UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

RELOJ CHECADOR MULTIENTRADA

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA

CON ESPECIALIDAD EN ELECTRONICA

P R E S E N T A :

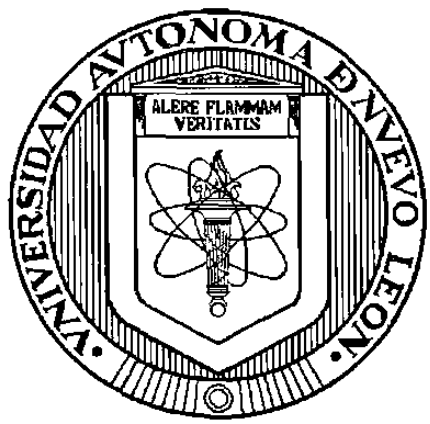
GUADALUPE ACOSTA VILLARREAL

CD. UNIVERSITARIA ENERO DE 1995

TM
Z5853
.M2
FIME
1995
A2


1020070693

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA



RELOJ CHECADOR MULTIENTRADA

TESIS
QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA INGENIERIA ELECTRICA
CON ESPECIALIDAD EN ELECTRÓNICA

PRESENTA
GUADALUPE ACOSTA VILLARREAL

Cd. Universitaria, San Nicolás de los Garza, N. L. Enero de 1995.

TM
258.3
.M2
FIME
1995
A2

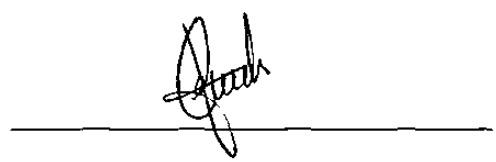
FONDO TESIS

16679

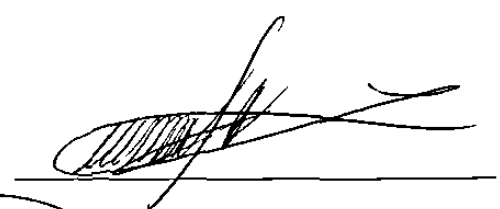
RELOJ CHECADOR MULTIENTRADA

Los Miembros del Comité aprueban la Tesis
de Maestría de Guadalupe Acosta Villarreal

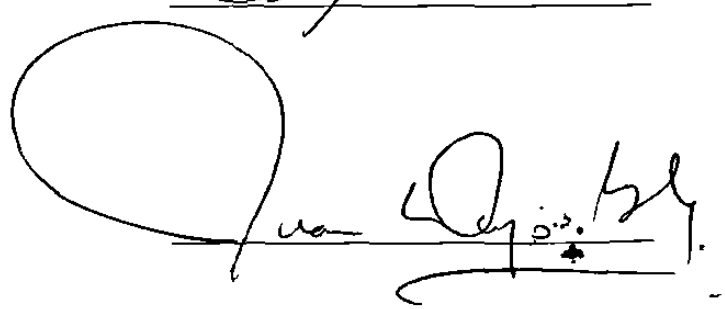
M. C. Luis Manuel Camacho Velázquez
Asesor



M. C. Luis M. Martínez Villarreal



M. C. Juan D. Garza González



Derechos Reservados © 1995
Ing. Guadalupe Acosta Villarreal

DEDICATORIA

Para mis padres: Silvino y Luisa y mis hermanos: Arnulfo, Esther, Héctor, Clara, Bertha, Faustino, Humberto, Leticia, Rolando y Saúl Silvestre.

Para mi Esposa Simona Santos de L.

y para mis Hijos

Orlando Guadalupe
Eduardo Haziel
e Iraida Edith

Por sus atenciones, sus esperas, por sus palabras, por su inagotable paciencia

AGRADECIMIENTOS.

Agradezco el apoyo recibido para la parte operativa al Director del Instituto Tecnológico de Cd. Victoria, Tam. Ing. Tomás Garza Wong y al Jefe de Recursos Humanos, Ing. Salomón Rodríguez; por el invaluable apoyo a la parte técnica a la Dra. Ludivina Barrientos Lozano; A los compañeros del Doctorado en Ingeniería Eléctrica de la Facultad de Ingeniería Mecánica y Eléctrica de la UANL, M. C. César Elizondo González y M. C. Víctor Vega Domínguez por toda su ayuda en el área de Hardware, especialmente al M.C. Juan José Darío Delgado Romero por toda la asesoría en los momentos difíciles del desarrollo de este trabajo; a mis hermanos Faustino, Humberto y Rolando por estar siempre atentos a las necesidades de software y hardware. Un agradecimiento especial al M. C. Luis Manuel Camacho Velázquez por todas las sugerencias para mejorar la presentación.

Guadalupe Acosta Villarreal

Cd. Universitaria, San Nicolás de los Garza N. L., Enero de 1995.

INDICE

INDICE DE FIGURAS ***PAGINA***

INDICE DE FIGURAS *PAGINA*

INDICE DE FIGURAS *PAGINA*

INDICE DE TABLAS *PAGINA*

INDICE DE DIAGRAMAS DE FLUJO *PAGINA*

INDICE DE RUTINAS *PAGINA*

INDICE DE RUTINAS *PAGINA*

INDICE DE RUTINAS ***PAGINA***

CAPITULO I
GENERALIDADES

El tiempo, resultado de la actividad de nuestro universo, se ha tratado de medir de diferentes maneras; con relojes de sol, arena, agua, mecánicos, eléctricos y electrónicos entre los más sobresalientes; en la actualidad cumplen importantes funciones, una de ellas es la de medir la duración de la jornada de trabajo.

Dada la importancia de la fuerza de trabajo; en las empresas se ha hecho necesario medir la duración de cada turno, horas trabajadas y horas extras.

Por lo cual se ha diseñado un dispositivo con características especiales, las que en páginas posteriores se enumerará.

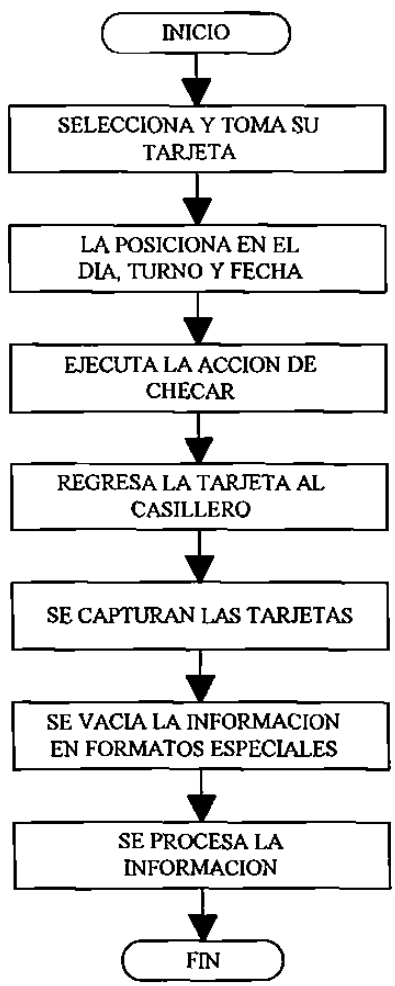


Diagrama de FlujoNo. 1.1: Acción de Checar Entrada o Salida y Manipulación de Datos.

Existe un reloj checador aún muy utilizado, el reloj electromecánico; el cual es económico y deja constancia de la acción del empleado "el checar" su entrada o su salida de su turno o sus horas extras; el sistema se divide en el dispositivo (reloj checador), Figura No. 1.1; las tarjetas donde se checa, los anaqueles donde se depositan las tarjetas, Figura No. 1.2 y las concentrados donde se vacían los datos capturados en las tarjetas y diversas formas de manipular dichos datos; el Diagrama de Flujo No. 1.1 muestra la secuencia que sigue esta acción.

Esta forma de control de entrada salida tiene la desventaja de ser problemática por el número de repeticiones que presenta en la captura de la información por lo que es propensa a errores; además el trabajador por descuido o rapidez al checar empalma las checadas, y otros errores de tipo involuntario de acuerdo al tipo de reloj que se use.

Actualmente existen en el mercado algunos modelos de relojes checadores electrónicos bastante económicos con el solo requisito de tener una computadora, cosa muy común actualmente; el modelo que presento contiene algunas características que no poseen los existentes en el mercado.

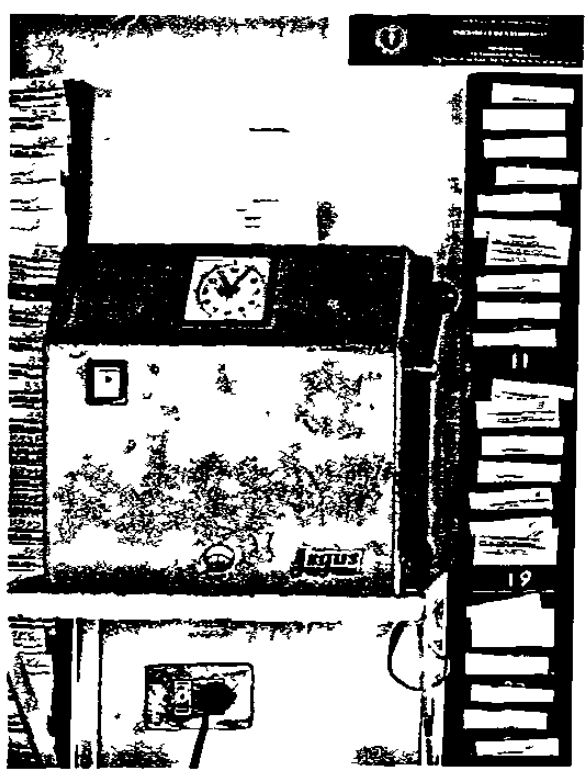


Figura No. 1.1: Reloj Checador Electromecánico.

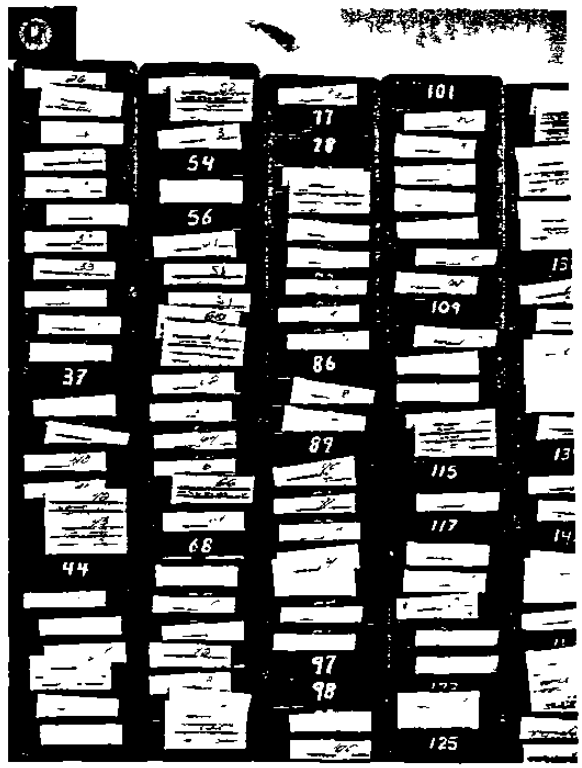


Figura No. 1.2:Tablero y Tarjetas que se Utilizan para Checar.

DESCRIPCION

El reloj checador electrónico se basa en el microcontrolador de motorola MC68HC11, el cual está configurado en modo expandido, los detalles se mencionarán en el capítulo de Hardware. Este dispositivo tiene las siguientes caracteristicas.

- Capacidad para 500 trabajadores (*).
- Cuenta con respaldo de baterías (*).
- Puntos de alarma periodicos o fijos
- Salida para conectar tableros de digitos de 7 segmentos
- Muestra de tiempo de 00 00 00 a 23 59 59 (*).
- Muestra de la temperatura ambiente en Grados Celsius.
- Entrada para código de barras
- Salida a impresora de tipo centronics del dato checado (*)

- Entrada de datos vía teclado (*).
- Reporte de Memoria disponible en número de caracteres (*).
- Reporte a impresora de la información capturada (*).
- Entrada de datos por computadora vía puerto serie (*).
- No es necesario tener la computadora encendida (*).
- Espacio en memoria para entrada de datos vía puerto serie del nombre del trabajador en 22 caracteres (ASCII) y 2 para su clave (BCD) (*).
- Módulo para trasladar datos a computadoras que no están al alcance del dispositivo (*).

La figura No. 3 muestra el diagrama a bloques de la estructura general de este dispositivo.

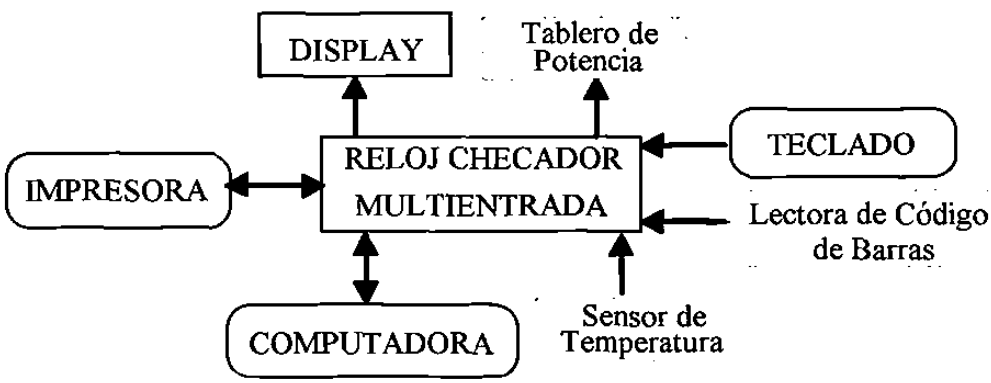


Figura No. 1.3: Diagrama a Bloques que Muestra el Reloj Checador .

El mapeo del microprocesador será como lo indica la Tabla No. 3, los primeros 32 kb serán de RAM y el resto de puertos; la RAM se divide en dos partes 16 kb para ram de nombres y 16 kb para ram de datos.

La ram de nombres contendrá la información del trabajador distribuída de la siguiente manera:

No. de Empleado	2 bytes
Nombre	22 bytes

Usando las normas internacionales, es decir manejando un código lo suficientemente común y compatible con los elementos que interactúan (reloj checador, impresora, computadora) como el ASCII (*American Standard Code for Information*

Interchange); podremos colocar en 16 Kb de RAM la clave y el nombre de 24 bytes de 682 empleados, de los cuales sólo pensaremos en 500 empleados quedando 4,384 bytes que se reservarán para otros usos.

El funcionamiento principal del dispositivo será como lo muestra el Diagrama de Flujo No. 1.2.

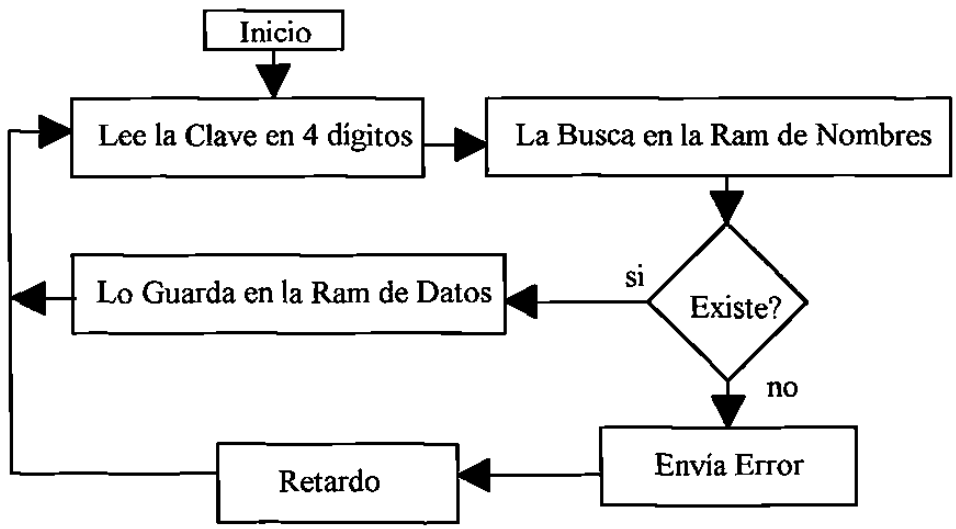


Diagrama de Flujo No. 1.2: del Funcionamiento del Reloj Checador

La información se acumulará en una RAM estática de 16 Kb donde cada 00:00:00 Horas colocará en la Ram la fecha correspondiente en el siguiente formato /dd mm aa (/ identificador de fecha, día, mes y año) en cada checada colocará hh mm (horas y minutos), a continuación se guardará la clave del empleado (cc cc) donde lo siguiente nos mostrará como quedará en la RAM; donde "/" es igual a un byte, "dd" un byte, "mm" un byte, "aa" un byte de igual forma la clave y la hora.

/ddmmaa*hhmm*cccc...hhmmcccc...../ddmmaahhmmccccchhmmcccc....

Obtenemos que por cada checada tenemos (hhmmcccc) 4 bytes (hh,mm,cc,cc) y agregamos en cada día 4 bytes de fecha, por lo que suponiendo que un empleado checara dos veces entrada y dos veces salida en un mismo día, se tendrían 16 bytes de checada por lo que 500 trabajadores checando 4 veces diarias, ocuparían aproximadamente 8 Kb + 4 bytes de fecha, nos quedarán un poco más de 8 Kb para checadas de horas extras y otros

eventos no considerados; como recuperar la información después de iniciar el primer turno del siguiente día e incluso de dos días consecutivos.

De las claves y nombres llegarán al dispositivo vía puerto serie de la computadora y generados mediante una base computacional, de total responsabilidad del usuario, [solamente se desarrollará las características mostradas con asterisco (*)] de igual forma los datos capturados por el dispositivo serán transferidos similarmente a como fueron recibidos, es decir por el puerto serie y presentados en un archivo en la computadora para poder ser manipulados y procesados por el sistema que hago referencia; pero teniéndolo, en forma interactiva, como lo indica la Figura No. 1.4.

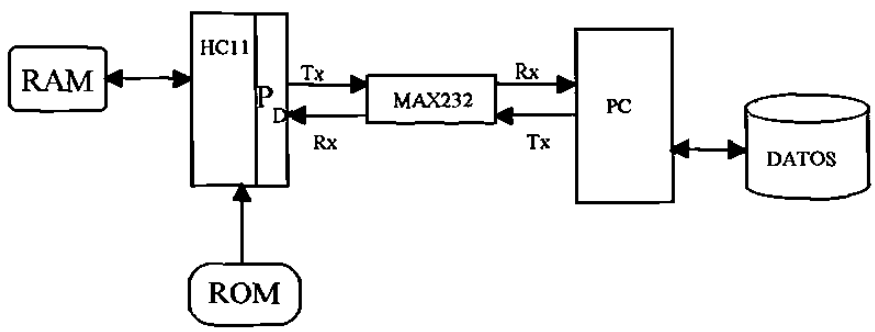


Figura No. 1.4: Flujo de la Información de la Microcomputadora al Dispositivo y Visceversa.

Los clave del empleado será digitada por éste en cada ocasión que necesite checar su entrada o salida, es obvio que al conectarse la lectora de código de barras este procedimiento se sustituirá; el teclado se conectará al puerto A del microcontrolador MC68HC11; al digitar el empleado su clave ésta será verificada en la base de datos almacenada en la ram de nombres y de localizarse, guardará la hora en el preciso momento de que digitó la primer tecla, además se imprimira opcionalmente estos datos con el formato siguiente:

---

hh:mm dd/mm/aa

CCCC NOMBRE DEL TRABAJADOR

Donde hh:mm es el formato de horas y minutos, dd/mm/aa es día/mes/año, la clave es representada por CCCC y el nombre del trabajador en 22 caracteres como se encuentra en la ram de nombres.

El sistema será totalmente interactivo con tres subsistemas denominados **operación normal , de consulta y de comunicación**; estos subsistemas serán controlados mediante una llave tipo de caja registradora denominada de supervisión, para operar los tres modos; el primer subsistema de operación normal se refiere al modo en que el trabajador tiene acceso al sistema mediante un menú que se mostrará en un display de cristal líquido de 2 líneas y 40 caracteres por línea, mostrado en la Figura No. 1.5.

Figura No. 1.5: Display de 80 Caracteres en Dos Líneas

El trabajador al teclear su clave aparecerá su nombre en 20 caracteres o el mensaje **esta clave no existe**, para dar de alta en la RAM de datos su checada necesitará confirmarlo oprimiendo la tecla de *enter* o en su defecto si hay un error, la de *escape* inmediatamente al teclear *enter* se guardará la hora de checada y la clave como se indicó anteriormente y se imprimirá en la impresora, si es que ésta existe, al implementar la lectora de código de barras se evitará la acción de teclear.

Para el subsistema de consulta contará con el menú de la Figura No. 1.6 en el cual se muestra la fecha y la hora en los extremos de la primer línea además las opciones (1) Poner a Tiempo y (2) Consultar.

Figura No. 1.6: Menú principal del Subsistema de Consulta.

Seleccionando 1 ó 2 el cursor (<) brincará de una línea a otra y con un *enter* seleccionará la opción deseada; de ser Poner a Tiempo, mostrará el menú de la Figura No. 1.7 , estas nuevas opciones son para modificar la fecha o la hora.

Figura No. 1.7: Menú para Modificar la Hora o la Fecha.

Para salir de cualquier opción basta con presionar la tecla de *escape* y se irá al menú principal.

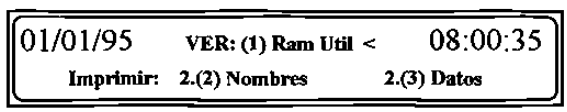


Figura No. 1.8: Menú de las Opciones de Consultar.

Si se selecciona el **(2) Consultar**, del menú principal se mostrará el menú de la Figura No. 1.8 el cual contiene tres opciones: **VER: (1) Ram Util**, la cual mostrará la cantidad de bytes disponibles enunciandolos como caracteres, para que el usuario determine el espacio de tiempo que puede mantener sin respaldar la información; **Imprimir: 2.(2) Nombres**, esta opción imprime el contenido de la base de datos que tiene almacenada el dispositivo; e Imprimir: **2.(3) Datos**, Imprime los claves y la hora en que el usuario ha checado, como lo indica la Figura No. 1.9.

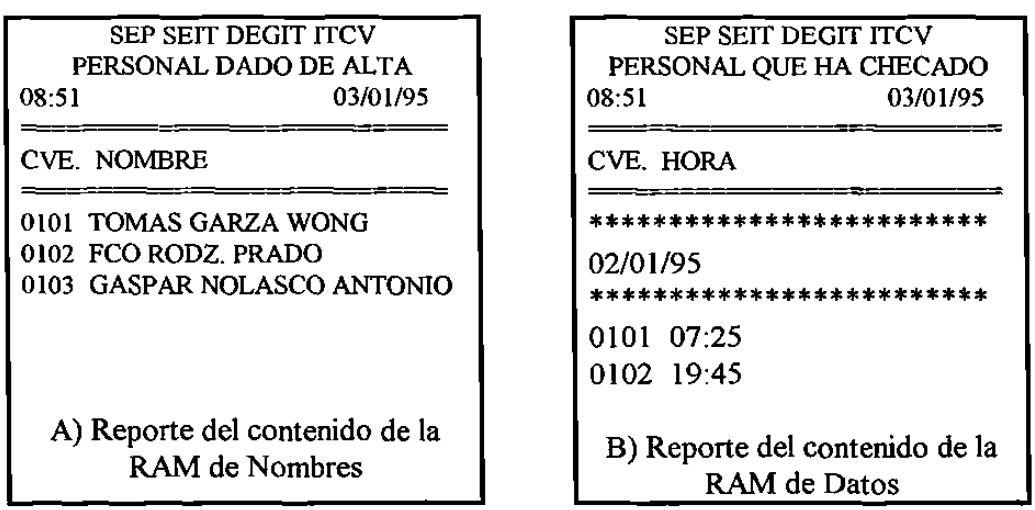


Figura No. 1.9: Reportes Obtenidos de el Menú Consultar.

Los datos se manipularán en la computadora inicialmente, con una base de datos ya indexada por clave, de menor a mayor de tal forma que generarán un archivo con clave de cuatro dígitos y presentados en BCD, ocupando ésta dos bytes y el nombre en 22 caracteres, completando 16 Kb con el hexadecimal $FF; este formato es importante, ya que el sistema en el dispositivo buscará la clave en función de este índice, ahorrandose tiempo en la búsqueda, este archivo será enviado a través del puerto serie y almacenado en la Ram para nombres, de manera similar pero en sentido contrario el dispositivo enviará vía puerto serie a la computadora, los datos almacenados en la ram de datos conteniendo /ddmmaaccccchhmm... donde "/" es el identificador de fecha en un byte, dd un byte en BCD de día , igual para mes y año; dos bytes de clave y dos bytes entre horas y minutos también en BCD. Al ir transfiriendo la información complementará con el hexadecimal $FF el byte enviado; el manipuleo de la información será responsabilidad del usuario, así como el nombre que se le dará a este archivo

La medición del tiempo se implementó por medio de una interrupción generando un pulso de reloj cada centésima de segundo; guardadas en registros especiales de segundos, minutos, horas, días, meses y años; por medio de una tabla se consulta el número de días que contiene el mes correspondiente; y mediante una operación sencilla se determina si el año es un múltiplo de cuatro para saber si es bisciesto; las horas son de 00:00:00 a 23:59:59 es decir en formato de 24 horas; esto implica que la selección del cristal que complementará el diseño debe ser seleccionado de tal forma que no complique la implementación de la interrupción, una forma sencilla es encontrar un potencia de 2 que nos permita hacer esto de tal forma que 2^{22} nos da una frecuencia que nos da divisiones exactas, la cual es de 4,194,304 Hz la cual con un mínimo de ajuste podremos lograr que el tiempo señalado en horas , minutos y segundos sea lo suficientemente exacto.

CAPITULO 2
HARDWARE

El sistema de reloj checador está basado en el microcontrolador de motorola MC68HC11 en modo expandido como lo muestra el diagrama a cuadros de la Figura No. 2.1, cuenta con el circuito 74LS373 para demultiplexar los datos de las direcciones , una memoria de 32 kb dividida en dos bloques, dos PIAs MC6821 para ampliación de puertos, para impresora y otras E/S un bloque para el división del mapa de memoria e interfases para el display de cristal líquido y puerto serie.

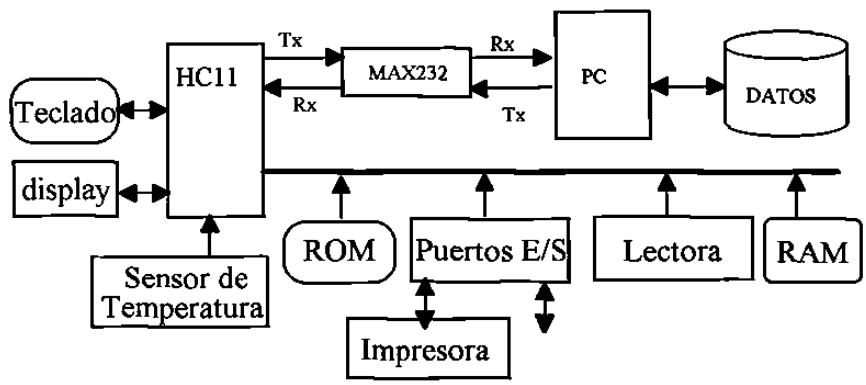


Figura No. 2.1: Diagrama a Bloques del Sistema Control de Personal.

DE LA CONEXION DEL MICROCONTROLADOR.

Primeramente hablaremos de la base fundamental del sistema denominado Reloj de control de Personal, que es el microcontrolador MC68HC11, este es un integrado que contiene además de un microprocesador de 8 bits, puertos de entrada salida, Ram y Rom internos, puerto serie y entradas analógicas:

El microcontrolador MC68HC11 que contiene un microprocesador de 8 bits que se implementó con un cristal de 4.194304 Mhz que es la parte fundamental del reloj cuyo bus trabaja a un cuarto de esta frecuencia. La representación gráfica se muestra en la Figura No. 2.2 cuya conexión se hizo tal como lo recomienda el fabricante con una resistencia R=10 MΩ. y los capacitores C=27 Pf.

El reset para el MC68HC11 es la implementación más simple la cual contiene una resistencia de 1 KΩ conectada a Vcc y un switch que conecta a tierra, como se indica en la Figura No. 2.3.

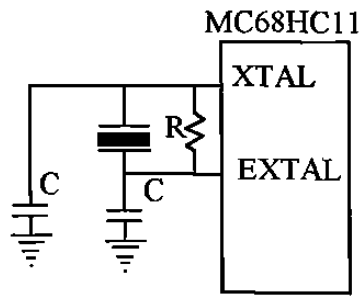


Figura No. 2.2: Conexión del Oscilador de 4.194303 al MC68HC11.

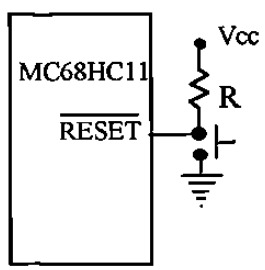


Figura No. 2.3: Circuito de Reset del MC68CH11.

EL TECLADO

Los dispositivos que se conectarán directamente al microprocesador con un mínimo de interfase serán el teclado, la llave de modo, el display, la Ram, el Puerto Serie y el sensor de temperatura. En la Figura No. 2.4 se muestra la conección del teclado en el puerto A del microcontrolador

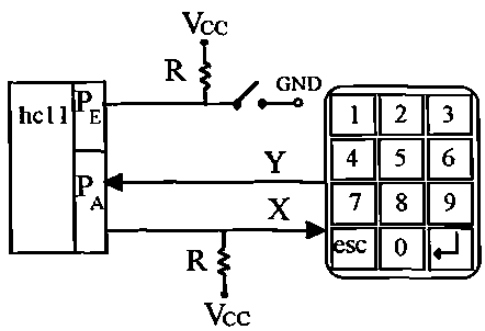


Figura No. 2.4: Conexión del Teclado con el Microprocesador.

Las "Y" del teclado se conectarán a los bits $A_{0..2}$ las cuales son entradas naturales del microcontrolador y la entrada "X" se conectarán a los bits $A_{3..6}$ y a VCC por medio de una resistencia de 4.7 KΩ las cuales son salidas naturales del puerto A del microcontrolador; esto con el propósito de enviar ceros por las salidas y monitorearlos por las entradas al cerrar el circuito por medio de tecla oprimida.

EL PUERTO SERIE

Se seleccionó el puerto Serie Asíncrono por ser compatible con el puerto serie de la computadora y para usar la misma fuente de voltaje del microcontrolador para implementar los voltajes que necesita el puerto (+12 y -12 Vcd) por medio de un Maxi MAX232, circuito integrado que permite entradas de 5 Vcd y salidas de ±12 Vcd, como se muestra en la Figura No. 2.5; además cumple con las especificaciones de la EIA y la recomendación V.28 de CCITT las cuales indican que pueden ser utilizados para las diferentes velocidades estándares de 300, 600, 1200, etc baudios; el máximo recomendado es de 19,200 baudios.

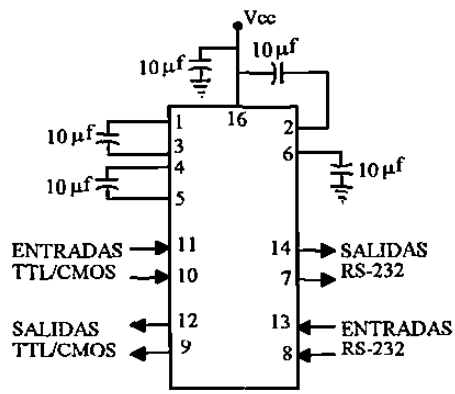


Figura No. 2.5: MAX232 que Permite Transformar el Nivel TTL a RS232.

DEMULTIPLEXANDO DATOS DE DIRECCIONES

Como los datos están multiplexados con las direcciones bajas, éstas se demultiplexan por medio del circuito 74LS373 y la señal del microcontrolador AS que proviene del pin 26 en el DIP de 48 pines, como lo muestra la Figura No. 2.6. conformando asi el bus de datos y el bus de direcciones.

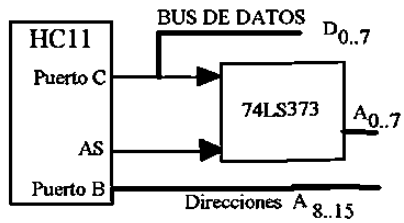


Figura No. 2.6: Conformación del Bus de Datos y el Bus de Direcciones.

EL MAPEO DE MEMORIA

El mapeo de memoria se logra mediante el decodificador de 3 a 8 número 74LS138 donde intervienen las direcciones de la $A_{15..12}$ y la señal E invertida que da la sincronía formando un mapeo de bloques de 4 Kb en la parte alta; como lo indica la Figura No. 2.7, se aprovecha que la primera mitad del mapa de memoria la ocupa la Ram del sistema y ésta la determina A_{15} en bajo, por lo tanto cuando A_{15} es alto se encuentra en la segunda mitad del mapa y, ésto se aprovecha para habilitar G1 del decoder y como G2A+G2B necesitan estar en bajo invertimos la señal E para tener la sincronía necesaria en la disposición de los 8 bloques en que se divide esta parte del mapa de memoria.

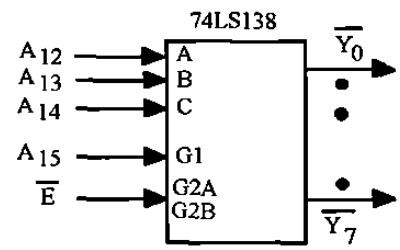


Figura No. 2.7: Decodificación de los Bloques de Memoria.

EL DISPLAY

El display que contiene diferentes características de las cuales se usará la interfase de 8 bits sin el uso de la bandera "*busy Flag*", quedando como lo muestra la Figura No 2.8. Algunas de las características de este diplay se mencionan a continuación:

- Es display es HD44780 capaz de interfasear con 4 bit u 8 bits con un microprocesador.
- Display RAM de datos de 80 x 8 bits.
- ROM generador de caracteres de 5x7 y 5x10 dots.
- Caracteres de la RAM pueden ser leídos por el microprocesador.
- Reset interno automático al activar la fuente de alimentación.

El mapeo de memoria está indicado en la Tabla No. 2.4, de aquí encontramos que la dirección reservada para el display es $D000 a la $DFFF aunque sólo dos direcciones se utilizan, el resto es espejo de la misma. El display de cristal líquido de 2 líneas por 40 columnas está conectado directamente al bus por medio de $\overline{Y}_5$, E, A_0, R / $\overline{w}$ y los datos.

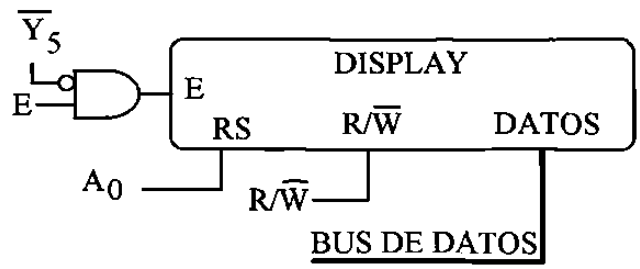


Figura No. 2.8: Conexión del Display en la Dirección $D000 a $DFFF.

Como el display necesita en RS=0 para escribir el código de instrucción y RS=1 para escribir datos, cualquier par de direcciones que terminen en 0 ó 1 dentro de $D000 a $DFFF servirán para direccionar este display, he seleccionado las dos primeras direcciones para manipularlo, como se indica en el capítulo de Software.

LA RAM DEL SISTEMA

La RAM del sistema es un Circuito de Ram Estática de 32 Kb en un solo integrado, dándonos modularidad y facilitando su direccionamiento como se indica en la Figura No. 2.9, esta RAM tiene velocidad de acceso entre 100 y 120 nseg lo cual permite un acceso lo suficientemente veloz para no entorpecer otras funciones del sistema; el circuito en mención es el D43256A, circuito integrado de 28 pines.

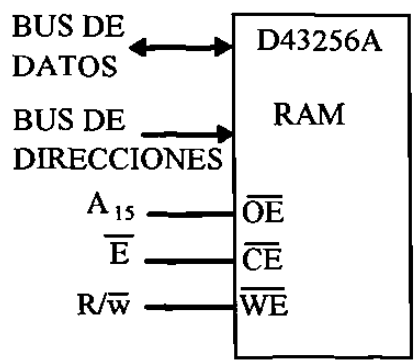


Figura No. 2.9: Direccionamiento de la Ram del Sistema.

Esta RAM es de las denominadas RAM estática (SRAM) por lo que no necesita la acción de refresco haciendo más simple el diseño, y también por no necesitar mayores volúmenes de memoria; su direccionamiento es muy sencillo; por encontrarse totalmente localizada en la primera mitad del mapa es decir cuando A_{15} es baja, aprovecho esta situación para conectarla a $\overline{OE}$ y negando la señal E del microcontrolador para amarrarla a $\overline{CE}$ y finalmente amarrar $R/\overline{W}$ del microcontrolador a write enable ($\overline{WE}$) con esto garantizo lectura y escritura de las direcciones $0000 a $7FFF.

LA IMPRESORA

La impresora dispositivo cuya interfase de comunicación tipo centronics consta de 16 líneas de las cuales 8 son datos y 8 son de control, de estos últimos entran a la impresora 3 bits y salen 5; la interfase para conectar ésta es por medio de un Adaptador de Interfase Periférico (PIA) MC6821, de motorola que consta de dos puertos A y B; el Puerto A lo usé de control y el B de datos de tal forma que A es de entrada salida y B sólo de salida; las señales implementadas fueron similares a las que manejan en sus interrupciones el BIOS las PCs y se describen en la Figura No. 2.10.

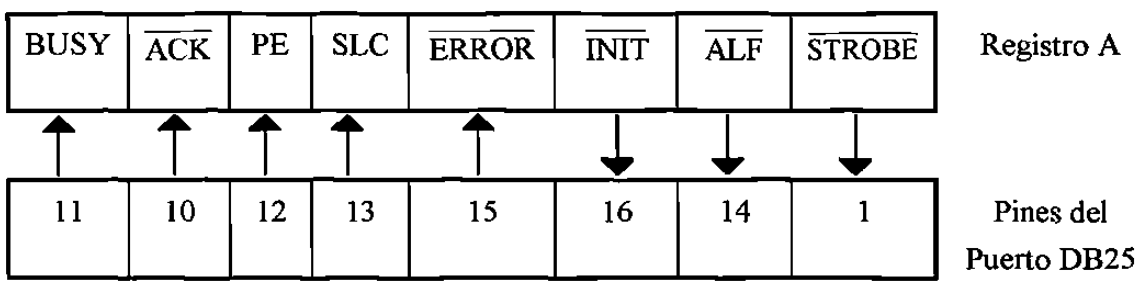


Figura No. 2.10: Conexión Entre el Puerto A del PIA MC6821 y los Pines de Control del Puerto de la Impresora.

Las señales que marca el estandar para cualquier impresora con conexión tipo Centronics se describe en la Tabla No. 2.1, las señales que se activan en bajo así están señaladas.

Los tiempos de las señales se muestran en la Figura No. 2.11 y 2.12; cabe mencionar que son tiempos aproximados para cualquier impresora de tipo centronics, por lo que el dispositivo se sincroniza por medio del software.

Pin	Señal	E/S
1	$\overline{\text{STROBE}}$	E
2	DATO 1	E
3	DATO 2	E
4	DATO 3	E
5	DATO 4	E
6	DATO 5	E
7	DATO 6	E
8	DATO 7	E
9	DATO 8	E
10	$\overline{\text{ACK}}$	S
11	BUSY	S
12	SIN PAPEL	S
13	ALTO	
14	$\overline{\text{ALF}}$	E
15	N.C.	
16	TIERRA	
17	TIERRA DE CHASIS	
18	N.C.	

Pin	Señal	E/S
19	TIERRA	
20	TIERRA	
21	TIERRA	
22	TIERRA	
23	TIERRA	
24	TIERRA	
25	TIERRA	
26	TIERRA	
27	TIERRA	
28	TIERRA	
29	TIERRA	
30	TIERRA	
31	$\overline{\text{INIT}}$	E
32	$\overline{\text{ERROR}}$	S
33	TIERRA	
34	N.C.	
35	N.C.	
36	N.C.	

Tabla No. 2.1: Configuración Centronics de la Impresora.

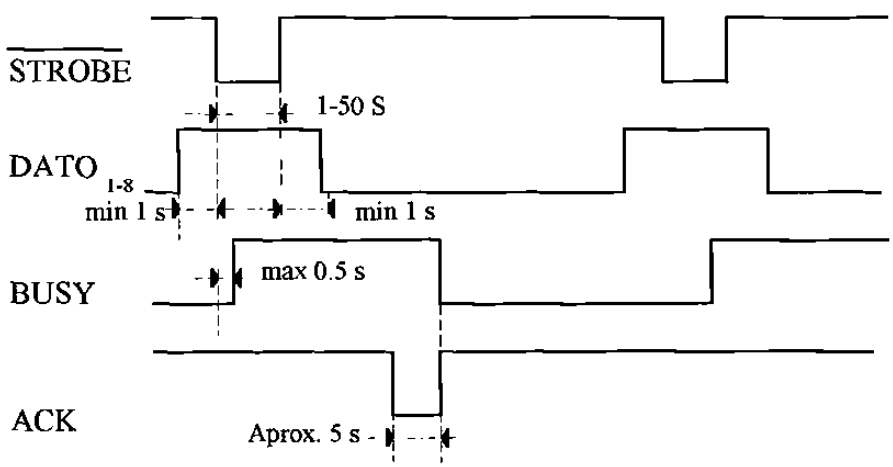


Figura No. 2.11: Diagrama de Tiempos de la entrada de Datos a la Impresora.

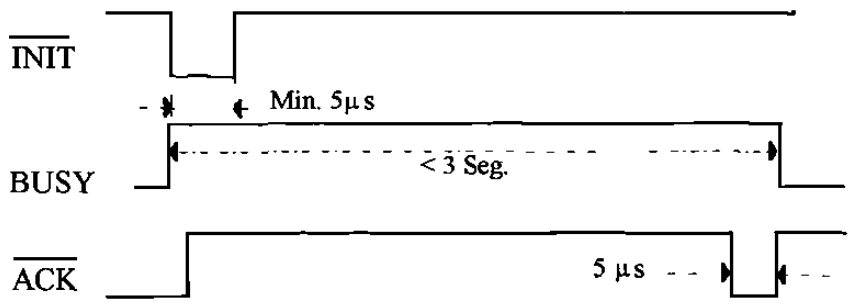


Figura No. 2.12: Diagrama de Tiempos de Inicialización de la Impresora.

Del direccionamiento, he seleccionado las direcciones de las $CC00..$CFFF con el arreglo que se muestra en la Figura No. 2.13, donde la selección se hizo de $A_{11}..A_0$: 11xx xxxx xxxx , como el decoder dá una señal negada, tomando las 4 señales más altas; el PIA necesita tres Chips Selects ($CS_0, CS_1, \overline{CS_2}$) del decoder tomamos uno, $\overline{Y_4}$, que es enviado al $\overline{CS2}$ y de los bloques subsecuentes (de 4 bits) tomamos los de mayor peso es decir A_{10} y A_{11} para la selección en alto.

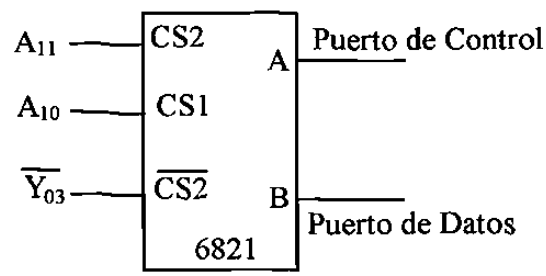


Figura No. 2.13: Configuración del PIA para Transformarlo en Puerto Centronics.

ESTANDAR DE LAS COMUNICACIONES DE DATOS SERIE.

Muchos periféricos como terminales, impresoras, y modems son diseñados para transferir datos en forma ASCII (American Standard Code for Information Interchange), es muy común esta forma de comunicación cuando se requieren bajas velocidades y/o los dispositivos se encuentran en lugares lejanos. Existen dos estandares de mayor uso, el EIA RS-232C y el lazo de corriente de 20 mA, en esta ocasión sólo se determinará el estandar RS-232C que es el estandar utilizado en este trabajo.

Pin	Descripción
1	Tierra de protección
2	Transmite Datos (Tx)
3	Recibe Datos (Rx)
4	Request to Send
5	Clear to Send
6	Data Set Ready
7	Tierra de Señal
8	Received Line Signal Detector
9	Reservada
10	Reservada
11	No Asignada
12	2a. Line Signal Detector
13	2a. Clear to Send
14	Transmite datos (Secundaria)
15	Receiver Signal Element Timing (DCE source)
16	Recibe datos (Secundaria)
17	Receiver Signal Element Timing (DCE source)
18	No asignada
19	Request to Send (Secundario)
20	Data Terminal Ready
21	Signal Quality Detector
22	Ring Indicador
23	Data Signal Rate Selector (DTE/DCE Source)
24	Transmit Signal Element Timing (DTE source)
25	No usada

Tabla No. 2.2: Líneas de la Interfase RS-232C.

EL ESTANDAR EIA RS-232C

Este estandar define la interfase entre terminales de datos y equipo de comunicaciones, empleando intercambio de datos serie binarios. El total de la interfase consiste de 25 líneas de datos, como se muestra en la Tabla No. 2.2; muchas de estas líneas tienen una aplicación muy especializada y la mayoría de las terminales de computadora solamente necesitan de 3 a 5 líneas para operar, en el tema que nos ocupa se utilizó la interfase de comunicación serie asíncrona de acuerdo a la conexión mostrada en la Figura No. 2.14.

COMUNICACION SINCRONA

Con la comunicación paralela es claro que en el borde de la onda del reloj se indica la transferencia de un byte completo, pero en la comunicación serie cada pulso de reloj indica la transferencia de un único bit, y es necesario un procedimiento para saber que bit

del byte se ha transmitido. Un método común es sincronizar la transmisión y la recepción de los registros de corrimiento como se indica en la figura No. 2.15.

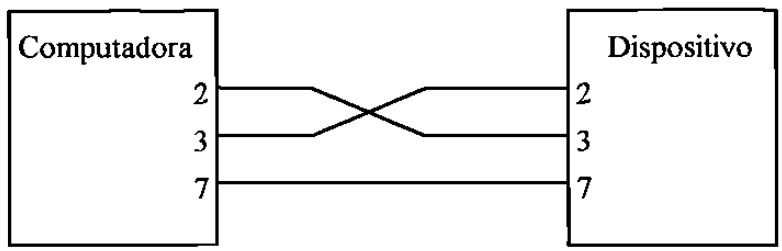


Figura No. 2.14: Conexión de la Interfase Serie RS-232C.

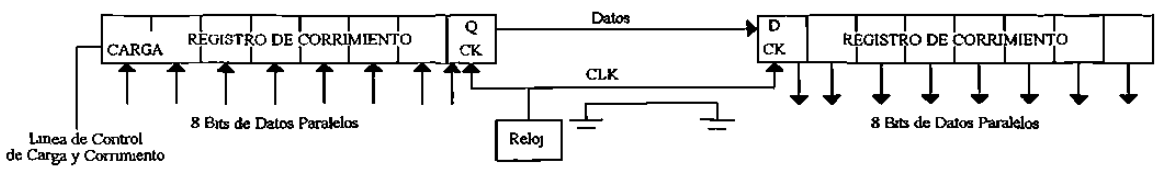


Figura No. 2.15: Transmisión de Datos Serie Síncrono.

Lo práctico de la comunicación síncrona depende de la habilidad de mantenerse sincronizados los dispositivos de transmisión y recepción después de ser sincronizados. La inicialización de la transmisión y recepción generalmente van acompañados de un caracter de identificación. Cuando una transferencia serie síncrona inicia, el receptor es puesto en el primer bit de flujo en modo observador. El primer caracter enviado por el registro trasmisor, es el caracter de identificación, consiste de un patrón predefinido conocido por el receptor. El receptor reorganiza el caracter identificador en el ciclo de reloj que es totalmente puesto en el registro de corrimiento e inicia contado bytes de ocho bits desde este punto.

La transmisión serie requiere de una señal de reloj adicional a las líneas de entrada salida de datos.

COMUNICACION ASINCRONA.

Esta forma de comunicación no necesita la línea de reloj adicional. Esta forma de comunicación depende del hecho de que dos relojes de aproximadamente la misma frecuencia se mantienen bastante bien sincronizados durante un corto período de tiempo.

Un trasmisor de datos asíncrono envía un bit denominado de inicio, seguido de 8 bits de datos y uno o dos bits de parada. El receptor asíncrono sincroniza su reloj con el bit de inicio y los 8 bits de datos usando el reloj receptor como guía.

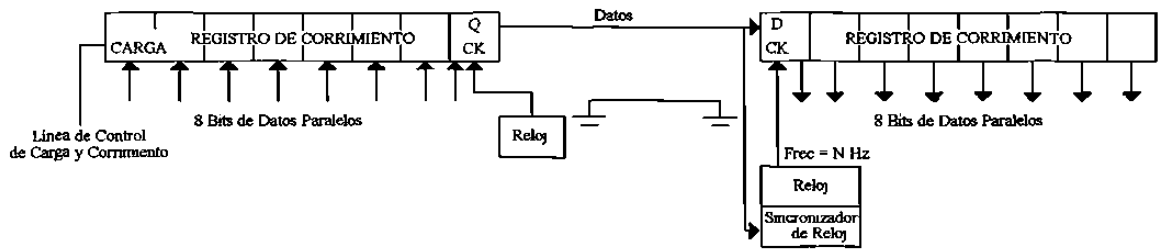


Figura No. 2.16: Transmisión de Datos Serie Asíncrono.

El bit de parada y el fin del flujo de datos son generalmente usados por el equipo receptor para calcular la desviación del reloj y dar una lectura exacta de datos.

La comunicación serie necesita mucho más circuitos de control que la comunicación paralela, por lo que existen dispositivos que nos permiten la transformación de comunicación paralela a serie como el UART (Universal Asynchronous Receiver Transmitter) el USRT (Universal synchronous Receiver Transmitter) y el ACIA (Asynchronous Communication Interface Adapter). En nuestro caso el MC68HC11 nos proveé señales de Trasmisión Recepción en base a un UART interno.

La transmisión de señales será en forma full duplex por lo que se requiere de dos cables uno transmite y otro recibe además de el de tierra, esta conexión se hizo con cable, Figura No. 2.16; aunque existen otras medios como radiofrecuencia, fibra óptica entre otros, este último es digno de mencionarse ya que puede utilizarse para grandes distancias sin que la información se degenere; en la transmisión también se utiliza un bit de paridad adicional el cual puede ser par o impar, este bit completará el número de unos total en el byte para serlo par o impar según sea el caso.

EL MICROCONTROLADOR MC68HC11

La base fundamental del sistema es un microcontrolador de motorola el MC68HC11, aqui se explicarán algunas generalidades y se enfocará principalmente en las características explotadas. Es un microcontrolador CMOS (HCMOS) de alta densidad que contiene en un solo integrado características especiales como puertos de E/S, bus de hasta 2 Mhz, timer de 16 bits, Interfase serie periférica (SPI), Interfase de Comunicación Serie

NRZ (SCI), Interrupciones de Tiempo Real, Ram, Rom y Eeprom internos y Convertidor A/D de 8 bits, el diagrama a bloques se muestra en la figura No. 2.17.

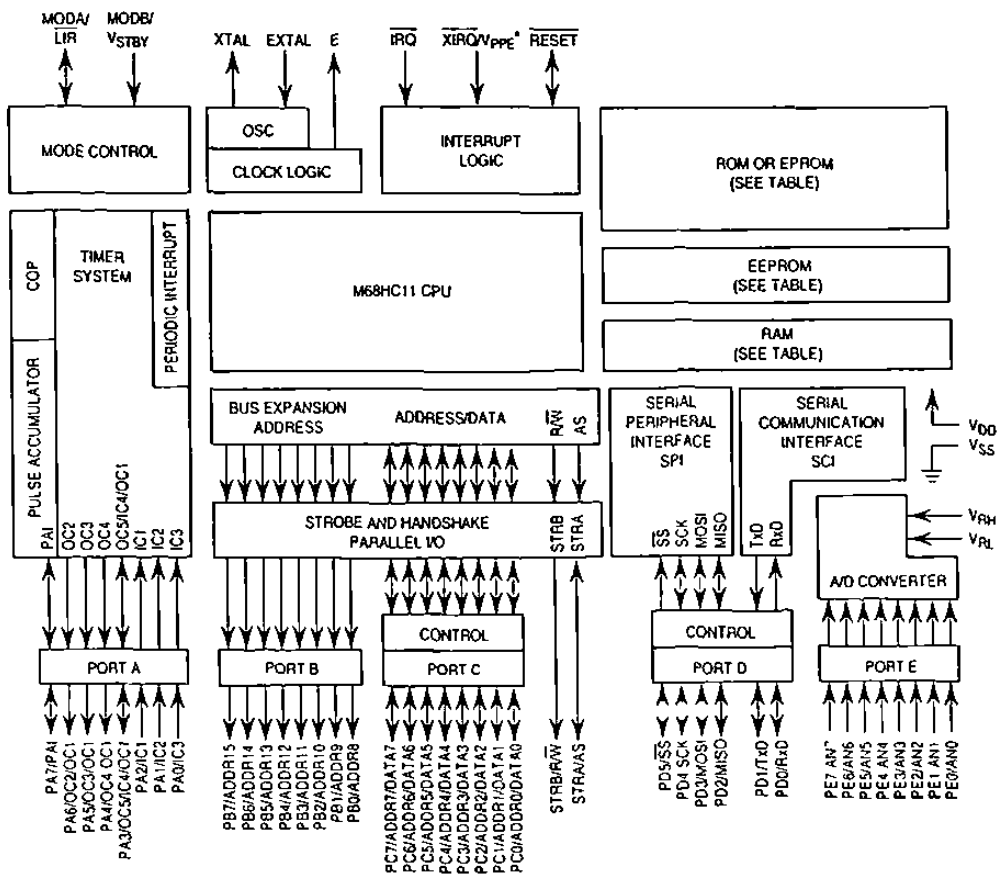


Figura No. 2.17: Diagrama a Bloques del Microcontrolador MC68HC11.

Este microcontrolador (MCU) posee tres modos de operación, modo de chip simple, modo expandido multiplexado, modo bootstrap y modo de prueba; para estas configuraciones utiliza dos pines MODA (25) y MODB (24) de acuerdo a la tabla No. 2.3.

El modo simple es donde se explotan más eficazmente todos los recursos del microcontrolador ya que se puede direccionar 8 bits del puerto A, 8 bits del puerto B, 8 Bits del puerto C, 5 bits del puerto D y 5 Bits del puerto E El modo Boostrap permite la acción de fetch desde el ROM de carga, este modo es muy versátil y puede ser usado para muchas funciones , pruebas, diagnósticos y programar la eeprom, este modo puede ser cambiado por programación. El modo de prueba fue implementado inicialmente para

pruebas al tiempo de fabricación, sin embargo puede ser utilizado para programar la eeprom, además de contar con un bit especial para pruebas especiales a los bits de control. Este modo se puede cambiar a otros por medio de programación.

MODB	MODA	MODO SELECCIONADO
1	0	Chip Simple
1	1	Expandido Multiplexado
0	0	Bootstrap Especial
0	1	Prueba Especial

Tabla No. 2.3: Configuración del Microcontrolador MC68HC11.

El DIP cuenta con 48 pines los cuales serán descritos a continuacción haciendo referencia al número de pin como lo muestra la Figura No. 2.18.

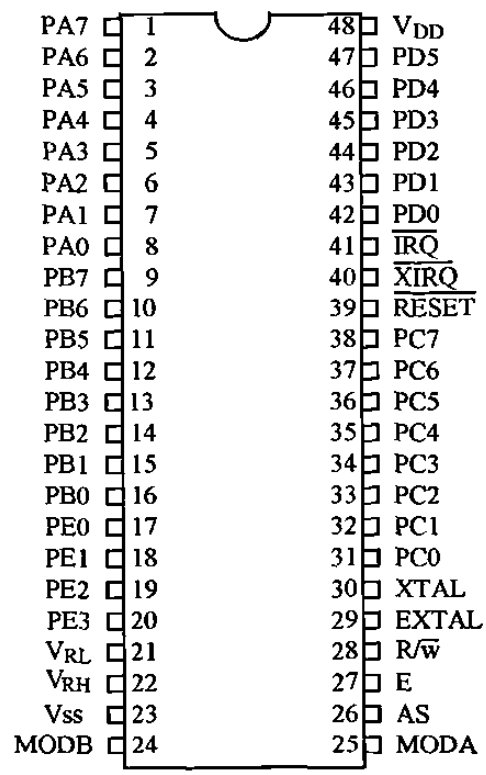


Figura No. 2.18: Pines del MC68HC11.

V_{DD} (48) y VSS (23)

La alimentación de energía al microcontrolador es el V_{DD} = +5 volts (±0.5V) y V_{SS} a tierra.

$\overline{\text{RESET}}$ (39)

Una señal activa en bajo se usa para inicializar el MCU.

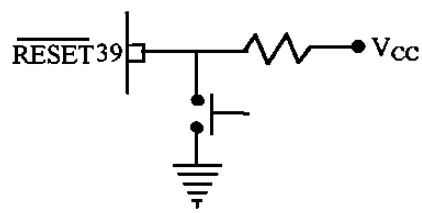


Figura No.2.19: Conexión del Reset Manual.

XTAL (30) y EXTAL (29)

Son las entradas para conectar el oscilador que configura al reloj maestro del sistema.

E (27)

Este pin proporciona una salida de reloj de un cuarto de la frecuencia del oscilador en XTAL y EXTAL, la cual sirve de referencia para los periféricos que se deseen controlar.

$\overline{\text{IRQ}}$ (41)

Este pin es una entrada que permite la posibilidad para aplicar interrupciones asíncronas al MCU, la cual es sensible al nivel bajo de la señal, aunque no se están usando actualmente la conexión se muestra en la Figura No. 2.20. Para la operación normal de este pin se requiere que una resistencia sea conectada a V_{DD}, en este caso se ha conectado una resitencia de 4.7 KΩ.

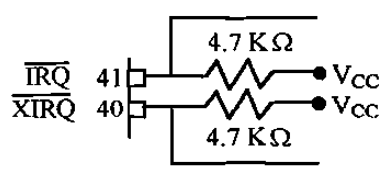


Figura No. 2.20: Conexión de las Entradas a Interrupción.

$\overline{XIRQ}$ (40)

Este pin es una entrada que permite la posibilidad para aplicar interrupciones asíncronas no enmascarables al MCU, la cual es sensible al nivel bajo de la señal. Durante el reset el bit X en el registro de código de condición se activa y para cualquier interrupción es enmascarada hasta que se habilita por software. Para la operación normal de este pin se requiere que una resistencia sea conectada a V_{DD}, en este caso se ha conectado una resitencia de 4.7 KΩ.

MODA (25) y MODB (24)

Son entradas para determinar los modos de operación de acuerdo a los niveles de voltaje presentes en ellas de acuerdo a la Tabla No.2.3. Aqui se ha seleccionado el modo expandido multiplexado.

V_{RL} (21) Y V_{RH} (22)

Estas entradas dan la referencia de voltaje para el circuito convertidor de las entrada análoga a digital; donde V_{RH} podra ser tan pequeña como 3 Vcd mayor que V_{RL}, y V_{RL} y V_{RH} estarán ente V_{SS} y V_{DD}

R / $\overline{W}$ (28)

Esta salida sirve para controlar la dirección de transfernecia del bus externo.

DIRECCIONES INICIO	FIN	OBSERVACIONES
0000	7FFF	RAM EXTERNA
8000	9FFF	LIBRES
A000	BFFF	RESERVADAS
C000	C3FF	PUERTOS DE E/S LIBRES
C400	C7FF	DISPLAY DE POTENCIA
C800	CBFF	LECTORA CODIGO DE BARRAS
CC00	CFFF	IMPRESORA
D000	DFFF	LCD
E000	FFFF	EEPROM

Tabla No. 2.4: Distribución del Mapeo de Memoria del MC68HC11.

AS (26)

Esta salida (Addres Strobe) es usada para demultiplexar las direcciones y datos del puerto C.

PUERTO A (8..1)

Este puerto es configurado para $A_{0..2}$ como entradas y $A_{3..6}$ como salidas, esta configuración es fija y A_7 puede programarse como E/S a través del registro PACTL (Pulse Accumulator Control Register) en nuestro caso aqui está conectado el teclado a los primeros 7 bits.

EL PUERTO B (16..9)

En el modo Expandido este puerto nos proporciona la mitad de direcciones de mayor peso es decir de la 8 a la 15.

EL PUERTO C (31..38)

En modo expandido este puerto nos proporciona las direcciones de la 0 a la 7 y los datos multiplexados.

EL PUERTO D (43..47)

Puede usarse como puerto de propósito general E/S o como SCI o SPI donde el bit 0 es Recibe datos (RxD) y el bit 1 Transmite Datos(TxD); en nuestro proyecto se está usando el puerto serie SCI.

EL PUERTO E (17..20)

Lo he utilizado para propósito general sólo de entrada, pero también puede usarse como entradas Análogo a Digital (A/D), en este puerto estará la llave provisionalmente hasta que se implemente el módulo sensor de temperatura.

LOS REGISTROS INTERNOS

ACUMULADORES A Y B

Los acumuladores A y B son registros de propósito general de 8 bits, son usados como operandos en cálculos aritméticos o para almacenar datos intermedios. En algunas operaciones estos dos registros se unen para formar el registro D de 16 bits como la indica la Figura No. 27. El MCU efectúa instrucciones con estos registros, como lo son la suma (ADD) la sustracción

(SUB), las operaciones de carga (LOAD) de descarga (STA) de comparación (CPA) y los asociados con ambos registros como ABA, SBA y SBA. También contiene otras instrucciones como ABX y ABY que agregan el contenido del registro B al contenido del registro X o Y, pero no existe una instrucción equivalente para el registro A. La instrucción TAP y TPA que transfieren instrucciones del registro de código de condición al registro A y viceversa pero no existen instrucciones equivalentes con el registro B. La instrucción que efectúa un ajuste a decimal al registro A (DAA) se usa después de operaciones aritméticas en decimal codificado en binario (BCD), y no existe el equivalente para el registro B.

Figura No. 2.21: Representación del Registro A y B dónde A∪ B es el Registro D.

EL REGISTRO INDICE X (IX)

El registro índice X (IX) el cual contiene 16 bits que pueden ser usados como contador o como almacenamiento temporal, además poseé el poder de agregarse a un offset de 8 bits para crear dirección efectiva

EL REGISTRO INDICE Y (IY)

El registro índice Y (IY) el cual contiene 16 bits que pueden ser usados como contador o como almacenamiento temporal, además poseé el poder de agregarse a un offset de 8 bits para crear dirección efectiva de igual forma que el registro IX; la diferencia estriba que IY requiere de un byte extra de código de máquina y un ciclo extra de tiempo de ejecución.

EL APUNTADOR DE LA PILA (SP)

El apuntador de la pila (Stack Pointer) es un registro de 16 bits que contiene la siguiente dirección disponible de la pila; en la **pila** o **stack** se guardan las direcciones de las instrucciones pendientes al hacer llamadas a subrutinas e interrupciones además de guardar datos temporales. Cuando se llama a una subrutina por medio de la instrucción JSR (Jump to SubRutine) o con BSR (Branch to SubRutine) las direcciones que siguen a la llamada de subrutina son automaticamente guardadas en la pila (PUSH), y cuando se ejecuta la instrucción

RTS, se ejecuta la acción de extraer la dirección siguiente de ejecutar, de la pila (PUL) y cargada al contador de programa (PC) y se continúa ejecutando el programa. Sí una interrupción es detectada la siguiente instrucción es puesta en la pila y también todos los registros se guardan en la pila, la ejecución continúa con la dirección indicada en el vector de interrupción hasta el fin de la rutina de interrupción (RTI). La instrucción RTI causa que los registros guardados en la pila sean devueltos a éstos (PUL), en orden inverso a como fueron guardados. Para guardar datos que están en los registros A o B las instrucciones son PSHA y PULA, y PSHB y PULB análogamente, para los registros X y Y son PSHX y PULX y PSHY y PULY.

EL CONTADOR DE PROGRAMA (PC)

El PC (Program Counter) es un registro de 16 bits, que contiene la siguiente instrucción que será ejecutada. Después del reset, el PC es inicializado con la dirección contenida en nuestro caso en las direcciones $FFFE..$FFFF, que es el vector de interrupción del $\overline{\text{RESET}}$

EL REGISTRO DE CODIGO DE CONDICION (CCR)

El CCR (Condition Code Register) es un registro de 8 bits que contiene 5 indicadores de código de condición C, V, Z, N, y H (Carry, Overflow, Zero, Negative y Half carry), dos bits de interrupción enmascarable I y X (Irq y Xirq) y un bit de parada S (Stop disable). El CCR es automaticamente actualizado por muchas de las instrucciones ejecutadas; por ejemplo LDAA y STAA activan o borran los banderas del códigos de condición N, Z y V en cambio PSH, PUL, ABX y ABY no afectan el CCR.

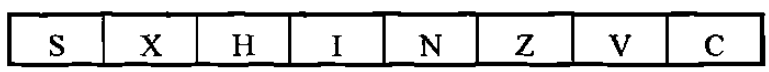


Figura No. 2.22: Configuración del Registro de Código de Condición.

Carry/Borrow (C)

El bit C se activa si la Unidad Aritmética Lógica (ALU) hace un acarreo o pide un préstamo en una operación aritmética. También se activa cuando ocurre un error en una multiplicación o división. Las instrucciones de corrimiento operan con y sin acarreo para facilitar operaciones de corrimiento de palabras múltiples.

Overflow (V)

El bit V se activa cuando una operación aritmética causa un desbordamiento, de otra forma el bit V es desactivado (puesto a cero).

Zero (Z)

El bit Z se activa cuando el resultado de una operación aritmética, lógica o una operación de manipulación de datos es cero; de otra forma el bit Z es puesto a cero.

Negative (N)

El bit N se activa cuando el resultado de una operación aritmética, lógica o una operación de manipulación de datos es negativa (MSB=1). De otra forma el bit N es puesto a cero. Por ejemplo para probar que el contenido de la localización de memoria es negativo se carga a un acumulador y se verifica el estatus del bit N.

Interrupt Mask (I)

El bit I es una máscara global que deshabilita todas las fuentes de interrupción enmascarables. Cuando el bit I se activa, las interrupciones se quedan pendientes, pero la operación del MCU continúa ininterrumpidamente hasta que el bit I es borrado. Después de cualquier reset el bit I se activa y puede ser borrado por software. Después que una interrupción es reconocida, el bit I se activa y luego los registros son colocados en el stack. Normalmente, el bit I es cero después del retorno de la interrupción.

Half Carry (H)

El bit H se activa cuando un acarreo ocurre entre los bits 3 y 4 de la ALU durante una instrucción ADD, ABA o ADC. De otra forma el bit H es borrado, este bit es usado durante operaciones en BCD.

X Interrupt Mask (x)

El bit X deshabilita la interrupción del pin $\overline{\text{XIRQ}}$. Después de cualquier reset X es activo por default y puede ser borrado por software. Cuando la interrupción $\overline{\text{XIRQ}}$ es reconocida los bits X e I son activados en seguida, los registros son guardados en la pila pero primeramente el vector de interrupción es apuntado después de terminar la interrupción; con un RTI causa que los registros sean restablecidos de la pila. El bit X se activa sólo por hardware ($\overline{\text{XIRQ}}$ o

$\overline{\text{RESET}}$). El bit X es borrado solo por instrucciones de programa (TAP cuando el bit asociado de A es cero, o RTI donde el bit 6 del valor cargado en CCR desde la pila ha sido borrado). No existe una acción de hardware para borrar X.

OTROS REGISTROS INTERNOS

A continuación explicaré algunos registros que se han utilizado en el modo expandido multiplexado.

En el registro INIT ($B03D) se utiliza para modificar las direcciones iniciales de RAM interna y de el block de registros, los cuales inicialmente estaban en las localidades $0000-$00FF y $1000-$103F respectivamente, en este registro se colocó el valor de $AB por lo cual las direcciones de la RAM interna a la dirección $A000 $A0FF y el block de registros internos de la $1000-$103F a la $B000-$B03F. De otros registros hablaremos en otros apartados. En la Tabla 2.5 se muestran todos los registros internos, nombres y direcciones.

	Bit 7	Bit 6	Bit 5	Bit 4	bit 3	Bit 2	Bit 1	Bit 0	
$B000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	Puerto A
$B001									Reservada
$B002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC
$B003	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	Puerto C
$B004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	Puerto B
$B005	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0	PORTCL
$B006									Reservada
$B007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
$B008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	Puerto D
$B009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
$B00A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	Puerto E
$B00B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC
$B00C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OC1M
$B00D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OC1D
$B00E	Bit 15	14	13	12	11	10	9	8	TCNT(h)
$B00F	Bit 7	6	5	4	3	2	1	0	TCNT(L)
$B010	Bit 15	14	13	12	11	10	9	8	TIC1(h)
$B011	Bit 7	6	5	4	3	2	1	0	TIC1(L)
$B012	Bit 15	14	13	12	11	10	9	8	TIC2(h)
$B013	Bit 7	6	5	4	3	2	1	0	TIC2(L)
$B014	Bit 15	14	13	12	11	10	9	8	TIC3(h)
$B015	Bit 7	6	5	4	3	2	1	0	TIC3(L)
$B016	Bit 15	14	13	12	11	10	9	8	TOC1(h)
$B017	Bit 7	6	5	4	3	2	1	0	TOC1(L)
$B018	Bit 15	14	13	12	11	10	9	8	TOC2(h)
$B019	Bit 7	6	5	4	3	2	1	0	TOC2(L)

Tabla No. 2.5a: Block de 64 Registros Internos del MC6811HC11.

$B01A	Bit 15	14	13	12	11	10	9	8	TOC3(h)
$B01B	Bit 7	6	5	4	3	2	1	0	TOC3(L)
$B01C	Bit 15	14	13	12	11	10	9	8	TOC4(h)
$B01D	Bit 7	6	5	4	3	2	1	0	TOC4(L)
$B01E	Bit 15	14	13	12	11	10	9	8	TI4/O5(h)
$B01F	Bit 7	6	5	4	3	2	1	0	TI4/O5(L)
$B020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1
$B021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2
$B022	OC1I	OC2I	OC3I	OC4I	I4/O5I	IC1I	IC2I	IC3I	TMSK1
$B023	OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F	TFLG1
$B024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
$B025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2
$B026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL
$B027	Bit 7	6	5	4	3	2	1	Bit 0	PACNT
$B028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
$B029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR
$B02A	Bit 7	6	5	4	3	2	1	Bit 0	SPDR
$B02B	TCLR	SCP2	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD
$B02C	R8	T8	0	M	WAKE	0	0	0	SCCR1
$B02D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
$B02E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR
$B02F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SCDR
$B030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL
$B031	Bit 7	6	5	4	3	2	1	Bit 0	ADR1
$B032	Bit 7	6	5	4	3	2	1	Bit 0	ADR2
$B033	Bit 7	6	5	4	3	2	1	Bit 0	ADR3
$B034	Bit 7	6	5	4	3	2	1	Bit 0	ADR4
$B035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT
$B036	MBE	0	ELAT	EXCOL	EXROW	T1	T0	PGM	EPROG
$B037									Reservada
$B038									Reservada
$B039	ADPU	CSEL	IRQE	DLY	CME	0	CR1	CR0	OPTION
$B03A	Bit 7	6	5	4	3	2	1	Bit 0	COPRST
$B03B	ODD	EVEN	ELAT	BYTE	ROW	ERASE	EELAT	EPGM	PPROG
$B03C	RBOOT	SMOD	MDA	IRVNE	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO
$B03D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT
$B03E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1
$B03F	EE3	EE2	EE1	EE0	NOSEC	NOCOP	ROMON	EEON	CONFIG

Tabla No. 2.5b: Block de 64 Registros Internos del MC6811HC11.

TIPOS DE DATOS

El MC68HC11 soporta los siguientes tipos de datos:

1) Por bit

2) Enteros de 8 y 16 bits con signo y sin signo

3) Fracciones sin signo de 16 bits.

4) Direcciones de 16 bits.

Los datos no usados fueron operaciones con signo y el especificado en el punto 3). En el Anexo B, se muestran los el concentrado de nemónicos del MC68HC11.

MODOS DE DIRECCIONAMIENTO

El MC68HC11 contiene 6 modos de direccionamiento que son el inmediato, directo, extendido, indexado, inherente y relativo. Todos los modos usan una dirección efectiva excepto el inherente. La dirección efectiva es la dirección de memoria desde la cual el argumento es buscado o cargado, o la dirección desde la cual se está ejecutando. La dirección efectiva puede ser especificada dentro de una instrucción o esta puede ser calculada.

Modo Inmediato

En este modo el argumento es contenido en el (los) byte(s) inmediados seguidos del código de instrucción. La dirección es señalada inmediatamente del código de instrucción.

El Modo Directo

El byte de menor orden de la dirección del operando es contenido en un byte simple seguido del código de operación, y el byte de más alto orden de la dirección se asume que es $00. Direcciones de $00-$FF son así accesadas directamente usando instrucciones de dos bytes. El tiempo de ejecución se reduce por la eliminación del acceso de memoria adicional requerida para la dirección del byte de mayor orden. En muchas aplicaciones, estos 256 bytes son reservados frecuentemente para referenciar datos. En los MCU's MC68HC11, el mapa de memoria puede ser configurado por combinaciones de registros internos, RAM, o memoria externa para ocupar estas direcciones.

Modo Extendido.

En este modo la dirección efectiva del argumento es contenida en los dos bytes seguidos del byte del código de instrucción. Existen instrucciones de tres bytes (o cuatro bytes si un prebyte es requerido). Uno o dos bytes son necesarios para el código de instrucción y dos para la dirección efectiva.

Modo Indexado.

En este modo, es necesario un offset de 8 bits sin signo contenido en la instrucción, es agregado al valor contenido en uno de los registros índices (IX o IY). La suma es la dirección efectiva. Este modo de direccionamiento siempre es referenciado a alguna localización de memoria en los 64 Kbyte de espacio direccionable. Estas son instrucciones de dos a cinco bytes, dependiendo de si requiere o no un prebyte.

Modo Inherente

En este modo la información necesaria para ejecutar la instrucción es contenida en el código de instrucción. Operaciones que usan sólo los registros índices o acumuladores, tales como instrucciones de control sin argumentos, son incluidos en este modo de direccionamiento. Estas son instrucciones de uno o dos bytes.

Modo Relativo.

Este modo se usa sólo en instrucciones Branch. Si la condición Branch es verdadera, se incluye un offset de 8 bits con signo a la instrucción y se agrega al contenido del contador de programa (PC) para formar la dirección efectiva de la bifurcación. De otra forma el control continúa con la siguiente instrucción. Conforman usualmente la instrucción por dos bytes.

RESETS E INTERRUPCIONES.

Las operaciones de reset y las operaciones de interrupción cargan al PC con un vector que apunta a la nueva localización desde donde la dirección va a ser ejecutada. Un reset automáticamente detiene la ejecución de la instrucción actual y fuerza al MCU a reiniciar el programa. Una interrupción suspende temporalmente la ejecución de un programa normal, iniciando un servicio adicional de acuerdo a la petición de la interrupción. Después que la interrupción ha sido terminada el programa continúa con la siguiente instrucción del programa principal.

RESETS

Existen cuatro posibles maneras de ejecutar un reset. El reset por encendido POR (Power-on Reset) y el reset externo normal parten del vector normal de reset y el COP (Computer Operating Properly) y el reset del reloj monitor que tienen cada uno su propio vector de reset.

Efectos del Reset.

Cuando una condición de reset es reconocida, los registros internos y los bits de control son forzados a inicializarse. Dependiendo de las causas del reset y el modo de operación, los vectores de reset pueden ser localizados desde una de 6 localizaciones posibles mostradas en la Tabla No. 2.6.

Mapa de Memoria.

Después del reset , el registro INIT ($B03D) es inicializado a $00 por lo que es necesario contar con esto para que el sistema coloque nuevamente el valor de $AB en el registro INIT.

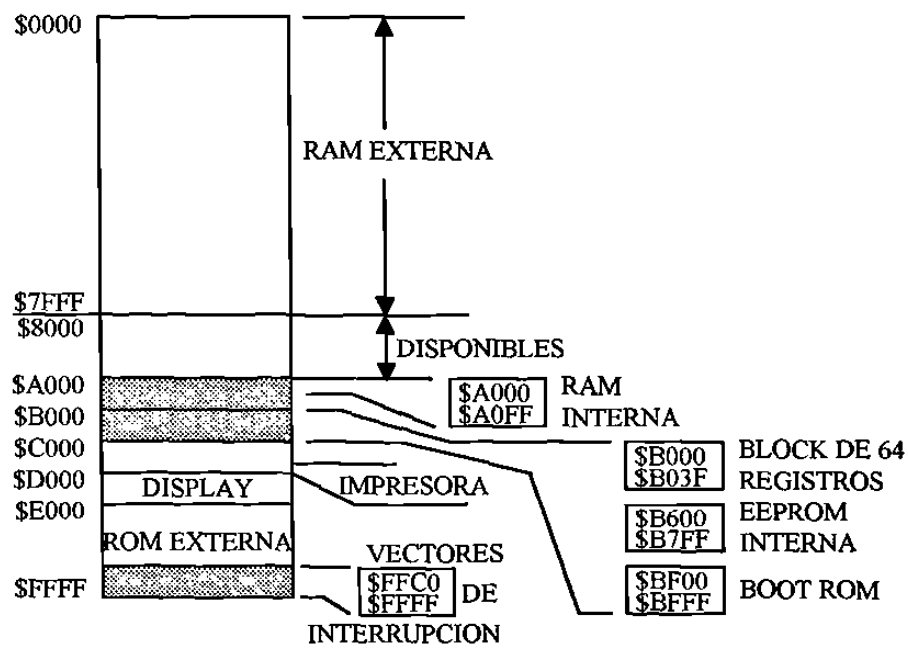


Figura No. 2.23: Representación Gráfica del Mapeo de Memoria.

Interrupciones de Tiempo Real (RTI).

La bandera de interrupción de tiempo real RTIF (bit 6 de TFLG2, $B025) es borrado y automaticamente las interrupciones por hardware son enmascaradas, éstas pueden ser inicializadas por software antes de que las RTI sean usadas.

Para una interpretación más objetiva del Reset y las Interrupciones se muestran en el Diagrama de Flujo No.2.1 y 2.2, la secuencia y las prioridades de éstas.

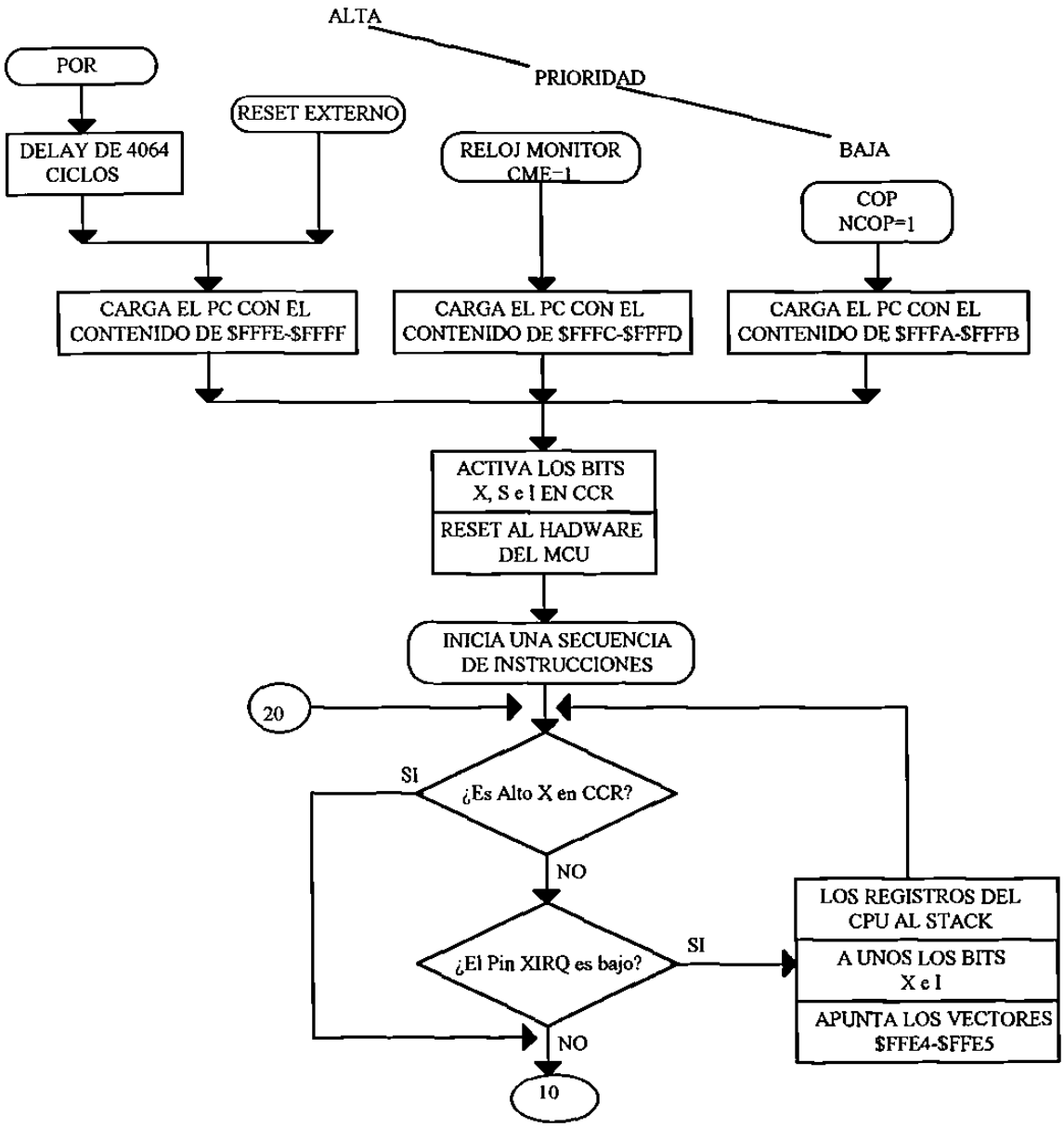


Diagrama de Flujo No. 2.1a: Prioridades del Reset (1 de 2).

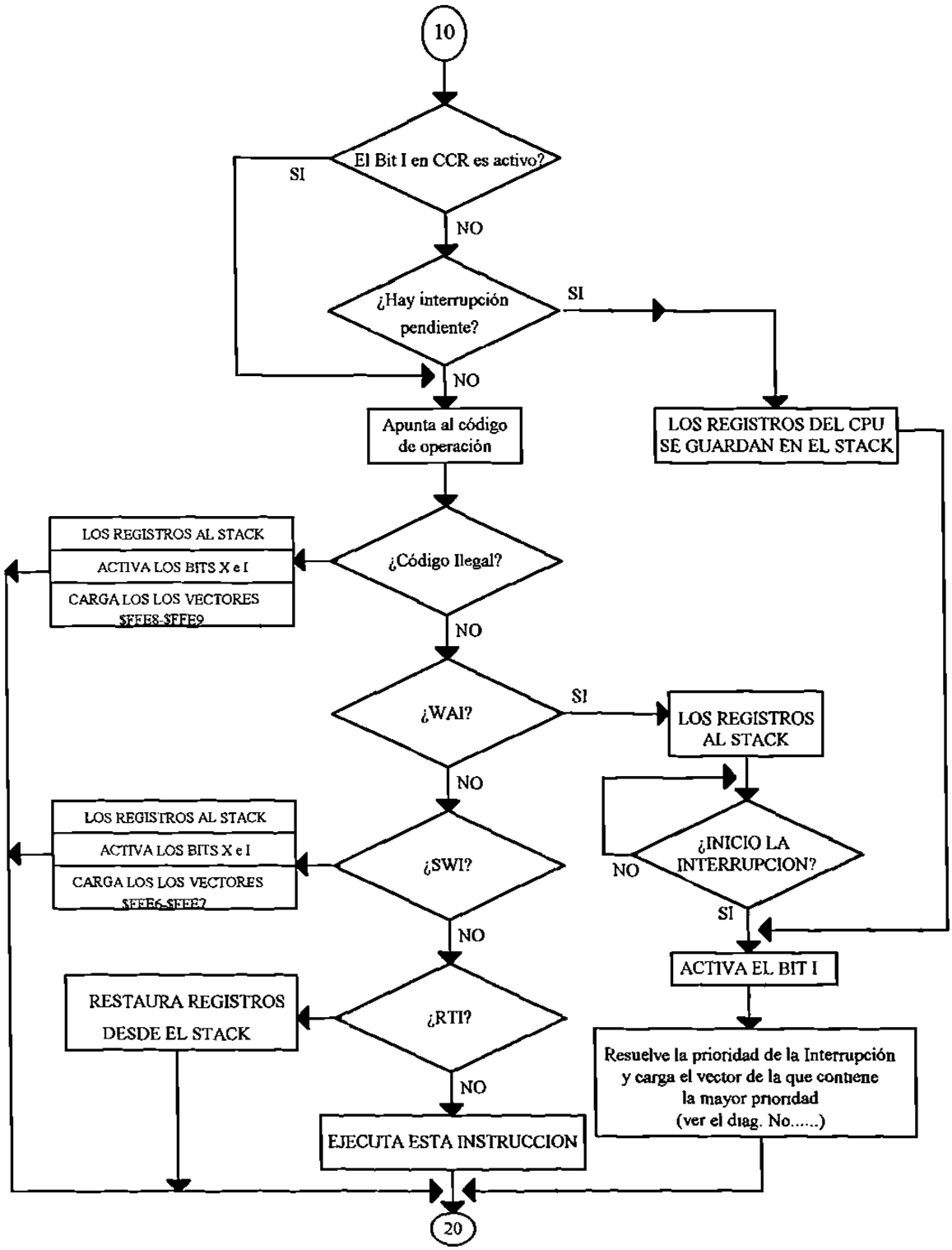


Diagrama de Flujo No. 2.1b: Prioridades del Reset (2 de 2)

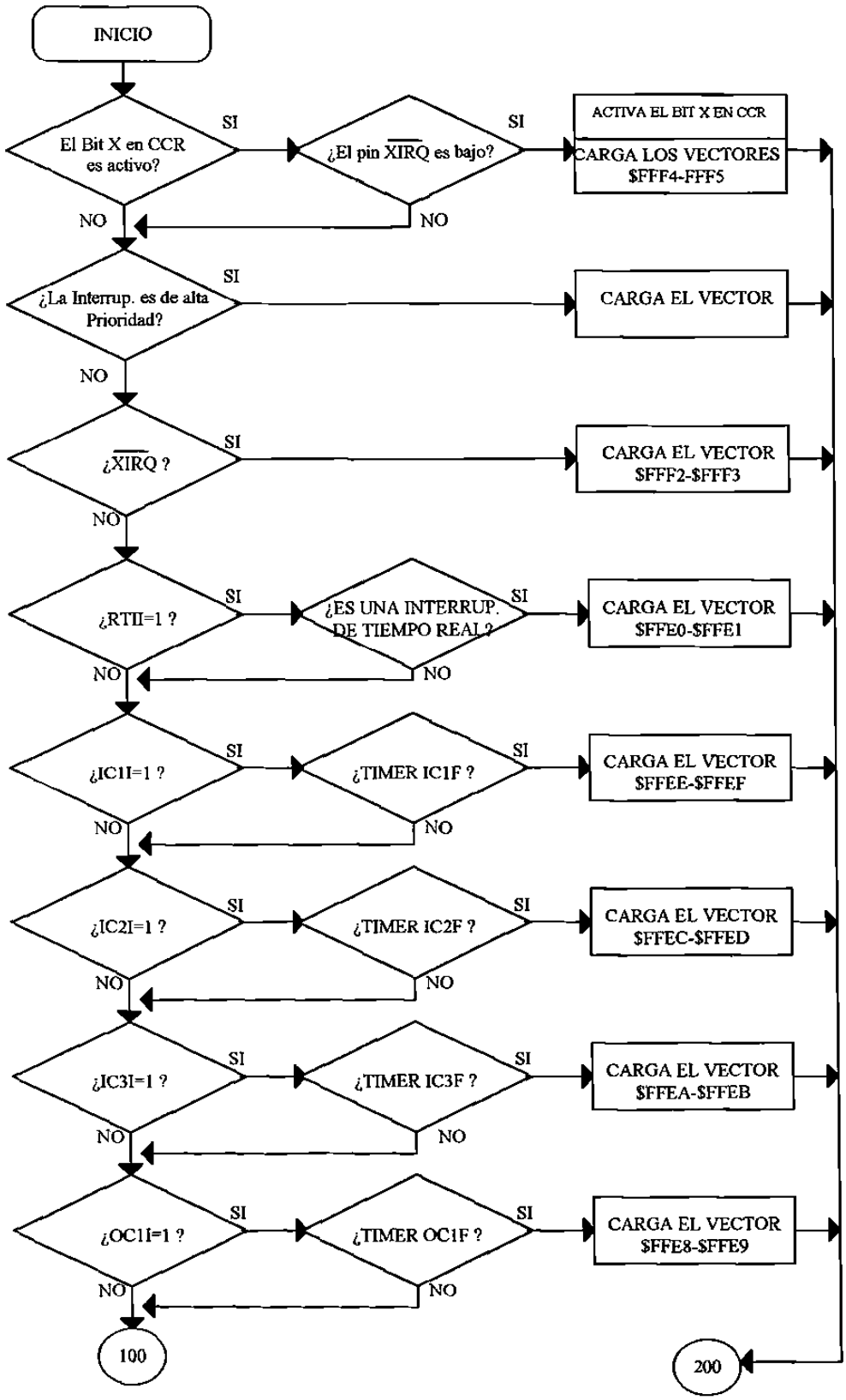


Diagrama de Flujo No. 2.2a: Prioridad de las Interrupciones (1 de 2).

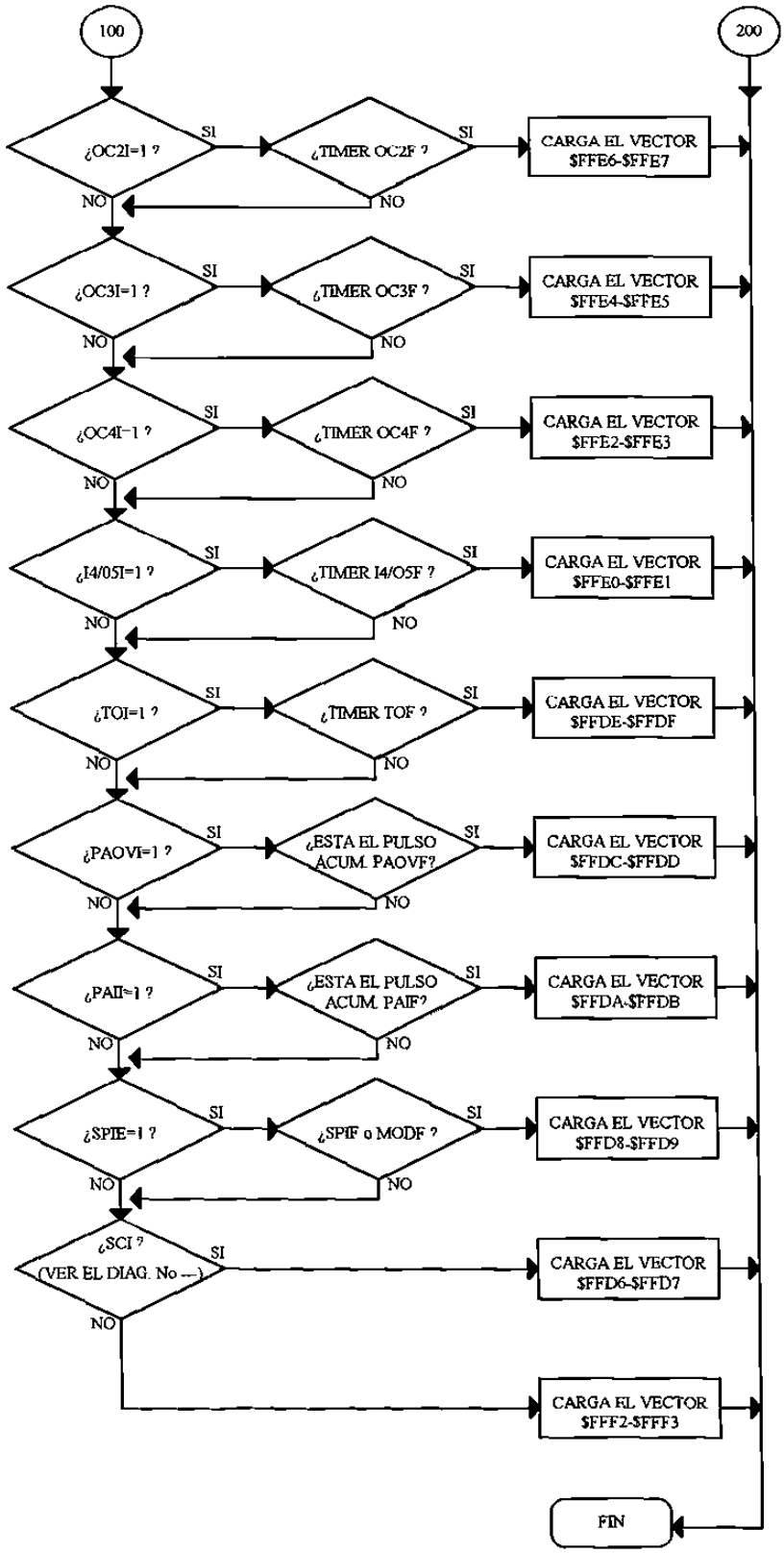


Diagrama de Flujo No. 2.2b: Prioridad de las Interrupciones (2 de 2).

INTERRUPCION DE TIEMPO REAL (RTI).

El Sistema de Tiempo del MC68HC11 está compuesto por cinco cadenas de relojes divisores. El principal incluye un contador de 16 bits de carrera libre, el cual es conducido por un preescalar programable. La salida del preescalar divide el sistema de reloj por 1, 4, 8 ó 16., la interacción de los registros se muestra en la Figura No.2.24. En nuestro caso solo se verán los principales que se han utilizado.

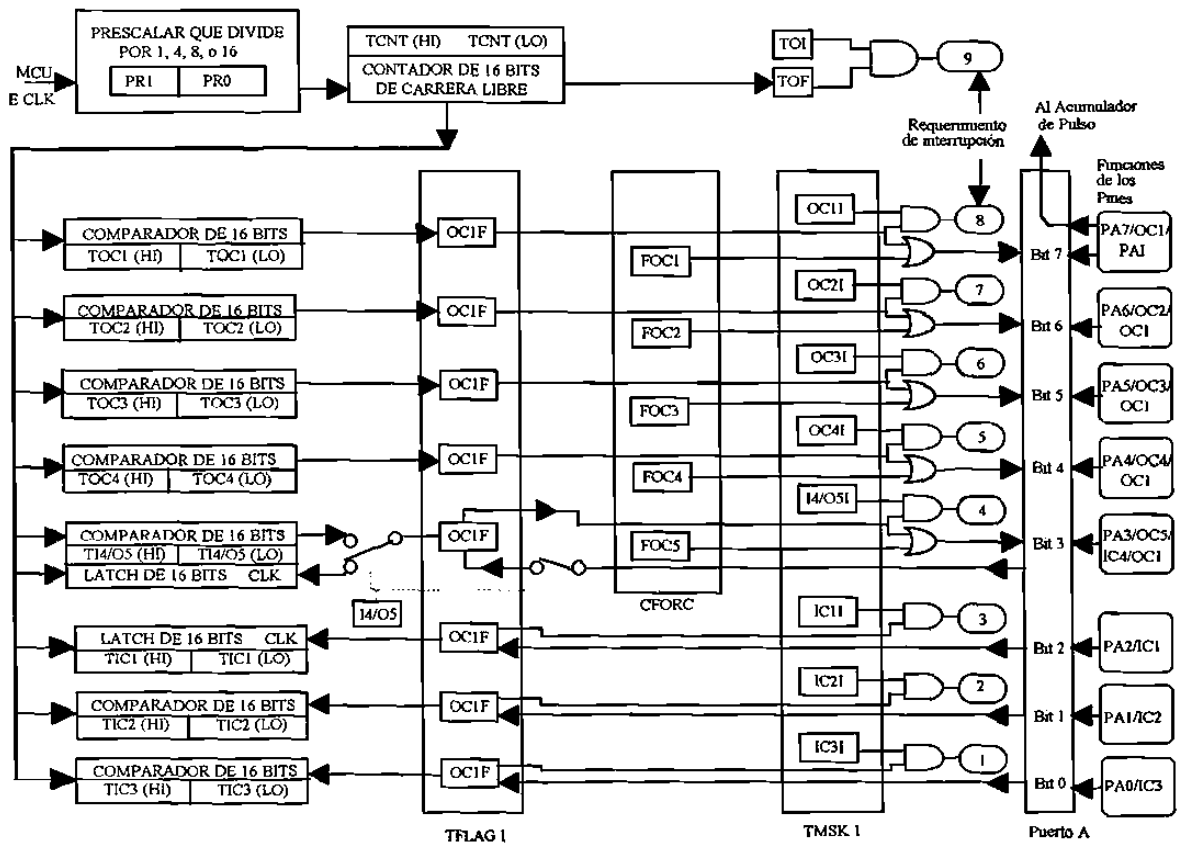


Figura No. 2.24: Diagrama a Bloques del Sistema de Timer

La función captura de entrada relaciona el tiempo y eventos externos ocurridos por el valor visto del contador de carrera libre donde un pulso es detectado con la entrada asociada al timer. El software puede cargar valores conocidos y usar éstos para calcular la periodicidad y duración de eventos; el cuales se usaran para implementar un reloj de tiempo real.

La RTI (Real-Time Interrupt) que se ha habilitado en nuestro proyecto, se usa para generar interrupciones con un rango periódico fijo, es configurado y controlado por dos

bits, (RTR1 y RTR0) y el registro de control del pulso acumulador (PACTL). El bit RTII y el registro TMSK2 habilitan la capacidad de generar la interrupción. Poseé cuatro diferentes rangos producidos de acuerdo a la frecuencia del oscilador y el valor de los bits RTR[1:0] como lo muestra la Tabla No. 2.7.

RTR[1:0]	E=1 MHz	E=2MHz	E=3MHz	E=X MHz
00	2.731 ms	4.096 ms	8.192 ms	$(E/2^{13})$
01	5.461 ms	8.192 ms	16.384 ms	$(E/2^{14})$
10	10.923 ms	16.384 ms	32.768 ms	$(E/2^{15})$
11	21.845 ms	32.768 ms	65.536 ms	$(E/2^{16})$

Tabla No. 2.7: Rangos de RTI.

El reloj fuente del la función RTI es un reloj de carrera libre que no puede ser parado o interrumpido, excepto por el reset. Para esta implementación se han usado sólo algunos registros del sistema del timer.

TMSK2 (Timer Interrupt Mask Register 2)

Este registro contiene los bits que habilitan la interrupción de tiempo real.

$B024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
RESET	0	0	0	0	0	0	0	0	

Figura No. 2.25: Representación del TMSK2.

TOI (Timer Overflow Interrupt Enable)

0 = La interrupción TOF deshabilitada.

1 = Requerimiento de interrupción cuando TOF se activa a uno.

RTII (Real Time Interrupt Enable)

0 = La interrupción RTIF deshabilitada.

1 = Requerimiento de interrupción cuando RTIF se activa a uno.

PAOVI (Pulse Accumulator Overflow Interrupt Enable)

El bit de control PAOVI permite configurar al acumulador de pulsos de sobreflujo para extraer o conducir a una interrupción y no afecta al estado del PAOVF en el registro TFLG2.

PAII (Pule Accumulator Input Edge).

Cuando este bit se activa una interrupción de hardware es requerida y es generada cada vez que el bit PAIF del registro TFLG2 es activo.

Bits[3:2] no se usan, siempre son cero.

PR[1:0] Timer Prescalar Select.

Estos bits son usados para seleccionar la proporción de la división del preescalar. Se refiere a la Tabla No. 2.8.

PR[1:0]	Preescalar
00	1
01	4
10	8
11	16

Tabla No. 2.8: El Preescalar del Timer.

TFLG2 (Timer Interrupt Flag Register 2)

Los bits de este registro indican que ha ocurrido un evento del sistema timer. Se asocia con los cuatro bits de mayor orden del TMSK2, los bits del TFLG2 permiten al subsistema operar en una interrupción. Cada bit de TFLG2 corresponde a un bit en TMSK2 en la misma posición.

$B025	TOF	RTIF	PAOVF	PAIF	-	-	-	-	TFLG2
RESET	0	0	0	0	0	0	0	0	

Figura No. 2.26: Representación del TFLG2.

TOF (Timer Overflow Interrupt Flag)

Se activa cuando TCNT cambia desde $FFFF a $0000

RTIF (Real Time Interrupt Flag)

Se activa automáticamente a uno cuando finaliza un período del RTI. Para borrar RTIF, se escribe un byte a TFLG2 con el bit 6 activo.

PAOVF (Pulse Accumulator Overflow Interrupt Flag) y PAIF (Pulse Accumulator Input Edge Interrupt Flag).

El estatus del bit PAOVF se activa cada vez que el acumulador de pulsos pasa de $FF a $00. Para borrar el bit de estatus, escribimos un uno en la correspondiente posición del bit de dato del registro TFLG2.

Bits[3:0] No se usan y siempre son cero.

PACTL (Pulse Accumulator Control Register)

Los bits RTR[1:0] de este registro seleccionan el rango para el sistema RTI (Real Time Interrupt). El resto son bits que controlan el acumulador de pulsos y las funciones IC4/OC5.

$B026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL
RESET	0	0	0	0	0	0	0	0	

Figura No. 2.27: Representación de los bits del PACTL.

DDRA7 (Data Direction for Port A Bit 7)

Este bit se utiliza para configurar el bit 7 del puerto A

0 = Entrada

1 = Salida

PAEN (Pulse Accumulator System Enable)

0 = Deshabilita el Acumulador de Pulsos

1 = Habilita el Acumulador de Pulsos.

PAMOD (Pulse Accumulator Mode)

0 = Contador de Eventos.

1 = Acumulación de Tiempo.

PEDGE (Pulse Accumulator Edge Control)

Este bit tiene diferentes usos dependiendo del estado del bit PAMOD, lo que se muestra en la Tabla No.2.9.

DDRA3 (Data Direction for port A Bit 3)

Se utiliza para configurar el bit 3 del puerto A.

0 = Entrada

1 = Salida

PAMOD	PEDGE	ACCION DEL RELOJ
0	0	En la caída de la onda PAI incrementa el contador
0	1	En el ascenso de la onda PAI incrementa el contador.
1	0	Un cero en PAI inhibe el contador
1	1	Un uno en PAI inhibe el contador

Tabla No. 2.9: Control de la Onda del Pulso Acumulador.

I4/O5 (Input Capture 4/Output Compare)

0 = Función de Comparador de Salida 5 Habilitado. (No IC4)

1 = Función de Captura de Entrada 4 habilitada (No OC5)

RTR[1:0] RTI Interrupt Rate Select.

Estos dos bits determinan el rango en el cual el sistema RTI requiere la interrupción. El sistema RTI es conducido por una división de E entre 2^{13} pulsos de reloj que son compensados independientemente del timer preescalar. Estos dos bits de control seleccionan un factor adicional de división de acuerdo a la Tabla No. 2.7.

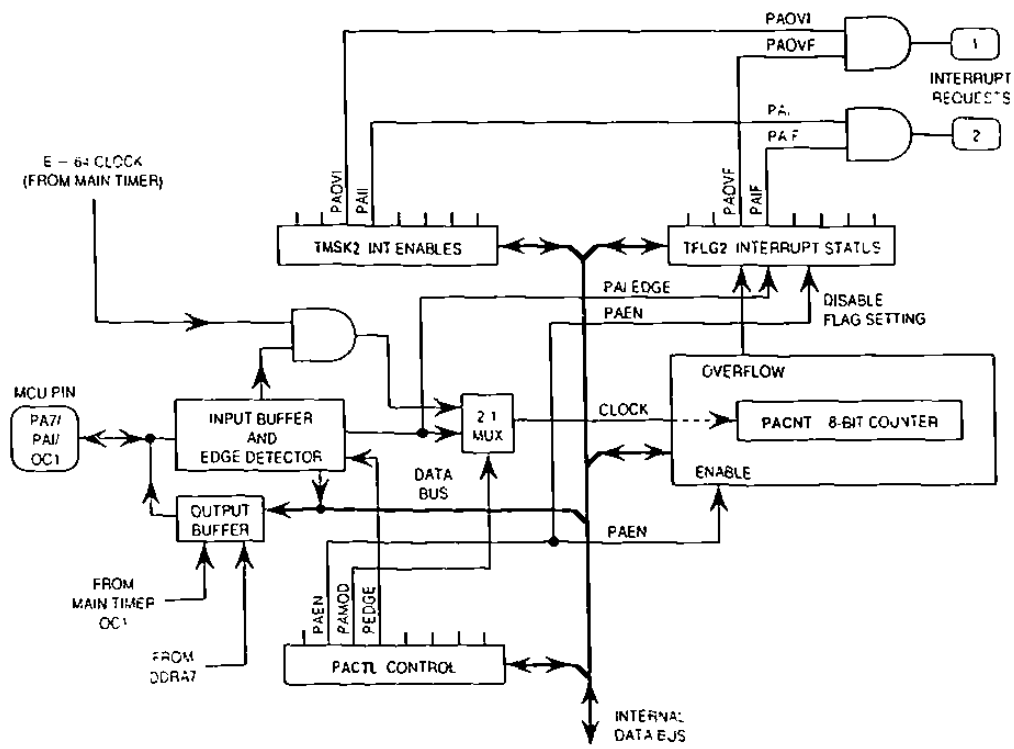


Figura No. 2.28: Acumulador de Pulsos.

ACUMULADOR DE PULSOS

El MCU tiene un contador de 8 bits que puede ser configurado para operar como un simple contador de eventos o para encerrar tiempo de acumulación, dependiendo del estado del bit PAMOD en el registro PACTL, de acuerdo al diagrama de la Figura No. 2.28.

INTERFASE SERIE

El MCU MC68HC11 contiene dos tipos de puertos serie el sistema de Interfase de Comunicación Serie SCI (Serial Comunication Interface) y el puerto Serie Síncrono SPI (Serial Peripherical Interface); en este caso está habilitado el SCI por las razones que se describen a continuación

Descripción General del SCI

El SCI es un sistema asíncrono full-duplex tipo UART (Universal Asynchronous Receiver Transmitter), usa el formato standar NRZ (NonReturn to Zero), un bit de inicio, ocho bits de datos y un bit de parada. Un generador de baudios que depende del oscilador del MCU. El transmisor y el receptor son funcionalmente independientes pero usan el mismo formato de datos y la misma velocidad Tx/Rx (Bauds) en la salida presentan niveles de voltaje TTL para trasladarse a los niveles de voltaje para la interfase RS232 (±12 V). El SCI incluye características avanzadas para asegurar la alta confiabilidad en las comunicaciones; el diagrama bloques del transmisor se muestra en la Figura No. 2.31 y el Receptor en la figura No. 2.32.

Registros del SCI y bits de Control.

$B008	0	0	$\overline{SS}$	SCK	MOSI	MISO	Tx	Rx	Puerto D
RESET	0	0	0	0	0	0	0	0	
$B009	0	0	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
RESET	0	0	0	0	0	0	0	0	
$B028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
RESET	0	0	0	0	0	1	0	0	

Figura No. 2.29: Registros de Datos de Comunicación Serie (SCDR).

El SCI es configurado y controlado por cinco registros BAUD, SCCR1, SCCR2, SCSR y SCDR. Agregando, el registro del puerto D, el registro de dirección para el puerto D (DDRD), y el puerto D en modo de OR alambrado en el registro de control SPI (SPCR) son relacionados en segundo orden en el sistema SCI.

$B02F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SCDR
RESET	1	1	1	1	1	1	1	1	

Figura No. 2.30: Representación del Registro de Datos de Comunicación Serie.

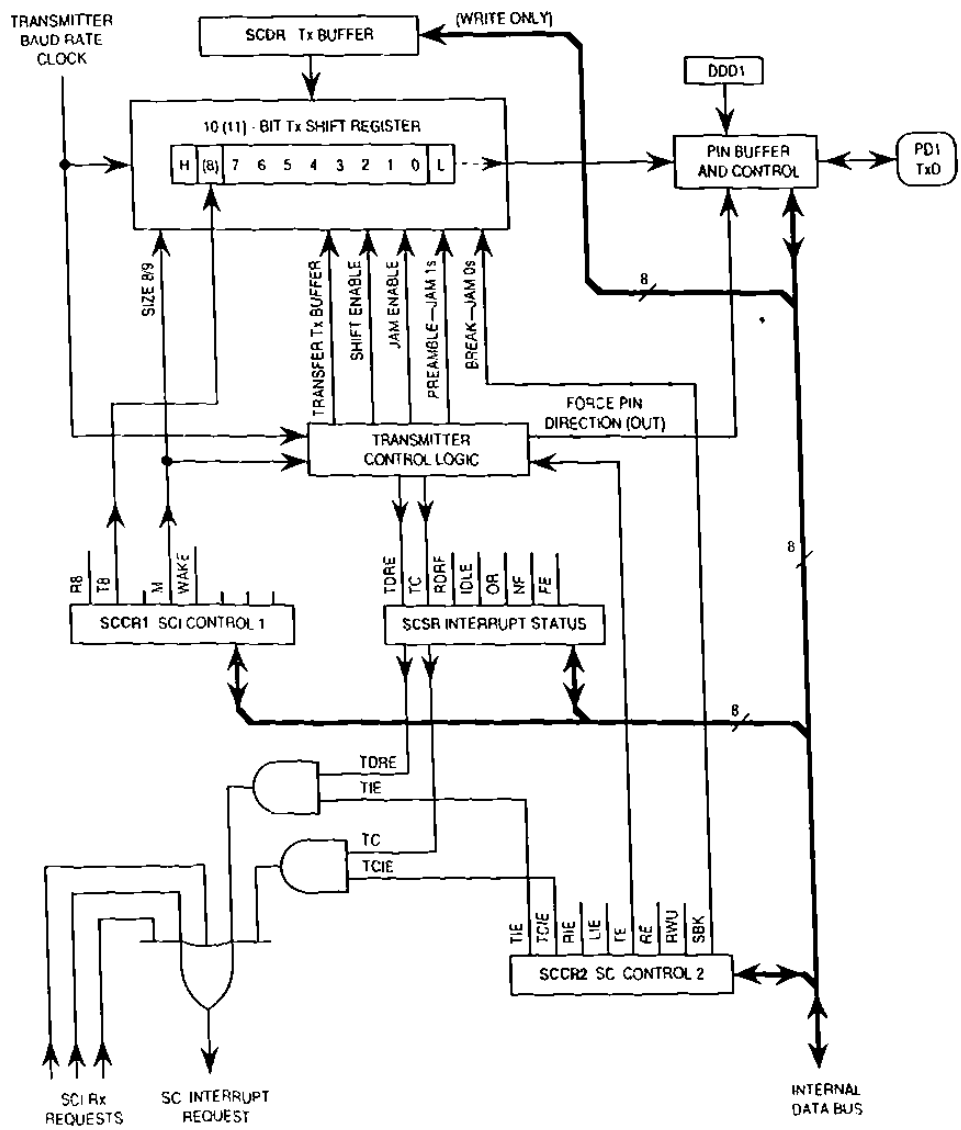


Figura No. 2.31: Diagrama a Bloques del Transmisor del SCI del MC68HC11.

Registro de Datos de Comunicación Serie (SCDR)

El SCDR (Serial Comunications Data Register) es un registro paralelo que tiene dos funciones. Es receptor de datos cuando está leyendo, y es un registro transmisor de datos cuando está el sistema transmitiendo.

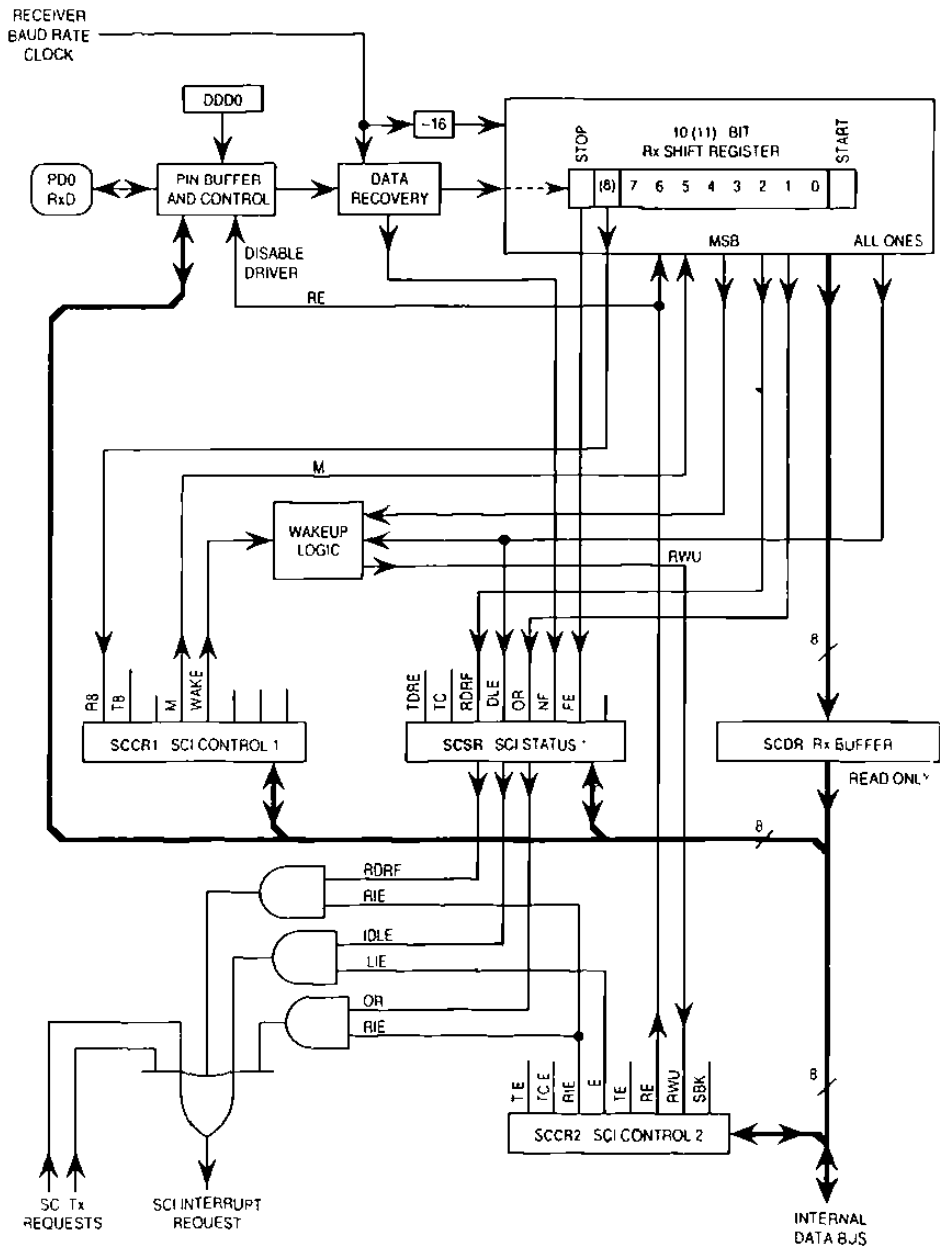


Figura No. 2.32: Diagrama a Bloques del Receptor del SCI del MC68HC11.

El Registro de Control 1 de Comunicación Serie (SCCR1)

El SCCR1 (Serial Communication Control Register 1) proporciona el bit para determinar la longitud de palabra y seleccionar el método para usar las características de Wake up.

R8: Recibe datos de 8 bits.

Si el bit M es activo, R8 carga el noveno bit en el receptor.

T8: Transmite datos de 8 bits.

Si el bit M es activo, R8 carga el noveno bit en el Transmisor Receptor.

Bit 5: No implementado. Siempre es cero.

$B02C	R8	T8	0	M	WAKE	0	0	0	SCCR1
RESET	1	1	0	0	0	0	0	0	

Figura No. 2.33: Representación del Registro de Control Serie del SCI.

M: Modo (Selección del formato de longitud de palabra)

0 = Un bit de inicio, 8 de datos y un bit de parada

1 = Un bit de inicio, 9 de datos y un bit de parada

WAKE: Activa Marca/Dirección Libre.

0 = Activa para reconocimiento de línea Disponible.

1 = Activa para marca de dirección.

Bit[2:0]: No implementados. Siempre son ceros.

El Registro de Control 2 de Comunicación Serie (SCCR2)

El SCCR2 (Serial Communication Control Register 2) habilita o deshabilita las funciones del SCI.

$B02D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
RESET	0	0	0	0	0	0	0	0	

Figura No. 2.34: Representación del Registro de Control 2 del SCI.

TIE: Transmit Interrupt Enable.

0 = Deshabilita la interrupción TDRE.

1 = Requerimiento de interrupción SCI cuando el estatus de la bandera TDRE está alto.

TCIE: Transmit Complete Interrupt Enable.

0 = Deshabilita la interrupción TC.

1 = Requerimiento de interrupción SCI cuando el estatus de la bandera TC está alto.

RIE: Receiver Interrupt Enable.

0 = Interrupciones RDRF y OR deshabilitadas.

1 = Requerimiento de interrupción SCI cuando el estatus de la bandera RDRF o OR está alto.

ILIE: Idle Line Interrupt Enable.

0 = Deshabilita las interrupciones IDLE.

1 = Requerimiento de interrupción SCI cuando el estatus de la bandera IDLE está alto.

TE: Transmitter Enable.

Cuando TE va desde cero a uno, una unidad de caracteres libres (unos lógicos) son enviados como un antecedente.

0 = Transmisor deshabilitado.

1 = Transmisor habilitado.

RE: Receiver Enable.

0 = Receptor deshabilitado.

1 = Receptor habilitado.

RWU: Receiver Wakeup Control.

0 = Receptor SCI normal.

1 = Habilita el Wakeup y las interrupciones del receptor inhibidas

SBK: Send BreaK.

Al final un caracter de tiempo de descanso es puesto en la cola y enviado cada vez que el SBK es puesto a uno. Tantas veces como el bit SBK es activado, los caracteres de espera son puestos en la cola y enviados.

0 = El Generador de espera apagado.

1 = El generador de códigos de espera activo.

El Registro de Status de la Comunicación Serie (SCSR).

El SCSR (Serial Communication Status Register) da las entradas para el circuito lógico de interrupción para la generación de las interrupciones del sistema SCI.

$B02E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR
RESET	1	1	0	0	0	0	0	0	

Figura No. 2.35: Representación del Registro de Estatus de Comunicación Serie.

TDRE: Transmit Data Register Empty Flag.

Esta bandera está activa cuando el SCDR está vacío. Borra la bandera TDRE para leer el SCSR con TDRE activo y escribir en SCDR.

0 = SCDR lleno.

1 = SCDR vacío.

TC: Transmit Complete Flag.

Esta bandera se activa cuando el transmisor está ocioso. Borra la bandera TC para leer el SCSR con TC activo y entonces escribir en el SCDR.

0 = Transmisor lleno.

1 = Transmisor libre.

RDRF: Receive Data Register Full Flag.

Esta bandera se activa si un caracter recibido está listo para leerse desde el SCDR. Borra la bandera RDRF al leer SCSR.

0 = SCDR vacío.

1 = SCDR lleno.

IDLE: Idle Line Detect Flag.

Esta bandera se activa cuando la línea receptora está libre. Una vez borrado, se mantendrá siempre así hasta que la línea receptora se haya activado y vuelva a quedar libre. La bandera IDLE es inhibida cuando RWU = 1. IDLE es borrada por la lectura de SCSR con IDLE activa.

0 = La línea Rx está ocupada.

1 = La línea Rx está libre.

OR: Overrun Error Flag.

OR se activa si un nuevo caracter es recibido después de que un caracter previo que fue recibido se ha leído desde SCDR. Se borra por la lectura de SCDR con el OR activo.

0 = No hay sobre escritura.

1 = Una sobreescritura es detectada.

NF: Noise Error Flag.

Se activa si hay ruido y si no está todo correcto. NF se borra con la lectura de SCSR y leyendo SCDR.

0 = Decisión unánime.

1 = Ruido detectado.

FE: Framing Error.

FE se activa cuando un cero es detectado cuando un bit de parada fue esperado. Se borra la bandera FE por la lectura de SCSR con FE activo.

0 = Un bit de parada detectado.

1 = Un cero detectado.

Bit 0: No está implementado. Siempre se lee cero.

El Registro de Baud rate.

Se usa el registro BAUD para seleccionar diferentes velocidades de Tx/Rx para el sistema SCI. Los bits SCP[1:0] funcionan como un preescalar para los bits SCR[2:0]. Sin embargo estos cinco bits proporcionan múltiples velocidades de Tx/Rx para una frecuencia de cristal dada. Normalmente, estos registros se escriben una vez durante la inicialización, ver la Tabla No. 2.10.

Selec. de Preescalar			Div. interna	Frecuencia del cristal en Mhz.		
SCP2	SCP1	SCP0	del reloj por	4.0	4.9152	8.0
0	0	0	1	62500	76800	125000
0	0	1	3	20833	25600	41667
0	1	0	4	15625	19200	31250
0	1	1	13	4800	5907	9600
1	0	0	39	1602	1969	3205
1	0	1	No se usa			
1	1	0	No se usa			
1	1	1	No se usa			

Tabla No. 2.10 Selección del Preescalar del Baud Rate.

BAUD: Baud Rate.

$B02B	TCLR	SCP2	SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD
RESET	0	0	0	0	0	U	U	U	

Figura No. 2.36: Representación del Registro BAUD.

TCLR: Clear Baud Rate Counters (Prueba).

SCP[2:0] Seleccionael preescalar del Baud Rate del SCI.
De acuerdo a la Tabla No.2.10.

SCR[2:0]: Selección del Baud Rate del SCI.
La selección de la recepción y transmisión del bit rate basado en el contenido de la salida desde el preescalar del baud rate. Como se muestra en la Tabla No. 2.11. Los bits preescalares SCP[2:0], determinan el mayor baud rate, y los bits SCR[2:0] seleccionan un submúltiplo binario adicional ($\div 1$, $\div 2$, $\div 4$, ... ,$\div 128$) del mayor baud rate. El resultado de estos dos divisores en serie es el reloj receptor de baud rate 16X. Los bit SCR[2:0] no son afectados por el reset y pueden ser cambiados en cualquier momento, sin embargo no pueden ser cambiados cuando la comunicación con el SCI está en proceso.

SCR[2:0]	Divide el Preescalar por	El mayor Baud Rate (Salida Preescalar de la Tabla No. 2.11)				
		131072	76800	32768	19200	4800
000	1	131072	76800	32768	19200	4800
001	2	65536	38400	16384	9600	2400
010	4	32768	19200	8192	4800	1200
011	8	16384	9600	4096	2400	600
100	16	8192	4800	2048	1200	300
101	32	4096	2400	1024	600	150
110	64	2048	1200	512	300	75
111	128	1024	600	256	150	-

Tabla No. 2.11: Selección de Baud Rate.

Banderas de Estatus e Interrupciones.

El transmisor SCI tiene dos banderas de estatus; las cuales pueden ser leídas por el software para saber la condición correspondiente. Alternativamente, un bit habilita un interrupción lo cual puede activarse para habilitar cada una de estas condiciones de estatus para generar una petición de interrupción cuando la correspondiente condición esté presente. Las banderas de estatus son automaticamente activadas por condiciones lógicas de hardware, pero pueden ser borradas por software, por lo que proveé de un mecanismo de sincronización que permite logicamente saber cuando el software tiene conocimiento de la condición del estatus.

Las banderas TDRE y TC están normalmente activas cuando el transmisor es primeramente habilitado (TE activo en uno). La bandera TDRE indica si existe espacio en la cola de transmisión para cargar otro caracter en el TDR. El bit TIE es la máscara de la interrupción local para TDRE. Donde TIE tiene cero, TDRE puede ser extraída. Cuando TIE y TDRE son unos una petición de interrupción está presente.

La bandera TC indica que el transmisor ha llenado la cola. El bit TCIE es la máscara de interrupción local para TC. Cuando TCIE es cero, TC puede ser extraída; Cuando TCIE y TC son unos, una petición de interrupción está presente.

Las Banderas de Recepción.

El Receptor del SCI contiene cinco banderas de estatus, tres de las cuales pueden generar peticiones de interrupción. Las banderas de estatus están activas por la lógica del SCI en respuesta a una condición específica en el receptor. Estas banderas pueden ser

leídas en cualquier momento por software. Para mayor claridad se desarrollan en el Diagrama de Flujo No. 2.3.

El RDRF se activa cuando un caracter ha sido recibido y transferido al registro paralelo RDR. La bandera OR se activa en lugar de RDRF cuando ocurre un desbordamiento. Un nuevo caracter es leído para ser transferido al RDR después que un caracter previo es removido del RDR.

Las banderas NF y FE dan información adicional acerca del caracter en el RDR, pero no generan requerimientos de interrupción.

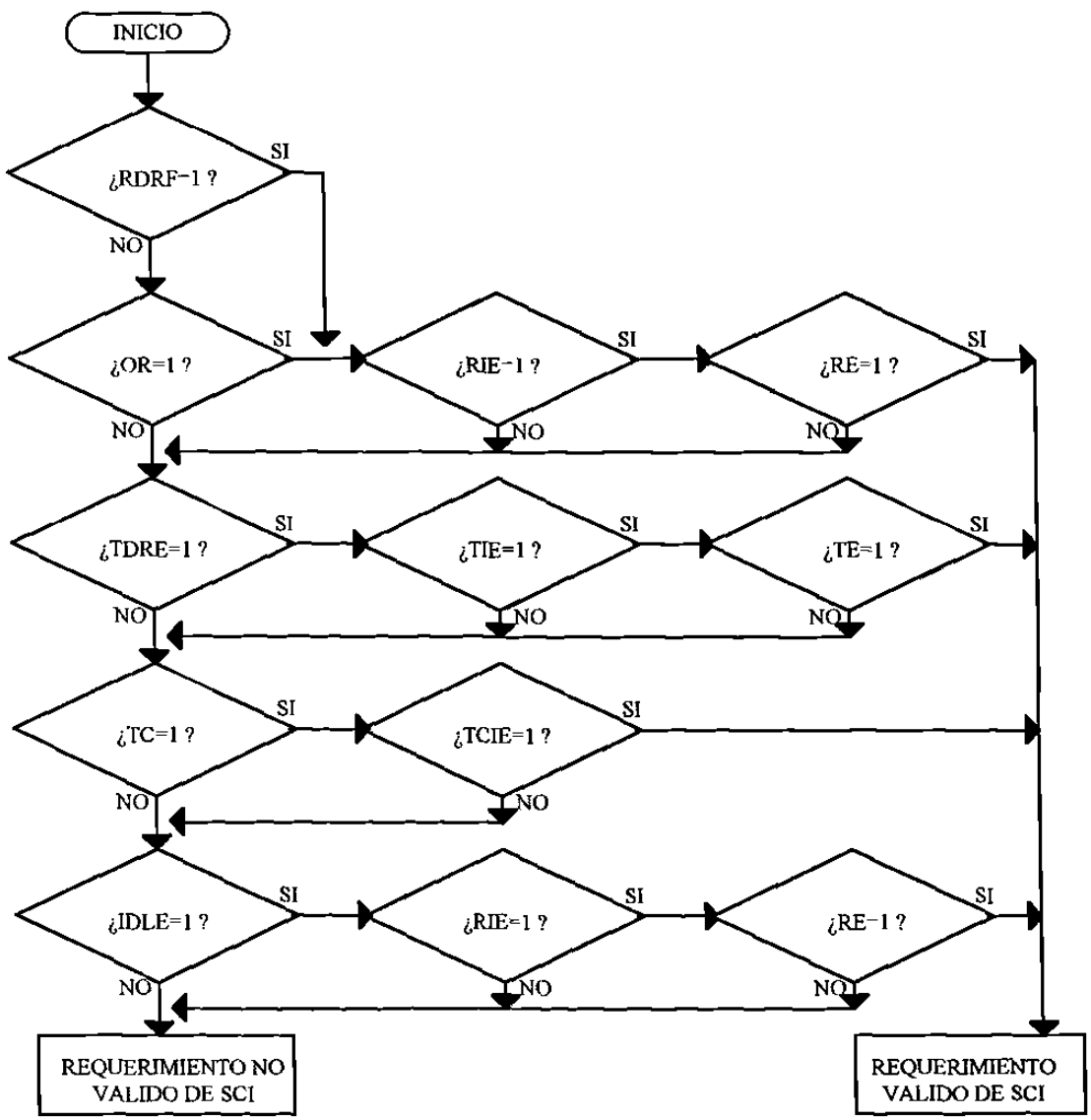


Diagrama No. 2.3: Diagrama de Flujo que Muestra el Origen de las Interrupciones del SCI.

Adaptador de Interfase Periférico (PIA)

El adaptador de interfase periférico MC6821 proporciona medios universales para comunicarse con la familia de microprocesadores de 8 bits como el MC68HC11.

Este dispositivo es capaz de interactuar directamente con el MCU, proporcionando dos buses de datos bidireccionales de 8 bits y cuatro líneas de control. La lógica de interfase externa no es necesaria para la mayoría de los dispositivos periféricos.

La configuración del PIA es programada por el MCU durante la inicialización. Cada una de las líneas de los puertos pueden ser programadas para actuar como entrada o salida, esto permite un alto grado de flexibilidad en la operación global de la interfase.

Características Generales

- Bus de Datos de 8 bits bidireccionales para comunicaciones con el MCU.
- Dos Puertos Bidireccionales de 8 bits (A y B).
- Dos controles de registros programables.
- 4 líneas de entrada de interrupción controladas individualmente utilizadas como controles periféricos de salida.
- Control lógico de entrada salida.
- Alta impedancia three-state y transistor directo al drive de línea de salida.
- Dos drive TTL con capacidad en todos los buffers de los puertos A y B.
- Compatible con TTL.

La configuración del PIA se muestra en la Figura No. 2.37; y a continuación se enumerarán las principales características de cada uno de sus pines.

Señales de Interfase del PIA.

El PIA interfiere al bus del MC68HC11 con un bus de datos bidireccionales de 8 bits, 3 chips selectores de línea, 2 registros selectores, 2 interruptores de demanda, 1 línea de Lectura/Escritura, 1 línea permisoria y un reset. Para asegurar una operación apropiada, el VMA puede ser usado como una parte activa de la decodificación de las direcciones.

Datos Bidireccionales (D_0-D_7):

Los datos bidireccionales permiten la transferencia de datos entre el MCU y el PIA. El bus de datos de salida son dispositivos three-state que permanecen en estado de alta impedancia excepto cuando el MCU ejecuta una operación de lectura. La línea de Lectura/Escritura está en el estado de lectura (alta) cuando el PIA es seleccionado para esta operación.

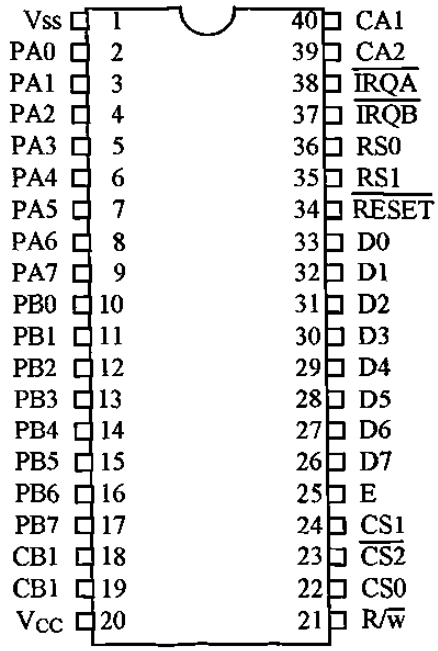


Figura No. 2.37: Estructura Física del MC6821.

Enable (E):

El pulso habilitador E, es la única señal de cronometraje que es proporcionada al PIA. El cronometraje de todas las otras señales es con referencia al inicio y salida de los bordes del pulso E.

Lectura/Escritura (L/E):

Esta señal es generada por el MCU para controlar la dirección de la transferencia de datos en el bus de datos. Un estado bajo en las líneas de Lectura/Escritura del PIA habilita la entrada y los datos son transferidos del MCU al PIA en la señal E. Un alto en las líneas de Lectura/Escritura habilita al PIA para una transferencia de datos del MCU. Los buffers de salida del PIA son habilitados cuando las direcciones son apropiadas y el pulso E está presente.

Reset:

Un cero activa la línea de reset, este estado es usado para poner todos los bits de registro en el PIA a un cero lógico (bajo). Esta línea puede ser usada como un reset de encendido-apagado y un reset maestro durante la operación del sistema.

Líneas Selectoras (CS0, CS1 y CS2):

Estas 3 señales de entrada son usadas para seleccionar el PIA. CS0 y CS1 tienen que ser altas y CS2 debe ser baja para la selección del dispositivo. La transferencia de datos es entonces ejecutada bajo el control de enable y las señales de Lectura/Escritura. Los chips selectores de línea son estables en la duración del pulso E. El dispositivo es deshabilitado cuando una de las líneas selectoras está inactiva.

Selector de Registro (RS0 y RS1):

Las dos líneas selectoras del registro son usadas para seleccionar varios registros interiores del PIA. Estas dos líneas son usadas en conjunción con el control interno de registro para seleccionar un registro particular que será escrito o leído. El registro y las líneas selectoras son estables en la duración del pulso E junto con la duración del ciclo de Lectura/Escritura.

Interruptor Request (IRQA e IRQB):

Un bajo activa el interruptor request (IRQA e IRQB) actúa para interrumpir directamente en cada actividad de eventos prioritarios del MCU. Estas líneas son "drenadores" (el chip no carga al dispositivo). Cada interrupción de demandas de línea tiene dos banderas internas que pueden causar que la interrupción de demanda se vaya a cero. Cada bit bandera es asociado con una interrupción. También, 4 bits interruptores son proporcionados en el PIA, los cuales pueden ser usados para inhibir una interrupción particular de un dispositivo. El interruptor en servicio del MCU puede ser usado por una rutina de software, en una prioridad básica, secuencialmente lee y evalúa los dos registros de control en cada PIA para las bandera que estén colocadas.

Líneas Periféricas de Interfase del PIA.

El PIA proveé dos puertos bidireccionales de 8 bits, 4 líneas de control/interrupción de línea para interactuar con dispositivos periféricos.

Puerto A (PA0-PA7):

Cada una de las líneas de datos puede ser programada para actuar como entrada o salida, en este caso se usó de este puerto, los tres primeros bits de salida y los últimos 5 bits de entrada como lo muestra la Figura No. 2.10. Esto es realizado por un "1" en el bit correspondiente de registro de datos para estas líneas las cuales son salidas. Un "0" en un bit del registro de dirección de datos causa que la línea de datos actúe como una entrada. Durante una operación de lectura, los datos programados en líneas actúan como entradas

apareciendo directamente en la línea del bus de datos del MCU. En modo entrada, la resistencia interna en estas líneas representa un máximo de 1.5 Ω.

Puerto B (PB0-PB7):

Los datos de las líneas del puerto B del PIA pueden ser programados para actuar cada uno como entrada o salida en una manera similar que PA0-PA7; en el proyecto que nos ocupa se programaron todas como salidas, en el caso de la impresora, por este puerto saldrán los datos sistema-impresora. Ellos tienen capacidad three-state permitiendo la entrada de una alta impedancia cuando la línea de datos es usada como entrada. En suma, los datos en las líneas de datos (PB0-PB7) serán leídas pobremente desde estas líneas programadas como salidas cuando los voltajes estén por abajo de 2 volts para un "high" o arriba de 0.8 volts para un "low"; como salidas, estas líneas son compatibles con TTL estandar.

Líneas de entrada (CA0 y CA2):

Las líneas de entrada CA0 y CA2 son líneas de entrada solamente que activan las banderas de interrupción de los registros de control. La transición activa para estas señales son también programadas por los dos registros de control.

Control (CA2):

La línea de control CA2 puede ser programada para actuar como un interruptor de entrada o como un control de salida. Como salida, esta línea es compatible con TTL estandar.

Control (CB2):

Las líneas de control CB2 pueden también ser programadas para actuar como un interruptor de entrada o como un control de salida. Como entrada, esta línea tiene alta impedancia de entrada y es compatible con TTL estandar. Esta línea es programada por el registro de control B.

Controles Internos

Un $\overline{RESET}$ tiene el efecto de poner en cero todos los registros del PIA. Esto colocará al PA_0-PA_7, PB_0-PB_7, CA2 y CB2 como entradas y deshabilitará todos los interruptores. El PIA puede ser configurado durante la ejecución del programa el cual seguirá al reset.

Estas son 6 localizaciones dentro del PIA accesibles al bus de datos del MCU: 2 registros periféricos, 2 registros de dirección de datos y 2 registros de control. La selección de estas posiciones es controlada por las entradas RS0 y RS1 junto con el bit 2 en el registro de control como se muestra en la Tabla No. 2.12.

RS0	RS1	CRA-2	CRB-2	LOCALIZACION DE SELECCION
0	0	1	X	Registro A
0	0	0	X	Registro de Direcciones de Datos A.
0	1	X	X	Control de Registro A
1	0	X	1	Registro B
1	0	X	0	Registro de Direcciones de Datos B
1	0	X	X	Control de Registro B

Tabla No. 2.12: Direcciones Internas del MC6821

Registro de Control CRA y CRB:

Los dos registros de control CRA y CRB permiten al MCU controlar la operación de las 4 líneas de control CA1, CA2, CB0 y CB2. En resumen ellos permiten al MCU habilitar las líneas de interrupción y monitorear el estatus de las banderas de interrupción. Los bits del 0 al 5 de los dos registros pueden ser escritos o leídos directamente por el MCU cuando los selectores apropiados y el registro selector de señales son aplicados. Los bits 6 y 7 de los dos registros son de sólo lectura y son modificados por interruptores internos de las líneas de control CA1, CA2, CB0 y CB2.

Acceso de Dirección al Bit de Control (CRA2 y CRB2):

El bit dos de cada registro de control (CRA y CRB), determinará la selección de cada registro de salida o la correspondiente dirección de datos. Un "1" en el bit dos permite acceso al registro, accesando un "0" causa que el registro de dirección pueda ser direccionado.

Banderas de Interrupcón (CRA6, CRA7, CRB6 y CRB7):

Los 4 bits de las banderas de interrupción son colocados por transiciones activas de señales en los cuatro líneas de control e interrupción donde estas líneas son programadas como entradas. Estos bits pueden ser colocados directamente desde el bus

de datos del MCU y ser reiniciados indirectamente por una operación de lectura de datos en la sección apropiada.

Control de las líneas de control CA2 y CB2, (CRA3, CRA4, CRA5, CRB3, CRB4 y CRB5):

Los bits 3, 4 y 5 de los dos registros de control son usados para controlar las líneas de control CA2 y CB3. Estos bits determinan si el control de líneas será un interruptor de entrada o una señal de control de salida. Si el bit CRA5 (CRB5) es bajo, CA2 (CB2) es una línea interruptora de entrada similar a CA0 (CB1). Cuando CRA5 (CRB5) es alto y CA2 (CB2) se vuelve en señal de entrada y puede ser usado para controlar la transferencia de datos. Cuando esté en el modo de salida CA2 y CB2 tienen ligeras diferencias en las características de carga.

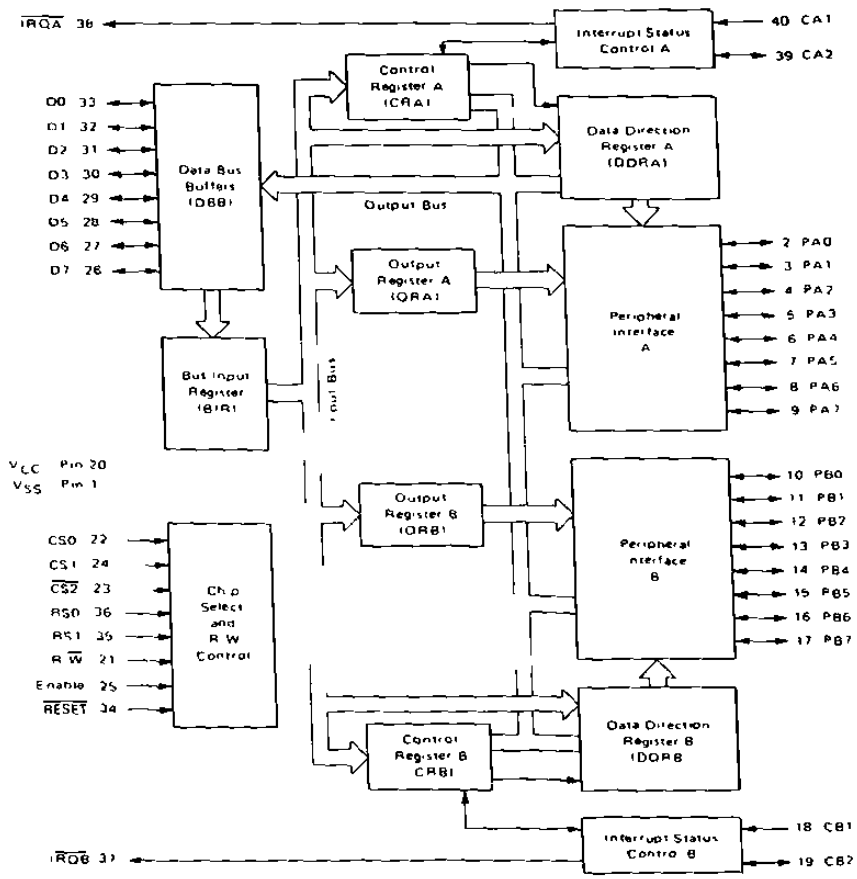


Figura No. 2.38: Diagrama a Bloques de la Estructura del MC6821.

El Control de las Líneas de Interrupción CA1 y CB1 (CRA-0, CRB-0, CRA-1 y CRB-1):

Los dos bits de menor orden de los registros de control son usados para el control de las líneas de entrada CA1 y CB1. Los bits CRA-0 y CRB-0 son usados para habilitar las señales de interrupción del MCU $\overline{IRQA}$ y $\overline{IRQB}$, respectivamente. Los bits CRA-1 y CRB-1 determinan la transición activa de las señales de entrada de CA1 y CB1.

Para un mejor entendimiento del PIA en la Figura No. 2.38, se muestran el diagrama a cuadros de este circuito.

166795

CAPITULO III
SOFTWARE

Las rutinas que se explicarán en este capítulo son las más importantes del sistema y el programa completo se muestra en el Anexo A.

Al inciar el sistema parte de la dirección $FFFE-$FFFF donde se guarda el vector de reset, aquí se guarda la dirección donde inicia el programa principal en nuestro caso es $C000.

El sistema al dar reset por alguna de sus formas primeramente configura el sistema, inicializando puertos e inicializando dispositivos y verifica el estado de la llave como se mencionó en capítulos anteriores, que contiene tres posiciones, con lo cual lo divide a su vez en tres partes o subsistemas; el Diagrama de Flujo No. 3.1 , el cual muestra estas tres secciones, el denominado subsistema normal donde se encuentra la posición para que el usuario (trabajador) cheque su entrada o salida y se mantendrá ahí mientras la llave no cambie de posición y de la misma forma se mantendrá en los otros subsistemas; el subsistema de consulta, donde el supervisor pondrá a tiempo el reloj, podrá saber cuanto espacio de memoria de datos tiene disponible y obtendrá listado de trabajadores en la base de datos y listado de los datos capturados; la tercer opción denominada subsistema de comunicación es para transmitir y recibir información desde y hacia la computadora que manipulará los datos.

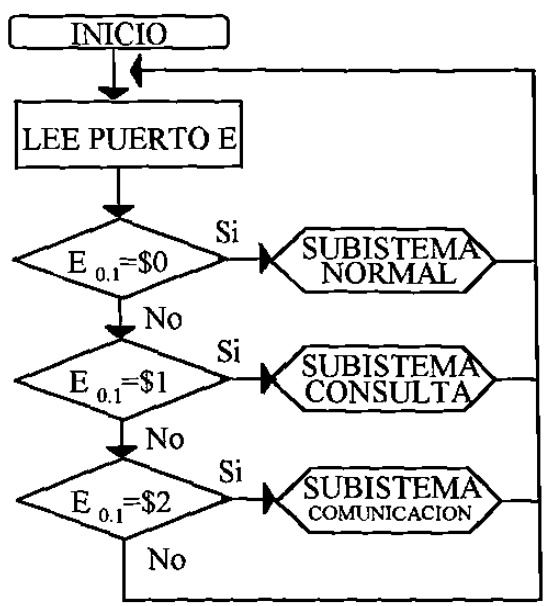


Diagrama de Flujo No. 3.1: Interacción con la Llave Supervisora.

Los subsistemas los denominaré E0, E1 y E2 en referencia al puerto y al dígito leído de él, por lo que en lo sucesivo lo haré con esta referencia.

Las rutinas implementadas para E0 se muestran en el Diagrama de FlujoNo. 3.2.

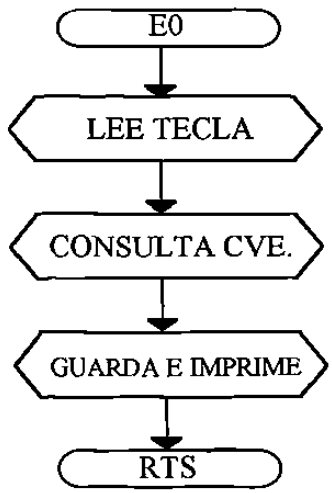


Diagrama de Flujo No. 3.2: Rutinas del Subsistema Normal (E0).

Donde **LEE TECLA** es una rutina donde el sistema esperará que se digite una tecla; al atrapar el sistema la primer tecla, a su vez retendrá las horas y minutos en una variable de paso, al teclear cada dígito se mostrará en el display el dato que se ha introducido al terminar de teclear la clave en 4 dígitos, se pasará a la rutina de **CONSULTA CLAVE**, en ella se buscará en la base de datos la clave correspondiente y si lo encuentra la mostrará en el diaplay, en esta fase será necesario confirmar si la clave tecleada es la correcta oprimiendo la tecla de *enter*, se trasladará a la rutina de **GUARDA E IMPRIME** en esta rutina verificará si existe memoria suficiente para guardar datos de ser así guardará clave y horas y minutos en la memoria de datos y además en forma opcional imprimirá los datos en el formato siguiente:

HH:MM DD/MM/AA
CCCC NOMBRE DEL EMPLEADO

Se dice en forma opcional por que de no encontrarse la impresora, la tomará como si existiera, es decir, la impresora es transparente para el sistema en este punto. Si el sistema verifica que no contiene memoria suficiente para guardar datos enviará un mensaje

de error al display y no permitiendo introducir datos hasta que el sitema libere a la memoria de su carga.

De no encontrar la clave desplegará el mensaje **"esta clave no existe"** Durará unos segundos y retornará a la opción inicial de esperar la primera tecla nuevamente.

En caso de que el trabajador se entere que tecleó en forma equivocada su clave, bastará con digitar la tecla de escape retornando a la condición inicial de **"espera la primera tecla"**.

Las rutinas que componen el susbsistema de consulta (E1) se muestran en el Diagrama de Flujo No. 3.3.

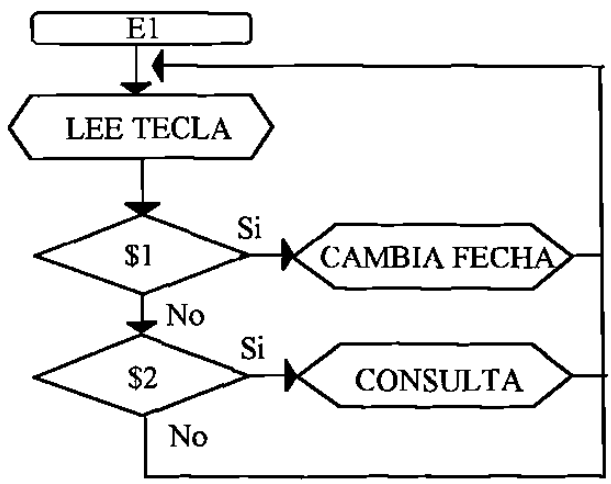


Diagrama de Flujo No. 3.3: Rutinas del Subsistema de Consulta (E1).

El menú del subsistema de consulta se muestra en la Figura No. 1.6, contiene dos opciones, por default es la uno, bastará con confirmar tecleando *enter*, o seleccionando con los números 1 ó 2 y tecleando *enter*, podrá notar al hacer esto que el cursor "<" se moverá de un renglón a otro para indicarle que opción está seleccionando.

Si selecciona la primera opción **"(1) Poner a Tiempo"**, aparecerá un nuevo menú la cual contiene dos opciones como lo muestra la Figura No.1.7, tecleando 1 y *enter* entrará directamente a cambiar la fecha y 2 y *enter* para cambiar la hora, para esto se tecleará lo que se haya seleccionado en forma completa y en el formato indicado y bajo la responsabilidad del supervisor.

LECTURA DE TECLADO

Para leer el teclado, el cual se encuentra en el puerto A con 3 bits de entrada y cuatro bits de salida; por los bits de salida se envían un tren de ceros, ya que estarán

amarrados a unos lógicos a través de una resistencia, y se hace un barrido a las entradas para detectar que switch del teclado ha sido cerrado (tecla oprimida); el diagrama de flujo de esta rutina se muestra en el Diagrama de Flujo No. 3.4.

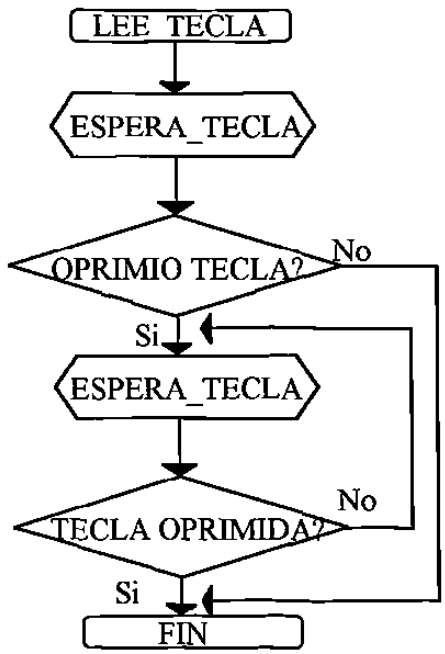


Diagrama de Flujo No. 3.4: Rutina de Lectura de Teclado.

SE ENVIA				SE LEE			TECLA
A_6	A_5	A_4	A_3	A_2	A_1	A_0	OPRIMIDA
1	1	1	0	1	1	0	3
1	1	1	0	1	0	1	2
1	0	1	0	0	1	1	1
1	1	0	1	1	1	0	6
1	1	0	1	1	0	1	5
1	1	0	1	0	1	1	4
1	0	1	1	1	1	0	9
1	0	1	1	1	0	1	8
1	0	1	1	0	1	1	7
0	1	1	1	1	1	0	ENTER
0	1	1	1	1	0	1	0
0	1	1	1	0	1	1	ESCAPE

Tabla No. 3.1: Lectura del Teclado.

Se lee primero el teclado si hay una tecla oprimida se espera ha que se suelte, de tal forma que no se repita el caracter ha que se hace mención; la tecla que se oprime responde a la Tabla No. 3.1.

EL DISPLAY

El display es un dispositivo que contiene su propio microprocesador para controlar sus funciones de tal forma que cada una está representada por un comando como se muestra en la Tabla No. 3.2.

INSTRUCCION	Código										DESCRIPCION
	RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
Borrar Display	0	0	0	0	0	0	0	0	0	1	Borra display y regresa el cursor a la posición inicial.
Regresa al Inicio	0	0	0	0	0	0	0	0	1	*	Regresa el cursor al inicio.
Modo SET	0	0	0	0	0	0	0	1	I/D	S	Activa el Cursos y Especifica lo que se indica.
Control Display ON/OFF	0	0	0	0	0	0	1	D	C	B	Activa ON/OFF parámetros del Display.
Corrimiento del Display y Cursor.	0	0	0	0	0	1	S/C	R/L	*	*	Mueve Cursor y Corre el Display Cambiando el Contenido de la DD RAM.
Función SET	0	0	0	0	1	DL	N	F	*	*	Activa la Interfase de Longitud de Dato, No. de Línea y Fonts.
Activa Dirección CG RAM	0	0	0	1	A_{CG}						Activa la Dirección de CG RAM
Activa Dirección de DD RAM	0	0	1	A_{DD}							Activa la Dirección de DD RAM
Escribe Datos en el CG o DD RAM	1	0	Escribe Datos								Escribe Datos en la DD RAM o en la CG RAM.

I/D=1: Incrementa en 1 I/D=0 Decrementa en 1. S=1: Acompaña Corrimeinto al Display S/C=1: Recorre el Dispaly S/C=0: Mueve el Cursor R/L=1: Recorre a la Derecha R/L=0: a la Izquierda. DL=1: 8 bits DL=0: 4 bits N=1: 2 Líneas N=0: 1 Línea F=1: 5x10 puntos F=0: 5x7 puntos	DD RAM: RAM de Despliegue de Datos. CG RAM: RAM Generadora de Caracteres. A_{CG}: Direcciones de la CG RAM A_{DD}: Direcciones de la DD RAM corresponde al cursor.

Tabla No. 3.2: Comandos del Display.

En este proyecto se configuró el display a caracteres de 5x7 puntos, recorre el cursor, con corrimiento de la DD RAM a la derecha, con interfase de 8 bits y como el display es de 2 líneas se colocó N=1. Bastará seguir el diagrama mostrado en la Diagrama de Flujo No 3.5 para inicializar el display.

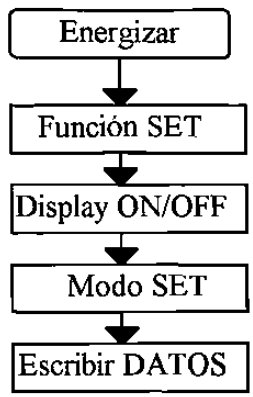


Diagrama de Flujo No. 3.5: Inicialización del Display.

Para desplegar información en el dispaly se guardó en un buffer que se reservó para ello en las direcciones $1B0 a $1FF de la RAM, esto permite guardar la hora cambiante a cada segundo en el momento preciso indicado por la interrupción. El despliegue se hizo atendiendo a las direcciones indicadas por el hardware de acuerdo al Diagrama de Flujo No. 3.6.

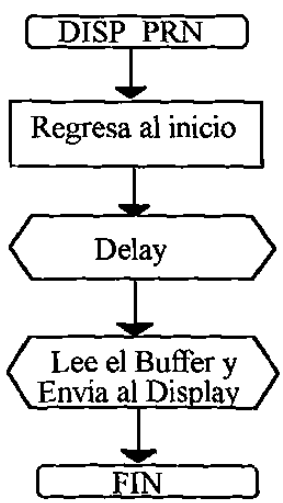


Diagrama de Flujo No. 3.6: Rutina que Despliega la Información Contenida en el Buffer al Display.

La rutina denominada *delay* es con el fin de estabilizar el bus interno del display y es de algunos milisegundos; la rutina que lee el buffer del display y lo envía a este depués de haber colocado el cursor en el inicio, aunque se puede desplegar directamente la información, es decir sin usar el buffer, se eligió el uso de éste ya que la interrupción se hacía más lenta y generaba errores visuales en las décimas de segundo (no errores reales).

LA IMPRESORA.

A la impresora se le asignó también un buffer con el fin de optimizar el tiempo ya que la velocidad de escritura en la impresora es más lenta que la escritura en la RAM, y la posición de los caracteres es más sencillo en el buffer que en la impresora; se tienen dos rutinas de impresión; una donde la impresora es transparente para el dispositivo y otra donde se verifica la presencia y buen funcionamiento de ésta. Ambas son muy similares y sólo difieren en la parte donde se verifica el estatus de la impresora, en el Diagrama de Flujo No. 3.7. se muestra esta última opción.

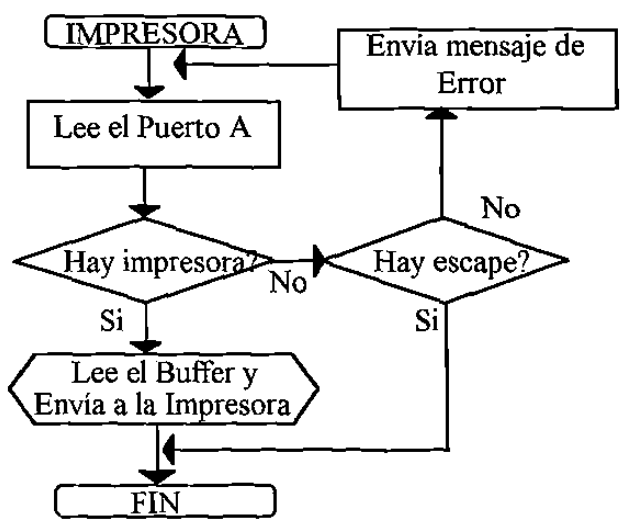


Diagrama de Flujo No. 3.7: Rutina que Imprime la Información Contenida en el Buffer de la Impresora.

LA INTERRUPCION DE TIEMPO.

Se crearon variables de tiempo, como lo son **CENTSEG, SEGUNDOS, MINUTOS, HORAS, DIA, MES Y AÑO** para guardar temporalmente estas referencias de tiempo; para los días del mes se creó una tabla que informa del número de días que tiene cada uno de ellos, esta tabla cambia en el segundo mes de acuerdo si es biciesto o

no. Para generar la interrupción se colocó un $80 en TMSK2 y TFLG2 cuya dirección es la $B024 y $B025.

$B024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
$B025	TOF	RTIF	PAOVF	PAIF	0	0	0	0	TFLG2

Figura No. 3.1: Representación del Registro TMSK2 del Timer.

Al generarse la interrupción incrementa las centésimas de segundo en CENTSEG y cuando sean 60 borra esta variable e incrementa SEGUNDOS de igual forma se hace con el resto de las variables; cada vez que se genera la interrupción envía al buffer del display la fecha y la hora; también verifica si es año bisciesto para modificar el número de días del mes de Febrero. La rutina de interrupción se muestra en el Diagrama de Flujo No. 3.8.

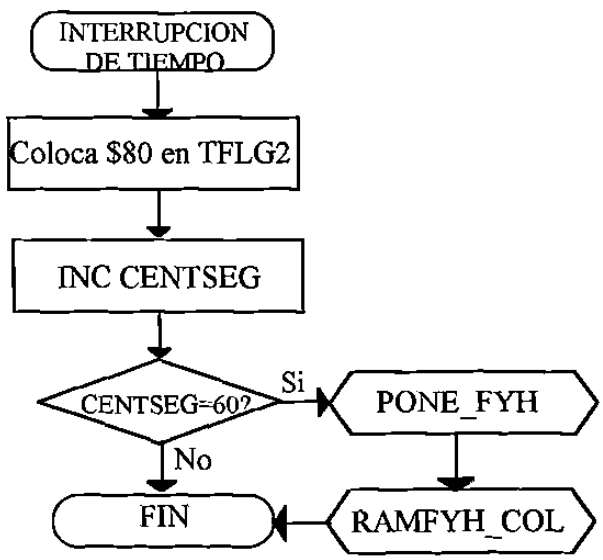


Diagrama de Flujo No. 3.8: Rutina de la Interrupción de Tiempo.

La rutina PONE_FYH es donde se incrementan los segundos, minutos, horas, días, meses y años, y se colocan en las variables correspondientes. La rutina RAMFYH_COL coloca la fecha y hora en el buffer del display en la posición correspondiente.

ANEXO A

RUTINAS DEL PROGRAMA PARA EL MICROCONTROLADOR MC68HC11

```
                        ORG $C000

HORAS           EQU $0100
MINUTOS         EQU $0101
SEGUNDOS        EQU $0102
CENTESEG        EQU $0103
AYO             EQU $0104
MES             EQU $0105
DIA             EQU $0106
CVE_2           EQU $0107
CVE_1           EQU $0108
CONT_TECLA      EQU $0109
PASO1           EQU $010A
HOR_CHECAD      EQU $010B       *DOS LOCALIDADES $10B Y $10C
ESTAT_PRN       EQU $010D       *BYTE DE ESTATUS DE LA IMPRESORA
UMC             EQU $10E        *BYTE DE UNIDADES DE MILLAR Y CENTENAS
DU              EQU $10F        *DECENAS Y UNIDADES PARA
                                *CANTIDAD DE RAM DE DATOS INI_RAM_NOM
                EQU $4000
FIN_RAM_NOM     EQU $4FFF

DAT_INI_RAM     EQU $3000       *INICIA RAM DE DATOS
DAT_FIN_RAM     EQU $3FF0       *FINALIZA RAM DE DATOS
DIR_ULT_CVE     EQU $3FFE       *CONTIENE LA PRIMERA DIRECCION
                                *LIBRE PARA DATOS
TECLA           EQU $B000
************************ DIRECCIONES DE LA IMPRESORA (PIA1)
CONTROL_PRN     EQU $CC00
DATOS_PRN       EQU $CC02
HOR_B_PRN       EQU $0130
FEC_B_PRN       EQU $0143
CVE_B_PRN       EQU $014D
NOM_B_PRN       EQU $0152
FIN1_B_PRN      EQU $016A       *FIN DEL BUFFER +1 (2 LINEAS)
FIN_BU_PRN      EQU $0188       * FIN DEL BUFFER +1  (3 LINEAS)
I_R2B_PRN       EQU $016A       * FIN+1 SEGUNDA LINEA DEL BUF
                                * TAMBIEN INICIO DE LA TERCERA FGB_L1P
                EQU $14B        *FIN DE CARGA DE LINEA UNO
FGB_L2P         EQU $168        *FIN DE CARGA DE LINEA DOS
FGB_L3P         EQU $185        *FIN DE CARGA DE LINEA TRES

************************ DATOS DEL DISPALY
RDDC            EQU $d000
RDDD            EQU $d001
************************DATOS DEL BUFFERS DEL DISPALY
B_DIS1_RAM      EQU $01B0       *BASE del BUFFER DISPLAY
DIS_REN2        EQU $01D8       *INICIO DEL RENG. 2 EN BUFFER
INI_DIS_NOM     EQU $01DD       *INICIA EL NOMBRE EN EL BUF. DISP.
DIS_B_HOR       EQU $01D0       *INICIA LA HORA EN EL BUF. DISP.
F_BUF_DISP      EQU $0200       *FIN DEL BUFFER DEL DISPLAY + 1
PON_TIEMPO      EQU $01B8       *INICIO DEL MENSAJE RENG. 1 EN BUF.DISP.
POS_CONS        EQU $01E2       *INICIO DEL MSJE. CONSULTAR
FL_PT           EQU $01CC       *POS. DE LA FLECHA EN EL BUFFER
FL_CONSU        EQU $01F3       *POS. DE FLECHA DE CONSULTAR
```

```
M21_RAM     EQU $01BA     *DESPLIEGA:(1)RAM UTIL
FL_M21R     EQU $01CB     *FLECHA DE RAM UTIL
M22_IN      EQU $01D9     *IMPRIME:(2) NOMBRES
FL_M22_IN   EQU $01F1     *FLECHA IMPRIME NOMBRES
FL_M23_IN   EQU $01FF     *FLECHA IMPRIME DATOS
TEC_M11     EQU $01B9     *POS. DEL BUF DE < 1 <-- TECLEE
FYH_1E0     EQU $01E0     *POS. DONDE INICIA CAMBIO DE FECHA
FYH_1F8     EQU $01F8     *POS. INICIA CAMBIO DE HORA
```

CONFIGURACION DE PUERTOS Y DISPLAY

```
            JSR CONF_DIS
            ldaa #$7e
            staa $00d0
            ldx #reloj
            stx $00d1
            ldaa #$80
            staa $B024
            LDAA #00          * INIC. REG. DE CONTROL DEL PIA
            STAA $CC01
            STAA $CC03
INIT_PIA    LDX  #$0704       *INICIALIZA PIA
            STX  CONTROL_PRN
            LDX  #$0FF04
            STX  DATOS_PRN
            LDAA #$07
            STAA CONTROL_PRN
```

PROGRAMA PRINCIPAL

```
            JSR BORRA_DIS
            LDX #$FF
            JSR ESPERA1
AQUI_DIAG
FUE_E0
            JSR E0
            JMP AQUI_DIAG
FUE_E1
            JSR E1
            JMP AQUI_DIAG
FUE_E2
            JMP AQUI_DIAG
FIN_PAL     JMP AQUI_DIAG
```

FEC_PON_RAM: PONE LA FECHA EN LA RAM DE DATOS CADA QUE SEAN LAS 00:00:00

```
                    LDX DIR_ULT_CVE
                    CPX #DAT_INI_RAM
                    BEQ PONE_FEC_RAM
                    LDX HORAS
                    CPX #$00
                    BNE FFEC_RAM
                    LDAA SEGUNDOS
                    CMPA #5
                    BGT FFEC_RAM
                    LDX DIR_ULT_CVE
                    DEX
                    DEX
                    DEX
                    DEX
                    LDAA ,X
                    CMPA #'/'
                    BEQ FFEC_RAM
PONE_FEC_RAM
                    LDX DIR_ULT_CVE
                    LDAA #'/'
                    STAA ,X
                    INX
                    LDAA DIA
                    STAA ,X
                    INX
                    LDAA MES
                    STAA ,X
                    INX
                    LDAA AYO
                    STAA ,X
                    INX
                    STX DIR_ULT_CVE
FFEC_RAM            RTS
```

E1: RUTINA QUE ENTRA A LA MODALIDAD E1 EN RAM

```
E1
                    LDAA #$20
                    JSR COL_BL_DIS
                    JSR DISP_PRN        *DESPLIEGA BUFFER DEL DISPLAY
                    JSR E1_MENU1
FIN_E1              RTS
```

E1_MENU1: RUTINA QUE COLOCA EN EL DISPLAY EL MENU1

```
E1_MENU1
                LDAA #$1
                STAA PASO1
                LDY #PON_TIEMPO  * Y  POSICION DEL NOM EN BUF.
                LDX #M_1         * X  POS. DEL MSJE EN RAM
                JSR B_MSJE_DIS   * RUTINA PARA COLOCAR MSJE EN BUF.
                LDY #POS_CONS
                LDX #M_2
                JSR B_MSJE_DIS
E1_OT_TEC       JSR DISP_PRN
                JSR ESPERA_TECLA
                CMPA #$1
                BNE FUE_CONS
                STAA PASO1
                LDAA #$20
                STAA FL_CONSU
                LDAA #'<'
                STAA FL_PT
                JMP E1_OT_TEC
FUE_CONS        CMPA #$2
                BNE E1_ESC
                STAA PASO1
                LDAA #$20
                STAA FL_PT
                LDAA #'<'
                STAA FL_CONSU
                JMP E1_OT_TEC
E1_ESC          CMPA #$B
                BEQ F_E1_M1
E1_FUE_CR       CMPA #$A
                BNE E1_OT_TEC
                LDAB PASO1
                CMPB #$1
                BEQ PON_T
                CMPB #$2
                BEQ CONSULTAR
PON_T
                LDY #TEC_M11
                LDX #HOR_M11
                JSR B_MSJE_DIS
                LDY #DIS_REN2
                LDX #C_MSJE
                JSR B_MSJE_DIS
                LDY #FYH_1E0
                LDX #FECHA_
                JSR B_MSJE_DIS
                JSR DISP_PRN     *DESPLIEGA BUF. DEL DISPLAY
                JSR CO_FYH_DI    *COLOCA FECHA Y HORA EN DISP.
```

```
                JMP F_E1_M1
CONSULTAR
                LDY #M21_RAM
                LDX #M_DES_2_1
                JSR B_MSJE_DIS
                LDY #M22_IN
                LDX #M_IMP_2_2
                JSR B_MSJE_DIS
                JSR MENU2_E1
F_E1_M1         RTS
```

CO_FYH_DI: COLOCA FECHA Y HORA EN DISPLA, ADEMAS ACTUALIZA FECHA Y HORA

```
CO_FYH_DI
                LDAA #$1            *COLOCA EL DEFAULT=1
                STAA PASO1
FYH_OTTEC       JSR ESPERA_TECLA    *REGRESA EN A LA TECLA OPRIMIDA
                CMPA #$1
                BNE FYH_2
                STAA PASO1
                LDX #FECHA_
                LDY #FYH_1E0
                JSR B_MSJE_DIS
                JMP FYH_OTTEC
FYH_2           CMPA #$2
                BNE FYH_ESC
                STAA PASO1
                LDX #HORA_
                LDY #FYH_1E0
                JSR B_MSJE_DIS
                JMP FYH_OTTEC
FYH_ESC         CMPA #$B
                BEQ FCO_FYH_DI
FYH_CR          CMPA #$A
                BNE CO_FYH_DI
                LDAA PASO1
                CMPA #$1
                BNE FUE2_FYH
                JSR FEC_RYD         *FECHA A LA RAM DEL DISPLAY FUE2_FYH
                CMPA #$2
                BNE CO_FYH_DI
                JSR HOR_RYD         *HORA A LA RAM DEL DISPLAY
                JMP FYH_OTTEC
FCO_FYH_DI      RTS
```

FEC_RYD: RUTINA QUE LEE EL TECLADO Y COLOCA EN LA RAM 2o. RENGLON LA FECHA Y LA ACTUALIZA

```
FEC_RYD
                LDY #DIS_REN2
                JSR LEE_2T
                STAA DIA
                LDAA #'/'
                STAA ,Y
                INY
                JSR LEE_2T
                STAA MES
                LDAA #'/'
                STAA ,Y
                INY
                JSR LEE_2T
                STAA AYO
FFEC_RYD        RTS
```

LEE_2T: LEE DOS TECLAS LAS COLOCA EN EL DISPALY Y LAS TRANSFORMA A BCD EN UN BYTE

```
LEE_2T
                PSHY
TEC1_           JSR ESPERA_TECLA
                CMPA #$9
                BGT TEC1_
                TAB
                ROLB
                ROLB
                ROLB
                ROLB
                ANDB #$F0
                PULY
                ADDA #$30
                STAA $120
                STAA ,Y
                INY
                PSHB
                PSHY
TEC2_           JSR ESPERA_TECLA
                CMPA #$9
                BGT TEC2_
                TAB
                PULY
                ADDA #$30
                STAA ,Y
                INY
                PULA
```

```
                ABA
F_LEE_2T        RTS
```

HOR_RYD: COLOCA EN EL 2o. RENGLON LA NUEVA HORA Y LA ACTUALIZA EN LA RAM DE HORA

```
HOR_RYD
                LDY #FYH_1F8
                JSR LEE_2T
                STAA HORAS
                LDAA #':'
                STAA ,Y
                INY
                JSR LEE_2T
                STAA MINUTOS
                LDAA #':'
                STAA ,Y
                INY
                JSR LEE_2T
                STAA SEGUNDOS
FHOR_RYD        RTS
```

MENU2_E1: RUTINA QUE SELECCIONA VER RAM UTIL, IMPRIME NOMBRES O IMPRIME DATOS

```
MENU2_E1
M2_OT_TEC
                JSR ESPERA_TECLA
                CMPA #$1
                BNE FUE_R_UTIL
                STAA PASO1
                LDAA #$20
                STAA FL_M22_IN
                STAA FL_M23_IN
                LDAA #'<'
                STAA FL_M21R
                JMP M2_OT_TEC
FUE_R_UTIL      CMPA #$2
                BNE IM_NOMBRES
                STAA PASO1
                LDAA #$20
                STAA FL_M21R
                STAA FL_M23_IN
                LDAA #'<'
                STAA FL_M22_IN
                JMP M2_OT_TEC
IM_NOMBRES      CMPA #$3
                BNE M2_ESCAPE
                STAA PASO1
                LDAA #$20
                STAA FL_M21R
```

```
                STAA FL_M22_IN
                LDAA #'<'
                STAA FL_M23_IN
                JMP M2_OT_TEC
M2_ESCAPE       CMPA #$B
                BEQ F_E1_MENU
M2_FUE_CR       CMPA #$A
                BNE M2_OT_TEC
                LDAB PASO1
                CMPB #$1
                BEQ VE_R_U
                CMPB #$2
                BEQ IMP_NOMS
                CMPB #$3
                BEQ IMP_DAT
VE_R_U
                JSR VE_RAM_D
                JMP F_E1_MENU
IMP_NOMS
                JSR CHECA_PRN
                LDAA ESTAT_PRN
                CMPA #$1
                BEQ NOEX1_PRN     ****
                JSR IMP_CABEZA
                LDX #INI_RAM_NOM
LE_OTNOM_R      PSHX
                LDX #FGB_L1P      *FIN DEL RENGLON1
                STX BU_PRN_FIN    *HSTA AQUI CARGA
                LDY #HOR_B_PRN    *INICIO DE CARGA
                LDAA #$20         *CARACTER A CARGAR (" ")
                JSR COL_CAR_BU    *COLOCA EN EL BUFFER
                LDY #HOR_B_PRN
                PULX
                CPX #FIN_RAM_NOM
                BEQ F_E1_MENU
                LDAA ,X
                CMPA #$FF
                BEQ F_E1_MENU
                JSR IMP_CLAVE     *CLAVE EN EL BUFFER L1
                INX
                INY
                LDAA ,X
                JSR IMP_CLAVE     *CLAVE EN EL BUF. L1
                INY
                LDAA #$20
                STAA ,Y
                LDAB #22
OT_NOM_CAR      INX
                INY
                LDAA ,X
                STAA ,Y
                DECB
                BNE OT_NOM_CAR
                PSHX
```

```
                    LDX #CVE_B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
                    PULX
                    INX
NOEX1_PRN           JMP LE_OTNOM_R    *LEE OTRO NOMBRE DE RAM
IMP_DAT
                    JSR PRN_DAT
F_E1_MENU           RTS PRN_NOM
```

VE_RAM_D: CUENTA LA RAM DE DATOS DISPONIBLE Y LA COLOCA EN EL DISPLAY

```
VE_RAM_D
                    CLR PASO1
                    CLR UMC
                    CLR DU
                    LDX DIR_ULT_CVE
                    LDD #DAT_FIN_RAM
                    SUBD DIR_ULT_CVE
                    PSHB
                    PSHA
                    PULX
                    INX
OT_CUENTA
                    DEX
                    BEQ MUEST_RAM_D
                    ldaa DU
                    ADDA #$1
                    TAB
                    DAA
                    STAA DU
                    CMPB #$9A
                    BNE OT_CUENTA
                    CLRA
                    STAA DU
                    LDAA UMC
                    ADDA #$1
                    TAB
                    DAA
                    STAA UMC
                    CMPB #$9A
                    BNE OT_CUENTA
                    CLRA
                    STAA UMC
                    LDAA PASO1
                    ADDA #$1
                    DAA
                    STAA PASO1
                    JMP OT_CUENTA
```

```
MUEST_RAM_D     LDAA #$20          *ES UN BLANCO
                JSR COL_BL_DIS     *BLANCOS AL 2DO. RENG. DEL DISP
                LDAA PASO1
                ANDA #$F
                ADDA #$30
                LDY #DIS_REN2      *POSICION DEL CARACTER
                STAA ,Y
                INY
                LDAB UMC
                JSR DER_ROTA
                ADDB #$30
                STAB ,Y
                INY
                LDAA UMC
                ANDA #$0F
                ADDA #$30
                STAA ,Y
                INY
                LDAB DU
                JSR DER_ROTA
                ADDB #$30
                STAB ,Y
                INY
                LDAA DU
                ANDA #$0F
                ADDA #$30
                STAA ,Y
                INY
                INY
                LDX #CAR_DIS
                JSR B_MSJE_DIS
RA_TEC          JSR ESPERA_TECLA
                CMPA #$B
                BNE RA_TEC
FVE_RAMD        RTS
```

PRN_DAT: IMPRIME DATOS DE LA RAM DE DATOS

```
PRN_DAT
                JSR CHECA_PRN
                LDAA ESTAT_PRN
                CMPA #$1
                BEQ NOEX3_PRN      ******
                JSR CABEZA1_PRN
                LDX #DAT_INI_RAM   *CARGA EN X EL INICIO DE LA RAM DE DATOS
                LDY #HOR_B_PRN     *INICIO DE CARGA PARA BUF PRN
LE2_OTNOM_R
                JSR VER_DIAG       *RUTINA PARA IMPRIMIR FECHA, REGRESA
                                   * LA POS. DEL DATO SIGUIENTE H:M
                PSHX
```

```
                LDX #FGB_L1P      *FIN DEL RENGLON1
                STX BU_PRN_FIN    *HSTA AQUI CARGA
                LDY #HOR_B_PRN    *POS. DEL INICIO DEL RENG. 1 DE BUF PRN
                LDAA #$20         *CARACTER A CARGAR (" ")
                JSR COL_CAR_BU    *COLOCA EN EL BUFFER
                LDY #HOR_B_PRN    *POS. DEL INICIO DEL RENG. 1 DE BUF PRN
                PULX
                CPX DIR_ULT_CVE   *ES LA ULTIMA DIRECCION DE DATOS?
                BGE FPRN_DAT      *SI ES ASI, SALTA AL FINAL
                                  *SI NO SALTA A LA RUTINA SIG. JSR
BYTE_A_2ASC                       *TRANSFORMA BCD A DOS ASCII
                LDAA #':'         *COLOCA LOS DOS PUNTOS
                STAA ,Y
                INY
                JSR BYTE_A_2ASC   *TRANSFORMA BCD A DOS ASCII
                LDAA #' '         *COLOCA UN ESPACIO EN BLANCO
                STAA ,Y
                INY
                JSR BYTE_A_2ASC   *LEE E IMPRIME PRIMER BYTE DE CVE
                JSR BYTE_A_2ASC   *LEE E IMPRIME SEGUNDO BYTE DE CVE
                PSHX              *GUARDA EL APUNTADOR DE DATOS
                LDX #CVE_B_PRN    * POS. DE FIN DE BUF PRN
                STX BU_PRN_FIN    * PARA IMPRIMIR
                LDX #HOR_B_PRN    *INICIO DE IMPRESION
                JSR IMP_BU_PRN    *IMPRIME
                PULX              *EXTRAE EL APUNTADOR DE DATOS
                JMP LE2_OTNOM_R   *LEE OTRO NOMBRE DE RAM
NOEX3_PRN
                                  ********
FPRN_DAT        RTS
```

BYTE_A_2ASC: LEE UN BYTE Y LO TRANSFORMA EN ASCII COLOCANDOLO EN EL BUFFER DE LA IMPRESORA Y LO IMPRIME

```
BYTE_A_2ASC
                                  *SI NO ES FIN DE DATOS
                LDAB ,X           *CARCA UN BYTE
                PSHX              *GUARDA POS. DEL DATO
                JSR DER_ROTA      *LO ROTA  B<-- DATO ROTADO (DECENAS)
                LDX #TABLA        *APUNTA A LA TABLA
                JSR RCOLOCA       *COLOCA VALOR EN BUFFER DEL DISP.
                INY               *INCREMENTA POS. DEL BUF. PRN
                PULX              *EXTRAE POS. DEL DATO
                LDAB ,X           *EXTRAE DATO
                PSHX
                LDX #TABLA        *EXTRAE EL EQUIVALENTE
                JSR RCOLOCA       *LO COLOCA EN EL BUF. PRN
                PULX
```

```
                INY             *INCREMENTA POS. DEL BUF. PRN
                INX             *INCREMENTA POS. DEL DATO
FBYTE_2ASC      RTS
```

VER_DIAG: VERIFICA EN LA RAM DE DATOS SI LA POSICION ES "/" DE SER ASI, IMPRIME FECHA.

```
VER_DIAG
                LDAA ,X
                CMPA #'/'       *ES '/' EL DATO DIRECCIONADO EN X?
                BNE F_VER_DIAG  * SI NO, FINALIZA
                INX             *SI SI, INCREMENTA A POS. DEL DIA
                LDY #FGB_L1P    *ULTIMA POSICION DEL RENGLON 1 BUF PRN
                STY BU_PRN_FIN  *LO ALMACENA
                LDY #HOR_B_PRN  *POS. DEL INICIO DEL RENG. 1 DE BUF PRN
                LDAA #'*'       *COLOCA '*' EN BUF PRN
                JSR COL_CAR_BU  *LO EJECUTA
                LDY #CVE_B_PRN  *CARGA INICIO DE RENG 2 DE BUF PRN
                LDY #FGB_L2P    *ULTIMA POSICION DEL RENGLON 2 BUF PRN
                STY BU_PRN_FIN  *LO ALMACENA
                LDY #CVE_B_PRN  *POS. DEL INICIO DEL RENG. 2 DE BUF PRN
                LDAA #' '       *COLOCA ' ' EN BUF PRN
                JSR COL_CAR_BU  *LO EJECUTA
                LDY #CVE_B_PRN  *Y <-- POS. DE BUF. DE PRN  RENG. 2
                JSR BYTE_A_2ASC *LEE DIA Y LO TRTANSFORMA A ASCII
                LDAA #'/'       *CARGA '/'
                STAA ,Y         *LO COLOCA EN BUF PRN
                INY             *INCREMENTA POS. DEL BUF PRN
                JSR BYTE_A_2ASC *LEE MES Y LO TRANSFORMA A ASCII
                LDAA #'/'       *CARGA '/'
                STAA ,Y         *LO COLOCA EN BUF PRN
                INY             *INCREMENTA POS. DEL BUF PRN
                JSR BYTE_A_2ASC *LEE A¥O Y LO TRANSFORMA A ASCII
                LDY #FGB_L3P    *ULTIMA POSICION DEL RENGLON 1 BUF PRN
                STY BU_PRN_FIN  *LO ALMACENA
                LDY #FIN1_B_PRN *POS. DEL INICIO DEL RENG. 1 DE BUF PRN
                LDAA #'*'       *COLOCA '*' EN BUF PRN
                JSR COL_CAR_BU  *LO EJECUTA
                PSHX
                LDX #FIN_BU_PRN * POS. DE FIN DE BUF PRN
                STX BU_PRN_FIN  * PARA IMPRIMIR
                LDX #HOR_B_PRN  *INICIO DE IMPRESION
                JSR IMP_BU_PRN  *IMPRIME
                PULX
F_VER_DIAG      RTS
```

CABEZA1_PRN: IMPRIME EL ENCABEZADO DE REPORTE DE DATOS DE LA RAM DE DATOS

```
CABEZA1_PRN
                JSR IMP_1CABEZA
                LDX #I6_CABEZA  *CVE. NOMBRES EN RENG. 1
```

```
                    LDY #HOR_B_PRN
                    JSR B_MSJE_DIS
                    LDX #FGB_L2P      *FIN DEL RENGLON2
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #CVE_B_PRN    *INICIO DE CARGA
                    LDAA #'='         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
F2_IMP_CAB          RTS
```

IMP_CABEZA: IMPRIME EL ENCABEZADO DE REPORTE DE NOMBRES DELA RAM DE DATOS

```
IMP_CABEZA
                    JSR IMP_2CABEZA
                    LDX #I5_CABEZA    *CVE. NOMBRES EN RENG. 1
                    LDY #HOR_B_PRN
                    JSR B_MSJE_DIS
                    LDX #FGB_L2P      *FIN DEL RENGLON2
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #CVE_B_PRN    *INICIO DE CARGA
                    LDAA #'='         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
F_IMP_CABEZA  RTS
```

IMP_1CABEZA: IMPRIME PARTE DEL ENCABEZADO DE REPORTE DE NOMBRES DE LA RAM DE DATOS

```
IMP_1CABEZA
                    JSR CHECA_PRN
                    LDAA ESTAT_PRN
                    CMPA #$00
                    BNE F_IMP_CABEZA
                    LDX #I1_CABEZA    *SEP SEIT DGIT ITCV
                    LDY #HOR_B_PRN
                    JSR B_MSJE_DIS
                    LDX #I21_CABEZ    *PERSONAL DADO DE ALTA
                    LDY #CVE_B_PRN
                    JSR B_MSJE_DIS
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
                    LDX #FGB_L1P      *FIN DEL RENGLON2
```

```
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #HOR_B_PRN    *INICIO DE CARGA
                    LDAA #$20         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    JSR FYH_B_PRN     *FECHA Y HORA EN BUF. PRN
                    LDX #FGB_L2P      *FIN DEL RENGLON2
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #CVE_B_PRN    *INICIO DE CARGA
                    LDAA #'='         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
F_IM1_CAB   RTS
```

IMP_2CABEZA: IMPRIME PARTE DEL ENCABEZADO DE REPORTE DE NOMBRES DE LA RAM DE DATOS

```
IMP_2CABEZA
                    JSR CHECA_PRN
                    LDAA ESTAT_PRN
                    CMPA #$00
                    BNE F_IMP_CABEZA
                    LDX #I1_CABEZA    *SEP SEIT DGIT ITCV
                    LDY #HOR_B_PRN
                    JSR B_MSJE_DIS
                    LDX #I2_CABEZA
                    LDY #CVE_B_PRN
                    JSR B_MSJE_DIS
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
                    LDX #FGB_L1P      *FIN DEL RENGLON2
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #HOR_B_PRN    *INICIO DE CARGA
                    LDAA #$20         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    JSR FYH_B_PRN     *FECHA Y HORA EN BUF. PRN
                    LDX #FGB_L2P      *FIN DEL RENGLON2
                    STX BU_PRN_FIN    *HSTA AQUI CARGA
                    LDY #CVE_B_PRN    *INICIO DE CARGA
                    LDAA #'='         *CARACTER A CARGAR
                    JSR COL_CAR_BU    * COLOCA EN EL BUFFER
                    LDX #I_R2B_PRN    * POS. DE FIN DE BUF PRN
                    STX BU_PRN_FIN    * PARA IMPRIMIR
                    LDX #HOR_B_PRN    *INICIO DE IMPRESION
                    JSR IMP_BU_PRN    *IMPRIME
F_IM2_CAB   RTS
```

IMP_CLAVE: IMPRIME LA CLAVE EN LE REPORTE DE NOMBRES DE LA RAM DE DATOS

```
IMP_CLAVE
                RORA
                RORA
                RORA
                RORA
                ANDA #$0F
                ADDA #$30
                STAA ,Y
                INY
                LDAA ,X
                ANDA #$0F
                ADDA #$30
                STAA ,Y
IMP_F_CVE       RTS
```

E0: RUTINA QUE ENTRA A LA MODALIDAD E0 EN RAM

```
E0
                JSR DISP_PRN        *DESPLIEGA BUFFER DEL DISPLAY
                JSR TECLADO         *LEE DEL TECLADO
FIN_E0          RTS
```

TECLADO: LEE LA TECLA, LA ROTA, LA DESPLIEGA Y LA ALMACENA EN RAM

```
TECLADO
                LDAA #$20
                JSR COL_BL_DIS      *COLOCA BLANCOS EN EL RENG. 2
                JSR DISP_PRN        *BUFFER DEL DISPLAY AL DISPLAY
                LDAA #$00           *CONTADOR DE TECLAS
                STAA CONT_TECLA
TECLA_NUEVA     JSR ESPERA_TECLA    *REGRESA EN A LA TECLA OPRIMIDA
                CMPA #$FF           *SI NO ES TECLA REGRESA
                BEQ FFIN_TECLA
                CMPA #$A            * ES <CR>
                BEQ ES_CAR_RET
                CMPA #$B            * ES <ESC>
                BEQ VE_ESCAPE
                PSHA
                JSR DISP_TECLA      * FUE CARACTER
                PULA
                LDAB CONT_TECLA     *AQUI YA SABEMOS QUE FUE CARACTER
                INCB
                STAB CONT_TECLA
PRIMERA         CMPB #$01
```

```
                BNE SEGUNDA
                ROLA
                ROLA
                ROLA
                ROLA
                ANDA #$F0
                STAA CVE_2
                JMP TECLA_NUEVA
SEGUNDA         CMPB #$02
                BNE TERCERA
                LDAB CVE_2
                ABA
                STAA CVE_2
                JMP TECLA_NUEVA
VE_ESCAPE       JMP ES_ESCAPE
TERCERA         CMPB #$3
                BNE CUARTA
                ROLA
                ROLA
                ROLA
                ROLA
                ANDA #$F0
                STAA CVE_1
                JMP TECLA_NUEVA
CUARTA          CMPB #$4
                BNE TECLA_NUEVA
                LDAB CVE_1
                ABA
                STAA CVE_1
                JSR BUSCA_CVE       *SI ENCUENTRA IMPRIME
                JMP TECLA_NUEVA *REGRESA Y BORRA RENG. 2
FFIN_TECLA      JMP FIN_TECLA
ES_CAR_RET      LDAB CONT_TECLA * < ENTER >
                CMPB #$4            *B = NUMERO DE TECLAS
                BNE TECLA_NUEVA

                JSR GUARDA_DATOS
                LDX #CVE_B_PRN      * FEC_B_PRN = POS. DEL BUF DE PRN
                LDY #DIS_REN2       * DIS_B_HOR = POS. DEL BUF DE DISP
                LDAB #27
CONT3_L_G       LDAA 0,Y
                STAA 0,X
                INX
                INY
                DECB
                BNE CONT3_L_G       *CONTINUA LEYENDO Y GRABANDO
                JSR CHECA_PRN
                LDAA ESTAT_PRN
                CMPA #$1
                BEQ NOEX2_PRN       *AQUI FALTA MSJE ERR_PRN
                LDX #FIN_BU_PRN     * POS. DE FIN DE BUF PRN
                STX BU_PRN_FIN      * PARA IMPRIMIR
                LDX #HOR_B_PRN      *INICIO DE IMPRESION
                JSR IMP_BU_PRN
```

```
NOEX2_PRN       JMP TECLADO
ES_ESCAPE
                LDAA #$20
                JSR COL_BL_DIS
                JMP TECLADO
FIN_TECLA       RTS
```

COL_BL_DIS: COLOCA BLANCOS EN EL SEGUNDO RENGLON DEL BUFFER DEL DISPLAY

```
COL_BL_DIS
                LDY #DIS_REN2
OT_BCO          STAA ,Y
                INY
                CPY #F_BUF_DISP
                BNE OT_BCO
F_CBD           RTS
```

GUARDA_DATOS: GUARDA HORA, MINUTOS, Y CLAVE DEL EMPLEADO

```
GUARDA_DATOS
                LDX DIR_ULT_CVE
                LDY HOR_CHECAD
                STY ,X
                INX
                INX
                LDAA CVE_2
                STAA ,X
                INX
                LDAA CVE_1
                STAA ,X
                INX
                STX DIR_ULT_CVE
F_GUARDA_DAT  RTS
```

BUSCA_CVE: VERIFICA LA EXISTENCIA DE LA CLVE GUARDADA EN LA RAM DENOMINADA CVE_1 Y CVE_2

```
BUSCA_CVE
                LDX #INI_RAM_NOM
OTRA_CVE        LDAA ,X
                CMPA #$FF
                BEQ CVE_NO_EXIS
                CMPA CVE_2
                BEQ FUE_CV2         *AQUI ESTA BUSCANDO C4C3
```

```
                    LDAB #24
                    ABX
                    CPX #FIN_RAM_NOM
                    BLT OTRA_CVE
CVE_NO_EXIS
                    LDY #INI_DIS_NOM  * A IY POSICION DEL NOM EN BUF.
                    LDX #ERNO_CVE     *A IX POS. DEL MSJE EN RAM
                    JSR B_MSJE_DIS    * RUTINA PARA COLOCAR MSJE EN BUF.
                    JMP DEJA_VER
FUE_CV2
                    INX
                    LDAA ,X
                    CMPA CVE_1        *BUSCA C2C1
                    BEQ FUE_CV1
                    LDAB #23
                    ABX
                    JMP OTRA_CVE
FUE_CV1             INX               *POCISION DEL NOMBRE EN RAM
                    LDY #INI_DIS_NOM  *POC. DEL NOM. EN BUFFER DEL DISP
                     LDAB #22
KARGA_CARAC         LDAA ,X
                    STAA ,Y
                    INX
                    INY
                    DECB
                    BNE KARGA_CARAC
                    JSR DISP_PRN      *DESPLIEGA BUFFER DEL DISPLAY
                    JMP FIN_BUSCA_CVE
DEJA_VER
                    JSR DISP_PRN
                    LDY #$5
                    JSR ESPERAY
                    LDAA #$20
                    JSR COL_BL_DIS    *COLOCA BLANCOS EN EL RENG. 2
                    LDAA #$00         *CONTADOR DE TECLAS
                    STAA CONT_TECLA
FIN_BUSCA_CVE RTS
```

B_MSJE_DIS: COLOCA EL MENSAJE EN EL BUFFER DEL DISPLAY INICIANDO EN IX Y FINALIZA CON $

```
B_MSJE_DIS
                    LDAA $0,X         *LEE MSJE DE LA POS. IX
                    CMPA #$24
                    BEQ #FB_MSJE_D
                    STAA ,Y
                    INX
                    INY
                    JMP B_MSJE_DIS
FB_MSJE_D           RTS
```

ESPERA_TECLA: VERIFICA EL TIEMPO DE ESPERA PARA QUE TECLEE LA TECLA SIGUIENTE

```
ESPERA_TECLA
LEE_OTRAT
                JSR FEC_PON_RAM
                LDX HORA            *COLOCA EN UN ANTES DE INICIAR
                STX HOR_CHECAD      * A OPRIMIR TECLA
                JSR FYH_B_PRN       *HORA Y FECHA EN BUF PRN RENG. 1
                JSR DISP_PRN
                JSR LEE_TECLA
                CMPA #$FF
                BNE YA_NO_TECLA
                JMP FIN_E_TEC
YA_NO_TECLA
                PSHA                * GUARDA LA TECLA OPRIMIDA
OTRA_NO_TEC
                JSR DISP_PRN
                JSR LEE_TECLA       *VERIFICA QUE SE HAYA DEJADO
                CMPA #$FF
                BNE OTRA_NO_TEC
                PULA                *
FIN_E_TEC       RTS
```

FYH_B_PRN: COLOCA EN EL BUFFER DEL DISPALY EN EL RENGLON 1 LA FECHA Y LA HORA EN LOS EXTREMOS

```
FYH_B_PRN
                LDX #HOR_B_PRN      * HOR_B_PRN = POS. DEL BUF DE PRN
                LDY #DIS_B_HOR      * DIS_B_HOR = POS. DEL BUF DE DISP
                LDAB #$5
CONT_L_G        LDAA 0,Y
                STAA 0,X
                INX
                INY
                DECB
                BNE CONT_L_G        *CONTINUA LEYENDO Y GRABANDO
                LDX #FEC_B_PRN      * FEC_B_PRN = POS. DEL BUF DE PRN
                LDY #B_DIS1_RAM     * DIS_B_HOR = POS. DEL BUF DE DISP
                LDAB #$8
CONT2_L_G       LDAA 0,Y
                STAA 0,X
                INX
                INY
                DECB
                BNE CONT2_L_G       *CONTINUA LEYENDO Y GRABANDO
F_FYH_BPRN      RTS
```

LEE_TECLA: LEE LA TECLA OPRIMIDA

```
LEE_TECLA
                LDAA #$70
                STAA TECLA
                LDAA TECLA
                ANDA #$07
                CMPA #$6
                BEQ LEYO_3
                CMPA #$5
                BEQ LEYO_2
                CMPA #$3
                BEQ LEYO_1
                LDAA #$68
                STAA TECLA
                LDAA TECLA
                ANDA #$07
                CMPA #$6
                BEQ LEYO_6
                CMPA #$5
                BEQ LEYO_5
                CMPA #$3
                BEQ LEYO_4
                LDAA #$58
                STAA TECLA
                LDAA TECLA
                ANDA #$07
                CMPA #$6
                BEQ LEYO_9
                CMPA #$5
                BEQ LEYO_8
                CMPA #$3
                BEQ LEYO_7
                LDAA #$38
                STAA TECLA
                LDAA TECLA
                ANDA #$7
                CMPA #$6
                BEQ LEYO_CR
                CMPA #$5
                BEQ LEYO_0
                CMPA #$3
                BEQ LEYO_ESC
                LDAA #$FF
                JMP FLEE_TECLA
LEYO_3          LDAA #$3
                JMP FLEE_TECLA
LEYO_2          LDAA #$2
                JMP FLEE_TECLA
LEYO_1          LDAA #$1
```

```
                JMP FLEE_TECLA
LEYO_6          LDAA #$6
                JMP FLEE_TECLA
LEYO_5          LDAA #$5
                JMP FLEE_TECLA
LEYO_4          LDAA #$4
                JMP FLEE_TECLA
LEYO_9          LDAA #$9
                JMP FLEE_TECLA
LEYO_8          LDAA #$8
                JMP FLEE_TECLA
LEYO_7          LDAA #$7
                JMP FLEE_TECLA
LEYO_CR         LDAA #$A
                JMP FLEE_TECLA
LEYO_0          LDAA #$0
                JMP FLEE_TECLA
LEYO_ESC        LDAA #$B
FLEE_TECLA
TER_TECLA       RTS
```

DISP_TECLA: DESPLIEGA LA TECLA EN LA POS. DEL DISPLAY

```
DISP_TECLA
                ADDA #$30
                LDX #DIS_REN2
                LDAB CONT_TECLA
                ABX
                STAA ,X
F_DISP_TEC      RTS
```

CONF_DIS: CONFIGURA EL DISPLAY

```
CONF_DIS
                LDAA #$038      *FUNCION SET b7=0, b6=0, b5=1
                STAA RDDC       *b4=DL=1= 8 bits
                                *b3= N=1= display de 2 lineas
                                *b2= F=0= 5x7 dots c/caracter
                                *b1=b0= no inporta
                LDX $FFF
                JSR ESPERA1
                                *AQUI ABAJO TENIA $E
                LDAA #$0E       *DISP ON/OFF b7..b4=0, b3=1
                STAA RDDC       *b2=b1=display on=cursor on=1
                                *b0=B=0= blink off
                LDX #$FF
                JSR ESPERA1
                LDAA #$06 *4    *MODO SET b7..b3=0=rs=r/w, b2=1
                STAA RDDC       *b1=1=I/D=decrementa -1
                                *b0=0=S=acompaña corrimiento de disp.
                LDX #$1FF
```

```
                JSR ESPERA1
                RTS
```

ESPERAY: Ciclo de Espera

```
ESPERAY
                LDX #$FFFF      *PERMITE VER UNOS SEG. EL NOMBRE
                JSR ESPERA1
                DEY
                BNE ESPERAY
F_ESPERAY       RTS
```

ESPERA1: ciclo de espera

```
ESPERA1
                                *APROX 4 MSEG POR CICLO.
                DEX
                NOP
                BNE ESPERA1
ESP1_FIN        RTS
```

BORRA_DIS: BORRA EL DISPLAY

```
BORRA_DIS
                LDAA #$01       *CLEAR DISPALY rs=r/w=0 b7..b1=0
                STAA RDDC
                LDX #$FF
                JSR ESPERA1     *VERIFICA SI BSY FLAG ES CERO
FIN_BD          RTS
```

COL_CAR_BU: COLOCA CARACTER EN EL BUFFER DONDE SE INDICA

```
COL_CAR_BU
CAR_OT          STAA ,Y
                INY
                CPY BU_PRN_FIN
                BNE CAR_OT
CAR_F_BU        RTS
```

IMP_BU_PRN: IMPRIME EL BUFFER DE LA IMPRESORA

```
IMP_BU_PRN

IMP_OTRO        LDAA ,X
                PSHX
                STAA DATOS_PRN
                JSR COLOKA_PRN
                PULX
                INX
                CPX BU_PRN_FIN    *ULIMO CARACTER A IMPRIMIR
                BNE  IMP_OTRO
F_IMP_BU_PRN    RTS
```

CHECA_PRN: VERIFICA SI EXISTE IMPRESORA, O TIENE ALGUN ERROR

```
CHECA_PRN
                LDAA #$00
                STAA ESTAT_PRN   *BORRA EL ESTATUS DE LA IMPRESORA
                LDAA CONTROL_PRN
                ANDA #$C0
                CMPA #$C0
                BNE CHEC_F_PRN
                LDAA #$01        *ESTATUS ERROR 1 DE IMPRESORA
                STAA ESTAT_PRN
                JMP CHEC_F_PRN
                LDX #ERR_PRN
                LDY #DIS_REN2
                JSR B_MSJE_DIS
                JSR DISP_PRN     *DELAY PARA VER LA IMPRESORA
                LDY #$FF
                JSR ESPERAY
CHEC_F_PRN  RTS
```

COLOKA_PRN: COLOCA EL CARACTER EN LA IMPRESORA EJECUTANDO LOS COMANDOS DE CONTROL CORRESPONDIENTES

```
COLOKA_PRN
                LDAA #$06
                STAA CONTROL_PRN
                LDAA #$07
                STAA CONTROL_PRN
                JSR PRN_BSY
F_COL_PRN       RTS
```

PRN_BSY: VERIFICA EL BUSY DE LA IMPRESORA

```
PRN_BSY
                    LDAA CONTROL_PRN
                    ANDA #$C0
                    CMPA #$40
                    BNE PRN_BSY
F_PRN_BSY RTS
```

reloj: Rutina de la Interrupción de Tiempo

```
reloj
                    ldaa #$80
                    staa $B025
                    ldaa CENTESEG
                    inca
                    staa CENTESEG
                    cmpa #$10
                    bne SAL
                    JSR PONE_FYH
                    JSR RAMFYH_COL
SAL                 rti
```

PONE_FYH: COLOCA EN LA MEMORIA FECHA Y HORA

```
PONE_FYH
                    clra
                    staa CENTESEG
                    ldaa SEGUNDOS
                    ADDA #$1
                    DAA             *AJUSTA A DECIMAL
                    STAA SEGUNDOS   *GUARDA SEGUNDOS
                    CMPA #$60
                    BNE F_PONE_FYH
                    CLRA            *BORRA SEGUNDOS
                    STAA SEGUNDOS
                    LDAA MINUTOS
                    ADDA #$1        *INCREMENTA MINUTOS
                    DAA             *AJUSTA A DECIMAL
                    STAA MINUTOS
                    CMPA #$60
                    BNE F_PONE_FYH  *SI YA SON 60 MINUTOS
                    CLRA            * MINUTOS  A CEROS
                    STAA MINUTOS
                    LDAA HORAS
                    ADDA #$1        *INCREMENTA HORAS
                    DAA             *AJUSTA A DECIMAL
                    STAA HORAS
                    CMPA #$24
                    BNE F_PONE_FYH  *SON 24 HORAS
```

```
                CLRA            *BORRA HORAS
                STAA HORAS
                LDAA DIA
                ADDA #$1        *INCREMENTA EL DIA
                DAA             *AJUSTA A DECIMAL
                STAA DIA        *GUARDA DIA
                LDAB MES        *A B EL MES
                CMPB #$10
                BLT MENOR_OCT   * EN REG B DEBE IR EL MES
                LDAA #$A
                ANDB #$0F
                ABA
                TAB
                LDAA DIA
                JMP MAYOR_SEP
MENOR_OCT
MAYOR_SEP       LDX #TDIA_MES   *CARGA EL APUNTADOR DE TDIA_MES
                ABX             * APUNTADOR + MES A IX
                LDAB ,X         *A B EL DIA DEL MES
                INCB
                CBA             *ES EL DIA INCREMENTADO
                BNE F_PONE_FYH
                LDAA MES
                INCA
                DAA
                STAA MES
                LDAB #$1        *PONE DIA PRIMERO EN EL NUEVO MES
                STAB DIA
                CMPA #$13
                BNE F_PONE_FYH
                LDAA #$1
                STAA MES        *COLOCA PRIMER MES DEL AÑO
                LDAA AYO
                INCA
                DAA
                STAA AYO        *INCREMENTA AÑO
*               JSR CHECA_BI    *VERIFICA SI ES BISCIESTO
F_PONE_FYH      RTS
```

CHECA_BI: VERIFICA SI EL MES ES FEB Y SI EL AÑO ES BISCIESTO

```
CHECA_BI
                LDX #XYEAR_BI
                LDAA AYO
                LDY #25
LEE_XYEAR       LDAB ,X
                CBA
                BEQ ABI_SI
                INX
                DEY
                BNE LEE_XYEAR
                LDAA #$28
```

```
                JMP ABI_NO
ABI_SI          LDAA #$29
ABI_NO          LDX #TDIA_MES
                INX
                INX
                STAA ,X
F_CHECA_BI      RTS
```

RAMFYH_COL: COLOCA LA FECHA Y LA HORA EN EL FUBER DEL DISPLAY

```
RAMFYH_COL
                LDY #B_DIS1_RAM
                LDAB DIA
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA      *COLOCA VALOR EN BUFFER DEL DISP. INY
                LDAB DIA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAA #'/'
                STAA ,Y
                INY
                LDAB MES
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAB MES
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAA #'/'
                STAA ,Y
                INY
                LDAB AYO
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDX #TABLA
                LDAB AYO
                JSR RCOLOCA
                ldY  #$1D0
                LDAB HORAS
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAB HORAS
                LDX #TABLA
                JSR RCOLOCA
```

```
                INY
                LDAA #':'
                STAA ,Y
                INY
                LDAB MINUTOS
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAB MINUTOS
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDAA #':'
                STAA ,Y
                INY
                LDAB SEGUNDOS
                JSR DER_ROTA
                LDX #TABLA
                JSR RCOLOCA
                INY
                LDX #TABLA
                LDAB SEGUNDOS
                JSR RCOLOCA
F_RAM_FYH       RTS
```

RCOLOCA: HACE UN AND CON B TOMA EL VALOR CORRESPONDIENTE Y LO COLOCA EN EL BUFFER DEL DISPLAY

```
RCOLOCA
                ANDB #$0F
                abx
                ldaa 0,x
                staa ,Y
RF_COLCOA       RTS
```

DISP_PRN: COLOCA LA FECHA Y LA HORA EN EL DISPLAY

```
DISP_PRN
                LDAA #$02
                STAA RDDC
                LDX #$FF
                JSR ESPERA1
                LDY #B_DIS1_RAM
OT1_DPRN        LDAA ,Y
                JSR CHAR_COL
                INY
                CPY #F_BUF_DISP
                BNE OT1_DPRN
F_DPRN_R1Y2 RTS
```

CHAR_COL: COLOCA EL CARACTER QUE VIENE EN EL REGISTRO A

```
CHAR_COL
                STAA RDDD
                LDX #$F
                JSR ESPERA1
F_CHAR_COL      RTS
```

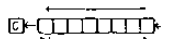
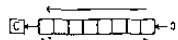
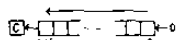
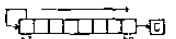
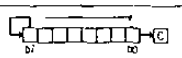
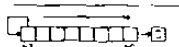
DER_ROTA: ROTA CUATRO VECES EL REGISTRO B A LA DERECHA

```
DER_ROTA
                RORB
                RORB
                RORB
                RORB
                ANDB #$0F
F_DER_ROTA      RTS

*TABLA
TABLA           fcb $30,$31     * 0,1
                fcb $32,$33     * 2,3
                fcb $34,$35     * 4,5
                fcb $36,$37             * 6,7
                fcb $38,$39             * 8,9

*TABLA DE A¥OS BISCIESTOS
XYEAR_BI
FCB $00,$04,$08,$12,$16,$20,$24,$28,$32,$36 FCB $40,$44,$48,$52,$56,$60,$64,$68,$72,$76 FCB
$80,$84,$88,$92,$96

CAR_DIS         FCC "<-- Caracteres Disponibles$"
C_MSJE          FCC "DD/MM/AA      ITCV DGIT SEP      HH:MM:SS$"
FECHA_          FCC "< Cambiar la fecha     $"
HORA_           FCC "       Cambia la hora >$"
HOR_M11         FCC "< 1 <-- Teclee --> 2 >$"
ERNO_CVE        FCC "<-- ESTA CLAVE NO EXISTE$"
ERR_PRN         FCC "01 LA IMPRESORA MUESTRA UN ERROR$"
CVE_TECLEA      FCC "TECLEE SU CLAVE$"
M_1             FCC "  (1) Poner a Tiempo<    $"
M_2             FCC "(2) Consultar$"
M_IMP_2_2       FCC "Imprimir: 2.(2) Nombres   2.(3) Datos$"
M_DES_2_1       FCC "VER: (1)Ram Util <$"
M_1_1           FCC "1.(1) HORA$"
M_1_2           FCC "1.(2) FECHA$"
I1_CABEZA       FCC "    SEP SEIT DEGIT ITCV    $"
I2_CABEZA       FCC "   PERSONAL DADO DE ALTA   $"
I21_CABEZ       FCC "  PERSONAL QUE HA CHECADO  $"
I3_CABEZA       FCC "DD/MM/AA           HH:MM:SS$"
I4_CABEZA       FCC "=========================$"
I5_CABEZA       FCC "CVE. NOMBRES           $"
I6_CABEZA       FCC "CLAVE HH:MM            $"
```

```
                ORG $100
HORA            FCB $21,$50,$59,$09     * HH:MM:SS:00
FECHA           FCB $94,$12,$03           * aa/mm/dd
                FCB $00,$00,$00,$00,$00,$00
                ORG $120
                FCB $FF,$FF,$FF,$FF,$FF,$FF,$FF
                FCB $00,00,00,00,00,00,00

                ORG $130
L1_PRN          FCC "HH:MM              DD/MM/AA"
                FCB 13,$A
L2_PRN          FCC "CCCC 1234567890123456789012"
                FCB 13,$A
L3_PRN          FCC "--------------------------"
                FCB 13,$A,13
BU_PRN_FIN      FCB $00
                ORG $190
TDIA_MES        FCB $00
                FCB $31             * ENERO
                FCB $28             * FEBRERO
                FCB $31             * MARZO
                FCB $30             * ABRIL
                FCB $31             * MAYO
                FCB $30             * JUNIO
                FCB $31             * JULIO
                FCB $31             * AGOSTO
                FCB $30             * SEPTIEMBRE
                FCB $31             * OCTUBRE
                FCB $30             * NOVIEMBRE
                FCB $31             * DICIEMBRE
                ORG $1A0
TBI1            FCB $00
                FCB $A              * ENERO
                FCB $14             * FEBRERO
                FCB $1E             * MARZO
                FCB $28             * ABRIL
                FCB $32             * MAYO
                FCB $3C             * JUNIO
                FCB $46             * JULIO
                FCB $50             * AGOSTO
                FCB $5A             * SEPTIEMBRE
                ORG $1B0
                FCC "DD/MM/AA     ITCV DGIT SEP     HH:MM:SS"
FCC "CCCC 1234567890123456789012          "
                END
```

ANEXO B

SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Operand	Cycles	Condition Codes S	X	H	I	N	Z	V	C
ABA	Add Accumulators	A + B ⇒ A	INH	1B		2			↕		↕	↕	↕	↕
ABX	Add B to X	IX + (00 B) ⇒ IX	INH	3A	—	3								
ABY	Add B to Y	IY + (00 B) ⇒ IY	INH	18 3A	—	4	—							
ADCA (opr)	Add with Carry to A	A + M + C ⇒ A	A IMM	89	ii	2			↕		↕	↕	↕	↕
			A DIR	99	dd	3								
			A EXT	B9	hh ll	4								
			A IND,X	A9	ff	4								
			A IND,Y	18 A9	ff	5								
ADCB (opr)	Add with Carry to B	B + M + C ⇒ B	B IMM	C9	ii	2			↕		↕	↕	↕	↕
			B DIR	D9	dd	3								
			B EXT	F9	hh ll	4								
			B IND,X	E9	ff	4								
			B IND,Y	18 E9	ff	5								
ADDA (opr)	Add Memory to A	A + M ⇒ A	A IMM	8B	ii	2			↕		↕	↕	↕	↕
			A DIR	9B	dd	3								
			A EXT	BB	hh ll	4								
			A IND,X	AB	ff	4								
			A IND,Y	18 AB	ff	5								
ADDB (opr)	Add Memory to B	B + M ⇒ B	B IMM	CB	ii	2			↕		↕	↕	↕	↕
			B DIR	DB	dd	3								
			B EXT	FB	hh ll	4								
			B IND X	EB	ff	4								
			B IND Y	18 EB	ff	5								
ADDD (opr)	Add 16-Bit to D	D + (M M + 1) ⇒ D	IMM	C3	jj kk	4		—			↕	↕	↕	↕
			DIR	D3	dd	5								
			EXT	F3	hh ll	6								
			IND X	E3	ff	6								
			IND,Y	18 E3	ff	7								
ANDA (opr)	AND A with Memory	A • M ⇒ A	A IMM	84	ii	2					↕	↕	0	
			A DIR	94	dd	3								
			A EXT	B4	hh ll	4								
			A IND X	A4	ff	4								
			A IND Y	18 A4	ff	5								
ANDB (opr)	AND B with Memory	B • M ⇒ B	B IMM	C4	ii	2					↕	↕	0	
			B DIR	D4	dd	3								
			B EXT	F4	hh ll	4								
			B IND,X	E4	ff	4								
			B IND Y	18 E4	ff	5								
ASL (opr)	Arithmetic Shift Left		EXT	78	hh ll	6					↕	↕	↕	↕
			IND X	68	ff	6								
			IND Y	18 68	ff	7								
ASLA	Arithmetic Shift Left A		A INH	48		2					↕	↕	↕	↕
ASLB	Arithmetic Shift Left B		B INH	58		2					↕	↕	↕	↕
ASLD	Arithmetic Shift Left D		INH	05		3					↕	↕	↕	↕
ASR	Arithmetic Shift Right		EXT	77	hh ll	6					↕	↕	↕	↕
			IND X	67	ff	6								
			IND Y	18 67	ff	7								
ASRA	Arithmetic Shift Right A		A INH	47		2					↕	↕	↕	↕
ASRB	Arithmetic Shift Right B		B INH	57		2					↕	↕	↕	↕
BCC (rel)	Branch if Carry Clear	? C 0	REL	24	rr	3								
BCLR (opr) (msk)	Clear Bit(s)	M • (mm) ⇒ M	DIR	15	dd mm	6					↕	↕		
			IND X	1D	ff mm	7								
			IND Y	18 1D	ff mm	8								
BCS (rel)	Branch if Carry Set	? C 1	REL	25	rr	3								

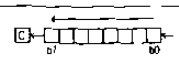
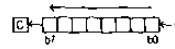
SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (1 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Operand	Cycles	Condition Codes S	X	H	I	N	Z	V	C
BEQ (rel)	Branch if = Zero	? Z = 1	REL	27	rr	3	—	—		—			—	
BGE (rel)	Branch if ≥ Zero	? N ⊕ V = 0	REL	2C	rr	3	—		—		—	—	—	—
BGT (rel)	Branch if > Zero	? Z + (N ⊕ V) = 0	REL	2E	rr	3		—	—	—	—		—	
BHI (rel)	Branch if Higher	? C + Z = 0	REL	22	rr	3	—	—						—
BHS (rel)	Branch if Higher or Same	? C = 0	REL	24	rr	3	—		—	—		—		—
BITA (opr)	Bit(s) Test A with Memory	A • M	A IMM A DIR A EXT A IND,X A IND,Y	85 95 B5 A5 18 A5	ii dd hh ll ff ff	2 3 4 4 5	—	—	—		↕	Δ	0	
BITB (opr)	Bit(s) Test B with Memory	B • M	B IMM B DIR B EXT B IND,X B IND,Y	C5 D5 F5 E5 18 E5	ii dd hh ll ff ff	2 3 4 4 5	—				↕	Δ	0	
BLE (rel)	Branch if ≤ Zero	? Z + (N ⊕ V) = 1	REL	2F	rr	3			—	—	—			
BLO (rel)	Branch if Lower	? C = 1	REL	25	rr	3	—	—	—	—				
BLS (rel)	Branch if Lower or Same	? C + Z = 1	REL	23	rr	3	—							
BLT (rel)	Branch if < Zero	? N ⊕ V = 1	REL	2D	rr	3	—		—					
BMI (rel)	Branch if Minus	? N = 1	REL	2B	rr	3	—							
BNE (rel)	Branch if not = Zero	? Z = 0	REL	26	rr	3								
BPL (rel)	Branch if Plus	? N = 0	REL	2A	rr	3	—		—					
BRA (rel)	Branch Always	? 1 = 1	REL	20	rr	3								
BRCLR(opr) (msk) (rel)	Branch if Bit(s) Clear	? M • mm = 0	DIR IND,X IND,Y	13 1F 18 1F	dd mm rr ff mm rr ff mm rr	6 7 8								
BRN (rel)	Branch Never	? 1 = 0	REL	21	rr	3								
BRSET (opr) msk) (rel)	Branch if Bit(s) Set	? (M) • mm = 0	DIR IND,X IND,Y	12 1E 18 1E	dd mm rr ff mm rr ff mm rr	6 7 8								
BSET (opr) msk)	Set Bit(s)	M + mm ⇒ M	DIR IND,X IND,Y	14 1C 18 1C	dd mm ff mm ff mm	6 7 8					↕	↕	0	
BSR (rel)	Branch to Subroutine	See Figure 3-2	REL	8D	rr	6							—	
BVC (rel)	Branch if Overflow Clear	? V = 0	REL	28	rr	3								
BVS (rel)	Branch if Overflow Set	? V = 1	REL	29	rr	3								
CBA	Compare A to B	A − B	INH	11		2					↕	↕	↕	↕
CLC	Clear Carry Bit	0 ⇒ C	INH	0C		2				—				0
CLI	Clear Interrupt Mask	0 ⇒ I	INH	0E		2				0				
CLR (opr)	Clear Memory Byte	0 ⇒ M	EXT IND,X IND,Y	7F 6F 18 6F	hh ll ff ff	6 6 7					0	↕	0	0

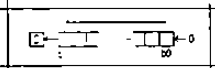
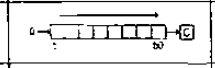
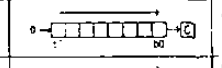
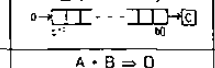
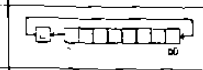
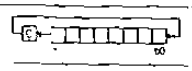
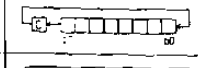
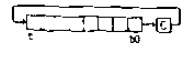
SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (2 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
CLRA	Clear Accumulator A	0 ⇒ A	A INH	4F		2	—				0	1		0
CLRB	Clear Accumulator B	0 ⇒ B	B INH	5F	—	2					0	1		0
CLV	Clear Overflow Flag	0 ⇒ V	INH	0A	—	2								
CMPA (opr)	Compare A to Memory	A − M	A IMM	81	ii	2	—				↕	↕		↕
			A DIR	91	dd	3								
			A EXT	B1	hh ll	4								
			A IND,X	A1	ff	4								
			A IND,Y	18 A1	ff	5								
CMPB (opr)	Compare B to Memory	B − M	B IMM	C1	ii	2	—				↕	↕		↕
			B DIR	D1	dd	3								
			B EXT	F1	hh ll	4								
			B IND,X	E1	ff	4								
			B IND,Y	18 E1	ff	5								
COM (opr)	One's Complement Memory Byte	$FF − M ⇒ M	EXT	73	hh ll	6	—				↕	↕		1
			IND,X	63	ff	6								
			IND,Y	18 63	ff	7								
COMA	One's Complement A	$FF − A ⇒ A	A INH	43	—	2					↕	↕		1
COMB	One's Complement B	$FF − B ⇒ B	B INH	53	—	2	—				↕	↕		1
CPD (opr)	Compare D to Memory 16-Bit	D − M : M + 1	IMM	1A 83	jj kk	5					↕	↕		↕
			DIR	1A 93	dd	6								
			EXT	1A B3	hh ll	7								
			IND,X	1A A3	ff	7								
			IND,Y	CD A3	ff	7								
CPX (opr)	Compare X to Memory 16 Bit	IX − M : M + 1	IMM	8C	jj kk	4					↕	↕		↕
			DIR	9C	dd	5								
			EXT	BC	hh ll	6								
			IND,X	AC	ff	6								
			IND,Y	CD AC	ff	7								
CPY (opr)	Compare Y to Memory 16 Bit	IY − M : M + 1	IMM	18 8C	jj kk	5					↕	↕		↕
			DIR	18 9C	dd	6								
			EXT	18 BC	hh	7								
			IND,X	1A AC	ff	7								
			IND,Y	18 AC	ff	7								
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19		2					↕	↕		↕
DEC (opr)	Decrement Memory Byte	M − 1 ⇒ M	EXT	7A	hh ll	6					↕	↕		
			IND,X	6A	ff	6								
			IND,Y	18 6A	ff	7								
DECA	Decrement Accumulator A	A − 1 ⇒ A	A INH	4A		2						↕	↕	
DECB	Decrement Accumulator B	B − 1 ⇒ B	B INH	5A		2						↕	↕	
DES	Decrement Stack Pointer	SP − 1 ⇒ SP	INH	34		3								
DEX	Decrement Index Register X	IX − 1 ⇒ IX	INH	09		3							↕	
DEY	Decrement Index Register Y	IY − 1 ⇒ IY	INH	18 09		4							↕	
EORA (opr)	Exclusive OR A with Memory	A ⊕ M ⇒ A	A IMM	88	ii	2					↕	↕		
			A DIR	98	dd	3								
			A EXT	B8	hh ll	4								
			A IND,X	A8	ff	4								
			A IND,Y	18 A8	ff	5								
EORB (opr)	Exclusive OR B with Memory	B ⊕ M ⇒ B	B IMM	C8	ii	2					↕	↕		
			B DIR	D8	dd	3								
			B EXT	F8	hh ll	4								
			B IND,X	E8	ff	4								
			B IND,Y	18 E8	ff	5								

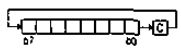
SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (3 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Operand	Cycles	S	X	H	I	N	Z	V	C
FDIV	Fractional Divide 16 by 16	D / IX ⇒ IX, r ⇒ D	INH	03		41	—		—	—		Δ	Δ	Δ
IDIV	Integer Divide 16 by 16	D / IX ⇒ IX, r ⇒ D	INH	02	—	41						↕	0	↕
INC (opr)	Increment Memory Byte	M + 1 ⇒ M	EXT	7C	hh ll	6					↕	↕	↕	—
			IND,X	6C	ff	6								
			IND,Y	18 6C	ff	7								
INCA	Increment Accumulator A	A + 1 ⇒ A	A INH	4C		2					↕	Δ	↕	
INCB	Increment Accumulator B	B + 1 ⇒ B	B INH	5C	—	2			—		↕	↕	Δ	
INS	Increment Stack Pointer	SP + 1 ⇒ SP	INH	31	—	3			—					
INX	Increment Index Register X	IX + 1 ⇒ IX	INH	08	—	3		—			—	Δ		
INY	Increment Index Register Y	IY + 1 ⇒ IY	INH	18 08		4						Δ		
JMP (opr)	Jump	See Figure 3-2	EXT	7E	hh ll	3				—	—	—		
			IND,X	6E	ff	3								
			IND,Y	18 6E	ff	4								
JSR (opr)	Jump to Subroutine	See Figure 3-2	DIR	9D	dd	5								
			EXT	BD	hh ll	6								
			IND,X	AD	ff	6								
			IND,Y	18 AD	ff	7								
LDAA (opr)	Load Accumulator A	M ⇒ A	A IMM	86	ii	2					↕	↕	0	
			A DIR	96	dd	3								
			A EXT	B6	hh	4								
			A IND,X	A6	ff	4								
			A IND,Y	18 A6	ff	5								
LDAB (opr)	Load Accumulator B	M ⇒ B	B IMM	C6	ii	2					↕	↕	0	
			B DIR	D6	dd	3								
			B EXT	F6	hh	4								
			B IND,X	E6	ff	4								
			B IND,Y	18 E6	ff	5								
LDD (opr)	Load Double Accumulator D	M ⇒ A, M + 1 ⇒ B	IMM	CC	jj kk	3					↕	↕	0	
			DIR	DC	dd	4								
			EXT	FC	hh	5								
			IND,X	EC	ff	5								
			IND,Y	18 EC	ff	6								
LDS (opr)	Load Stack Pointer	M ⇒ SP, M + 1 ⇒ SP	IMM	8E	jj kk	3					↕	↕	0	
			DIR	9E	dd	4								
			EXT	BE	hh	5								
			IND,X	AE	ff	5								
			IND,Y	18 AE	ff	6								
LDX (opr)	Load Index Register X	M ⇒ IX, M + 1 ⇒ IX	IMM	CE	jj kk	3					Δ	↕	0	
			DIR	DE	dd	4								
			EXT	FE	hh	5								
			IND,X	EE	ff	5								
			IND,Y	CD EE	ff	6								
LDY (opr)	Load Index Register Y	M ⇒ IY, M + 1 ⇒ IY	IMM	18 CE	jj kk	4					↕	↕	0	
			DIR	18 DE	dd	5								
			EXT	18 FE	hh	6								
			IND,X	1A EE	ff	6								
			IND,Y	18 EE	ff	6								
LSL (opr)	Logical Shift Left		EXT	78	hh	6			—		Δ	↕	↕	↕
			IND,X	68	ff	6								
			IND,Y	18 68	ff	7								
LSLA	Logical Shift Left A		A INH	48		2					↕	↕	↕	↕
LSLB	Logical Shift Left B		B INH	58		2					Δ	↕	↕	↕

SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (4 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	X	H	I	N	Z	V	C
LSLD	Logical Shift Left Double		INH	05	—	3			—		↕	↕	↕	↕
LSR (opr)	Logical Shift Right		EXT IND,X IND,Y	74 64 18 64	hh ll ff ff	6 6 7	—	—			0	↕	↕	↕
LSRA	Logical Shift Right A		A INH	44		2					0	↕	↕	↕
LSRB	Logical Shift Right B		B INH	54		2			—		0	↕	↕	↕
LSRD	Logical Shift Right Double		INH	04		3	—	—		—	0	↕	↕	↕
MUL	Multiply 8 by 8	A • B ⇒ D	INH	3D		10		—	—	—				↕
NEG (opr)	Two's Complement Memory Byte	0 − M ⇒ M	EXT IND,X IND,Y	70 60 18 60	hh ll ff ff	6 6 7		—	—		Δ	↕	↕	↕
NEGA	Two's Complement A	0 − A ⇒ A	A INH	40	—	2	—	—		—	Δ	Δ	Δ	↕
NEGB	Two's Complement B	0 − B ⇒ B	B INH	50	—	2		—	—		Δ	Δ	Δ	↕
NOP	No operation	No Operation	INH	01		2	—		—			—		
ORAA (opr)	OR Accumulator A (Inclusive)	A + M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	8A 9A BA AA 18 AA	ii dd hh ll ff ff	2 3 4 4 5	—		—		↕	↕	0	
ORAB (opr)	OR Accumulator B (Inclusive)	B + M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	CA DA FA EA 18 EA	ii dd hh ll ff ff	2 3 4 4 5	—		—		↕	↕	0	
PSHA	Push A onto Stack	A ↓ Stk,SP = SP − 1	A INH	36		3								
PSHB	Push B onto Stack	B ↓ Stk,SP − SP − 1	B INH	37		3								
PSHX	Push X onto Stack (Lo First)	IX ↓ Stk SP SP − 2	INH	3C		4								
PSHY	Push Y onto Stack (Lo First)	IY ↓ Stk SP SP − 2	INH	18 3C		5								
PULA	Pull A from Stack	SP SP + 1, A ↑ Stk	A INH	32	—	4								
PULB	Pull B from Stack	SP SP + 1, B ↑ Stk	B INH	33	—	4								
PULX	Pull X From Stack (Hi First)	SP SP + 2, IX ↑ Stk	INH	38	—	5	—							
PULY	Pull Y from Stack (Hi First)	SP = SP + 2 IY ↑ Stk	INH	18 38		6				—				
ROL (opr)	Rotate Left		EXT IND X IND Y	79 69 18 69	hh ll ff ff	6 6 7	—	—	—		Δ	↕	↕	↕
ROLA	Rotate Left A		A INH	49		2					↕	↕	↕	↕
ROLB	Rotate Left B		B INH	59	—	2	—	—			↕	↕	↕	↕
ROR (opr)	Rotate Right		EXT IND X IND Y	76 66 18 66	hh ll ff ff	6 6 7				—	↕	↕	↕	↕

SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (5 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	X	H	I	N	Z	V	C
RORA	Rotate Right A		A INH	46	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RORB	Rotate Right B		B INH	56	—	2	—	—	—	—	Δ	Δ	Δ	Δ
RTI	Return from Interrupt	See Figure 3–2	INH	3B	—	12	Δ	↓	Δ	Δ	Δ	Δ	Δ	Δ
RTS	Return from Subroutine	See Figure 3–2	INH	39	—	5	—	—	—	—	—	—	—	—
SBA	Subtract B from A	A – B ⇒ A	INH	10	—	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA (opr)	Subtract with Carry from A	A – M – C ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	82 92 B2 A2 18 A2	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
SBCB (opr)	Subtract with Carry from B	B – M – C ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C2 D2 F2 E2 18 E2	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
SEC	Set Carry	1 ⇒ C	INH	00	—	2	—	—	—	—		—	—	1
SEI	Set Interrupt Mask	1 ⇒ I	INH	0F	—	2	—	—	—	1	—	—	—	—
SEV	Set Overflow Flag	1 ⇒ V	INH	0B	—	2	—	—	—	—	—	—	1	—
STAA (opr)	Store Accumulator A	A ⇒ M	A DIR A EXT A IND,X A IND,Y	97 B7 A7 18 A7	dd hh ll ff ff	3 4 4 5	—	—	—	—	Δ	Δ	0	—
STAB (opr)	Store Accumulator B	B ⇒ M	B DIR B EXT B IND,X B IND,Y	D7 F7 E7 18 E7	dd hh ll ff ff	3 4 4 5	—	—	—	—	Δ	Δ	0	—
STD (opr)	Store Accumulator D	A ⇒ M, B ⇒ M + 1	DIR EXT IND,X IND,Y	DD FD ED 18 ED	dd hh ll ff ff	4 5 5 6	—	—	—		Δ	Δ	0	
STOP	Stop Internal Clocks	—	INH	CF	—	2		—	—		—	—	—	—
STS (opr)	Store Stack Pointer	SP ⇒ M M + 1	DIR EXT IND,X IND,Y	9F BF AF 18 AF	dd hh ll ff ff	4 5 5 6	—	—	—	—	Δ	Δ	0	
STX (opr)	Store Index Register X	IX ⇒ M M + 1	DIR EXT IND,X IND Y	DF FF EF CD EF	dd hh ll ff ff	4 5 5 6	—	—	—	—	Δ	Δ	0	
STY (opr)	Store Index Register Y	IY ⇒ M M + 1	DIR EXT IND,X IND Y	18 DF 18 FF 1A EF 18 EF	dd hh ll ff ff	5 6 6 6	—	—	—	—	Δ	Δ	0	—
SUBA (opr)	Subtract Memory from A	A – M ⇒ A	A IMM A DIR A EXT A IND,X A IND Y	80 90 B0 A0 18 A0	ii dd hh ll ff ff	2 3 4 4 5	—	—	—	—	Δ	Δ	Δ	Δ
SUBB (opr)	Subtract Memory from B	B – M ⇒ B	A IMM A DIR A EXT A IND,X A IND Y	C0 D0 F0 E0 18 E0	ii dd hh ll ff ff	2 3 4 4 5		—	—	—	Δ	Δ	Δ	Δ
SUBD (opr)	Subtract Memory from D	D – M M + 1 ⇒ D	IMM DIR EXT IND,X IND,Y	83 93 B3 A3 18 A3	jj kk dd hh ll ff ff	4 5 6 6 7	—	—	—	—	Δ	Δ	Δ	Δ

SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (6 DE 7)

Mnemonic	Operation	Description	Addressing Mode	Instruction Opcode	Instruction Operand	Instruction Cycles	Condition Codes S	X	H	I	N	Z	V	C
SWI	Software Interrupt	See Figure 3–2	INH	3F	—	14	—		—	1		—	—	—
TAB	Transfer A to B	A ⇒ B	INH	16	—	2		—		—	Δ	Δ	0	—
TAP	Transfer A to CC Register	A ⇒ CCR	INH	06	—	2	↕	↕	Δ	Δ	↕	↕	Δ	Δ
TBA	Transfer B to A	B ⇒ A	INH	17	—	2			—		Δ	Δ	0	—
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00	—	*			—	—			—	—
TPA	Transfer CC Register to A	CCR ⇒ A	INH	07	—	2	—	—	—			—	—	—
TST (opr)	Test for Zero or Minus	M – 0	EXT IND,X IND,Y	7D 6D 18 6D	hh ll ff ff	6 6 7	—		—	—	Δ	Δ	0	0
TSTA	Test A for Zero or Minus	A – 0	A INH	4D	—	2		—			Δ	Δ	0	0
TSTB	Test B for Zero or Minus	B – 0	B INH	5D	—	2					Δ	Δ	0	0
TSX	Transfer Stack Pointer to X	SP + 1 ⇒ IX	INH	30	—	3	—		—	—	—	—	—	
TSY	Transfer Stack Pointer to Y	SP + 1 ⇒ IY	INH	18 30	—	4							—	—
TXS	Transfer X to Stack Pointer	IX – 1 ⇒ SP	INH	35	—	3	—	—	—	—			—	—
TYS	Transfer Y to Stack Pointer	IY – 1 ⇒ SP	INH	18 35	—	4				—				
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E	—	**				—	—		—	—
XGDX	Exchange D with X	IX ⇔ D	INH	8F	—	3	—						—	—
XGDY	Exchange D with Y	IY ⇔ D	INH	18 8F	—	4								—

Cycle

* Infinity or until reset occurs

** 12 Cycles are used beginning with the opcode fetch. A wait state is entered which remains in effect for an integer number of MPU E Clock cycles (n) until an interrupt is recognized. Finally, two additional cycles are used to fetch the appropriate interrupt vector (14 + n cycles).

Operands

dd = 8 Bit Direct Address ($0000 –$00FF) (High Byte Assumed to be $00)
ff = 8-Bit Positive Offset $00 (0) to $FF (255) (Is Added to Index)
hh = High-Order Byte of 16 Bit Extended Address
ii = One Byte of Immediate Data
jj = High Order Byte of 16-Bit Immediate Data
kk = Low Order Byte of 16-Bit Immediate Data
ll = Low Order Byte of 16 Bit Extended Address
mm = 8-Bit Mask (Set Bits to be Affected)
rr = Signed Relative Offset $80 (–128) to $7F (+ 127)
(Offset Relative to Address Following Machine Code Offset Byte)

Operators

( ) Contents of register shown inside parentheses
⇒ Is transferred to
⇔ Is exchanged with
⇑ Is pulled from stack
⇓ Is pushed onto stack
• Boolean AND
+ Arithmetic Addition Symbol except where used as Inclusive-OR symbol in Boolean Formula
⊕ Exclusive OR
× Multiply
Concatenation
– Arithmetic subtraction symbol or Negation symbol (Two's Complement)

Condition Codes

Bit not changed
0 Bit always cleared
1 Bit always set
Δ Bit cleared or set depending on operation
↕ Bit can be cleared or set, or may become set

SET DE INSTRUCCIONES DEL MICROCONTROLADOR MC68HC11 (7 DE 7)

ANEXO C

TRABAJOS FUTUROS

TRABAJOS FUTUROS

- Optimización de Rutinas.
- Desarrollo de otras Interrupciones.
- Implementarle una Lectora de Código de Barras.
- Completar la Idea Original.

ANEXO D

BIBLIOGRAFIA

Microporcessor Microcontroller and Peripherical Data Vol I y II
MOTOROLA INC. 1988.

M68HC11 E Series
Technical Data
MOTOROLA INC. 1993.

M68HC11 E Series
Reference Manual
MOTOROLA INC. 1991.

8- bit MCU Applications Manual
MOTOROLA INC. 1992.

Microcomputer Interfacing
Bruce a. Artwick
Pretice-Hall, Inc. 1980.

Laboratory Automation Using the IBM PC
Stephen C. Gates with Jordan Becker.
Prentice Hall. 1989.

The Peter Norton PC Programer's Bible.
Peter Norton, Peter Aitken y Richard Wilton.
Microsoft Press. 3a. Edic. 1993.

DOS Guía para Usuarios Expertos.
Kris Jamsa.
Osborne/McGraw Hill. 1988.

CMOS DATABOOK
Nacional Semiconductor Corporation. 1981.

Memory Components Handbook
INTEL. 1985.

Apuntes del Curso del Microcontrolador MC68HC11
Instituto Tecnológico de Morelia, Mich.
Septiembre de 1983.
M.C. JUAN JOSE DARIO DELGADO ROMERO.

Lenguaje Ensamblador para Microcomputadoras IBM para Principiantes y Avanzados.
J. Terry Godfrey
PHH 1991.