



22
21

**UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO**

**FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN**

**"CAPTURA DE EVENTOS PARA UN
TURBOGENERADOR DE 300 MW DEL SISTEMA
CSCCT (CONTROL SUPERVISORIO CICLO
COMBINADO TULA) CON
MICROCONTROLADORES"**

T E S I S
QUE PARA OBTENER EL TITULO DE
INGENIERO MECANICO ELECTRICISTA

P R E S E N T A

HECTOR BECERRA HERNANDEZ

ASESOR: ING. JORGE BUENDIA GOMEZ

CUAUTITLAN IZCALLI, EDO. DE MEX.

1997

**TESIS CON
FALLA DE ORIGEN**



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
 UNIDAD DE LA ADMINISTRACION ESCOLAR
 DEPARTAMENTO DE EXAMENES PROFESIONALES

UNIVERSIDAD NACIONAL
 AVENIDA DE
 MEXICO

ASUNTO: VOTOS APROBATORIOS

DR. JAIME KELLER TORRES
 DIRECTOR DE LA FES-CUAUTITLAN
 P R E S E N T E .

AT'N: Ing. Rafael Rodríguez Ceballos
 Jefe del Departamento de Exámenes
 Profesionales de la F.E.S. - C.

Con base en el art. 28 del Reglamento General de Exámenes, nos permitimos comunicar a usted que revisamos la TESIS:

"Captura de Eventos para un Turbogenerador de 300Mw del

Sistema CSCOT (Control Supervisorio Cielo Combinado TuLa)
con Microcontroladores".

que presenta el pasante: Réctor Becerra Hernández

con numero de cuenta: 8501085-3 para obtener el TITULO de:
Ingeniero Mecánico Electricista

Considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VOTO APROBATORIO.

A T E N T A M E N T E .

"POR MI RAZA HABLARA EL ESPIRITU"

Cuatitlan Izcalli, Edo. de Mex., a 14 de Abril de 1997

PRESIDENTE	<u>Ing. Antonio Herrera Mejía</u>
VOCAL	<u>Ing. José Luis Rivera López</u>
SECRETARIO	<u>Ing. Jorge Buendía Gómez</u>
PRIMER SUPLENTE	<u>Ing. Ubaldo Ramírez Urizar</u>
SEGUNDO SUPLENTE	<u>Ing. Margarita López López</u>

[Handwritten signatures and dates]
29/06/97
25/07/97
060597

Con cariño:

A mis padres,

hermanos,

y novia

Con Atención especial:

Al Ing. Sergio Arguelles Martínez
y a todas las personas que
contribuyeron a la realización de
este proyecto

CONTENIDO

INTRODUCCION	I
I. PROBLEMA EN LA PRODUCCION DE ENERGIA ELECTRICA EN UNA CENTRAL TERMoeLECTRICA.	
1.1 Generalidades	1
1.2 Panel de alarmas	1
1.3 Registrador de eventos	2
II. ESCALAMIENTO DE SEÑALES	
2.1 Generalidades	3
2.2 Etapa escaladora de señal 1200A/110AC	6
III. ACONDICIONAMIENTO DE SEÑALES	
3.1 Generalidades	7
3.2 Filtro de corrientes transitorias	9
3.3 Detector de nivel con histeresis	10
3.4 Niveles logicos de voltaje	13
3.5 Utilidad del receptor en el HCPL-2730 y de la compuerta OR-EXC. para el simulador de eventualidades como niveles logicos de voltaje	14
IV. CONVERTIDOR DIGITAL-ANALOGICO	
4.1 Generalidades	16
4.2 Necesidad de multiplexar los niveles logicos de las eventualidades en un turbogenerador	16
4.3 Necesidad de utilizar un convertidor Digital-Analógico para transmision de señales	20
4.4 Convertidor Digital-Analógico	22
4.5 Resultados obtenidos en la experimentacion con el DAC-65	23
V. EL MICROCONTROLADOR 68HC11 Y EL PUERTO E	
5.1 Generalidades	23
5.2 El microcontrolador 68HC11	26
5.3 Sistema de control del convertidor A/D en el puerto E	27
5.4 Los registros en el microcontrolador 68HC11	28
5.5 La memoria en el microcontrolador 68HC11	31
5.6 Programacion de dispositivos entrada/salida	32
VI. DISEÑO DEL PROGRAMA PARA CAPTURA DE EVENTOS EN EL MICROCONTROLADOR 68HC11E9	
6.1 Generalidades	35
6.2 Simulacion de un MTS con el microcontrolador	38
6.3 Simulacion de Hardware registrador de secuencias	41
6.4 Subrutinas de inicio y control	43

6.5	Programa principal y la subrutina CLARO	45
6.6	Resultados obtenidos	46
6.7	Subrutina TIMER	47
6.8	Subrutinas SUBSEG y SUBMIL	48
6.9	Subrutina SUBHRS	50
6.10	Conversiones de código binario a BCD en TIMER	51
6.11	Programa BUFFALO en el 68HC11E9	54
6.12	Subrutina OUTSTRG en TIMER	55
6.13	Inicialización del reloj	55
VII. PRESENTACION DE RESULTADOS		
7.1	Generalidades	56
7.2	Presentacion en el display 1720A	57
7.3	Presentacion en un monitor de computadora	58
CONCLUSIONES		61
APENDICE A		
Listados		63
APENDICE B		
Tablas de registros e instrucciones del 68HC11E9		73
BIBLIOGRAFIA		93

INTRODUCCION

La central termoeléctrica "Fco. Pérez Ríos" ubicada en el Estado de Hidalgo. Es una de las plantas generadoras de energía eléctrica más importante en nuestro país.

La generación de energía eléctrica en una central de este tipo se basa en el aprovechamiento de la capacidad calorífica de un combustible fósil llamado combustóleo para llevar a cabo un trabajo que finalmente se ve reflejado en potencia eléctrica.

El trabajo que se realiza es el siguiente:

- 1) Cambiar el estado del agua de líquido a vapor.
- 2) Convertir la energía calorífica del vapor a energía cinética en los alabes de una turbina.
- 3) Convertir la energía cinética rotativa en el eje de una turbina a potencia eléctrica a través de un generador.

Una turbina cuyo eje lleva acoplado un generador eléctrico recibe el nombre de Turbogenerador.

En los últimos años con la demanda de energía eléctrica la central termoeléctrica "Fco. Pérez Ríos" decidió aprovechar la energía calorífica de los gases liberados a la atmósfera en los procesos termodinámicos gas-vapor. Este aprovechamiento de energía se llevó a cabo en una sección de la planta que desde entonces se conoce como "Sección Ciclo Combinado".

Un Turbogenerador es una máquina de grandes dimensiones y sumamente costosa. Debido a esto, están provistas de un sistema de protección. Ya que de lo contrario resultaría contraproducente llegar a una reparación o sustitución de las mismas.

El sistema de protección se encarga poner fuera de funcionamiento al Turbogenerador cada vez que se detecta una sobrecorriente en los devanados del generador eléctrico.

Las sobrecorrientes son producidas por eventos tales como Alta vibración del compresor. Alta temperatura en las chumaceras. Baja presión de aceite de lubricación. Muy alto movimiento axial del compresor. Sobrevelocidad del compresor. Paro de turbina, etc.

El departamento de "Control Supervisorio" de cualquier planta termoeléctrica se encarga de estudiar, analizar y prevenir estos eventos.

La ingeniería electrónica se ha desarrollado en este campo con la utilización de registradores de eventos. Estos sistemas electrónicos capturan en su memoria las eventualidades ocurridas durante la operación de una máquina en particular.

En la aeronáutica, un registrador de eventos nos puede proporcionar la información necesaria para saber la causa de algún siniestro en una nave aérea. Aun cuando no haya habido sobrevivientes.

Hace algunos años, la central termoeléctrica "Fco. Pérez Ríos" adquirió un registrador de eventos patentado por la compañía Jhon Fluke, MFG. CO. INC. Este sistema electrónico se instaló en la sección Ciclo Combinado. Desafortunadamente, el controlador 2400A-104/AA y la unidad lógica de procesamiento

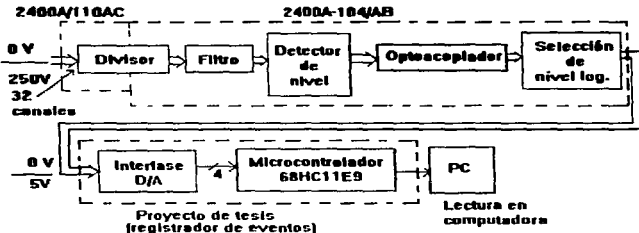
(Mainframe 2400A) del sistema dejaron de funcionar.

Mi labor de investigación en el departamento de control entre otras actividades fué la de encontrar una solución a este problema. Por la responsabilidad que implica solución del mismo, surge la necesidad de una propuesta de tesis.

El método a seguir es el siguiente:

- 1) Multiplexar las señales que recibía el controlador 2400A-104/AA para enviarlas al microcontrolador 68HC11.
- 2) Programar al microcontrolador 68HC11 para captura e identificación de la información enviada.
- 3) Sustituir a la unidad lógica de procesamiento (Mainframe 2400A) por una computadora personal PC.
- 4) Diseñar un programa en la computadora que pueda interpretar información enviada a través del puerto serie.

Los pasos a seguir pueden apreciarse claramente en la siguiente figura:



Registrador de eventos basado en el microcontrolador 68HC11E9 adaptado a la etapa acondicionadora de señal de un turbogenerador de 300MW

El capítulo 1 explica el problema que existe en una central termoelectrónica cuando ocurre una falla en el turbogenerador y no se cuenta con un registrador de eventos.

El capítulo 2 describe el escalamiento de señales que se lleva a cabo en la etapa 2400A/110AC en el registrador de eventos patentado por la compañía Jhon Fluke, MFG. CO. INC. para 32 señales de voltaje producidas por los eventos de un turbogenerador.

El capítulo 3 describe el acondicionamiento de las señales en las etapas 2400A-104/AB en el registrador de eventos patentado por la compañía Jhon Fluke MFG. CO. para las señales enviadas por el divisor 2400A/110AC.

El capítulo 4 trata sobre la transferencia de datos implementada con los convertidores DAC-08.

El capítulo 5 se refiere al microcontrolador 68HC11 y a la forma de operación del puerto E cuando funciona como convertidor analógico-digital.

El capítulo 6 trata sobre el diseño del programa en lenguaje ensamblador que utiliza el microcontrolador para la captura y registro en tiempo real de los eventos.

El capítulo 7 muestra algunas herramientas disponibles en el lenguaje de programación QBASIC para la comunicación con dispositivos conectados a la interfaz RS-232. También, se expone el algoritmo diseñado en este lenguaje para interpretar la información registrada por el microcontrolador una vez que ha sucedido una falla en el turbogenerador.

CAPITULO I. PROBLEMA EN LA PRODUCCION DE ENERGIA ELECTRICA EN UNA CENTRAL TERMOELECTRICA

1.1 GENERALIDADES

El problema básico en cualquier industria productiva es el de mantener un nivel de productividad mínimo necesario para satisfacer las demandas del mercado.

Cuando no se logra ser lo suficientemente productivos. Se generan pérdidas económicas que resultan ser sumamente cuantiosas.

La Comisión Federal de Electricidad es la única industria en México encargada de producir energía eléctrica. Su nivel de productividad excede las demandas de consumo diario en nuestro país.

Sin embargo, existen pérdidas económicas debidas a un mayor consumo de combustible en las centrales termoeléctricas. Lo que se manifiesta cuando alguna de las mismas no está produciendo la suficiente potencia.

La pérdida total ó parcial de potencia en una central termoeléctrica generalmente viene precedida por la inactividad en algún turbogenerador.

Las pérdidas son mínimas cuando se logra un arranque rápido en el turbogenerador. Pero no siempre sucede así. En muchas ocasiones los arranques pueden demorarse hasta 24 Hrs. cuando se desconoce por completo el "origen del disparo" (conjunto de eventualidades previas al paro del turbogenerador).

Para prevenir un "disparo" (paro del turbogenerador) se hace necesario tener un registro de los eventos previos.

1.2 PANEL DE ALARMAS

Las salas de control instaladas en cada central termoeléctrica cuentan con "paneles de alarmas" (Ver fig. 1.1). a cargo de un "operador" para tener un registro de eventos.

Un operador es la persona encargada de notificar las eventualidades observadas en un panel de alarmas.

En muchas ocasiones, el operador no está presente al momento de suceder los eventos, por lo que el "disparo de unidad" (paro del turbogenerador) es inevitable.

Otra desventaja que se presenta un panel de alarmas es la poca confiabilidad de sus componentes. Ya que los focos en los anunciadores son susceptibles de fundirse. Por otra parte, el cableado y la forma de transmisión de las señales representa un problema continuo.

Se cuenta con "probadores de alarmas" que ayudan a la reparación y mantenimiento de los paneles. Pero el problema no termina por solucionarse.

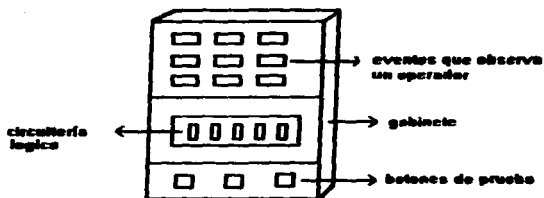


Fig. 1.1 Panel de alarmas a cargo de un operador

1.3 REGISTRADOR DE EVENTOS

Un "registrador de eventos" (Ver fig.1.2) es un sistema electrónico que registra en una memoria una serie de datos digitales relacionados con una secuencia de eventos. Sus características de operación dependen de cada fabricante. El registro lo efectúan repetidamente en periodos rápidos de tiempo (milésimas se segundo). Están constituidos por un circuito escalador de señal, una interfase acondicionadora de señal y un controlador.

Un registrador de eventos aplicado a las necesidades de la industria eléctrica proporciona las siguientes ventajas:

- 1) No requiere de un "operador" que observe un panel de alarmas y notifique las eventualidades.
- 2) Es un sistema electrónico implementado con un controlador ó microcontrolador que por sus características resultan ser sumamente confiables.
- 3) En caso de suceder un "disparo" se puede conocer su origen con toda precisión.
- 4) El arranque de un turbogenerador se puede llevar a cabo de forma inmediata.
- 5) Las pérdidas económicas en la producción de energía eléctrica se reducen a un mínimo.

Las desventajas que se tienen con un sistema electrónico de este tipo es en cuanto al número de señales que pueden procesarse con el controlador ó microcontrolador. Ya que los puertos de

entrada son limitados. Sin embargo, debemos señalar que un registrador de eventos es muy útil cuando es instalado en lugares estratégicos. Es decir, en las áreas de mayor interés en una central termoelectrónica.

Por otra parte, es posible tener una estadística del comportamiento de las eventualidades a largo plazo ocurridas en un turbogenerador mediante el uso de los archivos de eventos cotidianos en computadora. De tal manera que los "disparos de ciudad" tengan lugar en periodos más prolongados de tiempo mediante un análisis exhaustivo de la información estadística obtenida.

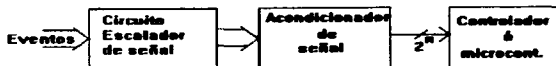


Fig. 1.2 Arquitectura de un registrador de eventos

CAPITULO II. ESCALAMIENTO DE SEÑALES

2.1 GENERALIDADES

Cuando hablamos de realizar un dibujo a "escala" nos referimos a hacer una representación gráfica de un objeto real con dimensiones diferentes. Las dimensiones de un dibujo a escala pueden ser mayores o menores que las del objeto en cuestión, pero siempre en forma proporcional.

Por ejemplo, una cámara fotográfica reduce las dimensiones de un objeto real a unos cuantos centímetros en un retrato, sin modificar en lo absoluto la imagen que percibimos al observar dicho objeto. En este caso, podemos decir que se ha hecho un "escalamiento de dimensiones".

En electrónica, un "escalamiento de señal" puede definirse como el aumento o disminución en forma proporcional de los valores que presenta una variable física.

Los valores de corriente y voltaje en una señal pueden disminuir o aumentar en forma proporcional a través de los devanados de un transformador eléctrico.

2.1.1 RELEVADORES

En la industria eléctrica existen muchos dispositivos electromecánicos de gran utilidad. Toda máquina electromecánica está provista de un arrollamiento eléctrico por el cual circula una corriente que produce una fuerza electromotriz utilizada para realizar un trabajo.

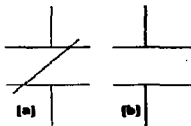
Un relevador es dispositivo electromecánico formado por una bobina eléctrica y dos laminillas de material conductor. Cuando la bobina es excitada, las laminillas tienden a unirse o

separarse entre sí.

En condiciones normales (no hay excitación en la bobina), los relevadores pueden ser de dos tipos (Ver fig. 2.1)

- 1) De contactos normalmente abiertos
- 2) De contactos normalmente cerrados

En ambos casos, la fuerza mecánica ejercida por un resorte se encarga de mantener separadas ó unidas a las laminillas. La fuerza electromotriz creada por la bobina tiene un efecto inverso al producido por la acción del resorte. Un relevador de contactos normalmente cerrados requiere mayor corriente de excitación en su bobina para cambiar de estado las laminillas que un relevador de contactos normalmente abiertos. (Ver Tabla 1).



**Fig. 2.1 (a) Contactos normalmente cerrados
(b) Contactos normalmente abiertos**

2.1.2 UTILIDAD DE UN RELEVADOR EN LAS SEÑALES DE EVENTOS

Un sistema de protección a base de relevadores se encarga de poner fuera de funcionamiento a un turbogenerador cada vez que se detecta una sobrecorriente en los devanados del generador eléctrico.

Las sobrecorrientes son producidas por eventos tales como Alta vibración del compresor. Alta temperatura en las chumaceras. Baja presión de aceite de lubricación, etc.

Si la bobina en un relevador se excita con una sobrecorriente producida por algún evento, entonces el cambio de estado en las laminillas nos indicará que existe un problema en los devanados del generador eléctrico.

Por otra parte, es posible generar una señal de voltaje para cada evento si conectamos una batería en serie con un relevador (ver fig. 2.2).

En consecuencia, un sistema de protección a base de relevadores puede ser conectado a un registrador de eventos (Ver fig. 2.3) para análisis, estudio y prevención de las eventualidades.

Estado del contacto	Voltaje aplicado
Cerrado	Señales de entrada superiores a 60V
Abierta	Señales de entrada inferiores a 40V

Tabla 1. Relación entre señales de entrada y el estado de los contactos

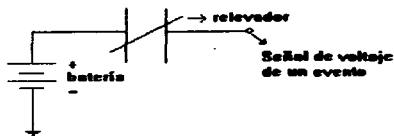


Fig. 2.2 Conexión serie batería - relevador

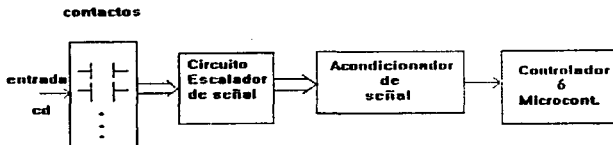


Fig. 2.3 Sistema de protección a base de relevadores conectado a un registrador de eventos.

Las señales de voltaje generadas por los eventos en un turbogenerador son escaladas por un circuito divisor de tensión en la etapa 2400A/110AC en el registrador de eventos patentado por la compañía Jhon Fluke, MFG. CO. INC.

2.2 ETAPA ESCALADORA DE SERIAL 1200A/110AC

El registrador de eventos está provisto de una etapa escaladora de señal que reduce el valor de las señales de voltaje producidas por cada uno de los eventos (Ver tabla 2).

La etapa escaladora de señal está formada por un arreglo resistivo en "cascada" conocido con el nombre de "divisor de tensión". (Ver fig. 2.4).

estado del contacto	Voltaje aplicado	Voltaje escalado
Cerrado	Señales superiores a 68Vcd.	2.47Vcd
Abierto	Señales inferiores a 48Vcd.	1.67Vcd

Tabla 2. Señales de voltaje escaladas por el divisor de tensión

El registrador puede escalar 32 señales de voltaje utilizando un "conector de entrada" (Ver fig. 2.5). Que contiene las primeras 32 resistencias R1 del divisor.

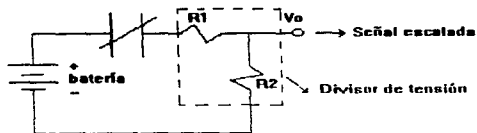
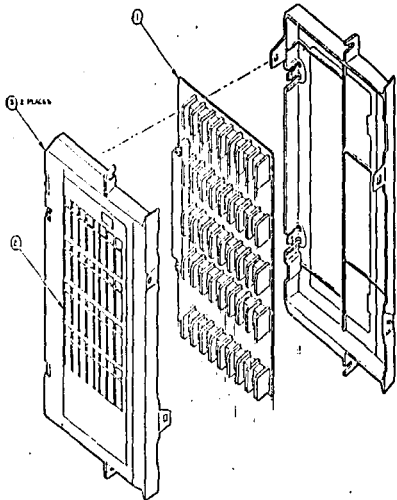


Fig. 2.4 Arreglo resistivo en "cascada" para escalar las señales de voltaje en un evento.

El conector de entrada acepta 32 señales de alto voltaje AC ó CD. Las resistencias se encuentran atornilladas y encapsuladas por bloques en una celda plástica cuyas dimensiones son 22x2.9x11 centímetros (Ver diagramas). Su peso aproximado es de 0.36 Kg. El valor de cada resistencia es de 10 Megohms. Pueden aplicarse como máximo $\frac{1}{2} V_{cd} \text{ ó } V_{cd}$ entre una terminal y tierra ó entre ambas terminales de una resistencia.

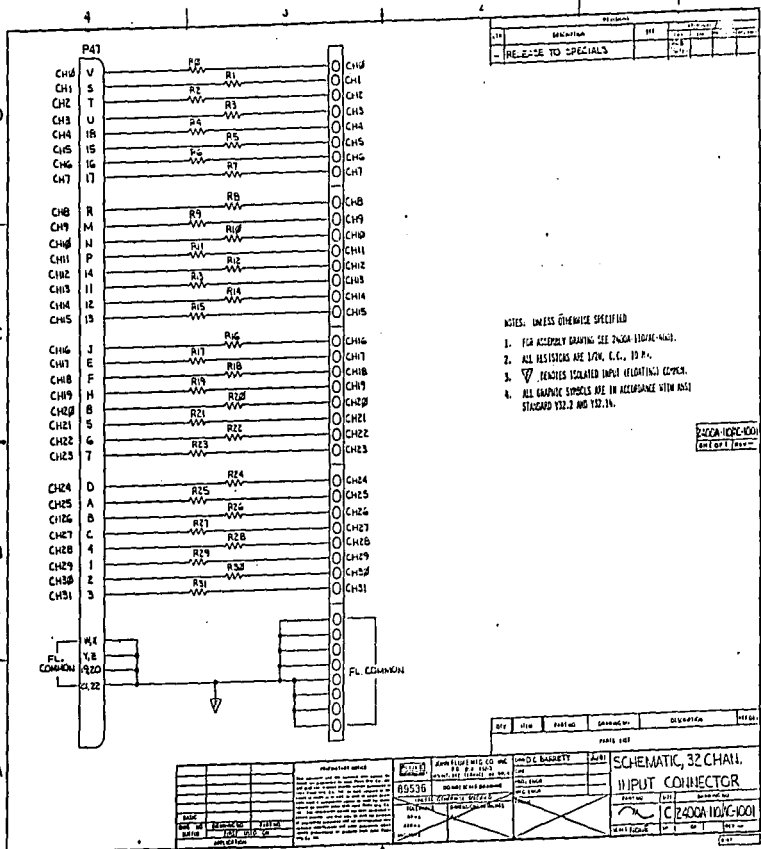
DATE	REV	PREPARED BY
RELEASE TO SPEAKERS		



NOTE: UNLESS OTHERWISE SPECIFIED
 1. ASSEMBLE PER DETAIL.
 2. TEST COMPLETED ASSEMBLY PER
 TEST PROCEDURE IN ENG-648-00.

1	3	4478	WALD COB	400000 1007 000
1	3	454731	SEALED SEA. 15 000 1007 000	
1	1	517153	INDUSTRIAL	22 000 1007 000

PART NO. 80536 TITLE INPUT CONNECTOR DATE 58-37 DRAWN BY CHECKED BY APPROVED BY	FINISH ASSY 22 CHAN INPUT CONNECTOR 58-37 01 2400A-100-01
--	---



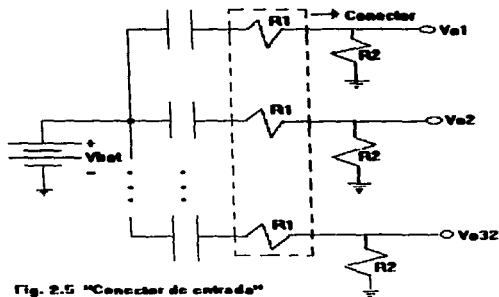


Fig. 2.5 "Conector de entrada"

Las siglas 1200A/110AC son utilizadas para identificar el conector como parte del registrador.

El segundo elemento resistivo $R2$ del divisor de tensión (Ver fig. 2.5) tiene un valor de 430 Kilo ohms. Estos 32 elementos resistivos no forman parte del conector 1200A/110AC. Se encuentran alojados en la tarjeta electrónica 2400A-104/AB (etapa acondicionadora de señal) del registrador.

CAPITULO III. ACONDICIONAMIENTO DE SEÑALES

3.1 GENERALIDADES

Con el desarrollo creciente y continuo de la tecnología e investigación en los países desarrollados. Ha sido posible lograr grandes avances científicos, en el área de la ingeniería electrónica. Como resultado de esto, la automatización y la implementación de sistemas digitales en casi todas las áreas se ha convertido en un proceso sin límites.

En las comunicaciones, se puede contar con un sistema de redes por computadoras a gran escala.

En la industria de manufactura, los procesos automatizados son rápidos y eficientes.

En la medicina, pueden visualizarse y ser diagnosticadas casi todo tipo de deficiencias en el organismo humano a través de las computadoras.

En las centrales termoeléctricas es posible tener un control sumamente confiable y poderoso con la tecnología de los microcontroladores.

3.1.1 ACONDICIONAMIENTO DE SEÑALES EN UN SISTEMA DE CONTROL

Como podemos apreciar, es posible establecer una relación directa entre el mundo en que vivimos y la tecnología. Parte de esta relación es la que se ha logrado en la ingeniería de control moderna mediante la creatividad y el ingenio humano.

El proceso creativo en este ámbito puede dividirse en tres partes:

- 1) Captura de las variables físicas que forman parte del fenómeno en estudio.
- 2) Acondicionamiento de las señales.
- 3) Diseño del control de las variables.

Todo tipo de sensores y transductores electrónicos son dispositivos que nos pueden proporcionar una señal eléctrica en función de una variable física. Por ejemplo, en la industria textil se hace necesario utilizar sensores de luz que envían una señal eléctrica en función de las rpm de un motor en una banda hiladora.

Una vez que se ha logrado tener una señal eléctrica producida por algún fenómeno en particular. Se hace necesario llevar a cabo un "acondicionamiento" de la misma. Esto se debe a que el fenómeno en estudio tiene características muy fluctuantes e indeterminadas, las cuales se ven reflejadas en las señales eléctricas que producen los transductores.

En la mayoría de los casos, un "acondicionamiento de señal" cumple con los siguientes objetivos:

- 1) Linealizar el comportamiento exponencial ó logarítmico de alguna variable física.
- 2) Eliminar fluctuaciones en la función lineal producidas por "ruido".
- 3) Convertir los valores determinados por la función lineal a niveles lógicos. En caso de utilizar un control digital.

3.1.2 ACONDICIONAMIENTO DE SEÑALES EN UN REGISTRADOR

Un registrador de eventos *no es un sistema de control*. Debido a esto, el acondicionamiento de señal únicamente realiza lo siguiente:

- 1) Elimina el "ruido" en las señales escaladas.
- 2) Convierte los valores de las señales escaladas a niveles lógicos.

Para lograr los dos objetivos anteriores es necesario contar con un filtro de corrientes transitorias, un detector de nivel con histeresis y un optoacoplador. Cada una de estas partes conforman la tarjeta electrónica 2400A-104/AB del registrador.

En las secciones siguientes explicaremos con detalle el funcionamiento del acondicionador de señal.

3.2 FILTRO DE CORRIENTES TRANSITORIAS

Cualquier circuito electrónico que maneja oscilaciones de voltaje periódicas, presenta pequeñas corrientes transitorias posteriores a cada oscilación. Aunque la duración de estas corrientes suele ser muy pequeña, es suficiente para causar problemas en la información procesada por los sistemas digitales.

En un registrador de eventos, cada señal de voltaje producida por el cambio de estado en un releador genera corrientes transitorias. Por ejemplo, un releador cuyos contactos están cerrados, presentan un voltaje escalado de $2.47V_{cd}$ cuando el voltaje en la batería es de $60V_{cd}$ (Ver Tabla 2, Sección 2.2). Al momento de ocurrir una sobrecorriente, es decir, la bobina del releador es excitada por una corriente mayor a la generada por la batería de $60V_{cd}$. El voltaje escalado de $2.47V_{cd}$ debe reducirse por lo menos a $1.67V_{cd}$ para que pueda interpretarse como un cambio de estado del releador de cerrado a abierto.

Sin embargo, el cambio de voltaje de $2.47V_{cd}$ a $1.67V_{cd}$ tarda en estabilizarse. Es decir, se presenta una señal de voltaje amortiguada con valores pico fluctuantes. (Ver fig. 3.1), durante un periodo de tiempo pequeño.

Estas señales amortiguadas distorsionan la información digital de entrada a un microprocesador ó microcontrolador. Por lo que deben ser eliminadas.

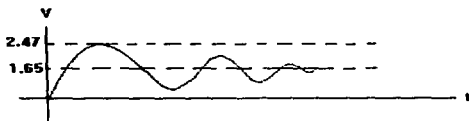


Fig. 3.1 Señal transitoria de voltaje

La solución del problema es posible mediante el uso de "filtros". Existen una gran variedad de ellos. Pero todos tienen una característica en común: Son arreglos resistivos-capacitivos (Ver fig. 3.2). Este es un filtro "pasa bajas de primer orden", utilizado por el registrador de eventos, los valores de R y C son de 1Kilo ohm y 0.01 microfaradios respectivamente.

Un filtro de este tipo suprime señales a frecuencias pequeñas ó bien, en periodos de tiempo relativamente largos. De tal manera que el registro de la información digital en un microprocesador ó microcontrolador es confiable.

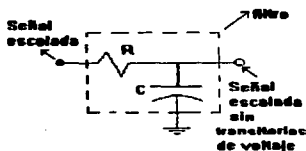


Fig. 3.2 FWR para bajas

3.3 DETECTOR DE NIVEL CON HISTERESIS

En muchas ocasiones deseamos que se lleve a cabo una serie de eventos especiales cuando realizamos una "acción determinada". Por ejemplo, cuando oprimimos un botón y suena una alarma, cuando nos acercamos a una puerta provista de sensores y ésta se abre para que entremos, cuando accionamos una palanca y todo un sistema de bombeo comienza a funcionar, etc.

La acción determinada que realizamos simplemente es activar un "mecanismo" con una "corriente eléctrica".

En electrónica, podemos activar "circuitos integrados" tales como computadoras, contadores, convertidores, memorias, microprocesadores, microcontroladores, optoacopladores, etc. con corrientes producidas por las "señales de salida" en circuitos analógicos como Detectores de Nivel, Convertidores D/A, etc.

3.3.1 OPTOACOPLADOR HCPL-2730

Cuando un sistema electrónico utiliza etapas distintas (Componentes analógicos y digitales) es necesario transmitir la señal de una etapa a otra a través de un circuito integrado llamado "Optoacoplador". (Ver fig. 3.3). Esto se hace con el fin de proteger los sistemas digitales de sobrecorrientes producidas en los componentes analógicos.

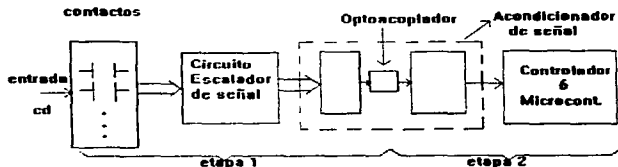


Fig. 3.3 Uso de un optoacoplador

Existen una gran variedad Optoacopladores. Su principio de operación es el de radiación de luz interna entre un transmisor y un receptor. El registrador de eventos utiliza el optoacoplador HCPL-2730 (Ver fig. 3.4). En este caso, el transmisor es un "diodo". El cual debe "polarizarse en directa" con una corriente de 5.33 mili amperes.

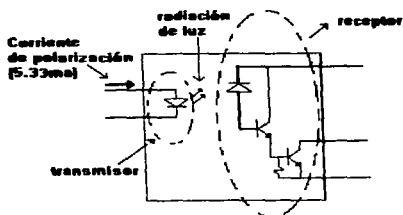


Fig. 3.4 Optoacoplador HCPL-2730

3.3.2 UTILIDAD DE UN DETECTOR DE NIVEL PARA POLARIZAR EL TRANSMISOR DE UN OPTOACOPLADOR

La corriente de polarización directa que recibe un diodo transmisor en un optoacoplador puede ser la señal de salida de cualquier circuito analógico.

El transmisor del optoacoplador HCPL-2730 recibe la señal de salida de un arreglo analógico llamado "Detector de Nivel con Histeresis". (Ver fig. 3.5). Este detector utiliza el circuito integrado LM339, las regiones de operación y su modo de funcionamiento se explicará en la siguiente sección.

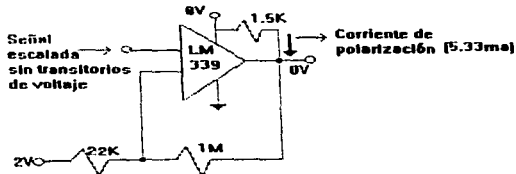


Fig. 3.5 Detector de nivel con histeresis

3.3.3 REGIONES DE OPERACION Y MODO DE FUNCIONAMIENTO DEL DETECTOR USADO EN EL REGISTRADOR DE EVENTOS

Este circuito recibe señales de $2.47V_{cd}$ y $1.67V_{cd}$, dependiendo del estado cerrado ó abierto de los contactos en el relevador emisor de señal.

Supongamos que el relevador tiene sus contactos cerrados, o bien, se envía un voltaje de entrada de $2.47V_{cd}$ al detector, éste lo compara con un voltaje "umbral superior" de $2.04V_{cd}$ (Ver fig. 3.6). Como es mayor, envía una señal de $0V_{cd}$ a la salida, lo que hace fluir una corriente de polarización de 5.33 mili amperes a través de la resistencia de 1.5 Kilo ohms (Ver fig. 3.5). Si por causas de "ruido", el detector cae en la región de "histéresis", es decir, en la región comprendida entre los voltajes $2.04V_{cd}$ y $1.87V_{cd}$ de la gráfica. La corriente de polarización mantiene su valor.

Si se presenta una eventualidad en el turbogenerador y el relevador respectivo cambia el estado de sus contactos, la señal enviada al detector cambia de $2.47V_{cd}$ a $1.67V_{cd}$, éste lo compara con el voltaje "umbral inferior" de $1.87V_{cd}$. Como es menor, envía una señal de $8V_{cd}$ a la salida, lo que elimina el flujo de la corriente de polarización (Ver fig. 3.7).

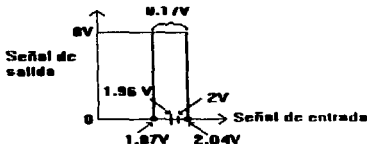


Fig. 3.6 Región de operación del detector

Si por razones de "ruido", el detector cae en la región de histéresis, la corriente de polarización sigue siendo nula.

Debemos señalar, que un circuito detector puede ser calibrado para trabajar en diversas regiones de operación con el mismo rango de histéresis únicamente ajustando el valor del voltaje de referencia. Como podemos apreciar, este valor ha sido ajustado a $2V_{cd}$ en el registrador de eventos.

El Detector descrito anteriormente es la última fase en la etapa de circuitería analógica que integra la etapa acondicionadora de señal del registrador. La segunda etapa está formada por el receptor del MCPL-2730 y de componentes digitales C.I. 74LS86 que son compuertas lógicas cuya función es asegurar un nivel lógico de voltaje para las señales enviadas por el receptor. En la siguiente sección veremos la lógica programada para esta compuerta, así como el funcionamiento del receptor y la relación que existe entre ambos.

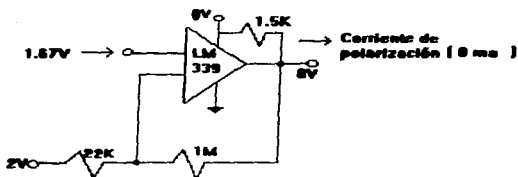


Fig. 3.7 Corriente de polarización nula en el detector

3.4 NIVELES LOGICOS DE VOLTAJE

Un nivel de lógico de voltaje es un valor comprendido entre $0V_{cd}$ y $5V_{cd}$. En electrónica digital, estos valores extremos son interpretados como "estados lógicos", es decir, "apagado ó encendido". Matemáticamente, corresponden a los números 0 y 1 del sistema de numeración binario.

Como sabemos, en un sistema de numeración, podemos efectuar operaciones aritméticas muy diversas. Los "circuitos integrados digitales" se han encargado de cubrir este campo. Por ejemplo, una "computera lógica" es un circuito integrado que puede realizar operaciones de suma, resta, y multiplicación. Una "memoria" almacena resultados para su posterior uso ó procesamiento. Un "microprocesador" efectúa operaciones lógicas y aritméticas a gran velocidad. Un "microcontrolador" reúne las características de una memoria y un microprocesador juntos, etc.

Gracias a estos avances tecnológicos se puede contar hoy en día con computadoras muy sofisticadas (inteligencia artificial) que ayudan al hombre en la solución de los problemas más complejos.

El problema decapturar eventos en un turbogenerador no es un problema de gran complejidad, pero si requiere del uso de circuitos integrados digitales. Por esta razón se hace necesario simular las eventualidades como niveles lógicos.

Parte de este trabajo se llevó a cabo en la primera etapa de circuitería analógica del registrador. Los pasos que integran esta etapa son:

- 1) Escalar los voltajes $40V_{cd} - 60V_{cd}$ a $1.67V_{cd} - 2.47V_{cd}$ respectivamente.
- 2) Eliminar transitorios de voltaje en las señales $1.67V_{cd}$ y $2.47V_{cd}$.
- 3) Detectar los cambios de voltaje de $1.67V_{cd}$ a $2.47V_{cd}$ y viceversa.

La segunda etapa del registrador de eventos comienza a partir del receptor del HCPL-2730. En las secciones siguientes hablaremos sobre el funcionamiento del mismo, así como de una compuerta lógica llamada "Or-Exclusiva" y la relación que existe entre ambos.

3.5 UTILIDAD DEL RECEPTOR EN EL HCPL-2730 Y DE LA COMPUERTA OR-EXCLUSIVA PARA SIMULACRO DE EVENTUALIDADES COMO NIVELES LOGICOS DE VOLTAJE

Como hemos estado mencionando, la utilidad de los circuitos integrados digitales es de sumo interés en la solución de problemas complejos. La magnitud de estos es muy diversa, dependiendo de la situación en la que nos encontremos. Sin embargo, debemos señalar que la aplicación más frecuente de estos sistemas radica en la solución de problemas técnicos industriales.

El problema de capturar eventos en un turbogenerador se vuelve complejo y fuera del alcance humano por las siguientes razones:

- 1) Los cambios diferenciales de temperatura, presión y nivel no pueden ser percibidos por una persona.
- 2) Las señales de cada evento que son enviadas por los relevadores tienen una duración de milisegundos.
- 3) Los eventos son impredecibles.

Luego, para poder abordar el problema y tratar de plantear una solución en base a sistemas digitales se hace necesario, como primer paso, simular los eventos como señales lógicas de voltaje.

3.5.1 FUNCIONAMIENTO DEL RECEPTOR EN EL HCPL-2730

El receptor es un circuito formado por un diodo D y dos transistores T1 y T2 denominado "Darlington" (Ver fig. 3.8). El diodo receptor D tiene la función de interruptor, es decir, abre ó cierra el circuito. Cuando se presenta un evento, el diodo transmisor del HCPL-2730 manda una señal radiante de luz al diodo receptor D. En este momento el circuito se cierra y comienza a fluir una corriente I a través de la resistencia de 33K, originando una caída de tensión en la misma, esto hace que el voltaje de salida V_o en el colector del transistor T2 sea la diferencia entre los 5V de la fuente y la caída de tensión resistiva. Esta diferencia corresponde a un nivel lógico de voltaje alto, 0 bien, a un estado de "encendido" que puede interpretarse como un 1 lógico en los sistemas digitales.

Cuando termina la duración del evento, el diodo transmisor deja de enviar radiación de luz al diodo receptor D. En consecuencia, el circuito se abre y la corriente I deja de fluir a través de la resistencia. Por lo que la señal de salida V_o cambia de un nivel lógico de voltaje alto a bajo. En otras palabras, se presenta un cambio de estado a "apagado" que corresponde a un 0 lógico en los sistemas digitales.

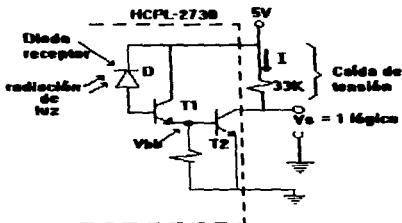


Fig. 3.8 Receptor del HCPL-2730

El siguiente paso a seguir en la simulación de eventos como niveles lógicos de voltaje es la implementación de un circuito lógico de "referencia". Ya que de lo contrario, sería imposible saber cuál fue el "estado inicial" del que partimos. Por ejemplo, si tenemos un estado lógico de 1 en V_o y éste cambia a 0, sería lógico pensar que acaba de ocurrir un evento. Sin embargo, esta afirmación es válida siempre y cuando sepamos que el 1 lógico es el estado inicial.

El circuito de referencia utilizado por el registrador de eventos está formado por una compuerta lógica llamada "Or-Exclusiva" que se explicará en la siguiente sección.

3.5.2 LA COMPUERTA OR-EXCLUSIVA COMO CIRCUITO DE REFERENCIA

Una compuerta lógica o compuerta binaria es el elemento básico en los sistemas digitales. Opera con el sistema de numeración binario (1 y 0 lógicos). Todos los sistemas digitales se construyen usando solo tres compuertas lógicas. A estas compuertas se les conoce con el nombre de AND, OR y NOT. (Ver fig. 3.9). Muchos problemas de lógica digital utilizan "combinaciones de compuertas". Uno de ellos, es el que se presenta cuando necesitamos detectar ó sentir un cambio de estado lógico. Es decir, de 1 a 0 ó viceversa.

Este problema es típico en un registrador de eventos, ya que necesitamos detectar el cambio de estado en los contactos de un relevador. La solución se ha logrado con la implementación de un "circuito de referencia" usando una combinación especial de compuertas llamada "compuerta Or-Exclusiva" (Ver fig. 3.10). Como podemos observar, este circuito es una simple conexión entre el receptor del HCPL-2730 y la compuerta. El estado lógico de referencia viene determinado por la posición del switch.

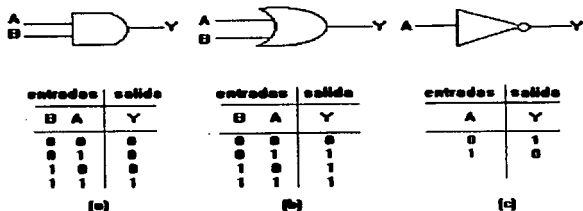


Fig. 3.5 (a) compuerta AND, (b) compuerta OR, (c) compuerta NOT

Si la posición del switch es "cerrada", tendremos un estado lógico de referencia igual a 1 a la salida de la compuerta cuando el nivel lógico de V_s sea de 0. Obviamente, si ocurre un evento, la señal de 0 lógico en V_s cambiará a 1 lógico, haciendo que exista un cambio de estado a la salida de la compuerta de 1 a 0 lógico. En este momento, podemos saber que ha ocurrido un evento.

Andógicamente, si la posición del switch es "abierta", tendremos un estado lógico de referencia igual a 0 a la salida de la compuerta cuando el nivel lógico de V_s sea de 0. Luego, al suceder una eventualidad, la señal de 0 lógico en V_s cambiará a 1 lógico, de tal manera que el estado inicial a la salida de la compuerta cambiará de 0 a 1 lógico, indicando la presencia de una eventualidad.

En el registrador de eventos, la posición del switch normalmente se encuentra cerrada (Ver diagramas).

El circuito de referencia es la última parte que integra la etapa acondicionadora de señal 2400A-104/AB en el registrador.

Los diagramas respectivos se muestran al final del capítulo.

La parte experimental de esta tesis comienza a partir del siguiente capítulo, donde hablaremos sobre un circuito multiplexor analógico diseñado especialmente para servir como interfase entre los 32 niveles lógicos de voltaje del registrador y un puerto de entrada al microcontrolador 68HC11E9.

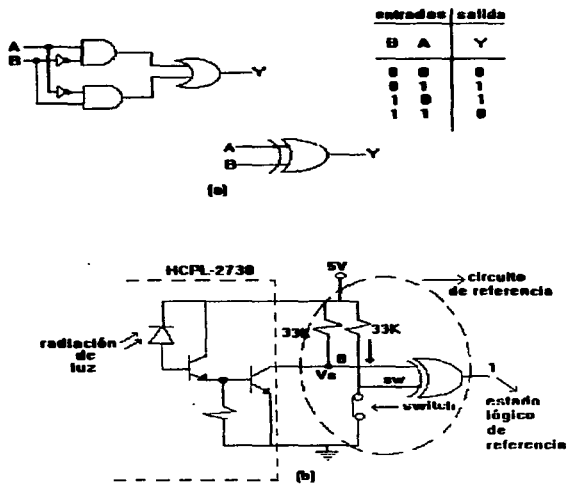
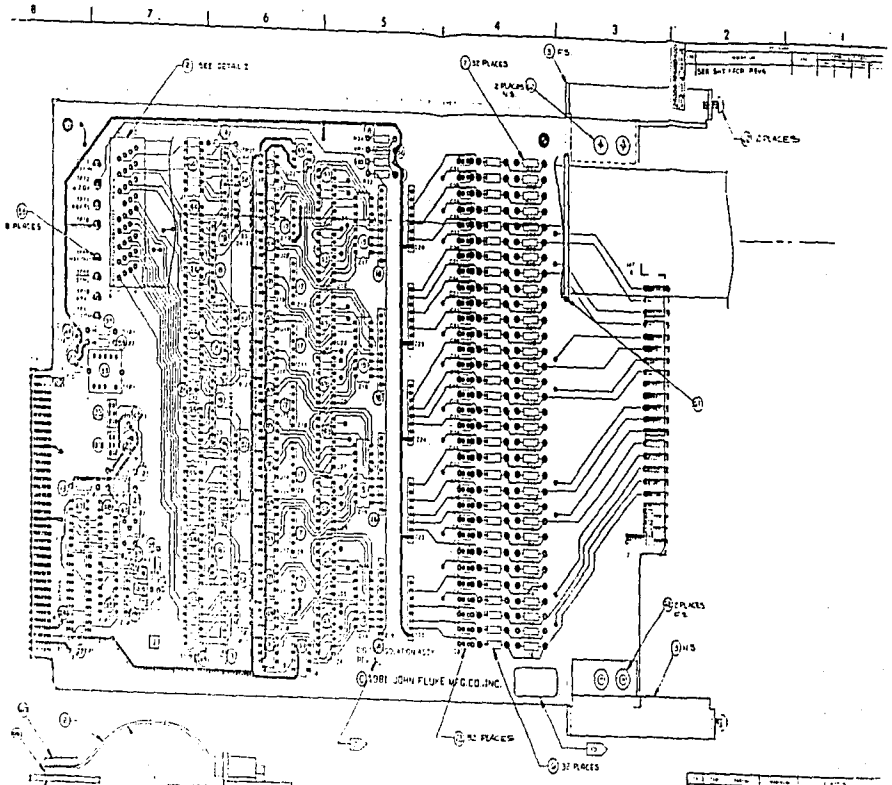


Fig. 3.10 (a)Compuerta Or-Exc. (b)Circuito de referencia sw=1



REV.	DATE	BY	CHKD.	DESCRIPTION
1				INITIAL SOLUTIONS
2				
3				
4				
5				
6				
7				
8				

CAPITULO IV . CONVERTIDOR DIGITAL-ANALOGICO

4.1 GENERALIDADES

Los aparatos telefónicos, son utilizados a diario por millones de personas en todo el mundo. Como sabemos, el modo de operación de un teléfono es muy simple, únicamente oprimimos una serie de teclas y establecemos la comunicación deseada. Si por alguna razón, tenemos la necesidad de cambiar la comunicación, simplemente colgamos y oprimimos nuevamente una serie de teclas diferente a la primera.

El proceso de seleccionar una línea de comunicación telefónica a través de una combinación determinada de teclas puede definirse en electrónica como "Multiplexar una señal". Todos los teléfonos modernos llevan en su construcción interna circuitos integrados llamados "multiplexores".

Existen muchas otras aplicaciones de los circuitos multiplexores, entre ellos destacan la construcción de interfaces para "comunicación digital". Cuando hablamos de comunicación digital, podemos pensar en el intercambio de información entre una computadora y sus periféricos. Por ejemplo, sería imposible enviar información de un teclado a la memoria de una computadora si esta no tuviese un "puerto de comunicación" para recibir los datos. En un puerto se enlazan líneas de comunicación (Ver fig. 4.1), a través de las cuales se genera un flujo de datos.

En muchas ocasiones, las líneas de comunicación difieren de la configuración del puerto, es decir, podemos tener 8 líneas de comunicación y nuestro puerto sólo puede admitir 4 de ellas. Para resolver este tipo de problemas se utilizan interfaces multiplexoras.

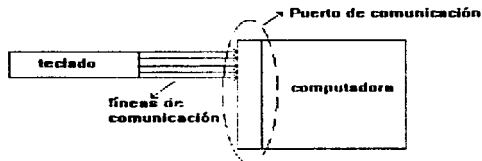


Fig. 4.1 líneas de comunicación entre un teclado y una computadora

4.2 NECESIDAD DE MULTIPLEXAR LOS NIVELES LOGICOS DE LAS ENENTUALIDADES EN UN TURBOGENERADOR.

Hace algunos años, el registrador de eventos instalado en la sección de ciclo combinado de la central termoelectrica "Fco. Pérez Ríos", contaba con un controlador identificado con las siglas 2400A-104/AA (Ver fig. 4.2) en comunicación con la etapa

acondicionadora de señal 2400A-104/AB descrita en el capítulo anterior. De la figura, podemos apreciar que el número de líneas de comunicación es de 32 (una por cada evento).

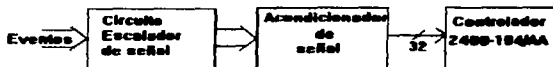


Fig. 4.2 Registrador de eventos en la control termomecánica hace algunos años.

Recientemente, el departamento de Control Supervisorio de la central decidió abordar el problema. Esta decisión, basada en un presupuesto económico y experiencia técnica enfocó, la solución a los siguientes puntos:

- 1) Conservar las tarjetas acondicionadoras de señal 2400A-104AB.
- 2) Utilizar un controlador versátil y económico que pudiese suplir las características del 2400A-104/AA ó mejorarlas.
- 3) Utilizar una computadora personal con un programa nuevo para sustituir el programa de prueba "A104AB" en la unidad de procesamiento Mainframe 2400A.

En el punto número 2, aparece el primer problema. Si nosotros cambiamos de controlador, las características técnicas de éste pueden ser iguales ó mejores al primero, pero las líneas de comunicación no son las mismas. Entonces, para poder llevar a cabo este punto se hace necesario diseñar un circuito multiplexor (Ver fig. 4.3), que pueda resolver el problema de comunicación entre las 32 líneas a la salida del acondicionador y las 2^n líneas de entrada al nuevo controlador.

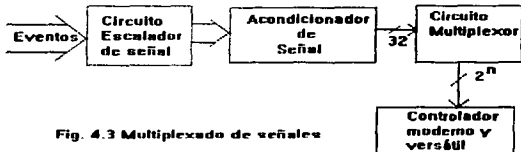


Fig. 4.3 Multiplexado de señales

4.3 NECESIDAD DE UTILIZAR UN CONVERTIDOR DIGITAL-ANALOGICO PARA TRANSMISION DE SEÑALES

En electrónica, existen multiplexores digitales y analógicos. Como su nombre lo indica, los primeros manejan datos digitales y los segundos datos analógicos. Los multiplexores constan principalmente de tres partes (Ver fig. 4.4), un circuito de líneas de entrada, una línea de salida y líneas de selección.

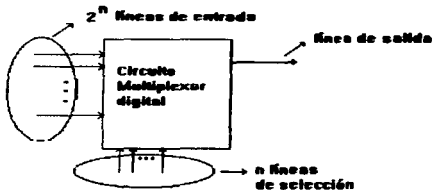


Fig. 4.4 Circuito multiplexor digital

Los multiplexores no pueden transferir un dato en la línea de salida si no se indica en las líneas de selección cual de todos los que conforman las líneas de entrada es el que va a ser transferido. Esto implica, diseñar un "circuito digital de selección" que se ajuste a nuestras necesidades. Más aún, si la transferencia de datos debe hacerse en milésimas de segundo al puerto de entrada en un controlador, el circuito digital de selección debe ser lo suficientemente rápido para evitar pérdidas de información. En este caso, podría utilizarse como circuito selector, las líneas en un puerto de salida del mismo controlador previamente programado, es decir, mediante el uso de software (lógica programada) llevar a cabo la selección (Ver fig. 4.5). Sin embargo, esto implicaría utilizar "memoria extra" del controlador, que por lo general es reducida. Existen circuitos integrados de memoria que pueden ser adaptados a los controladores para ampliar las capacidades de software de los mismos. Pero esta alternativa debe ser considerada en casos muy necesarios.

Sin embargo, el punto número 2 de la solución planteada por el departamento de control supervisorio, nos permite utilizar un controlador versátil y económico. En la actualidad, los controladores que cumplen con estas características son los "microcontroladores". Estos circuitos integrados, se han perfeccionado mucho, en tal magnitud, que podemos afirmar que son computadores del tamaño de un "chip".

Gracias a las capacidades que ofrecen los microcontroladores

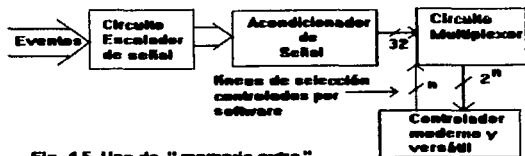


Fig. 4.5 Uso de "memoria extra"

hoy en día, se hace posible utilizar un convertidor digital-analógico para hacer la transmisión de datos (Ver fig.4.6), la señal analógica que llega al puerto de entrada en el microcontrolador, tiene una representación única como información digital en las líneas de salida del convertidor A/D (Análogo-Digital) interno. Esta relación entre ambos tipos de datos recibe el nombre de "resolución" y su valor depende del fabricante.

Luego, la implementación de una transmisión de datos con ayuda de un convertidor D/A se reduce simplemente a buscar la resolución apropiada.

Todo convertidor, necesita ser alimentado por una fuente de poder. Esto representa una desventaja cuando deseamos implementar la transmisión de datos, ya que se presenta "ruido" en la señal analógica de salida generado por la fuente. En la práctica, esta distorsión nos indicaría la aparición de eventos irreales, pero serían fácilmente reconocidos como tales por su naturaleza.

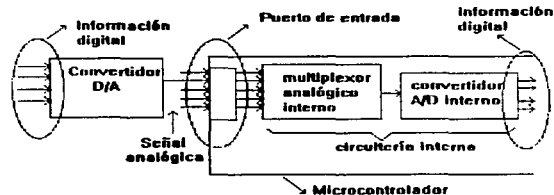


Fig. 4.6 Transmisión de datos a través de un convertidor D/A

En el mercado, existe una gran variedad de convertidores D/A que pueden servir para implementar una transmisión de datos. El componente utilizado en esta tesis, fue el circuito integrado

"DAC-08", que es un convertidor D/A (Digital-Analógico) de 8 bits En la siguiente sección hablaremos sobre el funcionamiento de este componente.

4.4 CONVERTIDOR DIGITAL-ANALÓGICO

En la sección anterior, se mencionó la forma de implementar una transmisión de datos con el uso de convertidores D/A. El convertidor utilizado en la experimentación de este proyecto, es un circuito integrado llamado "DAC-08" encapsulado en un empaque doble de 16 terminales (Ver fig. 4.7). Debemos señalar que cualquier convertidor D/A puede servir para éste propósito, siempre y cuando pueda ser ajustado a las características de operación del convertidor A/D interno. Todos los convertidores pueden ser ajustados para operar con un factor de conversión llamado "resolución". Esta cantidad, es la razón matemática que existe entre el valor de la señal analógica de salida y el valor decimal equivalente de la información digital de entrada a un convertidor. Por ejemplo, si los datos digitales de entrada en el circuito de la figura 4.7 son 11010110₂, o bien 214₁₀ y la señal analógica de voltaje es de 5V_{cd}, entonces el factor de resolución sería de 23.4×10^{-3} . Este número está determinado por el fabricante en los convertidores D/A internos, pero en los convertidores A/D existentes en el mercado, pueden ser ajustados a una resolución determinada según las necesidades personales.

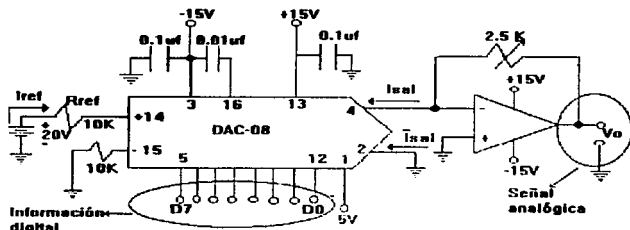


Fig. 4.7 Convertidor Digital-Analógico

En esta tesis, el convertidor DAC-08 se ajustó a una resolución de 20×10^{-3} . Por las razones antes mencionadas. Para lograr este factor de conversión fue necesario lo siguiente:

- 1) Construir una fuente regulada de voltaje para obtener los valores de +15V_{cd}, -15V_{cd}, +20V_{cd} y +5V_{cd} en la alimentación del DAC-08

- 2) Armar 3 circuitos similares más, para la simulación de 32 eventos como niveles lógicos de voltaje.

4.5 RESULTADOS OBTENIDOS EN LA EXPERIMENTACION CON EL DAC-08

En la sección 4.3, se habló sobre los efectos de "ruido" emitidos por la fuente de alimentación al convertidor D/A. Durante la experimentación en este proyecto, las fluctuaciones se presentaron en los bits de menos peso. Por ejemplo, si la información digital a la entrada del DAC-08 a ser transmitida era de 11010101, la salida del convertidor A/D registraba 11010111, como información recibida. Si la señal de entrada se cambiaba a 11010110, entonces obteníamos 11010111 en el convertidor. Este tipo de resultados, siguen siendo de utilidad, siempre y cuando el número binario de entrada forme parte del número binario a la salida, como en los casos previamente citados. En caso contrario, si los números binarios a la salida del convertidor se hubiesen registrado como 11010110, esta información sería de utilidad en un 90% para nuestros propósitos, puesto que la eventualidad no registrada en el bit 2^a tendría que ser averiguada personalmente en campo.

La parte experimental con Hardware se concluye con este capítulo. En los siguientes capítulos hablaremos acerca de la experimentación con software, donde manejamos programación de bajo y alto nivel, comenzaremos con lenguaje ensamblador, por lo que se hace necesario hablar previamente de las características técnicas del dispositivo a programarse.

CAPITULO V. EL MICROCONTROLADOR 68HC11 Y EL PUERTO E

5.1 GENERALIDADES

En la sección 4.3 del capítulo anterior, se habló de las capacidades que ofrece un microcontrolador moderno para poder implementar una transferencia de información con convertidores.

Hablar sobre microcontroladores sería muy extenso, ya que en la actualidad, existen varios fabricantes y múltiples modelos diferentes de chips. Mas aun, no es posible en un solo libro de texto cubrir todo lo referente a un microcontrolador en particular. En esta tesis, mencionaremos las características mas generales y destacaremos las capacidades mas atractivas que se utilizaron en este proyecto con el microcontrolador 68HC11E9 de MOTOROLA.

5.1.1 ¿QUE ES UN MICROCONTROLADOR?

Quando se vió la necesidad de realizar un gran numero de operaciones aritméticas y lógicas a gran velocidad, surgió la primera computadora en el mundo, utilizando la tecnología de la época. Lo que dió lugar a la primera generación de computadoras.

Gracias a los avances tecnológicos, las computadoras fueron perfeccionandose cada día mas, su tamaño se reducía gradualmente

y se incrementaba su capacidad. La libre competencia en el mercado no se hizo esperar, surgieron muchos modelos de computadoras y diversos fabricantes. Entonces, a través de la historia se fueron clasificando las computadoras de acuerdo a sus características técnicas en generaciones distintas. En la actualidad, nos encontramos en una generación muy avanzada de computadoras y la evolución en este campo sigue teniendo lugar en diversas partes del mundo a gran escala. Parte de este desarrollo técnico, ha dado lugar a la aparición de los "microcontroladores" en el mercado.

Al igual que una computadora, un microcontrolador puede realizar operaciones aritméticas y lógicas a gran velocidad, tiene los elementos básicos que integran una computadora (Ver fig. 5.1) que son, microprocesador, memoria y unidades de entrada/salida. La diferencia entre ambos, radica en su tamaño y capacidad. La mayoría de nosotros conocemos las dimensiones de una computadora personal. Un microcontrolador es tan pequeño, que puede ser alojado en una tarjeta electrónica de 17x8 centímetros con todo el hardware necesario para su uso. Sin embargo, un microcontrolador no puede sustituir a una computadora en su capacidad. Un microcontrolador se invento con la finalidad de ser programado e instalado en algun sistema de control. En la actualidad, podemos encontrar microcontroladores en los "ordenadores de lazo" (loop command), utilizados en el diseño de lazos de control en las centrales termoelectricas, también en, "sistemas de inyección electrónica de combustible" propios de automoviles sofisticados, así como "registradores de secuencia" en diversos sistemas electrónicos, etc.

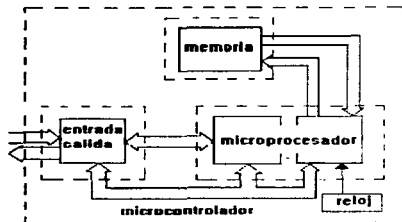


Fig. 5.1 Arquitectura de un microcontrolador

5.1.2 ¿COMO FUNCIONA UN MICROCONTROLADOR?

En la sección anterior, se mencionaron las partes principales de un microcontrolador, sus características físicas y se hizo una analogía con el funcionamiento de una computadora.

Debemos señalar, que un microcontrolador *no es una*

computadora y tampoco puede realizar las funciones de ésta. Pero las operaciones que realizan ambos son muy parecidas, así como el funcionamiento de sus mecanismos internos. En la siguiente tabla podemos apreciar algunas de éstas similitudes.

FUNCIÓN	COMPUTADORA	MICROCONTROLADOR
Operaciones aritméticas y lógicas	Unidad aritmética y lógica CPU	Unidad aritmética y lógica CPU
Almacenamiento de la información	Memoria RAM. ROM	Memoria RAM, ROM y EPROM
Entrada de información	Teclado, lápiz óptico	Teclado (Con el uso de un ensamblador)
Presentación de la información interna	Impresora, monitor	Impresora, monitor (Con interfase a un programa en lenguaje de alto nivel)
Comunicación entre usuario y la CPU	Intérprete de comandos	Intérprete de comandos
Comunicación con dispositivos digitales	Puertos serie y paralelos	Puertos bidireccionales de 8 a 16 bits

La mayoría de nosotros hemos utilizado una computadora, o por lo menos hemos visto lo que otras personas realizan con ella.

Las tareas que se pueden realizar hoy en día con una PC son muy diversas, pero la forma de trabajar de esta siempre es la misma. Es decir, realiza lo siguiente:

- 1) Recibe información externa.
- 2) Procesa la información.
- 3) Devuelve resultados.

Los microcontroladores trabajan exactamente igual, y su área de trabajo ha sido enfocada a la solución de problemas electrónicos. Es decir, ya no se hace necesario "tener conectada" una computadora y estar utilizando su hardware interno para implementación de "procesos digitales".

La información que recibe un microcontrolador, es un "conjunto de instrucciones" que le indican una tarea específica a realizar. Cada instrucción, es almacenada en un lugar único en la memoria interna del microcontrolador. Posteriormente, cada una de ellas pasa a ser procesada por una "unidad lógica y aritmética" dentro del CPU. Una vez concluido el procesamiento de una instrucción, el microcontrolador realiza lo indicado por esta. El procedimiento se repite una y otra vez hasta que todas las instrucciones hayan sido procesadas y ejecutadas.

Los resultados devueltos, pueden observarse en un monitor de computadora.

5.2 EL MICROCONTROLADOR 68HC11

Este microcontrolador, es un "circuito integrado" que contiene muchos dispositivos de "entrada/salida" y Varios tipos de memoria que podemos programar. Si requerimos más dispositivos, podemos hacer una expansión de hardware fácilmente.

Se cuenta con una capacidad de 12 Kilo Bytes en ROM y 512 Bytes en RAM (Ver fig. 5.2), todos los dispositivos de entrada/salida que observamos, pueden ser utilizados en el modo de operación A. Este modo, se conoce también como "simple". En el modo B ó "expandido", algunos dispositivos no pueden utilizarse, ya que algunas direcciones internas ocupan las líneas de entrada/salida. El CPU puede ejecutar un máximo de 307 instrucciones para control de datos, direcciones y líneas de entrada/salida. Los dispositivos de "comunicación serial" SPI nos permiten establecer comunicación con una computadora para presentación de resultados, así como de poder utilizar el "intérprete de comandos" del microcontrolador. Los 512 Bytes en EPROM, nos permiten almacenar las instrucciones del programa diseñado, en algunas ocasiones este espacio de memoria no es suficiente para nuestra aplicación, y se hace necesario utilizar los comandos del intérprete para mover instrucciones a chips externos de memoria. Podemos establecer y controlar comunicaciones con dispositivos externos a través de la interfase SPI. Estas conexiones externas, se llevan a cabo en los puertos paralelos de entrada/salida A, B, C, D y E. El "Timer System", nos permite medir intervalos de tiempo, controlar el tiempo de las señales de salida, medir fácilmente las RPM de un motor, generar pulsos de salida con mucha exactitud, etc. Podemos conectar sensores, muchos tipos de interfaces y circuitos analógicos al puerto E, ya que una de las capacidades del microcontrolador nos permite utilizar este puerto como sensor de voltaje.

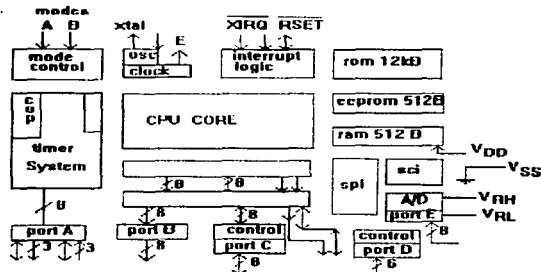


Fig. 5.2 Microcontrolador 68HC11

5.3 SISTEMA DE CONTROL DEL CONVERTIDOR A/D EN EL PUERTO E

Cuando programamos cualquier dispositivo entrada/salida en el 68HC11E9, activamos internamente un "sistema de control" propio del dispositivo.

El "sensor de voltaje", en el puerto E del 68HC11E9, es un dispositivo controlado por un sistema de "lazo cerrado" (Ver fig. 5.3). Inicialmente, el registro D/A tiene un voltaje de referencia V_{ref} igual a la mitad del voltaje maximo de conversion V_{max} . El comparador, resta un voltaje desconocido V_x (voltaje de entrada al puerto) del voltaje de referencia, si el resultado de la operación es negativo, significa que V_x es mayor que V_{ref} , luego, el controlador comienza a incrementar el valor de V_{ref} con un factor geométrico de $(V_{max} - V_{ref}) / (1/2^{n-1})$ hasta que el resultado de la resta en el comparador sea positivo, en este momento el valor de voltaje en el registro D/A se convierte en la salida digital.

Si el resultado de la resta entre el voltaje de referencia inicial y el voltaje desconocido es positivo, significa que V_x es menor que V_{ref} , luego, el controlador comienza a decrementar el valor de V_{ref} con un factor geométrico de $V_{ref} / (1/2^{n-1})$ hasta que el resultado de la resta en el comparador sea negativo, en este momento, el valor de voltaje en el registro D/A se toma como referencia V_{ref} y a partir de este valor se realizan los incrementos geométricos $(V_{ref} - V_{ref}) / (1/2^{n-1})$ en V_{ref} hasta que el resultado de la resta en el comparador sea positivo. En este momento, el valor en el registro D/A se convierte en la salida digital.

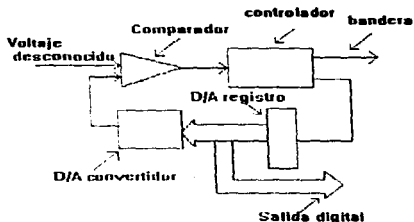


Fig. 5.3 "sensor de voltaje" en el puerto E

Como podemos darnos cuenta, se requiere un tiempo mínimo necesario para convertir el nivel de voltaje en la entrada. Por esta razón, el 68HC11E9 cuenta con un retenedor interno (Ver fig.

5.4) para asegurar una conversión correcta. Este tiempo de conversión, se ha calculado con un valor aproximado de $64_{\mu s}$.

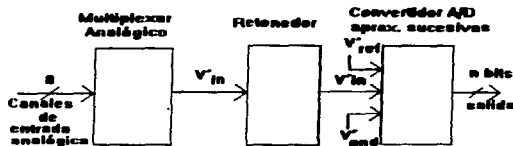
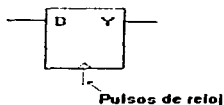


Fig. 5.4 "retenedor" de señales analógicas en el puerto E

5.4 LOS REGISTROS EN EL MICROCONTROLADOR 68HC11

Durante la ejecución de cualquier algoritmo en un microprocesador o microcontrolador, se llevan a cabo operaciones de almacenamiento y lectura de números binarios. Los cuales, son almacenados por un conjunto de "flip-flops". Un flip-flop es la "unidad mínima" de memoria en los sistemas digitales, esto significa, que tienen la capacidad de almacenar un bit de información. El flip-flop más común en los sistemas digitales es el "tipo D" (Ver fig.5.5), el cual cambia su estado de salida "Y" por una señal de entrada "D" cada vez que recibe un "pulso de reloj". Si diseñamos un "circuito síncrono" con dos o más flip-flops, podemos almacenar un conjunto de bits que nos representaran un número binario (Ver fig. 5.6). Un circuito de este tipo, se conoce como "registro". En un microcontrolador, los registros pueden transferir su información en forma paralela, serial o retenerla si el "control" en una parte de hardware del microcontrolador depende de la misma.



entrada	estado anterior	estado actual
D	Y	Y
0	0	0
0	1	0
1	0	1
1	1	1

Fig. 5.5 Tabla de funcionamiento del "flip-flop D"

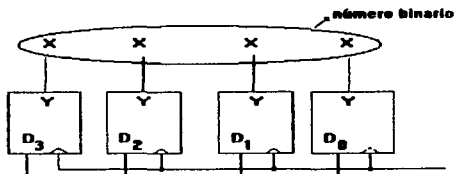


Fig. 5.6 Registro de cuatro bits, X=0 ó 1 lógicas

Los registros, pueden representarse con simbologías muy diversas, dependiendo de los detalles a ilustrar. Por ejemplo, si deseamos mostrar en un diagrama electrónico la presencia de un registro, lo podemos hacer mediante la interconexión de pequeñas cajas (Ver fig. 5.7(a)) simulando los flip-flops. Si la información binaria tiene a su cargo el control de circuitería interna, es aconsejable mostrar el estado de los flip-flops (Ver fig. 5.7 (b)) con unos y ceros. Cuando deseamos ilustrar las operaciones transferencia ó almacenamiento de una instrucción mnemónica, se utiliza la notación más común, donde el contenido del registro, se indica con un número hexadecimal (Ver fig.5.7 (c)), finalmente, si deseamos explicar una operación de corrimiento ó transferencia paralela en un registro, hacemos uso de flechas para indicar el flujo de información (Ver fig. 5.7 (d)).

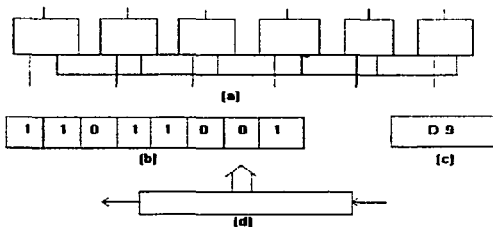


Fig. 5.7 Simbología utilizada para representar un registro

5.4.1 REGISTROS DEL MICROPROCESADOR EN EL 68HC11E9

Las operaciones de almacenamiento, transferencia, aritméticas, salto y decisión son ejecutadas por el CPU (microprocesador) del 68HC11E9. Como sabemos, la arquitectura básica de cualquier microprocesador consta de tres partes principales:

- 1) Sistema de control en direcciones, datos y señales.
- 2) Unidad lógica y aritmética.
- 3) Registros.

En esta estructura, existe un conjunto de líneas donde se intercambian datos entre la unidad lógica y los registros. De lo contrario, no sería posible llevar a cabo la ejecución de las instrucciones. El CPU en el microcontrolador 68HC11 tiene seis registros en su arquitectura (Ver fig. 5.8), cinco de ellos pueden almacenar datos de 16 bits y uno se encarga de controlar el estado de las "banderas". El registro D puede utilizarse por separado. Es decir, como dos registros de 8 bits independientes A y B para manipulación de datos. Los registros X y Y se utilizan generalmente para almacenamiento de direcciones. Cuando un algoritmo de programación requiere el uso del "Stack", el registro SP es utilizado para llevar a cabo direccionamientos indexados. El registro PC almacena el valor de las direcciones en la memoria EEPROM. El contenido de este registro varía de acuerdo a la ejecución del algoritmo. El registro CCR (Condition Code Register) recibe señales que dependen de la ejecución de cada una de las instrucciones, la lógica de interrupción y muchas instrucciones de salto dependen de las condiciones presentadas en este registro.

7	A	0	7	0	Acumuladores A y B
15		D		0	Acumulador D
15		X		0	Registro X
15		Y		0	Registro Y
15		SP		0	Apuntador del Stack
15		PC		0	Contador de prog.
S X H I N Z V C					Registro CCR

Fig. 5.8 Registros del CPU en el 68HC11E9

5.5 LA MEMORIA EN EL MICROCONTROLADOR 68HC11

Una memoria, es un dispositivo que puede almacenar números binarios por medio de señales digitales provenientes de un circuito electrónico, a través de un alambrado físico externo, o por la acción de instrumentos que sean capaces de generar campos eléctricos y magnéticos. Podemos definir a una memoria como un "conjunto de registros" hechos con flip-flops. El acceso a los registros, se hace de manera secuencial, por lo que cada uno tiene un número llamado "dirección" que lo identifica dentro del hardware de la memoria. Toda memoria en un microcontrolador interactúa con otros dispositivos. Esta interacción, puede lograrse con cuatro señales digitales que controlan el flujo de información al chip (Ver fig. 5.9), estas señales son:

- 1) Bus de direcciones.
- 2) Bus de datos.
- 3) Línea de lectura/escritura.
- 4) Señal de reloj.

El Bus de direcciones, es un grupo de líneas de entrada a la memoria, en donde se lleva a cabo una "selección" de registros. El Bus de datos, es un grupo de líneas entrada/salida en donde los números binarios son leídos ó accedidos a la memoria de acuerdo con el registro previamente seleccionado. Si algún dispositivo, en comunicación con la memoria va a realizar una operación, la línea de lectura/escritura indicará la naturaleza de ésta. El pulso de reloj, se encarga de activar al chip de memoria cada vez que se va a realizar una transferencia.

Los registros que integran una memoria, pueden representarse como un conjunto de "bloques" unidos entre sí, formando una "pila" (memoria), como se muestra en la figura 5.10, cada bloque contiene un "numero hexadecimal" de 8 bits. Este número, nos representa el dato contenido en un registro de la memoria. El número que identifica cada registro (dirección) es un número hexadecimal de 16 bits que se encuentra a la izquierda de cada bloque.

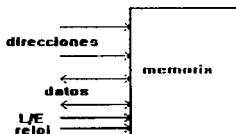


Fig. 5.9 Interacción de una memoria con dispositivos

8000	-
	-
	-
C239	86
C23A	C1
	-
	-
	33
FFFF	22

Fig. 5.10 Representación simbólica de una memoria

5.6 PROGRAMACION DE DISPOSITIVOS ENTRADA/SALIDA

Quando deseamos utilizar algún dispositivo de entrada/salida en el microcontrolador, necesitamos conocer los registros que "controlan su hardware". Estos registros de control, nos permiten habilitar y configurar cada dispositivo para que funcione de una manera específica. En condiciones normales (condiciones de reset), los registros de control se encuentran ubicados en las direcciones 1000 a 103F en el espacio de memoria del microcontrolador (Ver fig. 5.11). Si deseamos cambiar su posición en la pila, podemos hacer uso del registro INIT, el cual intercambia las direcciones de la memoria RAM con las direcciones de los registros.

En el apéndice B, pueden consultarse direcciones, bits y nombres de los 64 registros en el 68HC11E9.

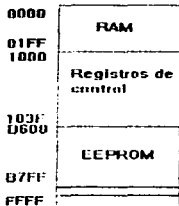


Fig. 5.11 Espacio de memoria en el 68HC11

En esta tesis utilizamos los siguientes:

- 1) El registro TMSKE del dispositivo "Contador de 16 bits"
- 2) El registro TFLG2 del dispositivo "Interrupcion de reloj"
- 3) El registro ADCTL del dispositivo "Sensor de voltaje"

- 4) El registro OPTION del dispositivo "Sensor de voltaje"
 5) Los registros ADRX del dispositivo "Sensor de voltaje"

El TMSK2 controla la velocidad de conteo del dispositivo. Es decir, el contador de 16 bits puede incrementar el valor de sus registros de 0000 a FFFF a razones de 32.77, 131.1, 262.1, y 524.3 milésimas de segundo, según lo indique el registro TMSK2.

Por ejemplo, si deseamos que el 68HC11E9 "realice una tarea determinada" cada 32.77 milésimas de segundo, hagamos que los bits de control TOI, PRI y PRO sean igual a 1.0 y 0 lógicos en el registro de control (Ver fig. 5.12(a)) respectivamente, mediante las instrucciones:

```
LDA #80
STAA TMSK2
```

El contador alcanzará su valor máximo en 32.77 ms, es decir el valor de FFFF, entonces enviará una señal al registro TFLG2, la cual, podrá ser detectada como un cambio de estado de 1 a 0 lógico en el bit TOF del registro (Ver fig. 5.12(b)). Para probar el estado de este bit, se utiliza una instrucción de 4 bytes con el siguiente formato:

etiqueta BRCLR 0.X.máscara.salto

en donde *etiqueta*, es la primera línea (Interrupt Vector) de un segmento de programa en ROM llamado "rutina de interrupción". O bien, la tarea que deseamos realizar a los 32.77 ms. Esta rutina, se ejecuta solamente cuando se ha presentado la señal en el registro, es decir, cuando su valor cambia de 1XXXXXX, a 0XXXXXX, este cambio de estado es posible conocerlo mediante el segmento de instrucción *BRCLR 0.X.máscara*, en donde 0.X nos representa la dirección del registro TFLG2 y *máscara* tiene el valor hexadecimal de 50. En caso de no existir ningún cambio de estado, la rutina de interrupción en ROM será ignorada y la ejecución pasará al espacio de memoria en EEPROM indicado por la etiqueta *salto*.

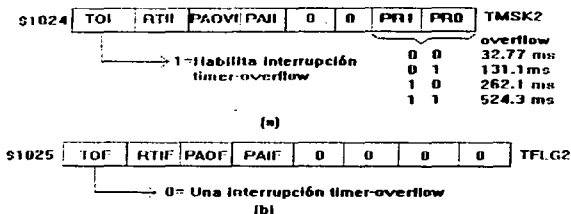


Fig. 5.12 Bits de control en los registros TMSK2 y TFLG2

Si utilizamos el sensor de voltaje en el 68HC11E9, para capturar señales analógicas de dispositivos externos conectados al puerto E. Debemos encender el sistema A/D colocando un 1 lógico en el bit de control ADPU (Ver fig. 5.13 (a)) del registro OPTION como sigue:

```
LDAA #80
STAA OPTION
```

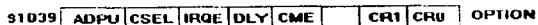
Además, podemos indicarle a nuestro dispositivo, que trabaje de manera continua sensando cuatro señales analógicas en los bits más bajos del puerto, colocando un 1 lógico en los bits SCAN, MULT y un 0 lógico en los bits CC y CB en el registro de control ADCTL (Ver fig. 5.13 (b)) con las siguientes instrucciones:

```
LDAA #30
STAA ADCTL
```

Para asegurar un valor correcto de conversión, es aconsejable generar un "retardo" de 64_{µs}, mediante el software:

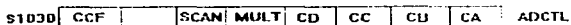
```
RETA    LDAB #31A
        DECB
        BNE RETA
```

Cada lectura de voltaje, en los bits 0-3 (bits más bajos) del puerto, tendrán un equivalente binario de 8 bits de acuerdo con una resolución de 20mV/bit. Estos números binarios, se almacenarán de manera continua en los registros ADR1, ADR2, ADR3 y ADR4 respectivamente.



Enciende el convertidor A/D
 0= Sistema A/D apagado
 1= Sistema A/D encendido

(a)



(b)

Selecciona el modo de operación
 0= modo simple
 1= modo múltiple

Selecciona el nibble superior ó inferior en el puerto L
 00= bits 0-3
 01= bits 4-7

Controla el No. de conversiones
 0= 4 conversiones y se detiene
 1= Convierte continuamente

Fig. 5.13 Bits de control en los registros OPTION y ADCTL

CAPITULO VI. DISEÑO DEL PROGRAMA PARA CAPTURA DE EVENTOS EN EL MICROCONTROLADOR 68HC11E9

6.1 GENERALIDADES

El problema de "capturar eventos", como la mayoría de los problemas electrónicos, tiene un grado de dificultad relativa, es decir, depende de nuestra imaginación, y del hardware, ó software que tengamos a nuestra disposición.

Para introducirnos al concepto del mismo, comenzaremos por describir la solución que inicialmente se había logrado con el controlador 2400A-104/AA del registrador patentado por Jhon Fluke MFG. CO. INC. y el alcance de su metodología.

Una vez que tengamos en mente el concepto básico, hablaremos sobre el método de solución obtenido en este proyecto con el 68HC11E9 y las ventajas que presenta utilizar un microcontrolador de estas características.

6.1.1 ¿QUE SIGNIFICA CAPTURAR UN EVENTO EN TERMINOS DIGITALES?

En términos digitales, capturar un evento se define como la acción de "registrar la aparición de un estado lógico" en un "momento determinado". Para ilustrar esto, supongamos que tenemos un número binario igual a 1101011, grabado como dato en una localidad de un chip de memoria. Si por alguna razón desconocida, este dato sufre un cambio en alguno de sus bits, es decir a 11010011, entonces podemos decir que el "cuarto bit" en nuestro dato "cambió de 1 a 0" lógico. Posteriormente, si por otra causa desconocida se produce un nuevo cambio de estado como 1110011, podemos afirmar que el "sexto bit" "cambió de 0 a 1" lógico. Si las causas desconocidas continúan presentándose, y nuestro dato sigue cambiando, entonces se producirán más números binarios y más afirmaciones.

Ahora, supongamos que después de N causas desconocidas nuestro dato deja de presentar cambios. Entonces, tendríamos un registro de N números binarios y N afirmaciones. Por lo tanto, podríamos decir que hicimos una "captura de N eventos".

6.1.2 NECESIDAD DE UTILIZAR UN CHIP DE MEMORIA PARA LA CAPTURA DE UN EVENTO

Como hemos visto, la acción de "registrar un evento" es equivalente a "escribir un número binario", donde cada evento es representado por un bit de dicho número. En sistemas digitales, el único dispositivo que nos sirve para almacenar números binarios es un "chip de memoria". Luego, esta debe interactuar con dispositivos externos "registradores de secuencia" para que la información que vaya a ser almacenada a la misma, pueda ser considerada como una secuencia ó captura de eventos.

6.1.3 FUNCIONAMIENTO DEL CONTROLADOR 2400A-104/AA

El controlador 2400A-104/AA, fue diseñado especialmente para capturar 32 eventos, es parte de una patente adquirida por la

Comisión Federal de Electricidad en el año de 1981. Actualmente, no se encuentra en funcionamiento. En esta tesis, únicamente será expuesto el concepto básico sobre el cual se basó la construcción de esta patente, con la finalidad de mostrar al lector lo que significa "capturar un evento" con sistemas digitales.

Este controlador, es una tarjeta electrónica con dimensiones físicas de 19.5x 35 centímetros. (Ver diagramas) en donde se encuentran alojados chips lógicos como son compuertas, flip-flops, buffers de tercer estado y una "memoria" entre otros. Esta circuitería, se agrupa en cuatro bloques principales (Ver fig. 6.1) que son:

- 1) Buffers de entrada.
- 2) Cristal oscilador.
- 3) Hardware registrador de secuencia.
- 4) Chip de memoria.

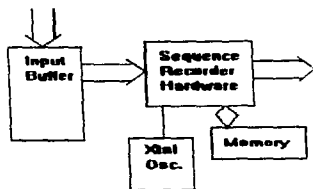


Fig. 6.1 Controlador 2400A-104/AA

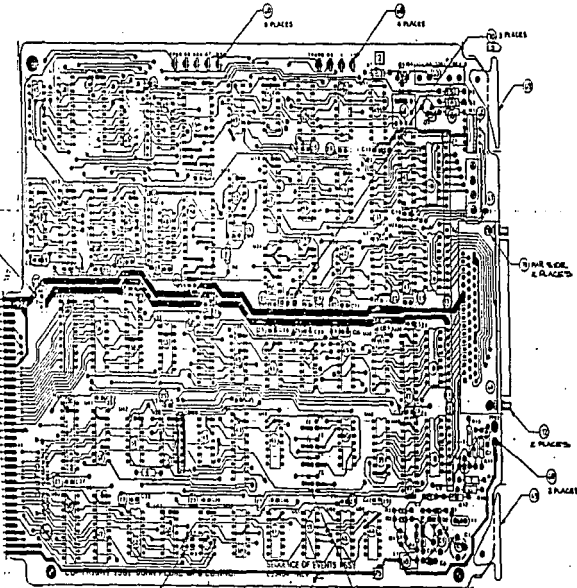
El Hardware registrador de secuencia, está formado a su vez por seis componentes interconectados entre sí (Ver fig. 6.2) que son:

- a) Circuito M.T.S. (Multi-Trigger Suppression).
- b) Circuito selector de M.T.S.
- c) Compuerta OR habilitadora de memoria.
- d) Circuito selector de datos.
- e) Contador de direcciones.

En este circuito, el M.T.S. se encarga de llevar a cabo el "reconocimiento de ocho eventos", cada vez que uno de los ocho eventos es "reconocido", se envía una señal de entrada a la compuerta OR, que a su vez, se encarga de activar al chip de memoria RAM con su salida. Esto sucede siempre que la compuerta recibe una señal de entrada proveniente de cualquiera de los cuatro M.T.S. Luego, el circuito selector de datos determina una combinación binaria única para representar el evento como un

8 7 6 5 4 3 2 1

DATE SHEET 1 FOR REVOLUTIONS



SEE SHEET 1 FOR NOTES

10 PLACES
11 PLACES
12 PLACES

DATE SHEET 1 FOR PARTS LIST

REV	DATE	BY	CHKD	APP'D	DESCRIPTION

<p>SEQUENCE OF EVENTS ASSEMBLY</p> <p>23554041000-1000-1000-1000</p> <p>REV. 1.0</p>	<p>DATE SHEET 1 FOR PARTS LIST</p>
--	------------------------------------

"dato" en función del MTS que lo reconoció y le asigna una "localidad de memoria" tomada del contador de direcciones. Posteriormente, se graba el dato y dirección en la memoria RAM.

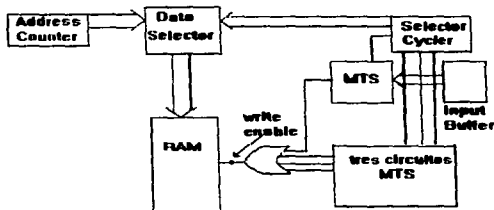


Fig. 6.2 Registrador de secuencias en el 2400-184AA

Como podemos darnos cuenta, el circuito MTS es el componente principal en la tarjeta, ya que el funcionamiento del resto de la circuitería depende del "reconocimiento" que lleva a cabo este dispositivo.

Los componentes que integran a un MTS, son dos registros síncronos y tres compuertas lógicas (Ver figura 6.3). En este diagrama, podemos apreciar el concepto de "reconocer un evento". Los datos datos paralelos a la entrada del registro UD, son transmitidos en forma serial al registro UB, el cual, lleva a cabo un corrimiento interno de los datos seriales de izquierda a derecha. Esta rotación, produce distintas señales de salida en el inversor, las cuales, se multiplican con el dato paralelo en transmisión (señal presente en la salida Q del registro UD). Debido a la sincronía de ambos registros, y a la presencia de la compuerta OR, la señal de entrada en la compuerta AND producida por la salida del inversor, será en todo momento la "negación del estado anterior del dato paralelo a transmitirse". Luego, si el estado de este dato no ha cambiado, obviamente la compuerta AND no hará "reconocimiento" alguno. En otras palabras, una vez que hayan sido reconocidos los ocho bits en la entrada del registro UB, no habrá señal de salida en la compuerta AND a menos que alguno de estos bits cambie su estado lógico.

La señal en la línea E (Chip Enable) del registro UD, se encarga de habilitar este componente. Esta señal de control, está a cargo del "Selector Cycler" mostrado en la figura anterior.

La señal en la línea P1 (Load), se encarga de llevar una cuenta binaria de 000 a 111 para enviar cada dato paralelo en forma serial a una razón de 4_{us} .

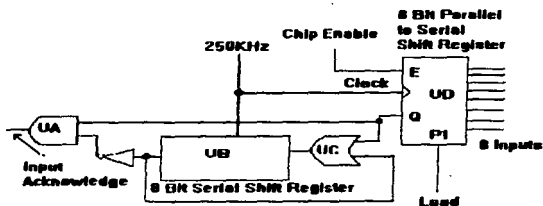


Fig. 6.3 Reconocimiento de eventos con un MTS

6.2 SIMULACION DE UN MTS CON EL MICROCONTROLADOR

Como hemos visto, un circuito MTS es el dispositivo básico en el hardware registrador de secuencia. Si existe la posibilidad de simular el funcionamiento de un dispositivo de esta naturaleza con algún algoritmo en el microcontrolador, entonces el hardware registrador de secuencia, también podrá ser simulado. Y por lo tanto, la tarjeta electrónica 2400A-104/AA del controlador será simulada por un programa completo en el microcontrolador 68HC11E9.

6.2.1 CARACTERISTICAS DE UN CIRCUITO MTS

La característica principal de un circuito MTS, radica en la capacidad de "reestructurar operaciones aritméticas en forma serial, a partir de un dato *paralelo*". Es decir, puede multiplicar o sumar los bits de un Byte con algún otro bit, que no forma parte del Byte y enviar un *bit de resultado*. Esta capacidad, representa una ventaja en el reconocimiento de un evento, pues si deseamos hacer un registro en RAM, únicamente asignamos al bit de resultado una dirección.

6.2.3 CARACTERISTICAS DEL SOFTWARE Y SIMULACION DE UN MTS

Si los bits de un dato *paralelo*, se encuentran en algún registro del 68HC11E9, "no pueden ser multiplicados o sumados en forma serial". Porque todas las operaciones aritméticas en software pueden aplicarse únicamente a un Byte, y no a los bits que lo integran. Luego, el reconocimiento de un evento resulta ser un proceso un poco complicado, pero no imposible.

Luego, *¿Cómo simular a un dispositivo con las características del MTS con el software del 68HC11?*

A lo largo de este capítulo, hemos estado hablando sobre

"reconocer un evento". pero no hemos definido con toda precisión lo que significa en términos digitales. La definición, es muy simple: " detectar el cambio de estado lógico en alguno(s) de los N bits en un número binario D." El circuito MTS, lo hacía multiplicando el complemento del bit N del estado anterior, con el bit N del estado actual.

En software, esto se implementa multiplicando el complemento del número D por los N números 2^k , donde $k=0,1,\dots,N-1$.

6.2.4 SUBROUTINA SIMULADORA DEL MTS

Esta subrutina, es parte de todo un programa creado para la simulación con software de las funciones hardware en el controlador 2400A-104/AA. Debemos señalar, que este programa, no sólo simula al controlador para su replazo, sino también posee algunas otras características interesantes que mencionaremos posteriormente.

La subrutina simuladora del MTS, se ha nombrado como ALARM, la cual, recibe un número 2^k representado por una variable denominada MASK y el valor $k+1$ en la variable CONT:

```
ALARM LDAA MASK
      PSHA
      LDAB CONT
      PSHB
      TSX
```

Posteriormente, toma lectura del número D en el puerto asociado al dispositivo entrada/salida. En este caso, utilizamos el "sensor de voltage" asociado a los puertos ADR1, ADR2, ADR3 Y ADR4 con direcciones 1031, 1032, 1033 y 1034 respectivamente. Cualquiera de estas direcciones, están representadas por la variable PUERTO:

```
LDY PUERTO
BIT LDAA 0,Y
```

Luego, se realizan K multiplicaciones entre el complemento del número D y los números 2^k hasta reconocer el evento N:

```
COMA
ANDA 1,X
DEC 0,X
CMPA #0
BEQ PASO
LSR 1,X
BRA BIT
```

Una vez que se ha logrado reconocer un evento, el número $k+1$ representado por CONT es utilizado para generar una clave ASCII que será guardada en la memoria RAM del 68HC11E9. Las diferentes claves, son almacenadas en una variable llamada ASCII.

```
PASO LDAA 0,X
```

```

STAA CONT
ADDA #30
NOCAM STAA ASCII

```

En seguida, se verifica si todavía puede haber la posibilidad de reconocer algunos eventos más en el registro, o si el evento que acaba de reconocerse a sido el último:

```

LSR 1,X
LDAA 1,X
CMPA #0

```

En caso de no haber posibilidad de reconocer un evento más, preparamos la variable **PUERTO** para hacer un nuevo reconocimiento de eventos en un registro diferente. Así mismo, hacemos que **CONT** sea igual a cero, para poder inicializarla nuevamente al valor $K+1$ en una subrutina llamada **CONTROL** que se encuentra al inicio del programa.

```

BNE MISMO
INC PUERTO+1
LSR 0,X
LDAA 0,X
STAA CONT

```

En caso contrario, si todavía tenemos que continuar haciendo reconocimiento de eventos en el registro, procedemos a salvar el valor de Z en **MASK**, antes de salir de nuestra subrutina.

```

MISMO STAA MASK
JSR MENSAJ
PULB
PULA
RTS

```

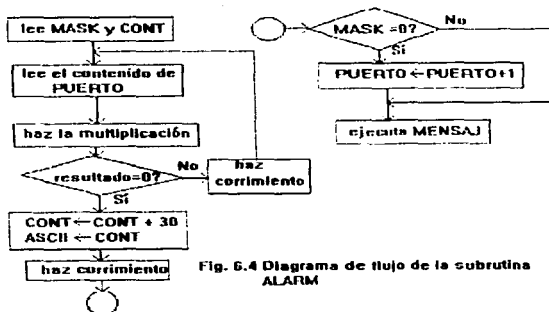


Fig. 6.4 Diagrama de flujo de la subrutina ALARM

6.3 SIMULACION DEL HARDWARE REGISTRADOR DE SECUENCIAS

En las primeras secciones de este capitulo, se hablo sobre el funcionamiento de esta circuiteria. El objetivo principal, de este hardware, es "asignar una direccion unica en el rango de 00000 a 11111 al evento N reconocido por cualquiera de los cuatro circuitos MTS y almacenar la informacion en RAM".

En nuestro programa, nombramos MENSALJ a la subrutina encargada de registrar en memoria RAM el evento N reconocido por la subrutina ALARM.

El "Contador de direcciones", es representado por una variable llamada DIREC. Esta Variable, es inicializada con la direccion 0121 en RAM:

MENSALJ LDY DIREC

```
CPY #0
BNE LECT
LDY #0121
```

En seguida, procedemos a verificar el tipo de dato que nos envia la subrutina ALARM. Es decir, si no hay reconocimiento, no hay ejecucion de MENSALJ. Por otra parte, si el reconocimiento sucedio en el "ultimo bit" (00000001) de algun puerto, procedemos a restaurar la direccion del "Selector de MTS" simulado por la variable PUERTO. Esto se debe, a que el incremento en esta variable depende del valor en el exponente del numero 2^N, pero la clave del evento reconocido es funcion del numero K+1. Por lo tanto, cuando K=0, PUERTO=PUERTO+1, pero si el evento 00000001, estuvo presente en algun puerto, MENSALJ no podra escribirlo porque el valor de PUERTO ha cambiado. Entonces, hacemos el siguiente ajuste:

```
LECT LDAA ASCII
      CMPA #30
      BEQ NADA
      CMPA #31
      BNE NORM
      LDY PUERTO
      DEX
      BRA ULTIM
```

El circuito "Selector de datos", envia una direccion en el rango de 00000 a 11111, proporcionada por el contador de direcciones con el evento, indicado por el selector de MTS. Posteriormente, el selector de datos envia esta informacion a la memoria RAM para su registro.

Nuestra subrutina MENSALJ, de acuerdo al valor de PUERTO, asigna cualquiera de las claves B1, B2, ..., B6, D1, D2, ..., D6, F1, F2, ..., F6 y E1, E2, ..., E6 al evento y luego se registran en la memoria RAM a partir de la localidad indicada por la variable DIREC, como se muestra a continuacion:

NORM LDY PUERTO


```

ULTIM CPX  ##1031
      BNE  COND2
      LDX  ##0104
      JSR  LETRA
      BRA  IDEN
COND2 CPX  ##1032
      BNE  COND3
      LDX  ##0105
      JSR  LETRA
      BRA  IDEN
COND3 CPX  ##1033
      BNE  LEY4
      LDX  ##0106
      JSR  LETRA
      BRA  IDEN
      LEY4 LDX  ##0107
      JSR  LETRA
      IDEN LDA  ASCII
      JSR  ACT
      STY  DIREC
      NADA RTS

```

Es probable que, algunas de las líneas del segmento de programa anterior, no hayan quedado muy claras. Esta subrutina, tiene anidadas otras subrutinas, y maneja direcciones que se han definido como constantes al inicio del programa principal. Sin embargo, la información que hemos proporcionado en estas secciones será de gran utilidad para comprender el propósito de las subrutinas secundarias. A continuación mostramos la lógica de funcionamiento de la subrutina MENSAJ:

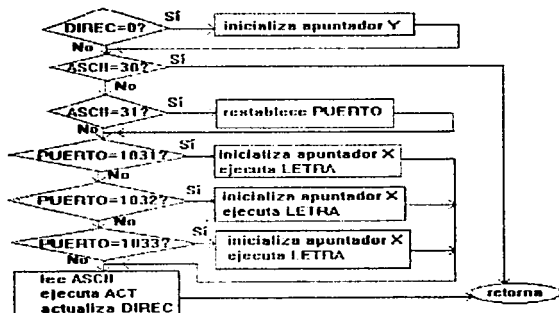


Fig. 6.5 Diagrama de flujo de la subrutina MENSAJ

6.3.1 SUBROUTINAS SECUNDARIAS

Como podemos darnos cuenta, las subrutinas ALARM y MENSAJ son de suma importancia en nuestro programa, pues tienen la tarea de simular el funcionamiento del hardware en el controlador 2400A-104/AA. Sin embargo, no pueden trabajar solas, es decir, se auxilian de dos pequeñas subrutinas para llevar a cabo sus funciones. Una de ellas, se encarga de encarga de transferir el contenido de alguna dirección indicada por el registro X a una dirección señalada por el registro Y.

La segunda, unicamente se encarga de almacenar algún valor contenido en el acumulador A en la dirección indicada por el registro Y, actualizando este al final de su ejecución. En nuestro programa, estas subrutinas se han identificado con los nombres de LETRA y ACT:

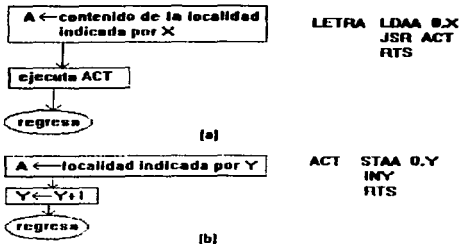


Fig. 6.6 Subrutinas secundarias en MENSAJ

6.4 SUBROUTINAS DE INICIO Y CONTROL

Cuando la subrutina ALARM... reconoce un evento, envia el parámetro ASCII a la subrutina MENSAJ para llevar a cabo el registro en RAM. Luego, el control pasa nuevamente a la subrutina ALARM para continuar con el reconocimiento de los eventos. Este proceso, debe ser constante, es decir, las 24hrs. del día.

La subrutina CONTROL, se encarga de inicializar las variables PUERTO, MASK y CONT cada vez que el programa a sido utilizado 32 veces.

Inicialmente, verificamos si nuestro programa se encuentra trabajando en alguno de los cuatro registros asociados al sensor de voltaje, de lo contrario, indicamos que lo haga en el registro ADRI:

```
CONTROL LDY PUERTO
```

```

CPY #0
BNE NEXT
LDY ##1031
STY PUERTO

```

Luego, para lograr un reconocimiento rápido, intentamos aprovechar el valor actual del factor 2^k , pero si este valor es igual a cero, entonces se comienza con el valor más alto, esto es MASK=80₁₆:

```

NEXT LDA  MASK
      CMPA #0
      BNE  NEXT2
      LDA  ##80
      STAA MASK

```

Análogamente, si utilizamos el valor de K+1 actual, el reconocimiento sería rápido si el evento ocupa una posición cercana al valor de CONT, de otra manera, tendrá que encontrarse este valor, haciendo CONT=9₁₀.

```

NEXT2 LDA  CONT
      CMPA #0
      BNE  PROX
      LDA  #9
      STAA CONT
PROX  JSR  ALARM
      RTS

```

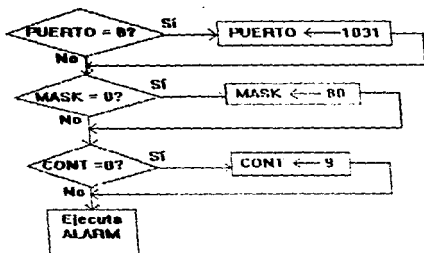


Fig. 6.7 Diagrama de flujo de la subrutina CONTROL

6.5 PROGRAMA PRINCIPAL Y LA SUBROUTINA CLARO.

En nuestro programa, utilizamos una subrutina llamada CLARO, cuya función es limpiar los valores de FF contenidos inicialmente en el área de memoria RAM donde se encuentran definidas las variables PUERTO, MASK, CONT y DIREC:

```
CLR PUERTO
CLR PUERTO+1
CLR MASK
CLR CONT
CLR DIREC
CLR DIREC+1
RTS
```



Fig. 6.8 Subrutina CLARO

Las subrutinas que han sido utilizadas, tienen una tarea específica a realizar, y se encuentran comunicadas entre sí a través de los parámetros. Sin embargo, en este tipo de programas (programas estructurados), se necesita de un "programa principal" que indique el "orden de ejecución" de cada subrutina.

Las primeras líneas que integran nuestro modulo principal, indican que la subrutina CLARO debe de ejecutarse primero y después la subrutina CONTROL:

```
JSR CLARO
SIGUE JSR CONTROL
```

Luego, pasamos a una condición, que obliga a trabajar de forma continua a las subrutinas hasta concluir la posibilidad de reconocer un mínimo de 32 eventos. Es decir, que todos los registros utilizados por nuestro dispositivo entrada/salida en el microcontrolador sean procesados.

```
LDX PUERTO
CPX #51035
BNE SIGUE
```

Posteriormente, grabamos en RAM un dato que indicará el fin de la transmisión serial:

```
LDA #504
STAA 0.Y
```

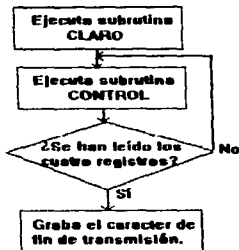


Fig. 6.9 Diagrama de flujo del módulo principal

6.6 RESULTADOS OBTENIDOS

Las dos subrutinas de simulación, ALARM y MENSAJ lograron cumplir su objetivo después de una serie de pruebas experimentales. A continuación, mostramos un ejemplo en el cual, suponemos que existe un evento N distinto en cada registro (Ver fig. 6.10).

La subrutina ALARM, se encarga de reconocer al evento y de asignarle un código ASCII de acuerdo con la posición que ocupa en el registro. Por otra parte, la subrutina MENSAJ, identifica el registro con alguna de las letras B, D, F o P. De tal manera, que se puede contar con 32 claves únicas para los 32 eventos que pudiesen presentarse en nuestro dispositivo de entrada/salida.

Cada una de estas claves, son registradas en la memoria RAM del 68HC11ES a partir de la localidad 0121 (Ver fig. 6.11).

Registro \$1031

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

Valor de ASCII al final de ALARM = 35
Letra asignada por MENSAJ = B (42=01000010)

Registro \$1032

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

Valor de ASCII al final de ALARM = 33
Letra asignada por MENSAJ = D (44=01000100)

Registro \$1033

1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Valor de ASCII al final de ALARM = 38
Letra asignada por MENSAJ = F (46=01000110)

Registro \$1034

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

Valor de ASCII al final de ALARM = 31
Letra asignada por MENSAJ = P (50=01010000)

Fig. 6.10 Un evento N en cada registro del dispositivo

```

0121 42 35 44 33 46 38 50 31   . . . . .
0122 . . . . .
0123 . . . . .

```

Fig. 6.11 Aspecto de la memoria 0121 a 0127 del microcontrolador después de ejecutarse la subrutina MENSAJ.

6.7 SUBRUTINA TIMER

Como un detalle adicional a nuestro programa, se decidió anexar una pequeña rutina llamada **TIMER** en la dirección 00D0. Esta rutina solo puede ser activada por la señal que produce el contador de 16 bits cuando alcanza la cuenta FFFF. La señal, puede ser percibida en el bit TOF del registro TFLG2 cuando cambia su estado de 1 a 0 lógico:

TIMER BRCLR 0.X,BIT7,IGNORA

donde 0.X es la dirección 1025 del registro TFLG2. **BIT7** es igual a #10000000, e **IGNORA** indica la aparición de una "interrupción ilegal", es decir, una señal desconocida que trató de activar la subrutina **TIMER**.

Cuando la interrupción es legal, el bit TOF en el registro TFLG2 debe ser regresado a su estado inicial de 1 lógico:

**LDAA #BIT7
STAA TFLG2**

Las instrucciones presentes en el cuerpo de **TIMER**, se explicarán con mayor detalle en secciones posteriores.

El propósito de **TIMER**, es registrar el "tiempo real" de los eventos, para saber con exactitud su hora de aparición durante un día de trabajo del **SEHUCLES**.

6.7.1 SUBRUTINA SUBMIL

La subrutina encargada de contar las milésimas de segundo, ha sido llamada como **SUBMIL**, y es parte de **TIMER**. Después de que se ha escrito un **LDAA** en el bit TOF del registro TFLG2 se hace un llamado a **SUBMIL**:

JSR SUBMIL

Inicialmente, comienza por comparar el número de milisegundos en su variable **MILI**, con los 999 milisegundos representados por la variable **FACTOR3**:

```

SUBMIL LDD MILI
        CPD FACTOR3

```

Si MILI > FACTOR3, entonces la ejecución pasa a la subrutina encargada de contar los segundos SUBSEG:

```

        BHI SUBSEG

```

Si MILI < FACTOR3, entonces utilizamos un factor de incremento llamado SFLUJO para incrementar la variable MILI:

```

        ADD SFLUJO
        STD  MILI

```

Teóricamente, el valor de este factor debe ser igual a 32 o 33, que es el tiempo necesario para que el contador de 16 bits cambie el estado lógico del bit TOF. Sin embargo, en la experimentación, hubo la necesidad de convertir el factor SFLUJO a una variable que oscilara entre 36 y 37 antes de abandonar SUBMIL:

```

        LDD SFLUJO
        CPD @0024
        BEQ SUMA
        DEC SFLUJO+1
        BRA SALIR
SUMA   INC SFLUJO+1
SALIR  RTS

```

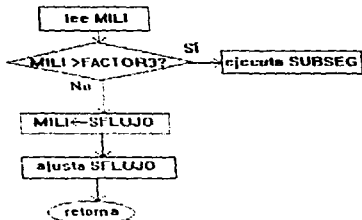


Fig. 6.12 Subrutina SUBMIL

6.8 SUBRUTINAS SUBSEG Y SUBMIN

Estas dos subrutinas, tienen la misma estructura. SUBSEG se encarga de contar los segundos y SUBMIN los minutos. En el caso

de SUBSEG, se comienza por limpiar la variable MILI, que tiene un valor inicial de 999 milisegundos y es puesto a 0 para iniciar una nueva cuenta:

SUBSEG CLR MILI

Luego, se compara el número de segundos en la variable SEG, con los 59 segundos representados por la constante FACTOR1:

**LDAA SEG
CMPA FACTOR1**

Si $SEG < FACTOR1$, entonces SEG incrementa su valor en un segundo. De lo contrario, salimos de SUBSEG:

**BEQ SUBMIN
INC SEG
RTS**

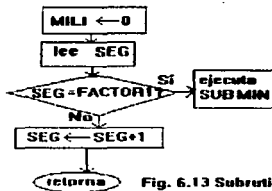


Fig. 6.13 Subrutina SUBSEG

La subrutina SUBMIN, comienza por limpiar la variable SEG, que tiene un valor inicial de 59 segundos:

SUBMIN CLR SEG

Luego, compara el número de minutos en la variable MIN, con los 59 minutos en FACTOR1:

**LDAA MIN
CMPA FACTOR1**

Si $MIN < FACTOR1$, entonces MIN incrementa su valor en un minuto. De lo contrario, salimos de SUBMIN:

**BEQ SUBHRS
INC MIN
RTS**

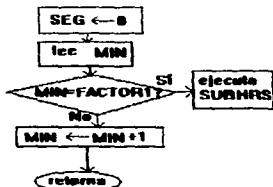


Fig. 6.14 Subrutina SUBMIN

6.9 SUBROUTINA SUBHRS

Esta subrutina, se encarga de contar las 24 Hrs. del día. Inicialmente, comienza por limpiar la variable MIN, que tiene un valor inicial de 59 minutos:

SUBHRS CLR MIN

Luego, compara el número de horas en la variable HORAS con las 23 horas representadas por la constante FACTOR2:

LDA HORAS
CMPA FACTOR2

Si HORAS < FACTOR2, entonces HORAS incrementa su valor una hora. De lo contrario, salimos de SUBHRS:

BEQ DIA
INC HORAS
BRA VUELTA
DIA CLR HORAS
RTS

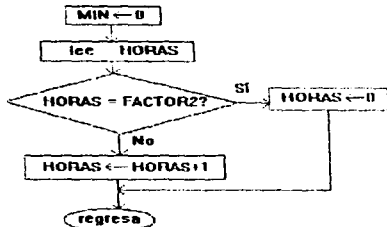


Fig. 6.15 Subrutina SUBHRS

6.10 CONVERSIONES DE CODIGO BINARIO A BCD EN TIMER

Otra tarea que tiene a su cargo la subrutina TIMER, es la de hacer conversiones continuas del código binario en las variables MILI, SEG, MIN y HORAS a código BCD. Con la finalidad, de poder presentar al usuario una información legible.

El primer paso, es limpiar el acumulador A:

```
CLRA
```

Luego, procedemos a cargar el acumulador B con el parámetro a convertir, e inicializamos el registro Y con una dirección en RAM donde podamos comenzar a registrar los valores de tiempo en código BCD, e iniciamos con la conversión:

```
LDAB HORAS  
LDY #0118  
JSR SUBCONV2
```

El proceso continúa, de manera análoga para el resto de las variables:

```
CLRA  
LDAB MIN  
LDY #011A  
JSR SUBCONV2  
CLRA  
LDAB SEG  
LDY #011C  
JSR SUBCONV2  
LDD MILI  
LDY #011E  
JSR SUBCONV1
```

6.10.1 SUBROUTINAS DE CONVERSION SUBCONV1 Y SUBCONV2

Quando hacemos un cambio de base M a N en un número D, efectuamos una división en la cual, el dividendo es el número D en base M y el divisor es la nueva base N. Luego, el "residuo" obtenido de esta operación aritmética, nos representa uno de los dígitos del número D en la nueva base N. Si deseamos obtener el resto de los dígitos en la base N del número D, continuamos realizando más divisiones entre los cocientes y la base N.

El cambio de base que realizan las subrutinas SUBCONV1 y SUBCONV2, es de la base 16 a 10 en las variables HORAS, MIN, SEG, y MILI.

SUBCONV1 realiza una división más que SUBCONV2, ya que la variable MILI es mayor en una decena que las demás.

Comenzamos por definir un divisor de 100₁₀, inicializando el registro Y con 64₁₆:

```
LDX #364
```

La variable MILI es enviada como un parámetro a la subrutina

SUBCONV1 en el acumulador D. por lo que efectuamos la división D/X:

IDIV

Posteriormente, transferimos el residuo (dígito de MILI en base 10) a la localidad indicada por el parámetro Y:

XGDX
STAB 0,Y

Para continuar nuestro algoritmo y poder obtener el resto de los dígitos en base 10 de la variable MILI, hacemos que el nuevo dividendo sea el cociente obtenido en la división anterior, y utilizamos como divisor el número 10_{10} :

XGDX
LDX #SA

Continuamos con la misma manera, hasta obtener la unidad como cociente:

IDIV
XGDX
STAB 1,Y
XGDX
STAB 2,Y
RTS

La subrutina SUBCONV2 hace lo mismo que SUBCONV1, con la diferencia de realizar una división menos. Esta subrutina, es aplicada a las variables HORAS, MIN y SEG:

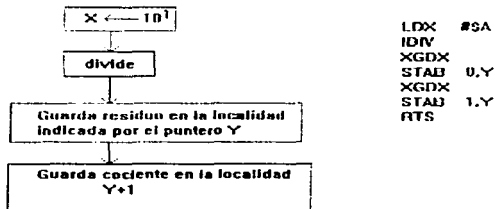


Fig. 6.16 Subrutina SUBCONV2

6.10.2 SUBROUTINA CODIGO EN TIMER

Con la finalidad de proporcionar una información legible al usuario en un "protocolo de comunicación", es decir, cuando un programa en computadora trabaja con datos enviados por un dispositivo en comunicación serial. Se hace necesario manipular toda la información digital en código ascii. por esta razón, TIMER activa la subrutina CODIGO para convertir los valores de HORAS.MIN.SEG. y MILI en BCD a código ascii:

JSR CODIGO

Se comienza por cargar al acumulador B con un contador igual al número de dígitos que integran la hora en tiempo real:

```
LDAB #09
```

Luego, enviamos el primer dígito de la variable HORA como parametro a la subrutina de conversión AUX:

```
NIDO LDAA 0,Y  
JSR AUX
```

Posteriormente, colocamos en el registro Y el valor de la dirección donde se encuentra el segundo dígito de la variable HORA:

```
INY
```

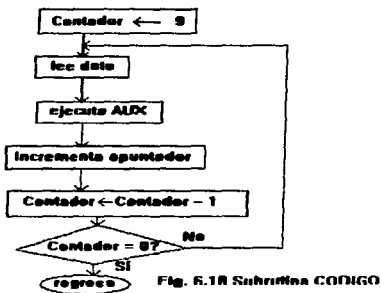
El algoritmo continua hasta completar los nueve dígitos, es decir, hasta que el contador en el acumulador B sea cero:

```
DECB  
BNE NIDO  
RTS
```

Si el microcontrolador se detiene a las 10:00:50:131 Hrs., el aspecto de la memoria RAM a partir de la localidad 0118 en el 68HC11E9 sería el siguiente:

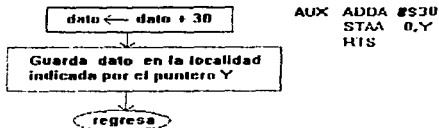
```
0118 31 33 30 39 35 30 32 33 31 xx xx xx xx xx xx xx  
0120 xx xx xx . . .  
.  
.  
.
```

Fig. 6.17 Variables HORAS,MIN,SEG y MILI



6.10.3 SUBROUTINA AUXILIAR

Esta subrutina, llamada AUX, tiene la tarea de sumar un 30_{16} a cada uno de los parámetros que le son enviados por CODIGO y almacenarlos en la dirección indicada por el registro Y:



6.11 PROGRAMA BUFFALO EN EL 68HC11E9

El chip 68HC11E9, tiene grabado un programa en su memoria ROM un llamado BUFFALO (Bit User Fast Friendly Aid to Logical Operation). Este programa, consta de cinco partes:

- 1) Inicialización.
- 2) Interprete de comandos.
- 3) Rutinas de entrada/salida.
- 4) Subrutinas de utilería.
- 5) Tabla de comandos.

En la inicialización, BUFFALO coloca todos los registros y dispositivos en condiciones normales ó de RESET, además de configurar su hardware para comunicación con la interfaz RS232.

El intérprete de comandos recibe los caracteres ASCII del comando pulsado en el teclado para activar la subrutina interna correspondiente.

Cuando utilizamos la interfaz RS232, BUFFALO utiliza una serie de subrutinas llamadas "driver subroutines", las cuales, se encuentran clasificadas de acuerdo con tres rutinas principales de entrada/salida llamadas INIT, INPUT y OUTPUT.

Las rutinas de utilería se encuentran disponibles para el usuario, todas ellas realizan operaciones de entrada/salida en la interfaz RS232.

Para identificar una tabla de comandos en BUFFALO, se utilizan tres pseudoinstrucciones, para identificar las tres partes que forman un comando, estas partes son, el número de caracteres del comando, el nombre y la dirección de inicio de la subrutina.

6.12 SUBROUTINA OUTSTRG EN TIMER

La subrutina TIMER en nuestro programa, activa una subrutina de utilería en BUFFALO, llamada OUTSTRG, la cual, envía una serie de caracteres en RAM a partir de la localidad indicada por el registro X a través de la interfaz RS232:

```

LDX #S0118
JSR OUTSTRG
IGNORA RTI
END

```

6.13 INICIALIZACION DEL RELOJ

Podemos asignar los valores iniciales de las variables HORAS, MIN.SEG y MILI para que el reloj simulado por el programa TIMER, inicie su conteo a partir de una hora predeterminada. La subrutina encargada de llevar a cabo esta tarea, es la subrutina SUBINI. Los valores de MILI y SEG varían rápidamente, por lo que se les asigna una valor de cero:

```

SUBINI CLR MILI
CLR SEG

```

Posteriormente, asignamos los valores deseados *valor1* y *valor2* en base 16 en las variables MIN y HORAS respectivamente:

```

LDA #valor1
STAA MIN

```

```
LDA #valor2
STAA HORAS
```

36: La última parte de SUBINI, inicializa la variable SFLUJO con

```
LDD #80024
STD SFLUJO
RTS
```

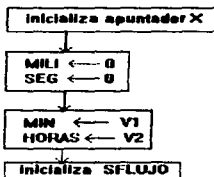


Fig. 6.20 Subrutina SUBINI

CAPITULO VII. PRESENTACION DE RESULTADOS

7.1 GENERALIDADES

Una vez que se ha logrado tener un registro de eventos en una memoria RAM. Se hace necesario, presentar los resultados en pantalla o display, de tal manera, que el usuario pueda interpretar de alguna manera la información capturada por el dispositivo. La presentación de resultados, en la mayoría de los registradores, se lleva a cabo a través de números. Por ejemplo, si en un display, aparecen los números 27, 2, 2, 6, esto nos indicará la aparición de seis eventos. Luego, procedemos a consultar una tabla, en la cual, podemos relacionar los números con sus eventos respectivos.

Desde el punto de la presentación incluyamos casos, es decir mostramos directamente la tabla de consulta en monitor, así como la hora exacta de las eventualidades.

Independientemente del formato elegido, cualquier presentación requiere de un protocolo de comunicación entre el dispositivo registrador y una unidad lógica de procesamiento con periféricos de salida. En este capítulo, comenzamos por describir el formato de presentación en el display 1720A de la mainframe 2400A en el registrador de eventos patentado por la compañía Jhon Fluke MFG. CO. INC., con el propósito de ilustrar el concepto de "presentación de resultados". Posteriormente, hablaremos sobre la presentación obtenida con un programa de computadora en el lenguaje BASIC que nos muestra las eventualidades registradas en la memoria RAM del 68HC11E9.

7.2 PRESENTACION EN EL DISPLAY 1720A

La comunicacion entre el controlador 2400A-104/AA y la unidad lógica mainframe 2400A, se hacia a través de la interfaz IEEE-488 Port 0. Es decir, la dirección IEEE-488 de la unidad lógica se configuraba a 00000.

Luego, se cargaba el programa "A104AB", el cual almacenaba en un arreglo de 32 elementos representado por la variable SI(x) los 32 estados lógicos en la RAM del controlador mediante la instruccion SEND SI(n)!!:

```
560 DIM SI(31%)
    INIT PORT0
    FOR J% = 0% TO 31%
        SEND SI(J%)!
        INPUT SI(J%)
    NEXT J%
```

Luego, cada uno de los estados lógicos se presentaban en una matriz de 4x8 elementos (Ver fig.7.1), utilizando la instruccion CPOS(M,N), donde M es el número de columnas y N el número de renglones:

```
PRINT CPOS (1,20); "2400A-104/AA TEST"
PRINT CPOS (3,21); "Touch screen";CPOS (4,21);"to display";
PRINT CPOS (5,21); "channel no.":
FOR I% = 0% TO 7%
PRINT USING "***", CPOS(I%+1%,2%);SI(I%);SI(I%+8%);SI(I%+16);
SI(I%+24%);
NEXT I%
```

Después de accionar un switch en el display 1720A, se podía mostrar una tabla de números para relacionarlos con la matriz de estados lógicos:

```
ON KEY GOTO 880
OFF KEY
880 PRINT CPOS(1%,20%);" 0  8  16  24 "
    PRINT CPOS(2%,20%);" 1  9  17  25 "
    PRINT CPOS(3%,20%);" 2 10 18  26 "
    PRINT CPOS(4%,20%);" 3 11 19  27 "
    PRINT CPOS(5%,21%);" 4 12 20  28 "
    PRINT CPOS(6%,22%);" 5 13 21  29 "
    PRINT CPOS(7%,23%);" 6 14 22  30 "
    PRINT CPOS(8%,24%);" 7 15 23  31 "
ON KEY /WAIT/GOTO 560
```


0 0 1 0	2400A-1840A TEST
0 0 1 0	
0 0 0 0	
1 0 0 0	Touch screen
0 0 0 0	Te display
0 0 0 1	
0 0 0 0	Channel No. 'S
0 0 0 0	
	SWITCH

Fig. 7.1 Presentación en el display 1720A

7.3 PRESENTACION EN UN MONITOR DE COMPUTADORA

La comunicación entre el 68HC11E9 y la computadora, se hace a través de la interfase RS232. Una vez que el microcontrolador ha registrado el código ascii de los eventos, y su tiempo real en la memoria RAM, esta información puede ser recibida por un canal de comunicación en una computadora personal mediante la instrucción OPEN COM. Así mismo, podemos asignar la información, en una variable de tipo STRING llamada `eventos$` con la función INPUT\$:

```
CLS
OPEN "COM2:9600,N,8,1." FOR RANDOM AS #1
COM(2) ON
eventos$ = INPUT$(35.1)
CLS
```

Luego, creamos una tabla con cuatro arreglos alfanuméricos STRING de ocho elementos cada uno, asignando a cada elemento una leyenda completa referente a un evento en particular:

```
PRINT "NO.":TAB(20):"LISTA DE EVENTOS":TAB(66):"HORA"
PRINT "----":TAB(20):"-----":TAB(66):"-----"
PRINT TAB(140):"HRS MIN SEG MSEG"
D$(1)-"Baja presión de aceite de lubricación (PAL-C051)"
D$(2)-"Alta temp. de aceite de lubricación (TAH-C001)"
D$(3)-"Alto nivel del condensador (LAH-C010)"
D$(4)-"Alta vibración del compresor (UAH-C031)"
D$(5)-"Alta temp. de chumaceras radial (TAH-C031)"
D$(6)-"Alta temp. en escape de turbina (TIAH-C017)"
D$(7)-"Muy baja presión de aceite de lubricación (PALL-C051)"
D$(8)-"Paro de turbina"
F$(1)-"Bajo nivel del condensador (LAL-C011)"
F$(2)-"Alta temp. chumaceras de empuje (TAH-C032)"
```

F8(3)="-Baja presión aceite del gobernador (PAL-C052)"
 F8(4)="-Arr. bomba aux. de aceite de lubricación (PAL-C001)"
 F8(5)="-Sobrevelocidad del compresor"
 F8(6)="-Alto movimiento axial (2AH-C031)"
 F8(7)="-Alto nivel B.P. succión domo (LAH-124)"
 F8(8)="-Alto nivel en trampa de drenaje lado descarga (LAH-C003)"
 P8(1)="-Alta presión diferencial ac. de lubricación (PDAH-C001)"
 P8(2)="-Oper. bomba aux. aceite de lubricación (PAH-C001)"
 P8(3)="-Disparo por sobrevelocidad del compresor"
 P8(4)="-Muy alto movimiento axial del compresor (2AMH-C031)"
 P8(5)="-Muy alto nivel de succión domo (LAHH-126A)"
 P8(6)="-Disturbio común en compresor (XA-202A)"
 P8(7)="-Bajo nivel TQ. de aceite de lubricación (LAL-C001)"
 P8(8)="-Baja presión diferencial aceite de sellos (PDAL-C001)"
 B8(1)="-Alta temp. descarga gas compresor (TAH-C203)"
 B8(2)="-Alta presión desfogue turbina (PAH-C017)"
 B8(3)="-Bajo nivel B.P. succión domo (LAL-C124)"
 B8(4)="-Disparo del compresor (XAH-203A)"
 B8(5)="-Muy baja presión de aceite de sellos (PDALL-C001)"
 B8(6)="-Valv. de control TTV posición cerrada (2A-C002)"
 B8(7)="-Muy alto nivel interfase domo (LAHH-129)"
 B8(8)="-Baja presión purga aire tablero (PAL-C031)"

Posteriormente, seleccionamos de **eventos**, la parte correspondiente al código ascii que nos representa la hora en tiempo real, para hacer una asignación a la variable llamada **horas** :

horas = MIDS(eventos\$.4.10)

De la cadena ascii representada por **horas**, asignamos las variables **HORAS**, **MIN**, **SEG** y **MILI** a las variables **HH\$**, **MM\$**, **SS\$** y **MS\$** respectivamente:

HH\$ = MIDS(horas\$.1.2)
MM\$ = MIDS(horas\$.3.2)
SS\$ = MIDS(horas\$.5.2)
MS\$ = MIDS(horas\$.7.3)

Para poder mostrar una presentación completa, es decir, visualizar número, leyenda y hora de cada evento (Ver fig. 7.2), en monitor, inicializamos un contador **i** con la posición de la cadena ascii en la variable **eventos\$**:

i = 14

A partir de esta posición, comenzamos a seleccionar de la tabla, aquellos eventos que nos indique la variable **vp\$**. Por cada evento seleccionado, nuestro programa envía número leyenda y hora al monitor. Este proceso, termina cuando el valor del contador **i** excede el número de caracteres en la variable **eventos\$** :

NO.	LISTA DE EVENTOS	HORA		
		HRS	MIN	SEG MSEG
29	Muy baja presión de aceite de sellos (PDALL-C001)	13:09:50	231	
3	Alto nivel del condensador (LAH-C010)	13:09:50	231	
16	Alto nivel en trampa de drenaje lado descarga (LAH-C003)	13:09:50	231	
17	Alta presión diferencial ac. de lubricación (PDAH-C001)	13:09:50	231	

Fig. 7.2 Presentación en un monitor de computadora

DO

```

vps = MIDS (eventoss,1,2)
SELECT CASE vps
CASE "D1"
PRINT "1";D$(1);" ";HH$:" ":"MM$:" ":"SS$:" ":"MS$
CASE "D2"
PRINT "2";D$(2);" ";HH$:" ":"MM$:" ":"SS$:" ":"MS$

.
.
.

CASE "B0"
PRINT "32";B$(8);" ";HH$:" ":"MM$:" ":"SS$:" ":"MS$
END SELECT
i = i + 2
LOOP UNTIL i > LEN (eventoss)
END

```

CONCLUSIONES

El registrador de eventos, implementado con el 69HC11E9, fue sometido a distintas pruebas experimentales durante su desarrollo. Los resultados obtenidos en cada etapa, fueron muy satisfactorios. Entre estos, se logró optimizar el espacio en memoria RAM, ya que las eventualidades fueron identificadas por una clave ascii mediante el software en el microcontrolador, y no por una dirección de memoria como se hacía en el controlador 2400A-104/AA. Por otra parte, la presentación de resultados fue mejorada en gran medida, puesto que cualquier persona puede interpretar fácilmente las eventualidades por su nombre y no por la posición de un número binario o en un display.

En lo referente al registro en tiempo real, el controlador 2400A-104/AA estaba programado para actualizar un registro de 32 bits, el cual, nos podía representar mediante un número binario una cuenta máxima de 999 999.9 segundos que equivalen a 277 Hrs. ó 11 1/2 días (Ver fig. 7.3). El sobreflujo en este registro podía apreciarse con un 1 lógico en el bit 31, la posición del punto decimal mediante el código binario en los bits 28 a 30 y el código BCD de cada dígito en los bits 0 a 27.

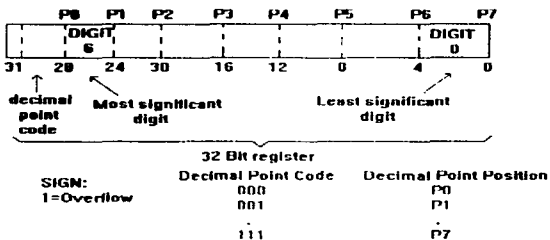


Fig. 7.3 Formato en el 2400A

Con el 69HC11E9, se logró registrar el tiempo real desde 32 milisegundos de segundo hasta un día como límite máximo. Esto se hizo con la actualización de estas localidades de memoria RAM, presentando los resultados de manera adecuada en un monitor de computadora (Ver fig. 7.4). Debemos señalar, que esta limitante fue debida a falta de memoria EEPROM, ya que de lo contrario podía haber existido la posibilidad de actualizar tres localidades mas en RAM que nos hubiesen representado el número de días, meses y años. Sin embargo, este problema puede ser resuelto colocando un chip de memoria en la zona wire-wrap de la tarjeta EVBU, configurando a la misma de tal manera que parte del código

de programación en EEPROM pueda ser transferido a este nuevo chip mediante el comando MOVE del programa BUFFALO y tener la posibilidad de aumentar el número de instrucciones en la zona de programación para un registro en RAM valores de tiempo real mucho más amplios. O bien, habilitando el chip 68HC1168TIP (reloj en tiempo real) montado en la tarjeta EVBU y liberando el espacio de memoria en EEPROM utilizado en la programación de la subrutina TIMER.

HORA

HRS MIN SEG MSEG
 13 : 09 : 50 : 231

Fig. 7.4 Presentación en monitor

Este proyecto de tesis, fué desarrollado a nivel de laboratorio, con todas las pruebas pertinentes. Para poder ser instalado definitivamente en el turbogenerador de 300MW, en la sección de ciclo combinado, sólo resta montar las piezas de los convertidores D/A en un circuito impreso (Ver fig. 7.5) con una entrada especial, donde puedan ser recibidas las 32 señales digitales provenientes del conector 2400A-104/AB-4002 en el acondicionador de señal 2400A-104/AB.

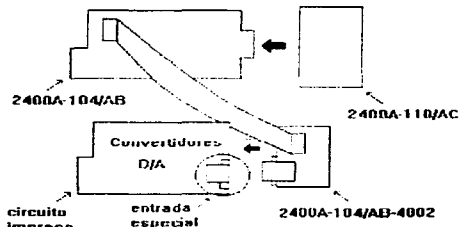


Fig. 7.5 Ultima paso a realizar para instalación

APENDICE A

"LISTADOS"

* ESTE PROGRAMA SIMULA UN REGISTRADOR DE 22 EVENTOS

 * HABILITA LOS REGISTROS

PORTE	EQD	41004
TMSK2	EQD	41004
TFLG2	EQD	41005
ADCTL	EQD	41009
OPTION	EQD	41009
OUTSTR	EQD	41007
BIT7	EQD	210000000

* TABLA DE CONSTANTES

	ORG	40100
FACTOR1	FCB	405
FACTOR2	FCB	417
FACTOR3	FCB	40007
B	FCB	440
C	FCB	444
D	FCB	446
E	FCB	450

* CREA UN ESPACIO DE MEMORIA PARA ALMACENAMIENTO DE VARIABLES

	ORG	40100
BUERTO	RMB	0
NASH	RMB	1
CONT	RMB	1
ASCI	RMB	1
TIRES	RMB	2
MILO	RMB	2
TA	RMB	1
TA	RMB	1
TA	RMB	1
TA	RMB	2

* TABLA DE RE-ENTRACION

ORG	40100
ORG	40100

* DEFINICION DE LOS REGISTROS

ORG	40100
ORG	4047
ORG	4050
ORG	4050
ORG	4050
ORG	4050
ORG	4050
ORG	4050

* TIEMPO DE OPERACION DEL CONVERTIDOR

ETA LDAB #11A
 DECB
 BNE RETH

* CAPTURA LAS SEÑALES

SIGUE JSR C:ARQ
 JSR CONTROL
 LDY PUERTO
 CPX #11000
 BNE SIGUE
 LDAA #604
 STAA O:Y

* INICIALIZA EL RELOJ

 JSR SUBINI

* HABILITA LAS INTERRUPCIONES

 CLI

* ESPERA UNA INTERRUPCION

 BRA SIGUE

* RUTINA DE INTERRUPCION

IMER BPLR C:Y:BIT:IGNORA
 LDAA #E100
 STAA #F000
 JSR SUBINI
 CLRA
 ENDE
 LDY AD:100
 CLR C:Y:BIT:IGNORA
 JSR SUBINI
 LDAB #10
 LDY #E100
 JSR SUBINI
 JSR SUBINI
 LDAB #00
 CLR C:Y:BIT:IGNORA
 JSR SUBINI
 LDY #E100
 JSR SUBINI
 LDY #E100
 JSR SUBINI
 LDA #E100
 JSR SUBINI
 STY C:Y:BIT:IGNORA
 ENI

IGNORA

SUBROUTINA QUE MARCA EL BIT DE CLEADA EN LOS REGISTROS ADAL PARA


```

CONDS  BRA IDEN
      CFX ##1033
      BNE LEY4
      LDX ##0106
      JSR LETRA
      BRA IDEN
LEY4   LDX ##0107
      JSR LETRA
IDEN   LDA# ASCII
      JSR ACT
      STY DIREC
NADA   RTS

```

* SUBROUTINA QUE LIMPIA EL ESPACIO DE MEMORIA RESERVADO PARA LAS VARIABLES
* Y MENSAJES

```

CLARO  CLR PUERTO
      CLR PUERTO+1
      CLR MASC
      CLR CONT
      CLR DIREC
      CLR DIREC+1
      RTS

```

* SUBROUTINA QUE SE ENCARGA DE CONTROLAR LA EJECUCION DEL PROGRAMA

```

CONTROL  LDY PUERTO
      CPY #0
      BNE NEXT
      LDY ##1031
      STY PUERTO
NEXT     LDA# MASC
      CMA #1
      BNE NEXTC
      LDA# BPRO
      STAA MASC
NEXTC   LDA# CONT
      CMA #0
      BNE BPRO
      LDA #1
      STAA CONT
BPRO    CMA ALARM
      RTS

```

* SUBROUTINA QUE GENERA EL CODIGO ASCII DE LOS MENSAJES

```

LETRA   LDA# #13
      CMA #1
      RTS

```

* SUBROUTINA QUE ACTUALIZA AL APUNTAADOR Y

```

ACT     STAA #Y
      INY
      RTS

```

* SUBROUTINA QUE CUENTA LOS MILLISEGUNDOS

```

SUBRIL  LDD  MILI
        CPD  FACTOR3
        BHI  SUBSEG
        ADDD SFLUJ0
        STD  MILI
        LDD  SFLUJ0
        CPD  #00024
        SEQ  SUMA
        DEC  SFLUJ0+1
        BNA  SALIR
SUMA    INC  SFLUJ0+1
SALIR  RTS

```

* SUBROUTINA QUE CUENTA LOS SEGUNDOS

```

SUBSEG  CLR  MILI
        LDAA SEC
        CMPA FACTOR1
        SEQ  SUBMIN
        INC  SEG
        RTS

```

* SUBROUTINA QUE CUENTA LOS MINUTOS

```

SUBMIN  CLR  SEC
        LDAA MIN
        CMPA FACTOR1
        SEQ  SUBHR3
        INC  MIN
        RTS

```

* SUBROUTINA QUE CUENTA LAS HORAS

```

SUBHR3  CLR  MIN
        LDAA HORAS
        CMPA FACTOR3
        SEQ  FIN
        INC  HORAS
        BNA  VUELTA
FIN      RTS
VUELTA  RTS

```

* SUBROUTINA QUE MUESTRA EL VALOR ABSOL DEL TIEMPO CAPTURADO

```

CODIGO  LDAA  ABS
NULO    LDAA  0
        CBR  ABS
        BNY
        DEC  ABS
        ANE  MILI
        BNE  NULO
        RTS

```

* SUBROUTINA QUE INICIALIZA LAS VARIABLES Y CONSTANTES

```

INICI1  LDA  #1000
        CLR  MILI
        CLR  SEC
        RTS

```

```
LDAA #011
STAA MIN
LDAA #00F
STAA HORAS
LDD #00024
STD SFLUJC
RTS
```

* SUBROUTINA QUE SUMA 01 PARA GENERAR EL CODIGO ASCII

```
AUX      ADDA #000
         STAA 0.1
         RTS
```

* SUBROUTINA QUE REALIZA LA CONVERSION BCD-DECIMAL DE MILLISEGUNDOS

```
SUBCONV1 LDX #064
         IDIV
         XGDX
         STAB 0.1
         XGDX
         LDX #5A
         IDIV
         XGDX
         STAB 1.1
         XGDX
         STAB 0.1
         RTS
```

* SUBROUTINA QUE REALIZA LA CONVERSION BCD-DECIMAL DE SEGUNDOS Y MINUTOS

```
SUBCONV2 LDX #0A
         IDIV
         XGDX
         STAB 0.1
         XGDX
         STAB 0.1
         RTS
```


CASE "D3"
PRINT "3"
CASE "D4"
PRINT "4"
CASE "D5"
PRINT "5"
CASE "D6"
PRINT "6"
CASE "D7"
PRINT "7"
CASE "D8"
PRINT "8"
CASE "F1"
PRINT "9"
CASE "F2"
PRINT "10"
CASE "F3"
PRINT "11"
CASE "F4"
PRINT "12"
CASE "F5"
PRINT "13"
CASE "F6"
PRINT "14"
CASE "F7"
PRINT "15"
CASE "F8"
PRINT "16"
CASE "F1"
PRINT "17"
CASE "F2"
PRINT "18"
CASE "F3"
PRINT "19"
CASE "F4"
PRINT "20"
CASE "F5"
PRINT "21"
CASE "F6"
PRINT "22"
CASE "F7"
PRINT "23"
CASE "F8"
PRINT "24"
CASE "B1"
PRINT "25"
CASE "B2"
PRINT "26"
CASE "B3"
PRINT "27"
CASE "B4"
PRINT "28"
CASE "B5"
PRINT "29"
CASE "B6"
PRINT "30"
CASE "B7"
PRINT "31"

```
CASE "B6"  
PRINT "B2 " ; B$(B) ; " " ; HH$ ; " " ; MM$ ; " " ; SS$ ; " " ; MS$  
END SELECT  
i = i + 2  
LOOP UNTIL i > LEN(eventos$)  
END
```

APENDICE B

**"TABLAS DE REGISTROS E
INSTRUCCIONES DEL
68HC11E9"**

INSTRUCTIONS, ADDRESSING MODES, AND EXECUTION TIMES

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cyc U	Condition Codes								
				Opcode	Operand(s)			S	X	H	N	Z	V	C		
ABA	Add Accumulators	A - B → A	INH	1B		1	2									
ABX	Add B to X	IX - 00 B → IX	INH	3A		1	3									
ABY	Add B to Y	IY - 00 B → IY	INH	18 3A		2	2									
ADCA (opt)	Add with Carry to A	A - M - C → A	A IMM A DIR A EXT A IND X A IND Y	89 99 B9 A9 18 A9	ii ds' i hh ii ff ff	2 2 3 2 3	2 2 4 4 5									
ADCB (opt)	Add with Carry to B	B - M - C → B	B IMM B DIR B EXT B IND X B IND Y	C9 D9 F9 E9 18 E9	ii dd hh ii ff ff	2 2 3 2 3	2 2 4 4 5									

Source Format	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Byte Cycle		Condition Codes										
				Opcode	Operand(s)			S	X	H	I	N	Z	V	C			
ADDA (opt)	Add Memory to A	A ← M ← A	A (MM)	8B	"	2	2											
			A (DR)	9B	dh	2	3											
			A (EXT)	BB	hh ll	3	4											
			A (IND, X)	AB	"	2	5											
			A (IND, Y)	1B AB	H	3	5											
ADDR (opt)	Add Memory to B	B ← M ← B	B (MM)	CB	"	2	2											
			B (DR)	DB	dh	2	3											
			B (EXT)	FB	hh ll	3	4											
			B (IND, X)	AB	ff	2	4											
			B (IND, Y)	1B FB	H	3	5											
ADDI (opt)	Add 16 Bits to D	D ← M ← I ← D	(MM)	C3	ff hh	3	4											
			(DR)	D3	dh	2	5											
			(EXT)	F3	hh ll	3	6											
			(IND, X)	E3	ff	2	6											
			(IND, Y)	1B E3	ff	3	7											
ANDA (opt)	AND A with Memory	A ← M ← A	A (MM)	84	"	2	2											
			A (DR)	94	dh	2	3											
			A (EXT)	B4	hh ll	3	4											
			A (IND, X)	A4	ff	2	4											
			A (IND, Y)	1B A4	ff	3	5											

ANDB (opt)	AND B with Memory	B ← M ← B	B IMM B DIR B EXT B IND,X B IND,Y	C4 D4 F4 E4 18 64	# dst hh ll H H	2 3 3 4 4 5	2 3 3 4 4 5	1 1 1 1 1 1	0 0 0 0 0 0
ASL (opt)	Arithmetic Shift Left		EXT IND,X IND,Y A INH B INH	78 68 18 68 48 58	hh ll H H H H	3 2 3 1 1	6 6 7 2 2	1 1 1 1 1	1 1 1 1 1
ASL ←									
ASL ←									
ASL ←									
ASLD	Arithmetic Shift Left Double		IMH	65	H	1	3	1	1
ASR (opt)	Arithmetic Shift Right		EXT IND,X IND,Y A INH B INH	77 67 18 67 47 57	hh ll H H H H	3 2 3 1 1	6 6 7 2 2	1 1 1 1 1	1 1 1 1 1
ASR ←									
ASR ←									
ASR ←									
BCC (opt)	Branch if Carry Clear	← C ← 0	REL	24	rr	2	3	2	3
BCLR (opt) (msb)	Clear Bit(s)	M ← 0	DIR IND,X IND,Y	15 10 18 10	dst H H H	3 2 3 4	6 6 7 8	1 1 1 1	0 0 0 0
BCC (msb)	Branch if Carry Set	← C ← 1	REL	25	rr	2	3	2	3
BEO (opt)	Branch if Zero	← Z ← 1	REL	27	rr	2	3	2	3

Source Format	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		S	X	H	I	N	Z	V	C	
				Opcode	Operand(s)									
BGE (rel)	Branch if \geq Zero	$\neg N \oplus V = 0$	REL	2C	rr	2	3							
BGT (rel)	Branch if $>$ Zero	$\neg Z \wedge (N \oplus V) = 0$	REL	2E	rr	2	3							
BHI (rel)	Branch if Higher	$\neg C \wedge Z = 0$	REL	22	rr	2	3							
BHS (rel)	Branch if Higher or Same	$\neg C = 0$	REL	24	rr	2	3							
BITA (opt)	Bits Test A with Memory	A=M	A INM	85	rr	2	2						I I 0	
			A DIR	86	rr	2	3							
			A EXT	8B	rr, R	3	4							
			A IND, X	85	rr	2	4							
			A IND, Y	8B	rr	3	5							
BITB (opt)	Bits Test B with Memory	B=M	B INM	C5	rr	2	2						I I 0	
			B DIR	D5	rr	2	3							
			B EXT	F5	rr, R	3	4							
			B IND, X	E5	rr	2	4							
			B IND, Y	FB	rr	3	5							
BLE (rel)	Branch if \leq Zero	$\neg Z \wedge (N \oplus V) = 1$	REL	2F	rr	2	3							
BLO (rel)	Branch if Lower	$\neg C = 1$	REL	25	rr	2	3							
BLS (rel)	Branch if Lower or Same	$\neg C \wedge Z = 1$	REL	23	rr	2	3							

BLT (rel)	Branch if - Zero	?N Φ V - 1	REL	2D	rr	2	3	
BMI (rel)	Branch if Minus	?N - 1	REL	2B	rr	2	3	
BNE (rel)	Branch if Not - Zero	?Z - 0	REL	26	rr	2	3	
BPL (rel)	Branch if Plus	?N 0	REL	2A	rr	2	3	
BRA (rel)	Branch Always	?1 - 1	REL	20	rr	2	3	
BRL (r/m)	Branch if Bit(s) Clear	?1*mm - 0	DIR	13	ddd mm rr	4	6	
(m/s)			IND.X	1F	ff mm rr	4	7	
(rel)			IND.Y	1B 1F	ff mm rr	5	8	
BRN (rel)	Branch Never	?1 0	REL	21	rr	2	3	
BRS (r/m)	Branch if Bit(s) Set	?1M*mm - 0	DIR	12	ddd mm rr	4	6	
(m/s)			IND.X	1E	ff mm rr	4	7	
(rel)			IND.Y	1B 1E	ff mm rr	5	8	
BSE (r/m)	Set Bit(s)	M*mm - M	DIR	14	ddd mm	3	6	1 1 0
(m/s)			IND.X	1C	ff mm	3	7	
			IND.Y	1B 1C	ff mm	4	8	
BSR (rel)	Branch to Special Op	See Special Ops	REL	8D	rr	2	5	
PVC (rel)	Branch if Overflow Clear	?V - 0	REL	28	rr	2	3	
BVS (rel)	Branch if Overflow Set	?V - 1	REL	29	rr	2	3	
CBA	Compare A to B	A - B	IFH	11		1	2	1 1 1 1
CLC	Clear Carry Bit	0 - C	IFH	0C		1	2	0
CLI	Clear Interrupt Mask	0 - I	IFH	0E		1	2	0

ESTD 1958
 SAMSUNG ELECTRONICS CO. LTD.
 SAMSUNG ELECTRONICS CO. LTD.

Source Formist	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		e	G	V	C	Condition Codes							
				Opcode	Operandisl					S	X	H	I	N	Z	V	C
CLR (opri)	Clear Memory Byte	0 ← M	EXT	7F	hh ll	3	6					0	1	0	0		
			IND,X	6F	ll	2	6										
			IND,Y	18 6F	ll	3	7										
CLRA	Clear Accumulator A	0 ← A	A INH	4F		1	2						0	1	0	0	
CLPB	Clear Accumulator B	0 ← B	B INH	5F		1	2							0	1	0	0
CLV	Clear Overflow Flag	0 ← V	INH	0A		1	2									0	
CMPA (opri)	Compare A to Memory	A - M	A INM	81	ll	2	2										
			A DIR	91	dl	2	3										
			A EXT	B1	hh ll	3	4										
			A IND,X	A1	ll	2	4										
			A IND,Y	18 A1	ll	3	5										
CMPB (opri)	Compare B to Memory	B - M	B INM	C1	ll	2	2										
			B DIR	D1	dl	2	3										
			B EXT	F1	hh ll	3	4										
			B IND,X	E1	ll	2	4										
			B IND,Y	18 E1	ll	3	5										
COM (opri)	1's Complement Memory Byte	1FF M ← M	EXT	73	hh ll	3	6									1	
			IND,X	63	ll	2	6										
			IND,Y	18 63	ll	3	7										

LCMM	1's Complement A	SFF - A - A	A INH	43		1	2		1	0	1	
COMB	1's Complement B	SFF - B - B	B INH	53					1	1	0	1
CPD (opt)	Compare D to Memory 16 Bit	D - M M + 1	IMM DIR EXT IND.X IND.Y	1A B3 1A 93 1A B3 1A A3 CD A3	jj kk dd hh # # #	4 3 4 3 3	5 6 7 7 7		1	1	1	1
CPX (opt)	Compare X to Memory 16 Bit	X - M M + 1	IMM DIR EXT IND.X IND.Y	8C 9C BC AC CD AC	jj kk dd hh # # #	3 2 3 2 3	4 5 6 6 7		1	1	1	1
CPY (opt)	Compare Y to Memory 16 Bit	Y - M M + 1	IMM DIR EXT IND.X IND.Y	18 8C 18 9C 18 BC 1A AC 18 AC	jj kk dd hh # # #	4 3 4 3 3	5 6 7 7 7		1	1	1	1
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19				1	1	1	1	1
DEC (opt)	Decrement Memory Byte	A - 1 - A	EXT IND.X IND.Y	7A 6A 18 6A	hh # # #	3 2 3	6 6 7		1	1	1	1
DECA	Decrement Accumulator A	A - 1 - A	INH	4A				1	1	1	1	1
DECP	Decrement Accumulator B	B - 1 - B	INH	5A				1	1	1	1	1
DES	Decrement Stack Pointer	SP - 1 - SP	INH	3A				1	1	1	1	1



Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Byte	Cyc	Condition Codes														
				OpCode	Operand(s)			S	X	H	N	Z	V	C								
DEX	Decrement Index Register X	$IX \rightarrow IX - 1$	IRH	09		1	3								1							
DEY	Decrement Index Register Y	$IY \rightarrow IY - 1$	IRH	09		1	2								1							
EORA (opt)	Exclusive OR with Memory	$A \oplus M \rightarrow A$	A IMM A DIR A EXT A IND X A IND Y	98 99 9B A5 A6	- H H H H	2 2 3 3 3	2 1 4 4 5									1				0		
EORB (opt)	Exclusive OR B with Memory	$B \oplus M \rightarrow B$	B IMM B DIR B EXT B IND X B IND Y	08 09 0B E5 E6	- H H H H	2 2 3 3 3	2 3 4 4 5									1				0		
FDIV	Fractional Divide 16 by 16	$D:IX \rightarrow IX, r \rightarrow D$	IRH	03		1	21													1	1	
IDIV	Integer Divide 16 by 16	$D:IX \rightarrow IX, r \rightarrow D$	IRH	02		1	41													1	0	1
INC (opt)	Increment Memory Byte	$M + 1 \rightarrow M$	EXT IND, X IND, Y	7C 6C 6E	H H H	3 2 3	6 6 7									1				1	1	
INCA	Increment Accumulator A	$A + 1 \rightarrow A$	A INH	4C		1	2													1	1	1
INCB	Increment Accumulator B	$B + 1 \rightarrow B$	B INH	5C		1	2													1	1	1

SP	Increment Stack Pointer	SP ← SP	WH	31		1	3	
IX	Increment Index Register X	IX ← IX	WH	6B		1	3	1
IY	Increment Index Register Y	IY ← IY	WH	18 0B		2	4	1
JMP (op1)	Jump	See Special Ops	EXT IND,X IND,Y	7E 6E 18 6E	hh ll H H	3 2 3	3 3 4	
JSR (op1)	Jump to Subroutine	See Special Ops	DIR EXT IND,X IND,Y	9D 8D AD 18 AD	dd hh ll H H	2 3 2 3	5 6 6 7	
LDA (op1)	Load Accumulator A	M → A	A DIR EXT IND,X IND,Y	05 06 05 AD 18 AD	H H hh ll H H	2 2 3 3 3	2 3 4 4 7	1 1 0
LDB (op1)	Load Accumulator B	M → B	B DIR EXT IND,X IND,Y	0F 06 16 E6 18 E6	H H hh ll H H	2 2 3 2 3	2 3 4 4 5	1 1 0

Source Format	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Byte Size	e V O	Condition Codes						
				Opcode	Operand(s)			S	X	H	I	N	Z	V
LDS (opt)	Load Double Accumulator D	$M \leftarrow A, M + 1 \leftarrow B$	IMM	CC ij kk	3	3					1	1	0	
			DIR	DC dff	2	4								
			EXT	FC hh ij	3	5								
			IND X	EC ff	2	5								
			IND Y	18 EC ff	3	6								
LDS (opt)	Load Stack Pointer	$M \leftarrow 1 \leftarrow SP$	IMM	BE ij kk	3	3					1	1	0	
			DIR	9E dd	2	4								
			EXT	BE hh ij	3	5								
			IND X	AE ff	2	5								
			IND Y	16 AE ff	3	6								
LDX (opt)	Load Index Register X	$M \leftarrow 1 \leftarrow IX$	IMM	CE ij kk	3	3					1	1	0	
			DIR	DE dd	2	4								
			EXT	FE hh ij	3	5								
			IND X	EE ff	2	5								
			IND Y	CD EE ff	3	6								
LDY (opt)	Load Index Register Y	$M \leftarrow 1 \leftarrow IY$	IMM	18 CE ij kk	4	4					1	1	0	
			DIR	18 DE dd	3	5								
			EXT	18 FE hh ij	4	6								
			IND X	1A EE ff	3	6								
			IND Y	18 EE ff	3	6								

LSL (op1)	Logical Shift Left		EXT IND.X IND.Y	28 hh H 68 H 168 H	3 2 3	6 4 7	1 1 1 1
LSLA			A INH	28	1	2	
LSLB			B INH	68	1	2	
LSLD	Logical Shift Left Double		INH	65	1	3	1 1 1 1
LSR (op1)	Logical Shift Right		EXT IND.X IND.Y	74 hh H 64 H 164 H	3 2 3	6 5 7	0 1 1 1
LSRA			A INH	44	1	2	
LSRB			B INH	54	1	2	
LSPD	Logical Shift Right Double		INH	64	1	3	0 1 1 1
MUL	Multiplies B by B	4xB → D	INH	3D	1	10	1
NEG (op1)	2's Complement Memory Byte	0 M → M	EXT IND.X IND.Y	70 hh H 60 H 160 H	3 2 3	6 6 7	1 1 1 1
NEG2	2's Complement A	0 A → A	A INH	40	1	2	1 1 1 1
NEG3	2's Complement B	0 B → B	B INH	50	1	2	1 1 1 1
NOF	No Operation	No Operation	INH	01	1	2	

Source Formist	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycles	Condition Codes									
				Opcode	Operands			S	X	H	I	N	Z	V	C		
ORAA (opt)	OR Accumulator A (Inclusive-OR)	A ← M ← A	A IMM	BA	h	2	2										0
			A DIR	9A	dd	2	3										
			A EXT	BA	hh s	3	4										
			A IND,X	AA	ff	2	4										
			A IND,Y	18 AA	ff	3	5										
ORAB (opt)	OR Accumulator B (Inclusive-OR)	B ← M ← B	B IMM	CA	h	2	2										0
			B DIR	DA	dd	2	3										
			B EXT	FA	hh s	3	4										
			B IND,X	EA	ff	2	4										
			B IND,Y	18 EA	ff	3	5										
PSHA	Push A onto Stack	A ← SP ← SP + 1	A INH	36		1	3										
PSHB	Push B onto Stack	B ← SP ← SP + 1	B INH	37		1	3										
PSHX	Push X onto Stack - Lo First	(X ← SP ← SP + 2)	INH	3C		1	4										
PSHY	Push Y onto Stack - Lo First	(Y ← SP ← SP + 2)	INH	3D		2	5										
PULA	Pop A from Stack	SP ← SP - 1 ← SP	A INH	3E		1	2										
PULX	Pop X from Stack	SP ← SP - 1 ← SP	B INH	3F		1	2										
PULY	Pop Y from Stack - Hi First	SP ← SP - 2 ← SP	INH	40		1	5										
PULZ	Pop Z from Stack - Hi First	SP ← SP - 2 ← SP	INH	41		2	6										

ROL repr	Rotate Left		EXT IND X IND Y A INH B INH	73 69 18 63 49 59	hh H H H H	2 3 1 1 1	6 6 7 2 2		1	1	1	1	1	1	1	1	1	1	
ROLA																			
ROLB																			
ROR repr	Rotate Right		EXT IND X IND Y A INH B INH	76 66 18 66 46 56	hh H H H H	3 2 3 1 1	6 6 7 2 2		1	1	1	1	1	1	1	1	1		
RORA																			
RORB																			
RTI	Return from Interrupt	Sw Special Ops	INH	3B	?	?	1	1	1	1	1	1	1	1	1	1	1	1	
RTS	Return from Subroutine	Sw Special Ops	INH	39			1	1	1	1	1	1	1	1	1	1	1	1	
SBA	Subtract B from A	A - B - A	INH	10			1	2											
SBCA repr	Subtract with Carry from A	A - M - C - A	A INH A DIR A EXT A IND X A IND Y	82 92 82 A2 18 A2	hh hd hh H H H	2 2 3 2 3	2 1 4 4 5		1	1	1	1	1	1	1	1	1	1	
SBCB repr	Subtract with Carry from B	B - M - C - B	B INH B DIR B EXT B IND X B IND Y	C2 D2 F2 E2 18 E2	hh dd hh H H H	2 2 3 2 3	2 3 4 4 5		1	1	1	1	1	1	1	1	1	1	

Source Form(s)	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		B	C	U	V	Condition Codes												
				Opcode	Operand(s)					S	X	H	I	N	Z	V	C					
SEC	Set Carry	$1 \rightarrow C$	INH	0D		1	2														1	
SEI	Set Interrupt Mask	$1 \rightarrow I$	INH	0F		1	2															1
SEV	Set Overflow Flag	$1 \rightarrow V$	INH	0B			2															1
STAB (opri)	Store Accumulator A	$A \rightarrow M$	A DIR A EXT A IND,X A IND,Y	07 B7 A7 1B A7	hh hh ll H H	2 3 2 3	3 4 4 5															1 1 1 1
STAB (opri)	Store Accumulator B	$B \rightarrow M$	B DIR B EXT B IND,X B IND,Y	D7 F7 E7 1B E7	dh hh ll H H	2 3 2 3	3 4 4 5															1 1 1 1
STD (opri)	Store Accumulator D	$A \rightarrow M, B \rightarrow M + 1$	DIR EXT IND,X IND,Y	DD FD ED 1B ED	dh hh ll H H	2 3 2 3	4 5 5 6															1 1 1 1
STOP	Stop Internal Clocks		INH	CF		1	2															
STS (opri)	Store Stack Pointer	$SP \rightarrow M, M + 1$	DIR EXT IND,X IND,Y	9F BF AF 1B AF	dh hh ll H H	2 3 2 3	4 5 5 6															1 1 1 1

STX (opt)	Store Index Register X	$Y \rightarrow MM + 1$	DIR EXT IND, X IND, Y	DF FF EF CF EF	dd hh H H	2 1 3 3	4 5 6 6		1	1	0
STY (opt)	Store Index Register Y	$Y \rightarrow MM + 1$	DIR EXT IND, X IND, Y	18 DF 18 FF 1A EF 18 EF	dd hh H H	3 4 3 3	5 6 6 6		1	1	0
SUBA (opt)	Subtract Memory from A	$A - M \rightarrow A$	A IMM A DIR A EXT A IND, X A IND, Y	80 90 B0 A0 18 A0	H dd hh H H H	2 2 3 2 3	2 3 4 4 5		1	1	1
SUBB (opt)	Subtract Memory from B	$B - M \rightarrow B$	B IMM B DIR B EXT B IND, X B IND, Y	C0 D0 F0 E0 18 E0	H dd hh H H H	2 2 3 2 3	2 3 4 4 5		1	1	1
SUBD (opt)	Subtract Memory from D	$D - MM + 1 \rightarrow D$	IMM DIR EXT IND, X IND, Y	83 93 B3 A3 18 A3	H dd hh H H H	3 2 3 2 3	4 5 6 6 7		1	1	1
SVC	Software Interrupt	See Special Ops	IMM	3F	H	1	11		1		

Source Formist	Operation	Boolean Expression	Addressing Mode for Operand	Machine Coding (Hexadecimal)		Bytes	Cycles	Condition Codes																
				Opcode	Operand(s)			S	X	H	N	Z	V	C										
TAB	Transfer A to B	A ← B	INH	06		1	2																	
TAP	Transfer A to CC Register	A ← CCR	INH	05		1	2	1	1	1	1	1	1	1	1	1	1	1	1	1				
TBA	Transfer B to A	B ← A	INH	17		1	2																	
TEST	TEST (Only in Test Modes)	Address/Bus Counts	INH	00		1	*																	
TFA	Transfer CC Register to A	CCR → A	INH	07		1	2																	
TST (opt)	Test for Zero or Minus	M = 0	EXT	7D	hh	3	6													1	1	0	0	
			IND X	6D	h	2	6																	
			IND Y	18 6D	h	3	7																	
TSTA		A = 0	A INH	4D		1	2																	
TSTB		B = 0	B INH	5D		1	2																	
TSX	Transfer Stack Pointer to X	SP ← 1 ← IX	INH	30		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
TSY	Transfer Stack Pointer to Y	SP ← 1 ← IY	INH	18 30		2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
TXS	Transfer X to Stack Pointer	IX ← 1 ← SP	INH	35		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
TVS	Transfer Y to Stack Pointer	IY ← 1 ← SP	INH	18 35		2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E		1	**																	
XGDY	Exchange D with X	IX ← D, D ← IX	INH	8F		1	3																	
YGDY	Exchange D with Y	IY ← D, D ← IY	INH	18 8F		2	4																	

Hardware Control Registers and Bits

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Name	Description
\$1000	---	---	---	---	---	---	---	---	FOCIA	IO Port A
\$1001	---	---	---	---	---	---	---	---		Reserved
\$1002	STAF	STAI	CWOM	HNSD	QIN	PLS	EGA	INVB	FOCC	Parallel IO Control Register
\$1003	---	---	---	---	---	---	---	---	FOCIIC	IO Port C
\$1004	---	---	---	---	---	---	---	---	FOCIDB	Output Port B
\$1005	---	---	---	---	---	---	---	---	FOCIDCL	Alternate Latched Port C
\$1006	---	---	---	---	---	---	---	---		Reserved
\$1007	---	---	---	---	---	---	---	---	DFIC	Data Buffer for Port C
\$1008	---	---	---	---	---	---	---	---	FOCIDD	IO Port D
\$1009	---	---	---	---	---	---	---	---	DFID	Data Buffer for Port D
\$100A	---	---	---	---	---	---	---	---	FOCIE	IO Port E
\$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	OC1C	Compare Control Register
\$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1E	OC1 Action Enable Register
\$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D	OC1 Action Data Register

Hardware Control Registers and Bits

Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Field	Description
\$100E	---	---	---	---	---	---	---	---	TCNT	Timer Counter Register
\$100F	---	---	---	---	---	---	---			
\$1010	---	---	---	---	---	---	---	---	IC1I	Input Capture 1 Register
\$1011	---	---	---	---	---	---	---			
\$1012	---	---	---	---	---	---	---	---	IC2I	Input Capture 2 Register
\$1013	---	---	---	---	---	---	---			
\$1014	---	---	---	---	---	---	---	---	IC3I	Input Capture 3 Register
\$1015	---	---	---	---	---	---	---			
\$1016	---	---	---	---	---	---	---	---	IC4I	Output Compare 1 Register
\$1017	---	---	---	---	---	---	---			
\$1018	---	---	---	---	---	---	---	---	IC5I	Output Compare 2 Register
\$1019	---	---	---	---	---	---	---			
\$101A	---	---	---	---	---	---	---	---	IC6I	Output Compare 3 Register
\$101B	---	---	---	---	---	---	---			
\$101C	---	---	---	---	---	---	---	---	IC7I	Output Compare 4 Register
\$101D	---	---	---	---	---	---	---			
\$101E	---	---	---	---	---	---	---	---	IC8I	Output Compare 5 Register
\$101F	---	---	---	---	---	---	---			
\$1020	OM0	OM1	OM2	OM3	OM4	OM5	OM6	OM7	IC1I1	Timer Counter Flag 1
\$1021	0	0	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	IC1I2	Timer Counter Flag 2
\$1022	OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I	IC1IF1	Timer Input Match Flag 1
\$1023	OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F	IC1IF1	Timer Input Match Flag 1
\$1024	TOI	RTI	PAOVI	PAIF	0	0	PAI	PRD	IC1IF2	Timer Input Match Flag 2
\$1025	TOF	RTF	PAOVF	PAIF	0	0	0	0	IC1IF2	Timer Input Match Flag 2
\$1026	DDIA7	PAEII	PANOD	PDGE	0	0	PII1I	PII1D	PAI1I	Phase Advance Control Flag
\$1027	---	---	---	---	---	---	---	---	PII2I	Phase Advance Control Flag

Hardware Control Registers and Bits

Addr	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Mask	Description
\$1026	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPRO	SPCR	SPI Control Register
\$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR	SPI Status Register
\$102A	—	—	—	—	—	—	—	—	SPCR	SPI Data Register
\$102B	TCLR	0	SCP1	SCP0	HCKB	SCI1	SCI1	SCI0	I2M0	SCI Control Register
\$102C	RB	TB	0	M	WAKE	0	0	0	SCCR1	SCI Control Register 1
\$102D	TIE	TCIE	RIE	ILIE	TE	RE	HWU	SIK	SCCR2	SCI Control Register 2
\$102E	TDRE	TC	RDRF	IDLE	CR	NF	FE	0	SCSR	SCI Status Register
\$102F	—	—	—	—	—	—	—	—	SC1R	SCI Data (I2R/I2W/TDR)
\$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL	AD Control Register
\$1031	—	—	—	—	—	—	—	—	AD1R	AD1 Data Register 1
\$1032	—	—	—	—	—	—	—	—	AD2R	AD1 Data Register 2
\$1033	—	—	—	—	—	—	—	—	AD3R	AD1 Data Register 3
\$1034	—	—	—	—	—	—	—	—	AD4R	AD1 Data Register 4
\$1035 to \$103A										Reserved
\$1039	ADPU	CSEL	IRDF	DLY	CH2	0	CH1	CH0	CCDR2	System Configuration Options
\$103A	—	—	—	—	—	—	—	—	—	CCDR1: Actual vs. CC1: Target Capacity
\$103B	ODD	EVEN	0	BYTE	ROW	ERASE	ECLAT	EEPROM	EEPROM	EEPROM Programming Control
\$103C	PRO0	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	IF13D	Input/Output Data Register
\$103D	HAM3	HAM2	HAM1	RAM0	REG3	REG2	REG1	REG0	IF1	Input/Output Register
\$103E	TIOP	0	OCCR	CBYP	DIRT	FCM	EGOP	TCOR	TEST1	Test Value Control Register
\$103F	0	0	0	0	NOSSC	NOCON	HOMON	DEON	CCDR3	Control Register 3: I2C/UART

BIBLIOGRAFIA

MICROCOMPUTER ENGINEERING

Gene H. Miller
Ed. Prentice-Hall, Inc.
A Simon & Schuster Company
Englewood Cliffs, New Jersey 1993

DATA ADQUISITION AND PROCESS CONTROL WITH THE M68HC11

Frederick F. Driscoll
Ed. Mc. Millan
1992

CRITERIO DE PLANTAS TERMoeLECTRICAS

Martiano Aguilar R.
Ed. Limusa
1982

DESIGN WITH MICROCONTROLLERS

Jhon B. Peatman
Ed. Prentice-Hall
1992

EVALUATION BOARD USER'S MANUAL

Motorola Inc.
MC68HC11EVBUMVAD1
1990

REFERENCE MANUAL

Motorola Inc.
MC68HC11
1990

PRINCIPIOS DIGITALES

Donald L. Tuckman
Ed. Mc Graw Hill
1982

CIRCUITOS LINEALES Y AMPLIFICADORES OPERACIONALES

Frederick F. Driscoll
Ed. Mc. Graw Hill
1990

DIGITAL ISOLATION ASSY

Jhon Fluke
Mountlake Terrace, Wa.
MFG. CO. INC.
1981

SEQUENCE OF EVENT OPTION

Jhon Fluke
Mountlake Terrace, Wa.
MFG. CO. INC.
1981

CHANNEL INPUT CONECTOR

Jhon Fluke
Mountlake Terrace, Wa.
MFG. CO. INC.
1981