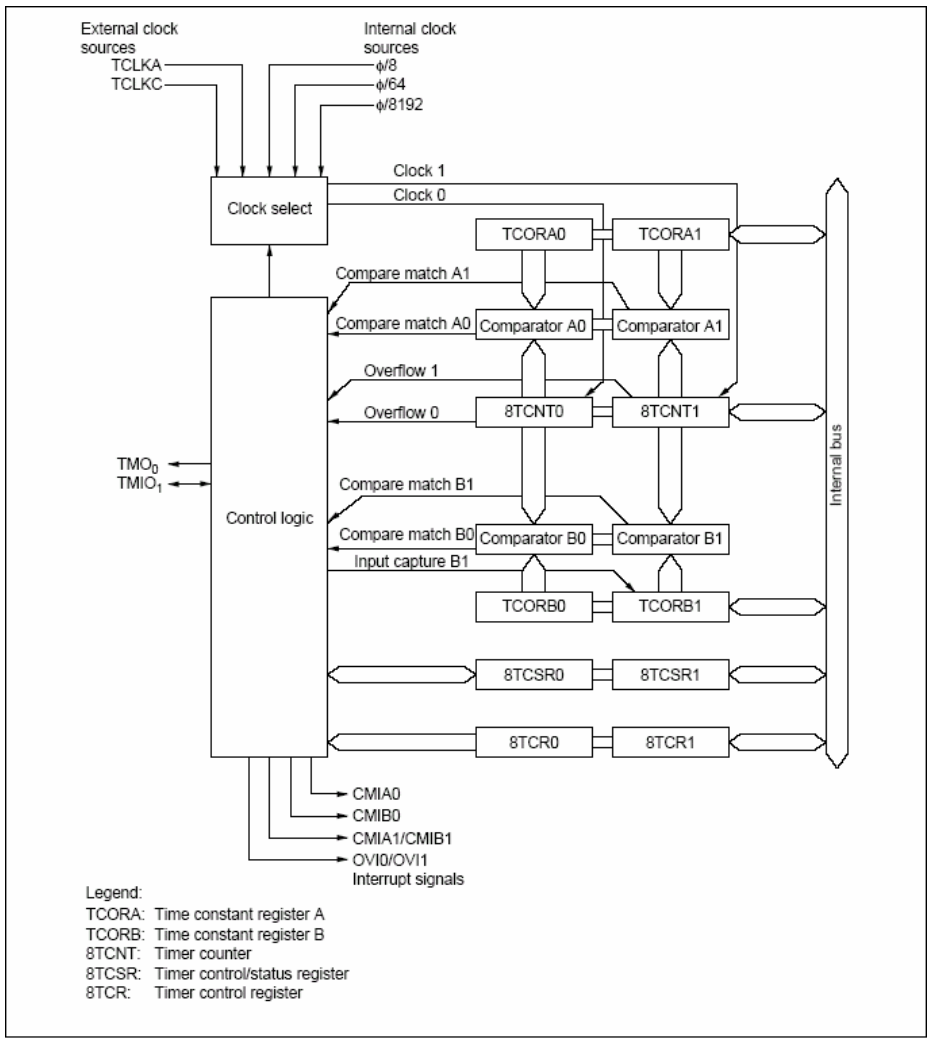


Figura 6. 1: diagrama de bloques para el temporizador de ocho bits.

Dependiendo del grupo y canal seleccionado así tenemos diferentes funciones para el módulo de temporización. En la figura 6.2 se detalla en una tabla las funciones y cada uno de los pines a usar.

Group	Channel	Name	Abbreviation	I/O	Function
0	0	Timer output	TMO_0	Output	Compare match output
		Timer clock input	TCLKC	Input	Counter external clock input
	1	Timer input/output	$TMIO_1$	I/O	Compare match output/input capture input
		Timer clock input	TCLKA	Input	Counter external clock input
1	2	Timer output	TMO_2	Output	Compare match output
		Timer clock input	TCLKD	Input	Counter external clock input
	3	Timer input/output	$TMIO_3$	I/O	Compare match output/input capture input
		Timer clock input	TCLKB	Input	Counter external clock input

Figura 6. 2: Tabla de funciones para e subsistema de temporización de 8 bits.

La ubicación física de los pines esta dada en la figura 6.3.

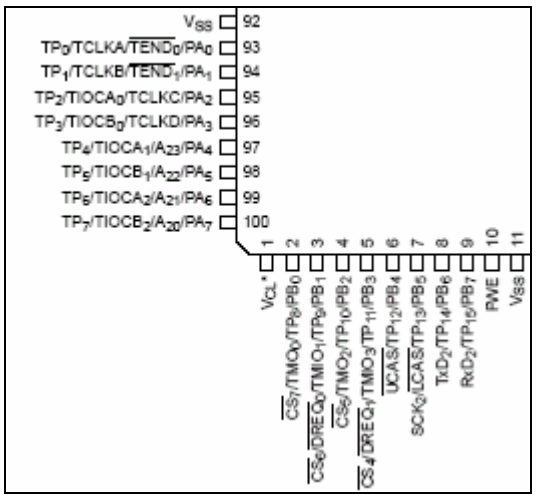


Figura 6. 3: Ubicación física de los pines de temporizadores de 8 bits.

6.2 Registros

Cada canal del módulo cuenta con 5 registros para manejar la operación y llevar el control de determinado canal. A continuación presentamos los registros y sus detalles para un solo canal (canal cero) ya que es similar para los otros tres canales, solo las direcciones de los registros son las que cambian. La figura 6.4 muestra las direcciones de los registros que contiene el canal 0.

0	H'FFF80	Timer control register 0	8TCR0	R/W	H'00
	H'FFF82	Timer control/status register 0	8TCSR0	R/(W)*2	H'00
	H'FFF84	Time constant register A0	TCORA0	R/W	H'FF
	H'FFF86	Time constant register B0	TCORB0	R/W	H'FF
	H'FFF88	Timer counter 0	8TCNT0	R/W	H'00

Figura 6. 4: Dirección de registro canal 0 temporizador de ocho bits.

6.2.1 Registro de conteo (8TCNTx):

Los registros 8TCNT son los registros de conteo que contienen los temporizadores, en general son habilitados para lectura y escritura e incrementan solo debido a los pulsos generados por las fuentes de reloj, ya sean externas o internas. Estas fuentes de reloj son seleccionadas por los bits de ajuste de reloj CKS2 a CKS0 en el registro de control de temporizador 8TCR.

Tomando en cuenta que los temporizadores de ocho bits pueden ser transformados en temporizadores de 16 bits, los registros de conteo, por ejemplo el 8TCNT0 y 8TCNT1, pueden ser concatenados para generar un registro de conteo de 16 bits si así es requerido. De igual manera puede hacerse para los canales 2 y 3.

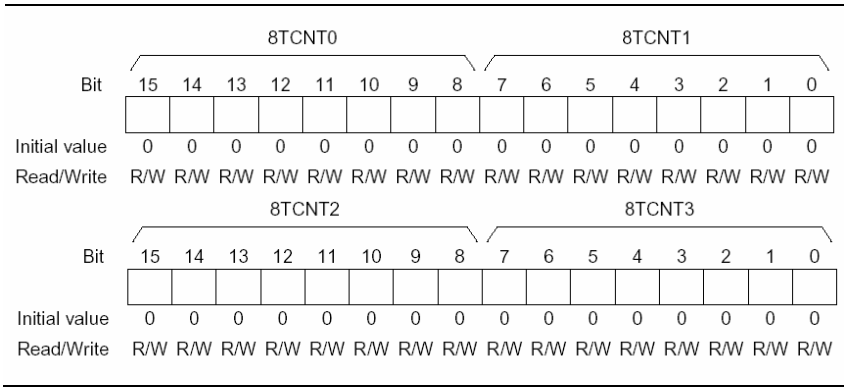


Figura 6. 5: Registro de conteo (8TCNTx).

Los 8TCNT pueden ser limpiados por una señal de entrada de comparación o por una comparación exitosa, el método de limpieza es seleccionado en el registro de control de temporizador de ocho bits (8TCR).

Cuando un registro de conteo alcanza el desbordamiento desde "FF" a "00", la bandera de desborde (OVF) se fija a uno en el registro de estado y control del temporizador.

> Nota: Cada registro de conteo es inicializado con "00" en una condición de reset.

6.2.2 Registro de constantes de tiempo (TCORxy):

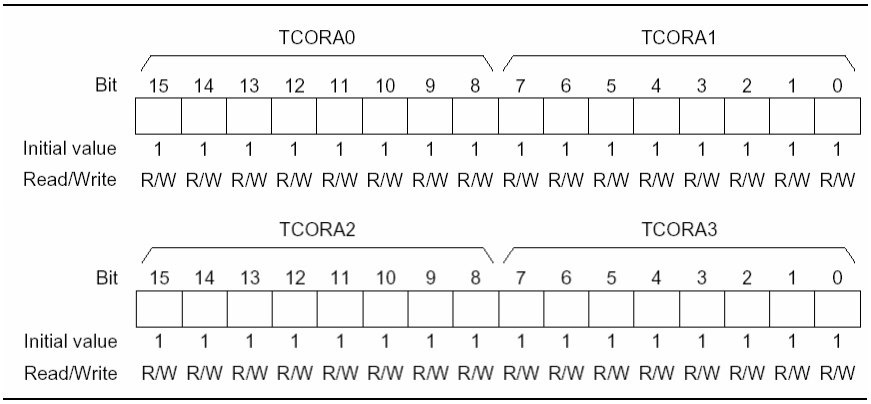


Figura 6. 6: Registro de constantes de tiempo (TCORxy).

TCORA0 a TCORA3 son registros de lectura y escritura; los cuales son comparados son los registros de conteo (8TCNT). Cuando una comparación es exitosa, la bandera (CMFA) es disparada y colocada a uno en el registro de estado del temporizador (8TCSR).

Para utilizar el temporizador de ocho bits como temporizador de 16 bits los registros, las constantes de temporización pueden ser concatenadas al igual que los registros.

El registro TCORB es utilizado al igual que el registro TCORA con la diferencia que este registro cumple con la característica especial de captura de eventos. Si el temporizador esta en captura de eventos y se detecta un evento, en el registro TCORB se guarda el dato que tenga el registro de conteo correspondiente.

> Nota: tomando en cuenta que después de un reset estos registros son inicializados con el valor de "FF".

6.2.3 Registro de control (8TCRx):

Es un registro que sirve para controlar la operación del módulo completo, es decir que controla los cuatro temporizadores que comprenden este módulo. La figura 6.5 muestra la configuración del registro de control de temporizadores de ocho bits.

Bit	7	6	5	4	3	2	1	0
	CMIEB	CMIEA	OVIE	CCLR1	CCLR0	CKS2	CKS1	CKS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figura 6. 7: Configuración de registro de control, temporizador de ocho bits.

Este es un registro de lectura y escritura que selecciona la fuente de reloj, fuente de limpieza y habilita o deshabilita las interrupciones en el módulo.

Bit 7: Compare Match Interrup Enable B (CMIEB): habilita o deshabilita la interrupción por comparación exitosa utilizando la comparación con el TCORB.

CMIEB	Descripción	
0	Interrupción deshabilitada	(Valor inicial)
1	Interrupción habilitada	

Figura 6. 8: CMIEB.

Bit 6: Compare Match Interrup Enable A (CMIEA): habilita o deshabilita la interrupción CMIA provocada cuando la bandera CMFA es fijada a uno en el registro de estado y control (8TCSR) del temporizador, por la comparación exitosa con TCORA.

CMIEA	Descripción	
0	Interrupción deshabilitada	(Valor inicial)
1	Interrupción habilitada	

Figura 6. 9: CMIEA.

Bit 5: Temporizador Overflow Interrup Enable (OVIE): Habilita o deshabilita la interrupción generada por un desbordamiento en el conteo, la interrupción es provocada por la bandera de OVF en el registro 8TCSR.

OVIE	Descripción	
0	Interrupción deshabilitada	(Valor inicial)
1	Interrupción habilitada	

Figura 6. 10: OVIE.

Bits 4 y 3: Counter Clear 1 and 0 (CCLR1, CCLR0): estos bits especifican la limpieza en el registro de conteo, es decir de que manera serán limpiados.

CCLR1	CCLR0	Descripción	
0	0	Limpieza deshabilitada	(Valor inicial)
0	1	Limpieza con A	
1	0	Limpieza con B	
1	1	Limpieza por entrada B	

Figura 6. 11:CCLR1 y CCLR0.

Bits 2 al 0: Clock Select 2 a 0 (CSK2 a CSK0): estos bits seleccionan la fuente de reloj que tendrá el módulo, esta puede ser interna o externa, tres fuentes de reloj interna pueden ser seleccionadas, todas dependientes del reloj del sistema (Φ).
Cuando se utiliza una fuente externa, tres tipos de conteos pueden ser seleccionados por flancos ascendentes, flancos descendentes, y ambos tipos de flanco (Toggle).

CSK2	CSK1	CSK0	Descripción
0	0	0	Entrada de reloj deshabilitada (Valor inicial)
0	0	1	Reloj interno Φ/8
0	1	0	Reloj interno Φ/64
0	0	1	Reloj interno Φ/8192
1	0	0	Canal 0 en modo de 16 bits
1	0	1	Reloj externo por flancos ascendentes
1	1	0	Reloj externo por flancos descendentes
1	1	1	Reloj externo por Toggle

Figura 6. 12: CSK2-CSK0.

6.2.4 Registro de Control y Estado (8TCSRx):

La figura 6.6 muestra la configuración del registro de control y estado del temporizador de 8 bits.

8TCSR0

Bit	7	6	5	4	3	2	1	0
	CMFB	CMFA	OVF	ADTE	OIS3	OIS2	OS1	OS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/W	R/W	R/W	R/W	R/W

8TCSR2

Bit	7	6	5	4	3	2	1	0
	CMFB	CMFA	OVF	—	OIS3	OIS2	OS1	OS0
Initial value	0	0	0	1	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	—	R/W	R/W	R/W	R/W

8TCSR1, 8TCSR3

Bit	7	6	5	4	3	2	1	0
	CMFB	CMFA	OVF	ICE	OIS3	OIS2	OS1	OS0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/W	R/W	R/W	R/W	R/W

Figura 6. 13: Registro de control y estado canal0.

Este es un registro de ocho bits que indica (dependiendo el arreglo de sus bits), la comparación exitosa, entrada por comparación, captura de eventos o desbordamiento.

Nota: Es inicializado con “00” por un reset.

Bit 7: Compare Match/Input Capture Flag B: Bandera de estado que indica la activación de TCORB por una comparación exitosa o un evento capturado.

CMFB	Descripción	
0	(Condición de limpia) Cuando CMFB=1, escribir un 0 para limpiar.	(Valor inicial)
1	8TCNT=TCORB 8TCNT es transferido a TCORB por una captura de eventos	

Figura 6. 14:CMFB.

Bit 6: Compare Match Flag A (CMFA): Bandera de estado que indica la activación de un TCORA por comparación exitosa.

CMFA	Descripción	
0	(Condición de limpia) Cuando CMFA=1, escribir un 0 para limpiar.	(Valor inicial)
1	8TCNT=TCORA	

Figura 6. 15:CMFA.

Bit 5: Temporizador Overflow Flag (OVF): bandera de estado que indica que el registro 8TCNT ha dado un desbordamiento de datos contados desde "FF" a "00".

OVF	Descripción	
0	(Condición de limpia) Cuando OVF=1, escribir un 0 para limpiar.	(Valor inicial)
1	8TCNT ha desbordado desde "FF" a "00"	

Figura 6. 16: OVF.

Bit 4: A/D Trigger Enable (ADTE): en combinación con el bit TRGE en el registro de control del convertidor de análogo a digital (ADCR), habilita o deshabilita el convertidor de análogo a digital por una comparación exitosa con A o un evento externo.

TRGE	ADTE	Descripción	
0	0	Conversión A/D activada por comparación con A o un evento externo en pin (ADTRG) deshabilitada	(Valor inicial)
0	1	Conversión A/D activada por comparación con A o un evento externo en pin (ADTRG) deshabilitada	
1	0	Conversión A/D activada por evento externo en pin (ADTRG) habilitada	
1	1	Conversión A/D activada por comparación con A habilitada	

Figura 6. 17: ADTE.

Bits 3 y 2: Output/Input Capture Edge Select B3 y B2 (OIS3, OIS2): Estos bits seleccionan, con la ayuda del ICE de los registros 8TCSR1/3 el nivel de salida de la comparación exitosa con B o el flanco de captura para un evento externo.

ICE Bit in 8TCSR1 (8TCSR3)	Bit 3 OIS3	Bit 2 OIS2	Description	
0	0	0	No change when compare match B occurs	(Initial value)
		1	0 is output when compare match B occurs	
	1	0	1 is output when compare match B occurs	
		1	Output is inverted when compare match B occurs (toggle output)	
1	0	0	TCORB input capture on rising edge	
		1	TCORB input capture on falling edge	
	1	0	TCORB input capture on both rising and falling edges	
		1		

Figura 6. 18:OIS3 y OIS2.

Bits 1 y 0: Output Select A1 y A0 (OS1, OS0): estos bits seleccionan la comparación exitosa con A.

OS1	OS0	Descripción	
0	0	No cambia cuando una comparación exitosa con A ocurre.	(Valor inicial)
0	1	0 es la salida cuando una comparación exitosa con A ocurre.	
1	0	1 es la salida cuando una comparación exitosa con A ocurre.	
1	1	La salida es invertida cuando una comparación exitosa en A ocurre (toggle).	

Figura 6. 19: OS0 u OS1.

Cuando la comparación exitosa con el registro es usada, el temporizador tiene como prioridad de salidas: salida toggle, salida 1 o salida 0. Por otro lado, si una comparación de A y B ocurre simultáneamente, la salida cambia de acuerdo con el nivel más alto de prioridad.

Los registros 8TNTC, TCORA, TCORB, 8TCR y 8TCSR son registros de 8 bits. Estos registros son conectados al CPU por un bus interno de 16 bit y pueden ser leídos y escritos como una palabra o como byte.

El convertidor análogo a digital puede solo ser disparado por el canal 0 cuando se compara con A. Si el bit ADTE esta puesto a uno cuando la bandera CMFA en el registro 8TCSR0 indica una comparación exitosa, la conversión de análogo a digital es solicitada. Si el bit TRGE en el registro ADCR (del convertidor) esta puesto a uno en este momento, el convertidor de análogo a digital comenzara a trabajar.

Si el bit ADTE en el registro 8TCSR0 esta puesto a uno, la conversión A/D que se disparara por una fuente externa en el pin ADTRG queda deshabilitada.

6.3 Operación

6.3.1 Conteo del temporizador:

Podemos seleccionar entre cuatro tipos de fuente de reloj para entregar los pulsos al contador. Tres de estas son internas y están relacionadas (fracciones) con el reloj del sistema: Φ/8, Φ/64 y Φ/8192. Estos deben de ser seleccionado con los bits CKS2-CKS0 del registro de control.

La cuarta fuente de reloj es una fuente de reloj externa la cual debe ser 1.5 ó 2.5 veces el valor del reloj del sistema para que un solo pulso (flanco ascendente o descendente) o ambos pulsos (toggle) sean detectados respectivamente.

6.3.2 Señal de Compare Match:

Cuando una comparación exitosa ocurre con TCORA o TCORB la salida será determinada por la configuración de los bits OIS3, OIS2, OS1, y OS0 del registro 8TCSR.

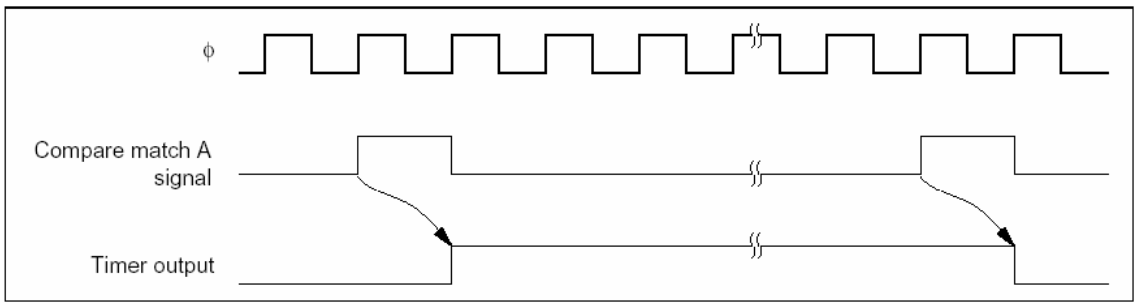


Figura 6. 20: Salida del temporizador activado por un Compare Match.

6.3.3 Limpieza del contador:

Dependiendo de la configuración del los bits CCLR1-CCLR0 del registro 8TCR determinamos si la limpieza del contador se efectuará con una comparación exitosa con TCORA o TCORB, o si se efectuará con la captura de la señal en TCORB (Input Capture).

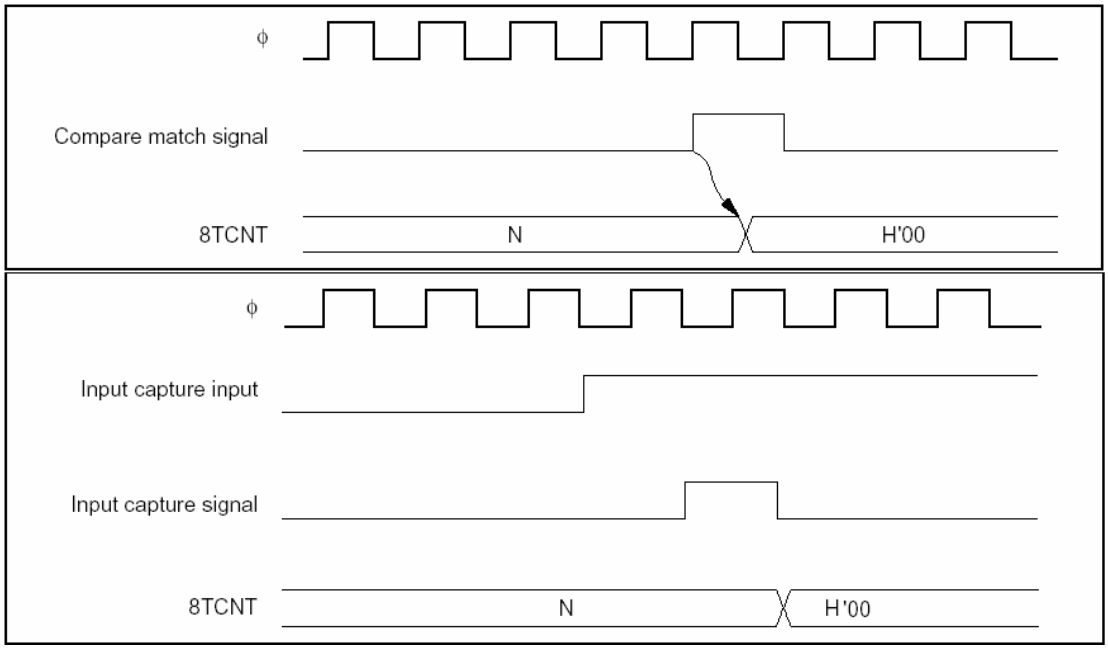


Figura 6. 21: Limpieza por Compare match o Input Capture respectivamente.

6.3.4 Señal de Input Capture:

La entrada de Input Capture puede ser un flanco ascendente, un flanco descendente o un Toggle. Cualquiera de estos tres según sea configurado podrá activar la señal de Input Capture de la cual dependerá el valor a ser capturado por el registro TCORB.

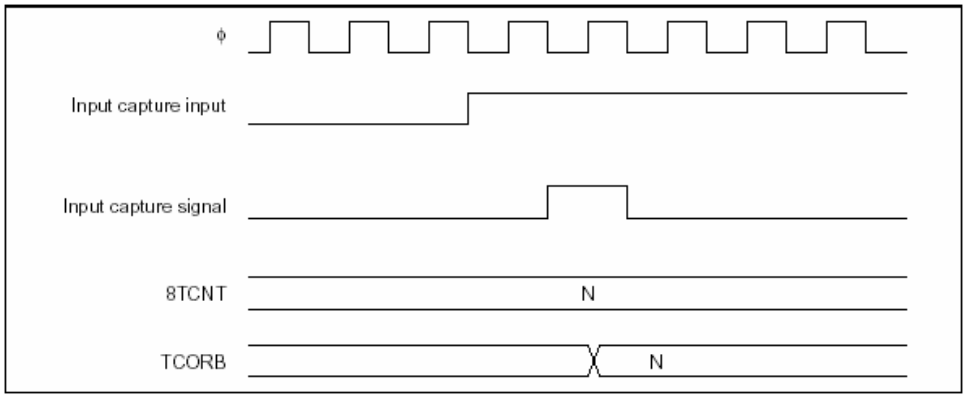


Figura 6. 22: Señal de Input Capture.

6.3.5 Banderas y tiempos de acción:

Las banderas CMFA y CMFB son activadas cuando tenemos una comparación exitosa con TCORA y TCORB. Ahora, la señal de Compare Match (C.M.) es dada en el momento que se va a incrementar el contador. Una vez teniendo esa señal de C.M. se activa la bandera.

En el caso de un Input Capture (I.C.), la señal se da en el momento que tenemos el I.C. y es entonces que se activa la bandera.

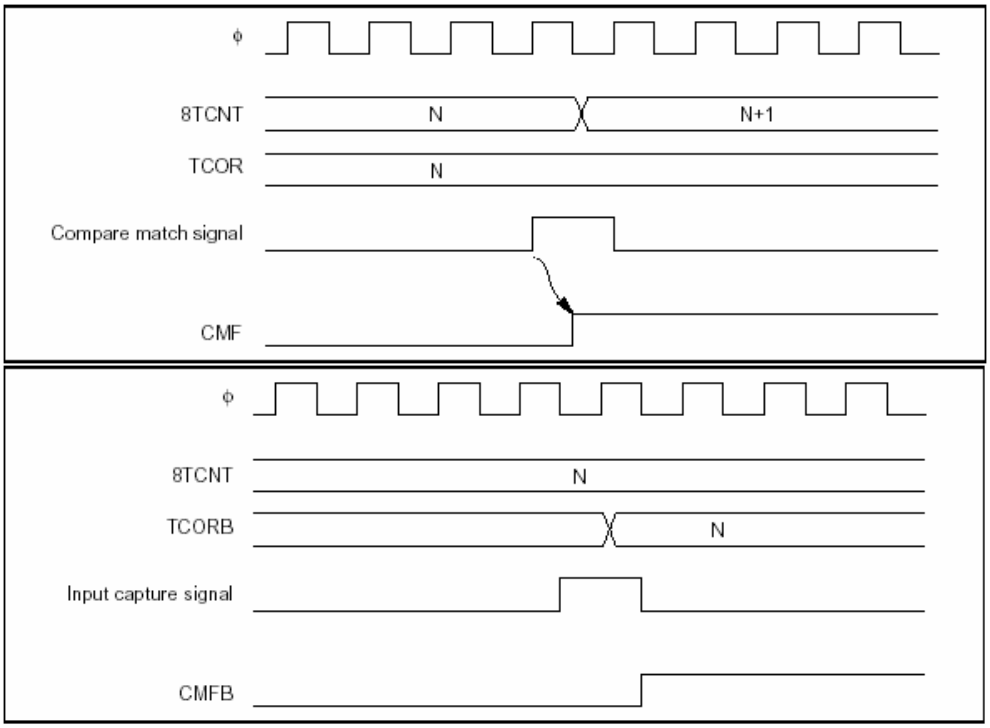


Figura 6. 23: Activación de banderas con el Compare match e Input Capture (respectivamente)

Por otro lado, también tenemos la señal de Overflow que activa la bandera de OVF de la siguiente manera:

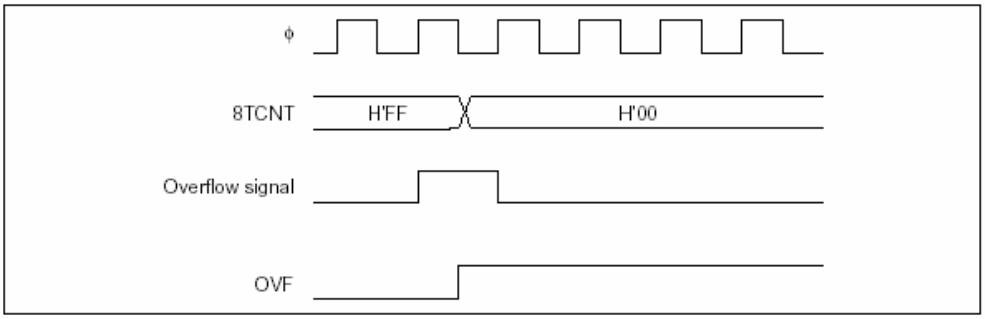


Figura 6. 24: Activación de la bandera de Overflow.

6.3.6 Operación en cascada:

Como se dijo anteriormente, los cuatro contadores de 8 bits pueden trabajar como dos contadores de 16 bits o inclusive trabajar en pareja (0-1 y 2-3) para que uno cuente los

eventos del otro. Esto se logra colocando los CKS2-CKS0 en una combinación de 100 en los registros 8TCRx correspondientes.

1. Contador de 16 bits:

Tomaremos el caso del canal 0 y 1:
Cuando tenemos la configuración anterior para los CKS2-CKS0 en el registro 8TCR0, los canales 0 y 1 van a funcionar como contador de 16 bits en donde el canal cero ocupa la parte alta.

- Compare Match:

Las banderas CMFA o CMFB serán disparadas cuando tengamos una comparación exitosa para los 16 bits o para los 8 bits menos significativos. Para las salidas tenemos que el comportamiento de los pines TMO_0 y $TMIO_1$ estarán configurados según los bits OIS3, OIS2, OIS1 y OIS0 del registro 8TCSR0 y 8TCSR1 y responderán a las condiciones de comparación de los 16 bits y de los 8 bits menos significativos respectivamente.

- Input Capture:

En este caso solo la bandera CMFB es activada en los registros 8TCSRx al momento de la captura, siempre y cuando el bit ICE este en uno. La captura del evento se hará por el pin $TMIO_1$ y dependerá de la configuración de los bits OIS2 y OIS3 de 8TCSR0.

- Limpieza de contador:

Los ocho bits menos significativos no pueden ser limpiados de manera independiente ya que la configuración de los bits CCLR1 y CCLR0 del canal uno es ignorada. La limpieza se realiza para los 16 bits si se han configurado los bits CCLR1 y CCLR0 del canal uno para una limpieza por evento.

- Bandera OVF:

Tenemos dos banderas disponibles, la del canal cero y la del uno. La primera se activará cuando el contador concatenado a 16 bits desborde y la segunda, solo cuando el conteo de los 8 bits menos significativos desborde.

> Nota: Estos mismos principios se mantienen para los canales 2 y 3, solo la dirección de los registros y los pines usados son diferentes.

2. Contador de eventos:

Si tenemos la configuración de 100 para los CKS2-CKS0 en el registro 8TCR1, el canal uno va a contar las comparaciones exitosas, con TCORA, del canal cero. Aquí, cana uno de los canales va a mantener la configuración de sus auxiliares (llamada a interrupción, salida de eventos…).

Nota:
- Este mismo principio se mantiene para los canales 2 y 3, solo la dirección d los registros y los pines usados son diferentes.
- Cuando el bit ICE de 8TCSR1 esta en uno, el registro de comparación TCORB para el canal cero no puede ser usado.
- No se debe de configurar los CKS2-CKS0 en el registro 8TCR0 y 8TCR1 en 100 al mismo momento ya que los contadores no operaran debido a que no habrá señal de reloj generada.

6.3.7 Input Capture:

El valor del contador puede ser transferido al registro correspondiere (o a los registros, dependiendo de la configuración adoptada) al momento de tener un evento en los pines TMIOx. La configuración del evento es también manipulada por el usuario.

- Operación en 8 bits:

Debemos de seleccionar la operación de Input capture con el bit ICE del canal a usar (uno o tres), configurar el tipo de evento que se va a detectar (con OIS3-OIS2) y seleccionar la señal de reloj que va a regir el contador para luego echar a andar el sistema.

Nota: Cuando se usa los canales uno o tres para Input Capture, no podemos utilizar los canales cero y dos para Compare Match.

- Operación en 16 bits:

Para cada pareja de contadores (0-1 o 2-3) debemos de configurar la función de Input Capture con el bit ICE en uno, luego definimos el tipo de evento a capturar en el registro del canal cero (con OIS3-OIS2 de 8TCSR0) y seleccionamos la señal de reloj que regirá el contador y posteriormente lo activamos.

6.4 Fuentes de interrupción.

El temporizador de 8 bits puede generar tres tipos de interrupción: comparación exitosa para A y B (CMIA y CMIB) y el desbordamiento (TOVI). La figura 6.12 muestra el nivel de prioridad que tienen las interrupciones del temporizador.

Interrupt Source	Description	Priority
CMIA	Interrupt by CMFA	High
CMIB	Interrupt by CMFB	↑
TOVI	Interrupt by OVF	Low

Figura 6. 25: Prioridad de interrupciones, temporizador de 8 bits.

El canal 0 y el canal 1 del temporizador comparten el mismo vector de interrupción para el caso de un desborde en el conteo. Este vector de interrupción es importante para el programador, el objetivo es conocer hacia donde bifurcara el PC al momento de que una interrupción en el temporizador de ocho bits sea ejecutada (para mayor información del vector de interrupciones de este temporizador vea sección 5.3.3 en el manual de Hardware H8/3069F).

A continuación presentamos las fuentes de interrupción para cada uno de los canales:

Channel	Interrupt Source	Description
0	CMIA0	TCORA0 compare match
	CMIB0	TCORB0 compare match/input capture
1	CMIA1/CMIB1	TCORA1 compare match, or TCORB1 compare match/input capture
0, 1	TOVI0/TOVI1	Counter 0 or counter 1 overflow
2	CMIA2	TCORA2 compare match
	CMIB2	TCORB2 compare match/input capture
3	CMIA3/CMIB3	TCORA3 compare match, or TCORB3 compare match/input capture
2, 3	TOVI2/TOVI3	Counter 2 or counter 3 overflow

Figura 6. 26: Fuentes de interrupción para cada canal.

Capitulo Siete: Convertidor Análogo-Digital [15]

[15] Para mayor información hacer referencia a la sección 15 del Manual de Hardware [B1].

7.1 Generalidades

El H8/3069F incluye 8 canales de conversión análoga-digital con resolución de 10 bits y con un método de aproximaciones sucesivas. Existen 3 fuentes posibles para iniciar la conversión: la primera de ellas es una activación por software, la siguiente es la activación por algún evento externo y por ultimo, la activación provocada por una comparación exitosa en el temporizador interno de 8 bits (Campare match, la cual será tratada en el capitulo del temporizador de 8 bits). Cualquiera de estos tres eventos iniciara la conversión en alguno de los modos siguientes:

- Modo Singular: cada unos de los canales es convertido separadamente y es convertido solo uno de ellos.
- Modo Scan: Conversión continua de uno hasta cuatro canales de manera sucesiva.

Al finalizar la conversión, el valor obtenido es transferido a cualquiera de los registros de datos (cuatro registros de 16 bits en total) en donde será almacenado hasta que el usuario haga uso de ese valor o hasta que la próxima conversión en el mismo canal tenga lugar.
Estas conversiones pueden ser bastante rápidas dependiendo de la frecuencia del sistema, pueden alcanzar un tiempo mínimo de 2.8 μs para una frecuencia de 25 MHz y además, al terminar la conversión, tenemos la posibilidad activar o no las interrupciones o el controlador de DMA (DMAC).

> Notas:
> - El voltaje de referencia para la conversión puede ser seleccionado externamente mediante el pin Vref.
> - En caso de no estar usando la conversión análoga-digital, esta puede ser apagada de manera a ahorrar energía.

La tabla siguiente muestra los pines a ser usados por el convertidor análogo-digital.

Pin Name	Abbrevi-ation	I/O	Function
Analog power supply pin	AV_{CC}	Input	Analog power supply
Analog ground pin	AV_{SS}	Input	Analog ground and reference voltage
Reference voltage pin	V_{REF}	Input	Analog reference voltage
Analog input pin 0	AN_0	Input	Group 0 analog inputs
Analog input pin 1	AN_1	Input	
Analog input pin 2	AN_2	Input	
Analog input pin 3	AN_3	Input	
Analog input pin 4	AN_4	Input	Group 1 analog inputs
Analog input pin 5	AN_5	Input	
Analog input pin 6	AN_6	Input	
Analog input pin 7	AN_7	Input	
A/D external trigger input pin	$\overline{ADTRG}$	Input	External trigger input for starting A/D conversion

Tabla 7. 1: Pines del convertidor A/D.

Los ocho canales de conversión están divididos en dos grupos: grupo cero (AN0 a AN3) y grupo uno (de AN4 a AN7), además tenemos los pines Vcc y Vss quienes son para la alimentación y a Vref el cual es para poner la referencia de voltaje para la conversión.

La figura 7.1 muestra el diagrama de bloques del convertidor A/D:

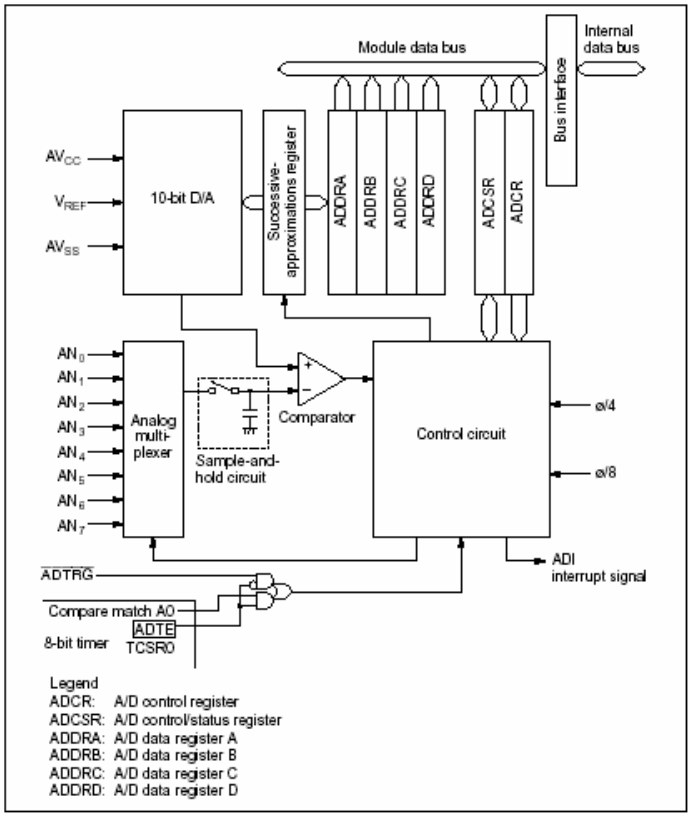


Figura 7. 1: Diagrama de broques del convertidor A/D

Como se mencionó anteriormente, tenemos 4 registros (ADDRA, ADDRB, ADDRC y ADDRD de 16 bits en donde se almacenará la información de la conversión, estos registros están separados en registros de 8 bits y están ubicados en las direcciones siguientes (tabla 7.2):

Address*1	Name	Abbreviation	R/W	Initial Value
H'FFFE0	A/D data register A H	ADDRAH	R	H'00
H'FFFE1	A/D data register A L	ADDRAL	R	H'00
H'FFFE2	A/D data register B H	ADDRBH	R	H'00
H'FFFE3	A/D data register B L	ADDRBL	R	H'00
H'FFFE4	A/D data register C H	ADDRCH	R	H'00
H'FFFE5	A/D data register C L	ADDRCL	R	H'00
H'FFFE6	A/D data register D H	ADDRDH	R	H'00
H'FFFE7	A/D data register D L	ADDRDL	R	H'00
H'FFFE8	A/D control/status register	ADCSR	R/(W)*2	H'00
H'FFFE9	A/D control register	ADCR	R/W	H'7E
Notes: *1 Lower 20 bits of the address in advanced mode. *2 Only 0 can be written in bit 7, to clear the flag.				

Tabla 7. 2: Direcciones de los registros de datos.

7.2 Configuración de Registros

7.2.1 Registro de Datos (ADDRn):

Para este convertidor análogo-digital tenemos cuatro registros de datos de 16 bits cada uno, solo de lectura, en donde se almacena el resultado de la conversión. Dicho resultado esta compuesto por 10 bits, los ocho bits mas significativos son almacenados en el byte alto del registro y los dos bits restantes son almacenados en el parte alta del byte menos significativo. Los bits del 0 al 5 del byte menos significativo son leídos siempre como cero.
Dicho de otra forma, tenemos un resultado de 10 bits almacenado con alineación a la izquierda en un registro de 16 bits, como se muestra en la figura 7.2.

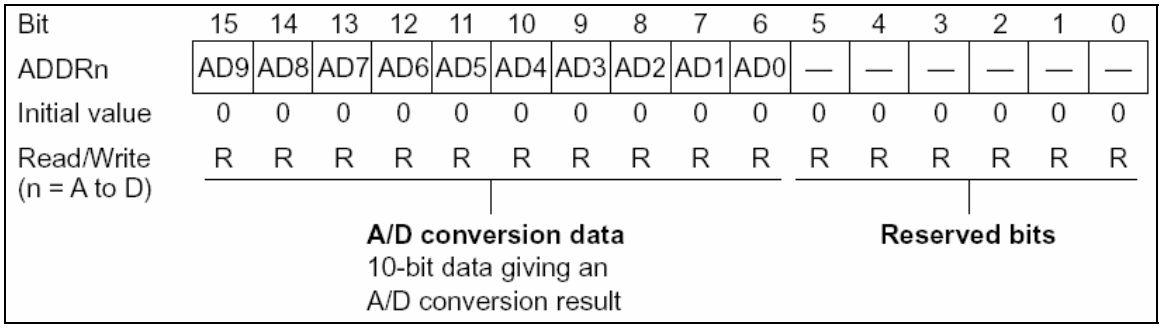


Figura 7. 2: Registros de Datos

Es necesario mencionar que el CPU lee y escribe en la parte alta del registro de datos directamente, pero para los dos bits menos significativos debe usar el registro temporal (TEMP). Debemos tener cuidado de no confundir la escritura y lectura de los registros por parte del CPU y la lectura (únicamente) por parte del usuario. La primera esta sujeta a

la capacidad de leer y escribir en el registro una vez se ha completado la conversión y la segunda a la capacidad de leer del registro el valor correspondiente a la conversión una vez esto le sea ordenado al CPU mediante las instrucciones respectivas.

Nota:
- Los cuatro registros ADDRA, ADDRB, ADDRC y ADDRD funcionan de la forma anterior.
- Los registros son inicializados con cero después de un Reset.

7.2.2 Registro de control de estado (ADCSR):

Este registro es un registro de escritura y lectura, inicializado siempre en cero después de un Reset, con el cual podemos seleccionar el modo de operación y controlar el convertidor análogo-digital. Dentro de las funciones de control tenemos la posibilidad de activar o desactivar la petición de interrupciones al finalizar el ciclo de conversión, iniciar o detener una conversión, visualizar si una conversión ha terminado o no, seleccionar la entrada de reloj y seleccionar el canal con el cual vamos a trabajar. A continuación se presenta el registro ADCSR:

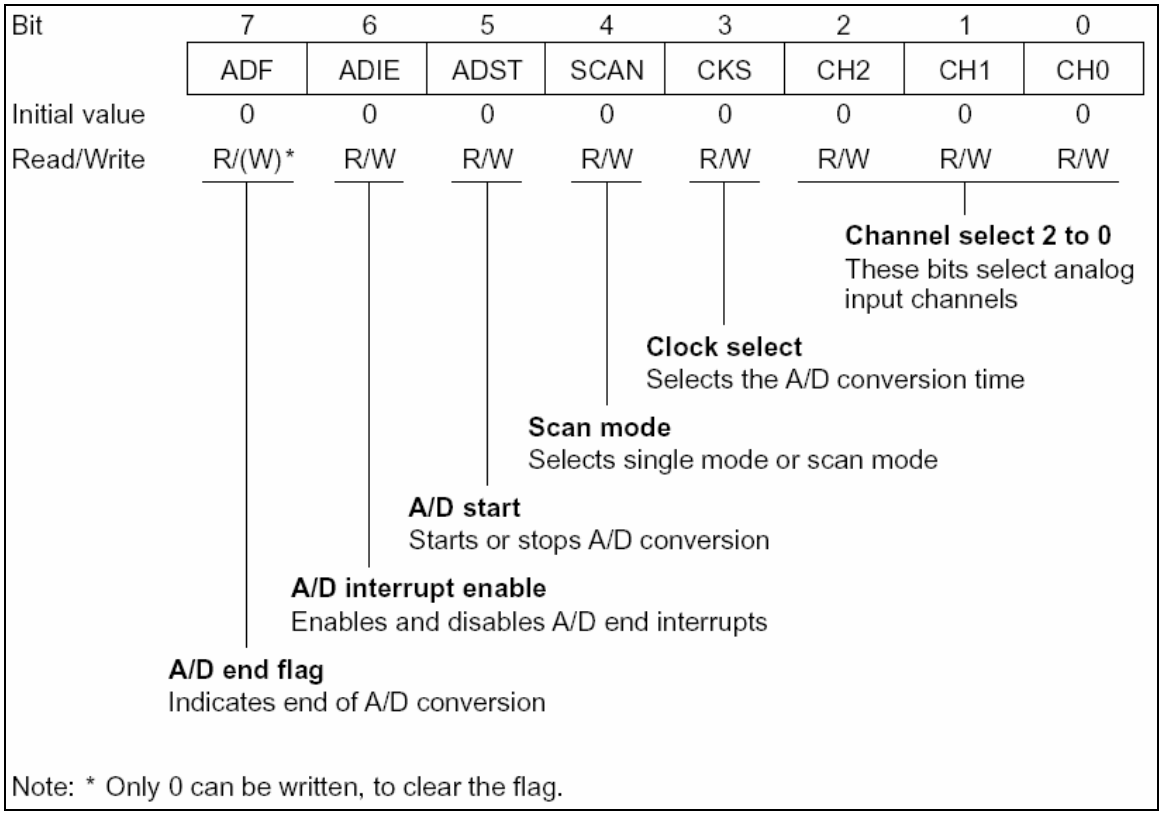


Figura 7. 3: A/D Control/Status Register (ADCSR)

Bit 7 (ADF): Un uno indica que la conversión ha terminado para el canal correspondiente (el modo Singular) o para todos los canales seleccionados (en modo Scan). Para poder limpiar esta bandera, debemos leerla y luego escribir cero.

Bit 6 (ADIE): Con uno habilitamos la petición de interrupciones (ADI) al finalizar la conversión, es decir al ponerse en uno el ADF.

Bit 5 (ADST): Este es puesto en uno para iniciar la conversión; puede ser puesto en uno por el usuario, por un evento externo o por una comparación exitosa (Compare Match) en el temporizador de 8 bits.
Si el convertidor esta en modo Singular entonces la conversión comenzará al poner uno en ADST y al finalizar, este bit regresará automáticamente a cero. Si estamos en modo Scan, la conversión iniciará de manera similar pero no se detendrá si hasta que el bit ADST sea puesto a cero por el usuario, Reset o Standby Mode.
Si ADST esta en cero, el convertidor esta detenido.

Bit 4 (SCAN): Este bit permite seleccionar entre el modo Singular (bit en cero) y el modo Scan (bit en uno).

Bit 3 (CKS): Aquí se configura el tiempo de conversión; si ponemos un uno, tendremos un tiempo máximo equivalente a 70 estados. De poner un cero, tendremos un tiempo máximo equivalente a 134 estados. Ver Tiempo de conversión.

Bits 2 a 0 (CH2 a CH0): Con estos tres bits seleccionamos el canal de entrada para el convertidor según la tabla siguiente:

Group Selection	Channel Selection		Description	
CH2	CH1	CH0	Single Mode	Scan Mode
0	0	0	AN_0 (Initial value)	AN_0
		1	AN_1	AN_0, AN_1
	1	0	AN_2	AN_0 to AN_2
		1	AN_3	AN_0 to AN_3
1	0	0	AN_4	AN_4
		1	AN_5	AN_4, AN_5
	1	0	AN_6	AN_4 to AN_6
		1	AN_7	AN_4 to AN_7

Tabla 7. 3: Selección de canal de entrada.

Nota: Para cambiar las últimas dos funciones es necesario pasar el ADST a cero, de lo contrario los cambios no tomarán efecto.

7.2.3 Registro de control (ADCR):

Este registro de lectura/escritura permite habilitar o deshabilitar el inicio de la conversión debido a un evento externo o debido a una señal de Compare Match dada por el temporizador de 8 bits.
Si el TRGE se encuentra en cero, la función de inicio de la conversión debido a los motivos antes mencionados estará desactivada (ver la tabla 7.4).

Bit 7 TRGE	Description	
0	Starting of A/D conversion by an external trigger or 8-bit timer compare match is disabled	(Initial value)
1	A/D conversion is started at the falling edge of the external trigger signal ($\overline{\text{ADTRG}}$) or by an 8-bit timer compare match	

Tabla 7. 4: TRGE del registro ADCR

Bits 6 al 1: Todos estos están reservados, y al momento de una lectura aparecerán como unos.

Bit 0: Este bit puede ser leído o escrito y solo debe de ser puesto en uno si el convertidor se activará a través del temporizador de 8 bits.

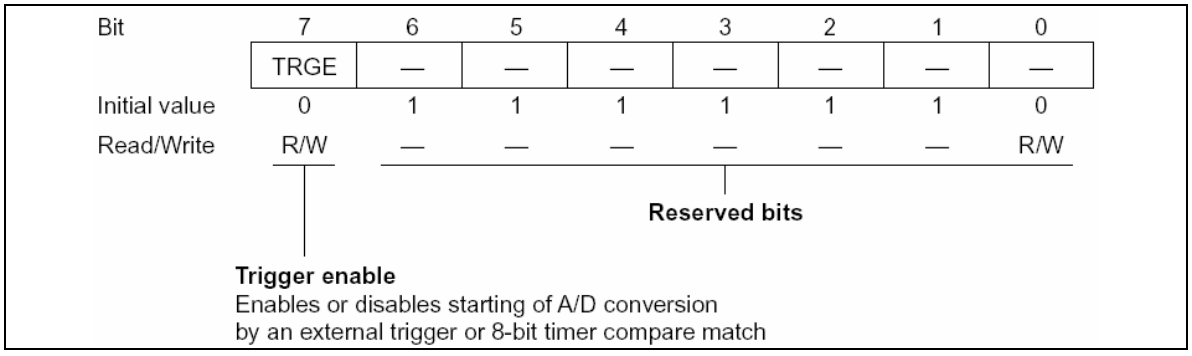


Figura 7. 4: Registro de control del convertidor.

7.3 Uso del registro TEMP

El convertidor análogo-digital tiene 4 registros de 16 bits en donde se almacenan los datos de la conversión. Ahora, para poder tener acceso a estos registros usamos un bus de 8 bits de modo que al momento de leer el dato resultante tendremos una lectura directa para el byte alto mientras que el byte bajo será almacenado en el registro temporal TEMP.
La secuencia es la siguiente: al momento de hacer la lectura, el byte alto es transferido el CPU y el bajo al registro TEMP, luego el valor de este último es transferido al CPU. La figura 7.7 ilustra un poco el flujo de los datos del registro al CPU.

Como dentro del registro de 16 bits el dato de 10 bits resultante de la conversión tiene alineación izquierda, no podemos hacer una simple lectura de el byte bajo de ADDRn ya que obtendríamos un valor errado y diferente del obtenido en la conversión.

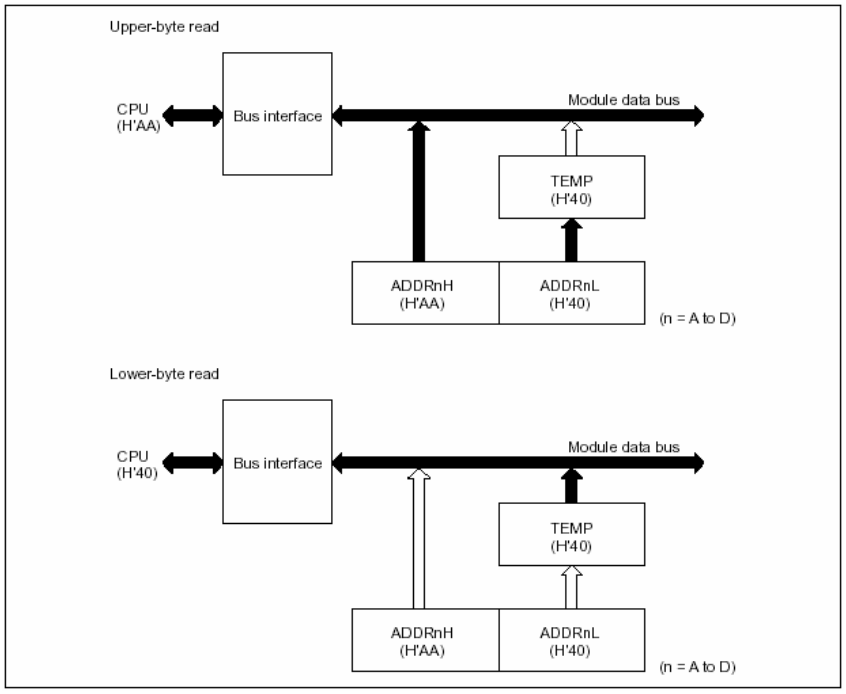


Figura 7. 5: Flujo de datos del convertidor al CPU.

7.4 Operación

7.4.1 Modo Singular (SCAN=0):

Este modo debe ser seleccionado cuando se desea hacer una sola conversión y de un solo canal a la vez. Previamente el canal debe ser seleccionado y el inicio de la conversión será dado por la presencia de un uno en ADST (puesto manualmente o de otra forma). Al terminar la conversión tendremos en el registro correspondiente el valor resultante y el ADST (el cual permanece en uno durante la conversión) regresará a cero, la bandera ADF se pondrá en uno y si esta activada la opción de interrupciones (ADIE=1), una interrupción será activada.
Para poder limpiar la bandera ADF debemos leer el registro ADCSR primero para luego escribir un cero en ADF.

En caso de querer detener la conversión, solo debemos poner un cero en ADST y de ser necesario cambiar el canal, debemos de detener primero la conversión para luego cambiar el canal con CH2-CH0.

7.4.2 Modo Scan (SCAN=1):

Este modo es ideal para monitorear uno o más canales casi al mismo tiempo.
Una vez este seleccionado el grupo de entradas análogas con el que se va a estar trabajando, ponemos el ADST en uno para iniciar la conversión. La operación comenzara con el primer canal del grupo y una vez la conversión para el primer canal termine, la conversión para el segundo empezará inmediatamente y los valores resultantes serán almacenadas en los registros correspondientes según la tabla siguiente:

Analog Input Channel		
Group 0	Group 1	A/D Data Register
AN_0	AN_4	ADDRA
AN_1	AN_5	ADDRB
AN_2	AN_6	ADDRC
AN_3	AN_7	ADDRD

Tabla 7. 5: Registros según grupos.

Una vez la conversión de todos los registros este completa, se activará la bandera ADF y posiblemente la interrupción correspondiente (de estar ADIE en uno). Debemos tener cuidado de leer los datos antes que sean sobre escritos por los nuevos valores ya que el proceso volverá a comenzar mientras tengamos ADST en uno. Para detener el proceso completo deberemos poner este último en cero.
De igual forma que para el modo anterior, para cambiar los canales (el grupo en este caso) en convertidor deberá estar detenido.

7.4.3 Muestreo y tiempo de conversión:

El convertidor análogo-digital posee un circuito de muestreo y retención para capturar la señal análoga a ser procesada. Este circuito introduce un retardo t_{spl} para el tiempo total de conversión t_{conv}, como se muestra en la figura 7.6.

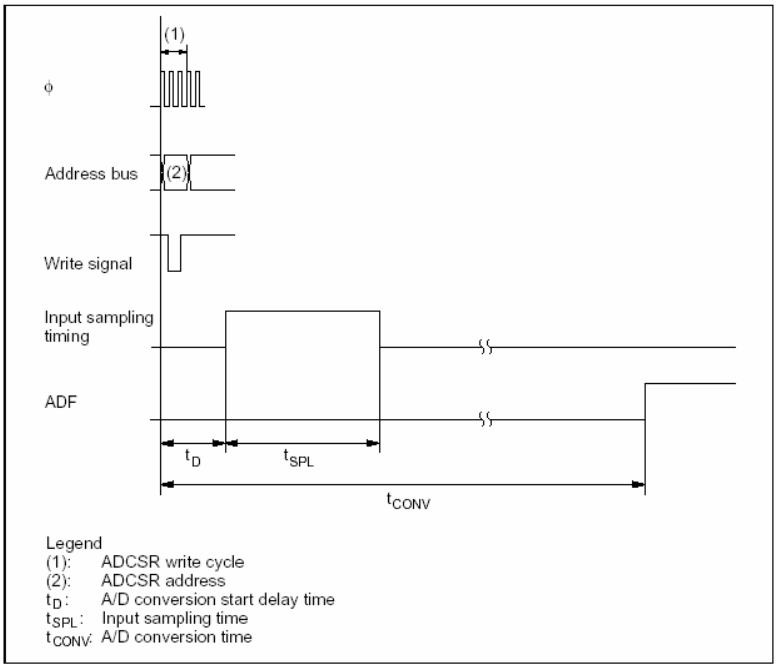


Figura 7. 6: Tiempo de conversión.

Podemos ver también que existe un tiempo de retraso t_D para que el muestreo inicie, después de haber puesto ADST en uno. Este tiempo depende del tiempo que toma tener acceso al registro ADCSR para la escritura del uno en ADST.

Ahora, los tiempos de conversión que se presentan a continuación se aplican únicamente para el modo de operación Singular. Para el modo de operación Scan tenemos los mismo tiempos solo para el primer canal a convertir, luego los tiempo son fijados a 128 estados y 66 estados para una configuración de CKS=0 y CKS=1 respectivamente.

		CKS = 0			CKS = 1		
	Symbol	Min	Typ	Max	Min	Typ	Max
Synchronization delay	t_D	6	—	9	4	—	5
Input sampling time	t_{SPL}	—	31	—	—	15	—
A/D conversion time	t_{CONV}	131	—	134	69	—	70

Note: Values in the table are numbers of states.

Tabla 7. 6: Tiempos de conversión.

7.4.4 Disparo externo de inicio:

La conversión puede ser iniciada por un evento externo detectado en el pin ADTRG cuando la señal pasa de alto a bajo. A ser detectado este evento, el bit ADST de ADCSR se pondrá en uno iniciando así la conversión. Para que esto ocurra debemos de tener configurado el bit TRGE del registro ADCR en uno y el ADTE del registro 8TCSR0 del temporizador de 8 bits en cero.

La anterior era una de las dos posibilidades de disparo externo usando el temporizador de 8 bits. La tabla siguiente muestra las diferentes combinaciones que se pueden hacer para las diferentes funciones de inicio de conversión:

TRGE*	Bit 4 ADTE	Description
0	0	A/D converter start requests by compare match A or external trigger pin ($\overline{\text{ADTRG}}$) input are disabled (Initial value)
	1	A/D converter start requests by compare match A or external trigger pin ($\overline{\text{ADTRG}}$) input are disabled
1	0	A/D converter start requests by external trigger pin ($\overline{\text{ADTRG}}$) input are enabled, and A/D converter start requests by compare match A are disabled
	1	A/D converter start requests by compare match A are enabled, and A/D converter start requests by external trigger pin ($\overline{\text{ADTRG}}$) input are disabled
Note: * TRGE is bit 7 of the A/D control register (ADCR).		

Tabla 7. 7: Combinaciones para el disparo externo.

7.5 Interrupciones:

Al finalizar una conversión se puede generar una llamada a interrupción siempre y cuando dicha petición no esta enmascarada, es decir que el bit ADIE del registro ADCSR debe estar en uno. De ser ese el caso, el cambio de estado de ADI a uno (fin de conversión) llamará a la interrupción indicada por el vector de interrupciones o podría activar el DMAC (DMA Controller). Si el segundo caso se cumple la interrupción del vector de interrupciones no se tomará en cuenta.

Interrupt Source	Origin	Vector Number	Vector Address*1		IPR
			Advanced Mode	Normal Mode*2	
ADI (A/D end)	A/D	23	H'005C to H'005F	H'002E to H'002F	

Tabla 7. 8: Vector de interrupción.

Capitulo Ocho: Convertidor Digital-Análogo [16]

[16] Para mayor información hacer referencia a la sección 16 del Manual de Hardware [B1].

8.1 Generalidades

La conversión D/A esta diseñada para generar como salida valores análogos de voltaje y es mucho mas fácil de comprender y simple que la conversión A/D. El H8/3069F incluye un convertidor D/A con dos canales disponibles con resolución de 8 bits para los datos a ser convertidos.

En una aplicación de microcontroladores un actuador podría utilizarse para controlar valores físicos como rotación, velocidad y un porcentaje de generación de calor a través de voltajes análogos. Estos voltajes análogos (entregados por el DAC) son salidas que se habilitan después de ejecutarse una conversión de bits que provienen de números digitales. Esta conversión de digital a análogo es provocada por un convertidor D/A integrado

8.1.1 Características.

- Resolución de ocho bits.
- Dos canales de salida.
- Tiempo de conversión: 10 µs Máximo (con carga capacitiva de 20-pF).
- Salida de voltaje: 0V a Vref (0 – 5V).
- La salida de voltaje puede ser mantenida aún en modo Standby.

8.1.2 Diagrama de bloques.

La figura 8.1 muestra el diagrama de bloques del convertidor D/A.

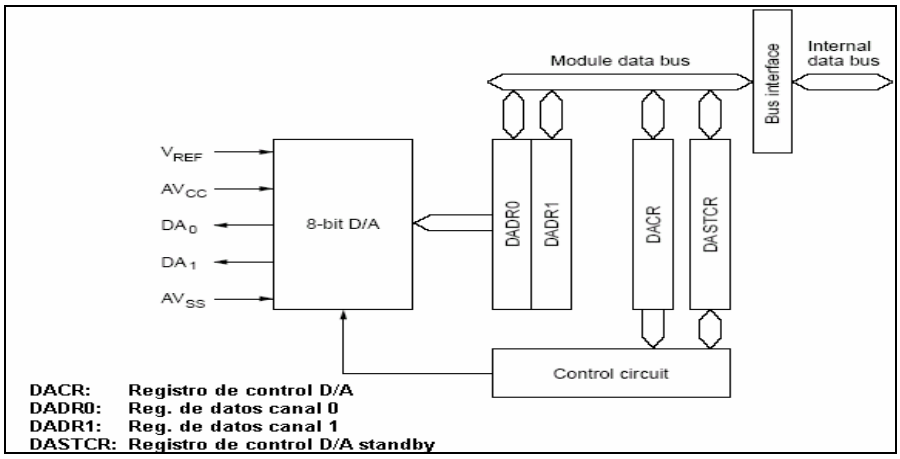


Figura 8. 1: Diagrama en bloques del módulo D/A.

El D/A tiene 5 pines externos los cuales aparecen en la figura 8.1. DA0 y DA1 están diseñados como salidas. VREF es el voltaje de referencia para la conversión, AVcc es el voltaje de alimentación del módulo y el pin Avss es el voltaje de tierra, estos dos últimos están separados de los otros pines de alimentación y tierra (Vcc y Vss). Debemos

mencionar que el convertidor no funcionará si no esta conectada la alimentación respectiva del módulo.

La figura 8.2 muestra la función de cada pin en el módulo D/A:

Pin Name	Abbreviation	I/O	Function
Analog power supply pin	AV_{CC}	Input	Analog power supply and reference voltage
Analog ground pin	AV_{SS}	Input	Analog ground and reference voltage
Analog output pin 0	DA_0	Output	Analog output, channel 0
Analog output pin 1	DA_1	Output	Analog output, channel 1
Reference voltage input pin	V_{REF}	Input	Analog reference voltage

Figura 8. 2: Pines para el convertidor DA

La ubicación física de los pines la muestra la figura 8.3:

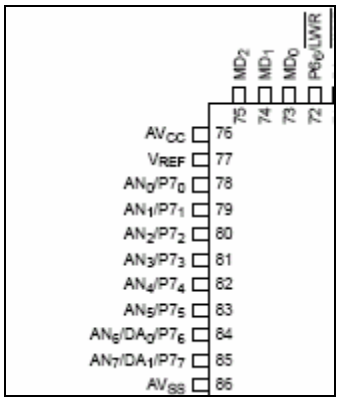


Figura 8. 3: Ubicación física de los pines.

8.2 Registros

Tenemos cuatro registros para poder controlar los dos canales de conversión D/A que nos brinda el H8/3069F. La figura 8.4 muestra el nombre y la dirección cada uno de los registros.

Address*	Name	Abbreviation	R/W	Initial Value
H'FFF9C	D/A data register 0	DADR0	R/W	H'00
H'FFF9D	D/A data register 1	DADR1	R/W	H'00
H'FFF9E	D/A control register	DACR	R/W	H'1F
H'EE01A	D/A standby control register	DASTCR	R/W	H'FE

Note: * Lower 20 bits of the address in advanced mode.

Figura 8. 4: Registros del módulo digita-análogo

8.2.1 Registro de datos DADRx (canales 0 y 1).

Los registros de datos (DADR0 y DADR1) son registros de lectura y escritura de ocho bits, estos almacenan el dato a convertir. Cuando la salida análoga es habilitada, el registro de datos del D/A es constantemente convertido y una señal análoga es enviada al pin de salida.

La figura 8.5 muestra el registro de datos del módulo D/A. Debemos tener en cuenta que ambos registros son propios para cada canal.

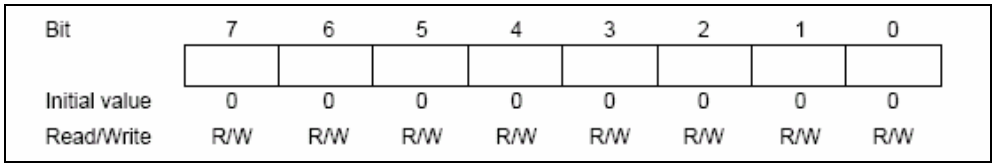


Figura 8. 5: Registro de datos canal 0 y 1, Módulo D/A.

> Nota: El registro de datos del D/A es inicializado por un H'00 después de un reset o de un Standby.

8.2.2 Registro de control (DACR).

El DACR es un registro de lectura y escritura de ocho bits que controla la operación del módulo D/A, este registro es inicializado con el valor de H'1F por un reset o por un Standby.

La figura 8.6 muestra el registro de control del módulo D/A.

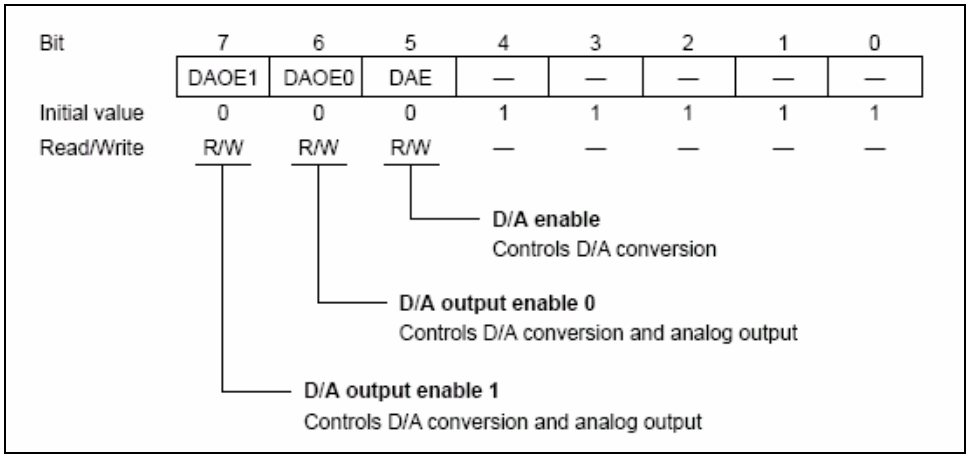


Figura 8. 6: Registro de control, módulo D/A.

Bit 7 D/A Output Enable 1 (DAOE1): Controla la conversión de la salida análoga del canal uno del módulo D/A.

DAOE1	Descripción
0	Salida DA1 esta deshabilitada.
1	Salida DA1 esta habilitada.

Bit 6 D/A Output Enable 0 (DAOE0): Controla la conversión de la salida análoga del canal cero del módulo D/A.

DAOE0	Descripción
0	Salida DA0 esta deshabilitada.
1	Salida DA0 esta habilitada.

Bit 5 D/A Enable (DAE): Controla la conversión D/A, conjuntamente con los bits DAOE0 y DAOE1. Cuando el bit DAE esta puesto a cero, la conversión análoga en los canales 0 y 1 es controlada independientemente. Cuando el bit DAE es puesto a uno, la conversión análoga es controlada de manera grupal, ahora, debemos decir que el resultado de salida en la conversión es siempre controlado independientemente por DAOE0 y DAOE1.

DAOE1	DAOE0	DAE	Descripción
0	0	-	Conversión D/A deshabilitada en ambos canales.
0	1	0	Conversión D/A habilitada en el canal 0.
1	0	0	Conversión D/A habilitada en el canal 1.
-	-	1	Conversión D/A habilitada en los dos canales.
1	1	-	Conversión habilitada en canales 0 y 1.

Cuando el bit DAE esta colocado a uno, aunque los bits DAOE0 y DAOE1 en DACR y el bit ADST en el registro ADCSR están en cero, la misma corriente es drenada desde la fuente de poder análoga durante la conversión de digital análogo así como también a la conversión de análogo a digital.

Bits 4 al 0 (Reserved): Estos bits no pueden ser modificados y siempre son leídos como uno.

8.2.3 Registro de control Standby del módulo D/A.

El registro DASTCR es un registro de lectura y escritura de ocho bits, que habilita o deshabilita la salida del módulo D/A si el microcontrolador se encuentra en modo Standby.

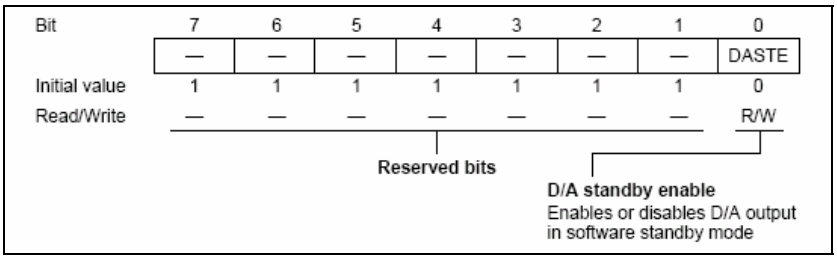


Figura 8. 7: Registro de control Standby.

Nota: El registro DASTCR es inicializado con el byte H'FE por un reset o por un modo Standby.

Bits 7 a 1 (Reserved): Estos bits no pueden ser modificados y siempre se leen como uno.

Bit 0 D/A Standby Enable (DASTE): habilita o deshabilita la salida del módulo D/A en modo Standby.

DASTE	Descripción	
0	Salida D/A en modo Standby deshabilitada.	(Valor inicial)
1	Salida D/A en modo Standby habilitada.	

Nota: Cuando el bit DASTE del registro DASTCR esta a uno, el registro DACR no es inicializado en el modo Standby.

8.3 Operación.

El módulo de conversión D/A tiene dos convertidores D/A construidos, que pueden ejecutar la conversión independientemente. Cada una de ellas es desarrollada constantemente mientras se habilite en DACR.

Si los valores de los registros DADR0 o DADR1 son modificados, una nueva conversión comienza inmediatamente. La conversión termina en una salida analógica cuando los bits DAOE0 y DAOE1 tienen un valor, dicha operación tendrán un tiempo máximo de conversión de 10 microsegundos,

Las salidas análogas del módulo tienen la siguiente ecuación:

Vout = (DADR/256) x Vref

El resultado de la conversión no cambia hasta que el valor en DADRx no sea modificado o el bit DAOEx sea puesto a cero. En este ultimo caso, el pin DAx se convierte en un pin de entrada.

Capitulo Nueve: SCI [17]

[17] Para mayor información hacer referencia a la sección 13 del Manual de Hardware [B1].

9.1 Generalidades

Existen tres canales independientes de comunicación serial (SCI), cada uno de ellos con las mismas funciones y con la posibilidad de ser apagados para conservar la energía. La SCI puede ser usada tanto para la comunicación asincrónica como para la comunicación sincrónica y para la comunicación entre dos procesadores. Además, también posee una interfase para la comunicación serial con una tarjeta inteligente[18] (Smart Card), dicha interfase va conforme el estándar ISO/IEC 7816-3 (Identification Card).

La capacidad y características de comunicación dependen del modo con el cual se este trabajando, pero debemos mencionar que ambos modos poseen características que permiten la comunicación Full-duplex, la posibilidad de transferir el bit mas significativo primero o el menos significativo primero, invertir los niveles lógicos al momento de la transmisión, selección de la señal de reloj (interna o externa) y cuatro tipos de interrupciones dos de las cuales pueden activar el DMAC (controlador DMA). Los modos de comunicación son los siguientes:

- Modo Asincrónico:

En este modo la comunicación serial de datos es sincronizada un canal a la vez (en el flanco negativo del bit de Start). El SCI puede comunicarse con Receptores-Transmisores Universales Asincrónicos mejor conocidos como UART, con adaptadores de interfase de comunicación asincrónica (ACIA) o con cualquier tipo chip que utilice el estándar de comunicación asincrónica. Además, puede comunicarse con dos o más microprocesadores usando la función de comunicación de microprocesadores. El formato es variado:

Largo de la trama:	7 o 8 bits
Bits de paro:	1 o 2 bits
Paridad:	Par/impar/ninguna
Bit de Microprocesador:	1 o 0
Método de detección de error:	Paridad, sobre escritura o Framing

- Modo Sincrónico:

Para este modo tenemos que la comunicación esta sincronizada con una señal de reloj y se puede dar con cualquier otro chip que tenga las funciones de comunicación sincrónica. A diferencia del modo anterior, solo tenemos un formato para la comunicación de datos:

Largo de la trama:	8 bits
Método de detección de error:	sobre escritura

A continuación presentamos el diagrama de bloque de la interfase serial (figura 7.1) y los pines asignados para cada uno de los canales (figura 7.2):

[18] Esta interfase no será tratada.

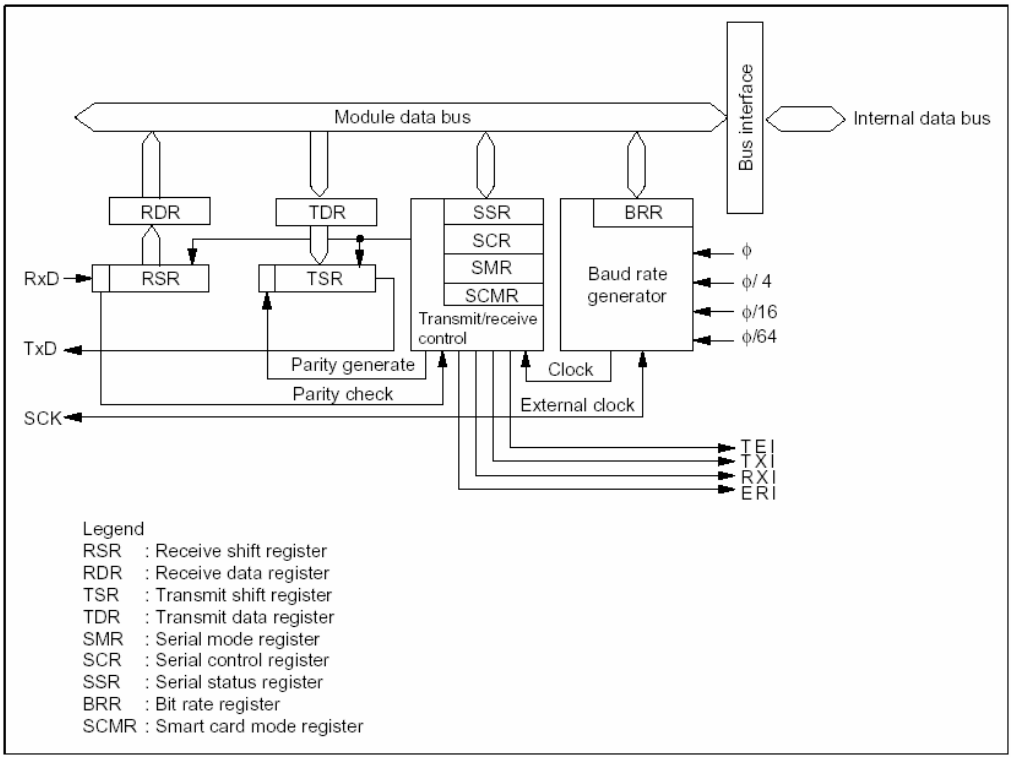


Figura 9. 1: Diagrama de bloques del SCI.

Channel	Name	Abbreviation	I/O	Function
0	Serial clock pin	SCK_0	Input/output	SCI_0 clock input/output
	Receive data pin	RxD_0	Input	SCI_0 receive data input
	Transmit data pin	TxD_0	Output	SCI_0 transmit data output
1	Serial clock pin	SCK_1	Input/output	SCI_1 clock input/output
	Receive data pin	RxD_1	Input	SCI_1 receive data input
	Transmit data pin	TxD_1	Output	SCI_1 transmit data output
2	Serial clock pin	SCK_2	Input/output	SCI_2 clock input/output
	Receive data pin	RxD_2	Input	SCI_2 receive data input
	Transmit data pin	TxD_2	Output	SCI_2 transmit data output

Figura 9. 2: Pines para el SCI.

9.2 Registros

Mediante los diferentes registros podemos seleccionar entre una comunicación sincrónica o asincrónica, especificar el formatos de los datos y velocidad de transferencia, controlar la sección de transmisión y recepción y hacer los cambio entre la interfase de comunicación serial y de Smart Card.
Todos los registros y sus direcciones están listados en la tabla siguiente:

Channel	Address*1	Name	Abbreviation	R/W	Initial Value
0	H'FFFB0	Serial mode register	SMR	R/W	H'00
	H'FFFB1	Bit rate register	BRR	R/W	H'FF
	H'FFFB2	Serial control register	SCR	R/W	H'00
	H'FFFB3	Transmit data register	TDR	R/W	H'FF
	H'FFFB4	Serial status register	SSR	R/(W)*2	H'84
	H'FFFB5	Receive data register	RDR	R	H'00
	H'FFFB6	Smart card mode register	SCMR	R/W	H'F2
1	H'FFFB8	Serial mode register	SMR	R/W	H'00
	H'FFFB9	Bit rate register	BRR	R/W	H'FF
	H'FFFBA	Serial control register	SCR	R/W	H'00
	H'FFFBB	Transmit data register	TDR	R/W	H'FF
	H'FFFBC	Serial status register	SSR	R/(W)*2	H'84
	H'FFFBD	Receive data register	RDR	R	H'00
	H'FFFBE	Smart card mode register	SCMR	R/W	H'F2
2	H'FFFC0	Serial mode register	SMR	R/W	H'00
	H'FFFC1	Bit rate register	BRR	R/W	H'FF
	H'FFFC2	Serial control register	SCR	R/W	H'00
	H'FFFC3	Transmit data register	TDR	R/W	H'FF
	H'FFFC4	Serial status register	SSR	R/(W)*2	H'84
	H'FFFC5	Receive data register	RDR	R	H'00
	H'FFFC6	Smart card mode register	SCMR	R/W	H'F2

Notes: *1 Indicates the lower 20 bits of the address in advanced mode.
*2 Only 0 can be written, to clear flags.

Tabla 9. 1: Registros del SCI

9.2.1 Receive Shift Register (RSR) y Receive Data Register (RDR):

La recepción de los datos se efectúa a través de dos registros (actuando como buffer) de manera a poder tener una recepción continua. El primero de ellos, el RSR, toma los datos de la entrada en el orden en que se reciben (el bit menos significativo primero) y los convierte a datos en paralelo. Cuando un byte entero ha sido recibido, este dato es transferido del RSR al RDR para su almacenamiento, y es hasta entonces que se completa

la operación de recepción. Después de transferir el dato al RDR, el RSR puede empezar a recibir el siguiente y tener así una continuidad de recepción mientras el dato anterior es procesado.
El contenido de ambos registros no puede ser modificado por el CPU y solamente el RDR puede ser leído de manera a extraer el dato recibido.

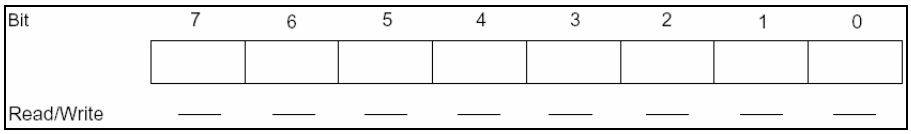


Figura 9.3: Receive Shift Register (RSR)

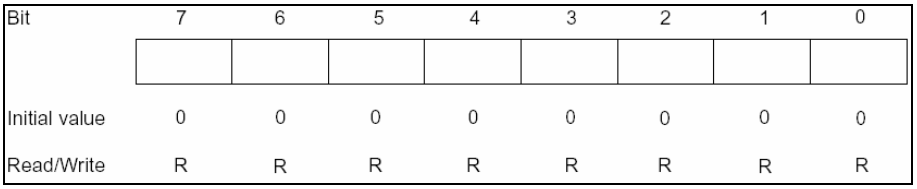


Figura 9.4: Receive Data Register (RDR)

Nota: Después de un Reset o de un Standby el RDR es inicializado en H´0000.

9.2.2 Transmit Shift Register (TSR) y Transmit Data Register (TDR):

El principio es similar al de los registros anteriores. El TSR solo es un registro de intercambio, el dato que recibe del TDR es enviado bit a bit por el pin asignado comenzando por el bit menos significativo. Cuando la transmisión del byte ha concluido y por consiguiente el TSR este vacío, el SCI automáticamente cargará el nuevo valor a ser enviado. Durante el momento de la transmisión es posible poner en el TDR el siguiente valor a ser procesado de modo a tener una transmisión continua.
El dato contenido en el TDR va a ser cargado en el TSR de manera continua siempre y cuando la bandera TDRE del registro SSR este puesto en uno.

El contenido de TDR si puede ser leído y escrito en cualquier momento mientras que el TSR no puede ser modificado por el CPU directamente.

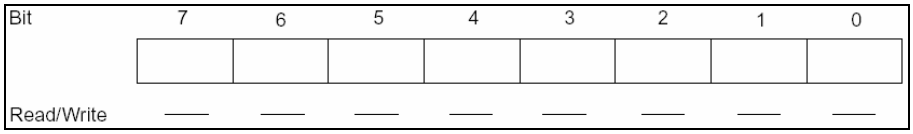


Figura 9.5: Transmit Shift Register (TSR).

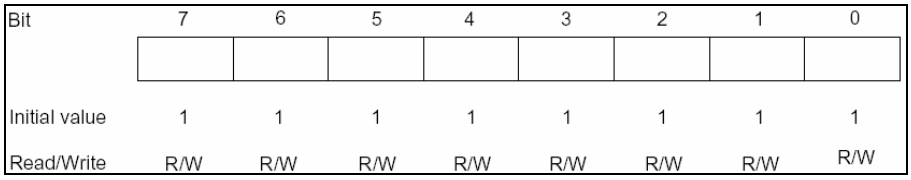


Figura 9.6: Transmit Data Register (TDR).

9.2.3 Serial Mode Register (SMR):

Este registro de lectura/escritura nos permite seleccionar el formato con el cual trabajará el SCI y la fuente de reloj para el generador de Baudios.

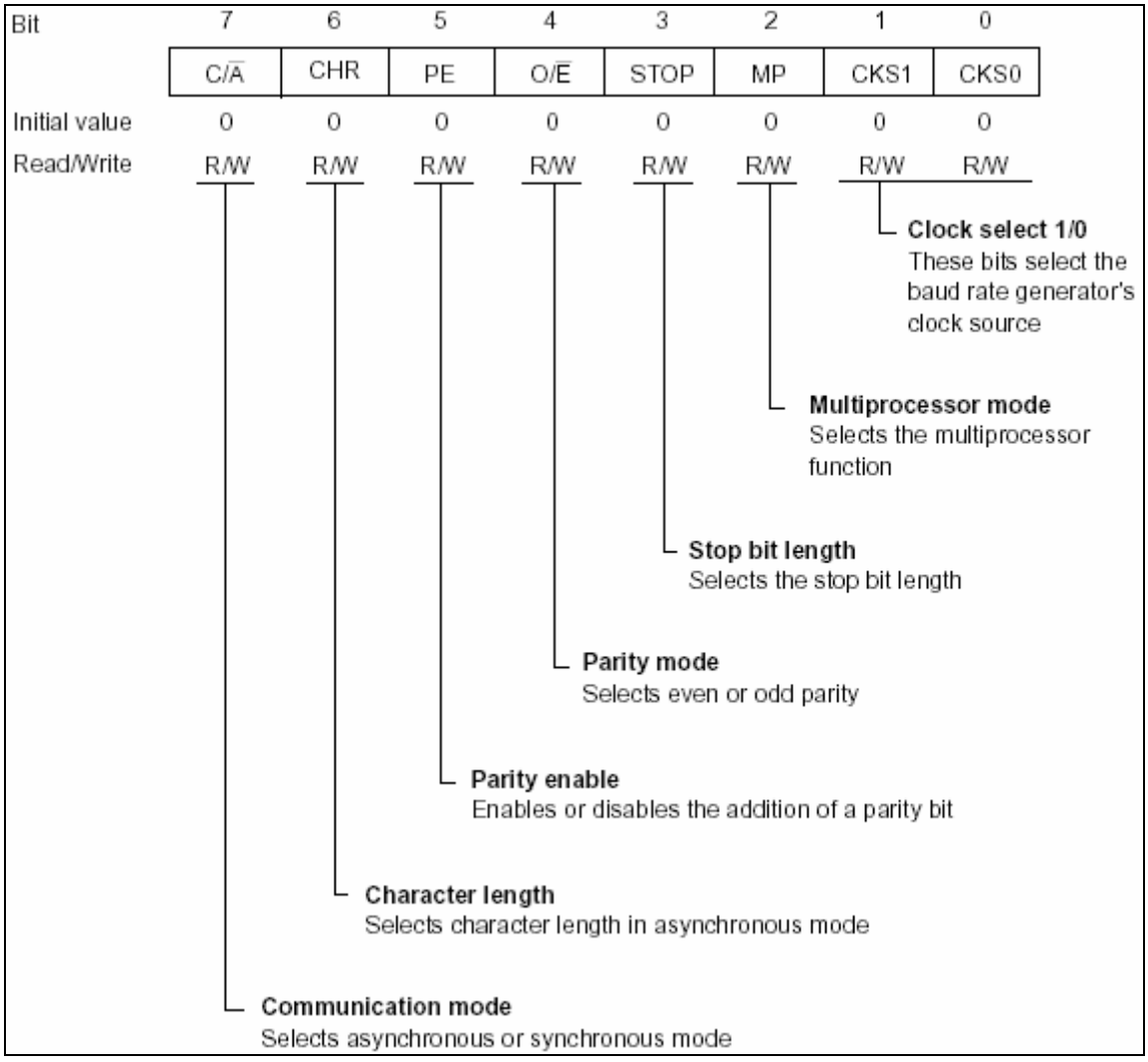


Figura 9.7: Serial Mode Register (SMR)

Bit 7 (C/A):
Este bit comparte función con el bit SMIF del registro SCMR y juntos pueden seleccionar la interfase de transmisión y el modo en el cual va a operar dicha interfase. Dado que no trabajaremos por el momento con la interfase de Tarjeta Inteligente (Smart Card), el bit SMIF deberá permanecer en cero.

Con la interfase SCI seleccionada, podemos operar en modo sincrónico poniendo el bit 7 en uno y para modo asincrónico, ponerlo en cero.

Bit 7 C/$\overline{A}$	Description	
0	Asynchronous mode	(Initial value)
1	Synchronous mode	

Figura 9. 8:C/A.

Bit 6 (CHR):
Aquí podemos seleccionar lo largo de la trama de datos, 7 bits si ponemos el bit en uno y 8 bits si lo ponemos en cero. Ahora, esto aplica nada más para cuando se esta trabajando en modo asincrónico, para el modo sincrónico el largo de la trama ya esta definido a 8 bits.
Cuando seleccionamos la longitud a 7 bits, el bit más significativo es el que no es transmitido.

Bit 6 CHR	Description	
0	8-bit data	(Initial value)
1	7-bit data*	
Note: * When 7-bit data is selected, the MSB (bit 7) of TDR is not transmitted.		

Figura 9.9: CHR.

Bit 5 (PE):
Habilitador de paridad. Con este bit habilitamos o no la función del chequeo de paridad en la recepción de datos. Debemos mencionar que esta función, al igual que la anterior es obviada cuando se trabaja en modo sincrónico.

Bit 5 PE	Description	
0	Parity bit not added or checked	(Initial value)
1	Parity bit added and checked*	
Note: * When PE bit is set to 1, an even or odd parity bit is added to transmit data according to the even or odd parity mode selection by the O/$\overline{E}$ bit, and the parity bit in receive data is checked to see that it matches the even or odd mode selected by the O/$\overline{E}$ bit.		

Figura 9.10: PE.

Bit 4 (O/E):
La selección de la paridad par (O/E=1) o impar (O/E=0) solo es validad si el bit anterior ha sido activado.

Bit 4 O/$\overline{E}$	Description	
0	Even parity*1	(Initial value)
1	Odd parity*2	
Notes: *1 When even parity is selected, the parity bit added to transmit data makes an even number of 1s in the transmitted character and parity bit combined. Receive data must have an even number of 1s in the received character and parity bit combined. *2 When odd parity is selected, the parity bit added to transmit data makes an odd number of 1s in the transmitted character and parity bit combined. Receive data must have an odd number of 1s in the received character and parity bit combined.		

Figura 9.11: O/E.

Bit 3 (STOP):
Seleccionamos la cantidad de bits de paro para la transmisión asincrónica, uno si STOP=0 o dos si STOP=1. Nuevamente esta función solo es valida para el modo asincrónico, en el modo sincrónico no existe bit de paro.

Bit 3 STOP	Description	
0	1 stop bit*1	(Initial value)
1	2 stop bits*2	
Notes: *1 One stop bit (with value 1) is added to the end of each transmitted character. *2 Two stop bits (with value 1) are added to the end of each transmitted character.		

Figura 9.12: STOP.

Bit 2 (MP):
Aquí seleccionamos la función de comunicación Multiprocesadores, con el cual podemos comunicarnos entre varios procesadores. Esta función solo es valida para el modo asincrónico, y cuando esta activada los estados de PE y O/E son ignorados.

Bit 2 MP	Description	
0	Multiprocessor function disabled	(Initial value)
1	Multiprocessor format selected	

Figura 9.13:MP.

Bits 1 y 0 (CKS1/0):
Con la combinación de estos bits seleccionamos la fuente de reloj para el generador de BAUDIOS el cual controla la velocidad de la transmisión. Cuatro fuentes pueden ser seleccionadas: Φ, Φ/4, Φ/16 y Φ/64.

Bit 1 CKS1	Bit 0 CKS0	Description	
0	0	φ	(Initial value)
0	1	φ/4	
1	0	φ/16	
1	1	φ/64	

Figura 9.14: CKS1/0.

9.2.4 Serial Control Register (SCR):

El SCR es un registro de lectura/escritura el cual es inicializado en cero después de un Reset o Standby y donde podemos activar o desactivar la transmisión y recepción, la salida de reloj en modo asincrónico, las interrupciones y seleccionar la fuente de reloj para la transmisión/recepción.

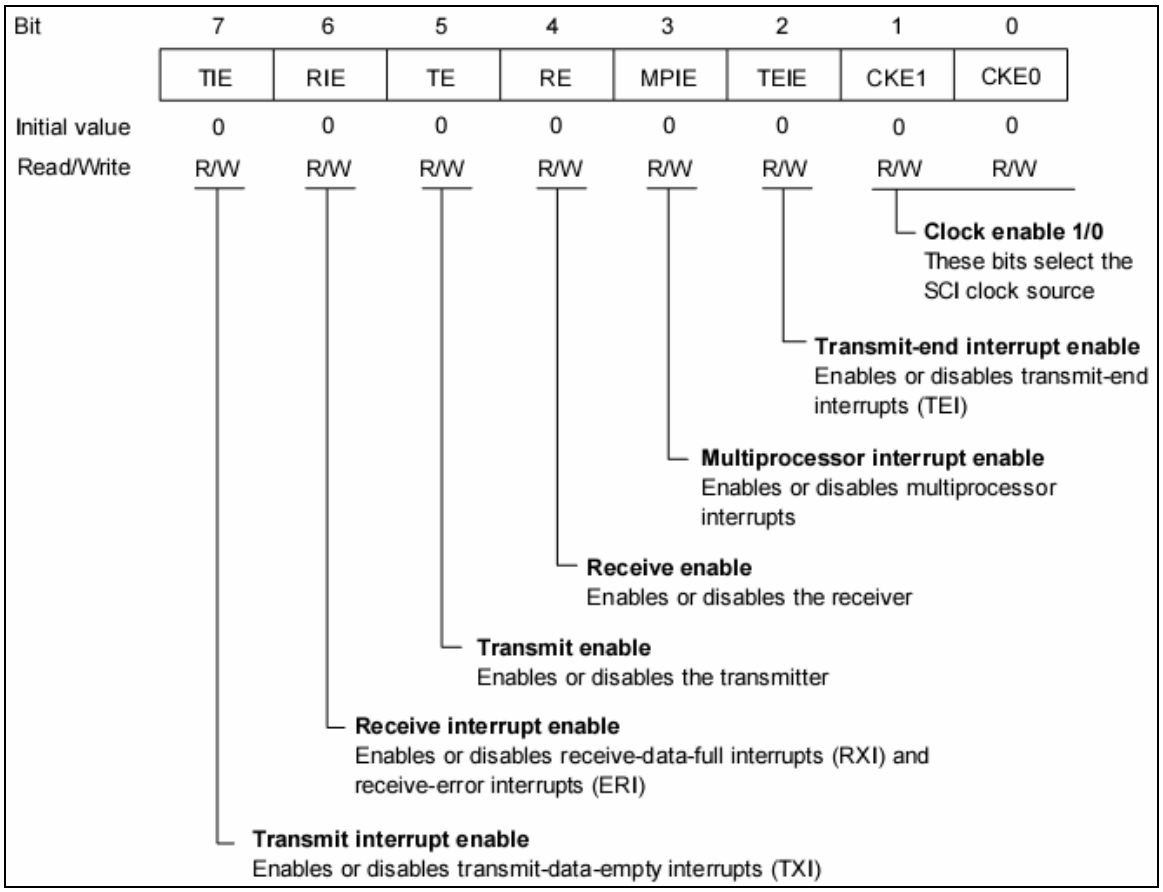


Figura 9. 15: Serial Control Register (SCR).

Bits 7 y 6 (TIE y RIE):
Recordemos que para poder transmitir un dato es necesario transferir el valor deseado de TDR a TSR para que luego este sea enviado por el pin correspondiente y de igual manera es necesario, después de la recepción de un dato, transferir el valor recibido de RSR a RDR para luego poder manipularlo. Ahora, cada vez que una de estas dos operaciones ha terminado, transferencia de TDR a TSR o de RSR a RDR, podemos hacer llamadas a interrupciones, siempre y cuando los bits TIE y RIE lo permitan.
Si estos bits están en cero, entonces la llamada a la interrupción no se efectuara y el caso contrario cuando estos están en uno.

Bit 7 TIE	Description	
0	Transmit-data-empty interrupt request (TXI) is disabled*	(Initial value)
1	Transmit-data-empty interrupt request (TXI) is enabled	
Note: * TXI interrupt requests can be cleared by reading the value 1 from the TDRE flag, then clearing it to 0; or by clearing the TIE bit to 0.		
Bit 6 RIE	**Description**	
0	Receive-data-full (RXI) and receive-error (ERI) interrupt requests are disabled*	(Initial value)
1	Receive-data-full (RXI) and receive-error (ERI) interrupt requests are enabled	
Note: * RXI and ERI interrupt requests can be cleared by reading the value 1 from the RDRF, FER, PER, or ORER flag, then clearing the flag to 0; or by clearing the RIE bit to 0.		

Figura 9. 16:TIE y RIE.

Bits 5 y 4 (TE y RE):
Con estos bits habilitamos (poniendo uno) o deshabilitamos (poniendo cero) el inicio de la operación de transmisión y recepción para el SCI.

Bit 5 TE	Description	
0	Transmitting disabled*1	(Initial value)
1	Transmitting enabled*2	
Notes: *1 The TDRE flag is fixed at 1 in SSR. *2 In the enabled state, serial transmission starts when the TDRE flag in SSR is cleared to 0 after writing of transmit data into TDR. Select the transmit format in SMR before setting the TE bit to 1.		
Bit 4 RE	**Description**	
0	Receiving disabled*1	(Initial value)
1	Receiving enabled*2	
Notes: *1 Clearing the RE bit to 0 does not affect the RDRF, FER, PER, and ORER flags. These flags retain their previous values. *2 In the enabled state, serial receiving starts when a start bit is detected in asynchronous mode, or serial clock input is detected in synchronous mode. Select the receive format in SMR before setting the RE bit to 1.		

Figura 9. 17: TE y RE.

Bit 3 (MPIE):
Activa (con uno) o desactiva (con cero) la llamada a interrupción para la comunicación múltiple entre procesadores pero solo para el modo asincrónico y cuando el bit MP del registro SMR esta puerto en uno.

Bit 3 MPIE	Description
0	Multiprocessor interrupts are disabled (normal receive operation) (Initial value) Clearing conditions (1) The MPIE bit is cleared to 0 (2) MPB = 1 in received data
1	Multiprocessor interrupts are enabled* Receive-data-full interrupts (RXI), receive-error interrupts (ERI), and setting of the RDRF, FER, and ORER status flags in SSR are disabled until data with the multiprocessor bit set to 1 is received.
Note: * The SCI does not transfer receive data from RSR to RDR, does not detect receive errors, and does not set the RDRF, FER, and ORER flags in SSR. When it receives data in which MPB = 1, the SCI sets the MPB bit to 1 in SSR, automatically clears the MPIE bit to 0, enables RXI and ERI interrupts (if the TIE and RIE bits in SCR are set to 1), and allows the FER and ORER flags to be set.	

Figura 9. 18:MPIE.

Bit 2 (TEIE):
Habilita (con uno) o deshabilita (con cero) la llamada a interrupción al final de una transmisión. Esto si el TDR no contiene ningún dato valido para ser transferido al TSR cuando el dato anterior ya esta por ser terminado de enviar.

Bit 2 TEIE	Description	
0	Transmit-end interrupt requests (TEI) are disabled*	(Initial value)
1	Transmit-end interrupt requests (TEI) are enabled*	
Note: * TEI interrupt requests can be cleared by reading the value 1 from the TDRE flag in SSR, then clearing the TDRE flag to 0, thereby also clearing the TEND flag to 0; or by clearing the TEIE bit to 0.		

Figura 9. 19: TEIE.

Bits 1 y 0 (CKE1/0):
Al igual que el bit 7 del SMR (C/A), estos dos bits cambian de función cuando se esta trabajando con la interfase SCI o con la Smart Card (con la cual no vamos a trabajar). Para a interfase SCI, estos bits seleccionan la fuente de reloj y activan o desactivan la salida de reloj por el pin SCK. Dependiendo de la configuración de los bits CKE1 y CKE0, el bit SCK puede ser usado de las diferentes formas presentadas a continuación:

Bit 1 CKE1	Bit 0 CKE0	Description	
0	0	Asynchronous mode	Internal clock, SCK pin available for generic input/output*1
		Synchronous mode	Internal clock, SCK pin used for serial clock output*1
0	1	Asynchronous mode	Internal clock, SCK pin used for clock output*2
		Synchronous mode	Internal clock, SCK pin used for serial clock output
1	0	Asynchronous mode	External clock, SCK pin used for clock input*3
		Synchronous mode	External clock, SCK pin used for serial clock input
1	1	Asynchronous mode	External clock, SCK pin used for clock input*3
		Synchronous mode	External clock, SCK pin used for serial clock input

Notes: *1 Initial value
*2 The output clock frequency is the same as the bit rate.
*3 The input clock frequency is 16 times the bit rate.

Figura 9. 20: CKE1/0.

Notas:

- La configuración de CKE0 solo es valida para el modo asincrónico y cuando el reloj interno es seleccionado (CKE1=0) y es ignorada para CKE1=0 o en modo sincrónico.
- Antes de programar los bits CKE1 y CKE0 es recomendable seleccionar el modo de operación en el registro SMR.

9.2.5 Serial Status Register (SSR)

Este registro de lectura/escritura contiene los valores y los estados de las banderas que indican el estado de operación de la comunicación SCI. Es necesario mencionar que para poder limpiar las banderas es necesario escribir un cero en los bits respectivos después de haberlos leído (cuando su valor era de uno durante la lectura). Además, existe la limitante de no poder escribir un uno en los bits TDRE, RDRF, ORER, PER Y FER.

En la siguiente Figura se muestra cada uno de bits del registro SSR y posteriormente se detallara su uso:

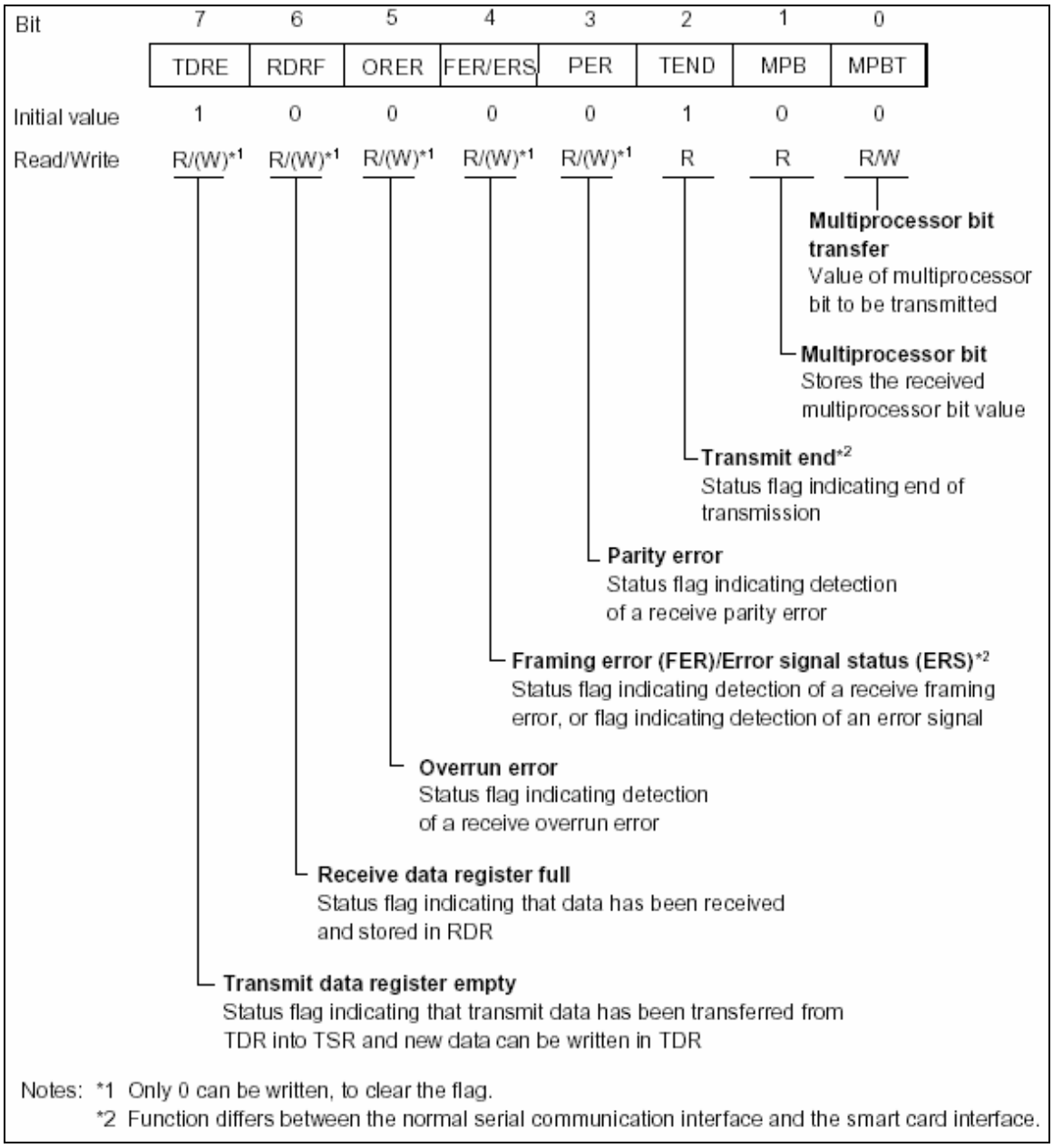


Figura 9. 21: Serial Status Register (SSR)

Bits 7 y 6 (TDRE y RDRF):

Estos dos bits hacen posible el juego entre los dos registros (lo cuales actúan como buffer) permitiendo que la transmisión o recepción sean continuas.

Cuando hay un uno, el primero nos indica que el valor introducido en el TDR ya ha sido transferido al TSR para su transmisión, de modo que el TDR esta vació y disponible para recibir el nuevo valor. Por otro lado, el segundo, indica que se ha recibido satisfactoriamente un dato en el RDR. Dicho dato ya fue recibido primeramente por el RSR y luego transferido al RDR.

Bit 7 TDRE	Description
0	TDR contains valid transmit data Clearing conditions Read TDRE when TDRE = 1, then write 0 in TDRE The DMAC writes data in TDR
1	TDR does not contain valid transmit data (Initial value) Setting conditions The chip is reset or enters standby mode The TE bit in SCR is cleared to 0 TDR contents are loaded into TSR, so new data can be written in TDR

Bit 6 RDRF	Description
0	RDR does not contain new receive data (Initial value) Clearing conditions The chip is reset or enters standby mode Read RDRF when RDRF = 1, then write 0 in RDRF The DMAC reads data from RDR
1	RDR contains new receive data Setting condition Serial data is received normally and transferred from RSR to RDR

Note: The RDR contents and the RDRF flag are not affected by detection of receive errors or by clearing of the RE bit to 0 in SCR. They retain their previous values. If the RDRF flag is still set to 1 when reception of the next data ends, an overrun error will occur and the receive data will be lost.

Figura 9. 22: TDRE y RDRF.

Bit 5 (ORER):
Esta bandera indica con un uno que ha habido un error de sobre escritura y la recepción ha sido finalizado de manera anormal.

Bit 5 ORER	Description
0	Receiving is in progress or has ended normally*1 (Initial value) Clearing conditions The chip is reset or enters standby mode Read ORER when ORER = 1, then write 0 in ORER
1	A receive overrun error occurred*2 Setting condition Reception of the next serial data ends when RDRF = 1

Notes: *1 Clearing the RE bit to 0 in SCR does not affect the ORER flag, which retains its previous value.
*2 RDR continues to hold the receive data prior to the overrun error, so subsequent receive data is lost. Serial receiving cannot continue while the ORER flag is set to 1. In synchronous mode, serial transmitting is also disabled.

Figura 9. 23: ORER.

Bit 4 (FER/ERS):
La función de este bit difiere dependiendo de la interfase con la cual estamos trabajando (serial o Smart Card). En nuestro caso, en donde solo trabajaremos con la interfase serial, este bit indicará con un uno que la transmisión finaliza de manera anormal debido a un error en la trama, cuando la comunicación es del tipo asincrónica.

Bit 4 FER	Description
0	Receiving is in progress or has ended normally*1 (Initial value) Clearing conditions The chip is reset or enters standby mode Read FER when FER = 1, then write 0 in FER
1	A receive framing error occurred*2 Setting condition The stop bit at the end of the receive data is checked and found to be 0
Notes: *1	Clearing the RE bit to 0 in SCR does not affect the FER flag, which retains its previous value.
*2	When the stop bit length is 2 bits, only the first bit is checked. The second stop bit is not checked. When a framing error occurs the SCI transfers the receive data into RDR but does not set the RDRF flag. Serial receiving cannot continue while the FER flag is set to 1. In synchronous mode, serial transmitting is also disabled.

Figura 9. 24:FER/ERS.

Bit 3 (PER):
Indica con un uno cuando ha existido un error de paridad en la transmisión en modo asincrónico y la dicha transmisión ha sido detenida anormalmente.

Bit 3 PER	Description
0	Receiving is in progress or has ended normally*1 (Initial value) Clearing conditions The chip is reset or enters standby mode Read PER when PER = 1, then write 0 in PER
1	A receive parity error occurred*2 Setting condition The number of 1s in receive data, including the parity bit, does not match the even or odd parity setting of O/$\overline{E}$ in SMR
Notes: *1	Clearing the RE bit to 0 in SCR does not affect the PER flag, which retains its previous value.
*2	When a parity error occurs the SCI transfers the receive data into RDR but does not set the RDRF flag. Serial receiving cannot continue while the PER flag is set to 1. In synchronous mode, serial transmitting is also disabled.

Figura 9. 25: PER.

Bit 2 (TEND):
La función de este bit también depende de la interfase con la cual se esta trabajando: serial o Smart Card. Nuevamente es de especificar que solo usaremos la primera.
En ese caso, esta bandera indicaría con un uno que cuando el último carácter del dato ha sido enviado por el TSR no había un dato valido en el TDR de modo que la transmisión termina de una manera adecuada.

Bit 2 TEND	Description	
0	Transmission is in progress Clearing conditions Read TDRE when TDRE = 1, then write 0 in TDRE The DMAC writes data in TDR	
1	End of transmission Setting conditions The chip is reset or enters standby mode The TE bit in SCR is cleared to 0 TDRE is 1 when the last bit of a 1-byte serial transmit character is transmitted	(Initial value)

Figura 9. 26:TEND.

Bits 1 y 0 (MPB y MPBT):
Estos bit funcionan únicamente cuando esta activa la opción de comunicación con múltiples procesadores. En ellos se guarda el valor del dato recibido o agregado para la transmisión en el modo *Multiprocessor*. El ultimo de los dos (MPBT) es ignorado cuando se esta trabajando en de manera sincrónica, cuando no se esta trasmitiendo nada por el SCI o cuando el modo *Multiprocessor* no ha sido habilitado.

Bit 1 MPB	Description	
0	Multiprocessor bit value in receive data is 0*	(Initial value)
1	Multiprocessor bit value in receive data is 1	
Note: * If the RE bit in SCR is cleared to 0 when a multiprocessor format is selected, MPB retains its previous value.		
MPBT	**Description**	
0	Multiprocessor bit value in transmit data is 0	(Initial value)
1	Multiprocessor bit value in transmit data is 1	

Figura 9. 27: MPB y MPBT.

9.2.6 Bit Rate Register (BRR):

Este registro de lectura/escritura es inicializado con H´FF después de un Reset o Standby, junto con los bits CKE1/0 del registro SMR establecen la tasa de transferencia para la comunicación serial. Es posible tener, para cada uno de los canales, su propia tasa de transferencia.

Bit	7	6	5	4	3	2	1	0
Initial value	1	1	1	1	1	1	1	1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figura 9. 28: Bit Rate Register (BRR).

Los valores para el BRR son obtenidos de manera diferente según el modo de operario (asincrónico y sincrónico) y los cálculos a realizar se muestran en la figura 9.10.

Asynchronous mode:

$$N = \frac{\phi}{64 \times 2^{2n-1} \times B} \times 10^6 - 1$$

Synchronous mode:

$$N = \frac{\phi}{8 \times 2^{2n-1} \times B} \times 10^6 - 1$$

B: Bit rate (bit/s)
N: BRR setting for baud rate generator ($0 \leq N \leq 255$)
ϕ: System clock frequency (MHz)
n: Baud rate generator clock source (n = 0, 1, 2, 3)
(For the clock sources and values of n, see the following table.)

Figura 9. 29: Cálculos para el BRR

El valor de n depende de la fuente de reloj que se ha hecho con los bits CKE1 y CKE0 del registro SMR según la siguiente tabla:

		SMR Settings	
n	Clock Source	CKS1	CKS0
0	ϕ	0	0
1	ϕ/4	0	1
2	ϕ/16	1	0
3	ϕ/64	1	1

Tabla 9. 2: Efecto de CKE1/0 para los cálculos de la tasa de intercambio.

Además, es posible calcular el porcentaje de error de la tasa de transferencia en modo asincrónico según la ecuación numero 1:

Ecuación número 1:

$$\text{Error (\%)} = \left\{ \frac{\phi \times 10^6}{(N + 1) \times B \times 64 \times 2^{2n-1}} - 1 \right\} \times 100$$

9.3 Funcionamiento

Como dijimos anteriormente, el SCI puede efectuar la comunicación serial de dos maneras: de forma asincrónica en donde la sincronización es hecha carácter por carácter y de forma sincrónica en donde la sincronización se hace con pulsos de reloj.

A continuación presentaremos el modo de operación y las características de cada uno de los modos de comunicación antes mencionados. Los detalles del funcionamiento serán presentados en la guía de laboratorio numero nueve.

9.3.1 Modo asincrónico:

En este modo cada una de las tramas, de 7 ó 8 bits, que son enviadas o recibidas inicia con un bit de Start y finalizan con uno o dos bits de Stop (según se ha configurado). Además, puede o no tener un bit para verificar la paridad, en caso que esta opción haya sido activada. La figura siguiente muestra la constitución de la trama:

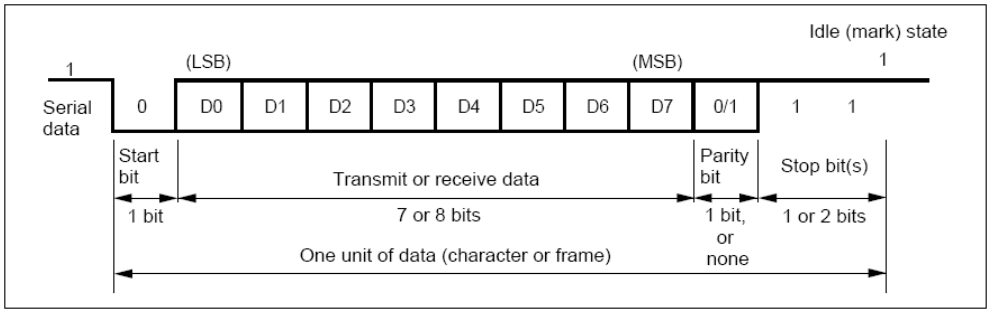


Figura 9. 30: Trama de comunicación serial en modo asincrónico.

Como se puede ver, la línea de transmisión permanece en estado alto cuando no esta siendo usada. Al detectar un flanco negativo, lo cual indica un bit de inicio, la trama comenzará a transmitirse. El dato a transferir es colocado después de este bit con el bit menos significativo al principio, luego tendremos el bit de paridad y uno o dos bits en alto, los cuales son los bits de paro. Las diferentes combinaciones para la configuración pueden apreciarse mejor en la figura 9.12.

Por otro lado, debemos mencionar que la transmisión y recepción de dentro del SCI son totalmente independientes de modo que se puede tener una comunicación full-duplex, y como ambas funciones tienen doble registro, permitiendo escritura y transmisión a la vez, es posible tener una comunicación continua.

SMR Settings				Serial Communication Format and Frame Length
CHR	PE	MP	STOP	1 2 3 4 5 6 7 8 9 10 11 12
0	0	0	0	S 8-bit data STOP
0	0	0	1	S 8-bit data STOP STOP
0	1	0	0	S 8-bit data P STOP
0	1	0	1	S 8-bit data P STOP STOP
1	0	0	0	S 7-bit data STOP
1	0	0	1	S 7-bit data STOP STOP
1	1	0	0	S 7-bit data P STOP
1	1	0	1	S 7-bit data P STOP STOP
0	—	1	0	S 8-bit data MPB STOP
0	—	1	1	S 8-bit data MPB STOP STOP
1	—	1	0	S 7-bit data MPB STOP
1	—	1	1	S 7-bit data MPB STOP STOP

Figura 9. 31: Combinaciones posibles para el funcionamiento en modo asincrónico

Leyenda:

S:	Start bit
STOP:	Stop bit
P:	Parity bit
MPB:	Multiprocessor bit

Sincronización:

El reloj para la transmisión/recepción puede provenir de dos fuentes posibles: interna o externa, según estén configurados los bits C/A del registro SMR y CKE1/0 del registro SCR.

En el primer caso, la señal generada puede ser transmitida y además la frecuencia de esta señal es igual a la tasa de transferencia de la trama. La fase es alineada y sincronizada en el momento que se detecta el flanco negativo del bit de Start de modo a dejar el flanco positivo justo al centro y capturar/enviar el dato en ese momento.

En el segundo caso, reloj externo, este debe tener un frecuencia equivalente a 16 veces la tasa de transferencia deseada.

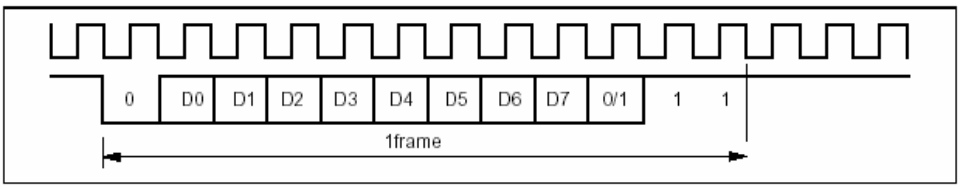


Figura 9. 32: Relación entre la fase y la salida de reloj (reloj interno).

9.3.2 Modo de Multiprocesador:

Este modo es usado para la comunicación asincrónica entre dos o más procesadores mediante una misma línea de comunicación serial. En este modo la comunicación se hace en dos tiempos, primero, el transmisor debe identificar al receptor mediante el envió de un numero. Este numero es tomado por todos lo posibles receptores y es comparado con el numero configurado en sus registros, de ser igual ese procesador va a recibir toda la información a ser enviada. Los demás procesadores ignoraran toda transferencia hasta recibir nuevamente un número para comparar.

Ahora, para distinguir cuando se esta enviando este numero y cuando se esta enviando información, utilizamos el bit llamado *Multiprocessor bit*. Cuando este esta en uno, se indica que la trama que lo acompaña hace referencia al número de identificación (leído por todos). Cuando esta en cero, la trama que lo acompaña es información (leído solamente por el receptor).

Una vez se haya identificado el procesador con el cual nos vamos a comunicar, comienza el envió de la información (con el *Multiprocessor bit* en cero). En la figura 9.14 podemos observar un ejemplo de la comunicación en modo multiprocesador, en la etapa de identificación se selecciono el procesador A enviando el numero H'01 acompañado del *Multiprocessor bit* en uno. Luego se envía el dato H'AA (acompañado del *Multiprocessor bit* en cero), y solo el procesador seleccionado lo recibe.

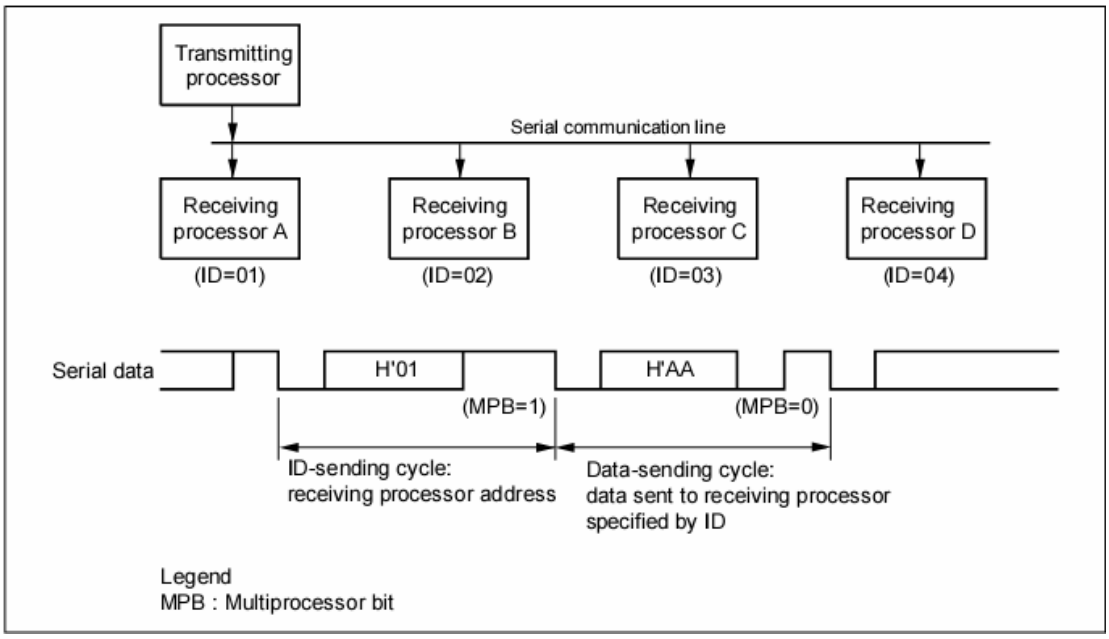


Figura 9. 33: Comunicación en modo Multiprocesor

9.3.3 Modo sincrónico:

En este modo la sincronización se logra usando una señal de reloj (interna o externa, seleccionable) lo cual permite transmisiones a alta frecuencia. Si bien las opciones de transmisión y recepción comparten la misma señal de reloj, estas son completamente independientes de modo que podemos tener comunicación full-duplex al igual que para la comunicación asincrónica. Además, tenemos la misma característica de buffer lo cual permite escritura/lectura y envió al mismo tiempo haciendo que la comunicación sea lo más continua posible.

Los datos son transmitidos del bit menos significativo al bit mas significativo (sea envió o recepción), los bits son puestos en la línea al detectar un flanco negativo de la señal de reloj y son verificados en el flanco positivo. Ahora, al terminarse una trama de datos (fijada a 8 bits) es posible que la comunicación termine o continué dependiendo si esta o no en modo de transmisión continúa. De no estarlo, el reloj queda en estado alto y la línea de transmisión queda en el estado del último bit enviado.

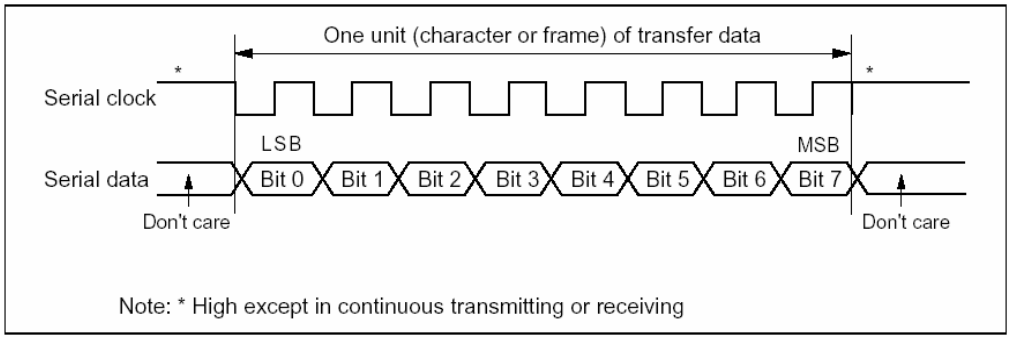


Figura 9. 34: Formato de dato en la comunicación sincrónica

9.4 Interrupciones

Como dijimos anteriormente, el SCI tiene 4 fuentes de llamada a interrupción que pueden ser activadas o desactivadas mediante los bits TIE, RIE, y TEIE del registro SCR. Cada llamada es atendida separadamente por una interrupción, las fuentes son:

- Transmit-end interrupt (TEI), cuando la bandera de fin de transmisión TEND se pone en uno.
- Receive-error interrupt (ERI), cuando las banderas ORER, PER o FER están en uno
- Receive-data-full interrupt (RXI), cuando la bandera RDRF del SSR es puesta en uno. Esta llamada podría eventualmente activar el DMAC de estar activada la opción. De estar activada, la bandera limpiará automáticamente la bandera RDRF
- Transmit-data-empty interrupt (TXI), cuando la bandera TDRE es puesta a uno en el registro SSR. Al igual que la llamada anterior, esta puede activar el DMAC y la transferencia de datos usando esta opción limpia automáticamente la bandera.

Interrupt Source	Description	Priority
ERI	Receive error (ORER, FER, or PER)	High
RXI	Receive data register full (RDRF)	
TXI	Transmit data register empty (TDRE)	
TEI	Transmit end (TEND)	Low

Tabla 9. 3: Prioridad en las interrupciones para el SCI.

Capitulo Diez: WatchDog Timer [19]

[19] Para mayor información hacer referencia a la sección 12 del Manual de Hardware [B1].

10.1 Generalidades

El WatchDog Timer puede operar de dos formas diferentes y desencadenar acciones diferentes dependiendo de la configuración de sus registros.
Su función principal es operar como guardián de la integridad del funcionamiento del programa, para ello el contador TCNT debe ser sobre escrito cada cierto tiempo, antes que desborde. Al haber un desbordamiento, el sistema completo se reiniciara (Reset). Por otro lado tenemos también la opción de usar el TCNT como un simple contador de intervalo de ocho bits el cual cada vez que desborde llamara a una interrupción.

Las principales características de este WDT son entonces las siguientes:

- Al haber un Overflow en el TCNT se puede general un reinicio completo del sistema o llamar a una interrupción dependiendo de la configuración.
- Posibilidad de funcionamiento como temporizador interno de intervalo.
- Tenemos la opción de seleccionar la fuente de reloj entre ocho posibilidades, todas dependientes del Clock del sistema.

A continuación presentamos el diagrama de bloques del sistema:

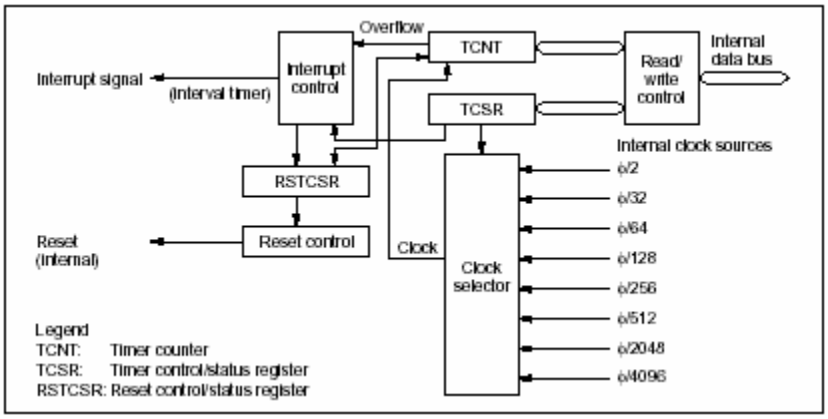


Figura 10. 1: Diagrama de bloques del WDT

10.2 Configuración de registros

Tenemos tres registros de ocho bits para poner en funcionamiento este temporizador especial, dos de ellos son de control y el otro es el registro de conteo. Las direcciones y los nombres de cada uno de ellos son especificados en la tabla siguiente:

Address*1					
Write*2	Read	Name	Abbreviation	R/W	Initial Value
H'FFF8C	H'FFF8C	Timer control/status register	TCSR	R/(W)*3	H'18
	H'FFF8D	Timer counter	TCNT	R/W	H'00
H'FFF8E	H'FFF8F	Reset control/status register	RSTCSR	R/(W)*3	H'3F

Notes: *1 Lower 20 bits of the address in advanced mode.
*2 Write word data starting at this address.
*3 Only 0 can be written in bit 7, to clear the flag.

Tabla 10. 1: Nombre y dirección de cada registro para el WDT

10.2.1 Timer Counter (TCNT):

Este registro cumple la función de contador accedente de ocho bits. El conteo inicia después de recibir el la señal de Start mediante el registro TCSR (TME en uno) y el valor se incrementa con cada pulso de la señal de reloj seleccionada mediante los bits CKS2-CKS0 del registro TCSR.
Cuando el TCNT se desborda pasando de H´FF a H´00 se activa la bandera de Overflow OVF la cual hace posible el Reset o la llamada a la interrupción. Para poder inicializar este contador es necesario hacer un Reset o poner un cero en TME (lo cual detendría el conteo).

Bit	7	6	5	4	3	2	1	0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note: TCNT is write-protected by a password. For details see section 12.2.4, Notes on Register Access.

Figura 10. 2: Timer Counter (TCNT)

10.2.2 Timer Control/Status Register (TCSR):

Este registro de lectura escritura permite seleccionar la fuente de reloj para el conteo de pulsos y controlar y configurar el WDT:

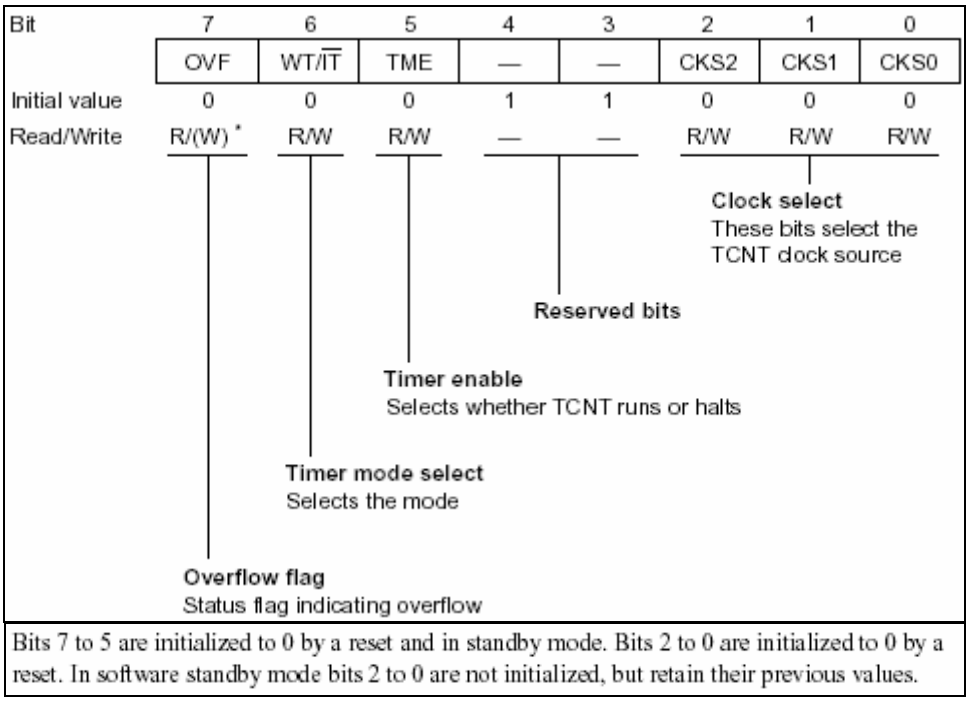


Figura 10. 3: Timer Control/Status Register (TCSR)

Bit 7 (OVF): Este bit nos indica que el contador ha desbordado pasando de H´FF a H´00.

Bit 7 OVF	Description	
0	[Clearing condition] Cleared by reading OVF when OVF = 1, then writing 0 in OVF	(Initial value)
1	[Setting condition] Set when TCNT changes from H'FF to H'00	

Figura 10. 4: OVF

Bit 6 (WT/IT): Aquí hacemos la selección de la función del WDT. Configuramos si se trabajará en la modalidad de WatchDog Timer como tal (WT/IT en uno) generando un Reset con cada Overflow del TCNT o si se trabajará con un contador de intervalo (WT/IT en cero) generando una llamada a interrupción con cada Overflow.

Bit 6 WT/IT	Description	
0	Interval timer: requests interval timer interrupts	(Initial value)
1	Watchdog timer: generates a reset signal	

Figura 10. 5: WT/IT.

Bit 5 (TME): A través de este bit iniciamos (con uno) o detenemos (con cero) el conteo del TCNT.

> Nota: Si estamos trabajando en el modo de WatchDog Timer, debemos de limpiar el SSBY (Software Standby) del registro SYSCR antes de poner en uno el TME. Cuando el SSBY se pone en uno el TM deberá estar en cero.

Bit 5 TME	Description	
0	TCNT is initialized to H'00 and halted	(Initial value)
1	TCNT is counting	

Figura 10. 6: TME.

Bit 2 al 0 (CKS2-CKS0): Con estos tres bits haremos la selección de la fuente de reloj, la cual se obtiene haciendo una pre-escala del clock del sistema según la siguiente tabla:

Bit 2 CKS2	Bit 1 CKS1	Bit 0 CKS0	Description	
0	0	0	ϕ/2	(Initial value)
		1	ϕ /32	
	1	0	ϕ /64	
		1	ϕ /128	
1	0	0	ϕ /256	
		1	ϕ /512	
	1	0	ϕ /2048	
		1	ϕ /4096	

Figura 10. 7: CKS2-CKS0.

10.2.3 Reset Control/Status Register (RSTCSR):

Este registro de lectura/escritura indica cuando una señal de Reset ha sido generada por un Overflow en la modalidad de WatchDog Timer, y controla la señal de salida de este Reset. El bits involucrado para este registro solo puede ser inicializado mediante un reset proveniente del pin físico de Reset, el reset provocado por el WDT no tiene ningún efecto sobre ellos.

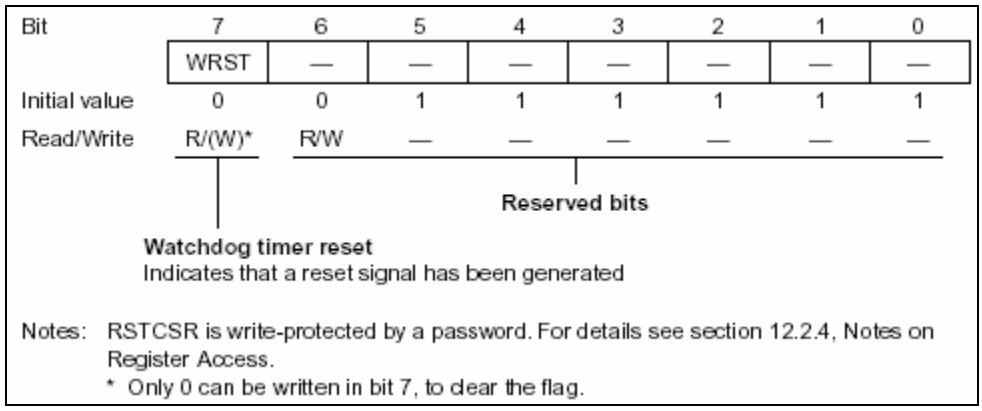


Figura 10. 8; Reset Control/Status register

Bit 7 (WRST): Cuando se esta trabajando en modalidad de WatchDog Timer, este bit indica que se ha generado un Reset general del H8/3069F debido a un Overflow en el TCNT.

Bit 7 WRST	Description
0	[Clearing condition] Reset signal at $\overline{RES}$ pin. Read WRST when WRST =1, then write 0 in WRST. (Initial value)
1	[Setting condition] Set when TCNT overflow generates a reset signal during watchdog timer operation

Figura 10. 9;WRST.

Nota: Todos los demás bits están reservados y deben tener el valor indicados en la figura descriptiva del registro.

10.2.4 Acceso a los registros:

Los registros de este subsistema son un poco diferentes a los de los demás en cuanto a acceso se trata. La lectura de estos no sufre mayores cambios, siempre se ejecuta a través de las instrucciones y con el procedimiento normal de lectura, solo es de tener cuidado con la dirección del registro que estamos leyendo. A continuación presentamos una tabla con las direcciones de lectura:

Address*	Register
H'FFF8C	TCSR
H'FFF8D	TCNT
H'FFF8F	RSTCSR
Note: * Lower 20 bits of the address in advanced mode.	

Figura 10. 10: Direcciones de lectura de registros

Ahora, para la escritura en estos registros debemos usar instrucciones de transferencia de palabras, no podemos hacerlo con manipulación de byte o bite. Además, es necesario

respetar los diferentes códigos de acceso para cada registro, los cuales deben de estar escritos en la parte alta de la palabra que vamos a transferir como se indica en la figura 10.5:

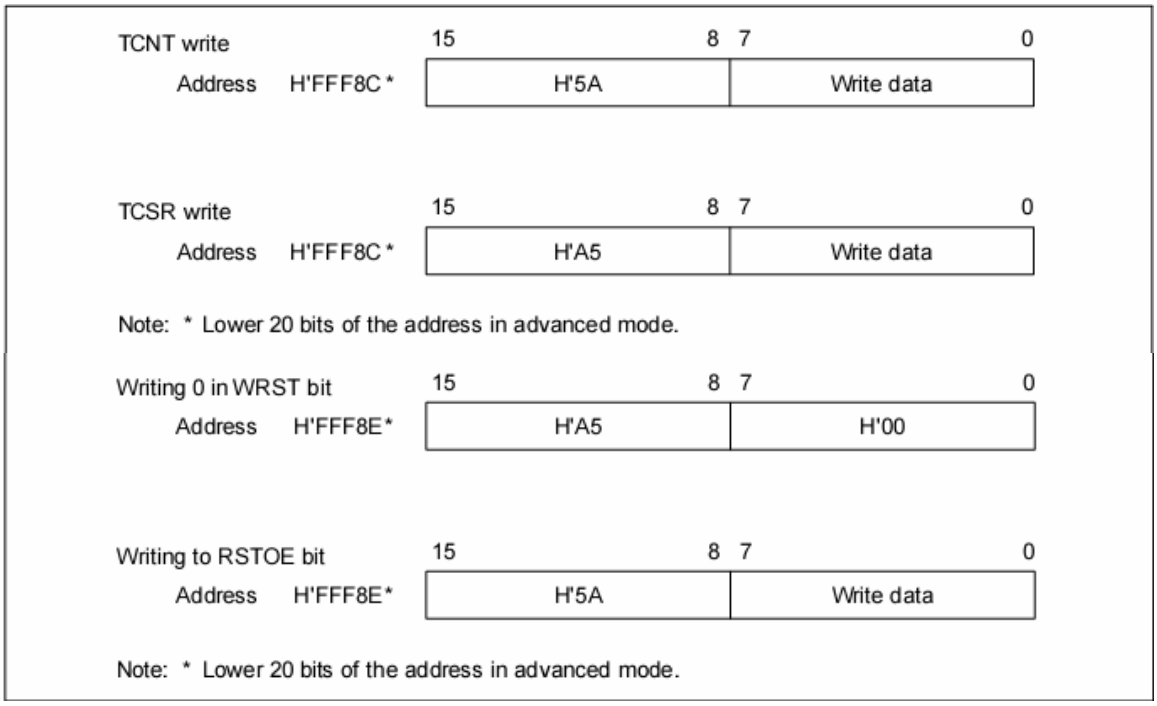


Figura 10. 11: Direcciones y códigos de acceso para la escritura.

Si bien es cierto que los cuatro registros comparten solo dos direcciones, es posible hacer la diferencia entre ellos con la palabra clave.

10.3 Modos de operación

10.3.1 WatchDog Timer:

Para que el temporizador WDT funcione en esta modalidad es necesario poner los bits WT/IT y TME del registro TCSR en uno. Para evitar que el contador tenga un Overflow el software deberá escribir cada cierto tiempo H´00 en el registro TCTN. Con esto estamos logrando tener un monitoreo de la ejecución de dicho software, evitando que el sistema se quede en un bucle infinito o demasiado largo.
En caso de generarse un error de este tipo o por cualquier otro motivo que haga que el sistema se caiga, un Reset, que dure 518 estados, será generado. Este Reset tiene el mismo vector que el que es generado por el pin físico, pero gracias al bit WRST del registro RSTCSR podemos hacer la diferencia que quien provoco dicha acción. La siguiente figura ilustra bastante bien el funcionamiento de esta modalidad:

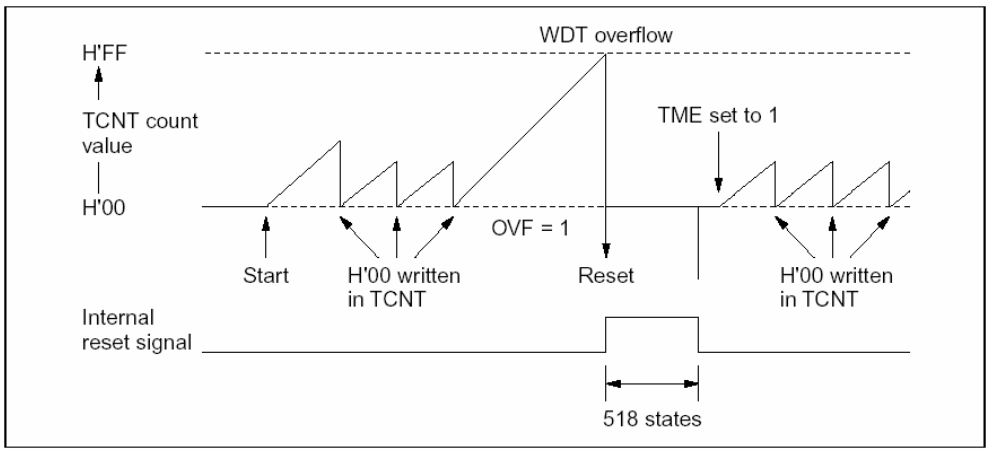


Figura 10. 12: Modo de WatchDog Timer

Nota: En caso de tener un Reset por pin físico y otro generado por el WDT, la prioridad la tendrá el del pin físico.

10.3.2 Modo de temporización de intervalo:

Para trabajar bajo esta modalidad es necesario tener los bits WT/IT y TME del registro TCSR configurados en cero y en uno (respectivamente). Este modo nos permite llamar una interrupción determinada cada cierto tiempo (intervalo de tiempo establecido), al momento que el TCNT se desborde como lo ilustra la figura siguiente:

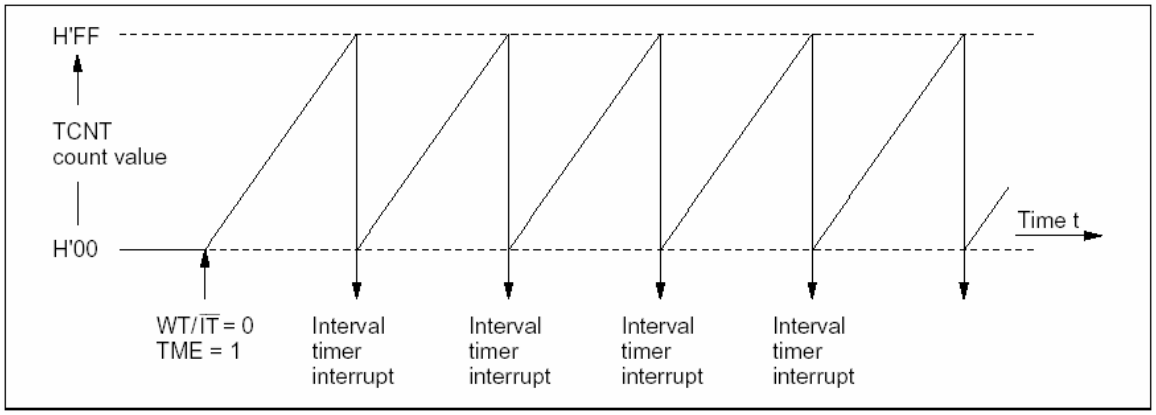


Figura 10. 13: Modo de temporización de intervalo.

10.4 Interrupciones

Cada vez que el bit OVF del registro TCSR se pone en uno y que el modo de operaciones seleccionado sea el de temporización de intervalo, se llamará a la interrupción interna de temporización (WOVI). A continuación presentamos un fragmento de la tabla de vectores

de interrupción para poder ubicar la dirección donde deberá comenzar el código de la interrupción llamada.

Interrupt Source	Origin	Vector Number	Vector Address*1 Advanced Mode	Vector Address*1 Normal Mode*2	IPR	Priority
NMI	External pins	7	H'001C to H'001F	H'000E to H'000F		High
IRQ_0		12	H'0030 to H'0033	H'0018 to H'0019	IPRA7	
IRQ_1		13	H'0034 to H0037	H'001A to H'001B	IPRA6	
IRQ_2		14	H'0038 to H'003B	H'001C to H'001D	IPRA5	
IRQ_3		15	H'003C to H'003F	H'001E to H'001F		
IRQ_4		16	H'0040 to H'0043	H'0020 to H'0021	IPRA4	
IRQ_5		17	H'0044 to H'0047	H'0022 to H'0023		
Reserved		18	H'0048 to H'004B	H'0024 to H'0025		
		19	H'004C to H'004F	H'0026 to H'0027		
WOVI (interval timer)	Watchdog timer	20	H'0050 to H'0053	H'0028 to H'0029	IPRA3	

Figura 10. 14: Tabla de vector de interrupciones (fragmento).

Capitulo Once: TPC [20]

[20] Para mayor información hacer referencia a la sección 11 del Manual de Hardware [B1].

11.1 Generalidades

Este subsistema[21] provee cuatro grupos (grupo cero a grupo 3) de salida de pulsos basado en el temporizador de 16 bits. Cada uno de ellos es independiente y pueden funcionar simultáneamente y con diferentes señales de disparo. Además, contienen 4 bits de salida de modo que tenemos un total de 16 salidas posibles (TP_0-TP_{15}) las cuales están multiplexadas con los puertos A y B.

La siguiente figura muestra el diagrama de bloques del TPC:

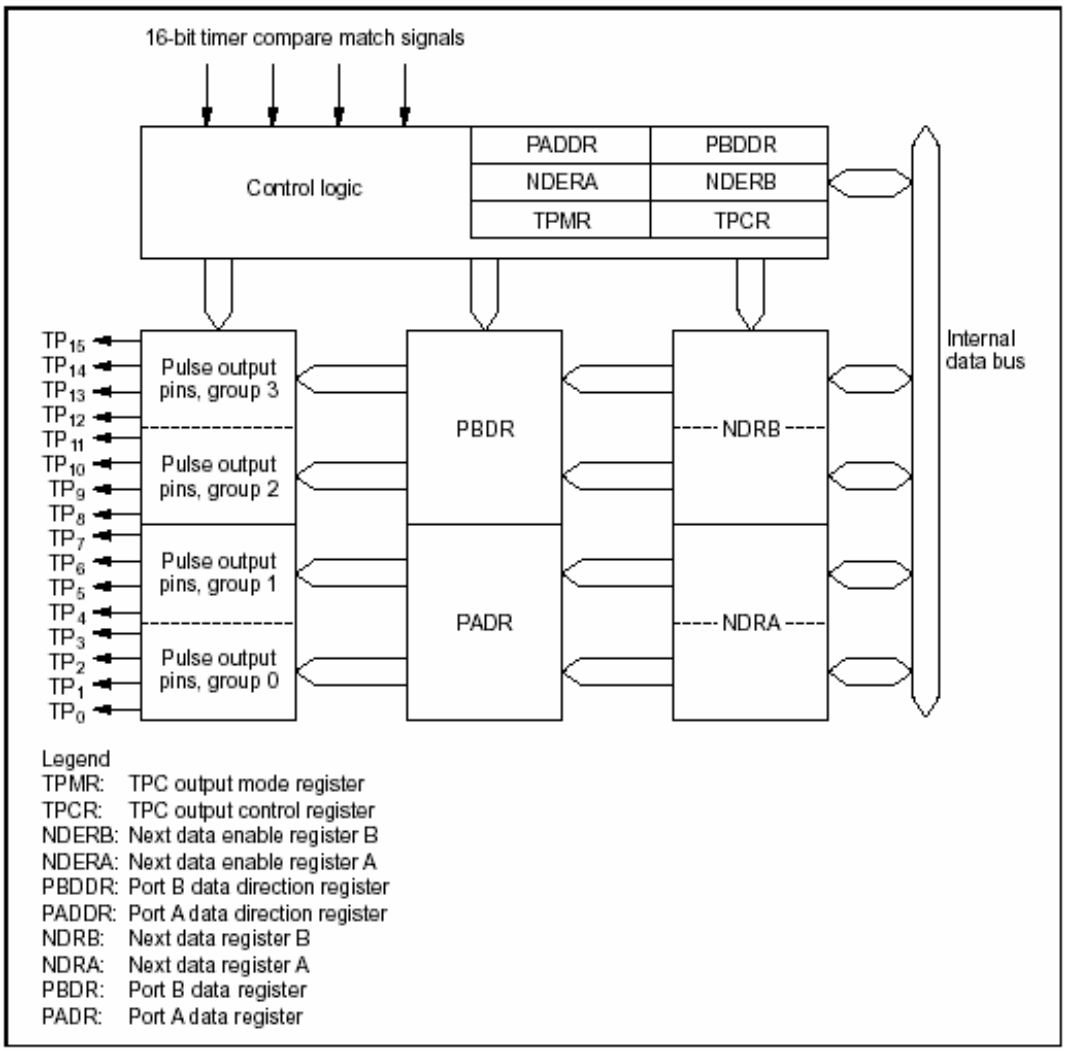


Figura 11. 1: Diagrama de bloques del TPC

[21] El TPC no posee llamada a interrupciones.

11.2 Registros de configuración

La figura 11.2 muestra todos los registros que se ven involucrados para la configuración y correcto funcionamiento de este subsistema, a continuación los describiremos.

Address*1	Name	Abbreviation	R/W	Function
H'EE009	Port A data direction register	PADDR	W	H'00
H'FFFD9	Port A data register	PADR	R/(W)*2	H'00
H'EE00A	Port B data direction register	PBDDR	W	H'00
H'FFFDA	Port B data register	PBDR	R/(W)*2	H'00
H'FFFA0	TPC output mode register	TPMR	R/W	H'F0
H'FFFA1	TPC output control register	TPCR	R/W	H'FF
H'FFFA2	Next data enable register B	NDERB	R/W	H'00
H'FFFA3	Next data enable register A	NDERA	R/W	H'00
H'FFFA5/ H'FFFA7*3	Next data register A	NDRA	R/W	H'00
H'FFFA4/ H'FFFA6*3	Next data register B	NDRB	R/W	H'00

Notes: *1 Lower 20 bits of the address in advanced mode.
*2 Bits used for TPC output cannot be written.
*3 The NDRA address is H'FFFA5 when the same output trigger is selected for TPC output groups 0 and 1 by settings in TPCR. When the output triggers are different, the NDRA address is H'FFFA7 for group 0 and H'FFFA5 for group 1. Similarly, the address of NDRB is H'FFFA4 when the same output trigger is selected for TPC output groups 2 and 3 by settings in TPCR. When the output triggers are different, the NDRB address is H'FFFA6 for group 2 and H'FFFA4 for group 3.

Figura 11. 2: Registros del TPC.

PxDDR: Port x Data Direction Register:

Como dijimos anteriormente, este subsistema utiliza los puertos A y B de modo que para su correcto funcionamiento debemos de configurar estos como salida (poniendo un uno) en los registros PADDR y PBDDR. Debemos mencionar que el puerto A corresponde a las salidas del grupo 0 y 1, de modo que abarca los pines TP_0-TP_7, y el puerto B es para los grupos 2 y 3 los cuales usan los pines TP_8 –TP_{15}.

Bit	7	6	5	4	3	2	1	0
	PA_7DDR	PA_6DDR	PA_5DDR	PA_4DDR	PA_3DDR	PA_2DDR	PA_1DDR	PA_0DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port A data direction 7 to 0
These bits select input or output for port A pins

Bit	7	6	5	4	3	2	1	0
	PB_7DDR	PB_6DDR	PB_5DDR	PB_4DDR	PB_3DDR	PB_2DDR	PB_1DDR	PB_0DDR
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port B direction 7 to 0
These bits select input or output for port B pins

Figura 11. 3: Registros PxDDR para el TPC.

PxDR: Port x Data Register:

En estos registros se almacena la variable que va a ser transmitida vía los puertos A y B. Cada vez que las salidas del TPC son activadas, el valor de la salida es transferido a este registro para que sea puesto en los pines correspondientes de los puertos. Para mayor información es recomendable ver el capitulo 4.

Bit	7	6	5	4	3	2	1	0
	PA_7	PA_6	PA_5	PA_4	PA_3	PA_2	PA_1	PA_0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Port A data 7 to 0
These bits store output data
for TPC output groups 0 and 1

Bit	7	6	5	4	3	2	1	0
	PB_7	PB_6	PB_5	PB_4	PB_3	PB_2	PB_1	PB_0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*	R/(W)*

Port B data 7 to 0
These bits store output data
for TPC output groups 2 and 3

Figura 11. 4: Registros PxDR para el TPC.

Nota: Estos registros son de lectura/escritura pero los bits que son seleccionados, por NDERx, para funcionar como parte del TPC cambian a ser los de lectura, la escritura para ellos se hará a través de los registros NDRx.

NDRx: Next Data Register x:

Este registro de 8 bits almacena el dato que va a ser puesto en la salida del TPC una vez se de el evento de comparación exitosa en el temporizador de 16 bits. Es decir que cuando se de el evento, el valor del este registro será transferido al PxDR correspondiente. Ahora, la dirección del registro NDRx depende de los grupos que se estén usando (0, 1, 2 o 3) y de si las señales de disparo serán diferentes para los grupos 0-1 y 2-3.

Por ejemplo: si la señal de disparo es la misma para los grupos 0-1 entonces la dirección del registro es la H´FFFA5 y los 8 bits serán puestos en la salida (los 4 menos significativos para el grupo cero y los 4 mas significativos para el puerto uno). Si los grupos tienen señales de disparo diferentes, tendremos dos direcciones, H´FFFA5 y H´FFFA7, para el grupo uno y el grupo cero respectivamente. En este ultimo caso, si se da el evento relacionado con el canal uno, la dirección a tomar en cuenta para el registro NDRx es la H´FFFA5 en donde los 4 bits mas significativos serán los de salida y los 4 menos significativos no serán tomados en cuenta (tendrán un valor de lectura reservado de uno). El mismo principio aplica si el evento es el relacionado con el del canal cero, la dirección seria H´FFFA7 y los valores de salida serian los 4 menos significativos.

Nota: En caso de tener la misma señal de disparo solo la H´FFFA5 es tomada en cuenta y la H´FFFa7 tendrá valores de lectura reservado en uno.

Esto mismo aplica con los grupos 2-3 con la única diferencia de la dirección de los registros. Las siguientes figuras ilustran un poco el fenómeno y las direcciones de los registros que deben ser usadas:

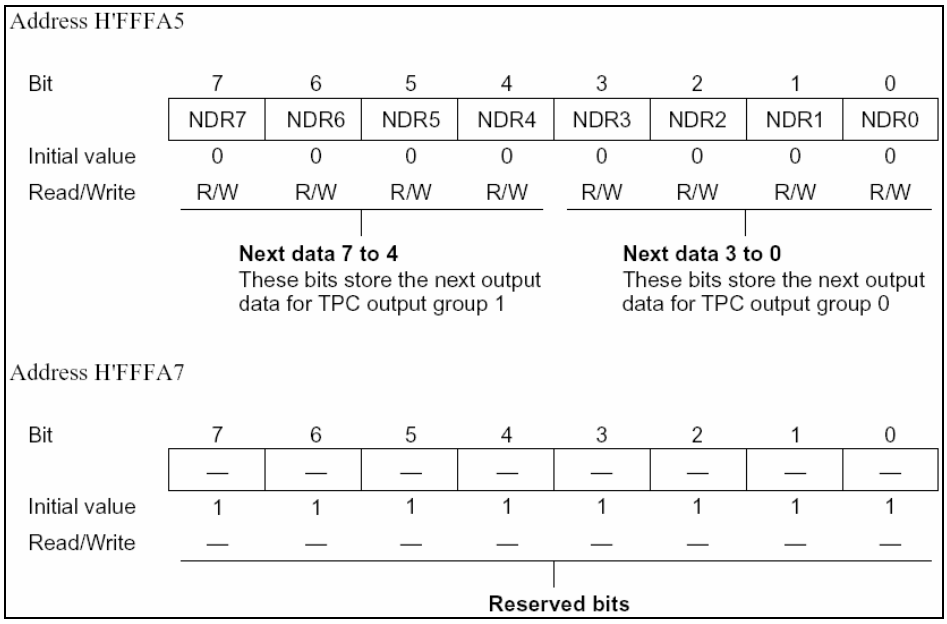


Figura 11. 5: Misma señal de disparo para los grupos 0-1.

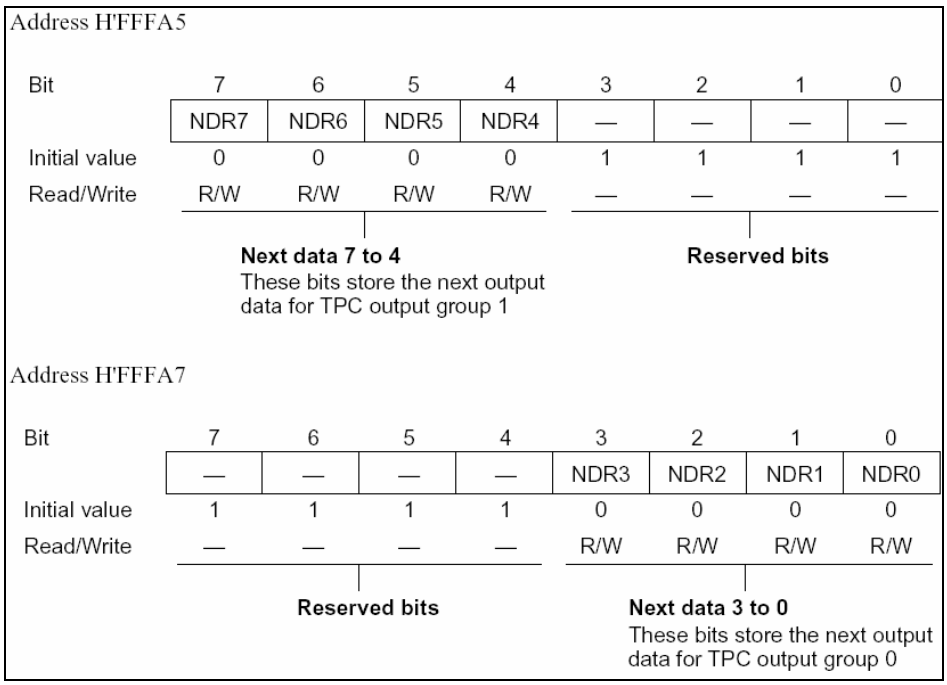


Figura 11. 6: Señal de disparo diferente para los grupos 0-1.

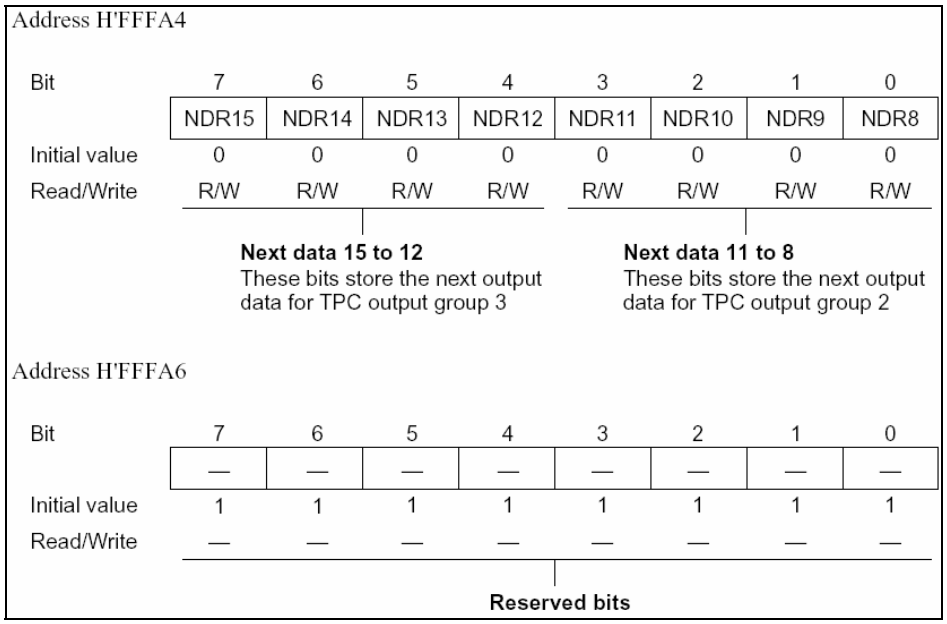


Figura 11. 7: Misma señal de disparo para los grupos 2-3.

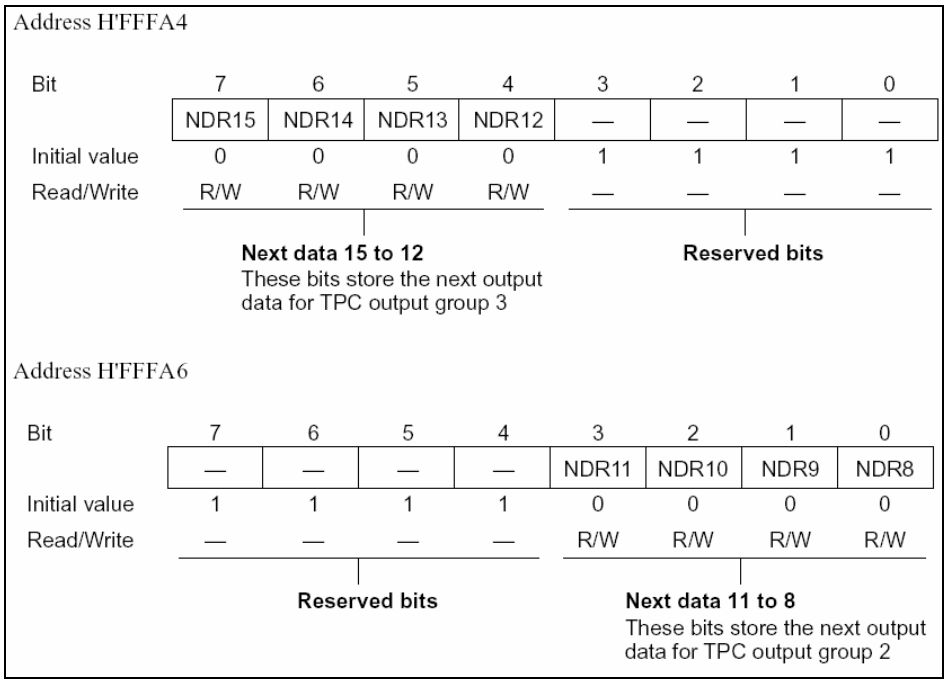


Figura 11. 8: Señal de disparo diferente para los grupos2-3.

NDERx: Next Data Enable Register x:

Este registro de 8 bits permite activar o desactivar bit por bit las salidas de TPC para los grupos 0-1 y 2-3. Al colocar un uno para el TP_0 y un cero para el TP_1 por ejemplo, al darse la comparación exitosa para el grupo cero, el dato del registro NDRA será transferido al PADR para sacarlo por el pin correspondiente. Como la señal de disparo es la misma para el TP_0 y el TP_1 es de pensar que el dato también será actualizado para el

pin que corresponde al $TP_{1,}$ pero esto no se da ya que esa salida no ha sido activada en el NDERA. En ese caso (cero en NDERx) los datos de NDRx son ignorados.

Bit	7	6	5	4	3	2	1	0
	NDER7	NDER6	NDER5	NDER4	NDER3	NDER2	NDER1	NDER0
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Next data enable 7 to 0
These bits enable or disable TPC output groups 1 and 0

Bit	7	6	5	4	3	2	1	0
	NDER15	NDER14	NDER13	NDER12	NDER11	NDER10	NDER9	NDER8
Initial value	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Next data enable 15 to 8
These bits enable or disable TPC output groups 3 and 2

Bits 7 to 0 NDER7 to NDER0	**Description**	
0	TPC outputs TP_7 to TP_0 are disabled ($NDR7$ to $NDR0$ are not transferred to PA_7 to PA_0)	(Initial value)
1	TPC outputs TP_7 to TP_0 are enabled ($NDR7$ to $NDR0$ are transferred to PA_7 to PA_0)	

Bits 7 to 0 NDER15 to NDER8	**Description**	
0	TPC outputs TP_{15} to TP_8 are disabled ($NDR15$ to $NDR8$ are not transferred to PB_7 to PB_0)	(Initial value)
1	TPC outputs TP_{15} to TP_8 are enabled ($NDR15$ to $NDR8$ are transferred to PB_7 to PB_0)	

Figura 11. 9: Registros de activación del TPC.

TPCR: TPC Output Control register:

Con este registro configuramos la señal de disparo que hará efectiva la activación de la salida para el TPC. Podemos ver que los cuatro grupos son totalmente independientes y pueden funcionar de forma simultánea.

Según sea la combinación de los bits correspondientes al grupo, así será dicha señal de disparo, las combinaciones posibles para un grupo se muestran en la figura 11.11 y el mismo principio se aplica para los demás grupos.

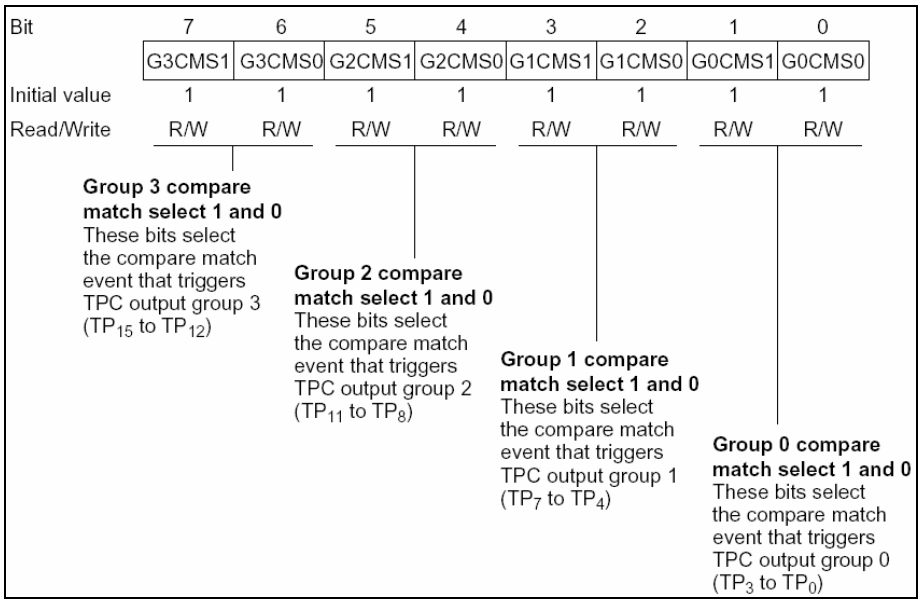


Figura 11. 10: TPCR para el TPC

Bit 7 G3CMS1	Bit 6 G3CMS0	Description
0	0	TPC output group 3 (TP_{15} to TP_{12}) is triggered by compare match in 16-bit timer channel 0
	1	TPC output group 3 (TP_{15} to TP_{12}) is triggered by compare match in 16-bit timer channel 1
1	0	TPC output group 3 (TP_{15} to TP_{12}) is triggered by compare match in 16-bit timer channel 2
	1	TPC output group 3 (TP_{15} to TP_{12}) is triggered by compare match in 16-bit timer channel 2 (Initial value)

Figura 11. 11: Configuración de la señal de disparo.

TPMR: TPC Output Mode Register:

Aquí se selecciona si la salida para cada grupo va a ser *Normal* o *Non-Overlapping*. Una salida normal indica que los valores de salida cambian cuando tenemos una comparación exitosa en el canal A del temporizador de 16 bits y la salida Non-Overlapping indica que un uno y un cero serán puestos como salida cuando tengamos una comparación en el canal A y B respectivamente.

A continuación ilustramos lo ante dicho para un solo grupo (grupo tres) pero esto mismo aplica para los demás:

Bit 3—Group 3 Non-Overlap (G3NOV): Selects normal or non-overlapping TPC output for group 3 (TP_{15} to TP_{12}).

Bit 3 G3NOV	Description
0	Normal TPC output in group 3 (output values change at compare match A in the selected 16-bit timer channel) (Initial value)
1	Non-overlapping TPC output in group 3 (independent 1 and 0 output at compare match A and B in the selected 16-bit timer channel)

Figura 11. 12: TPMR para el grupo tres.

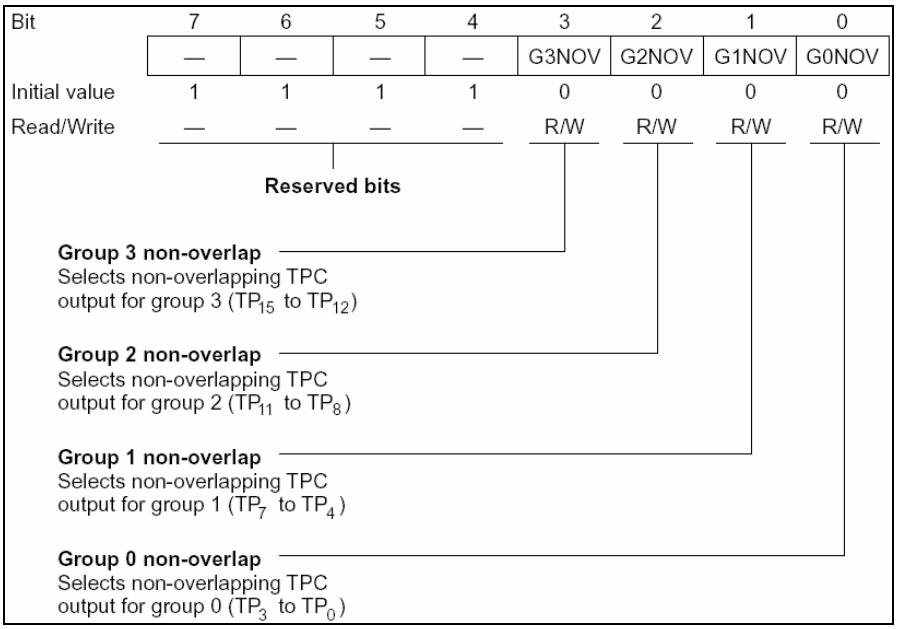


Figura 11. 13: TPMR para el TPC.

11.3 Operación

11.3.1 Condiciones de funcionamiento:

Para poder activar el TPC es necesario tener configurado los puertos o los pines deseados como salida (con el PxDDR) y activar el NDERx para los pines deseados, todo colocando un uno donde sea requerido. Es justo recalcar le dependencia del TPC con la configuración de los puertos, es por eso que presentamos la figura 11.14 en donde se muestra dicha relación:

NDER	DDR	Pin Function
0	0	Generic input port
	1	Generic output port
1	0	Generic input port (but the DR bit is a read-only bit, and when compare match occurs, the NDR bit value is transferred to the DR bit)
	1	TPC pulse output

Figura 11. 14: Condiciones de funcionamiento del TPC

11.3.2 Salida de TPC activada por Compare Match:

Al estar configurado correctamente y darse la condición de comparación, los datos que se encuentren en el NDRx serán transferidos al PxDR para poder ser sacador por el puerto. Es posible estar cambiando este valor de salida siempre y cuando lo hagamos antes que la próxima comparación exitosa se de, de modo que la salida sea actualizada en ese momento. Para ello es necesario que no este configurado como Non-Overlapping.

En la siguiente figura se muestra un ejemplo de la salida de los pulsos por los grupos 2 y 3 activada por la comparación exitosa en el canal A del temporizador. Podemos ver después de la señal de activación entregada por el temporizador tenemos la transferencia de datos del NDRB al PBDR y por consiguiente a los pines.

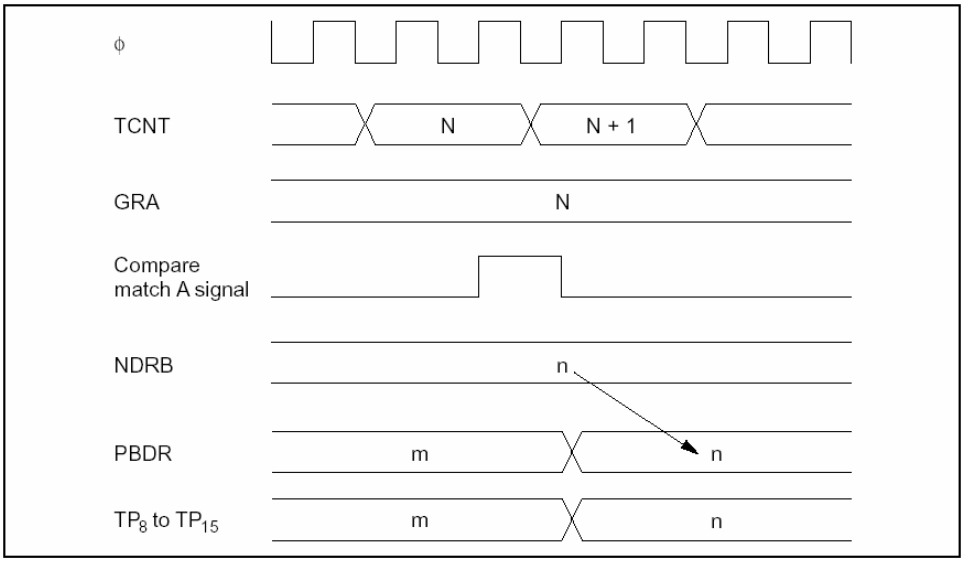


Figura 11. 15: Ejemplo de activación del grupo 2 y 3 por comparación en canal A.

11.3.3 Salida de TPC activada por Input Capture:

El TPC puede ser activado por un Input Capture así como es activado por el Output Compare. Lo único que necesitamos es que el canal del temporizador que se este empleando este configurado como Input Capture. El TPC recibirá siempre la señal del evento entregada por el temporizador y automáticamente, al siguiente pulso de reloj, tomará el valor de NDRx y lo pondrá en PxDR como se muestra a continuación:

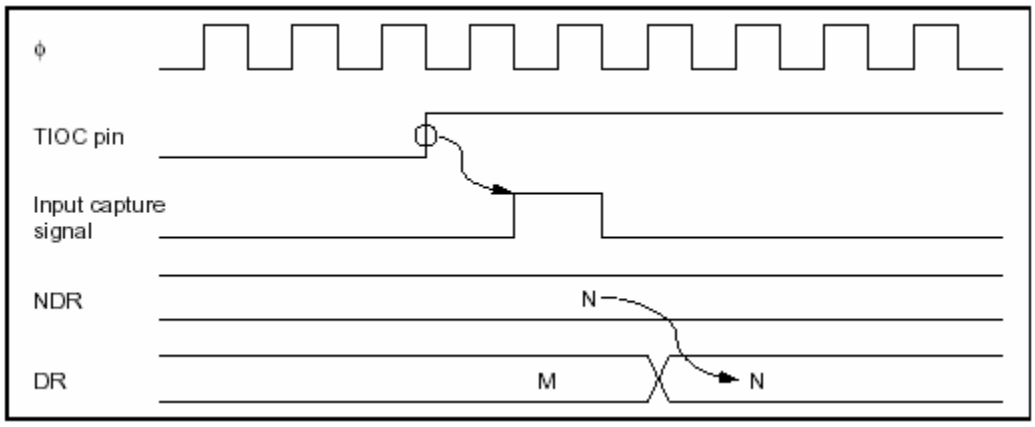


Figura 11. 16: Activación del TPC por Input Capture.

Capitulo Doce: Guías de laboratorio

Introducción:

Para poder completar la información teórica entregada, en forma de capítulos, con una formación práctica que le ayude al estudiante entrar en contacto con el dispositivo, se han creado una serie de guías prácticas. Con estas guías se pretende ilustrar, mediante ejemplos comentados, la configuración y utilización de cada subsistema de manera de orientar al estudiante en su primer contacto y encaminarlo a que sea capaz de crear posteriormente sus propias aplicaciones.

En el fólder anexo (llamado "Guías") se presentan las guías de laboratorio para cada uno de los subsistemas presentados y tratados a lo largo del trabajo de graduación junto con los diferentes ejemplos comentados[22].

Manual de laboratorio[23]:

Las guías que se encuentran en el manual de laboratorio son las siguientes:

Guía numero 2: Construcción del Sistema Mínimo.
Guía numero 3: Uso de los programas.
Guía numero 4: Uso de puertos
Guía numero 5: Temporizador de 16 bits.
Guía numero 6: Temporizador de 8 bits
Guía numero 7: Convertidor Análogo-Digital
Guía numero 8: Introducción al Convertidor DA.
Guía numero 9: Comunicación Serial.
Guía numero 10: Introducción al WDT.
Guía numero 11: Introducción al TPC

[22] La primera guía es la numero dos, la guía numero uno no existe. La nomenclatura se hizo de esta forma para que las guías lleven una correlación con la nomenclatura de los capítulos.
[23] Puede tener acceso al Manual de laboratorio a través de este vínculo.

Conclusiones:

Conclusiones del trabajo:

- Se presentó toda la información teórica y práctica que se consideró necesaria para que un estudiante, con conocimientos previos de microcontroladores, pueda aprender a utilizar el microcontrolador aquí presentado: H8-3069F.
  Dicha información consta de los Capítulos Teóricos, la introducción teórica incluida en cada guía, explicación del procedimiento de los ejemplos y los comentarios de cada uno de ellos.

- Se entregaron las dos etapas del sistema el cual estaba compuesto por el sistema mínimo y por el módulo de evaluación. Con dichos módulos se realizaron las prácticas de laboratorio con el grupo de estudiantes seleccionado por lo que se afirma que éste está en estado funcional.

- El hardware fue desarrollado para que pudiera trabajar tanto de manera unitaria como en conjunto, entendiéndose por esto que el sistema mínimo puede trabajar perfectamente solo o con los diferentes módulos acoplables.
  Por otro lado, el conjunto de módulos fueron desarrollados para que trabajen de forma adecuada con cada una de las aplicaciones de la guía y respondan a sus exigencias: Acoples de voltaje y protecciones,
  Pantallas con sus respectivos circuitos (LCD y Matriz de LED),
  LED para fácil visualización…

- En cuanto a elementos dañados en los impresos, estos son fácilmente remplazables debido a que la mayoría han sido montados sobre bases (en cuanto a integrados respecta) y para los elementos genéricos solo es necesario tener los conocimientos básicos de soldadura para poder remplazarlos. Ahora, por el espacio de acceso, el sistema trató de dejarse lo más modular posible de manera de facilitar el cambio de piezas.

- Las prácticas se realizaron tanto con un grupo de estudiantes con conocimientos previos de microcontroladores así como con uno que no los tenía. Los resultados fueron totalmente diferentes, el primer grupo tuvo una mejor absorción de la información mientras que el segundo presentó dificultades con los conocimientos básicos. Esto implica que para poder tener un mejor resultado es necesario que los estudiantes tengan un conocimiento previo del tema o estén cursando la materia de microcontroladores de manera a contar con las bases y el apoyo necesario.

- En el mercado es posible encontrar dispositivos similares al que se ha creado (sistema mínimo) pero muy pocos presentan la modularidad que el nuestro incluye (sistemas de evaluación).
  Podemos ver que si solo se deseara utilizar el sistema mínimo, este puede ser adquirido por unos $35-$40 en cualquier tienda especializada de electrónica (en

estados Unidos o Japón) pero se tendría la gran limitante que los estudiantes no tendrían conocimiento alguno sobre el equipo con el que están trabajando. Ahora, la solución que se plantea sobre la creación del sistema mínimo lleva dos objetivos: Que los estudiantes conozcan a fondo el sistema con el cual trabajan para que en alguna eventualidad sepan como repararlo y que apliquen los conocimientos adquiridos a lo largo de la carrera al momento de la fabricación.

Si bien es cierto que la solución que se está ofreciendo puede resultar un poco más costosa que la comercial, sus beneficios educativos y académicos justifican la inversión.

- La inversión total de los módulos del sistema, tanto los dos sistemas mínimos como el modulo didáctica, fue de $160 aproximadamente. El detalle de los gastos figura en el capitulo dos del trabajo de graduación.

Conclusiones del dispositivo:

- La velocidad de operación y juego de instrucciones del Microcontrolador, permite la realización de tareas mucho más complejas y precisas que los microcontroladores que tenemos en el mercado regional en la actualidad.

- Su amplia gamma de subsistemas lo hacen una herramienta poderosa para la ingeniería electrónica. Internamente el microcontrolador cuenta con más de 10 subsistemas, incluyendo los nuevos, lo cual es superior a la cantidad de subsistemas en los microcontroladores de uso regional y convencional.

- Una de las características por las cuales este dispositivo se vuelve una atractiva herramienta, es precisamente por su característica de máxima explotación de recursos, pues las subdivisiones que se encuentran en cada subsistema nos permite un amplio margen de aplicación no solo una vez sino que varias veces. Un ejemplo claro de ello es el número de canales que tienen los módulos de temporización y de puertos, entre otros.

- La arquitectura interna del microcontrolador, permite el uso de instrucciones para manejo de palabra doble (32 bits), esto sin duda es una gran ventaja puesto que el manejo de operaciones matemáticas o de cálculo se vuelven hasta cierto punto más fácil.

- Instrucciones como la de multiplicación y división son incluidas en el microcontrolador lo cual facilita que rutinas de cálculo en las cuales en el pasado debían ser efectuadas por subrutinas, ahora se realizan de manera directa.

- Los diversos modos de operación de la CPU (cambio de bancos de memoria) permite una amplia gama de alternativas para la programación de manera que es

posible acomodándose más fácilmente a la necesidad que el programador requiera para su aplicación.

- El controlador de bus del cual se hace mención, aunque no es estudiado a profundidad en este documento, es una aplicación útil para el manejo de dispositivos externos como memorias. Si tenemos en cuenta el uso del DMAC y la máxima capacidad en memoria externa que se puede direccionar (16MB en modo extendido) vemos que el dispositivo se convierte en un dispositivo de gran capacidad de memoria.

- Un nuevo subsistema empleado llamado TPC o control de patrones de tiempo, permite la emulación de sistemas secuénciales o PWM arbitrario, así como también la aplicación de un controlador de patrones de tiempo hacia los puertos A y B. Este tipo de subsistema permite el manejo de dispositivos externos que requieran el control con entradas sincrónicas o asíncronas (como memorias, FLIP FLOP, etc.). Esta actividad se hace sin forzar la CPU puesto que es un subsistema aparte y lo único que se requiere es la configuración de los registros.

- Siete interrupciones externas, 36 internas y tres niveles seleccionables de la prioridad de interrupciones, realizan un control casi total de los diversos, subsistemas del microcontrolador.

- Para el control digital de señales la opción de muestreo y retensión de datos realizada por el temporizador de 8 bits y el subsistema AD puede ser activada configurando únicamente los registros. La lectura de datos se realiza de manera que tal que la CPU se esfuerza en lo más mínimo para llevar a cabo esta función. Esto optimiza un sistema de control digital.

- Subsistemas como el convertidor DA pueden generar funciones de salida especiales programadas desde el compilador, el hacer un generador de funciones desde el microcontrolador se vuelve una tarea realizable.

- Las herramientas proporcionadas para la programación incluyen compilación en C, C++ y ensamblador, lo cual permite un amplio desarrollo de aplicaciones, tomando como base de diseño la programación a bajo nivel y/o alto nivel.

Recomendaciones:

1. Para poder tener un mejor resultado de aprendizaje será necesario que los estudiantes que utilicen este sistema e información tengan un mínimo de conocimiento de microcontroladores o que estén cursando la materia en ese momento. De esta manera se asegura que el centro de atención sea cómo utilizar los subsistemas en este nuevo microcontrolador y no en sí la explicación del subsistema en general (para el caso de los elementos comunes).

Para el caso de los subsistemas totalmente nuevos, una buena base sobre el tema será más que suficiente para relacionarse con el sistema e ir aprendiendo junto con la información y los ejemplos dados.

2. Se debe procurar tener un voltaje adecuado para la alimentación de los reguladores de voltaje que alimentan el sistema mínimo y la tarjeta de evaluación. Para el primero es necesario que a la salida del regulador se tengan 5V de manera que en la entrada es necesario tener un voltaje superior: entre 6V y 10V estará bien. De igual forma para el voltaje de alimentación del regulador del módulo de entrenamiento, éste debe ser de al menos 15V de manera que entregue los 12 voltios deseados.

3. Físicamente, para el puerto A (pines del 25 al 32) se ha dejado una conexión a un buffer TTL con opción para configuración IN/OUT y después de pasar por éste, los datos pasan a los conectores de 40 pines.
   Para evitar el riesgo de arruinar algunos pines de este puerto, se recomienda siempre dejar el Jumper que controla la opción de IN/OUT en la posición OUT a menos que se vaya a hacer uso de la opción IN. Esto debido a que si se configura el puerto como salida y el buffer como entrada, habrá una colisión de valores y el resultado podría ser impredecible e indeseable.
   La ubicación del Jumper del buffer la podemos ver en la figura R.1.

4. La entrada tratada por el operacional para la función de conversión AD debe de tener un voltaje de entrada máximo de 10 voltios debido a que si el voltaje entregado, por el divisor de voltaje, al pin de entrada analógica excede los 5.8 voltios (limite planteado en las características eléctricas del dispositivo – sección 21 del manual de hardware -) se corre el riesgo de arruinar el pin de entrada.

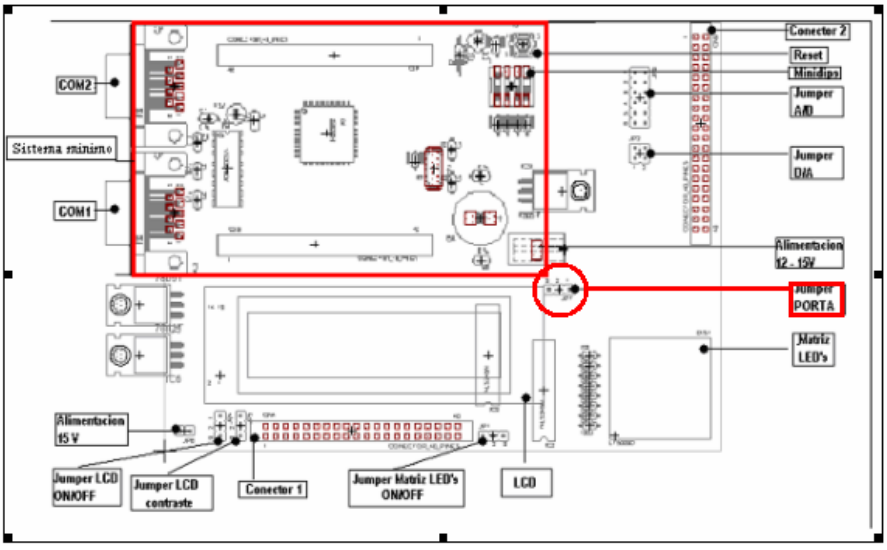


Figura R. 1: Ubicación del Jumper del puerto A.

5. Cabe la posibilidad de no querer utilizar el software que se ha recomendado debido a las limitantes de ser una versión de demostración, por lo que se proporcionan ciertos enlaces en donde podemos encontrar software similar. Debemos mencionar que para dichos software los programas no han sigo probados y además existe la posibilidad que los comandos o librerías varíen un poco.
   Los enlaces son los siguientes:

   - http://www.htsoft.com/products/H8300c.php (Pagado)
   - http://www.kpitgnutools.com/index.php (GNU)
   - GNU Dev Tools for the Hitachi H8/300[HS]: http://www.freebyte.com/programming/assembler/ (Libre)
   - http://sourceforge.net/project/showfiles.php?group_id=24580
   - http://sourceforge.net/project/showfiles.php?group_id=24580
   - http://www.gnuh8.org.uk/links.htm#compilers
   - http://www.renesasrulz.com/index.php?name=Web_Links&req=viewlink&cid=1

Bibliografía:

Manual de Hardware:

[B1] "Hitachi Hardware Manual for H8/3069 F-ZTAT™."
"Hardware Manual H8-3069F" (Renesas)

Artículo de aplicación y descripción del microcontrolador:

[B2] "Hitachi Release 16-Bit Microcontroller with On-Chip Large Flash Memory."

Revista de Renesas:

[B3] "Special Features: Flash Microcomputers," Renesas EDGE, vol. 11, pp. 02-05, 2006.

Manual de programación de la familia H8:

[B4] "H8 Programming Manual."

Líneas guías y recomendaciones para la fabricación de circuitos impresos:

[B5] "Printed Circuit Board layout Guidelines, Software, Trace Calculator, Tutorials, Assembly Resources": http://www.smps.us/pcb-design.html.
[B6] "Manual de seguridad para operaciones en actividades electrónicas": http://www.sprl.upv.es/mselectronica1.htm
[B8] "Printed Circuits Board"

Cursos interactivos proporcionados por Renesas:

[B7] "Renesas Interactive Course": www.renesasinteractive.com

Manual de usuario para el programa HEW y el FDT:

[B9] Hitachi Embedded Workshop
[B10] Renesas FLASH Development Toolkit

Hojas técnicas y cursos sobre los dispositivos utilizados:

[B11] Hoja técnica de HD44780U (LCD)
[B12] Curso sobre el manejo de LCD
[B16] Matriz de LED LT5003D

Explicación de protocolos y subsistemas:

[B13] WatchDog Timer (Santa Maria)
[B14] WDT (Microship)
[B15] Handshaking Notes:
http://www.cl.cam.ac.uk/~djg11/wwwhpr/fourphase/fourphase.html

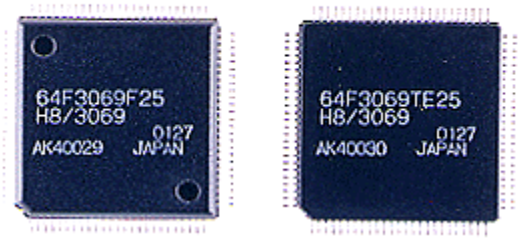
Trabajo de graduación (Referencia para las guías de laboratorio)

Diseño e implementación del prototipo de un módulo de aprendizaje para el uso de un microcontrolador H8/3069F (H8/300H Core) fabricado por Renesas y sus guías de desarrollo.

Anexos[24]

1. Hitachi Releases 16-Bit Microcontroller with On-Chip Large Flash Memory

-- Suitable for control of storage devices such as CD-R/RW and DVD-ROM/RAM, and incorporating large 512- KByte on-chip flash memory for shorter development time –



16-bit microcontroller “H8/3069F” with 512-Kbyte flash memory

Este es el encabezado del artículo que se encuentra en la página http://www.hitachi.com/New/cnews/E/2001/0802/index.html. Dicho artículo puede ser entregado en versión PDF por los estudiantes, de ser solicitado.

[24] Los demás anexos han sido presentados en la carpeta de anexos.

2. Renesas EDGE Vol. 11

Esta es la portada de la revista de Renesas: EDGE volumen 11 publicada en 2006 en donde aparece el articulo *Flash Microcomputers* de donde se extrajo la información de dos de las compañías que están usando este Microcontrolador en la actualidad: *Toyota Motor Corporation* y *Nikon Corporation.*

3. Tabla de objetivos educacionales[25]:

1. Conocimiento – Repetir la información de manera integra. [Ejemplo: Liste los pasos en el procedimiento de calibración de un cronógrafo de gas.]

2. Comprensión – Demostrar el conocimiento de los términos, conceptos y principios. [Ejemplo: Explicar con sus propias palabras el concepto de la presión de vapor. Interprete la forma de la onda de salida.]

3. Aplicación -- Aplicando conceptos y principios para la resolución de problemas. [Ejemplo: Calcule la probabilidad que dos muestras sean diferentes por mas del 5%. Resuelva la ecuación con los valores dados.]

4. Análisis -- Reducir al menor elemento, formular explicaciones teóricas o modelos matemáticos o lógicos parra el fenómeno observado. [Ejemplo: Interprete las diferencias entre las respuestas teóricas y experimental para determinado evento.]

5. Síntesis – Crear combinando elementos. [Ejemplo: Formule un algoritmo de control basado en los procesos estudiados en el laboratorio.]

6. Evaluación – Juzgar el valor de las opciones, escogiendo la mejor alternativa y justificando la elección. [Ejemplo: De las opciones disponibles, escoja la mejor para medir la respuesta del sistema y justifique su respuesta.]

[25] Extraído de ***A Colloquy on Learning Objectives For Engineering Education Laboratories***