UNIVERSIDAD POLITECNICA DE LAS PALMAS

INGENIERIA TECNICA DE TELECOMUNICACION

PROYECTO FIN DE CARRERA

TITULO: DESARROLLO DE LA MCS-48. APLICACIONES:

- MEDIDOR DIGITAL DE PRESION SANGUINEA INCRUENTA.
- CERRADURA CODIFICADA.

AUTOR : MANUEL J. VALOIRA OJEDA.

TUTOR: SEBASTIAN SUAREZ GIL.

FEBRERO - 86

PREFACIO

En los últimos años el desarrollo en el campo de la microelectrónica ha supuesto un gran avance en todos las ramas de la ciencia. La aparicion de los circuitos integrados a gran escala ha hecho posible la realizacion de proyectos y diseños que sin estos no hubiese sido posible.

La aparicion en 1972 del procesador 8008 de Intel supuso el comienzo de la era de los microprocesadores.

En 1973 Intel lanza el 8080 , mas potente que su antecesor. Los datos estaban ahi, se habia reducido en mas de un 50 por 100 los costes de un sistema minimo.

En cuanto a la integracion, se pasaba de 20 componetes por sistema (8008) a solo 5 componentes con el nuevo microprocesador. El gran paso se dio con la aparicion en 1976 de 8048 de Intel, no se trata ya de un microprocesador sino de un microcomputador " monochip". Si, efectivamente, se trata de un circuto integrado con todas las funciones necesarias para formar un sistema microcomputerizado.

El objetivo de este proyecto de fin de carrera es hacer un amplio estudio y desarrollo de la familia de microcomputadores MCS-48 de Intel.

El primer capitulo de este proyecto hace una descripcion detallada del Hardware de la MCS-48, describiendose su extructura, tanto interna como externa asi

como su funcionamiento.

El segundo capitulo esta dedicado al Software. En el se describen las transferencias de datos, los direccionamientos de registros asi como, la totalidad de las instrucciones.

En el tercer capitulo se hace una comparacion entre este microcomputador y el popular 8085 de Intel, viendo las caracteristicas de uno y del otro. La ultima parte del capitulo muestra las aplicaciones mas comunes de la MCS-48.

El capitulo cuarto es una parte practica, en donde se realiza un programa Macro-ensamblador MCS-48 en lenguaje BASIC.

Este ensamblador es una herramienta muy util a la hora de crear programas en codigo maquina para este microcomputador.

Los dos ultimos capitulos del proyecto son aplicaciones practicas. El capitulo 5 contiene el diseño y montaje de una cerradura codificada en la que una unica combinacion , activa un dispositivo de apertura. Este diseño es capaz tambien de activar una alarma en caso que la combinacion sea erronea.

El capitulo 6 describe el diseño y montaje de un medidor digital de presion sanguinea incruenta.

Estos dos diseños estan basados en la MCS-48.

Por ultimo decir que el proyecto se complementa con 6 apaendices donde cada uno hace referencia a un unico capitulo.

INDICE.

CAPITULO 1 : LA FAMILIA MCS-48. HARDWARE.

CAPITULO SEGUNDO. REPERTORIO DE INSTRUCCIONES DEL 8048. SOFTWARE.

CAPITULO TERCERO. APLICACIONES DE LA MCS-48. ESTUDIO COMPARATIVO CON EL MICROPROCESADOR 8085 DE INTEL.

APENDICES:

1.1 INTRODUCCION A LA FAMILIA MCS-48

Recientemente los avances en tecnologia NMOS han permitido a Intel la creación de los primeros microcomputador monochip conteniendo en una sola pastilla todas las funciones requeridas por un sistema digital de procesamiento .

Tales microcomputadores ,su variaciones y perifericos opcionales componen lo que se denomina la familia MCS-48.

La cabeza de la familia es el microcomputador 8048, el cual contiene en una sola pastilla de 40 pines las siguientes funciones:

-CPU de 8bits

-1Kx8 bytes de memoria ROM.

-64x8 bytes de memoria RAM.

-27 Lineas de Entrada /Salida (3 puertos de 8 bits.)

-Contador / Timer de 8 bits programable.

Un repertorio de aproximadamente 90 instrucciones, de uno o dos ciclos cada una, hacen de 8048 microcomputador una herramienta muy potente y versatil en sistemas digitates.

El 8048, es tambien, un micro computador de bajo costo que solo requiere una tensión de 5V simple para su funcionamiento.

1.2 OTROS MIEMBROS DE LA MSC-48.

La familia MCS-48 de microcomputadores comienza con el 8048 y 8748, este ultimo una versión identica al 8048 pero con memoria EMPROM. El 8035 es una copia identica del 8048 pero sin memoria ROM interna.

Posteriormente nació el 8049 que es un micro computador totalmente compatible con el 8048 tanto en instruciones como en el patillaje pero con el doble de memoria RAM, y doble de memoria ROM que su antecesor. Del 8049 se deriva el 8039,que es toltamente identico pero sin memoria ROM interna.

El 8021 es un micro computador de la MCS-48 de muy bajo costo;(el mas barato de la familia). Este contiene un extracto de las instruciones del 8048, es decir que el numero de instrucciones que posee es menor que el 8048, pero, eso si, todas son compatibles con las del 8048. El 8021 solo necesita una resistencia para generar la frecuencia de reloj.

El 8022 es un microcomputador basado en el 8021 con más memoria RAM, mas lineas I/O y ademas un convertidor A/D incorporado.

El diseño de un sistema basado en microcomputador resultaria demasiado caro si no se dispusiese del 8748, el cual, gracias a su memoria EPROM permite la modificación del programa cuantas veces se desee.

Una vez que se ha cnseguido el prototipo, el 8748 puede ser sustituido por un 8048, 8049, 8021, que son de mucho mas bajo costo que el 8748.

El 8748 permite un prediseño mucho más facil y comodo que las verciones ROM.

Para mas informacion de los diferentes miembros de la MCS-48 ver la figura 1-A (Figura 1 del apendice A).

1.3 CARACTERISTICAS TECNICAS BASICAS.

Los microcomputadores que conponen la familia MCS-48 poseen las siguientes caracteristicas:

- Alimentacion simple de +5 voltios y GND.
- Capsula normalizada de 40 y 28 pines segun el tipo de microconputador.
- Compatibilidad de pines entre las versiones ROM y EPROM
- Existen versiones de 10, 5, y 2.5 uS. de periodo en todos los modelos

- Todas las instrucciones ocupan uno o dos bytes de memoria como maximo.
- Todas las instrucciones se realizan en un ciclo maquina o dos, como maximo.
- Tienen la posibilidad de ejecucion del programa paso a paso mediante la entrada SS (Single Step).
- 8 niveles de Stack. (Pila de 8 registros). Ampliables a 16 niveles por programa.

- Existencia de dos bancos de registros de trabajo de 8 registros cada uno.Estos registros agilizan las operaciones con el microprocesador.
- La frecuencia de reloj puede ser generada externamente mediante :

    - Cristal de cuarzo
    - Red inductiva.
    - Puertas TTL.

- Salida de reloj opcional. (Por programa).

1.4 PERIFERICOS

El periferico mas utilizado por la MCS-48 es el 8243, este es un puerto de 16 lineas de Entrada/Salida en un encapsulado de 24 patillas. Las 16 lineas se distribuyen en 4 puertos de 4 bits cada uno. Esta pastilla ha sido diseñada para la MCS-48 especialmente.

La MCS-48 es compatible con la mayoria de todos los perifericos de la familia MCS-80/85 y en especial con aquellos que posean un bus multiplexado.

1.5 ARQUITECTURA EXTERNA DEL MICROCOMPUTADOR 8048.

El estudio que vamos a realizar a partir de ahora servira no solo para el 8048 sino para el 8748, 8035, 8049, 8039 que son todos microcomputadores de 40 patillas,totalmente conpatibles entre si.

La figura 2-A muestra una configuracion simplificada del 8048 y su patillaje.

A continuacion pasamos a describir todas las patillas y lineas que conponen el 8048.

INT : Linea de peticion de interrupcion. Es activa a nivel bajo

RESET : Linea de inicializacion de la CPU. Es activa a nivel bajo. Cuando esta linea es activada el contador de programa salta a la direccion 0000H de la ROM. Esta linea tambien se utiliza durante el proceso de programacion.

RD : Orden de lectura del Bus exterior,activa a nivel bajo. Esta linea se utiliza con memorias y perifericos internos.

WR : Orden de escritura del bus exterior. Activa a nivel bajo.

XTAL1 , XTAL2 :Entradas para generar la frecuencia de reloj.

ALE : Esta linea indica que lo que hay en el bus es una direccion (ALE=1) o un dato (ALE=0).

EA : Esta linea de entrada se utiliza para inhibir la memoria ROM interna y forzar los ciclos de busqueda en una memoria externa. Dicho en otras palabras sustituimos la memoria de programa interna por una externa.Esta linea es activa a

nivel alto.

SS : Esta entrada se utiliza para la ejecucion paso a paso del programa. Funciona por flanco de bajada.

PSEM : Esta salida se utiliza para la habilitacion de memorias ROM externas. Es activa a nivel bajo. Esta linea da un cero cada vez que se accede a una memoria externa. Se suele utilizar como Chip Selec de las memorias externas.

PROG : Entrada del impulso de grabacion (25 voltios) para el 8748. En las versiones sin EPROM es una salida de STROBE para los perifericos.

VDD : Esta entrada se utiliza para programar aplicandole 25 voltios durante todo el proceso de programacion. Si no se esta programando esta linea ha de estar a 5 voltios.En las versiones con alimentacion STANDBY esta linea se utiliza solo para alimentar la RAM.

T0 : Esta linea puede funcionar de tres formas diferentes que son:

a) Como entrada de muestreo testeable por la CPU mediante las instrucciones;

JT0 : Salto si la linea T0 esta a 1.

JNT0: Salto si la linea T0 esta a 0.

b) Como salida de reloj utilizando la instruccion;

ENTOCLK: Enable TO for Clock.

c) Tambie se utiliza para programar el 8748.

T1 : Esta linea puede funcionar de dos formas diferentes que son:

a) Como entrada de muestreo testeable por la CPU mediante las instrucciones;

JT1 : Salto si la linea T1 esta a 1

JNT1: Salto si la linea T1 esta a 0

b) Como entrada de reloj del TIMER/COUNTER.

VCC : Linea de alimentacion del chip, ecepto de la RAM en las versiones con STANDBY.

GND : Linea de tierra.

P10-P17 : Puerto de Entrada /Salida numero 1

P20-P27 : Puerto de Entrada /Salida numero 2

Db0-Db7 : Puerto de Entrada /Salida numero 0. Este puerto ademas de usarse como tal, es tambien, utilizado por el microcomputador como bus de datos/direcciones para comunicarse con el exterior. Desde el punto de vista fisico es muy

similar al bus multiplexado del microprocesador 8085.

1.6 EXTRUCTURA INTERNA DEL 8048.

En este apartado haremos un estudio exaustivo de la arquitectura interna del 8048. Para esto,estudiaremos el diagrama de bloques de la figura 3-A.

1.6.1 - MEMORIA DE PROGRAMA.

La memoria de programa es una memoria ROM o EPROM de uno o dos Kbytes segun el microcomputador que sea, pero en todos ellos hay que distinguir tres posiciones de especial importancia. Estas posiciones son:

Direccion 0000H : Esta es la primera direccion de la ROM,se accede a ella mediante la aplicacion de un reset exterior o mediante un JUMP incondicional. En esta posicion se coloca la primera instruccion a ser ejecutada tras un reseteo.

Direccion 0003H : Esta es la cuarta direccion de la ROM,se accede a ella mediante la activacion de la linea INT siempre y cuando la interrupcion este habilitada por programa. En esta posicion se coloca la primera instruccion a ser ejecutada tras una peticion de interrupcion.

Direccion 0007H : Esta es la octava posicion de la ROM, se accede a ella cuando se produce un overflow en el TIMER/COUNTER del microcomputador,siempre y cuando se habilite

la interrupcion de overflow del TIMER/COUNTER. En esta posicion se coloca la primera instruccion a ser ejecutada tras un overflow del TIMER/COUNTER.

Las demas posiciones de la ROM no tienen ninguna funcion especial.

1.6.2 - MEMORIA DE DATOS

Segun el tipo de microcomputador la memoria RAM puede ser de 64 bytes o 128 bytes, es decir de 64 o 128 palabras de 8 bits. Dentro de la RAM existen una serie de registros de especial importancia. Ver figura 4-A.

Los registros R0 y R1 se utilizan para direccionar de forma indirecta cualquier posicion de la RAM. Estos registros residen en las posiciones 0 y 1 respectivamente de la RAM. Estos registros reciben el nombre de registros punteros.

Las ocho primeras posiciones de la RAM (direcciones 0-7) forman un grupo de registros que se denomina Banco 0 de registros de trabajo. Estos registros se pueden utilizar como registros temporales para agilizar el trabajo de la CPU.

Existen una serie de instrucciones relaccionadas con estos registros tales como incrementos, complementacion, suma directa con el acumulador etc, que hacen de estos registros una herramienta muy potente.

Los registros R0 yR1 ademas de ser los registros punteros tambien funcionan como registros de trabajo normales.

Mediante la instruccion SEL RB se puede acceder a un segundo banco de trabajo de 8 registros igual que el anterior pero en las direcciones 24 a 31 ambas inclusive. Cuando se selecciona este segundo banco de trabajo el primero desaparece

como tal, pero el contenido de los registros se conserva. Esta caracteristica, permite en potencia disponer de 16 registros de trabajo, con la salvedad de que solo podemos utilizar de forma directa solo el banco de trabajo seleccionado.

En el segundo banco de trabajo los registros R24 y R25 se utilizan como registros punteros, al igual que R0 y R1 del primer banco.

Las posiciones 8 a 23 de la RAM son utilizables por la CPU para crear una pila de 8 niveles, es decir de 8 registros.

Por ultimo decir que cualquier posicion de la RAM puede ser utilizada como registro normal de datos, incluso los registros de trabajo y la pila.

1.6.3 - LINEAS DE ENTRADA/ SALIDA. PUERTOS.

El 8048 posee 27 lineas que se pueden utilizar tanto como entrada como salida de datos.

Estas lineas estan agrupadas en 3 puertos de 8 bits cada uno que pueden ser utilizados como entradas ,salidas o registros bidireccionales.

Ademas el 8048 posee tres linesa de "Test" que pueden modificar la secuencia del programa cuando dichas lineas son testeadas pro la CPU. Esta lineas son T0, T1, e INT.

1.6.3.1 - PUERTOS 1 Y 2.(QUASI BIDIRECCIONALES).

Estos dos puertos son de caracteristicas identicas. Para comprender el funcionamiento de estos puertos ver la figura 5-A.

Cuando la CPU escribe un dato en el puerto, este queda memorizado en el LACH tipo D que posee cada linea del puerto. Este dato permanece ahi hasta que es reescrito de nuevo por la CPU. En cambio cuando se trata de la lectura de un dato es necesario que dicho dato este presente a la entrada del puerto en el momento en que la CPU ordena la lectura de dicho puerto.La razon es que no existe LACH de entrada tan solo existe un BUFFER. (ver fig 5-A).

Estos puertos son compatibles con TTL tanto en entrada como en salida de datos.

A continuacion vamos a explicar el termino QUASI bidireccional. Se dice que estos dos puertos tienen una extructura QUASI bidireccional porque pueden funcionar tanto como entrada como salida, pero que la entrada esta condicionada por la salida.

Dicho de otra forma, para que una linea se pueda utilizar como entrada es necesario antes poner a uno dicha linea.

La razon es la siguiente:

Supongamos que mandamos un cero al lach de salida, esto hace que Q=0 y Q=1. Observando la figura 5-A vemos que en estas condiciones T1 queda cortado y T2 queda saturado.El circuito equivalente seria el de la figura 6-A.

Como vemos en dicha figura el Pin queda practicamente a masa por medio de T2 que se comporta como un corto, y en consecuencia esta linea no se podra utilizar como entrada.

Supongamos ahora que mandamos un uno al lach de salida, esto hace que Q=1 y Q=0 y por tanto T1 queda en corte pues la salida de la puerta AND esta a cero Tras la orden de escritura y T2 queda en corte tambien,pues esta unida directamente a Q . En esta condiciones el circuito equivalente seria el mostrado en la figura 7-A.

Como se ve en dicha figura, el PIN queda a uno "flotante" gracias al resistor de 50 Kohm. En esta condiciones este pin se puede utilizar como entrada siempre que se de la orden IN al buffer de entrada.

Resumiendo:

La salida es siempre posible en un Puerto de este tipo.

La entrada solo se puede hacer si antes ponemos a uno dicha linea.

La extructura QUASI bidireccional permite en todo momento utilizar las lineas de un puerto como entradas y salidas al mismo tiempo.

Una circuiteria adicional a la extructura de estos puertos, permite realizar mediante instrucciones, operaciones logicas con los datos presentes en dichos puertos,sin necesidad de recurrir al acumulador.

1.6.3.2 - PUERTO 0. BUS BUFFER

Este puerto de 8 bits tiene dos modo de funcionamiento diferentes:

MODO A: BUS BUFFER.

En este modo el puerto se comporta como un bus bidirecional semejante al bus multiplexado de 8085. Por medio de este bus el microcomputador se comunica con el exterior. Este modo se selecciona automaticamente mediante instrucciones que hacen referencia al bus.

MODO B: PUERTO QUASI BIDIRECCIONAL.

En este modo el puerto se comporta como tal, es decir como puerto Quasi bidireccional al igual que los puertos 1 y 2 antes estudiados. Este modo se selecciona automaticamente mediante instrucciones que hacen referencia al port,estas instrucciones son IN y OUT.

Tanto un modo como otro hacen que se generen las señales de WR y RD, pero estas son utiles solo cuando se trabaja como bus. Cuando este puerto no se utiliza y esta en modo BUS Buffer las lineas quedan en alta impedancia, esto evita muchos problemas a la hora de conexionado con perifericos.

1.6.3.3 - ENTRADAS DE TEST T0, T1, INT.

Estas tres lineas son testeables mediante instrucciones de salto incondicinal. Pueden bifurcar el programa sin necesidad de recurrir a la utilizacion de un puerto, y ademas tienen la ventaja de no utilizar el acumulador para nada.

Las lineas T0 y T1 son iguales entre si. Estas lineas no crean una interrupcion pues solo son texteables por la CPU mediante instrucciones especiales. En cambio, la linea INT si

crea una interrupcion,cuando esta habilitada, saltando automaticamente a la direccion 0003 de la ROM.

Ademas de estas funciones de testeo, estas lineas tienen otras funciones que se veran mas adelante.

La linea INT cuando no esta habilitada funciona como una entrada de testeo mediante la instruccion JNI (Salto si la linea INT esta a uno).

1.6.4 - CONTADOR DE PROGRAMA Y PUNTERO DE PILA.

El contador de programa como vemos en la figura 3-A esta implementado fuera de la memoria RAM. Este esta formado por un registro de 12 bits como se muestra en la figura 8-A.

Mediante los bits CP0-CP9 se selecciona hasta 1K de memoria ROM (8048). Con el siguiente bit (CP10) se puden seleccionar hasta 2K de memoria (8049). El ultimo bit de contador de programa CP11 permite seleccionar hasta 4K de memoria ROM.

Puesto que el 8049 posee como maximo 2K bytes de memoria ROM el bit CP11 del contador de programa no se pondra a uno nunca. Este bit solo se pone a uno cuando se accede a una expansion de memoria ROM externa. La razon de esto es que los 4K bytes de memoria se dividen en dos bancos de memoria, cada uno de dos 2Kbytes.

El bit CP11 solo se puede poner a 1 o 0 por programa, es decir mediante una instruccion, nunca por el incremento natural del contador de programa.

Cuando la linea de reset es activada, el contador de programa pone todos los bits a cero.

Una interrupcion o llamada a subrutina hace que el contenido del contador de programa sea guardado en una pila. Esta pila esta formada por 8 niveles de 16 bits cada uno. Ver figura 9-A.

La pila esta contenida en la RAM entre los registros R8 y R23, ambos inclusive. Los 12 bits menos significativos de cada nivel, se utilizan para almacenar el contador de programa y los cuatro restantes para almacenar parte de la palabra de estado del procesador. (PSW).

La seleccion de uno de los 8 niveles de la pila se realiza mediante 3 bits que forman el puntero de pila (Stack Pointer). Estos 3 bits S0, S1, S2 se encuentran en la palabra de estado del procesador (PSW). La tabla de la figura 10-A muestra la seleccion de los niveles de la pila segun el valor del Stack Pointer (S0, S1, S2).

En un principio (Tras un Reset por ejemplo) el Stack Pointer se encuentra a 000. Cuando se produce una interrupcion o una llamada a subrutina, el contenido del contador de programa se almacena en la pila, junto con los cuatro bits mas significativos de la PSW y automaticamente el Stack Pointer se incrementa una unidad. Este proceso se realiza cada vez que se produce una interrupcion o llamada a subrutina.

La pila del 8048 es rotativa, es decir, que cuando esta llena y se introduce otro dato, este se almacena al principio de la pila, con lo que el dato anterior se destruye. Por esta razon siempre hay que vigilar es estado del Stack Pointer.

1.6.5 - PALABRA DE ESTADO DEL PROCESADOR. PSW.

La PSW (Processor Status Word) es un registro de 8 bits que puede ser leido o cargado por medio del acumulador y que se encarga de informar de algunos parametros. La figura 11-A muestra la distribucion de la PSW que a continuacion pasamos a describir:

S0,S1,S2 : Estos tres bits forman el puntero de pila, los cuales seleccionan uno de los ochos niveles de la pila.

El bit PSW3 no se usa y siempre esta a uno.

BS : (Bank Swich).Este bit es el conmutador del banco de trabajo y nos dice en todo momento cual de los dos bancos de trabajo esta seleccionado.

Si BS=0 ; Banco 0 de trabajo.

Si BS=1 ; Banco 1 de trabajo.

F0 : Este es un flip-flop controlable por el usuario que puede ser borrado, complementado, o examinado mediante la instruccion de salto incondicional JF0.

Nota: Este flip-flop no tiene nada que ver con la entrada testeable T0.

AC : Carry auxiliar. Este bit se pone a uno cuando se produce un acarreo BCD en el acumulador. Naturalmente tras utilizar la instruccion DAA (Ajuste decimal).

CY : Carry. Este bit se pone a uno cuando el contenido del acumulador, tras una operacion, es mayor que OFFH.

1.7 - INTERRUPCIONES.

La peticion de interrupcion del 8048 se realiza aplicando un cero a la linea INT. En la figura 11.A vemos la implementacion del hardware de la interrupcion.

La linea INT ataca a un flip-flop tipo D. La entrada de reloj es gobernada por la señal ALE y se produce en el ultimo ciclo de cada instruccion gracias a la puerta AND.

De esta forma se garantiza que tras cada instruccion la bascula D memorize el valor presente en la entrada INT.

La bascula RS (en la parte de abajo de la figura) es la que habilita o no la interrupcion segun el estado de sus entradas. Estas entradas son Enable Interrup (EN I) y Disable Interrup (DIS I) que son activadas por programa.
Cuando EN I esta a 1 la salida del RS se pone a 1 con lo que la puerta AND queda abierta, permitiendo a la linea INT activar la bascula RS de " Interrupcion en proceso", automaticamente el procesador reliza la instruccion en curso y luego salta a la posicion 0003 de la ROM.

En cambio, si la linea DIS E esta activada, hace que la puerta AND quede cerrada con lo que la linea INT nunca podra activar a la bascula de "Interrupcion en proceso.

Antes de que el procesador salte a la posicion 0003 del programa el contenido del contador de programa y la PSW son almacenados en la pila. EL final de una subrutina de interrupcion es señalado mediante un return.

Cualquier otra peticion de interrupcion durante la ejecucion de esta, es ignorada hasta que se produce el segundo ciclo de la instruccion RETURN. La razon es que cuando se produce una interrupcion, el procesador activa la linea DIS I para inhibir una nueva peticion de interrupcion. Por otro lado, tras finalizar la interrupcion, el codigo de la instruccion RETURN activa la linea EN I con lo que se habilita de nuevo la linea de interrupcion.

Para habilitar o inhibir las interrupciones se utilizan las instrucciones EN I y DIS I respectivamente.

Un reset exterior desabilita las interrupciones.

La peticion de interrupcion debe ser deshabilitadad (INT=1) antes de que se produzca el Return, pues de otro modo se volveria a realizar dicha subrutina de interrupcion.

NOTA: Hay perifericos que suspenden la peticion de interrupcion una vez han recibido la señal de "reconocimiento" de la interrupcion. INTA (8085).

El 8048 no posee tal señal,por lo que cuando haga falta se podra generar a travez de una de las lineas I/O de este.

Mientras no se habilite la linea INT, esta puede ser usada como entrada testeable mediante la instruccion; JNI

1.8 - TIMER / COUNTER.

Como sabemos el Timer/counter es un contador de 8 bits que puede ser programado y leido a travez del acumulador mediante las instrucciones:

MOV A,T : Transferir el Acc al Timer Counter.

MOV T,A : Transferir el Timer Counter al Acc.

En la figura 12.A se muestra el hardware del Timer /Counter.

Este contador puede funcionar mediante una frecuencia generada internamente (TIMER) ó externamente a travez de la linea T1 (COUNTER). En el primer caso servirá para crear tiempos de retardo (Delays) y en el segundo como contador de sucesos externos.

- Un reseteo no afecta al contenido del Timer Counter.

- Para detener la cuenta se utiliza la instruccion STOPTCNT. O aplicando un reset exterior.

- El maximo valor de contaje es OFFH y automaticamente pasa a OOH continuando la cuenta indefinidamente.

- El paso de FF a 00 pone a uno el flag de overflow del Timer/Counter. Ver figura.

- Este uno en el flag de overflow genera una interrupcion que hace que el programa bifurque a la direccion 0007 de la ROM, siempre y cuando se habilite la interrupcion por programa.

-Las instrucciones EN TCNT1 y DIS TCNT1 se habilita e inhibe la interrupcion del Timer/Counter respectivamente.

-El flag de overflow puede ser testeado mediante la instruccion JTF (Salto si el flag del Timer esta a uno).

-El flag de overflow se pone a cero mediante un reset exterior o tras la ejecucion de la instruccion JTF.

-Una vez terminada la rutina de tratamiento de Overflow, tras el RETURN, el procesador deshabilita de forma automatica el flag de overflow. De este modo se permite utilizar de nuevo la rutina de tratamiento.

-Si se está ejecutando una interrupcion exterior (INT), aunque se produzca una interrupcion de overflow el procesador ignora ésta hata que se realize la primera.

-La interrupcion externa tiene prioridad sobre la interrupcion de overflow.

-Para dar la orden de comienzo de cuenta o continuación de la misma existen dos instrucciones:

START CNT: arranque del contador.

START T : arranque del Timer.

La primera de estas instrucciones permite arrancar el contador en modo COUNT y la segunda en modo TIMER. Dicho de otra forma para seleccionar uno de los dos modos del TIMER/COUNT se utiliza una de estas dos instrucciones.

1.8.1 - MODOS DE FUNCIONAMIENTO DEL TIMER/ COUNT.

1.8.2 - COMO CONTADOR:

Como se ve en la figura 12.A la patilla T1 del 8048 se utiliza como entrada de reloj en el modo COUNT. La instruccion START CNT hace que la patilla T1 se conecte directamente a la entrada de reloj del contador.

El contador funciona por flanco de bajada de la señal de reloj.

La maxima frecuencia a la que puede trabajar el contador en este modo es la frecuencia del cristal dividida por 45.

Por ejemplo; si la frecuencia del cristal es 6Mhz la maxima frecuencia del contador sera ;

6 / 45 = 0.1333333 Mhz. = 133.333 Khz.

La señal aplicada a T1 debe ser pulsos de 500 ns. de anchura , como minimo.

1.8.3 - COMO TIMER:

El comienzo de la cuenta en este modo se realiza mediante la instruccion SRART T. Esta instruccion conecta el contador con una señal de reloj interna, la cual se obtiene pasando la señal del cristal (dividida por 15) por un preescaler que la divide por 32. Ver figura 12.A.

Como ejemplo, si la frecuencia de reloj es de 6 Mhz se obtiene una frecuencia de de 12 Khz tras pasar por el preescaler.

- El menor tiempo de retraso (delay) se obtiene programando OFFH en el contador. (Delay = 80 us.).

- El mayor Delay se consigue programando OOH en el contador. Este valor permite un delay de 20 ms.

- El comtador permite programar 256 delays comprendidos entre 80us. y 20ms.

- El final del delay se detecta mediante el flag de overflow.

- Si se habilita la interrupcion de overflow, se produce un salto a la direccion 0007 de la ROM tras el delay.

Resumiendo:

La unica diferencia entre COUNT y TIMER es que en el primer caso la frecuencia de reloj es externa,a travez de T1 y en el segundo la frecuencia es interna.

EL modo COUNT se selecciona mediante START CNT.

EL modo TIMER se selecciona mediante START T.

Para parar el TIMER / COUNT se utiliza la misma instruccion STOP TCNT ó aplicando un reset.

En el apartado de las instrucciones, se vera con mas detalle todas las instrucciones referentes al TIMER/COUNTER.

1.9. * CIRCUITO DE RELOJ DEL 8048 Y TIMING.

El generador de Timing del 8048 es totalmente interno y solo necesita de un cristal, inductancia ó puertas TTL para su funcionamiento. Ver figura 13.A.

Como se ve en la figura, las dos patillas XTAL1 y XTAL2 se utilizan para generar la frecuencia de reloj mediante un dispositivo externo. A continuacion pasamos a describir los bloques que forman el circuito de Timing del 8048.

1.9.1 * STATE COUNT:

Este bloque es un divisor por 3 que se utiliza para generar la frecuencia de reloj interna. Por ejemplo si la frecuencia del cristal es de 6 Mhz la frecuencia de reloj del 8048 sera 2 Mhz.

Mediante la instruccion EN T0 CLK se habilita la patilla T0 como salida de reloj.En la figura esto equivale a colocar el conmutador en la posicion 2.

Mediante un reset el conmutador pasa de la posicion 2 a 1 con lo que el pin T0 quedaria de nuevo como entrada testeable.

1.9.2 * CYCLE COUNTER:

Este bloque es un divisor por 5 que es necesario para crear un ciclo maquina consistente en 5 periodos de reloj. En este bloque tambien se genera la señal ALE. Ver figura 14.A.

Como se ve en esta figura la señal ALE es generada entre el tercero y cuarto periodo de reloj.Esta señal siempre esta presente en la patilla ALE y es activa en su flanco de bajada.

La PSEN se genera cuando se accede a una memoria ROM externa y como sabemos es activa a nivel bajo.

RD y WR se generan cuando se hace una lectura o escritura en el bus respectivamente. Estas señales solo son utiles cuando se accede a una memoria o periferico externo.

El terminar PROG cuando no se utiliza para la programacion funciona como una señal de STROBE "dato preparado" para utililizar con perifericos como el 8243.

1.10 * LINEA DE RESET. CONSIDERACIONES.

La linea de reset del 8048 esta conectada a una puerta Trigger-Smith que inicializa el sistema. Ver fig 14.A.

El 8048 tan solo requiere un condensador de 1 uF para crear la constante de tiempo de reset ya que la resistencia viene integrada. Esta resistencia es de 200 Kohm.

La entrada de reset es compatible con TTL.

En un principio, tras dar alimentacion al microcomputador, la linea de reset debe permanecer a nivel bajo por lo menos 10 msg. hasta que la alimentacion se estabilize.

Una vez en pleno funcionamiento (alimentacion estabilizada) si se quiere aplicar un reset, es necesario que la linea de reset permanezna a cero, al menos, 12.5 us. para una frecuencia de 6 Mhz.

Tras un reseteo el sistema sufre los siguientes cambios:

1) El Contador de Programa se pone a cero.

2) El Stack Pointer se pone a Cero.

3) Se selecciona el banco de registro de trabajo 0 (RAM).

4) Se selecciona el banco de memoria 0 (ROM).

5) Pone el bus en alta impedancia exepto si la linea EA esta a cinco voltios.

6) Coloca los puertos 1 y 2 en modo entrada.

7) Deshabilita las interrupciones, tanto la externa (INT) como la interna (TIMER COUNTER).

8) Para la cuenta del TIMER COUNTER.

9) Pone a cero el flag de overflow del TIMER COUNTER.

10) Deshabilita la salida de reloj por el pin T0.

12) pone a cero el flag F0.

1.11 * EJECUCION PASO A PASO. SINGLE STEP.

Como sabemos, el 8048 posee una entrada que permite la ejecucion del programa paso a paso. Esta entrada es la linea SS la cual es activa a nivel bajo.

Cuando la linea SS se pone a cero, el procesador termina de ejecutar la instruccion en curso y manda el contenido del contador de programa (proxima instruccion a ejecutar) al BUS BUFFER y a los 4 bits menos significativos del PORT 2 (P20-P23). En esta condiciones el procesador se detiene, por tiempo indefinido, hasta que la linea SS es desactivada.

Una vez que se desactiva la linea SS, el programa corre libremente hasta que se vuelve a activar dicha linea.

Este proceso se realiza cada vez que se activa la linea SS. En realidad la patilla SS es una linea de parada del procesador que detiene la ejecucion del programa en el ciclo FETCH (Busqueda) de la instruccion.

La figura 15.A muestra un cronograma de funcionamiento de la linea Single Step. A continuacion pasamos a describir dicho proceso.

1) El procesador recibe la orden de parada aplicando un cero a la linea SS.

2) El procesador responde deteniendo el programa en el ciclo de busqueda de la siguiente instruccion, presentando a continuacion el contenido del contador de programa en el bus.

NOTA: Los 8 bits menos significativos del C.P. son mandados al puerto 0 (BUS) y los 4 restantes son mandados al puerto 2 (P20-P23).

3) El procesador se da por enterado de la parada poniendo a uno la linea ALE durante el tiempo que la linea SS permanezca a nivel bajo. Mientras SS = LOW, en los puertos 0 y 2 estara presente la direccion de la proxima instruccion a ejecutar.

4) Una vez que SS se pone a uno, el procesador continua la ejecucion del programa, la señal ALE vuelve a su modo natural, y el contenido de los puertos 0 y 2 se pierde.

Para poder ejecutar el programa instruccion por instruccion, es necesario que la la linea SS pase a cero tan pronto como la señal ALE tambien pase a cero.(Estado S4 de reloj). Esto se puede conseguir mediante el circuito mostrado en la figura 16.A.

Con este circuito se consigue que el procesador realize una sola instruccion cada vez que se pulsa el pulsador. El funcionamiento del circuito es el siguiente:

La señal SS se genera por medio de un flip-flop tipo D con entradas de PRESET (Puesta a uno) y de CLEAR (Puesta a cero), donde el Preset tiene preferencia sobre el Clear.

Mediante dos puerta NAND se implementa un flip-flop SR cuya salida esta a uno cuando el pulsador esta en estado de reposo.

El conmutador que aparece en la figura permite seleccionar la ejecucion paso a paso o ejecucion libre segun este en la posicion 1 y 2 respectivamente.

En la posicion 1 el preset queda desactivado (a uno) y el control de la bascula pasa a la entrada clear.Esta entrada esta unida a ALE como vemos en la figura.

Cuando ALE esta a nivel bajo, Clear tambien lo esta , con lo que la salida del biestable pasa a cero, activando la linea SS del procesador.

En esta condiciones el programa se detiene y el procesador pone a uno la salida ALE. Esto hace que se desactive la linea Clear.

En esta condiciones, si se presiona el pulsador, se produce un pulso en la entrada de reloj de la bascula, que hace que el nivel alto de la entrada pase a la salida.

Este uno a la salida de la bascula, permite que el programa corra hasta que la señal ALE pasa a cero (Estado S4 de reloj), en cuyo instante el procesador se para pues la salida de la bascula vuelve a cero por medio de la entrada clear.

De este modo, solo se ejecuta una instruccion cada vez que se presiona el pulsador.

En la posicion 2 del conmutador el Preset queda activado continuamente, esto hace que la salida de la bascula D este a uno, y por tanto el programa corre libremente. En esta condiciones el pulsador de Single Step queda anulado.

1.12 * MODO POWER DOWN.

Los microcomputadores 8048, 8049, 8039, y 8035L tienen la capacidad de poder desconectar su alimentacion, sin que la memoria RAM pierda su contenido. Esto es lo que se denomina modo "Power Down".

La forma de conseguir esto es muy secilla. Como vemos todos estos microcomputadores son de memoria ROM, es decir que vienen programados de fabrica, y por tanto no necesitan la entrada VDD para ser programados.

Lo que ha hecho INTEL es utilizar dicha entrada para alimentar la RAM a parte de todo el sistema. De esta forma se puede desconectar la alimentacion (VCC) y mantener la RAM alimentada por medio de VDD.

Mediante este sistema el consumo se reduce entre un 85 y 90 por ciento.

La figura 17.A muestra el cronograma del Modo "Power Down". Como se ve en la figura es necesario activar el reset antes de desconectar la alimentacion (Vcc) para evitar alguna posible alteracion de la memoria RAM.

Tambiel es necesario guardar el contenido del contador de programa y la PSW en la pila, ademas de los parametros que se estimen necesarios.

Hay que crear una pequeña subrutina que realize todo lo anterior y que comunique a un circuito exterior que el procesador se encuentra preparado para entrar en modo "Power Down".Para mas informacion sobre este modo consultar el manual MCS-48 de Intel.

1.13 * ACCESO A MEMORIA EXTERNA.

La memoria de programa del 8048 es direccionada por la CPU de forma interna, es decir, sin que las direcciones o datos que existen en el bus salgan al exterior.

La MCS-48 esta implementada de tal forma que la memoria de programa comienza a direccionarse en la posicion 000H de memoria, dicho de otra forma que la memoria de programa interna esta al principio.

La entrada EA permite al usuario desactivar esta memoria interna y obligar a la CPU a realizar los ciclos de busqueda en una memoria ROM externa.

Aplicando 5 voltios a la linea EA la CPU realiza los ciclos de busqueda a travez de BUS externo y los 4 bits menos significativos del port 2.

Esto permite sustituir de forma total la memoria de programa interna por una externa. Esto es muy util a la hora de diseño y testeo de sistemas.

INPORTANTE: La conmutacion de la linea EA debe hacerse con la linea de reset activada para evitar que el sistema se caiga o estropee.

En los modelos que no poseen memoria interna de programa (8039, 8035...) la linea EA siempre debe estar a 5 voltios.

En los modelos con ROM/EPROM interna la linea EA suele estar a masa (Desactivada) a no ser que se desee invalidar la memoria interna de programa.

La line EA ademas de utilizarse para forzar los ciclos de busqueda internos-externos tambien permite "leer " el contenido de la memoria de programa interna. Esto se vera mas adelante.

1.14 * 8047. PROGRAMACION Y VERIFICACION.

Como sabemos, el 8047 es el unico microcomputador de la MCS 48 que posee una memoria EPROM y por tanto es el que se utiliza en el diseño de sistema.

El proceso de programacion es muy sencillo y consiste en los siguientes pasos:

1) Aplicar una direccion y memorizarla.

2) Aplicar un dato o instruccion y memorizarla.

3) Aplicar un impulso de programacion. (PROG).

4) Verificacion de lo grabado.

5) Volver al paso 1.

La figura 18.A muestra el diagrama de tiempos del proceso de programacion.

Las patillas que intervienen en el proceso de programacion son las siguientes:

XTAL1, XTAL2 : Entradas de reloj

RESET :Esta linea no solo se utiliza en la inicializacion del sistema, sino que ademas se utiliza para dar la orden de memorizacion de las direcciones.

RESET = 0 : Inicializacion del sistema.

RESET = 1 : Memorizacion de las direcciones.

TO : Esta linea permite seleccionar uno de los siguientes modos:

TO = 0 : Modo PROGRAMACION.

TO = 1 : Modo VERIFICACION.

EA : Esta linea cuando se pone a 23 voltios permite que el 8748 se pueda programar o verificar. Viene a ser algo asi como la llave del proceso de programacion.

BUS BUFFER : Este puerto se utiliza para introducir las direcciones y datos a programar ademas de utilizarse como salida de datos en el modo verificacion.

P20-P21 : Estas dos lineas del puerto 2 se utilizan para introducir los dos bits mas significativos del bus de direcciones y poder, de esta forma, seleccionar toda la memoria del 8047 (1 K Byte).

VDD : Esta linea se utiliza para alimentar los circuitos de programacion internos del 8048. Su valor en el proceso de programacion es de 25 voltios.

PROG : Esta linea se utiliza para dar la orden de programacion. Esta orden consiste en un pulso de 23 voltis de amplitud y una duracion de 50 ms.

A continuacion pasamos a describir la secuencia de programacion segun el cronograma de la figura 18.A

Las condiciones iniciales son :

- VDD = 5 voltios.
- Reloj aplicado.

- Reset = 0 voltios.

- T0 = 5 voltios.;En este estado el Bus queda flotante.

- EA = 5 voltios.

La secuencia es la siguiente:

0) T0 = 0 V. Se selecciona el modo programacion.

1) EA = 23 V. Se activa el proceso de programacion.

2) Se aplica la direccion a programar al bus bufer y a las lineas P20-P21

3) RESET =5V. Orden de memorizacion de las direcciones.

4) Se aplica el byte a ser programado al bus.

5) La linea PROG se lleva a 0 voltios.

6) La linea VDD se pone a 25 voltios. Alimentacion de programacion.

7) Se manda a la linea PROG un pulso de 23V y 50 msg. En este momento se produce la programacion.

8) VDD= 5V. La alimentacion de los circuitos de programacion vuelve a su estado de reposo.

9) T0 = 5V. Se selecciona el modo de verificacion. aqui se octiene por el bus, de forma automatica, el dato programamado anteriormente.

10) T0 = 0V. Seleccion del modo programacion.

11) RESET =0V. Puesta del bus en alta impedancia. Esto permite colocar en el bus la proxima direccion si que se produzca conflicto en el bus.

12) Saltar al paso 2.

1.14.1 * BORRADO DE LA EPROM.

El borrado de la Eprom del 8748 se raliza mediante luz cuya longitud de onda sea menor de 4000 Angston. Por poner un ejemplo la luz sola y de fluorescentes tienen una longitud de onda comprendida entre 3000 y 4000 A.

Con luz fluorescente, el borrado tardaria unos tres años, mientras que con luz solar seria de un año. Naturalmente no podemos esperar tanto tiempo.

De todos modos para evitar que se pueda borrar por accidente, conviene tapar la "ventanilla" del 8047 con un material opaco, una vez que se halla grabado la memoria.

El borrado de la eprom se debe hacer con una luz ultravioleta cuya longitud de onda dea de unos 2500 A.

El tiempo que tarda la memoria en ser borrada depende de la intensidad de la fuente luminosa en gran medida. Como ejemplo decir que con una lampara de 1200 uW/cm2 el tiempo de borrado es de unos 15-20 minutos.

Decir como dato importante que tras borrar la memoria del 8748 todos los bits quedan a cero.

1.15 * LECTURA DE LA MEMORIA INTERNA DE PROGRAMA.

Como vimos en el apartado anterior es posible "leer" el contenido de la ROM / EPROM interna del 8048 utilizando el modo verificacion del proceso de programacion.

Dicho de otra forma, el usuario puede verificar el contenido de la memoria de programa del 8048, sin necesidad de que la CPU intervenga para nada.

El proceso de lectura es el siguiente: Ver figura 19.A.

1) RESET = 0 ;Inicializacion de la CPU.

2) T0 = 5V. Seleccion del modo verificacion en el 8748. En los modelos sin EPROM no es necesario poner esta linea a 5V pues en dichos modelos no existe el modo programacion, solo existe el modo verificacion y por tanto T0 no se utiliza para la seleccion de modos.

3) Activacion del proceso de verificacion.

EA= 23 V. Para el 8047.

EA= 12 V. Para los modelos con ROM.

4) Colocar en el bus buffer y P20-P21 la direccion de memoria a "leer".

5) RESET = 1 : Orden de memorizacion de la direccion presente a la entrada del bus. La puesta a uno del reset hace que el contenido del bus sea cargado en el contador de programa. De forma automatica, el contenido de la direccion de memoria señalada por el C.P. es presentado a la salida del bus.

La figura 20.A muestra un circuito que permite la lectura de la ROM interna. A continuacion pasamos a describir dicho circuito.

Los 8216 son buffers bidireccionales que permiten encausar la salida de datos y entradas de direcciones. Mediante la patilla DIEN se selecciona el sentido de la transferencia.

Si DIEN = 0 ; las direcciones entran a travez de I0-I4.

Si DIEN = 1 ; los datos sale a travez de D0-D4.

La bascula tipo D (7474) garantiza que la puesta a "uno" o "cero" del reset se produzca justo en el flanco de subida de la señal ALE.

Por ultimo, antes de desconectar la linea EA es necesario poner el reset a cero para evitar que se produzcan daños en el microcomputador.

2.1 - REPERTORIO DE INSTRUCCIONES DEL 8048.

2.1.1 - INTRODUCCION.

En este capitulo vamos a estudiar las instrucciones del 8048 a fondo, asi como la transferencia de datos entre los registros que componen el 8048.

Todas las instrucciones explicadas en este capitulo sirven para cualquiera de los diferentes microcomputadores de la familia MCS-48 salvo que se indique lo contrario.

Todas las instrucciones ocupan 1 o 2 octetos como maximo, siendo el 70 por ciento de las instrucciones de solo un octeto.

Todas las instrucciones se realizan en 1 o 2 ciclos maquina como maximo, o lo que es lo mismo en 5 y 10 periodos de reloj respectivamente. El 50 % de dichas instrucciones se ejecutan en un solo ciclo maquina.

2.2 - TRASFERENCIA DE DATOS.

En el 8048 el acumulador es el lugar de paso de la mayoria de las transferencias de datos.

La transferencia entre el acumulador y los registros del banco de trabajo es "directa" y el registro seleccinado es especificado por la propia instruccion.

Por ejemplo:

```
MOV A,R0 ; LLevar a A el contenido de R0.
MOV A,R1 ; LLevar a A el contenido de R1.
```

La transferencia entre la RAM y el acumulador es de forma "indirecta" direccionada por los registros R0 y R1 del banco de trabajo. Por ejemplo:

```
MOV @ R0,A ; Transferir el contenido de A a R0.
```

Nota: El simbolo " @ " se utiliza par referirse a los direccionamientos indirectos.

Los registros R0 y R1 tambien se pueden utilizar para direccionar una memoria RAM externa.

IMPORTANTE : El direccionamiento de RAM externa requiere dos ciclos maquina, mientras que el interno tan solo requiere uno.

El contenido de las posiciones de la ROM pueden ser transferidos directamente al acumulador y a los registros de trabajo.

La PSW puede ser cargada y modificada por el acumulador de forma directa. Por ejemplo:

```
MOV PSW,A ; Transferir el contenido de A a la PSW.

MOV A,PSW ; Transferir el contenido de la PSW a A.
```

Los registros del banco de trabajo pueden ser cargados de forma inmediata sin necesidad de recurrir al acumulador.

Por ejemplo:

MOV R,# dato : Cargar R con el dato

Nota: El simbolo " # " es usado para indicar un direccionamiento inmediato.

Tambien, los registros de trabajo pueden ser incrementados y decrementados sin necesidad de utilizar el acumulador.

2.1.3 - OPERACIONES CON EL ACUMULADOR.

En el acumulador se pueden hacer las siguientes operaciones:

- Sumas, con y sin acarreo.
- Operaciones logicas: AND, OR y ORX
- Se puede intercambiar el contenido de cualquier posicion de memoria RAM con el acumulador.
- Se puede intercambiar los 4 bits menos significativos del acumulador con los 4 bits menos significativos de cualquier posicion de la memoria RAM. Esto es ideal para trabajar con numeros en BCD.
- Existe una instruccion que permite intercambiar los cuatro bits menos significativos del acumulador con los cuatro mas significativos de este.Esta instruccion

supone una comodidad en operaciones BCD.

- El acumulador puede ser incrementado y decrementado.
- El acumulador puede ser borrado y complementado.
- En el acumulador se pueden hacer desplazamientos a la derecha y a la izquierda ,tanto a travez del carry como fuera de este.
- En el 8048 no existe la substraccion. Esta se consigue sumando al minuendo el complemento del substraendo y el resultado se vuelve a complentar.

2.1.4 - INDICADORES. fLAGS.

En el 8048 existen cuatro indicadores accesibles por el usuario. Son :

CY : Acarreo en las operaciones con el acumulador.

CA : Acarreo auxiliar. Solo en operaciones BCD.

F0 : Flag F0 de uso general.

F1 : Flag F1 de uso general.

Los tres primeros indicadores estan presentes en la PSW y son guardados en la pila cuando se accede a una subrutina.F1 no esta dentro de la pila y por tanto no hay forma de conservar su valor cuando se salta a una subrutina.

Estos indicadores pueden ser puestos a cero, complementados y testeados mediante instrucciones de salto condicional.

Por ultimo decir que los indicadores F0 y F1 no tienen nada que ver con las entradas testeables T0 y T1.

2.1.5 - INSTRUCCIONES DE BIFURCACION.

Como sabemos el contador de programa del 8048 permite direccionar hasta 4 Kbytes de memoria. Estos 4 Kbytes se dividen en dos bancos de memoria cada uno de 2 Kbytes.

El banco de memoria seleccionado depende del valor del bit mas significativo del contador de programa (CP11).Este bit se pone a cero o a uno mediante dos instrucciones, nunca por el incremento natural del contador de programa.

2.1.6 - SALTO INCONDICIONAL.

Las instruccion de salto incondicional permite el salto a cualquier posicion de memoria dentro del banco de memoria en que se este trabajando. Dicho de otra forma esta instruccion permite un salto de 2Kbytes como maximo. La razon es que el bit CP11 no es modificable mediante esta instruccion.

Si queremos saltar a una posicion del banco de memoria no seleccionado, primero es necesario seleccionar dicho banco y luego seleccionar la posicion mediante una instruccion de salto.

Para seleccionar uno u otro banco de memoria se utilizan las siguientes instrucciones:

SEL MB0 : Seleccion del banco de memoria 0.

SEL MB1 : Seleccion del banco de momoria 1.

Tras un reset se selecciona el banco 0.

El cambio de banco de memoria tiene lugar solo cuando se ejecuta una instruccion de salto. En otras palabra, la ejecucion de la instruccion SEL MB se realiza solo cuando mas adelante existe una instruccion de salto. Por esta razon es conveniente poner ambas instrucciones seguidas para evitar posibles errores.

Una vez se salta al otro banco de memoria, el programa continua ejecutandose en dicho banco y los siguientes saltos seran en este banco, hasta que una nueva instruccion SEL MB seguida de una de salto es ejecutada.

El salto a una subrutina que este ubicada en el banco contrario se realiza mediante una instruccion SEL MB seguida de la instruccion de llamada a subrutina. (CALL).

Una vez ejecutada la subrutina, se retorna al banco original, pues el bit CP11 tambien se almacena en la pila.

Por ultimo decir que existe una instruccion de salto incondicional, indirecto, direccionado por el acumulador.

2.1.7 - SALTOS CONDICIONALES.

Los saltos condicionales pueden examinar las siguientes entradas y estados internos:

- Entrada de testeo T0.
- Entrada de testeo T1.
- Entrada de testeo INT.
- Acumulador a cero.
- Cualquier bit del acumulador.
- Indicador de acarreo.
- Indicador F0.
- Indicador F1.

Con los saltos condicionales se puede acceder a cualquier posicion dentro de una pagina de 256 bits. Esto es porque en esta clase de saltos tan solo se dispone de un octeto para direccionar la memoria ya que el otro octeto es el codigo de la instruccion.

Cuando se ejecuta una instruccion de salto condicional, la direccion de salto, (octeto), es cargada en el byte menos significativo del contador de programa. Los restantes bits del contador de programa son conservados. Esto permite saltar a cualquier posicion de la pagina dada por CP8 - CP11.

2.1.8 - SUBRUTINAS.

El 8048 solo dispone de una instruccion de salto a subrutina. Esta es la instruccion CALL, que como vemos es una llamada a subrutina incondicional.

Como vimos en el apartado anterior se puede acceder, mediante esta instruccion, a cualquier posicion de memoria dentro del banco de memoria (2K bytes).

Para cambiar de banco de memoria se recurre a la instruccion SEL MB.

El retorno de la subrutina es al banco original donde se produjo la llamada, pues el bit CP11 del contador de programa tambien fue salvaguardado en la pila cuando se produjo el salto.

Como nuevo, decir que existen dos instrucciones de RETURN. Una reestablece la PSW tras el retorno y la otra no.

2.1.9 - INSTRUCCIONES DEL TEMPORIZADOR.

Como sabemos, el temporizador puede ser cargado y leido por medio del acumulador ya sea funcionando o parado.

Existe dos instrucciones de marcha distintas, una se utiliza para arracar en modo COUNTER y la otra para arrancar en modo TIMER.

Existe una unica instruccion de parada comun a los dos modos. Ademas existe una instruccion que habilita la interrupcion del TIMER/COUNTER y otra que la inhnibe.

2.1.10 - INSTRUCCIONES DE CONTROL.

Existe una instruccion para habilitar la linea de interrupcion externa (INT) y otra para inhnibirla.

Tras un reset las interrupciones quedan inhnibidas. Es necesario pues habilitarlas tras el reset si se van a utilizar.

Cuando se esta ejecutando una subrutina de ejecucion el procesador inhnibe, de forma automatica las interrupciones, volviendolas a habilitar una vez se ejecuta la instruccion return.

Existen cuatro instrucciones de seleccion de memoria. Dos para seleccionar uno de los dos bancos de registros de trabajo de la RAM, y otras dos para seleccionar uno de los dos bancos de memoria de la ROM.

Existe una instruccion para habilitar la patilla TO como salida de reloj. En caso de que se desee inhnibir la salida de reloj por TO es necesario hacer un reset.

2.1.11 - INSTRUCCIONES DE ENTRADA Y SALIDA.

Los puertos 1 y 2 pueden ser leidos y cargados por medio del acumulador, mediante dos instrucciones respectivamente. Teniendo en cuenta, que la salida se almacena en los lach de salida sin embargos, la entrada no.

Existen instrucciones que permiten hacer operaciones logicas entre los puertos y cualquier posicion de la memoria de programa directamente, y el resultado permanece en el puerto (lachs).

En cuanto al puerto 0 (BUS BUFFER) ,decir que puede hacer lo mismo que los puertos 1 y 2 exepto programar las lineas de un puerto, con entradas y salidas a la vez.

En este puerto, las lineas en un momento dado, son todas entradas o todas salidas.

Cuando este puerto se utiliza como bus funciona de forma totalmente bidireccional generando las señales WR y RD.

Las lineas P20-P23 del port 2 se utilizan para los cuatro bits mas significativos del bus cuando el puerto 0 se esta utilizando como bus.

2.2 - REPERTORIO DE INSTRUCCIONES.

En las siguientes paginas se describe las instrucciones de la MCS-48 detalladamente. En la figura 2.B se muestra una lista de las instrucciones agrupadas de forma funcional. En este apartado vamos a describir una a una cada instruccion en orden alfabetico.

La lista alfabetica de las instrucciones incluye:

- Nombre nemotecnico (Ensamblador).
- Codigo maquina.
- Descripcion verbal de la instruccion.
- Descripcion simbolica.
- Ejemplo en lenguaje ensamblador.

A continuacion se da una lista de los simbolos y abreviaciones utilizados en la descripcion de las instrucciones.

A	Acumulador.
CA	Carry auxiliar.
addr	Direccion de 12 bits (Memoria de programa)
Bd	Designador de uno de los 8 bit de un byte.
BS	Swiches de banco.
BUS	Port 0. Bus buffer.
C	Carry.
CLK	Clock.
CNT	Contador interno.
CRR	Registro de conversion de resultados.
D	Digito de 4 bits.
data	numero o expresion de 8 bits.
DBF	Flip flop del banco de memoria.
F0,F1	Flag 0 y flag 1 respectivamente.
I	Interrupcion.
P	Operacion en pagina.
PC	Contador de programa.
Pp	Designador del puerto.(p=1,2 o 4-7).
PSW	Palabra de estado del procesador.
Rr	Designador de registro. (r=0,1 o 0-7).
SP	Stack pointer.
T	Timer interno.
TF	Timer flag.
T0,T1	Entradas T0 y T1.
X	Nemonico de memoria RAM externa
#	Prefijo de dato inmediato.

@	Prefijo de direccionamiento indirecto.
$	Valor actual del Contador de programa.
(X)	Contenido de X.
((X))	Contenido de la posicion direccionada por X.

Sin mas preambulo, pasamos a describir las instrucciones de la familia MCS-48.

ADD A,Rr : Sumar el cotenidodel registro Rr a A.

0 1 1 0 1 r r r

El contenido de uno de los 8 registros del banco de trabajo es sumado al acumulador. EL carry es afectado.

(A) -- (A) + (Rr) r= 0-7

ADD A, @Rr : Sumar a A una posicion de la RAM.

0 1 1 0 0 0 0 r

El contenido de una posicion de memoria RAM direccionada por uno de los registros R0, R1 es sumado al acumulador. El carry es afectado.

(A) -- (A) + ((Rr)) r=0-1

ADD, #data : Sumar a A un dato inmediato.

0 0 0 0 0 0 1 1 d d d d d d d d

El dato especificado (1 byte) es sumado al acumulador.Esta instruccion es de 2 ciclos. El carry es afectado.

(A) -- (A) + dato

ADDC A, Rr : Sumar a A el registro Rr y el carry.

0 1 1 1 1 r r r

El contenido del bit de carry es sumado al acumulador y luego puesto a cero. A continuacion el registro Rr es sumado al acumulador. El carry es afectado.

(A) -- (A) + (Rr) + (C) r=0-7

ADDC A, @Rr : Sumar a A con carry una posicion de la RAM.

0 1 1 1 0 0 0 r

El contenido del bit de carry es sumado al acumulador y luego es puesto a cero. A continuacion el contenido de la posicion direccionada por r es sumada al acumulador. El carry

es afectado.

(A) -- (A) + ((Rr)) + (C) r=0-1

ADDC A, #data : Sumar a A con carry un dato inmediato.

0 0 0 1 0 0 1 1 d d d d d d d d

Esta es una instruccion de 2 ciclos. EL contenido del bit de carry es sumado al acumulador y luego puesto a cero. A continuacion se suma al acumulador el dato especificado. EL carry es afectado.

(A) -- (A) + data + (C)

ANL A, Rr : Operacion logica AND entre Rr y A.

0 1 0 1 1 r r r

Se hace una operacion AND entre el acumulador y el registro Rr especificado. El resultado permanece en el acumulador. Ningun flag es afectado.

(A) -- (A) AND (Rr) r=0-7

ANL A, @Rr : Logica AND entre A y memoria.

0 1 0 1 0 0 0 r

Se hace una operacion AND entre el acumulador y una posicion de memoria RAM direccionada por Rr

(A) -- (A) AND ((Rr)) r=0-1

ANL A, #data : Logica AND entre A y un dato inmediato.

0 1 0 1 0 0 1 1 d d d d d d d d

Se hace una operacion AND entre el acumulador y un dato inmediato. Esta es una instruccion de 2 ciclos.

(A) -- (A) AND dato.

ANL BUS, #data : Logica AND entre BUS y un dato inmediato

1 0 0 1 1 0 0 0 d d d d d d d d

Se hace una operacion AND entre el puerto 0 (BUS) y un dato inmediato y el resultado permanece en el puerto. Esta es una instruccion de 2 ciclos.

(BUS) -- (BUS) AND data

ANL Pp, #data : Logica AND entre un puerto y un dato.

1 0 0 1 1 0 p p d d d d d d d d

Se hace una operacion AND entre el puerto 1 ó 2 y un dato inmediato. El resultado permanece en el puerto. Esta es una instruccion de 2 ciclos.

(Pp) -- (Pp) AND dato p=1-2

ANL Pp, A : Logica AND entre un puerto periferico y A.

1 0 0 1 1 1 p p

Se hace una operacion AND entre uno de los 4 puertos del periferico 8243 y el acumulador (bits 0-3). El resultado permanece en el puerto. Los puertos del 8243 son de 4 bits cada uno y estan numerados del 4 a 7. Esta instruccion es de 2 ciclos.

(Pp) -- (Pp) AND (A)

pp	Puerto
00	4
01	5
10	6
11	7

La tabla de la derecha muestra el valor de los bits pp para direccinar cada uno de los puertos.

Para mayor informacion ver el periferico 8243.

CALL address : LLamada a subrutina.

a a a 1 0 1 0 0 a a a a a a a a

Esta es una instruccion de 2 ciclos. Los bits a0 - a10 son para especificar la direccion de salto dentro de la ROM.

Esta instruccion, cuando se ejecuta, realiza el siguiente proceso.

- Guarda el C.P. en la pila junto con la PSW (Bits 4-7).
- Incrementa el stack pointer (bits 0-2 de la PSW).
- Carga en el C.P. la direccion especificada.

La instruccion CALL no puede colocarse en las direcciones 2046-2047 ni en la 4094-4095 por ser las ultimas direcciones de cada bloque de memoria.

((SP)) -- (CP), (PSW 4-7)

(SP) -- (SP) +1

(CP 8-10) -- (addr 8-10)

(CP 0-7) -- (addr 0-7)

(CP11) -- DBF

CLR A : Limpiar el acumulador.

0 0 1 0 0 1 1 1

El contenido del acumulador es puesto a cero.

A -- OOH

CLR C : Limpiar el carry.

1 0 0 1 0 1 1 1

El contenido del flag de carry es puesto a cero.

C -- 0

CLR F1 : Limpiar el flag F1.

1 0 1 0 0 1 0 1

El contenido del flag F1 es puesto a cero.

(F1) -- 0

CLR F0 : Limpiar el flag F0.

1 0 0 0 0 1 0 1

EL contenido del flag F0 es puesto a cero

(F0) -- 0

CPL A : Complementar el contenido del acumulador.

0 0 1 1 0 1 1 1

El contenido del acumulador es complementado. Es extrictamente complemento a uno donde los unos son cambiados por cero y viceversa.

(A) -- NOT (A)

CPL C : Complementar el bit de CARRY.

1 0 1 0 0 1 1 1

El flag de carry es complementado. Se pone a uno cuando esta a cero y viceversa.

(C) -- NOT (C)

CPL F0 : Complementar el flag F0.

1 0 0 1 0 1 0 1

El flag F0 es complementado. Se pone a uno cuando esta a cero y viceversa.

F0 -- NOT (F0)

CPL F1 : Complementar el flag F1.

1 0 1 1 0 1 0 1

El flag F1 es complementado. Se pone a uno cuando esta a cero y viceversa.

DAA : Ajuste decimal del acumulador.

0 1 0 1 0 1 1 1

Los 8 bits del acumulador son ajustados para formar dos numeros de 4 bits en codigo BCD de la siguiente forma.

El procesador analiza los 4 LSB del acumulador y si el contenido es superior a nueve o el CA (carry auxiliar) esta a uno, le suma seis.

A continuacion, analiza los 4 MSB del acumulador y si el contenido es superior a seis o el CY esta a uno, le suma seis.

El CA se produce cuando al sumar dos numeros, ya puestos en BCD se produce un acarreo del cuarto al quinto bit del acumulador.

La instruccion DDA se utiliza siempre que se desee convertir el contenido del acumulador en un numero BCD.

DEC A : Decrementar el acumulador.

0 0 0 0 0 1 1 1

El contenido del acumulador es decrementado en una unidad.

(A) --(A) - 01H

DEC Rr : Decrementar el registro Rr.

1 1 0 0 1 r r r

El contenido del registro de trabajo especificado es decrementado en una unidad.

(Rr) -- (Rr) - 01H r=0-7

DIS I : Deshabilitar la interrupcion externa.

0 0 0 1 0 1 0 1

Esta instruccion deshabilita la linea de interrupcion externa INT. Un nivel bajo en esta patilla no tiene efecto.

DIS TCNTI : Deshabilitar la interrupcion del TIMER/COUNT.

0 0 1 1 0 1 0 1

Esta instruccion deshabilita la linea de interrupcion del TIMER/COUNTER. Aunque se produzca un overflow en el contador este es ignorado por la CPU.

DJNZ Rr, address : Decremento de Rr y testeo.

1 1 1 0 1 r r r a a a a a a a a

Esta es una instruccion de 2 ciclos. El registro Rr es decrementado una unidad, y testeado su contenido. Si el registro Rr es cero, el programa sigue su curso.

Si el contenido del registro es distinto que cero, el programa salta a la direccion especificada (a0-a7).

Se ve que el salto es paginado, es decir que podemos saltar a cualquiera de las 256 posiciones de la pagina en curso.

NOTA: si esta instruccion se encuentra en la ultima posicion de una pagina (255), el salto sera dentro de la siguiente pagina. La razon de esto es que el contador de programa se incrementa en una unidad tras la ejecucion de la instruccion.

(Rr) -- (Rr) - 1 r=0-7

(CP) -- (CP) +1

SI Rr no es cero entonces: (CP0-7) -- a0-a7

EN I : Habilitacion de la interrupcion externa.

0 0 0 0 0 1 0 1

Esta instruccion habilita la linea INT de interrupcion exterior. Esta instruccion no se encuentra en el 8021.

EN TCNTI : Habilitacion de la interrupcion del TIMER/COUNT

0 0 1 0 0 1 0 1

La interrupcion del TIMER/COUNTER es habilitada. Un overflow en el Timer /counter inicia la secuencia de interrupcion.Esta instruccion no se encuentra en el 8021.

ENTO CLK : Habilitacion de la salida de reloj.

0 1 1 1 0 1 0 1

La patilla TO es habilitada para actuar como salida de reloj. Esta funcion es dehabilitada unicamente mediante un reset del sistema. Esta instruccion no se encuentra en el 8021 ni en el 8022.

IN A, Pp : Entrada de un puerto al acumulador.

0 0 0 0 1 0 p p

Esta es una instruccion de dos ciclos. El dato presente en el puerto Pp (1 ó 2) es transferido al acumulador.

En el 8021 el puerto 2 es de solo 4 bits, esto causa que los 4 bits del puerto sean colocados en las posiciones A0-A3 del acumulador, mientras que A4-A7 son puestas a cero.

(A) -- (Pp) p=1-2

INC A : Incrementar el contenido del acumulador.

0 0 0 1 0 1 1 1

El contenido del acumulador es incrementado en una unidad.

(A) -- (A) + 1

INC Rr : Incrementar el contenido del registro Rr.

0 0 0 1 1 r r r

El contenido del registro de trabajo especificado es incrementado en una unidad.

(Rr) -- (Rr) + 1 r=0-7

INC @Rr : Incrementar una posicion de la RAM.

0 0 0 1 0 0 0 r

El contenido de la posicion de memoria direccionada por el registro Rr (0 ó 1) es incrementado en una unidad.

NOTA: Hay que tener en cuenta la cantidad de memoria RAM que podemos direccionar segun el microcomputador

((Rr)) -- ((Rr)) +1 r=0-1

IN A, P0 : Entrada del puerto 0 al acumulador.

0 0 0 0 1 0 0 0

Esta instruccion solo la tienen los microcomputadores 8021 y 8022. Es semejante a la instruccion IN A,BUS exepto que la señal RD no se genera.

INS A,BUS : Lectura del bus.

0 0 0 0 1 0 0 0

Esta es una instruccion de dos ciclos. El dato presente en el bus port es transferido (leido) al acumulador cuando la señal RD pasa a nivel bajo.

(A) -- (BUS)

JBb address : Salto si el bit " b" de A esta a uno.

b b b 1 0 0 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Si el bit Bb del acumulador esta a "uno" el programa salta a la posicion especificada en la instruccion (a0-a7). Este es un salto paginado a 256 bits.

Si Bb= 1 entonces : (CP0-7) -- a0-a7

Si Bb= 0 entonces : (CP) -- (CP) + 2

b=0-7

JC address : Salto si el carry esta a uno.

1 1 1 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Si el bit de carry esta a "uno" el programa salta ala direccion especificada. Este es un salto paginado a 256 bits.

Si C=1 entonces : (CP0-7) -- a0-a7

si C=0 entonces : (CP) -- (CP) + 2

JF0 address : Salto si el flag F0 esta a uno.

1 0 1 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Si el flag F0 esta a "uno" el programa salta a la direccion especificada. Este es un salto paginado a 256 bits.

(CP0-7) -- a0-a7 si F0=1

(CP) -- (CP) + 2 si F0=0

JF1 address : Salto si el flag F1 esta a uno.

0 1 1 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Si el flag F1 esta a "uno" el programa salta a la direccion especificada. Este es un salto paginado de 256 bits.

(CP0-7) -- a0-a7 si F1=1

(CP) -- (CP) + 1 si F1=0

JMP address : Salto incondicinal.

a a a 0 0 1 0 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Los bits 0-10 del contador de programa son reemplazados por la direccion especificada en la instruccion (a0-a10).

El valor del bit CP11 es determinado por la mas reciente instruccion SEL MB.

(CP8-10) -- a8-a10

(CP0-7) -- a0-a7

(CP11) -- DBF

JMPP @A : Salto incondicional indirecto.

1 0 1 1 0 0 1 1

Esta es una instruccion de dos ciclos. El contenido de la memoria de programa señalada por el acumulado es sustituido en el contador de programa (BITS 0-7).Este es un salto paginado.

(CP0-7) -- ((A))

JNC address : Salto si no hay carry.

1 1 1 0 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si el bit de carry esta a cero. Este es un salto paginado.

(CP0-7) -- a0-a7 si C=0

(CP) -- (CP) +2 si C=1

JNI address : Salto si la linea INT es cero.

1 0 0 0 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. Esta instruccion testea la entrada INT, si esta a cero salta a la direccion especificada. El salto es paginado a 256 bits.

NOTA: Para que esta instruccion se pueda ejecutar, es necesario que la linea INT este deshabilitada como interrupcion.

(CP0-7) -- a0-a7 si linea INT =0

(CP) -- (CP) +1 si linea INT =1

JNTO address : Salto si la linea T0 es cero.

0 0 1 0 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si la linea de test T0 esta a cero. Este es un salto paginado a 256 bits. Esta instruccion no se encuentra en el 8021.

(CP0-7) -- a0-a7 si linea T0 =0

(CP) -- (CP) +1 si linea T0 =1

JNT1 address : Salto si la linea T1 es uno.

0 1 0 0 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si la linea de test T1 esta a cero. Este es un salto paginado a 16 bits.

(CP0-7) -- a0-a7 si linea T1=0

(CP) -- (CP) +1 si linea T1=1

JNZ address : Salto si el acumulador es distinto de cero.

1 0 0 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de 2 ciclos. El programa salta a la direccion especificada si el contenido del acumulador es distinto de cero. Este es un salto paginado a 256 bits.

(CPO-7) -- a0-a7 si A distinto de cero.

(CP) -- (CP) +1 si A igual a cero.

JTF address : Salto si el flag de overflow esta a uno.

0 0 0 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de 2 ciclos. El programa salta a la direccion especificada si el flag de overflow del TIMER/CONTER esta a uno, esto es si se ha producido un overflow. Este es un salto paginado a 256 bits. Cuando se ejecuta esta instruccion el flag de overflow es puesto a cero automaticamente.

(CPO-7) -- a0-7 si TF =1

(CP) -- (CP) + 2 si TF =0

JTO address : Salto si la linea TO esta a uno.

0 0 1 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si la linea de test TO esta a uno. Este es un salto paginado.

(CPO-7) -- a0-a7 si linea TO =1

(CP) -- (CP) + 2 si linea TO =0

JT1 address : Salto si la linea T1 esta a uno.

0 1 0 1 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si la linea de test T1 esta a uno.

Este es un salto paginado.

(CP0-7) -- a0-a7 si linea T1 =1

(CP) -- (CP) +2 si linea T1 =0

JZ address : Salto si el acumulador es cero.

1 1 0 0 0 1 1 0 a a a a a a a a

Esta es una instruccion de dos ciclos. El programa salta a la direccion especificada si el contenido del acumulador es cero. Este es un salto paginado.

(CP0-7) -- a0-a7 si A es igual a cero

(CP) -- (CP) + 2 si A es distinto de cero.

MOV A, #data : Cargar el dato en el acunulador.

0 0 1 0 0 0 1 1 d d d d d d d d

Esta es una instruccion de dos ciclos. El dato de 8 bits especificado es cargado en el acumulador.

(A) -- data

MOV A, PSW : Cargar la PSW en el acumulador.

1 1 0 0 0 1 1 1

El contenido de Palabra de estado del procesador (PSW) es cargada en el acumulador.

(A) -- (PSW)

MOV A, Rr : Cargar el registro Rr en el acumulador.

1 1 1 1 1 r r r

El contenido del registro de trabajo especificado (R0-R7) es cargado en el acumulador.

(A) -- (Rr) r=0-7

MOV A, @Rr : Cargar el contenido de la RAM en A.

1 1 1 1 0 0 0 r

El contenido de la posicion de memoria RAM direccionada por el registro de trabajo (R0 ó R1) es cargado en el acumulador.

(A) -- ((Rr)) r=0-1

MOV A, T : Cargar el contenido del TIMER/COUNTER en A.

0 1 0 0 0 0 1 0

El contenido del contador del Timer/counter es cargado en el acumulador. Esta instruccion se ejecuta incluso con el contador en marcha.

(A) -- (T)

MOV PSW, A : Cargar el contenido de A en la PSW.

1 1 0 1 0 1 1 1

El contenido del acumulador es cargado en la PSW. Todos los flag y el stack pointer son afectados por esta instruccion.

(PSW) -- (A)

MOV Rr, A : Cargar el contenido de A en el registro Rr.

1 0 1 0 1 r r r

El contenido del acumulador es cargado en el registro de trabajo especificado.

(Rr) -- (A) r =0-7

MOV Rr, #data : Cragar el dato en el registro Rr.

1 0 1 1 1 r r r d d d d d d d d

Esta es una instruccion de dos ciclos. El dato especificado (d0-d7) es cargado en el registro de trabajo especificado (R0-R7).

(Rr) -- dato. r =0-7

MOV @Rr, A : Llevar el acumulador a la memoria de datos.

1 0 1 0 0 0 0 r

Esta es una instruccion de dos ciclos. El contenido del acumulador es almacenado en la posicion de memoria RAM direccionada por el registro R0 ó R1. Notese que para direccionar la RAM solo hacen falta 5 de los 8 bits.

((Rr)) -- (A) r =0-1

MOV @Rr, #data : Cargar el dato en la memoria de datos.

1 0 1 1 0 0 0 r d d d d d d d d

Esta es una instruccion de dos ciclos. El dato especificado (d0-d7) es almacenado en la posicion de memoria direccionada por R0 ó R1.

((Rr)) -- dato r =0-1

MOV T, A : Cargar el acumulador en el Timer/ Counter.

0 1 1 0 0 0 1 0

El contenido del acumulador es cargado en el contador interno del microcomputador.

(T) -- (A)

MOVD A, Pp : Cargar el puerto Pp (4-7) en el acumulador.

0 0 0 0 1 1 p p

Esta es una instruccion de dos ciclos.Esta instruccion solo se puede utilizar con el 8243.

El dato presente en uno de los puertos del 8243 (P4-P7) es llevado al acumulador (bits A0-A3).

Los bits A4-A7 del acumulador son puestos a cero.

(A0-A3) -- Pp

(A4-A7) -- 0 p =4-7

NOTA: La siguiente tabla muestra como se seleccinan los cuatros puertos del 8243.

p p	Puerto
0 0	4
0 1	5
1 0	6
1 1	4

MOVD Pp, A : Llevar el acumulador ar puerto Pp (4-7).

0 0 1 1 1 1 p p

Esta es una instruccion de dos ciclos. Esta instruccion solo se puede utilizar con el 8243.

El dato presente en el acumulador (bits A0-A3) son llevados al puerto seleccionado (P4-P7).

Ver la nota de la anterior instruccion.

(Pp) -- (A0-A3) p=4-7

MOVP A, @A : Cargar el acumulador con la pagina actual.

1 0 1 0 0 0 1 1

El contenido de la memoria de programa direccionada por el acumulador es cargado en el acumulador. Esta instruccion es de dos ciclos.

En el primer ciclo carga el contenido del acumulador en los 8 bits menos significativos del contador de programa.

En el segundo ciclo el contenido de la memoria direccionado por el contador de programa es cargado en el acumulador.

Tras esta operacion se reestablece el antiguo valor del contador de programa.

(CP0-CP7) -- (A)

(A) -- ((CP))

NOTA: Si esta instruccion se coloca en la posicion 255 de una pagina el direccionamiento se realizara en la siguiente pagina.La razon es que cuando la instruccion es llevada al DECODIFICADOR DE INSTRUCCIONES el contador de programa se incrementa en una unidad de forma automatica.

MOVP3 A, @A : Cargar el acumulador con la pagina 3.

1 1 1 0 0 0 1 1

Esta es una instruccion de dos ciclos. El contenido de la memoria de programa direccionada por el acumulador es cargado en el acumulador. Al contrario que la instruccion anterior la pagina es fija y es la numero tres.

(CP0-CP7) -- (A)

(CP8-CP11) -- 0011 :Seleccion de la pagina 3.

(A) -- (CP)

MOVX A, @Rr : LLevar el contenido de una RAM exterior a A

1 0 0 0 0 0 0 r

Esta es una instruccion de dos ciclos. El contenido de una posicion de memoria de datos externa direccionada por el registro Rr (R0 ó R1) es cargada en el acumulador.

(A) -- ((Rr)) r=0-1

MOVX @Rr, A : LLevar a A el contenido de una RAM exterior

1 0 0 1 0 0 0 r

Esta es una instruccion de dos ciclos. El contenido del acumulador es llevado a una posicion de memoria de datos externa direccionada por el registro Rr (R0 ó R1).

((Rr)) -- (A)

NOP : No operacion.

0 0 0 0 0 0 0 0

Esta instruccion no hace nada. El programa continua en la siguiente instruccion.

ORL A, Rr : OR entre A y una mascara registro Rr.

0 1 0 0 1 r r r

Esta instruccion realiza la operacion logica OR entre el acumulador y uno de los registros de trabajo seleccionado, (R0-R7).

EL resultado de la operacion permanece en el acumulador.

(A) -- (A) OR (Rr) r=0-7

ORL A, @Rr : OR entre A y una mascara de memoria.

0 1 0 0 0 0 0 r

Esta instruccion realiza la operacion logica OR entre el acumulador y una posicion de la memoria de datos direccionada por Rr (R0 ó R1).

El resultado de la operacion permanece en el acumulador.

(A) -- (A) OR ((Rr)) r=0-1

ORL A, #data : OR entre A y una macara inmediata.

0 1 0 0 0 0 1 1 d d d d d d d d

Esta es una instruccion de dos ciclos. Realiza una operacion logica OR entre el acumulador y el dato especificado. El resultado de la operacion permanece en el acumulador.

(A) -- (A) OR Data

ORL BUS, #data :OR entre BUS y una mascara inmediata.

1 0 0 0 1 0 0 0 d d d d d d d d

Esta es una instruccion de dos ciclos. Realiza un operacion logica OR entre el el PORT 0 y el dato especificado. El resultado de la operacion permanece en el puerto.

(BUS) -- (BUS) OR Data

ORL Pp, #data :OR entre el PORT y mascara inmediata.

1 0 0 0 1 0 p p d d d d d d d d

Esta es una instruccion de dos ciclos. Realiza la operacion logica OR entre el el puerto "p" (1 ó 2) y el dato especificado.

(Pp) -- (Pp) OR Data p=1-2

ORLD Pp, A : OR entre el PORT 4-7 y el acumulador.

1 0 0 0 1 1 p p

Esta es una instruccion de dos ciclos. Solo se puede utilizar con el 8243.

Esta instruccion realiza una operacion logica OR entre el acumulador (bits A0-A3) y uno de los puertos del 8243.

El resultado de la operacion permanece en el puerto.

(Pp) -- (Pp) OR (A0-A3) p=4-7.

OUTL P0, A : Mandar el contenido de A al puerto 0.

1 0 0 1 0 0 0 0

Esta instruccion manda el contenido del acumulador al puerto 0 del 8021 y 8022.

OUTL BUS, A : Mandar el contenido de A al BUS.

0 0 0 0 0 0 1 1

Esta es una instruccion de dos ciclos. El contenido del acumulador es transferido al puerto 0 (BUS) y lacheado. La informacion permanece en el puerto hasta que sea escrito de nuevo. La instruccion IN A,BUS no destruye la informacion presente en el Lach de salida de este puerto.

Las instrucciones MOVX, AND, OR con este puerto destruyen la informacion presente en este LACH.

(BUS) -- (A)

OUTL Pp, A : Manda el contenido de A al puerto 1 ó 2.

0 0 1 1 1 0 p p

Esta es una instruccion de dos ciclos. El contenido del acumulador es mandado al puerto Pp (1 ó 2) y lacheado.

(Pp) -- (A) p=1-2

RAD : LLevar el resultado de conversion a A (Solo 8022)

1 0 0 0 0 0 0 0

Esta es una instruccion de dos ciclos. El contenido del registro de conversion del convertidor analogico digital es cargado en el acumulador.

(A) -- (CRR)

RET : Return sin reestablecer la PSW.

1 0 0 0 0 0 1 1

Esta es una instruccion de dos ciclos. En el primer ciclo el stack pointer es decrementado una unidad. En el segundo ciclo se carga el contador de programa con la direccion de retorno. La PSW (bits 4-7) no se reestablece.

(SP) -- (SP) - 1

(CP) -- ((SP))

RETI : Return desde la interrupcion (solo 8022).

1 0 0 1 0 0 1 1

Esta es una instruccion de dos ciclos. Solo es aplicable al 8022. En el primer ciclo se decrementa el stack pointer. En el segundo se reestablece la direccion de retorno y se habilita de nuevo la linea de interrupcion.

(SP) -- (SP) -1

(CP) -- ((SP)) +1

RETR : Return con reestablecimiento de la PSW.

1 0 0 1 0 0 1 1

Esta es una interrupcion de dos ciclos. En el primer ciclo el stack pointer es decrementado una unidad. En el segundo ciclo el contador de programa y la PSW (bits 4-7) son reestablecidos.

(SP) -- (SP) - 1

(CP) -- ((SP))

(PSW4-7)-- ((SP))

RL A : Rotar a la izquierda sin el carry.

1 1 1 0 0 1 1 1

El contenido del acumulador es rotado un bit a la izquierda. El bit A7 es introducido en la posicion A0 del acumulador.

(An + 1) -- (An)

(A0) -- (A7) n=0-7

RLC A : Rotar a la izquierda con carry.

1 1 1 1 0 1 1 1

El contenido del acumulador es rotado un bit a la izquierda. El bit A7 es introducido en el carry. El carry es introducido en la posicion A0 del acumulador.

(An + 1) -- (An) n=0-7

(A0) -- (C)

(C) -- (A7)

RR A : Rotar a la derecha sin carry.

0 1 1 1 0 1 1 1

El contenido del acumulador es rotado un bit a la derecha. EL bit A0 del acumulador es introducido en el bit A7.

(An) -- (An + 1) n=0-6

(A7) -- (A0)

RRC A : Rotar a la derecha con carry.

0 1 1 0 0 1 1 1

El contenido del acumulador es rotado un bit a la derecha. El bit A0 del acumulador es introducido en el carry. El carry es introducido en el bit A7.

(An) -- (AN + 1)

(A7) -- (C)

(C) -- (A0) n=0-6

SEL AN0 : Seleccionar la entrada analogica cero.

1 0 0 1 0 1 0 1

Esta instruccion solo es aplicable al 8022.

SEL AN1 : Seleccionar la entrada analogica uno.

1 0 0 0 0 1 0 1

Esta instruccion solo es aplicable al 8022.

SEL MB0 : Seleccion del banco de memoria 0.

1 1 1 0 0 1 0 1

El bit CP11 del contador de programa es puesto a cero en la proxima instruccion JMP o CALL.

Todo los direccionamientos de la memoria de programa seran entre las direcciones 0 a 2047.

(DBF) -- 0

SEL MB1 : Seleccion del banco de memoria 1.

1 1 1 1 0 1 0 1

EL bit CP11 del contador de programa es puesto a uno en la proxima instruccion JMP o CALL.

Todo los direccionamientos de la memoria de programas seran entre las direcciones 2048 a 4095.

(DBF) -- 1

SEL RB0 : Seleccion del banco de registros 0.

1 1 0 0 0 1 0 1

EL bit 4 (BS) de la PSW es puesto a cero. Esto hace que se seleccione el banco 0 de registros de trabajo. (Localizaciones 0-7 de la RAM).

(BS) -- 0

SEL RB1 : Seleccion del banco de registros 1.

1 1 0 1 0 1 0 1

El bit 4 (BS) de la PSW es puesto a uno. Esto hace que se seleccione el banco 1 de registros de trabajo. (Localizaciones 24-31 de la RAM).

Se recomienda utilizar estos registros cuando se accede a subrutinas de interrupcion, pues el cambio al banco 1 no destruye la informacion del banco 0.

Cuando se produce el RETR la PSW (Bits 4-7) es reestablecida y por tanto el banco 0 se selecciona de nuevo.

(BS) -- 1

STOP TCNT : Parada del contador del TIMER COUNTER.

0 1 1 0 0 1 0 1

Esta intruccion se utiliza para detener la cuenta del TIMER COUNTER.

STRT CNT : Arranque del COUNT.

0 1 0 0 0 1 0 1

Esta instruccion habilita la entrada T1 como entrada de reloj del TIMER COUNTER y comienza la cuenta.

El contador es incrementado una unidad cada vez que la entrada T1 pasa de uno a cero.

STRT T : Arranque del TIMER.

0 1 0 1 0 1 0 1

Esta instruccion conecta la entrada de reloj del TIMER COUNTER a un preescaler que produce un pulso cada 32 ciclos maquina. De esta manera el contador se incrementa una unidad cada 32 x 15 periodos de reloj.

Esta instruccion pone a cero el prescaler pero no el contador.

Para mayor informacion ver el TIMER COUNTER en el capitulo 1.

SWAP A :

0 1 0 0 0 1 1 1

Esta instruccion intercambia los 4 bits menos significativos del acumulador con los 4 mas significativos de este.

(A0-A3) --- (A4-A7)

XCH A, Rr : Intercambio Acunulador - Registro Rr.

0 0 1 0 1 r r r

Esta instruccion intercambia el contenido del acumulador con el registro de trabajo seleccionado.

(A) --- (Rr) r=0-7

XCH A, @Rr : Intercambio Acumulador - Memoria de datos

0 0 1 0 0 0 0 r

El contenido del acumulador es intercambiado con la posicion de la memoria de datos direccionada por el registro Rr

(A) --- ((Rr)) r=0-1

XCHD A, @Rr : Intercambio A - Memoria de datos (4 bits)

0 0 1 1 0 0 0 r

Esta instruccion intercambia los 4 bits menos significativos del acumulador con los 4 bits menos significativos de la posicion de memoria direccionada por el registro Rr.

Los bits mas significativos quedan igual.

(A0-A3) --- ((Rr)) r=0-1

XRL A, Rr : OR exclusiva entre A y el registro Rr.

1 1 0 1 1 r r r

Esta instruccion realiza una operacion OR exclusiva entre el acumulador y el registro Rr seleccionado.

(A) -- (A) XOR (Rr) r=0-7

XRL A, @Rr : OR exclusiva entre A y memoria.

1 1 0 1 0 0 0 r

Esta instruccion realiza una operacion OR exclusiva entre el acumulador y la posicion de memoria de datos direccionada por Rr.

(A) -- (A) XOR ((Rr)) r=0-1

XRL A, #data : OR exclusiva entre a y el dato.

1 1 0 1 0 0 1 1 d d d d d d d d

Esta es una instruccion de dos ciclos. Realiza una operacion OR exclusiva entre el acumulador y el dato especidicado.

(A) -- (A) XOR dato

APLICACIONES DE LA MCS-48

ESTUDIO COMPARATIVO ENTRE LA MCS 48 Y LA MCS 85.

3.1 - INTRODUCCION.

En este capitulo vamos a hacer un pequeño estudio entre ambas familias, mas concretamente entre el microprocesador 8085 y el microcomputador 8048. La razon de este estudio es conocer las ventajas e inconvenientes de cada uno de estos chips, asi como el campo de aplicacion mas idoneo para cada microprocesador.

La razon para utilizar como punto de comparacion del 8048 el 8085 es que este ultimo ha sido y es uno de los microprocesadores mas usado en el diseño de sistemas.

La primera y gran diferencia entre ambos es que el 8048 es un microcomputador monochip,es decir un sistema completo formado por una RAM, una ROM, un contador interno programable, puertos de entrada/salida y la CPU propiamente dicha, mientras que el 8085 es un microprocesador conteniendo unicamente la CPU.

Sin mas preambulo pasamos a describir las caracteristicas de uno y otro.

3.2 - HARDWARE.

El estudio comparativo del Hardware del 8048 y el 8085 lo vamos a hacer basandonos en el diagrama de bloques de ambos chips. Figura 1.C

Como sabemos, la unidad central de proceso (CPU) esta compuesta por la unidad de control (C.U) y por la unidad aritmetica logica (ALU) ademas de una serie de registros que pasamos a describir a continuacion.

3.2.1 - CONTADOR DE PROGRAMA.

El contador de programa del 8085 es de 16 bits, esto permite direccionar de forma directa hasta 64 Kbytes mientras que el del 8048 es de 12 bits, con lo que solo se pueden direccionar de forma directa hasta 4Kbytes.

Como vemos en la figura 1.C el contador de programa del 8048 esta dividido en C.P.alto y C.P.bajo. Esto permite direccionar la memoria de programa en paginas de 256 bits.

Mediante el C.P.bajo (8 bits) se selecciona cualquier posicion dentro de una pagina, y mediante el C.P.alto se seleccionan la pagina deseada.

Esta caracteristica del C.P. del 8048 le da una gran potencia y flexibilidad en el manejo de memorias.

3.2.2 - DECODIFICADOR DE INSTRUCCIONES.

El decodificador de instrucciones es de ocho bits tanto en uno como en otro sistema, pues se trata de procesadores de 8 bits.

En el 8048 el registro de instruccion y el decodificador son uno solo mientras que en el 8085 estan separados.

3.2.3 - ALU. UNIDAD ARITMETICA LOGICA.

Tanto la ALU como el acumulador del 8048 es muy semejante al del 8085, la diferencia estriba en algunas operaciones con el acumulador. Por ejemplo el 8048 no posee la substraccion mientras que el 8085 si. Una caracteristica del 8048 es que posee una especie de segundo acumulador almacenador que agiliza las operaciones con la ALU.

3.2.4 - REGISTROS DE TRABAJO.

El 8085 posee seis registros de trabajo numerados B, C, D, E, H, L. Mientras, el 8048 posee ocho registros de trabajo numerados R0 a R7. Estos registros son muy importantes pue agilizan el trabajo del procesador.

Mediante una instruccion, en el 8048, se puede crear otro banco de ocho registros de trabajo.

Cuando se crea uno de los bancos de registros, el otro banco desaparece como banco, pero su contenido no. Esto permite disponer, en un momento dado, de 16 registros de trabajo.

Estos registros se pueden incrementar, decrementar etc.

3.2.5 - PALABRA DE ESTADO DEL PROGRAMA. PSW.

La PSW del 8048 es una palabra de 8 bits que nos informa en todo momento de ciertos parametros de un programa tales como el carry, el carry auxiliar etc. Tres de los ocho bits de la PSW se utilizan para guardar el puntero de pila. Efectivamente, el puntero de pila del 8048 (S.P.) es de solo tres bits y esta implementado dentro de la PSW del procesador.

El puntero de pila del 8085 es de 16 bits. La razon de que el S.P. del 8048 sea de solo tres bits es que este posee una pila interna de 8 niveles dentro de la RAM , mientras que el 8085 implementa la pila en una memoria RAM externa y por tanto no tiene limites en cuanto a tamaño de la pila.

En el 8085 es necesario definir la pila, esto es cargar el stack pointer. En el 8048 no es necesario esta operacion, pues tras un reset inicial, el stack se pone a cero.

3.2.6 - CIRCUITOS DE CONTROL Y TEMPORIZACION.

Como vemos en la figura 1.C varias de la señales generadas por este bloque son las mismas en ambos chips, por ejemplo la señal ALE, RD, WR, RESET IN, X1 y X2.

Nuestro interes se centra en las señales que son diferentes.

El 8085 posee una entrada para detener la ejecucion del programa. Esta entrada HOLD, detiene la ejecucion del programa y habilita las interrupciones. La linea HLDA indica al exterior que el procesador se encuentra en estado HOLD.

El 8048 no posee esta linea, es mas, tampoco posee una instruccion HALT como el 8085.

En el 8048 se podria simular una especie de estado HALT mediante un salto a un bucle cerrado, del que solo se saldria mediante una interrupcion o reset. La linea HOLD se podria simular mediante una de las entradas de testeo del 8048.

El 8048 posee dos entradas de testeo T0 y T1 que son, valga la rebundancia, testeables mediantes instrucciones de salto. Esta lineas, aunque no llegan a ser lineas de interrupcion son muy utiles.

El 8048 no posee las lineas de estados S0, S1 e IO/M. Esta ultima, IO/M, se puede crear mediante una de las linea de uno de los puerto del 8048.

El 8048 tampoco posee la entrada de READY pero se puede implementar con suma facilidad mediante las entradas testeables o por un puerto.

La señal CLK OUT del 8085 se puede obtener en el 8048 mediante una instruccion que convierte la entrada de testeo T0 en una salida de reloj.

La linea DE RESET OUT del 8085 se puede implementar mediante una linea de puerto del 8048.

El 8048 posee una entrada SS que permite la ejecucion paso a paso del programa (Single Step). Esta linea permite un diseño de sistemas mucho mas facil.

Otra linea importante en el 8048 es la linea EA, que permite inhibir la memoria ROM interna y forzar los ciclos de busqueda en una memoria ROM/ EPROM externa. Esto es muy util en la fase de diseño y comprobacion.

La linea PSEN del 8048 indica cuando se esta realizando un ciclo de lectura en una memoria ROM/EPROM externa. Esta linea suele utilizarse como "chip selec".

El 8048 no posee las lineas SID y SOD para la transmision / recepcion serie.

3.2.7 - INTERRUPCIONES.

En cuanto a lineas de interrupcion, el 8085 se lleva la palma, valga la expresion. Si, porque mientras el 8048 solo posee una linea de interrupcion, el 8085 posee cinco, de las cuales, una es vectorizada a 8 niveles. Todo esto hace que el 8085 pueda crear 12 interrupciones diferentes frente a 1 del 8048.

Para solventar la carencia de lineas de interrupcion del 8048 Intel ha creado el 8214

Ademas, el 8048 no posee la linea INTA de reconocimiento de la interrupcion, aunque se puede simular mediante una de las lineas de los puertos.

La linea de interrupcion del 8048 es unica y produce un salto a la direccion 0003 de la ROM interna.

Las lineas de testeo T0 y T1 del 8048 aunque no son lineas de interrupcion en el sentido extricto, permiten la bifurcacion del programa cuando son testeadas.

3.2.8 - MEMORIA DE PROGRAMA.

El 8048 posee una memoria ROM interna de 1 Kbyte mientras que el 8085 no posee ninguna. Esta es una de las grandes ventajas del microcomputador frente al microprocesador.

El 8047 es la version con EPROM del 8048.

El 8049 es el modelo con doble memoria ROM que el 8048.

El contador de programa del 8048 es de 12 bits, esto supone que puede direccionar 4 Kbytes de ROM.

Estos 4 Kbytes estan divididos en dos bancos de memoria de 2Kbytes cada uno.

La seleccion de uno u otro banco de memoria se realiza mediante dos instruccion que ponen a "cero" o a "uno" el bit mas significativo del contador de programa (CP11).

CP11 es un flip flop que se pone a "cero" o a "uno" mediante instruccion, nunca por el incremento natural del contador de programa.

3.2.9 - TIMER / COUNTER.

El 8048 posee un contador interno de 8 bits programable.

Este puede ser cargado y leido por medio del acumulador.

Ademas permite la utilizacion de una frecuencia de reloj generada internamente (Modo TIMER) ó una frecuencia de reloj externa (Modo COUNTER).

Tambien existe la posibilidad de crear una interrupcion cuando se produce un overflow en el contador. Efectivamente cuando el contador llega al maximo de cuenta, se pone a "uno" un flag de overflow generandose una peticion de interrupcion.

Si la interrupcion de overflow se habilita por programa automaticamente se salta a la posicion 0007 de la ROM donde el programador coloca la rutina de tratamiento.

Como sabemos, el 8085 no posee ningun contador interno programable.

Para hacernos una idea, este contador interno es muy semejante al contador de la RAM 8155 / 8156 de la familia MCS-85 pero con la ventaja que la frecuencia de reloj puede ser interna o externa segun se desee.

El modo TIMER se utiliza para generar retardos (DELAYS) mientras que el modo COUNTER se utiliza para contar impulsos externos.

3.2.10 - PUERTOS,

El 8048 posee 27 lineas de entrada / salida distribuidas en tres puertos de 8 bits ademas de la lineas de testeo T0, T1 e INT.

La linea INT si esta habilitada funciona como entrada de interrupcion como ya vimos. Cuando la interrupcion esta deshabilitada esta linea funciona como una entrada testeable mediante instrucciones de salto.

La linea T0, ademas de entrada testeable mediante instrucciones de saltos, puede funcionar como salida de Reloj opcional (CLK OUT) mediante una instruccion.

La linea T1, ademas de entrada de testeo, se utiliza como entrada de reloj del TIMER /COUNTER cuando este se utiliza en modo COUNT.

Los puertos del 8048 se numeran Puerto 0, 1, 2. Ver figura 1.C.

Una caracteristica muy importante de estos puertos, que le da una gran potencia, es que se pueden realizar operaciones logicas AND y OR entre los datos presentes en el puerto y cualquier posicion de memoria, o entre el puerto y el acumulador. Esto permite la creacion de mascaras en la memoria, que se pueden utilizar con los datos en los puertos para controlar cualquier dispositivo.

Los puertos 1 y 2 son identicos entre si en cuanto a su extructura fisica. Pueden funcionar tanto como entrada ó como salida de datos. Como salida posee unos lachs para almacenar la informacion, de esta forma la informacion queda en el puerto hasta que sea re-escrito de nuevo.

Como entrada no existe ningun lach, esto hace necesario que la informacion este presente a la entrada del puerto en el momento de producirse la lectura.

Por ultimo decir que cada linea del puerto puede ser programada individualmente como entrada o salida segun convenga. Esto es posible gracias a que estos dos puertos poseen una extructura " quasi-bidireccional ".

La unica diferencia entre el puerto 1 y 2 es que el puerto 2 tiene la posibilidad de desdoblarse en dos puertos de 4 bits. La razon de esto es que los 4 bits menos significativos del puerto 2 se utilizan para contener los 4 bits mas significativos del bus de direcciones cuando se accede a una memoria externa.

El puerto 0 ademas de utilizarse como puerto, se utiliza como BUS bidireccional del sistema que permite conectar el 8048 con el exterior. Este BUS es muy parecido al bus multiplexado del 8085, por esta razon el 8048 es directamente interfasable con las memorias y perifericos de la MCS- 85.

El puerto 0 junto con los 4 bis menos significativos del puerto 2 forman el bus de datos /direcciones del 8048.

3.2.11 - MEMORIA DE DATOS.

El 8048 posee una memoria RAM interna de 64 palabras de 8 bits. Dentro de la RAM se pueden crear un banco de registros de trabajo de 8 registros y una pila de 8 niveles (16 registros) esto reduce la memoria RAM a 40 palabras.

Naturalmente si la pila no se utiliza estos registros son utilizables como posiciones normales de RAM.

Tambien se puede crear un segundo banco de registros de trabajo, pero al crearlo el anterior banco desaparece como tal, y sus registros funcionan como posiciones normales de RAM.

3.3 - SOFTWARE.

Este apartado lo vamos a dedicar al estudio de instrucciones del 8048 de forma superficial, es decir sin describir niguna instruccion en concreto.

Este sera un estudio comparativo entre las instrucciones del 8048 y el 8085, pero siempre haciendo mayor incapie en el 8048.

El estudio se ha hecho agrupando el repertorio de instrucciones en bloques funcionales, es decir, las instrucciones de salto por un lado, las de transferencia por otro, etc.

Veremos instrucciones que posee uno y el otro no tiene,instrucciones comunes a ambos y tambien alguna instrucciones de especial interes del 8048.
Para enpezar, decir que el 8085 posee mas instrucciones que el 8048, cocretamente 116 y 96 respectivamente.

El 8048 posee instrucciones de uno o dos octetos como maximo, donde el 70 % son solo de un octeto.

El 8085 posee instrucciones de uno, dos y tres octetos.
Comparando uno con otro se deduce que para realizar un programa concreto se necesitara mas memoria en el 8085 que en el 8048.

En otras palabras, un programa en codigo maquina del 8048 ocupa menos lugar que en codigo maquina del 8085.

Esto es una doble ventaja, por un lado aprovechamiento de la memoria, y por otro lado menor tiempo de ejecucion.

3.3.1 - INSTRUCCIONES CON EL ACUMULADOR.

El acumulador del 8048 se puede incrementar, decrementar, rotar a izquierda, rotar a derecha, a travez del carry o fuera de el. Tambien se puede poner a cero y complementar, hacer operaciones logicas AND, OR y OR exclusiva lo mismo que en el

8085.

EL 8048 al igual que el 8085 posee la operaciones de suma con y sin carry.

Lo que no tiene el 8048 es la instruccion de subtraccion, es decir no existe la resta. La operacion de resta se debe hacer por suma al minuendo el complemento del sustraendo.

El 8048 posee una serie de instrucciones interesantes que no vamos a abordar en este capitulo, como ejemplo existe una instruccion de decremento del acumulador y salto si no es cero.

En cuanto operaciones en BCD con el acumuador, el 8048 posee una serie de instrucciones que lo hacen mas potente que el 8085 para este tipo de operaciones.

Una caracteristica muy importante del acumulador del 8048 es que puede hacer operaciones logicas AND y OR directamente con los puertos.

Existe una instruccion de salto si el bit "n" del acumulador esta a "uno".

Tambien existe saltos paginados por medio del acumulador.

3.3.2 - INSTRUCCIONES DE ENTRADA SALIDA.

El 8048 posee la instrucciones IN y OUT para introducir y sacar datos por los puertos al igual que el 8085.

Posee unas instrucciones de transferencia especiales, que solo se utilizan con el periferico 8243. El estudio de estas instrucciones se detalla en el capitulo de instrucciones del 8048.

El 8048 posee una serie de instrucciones muy interesantes que le permiten la realizacion de operaciones AND y OR entre los puertos y datos inmediatos, quedando el resultado de la operacion en el mismo puerto.

3.3.3 - INSTRUCCIONES CON LOS REGISTROS.

El 8085 posee las instrucciones de incremento y decremento de registros como el 8085.

Tambien posee la instruccion de incremento de cualquier posicion de la memoria RAM, sin embargo carece del decremento.

El 8048 puede hacer operaciones AND y OR entre los puertos y los registros de trabajo.

3.3.4 - INSTRUCCIONES DE SALTO.

El 8048 posee la instrucciones de salto incondicional ademas de las de salto condicinal como son:

Salto si hay carry,salto si no lo hay, salto si el acumulador esta a cero.

No posee la instruccion de salto si el acumulador es par.

El 8048 posee una serie de instrucciones de salto, que no posee el 8085 tales como salto si la entrada de testeo T0 esta a uno, salto si la entrada T1 esta a uno, salto si el flag de overflow del TIMER COUNTER esta a uno, etc.

El 8085 posee una instruccion doble muy interesante que decrementa un registro y luego salta a la posicion especificada si el contenido de dicho registro es distinto de

cero.

3.3.5 - SUBRUTINAS.

El 8048 posee solo tres instrucciones relacionadas con las subrutinas.

Una es CALL incondicional y las otras dos son de RETURN. Uno de los RETURN reestablece la palabra de estado del procesador y el otro no lo hace.

El 8048 no posee ninguna instruccion CALL condicional.

3.3.6 - FLAGS.

El 8048 puede poner a cero y complementar los flag F0, F1 y el de carry.

Los flags F0 y F1 son basculas internas de uso general que el programador puede testear mediante instrucciones de salto incondicional.

Los flags F0, F1 no tienen nada que ver con las entradas testeables T0 y T1.

3.3.7 - TRANSFERENCIA DE DATOS.

El 8048 puede realizar transferencias entre el acumulador y los registros de trabajo y entre la memoria de datos y el acumulador.

Tambien puede intercambiarse el contenido del acumulador con los registros de trabajo y con cualquier posicion de la memoria de datos.

El 8048 puede realizar transferencias entre el acumulador y una memoria externa.

Tambien puede intercambiarse los cuatros bits menos significativos del acumulador con los cuatro bits menos significativos de unos de los registros de trabajo o de la RAM. Esta propiedad es muy interesante en operaciones BCD.

La PSW puede ser transferida al acumulador y viceversa.

3.3.8 - TIMER COUNTER.

El contenido del contador del TIMER COUNTER puede ser transferido al acumulador y viceversa.

Existen instrucciones de marcha y paro del contador e instrucciones para habilitar o inhibir la interrupcion de overflow del TIMER COUNTER.

3.3.9 - CONTROL.

Las instrucciones de control del 8048 se reducen a siete. Dos son para habilitar o inhibir la linea de interrupcion exterior. Otras dos para seleccionar uno de los dos banco de memoria de programa. Otras dos para seleccionar uno de los dos bancos de registros de trabajo y la ultima para habilitar la salida de reloj por la patilla TO.

3.4 - APLICACIONES DE LA MCS-48.

Como regla general los microordenadores se deben emplear en aplicaciones de gran consumo. La grabacion de la ROM interna en fabrica mediante el proceso de mascara solo es rentable cuando el numero de chips es alto. (10000 unidadesd o mas).

Con caracter general, los microcomputadores se emplean en sistemas dedicados al control, que reunan las siguientes propiedades:

- Su precio es un factor critico.
- Precisan una capacidad de memoria pequeña.
- Se requiere un procesamiento en tiempo real.
- Manipulacion de bits y capacidad decimal.
- Pocos perifericos y arquitectura eficiente.

Con la aparicion en 1976 de la MCS-48 se logro reducir a un chip la cantidad de pastillas para formar un sistema minimo.Ver figura 2.C.

Con el 8085 se necesitaban tres pastillas y si nos remontamos en el tiempo, en 1972 cuando INTEL lanzo el 8008 la cantidad de chips para la realizacion de un sistema se elevaba a nada menos que 20.

La aparicion de la MCS-48 tambien ha permitido la reduccion de los costes de un sistema.

La figura 3.C muestra la evolucion de los costes de un sistema entre los años 73 a 79.

Como se ve en la figura el coste de un sistema con el 8048 en el 77 era de unos 20$ y en el 79 este se redujo a solo un dolar.

En cambio con el 8080 en el 77 el precio de un sistema se elevaba a unos 40$ y con el 8008 el precio estaba por encima de los 80$.

Las principales caracteristicas de la MCS-48 son:

-Sencillez de los montajes.

-Bajo consumo.

-Compatibilidad con perifericos de la MCS-85.

-27 lineas de entrada y salida.

-Repertorio de instrucciones potente.

-Abaratamiento de los costes a gran escala.

Estas caracteristicas junto con otras que no hemos mencionado hacen de la MCS-48 una familia ideal para pequeños sistemas de control donde no sea necesario un manejo masivo de datos ni programas demasiado extensos.

Por otro lado al ser la cantidad de componentes minima permite la realizacion de sistemas mas reducidos esto hace que la MCS-48 sea ideal para aparatos donde el espacio es vital.

Las aplicaciones mas importantes de la MCS-48 son:

- En la industria del automovil.

- En los electrodomesticos.
- En Terminales inteligentes.

En automoviles se utiliza para controlar y regular el consumo de carburante, el panel de control, el estado de los frenos, temperatura del aire acondicionado etc.

En los electrodomesticos se utiliza para controlar los procesos, temperaturas, etc.

La MCS-48 es de gran utilidad para la realizacion de terminales inteligente de bajo costo,tales como :

- Adaptadores de linea.
- Interfase de perifericos.
- Impresoras.
- Consolas de bajo costo.
- Drivers para teclado-display.
- Calculadoras de proposito general.

Otras aplicaciones:

- Instrumentos de medida.
- Sistema de alarmas.
- Modems.
- Equipos de audio y video.
- Paneles de control.
- Comunicaciones moviles.
- Maquinas de coser.

CAPITULO CUARTO: MACROENSAMBLADOR MCS-48

4.1 - INTRODUCCION.

Hoy en dia existen multitud de herramientas para el desarrollo de sistemas digitales tales como; emuladores, analizadores logicos, simuladores de ROM, ensambladores etc.

Todo esto para hacer el diseño de sistemas digitales mas facil y menos tedioso.

Desde la primera vez que empezamos a trabajar con la MCS-48 y sobre todo a la hora de realizar un programa en codigo maquina nos dimos cuenta de la necesidad que representaba disponer de un ensamblador de este microcomputador. asi pues decidi realizar un mini ensamblador que fuese capaz de darme el codigo maquina de un nemonimo que se introdujera por teclado. De esta manera realize un programa que esperaba a que se le introdujese un dato por teclado, buscaba el codigo maquina correspondiente y lo presentaba en pantalla.

Pronto me di cuenta que lo ideal seria que el ensamblador fuera capaz de trabajar con etiquetas de direcciones y olvidarme de las direcciones en hexadecimal.

Dicho y hecho, aunque costo lo suyo ¡ ,pues tuve que crear un fichero para memorizar las direcciones con su etiqueta correspondiente, ademas era necesario que el ordenador fuera capaz de distinguir las etiquetas de los

codigos del ensamblador y asi muchas mas dificultades, las cuales comentaremos mas adelante.

Ya parecia tener mi ensamblador terminado, cuando,me di cuenta que solo tenia el principio.

Un ensamblador debe tener la posibilidad de modificar la lineas, de insertar lineas nuevas, borrar lineas innecesarias, puesto en ello, realize un menu donde se daba acceso a todas estas posibilidades.

Para completar el programa añadi la posibilidad de poder imprimir los resultados del ensamblado por impresora.

Tras la utilizacion del programa, detecte la necesidad de poder almacenar los programas una vez realizados, de esta forma el menu del "Macroensamblador" se vio aumentado, con la posibilidad de de almacenar nuestros programas en disco y cargarlos en nuestro programa cuando se deseara.

Esta es la pequeña historia de como nuestro mini-ensamblador se convirtio en macro-ensamblador.
Por ultimo decir que el Macroensamblador esta sujeto a mejoras, una de esta mejoras podria ser mejorar la velocidad de ensamblado obtimizando el numero de aperturas y cierres de ficheros

Otra de las mejoras podria ser mejorar la insercion y modificacion de lineas utilizando un editor de pantalla.

4.2 - MACROENSAMBLADOR MCS-48.PROGRAMA.

En el apendice D aparece el listado del programa ensamblador. Este programa esta realizado en BASIC avanzado, mas concretamente en basic del IBM AT.

En los siguientes parrafos comentaremos el programa paso a paso, explicando las variables y ficheros utilizados.

Si mas preambulo pasamos a describir el programa:

En la lineas 130 a 230 aparece el menu principal del prográma, en el vemos que se puede editar, ensamblear, modificar lineas etc.

Cada una de estas opciones se selecciona pulsando el numero que esta a su izquierda.

Las lineas 250 a 350 se utilizan para detectar cual de las opciones ha sido pulsada y saltar a dicha opcion.

Antes de empezar a comentar opcion por opcion vamos a describir todos los ficheros utilizados en este programa.

FICHERO "FUENTE": Este fichero es donde se almacena el programa tal como nosotros lo introducimos por teclado, es decir : Etiqueta, instruccion en ensamblador, y comentario.

FICHERO "OBJETO": Este fichero lo crea el programa para almacenar en el las etiquetas,las instrucciones en ensamblador, el comentario, el codigo maquina, y el numero de linea, este ultimo se utiliza para facilitar las tareas de modificacion e insercion.

FICHERO "ENS": Este fichero tienes dos campos, uno lo forman las instrucciones en hexadecimal y el otro lo forma el codigo maquina correspondiente. El fichero este hay que crearlo manualmente,es decir, hay que hacer un pequeño programita cargador. Una vez cargado el fichero hay que

destruir el programa cargador. La razon de lo anteriormente expuesto es que este fichero solo sera leido por el programa ensamblador,nunca modificado. El listado de este fichero viene en el apendice E asi como el programa cargador.
El fichero ENS es el que le "dice" al programa ensamblador que codigo maquina tiene cada instruccion.

FICHERO "FUEN2" : Este fichero temporal que se utiliza en la opcion de insertar lineas para almacenar parte del FUENTE.

FICHERO "DIR" : Este es el quinto y ultimo fichero utilizado, en el se almacenan toda las etiquetas introducidas asi como su direccion hexadecimal.

Por ultimo, y referente a los ficheros, decir que los cinco son de acceso aleatorio.

4.2.1 - OPCION (1) EDICION.

Volviendo al desarrolo del programa llegamos a la primera opcion: Edicion del programa.(lineas 370-590). Aqui tenemos un pequeño menu que nos dice:

- Pulse cualquier tecla para comenzar la edicion.
- Escriba END para finalizar la edicion.

La primera premisa es para mantener en pantalla el menu hasta que se pulse cualquier tecla. La segunda premisa es la unica forma de salir de la edicion y volver al menu

principal.

La edicion del programa lo unico que hace es guardar las instrucciones introducidas por teclado (variable A$) e irlas almacenando consecutivamente en el fichero "FUENTE".

4.2.2 - OPCION (2) ENSAMBLADO.

Siguiendo el programa llegamos a la opcion de ensamblado. Esta es el alma del programa.

En la linea 620 el programa nos pregunta por la direccion de comienzo (variable ORG) esta es la direccion en hexadecimal donde va a estar nuestro programa en la ROM/EPROM.

La primera parte del ensamblado va desde las direcciones 660 hasta la 1140.

En esta primera parte las lineas del fichero "FUENTE" se desglozan en cinco variables A1$,A2$,A3$,A6$,A7$. Estas cinco variables contendran respectivamente etiqueta, instruccion ensamblador, comentario, valores inmediatos, salto a etiquetas. Para comprender mejor sirva el siguiente ejemplo:

```
                          A$
UNO: ADD A.#34H ;Esto es un ejemplo.
A1$    A2$   A6$        A3$

DOS: JMP*UNO ;Esto es otro ejemplo.
A1$ A2$  A7$             A3$
```

El programa identifica los simbolos : ; * # delimitando las variables anteriormente expuestas, todo este proceso se realiza en las lineas 710-970.

Las variables A, B, C, y E se utilizan para almacenar las posiciones de los simbolos anteriormente expuestos.

En la linea 980 de programa se optiene A5$ que es la variable que contiene la direcciones hexadecimales.

Todas estas variables ;A1$,A2$,A3$... se van almacenando en el programa "OBJETO" una tras otra hasta que se detecta la instruccion "END" .

La linea 1020 es de vital importancia, ya que es la que detecta si hay etiqueta y la guarda en el fichero "DIR"

Las lineas 990-1010 detectan si la instruccion en cuestion es de uno o dos octetos, y en consecuencia, incrementan la variable ORG.

La segunda parte de la opcion de ensamblado es la que se encarga de obtener el codigo maquina de las instrucciones (A4$). Esto se realiza entre las lineas 1150 -1580.

ER es la variable que cuenta el numero de errores que se van produciendo.

M y N son dos variables utilizadas en la presentacion en pantalla controlan en todo momento el numero de linea que salen por pantalla (20 en concreto).

B1$-B7$ son las variables intermedias del fichero "OBJETO".

En las linea 1190-1290 se obtiene de nuevo las variables A1$-A7$ (ecepto A4$) del fichero "OBJETO".

En las lineas 1300-1370 se busca el codigo maquina (A4$) de la instruccion (A2$) en el fichero "ENS". La variable CNT se utiliza como control del bucle para detectar cuando se ha sobrepasado la longitud del fichero "ENS" y por tanto detectar un error sintactico.

En la lineas 1400-1410 se mira si la instruccion es algun tipo de salto (variable A7$ <>" ") o si hace referencia a un dato (A6$ <> " ") y en cosecuencia le añade a A4$ el dato o la direccion corespondiente.

En la lineas 1460-1560 vuelve a almacenar las variables A1$, A2$... incluyendo A4$ en el fichero objeto.

Cuando se detecta la instruccion END (linea 1190) el ensamblado queda concluido y el programa salta a la direcciones 1590 -1790 donde aparece en pantalla el mensaje de "Ensambado completo", el numero de errores asi como un pequeño menu que nos permite visualizar las paginas del fichero "OBJETO" asi como volver al menu principal.

4.2.3 - OPCION (7) SACAR POR IMPRESORA.

Entre las lineas 2450-2640 esta la opcion para sacar por impresora el fichero "OBJETO"

La forma de impresion se realiza mediante un bucle controlado por la variable c que mediante la instruccion LPRINT va imprimiendo linea por linea el fichero asi hasta detectar la instruccion END donde vuelve automaticamente al menu principal.

4.2.4 - OPCIONES (5) Y (6) GUARDAR / CARGAR FICHERO.

Las lineas 2190 a 2320 contiene las opciones para almacenar o cargar un programa. La primera de la opciones es "guardar fichero" aqui lo que hace el programa es almacenar nuestro programa residente en el FUENTE y en el OBJETO en un fichero cuyo nombre lo ponemos nosotros (linea 2210).

El programa crea dos fichero con nuestro nombre pero con distintas extensiones. Una de las extensiones es ".OBJ" haciendo referencia al objeto y la otra es ".FUE" haciendo referencia al fuente.

La transferencia de informacion se realiza con un RENAME por lo que tras esta operacion los ficheros FUENTE y OBJETO desaparecen.
La segunda opcion es lo mismo, pero a la inversa.

4.2.5 - OPCION (3) MODIFICAR LINEAS.

Siguiendo nuestro programa llegamos a la opcion para modificar lineas.

Las lineas 2460-2490 presentan un pequeño menu para poder visualizar las diferentes pagina del "OBJETO".

La modificacion se realiza introduciendo primero el numero de linea a modificar y luego la instruccion que deseemos poner en esa linea. El programa entonces se va al "FUENTE" y modifica dicha linea.

Notese que la modificacion se hace en el fichero "FUENTE" esto implica que tras las modificaciones hay que volver a ensamblar para que el fichero "OBJETO" se actualice.

4.2.6 - OPCION (4) INSERTAR LINEAS.

Entre la lineas 2970 y 3520 se encuentra la opcion para insertar lineas. Esta opcion es mas compleja que la modificacion pues requiere guardar temporalmente todo lo que queda tras la insercion en un fichero hasta que la insercion quede concluida y luego volverlo a añadir al "FUENTE".

En las linea 2980-3070 aparece el menu de la opcion y en las lineas 3070 3220 se detecta que tecla ha sido pulsada. En la linea 3140 se pregunta en que linea va a comenzar la insercion (variable CON).

A continuacion todo el fichero "FUENTE" desde la linea CON hasta el fina se almacena en el fichero temporal. (FUEN2). Una vez concluida la operacion se procede a introducir las instrucciones que forma la insercion y terminando esta con el simbolo "/".

Cuando el programa detecta el simbolo "/" carga el fichero FUEN2 en el FUENTE tras la ultima linea de la insercion.

4.2.7 - OPCION (8) FINALIZACION.

La ultima opcion del programa es la de Finalizacion. Aunque parezca absurdo esta opcion es necesaria, pues a parte de salir del sistema sin la necesidad de un control BREAK evita que cuando lo vayamos a utilizar de nuevo se generen errores de fichero.

Si, efectivamente, si no destruyesemos los ficheros FUENTE y OBJETO al final del programa, cuando lo volviesemos a utilizar para editar un programa mas corto, las ultimas instrucciones del anterior programa se solaparian con las del ultimo,lo cual no nos interesa.

Lo mismo sucede con los ficheros DIR y FUEN2.

Por estas razones SIEMPRE hay que terminar con la opcion de finalizacion.

En la linea 3560 el programa nos pregunta a modo de recordatorio si queremos conservar nuestro programa. En el caso que ya lo hubiesemos conservado mediante la opcion (5) los ficheros FUENTE y OBJETO no existen y por tanto no hay que destruirlos. Si por equivocacion, y en el caso anterior, pusieramos que SI deseamos conservarlo, se produciria un error pues los ficheros FUENTE y OBJETO ya no existian.Este error no tiene efectos sobre nuestro programa ya conservado. Si este error se produce, basta con hacer un RUN y ejecutar la opcion (8) correctamente.

CAPITULO QUINTO:

DISEÑO DE UNA CERRADURA CODIFICADA BASADA EN LA FAMILIA MSC-48.

5.1 INTRODUCCION.

El repertorio de instrucciones de la MCS-48, permite un facil control de teclados y display, tanto en logica Hexadecimal como BCD.

Por otro lado, sus tres puertos facilitan su conexion con otros perifericos, simplificando enormemente el Hardware de los diseños.

En este capitulo vamos a describir el diseño y nombre de lo que podriamos llamar una "Cerradura Codificada".

En exencia, este circuito es un rele que solo se activa con una determina combinacion de numeros que introducimos por el teclado.

En principio, se ve que necesitaremos un teclado para introducir los datos. Por otro lado necesitaremos un circuito que analise la combinacion introducida, y que sea capaz de detectar si es la combinacion correcta, y luego, que sea capaz de activar el mecanismo de apertura (Rele, Triac).

Todo lo anteriormente expuesto, se podria completar con un Automata, pero este seria un tanto complicado y necesitaria demasiado Hardware.

Mejorando nuestro diseño, podriamos ponerle una alarma, esto es, que si a la tercera o cuarta vez que introducimos la combinacion esta es erronea, el circuito sea capaz de activar una sirena, durante un tiempo determinado'

Con esta ultima idea, hemos introducido la idea de un temporizador y de un contador de oportunidades (para la alarma).

El concepto de oportunidad es necesario, pues siempre puede ocurrir que pulsemos un numero equivocado. Por esta razon si el numero de oportunidades es pequeño (tres o cuatro) garantizamos por un lado, que cualquier intruso no pueda probar todas las combinaciones, y por otro lado, evitamos el riesgo de alarma por fallo accidental.

Otra duda que se nos plantea es, tras la alarma, es decir, una vez concluye la alarma, por un lado, podriamos hacer que el programa se bloquease(halt) y quedase asi hasta un reset, o se podria volver al principio del programa, con lo que el sistema queda en condiciones de aceptar una nueva combinacion.

Cada una de estos dos posibilidades tiene sus ventajas e inconvenientes.

La primera seria ideal para lugares en donde la alarma no tenga utilidad (por ejemplo: sitios solitarios y alejados). La segunda posibilidad, es ideal para cerradura de puertas de casas, coches, etc., donde la alarma "auyentaria" al intruso.

En nuestro montaje vamos a considerar la posibilidad primera (Reset).

Tras ésta introducción vamos a pasar a la explicación de nuestro diseño.

5.2 - DIAGRAMA DE BLOQUES.

En la figura 1.E (figura 1. del apendice E), aparece el diagrama de bloques de nuestro sistema. El corazón de la "Cerradura" es un 8035 o 8039 de la MCS-48. Este microcomputador se ayuda de un 8212 como decodificador de direcciones y una 2716 como memoria de programa.

En caso de utilizar un 8048, 8049, o 8748, el sistema se reduciría a una única pastilla, además de unos pocos componentes discretos.

Pasamos a continuación a describir el funcionamiento.

Como vemos en la figura 1.E el teclado del sistema, está unido al puerto P1 del microcomputador.

La resistencias "Pul Up" del Bus, en realidad no hacen falta, pues los puestos del MCS-48 están internamente en "Pull Up".

El Port 0 (Bus), se utiliza como Bus datos/direcciones del sistema exterior.

Es de importancia observar, como parte del puerto 2 (lineas P20-P22) son utilizadas como los Bits más significativos del Bus de direcciones, (A3-A-10). Esto es necesario para poder direccionar la totalidad de la memoria de la 2716 (2KBytes). El 8212 está como Lach de direcciones, pues la 2716 tiene separados los buses de datos y direcciones. Las lineas P25-P27 del Port 2, se utilizan como salidas para gobernar los dispositivos de

alarma y apertura. La linea P-25 se utiliza para activar el rele de "Apertura".

Este rele solo se activa cuando la combinacion introducida por teclado coincide con la combinación de apertura.

Hagamos cuentas; si la combinación consta de cuatro número, por ejemplo: 1876, una sencilla operación, nos dá el número total de combinaciones:

Esto es 10 E 4 = 10.000 combinaciones, o sea que con 4 números hay 10.000 combinaciones distinta, y una sola de ésta es la que activa el relé de apertura.

En caso de que fuese una combinación de 6 números tenemos que ; 10 E 6 = 1.000.000, si un millón de combinaciones.-

Volviendo a la figura 1.E

La linea P-25, es la encargada de activar la alarma, esta puede ser una sirena, una luz, etc.

La linea P-25 está unida a un led que se iluminará cuando la combinación introducida sea errónea, viene a ser como un indicador de que el sistema está trabajando.

Este led es necesario, pues a veces introducimos un número, y sin embargo introducimos otro (por Ç: Cuando se llama por teléfono).

Si no existiese este "Led" trás introducir la combinación y ver que no se activa la "Apertura" nos quedaría la duda si el sistema está mal, o la combinación introducida és errónea.

En la figura vemos que la linea EA la hemos llevado a alimentación (HIGH), con ésto conseguimos inhibir la memoria ROM interna y habilitar la externa (2716).

5.3 - SOFWARE. PROGRAMA GENERAL.

Una vez explicado el HARDWARE del sistema de la cerradura codificada, pasemos en éste apartado a explicar el programa que controlará al micocomputador.

En la figura 2.E se muestra lo que hemos llamado programa general, entendido por general, un organigrama de fácil asimilación, para comprender lo mejor posible el funcionamiento del programa.

Este organigrama es una primera aproximación a lo que será el programa elaborado, que más tarde comentaremos.

Para empezar, trás un reset el microcomputador es cargado con las condiciones iniciales, tales como inicializar, variables y registros, pero ésto lo veremos detalladamente más adelante.

Siguiendo el flujo, se llega a una estructura de decisión en donde se pregunta: ¿Hay tecla pulsada¿ En caso negativo el bucle retorna de nuevo al principio, con ésto se pretende mantener al microcomputador en un bucle de espera.

En caso que se pulse una tecla, el programa continua almacenando el valor de la tecla pulsada, en una posición de momoria. Continuando el flujo se llega a un bloque de decisión donde se pregunta ¿Es la 4ª tecla¿, ésto es en el caso de que la combinación de apertura sea de 4 números. En éste bloque se controla el número de teclas pulsadas, mientras el número de teclas pulsada sea menor que 4, el

bucle retorna al principio en busca del siguiente número.

Una vez se han introducido los números que forman la combinación de entrada, el programa sigue hasta el siguiente bloque, donde se hace una comparación ante los datos introducidos por teclado y la combinación interna de apertura (Residente en la ROM).

Más abajo aparece otro bloque de decisión donde se pregunta: Coinciden ambas combinaciones ?. En caso afirmativo, significa que hemos dado con la combinación correcta y por tanto se activa el relé de apertura.

En nuestro caso, como el diseño está orientado para archivar el encendido de un automóvil, es necesario que el relé permanezca activado hasta que se le corte el suministro de energía (mediante la llave de encendido), por ésta razón se lleva al microcomputador a un estado de retención HALT. La única forma de sacar al microcomputador de éste estado es mediante un reset exterior, o sea desconectandole la alimentación.

Volviendo a la comparación de las combinaciones, en caso que no coincidan, el programa bifurca a otro bloque de decisión donde se mira si es la última oportunidad.

Si, pues en la pregunta anterior vimos la necesidad de dar varias oportunidades (dos o tres) para evitar que un error circunstancial (falso contacto, mala pulsación) lleve al programa a un estado de alarma.

En caso que no sea la última oportunidad, el programa vuelve al principio, estando en condiciones de aceptar una nueva combinación.

Cuando el número de combinaciones introducidas sugiere la cantidad estipulada (nº de oportunidades) el programa salta a la subrrutina de actuación de alarma.

La alarma (activación) puede ser temporal (por ejemplo un minuto trás la alarma) o indefinida.

En el primer caso el programa volveria al principio, en el segundo caso se quedaria en un estado HALT (parado), del que solo se saldría mediante un reset o desconectar la alimentación del sistema.

5.4 - SOFTWARE: PROGRAMA ELABORADO.-

Una vez explicado y comprendido el programa general que controla el sistema, vamos a explicar el organigrama de la figura 3.E.

En éste organigrama, más elaborado, aparecen las variables de control de los bucles, las direcciones destinadas a almacenar las teclas introducidas, etc.

Para comprender éste organigrama es necesario conocer bien los microcomputadores de la familia MCS-48, pues hacemos referencia a los registros de éste microcomputador.

El registro R7 es inicializado con el valor "04H, éste registro controlará el número de oportunidades, y como vemos, el número de oportunidades será cuatro.

El registro R0 (registro puntero RAM) será el encargado de direcciones la RAM, se inicializa con el valor 0AH, que es la dirección de RAM donde vamos a guardar la 1ª tecla, o sea que las 4 teclas introducidas por teclado se pretende almacenar en las direcciones 00Ah - 00Dh, ambas

inclusives. R6, se utilizará para controlar el número de teclas que han entrado, lo inicializamos con el valor 04h, pues las combinaciones serán de 4 números.

En un principio el programa permanece en espera que se pulse una tecla. Mediante un bucle detector en que momento se pulsa una tecla, pues al pulsar una tecla el puesto del teclado experimenta una variación de nivel, que se detecta en el acumulador.

Una vez se pulsa una tecla y es detectada, ésta se guarda temporalmente, se espera un retardo de 10 msg, y se vuelve a leer el puesto del teclado para ver si la misma tecla continúa pulsada. Con ésto se consigue evitar falsa pulsaciones debido a ruidos y rebotes.

Una vez comprobada que se trata de una verdadera pulsación, ésta tecla se guardará en la dirección de memoria señalada por R0.

Ya guardada la tecla, se incrementa la dirección (para alimentar la próxima tecla, y se decrementa R6 (nº de teclas pulsadas).

Cuando R6=0, o lo que es lo mismo, cuando ya han sido pulsadas 4 teclas, se procede a la comparación. Antes de la comparación es necesario definir una serie de variables. El registro R5, será el encargado de direccionar el lugar de la memoria Rom, donde se encuentra la combinación de apertura, la combinación de apertura debe estar en la Rom, en las direcciones 0FA a 0FD ambas inclusive, R5 se inicializa con el valor "FA".

R0 continua siendo el registro que direcciona la RAM en las posiciones 00A a 00D, en donde esta la combinacion introducida por teclado.

R3 sera el registro que controlara el bucle de comparaciones, es decir habrá que hacer cuatro comparaciones y por esta razon'se inicializa con el valor "04H".

El resto del programa es de facil comprension por lo que no es necesario comentarlo.

5.5 SOFTWARE. ENSAMBLADO DEL PROGRAMA.

En la figura 4.D aparece el listado del programa ya depurado y con las instrucciones en codigo maquina.

Este programa ha sido editado,depurado y ensamblado mediante el macroensamblador MCS-48 que ha sido comentado en el capitulo cuarto de este proyecto.

La comparacion entre la tecla pulsada y la combinacion en memoria es ralizada mediante la instruccion OR Exclusiva, pues el 8048 no posee instrucciones de comparacion entre registros.

El retardo de 10 ms se consigue mediante un doble bucle controlado por los registros R4 y R2.

5.6 CONCLUSION.

El diseño de la cerradura codificada ha sido montado y probado en la practica comprobandose su perfecto funcionamiento. En la figura 5.D aparece el esquema de la placa del circuito a escala 1:1.

CAPITULO SEXTO: MEDIDOR DIGITAL DE PRESIONES BASADO EN LA MCS-48.

6.1 INTRODUCCION.

En este capitulo vamos a describir el funcionamiento y diseño de un medidor de presiones digital que trabajara entre los rangos de 0 a 250 mm de mercurio, basandonos en uno de los microcomputadores de la MSC-48.

El sistema ademas de poder medir presiones sera capaz de detectar unos limites de presion maximo y minimo preestablecidos por el operador.

El limite de 250 mm Hg es debido exclusivamente al transductor utilizado, es decir, que el transductor elegido trabaja linealmente en la zona de 0 a 250 mm Hg. En caso de que se quisiera trabajar en otro rango de presiones solo bastara cambiar de transductor.

En nuestro caso el transductor utilizado es un transductor marca FOXBORO modelo 1800-03-01-00B-0. Informacion sobre este transductor la podemos encontrar en el apendice F de este proyecto.

Este transductor puede medir presiones a partir de fluidos y gases. Para mayor informacion sobre transductores de presion se recomienda el proyecto " SISTEMA DE DETECCION DE LIMITES DE PRESION Y FRECUENCIA RESPIRATORIA BASADO EN MICROPROCESADOR." del alumno Jose

Quintana Segura.

6.2 DIAGRAMA DE BLOQUES.

En la figura 1 del apendice F (1.F) aparece el diagrama de bloques del medidor de presiones, al completo, y que acontinuacion pasamos a describir.

La señal procedente del trasductor es amplificada a los niveles necesarios por el amplificador diferencial (AD) de entrada. Dicha señal analogica es distribuida por un lado a los detectores de limite superior de alarma e inferior y por otro lado a un semi convertidor analogico-digital (A/D).

El semi-convertidor A/D es en esencia un combertidor analogico/tiempo, es decir que es capaz de convertir una señal analogica en un intervalo de tiempo proporcional a esta.

El tiempo transcurrido desde que se activa la señal CR (Comienzo de rampa) hasta que se activa la señal FR (Fin de rampa) es proporcional al voltaje de la señal analogica presente a la entrada.

El microcomputador cuenta el tiempo que transcurre entre las dos señales CR y FR y obtiene de este modo un valor numerico (digital) proporcional a la señal analogica (presion) presente a la entrada.

Los bloques de deteccion de limites son basicamente unos comparadores de la señal proveniente del transductor y una señal continua (voltaje) preestablecida por los potenciometros PI y PS (Potenciometro de fijacion de limite inferior y superior respectivamente).

Estos bloques generan las señales AI y/o AS (Alarma inferior y superior respectivamente), en el supuesto caso que la señal analogica de presion se salga de los niveles establecidos por los potenciometros PS y PI.

Los potenciometros PS y PI deben ser lineales y seran los que el operador utilizara para establecer los limites de alarma superior e inferior.

Cuando se produce una alarma de presion, ya sea alta (AS) o baja (AI) se dispara a travez de una puerta OR un temporizador programable mediante el potenciometro PR (Potenciometro de retardo).

La necesidad de este bloque de retardo es para evitar que cualquier sobre-pico de presion momentaneo active la alarma.

El funcionamiento de este bloque es muy simple, cualquiera de las señales (AI o AS) dispara el temporizador, cuando esto ocurre transcurre un tiempo programado por el potenciometro PR.

Una vez finalizado este tiempo de retardo el bloque genera la señal ALA ,la cual comunica al microcomputador que se ha producido una alarma de presion.

La señal ALA comunica al microcomputador que se ha producido una alarma, pero no le dice si es alarama por presion alta o baja.

El microcomputador detecta que tipo de alarma se ha producido mediante el valor obtenido a partir del semi convertidor analogico digital.

Una vez se produce la señal ALA el microcomputador activa algun dispositivo de alarma (sumbador, altavoz, etc) a travez de la linea P26.

Observando el dibujo se detecta la presencia de un conmutador de tres posiciones y un comun. Este conmutador sirve para poder visualizar por display no solo la presion sino los limites de alarma superior e inferior.

En la posicion 1 se visualiza el valor del limite superior y en la 3 se visualiza el valor del limite inferior.

Generalmente el conmutador estas en la posicion 2, que es la que visualiza la presion proveniente del transductor.

La visualizacion de la presion se hace mediante tres displays conectados atravez del puerto 1 del microcomputador.

Las señales P24 y P25 se utilizan para seleccionar los displays.

6.3 SECCION ANALOGICA.

En la figura 2.F (figura 2 del apendice F) aparace el esquema de toda la parte analogica del medidor de presiones.

El transductor T esta alimentado mediante una fuente de corriente formada por un amplificador operacional de IC1 una resistencia 2K7 y un zener de 4.3 voltios.

La corriente que circula entre las patillas 6 y 7 de IC1 es aproximadamente de 1,5 mA, mas concretamente de 4.3/2K7=1,59 mA.

Los tres amplificadores operacinales de IC1, restantes, forman un amplificador diferencial de alta impedancia de entrada, que amplifican la señal proveniente de las patillas 4 y 10 del transductor.

El amplificador esta diseñado para dar una ganancia variable entre 60 y 100 mediante el potencimetro P1. El rango de salida del transductor es de 0 a 150 m voltios, con lo que una ganancia de 100 hace que tengamos a la salida del amplificador unos 15 voltios maximos.

El conjunto de amplificadores IC2 forman unos seguidores de tension para evitar cargar la señal proveniente del amplificador diferencial y de los potenciometros P2.

El potenciometro P2 de arriba es el que determina el limite superior de alarma y el potenciometro P2 de abajo limita el limite inferior de alarma.

IC3 son dos comparadores que se utilizan para compara la señal de presion con los limites de alarma.

La patilla 1 de IC3 se pone a "uno" cuando la señal analogica sobrepasa el limite de presion maximo. De igual forma la patilla 7 de IC3 se pone a "uno" cuando la señal analogica queda por debajo del limite inferior de alarma.

Con las dos puertas NOR de arriba se forma una puerta OR que dispara a un monoestable formado por otras dos puertas NOR un condensador de 33 Mfaradios y un potenciometro P3. Este potenciomtro P3 es para programar el tiempo de retardo.

Cuando se produce una alarma se dispara el monoestable y la patilla 3 de IC5 permanece a nivel bajo hasta que transcurre el tiempo de retardo,tras lo cual, la puerta AND formada por las patillas 5,6 y 10 de IC5 queda transparente a la señal de alarma proveniente de IC3.

En caso que la alarma continue tras el periodo de retardo, la señal ALA pasara a nivel alto siendo detectado por el microcomputador.

El conjunto formado por IC6, IC7 y el 555 es lo que se ha denominado semi convertidor A/D en el diagrama de bloques. El 555 genera una onda cuadrada que ataca a un integrador formado por IC7. A la salida del integrador tenemos una onda en forma de diente de sierra.

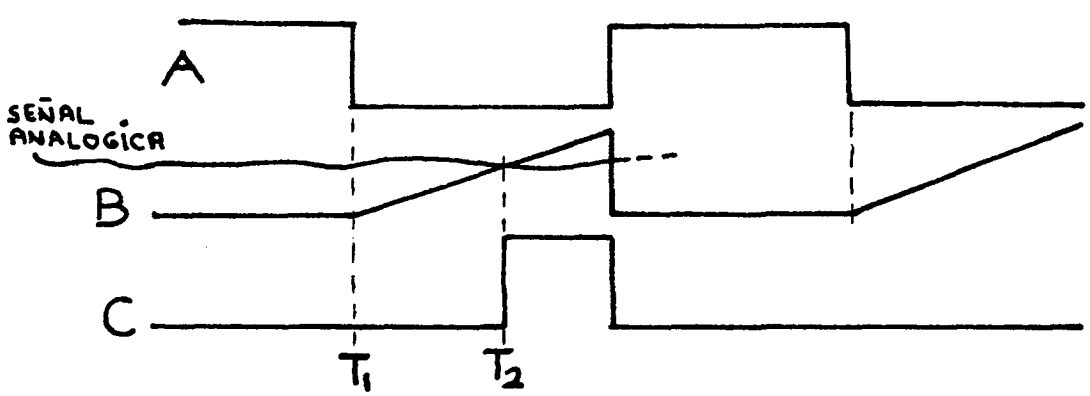
La pendiente de la rampa del diente de sierra se modifica medianre el potenciometro de 20 K.

IC6 actua como comparador entre la señal analogica del transductor y la rampa generada.

La seña CR se genera cuando la onda proveniente del 555 (A) pasa nivel bajo y es justo en el momento que comienza la rampa.

La señal FR se genera en el momento en que el voltaje de rampa es ligeramente superior al voltaje analogico, esto supone que el comprador de a su salida un "uno".

Esto se vera mejor en el siguiente cronograma:



La señal A es la onda cuadrada presente a la salida (3) del 555,cuado dicha señal pasa a cero el integrador comienza la rampa. Mientras la señal analogica (ANA) sea mayor que la rampa la salida del comparador IC6 permanece a nivel "cero".

Cuando la rampa alcanza el valor de la señal analogica el comparador pasa a "uno" (señal C).

El tiempo transcurrido desde el intervalo T1 a T2 es proporcional al voltaje de la señal analogica presente a la entrada.

Por ultimo decir que la tension de alimentacion de esta etapa analogica es de +/- 12 voltios para los integrados de alimentacion simetrica y +12 voltios para los integrados de alimentacion simple.

6.4 SECCION DIGITAL.

En la figura 3.F tenemos el esquema electronico de lo que se ha denominado seccion digital. Esta seccion esta compuesta por el microcomputador, 3 decodificadores BCD-7 segmentos (4511) y los tres displays (TIL 313) ademas de una serie de componentes discretos.

Los circuitos integrados 4511 son decodificadores BCD-7 segmentos con LACHES de entrada, osea que son capaces de memorizar la entrada (BCD) y manterner los displays con un valor determinado aunque la entrada este a otro valor.

La patilla 5 es la que activa los LACHES de entrada. Las patillas 1,2,6 y 7 forman la entrada BCD. Las patillas 9,10,11,12,13,14,15 forman la salidas a lis displays de 7 segmentos.

Las patillas 3,4 y 16 han de estar a 5 Voltios en los tres 4511 (aunque en la figura 3.F solo aparezca el de la derecha). La patilla 8 es la masa.

Lo que aparece como un bloque en la figura con la inscripcion 7 x 330 son en realidad 7 resistencias de 330 ohmios cada una, y son necesarias para limitar la corriente a travez de los displays.

El circuito integrado TIL 313 es un display de 7 segmentos de catodo comun. El catodo se encuentra en las patillas 2 y 9.

Los decodificadores estan unidos al microcomputador a travez del puerto 1 (patillas 27-34) . Los dos 4511 de la derecha estan gobernados mediante la salida P24 del puerto 2. El microcomputador activa estos dos displays a la vez mandando por el puerto 1 dos numeros BCD. Uno de estos numeros BCD es mandado al display de la derecha que sera el que represente las unudades. EL otro numero BCD sera mandado al 4511 del centro y es el que representara las decenas.

El 4511 de la izquierda es gobernado mediante la patilla P25 del puerto 2. Observando el dibujo vemos la posibilidad de añadir otro display para formar las unidades de 1000. Este display añadible seria gobernado por la linea P25.

La linea INT del microcomputador va unida a la señal FR (fin de rampa) del semiconvertidor A/D. La linea T0 del microcomputador va unida a la señal CR (comienzo de rampa) del semicovertidor A/D.

La linea P27 del microcomputador va unida a la señal ALA proveniente del temporizador en la seccion analogica.

La linea de reset del micro tambien sera accesible mediante un pulsador para resetear el sistema en un momento dado.

La linea P26 del microcomputador activa un sumbador en el momento que se produzca una alarma.

6.5 PROGRAMA

En la figura 4 del apendice F aparece un organigrama simplificado del programa de control del medidor de presiones digital.

Tras unas conidiciones iniciales que describiremos mas adelante lo primero que hace es meterse en un bucle de espera hasta que se genere la señal de comienzo de rampa.

Cuando se detecta la señal CR el programa comieza a incrementar el registro A mientras no se produzca la señal del final de rampa (FR).

Una vez activada la señal FR el contenido del registro A es mandado a los displays. El valor contenido en los displays es un numero que representa de forma digital el valor analogico (presion) del amplificador de entrada.

Tras presentar el valor en el display, el programa "mira" si se ha producido algun tipo de alarma mediante el muestreo de la señal presente en P27 (señal ALA). En caso que la señal ALA este activada, el microcomputador activara el dispositivo de alarma a travez de la linea P 26, y pasara a un estado de inoperancia (HALT) hasta que el sistema sea reseteado.

En caso de que no se halla producido la señal ALA el programa vuelve al principio reiterandose todo el proceso.

En la figura 5.F aparece un organigrama mas elaborado del medidor de presiones. En este aparecen los registros utilizados asi como las condiciones iniciales.

Hemos utilizado una serie de numeros introducidos en un circulo para hacer referencia mas detallada a las partes del programa.

Las condiciones iniciales son poner la PSW del microcomputador a cero para evitar que el bit de C y CA queden a "uno" al endecder el sistema, adema hay un salto a la direccion 00AH que es donde comienza el programa.

Los registros R0 y R1 son utilizados para almacena el valor obtenido tras la conversion A/D. R0 guarda los dos numeros BCD menos significativos (unidades y decenas) y R1 guarda las centenas.

En el punto 1 se espera a que comienze la rampa (T0=low). En el punto 2 se hace la coversion A/D mediante el incremento y ajuste decimal del acumulador. La instruccion DAA es necesaria para evitar los numeros hexadecimales mayor que nueve (A,,B,C,D y E).

Mediante el carry C se detecta cuando el acumulador pasa de 99 en decimal y en consecuencia se incrementa el registro R1.

El punto 2 se repite tantas veces como la señal INT este a cero,osea mientras no se produzca el final de rampa.

En el punto 3 se guarda el valor de la conversion en el registro R0 y se espera un tiempo muerto hasta que la señal T0 pase a 1.

En el punto 4 se manda el contenido de los registros R0 y R1 a los displays, esto se hace una vez cada 255 conversiones A/D para evitar un molesto parpadeo que se produce en el display de las unidades. El registro R5 es el que controla la visualizacion cada 255 conversiones.

En el punto 5 se detecta si se ha producido la señal de alarma (ALA).

En el punto 6 se activa el dispositivo de alarma.

En la figura 6.F aparece el listado del programa ya ensamblado y depurado y en la figura 7.F aparecen los cliches de las placas que componen el sistema.

	FUNCTION	PART NUMBER	DESCRIPTION	COMMENTS
MCS-48™	Microcomputers	8021	1K ROM Program Memory — 10 μsec Cycle	Compatible versions of the single chip microcomputers provide mask programmed, light erasable, or no internal program memory.
		8048	1K ROM Program Memory	
		8049	2K ROM Program Memory	
		8035	No Program Memory 64 x 8 RAM — 2.5 μsec Cycle	
		8035L	8035 with Power Down Mode	
		8039	No Program Memory 128 x 8 RAM	
		8048 8	1K ROM Program Memory	
		8748 8	1K EPROM Program Memory — 5.0 μsec Cycle	
		8035 8	No Program Memory	
	Memory and I/O Expanders	8355	2K x 8 ROM with 16 I/O Lines	Compatible devices allow direct expansion of MCS-48 functions with no additional external components.
		8755A	2K x 8 EPROM with 16 I/O Lines	
		8155/56	256 x 8 RAM with 22 I/O Lines and Timer	
	I/O Expander	8243	16 Line I/O Expander	Low Cost I/O Expander
Compatible MCS-80/85™ Components	Standard ROMs	8308	1K x 8 450 ns	Allow low cost external expansion of Program Memory. The 8308 is interchangeable with 8708 and the 2316E with the 2716.
		2316E	2K x 8 450 ns	
	Standard EPROM	8708	1K x 8 450 ns Light Erasable	User programmable and erasable.
		2716	2K x 8 450 ns Light Erasable	
	Standard RAMs	8111A 4	256 x 4 450 ns Common I/O	Data memory can be easily expanded using standard NMOS RAMs. The 5101 CMOS equivalent reduces standby power to 75 nW/bit
		8101A 4	256 x 4 450 ns Separate I/O	
		5101	256 x 4 650 ns CMOS	
	Standard I/O	8212	8-Bit I/O Port	Serves as Address Latch or I/O port.
		8255A	Programmable Peripheral Interface	Three 8-bit programmable I/O ports.
		8251A	Programmable Communicating Interface	Serial Communications Receiver/Transmitter
	Standard Peripherals	8205	1 of 8 Binary Decoder	MCS 80 peripheral devices are compatible with the MCS 48 allowing easy addition of such specialized interfaces as the 8279 Keyboard/Display Interface. Future MCS-80/85 devices will also be compatible.
		8214	Priority Interrupt Controller	
		8216	Bi-directional Bus Driver	
		8226	Bi-directional Bus Driver (Inverting)	
		8253	Programmable Interval Timer	
		8259	Programmable Interrupt Controller	
		8279	Programmable Keyboard/Display Interface (64 Keys)	
		8278	Programmable Keyboard/Display Interface (128 Keys)	
	Universal Peripheral Interface	8041	ROM Program Memory	User programmable to perform any custom I/O and control functions.
		8741	EPROM Program Memory	

FIG. 1.A Componentes de la MCS-48 y perifericos.

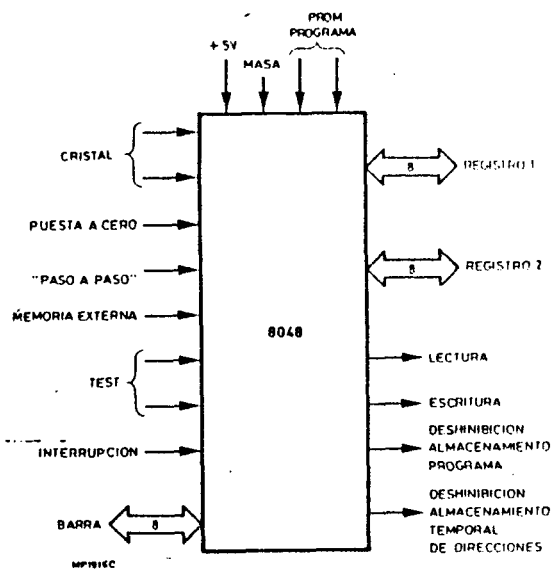


FIG. 2.A Extructvra externa y simplificada del 8048.

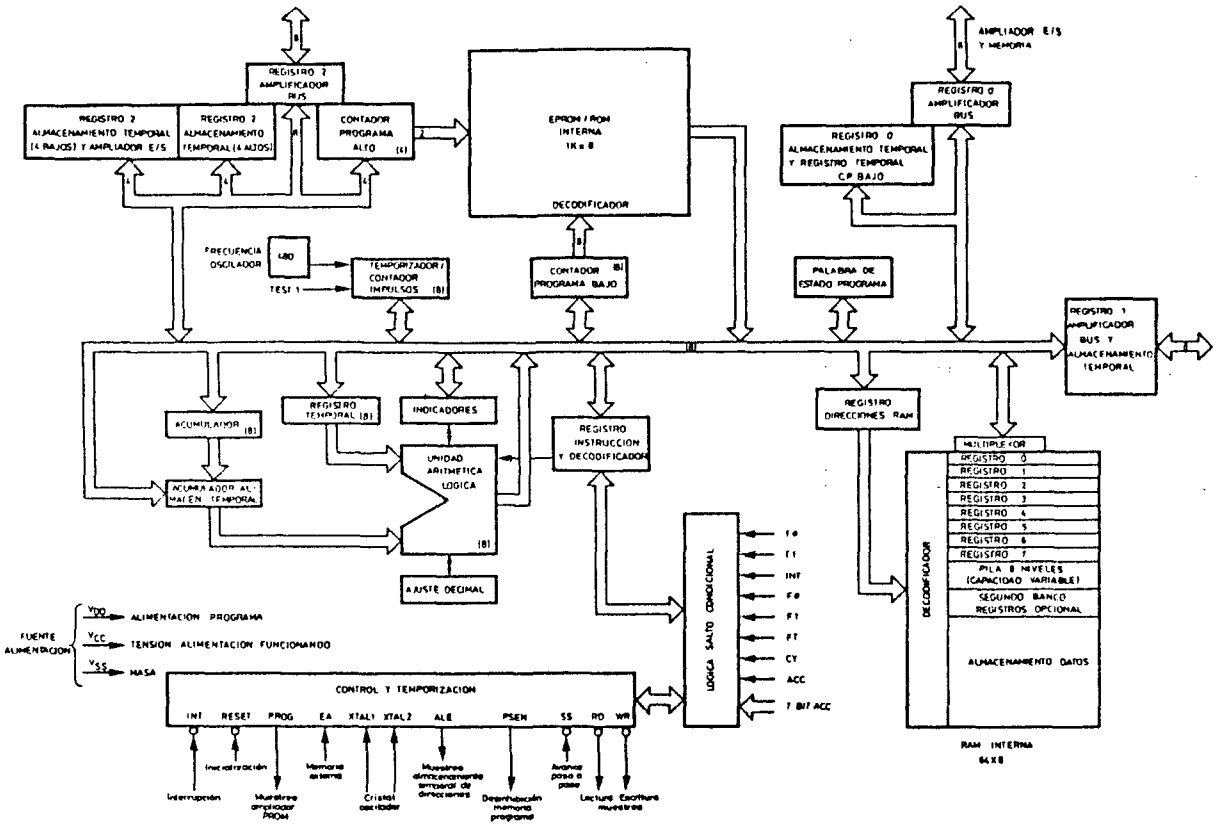


FIG. 3.A Arquitectura interna del 8048.

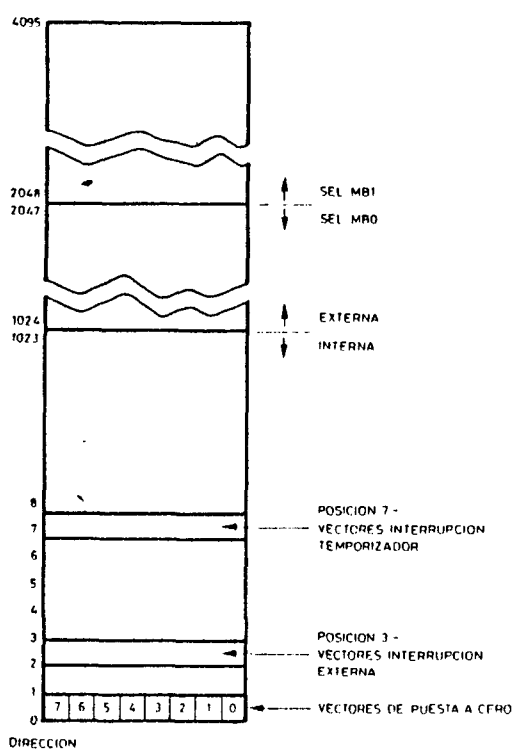


FIG 4.A Memoria de programa.

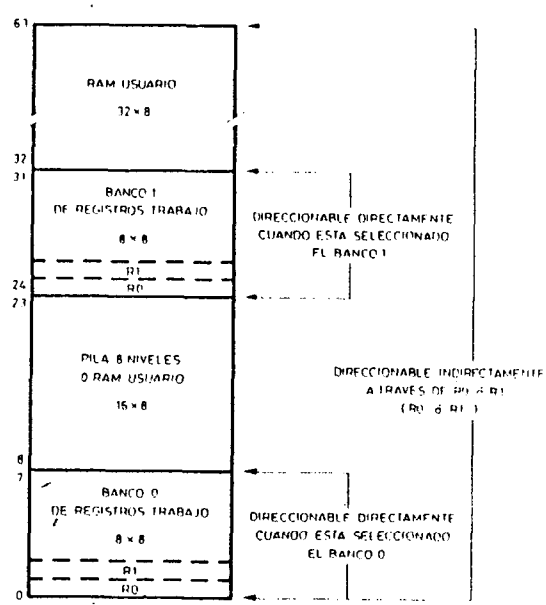


FIG 4.A Memoria de datos.

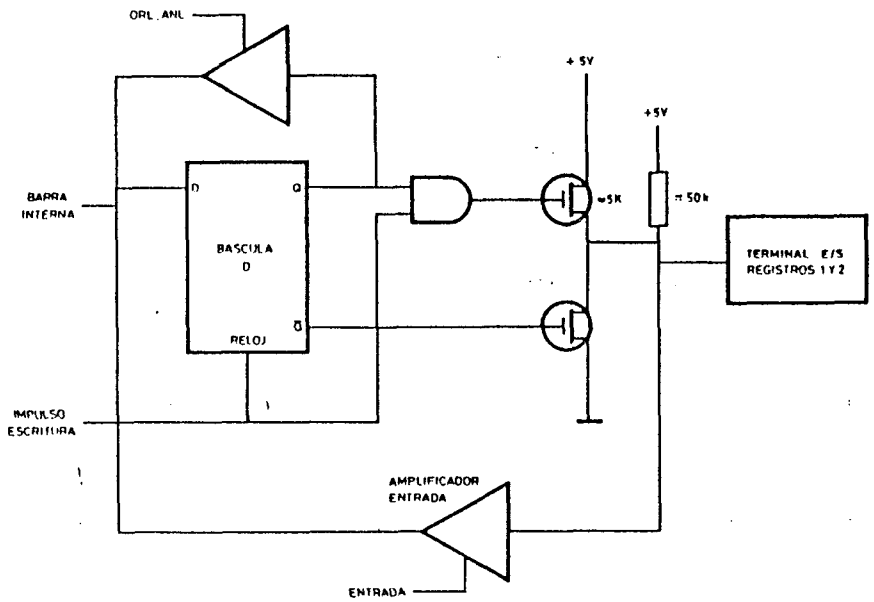


FIG 5.A Extructura quasi-bidirecciona.

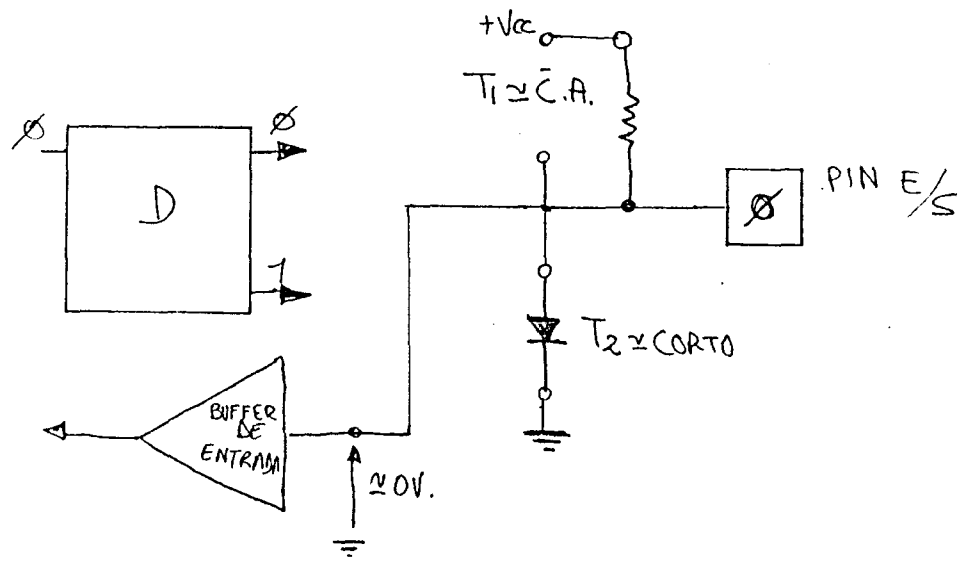


FIG 6.A Circuito equivalente cuando se programa un cero de salida.

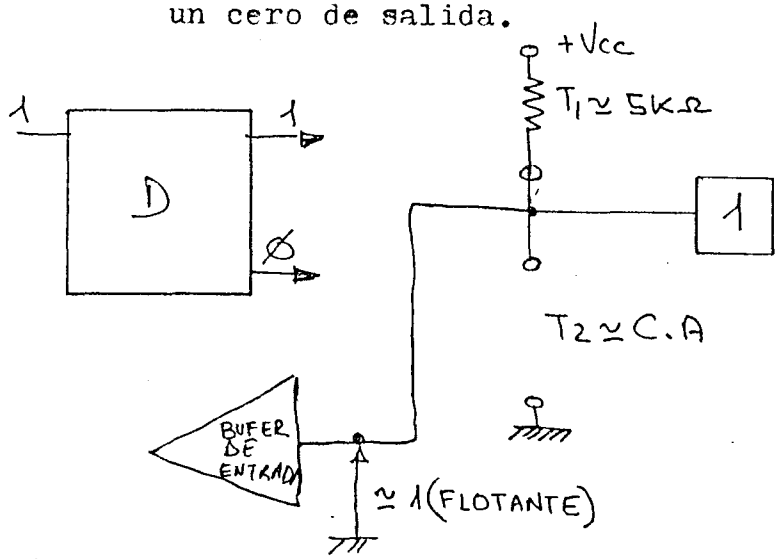


FIG 7.A Circuito equivalente cuando se programa un uno de salida.

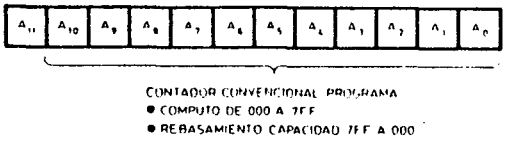


FIG 8.A Contador de programa del 8048.

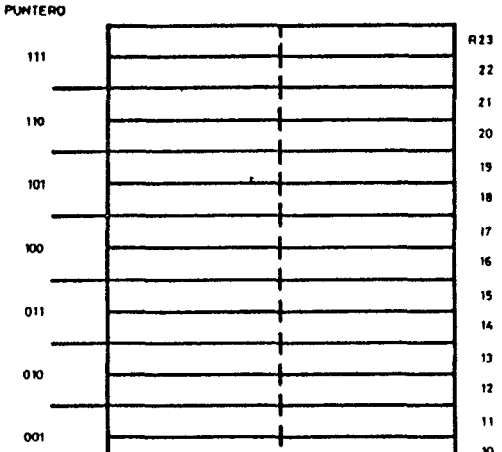
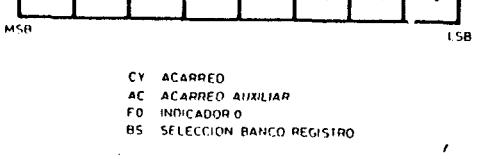


FIG 9.A Pila del 8048. Esta pila esta formada por 8 niveles numerados del 000 al 111. Cada nivel esta compuesto por dos registros de 8 bits cada uno.



CY ACARREO
AC ACARREO AUXILIAR
F0 INDICADOR 0
BS SELECCION BANCO REGISTRO

FIG 10.A Palabra de estado del procesador.

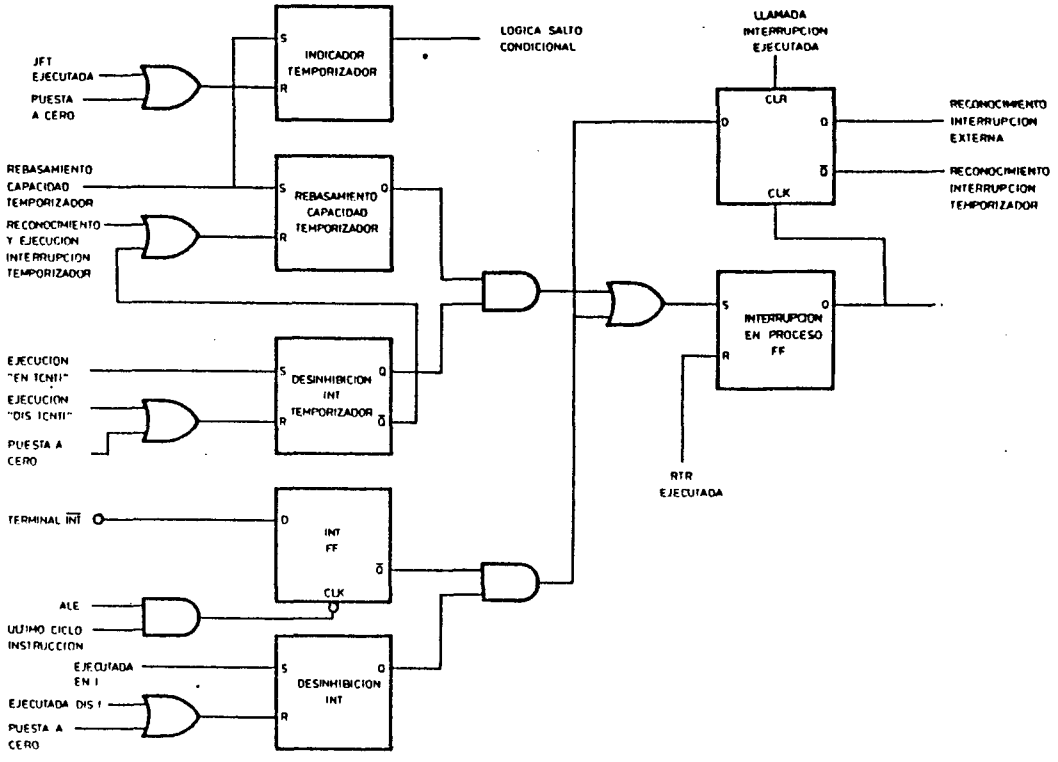


FIG 11.A Logica de interrupciones

FIG 12.A Diagrama del Temporizador interno

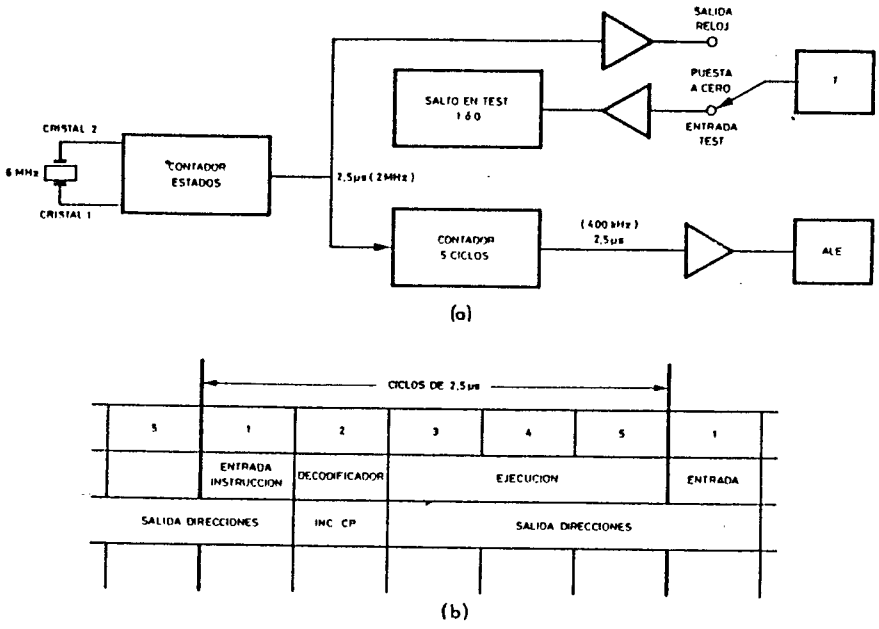


FIG 13.A Circuitos de TIMING del 8048. Ciclo maquina.

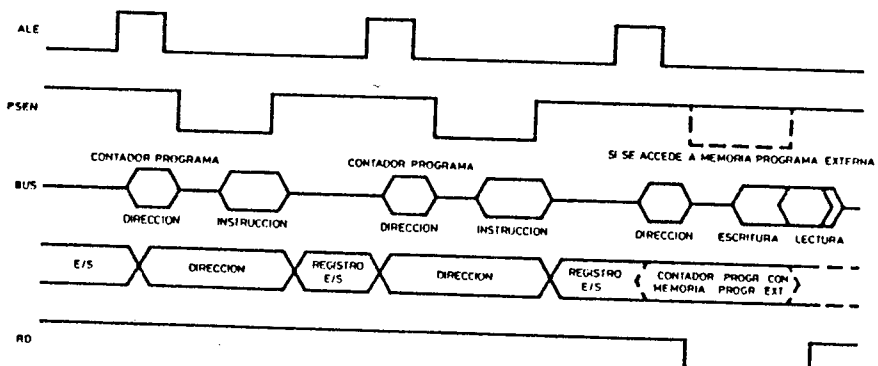


FIG 13.A Timing basico del 8048.

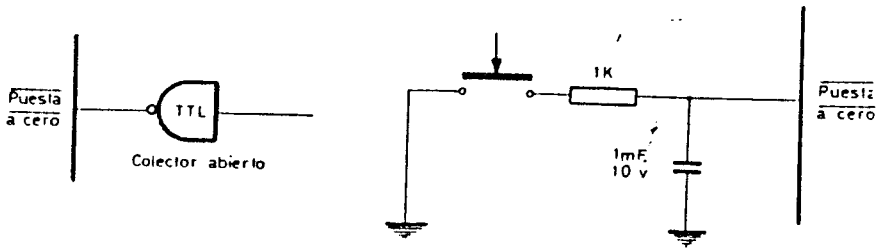


FIG 14.A Reseteo del 8048.

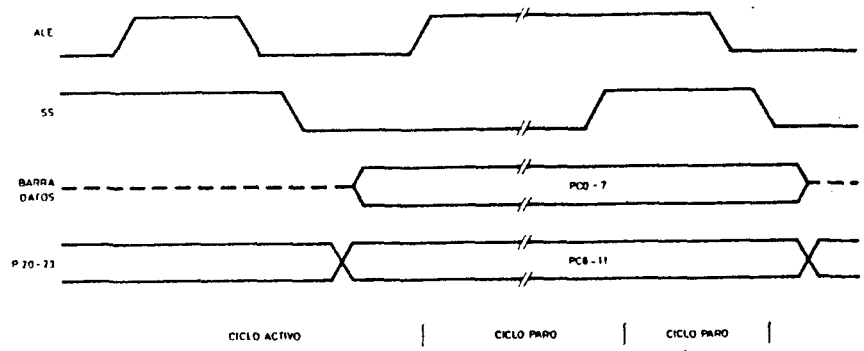


FIG 15.A Cronograma de ejecucion paso a paso.

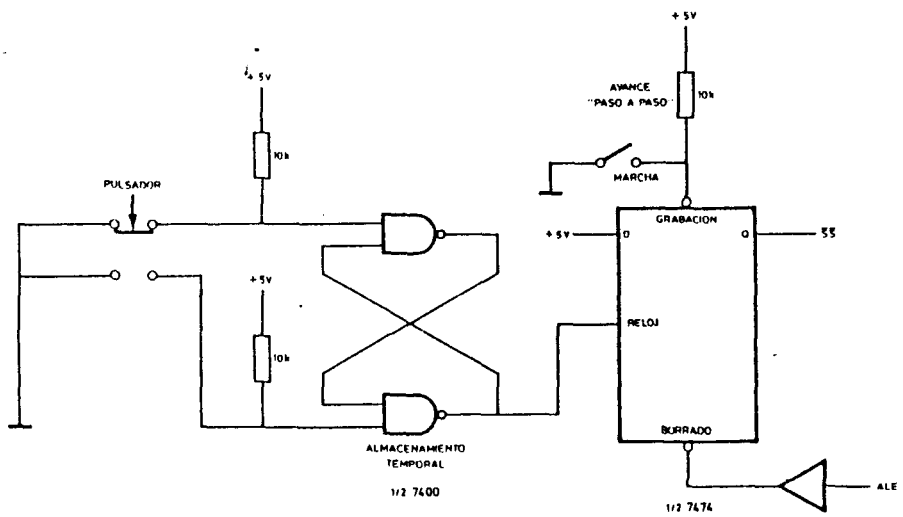


FIG 16.A Circuiteria extra para la ejecucion paso a paso.

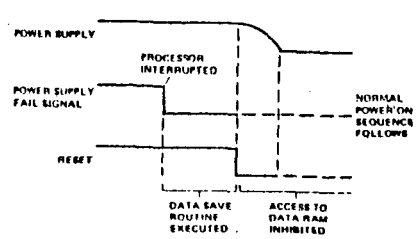


Fig 17.A Cronograma en modo "POWER DOWN"

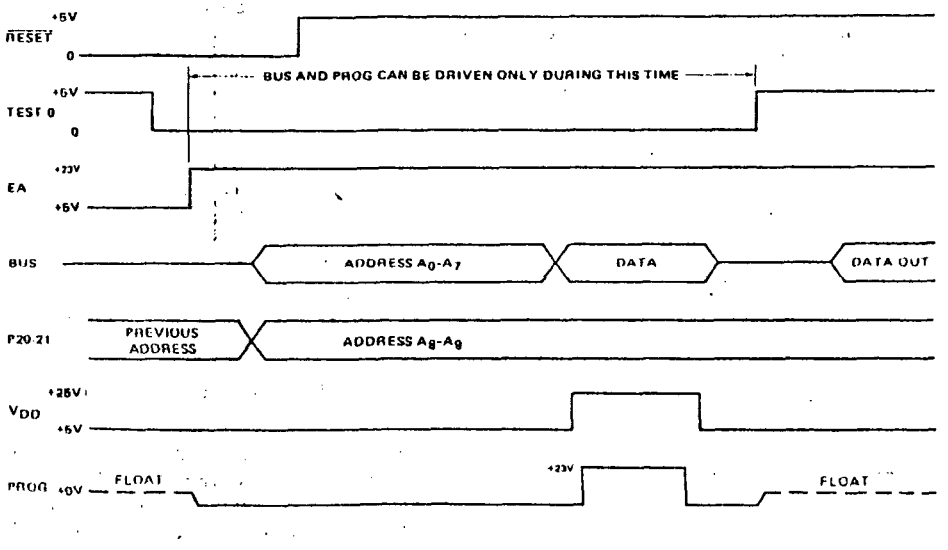


FIG 18.A Cronograma de programacion del 8748 (EPROM)

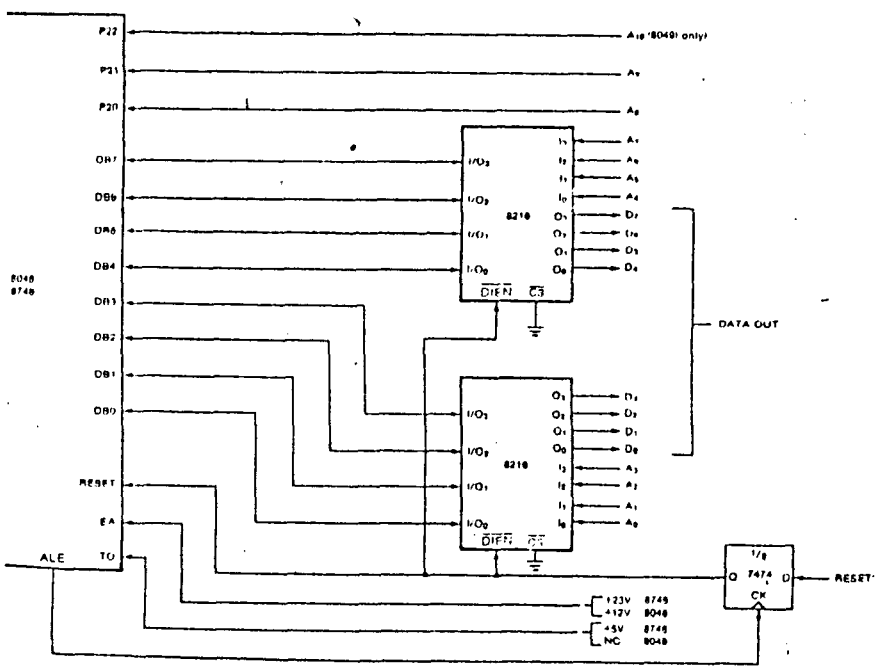


FIG 20.A Circuiteria para la lectura de la ROM del 8048 desde el exterior.

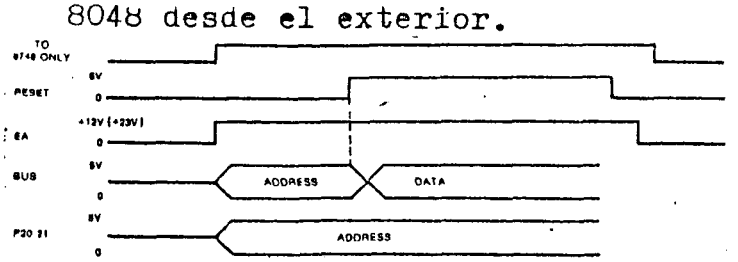
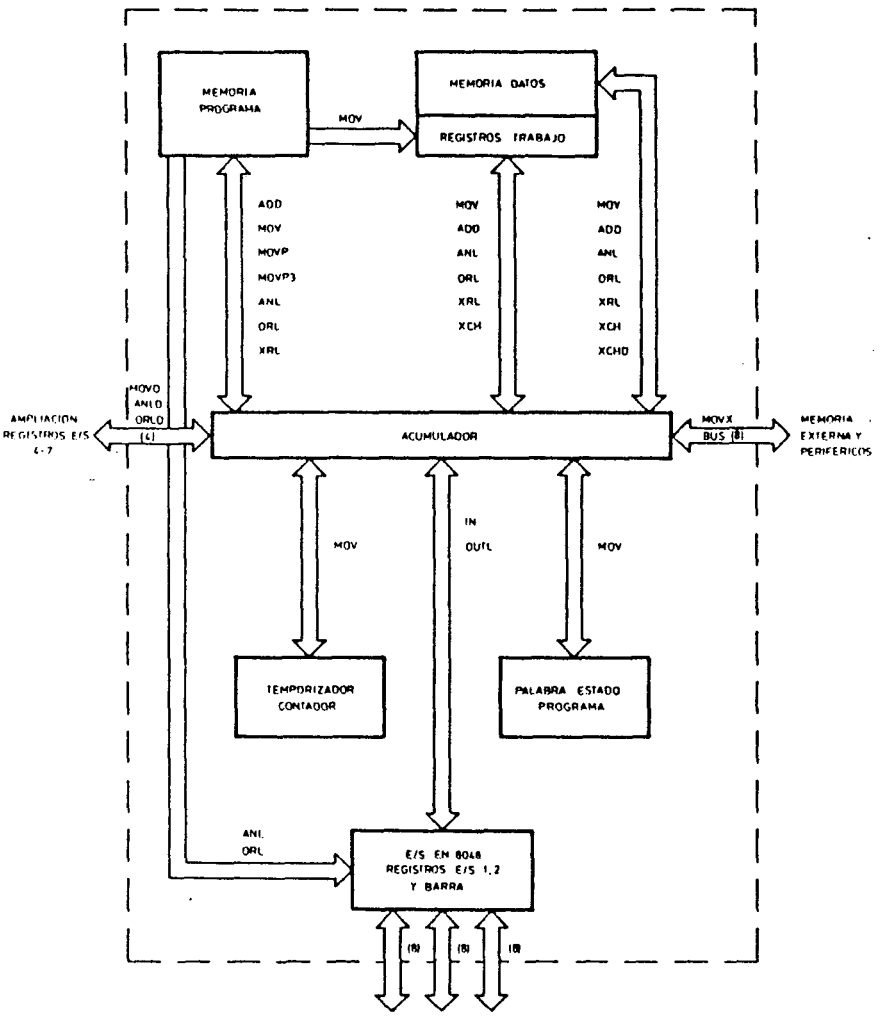


FIG 19.A Cronograma de la lectura de la ROM interna.

A P E N D I C E B

_ REFERENTE AL CAPITULO SEGUNDO.



Transferencia de datos en el 8048.

FIG 1.B

Nemónico	Descripción	Código		Código hexadecimal	N.º octetos	N.º ciclos
Acumulador						
ADD A,R	Suma del registro R a A	0110	1rrr	–	1	1
ADD A,@R	Suma del dato direccionado por R a A	0110	000r	–	1	1
ADD A, # data	Suma inmediata de A	0000	0011	03	2	2
ADDC A,R	Suma de R a A con acarreo	0111	1rrr	–	1	1
ADDC A @R	Suma del dato direccionado por R a A con acarreo	0111	000r	–	1	1
ADDC A, # data	Suma inmediata de A con acarreo	0001	0011	13	2	2
ANL A,R	AND entre el registro R y A	0101	1rrr	–	1	1
ANL A,@R	and entre el dato direccionado por R y A	0101	000r	–	1	1
ANL A, # data	AND inmediata con A	0101	0011	53	2	2
ORL A,R	OR entre el registro R y A	0100	1rrr	–	1	1
ORL A,@R	OR entre el dato direccionado por R y A	0100	000r	–	1	1
ORL A, # data	OR inmediato con A	0100	0011	43	2	2
XRL A,R	OR exclusivo entre el registro R y A	1101	1rrr	–	1	1
XRL A,@R	OR exclusivo entre el dato direccionado por R y A	1101	000r	–	1	1
XRL A, # data	OR exclusivo inmediato con A	1101	0011	D3	2	2
INC A	Incrementar A	0001	0111	17	1	1
DEC A	Decrementar A	0000	0111	07	1	1
CLR A	Poner a cero A	0010	0111	27	1	1
CPL A	Complementar el contenido de A	0011	0111	37	1	1
DA A	Ajuste decimal de A	0101	0111	57	1	1
SWAP A	Los bits 0 a 3 son cambiados por los bits 4 a 7	0100	0111	47	1	1
RL A	Rotación de A a la izquierda	1110	0111	E7	1	1
RLC A	Rotación de A a la izquierda a través del acarreo	1111	0111	E7	1	1
RR A	Rotación de A a la derecha	0111	0111	E7	1	1
RRC A	Rotación de A a la derecha a través del acarreo	0110	0111	67	1	1
Entrada-Salida						
IN A,P	Entrar el contenido de un registro de E/S (1 ó 2) y transferirlo a A	0000	10pp	–	1	2
OUTL P, A	Sacar el contenido de A y transferirlo a un registro de E/S (1 ó 2)	0011	10pp	–	1	2
ANL P, # data	AND inmediata con un registro de E/S (1 ó 2)	1001	10pp	–	2	2
ORL, P, # data	OR inmediato con un registro de E/S (1 ó 2)	1000	10pp	–	2	2
INS A, BUS	Entrar el contenido de la barra y transferirlo a A	0000	1000	08	1	2
OUTL BUS, A	Sacar el contenido de A y transferirlo a la barra	0000	0010	02	1	2
ANL BUS, # data	AND inmediata con la barra	1001	1000	98	2	2
ORL BUS, # data	OR inmediata con la barra	1000	1000	88	2	2
MOVD A,P	Entrar el contenido de un registro de E/S externo (4 – 7) y transferirlo a A	0000	11pp	–	1	2
MOVD P,A	Sacar el contenido de A y transferirlo a un registro de E/S externo (4 – 7)	0011	11pp	–	1	2
ANLD P,A	AND entre un registro de E/S externo y A	1001	11pp	–	1	2
ORLD P,A	OR entre un registro de E/S externo y A	1000	11pp	–	1	2
Registros						
INC R	Incrementar el registro R	0001	1rrr	–	1	1
INC @R	Incrementar el dato direccionado por R	0001	000r	–	1	1
DEC R	Decrementar el registro R	1100	1rrr	–	1	1
Saltos						
JMP addr	Salto incondicional a la dirección especificada	$a_{10}a_9a_80$	0100	–	2	2
JMPP @A	Salto indirecto (direccionado por A)	1011	0011	B3	1	2
DJNZ R,addr	Decremento y examen de R. Salta si (R) ≠ 0	1110	1rrr	–	2	2
JC addr	Salto si el acarreo es 1	1111	0110	F6	2	2
JNC addr	Salto si el acarreo es 0	1110	0110	E6	2	2
JZ addr	Salto si (A) = 0	1100	0110	C6	2	2
JNZ addr	Salto si (A) ≠ 0	1001	0110	96	2	2
JTO addr	Salto si TO es 1	0011	0110	36	2	2
JNTO addr	Salto si TO es 0	0010	0110	26	2	2
JT1 addr	Salto si T1 es 1	0101	0110	56	2	2
JNT1 addr	Salto si T1 es 0	0100	0110	46	2	2
JFO addr	Salto si el indicador 0 (F0) es 1	0111	0110	B6	2	2
JF1 addr	Salto si el indicador 1 (F1) es 1	0111	0110	76	2	2
JTF addr	Salto si el indicador del temporizador/contador (FT) es 1)	0001	0110	16	2	2
JN1 addr	Salto si la entrada de interrupción externa (INT) es 0	1000	0110	86	2	2
JBb addr	Salto si el bit b (1 entre 8) de A es 1	$b_2b_1b_01$	0010	–	2	2
Subrutinas						
CALL	Salto a subrutina	$a_{10}a_9a_81$	0100	–	2	2
RET	Retorno de subrutina	1000	0011	83	1	2
RETR	Retorno de subrutina y restablecimiento del estado inicial	1001	001	93	1	2

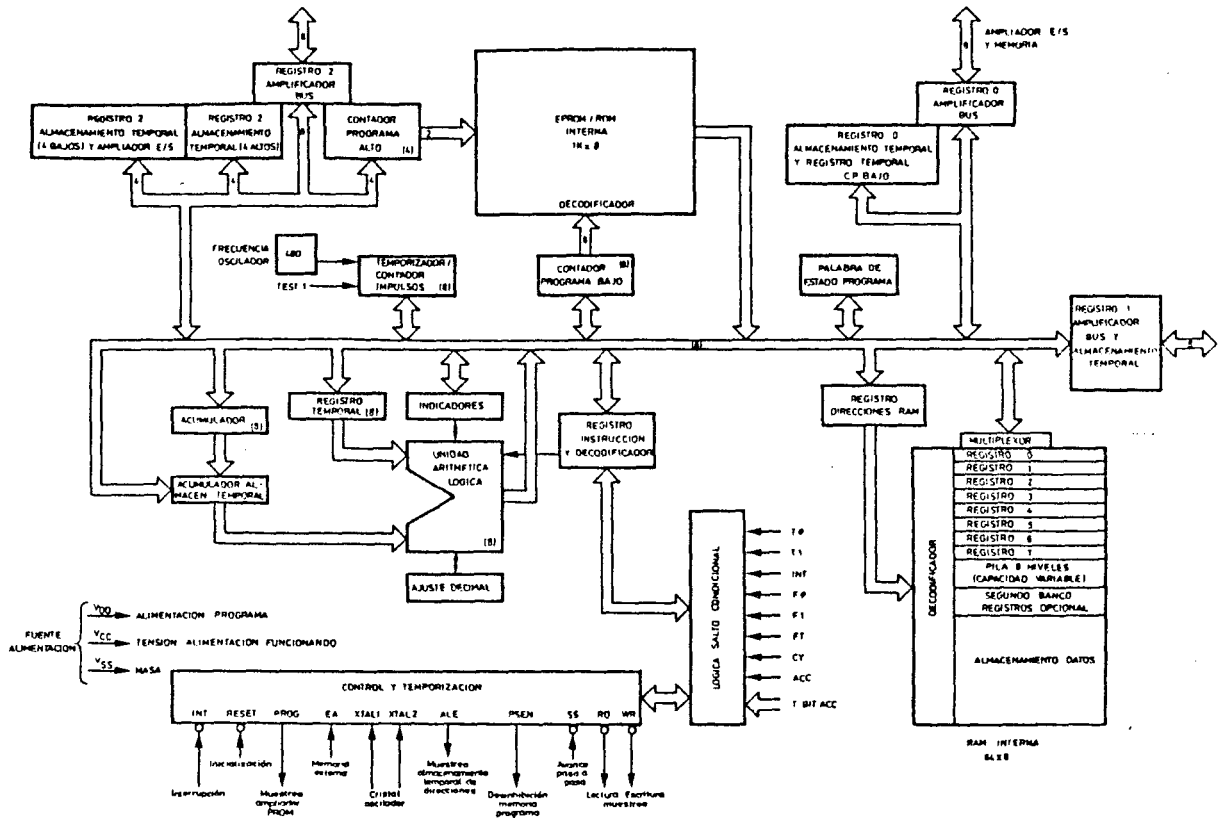
Nemónico	Descripción	Código		Código hexadecimal	N.º octetos	N.º ciclos
Indicadores						
CLR C	Puesta a cero de acarreo	1001	0111	97	1	1
CPL C	Complementar el acarreo	1010	0111	A7	1	1
CLR F0	Puesta a cero de F0	1000	0101	85	1	1
CPL F0	Complementar F0	1001	0101	95	1	1
CLR F1	Puesta a cero de F1	1010	0101	A5	1	1
CPL F1	Complementar F1	1011	0101	B5	1	1
Transferencia de datos						
MOV A,R	Transferencia de R a A	1111	1rrr	–	1	1
MOV A, @R	Transferencia de un dato direccionado por R a A	1111	000r	–	1	1
MOV A, # data	Carga inmediata dé A	0010	0011	23	2	2
MOV R,A	Transferencia de A a R	1010	1rrr	–	1	1
MOV @R,A	Transferencia de A a una posición de memoria direccionada por R	1010	000r	–	1	1
MOV R, # data	Carga inmediata de R	1011	1rrr	–	2	2
MOV @R, # data	Transferencia inmediata a una posición de memoria direccionada por R	1011	000r	–	2	2
MOV A,PSW	Transferencia de la palabra de estado (PSW) a A	1100	0111	C7	1	1
MOV PSW,A	Transferencia de A a la palabra de estado (PSW)	1101	0111	D7	1	1
XCH A,R	Cambio de A y R	0010	1rrr	–	1	1
XCH A,@R	Cambio de A y de un dato direccionado por R	0010	000r	–	1	1
XCHD A,@R	Cambio de A y 4 bits de un dato direccionado por R	0011	000r	–	1	1
MOVX A,@R	Transferencia de un dato direccionado por R a A	1000	000r	–	1	2
MOVX @R,A	Transferencia de A en la posición de memoria direccionada por R	1001	000r	–	1	2
MOVP A,@A	Transferencia de un dato direccionado por A a A	1010	0011	A3	1	2
MOVP3 A,@A	Transferencia de un dato direccionado por A a A	1110	0011	E3	1	2
Temporizador/contador						
MOV A,T	Transferencia del temporizador/contador a A	0100	0010	42	1	1
MOV T,A	Transferencia de A al temporizador/contador	0110	0010	62	1	1
STRT T	Arranque del temporizador	0101	0101	55	1	1
STRT CNT	Arranque del contador de impulsos	0100	0101	45	1	1
STOP TCNT	Para del temporizador o contador	0110	0101	65	1	1
EN TCNTI	Desinhibición de las interrupciones del temporizador/contador	0010	0101	25	1	1
DIS TCNTI	Inhibición de las interrupciones del	0011	0101		1	1

Nemónico	Descripción	Código		Código hexadecimal	N.º octetos	N.º ciclos
Control						
EN I	Desinhibición de las interrupciones externas	0000	0101	05	1	1
DIS I	Inhibición de las interrupciones externas	0001	0101	15	1	1
SEL RB0	Selección del banco de registros 0	1100	0101	C5	1	1
SEL RB1	Selección del banco de registros 1	1101	0101	D5	1	1
SEL MB0	Selección del banco de memoria 0	1110	0101	E5	1	1
SEL MB1	Selección del banco de memoria 1	1111	0101	F5	1	1
ENTO CLK	Salida del reloj interno por TO	0111	0101	75	1	1
NOP	No operación	0000	0000	00	1	1

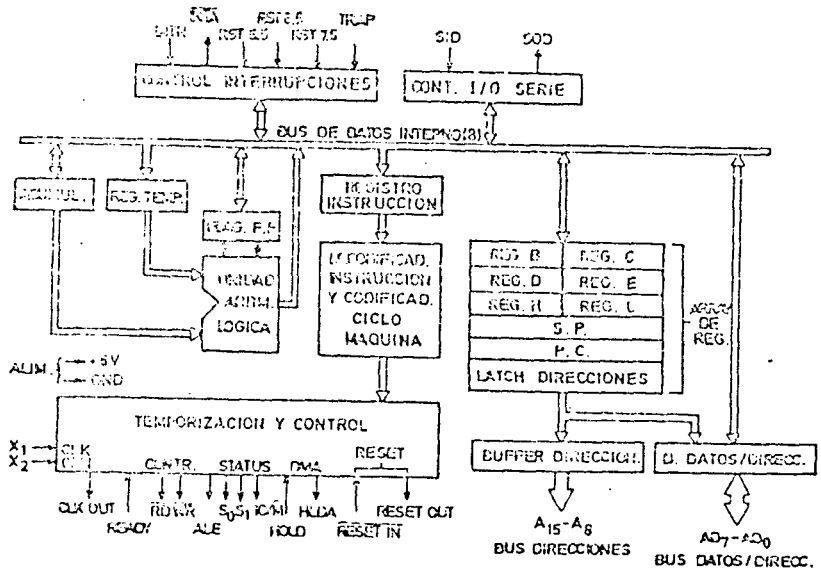
FIG 2.B REPERTORIO DE INSTRUCCIONES DE LA MCS-48.

A P E D I C E C

_ REFERENTE AL CAPITULO TERCERO.



MICROCOMPUTADOR 8048



MICROPROCESADOR 8085

FIG 1.C Estructura interna del 8048 y 8085 de Intel

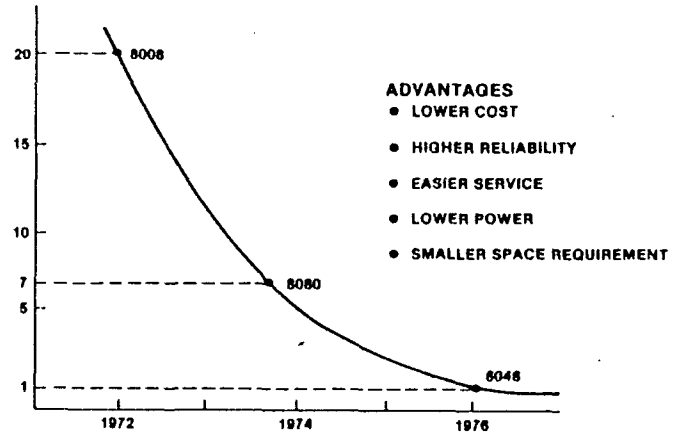


FIG 2.C Incidencia de la MCS 48 en la disminucion de los componentes del sistema.

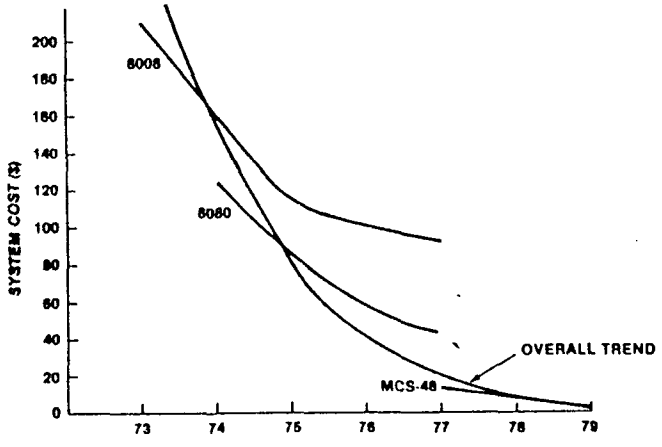


FIG 3.C Incidencia de la MCS 48 en la disminucion de los costes de un sistemas

A P E N D I C E D

REFERENTE AL CAPITULO CUARTO.

```
10 REM  ************** MACRO ENSAMBLADOR MCS 48  ***************
20 D2$="0"
30 CON=0
40 VD=0
50 VF=0
60 VO=0
70 VF2=0
80 KEY OFF
90 CLS
100 GOSUB 120
110 GOTO 290
120 PRINT:PRINT:PRINT:PRINT:PRINT
130 PRINT TAB(25)"*************************"
140 PRINT TAB(25)"MACRO ENSAMBLADOR MCS-48"
150 PRINT TAB(25)"*************************"
160 PRINT
170 PRINT TAB(25)"   ---- M E N U ----   "
180 PRINT
190 PRINT
200 PRINT TAB(20)"(1) - EDITAR"
210 PRINT TAB(20)"(2) - ENSANBLAR"
220 PRINT TAB(20)"(3) - MODIFICAR LINEAS"
230 PRINT TAB(20)"(4) - INSERTAR LINEAS "
240 PRINT TAB(20)"(5) - GUARDAR FICHERO "
250 PRINT TAB(20)"(6) - CARGAR FICHERO  "
260 PRINT TAB(20)"(7) - IMPRIMIR "
270 PRINT TAB(20)"(8) - FINALIZACION "
280 RETURN
290 A$=INKEY$
300 IF A$="" THEN GOTO 290
310 IF A$="1" THEN GOTO 410
320 IF A$="2" THEN GOTO 700
330 IF A$="3" THEN GOSUB 2700
340 IF A$="4" THEN GOTO 3050
350 IF A$="5" THEN GOSUB  2240
360 IF A$="6" THEN GOSUB  2370
370 IF A$="7" THEN GOTO 2500
380 IF A$="8" THEN GOTO 3610
390 IF A$= "" THEN GOTO 290
400 GOTO 90
410 REM  ************************ EDICION DEL PROGRAMA  ************
420 CLS
430 DR=0
440 VF=1
450 PRINT:PRINT:PRINT:PRINT:PRINT
460 PRINT TAB(25)"(1) EDICION DEL PROGRAMA"
470 PRINT TAB(25)"----------------------"
480 PRINT:PRINT
490 PRINT TAB(20)"- PULSE CULAQUIER TECLA PARA COMENZAR LA EDICION."
500 PRINT TAB(20)"- ESCRIBA <END> PARA FINALIZAR LA EDICION."
```

```
510 IF INKEY$="" THEN GOTO 510
520 OPEN "FUENTE" AS#1 LEN=37
530 FIELD#1,37 AS F$
540 CLS
550 PRINT"LAS INSTRUCCIONES DEBEN ESCRIBIRSE CON MAYUSCULAS.
560 PRINT
570 LINE INPUT;A$
580 PRINT
590 CON=CON+1
600 LSET F$=A$
610 PUT #1,CON
620 IF A$="END" THEN CLOSE#1: GOTO 90
630 GOTO 570
640 OPEN "FUENTE" AS#1 LEN=37
650 FIELD#1,37 AS F$
660 LSET F$=A$
670 PUT #1,CON
680 CLOSE#1
690 RETURN
700 REM  ************************  ENSAMBLADO  ********************
710 CLS
720 VO=1
730 INPUT "DIRECCION DE COMIENZO (ORIGEN)",ORG
740 F=1
750 OPEN "DIR" AS#2 LEN=9
760 OPEN "FUENTE" AS#1 LEN=37
770 OPEN "OBJETO" AS#3 LEN=47
780 CLS
790 PRINT "ENSAMBLANDO........."
800 IF MID$(A$,1,3)="END" THEN GOTO 1250
810 FIELD#1,37 AS F$
820 GET #1,F
830 A$=F$
840 L=LEN(A$)
850 A=0
860 B=0
870 E=0
880 C=LEN(A$)
890 REM
900 FOR D=1 TO LEN(A$)
910 IF MID$(A$,D,1)=":"THEN A=D
920 IF MID$(A$,D,1)=";"THEN C=D
930 IF MID$(A$,D,1)="*"THEN B=D
940 IF MID$(A$,D,1)="#" THEN E=D
950 NEXT D
960 A1$="      ":A2$="          ":A3$="                    "
970 A6$="      " :A7$="   ":A5$="      "
980 IF A>0 THEN A1$=LEFT$(A$,A-1)
990 A3$=RIGHT$(A$,L-C)
1000 IF E>0 THEN GOTO 1050
```

```
1010 IF B=0 THEN GOTO 1080
1020 IF A=0 THEN A2$=LEFT$(A$,B-1) ELSE A2$=MID$(A$,A+1,B-1-A)
1030 IF C=L THEN A6$=RIGHT$(A$,L-B) ELSE A6$=MID$(A$,B+1,C-B-1)
1040 GOTO 1090
1050 IF A=0 THEN A2$=LEFT$(A$,E) ELSE A2$=MID$(A$,A+1,E-A)
1060 IF C=L THEN A7$=RIGHT$(A$,L-E) ELSE A7$=MID$(A$,E+1,C-E-1)
1070 GOTO 1090
1080 IF C=L THEN A2$=MID$(A$,A+1,C-A) ELSE A2$=MID$(A$,A+1,C-A-1)
1090 REM
1100 A5$=HEX$(ORG)
1110 IF A6$ ="      " THEN  GOTO 1130  ELSE ORG=ORG+2
1120 GOTO 1140
1130 IF A7$="  " THEN ORG=ORG+1 ELSE ORG=ORG+2
1140 IF A1$><"      " THEN GOSUB 2020
1150 FIELD#3,4 AS B5$,5 AS B1$,10 AS B2$,15 AS B3$,6 AS B4$,5 AS B6$,2
1160 LSET B1$=A1$
1170 LSET B2$=A2$
1180 LSET B3$=A3$
1190 LSET B6$=A6$
1200 LSET B5$=A5$
1210 LSET B7$=A7$
1220 PUT #3,F
1230 F=F+1
1240 GOTO 800
1250 ER=0
1260 CLOSE
1270 N=0:M=CON
1280 OPEN "OBJETO" AS#3 LEN=47
1290 OPEN "EN2" AS#2 LEN=16
1300 OPEN "DIR" AS#1 LEN=9
1310 C=1
1320 B2$="   "
1330 IF MID$(B2$,1,3)="END" THEN GOTO 1660
1340 FIELD #3,4 AS B5$,5 AS B1$,10 AS B2$,15 AS B3$,6 AS B4$,5 AS B6$,
1350 GET #3,C
1360 A1$=B1$
1370 A2$=B2$
1380 A6$=B6$
1390 A3$=B3$
1400 A7$=B7$
1410 A5$=B5$
1420 CNT=1
1430 FIELD#2, 10 AS C2$,6 AS C4$
1440 GET #2,CNT
1450 IF C2$=A2$ THEN GOTO 1490
1460 IF CNT>224 THEN GOTO 1530
1470 CNT=CNT+1
1480 GOTO 1440
1490 A4$=C4$
1500 IF A7$<>"   " THEN A4$=MID$(A4$,1,2) + A7$
```

```
1510 IF A6$><"      " THEN GOSUB 2090
1520 GOTO 1550
1530 A4$="ERR #1"
1540 ER=ER+1
1550 FIELD#3,4 AS B5$,5 AS B1$,10 AS B2$,15 AS B3$,6 AS B4$,5 AS B6$,2
1560 LSET B1$=A1$
1570 LSET B2$=A2$
1580 LSET B3$=A3$
1590 LSET B5$=A5$
1600 LSET B4$=A4$
1610 LSET B6$=A6$
1620 LSET B7$=A7$
1630 PUT #3,C
1640 C=C+1
1650 GOTO  1330
1660 CLS
1670 CLOSE
1680 N=0
1690 PRINT:PRINT "ENSAMBLADO COMPLETO : ";ER;" ERRORES."
1700 PRINT
1710 PRINT TAB(20) "ERR #1 : Error sintactico."
1720 PRINT TAB(20) "ERR #2 : Error de etiqueta."
1730 PRINT:PRINT
1740 PRINT TAB(20) "- PULSE < M > PARA VOLVER AL MENU PRINCIPAL."
1750 PRINT TAB(20) "- PULSE < A > PARA VISUALIZAR LA PAGINA ANTERIOR."
1760 PRINT TAB(20) "- PULSE < P > PARA VISUALIZAR LA PAGINA POSTERIOR.
1770 A$=INKEY$
1780 IF A$="M" THEN GOTO 90
1790 IF A$="A" THEN GOTO 1820
1800 IF A$="P" THEN GOTO 1850
1810 GOTO 1770
1820 IF N>0 THEN N=N-20
1830 GOSUB 1890
1840 GOTO 1770
1850 N=N+20
1860 GOSUB 1890
1870 GOTO 1770
1880 REM  ************ IMPRESION POR PANTALLA DEL OBJETO  **********
1890 CLS
1900 M=N+20
1910 PRINT"DIR  :ETQ   :INSTRUCCION      :MAQ   :COMENTARIO      :NUM.LI
1920 PRINT"-----:------:-----------------:------:----------------:------
1930 OPEN "OBJETO" AS#1 LEN= 47
1940 FIELD #1,4 AS B5$,5 AS B1$,10 AS B2$,15 AS B3$,6 AS B4$,5 AS B6$,2
1950 FOR C=N+1 TO M
1960 GET #1,C
1970 IF MID$(B2$,1,3)="END" THEN GOTO 2000
1980 PRINT B5$;" :";B1$;" :";B2$;B7$;B6$;":";B4$;";";B3$;":";C
1990 NEXT C
2000 CLOSE #1
```

```
2010 RETURN
2020 FIELD #2,5 AS D1$,4 AS D5$
2030 VD=1
2040 DR=DR+1
2050 LSET D1$=A1$
2060 LSET D5$=A5$
2070 PUT #2,DR
2080 RETURN
2090 FIELD #1,5 AS D1$,4 AS D5$
2100 FOR B=1 TO DR
2110 GET #1,B
2120 IF D1$=A6$ THEN GOTO 2170
2130 NEXT B
2140 A4$="ERR #2"
2150 ER=ER+1
2160 RETURN
2170 IF RIGHT$(D5$,1)=" " THEN D5$=D2$ +LEFT$(D5$,3)
2180 IF RIGHT$(D5$,1)=" " THEN GOTO 2170
2190 A4$=MID$(A4$,1,2)+MID$(D5$,3,2)
2200 IF A2$="JMP        " THEN A4$=MID$(D5$,2,1) +MID$(A4$,2,3)
2210 IF A2$="CALL       " THEN A4$=MID$(D5$,2,1) +MID$(A4$,2,3)
2220 RETURN
2230 REM  ***************  GUARDAR FICHERO  ********************
2240 CLS
2250 INPUT "¿ QUE NOMBRE DESEA UTILIZAR : ";FIC$
2260 CLS
2270 PRINT"CREANDO FICHERO........"
2280 FIO$=FIC$ + ".OBJ"
2290 FIF$=FIC$ + ".FUE"
2300 NAME "OBJETO" AS  FIO$
2310 NAME "FUENTE" AS FIF$
2320 OPEN "FUENTE" AS#1
2330 OPEN "OBJETO" AS#2
2340 CLOSE
2350 RETURN
2360 REM  ***************  CARGAR FICHERO  ********************
2370 CLS
2380 INPUT "NOMBRE DEL FICHERO ";FIC$
2390 PRINT"CARGANDO FICHERO......."
2400 OPEN "FUENTE" AS#1
2410 OPEN "OBJETO" AS#2
2420 CLOSE
2430 KILL "FUENTE":KILL "OBJETO"
2440 FIO$=FIC$ + ".OBJ"
2450 FIF$=FIC$ + ".FUE"
2460 NAME FIO$ AS "OBJETO"
2470 NAME FIF$ AS "FUENTE"
2480 RETURN
2490 REM  ******************  IMPRESION  ********************************
2500 CLS
```

```
2510 REM LPRINT CHR$(15);
2520 PRINT"IMPRIMIENDO............."
2530 LPRINT"NUM.LIN.:DIR  :ETQ   :INSTRUCCION       :MAQ   :COMENTARIO
          "
2540 LPRINT"--------:-----:------:----------------:------:----------
----------"
2550 C=1
2560 OPEN "OBJETO" AS#1 LEN= 47
2570 FIELD #1,4 AS B5$,5 AS B1$,10 AS B2$,15 AS B3$,6 AS B4$,5 AS B6$,2
2580 IF MID$(B2$,1,3)="END" THEN GOTO 2630
2590 GET #1,C
2600 LPRINT C;TAB(9)":"; B5$;" :";B1$;" :";B2$;B7$;B6$;":";B4$;";";B3$
2610 C=C+1
2620 GOTO 2580
2630 CLOSE #1
2640 LPRINT:LPRINT "ENSAMBLADO COMPLETO :";ER;" ERRORES."
2650 LPRINT
2660 LPRINT TAB(20) "ERR #1 : ERROR SINTACTICO."
2670 LPRINT TAB(20) "ERR #2 : ERROR DE ETIQUETA."
2680 LPRINT:LPRINT:LPRINT:LPRINT
2690 GOTO 90
2700 REM  ****************  MODIFICAR LINEA  ***************************
2710 CLS
2720 PRINT:PRINT:PRINT:PRINT:PRINT
2730 PRINT TAB(25)"(3) - MODIFICACION DE LINEAS."
2740 PRINT TAB(25)"-----------------------------"
2750 PRINT:PRINT
2760 PRINT TAB(20) "- PULSE <M> PARA VOLVER AL MENU PRINCIPAL."
2770 PRINT TAB(20) "- PULSE <A> PARA VISUALIZAR LA PAGINA ANTERIOR."
2780 PRINT TAB(20) "- PULSE <P> PARA VISUALIZAR LA PAGINA POSTERIOR."
2790 N=0
2800 A$=INKEY$
2810 IF A$="" THEN GOTO 2800
2820 IF A$="M" THEN GOTO 90
2830 IF A$="P" THEN N=N+20
2840 IF A$="A" AND N>0 THEN N=N-20
2850 GOSUB 2920
2860 LINE INPUT;A$
2870 IF LEN(A$)<2 THEN GOTO 2820
2880 CON=VAL(MID$(A$,1,5))
2890 A$=MID$(A$,10,10)
2900 GOSUB 640
2910 GOTO 2860
2920 CLS
2930 M=N+20
2940 PRINT"NUN.LIN :CONTENIDO
2950 PRINT"---------:-------------------------------------------------"
2960 OPEN "FUENTE" AS#1 LEN=37
2970 FIELD#1,37 AS F$
2980 FOR C=N+1 TO M
2990 GET #1,C
3000 PRINT C; TAB(9)":";F$
```

```
3010 NEXT C
3020 CLOSE#1
3030 RETURN
3040 REM  ********** INSERTAR LINEAS  **********
3050 CLS
3060 CLS
3070 PRINT:PRINT:PRINT:PRINT:PRINT
3080 PRINT TAB(25) "(4) - INSERCION DE LINEAS."
3090 PRINT TAB(25) "-------------------------"
3100 PRINT:PRINT
3110 PRINT TAB(20)"- PULSE  <M> PARA VOLVER AL MENU PRINCIPAL."
3120 PRINT TAB(20)"- PULSE  <A> PARA VISUALIZAR LA PAGINA ANTERIOR."
3130 PRINT TAB(20)"- PULSE  <P> PARA VISUALIZAR LA PAGINA POSTERIOR."
3140 A$=INKEY$
3150 IF A$="" THEN GOTO 3140
3160 IF A$="M" THEN GOTO 90
3170 IF A$="P" THEN N=N+20
3180 IF A$="A" AND N>20 THEN N=N-20
3190 CLS
3200 GOSUB 2930
3210 INPUT "NUMERO LINEA A INSERTAR ¿ ",CON
3220 CON2=CON
3230 OPEN "FUEN2" AS#2 LEN=37
3240 FIELD #2, 37 AS F2$
3250 OPEN "FUENTE" AS#1 LEN=37
3260 FIELD #1, 37 AS F$
3270 D=1
3280 GET #1,CON
3290 LSET F2$=F$
3300 PUT #2,D
3310 CON=CON+1
3320 D=D+1
3330 IF MID$(F$,1,3)<>"END" THEN GOTO 3280
3340 CLOSE
3350 CLS
3360 PRINT "- COMIENZE A INSERTAR...."
3370 PRINT "- PARA FINALIZAR LA INSERCION PULSE </>."
3380 PRINT "-----------------------------------------"
3390 PRINT
3400 CON=CON2
3410 LINE INPUT;A$
3420 PRINT
3430 IF A$= "/" THEN GOTO 3470
3440 GOSUB 640
3450 CON=CON+1
3460 GOTO 3410
3470 OPEN "FUEN2" AS#2 LEN=37
3480 FIELD #2, 37 AS F2$
3490 OPEN "FUENTE" AS#1 LEN=37
3500 FIELD #1, 37 AS F$
```

```
3510 C=1
3520 FOR C=1 TO D-1
3530 GET #2,C
3540 LSET F$=F2$
3550 PUT #1,CON
3560 CON=CON+1
3570 NEXT C
3580 CLOSE
3590 GOTO 3050
3600 REM  ***************  FINALIZACION  ******************************
3610 CLS
3620 IF VD=1 THEN KILL "DIR"
3630 IF VF2=1 THEN KILL "FUEN2"
3640 PRINT "DESEA CONSERVAR SU PROGRAMA ¿ (S/N)"
3650 A$=INKEY$
3660 IF A$="" THEN GOTO 3650
3670 IF A$="S" THEN GOTO 3700
3680 IF A$="N" THEN GOTO 3720
3690 GOTO 3650
3700 GOSUB 2240
3710 GOTO 3740
3720 IF VF=1 THEN KILL "FUENTE"
3730 IF VO=1 THEN KILL "OBJETO"
3740 CLS
3750 FOR C=1 TO 18:PRINT
3760 NEXT C
3770 PRINT TAB(25) "MACRO ENSAMBLADOR MCS-48. (M.V.O    ** 1985 **)
3780 END
3790 N=0
```

LISTADO DEL FICHERO ENS2 QUE CONTIENE LOS CODIGOS MAQUINA Y ENSAMBLADOR DE LA FAMILIA MCS-48.

1	ADD A.R0	68
2	ADD A.R1	69
3	ADD A.R2	6A
4	ADD A.R3	6B
5	ADD A.R4	6C
6	ADD A.R5	6D
7	ADD A.R6	6E
8	ADD A.R7	6F
9	ADD A.@R0	60
10	ADD A.@R1	61
11	ADD A.#	03XX
12	ADDC A.R0	78
13	ADDC A.R1	79
14	ADDC A.R2	7A
15	ADDC A.R3	7B
16	ADDC A.R4	7C
17	ADDC A.R5	7D
18	ADDC A.R6	7E
19	ADDC.R7	7F
20	ADDC A.@R0	70
21	ADDC A.@R1	71
22	ADDC A.#	13XX
23	ANL A.R0	58
24	ANL A.R1	59
25	ANL A.R2	5A
26	ANL A.R3	5B
27	ANL A.R4	5C
28	ANL A.R5	5D
29	ANL A.R6	5E
30	ANL A.R7	5F
31	ANL A.@R0	50
32	ANL A.@R1	51
33	ANL A.#	53XX
34	ANL BUS.#	98XX
35	ANL P0.#	98XX
36	ANL P1.#	99XX
37	ANL P2.#	9AXX
38	ANL P4.A	9C
39	ANL P5.A	9D
40	ANL P6.A	9E
41	ANL P7.A	9F
42	CALL	X4XX
43	CLR A	27
44	CLR C	97
45	CLR F1	A5
46	CLR F0	85
47	CPL A	37
48	CPL C	A7
49	CPL F0	95
50	CPL F1	B5

51	DAA	57
52	DEC A	07
53	DEC R0	C8
54	DEC R1	C9
55	DEC R2	CA
56	DEC R3	CB
57	DEC R4	CC
58	DEC R5	CE
59	DEC R6	CE
60	DEC R7	CF
61	DIS I	15
62	DIS TCNTI	35
63	DJNZ R0	E8XX
64	DJNZ R1	E9XX
65	DJNZ R2	EAXX
66	DJNZ R3	EBXX
67	DJNZ R4	ECXX
68	DJNZ R5	EDXX
69	DJNZ R6	EEXX
70	DJNZ R7	EFXX
71	EN I	05
72	EN TCNTI	25
73	ENTO CLK	75
74	IN A.P0	08
75	IN A.P1	09
76	IN A.P2	0A
77	INC A	17
78	INC R0	18
79	INC R1	19
80	INC R2	1A
81	INC R3	1B
82	INC R4	1C
83	INC R5	1D
84	INC R6	1E
85	INC R7	1F
86	INC @R0	10
87	INC @R1	11
88	JB0	12XX
89	JB1	32XX
90	JB2	52XX
91	JB3	72XX
92	JB4	92XX
93	JB5	B2XX
94	JB6	D2XX
95	JB7	F2XX
96	JC	F6XX
97	JF0	B6XX
98	JF1	76XX
99	JMP	X4XX
100	JMPP @A	B3

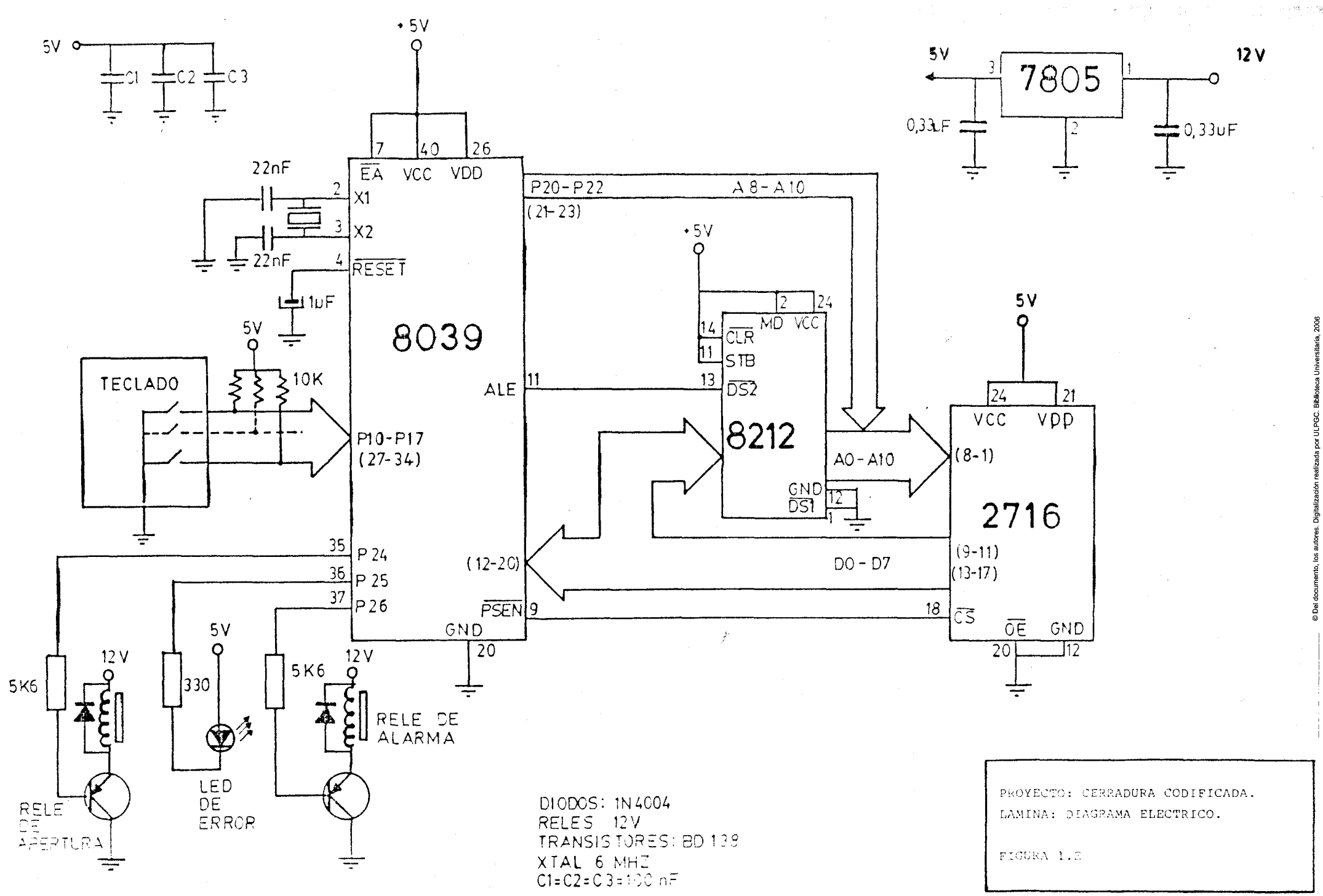
```
101             JNC             E6XX
102             JNI             86XX
103             JNTO            26XX
104             JNT1            46XX
105             JNZ             96XX
106             JTF             16XX
107             JTO             36XX
108             JT1             56XX
109             JZ              C6XX
110             MOV A.#         23XX
111             MOV A.R0        F8
112             MOV A.R1        F9
113             MOV A.R2        FA
114             MOV A.R3        FB
115             MOV A.R4        FC
116             MOV A.R5        FD
117             MOV A.R6        FE
118             MOV A.R7        FF
119             MOV A.@R0       F0
120             MOV A.@R1       F1
121             MOV A.T         42
122             MOV PSW.A       D7
123             MOV A.PSW       C7
124             MOV R0.A        A8
125             MOV R1.A        A9
126             MOV R2.A        AA
127             MOV R3.A        AB
128             MOV R4.A        AC
129             MOV R5.A        AD
130             MOV R6.A        AE
131             MOV R7.A        AF
132             MOV R0.#        B8XX
133             MOV R1.#        B9XX
134             MOV R2.#        BAXX
135             MOV R3.#        BBXX
136             MOV R4.#        BCXX
137             MOV R5.#        BDXX
138             MOV R6.#        BEXX
139             MOV R7.#        BFXX
140             MOV @R0.A       A0
141             MOV @R1.A       A1
142             MOV @R0.#       B0XX
143             MOV @R1.#       B1XX
144             MOV T.A         62
145             MOVD A.P4       0C
146             MOVD A.P5       0D
147             MOVD A.P6       0E
148             MOVD A.P7       0F
149             MOVD P4.A       3C
150             MOVD P5.A       3D
```

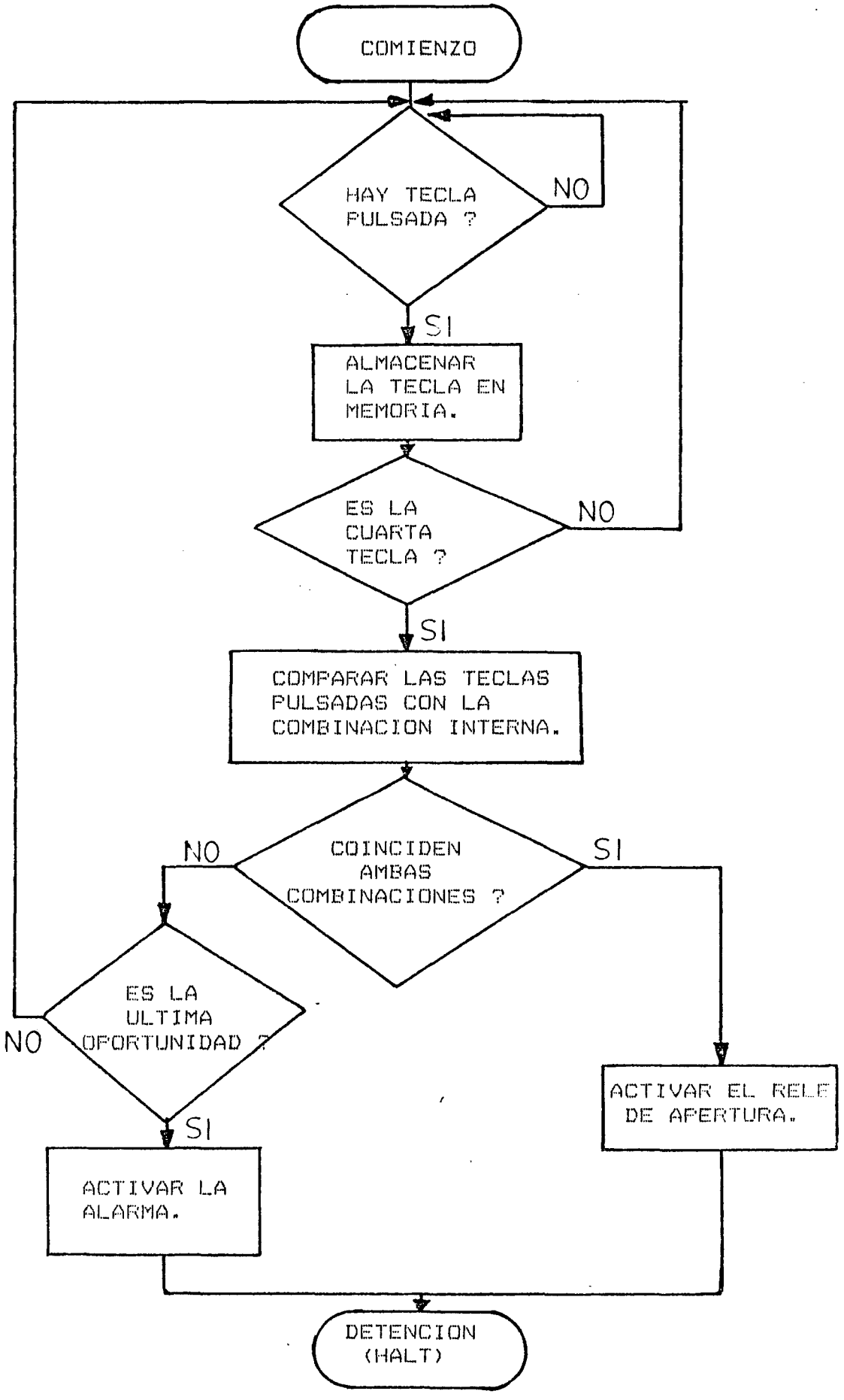
```
151             MOVD P6.A       3E
152             MOVD P7.A       3F
153             MOVP A.@A       A3
154             MOVP3 A.@A      E3
155             MOVX A.@R0      80
156             MOVX A.@R1      81
157             MOVX @R0.A      90
158             MOVX @R1.A      91
159             NOP             00
160             ORL A.R0        48
161             ORL A.R1        49
162             ORL A.R2        4A
163             ORL A.R3        4B
164             ORL A.R4        4C
165             ORL A.R5        4D
166             ORL A.R6        4E
167             ORL A.R7        4F
168             ORL A.@R0       40
169             ORL A.@R1       41
170             ORL A.#         43XX
171             ORL BUS.#       88XX
172             ORL P0.#        88XX
173             ORL P1.#        89XX
174             ORL P2.#        8AXX
175             ORLD P4.A       8C
176             ORLD P5.A       8D
177             ORLD P6.A       8E
178             ORLD P7.A       8F
179             OUTL P0.A       90
180             OUTL BUS.A      02
181             OUTL P1.A       39
182             OUTL P2.A       3A
183             RAD             80
184             RET             83
185             RETI            93
186             RETR            93
187             RLA             E7
188             RLC A           F7
189             RRA             77
190             RRC A           67
191             SEL AN0         95
192             SEL AN1         85
193             SEL MB0         E5
194             SEL MB1         F5
195             SEL RB0         C5
196             SEL RB1         D5
197             STOP TCNT       65
198             STRT CNT        45
199             STRT T          55
200             SWAP A          47
```

```
201             XCH A.R0        28
202             XCH A.R1        29
203             XCH A.R2        2A
204             XCH A.R3        2B
205             XCH A.R4        2C
206             XCH A.R5        2D
207             XCH A.R6        2E
208             XCH A.R7        2F
209             XCH A.@R0       20
210             XCH A.@R1       21
211             XCHD A.@R0      30
212             XCHD A.@R1      31
213             XRL A.R0        D8
214             XRL A.R1        D9
215             XRL A.R2        DA
216             XRL A.R3        DB
217             XRL A.R4        DC
218             XRL A.R5        DD
219             XRL A.R6        DE
220             XRL A.R7        DF
221             XRL A.@R0       D0
222             XRL A.@R1       D1
223             XRL A.#         D3XX
224             DB              XX
225             END             **
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
```

A P E N D I C E E

_ REFERENTE AL CAPITULO QUINTO.

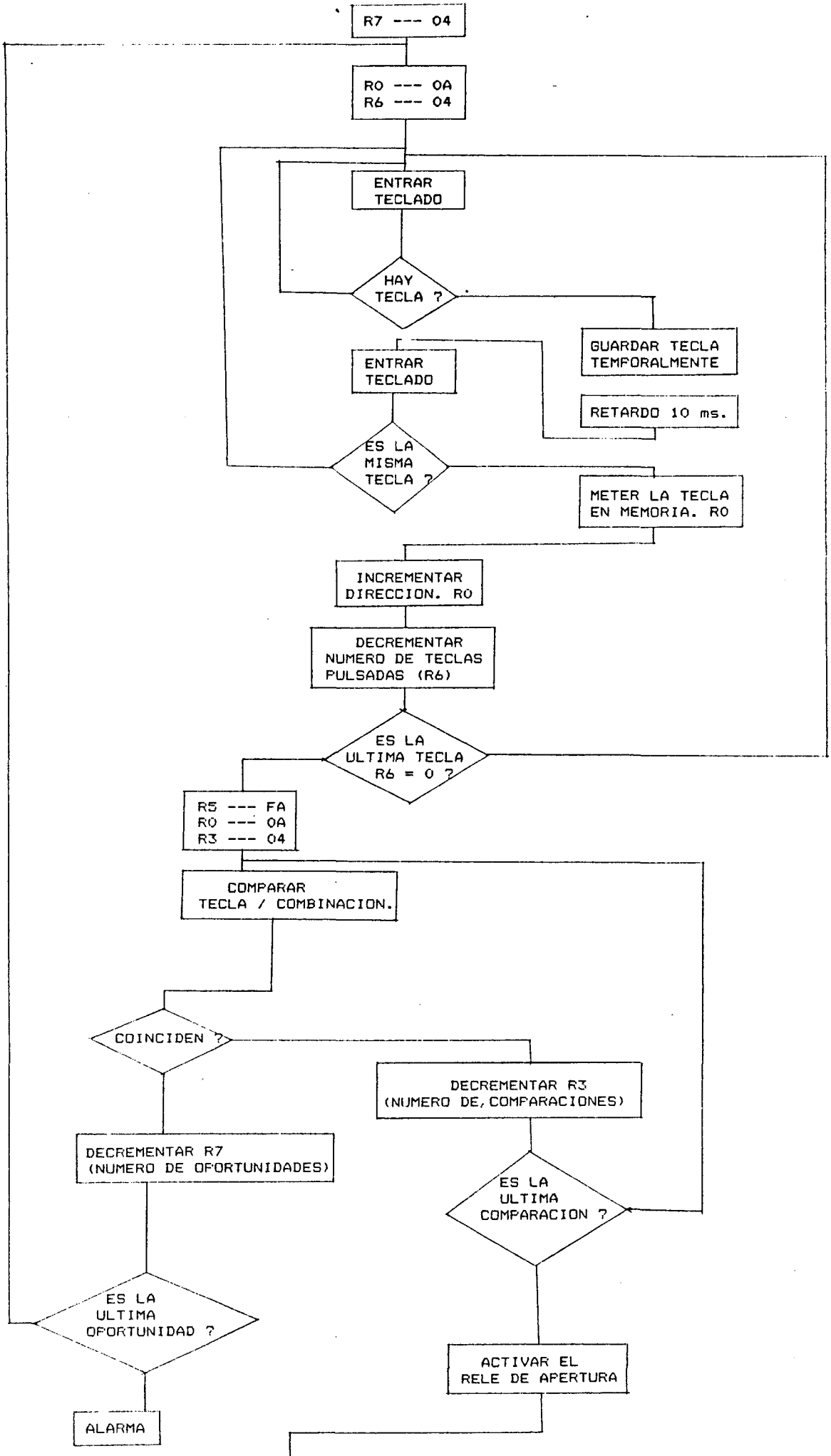




PROYECTO:CERRADURA CODIFICADA

LAMINA: ORGANIGRAMA SIMPLIFICADO-

FIGURA 2.E



PROYECTO: CERRDURA CODIFICADA

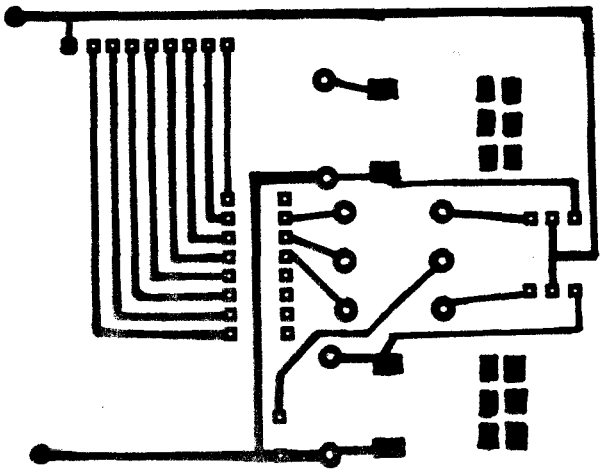
LAMINA: ORGANIGRAMA ELABORADO.

FIGURA 3.E

FIG 4.E ENSAMBLADO CERRADURA CODIFICADA.

NUM.LIN.	DIR	ETQ	INSTRUCCION		MAQ	COMENTARIO
1	0		MOV R7.#	04	BF04	;
2	2	CINCO	MOV R0.#	20	B820	;
3	4		MOV R6.#	04	BE04	;
4	6	UNO	IN A.P1		09	;
5	7		CPL A		37	;
6	8		JZ	UNO	C606	;
7	A		CPL A		37	;
8	B		MOV A.#	FF	23FF	;
9	D		OUTL P2.A		3A	;
10	E		MOV R1.A		A9	;
11	F		MOV R2.#	19	BA19	;
12	11	DOS	MOV R4.#	64	BC64	;
13	13	BUCLE	DEC R4		CC	;
14	14		MOV A.R4		FC	;
15	15		JNZ	BUCLE	9613	;
16	17		DEC R2		CA	;
17	18		MOV A.R2		FA	;
18	19		JNZ	DOS	9611	;
19	1B		NOP		00	;
20	1C		IN A.P1		09	;
21	1D		CPL A		37	;
22	1E		JZ	UNO	C606	;
23	20		CPL A		37	;
24	21		XRL A.R1		D9	;
25	22		NOP		00	;
26	23		JNZ	UNO	9606	;
27	25		MOV A.R1		F9	;
28	26		MOV @R0.A		A0	;
29	27		INC R0		18	;
30	28	SEIS	IN A.P1		09	;
31	29		CPL A		37	;
32	2A		JNZ	SEIS	9628	;
33	2C		DEC R6		CE	;
34	2D		MOV A.R6		FE	;
35	2E		JNZ	UNO	9606	;
36	30		MOV R5.#	FA	BDFA	;
37	32		MOV R0.#	20	B820	;
38	34		MOV R3.#	04	BB04	;
39	36	PRIN	MOV A.R5		FD	;
40	37		MOVP A.@A		A3	;
41	38		XRL A.@R0		D0	;
42	39		JNZ	TRES	9647	;
43	3B		INC R5		1D	;
44	3C		INC R0		18	;
45	3D		DEC R3		CB	;
46	3E		MOV A.R3		FB	;

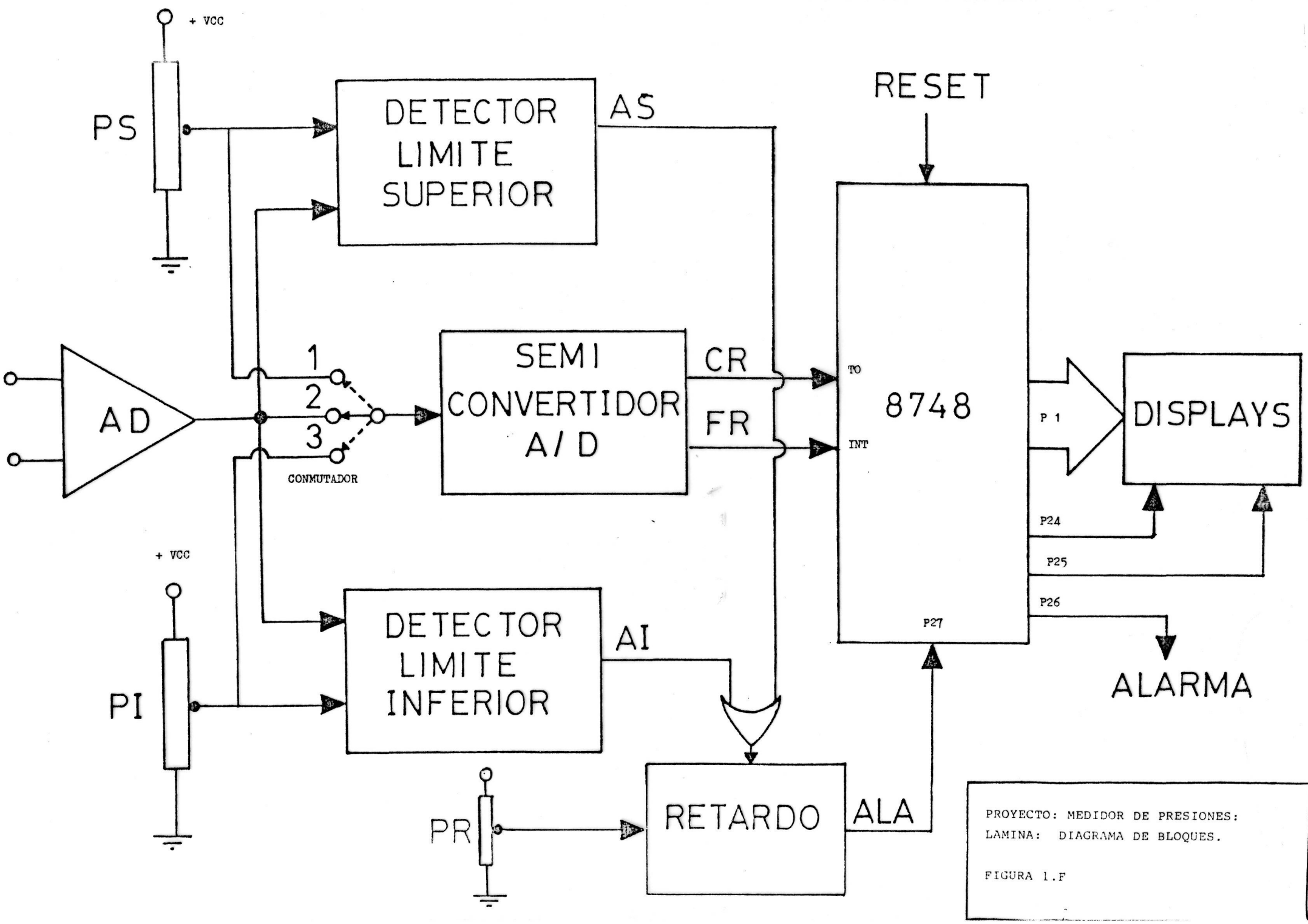
```
47      :3F   :       :JNZ          PRIN :9636  ;
48      :41   :       :MOV A.#   EF      :23EF  ;
49      :43   :       :OUTL P2.A         :3A    ;
50      :44   :CTO    :NOP               :00    ;
51      :45   :       :JMP          CTO  :0444  ;
52      :47   :TRES   :DEC R7            :CF    ;
53      :48   :       :MOV A.#   DF      :23DF  ;
54      :4A   :       :OUTL P2.A         :3A    ;
55      :4B   :       :MOV A.R7          :FF    ;
56      :4C   :       :JZ           ALAR :C650  ;
57      :4E   :       :JMP          CINCO:0402  ;
58      :50   :ALAR   :MOV A.#   BF      :23BF  ;
59      :52   :       :OUTL P2.A         :3A    ;
60      :53   :       :JMP          CTO  :0444  ;
61      :55   :       :END               :/     ;

ENSAMBLADO COMPLETO : 0  ERRORES.

                    ERR #1 : ERROR SINTACTICO.
                    ERR #2 : ERROR DE ETIQUETA.
```

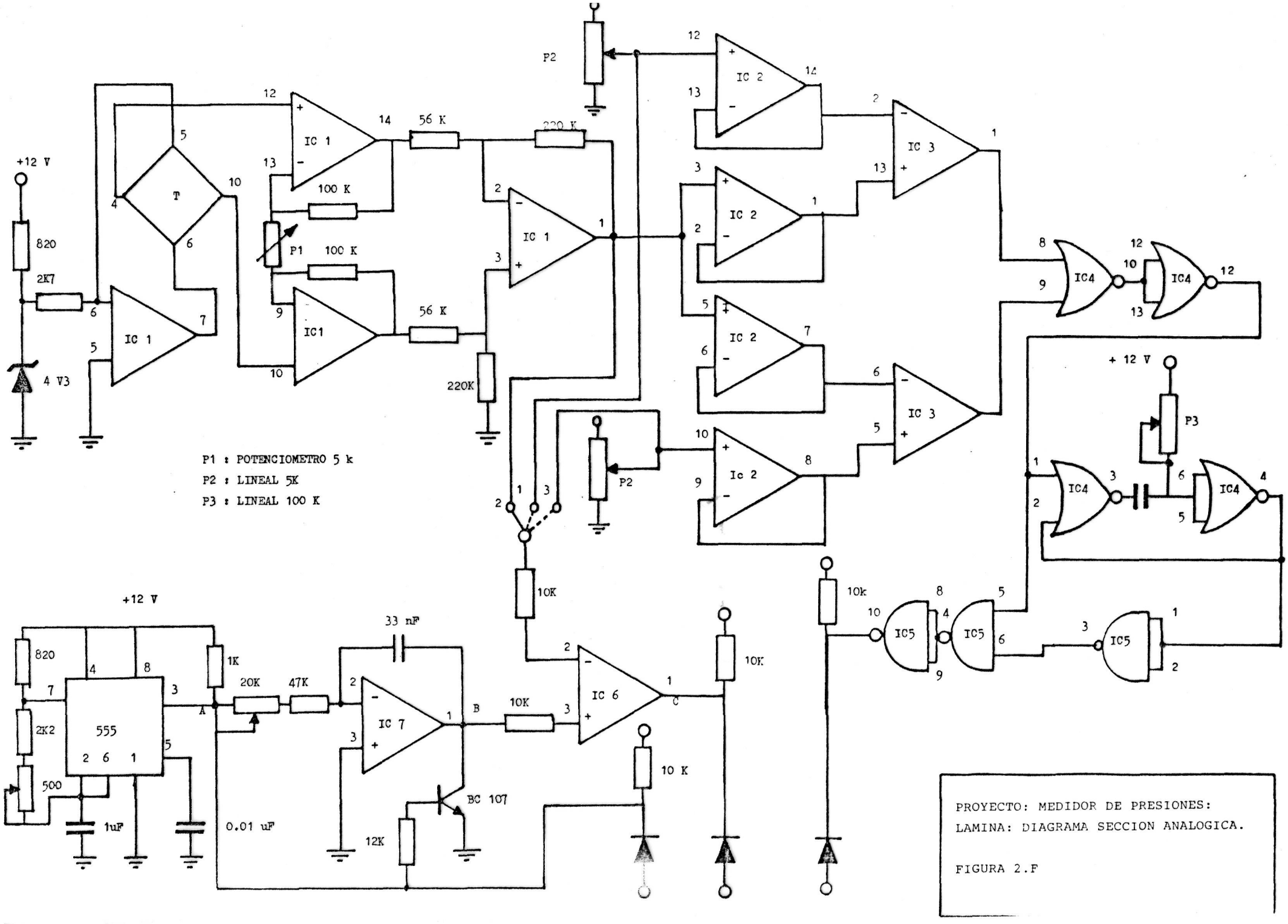


PLACA DE LOS RELES (VISTA SUPERIOR)

A P E N D I C E F

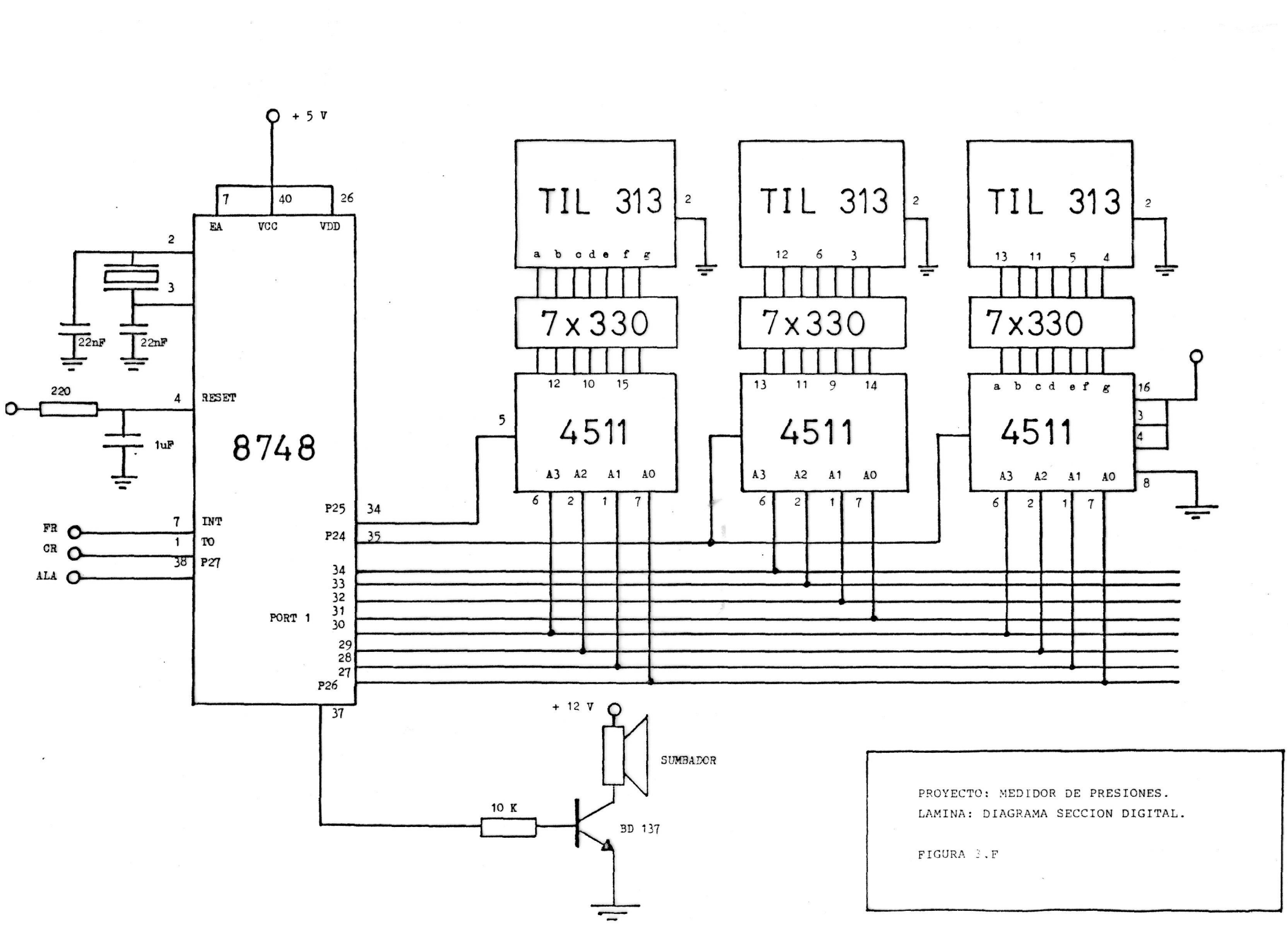
_ REFERENTE AL CAPITULO SEXTO.





PROYECTO: MEDIDOR DE PRESIONES:
LAMINA: DIAGRAMA SECCION ANALOGICA.

FIGURA 2.F



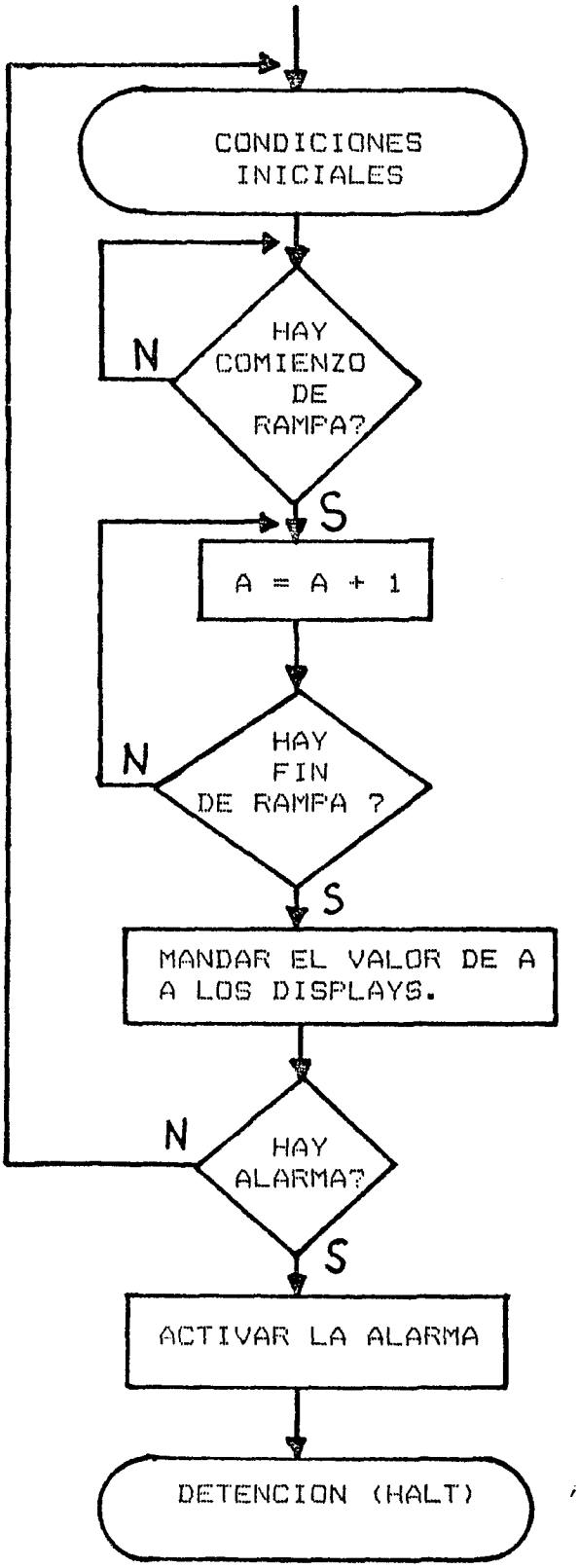


FIG 4.F ORGANIGRAMA SIMPLIFICADO DEL MEDIDOR DE PRESIONES.

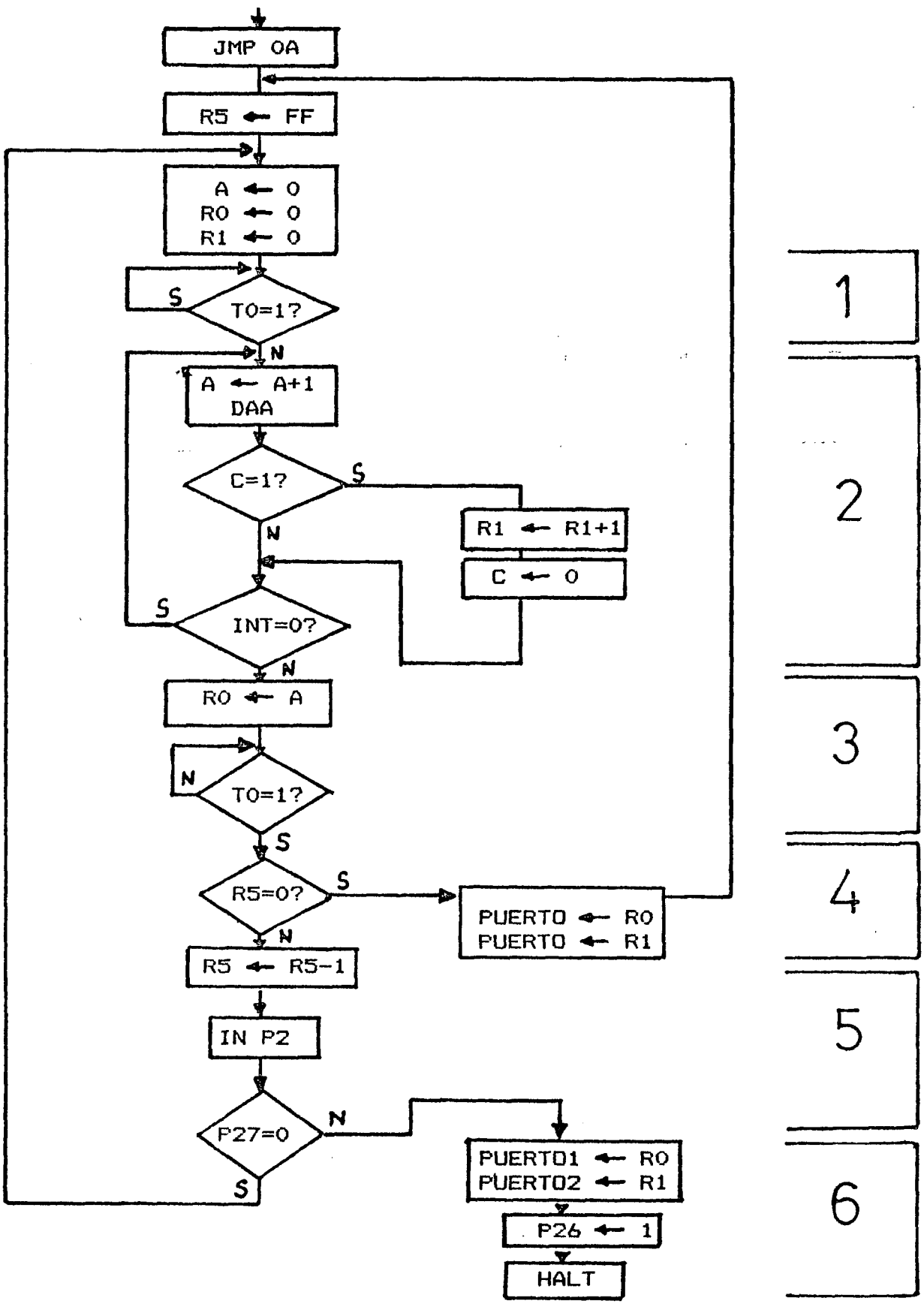
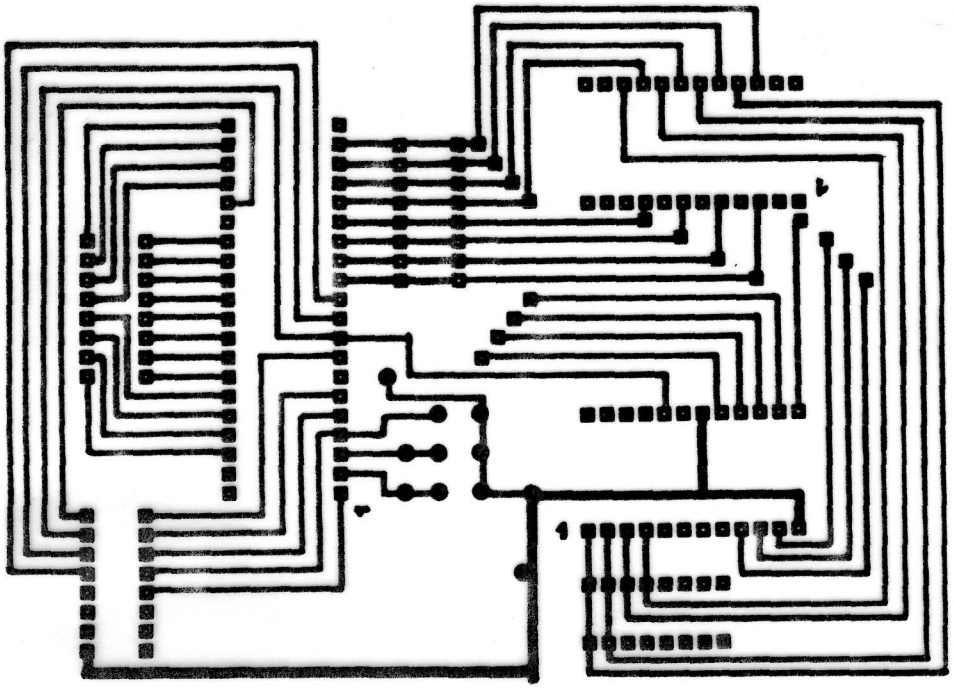


FIG 5.F ORGANIGRAMA ELABORADO DEL MEDIDOR DE PRESIONES.

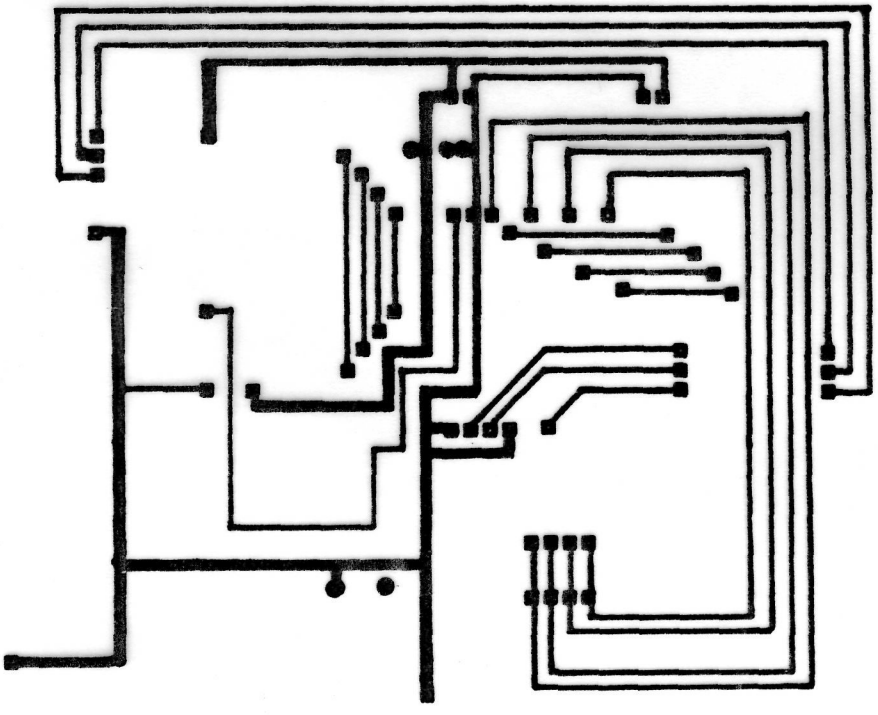
FIG 6.F ENSAMBLADO MEDIDOR DE PRESIONES.

NUM.LIN.	DIR	ETQ	INSTRUCCION		MAQ	COMENTARIO
1	0		DIS I		15	;
2	1		JMP	UNO	040C	;
3	3		MOV A.R7		FF	;
4	4		MOV A.R7		FF	;
5	5		MOV A.R7		FF	;
6	6		MOV A.R7		FF	;
7	7		MOV A.R7		FF	;
8	8		MOV A.R7		FF	;
9	9		MOV A.R7		FF	;
10	A		MOV A.R7		FF	;
11	B		MOV A.R7		FF	;
12	C	UNO	CLR A		27	;
13	D		MOV PSW.A		D7	;
14	E		MOV R5.#	FF	BDFF	;
15	10	DOS	CLR A		27	;
16	11		MOV R0.A		A8	;
17	12		MOV R1.A		A9	;
18	13	TRES	NOP		00	;
19	14		JTO	TRES	3613	;
20	16		MOV A.#	00	2300	;
21	18	CTRO	INC A		17	;
22	19		DAA		57	;
23	1A		JNC	CINCO	E61E	;
24	1C		INC R1		19	;
25	1D		CLR C		97	;
26	1E	CINCO	JNI		86XX	;
27	20		MOV R0.A		A8	;
28	21	SEIS	NOP		00	;
29	22		JNTO	SEIS	2621	;
30	24		DEC R5		CE	;
31	25		MOV A.R5		FD	;
32	26		JZ	OCHO	C63F	;
33	28		IN A.P2		0A	;
34	29		ANL A.#	80	5380	;
35	2B		JZ	DOS	C610	;
36	2D		MOV A.#	EF	23EF	;
37	2F		OUTL P2.A		3A	;
38	30		MOV A.R0		F8	;
39	31		OUTL P1.A		39	;
40	32		MOV A.#	FF	23FF	;
41	34		OUTL P2.A		3A	;
42	35		MOV A.#	DF	23DF	;
43	37		OUTL P2.A		3A	;
44	38		MOV A.R1		F9	;
45	39		OUTL P1.A		39	;
46	3A		MOV A.#	FF	23FF	;
47	3C		OUTL P2.A		3A	;
48	3D		MOV A.#	BF	23BF	;
49	3F		OUTL P2.A		3A	;
50	40	SIETE	NOP		00	;

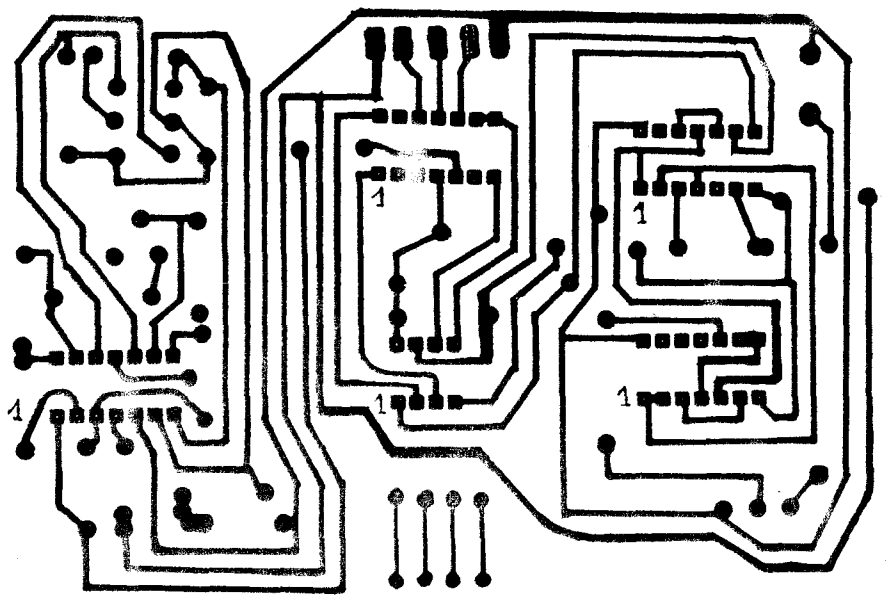
```
51      :41   :       :JMP          SIETE:043C  ;
52      :43   :OCHO   :MOV A.#   EF      :23EF  ;
53      :45   :       :OUTL P2.A        :3A    ;
54      :46   :       :MOV A.R0         :F8    ;
55      :47   :       :OUTL P1.A        :39    ;
56      :48   :       :MOV A.#   FF     :23FF  ;
57      :4A   :       :OUTL P2.A        :3A    ;
58      :4B   :       :MOV A.#   DF     :23DF  ;
59      :4D   :       :OUTL P2.A        :3A    ;
60      :4E   :       :MOV A.R1         :F9    ;
61      :4F   :       :OUTL P1.A        :39    ;
62      :50   :       :MOV A.#   FF     :23FF  ;
63      :52   :       :OUTL P2.A        :3A    ;
64      :53   :       :JMP          UNO :040C  ;
65      :55   :       :END              :**    ;

ENSAMBLADO COMPLETO : 0  ERRORES.

                    ERR #1 : ERROR SINTACTICO.
                    ERR #2 : ERROR DE ETIQUETA.
```



PLACA MICROCOMPUTADOR.(VISTA SUPERIOR)



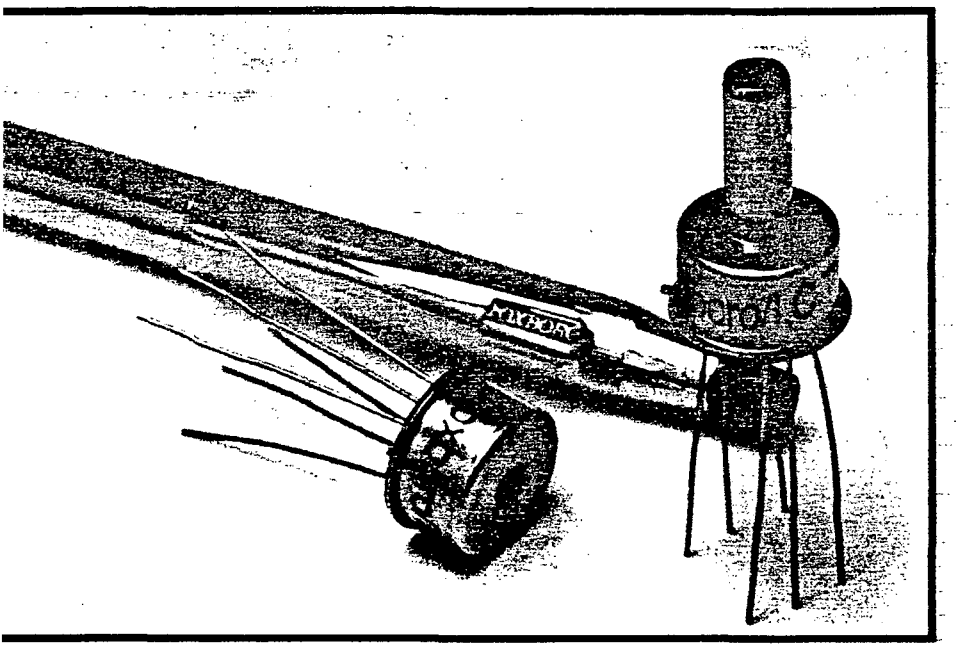
PLACA MICROCOMPUTADOR.(VISTA INFERIOR)



PLACA ANALOGICA (VISTA SUPERIOR)

Pressure Transducer, Printed Circuit Board Mountable

Model 180



Applications

The Model 1800 has been designed for the OEM user who needs an accurate pressure sensor and desires to use in-house signal conditioning for the greatest value added. These transducers should be employed in applications requiring a high degree of accuracy for the combined effects of linearity, hysteresis, repeatability and where long term stability is an important parameter. Specific applications are:

- **Medical (eye, ear, bladder, and blood pressure)**
- **Avionics (cabin pressure, altitude and air speed)**
- **Process control P to I (pressure to electric current) convertors when transmitting pressure data over long distances**

Other important applications are:

- **Liquid level determination**
- **Leak detection**
- **Automotive diagnostic testing**

Features/**Benefits**

Total static accuracy (linearity, hysterisis and repeatability) as low as ±0.1% Span maximum.
Affords highly reproducible measurements.

Designed for OEM applications.
Integrates easily into products or systems.

Sensor fabricated by high volume silicon technology.
Maximum economy.

Performance graded, each unit 100% computer tested.
Optimizes cost vs. performance for OEMs.

Solid state sensor, no moving parts.
Reliable — has a life expectancy greater than 10 million cycles. MBTF (mean time between failure) greater than 13 years.

Packaged in TO-8 (½" dia.) transistor housing.
Printed circuit board mountable, requires minimum space.

Available in four package options.
Provides for maximum design flexibility.

Wide selection of pressure ranges up to 100 PSI, gauge and absolute, English and metric.
Allows selection of range best suited to application.

Description

This linear integrated circuit transducer senses pressure by means of a silicon diaphragm into which a fully active Wheatstone bridge has been diffused. The piezo-resistive properties of silicon are utilized to produce a linear output signal proportional to the applied pressure. The high signal level of silicon strain gages can then be easily amplified or further conditioned as necessary to accommodate the user's need. The sensing chip is mounted in a standard TO-8 package which is compatible with printed circuit board mounting.

Transductor → 1800 – 03 – 01 – 00B – 0

Package Options and Dimensions

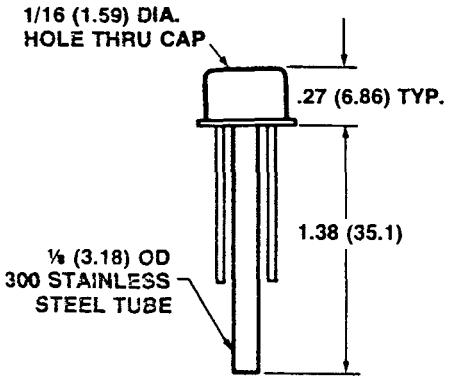


*FIGURE 1 (BOTTOM ENTRY)

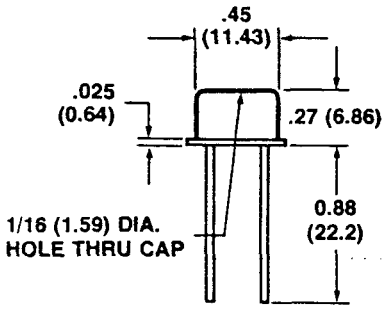


FIGURE 2 (TOP ENTRY)

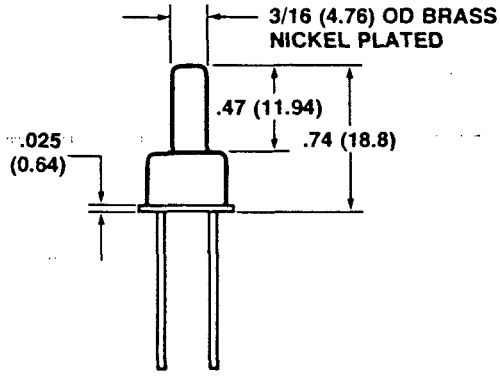


FIGURE 3 (TOP ENTRY)

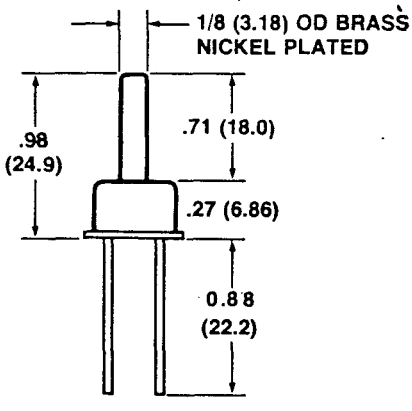


FIGURE 4 (TOP ENTRY)

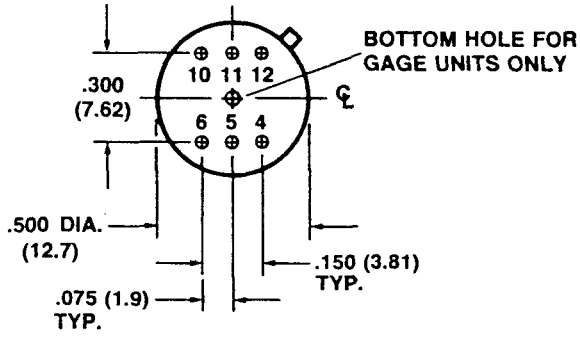
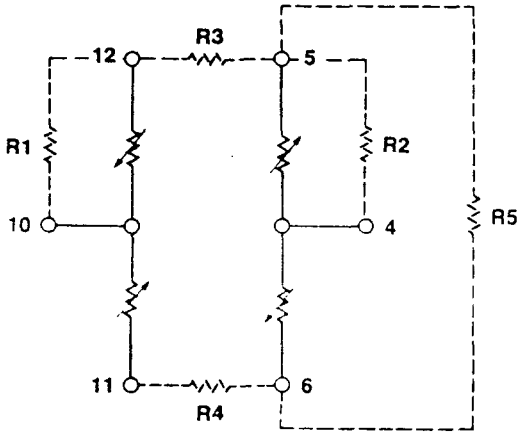


FIGURE 5 (BOTTOM VIEW)

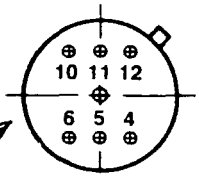
NOTE: (1) All dimensions are in inches. (2) All dimensions in parentheses are in millimeters. (3) All leads are 0.018" dia. — 7/8" long.
*Not available in pressure ranges greater than 50 PSIG (or equivalent), or in absolute versions.

Wiring Diagram



R1, R2, R3, R4, and R5 are (100 PPM) metal film resistors used for temperature compensation. Actual values are computer determined. Ohmic *values* are supplied with all pressure transducers allowing customers the flexibility to install their own resistors and reduce costs.

These resistors can also be supplied with production shipments at customers option. Please consult with your sales representative.



BOTTOM VIEW

TOP ENTRY
ELECTRICAL CONNECTION

PIN 5 + INPUT
PIN 6 – INPUT
PIN 4 + OUTPUT
PIN 10 – OUTPUT

BOTTOM ENTRY
ELECTRICAL CONNECTION

PIN 5 + INPUT
PIN 6 – INPUT
PIN 10 + OUTPUT
PIN 4 – OUTPUT

Specifications

General

Standard Pressure Ranges	0 to 5 PSIG	0 to 5 PSIA	0 to 0.5 kg/cm² gage	0 to 150 inches H_2O gage
	0 to 10 PSIG	0 to 10 PSIA	0 to 1.0 kg/cm² gage	0 to 250 inches H_2O gage
	0 to 15 PSIG	0 to 15 PSIA	0 to 3.0 kg/cm² gage	0 to 30 inches Hg gage
	0 to 25 PSIG	0 to 25 PSIA	0 to 7.0 kg/cm² gage	0 to 30 inches Hg abs
	0 to 50 PSIG	0 to 50 PSIA	0 to 0.5 kg/cm² abs	0 to 300 mm Hg gage
	0 to 100 PSIG	0 to 100 PSIA	0 to 1.0 kg/cm² abs	0 to 760 mm Hg gage
			0 to 3.0 kg/cm² abs	0 to 760 mm Hg abs
			0 to 7.0 kg/cm² abs	

To Order	See ordering information
Maximum Pressure	2X rated range
Media Compatibility	*Top Entry Models*: (Reference package options, Figures 2 and 3) Wetted sensor materials (i.e., parts that actually contact media) are the tube (nickel), housing (nickel), and the silicone gel. Restricted to gases and liquids that are non-corrosive. Figure 4 wetted materials are nickel, aluminum, gold, silicon dioxide and either silicone rubber or epoxy. *Bottom Entry Model*: (Figure 1) In this model, the media contacts the opposite side of the sensor circuit side. It therefore has greater media compatibility. Wetted materials are 300 series ss, solder (tin and lead), silicon dioxide and the silicone rubber RTV 116 or CF 3600 epoxy. (Note: Can only be ordered in ranges from 5 to 50 PSIG and equivalents.) If this pressure transducer is to be used in a media environment other than non-corrosive gases and liquids, please contact the factory.

Electrical
(at 77°F (25°C) unless otherwise stated)

Input Excitation	1.5 mA (constant current) fuente de corriente CC
Output Signal	See Performance Chart
Insulation Resistance	100 megohms at 50 V dc
Bridge Resistance (Input Impedance)	4700 ohms typical, 6000 ohms maximum
Electrical Connections	Printed circuit board mount TO-8; see wiring diagram

Mechanical

Weight	0.11 ounces (3.0 grams)
Pressure Connections	See Package Options
Case Material	Nickel
Dimensions	See Package Options and Dimensions

Environmental

Operating Temperature	−40°F (−40°C) to 250°F (121°C)
Compensated Temperature Range	30°F (−1°C) to 130°F (54°C)
Vibration	10 G's rms, 20 to 2000 Hz
Shock	100 G's, 11 milliseconds
Life	10 million cycles
Stability (typical)	±0.2% span/6 months

Performance *(maximum values unless otherwise stated)*

Class	*Static Accuracy % Span	Zero Pressure Output @ 77°F (25°C) mV	† Span Voltage @ 77°F (25°C) mV at rated excitation	Thermal Accuracy Expressed as % of Span Voltage Temp. Range 30°F (−1.1°C) to 130°F (54.4°C	
				Zero Pressure	Span
A	±0.1	±5	100 ± 25	±0.5	±0.5
B	±0.25	±5	75 to 150	±1.0	±1.0
C	±0.5	±10	50 min	±2.0	±2.0
D	±1.0 typical	±15 typical	50 typical	±5.0 typical	±5.0 typical

Note: Performance specifications are achieved using temperature compensation resistors.

*Static accuracy is a combination of linearity, hysterisis and repeatability @ 77°F (25°C) per ISA S51.1.

† Span is defined as the algebraic difference in output signal level between full scale rated pressure and zero pressure [illegible] rated input excitation power

Ordering Information

Model 1800

1800 [0][3] — [0][1] — [0][0][B] — [0]

Pressure Connections

NOTE 1:

00	Top Entry; Fig. 2 No Tube
01	Top Entry; Fig. 3 3/16" OD x 15/32" LG; Brass, Nickel Plated Tube
03	Bottom Entry; Fig. 1 1/8" OD x 1-3/8" LG; 300 Series Stainless Steel Tube
04	Top Entry; Fig. 4 1/8" OD x 23/32" LG; Brass, Nickel Plated Tube

Pressure Ranges

01	0 to 5 PSIG
02	0 to 10 PSIG
03	0 to 15 PSIG
06	0 to 25 PSIG
08	0 to 50 PSIG
* 09	0 to 100 PSIG
* 31	0 to 5 PSIA
* 32	0 to 10 PSIA
* 33	0 to 15 PSIA
* 34	0 to 25 PSIA
* 35	0 to 50 PSIA
* 36	0 to 100 PSIA
76	0 to 0.5 kg/cm² gage
77	0 to 1.0 kg/cm² gage
78	0 to 3.0 kg/cm² gage
* 79	0 to 7.0 kg/cm² gage
* 96	0 to 0.5 kg/cm² abs
* 97	0 to 1.0 kg/cm² abs
* 98	0 to 3.0 kg/cm² abs
* 99	0 to 7.0 kg/cm² abs
C1	0 to 150 inches H_2O gage
C0	0 to 250 inches H_2O gage
K0	0 to 30 inches Hg gage
* L0	0 to 30 inches Hg abs
P0	0 to 300 mm Hg gage
P1	0 to 760 mm Hg gage
* Q0	0 to 760 mm Hg abs

* Reference **NOTE 2.**

Electrical

Input

0	1.5mA

Output Span

0	see performance

Performance

NOTE 3

CLASS	LINEARITY, HYSTERESIS, AND REPEATABILITY % SPAN	ZERO PRESSURE OUTPUT @ 77°F (25°C) mV	SPAN VOLTAGE @ 77°F (25°C) @ RATED EXCITATION mV	THERMAL ACCURACY EXPRESSED AS % OF SPAN VOLT Temp Range 30°F (−1°C) to 130°F (54°C)	
				ZERO PRESSURE	SPAN
A	±0.10	±5	100 ± 25	±0.5	±0.5
B	±0.25	±5	75 to 150	±1.0	±1.0
C	±0.50	±10	50 minimum	±2.0	±2.0
D	±1.00 typical	±15 typical	50 typical	±5.0 typical	±5.0 typical

Electrical Connections

0	printed circuit board mountable

NOTE 1: Refer to data sheet package options and dimensions when making pressure connection selection.

NOTE 2: Pressure ranges indicated with asterisks are not available with the bottom entry *(03) ss tube.*

NOTE 3: Output voltage is determined by the Performance Grade class option. Choice of A, B, C, or D is entered directly with the sales order using the letter designation.

GENERAL NOTE: Regarding Media Compatibility for top entry and bottom entry models, refer to Media Compatibility Statement in data sheet.

Model 1800 Ordering Example
Assume an order for a bottom entry (Fig. 1), 25 PSIG range, class B pressure transducer.
Order entry would be: 1800-03-06-00B-0

Represented by:

[illegible] Jarrate S.L.
[illegible] Circuit [illegible]
Tel. [illegible] 08034 Barcelona

Sales Terms: Net 30 days, FOB San Jose, CA 95134. All standard models are stock items and can be shipped immediately. Prices and specifications are subject to change without notice.

Warranty: Foxboro/I.C.T., Inc. products are warranted against defects of materials and workmanship for 12 months from date of shipment.