UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIA Y TECNOLOGÍA

ESCUELA DE TECNOLOGÍA ELECTRONICA

DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE ENTRENAMIENTO BASADO EN ÉL MICROCONTROLADOR AT89C51

DISEÑO DEL PROYECTO DE TESIS DE GRADUACIÓN PREVIA LA OBTENCIÓN DEL TITULO DE TECNÓLOGO EN ELECTRÓNICA

Por: Francisco E. Alvarado C.

Director : Ing. Francisco E. Vásquez C.

Cuenca **Ecuador**

Agradecimientos:

Primero deseo agradecer a Dios por haberme permitido que llegue a este punto de mi vida.

A mis padres, por su constante apoyo durante los buenos y malos momentos ; y por su incondicionalidad hacia mí.

A mi esposa Antonieta y a mis hijos, Lis y Andy.

A todos los profesores que de una u otra manera me han brindado sus conocimientos, durante mi carrera y al final de la misma.

Al Ing. Francisco Vásquez, por el apoyo desinteresado para realizar este trabajo.

Al Ing. Eduardo Maldonado y al Ing. Jorge Morocho, por la ayuda que supieron dispensarme.

Francisco Eduardo

Dedicatoria:

Con todo cariño dedico esta tesis a mi Madre y Padre, que siempre han estado apoyándome en todas las grandes y pequeñas decisiones que he tomado durante mi vida.

Como también a mi Esposa, quien ha sido mi sostén fundamental para llegar a esta culminación .

De igual manera, a mis hijos Lisseth y Andrés, para que esto les sirva de incentivo y así puedan lograr todas las metas que ellos se propongan.

Francisco Eduardo

Responsabilidad:

Las ideas y opiniones vertidas, sin perder la óptica y finalidad del fabricante, en esta tesis son de exclusiva responsabilidad del autor. Además, cada práctica debe servir como repaso; y, no como un programa definitivo. Ya que cada tema puede ser resuelto de varias formas.

Francisco Eduardo Alvarado Correa

INDICE DE MATERIA

INDICE DE FIGURAS

INDICE DE TABLAS

Alvarado Correa Francisco Eduardo
Trabajo de Graduación
Ing., Vásquez Francisco
2005 / 11 / 20

DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA DE ENTRENAMIENTO BASADO EN EL MICROCONTROLADOR AT89C51

MICROPROCESADORES Y MICROCONTROLADORES

EL Microprocesador

Básicamente un microprocesador es un chip con integración LSI, que tiene una *Unidad de Procesamiento Central*; en tal virtud es el encargado de generar las varias señales de control para un sistema básico de microcomputador. Buscar y trasladar datos, instrucciones de los diferentes circuitos integrados de su sistema; como son:

- Memorias externas ROM y / o RAM,
- Decodificadores,
- Manejadores de periféricos,

También, efectuar operaciones matemáticas y lógicas de acuerdo a las instrucciones que lo gobierna. Estas instrucciones pueden ser cambiadas para modificar secuencias de operación, alternar los varios elementos lógicos encontrados en él. Dado que permite reemplazar a una cantidad de dispositivos lógicos, tales como: flip-flop´s, compuertas lógicas, contadores, lactches, representa una ventaja; desde el punto de vista:

- De la facilidad de diseño,
- Versatilidad,
- Mantenimiento,
- Costo,
- Consumo energético, etc.

Entonces, para implementar un sistema con este dispositivo hay que estar en conocimiento de:

1. El software de programación que emplea para su programación, que puede ser assembler u otro de mas alto nivel.

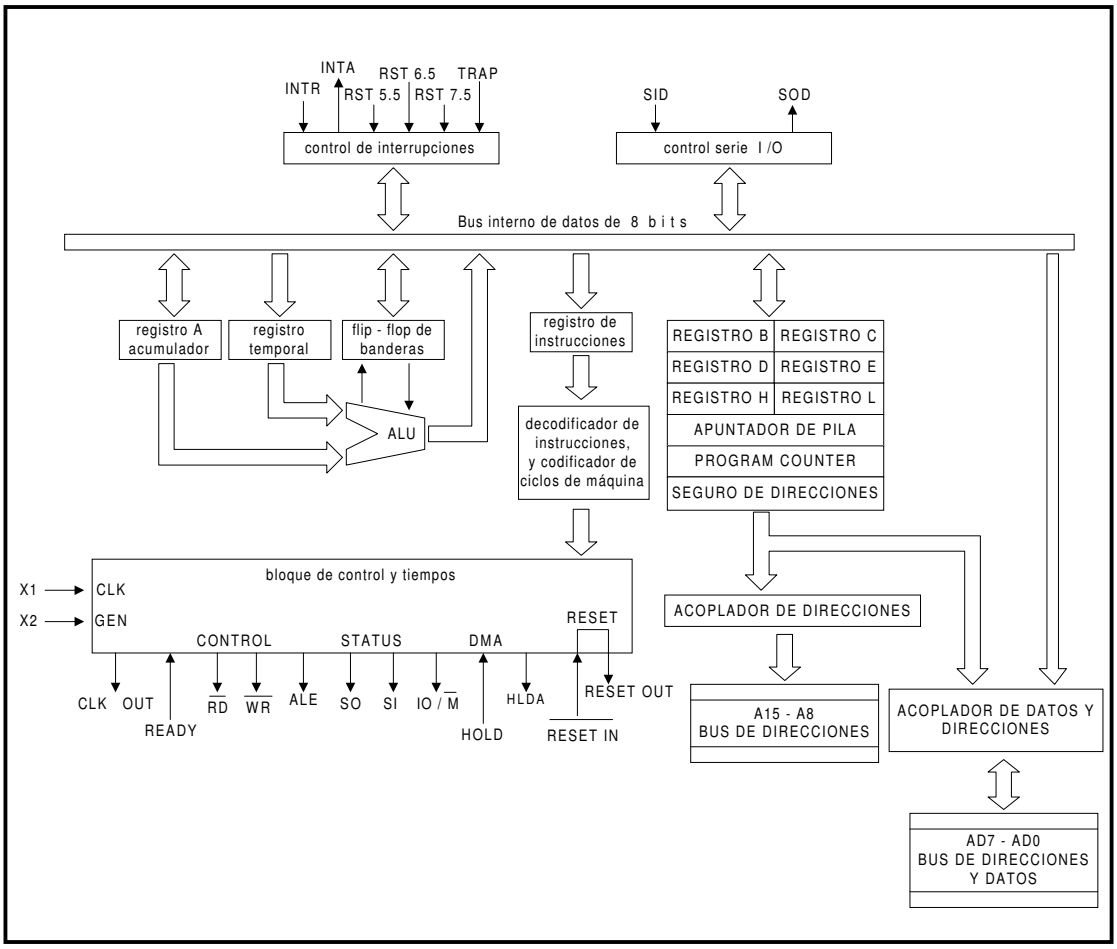
2. La forma de intercomunicación con los otros elementos de apoyo.

3. Lo relacionado a la numeración binaria, y sus equivalencias decimal y hexadecimal. Cekit, 1992.

Para explicar brevemente como funciona el microprocesador, nos referiremos al 8085 de la INTEL:

: **El 8085** es un microprocesador de 8 bits y es una versión mejorada del 8080, con el cual es totalmente compatible. Su constitución interna es:

Figura 1: Microprocesador 8085



Como se puede ver tiene lo siguiente:

Bloque de control y tiempos.- Es el encargado de establecer el control y los tiempos requeridos para ejecutar las instrucciones de la memoria.

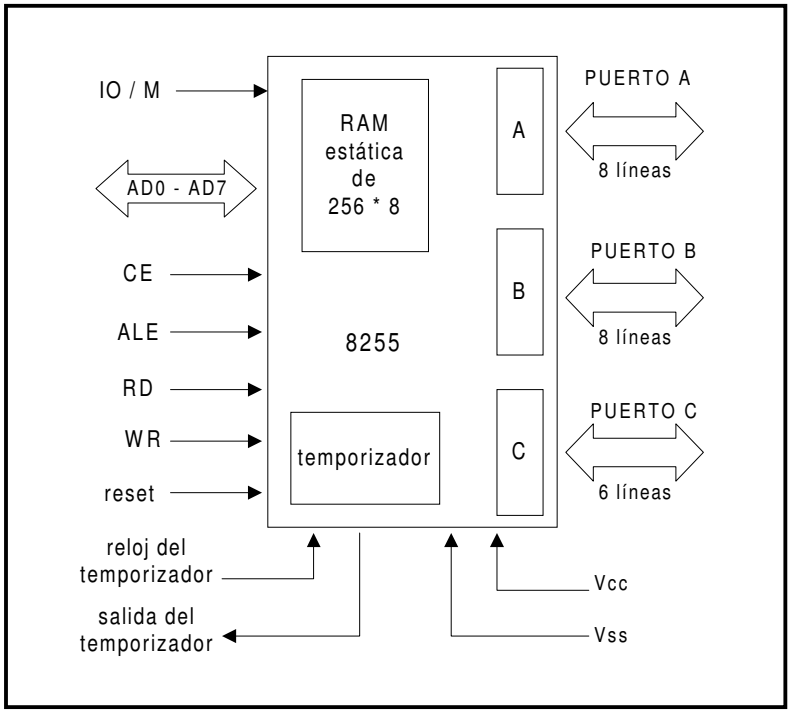
Bus de direcciones.- Contiene el byte más significativo del conjunto de bits de la dirección, es decir va desde A15 a A8.

Bus de direcciones y datos.- Este bus puede multiplexarse para manejar datos y direcciones. Produciendo el byte menos significativo del direccionamiento; y transmitir o recibir datos de los periféricos.

Pero por si solo no puede efectuar ningún trabajo, por no tener una memoria que almacene las instrucciones; debiéndose apoyar de una circuitería externa. En ésta se colocan memorias, buffer´s de salida, manejadores de puertos, etc. El más común que se utiliza es el soporte 8155.

Este chip contiene una RAM interna de 256 bytes, 3 puertos que pueden programarse como input / output en paralelo, un temporizador, y, un bus multiplexado de datos / direcciones:

Figura 2: PIA 8255



En la RAM interna se puede almacenar un programa pequeño; que puede servir para aplicaciones demostrativas, y aquellas que no requieran de mucha complejidad.

"Por software se puede a los puertos A – B – C configurar como ingresos o egresos de datos; las líneas del puerto C se las toma también como señales de petición de interrupciones, o, como preparación / conformidad para la transmisión de datos de los puertos A y B. Entre tanto el temporizador programable es básicamente un contador direccionable de 14 bits". *Cekit, 1992*

La unión de estos dos integrados da paso a un microcomputador o microcontrolador, ya que no puede funcionar ninguno de los dos por separado.

El Microcontrolador

En el año de 1976 ya se tenía una alta densidad de integración de componentes electrónicos en los chips, situación que facilitó la implementación de sistemas de control / comando; dándose paso a un dispositivo llamado microcomputador monopastilla.

Este microcomputador monopastilla integra interiormente, en una sola pastilla, los siguientes subsistemas:

- CPU,
- RAM,
- ROM o EPROM,

- Entradas / salidas,
- Contadores y temporizadores,
- Conversores ADC / DAC,
- Gestión de interrupciones,
- Wacth Dog Timer,
- Latches para los periféricos,
- Registros más dúctiles,
- Circuitos de reloj, etc.

"De una forma muy simple se puede decir que cuando los sistemas basados en los microprocesadores se especializan en aplicaciones industriales, en ambientes eléctricos adversos, aparece la versión industrial del microcomputador monopastilla; que vendría a ser el microcontrolador". *González, 1992*.

Puede ser de 8, 16, 32 bits la cantidad que puede manejar la CPU; permite por un procesador booleano tratar bit a bit; los puertos, que son bidireccionales, pueden manejar directamente a displays, transistores, optoacopladores que a su vez manejan a las cargas más grandes.

La RAM interna es capaz de tener 128 o 256 bytes y se la puede ampliar mediante una memoria externa; además permite una comunicación standard full dúplex con algún otro microcontrolador y/o computador; a las interrupciones se las prioriza para su ejecución.

Además de todo esto, los microcontroladores de 8 bits presentan ciertas particularidades como:

> La CPU es de 8 bits.

> Maneja un procesador booleano, tratamiento bit a bit.

> Comunicación full dúplex.

> Fuentes de interrupciones priorizadas.

> Espacio de memoria externa de 64 Kb, tanto para datos como programas.

> Los 4 puertos son de 8 bits.

> 6 modos de interrupción con prioridad.

> 2 interrupciones externas.

> Oscilador interno.

> Frecuencia de reloj de hasta 30 MHz.

> Alta inmunidad al ruido eléctrico.

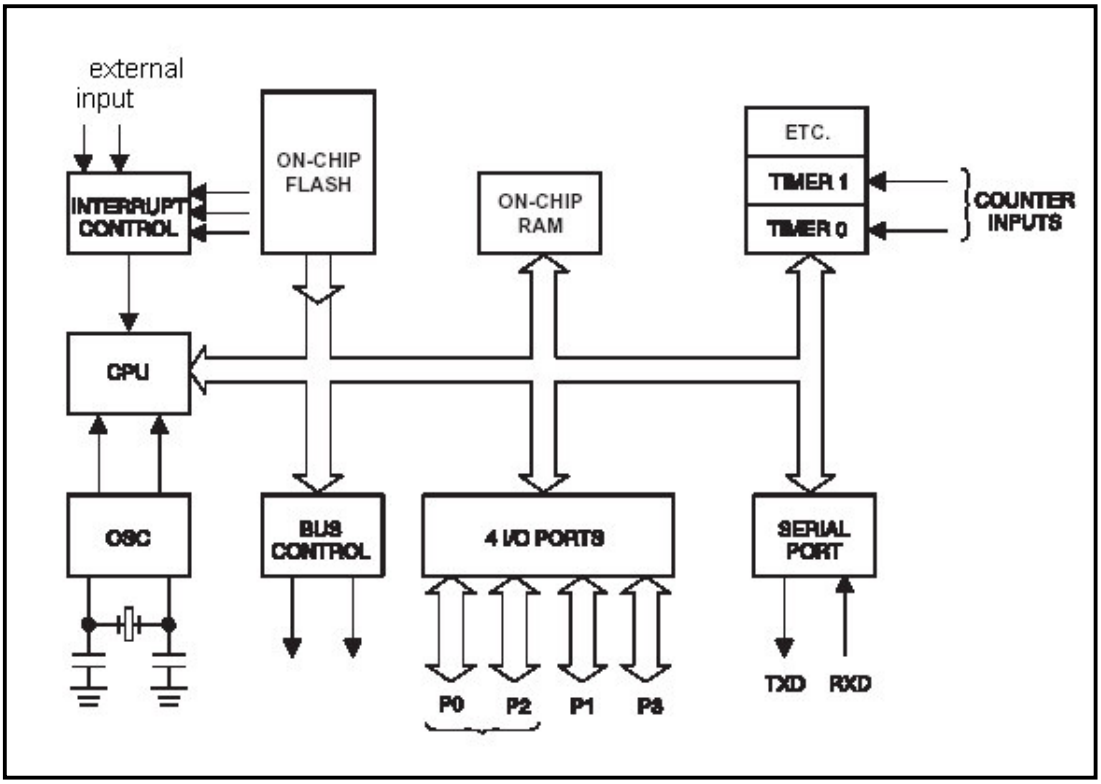
> Protección de memoria por encriptación.

> Tiene versiones con Watch Timer Dog, que vigila el desempeño óptimo de la CPU.

> Posibilidad de insertar funciones bajo diseño específico ASIC.

El diagrama siguiente muestra la configuración genérica de un microcontrolador 89XXXX:

Figura 3: Estructura interna del AT89XXXX



Este diagrama genérico muestra que, según su estructura, tiene:

~ Un tipo de memoria que permite almacenar un programa definido, constituye la memoria del programa, la cual es del tipo flash EPROM.

~ Maneja una RAM de 128 / 256 bytes, según el microcontrolador, que es el almacenamiento temporal para las instrucciones del programa.

~ Según el tipo de microcontrolador, tiene 2 o 3 timer / counter. Los cuales sirven para temporizar acciones, o, realizar secuencias de conteo.

~ Dos pines de control para la comunicación del puerto serial del microcontrolador con otro elemento, son RXD y TXD.

~ Una CPU que es la encargada de controlar que el microcontrolador se maneje con seguridad al ejecutar una programación. La conforman:

- Una Unidad Aritmético Lógica.
- Un banco de registros.
- Una lógica de control de Bus, interno y externo, de datos y direcciones.
- Una lógica de control de las interrupciones, llevada a cabo por prioridades.
- Una lógica de control del chip.

~ Posee 4 puertos, P0 – P1 – P2 – P3, que pueden funcionar en forma bidireccional dependiendo de la tarea dada por software. Cada uno tiene un latch y un sistema de buffer que les permite un fan –out apreciable; pudiendo manejar directamente a displays, entradas TTL´s, a transistores, leds, optoacopladores en todas sus variedades.

~ Un sistema de amplificador inversor que puede ser configurado como un oscilador. Empleando indistintamente un cristal de cuarzo o una red RC; permite también que al microcontrolador se lo controle por una señal de reloj externa.

Para programarlo se cuenta con un software de directivas assembler de fácil entendimiento; o si se quiere también se lo puede hacer por lenguaje C, por Basic, o por el BASCOM. Para directivas assembler se tiene el MCS – 51 que es compatible con la gran mayoría de los microcontroladores de la familia Atmel, y, el Sides. Pudiendo funcionar él solo en una aplicación.

1 Hardware de las Series AT89.

1.1 Introducción

Dentro de la familia de microcontroladores AT89 se encuentran los AT89X51 y AT89X52, como los principales; los mismos que tienen sus variantes.

La siguiente tabla describe a los más populares:

Tabla 1: Microcontrolador flash de Atmel

Equipo	Memoria de programa	Bytes memoria datos	Timers 16-bits	Tecnología
AT89C1051	1K Flash	64 RAM	1	CMOS
AT89C2051	2K Flash	128 RAM	2	CMOS
AT89C51	4K Flash	128 RAM	2	CMOS
AT89C52	8K Flash	256 RAM	3	CMOS
AT89C55	20K Flash	128 RAM	3	CMOS
AT89S8252	8K Flash	256 RAM 2K EEPROM	3	CMOS
AT89C53	12K Flash	256 RAM	3	CMOS

Siendo las características más comunes:

- CPU de 8 bits optimizada para aplicaciones de control.
- Capacidad de procesamiento extensivo booleano.
- Memoria de programa on – chip flash.
- RAM on – chip para datos.
- Líneas de entrada / salida direccionables individualmente y bidireccionables.
- Timers / counters de 16 bits.

- Máximo dúplex del tipo UART.
- Múltiples estructuras de interrupción: fuente / vector / prioridad.
- Oscilador de reloj interno.
- EEPROM interna, dada en las series AT89S.
- Interface de bus serial SPI, dada en las series AT89S.
- Watch timer dog, dada en las series AT89S.

1.2 Descripción General

Para entender como se estructura internamente estos microcontroladores, se visualizará en diagramas de bloques los siguientes:

Figura 4: Diagrama central de bloques del AT89CXX

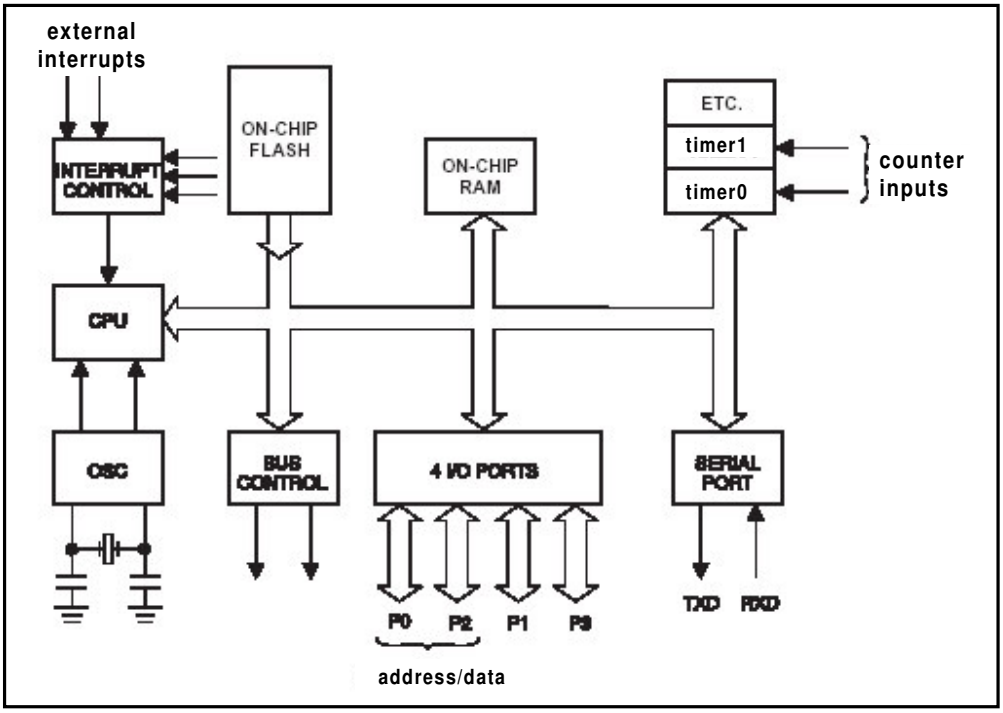
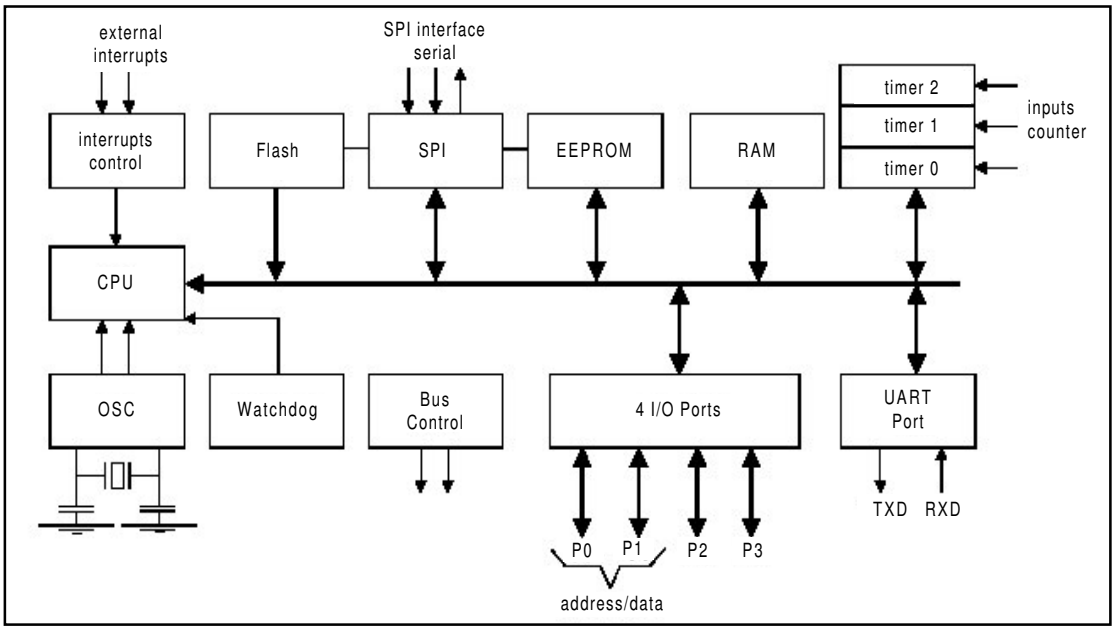


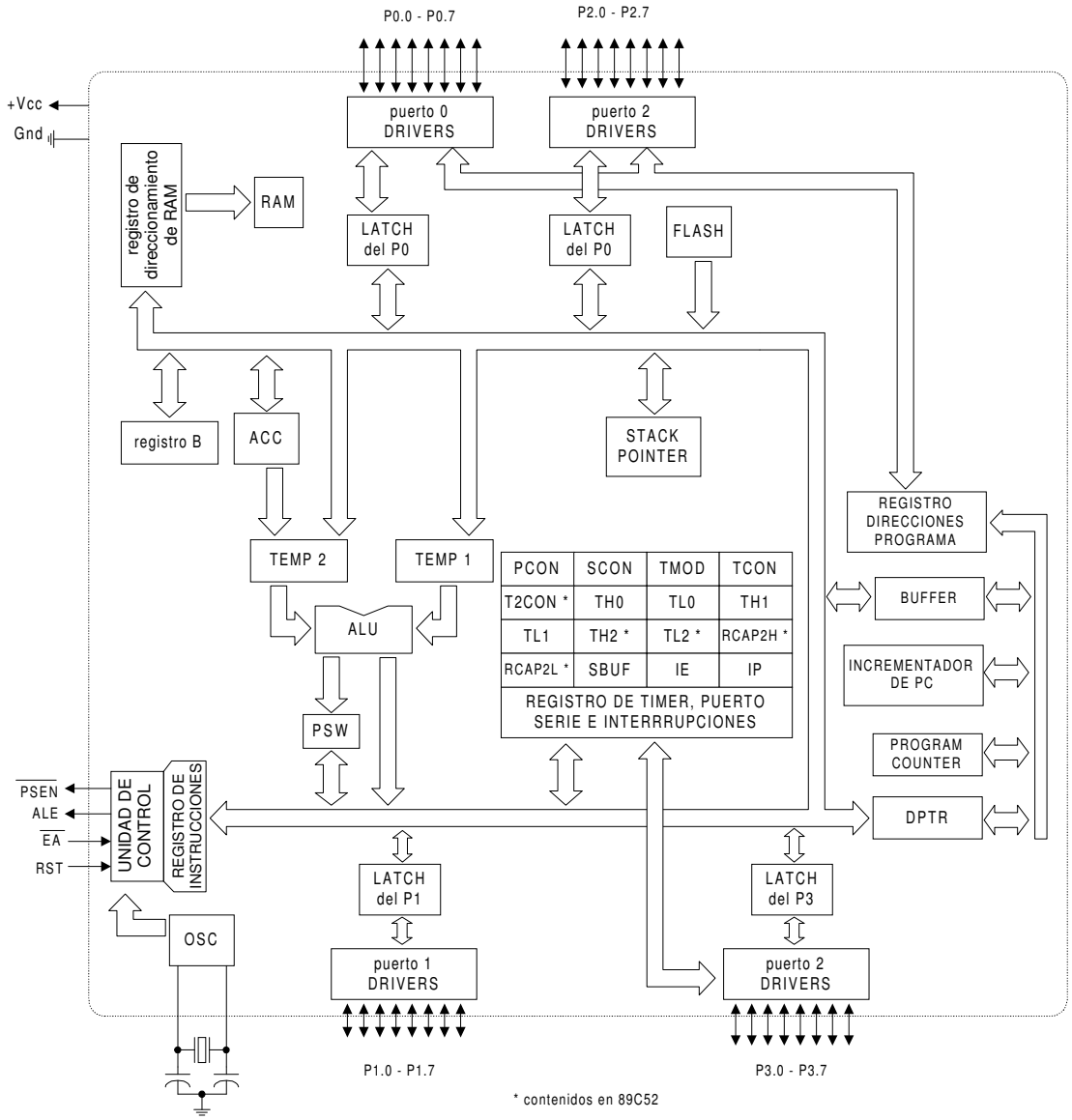
Figura 5: Diagrama central de bloques del AT89SXX



Mientras que la estructura del AT89C es simple, se puede visualizar que la serie AT89S es más compleja, ya que incorpora un puerto UART para la interface serial, un Watch Timer Dog para monitorear el funcionamiento de la CPU, una EEPROM interna.

Ahora, al detallar los bloques anteriores se tiene:

Figura 6: Distribución interna de un AT89C51



Vcc : Es la alimentación positiva de +5Vcd para el microcontrolador.

Gnd: Comprende el terminal negativo.

/PSEN: (*program store enable*); habilita la lectura de la memoria externa. La memoria externa tiene dos modalidades, de *datos* y de *programa*. Para diferenciarlas se usa ésta. No se activa cuando se esta ejecutando el contenido de la memoria interna del microcontrolador.

$\overline{\text{EA}}$/Vpp: Permite el acceso externo. Al mantenerse en un nivel alto solo se ejecuta el programa de la memoria interna; a menos que el program counter supere de 1FFF (8K) para el 89C52, y 0FFF (4K) para el 89C51.

Pero si se mantiene en nivel bajo se ejecuta el programa de la memoria externa, sin considerar la dirección de programa. Esto es:

EA = 1 ------- actúa como microcontrolador.

EA = 0 ------- actúa como microprocesador.

El mismo pin sirve para dar la señal de Vpp para programación.

/ALE-PROG: ALE (*address latch enable*) es un pulso que emite el microcontrolador que sirve para asegurar el byte bajo del bus de direcciones en el acceso de la memoria externa. ALE se emite a un rango de 1 / 6 de la frecuencia de reloj. PROG es el ingreso de los pulsos de programación de la memoria interna.

RXD: Según los modos, es la entrada del puerto serie.

TXD: Según los modos, es la salida del puerto serial.

/INT0: Entrada de la interrupción externa 0.

/INT1: Entrada de la interrupción externa 1.

T0: Entrada externa del temporizador / contador 0 (timer 0).

T1: Entrada externa del temporizador / contador 1 (timer 1).

/WR: ----- Señal de escritura para dispositivos externos.

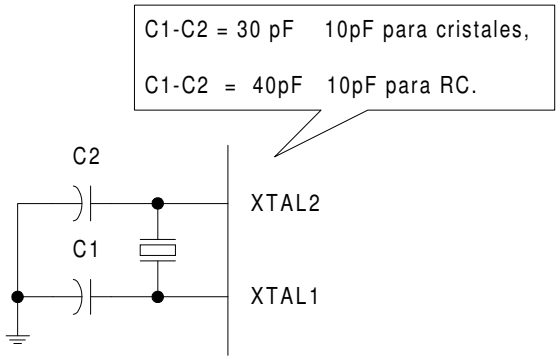
/RD: ---- Señal de lectura para dispositivos externos

RST: Inicializa el sistema. Se logra poniendo a nivel alto por un tiempo.

Las señales del clock son: XTAL1 de entrada, y XTAL2 de salida; de un amplificador inversor que puede ser configurado como un chip oscilador. Este clock puede montarse por un cristal de cuarzo, o por una red RC.

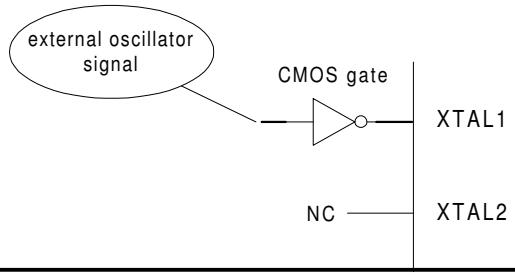
Siendo ésta:

Figura 7: Red RC



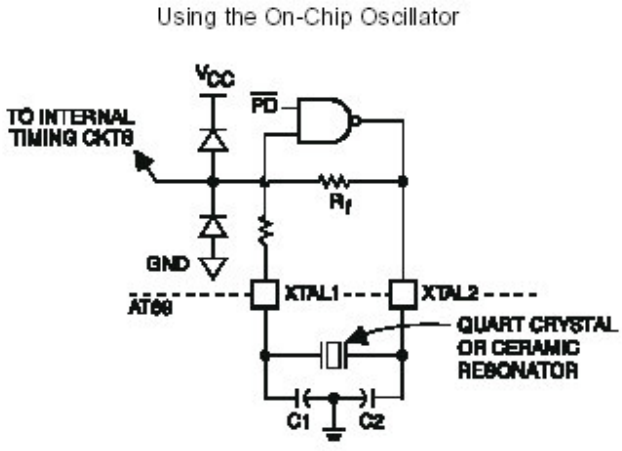
Entre tanto para que funcione el microcontrolador con una señal de clock externa es:

Figura 8: Clock externo



La circuitería para utilizar un oscilador interno es:

Figura 9: Oscilador interno

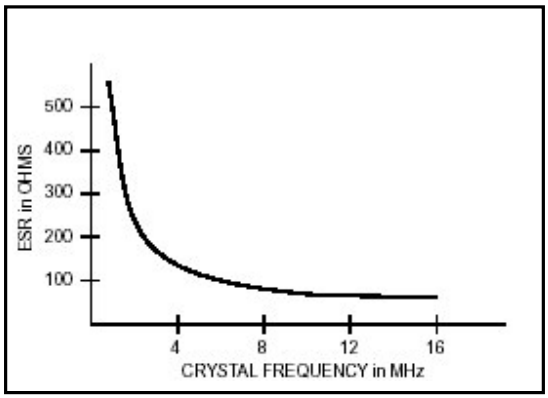


Las especificaciones del cristal son:

- ESR (Equivalent Series Resistanse: *resistencia serie equivalente*):

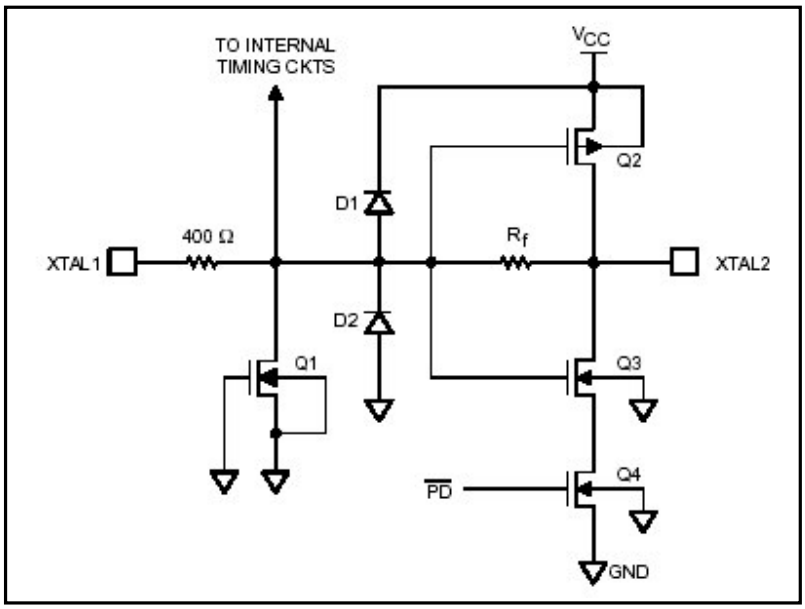
- CO (Shunt Capacitance: *capacitancia en paralelo*) : con un valor de 7 pF máximo.
- CL (Load Capacitance: *capacitancia de carga*): siendo de 30 pF + 3 pF.
- Driver level: 1mW.

Figura 10: Frecuencia vs ESR



Entre tanto la circuitería del oscilador del AT89C51 se manifiesta así:

Figura 11: Circuito Oscilador



EL tipo de oscilador difiere con la versión NMOS del circuito.

⊇ *Acumulador*.- El ACC es un registro. En los mnemónicos de las instrucciones para él se refieren solo como A.

⊇ *Registro B*.- Este registro es usado en las operaciones de producto y cociente. Pero en otras instrucciones puede emplearse como registro de cojín de marca.

⊇ *Stack Pointer*.- Es un registro amplio de 8 bits. Es incrementado antes que el dato sea almacenado durante las ejecuciones PUSH y CALL.

- *Mientras tanto en el STACK podría residir cualquier sitio de la RAM interna; el STACK POINTER se inicializa a 07H después del RESET. Esto hace que el stack comience en 08H.*

⊇ *Data Pointer*.- El data pointer (DPTR) consiste de un byte alto (DPH), y un byte bajo (DPL). Su función es retener una dirección de 16 bits. También se lo puede tratar como dos registros independientes de 8 bits cada uno; o como uno solo de 16 bits.

⊇ *Serial Data Buffer*.- El buffer de datos serial es actualmente dos registros separados, un buffer transmisor y uno receptor.

- Cuando un dato es movido al SBUF, este va para el transmisor en donde es retenido para transmisión serial, →(al mover un byte al SBUF se

inicializa la transmisión)← . Cuando el dato es movido desde el SBUF, este viene desde el buffer receptor.

⊇ *Special Function Register*.-

Tabla 2: Mapa de los SFR, lo que está en ( ) existe en el 89C52

	8 bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)	(T2MOD)	(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

De acuerdo a esto, no todo los registros están ocupados (*los direcciones desocupadas no son implementadas en el chip*). El acceder a la lectura de éstas direcciones suele regresar datos aleatorios; la escritura no tiene efecto. El usar el software no escribiría unos en las locaciones no implementadas, ya que estas serán usadas para nuevas características en aplicaciones futuras.

⊇ *Timers Registers*.- Estos son registros pares: TH0 – TL0 ; TH1 – TL1 ; TH2 – TL2 ; son registros de 16 bits para contadores / temporizadores.

⊇ *Registros de Captura*.- El par de registros (RCAP2H, RCAP2L) son los registros de captura para el Timer 2 en modo "capture". En este modo; en respuesta de una transmisión por el pin T2EX del 89C52; TH2 y TL2 son copiados al interior de RCAP2H y RCAP2L.

El Timer 2 también tiene un modo de autorrecarga de 16 bits, por lo que RCAP2H y RCAP2L retienen estos valores para este modo.

⊇ *Registros de control*.- Son los empleados en funciones especiales (SFR), y son: IP – IE – TMOD – TCON – T2CON – T2MOD – SCON y PCON; contienen los bits de control y estado para la interrupción del sistema, los timers / counters y el puerto serial.

1.3 Estructura y operación de los puertos

Todos los puertos del AT89C51 y AT89C52 son bidireccionales, en donde cada uno posee un latch, →(*SFR del P0 hasta el P3*)← , un driver de salida y un buffer de ingreso.

Los drivers de salida de Puerto 0 y Puerto 2, más el buffer de ingreso del Puerto 0 son empleados en el acceso a la memoria externa. Para ésta aplicación, del Puerto 0 se obtiene

el byte bajo de direcciones de la memoria externa, que es multiplexado en el tiempo para ser escrito o leído.

Por el Puerto 2 se obtiene el byte alto de direcciones de la memoria externa, que es de una amplitud de 16 bits; y también en los accesos de memoria de 16 bits. En los accesos de memoria externa con direccionamiento de 8 bits, los pines del Puerto 2 emiten el contenido del registro P2 del SFR.

Los pines del Puerto 3 y dos pines del Puerto 1 (*en el AT89C52*) son polivalentes. A más de ser pines de un Puerto, son a la vez los que proveen características especiales dadas a continuación:

Tabla 3: Función de pines

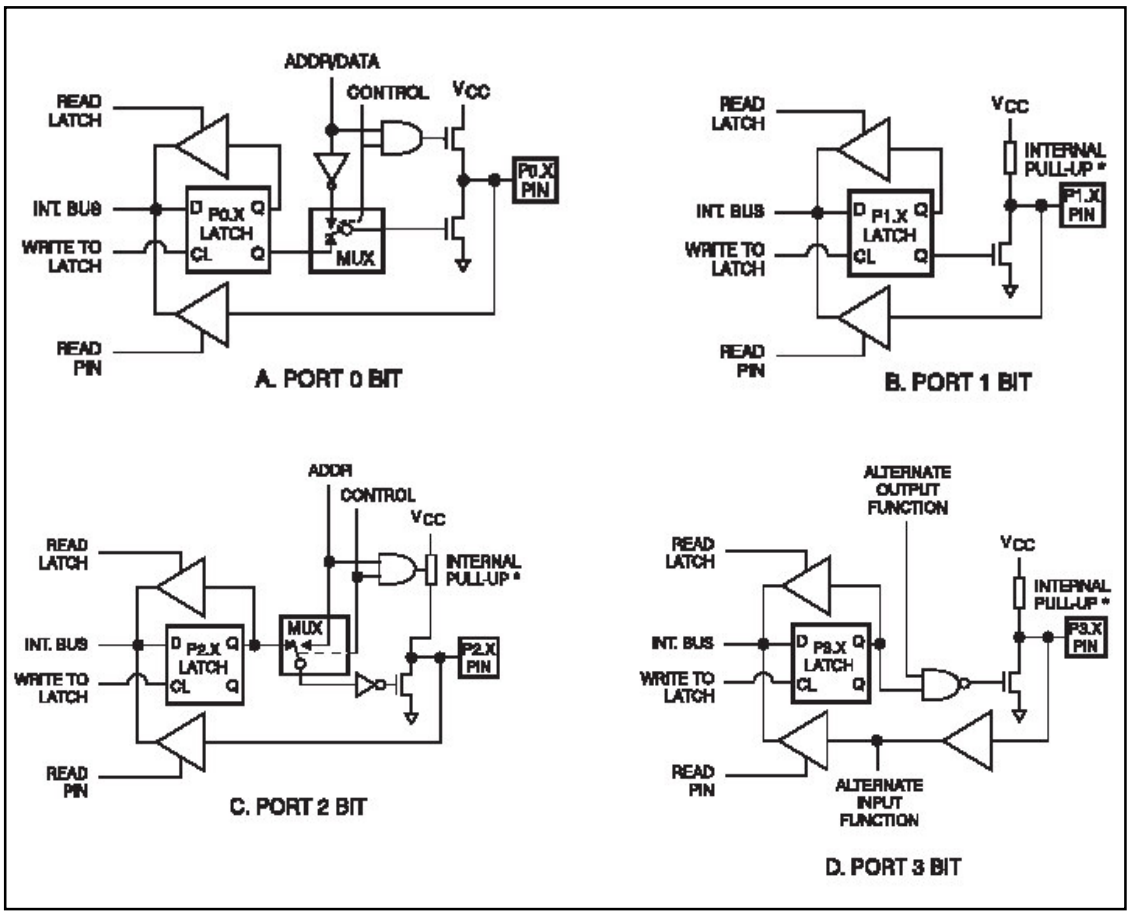
Pines	Función alternativa
P1.0(1)	T2 (entrada externa al timer/counter 2)
P1.1 (1)	T2EX (disparador externa al timer/counter 2 en captura/recarga)
P3.0	RXD (entrada puerto serial)
P3.1	TXD (salida puerto serial)
P3.2	INT0 (interrupción externa 0)
P3.3	INT1 (interrupción externa 1)
P3.4	T0 (entrada externa al timer/counter 0)
P3.5	T1 (entrada externa al timer/counter 1)
P3.6	WR (strobe para escritura en memoria externa de datos)
P3.7	RD (strobe para lectura en memoria externa de datos)

(1) solo son disponibles en el AT89C52

1.3.1 Configuración INPUT / OUTPUT de los Puertos

Para entender como configurarlos se verá la estructura de cada uno:

Figura 12: Búfferes de IN /OUT, y los latches para los bits de puerto



Cada uno de los pines en el Puerto correspondiente contiene un latch para el bit y un buffer de I / O. El latch, para el bit de los SFR de los Puertos, es presentado por un flip – flop tipo "D"; con una entrada del bus interno y reloj, responde a una señal de *"escribir en el latch"* que viene desde la CPU.

La salida del flip – flop, "Q", es colocada sobre el bus interno en respuesta a una señal de *"leer el latch"* proveniente desde la CPU. El nivel del pin de por sí es colocado en el bus interno en respuesta de la señal *"leer pin"* que viene desde la CPU. Algunas instrucciones que leen el puerto, activan la señal *"leer el latch"*; y otras la señal *"leer pin"*.

De igual manera, los drivers de salida de los Puertos 0 y 2 pueden ser conectados a un bus interno de direcciones y direcciones / dato; ya que se tiene una señal de control interna para usarla al acceder a la memoria externa, los pines del Puerto 2 sobrantes no cambian, pero "1" son escritos en el SFR del Puerto 0.

Si el bit en el latch del Puerto3 contiene un "1", entonces el nivel de salida es controlado por la señal "*alternative output function"*, tal como lo indica el diagrama correspondiente.

- *El nivel de los pines del Puerto 3 son siempre disponibles en "alternative output function"* .

Los Puertos 1 – 2 – 3 tienen pullups internas; entre tanto el Puerto 0 tiene salidas a drain abierto. Pero cada pin de los puertos mencionados pueden emplearse como salida o entrada independiente; los Puertos 0 y 2 no podrían ser usadas para propósito general cuando están funcionando como bus de address / data.

- *Para ser usado como una entrada, el bit del latch del Puerto contendrá un "1", que apagará al driver de salida del FET. Enton ces los pines de los Puertos 1 – 2 – 3, es jalado a un "alto" por la pullup interna, pero puede ser jalado a un "bajo" por una fuente externa.*

Ahora, el Puerto 0 no tiene pullups internas, el FET pullup en el driver de salida es usado solamente cuando el Puerto emite "unos", lo que se da al acceder a la memoria externa. De otra manera, el FET se halla apagado. Consecuentemente las líneas del Puerto 0 que están siendo usadas, son salidas a drain abierto.

Al escribir un "1" en el bit del latch del Puerto permite que ambos FET se apaguen, adquiriendo el pin un estado flotante; por ésta situación puede ser usarse como entrada de alta impedancia. Al momento de producir un RESET, se escribe automáticamente "unos" en las salidas de los Puertos.

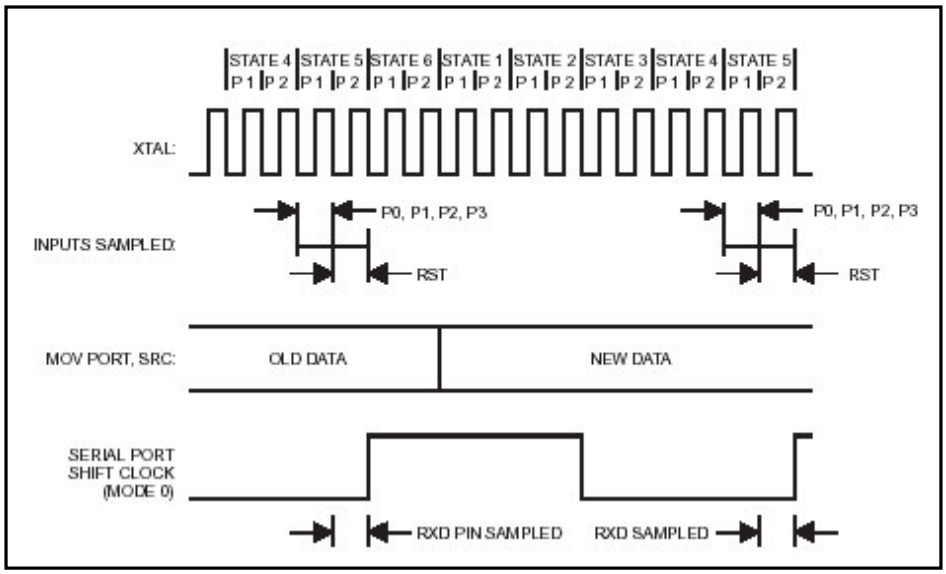
1.3.2 Escritura en los Puertos

Al cambiar el valor del latch del Puerto por una instrucción, éste nuevo valor llega al latch durante el S6P2 del ciclo final de la instrucción. No obstante, los latches de los Puertos son muestreados por sus buffers de salida, solamente durante la fase 1 de cualquier período de reloj.

- *Pero durante la fase 2, el buffer de salida mantiene el valor muestreado previamente en la fase1.*

Por lo que el nuevo valor en el latch del Puerto no aparece actualmente en el pin de salida hasta la siguiente fase1, que vendría a ser S1P1 del siguiente ciclo de máquina:

Figura 13: Secuencia de operación del puerto

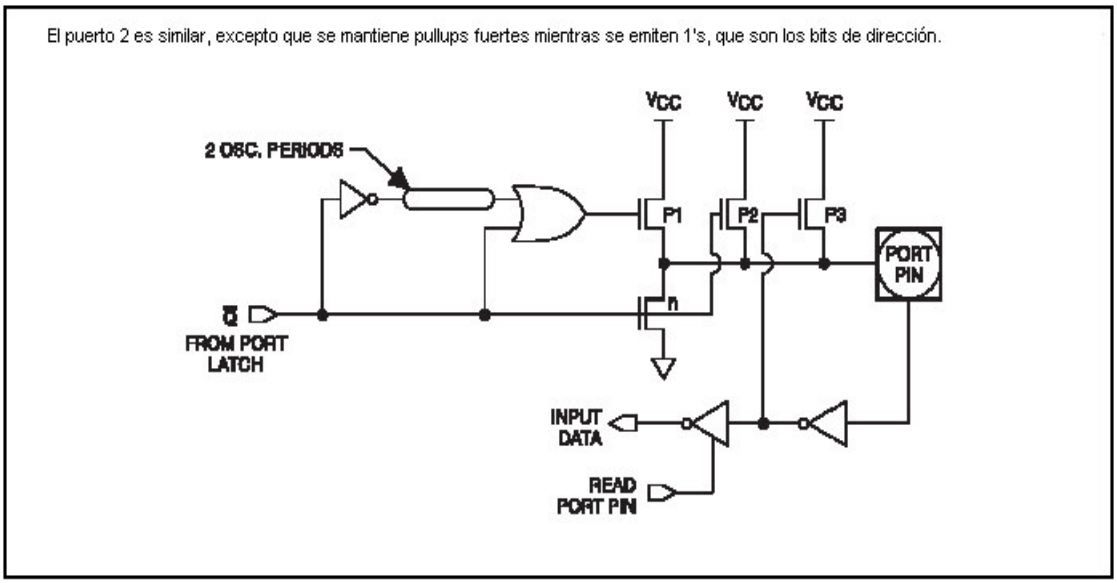


Bien, si se requiere una transición de cambio de "0" a "1", en el Puerto 1 – 2 o 3, un pullup adicional es encendido durante S1P1 y S1P2 del ciclo en que la transición ocurre; ésto incrementa la velocidad de la transición.

El pullup externo puede ser fuente de cerca 100 veces la corriente que puede manejar la pullup interna:

- *Esta pullup externa es un transistor de efecto de campo "FET" , y no un resistor lineal.*

Figura 14: Configuración de la pull-up´s internas de los P1 y P3



La figura muestra que el pullup no es más que 3 FET:

- Un n-FET enciende cuando un "1" lógico se aplica al gate, y se apaga cuando un "0" lógico va al gate. Un p -FET lo opuesto, es decir, con un "0" enc iende y con un "1" apaga.

  El transistor p-FET1 es activado por dos períodos del oscilador después de un transición de 0 a 1 en el latch del Puerto. Mientras que p-FET1 es activado, éste activa también al p-FET3 por medio del inversor. Este inversor más el p-FET3 forman un latch que mantienen el "1".

  Si el pin emite un "1", cualquier pulso negativo de una fuente externa puede apagar al p-FET3, produciendo que caiga en un estado flotante. El p-FET2 es un

pullup débil que es activado siempre y cuando el n-FET este apagado; todo esto es tradicional en el estilo CMOS. Pero p-FET2 es cerca de 1/10 de la potencia del p-FET3.

Su función es realmacenar un "1" para el pin, en la eventualidad de que el pin perdiera su "1" en un glitch.

Ahora bien, los buffers de salida de los Puertos 1 – 2 y 3 pueden manejar cada uno 4 entradas TTL´s. Como los pines son CMOS, pueden ser manejados por salidas de colector abierto y drain abierto, con el inconveniente que las transiciones se vuelven lentas. Esto es porque al ingresar un "0" el pullup p -FET3 se apaga, permitiendo que solamente el p-FET2 , pullup débil, maneje la transición.

1.4 Modos para operar a bajo consumo

Estos dos modos se logran seteando el bit correspondiente en el registro PCON, localizado en el SFR, que es:

Tabla 4: PCON (Registro de Control de Poder), no es direccionable por bit.

SMOD	----	----	----	GF1	GF0	PD	IDL

SMOD	bit doblador de rango de baudios. Si usa el Timer 1 para generar baudios y SMOD=1; el rango de baudios es doblado cuando el puerto serial esta en modo 1,2, o 3.
----	no implementado.
----	no implementado.
GF1	bit flag de propósito general
GF0	bit flag de propósito general
PD	bit de Power Down. Seteando este bit se activa el modo Power Down.
IDL	bit de modo IDLE. Setando este bit, se activa el modo de operación. Si se setean PD y IDL al mismo tiempo, PD tiene preferencia.

No se debe tratar de escribir en los espacios no implementados.
Al resetear todos los bits van a 0.

1.4.1 Modo IDLE

La instrucción que setea el PCON.0 es la última en ejecutarse antes de comenzar el modo; la señal del clock es atenuada por la CPU, en la puerta lógica, pero no las funciones de interrupción, de Timer y del Puerto Serial.

El estado de la CPU es completamente guardado; el Stack Pointer, el Program Counter, el Program Status Word, el Acumulador y todos los otros registros mantienen sus datos también. Los pines quedan atrapados en los niveles lógicos que tenían antes de activar éste modo, mientras que ALE y PSEN mantienen un nivel lógico "alto". Se lo puede dar por terminado así:

- Por la activación de cualquier interrupción habilitada, la que causará que PCON.0 sea limpiado por hardware. La instrucción será atendida, y, la siguiente

instrucción será RETI que es la anterior a la que puso el microcontrolador en el modo idle.

Ahora, los bits de los flags GF0 y GF1 pueden ser empleados para indicar si ha ocurrido una interrupción durante un funcionamiento normal, o, durante el modo idle. Es decir, la instrucción que activo el modo idle puede también activar a uno o a ambos flags. Al haber finalizado el modo por una interrupción, la rutina de la interrupción puede examinar a la vez los flags.

- La otra manera de dar por terminado el modo es con un reset del hardware. Después de que el reloj esté operativo, la señal en el reset permanecerá en "alto" por lo menos dos ciclos de máquina (*24 períodos oscilatorios*) para completar el reset.

La señal del reset limpia el bit idle en forma directa y asíncrona. A la vez la CPU reasume la ejecución del programa desde el último sitio en que fue interrumpida. Dado que el reset tarda de 2 a 3 ciclos establecerse, se tiende a inhibir la escritura en la RAM pero no el acceso a los puertos; por ésto, la siguiente instrucción a la que invoca al modo idle no deberá ser escritura de la RAM interna.

1.4.2 Modo Power Down

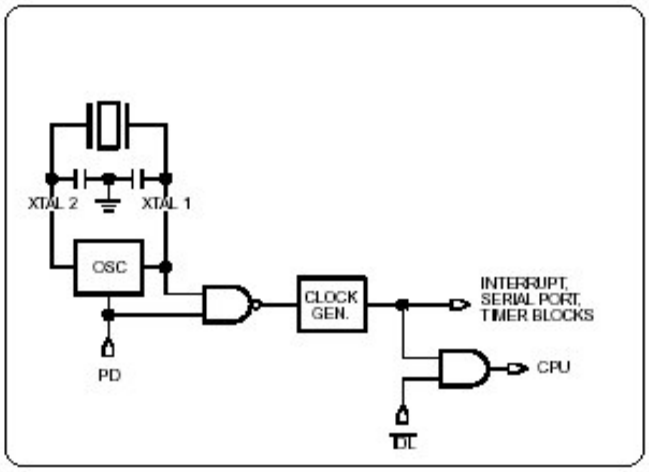
La instrucción que activa al PCON.1 es la ultima en ejecutarse antes de pasar al modo. El clock interno se detiene, y todas las funciones se paralizan pero la RAM interna y los

SFR´s son mantenidos. Los valores en los pines de los puertos se mantienen por sus respectivos SFR; mientras que ALE y PSEN son ‘bajos”.

Para salir del modo Power Down solo se es posible por un reset del hardware, el cual redefine los SFR´s pero no altera la RAM interna. Aquí la tensión puede ser reducida a 2 Vcd; pero no se lo hará antes de llamar al modo, y, se cuidará que la tensión sea restaurada antes de finalizar el modo.

El reset también libera al oscilador; éste reset no se activará antes de restaurar la tensión a la normal y se mantendrá el tiempo suficiente para asegurar el rearranque del equipo.(*mínimo 10 ms*)

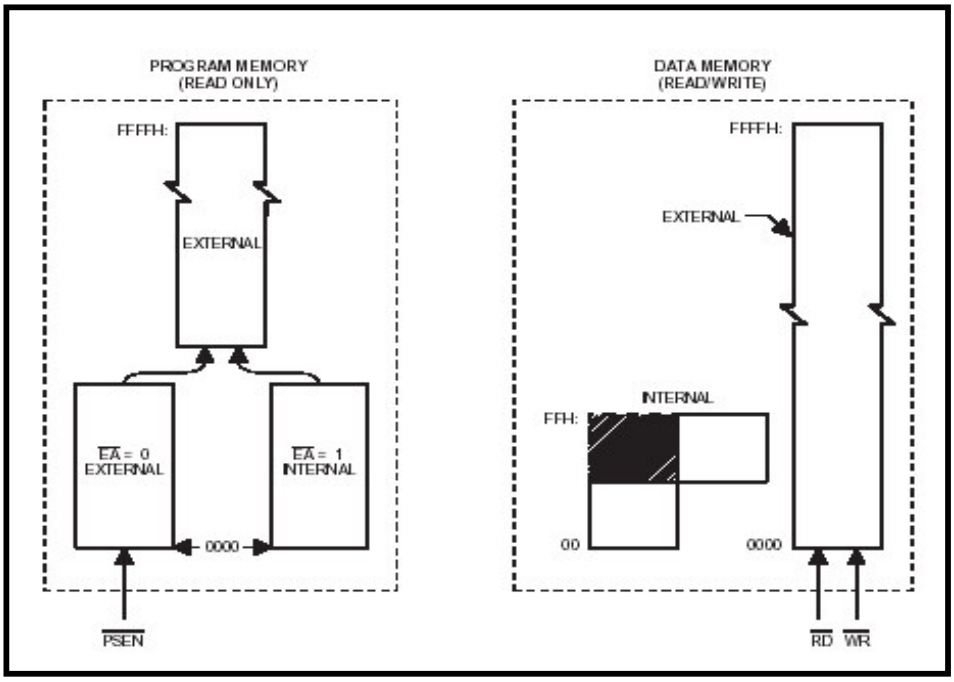
Figura 15: Hardware de modo Idle y Power down



1.5 Organización de la Memoria

Todos los microcontroladores Flash Atmel poseen separación de los espacios de memoria de programa y datos.

Figura 16: Estructura de la memoria del AT89C51/LV51 y AT89C51/LV52



Esta separación permite que la memoria de datos sea accesada por una dirección de 8 bits, que puede ser más rápidamente almacenada y manipulada por una CPU de 8 bits. Sin embargo, direcciones de memoria de datos de 16 bits pueden también ser generados por el registro DPTR.

La *memoria de programa* solo permite lectura; tiene como máximo 64 Kbytes de memoria direccionable directamente. La señal PSEN se usa para trabajar con una memoria de programa externo.

La *memoria de datos* ocupa un espacio de dirección separado desde la memoria de programa; un máximo de 64 Kbytes de memoria externa pueden ser direccionados directamente en el espacio de memoria de datos externa.

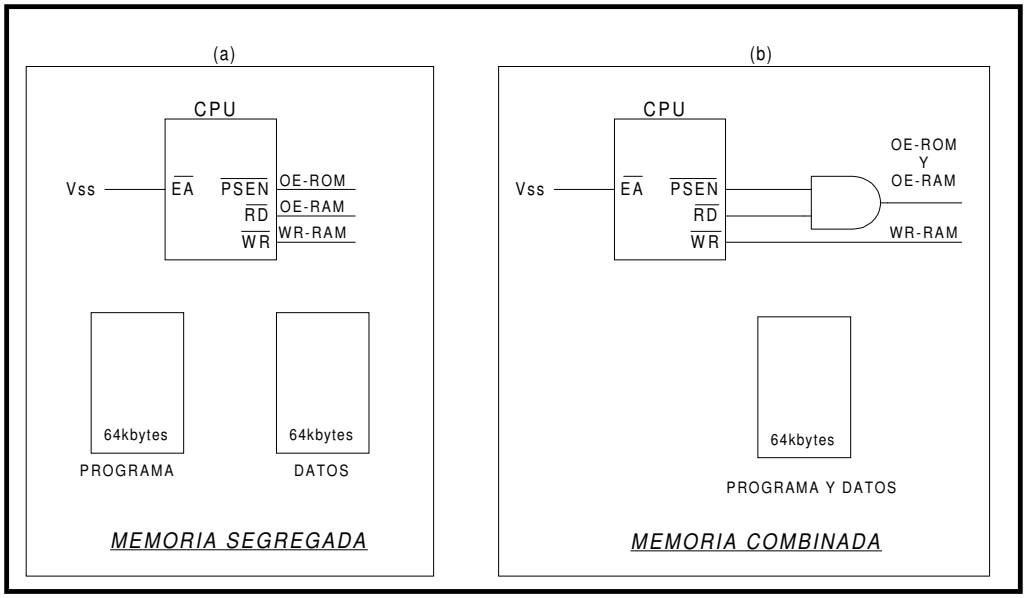
Por otra parte, la CPU genera las señales de lectura (READ) y escritura (WRITE) cuando se procede a trabajar con memoria externa.

Recopilando, se debe considerar que la memoria de programa y de datos pueden compartir el mismo espacio de direcciones o estar separadas. Tornándose en:

> Es *memoria combinada* cuando las señales RD y PSEN van a una compuerta lógica AND, y su salida actúa como "strobe" de memoria externa de programas y datos.

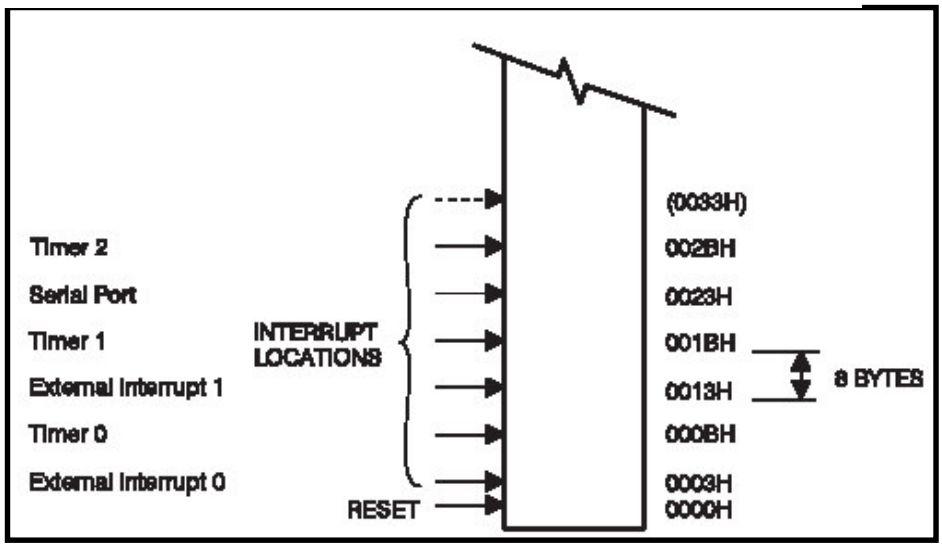
> Es *memoria segregada* cuando la memoria de datos es separada en espacio de la memoria de programa.

Figura 17 (a) y (b): Combinación de memoria



1.5.1 Memoria de Programa

Figura 18: Posiciones iniciales en la memoria de programa



El gráfico muestra un mapa de la parte más baja de la memoria de programa; es decir, que al producirse un RESET la memoria comienza a ejecutar desde la dirección 0000H. De igual manera, a cada interrupción se le asigna un lugar fijo dentro de la memoria de programa.

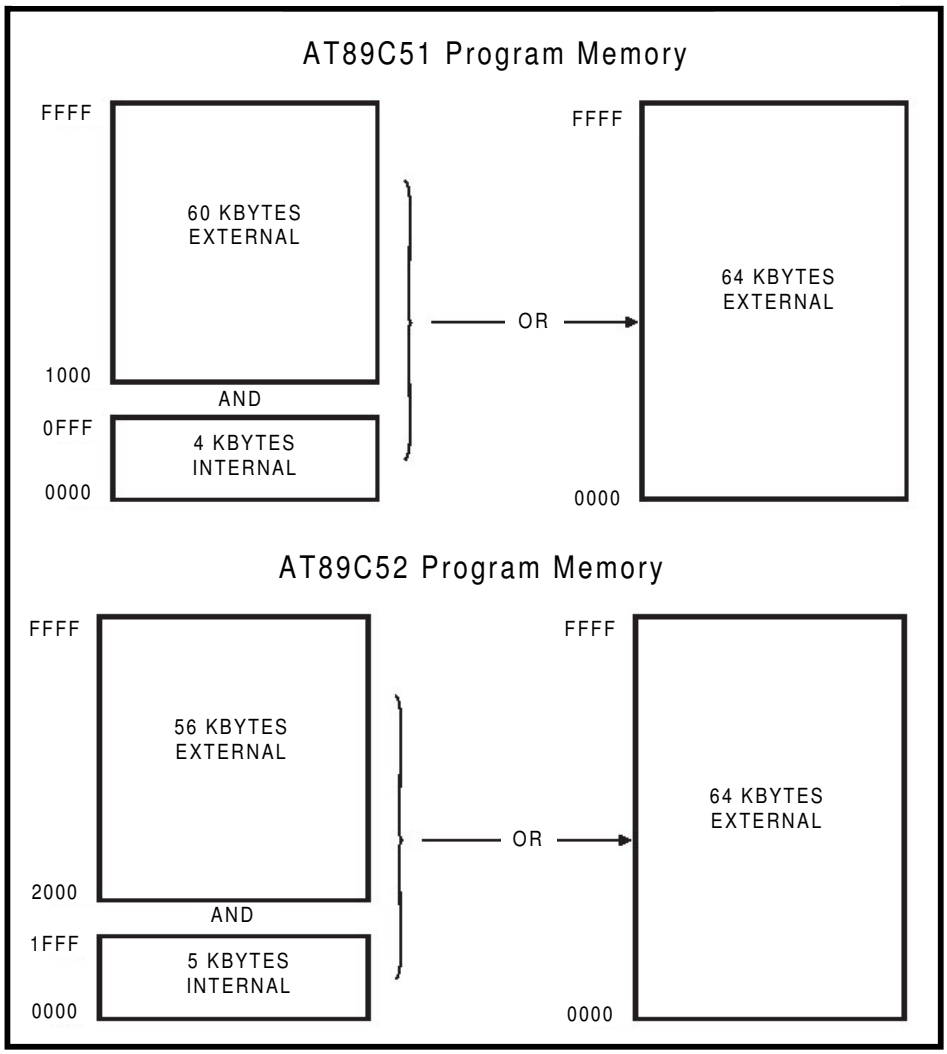
Entonces, al producirse una orden de éstas la CPU salta a la localización que es y ejecuta la subrutina encontrada.

Ejemplo:

- Dada la "interrupción externa 0", asignada como 0003H, la CPU b usca ésta y luego efectúa el servicio de subrutina hallado. Pero si la interrupción no es usada, dicha localización se puede emplear para propósitos generales.

La *expansión de la memoria de programa*, ya explicada anteriormente, sería:

Figura 19: Posiciones de la memoria de programa



- En el microcontrolador AT89C51, on – flash memory es de 4 kbytes, colocando el pin EA = +Vcc (*"1"lógico*) la CPU manda ciclos de búsqueda desde la dirección 0000H hasta la 0FFFH dentro de la on – flash memory; y desde la 1000H hasta FFFFH en la memoria de programa externo.

- En el microcontrolador AT89C52, con 8 kbytes de on – flash memory, colocando el pin EA = +Vcc (*"1"lógico*) la CPU manda ciclos de búsqueda desde la

dirección 0000H hasta la 1FFFH dentro de la on – flash memory; y desde la 2000H hasta FFFFH en la memoria de programa externo.

Pero si EA = Gnd (*"0" lógico*) todas los ciclos direcciones de búsqueda son externos.

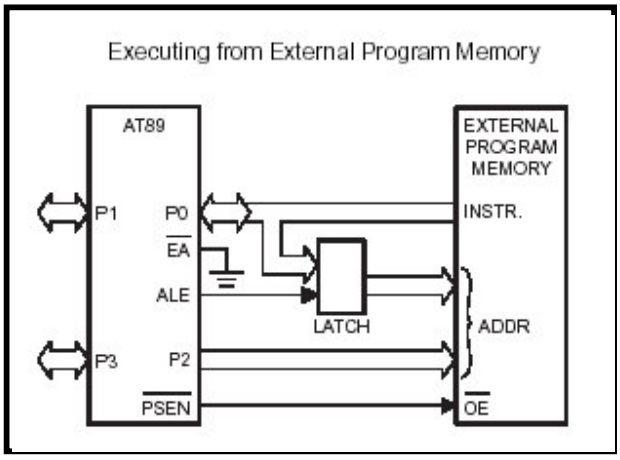
Ahora, todos los espacios de interrupciones tienen una largura de "8 bytes"; permitiendo albergar a una subrutina corta. En cambio, para subrutinas largas se pueden emplear los "saltos" para ir a secuencias de instrucciones.

La dirección más baja de la memoria de programa puede encontrarse en la Flash interna o en la memoria externa; para seleccionar alguna de éstas dos, se aprovecha la señal de EA del microcontrolador.

1.5.1.1 Accediendo a la Memoria de Programa Externo

Para efectuar esto se presenta la conexión necesaria:

Figura 20: Ejecutando memoria externa, hardware



Se puede acceder a la memoria externa para lo que son "datos" o "direcciones", para lo cual se emplea la señal PSEN como un "STROBE".

Funcionamiento:

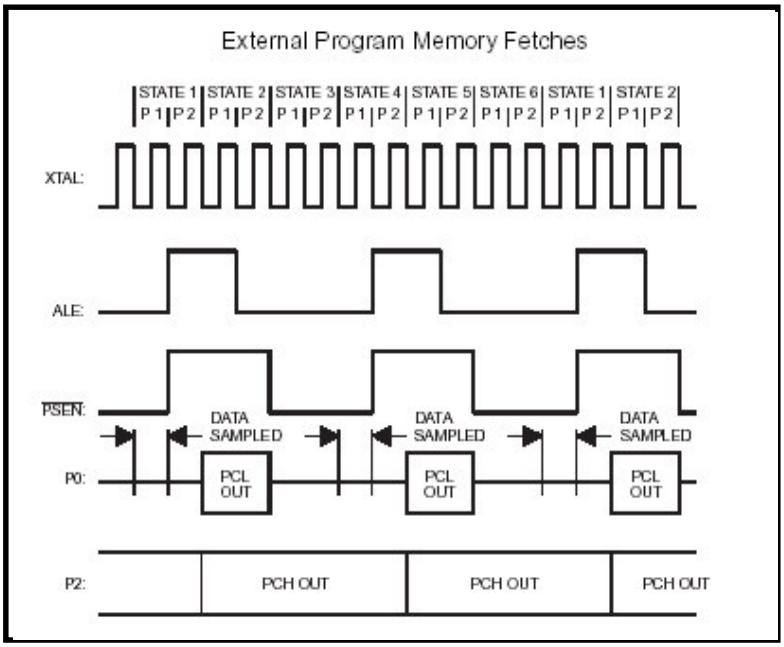
> El Puerto 0 emite el "byte bajo" del "Program Counter" (PCL) como una dirección y luego va hacia el estado flotante mientras espera por la llegada del "byte de código". En este tiempo el "byte bajo" del PC es válido en el Puerto 0, entre tanto la señal ALE introduce al latch una dirección.

Ahora el Puerto 2 emite el "byte alto" del "Progr am Counter" (PCH), luego PSEN habilita la memoria externa; el microcontrolador lee el "byte de código".

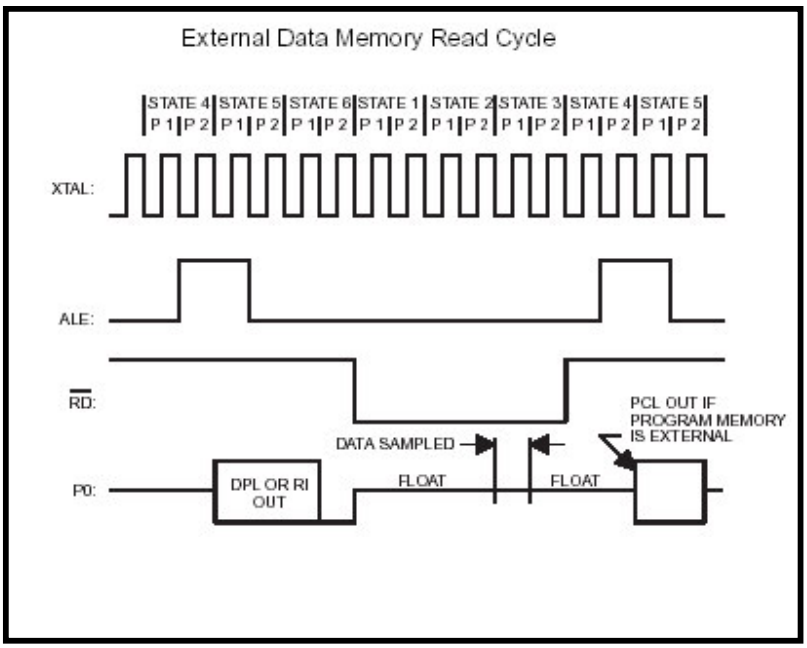
Además de "RD" y "WR" para trabajar con la memoria; los siguientes diagramas muestran los cronogramas de operación correspondientes:

Figura 21 (a), (b), (c): Temporización en memoria externa

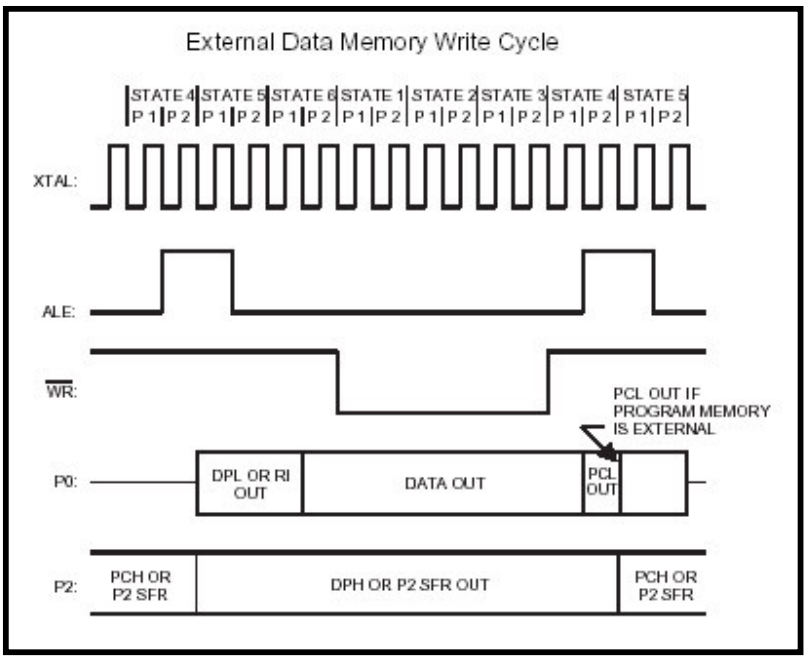
- a -



- b -



- c -



- Los diagramas muestran las señales de "STROBE" y de los "PUERTOS" operando en el tiempo con el reloj interno.

- Los tiempos de ascenso y descenso son dependientes de la carga que pueda manejar cada pin de Puerto. Hay con frecuencia cerca de 10 ns pasar de 0.8 a 2.0 Vcd

.

- Los tiempos de propagación de las formas de onda varían con la temperatura, tipo de manufactura del chip, carga del pin, tensión de alimentación; la forma de onda de XTAL puede variar desde 25 a 125 ns.

Los ciclos de búsqueda en la memoria de programa externo siempre usan direccionamiento de 16 bits, siendo las instrucciones que lo hacen:

MOVX @ DPTR ..... para 16 bits,

MOVX @ Ri ..... para 8 bits.

El "byte alto" (*dirección de 16 bits*) sale por el Puerto 2 y es sujeto por el ciclo de escritura (*en el tiempo*). Cabe recalcar que los drivers del Puerto 2 utilizan fuertes pullups durante todo el tiempo que emiten un "1", que es con:

MOVX @ DPTR instrucción

Durante éste tiempo el latch del puerto 2 no debe contener "unos", los contenidos del SFR no son modificados. Si el ciclo de memoria externa no es inmediatamente seguido por otro ciclo de memoria externa, el contenido del SFR no cambiaría, reapareciendo en el siguiente ciclo.

Con una dirección de 8 bits, MOVX @ Ri , el contenido del SFR del Puerto 2 permanecerá en los pines por todo el ciclo de memoria externa, facilitando la paginación.

En todos los casos el "byte bajo" de direcciones es multiplexado en el tiempo con el "byte de dato" en el Puerto 0, siendo los manejadores de las señales "dirección / dato" ambos FET del buffer de salida; aplicado así no requiere de pullups externas ya que no son salidas de drain abierto.

~ La señal ALE se empleará para capturar el "byte de direcciones" en el interior del latch externo. Este byte es válido en la transición negativa de ALE; luego en el ciclo de escritura los datos del byte aparecerán para ser escritos en el Puerto 0 justo antes de activar a WR, y , permanecerá allí hasta la desactivación de la señal.

En el ciclo de lectura el byte ingresado es aceptado por el Puerto 0 justo antes de que se deshabilite la señal de lectura.

Como precaución no se debe escribir ninguna información en el Puerto 0, ya que falsearía los datos; por esto la CPU, en los accesos de memoria externa, escribe 0FFFH en el latch del Puerto.

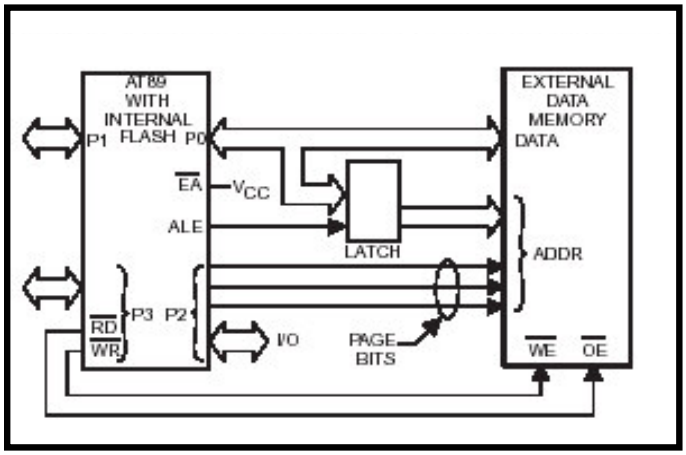
Para acceder a la memoria externa, se tiene las condiciones:

- Cuando la señal EA es activada,
- Cuando el PC contiene un número demasiado alto, más que 0FFFH; y 1FFFH para el AT89C52.

1.5.2 Memoria de datos

Para manipular datos en una *memoria externa* se tiene el siguiente hardware:

Figura 22: Accesando memoria de datos externa



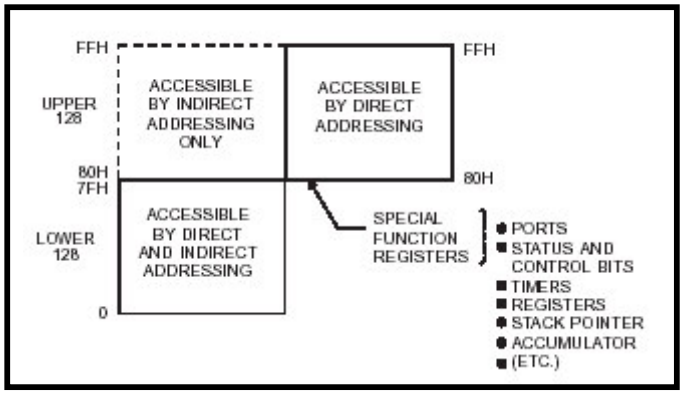
Se puede acceder a los 2 kbytes como máximo, en una RAM externa. Las directivas del programa son ejecutadas desde la memoria interna del microcontrolador; el Puerto 0 es bus multiplexado de dirección / dato, mientras la CPU produce las señales "RD" y "WR" necesarias para manejar la RAM externa.

La amplitud de la dirección puede ser de 1 o 2 bytes; direcciones de 1 byte son usadas conjuntamente con una o más líneas para paginar la RAM.

En cambio, direcciones de 2 bytes son empleadas en caso en donde el "byte de direcciones alto" sea emitido por el Puerto 2.

La *memoria interna* se subdivide en interiormente en tres bloques, que consideran los 128 bytes más bajos, los 128 bytes más altos y el espacio de los SFR:

Figura 23: Memoria de datos interno



Según la gráfica: el más alto direccionamiento directo a partir de 07H, ocupan un espacio de memoria; y, el más alto direccionamiento indirecto a partir de 07H ocupa un espacio de memoria diferente. De ésta forma se muestra los 128 bytes más altos y el espacio que tiene el SFR, en el mismo bloque de direcciones, yendo desde la 80H hasta la FFH aunque físicamente son independientes.

Las direcciones de memoria de datos interna siempre son de una amplitud de 1 byte, que implica un espacio de direcciones de 256 bytes para el AT89C51 y de 384 para el AT89C52 considerando también el SFR.

La gráfica siguiente muestra la distribución de los 128 bytes más altos:

Figura 24: Ubicación de los 128 bytes más bajos de la RAM interna

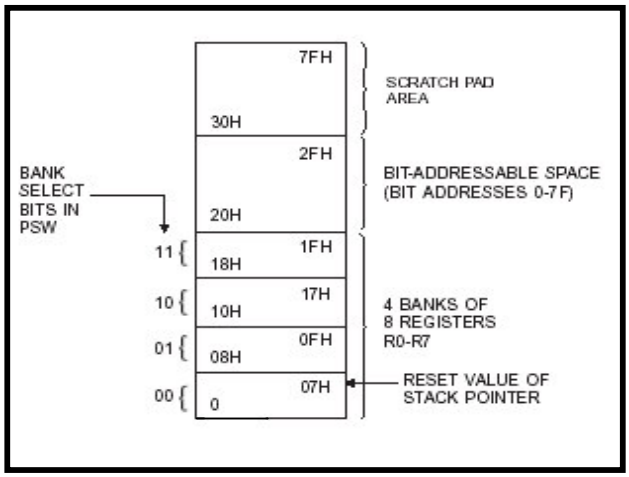
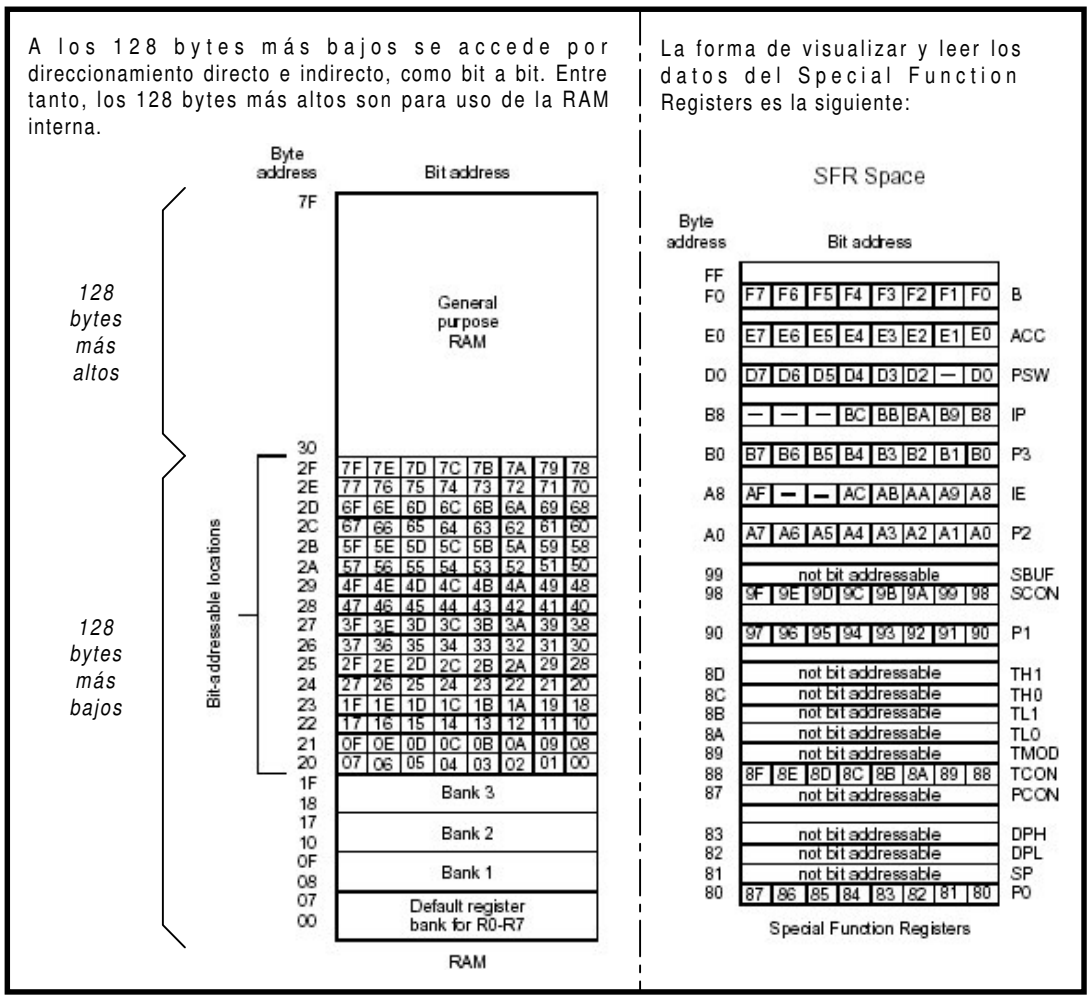


Figura 25: Direcciones de la memoria interna del microcontrolador 89C.



En el espacio del *Special Function Registers (SFR)* se localizan:

- Los latches de los Puertos,
- Los timers / counters,
- Los controles de lo periféricos, etc.

Solo se puede tener acceso en "direccionamiento directo", algunas direcciones se pueden tratar como "byte o bit a bit".

En la siguiente tabla se verifica todo aquel registro que se direcciona "bit a bit":

Tabla 5: Registro de Funciones Especiales <SFR>

Symbol	Name	Address
ACC(1)	Accumulator	0E0H
B(1)	B Register	0F0H
PSW(1)	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
P0(1)	Port 0	80H
P1(1)	Port 1	90H
P2(1)	Port 2	0A0H
P3(1)	Port 3	0B0H
IP(1)	Interrupt Priority Control	0B8H
IE(1)	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
TCON(1)	Timer/Counter Control	88H
T2CON(1)(2)	Timer/Counter 2 Control	0C8H
T2MOD(2)	Timer/Counter 2 Mode Control	0C9H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
TH2(2)	Timer/Counter 2 High Byte	0CDH
TL2(2)	Timer/Counter 2 Low Byte	0CCH
RCAP2H(2)	T/C 2 Capture Reg. High Byte	0CBH
RCAP2L(2)	T/C 2 Capture Reg. Low Byte	0CAH
SCON(1)	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

Notes: 1. Bit addressable
2. AT89C52 only

Luego de darse un RESET por hardware, los registros asumen los valores:

Tabla 6: Contenido de los SFR justo después de energizar el micro o dar un reset

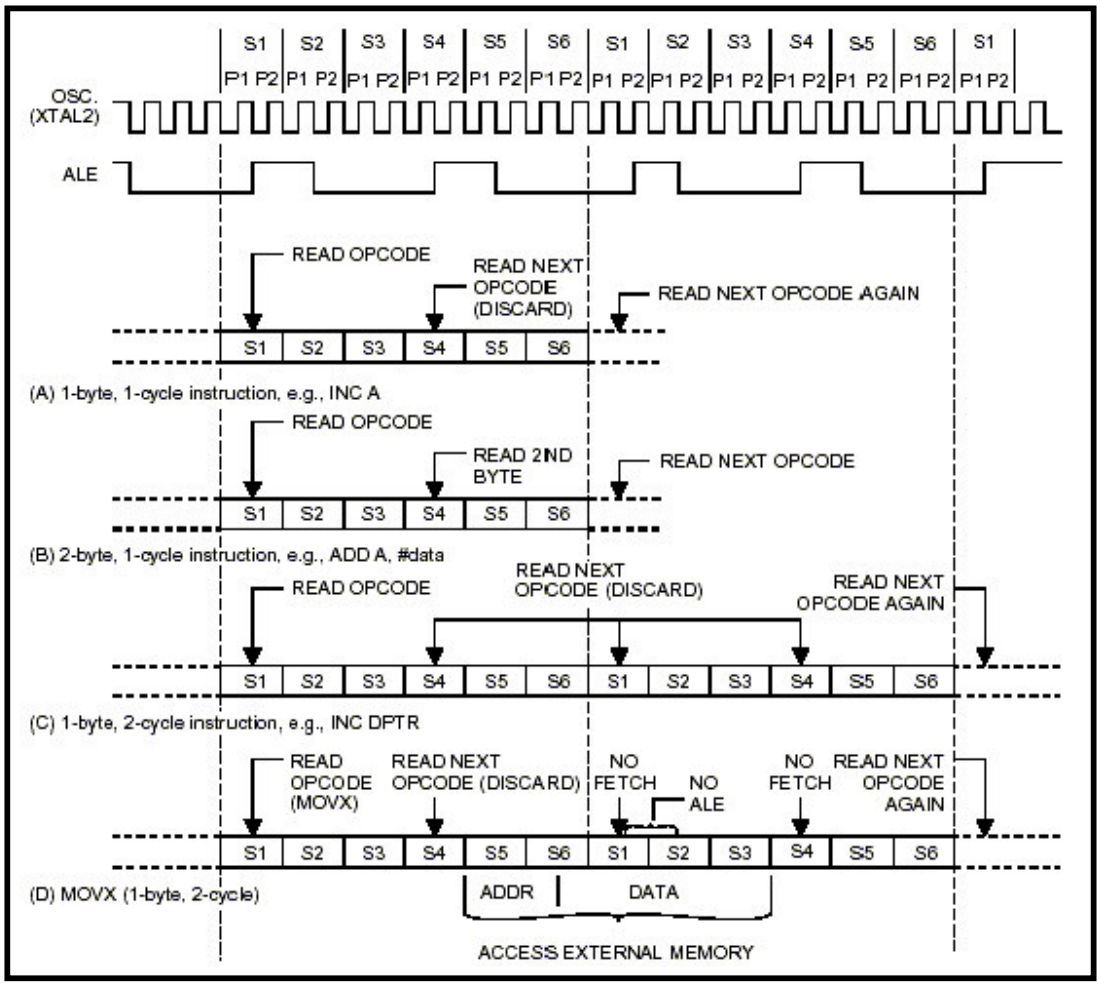
Register	Value in Binary
ACC(2)	00000000
B(2)	00000000
PSW(2)	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
PO(2)	11111111
P1(2)	11111111
P2(2)	11111111
P3(2)	11111111
IP(2)	80C51 XXX00000, 80C52 XX000000
IE(2)	80C51 0XX00000, 80C52 0X000000
TMOD	00000000
T2MOD(3)	XXXXXX00
TCON(2)	00000000
T2CON(2)(3)	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
TH2(3)	00000000
TL2(3)	00000000
RCAP2H(3)	00000000
RRAP2L(3)	00000000
SCON(2)	00000000
SBUF	Indeterminate
PCON	CMOS 0XXX0000

Notes: 1. X = Undefined 2. Bit Addressable 3. AT89C52 only

1.5.3 Ciclo de Máquina

Un ciclo de máquina consiste en una secuencia de 6 estados, S1 a S6; a cada uno lo conforma 2 períodos oscilatorios. O sea, cada ciclo consta de 12 períodos oscilatorios. Que en tiempo de duración depende del oscilador externo, cristal o RC.

Figura 26: Secuencias de operación



Cada estado se subdivide en *fase 1 (P1)* y *fase 2 (P2)*; normalmente en el programa, 2 ciclos de búsqueda son generados durante cada ciclo de máquina; ésto se da igual si la instrucción que está siendo ejecutada no lo necesitase. Si la instrucción que se está procesando no necesita más bytes de código, la CPU ignora el ciclo de búsqueda extra y el PC no es incrementado.

Con relación al gráfico, literal A y B; la ejecución de una instrucción de un ciclo de máquina comienza en el estado 1 del ciclo, cuando el "opcode" (*código de operación*) es latcheado en el interior del registro de instrucción.

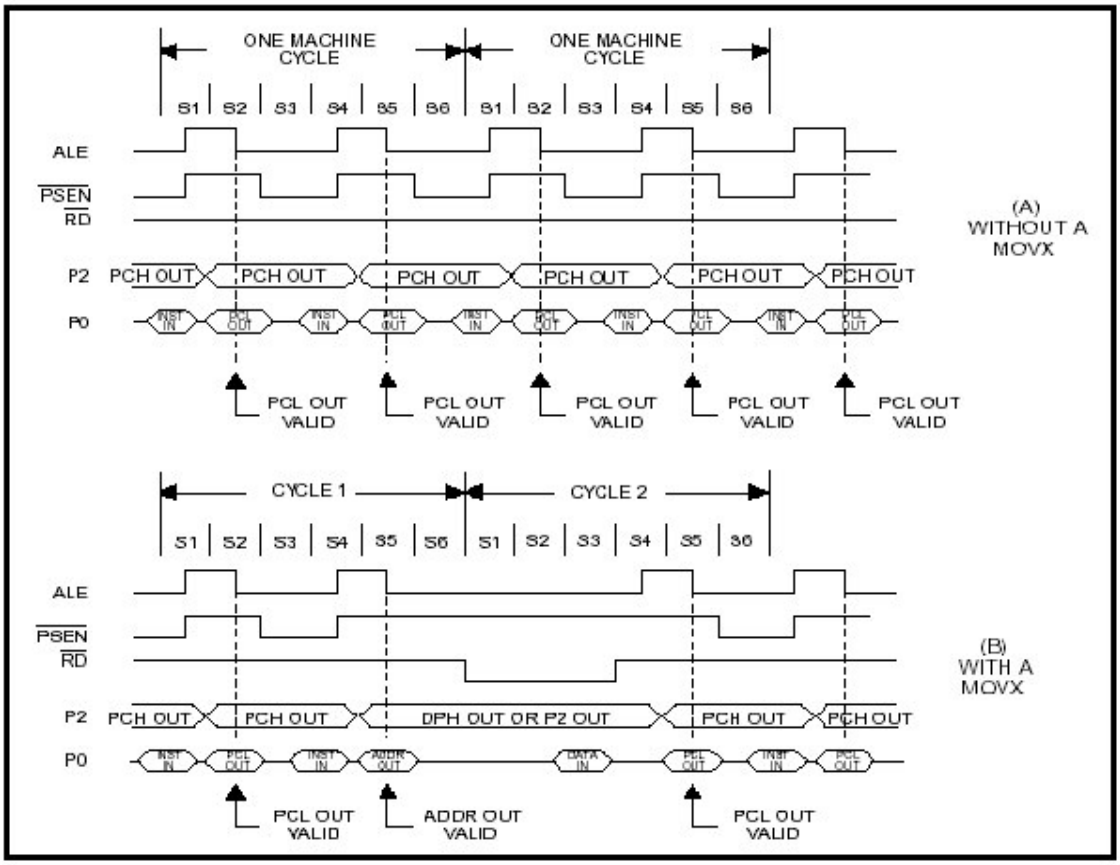
Un segundo ciclo de búsqueda ocurre durante S4 del mismo ciclo; la ejecución completa se da en el estado S6.

Luego se ve que la instrucción MOVX (literal D) tiene dos ciclos de máquina para ejecutarse, y durante el segundo ciclo de la instrucción no se genera uno de búsqueda. Siendo ésta la única vez que el programa se salta los fetchs.

Lo descrito se da en la memoria externa o interna, y los tiempos de ejecución no dependen de que memoria se utilice.

Veamos ahora las *señales y tiempos* involucrados en los fetches cuando la memoria de programa es interna o externa:

Figura 27: Ciclos de ejecución de la memoria de programa externo.



Si es la memoria externa PSEN es habilitado dos veces por ciclo de máquina, como se observa en el literal A. Sí un acceso a la memoria de datos ha sucedido, literal B, las habilitaciones anteriores son saltadas; porque el bus de direcciones y datos están siendo usadas por el acceso a la memoria de datos.

Un ciclo del bus de memoria de datos toma el doble de tiempo que un ciclo del bus de memoria de programa. También se muestra el tiempo de las direcciones emitidas por el P0 y P2, las señales ALE y PSEN.

Cuando la CPU está ejecutando desde la memoria de programa interna, PSEN no se activa y las direcciones del programa no son emitidas. Sin embargo, ALE continua activándose dos veces por ciclo de máquina y es por eso disponible como una señal de salida tipo clock.

Tomar en consideración que un ALE es saltado durante la ejecución de la instrucción MOVX.

1.6 Programación de la EPROM interna

1.6.1 Características

El tipo de memoria interna flash permite ser reprogramada en operación o por algún programador convencional. Para esto, al pin Vpp del AT89C51 / AT89C52 se debe colocar un pulso de +12 Vcd por un lapso de 100 ms en el pin PROG, por byte a escribir.

Generando un tiempo de programación aproximado de 1.5 ms por byte, por lo que se tardarían 6 segundos para los 4 kbytes y, 12 segundos para los 8 kbytes.

1.6.2 Seguros del programa

En la gran mayoría de microcontroladores, sin excepción de marca, se suele asegurar la programación para impedir cualquier violación del mismo. Suelen tener dos alternativas:

1. Por cadena de encriptación, y,
2. Por bits de seguridad.

En este caso, para los ATMEL, se dispone de bits que se pueden programar o desprogramar para dar ciertas características de seguridad:

Tabla 7: Microcontroladores flash AT89C51 y AT89C52

Device Name	Flash Bytes	Ckt Type	V_{pp}	Time Required to Program Entire Array
AT89C51	4K	CMOS	12V	6 seconds
AT89C52	8K	CMOS	12V	12 seconds

Tabla 8: Protección del programa

Device	Lock Bits
AT89C51	LB1, LB2, LB3
AT89C52	LB1, LB2, LB3
AT89C2051	LB1, LB2
AT89C1051	LB1, LB2

El procedimiento para programar los bits de seguridad es detallado en los data sheets.

Tabla 9: Programación de los bits de seguro y sus características

Programar bits de seguro				Tipo de protección
Modo	LB1	LB2	LB3	
1	Off	Off	Off	Deshabilitadas características de seguridad
2	On	Off	Off	La instrucción MOVC es ejecutada desde la memoria externa de programa,estan deshabilitadas por el fetching del código de bytes de la memoria interna.EA es muestreado y latcheado sobre el reset, y las ayudas de programación de la flash es deshabilitada.
3	On	On	Off	Igual a 2, también la verificación es deshabilitada.
4	On	On	On	Igual a 3, también la ejecución externa es deshabilitada.

Al ser programado el "lock bit 1", el nivel lógico de EA es muestreado y latcheado durante el RESET. Si es que el microcontrolador enciende sin producirse un RESET, el latch inicializa con un valor aleatorio que es mantenido hasta que el RESET ocurra.

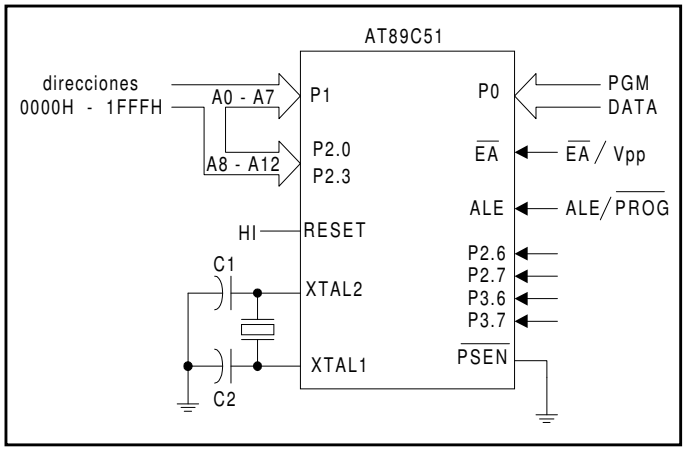
>> *El valor latcheado de EA concordaría con el nivel lógico de corriente que maneje el pin, para un funcionamiento apropiado.*

>> *Borrando la flash también se borraría los bits de seguridad, pero el microcontrolador retornará a su plena capacidad.*

1.6.3 Procedimientos para grabar y verificar

Para proceder a la *grabación de la flash*, se debe seguir los siguientes pasos:

Figura 28: Proceso de grabación de la flash.



- α Colocar una señal de reloj entre 6 y 24 MHz.
- α Direccionar la posición de memoria que se ha de cargar, realizado por los pines del P1 y P2.
- α Ingresar el dato en las líneas del P0.
- α Activar el pulso de programación en EA / Vpp.
- α Activar la correcta combinación de las señales de control:

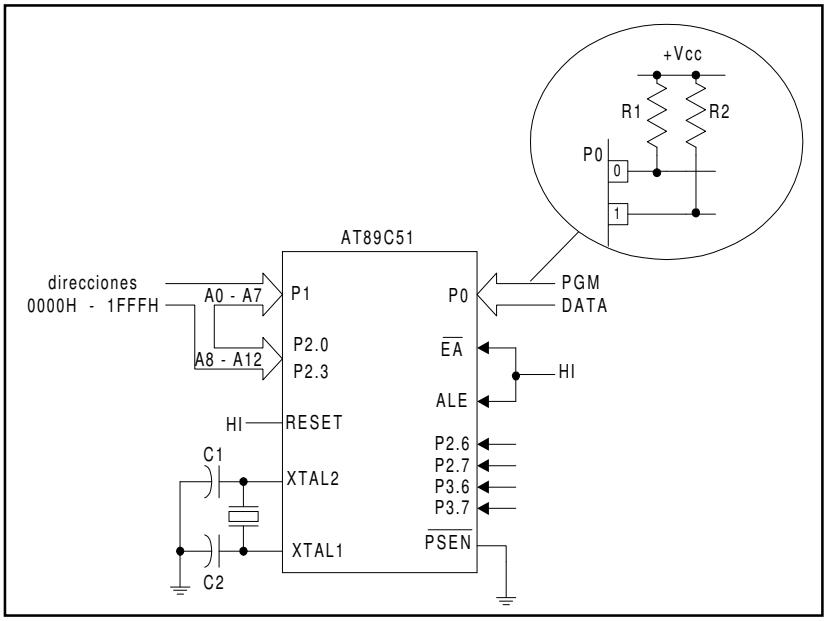
Tabla 10: Modos de programación de la EPROM flash del AT89C51

Mode		RST	PSEN	ALE/PROG	EA/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	(1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Nota: El borrado requiere de un pulso de 10 ms en PROG

Para la *verificación de la memoria*, siempre y cuando el programa no tenga activados los bits de seguridad, se procede así:

Figura 29: Proceso de verificación de la grabación.

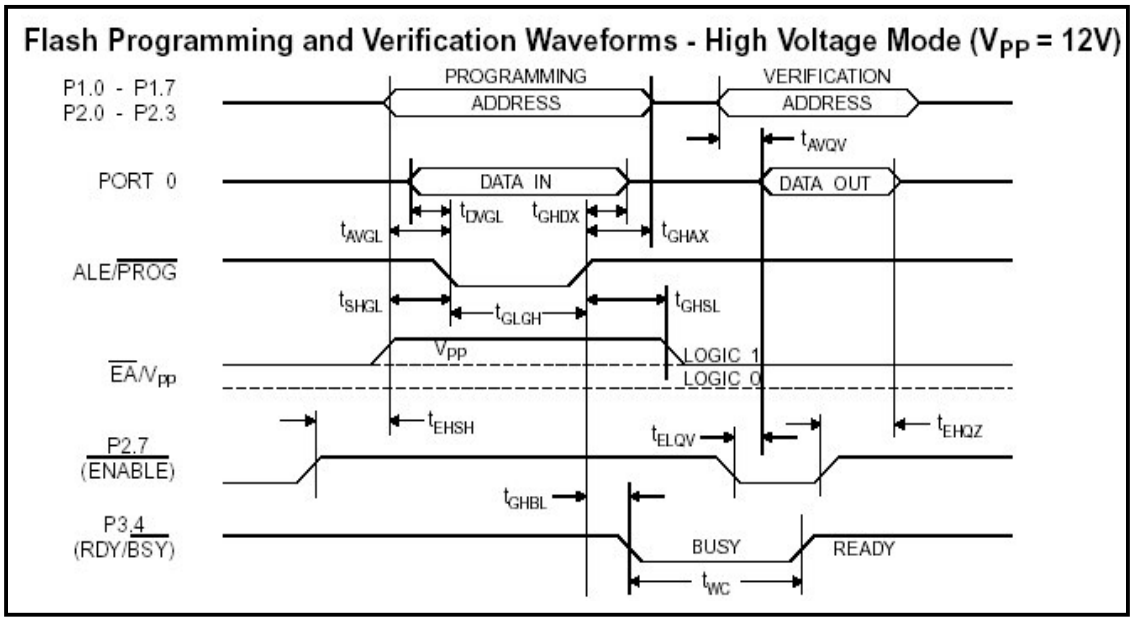


- Las direcciones de los lugares de la información en la memoria son aplicadas al P1 y los pines del P2 correspondientes, y al resto de los pines se les dará los niveles de tensión expuestos en la tabla anterior.

- La información es obtenida en los pines del P0, las cuales llevarán resistencias pullups externas.

El cronograma secuencial de estas actividades es el siguiente:

Figura 30: Temporización para la grabación y verificación de la flash del AT89C51.



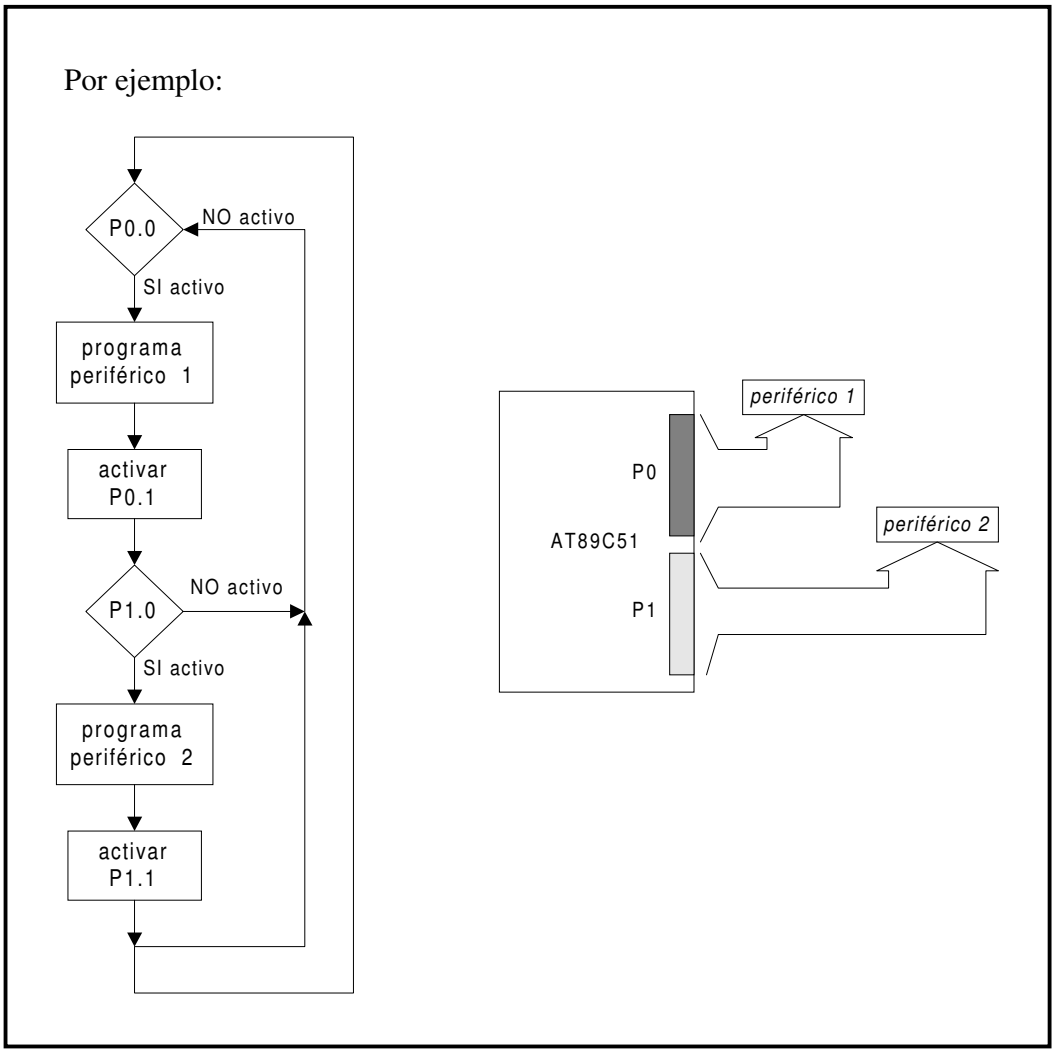
1.7 Interrupciones

1.7.1 Introducción

Dentro de un sistema, la comunicación asíncrona que existe entre los periféricos y la CPU, bidireccional, puede ser de las formas:

- ℜ Consultas (*polling*): Se comprueban cíclicamente; por medio de instrucciones del programa; los registros de estado de los dispositivos de I / O.

Figura 31: Consultas para efectuar una interrupción.



El programa pregunta si se ha activado el bit 0 del Puerto 0, o sea que el periférico 1 requiere de atención; al finalizar activa el bit 1 del Puerto 0 para indicar que ha concluido la secuencia. Igual descripción para el periférico 2.

Este método presenta desventajas:

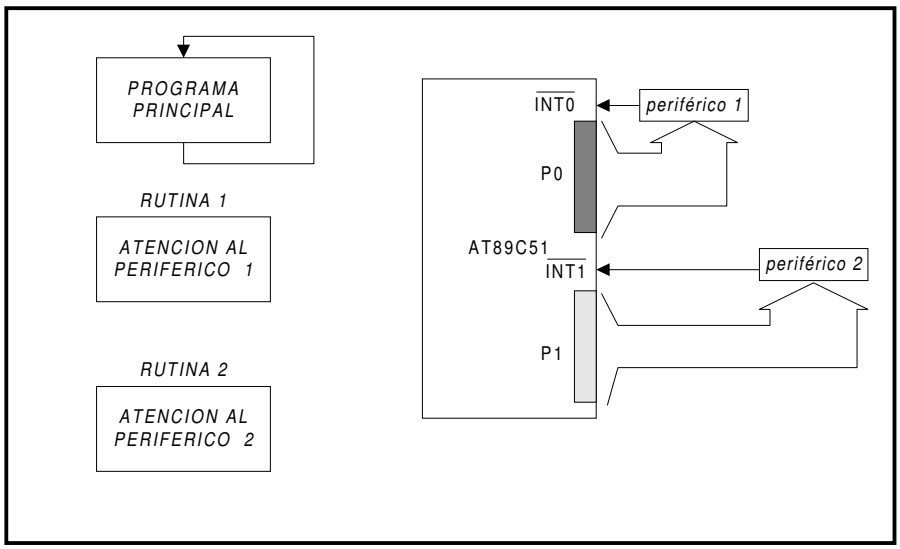
1. Para cada secuencia de programa se tiene que averiguar el estado de los bits de consulta.
2. La atención del periférico es después de efectuada la consulta, y no cuando se pide la intervención de la CPU .

ℜ *Interrupciones*: Se establece un servicio directo entre la CPU y los periféricos, siempre que a CPU lo decida.

Las características son:

- Es inmediato,
- Si ha ocurrido una interrupción que trastocaría el programa, se la puede inhibir.
- Permite que a las consultas se las pueda descartar en la mayor parte de los programas.
- No depende del control de procesos realizados en tiempo real.
- Es importante porque ejecuta un subproceso en el instante preciso y, luego de terminarlo, la CPU retorna al punto donde dejo el programa principal.

Figura 32: Consultas para efectuar una interrupción (mejorado).



Este pedido de interrupción se inicia al:

- ¬ Solicitarla por un agente que da la orden en los pines específicos, en este caso es externa.
- ¬ Ser generada por el propio chip, siendo ahora interna.

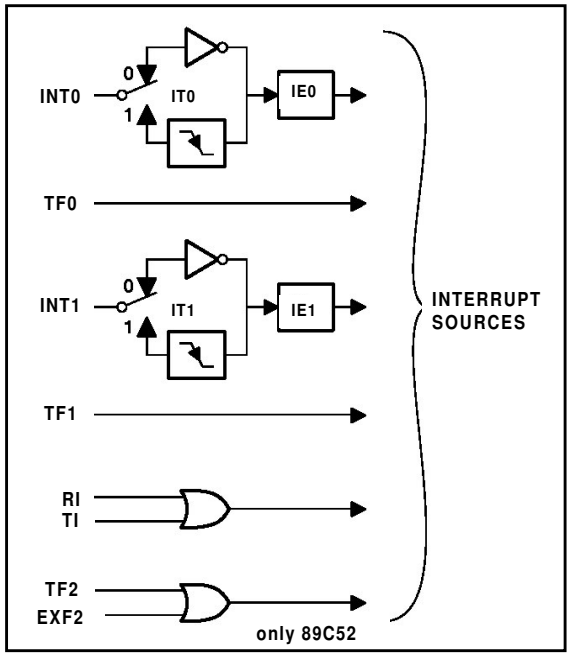
.1.7.2 Aspectos Particulares

Los microcontroladores ATMEL 89C51 tienen implementados 5 niveles de interrupción:

- α 2 interrupciones externas,
- α 2 interrupciones producidas por timer,
- α 1 interrupción del puerto serial.

Mientras que el ATMEL 89C52, tiene 6; son las mismas que las anteriores con la adición de un timer extra.

Figura 33: Fuentes de interrupción



Para lograr habilitar o deshabilitar a las interrupciones se tiene un registro que posibilita hacerlo con cada una de ellas, por "set" o "clear" del bit adecuado; éste registro es el IE localizado en el SFR. Así mismo, se cuenta con un bit que controla todo el registro:

Tabla 11: IE (Registro Habilitador de Interrupciones), es direccionable por bit

(MSB)							(LSB)
EA	**----**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

EA	IE.7	Bit de control del registro. Si EA = 0 se deshabilita todas las interrupciones ; si EA = 1 se habilita todas las interrupciones. Pero cada interrupción es controlada por cada bit de control.
----	IE.6	no implementado.
ET2	IE.5	Bit habilitador de interrupción del timer2 (AT89C52).
ES	IE.4	Bit habilitador de la interrupción del puerto serial.
ET1	IE.3	Bit habilitador de la interrupción del timer 1.
EX1	IE.2	Bit habilitador de la interrupción externa 1
ET0	IE.1	Bit habilitador de la interrupción del timer 0.
EX0	IE.0	Bit habilitador de la interrupción externa 0.

El "set" o "clear" de los bits se lo puede hacer por software, con los mismos resultados como si fueran por hardware; o sea, se pueden controlar por software. La posición IE.5 no contiene bit, en el registro, no debe escribirse por software; ya que es para implementaciones posteriores.

1.7.3 Tipos de Interrupciones

1.7.3.1 Interrupciones Externas

Estas son INT0 e INT1, las mismas que pueden generarse por la activación de un nivel o de una transición; pero dependiendo de los bits localizados en el registro TCON IE.0 e IE.1 . Son disparadas por un "bajo" colocado en el pin correspondiente; pero si se detecta un alto en un ciclo y un bajo en el siguiente, el flag IE.X es seteado.

Tabla 12: TCON (Registro de Control de Timer/Counter), es direccionable por bit

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Flag de overflow del timer 1. Es seteado por hardware cuando se da un overflow en el timer/counter 1. Limpiado por hardware cuando el procesador vectoriza la rutina de interrupción.
TR1	TCON.6	Bit de arranque del timer 1. Es seteado/limpiado por software para encender el timer / counter 1.
TFO	TCON.5	Flag de overflow del timer 0. Es seteado por hardware cuando se da un overflow en el timer/counter 0. Limpiado por hardware cuando el procesador vectoriza la rutina de interrupción.
TR0	TCON.4	Bit de arranque del timer 1. Es seteado/limpiado por software para encender el timer / counter 1.
IE1	TCON.3	Flag de flanco para la interrucpción externa 1. Es seteado por hardware cuando se detecta un flanco en el pin de la interrupción. Luego de procesar la interrupción, es limpiado por hardware.
IT1	TCON.2	Bit de control de interrupción externa 1. Seteado /limpiado por software para especificar que se ha dado un flanco descendente de disparo.
IE0	TCON.1	Flag de flanco para la interrucpción externa 0. Es seteado por hardware cuando se detecta un flanco en el pin de la interrupción. Luego de procesar la interrupción, es limpiado por hardware.
IT0	TCON.0	Bit de control de interrupción externa 0. Seteado /limpiado por software para especificar que se ha dado un flanco descendente de disparo.

Al peticionar el servicio de rutina sea vectorizado, el hardware del microcontrolador limpia el flag que señaló la interrupción; solamente si fue activada por una transición.

Si fue activada por un nivel, entonces la fuente de requerimiento externa controla al flag, y su "clear" se lo hace por una instrucción de programa.

1.7.3.2 Interrupciones por Timer

Para los Timers 0 y 1, las interrupciones son generadas por TF0 y TF1 que se pone en "set" por un movimiento en el mismo registro. A excepción del Timer 0 en modo 3.

Cuando una interrupción por Timer se ha producido, el hardware interno del microcontrolador limpia el flag una vez que se ha vectorizado la rutina de servicio.

Para el Timer 2 implementado en el AT89C52, se lo hace por el manejo de los bits: TF2 y EXF2, empleando una lógica OR entre las dos señales; el flag se borra por software y no por hardware. Pero accediendo primero al T2CON:

Tabla 13: T2CON (Registro de Control de Timer/Counter 2), direccionable por bit

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2	T2CON.7	Flag de overflow del timer 2. Es seteado por hardware y limpiado por software. TF2 no pude ser seteado ya sea por RCLK = 1 o TCLK = 1.
EXF2	T2CON.6	Flag externo del timer 2. Seteado en los modos de captura o recarga causado por una transición negativa sobre T2EX, y EXEN = 1. Cuando la interrupción del timer 2 es habilitada, EXF2 = 1 causa que la CPU vectorize a la rutina de interrupción del timer2. EXF2 será limpiado por software.
RCLK	T2CON.5	Flag de recepción de reloj. Cuando es seteado, impulsa al puerto serial a usar pulsos de overflow como reloj para la recepción en los modos 1 y 3. RCLK = 0 provoca que el overflow del timer 1 sean usados como reloj de recepción.
TCLK	T2CON.4	Flag de reloj de transmisión. Al ser 1, induce al puerto serial a usar los pulsos de overflow del timer 2 para el reloj de transmisión en modos 1 y 3. Al ser 0, los overflows del timer 1 son usados para reloj de transmisión.
EXEN2	T2CON.3	Flag habilitador externo del timer 2. En 1, permite capturar o recargar como un resultado de una transición negativa en T2EX si el timer 2 no esta siendo usado como reloj del puerto serial. EXEN = 0 hace que el timer 2 ignore los eventos en T2EX.
TR2	T2CON.2	Control de software on/off para el timer 2. Una lógica de 1 arranca el timer.
C/$\overline{T2}$	T2CON.1	Selector de timer o counter . 0 = timer interno , 1 = counter externo disparado por flancos descendentes.
CP/$\overline{RL2}$	T2CON.0	Flag de captura/ recarga. En 1 captura sobre una transición negativa en T2EX si EXEN = 1. En 0, autorrecarga ya sea con overflows del timer 2, o, transiciones negativas en T2EX si EXEN = 1. Este bit es ignorado y el timer es forzado a autorecarga en el overflow del timer 2.

1.7.4 Prioridad entre Interrupciones

Cada una de las interrupciones puede programarse para uno o dos niveles de prioridad, por la intervención del registro IP del SFR:

Tabla 14: IP (Registro de Prioridad de Interrupción),es direccionable por bit.

----	----	PT2	PS	PT1	PX1	PT0	PX0

----	IP.7	No implementado.
---	IP.6	No implementado.
PT2	IP.5	Define el nivel de prioridad de la interrupción del timer 2 (solo en AT89C52).
PS	IP.4	Define el nivel de prioridad de la interrupción del puerto serial.
PT1	IP.3	Define el nivel de prioridad de la interrupción del timer 1.
PX1	IP.2	Define el nivel de prioridad de la interrupción externa 1.
PT0	IP.1	Define el nivel de prioridad de la interrupción del timer 0.
PX0	IP.0	Define el nivel de prioridad de la interrupción externa 0.

El IP es limpiado después de resetear el sistema, lo hace para colocarse en la interrupción de más bajo nivel por "default". Para lo cual se sigue los siguientes pasos:

- Setear el bit EA en el registro IE.
- Setear el correspondiente bit habilitador individual de la interrupción en el registro IE.
- Comenzar el servicio de rutina correspondiente al vector direccionado.

Tabla 15: Prioridad y dirección de Interrupciones.

Fuente de interrupción	Fuente de interrupción	Nivel de prioridad
IE0	0003H	más alto
TF0	000BH	
IE1	0013H	
TF1	001BH	
R1 & T1	0023H	
TF2 & EXF2 (1)	002BH	más bajo

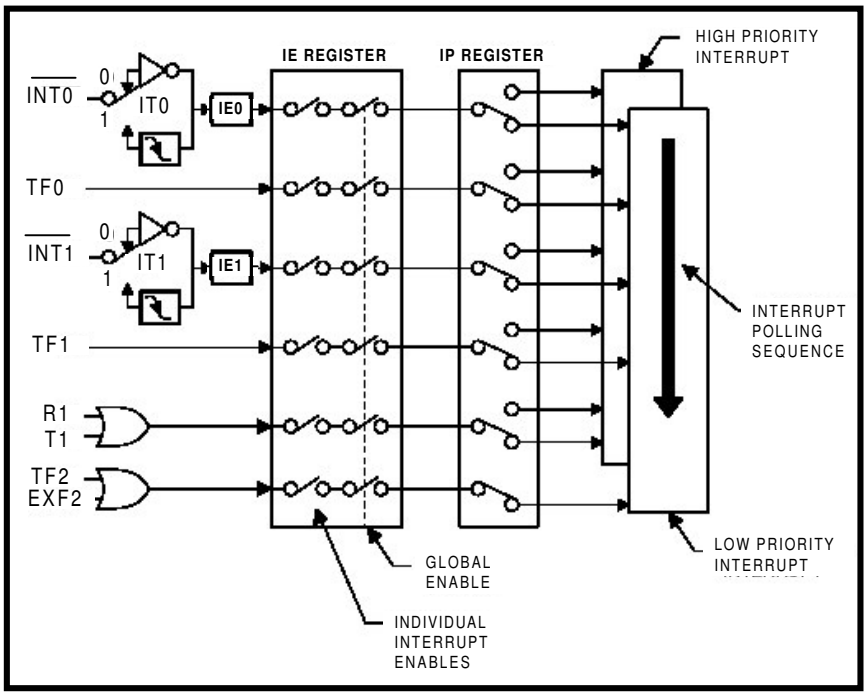
(1) Solo disponible en el AT89C52.

Se debe tener presente las próximas consideraciones:

- Una interrupción de bajo nivel puede ser cortada por una interrupción de alto nivel, pero no por otra de más bajo nivel. Una prioridad del más alto nivel no puede ser interrumpida por otra fuente externa.

- Si dos requerimientos de interrupción, con diferente nivel de prioridad, se reciben simultáneamente; se atiende aquel requerimiento que tenga el más alto nivel. Lo que se determina por una secuencia interna de consulta a los requerimientos. Así, dentro de cada nivel de prioridad la secuencia de consulta determina una segunda estructura prioritaria.

Figura 34: Sistema de Control de Interrupciones



1.7.5 Secuencias de la Interrupción

Los flags de interrupción son muestreados en el S5P2 de cada ciclo de máquina, y consultados en el siguiente ciclo de máquina.

Si se encontrase el "SET" en alguna condición de i nterrupción en el ciclo anterior a S5P2, el ciclo de consulta la encontraría; y, el sistema de interrupción vectorizará a la rutina correspondiente mediante un "LCALL". Proveído este LCALL generado por el hardware, solo se bloqueará por:

- ∉ Una interrupción de igual o más alto nivel de prioridad, que esté en progreso en ese momento.
- ∉ No haya finalizado el ciclo de la instrucción que en ese instante se esté procesando.
- ∉ La instrucción en proceso es "RETI" o se esté produciendo un acceso a los registros IP o IE.

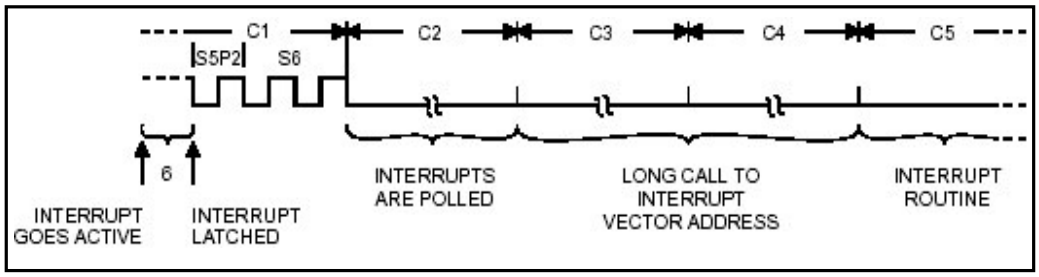
La segunda instancia asegura que la instrucción que se está procesando será completada antes de vectorizar cualquier servicio de interrupción. Entre tanto la instancia 3 asegura que si la instrucción en progreso es RETI, o cualquier acceso a IP o IE; entonces la instrucción más pequeña será ejecutada antes de vectorizar cualquier interrupción.

Si el flag de una interrupción activa no está siendo tomado en consideración por alguna de las condiciones ya mostradas, puede suceder lo siguiente:

- Que la petición de bloqueo haya desaparecido y que el flag siga activado, la interrupción ahora la atenderá el microcontrolador.

- Que la petición de bloqueo desaparezca, y que en ese intervalo de tiempo el flag se desactiva; la petición de interrupción será negada y se tendrá que volver a pedirla.

Figura 35: Respuesta de tiempo de una interrupción



Cabe anotar que si una interrupción del más alto nivel de prioridad está activa antes de S5P2 del ciclo de máquina etiquetado como C3, y de acuerdo a las reglas expuestas anteriormente, ésta interrupción será vectorizada durante C5 y C6. Sin que ninguna de las rutinas de las interrupciones de más baja prioridad haya sido ejecutada.

En algunos casos, luego del LCALL (generado por hardware), a las rutinas de interrupción se borra el flag que ha generado la interrupción; pero nunca se puede hacerlo con los flags del Puerto Serial y Timer 2. A las dos se las borra por software.

Este LCALL empuja el contenido del Program Counter al interior del Stack, y recarga el PC con la dirección que depende de la interrupción atendida:

Tabla 16: Secuencia de interrupción

Interrupción	Fuente	Dirección
Externa 0	IE0	0003H
Timer 0	TF0	000BH
Externa 1	IE1	0013H
Timer 1	TF1	001BH
Puerto Serial	RI or TI	0023H
Timer 2	TF2 or EXF2	002BH
Reset	RST	0000H

Cuando se vectoriza a una interrupción el flag que la causa es automáticamente limpiado por hardware. La excepción es las interrupciones del puerto serial RI y TI , y, TF2 y EXF2 interrupciones del timer 2. Ya que hay dos fuentes posibles de interrupción para cada una de éstas, no es práctico para la CPU limpiar el flag indicado. Estos bits serían probados en el ISR para determinar la fuente la interrupción, para luego el flag ser limpiado por software.

En conclusión, cuando una interrupción es aceptada pasa lo siguiente:

- La instrucción corriente completa su operación,
- El PC se guarda en el Stack,
- El estado de la instrucción corriente es guardado interiormente,
- Las interrupciones son bloqueadas al nivel de las interrupciones,
- El PC es recargado con el vector dirección de la rutina de interrupción,
- Se ejecuta la rutina establecida.

Es por esta razón que se debe finalizar los pedidos de interrupción con la instrucción "RETI", para retornar luego a la ultima dirección antes del pedido.

1.7.6 Simulación de un 3er nivel de interrupción

Los dos niveles vistos dentro de las prioridades de interrupción son producidos por el hardware interno del microcontrolador; pero al necesitarse un tercer nivel se lo puede implementar por medio del software, produciendo los mismos efectos.

Se establece primero el nivel de prioridad más alto con el registro IP; las rutinas de servicio para la interrupción de prioridad 1 son presumidos para ser interrumpidos por interrupciones de prioridad 2; escribiéndolos así:

```
PUSH        IE
  MOV       IE,#MASK
CALL        LABEL
**********************

ejecución del servicio de rutina

**********************
POP  IE
RET
LABEL: RETI
```

Luego de que cualquier interrupción de prioridad 1 es reconocida, el registro IE es redefinido para deshabilitar todo, excepto la interrupción de prioridad 2. Después un

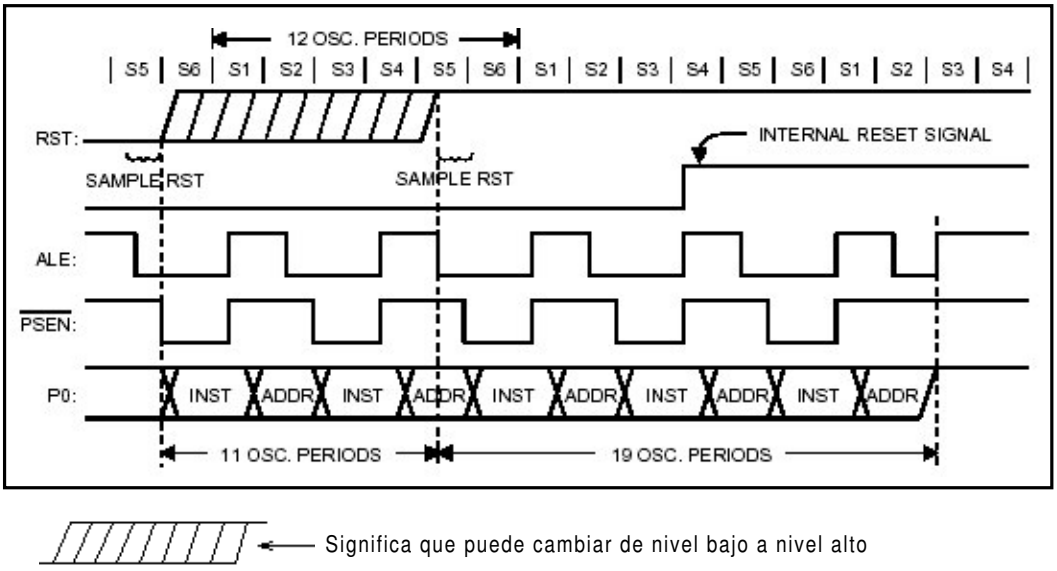
CALL a LABEL ejecuta la instrucción RETI; que limpia el flip – flop de la interrupción de prioridad 1 en progreso. En este punto, cualquier nivel de interrupción de prioridad 1 puede ser atendida, pero solamente interrupciones de prioridad 2 son habilitadas.

POP IE realmacena el byte habilitador original; un RET es usado para terminar la rutina. El software adicional agrega 10 ms, con un clock de 12 MHz, a la interrupción de prioridad 1. (*atmel.net*).

1.7.7 Situación del RESET

El pin de ingreso del RESET va hacia una *Schmitt Trigger*. Se cumple un "RST" cuando por 2 ciclos de máquina como mínimo se mantiene en nivel lógico "alto"; ésto es por 24 períodos de oscilación mientras el clock está operando. Entonces, la CPU responde generando un reset interno:

Figura 36: Tiempos de un RESET

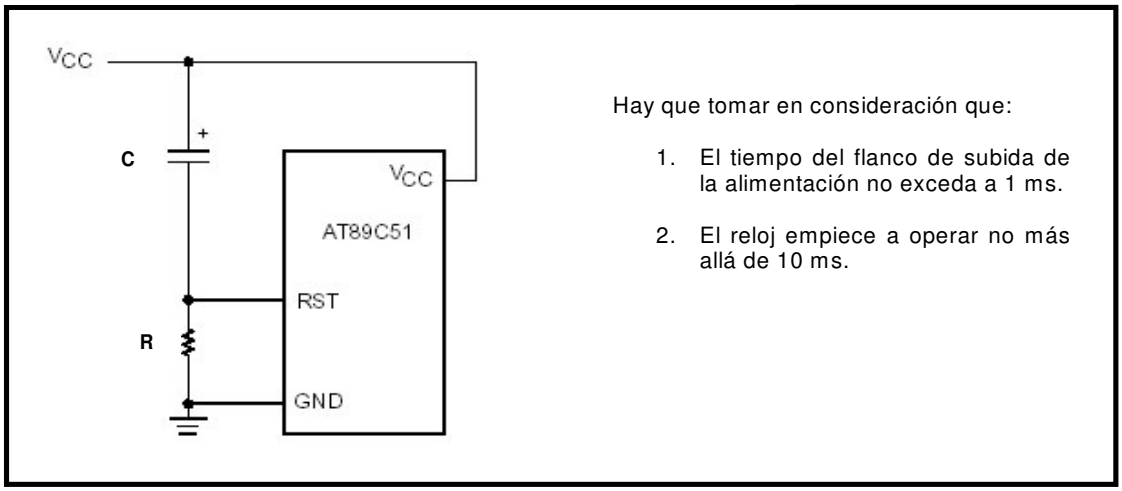


La señal externa del reset es asíncrona al reloj. El pin es muestreado durante en el ciclo 2 estado 5 S5P2 de cada ciclo de máquina; entre tanto, los pines de los puertos mantendrán por 19 períodos de oscilación más sus tareas a partir de la detección del "alto" en el reset. Esto es, después de que la señal externa este aplicada por 19 a 31 períodos de reloj.

Ahora, mientras el pin de RST = 1; las señales ALE y PSEN son jaladas a un nivel "alto", luego el RST es jalado a un "bajo". Esto llevaría un tiempo de 1 a 2 ciclos de máquina para que ALE y PSEN arranquen con el clock. Dada ésta situación, los demás equipos periféricos no se pueden sincronizar con los tiempos internos del microcontrolador.

Además, al inicializar el microcontrolador se puede generar un reset automático con la implantación de una red RC:

Figura 37: RESET inicial



1.8 Temporizadores y Contadores

1.8.1 Generalidades

El microcontrolador AT89C51 posee 2 registros de 16 bits: el Timer 0 y el Timer 1; los mismos que se pueden configurar como *temporizadores* o como *contadores*. En añadidura, el AT89C52 tiene los mismos más uno: el Timer 2; configurables en idéntica situación.

> En el instante de configurarse como Timer, el registro es incrementado con cada ciclo de máquina. De ésta forma, el registro contabiliza los ciclos de máquina; considerando que un ciclo de máquina consiste de 12 períodos de oscilación. Por lo que el porcentaje del conteo es 1/12 de la frecuencia del oscilador (*reloj*).

> Actuando como un *contador*, es incrementado en respuesta a la transición de un "alto" a un "bajo"; por la señal que se aplique al pin correspondiente T0 – T1 – T2. Esta entrada es muestreada durante el S5P2 de cada ciclo de máquina, al darse un "flanco descendente" el contador es incrementado. Apareciendo un nuevo valor en el registro durante S3P1 del ciclo siguiente, tomando como base el ciclo en que fue detectado. En virtud que se requiere de 2 ciclos de máquina para detectar el flanco, el porcentaje de conteo máximo es 1/24 de la frecuencia del oscilador.

Para la señal externa no existen restricciones en "duty cicle", pero deberá estar presente por lo menos un ciclo completo para asegurar el flanco antes de los cambios. Por otra parte, el T0 y T1 tienen 4 modos de operación; y, el T2, 3 modos de operación.

1.8.2 Situación de los Timer´s

Para tratarlos como *"contador o temporizador"* se debe seleccionar en el registro TMOD del SFR los bits C/T. Ahora, los Timer 0 y Timer 1 poseen 4 modos de operación, seleccionable por los bits M0 y M1; en el mismo TMOD. Los modos 0 – 1 – 2 son los idénticos para los elementos, pero el modo 3 es diferente.

Tabla 17: TMOD (Registro de modos de Timer/Counter)

MSB **LSB**

GATE	C / $\overline{T}$	M1	M0	GATE	C / $\overline{T}$	M1	M0
Timer 1				Timer 0			

GATE	Habilita la entrada externa: si GATE = 1, se habilita INT0 si TR0 = 1 (control por hardware). Si GATE = 0, se deshabilita INT0 y depende exclusivamente de TR0 (control por software).
C / T	Selector de timer o counter. Set = counter, Clear = timer
M0 y M1	Selectores de modo

Tabla 18: Modos de los Timer/Counter (s)

M1	M0	Modo	Modo de operación
0	0	0	Timer de 13 bits Timer/counter con THz de 8 bits y TLx como un preescalar de 5 bits
0	1	1	Timer de 16 bits Timer/counter con THz y TLx en cascada. No hay preescalares.
1	0	2	Timer/counter de 8 bits con autorecarga
1	1	3	Timer/counter (s) específicos

Tabla 19: Registros que se relacionan con los Timer/Counter (s)

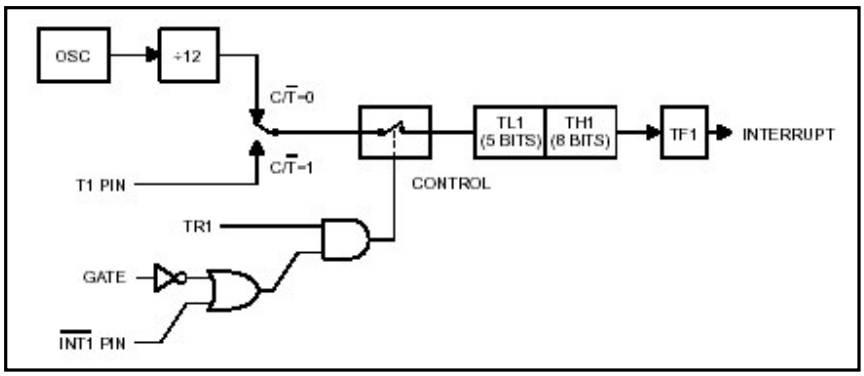
Timer SFR	Purpose	Address	Bit-Addressable
TCON	Control	88H	Yes
TMOD	Mode	89H	No
TL0	Timer 0 low-byte	8AH	No
TL1	Timer 1 low-byte	8BH	No
TH0	Timer 0 high-byte	8CH	No
TH1	Timer 1 high-byte	8DH	No
T2CON(1)	Timer 2 control	C8H	Yes
T2MOD(1)	Timer 2 Mode	C9H	No
RCAP2L(1)	Timer 2 low-byte capture	CAH	No
RCAP2H(1)	Timer 2 high-byte capture	CBH	No
TL2(1)	Timer 2 low-byte	CCH	No
TH2(1)	Timer 2 high byte	CDH	No

Note: 1. AT89C52 only.

1.8.2.1 T0 y T1 en Modo 0

Ambos Timer´ s son contadores de 8 bits con una división preescalar de 32 bits.

Figura 38: Timer/Counter 1.- Modo 0: contador de 13 bits



Aquí el Timer es configurado como un registro de 8 bits; cuando el contador da una vuelta pasando todos los "unos" a "ceros", el flag de interrupción del timer TF1 es seteado.

La entrada computada es habilitada por timer cuando TR1=1, y, GATE=0 o INT1=1. Seteando GATE=1 permite al timer ser controlado por la entrada externa INT1, facilitando un pulso para medición. TR1 es el bit de control en el SFR TCON (GATE), estando en TMOD.

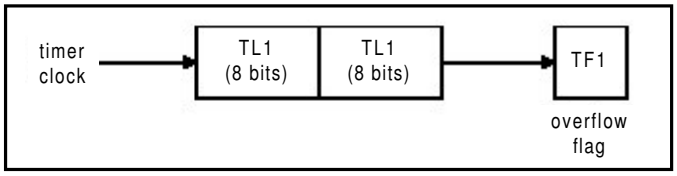
Conformando al registro se hallan los 8 bits de TH1, y los 5 bits más bajos de TL1; siendo los 3 bits más altos de TL1 indeterminados y por lo tanto no serán tomados en cuenta. El simple seteo del flag TR1 no borra los registros.

Lo descrito aquí es igual para el T0 y T1; a excepción del TR0, TF0 e INT0 que reemplazan las correspondientes señales del Timer1, en la gráfica anterior. Existen dos bits diferentes para GATE, los que son para el T1 y T0.

1.8.2.2 T0 y T1 en Modo 1

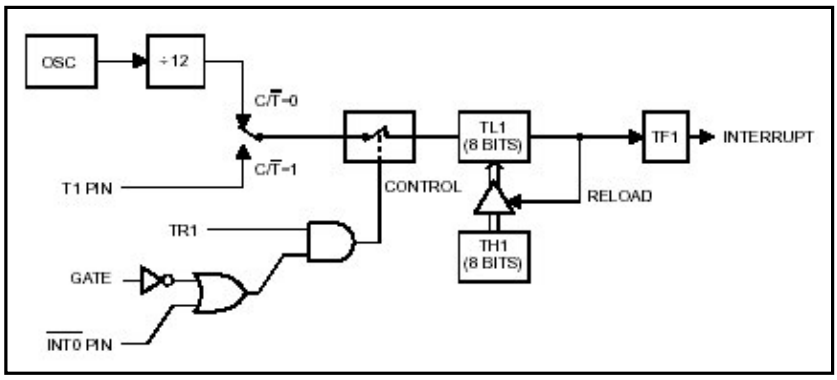
Básicamente es lo mismo del Modo 0, con la excepción que el registro del Timer funciona con 16 bits. El reloj se lo aplica por igual al TL1 / TH1; al recibir los pulsos el timer cuenta en ascenso comenzando en 0000H – 0001H – 0002H, etc. Al producirse un "overflow", de FFFFH a 0000H, se activa el flag de overflow, el timer continua contando. El flag de overflow es leído o escrito por software.

Figura 39: Timer/Counter 1.- Modo 1: Contador de 16bits



1.8.2.3 T0 y T1 en Modo 2

Figura 40: Timer/Counter 1.- Modo 2: Autorrecarga de 8 bits



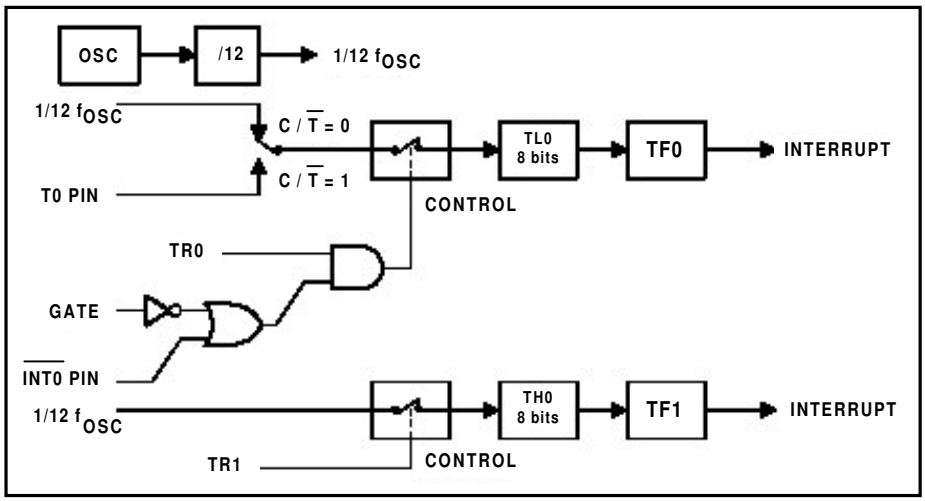
Este modo configura al timer como un contador de 8 bits (*TL1*) con recarga automática. Un overflow de TL1 no solo setea TF1, sino también recarga TL1 con el contenido de TH1, que es preseteado por software. La recarga permite a TH1 desengancharse.

1.8.2.4 Timers en Modo 3

El timer 1 simplemente mantiene su conteo. El efecto es el mismo que seteando TR1=0. El timer 0 en éste modo establece que TL0 y TH0 son dos registros totalmente separados. TL0 emplea los bits de control del timer 0 : C/T ; GATE ; TR0 ; INT0; y TF0.

TH0 se bloquea hacia el interior en función del timer (*contando los ciclos de máquina*) y por el empleo de TR1 y TF1 desde el timer 1. Así TH0 ahora controla la interrupción del timer 1.

Figura 41: Timer/Counter .- Modo 3: Dos contadores de 8 bits



1.8.2.5 Timer 2

Igual, el timer 2 es un contador / temporizador presente solo en el AT89C52; en relación a los ya expuestos es más poderoso. Son agregados 5 registros extras para funciones especiales, que acomodan al timer 2.

Y son:

- Los registros del timer TL2 – TH2,
- El registro de control del timer T2CON,

® Los registros de captura RCAP2L – RCAP2H.

Al igual que los timer´ s anteriores, se puede tratarlo como un *temporizador* o *contador de eventos*; de acuerdo a la configuración del bit C/T del T2CON. (*referirse a la tabla 13*) Este timer posee 3 modos de operación, tal como se observa en la tabla:

Tabla 20: Modos de operación del Timer 2

RCLK + TCLK	CP/$\overline{RL2}$	TR2	Mode
0	0	1	Autorecarga 16 bits
0	1	1	Captura 16 bits
1	X	1	Generador de baudios
X	X	0	(off)

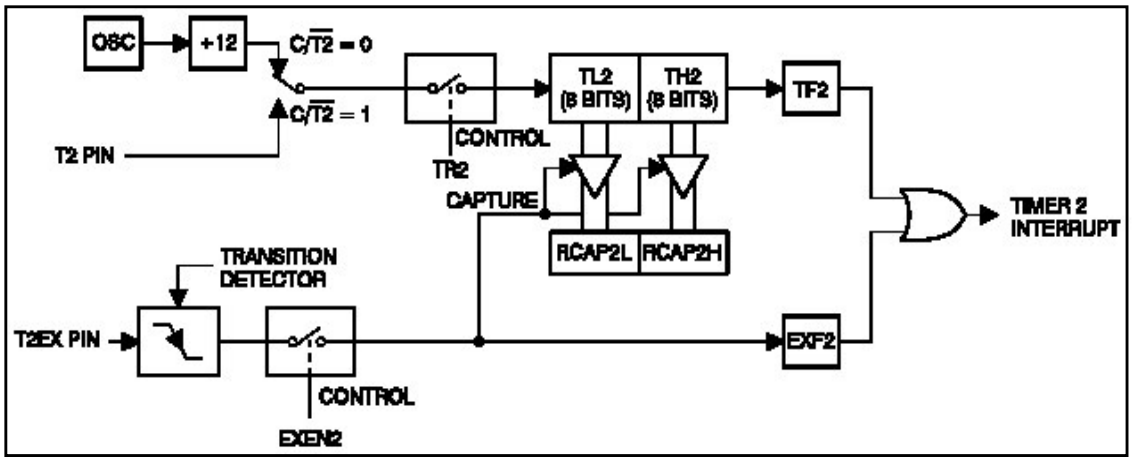
> En el *modo captura*, el bit EXEN2 en el T2CON selecta dos opciones:

EXEM = 0 ; es un temporizador a 16 bits, o, un contador en donde un overflow setea el bit TF2; bit de overflow; que puede ser empleado para iniciar una interrupción.

EXEM = 1 ; la operación es la misma; pero ahora, por una transición de señal en "flanco descendente" aplicada al pin T2EX causará que el valor de los registros TL2 y TH2 sean capturados en los registros RCAP2L y RCAP2H (respectivamente).

Esto se lo demuestra en la gráfica:

Figura 42: Timer/Counter 2.- Modo Captura

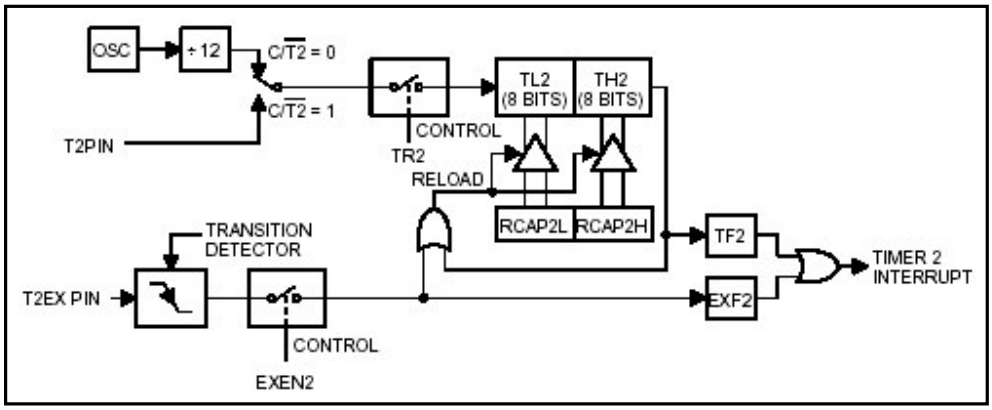


> En el *modo autorrecarga*, se maneja también el bit EXEM:

EXEM = 0 ; luego de que el timer 2 da un giro setea el TF2 , y, además recarga los registros RCAP2L y RCAP2H con valores de 16 bits; que anteriormente se cargaron por software.

EXEM = 1 ; el funcionamiento del timer es similar al anterior; pero por un flanco descendente en la entrada externa, también dispara la recarga a 16 bits y setea el EXF2.

Figura 43: Timer/Counter 2.- Modo Autorrecarga (DCEN = 0)



× El *modo generador de velocidad* para comunicación serial se explicará, por conveniencia, más adelante; en conjunto con lo relacionado al Puerto Serial.

1.9 Protocolos de Comunicación

1.9.1 Generalidades

El puerto serial del microcontrolador es del tipo *'full dúplex"* , siendo un mediador en la "transmisión y recepción" simultánea. En el momento de recibir el buffer que le apoya permite recibir un segundo byte, antes que el byte previo haya sido leído en el registro receptor.

- *Pero puede suceder que el primer byte no haya sido leído en el tiempo que el segundo byte se completa, por lo que uno de los dos bytes se pierde.*

Para acceder a los registros transmisor y receptor, se manipula SBUF en el SFR. Por la escritura en el SBUF se carga el registro transmisor; y leyendo el SBUF, se accede al registro receptor por separado. Los modos de operación del puerto serial son en modo 0, 1 , 2 y 3.

1.9.1.1 Entorno Multiprocesador

Los modos 2 y 3 tienen una previsión especial para la comunicación multiprocesadora. Aquí se recibe 9 bits seguidos por 1 bit de paro, para lo cual se trabaja con el registro SCON del SFR:

Tabla 21: SCON (Registro Control del Puerto Serial)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Especifica el modo del puerto serial.
SM1	SCON.6	Especifica el modo del puerto serial.
SM2	SCON.5	Habilita la comunicación multiprocesador en modos 2 y 3. Así, si SM2 = 1, luego RI no es activado si el noveno bit recibido es 0. En modo 1, RI no es activado si un bit válido de stop es recibido. En modo 0, SM2 será 0.
REN	SCON.4	Es seteado/limpiado por software para habilitar / deshabilitar la recepción.
TB8	SCON.3	Es el noveno que es transmitido en los modos 2 y 3. Seteado/limpiado por software.
RB8	CON.2	En los modos 2 y 3, es el noveno bit de dato que es recibido. En modo 1, si SM2=1, RB8 es el bit de stop que es recibido. En modo 0, RB8 no es usado.
TI	SCON.1	Flag de interrupción de transmisión. Seteado por hardware al final de la transmisión del 8avo bit en el modo 0, o al comienzo del bit de stop en los otros modos. Debe desactivarse por software.
RI	SCON.0	Flag de recepción. Se activa por hardware la finalizar la recepción del 8avo bit en el modo 0, o hacia la mitad del intervalo de tiempo del bit de stop en los otros modos (excepto ver SM2). Desactivar por software.

El noveno bit se coloca en el RB8, luego va el bit de paro. El puerto serial puede programarse de tal manera que cuando se reciba el bit de paro, la interrupción del puerto serial sea activada solo sí RB8 = 1; se logra con la habilitación del bit SM2. Mediante la explicación siguiente se podrá entender como emplear la interrupción serial en la comunicación multiprocesadora:

- En el instante que el procesador maestro transmitirá un bloque de datos a uno de los varios esclavos, primero envía un byte de dirección identificando al esclavo a usar. El byte de dirección se diferencia de un byte de datos por tener:

  - el byte de dirección, el noveno dígito = 1,
  - el byte de datos, el noveno dígito = 0,

  teniendo el SM2 = 1 ningún esclavo podrá ser interrumpido por un byte de datos. Entre tanto, el byte de dirección puede interrumpir a todos los esclavos, que a su vez lo analizan para identificar cual de ellos está siendo requerido. Entonces, el esclavo direccionado borra el bit SM2, e ignora el byte de datos.

El SM2 no tendrá efecto en el modo 0, pero se empleará en la verificación del bit de paro en el modo 1. En el modo 1 señalado como receptor, con SM2 = 1, la interrupción del receptor no se activará a menos que el bit de paro esté válido.

1.9.2 Modos del Puerto Serial

1.9.2.1 Modo 0

En éste modo, el dato serial ingresa y sale a través del pin RXD, en cambio TXD es la señal de clock. Aquí, 8 datos son recibidos y transmitidos comenzando por el LSB; los baudios son fijados a 1/12 de la frecuencia de reloj.

Figura 44: Modo 0 del Puerto Serial.- Transmisión / Recepción

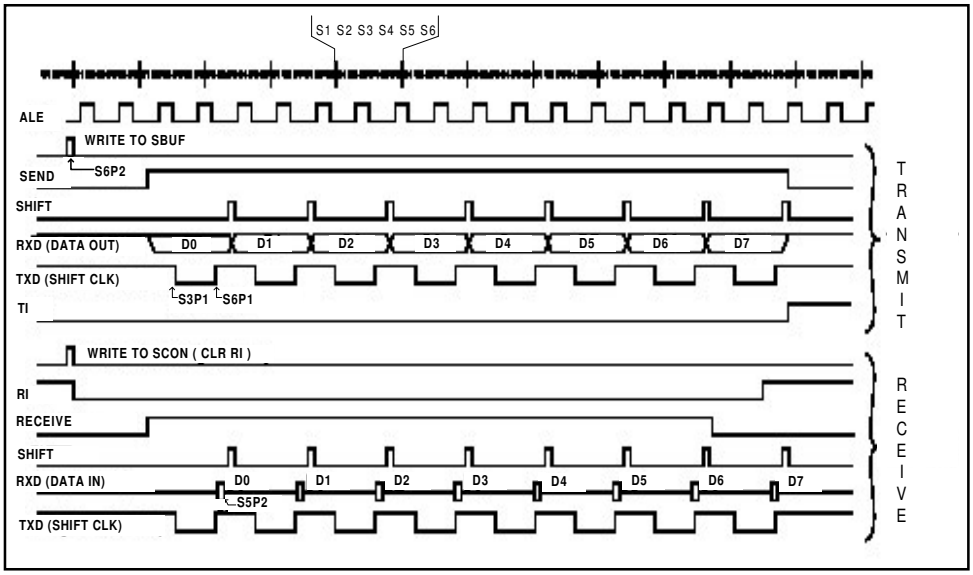
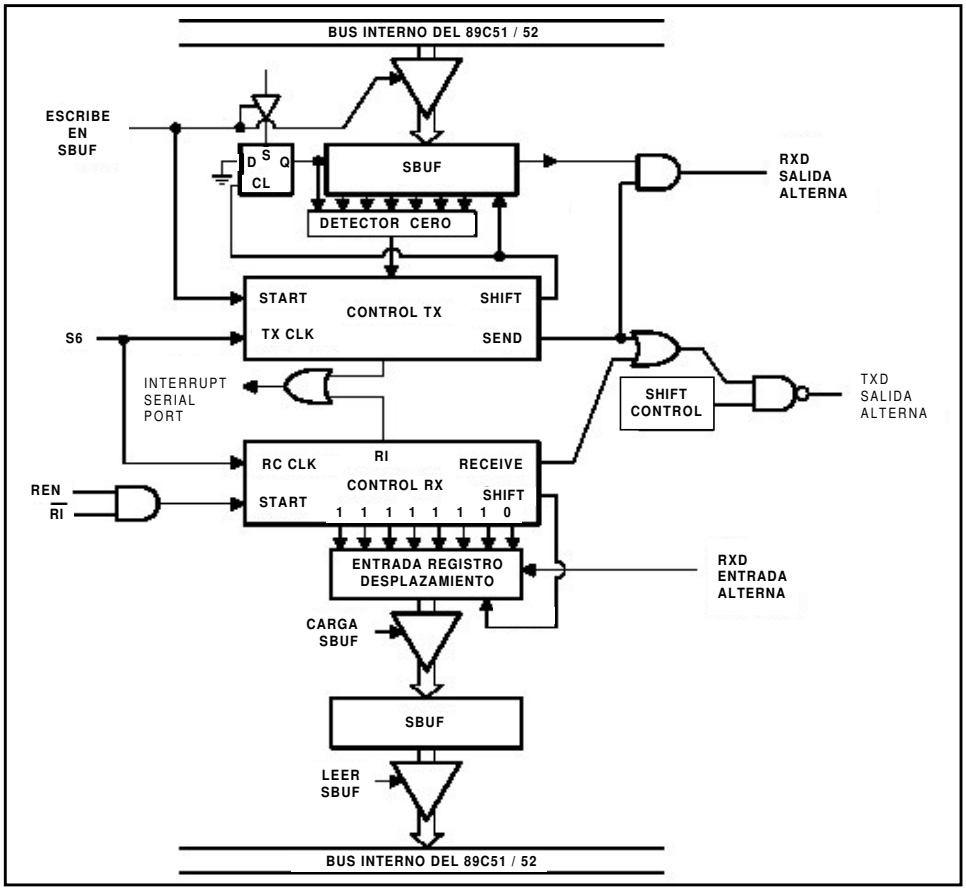


Figura 45: Modo 0 del Puerto Serial.- Transmisión / Recepción: hardware



La transmisión se inicializa al usar cualquier instrucción que maneje el SBUF como un registro de destino; la señal "Write to SBUF", en el estado S6P2, carga también un "1" en el interior de la novena posición. Esto en el registro de desplazamiento e indica el bloque de control TX para comenzar a transmitir.

El tiempo interno empleado desde la señal "Write to SBUF" y la activación de "SEND" comprende un ciclo completo de máquina.

* SEND transfiere el dato del registro de desplazamiento en forma alternada por el pin P3.0 (*RXD*), a la vez permite una salida de reloj simétrica por el pin P3.1 (*TXD*). Esta salida de reloj se mantiene a "nivel bajo" durante los estados S3 – S4 – S5 de cada ciclo de máquina; y en "nivel alto" en los estados S6 – S1 – S2. En el S6P2 de cada ciclo de máquina en donde SEND es activo, el contenido del registro de desplazamiento transmisor es corrido una posición a la derecha.

Por cada bit de dato desplazado al exterior por la derecha, por la izquierda se coloca un "bit 0". Cuando el MSB del byte de dato se halla en la posición de salida del registro desplazador, el "1" que se cargó inicialmente en la novena posición es el mismo que está a la izquierda del MSB. Y las demás posiciones a la izquierda de éste contienen "ceros.

Esta condición previene al bloque de control TX para que efectuase un último desplazamiento; luego se desactiva SEND, y, setea la señal de interrupción TI. Ambas situaciones pasan en la fase S1P1, del décimo ciclo de máquina después de "Write to SBUF".

Entre tanto, la recepción es inicializada por la condición “REN= 1” y “REN=0”. En el S6P2 del siguiente ciclo de máquina, la unidad de control RX escribe los bits 11111110 en el registro de desplazamiento receptor; y, activa el RECEIVE en la siguiente fase del reloj.

< RECEIVE habilita el reloj alternado en el pin P3.1 (*TXD*); la señal de reloj efectúa la transición de S3P1 y S6P1 de cada ciclo de máquina. En el S6P2 de cada ciclo de máquina en que RECEIVE es activo, el contenido del registro de desplazamiento receptor es corrido una posición a la izquierda. El valor que ingresa por la derecha es aquel que fue muestreado en el pin P3.0, en la fase S5P2 del mismo ciclo de máquina.

Ahora, los bits de datos ingresan por la derecha; y, los “unos” son desplazados por la izquierda hacia el exterior. Cuando el “0” que se cargó inicialmente se halle en el extremo izquierdo del registro desplazador, sería la indicación para que le bloque de control RX haga un último desplazamiento y cargue el SBUF. En el S1P1 del décimo ciclo de máquina, después de escribir en el SCON (*borrado de RI*), RECEIVE es borrado y el flag de interrupción RI es seteado.

1.9.2.2 Modo 1

En ésta opción se transmite (*a través de TXD*), o se recepta (*a través de RXD*); subdivididos en:

⊂ un bit de arranque “0”,

- 8 bits de datos, los primeros LSB,
- un bit de par “1”.

En la recepción el bit de paro se introduce en el RB8, del SCON. En el microcontrolador AT89C51 la velocidad en baudios se la determina por el overflow del timer 1; y en AT89C52, dicha velocidad es determinada por el timer 1 o timer 2 o ambos, tomando como base los overflows de cada uno. En este caso, un timer es transmisor y el otro receptor.

Figura 46: Modo 1 del Puerto Serial.- Temporización de tareas

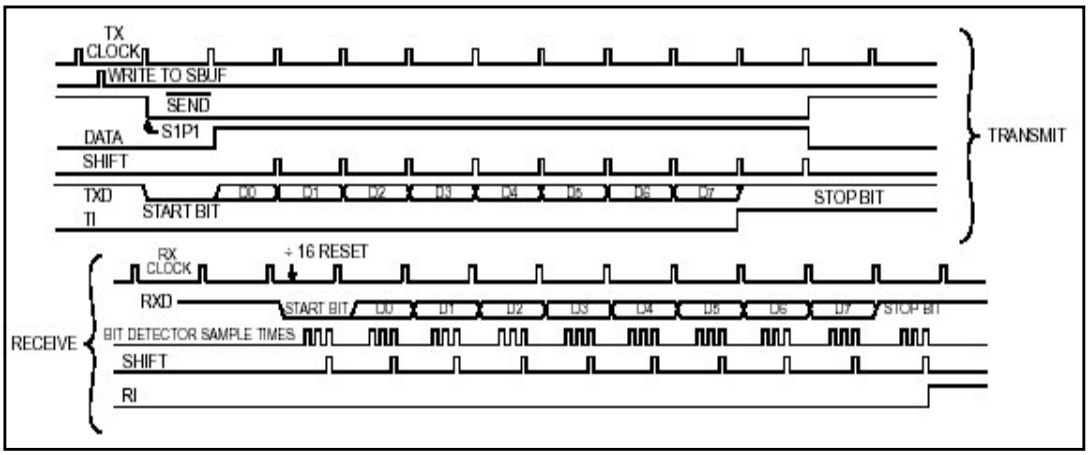
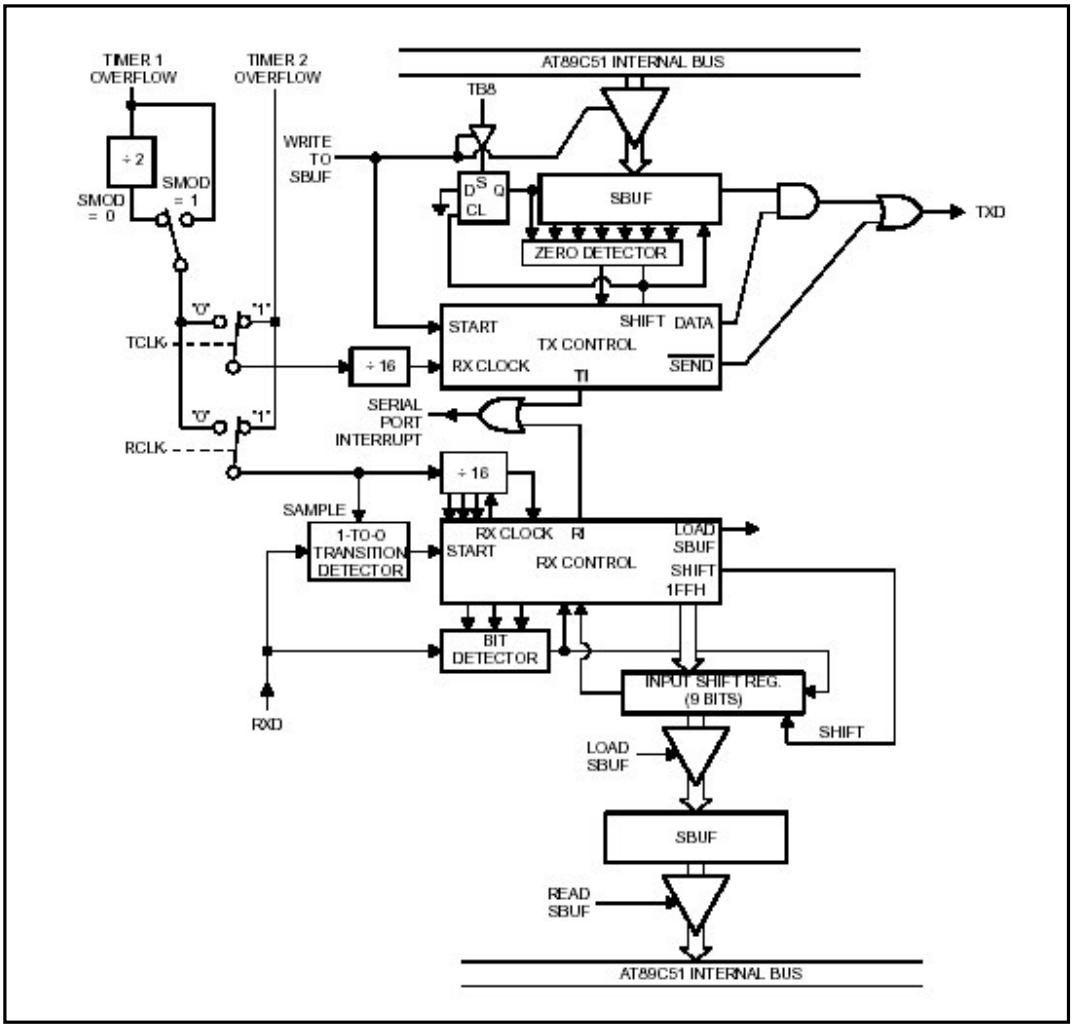


Figura 47: Modo 1 del puerto Serial.- hardware



La transmisión es inicializada por cualquier instrucción que maneje el SBUF, como un registro de destino. La señal "write to SBUF" carga también un "1" en la posición del noveno bit, del registro de desplazamiento transmisor, y alerta a la unidad de control TX que una petición de transmisión se ha hecho. A transmitir empieza en realidad cuando comienza el S1P1 del siguiente ciclo de máquina que resulta después del ciclo de sobrepasamiento del contador del timer dividido para 16; de ésta manera, el tiempo de bit es sincronizado por el contador dividido para 16 y no por la señal "write to SBUF".

Pero, la transmisión se comienza cuando SEND es activada, que coloca el bit de arranque en TXD; un tiempo más tarde (*el tiempo que se demora un bit*) la señal DATA es activada habilitando la salida del bit, del registro de desplazamiento transmisor por medio de TXD. El primer pulso de desplazamiento pasa "un tiempo de bit" más tarde de la salida del primer bit de dato.

Como los bits de datos salen por la derecha, por la izquierda se colocan "ceros". Cuando el MSB del byte de dato está en la posición de salida del registro desplazador, el "1" que se cargó inicialmente, en la novena posición, es aquel que está a la izquierda del MSB y todas las locaciones a la izquierda de éste contienen "ceros". Ahora, con dicha situación se indica a la unidad de control TX hacer un último desplazamiento, luego desactivar SEND y setear el flag de interrupción TI. Todo ésto ocurre en el décimo sobrepasamiento del divisor por 16, después de la señal "write to SBUF".

En cambio para empezar a recibir se debe colocar un "flanco descendente" en el pin RXD. Por ésto, se lo hace un muestreo en el rango de 16 veces que establece el rango de baudios.

- Al detectarse una transición, el contador divisor por 16 es inmediatamente reseteado y, el dato 1FFH es escrito en el interior del registro desplazador. Poniendo a cero el contador divisor por 16 sincroniza los tiempos de sobrepasamientos con los tiempos de bits entrantes.

Los 16 estados del contador dividen en 16 intervalos cada "tiempo de bit"; en el 7mo, 8avo y 9eno estado del contador de cada tiempo de bit, el detector de bit muestrea el valor de RXD. El valor que es aceptado es aquel que fue visto en los 2 o 3 últimos muestreos.

Se lo hace para descartar ruido; en éste orden, si el valor aceptado durante el primer tiempo de bit no es "0", el circuito receptor es reseteado y la unidad continua observando por otra transición. Si el bit de arranque es válido, éste es desplazado al interior del registro desplazador de ingreso; y la acogida del resto de bits de la estructura se procede así.

> Como el ingreso de los datos es desde la derecha, los "uno s" son sacados por la izquierda. Cuando el bit de arranque llega a la última posición izquierda, en el registro desplazador, se avisa al bloque de control para que haga un último desplazamiento, cargando SBUF y RB8, y seteando RI.

Para hacer lo último descrito, se da *"sí y solamente sí"* se reúne las condiciones siguientes en el momento en que el pulso final se genera:

- RI = 0,
- Si SM2 = 0, o el bit recibido de paro es 1.

Si algunas de éstas dos situaciones no se reúnen, la estructura recibida es perdida irremediablemente. Pero si ambas situaciones se reúnen, el bit de paro se coloca en RB8 y los 8 bits de datos van al SBUF, y RI es activado. En éste tiempo, si las condiciones que se

indicó se cumpliesen o no, la unidad continua observando por una transición (*flanco descendente*) en el pin RXD.

1.9.2.3 Modo 2 y 3

Por el pin TXD se transmiten 11 bits, o se los recepta por RXD:

- un bit de arranque ‘0”,
- 8 bits de datos, primero el LSB,
- un noveno bit programable,
- un bit de paro.

Para transmitir, el noveno bit (TB8) se le puede asignar un valor de 0 o 1; y para recibir, va en el RB8 del SCON. La velocidad en baudios se programa a 1/32 o 1/64 de la frecuencia de reloj en modo 2. En el modo 3, tendría una variación generada de velocidad por el timer 1 o 2; dependiendo del estado de TCLK y RCLK. La parte de la recepción es exactamente la misma que en el modo 1, pero la parte de la transmisión difiere del modo 1 solo en el noveno bit del registro desplazador transmisor.

Figura 48: Modo 2 del Puerto Serial.- Temporización de tareas

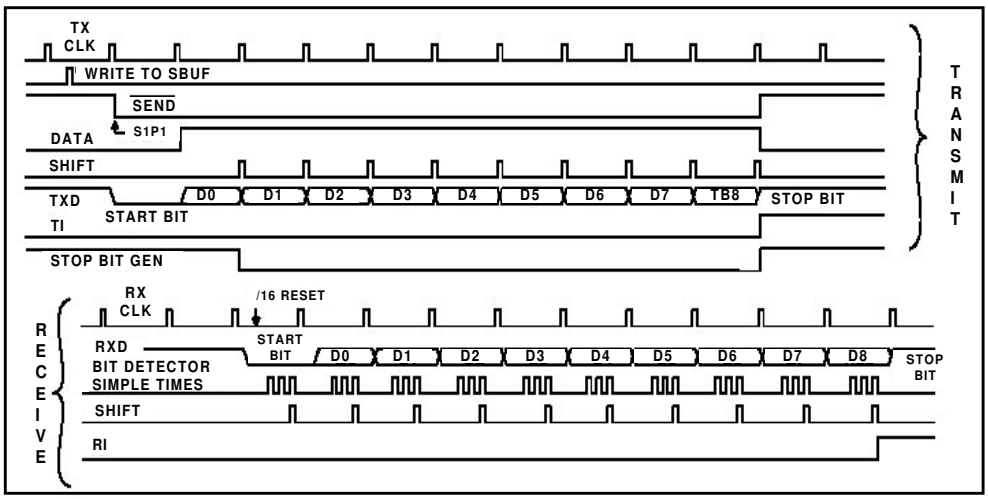


Figura 49: Modo 2 del Puerto Serial.- hardware

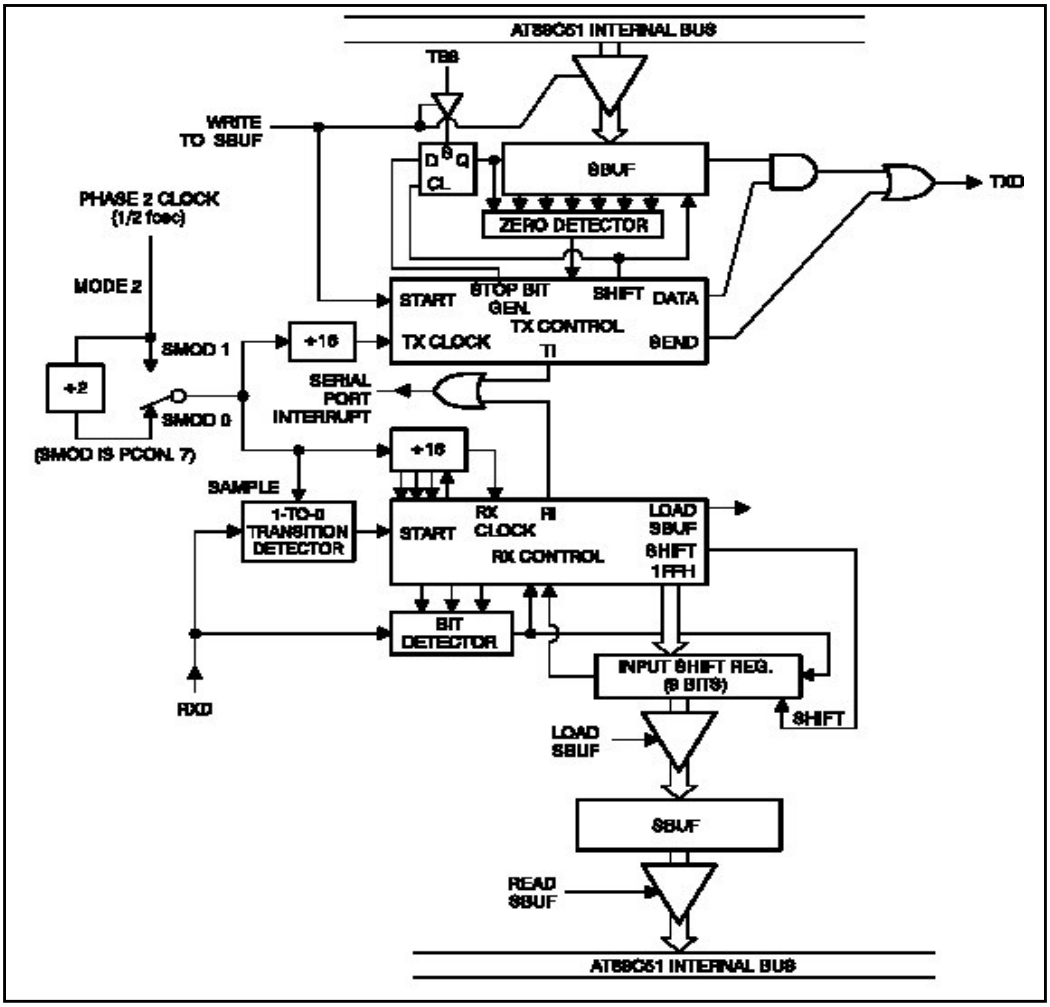


Figura 50: Modo 3 del Puerto Serial.- Temporización de Tareas

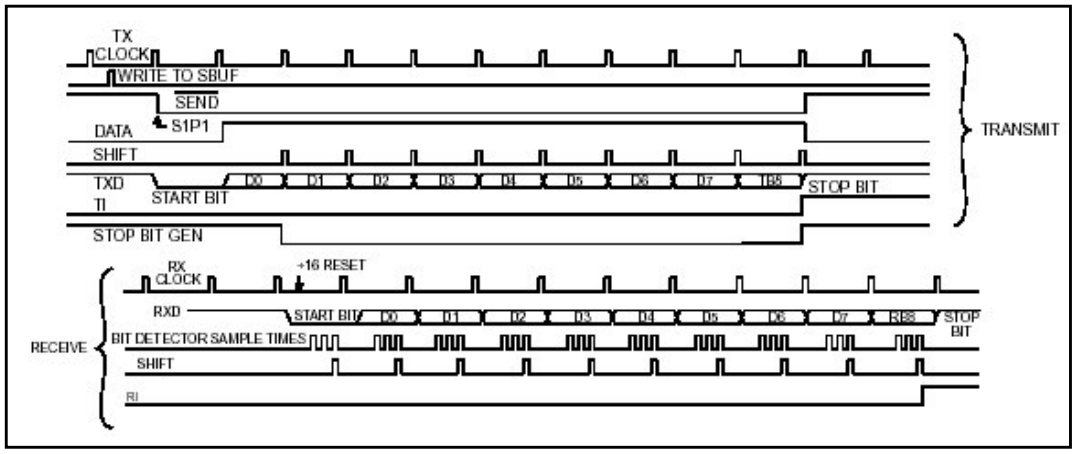
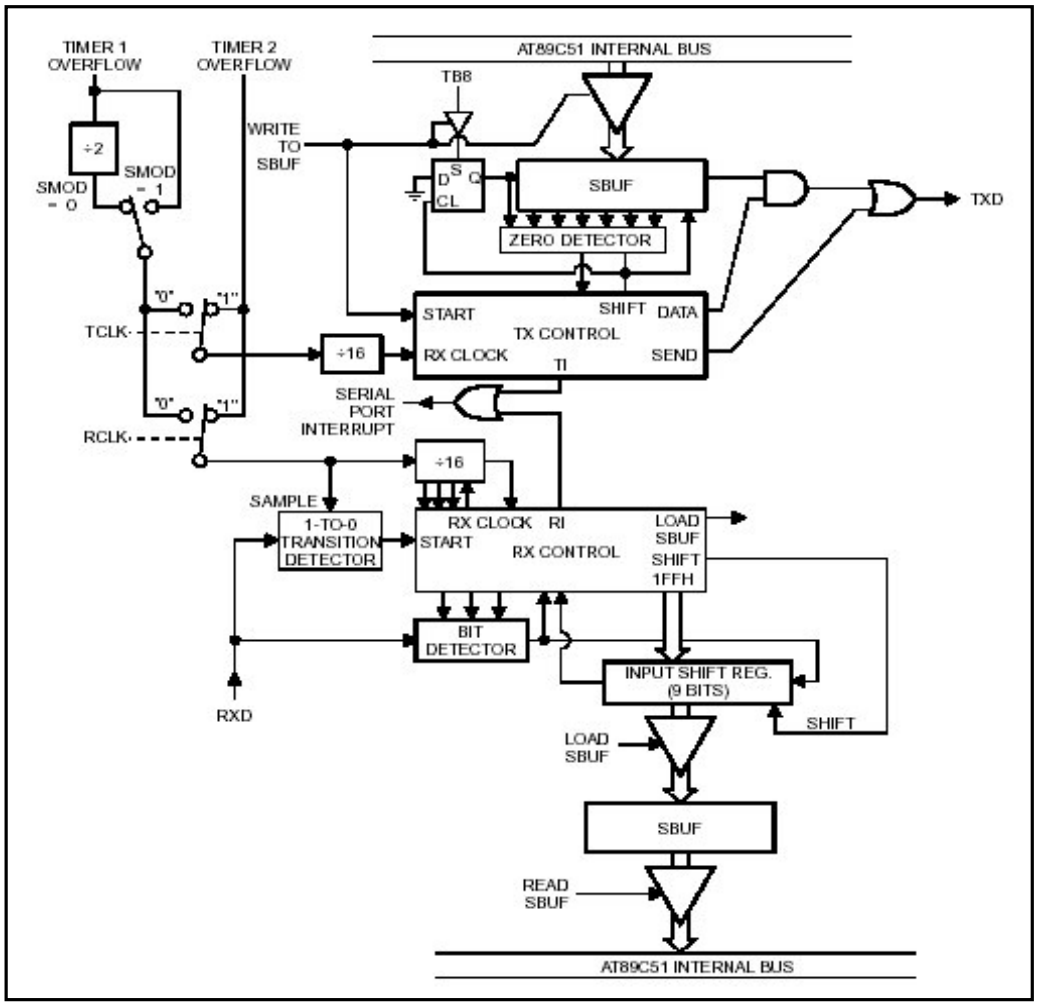


Figura 51: Modo 3 del Puerto Serial.- Hardware; RCLK,TCLK y Timer 2 solo están en el 89C52



La transmisión se inicializa por cualquier instrucción que emplee el registro SBUF como destino. La señal "write to SBUF" carga también el TB8 en el interior de la posición del noveno bit del registro de desplazamiento transmisor y, señala a la unidad de control TX que una transmisión es requerida.

La *transmisión* inicializa en el S1P1 de ciclo de máquina siguiente al sobrepasamiento del contador divisor por 16; así, los tiempos de bit son sincronizados por contador, y no por la señal "write to SBUF". Comenzando al activar SEND que coloca el bit de arranque en el TXD. Un tiempo de bit más tarde, la señal DATA es activada habilitando la salida del bit del registro de desplazamiento TXD. El primer pulso desplazador ocurre un tiempo de bit después de que ha salido el primer bit de datos.

En ese mismo momento se envía un bit de paro que se coloca en la novena posición de bit del registro desplazador; después de eso, solo "ceros" son almacenados.

Así, como los bits de datos salen por la derecha, los "ceros" son colocados por la izquierda. Cuando el bit TB8 está en la posición de salida del registro, entonces el bit de paro está justo a la izquierda del TB8 y, las demás posiciones a la izquierda de él contienen ceros. Esta condición alerta a la unidad de control TX para hacer un último desplazamiento, luego desactiva a SEND y, setea TI.

Esto se da el 11avo sobrepasamiento del contador divisor por 16, después de escribir "write to SBUF".

La *recepción* es iniciada por un "flanco descendente" en RXD; por tal motivo, RXD es muestreado a una razón de 16 veces según lo establecido por el rango de baudios. Al darse la transición, el contador es inmediatamente reseteado y el dato 1FFH es escrito en el ingreso al registro desplazador.

En el 7mo, 8avo y 9eno estado del contador con un tiempo de bit, el detector de bit muestrea al RXD; el valor que se acepte será el observado por los 2 o 3 últimos muestreos.

Si el valor aceptado durante el primer tiempo de bit no es 0, los circuitos receptores son reseteados y la unidad continua observando otra transición. Si el bit que se prevee para el arranque es válido, éste es colocado en el interior del registro de desplazamiento, así mismo con el resto de la estructura de ese byte.

Como los datos ingresan por la derecha, por la izquierda se desplazan los "unos". Al llegar la posición izquierda el bit de arranque, ésto avisa al bloque de control para que haga un último desplazamiento y cargue al RB8 y SBUF, y setee al RI.

La señal para cargar el SBUF y RB8 y setear el RI son generadas *sí y solamente sí* se dan las siguientes situaciones en el momento del pulso final de desplazamiento:

- ⊗ RI = 0,
- ⊗ Si SM2 = 0, o el 9no dato recibido es 1.

Si alguna de éstas condiciones no se da, la estructura recibida se pierde irremediablemente, y RI no es seteado. Pero si ambas condiciones se reúnen, el noveno bit de dato va al RB8, y el primer byte al SBUF.

Un tiempo de bit más tarde, se cumplan o no las condiciones especificadas, la unidad vuelve a muestrear el flanco descendente en RXD.

1.9.3 Velocidad de Comunicación

Para fijar la velocidad en el *modo 0*, se procede:

$$\text{Rango de Baudios} = \frac{\text{frecuencia del oscilador del micro}}{12}$$

Para el *modo 2* se debe considerar que el encendido del "bit SMOD" en el registro PCON del SFR:

Si SMOD = 0, el porcentaje de la velocidad es 1/64 de la frecuencia de reloj,

Si SMOD = 1, el porcentaje de la velocidad es 1/32 de la frecuencia de reloj.

Pudiendo aplicar la ecuación:

$$\text{Rango de Baudios} = \frac{2^{\text{SMOD}}}{12} * \text{frecuencia del oscilador del micro}$$

En los microcontroladores AT89C51 se emplea el sobrepasamiento del timer 1 para determinar la velocidad en los modos 1 y 3; mientras que el AT89C52, usa para lo mismo, los timers 1 y/o 2.

1.9.3.1 Empleo del T1 para generar velocidad

Empleado así el timer 1, la velocidad en los modos 1 y 3 son determinados por el rango de overflow y el valor del SMOD, >*la interrupción del timer 1 deberá ser deshabilitada para ésta aplicación* < , siendo:

$$\text{Rango de Baudios en los modos 1 y 3} = \frac{2^{SMOD}}{32} * \text{rango de sobreflujo del timer 1}$$

Al timer se lo puede también configurar como *contador* o *temporizador*, en cualquiera de sus tres modos; para las aplicaciones más típicas se lo hace como temporizador, en el modo auto-recarga. Siendo así, el rango de velocidad está dado por:

$$\text{Rango de Baudios en los modos 1 y 3} = \frac{2^{SMOD}}{32} * \frac{\text{frecuencia del oscilador}}{12 \times [\,256 - (TH1)]}$$

Para lograr rangos muy bajos, permitiendo a la interrupción del timer 1 habilitarse, y configurándolo como temporizador de 16 bits. Entonces, la interrupción es usada como recarga a 16 bits por software.

En la siguiente tabla se visualizan algunas velocidades obtenidas a partir del timer 1:

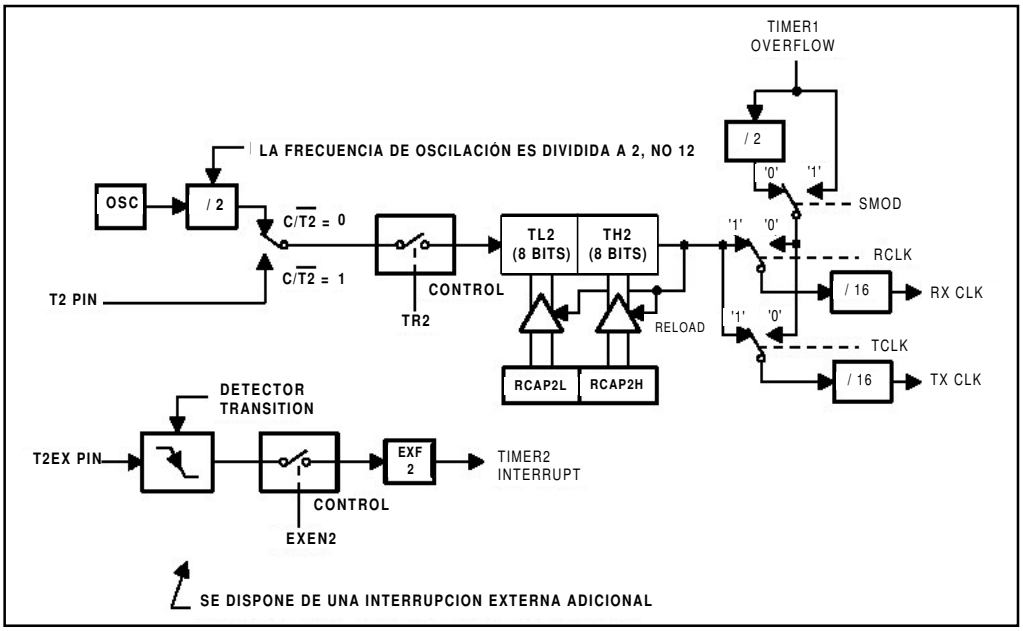
Tabla 22: Rangos de baudios comúnmente generados por el timer 1

Rango de baudios	frecuencia fosc	SMOD	Timer 1		
			C/$\overline{T}$	Modo	Valor recarga
Mode 0 Max: 1 MHz	12 MHz	X	X	X	X
Mode 2 Max: 375K	12 MHz	1	X	X	X
Modes 1, 3: 62.5K	12 MHz	1	0	2	FFH
19.2K	11.059 MHz	1	0	2	FDH
9.6K	11.059 MHz	0	0	2	FDH
4.8K	11.059 MHz	0	0	2	FAH
2.4K	11.059 MHz	0	0	2	F4H
1.2K	11.059 MHz	0	0	2	E8H
137.5	11.986 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

1.9.3.2 Empleo del T2 para generar velocidad

Encontrado en AT89C52, se logra configurarlo con el seteo de los bits TCLK y/o RCLK en el T2CON, y así selectarlo como generador de baudios. Dado ésto, el rango de baudios de transmisión y recepción pueden ser diferentes.

Figura 52: Timer 2 como generador de baudios



El modo empleado para la generación de baudios es similar al modo autorrecarga, en el cual un sobrepasamiento en el TH2 recarga los registros del timer2 (RCAP2L – RCAP2H) con un valor de 16 bits, los cuales fueron preseteados por software.

En éste caso el rango de baudios en modo 1 y 3 es acorde a la ecuación:

$$\text{Rango de Baudios en modo 1 y 3} = \frac{\text{rango de sobreflujo del timer 2}}{16}$$

Puede ser configurado como contador o temporizador, en las aplicaciones más típicas se lo configura como temporizador con C/T2 = 0. Normalmente como temporizador incrementa en cada ciclo de máquina a razón de 1/12 de la frecuencia del oscilador; pero

como generador de baudios incrementa a razón de ½ de la frecuencia del oscilador. Aplicándose:

$$\text{Rango de Baudios en modo 1 y 3} = \frac{\text{frecuencia del oscilador}}{32 * [\,65536 - (\,RCAP2H\,,\,RCAP2L\,)]}$$

En donde RCAP2H y RCAP2L corresponde al contenido de ellos, asignados en valor absoluto.

El gráfico anterior es válido solamente sí *RCLK + TCLK = 1* en el TCON; es decir, que puede actuar como generador de baudios solo bajo ésta condición. Un sobrepasamiento de TH2 no setea el flag TF2 ni genera una interrupción; es por ésto que la interrupción del timer 2 no tiene que deshabilitarse. Además, si EXEN2 se activa, un flanco descendente en T2EX setea a EXF2 pero no produce una recarga desde (RCAP2H – RCAP2L) a (TH2, TL2).

Pero si el timer 2 se usa como generador de baudios, el flag T2EX puede usarse como interrupción externa extra.

Al operar el timer 2 como generador se aconseja no leer ni escribir en los registros TH2 y TL2; con éstas condiciones, el T2 es incrementado en cada tiempo de estado, los resultados de leer o escribir podrían dar una falsa exactitud. Los registros RCAP podrían

ser leídos pero no escritos, por que la escritura solaparía una recarga y causa errores de escritura y/o recarga.

En tal caso, si se desea hacer éstas actividades primero desactívese el TR 2 antes de acceder al timer 2 o a los registros RCAP.

2 Software de las Series AT89

2.1 Program Status Word

Los microcontroladores de la familia ATMEL utilizan el mismo software. Las instrucciones permiten el trato de un bit específico de un byte, o tratar al byte como tal; realizando operaciones como: funciones lógicas, operaciones BCD, etc.

Para conocer el estado de la CPU se tiene el "PSW" (*program status word*), encontrándose en el SFR, constituido así:

Tabla 23: PSW (Palabra de Estado del Programa)

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Flag de acarreo
AC	PSW.6	Flag de acarreo auxiliar
F0	PSW.5	Flag de propósito general para el usuario
RS1	PSW.4	Bit 1 selector de banco de registro
RS0	PSW.3	Bit 0 selector de banco de registro
OV	PSW.2	Flag de overflag
—	PSW.1	Flag no definido
P	PSW.0	Flag de paridad. Seteado/ limpiado por hardware en cada ciclo de instrución para indicar un numero para o impar de 1´s presentes en el acumulador.

Nota: Los valores correspondientes de RS0 y RS1 seleccionan el correspondiente banco de registros:

RS1	RS0	Banco de Registros	Dirección
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

- El bit de *CARRY* se adiciona para servir como un bit de acarreo en las operaciones aritméticas, pero también sirve como acumulador para un número en operaciones booleanas.

- Los bits *RS0 y RS1* seleccionan uno de los 4 bancos de registros. A los cuales se refieren algunas instrucciones, como R7 a R0.

- El *bit de paridad P* refleja el numero de "unos" en el acumulador:

  P = 0..... acumulador con un numero impar de "1",

  P = 1..... acumulador con numero par de "1".

- Dos bits son para propósitos generales.

2.2 Set de Instrucciones

A continuación se presenta el conjunto de las instrucciones que se maneja:

Tabla 24: Instrucciones que afectan al seteo de los flags

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Tabla 25: Set de instrucciones y los modos de direccionamiento

Rn	Registros R7 a R0 del banco de registro corriente selectado
direct	Dirección de posiciones de datos interno de 8 bits. Esta será una posición de la RAM interna (0-127) o un SFR(por ejemplo, puerto I/O, registro de control,registro de estado, etc (128-255)).
@Ri	Posición de dato de la RAM interna de 8 bits (0-255) direccionada indirectamente a través de R1 o R0.
#data	Constante de 8bits incluida en la instrucción.
#data16	Constante de 16bits incluida en la instrucción.
addr16	Dirección de 16 bits. Usada por LCALL y LJMP. El salto puede ser dentro de los 64Kbyteen el espacio de memoria de programa.
addr11	Dirección de 11 bits. Usada por ACALL y aJMP. El salto puede ser dentro de la página de 2Kbytes dela memoria de programa a partir del primer byte de la siguiente instrucción.
rel	Salto relativo en formato de 8 bits en complemento a 2. Utilizado por SJMP y todos los saltos incondicionales. El rango del salto (8 bits) está comprendido entre -128 a +127 bytes a partir del primer byte de la siguiente instrucción.
bit	Bit direccionado directamente en la RAM interna de dato, o SFR

Tabla 26: Sumario de las Instrucciones

	0	1	2	3	4	5	6	7
0	NOP	JBC bit,rel [3B, 2C]	JB bit, rel [3B, 2C]	JNB bit, rel [3B, 2C]	JC rel [2B, 2C]	JNC rel [2B, 2C]	JZ rel [2B, 2C]	JNZ rel [2B, 2C]
1	AJMP (P0) [2B, 2C]	ACALL (P0) [2B, 2C]	AJMP (P1) [2B, 2C]	ACALL (P1) [2B, 2C]	AJMP (P2) [2B, 2C]	ACALL (P2) [2B, 2C]	AJMP (P3) [2B, 2C]	ACALL (P3) [2B, 2C]
2	LJMP addr16 [3B, 2C]	LCALL addr16 [3B, 2C]	RET [2C]	RETI [2C]	ORL dir, A [2B]	ANL dir, A [2B]	XRL dir, a [2B]	ORL C, bit [2B, 2C]
3	RR A	RRC A	RL A	RLC A	ORL dir, #data [3B, 2C]	ANL dir, #data [3B, 2C]	XRL dir, #data [3B, 2C]	JMP @A + DPTR [2C]
4	INC A	DEC A	ADD A, #data [2B]	ADDC A, #data [2B]	ORL A, #data [2B]	ANL A, #data [2B]	XRL A, #data [2B]	MOV A, #data [2B]
5	INC dir [2B]	DEC dir [2B]	ADD A, dir [2B]	ADDC A, dir [2B]	ORL A, dir [2B]	ANL A, dir [2B]	XRL A, dir [2B]	MOV dir, #data [3B, 2C]
6	INC @R0	DEC @R0	ADD A, @R0	ADDC A, @R0	ORL A, @R0	ANL A, @R0	XRL A, @R0	MOV @R0, @data [2B]
7	INC @R1	DEC @R1	ADD A, @R1	ADDC A, @R1	ORL A, @R1	ANL A, @R1	XRL A, @R1	MOV @R1, #data [2B]
8	INC R0	DEC R0	ADD A, R0	ADDC A, R0	ORL A, R0	ANL A, R0	XRL A, R0	MOV R0, #data [2B]
9	INC R1	DEC R1	ADD A, R1	ADDC A, R1	ORL A, R1	ANL A, R1	XRL A, R1	MOV R1, #data [2B]
A	INC R2	DEC R2	ADD A, R2	ADDC A, R2	ORL A, R2	ANL A, R2	XRL A, R2	MOV R2, #data [2B]
B	INC R3	DEC R3	ADD A, R3	ADDC A, R3	ORL A, R3	ANL A, R3	XRL A, R3	MOV R3, #data [2B]
C	INC R4	DEC R4	ADD A, R4	ADDC A, R4	ORL A, R4	ANL A, R4	XRL A, R4	MOV R4, #data [2B]
D	INC R5	DEC R5	ADD A, R5	ADDC A, R5	ORL A, R5	ANL A, R5	XRL A, R5	MOV R5, #data [2B]
E	INC R6	DEC R6	ADD A, R6	ADDC A, R6	ORL A, R6	ANL A, R6	XRL A, R6	MOV R6, #data [2B]
F	INC R7	DEC R7	ADD A, R7	ADDC A, R7	ORL A, R7	ANL A, R7	XRL A, R7	MOV R7, #data [2B]

Note: Key: [2B] = 2 Byte, [3B] = 3 Byte, [2C] = 2 Cycle, [4C] = 4 Cycle, Blank = 1 byte/1 cycle

Tabla 26: Continuación...

	8	9	A	B	C	D	E	F
0	SJMP REL [2B, 2C]	MOV DPTR,# data 16 [3B, 2C]	ORL C, /bit [2B, 2C]	ANL C, /bit [2B, 2C]	PUSH dir [2B, 2C]	POP dir [2B, 2C]	MOVX A, @DPTR [2C]	MOVX @DPTR, A [2C]
1	AJMP (P4) [2B, 2C]	ACALL (P4) [2B, 2C]	AJMP (P5) [2B, 2C]	ACALL (P5) [2B, 2C]	AJMP (P6) [2B, 2C]	ACALL (P6) [2B, 2C]	AJMP (P7) [2B, 2C]	ACALL (P7) [2B, 2C]
2	ANL C, bit [2B, 2C]	MOV bit, C [2B, 2C]	MOV C, bit [2B]	CPL bit [2B]	CLR bit [2B]	SETB bit [2B]	MOVX A, @R0 [2C]	MOVX wR0, A [2C]
3	MOVC A, @A + PC [2C]	MOVC A, @A + DPTR [2C]	INC DPTR [2C]	CPL C	CLR C	SETB C	MOVX A, @RI [2C]	MOVX @RI, A [2C]
4	DIV AB [2B, 4C]	SUBB A, #data [2B]	MUL AB [4C]	CJNE A, #data, rel [3B, 2C]	SWAP A	DA A	CLR A	CPL A
5	MOV dir, dir [3B, 2C]	SUBB A, dir [2B]		CJNE A, dir, rel [3B, 2C]	XCH A, dir [2B]	DJNZ dir, rel [3B, 2C]	MOV A, dir [2B]	MOV dir, A [2B]
6	MOV dir, @R0 [2B, 2C]	SUBB A, @R0	MOV @R0, dir [2B, 2C]	CJNE @R0, #data, rel [3B, 2C]	XCH A, @R0	XCHD A, @R0	MOV A, @R0	MOV @R0, A
7	MOV dir, @R1 [2B, 2C]	SUBB A, @R1	MOV @R1, dir [2B, 2C]	CJNE @R1, #data, rel [3B, 2C]	XCH A, @R1	XCHD A, @R1	MOV A, @R1	MOV @R1, A
8	MOV dir, R0 [2B, 2C]	SUBB A, R0	MOV R0, dir [2B, 2C]	CJNE R0, #data, rel [3B, 2C]	XCH A, R0	DJNZ R0, rel [2B, 2C]	MOV A, R0	MOV R0, A
9	MOV dir, R1 [2B, 2C]	SUBB A, R1	MOV R1, dir [2B, 2C]	CJNE R1, #data, rel [3B, 2C]	XCH A, R1	DJNZ R1, rel [2B, 2C]	MOV A, R1	MOV R1, A
A	MOV dir, R2 [2B, 2C]	SUBB A, R2	MOV R2, dir [2B, 2C]	CJNE R2, #data, rel [3B, 2C]	XCH A, R2	DJNZ R2, rel [2B, 2C]	MOV A, R2	MOV R2, A
B	MOV dir, R3 [2B, 2C]	SUBB A, R3	MOV R3, dir [2B, 2C]	CJNE R3, #data, rel [3B, 2C]	XCH A, R3	DJNZ R3, rel [2B, 2C]	MOV A, R3	MOV R3, A
C	MOV dir, R4 [2B, 2C]	SUBB A, R4	MOV R4, dir [2B, 2C]	CJNE R4, #data, rel [3B, 2C]	XCH A, R4	DJNZ R4, rel [2B, 2C]	MOV A, R4	MOV R4, A
D	MOV dir, R5 [2B, 2C]	SUBB A, R5	MOV R5, dir [2B, 2C]	CJNE R5, #data, rel [3B, 2C]	XCH A, R5	DJNZ R5, rel [2B, 2C]	MOV A, R5	MOV R5, A
E	MOV dir, R6 [2B, 2C]	SUBB A, R6	MOV R6, dir [2B, 2C]	CJNE R6, #data, rel [3B, 2C]	XCH A, R6	DJNZ R6, rel [2B, 2C]	MOV A, R6	MOV R6. A
F	MOV dir, R7 [2B, 2C]	SUBB A, R7	MOV R7, dir [2B, 2C]	CJNE R7, #data, rel [3B, 2C]	XCH A, R7	DJNZ R7, rel [2B, 2C]	MOV A, R7	MOV R7, A

Note: Key: [2B] = 2 Byte, [3B] = 3 Byte, [2C] = 2 Cycle, [4C] = 4 Cycle, Blank = 1 byte/1 cycle

≅ Desglose de las instrucciones por tipo de operación:

Tabla 27: Desglose de instrucciones

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,R_n	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@R_i	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,R_n	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@R_i	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,R_n	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@R_i	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	R_n	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@R_i	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	R_n	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@R_i	Decrement indirect RAM	1	12
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12

Note: 1. All mnemonics copyrighted © Intel Corp., 1980.

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS				
ANL	A,R_n	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@R_i	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,R_n	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@R_i	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,R_n	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@R_i	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12

Tabla 27: Continuación...

LOGICAL OPERATIONS (continued)				
Mnemonic		Description	Byte	Oscillator Period
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,R_n	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@R_i	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	R_n,A	Move Accumulator to register	1	12
MOV	R_n,direct	Move direct byte to register	2	24
MOV	R_n,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,R_n	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@R_i	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@R_i,A	Move Accumulator to indirect RAM	1	12
MOV	@R_i,direct	Move direct byte to indirect RAM	2	24
MOV	@R_i,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@R_i	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move Exernal RAM (16-bit addr) to Acc	1	24
MOVX	@R_i,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,R_n	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@R_i	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@R_i	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24

Tabla 27: Continuación...

PROGRAM BRANCHING

Mnemonic		Description	Byte	Oscillator Period
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	R_n,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@R_i,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	R_n,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

≅ Códigos de operación para cada una de las instrucciones:

Tabla 28: Codificación de las Instrucciones

Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr,code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr,code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr

Hex Code	Number of Bytes	Mnemonic	Operands
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2

Tabla 28: Continuación...

Hex Code	Number of Bytes	Mnemonic	Operands
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

2.3 Modos de direccionamiento

2.3.1 Direccionamiento Directo

El operando es especificado por un campo de acción de 8 bits en la instrucción; la RAM y los Registros de Funciones Especiales se pueden direccionar así:

```
MOV     A,7FH
ADD     A,#3BH
```

2.3.2 Direccionamiento Indirecto

La instrucción especifica un registro que contiene la dirección del operando; tanto la RAM interna como la memoria externa pueden ser accesadas por este medio:

@R0 - *@R1*.- Para los datos de 8 bits;

DPTR .- Para las datos de 16 bits.

2.3.3 Instrucciones para Registros

Los bancos de registros contienen 8 registros cada uno, *R0 a R7*, pueden ser accesados por instrucciones sumando 3 bits para especificar el registro, esto relacionado al op-code:

rrr ..... en la columna de codificación se indica el registro solicitado en la instrucción:

r	r	r	
0	0	0	R0
0	0	1	R1
0	1	0	R2
—	—	—	— — — — — — — — — —
1	1	1	R7

Para seleccionar el banco de registro se lo hace por medio del PSW.

2.3.4 Constantes Inmediatas

El valor de la constante puede seguir al op-code en la memoria de programa. Así:

```
MOV          A, #100  ;  Carga el acumulador con el número decimal 100.
```

2.3.5 Direccionamiento Indicado (*indexed*)

Únicamente la memoria de programa se puede acceder por esta vía, se lo emplea en la lectura de tablas. Un registro base de 16 bits apunta a la base de la tabla, el contenido del acumulador es el offset que permite acceder a la lectura de esa posición de la tabla:

. *La dirección de la tabla es la suma del acumulador y el puntero base.*

Se puede usar otro tipo de direccionamiento indexado para los saltos:

. *La dirección de salto es la suma del puntero base y el acumulador.*

2.3.6 Direccionamiento implícito

Algunas instrucciones no requieren de especificar el destino ya que esta incluido en el código de la instrucción:

INC A

INC DPTR

2.4 Tipo de Instrucciones

2.4.1 Instrucciones Aritméticas

Tabla 29: Instrucciones Aritméticas

Mnemonic	Operation	Addressing Modes				Execution Time (µS)
		Dir	Ind	Reg	Imm	
ADD A, <byte>	A = A + <byte>	X	X	X	X	1
ADDC A, <byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A, <byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator Only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer Only				2
DEC A	A = A – 1	Accumulator Only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B Only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B Only				4
DA A	Decimal Adjust	Accumulator Only				1

Nota: se asume un reloj de 12 MHz.

~ La instrucción INC DPTR opera con un puntero de datos de 16 bits, utilizado para la memoria externa.

- Se puede incrementar un byte sin el uso del acumulador.
- La instrucción MUL AB realiza el producto del acumulador y el registro B y permite un cociente de 8 bits en el acumulador con un remanente de 8 bits en el registro B.
- La instrucción DA A se usa en operaciones BCD, colocada siempre después de las instrucciones ADD y ADDC. DA A no convierte un número binario a BCD, sino produce un ajuste BCD del resultado, como un segundo paso en la suma de dos bytes BCD.

2.4.2 Instrucciones Lógicas

Tabla 30: Instrucciones Lógicas

Mnemonic	Operation	Addressing Modes				Execution Time (µS)
		Dir	Ind	Reg	Imm	
ANL A, <byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte> ,A	<byte> = <byte> .AND. A	X				1
ANL <byte> ,#data	<byte> = <byte> .AND. #data	X				2
ORL A, <byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte> ,A	<byte> = <byte> .OR. A	X				1
ORL <byte> ,#data	<byte> = <byte> .OR. #data	X				2
XRL A, <byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte> ,A	<byte> = <byte> .XOR. A	X				1
XRL <byte> ,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator Only				1
CPL A	A = .NOT. A	Accumulator Only				1
RL A	Rotate ACC Left 1 bit	Accumulator Only				1
RLC A	Rotate Left through Carry	Accumulator Only				1
RR A	Rotate ACC Right 1 bit	Accumulator Only				1
RRC A	Rotate Right through Carry	Accumulator Only				1
SWAP A	Swap Nibbles in A	Accumulator Only				1

Nota: se asume un reloj de 12 MHz.

- Se tiene instrucciones lógicas: AND, OR, XOR, NOT.

~ Las instrucciones de rotación RL A - RR A desplazan un bit, en el acumulador, a la izquierda y derecha respectivamente sin afectar al bit de acarreo.

~ Las instrucciones de rotación RLC A - RRC A desplazan el contenido del acumulador implicando el bit de acarreo.

~ La instrucción SWAP A intercambia el "nibble bajo" con el "nibble alto" del acumulador. Se la emplea con operaciones BCD.

2.4.3 Instrucciones de Transferencia de datos

Se considera según el lugar de la transferencia:

- *Movimiento en la RAM interna o en los SFR:*

Tabla 31: Instrucciones de Transferencia

Mnemonic	Operation	Addressing Modes				Execution Time (µS)
		Dir	Ind	Reg	Imm	
MOV A, <src>	A = <src>	X	X	X	X	1
MOV <dest> ,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, #data16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>, "@SP" ; DEC SP	X				2
XCH A, <byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A, @Ri	ACC and @Ri exchange low nibbles		X			1

Nota: se asume un reloj de 12 MHz.

~ La instrucción PUSH primero incrementa el Stack Pointer, luego copia el byte en el interior de la pila. Para un direccionamiento directo se usa PUSH – POP,

identificando al byte que se esta guardando o realmacenando. Pero la pila es accesada por direccionamiento indirecto con el registro SP.

~ La instrucción MOV DPTR, #dato de 16 bits, permite transferir datos para inicializar el DPTR. Empleado en el manejo de tablas en la memoria externa.

~ La instrucción XCH A, byte intercambia los datos del acumulador con el byte planteado; XCHD A,@Ri es similar, pero solo intercambia el nibble bajo, se la usa en BCD.

- *Movimiento sobre la RAM externa:*

Tabla 32: Instrucciones de transferencia en la memoria de datos externa

Address Width	Mnemonic	Operation	Execution Time (µs)
8 bits	MOVX A, @Ri	Read external RAM @ Ri	2
8 bits	MOVX @Ri,A	Write external RAM @ Ri	2
16 bits	MOVX A, @DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @ DPTR	2

~ Solo se las puede usar por direccionamiento indirecto; pudiendo ser: R0 o R1 de un banco de registro, o el @DPTR.

~ Se debe notar que el acumulador es empleado como registro fuente o destino.

~ Al emplear la 3era y 4ta instrucción se debe tener en cuenta que se va trabajar con el P2, esto es un inconveniente ya que se pierde un puerto.

~ Las líneas de control de RD y WR son activadas cuando se usa solamente con la instrucción MOVX.

- *Movimiento en tablas:*

Tabla 33: Instrucciones de lectura de tablas

Mnemonic	Operation	Execution Time (µs)
MOVC A, @A + DPTR	Read Pgm Memory at (A + DPTR)	2
MOVC A, @A + PC	Read Pgm Memory at (A + PC)	2

~ Las dos instrucciones para el manejo de tablas son las presentadas, independiente del cualquier algoritmo que se cree para el efecto.

~ Las tablas pueden ser leídas y no actualizadas.

~ Si la tabla es en la memoria externa, se tiene la señal PSEN para permitirla.

~ La 1era instrucción permite la lectura de tablas en accesos no superiores a 256. Siendo el acceso cargado en el acumulador mientras que DPTR establece el inicio de la tabla, y la lectura va al acumulador.

~ La 2da instrucción es semejante pero ahora el PC actúa indicando la dirección de la base de la tabla.

2.4.4 Instrucciones Booleanas

Tabla 34: Instrucciones Booleanas

Mnemonic		Operation	Execution Time (µs)
ANL	C,bit	C = C .AND. bit	2
ANL	C,/bit	C = C .AND. .NOT. bit	2
ORL	C,bit	C = C .OR. bit	2
ORL	C,/bit	C = C .OR. .NOT. bit	2
MOV	C,bit	C = bit	1
MOV	bit,C	bit = C	2
CLR	C	C = 0	1
CLR	bit	bit = 0	1
SETB	C	C = 1	1
SETB	bit	bit = 1	1
CPL	C	C = .NOT. C	1
CPL	bit	bit = .NOT. bit	1
JC	rel	Jump if C = 1	2
JNC	rel	Jump if C = 0	2
JB	bit,rel	Jump if bit = 1	2
JNB	bit,rel	Jump if bit = 0	2
JBC	bit,rel	Jump if bit = 1; CLR bit	2

- Los 128 bytes de la RAM interna pueden manipularse por "bit a bit". Al igual que los pines de los puertos.
- Se tienen operaciones de complementar, limpiar, OR, AND, MOV.
- La dirección de destino indicada, para los saltos, es indicada por el assembler o por una dirección en la memoria de programa. La dirección de destino ensamblada es el *byte de offset relativo.*

Estando en complemento a dos, señala el byte de offset que es sumado al PC si el salto es ejecutado. El rango de salto es (–128) a (+128) de los bytes relativos de la memoria de programa, al primer byte siguiente a la instrucción.

2.4.5 Instrucciones de bifurcación

Se tiene dos situaciones:

- *Bifurcaciones incondicionales:*

Tabla 35: Saltos incondicionales

Mnemonic	Operation	Execution Time (µs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interurpt	2

~ A pesar que solo se muestra JMP que es genérico, utilizable si al programar no se cuida el tipo de salto, se debe notar que en realidad existen 3 para éste fin: SJMP – LJMP – AJMP. Las mismas que se distinguen por la dirección destino.

~ La instrucción JMP @A+DPTR es un salto indirecto. Suma el byte contenido en el acumulador con los 16 bits del puntero de datos DPTR, el resultado lo carga en el PC. Siendo la dirección de la siguiente búsqueda de la instrucción.

~ A la vez se visualiza una instrucción CALL, pero en realidad hay 2: LCALL – ACALL, que se puede usar en forma genérica si al programar no se cuida las direcciones de "llamado".

~ Al finalizar una subrutina se deberá colocar RET para regresar y ejecutar la siguiente instrucción a CALL; pero RETI es usado para regresar de una rutina de interrupción.

- *Bifurcaciones Condicionales:*

Tabla 36: Saltos Condicionales

Mnemonic		Operation	Addressing Modes				Execution Time (µS)
			Dir	Ind	Reg	Imm	
JZ	rel	Jump if A = 0	Accumulator Only				2
JNZ	rel	Jump if A ≠ 0	Accumulator Only				2
DJNZ	<byte>,rel	Decrement and jump if not zero	X		X		2
CJNE	A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE	<byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

~ Permiten al equipo mantener una secuencia de programa lógico, con una contestación *"SI"* o *"NO"* a una pregunta específica.

~ Dado que la dirección de destino está implícita en la instrucción y es el operando el señalador del salto en el formato de offset relativo, la distancia de salto es limitada a (-128) a (+127) bytes refiriéndose a la siguiente instrucción a la de salto.

~ En la programación solo se debe indicar con una etiqueta la dirección de salto.

~ Normalmente DJNZ se emplea para secuencias de temporizaciones por software.

~ CJNE permite establecer una comparación de 2 bytes y saltar a otra dirección si los dos no son iguales.

3 Software Assembler MCS-51

3.1 Generalidades

Este software opera en D.O.S proveyendo una amplia versatilidad de operación; además, es compatible con la familia de microcontroladores Atmel. Para poder llamarlo se tipea:

ASEM<source>[<hex>[<list>]][/INCLUDES:p][/DEFINE:s:v:t][/COLUMNS]

En donde:

<source> : es el código fuente assembler;
<hex> : es el archivo de salida en formato hexadecimal;

/INCLUDE: se usa para que el ensamblador busque un grupo específico para incluir en los archivos que no pueden ser encontrados en el directorio de trabajo. "p" puede ser cualquier numero de directorio separado por el caracter ";" (punto y coma).

/DEFINE: ésta opción es usada para seleccionar variantes de programa particulares, desde la línea de comando que sería implementada con ensamblado condicional. Permite definir un símbolo "s", con un valor "v", y un segmento "t"; en donde "v" y "t" son opcionales.

Si se omite el segmento, por defaults se coloca NUMBER; si el valor del símbolo es "0", "v" es omitido. Ahora, "v" seria cualquier constante numérica:

C = CODE

D = DATA

I = IDATA

X = XDATA

B = BIT

N = NUMERO (*por defaults*)

Ejemplos:

ASEM PROGRAM

Se genera el ensamblado del programa con la terminación ".A51"; luego se produce los archivos ".HEX" y ".LST".

ASEM UNIVERSL /D:Eva_Board:8000H:C

Se ensamblaría el programa UNIVERSL.A51, mientras que el CODE símbolo EVA_BOARD sería predefinido con el valor 8000 durante el ensamblado.

Una vez efectuado el ensamblado, se puede visualizar en pantalla los errores cometidos; así:

MCS-51 Family Cross Assembler ASEM-51 V1.2

APPLICAT.A51(14): será conocido en el primer paso

USERBITS.INC(6): prueba dividir por cero

DEFINES.INC(37): símbolo no definido

APPLICAT.A51(20): símbolo no definido

APPLICAT.A51(27): no se encuentra la sentencia END

5 errores detectados

De esta manera, se muestran los errores ocurridos con el nombre de la fuente o del archivo incluido; el valor de la línea local en donde se lo encontró, y el mensaje de error correspondiente.

Incluido en el programa se halla un convertidor de hexadecimal a binario, que permite a partir de un archivo con ".hex" se obtenga uno con ".bin". Q ue se lo puede emplear en la grabación o en otras aplicaciones puntuales. Se lo puede llamar así:

HEXBIN<hex>[<bin>][/OFFSET:o][/LENGTH:1][/FILL:f]

En donde, hex es el archivo hexadecimal, y, bin el archivo de salida binario; las opciones OFFSET, LENGTH y FILL son para controlar el archivo binario de salida.

3.2 Lenguaje Assembler

3.2.1 Sentencias

En los archivos fuente se hallan sentencias que pueden ser de las siguientes formas:

[símbolo:] [instrucción[argumento]] [;comentario]

símbolo instrucción argumento [;comentario]

$control [argumento] [;comentario]

Se tiene las consideraciones:

a.- Lo escrito entre corchetes es opcional,

b.- La máxima largura de las líneas del código fuente es de 255 caracteres,

c.- Luego del ";" se asume como comentario,

d.- Para separar los léxicos se emplea el "tab" o "espacios en blanco".

Ejemplos:

AQUI: MOV A,#0FFH ;define nivel AQUI y carga A con #0FFH

YEAR EQU 2002 ;define símbolo para el año actual

$INCLUDE(80C517.MCU) ;incluye el registro de definiciones del SAB80C517

3.2.2 Símbolos

Son nombres definidos para direcciones o números; la largura máxima es de 31 caracteres, por lo que más allá de ello son ignorados. Estos podrían consistir de letras, dígitos, "_", y "?". Se debe tomar en consideración que el lenguaje de palabras assemblers no será redefinido para usarse como símbolo.

Ejemplo: Es_esto_realmente_ un_Símbolo?

3.2.3 Constantes

Las constantes numéricas consisten en una secuencia de dígitos, seguidos por un índice especifico. El primer carácter será un dígito decimal siempre; siendo éstos:

Tabla 37: Constantes

constante	dígito	índice
binario	0 - 1	B
octal	0 - 7	Q o O
decimal	0 - 9	D o ninguno
hex	0 - F	H

Por ejemplo:

11111111B	binario
177Q	octal
177o	octal
127D	decimal
127	decimal
07FH	hexadecimal

Un caracter constante podría ser usado siempre y cuando un valor numérico sea asignado; siendo dicho caracter constante el que se presenta encerrado entre comillas simples o dobles. Ejemplo:

'X'	constante de 8 bits: 58H
"a@"	constante de 16 bits: 6140H
""	constante de 8 bits: 27H

Con la sentencia DB, la largura de los caracteres puede ser cualquiera; dado ésto se llamaría a una fila de caracteres:

"Esto es solamente texto!"

3.2.4 Expresiones

Las expresiones aritméticas son compuestas de operandos, operadores y paréntesis. Los operandos podrían ser definidos por símbolos, constantes o por símbolos assembler especiales; y ser tratados como números no señalados de 16 bits.

Los símbolos assembler especiales que pueden ser usados como operandos son:

AR0, – , AR7	direccionamiento directo o registro R0 a R7
$	cuenta la localización del segmento activo corrientemente (arranca con la dirección de la sentencia assembler corriente)

Entre tanto, los operadores implementados son:

Unary operators: + identity: +x = x

- two' s complement: -x = 0-x

NOT one' s complement: NOT x = FFFFH-x

HIGH high order byte

LOW low order byte

Binary operators: + unsigned addition

- unsigned subtraction

* unsigned multiplication

/ unsigned division

MOD unsigned remainder

SHL logical shift left

SHR logical shift right

AND logical and

OR logical or

XOR exclusive or

. bit operator used for bit-adressable locations

EQ or = equal to

NE or <> not equal to

LE or <= less or equal than

LT or < less than

GT or > greater than

GE or >= greater or equal than

Resultados son:
0 si FALSE
FFFF si TRUE

Observar las siguientes consideraciones:

- Los caracteres que no son especiales (SHR – AND) serán separados de sus operandos por un ‘tab” o un espacio en blanco.

- Las expresiones generales son evaluadas de izquierda a derecha de acuerdo al operador de origen, que sería anulado por el paréntesis.

Los operadores de procedencia son:

Operator precedence:

( )		^ highest
+ - NOT HIGH LOW	(unary)	|
.		|
* / MOD		|
SHL SHR		|
+ -	(binary)	|
EQ = NE <> LT < LE <= GT > GE >=		|
AND		|
OR XOR		v lowest

3.2.5 Pseudo Instrucciones

Son símbolos léxicos en los que se escriben en letras minúsculas, mientras que los códigos claves assembler van en letras mayúsculas. Los argumentos de las instrucciones son representados por <arg>,<arg1>; las expresiones son representadas por <exp>,<exp1>.

Lo escrito entre corchetes es opcional; los "..." (tres puntos seguidos) significan siempre "una lista con cualquier numero de elementos".

Las pseudo instrucciones usadas son:

DB <arg1>[,<arg2>[,<arg3> ...]]

Reserva e inicializa un numero de bytes con los valores definidos por los argumentos. Los argumentos serían expresiones o caracteres de cualquier largura. DB se permite solo en el segmento CODE.

DW <expr1>[,<expr2>[,<expr3> ...]]

Reserva e inicializa un numero de palabras con los valores definidos por los argumentos. Cada argumento podría ser una expresión arbitraria y requiere de dos bytes de espacio. DW solo se permite en el segmento CODE.

DS <expr>

Reserva un número de bytes no inicializados en el segmento corriente. El valor de <expr> se conocerá en el primer paso. DS se permite en cada segmento.

DBIT <expr>

Reserva un número de bits no inicializados. El valor de la expresión se conocerá en el paso 1. DBIT solo se permite en el segmento BIT.

NAME <symbol>

Define el nombre para el módulo de programa, el cual será un símbolo legal. Debiéndose dar solamente una instrucción NAME en todo el programa. Puesto que la instrucción ha sido introducida por propósito de compatibilidad solamente, el símbolo no se usa frecuentemente; ya que se podría redefinir en la secuencia del programa.

ORG <expr>

Coloca la localización del contador del segmento común al valor de la expresión. Adicionalmente la sentencia ORG podría ser usada para generar segmentos de programa que cargarían diferentes localizaciones. El valor de la expresión se conocerá en el primer paso; y, el valor por default del contador de localizaciones del programa (program counter) comienza en "0".

USING <expr>

Coloca el banco de registros usados por la expresión, comprendiendo desde el banco 0 al banco 3; ésta instrucción solamente afecta a los valores de los símbolos especiales

assembler AR0....AR7 (registros R0....R7) del banco escogido. El valor de la expresión se conocerá en el primer paso; además, por default es seleccionado el banco 0.

END

Comprende la última sentencia del programa (archivo fuente); después de ella solo se permiten líneas en blanco o solamente comentarios.

<symbol> EQU <expr>

<symbol> SET <expr>

La instrucción EQU define una constante simbólica del tipo NUMBER. Pero el símbolo definido por EQU nunca puede ser modificado. La instrucción SET define un valor simbólico del tipo NUMBER, el mismo podría ser modificado con la aplicación de futuras sentencias SET.

El valor de la expresión se conocerá en el primer paso. Lo señalado por SET no puede ser redefinido por EQU, y lo definido por EQU no puede ser cambiado por SET. Una vez en el paso 2, las referencias adelantadas para un SET de un símbolo siempre se evalúan para el último valor, el símbolo ha sido SET en el paso 1.

<symbol> CODE <expr>

<symbol> DATA <expr>

<symbol> IDATA <expr>

<symbol> BIT <expr>

<symbol> XDATA <expr>

Todas estas instrucciones definen direcciones simbólicas para los 5 segmentos de memoria del 8051. Para los símbolos tipo DATA, IDATA y BIT el valor de la expresión no excederá de 0FFH. El valor de la expresión se conocerá en el primer paso. Una vez definida uno de las instrucciones, los símbolos no pueden ser redefinidos.

CSEG [AT <expr>]

DSEG [AT <expr>]

ISEG [AT <expr>]

BSEG [AT <expr>]

XSEG [AT <expr>]

Estas instrucciones conectan a uno de los 5 segmentos de memoria del 8051, y opcionalmente colocan en el contador de localizaciones del segmento en una dirección en particular <expr>. Al omitirse AT <expr> , el contador de localizaciones guarda el valor previo; el valor de la <expr> se conocerá en el primer paso. Al arrancar el programa por default el segmento es CODE y todos los conteos de localizaciones se setean a ‘0”.

- Cada vez que se ensambla una expresión es asignado un tipo de segmento, tomando en cuenta si es un operador u operando; el tipo de segmento indica el espacio a direccionar si es que se uso como una dirección. Son posible 6 tipos de segmentos:

  CODE

  DATA

  IDATA

  BIT

XDATA

NUMBER

Las demás expresiones tienen a NUMBER como tipo de segmento seleccionado, por lo que es asumido como el más bajo. Sin embargo, en algunos casos se podría usar para asignar un tipo de segmento en particular.

Para poder evaluar a los segmentos se tiene las siguientes reglas:

1. Las constantes numéricas tiene el mas bajo tipo; por lo tanto son segmentos del tipo NUMBER.
2. Los símbolos son asignados a un tipo de segmento durante la definición. Ahora, los símbolos que son definidos por EQU o SET no tienen tipos de segmentos; los niveles obtienen el tipo de segmento del corriente activo.
3. El resultado de una operación (+, -, NOT, HIGH, LOW) tendrán el tipo de segmento de estos operandos.
4. Los resultados de toda operación binaria (excepto + , - , y “:”) no tendrían tipo de segmento.
5. Si solamente un operando en una operación binaria, de suma o resta, tuvo un tipo de segmento, el resultado tendrá el mismo segmento. Y, en otros casos no tendrá segmento.
6. El resultado de una operación de bit “:” tendría siempre el tipo de segmento de BIT.

3.2.6 Controles Assembler

Este programa (ASEM-51) implementa un numero de controles que influyen en el ensamblado y en la generación del archivo LST (lista). Para lo cual se tiene:

- Controles Primarios, que pueden ser usados solo al principio del programa y permanecen efectivos por todo el ensamblado. Ellos solo podrían ser precedidos solo por una sentencia de control, comentarios o líneas en blanco. Si se emplea el mismo control primario varias veces con diferentes parámetros, el último es el que cuenta.

- Controles Generales, se pueden usar en todas partes de la programación; ellos cumplen una acción simple, o se mantienen efectivos hasta que son cancelados o cambiados por la siguiente sentencia de control.

Los controles assembler podrían tener un tipo de operando number o string: son operandos tipo Number las expresiones aritméticas conocidas en el paso 1; son operandos tipo String los caracteres encerrados entre paréntesis en lugar de comillas.

Se expone las explicaciones más detalladas de las implementaciones de los controles y sus abreviaciones:

Control	Type	Default	Abbreviation	Meaning

$DATE(string) P ' ' $DA inserts date string into page header

$DEBUG P $NODEBUG $DB (currently dummy)

$NODEBUG P $NODB (" ")

$EJECT G $EJ start a new page in list file

$INCLUDE(file)G $IC include a source file

$LIST G $LIST $LI list subsequent source lines

$NOLIST G $NOLI don' t list subsequent source lines

$MOD51 P $MOD51 $MO enable predefined SFR symbols

$NOMOD51 P $NOMO disable predefined SFR symbols

$PAGING P $PAGING $PI enable listing page formatting

$NOPAGING P $NOPI disable listing page formatting

$PAGELENGTH(n) P n=64 $PL set lines per page for listing

$PAGEWIDTH(n) P n=132 $PW set columns per line for listing

$PHILIPS P MCS-51 --- switch on 83C75x family support

$SYMBOLS	P	$SYMBOLS	$SB	create symbol table
$NOSYMBOLS	P		$NOSB	don' t create symbol table
$NOTABS	P	use tabs ---		don' t use tabs in list file
$TITLE(string)	G	copyright	$TT	inserts title string into page header
$XREF	P	$NOXREF	$XR	create cross reference
$NOXREF	P		$NOXR	don' t create cross reference

θ **Controles Primarios**

⊗ **$DATE (string) >** Inserta la hilera de la fecha en el archivo "lista", en el encabezado. Si $DATE es especificado, la fecha actual es colocada; la largura de la hilera de la fecha será máximo de 11 caracteres. Por default no se coloca la fecha; además, éste control no tendrá efecto si el control $NOPAGING ha sido especificado.

⊗ **DEBUG >** Figurativo. Solo para propósitos de compatibilidad.

⊗ **$NODEBUG >** Figurativo. Para propósitos de compatibilidad.

⊗ **$MOD51 >** Enciende el constructor de los registros especiales del 8051 y las definiciones de interrupción de símbolos. Por default.

⊗ **$NOMOD51 >** Apaga el constructor de los registros especiales del 8051 y las definiciones de interrupción de símbolos.

⊗ **$PAGING >** Enciende el formateador de página en el archivo lista. Por default.

⊗ **$NOPAGING >** Apaga el formateador de página en el archivo lista.

⊗ **$PAGELENGTH (n) >** Coloca el largo de página del archivo lista a "n" líneas.

(12 ⇐ n ⇐ 65535). Por default n = 64

Este control no tiene efecto si se ha especificado el control $NOPAGING.

⊗ **$PAGEWIDTH (n) >** Coloca el archivo lista con "n" columnas.

(72 ⇐ n ⇐ 255). Por default n = 132.

⊗ **$PHILIPS >** Da la opción de soporte de los microcontroladores de la Philips 83C75X. Esta deshabilita las instrucciones LJMP – LCALL – MOVX, y, también las pseudo instrucciones XDATA y XSEG. Entre tanto, los saltos genéricos son llamados siempre al ensamblar para una dirección absoluta.

- ⊗ **$SYMBOLS >** Genera una tabla de símbolos al finalizar el archivo lista. Esto por default. Se neutraliza al especificarse el comando $XREF.

- ⊗ **$NOSYMBOLS >** Suprime la tabla de símbolos al finalizar el archivo lista. Cuando se activa $XREF no tiene efecto éste comando.

- ⊗ **$NOTABS >** Dilata todos los caracteres tabs en la salida del archivo lista a espacios en blancos.

- ⊗ **$XREF >** Genera una referencia de cruce en vez de una tabla de símbolos. Note que ésta es ensamblada ligeramente mas lento, y ocupa cerca de 67% más espacio de memoria.

- ⊗ **NOXREF >** Genera una tabla de símbolos en lugar de una referencia de cruce. Esto por default.

θ **Controles Generales**

- ∅ **$EJECT >** Coloca una nueva página en el archivo lista. Pero este control no tiene efecto si $NOPAGING ha sido activado.

- ∅ **$INCLUDE (file) >** Incluye un archivo fuente externo dentro del assembler del programa justo detrás de la sentencia $INCLUDE. Si el archivo incluido no se

puede encontrar en el directorio por default, entonces la ruta especificada por el comando es buscada.

Los archivo incluidos podrían estar anidados.

∅ **$NOLIST >** Después de este control las líneas del código fuente son mas cortas, proveído ello no tiene cabida para errores, hasta que la siguiente sentencia ocurre.

∅ **$LIST >**Genera un listado del source file luego del ensamblado, mostrando la posición, el código que le corresponde, y su "source".

∅ **$TITLE (string) >** Inserta una hilera de caracteres para el titulo en el encabezamiento del archivo lista. El titulo sería bloqueado si se activa WIDTH; o por default, que es en donde va el copyright del programa.

El control no tiene efecto cuando se ha especificado $NOPAGING.

3.2.7 Assembler Condicionales

Se usan para configurar fácilmente los controles y soportar las aplicaciones de los programas, siendo:

IF <expr>

IFDEF <symbol>

IFNDEF<symbol>

ELSE

ENDIF

La sentencia IF opera de la siguiente forma:

Al darse la expresión <expr>, en la sentencia IF, diferente de "0" (verdadero); entonces las sentencias de 1 = n son ensambladas y aquellas de "n+1" a "n+m" son ignoradas.

```
IF <expr>
  <statement 1>
  <statement 2>       ;assembled when <expr> is TRUE
     .
  <statement n>
ELSE
  <statement n+1>
  <statement n+2>     ;assembled when <expr> is FALSE
     .
  <statement n+m>
ENDIF
```

Pero al darse que la expresión <expr> es igual a "0" (falso) se produce lo inverso. La sentencia principal "1" es ignorada y se ensamblan las sentencias "n+1" a "n+m".

También se realiza aun cuando las bifurcaciones IF o ELSE no contengan todas sentencias.

Al darse ésto, ELSE no tiene encerrada ninguna sentencia, la misma estructura puede simplificarse a:

```
IF <expr>
  <statement 1>
  <statement 2>       ;assembled when <expr> is TRUE
     .
  <statement n>
ENDIF
```

El valor de la <expr> se conocerá en el paso 1. Ejemplo:

```
TARGET EQU 0    ;configuration:  1 for application board
                ;--------------  0 for evaluation board
IF TARGET
  ORG 0         ;program start address of application board
ELSE
  ORG 08000H    ;program start address of evaluation board
ENDIF
```

Ahora bien, las sentencias IFDEF y IFNDEF operan de manera igual:

```
IFDEF <symbol>
  .
ELSE
  .
ENDIF
```

- Cuando el símbolo es definido en el programa, las sentencias en la bifurcación IFDEF son ensambladas, pero aquellas en ELSE son ignoradas.
- Cuando el símbolo no es definido en el programa, pasa lo contrario.

```
IFNDEF <symbol>
.
ELSE
.
ENDIF
```

- Cuando el símbolo es definido en el programa, las sentencias en la bifurcación IFNDEF son ignoradas, y, las de ELSE son ensambladas.
- Pero al no definirse el símbolo en el programa pasa lo contrario.

Ejemplo:

```
;EVA_537 EQU 0     ;symbol undefined: 80C537 application board
                   ;symbol defined:   80C537 evaluation board
IFNDEF EVA_537
```

```
  CLOCK EQU 16     ;clock frequency of application board
  CSEG  AT  0      ;program start address of application board
ELSE
  CLOCK EQU 12     ;clock frequency of evaluation board
  CSEG  AT  08000H ;program start address of evaluation board
ENDIF
```

- *Frecuentemente el programa es configurado para una aplicación 'board".*
- *Los símbolos de IFDEF y IFNDEF pueden o no ser definidos en el paso 1.*
- *Todas las condicionantes pueden ser anidadas en cualquier longitud.*

I Situaciones Especiales

Después que el ASEM-51 produce un archivo INTEL-hex reemplazando a un modulo objeto, el código fuente de un programa de aplicación para el 8051 tiene que residir en un archivo único. Consecuentemente todas las pseudo-instrucciones, que se trata con segmentos relocalizables o símbolos externos, no tendrán que ser implementadas:

```
PUBLIC
EXTRN
SEGMENT
RSEG
```

El comando SET no puede redefinir símbolos especiales assembler: registros. Además, Macros no pueden ser soportadas. Más ahora solamente los siguientes controles assembler y sus abreviaciones tendrían que implementarse:

	primary controls	abbrev.	general controls	abbrev.
Intel-controls	$DATE (<string>)	$DA	$EJECT	$EJ
	$DEBUG	$DB	$INCLUDE (<file>)	$IC
	$NODEBUG	$NODB	$LIST	$LI
	$MOD51	$MO	$NOLIST	$NOLI
	$NOMOD51	$NOMO	$TITLE (<string>)	$TT
	$PAGING	$PI		
	$NOPAGING	$NOPI		
	$SYMBOLS	$SB		
	$NOSYMBOLS	$NOSB		
	$PAGELENGTH (<lines>)	$PL		
	$PAGEWIDTH (<columns>)	$PW		
ASEM-51 controls	$NOTABS	-----		
	$PHILIPS	-----		

3.3 Formato del Archivo Lista

Se dispone de información ya que consta de:

a.- El encabezado de la página,

b.- El encabezado del archivo,

c.- Las líneas del titulo,

d.- El diagnostico de errores,

e.- La tabla de símbolos o listado de referencia de cruce.

```
ASEM-51 V1.2      Copyright (c) 1996 by W.W. Heinz      PAGE 1

      MCS-51 Family Cross Assembler   A S E M - 5 1   V 1.2
      ======================================================

       Source File:    DEMO.A51
       Object File:    DEMO.HEX                              Program
       List File:      DEMO.LST              per line

Line I     Addr      Code      Source

   1:                                 ;A sample List File Demo
   2:                                 ;------------------------------
   3:                            $NOMOD51                  ;no 8051 SFR
   4:          N      004F      $PAGEWIDTH (79)           ;79 columns
   5:                            $NOTABS                   ;expand tabs
                                   .
                                   .
                                   .
                                 END

                      register banks used:  0

                      18  errors detected

              C R O S S - R E F E R E N C E - L I S T I N G
              =============================================
```

* Al principio se tiene la información del copyright del programa, y el número de página; eventualmente se visualiza la fecha.

* El encabezado del archivo, solo primera página, genera:

- Identificación del ensamblador,
- Enlista todos los archivos ingreso y salida, y,
- Marca las columnas para las líneas de título.

* Las líneas del título:

- Columna "Line": contiene el número de líneas del archivo fuente.
- Columna "I": muestra el nivel de anidamiento del archivo.
- Columna "Addr": muestra la dirección de comienzo de la línea que frecuentemente es el segmento activo. Las direcciones de los segmentos CODE y XDATA tienen 4 dígitos numéricos, y, para los otros son de 2 dígitos numéricos. Para las líneas que no pueden ser asignadas a un segmento particular, el campo Addr es con espacios en blanco a la izquierda.
- Columna CODE: contendría los 4 bytes del código generado, los que son mas que suficiente para toda instrucción del 8051. Este es en hexadecimal.
- Columna Source: finalmente contiene el código fuente original.

* Al finalizar el programa se tiene el diagnóstico de posibles errores ocurridos y detectados, los que se visualizan en cantidad. Así mismo, el banco de registros usado.

3.4 Mensajes de Error

ERROR	*DESCRIPCION*
address out of range	La dirección de una instrucción de JMP o CALL no puede ser alcanzada con el modo de direccionamiento seleccionado
attempt to divide by zero	Durante la evaluación de una expresión en tiempo de ensamblado, el assembler no tuvo división por cero
binary operator expected	En ésta posición de una expresión, solo operadores binarios son permitidos
comma expected	Habría un caracter "," en la posición marcada
commands after END statement	La sentencia END está seguida por sentencias assembler adicionales
constant out of range	Una constante numérica es más grande que 65535
ENDIF statement expected	Existe una instrucción IF, IFDEF o IFNDEF que no termina con una instrucción ENDIF
expression out of range	El resultado de una expresión está grande o pequeño para tal propósito
file name expected	Habría un nombre de archivo válido en esa posición
illegal character	Una sentencia contiene caracteres que no son permitidos en el lenguaje assembler MCS-51

illegal constant	Existe un error de sintaxis en una constante numérica
illegal control statement	Una sentencia está comenzando con un keyword desconocido al principio de $
illegal operand	En ésta posición de una expresión, se ha esperado un operando válido
illegal operator	Una expresión contiene un caracter especial o un keyword en vez de un operador lógico o aritmético, que no está permitido en ese lugar
illegal statement syntax	Una sentencia contiene un elemento de sintaxis, que no es permitido en ese contexto
invalid base address	Una dirección DATA no direccionable bit a bit ha sido usada en el lado izquierdo de un operador "."
invalid bit number	Un número más grande que 7 ha sido usado en el lado derecho de un operador "."
invalid instruction	La instrucción ha sido previamente deshabilitada con el control $PHILIPS
module name already defined	Hay más de un NAME en el programa
must be known on first pass	El resultado de una expresión podrá evaluarse totalmente en el paso 1 del ensamblado
must be preseded by IF	Una instrucción ELSE o ENDIF ocurrió sin haber precedido IF, IFDEF o IFNDEF
no END statement found	El programa finalizó sin la sentencia END

not allowed in BIT segment	La instrucción no es permitida en un segmento BIT
only allowed in BIT segment	La instrucción solo es permitida en el segmento BIT
only allowed in CODE segment	La instrucción solo es permitida en el segmento de CODE
operand expected	La instrucción finalizó antes de estar completa la sintaxis
phase error	Sobre el paso 2, un valor de un símbolo fue evaluado con otro valor en el paso 1
preceded by non-control lines	Un control primario ocurrió con una sentencia que no existe en los controles assembler
segment limit exceeded	La localización del contador excede los bordes del segmento corriente
segment type mismatch	El tipo de segmento de un operando no iguala al tipo de instrucción
string exceeds end of line	Una hilera de caracteres no está apropiadamente terminada por comillas
symbol already defined	Prueba redefinir un símbolo que alrededor está definido
symbol name expected	Relacionaría un símbolo válido de nombre en esta posición
symbol not defined	El símbolo en referencia nunca ha sido definido
too many closing parentheses	Una expresión contiene más paréntesis cerrados que abiertos

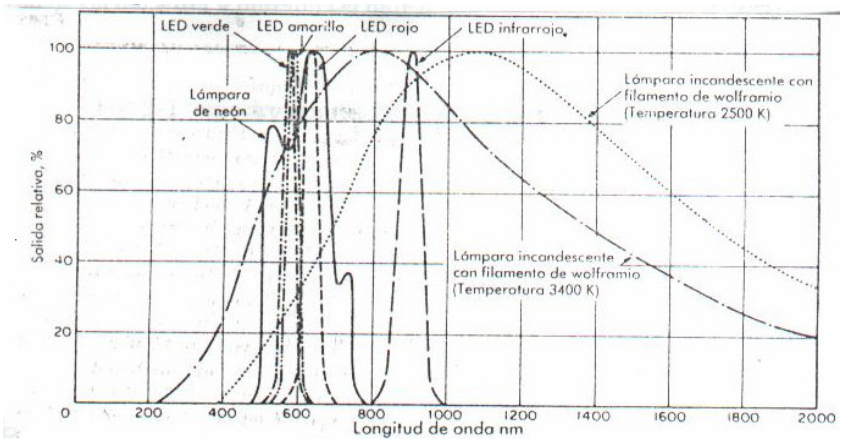
too many opening parentheses	Una expresión contiene más paréntesis abiertos que cerrados
too many operands	Una expresión contiene más operandos que los esperados
unary operator expected	En la posición de la expresión, son seguidos muchos operadores

4 Hardware utilizado por el microcontrolador AT89C51

4.1 Leds y Arreglos de leds.

4.1.1 ***Leds***.- (*light-emitting* diode), Estos componentes de estado sólido constituyen una nueva fuente de luminosidad con un espectro de luz muy angosto; en comparación con otras fuentes como: lámparas de filamento, lámparas de vapor o de gas, lámparas fluorescentes. Se utilizan en la visualización dentro de la optoelectrónica.

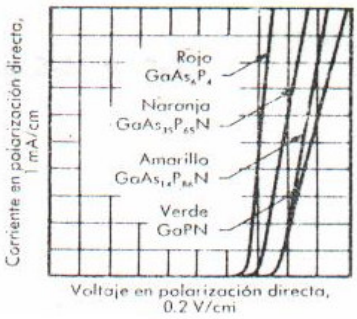
Figura 53: Espectros normalizados de varias fuentes luminosas



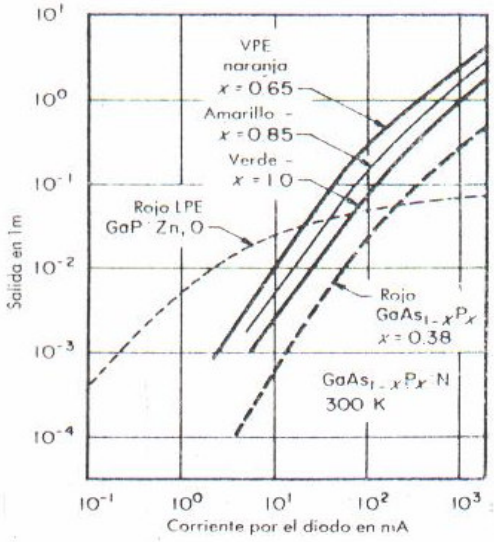
Teoría de conducción.- Tomando en consideración la teoría de conducción de la electrónica de estado sólido, en una unión p-n al existir una recombinación de huecos o electrones con otros portadores diferentes se genera una energía luminosa (fotones), y una energía calorífica (fonones). Esta energía es irradiada al medio. Esto se debe a que al desprenderse el electrón de la última capa de valencia del semiconductor irradia estas energías, pero en el caso de la luminosa es transparente. Por medios mecánicos se da color a esta luz, lográndose colores diferentes.

Pero se debe anotar que la cantidad de luminosidad esta en proporción de la cantidad de corriente que circula por él, no así el color que es de acuerdo al material utilizado.

Figura 54: Curvas características de un LED



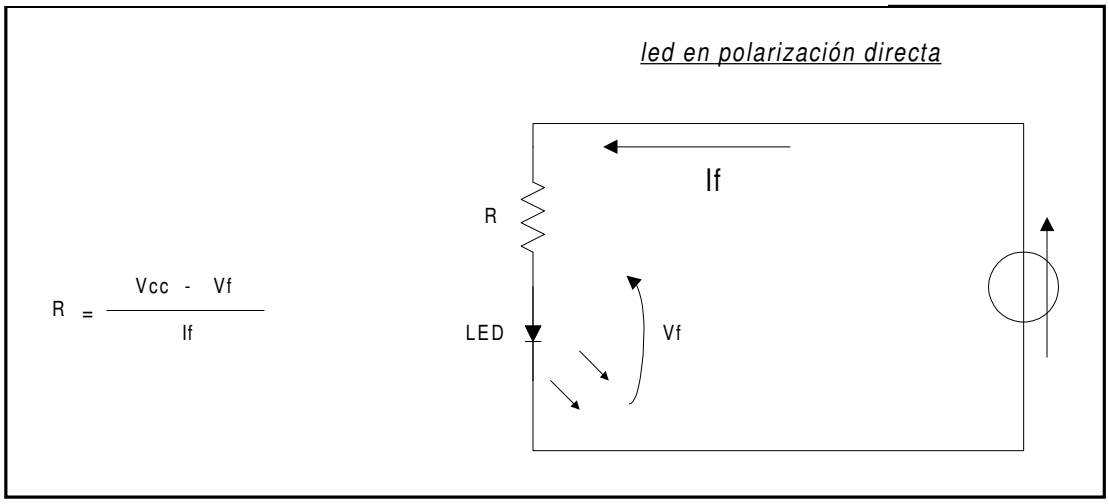
Corriente de polarización directa en función del voltaje de polarización directa.



Gráficas de intensidad luminosa producida en función de la densidad de corriente. LPE se refiere a un proceso epitaxial de fase líquida; VPE se refiere a un proceso epitaxial de fase gaseosa (de vapor).

Dado que el led debe tener una cierta cantidad de voltaje y de corriente, se lo suele conectar con una resistencia limitadora en serie. Entonces, la corriente de polarización directa que crece grandemente luego de la inflexión de la curva queda controlada; siendo la conexión y ecuación las siguientes:

Figura 55: Polarización de un led



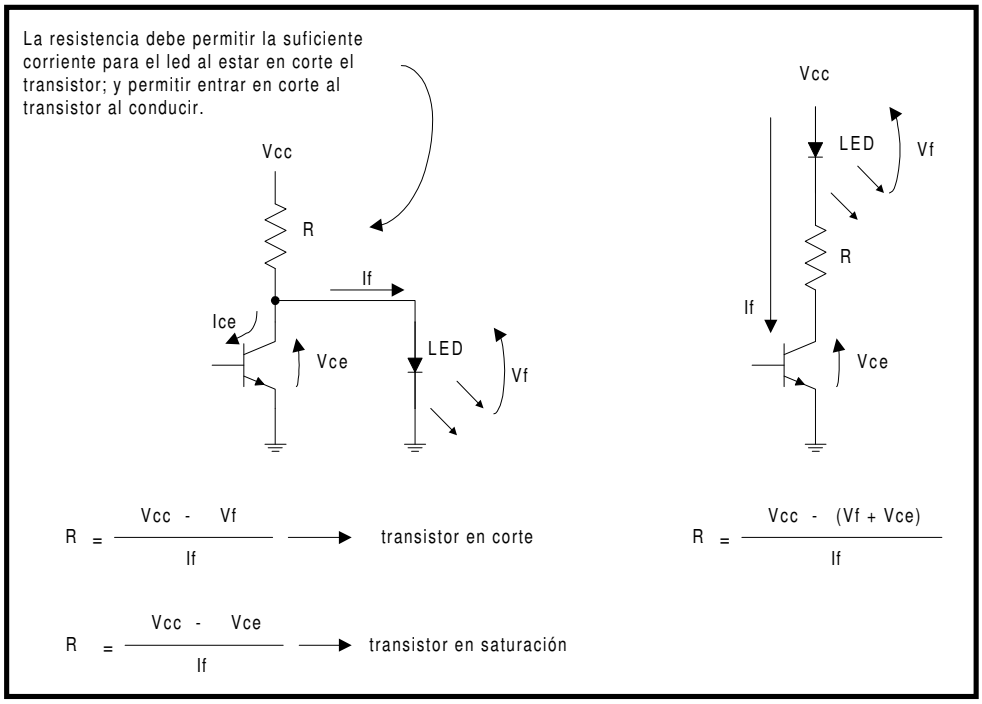
Es recomendable que cuando se tenga varios leds, cada uno lleve su propia resistencia; ya en la práctica con solo 4 o 5 milicandelas suficiente para lograr una visibilidad adecuada dentro de un ambiente brillante.

Y Dentro de las ventajas que ofrecen se tiene:

- El bajo consumo de potencia los hace compatibles con la gran mayoría de los circuitos electrónicos.
- La interfaz de conexión es sencilla.
- Dado que son a partir de materiales semiconductores, su vida útil es más larga y su coste de mantenimiento es realmente mínimo.
- Son inherentes a los ruidos del circuito, y tampoco lo producen.
- La respuesta es rápida, sin picos de corriente, ni períodos de establecimiento como otras lámparas.
- El diseñador le puede dar la forma que más requiera.

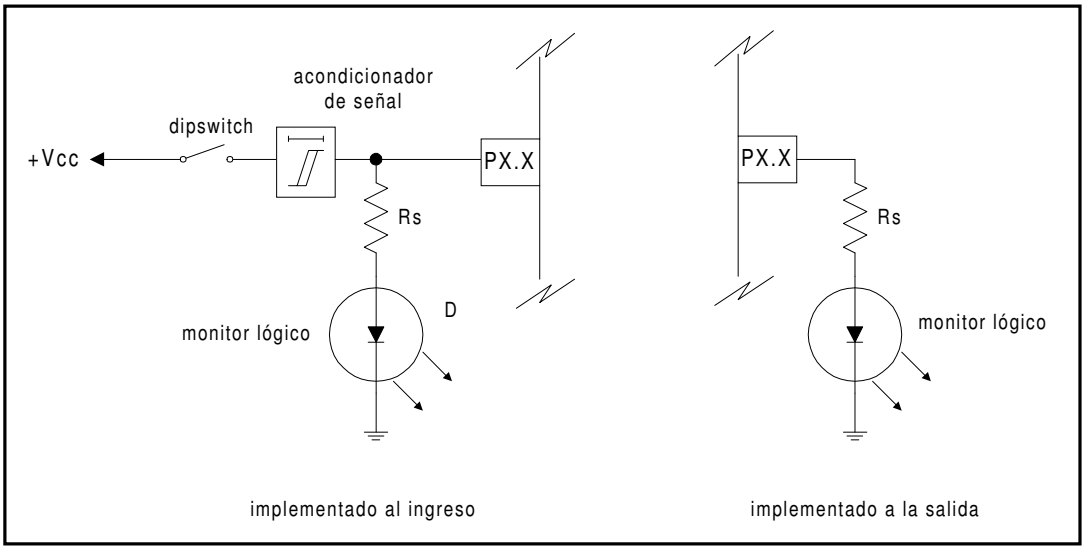
≅ Los circuitos de excitación básicos son:

Figura 56: Led como monitor lógico



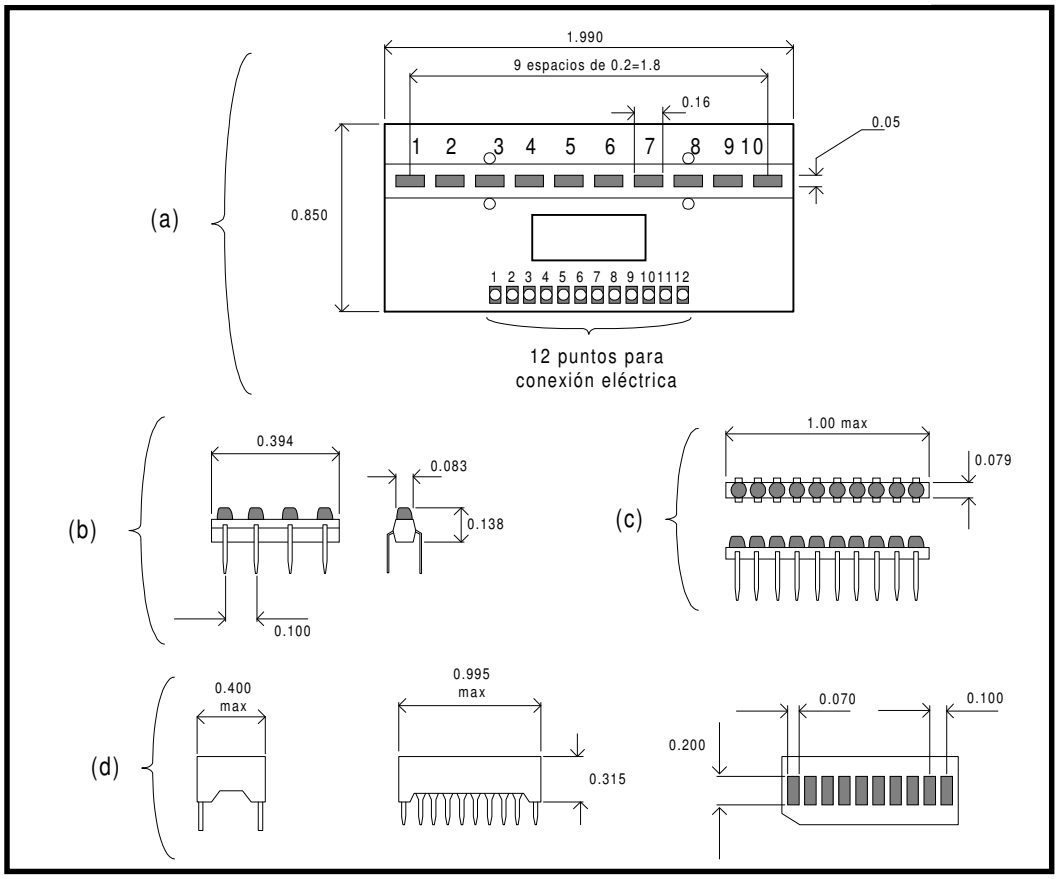
Entre tanto, para los circuitos integrados sería:

Figura 57: Monitor Lógico en un pin de puerto del microcontrolador

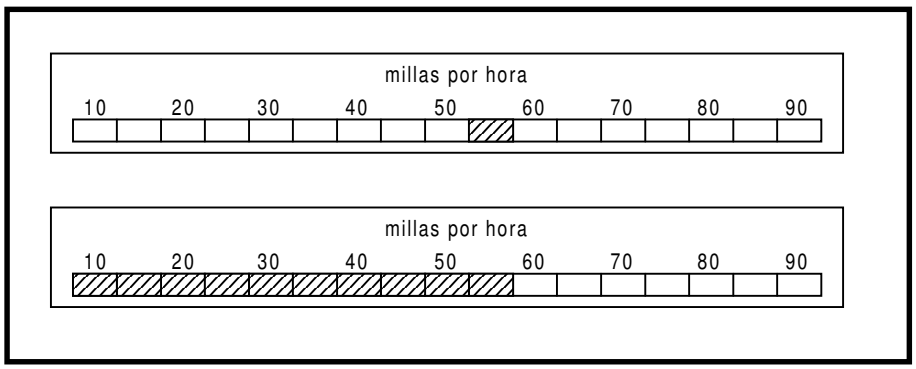


4.1.2 ***Arreglos de leds.-*** Al darse un arreglo a un grupo de leds, se obtiene:

Figura 58: Acomodo de leds en barras



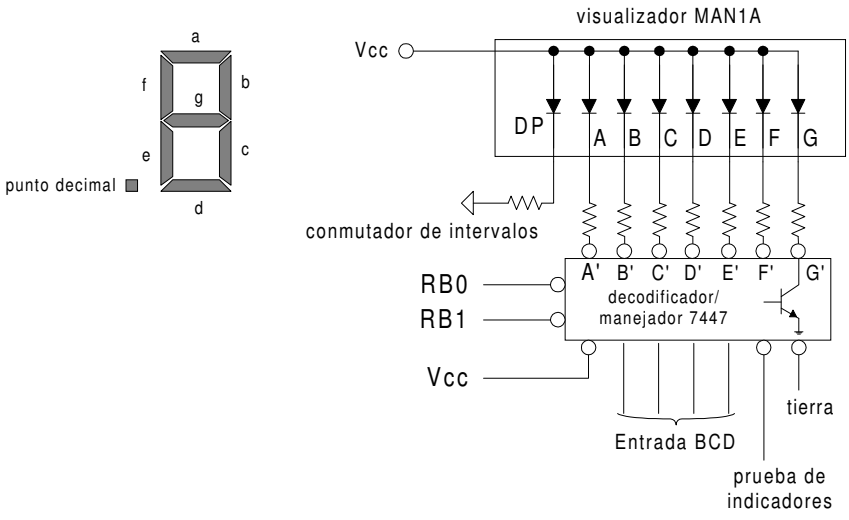
Ejemplos de encapsulados con varios led, a)módulo NSM3914 de National Semiconductor; b)BPX 84 de Litronix; c)TIL280 de Texas Instruments; d)MV57164 de General Instrument. Todas las dimensiones están en pulgadas.



Visualizador de barras para aplicaciones como velocímetro. a) Indicación puntual, en que el segmento prendido (visualizador de punto móvil) indica la posición; b) Indicación acumulada en que la posición se denota por el fin de la zona iluminada (visualizador de gráfica de barras).

- Displays

Figura 59: Leds en un display de 7 segmentos



Código BCD de entrada				Estado de salida del decodificador							Caracter de salida
2^3	2^2	2^1	2^0	A'	B'	C'	D'	E'	F'	G'	
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	1	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	1	1	0	0	9

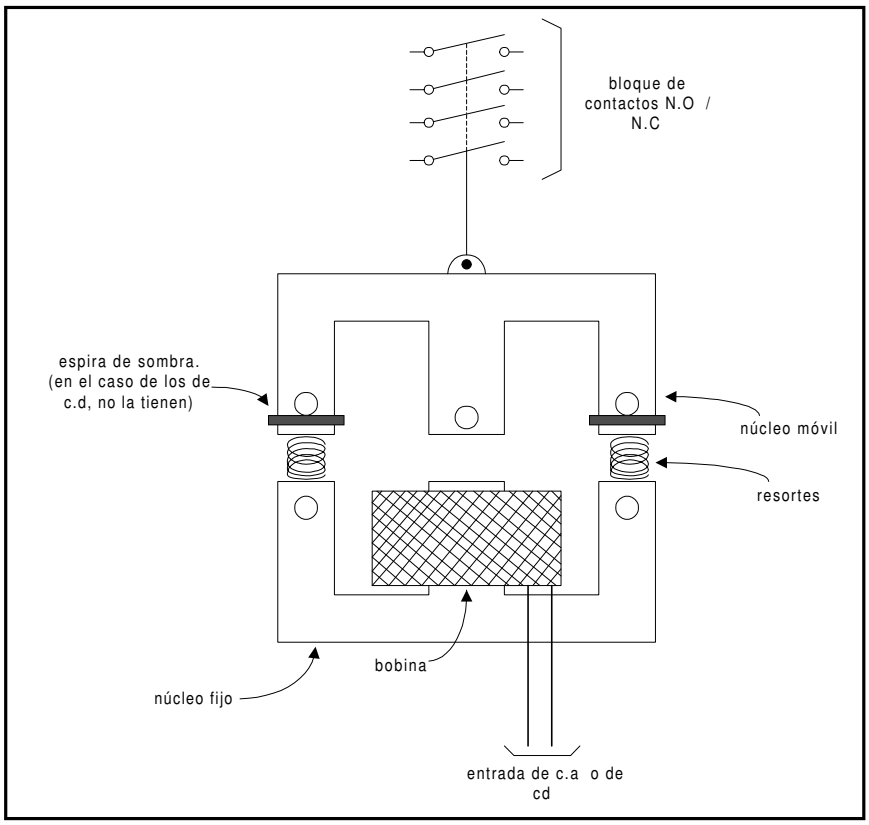
Conjunto de caracteres en un visualizador de siete segementos

4.2 Microrrelés.

Básicamente, un relé es un elemento de comando el cual permite el control a distancia, y entrar en un circuito como elemento auxiliar de contactos cuando se requiere. Esta conformado por un circuito electromagnético y bloques de contactos.

* *Circuito electromagnético*.- Consta de una bobina y un núcleo magnético:

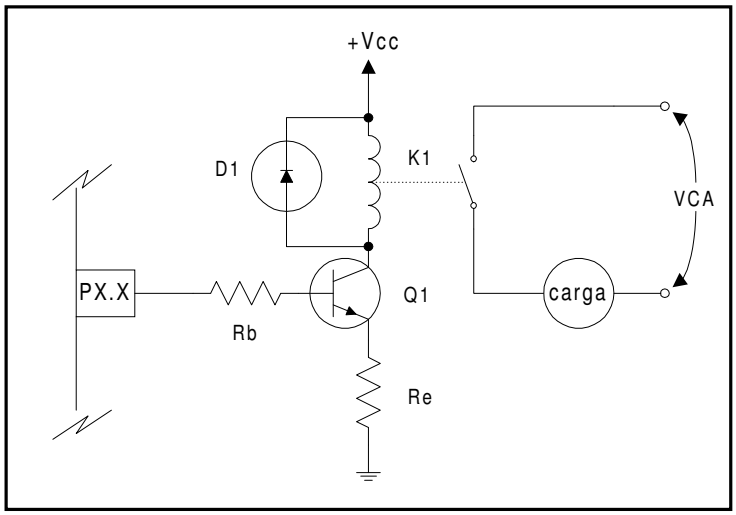
Figura 60: Partes de un contactor genérico



Al energizar la bobina se crea un campo magnético y, el magnetismo formado en la parte fija atrae a la parte móvil. A la vez el núcleo móvil, que es solidario con el bloque de contactos, arrastra al bloque de contactos cerrándolos y/o abriéndolos. Se los encuentra en los circuitos electrónicos en esencia, ya que permiten implementar un sistema de interfaz para el circuito de potencia.

Siendo los de cd los que se usan. Las tensiones que se encuentran en el mercado son: 5V, 6V, 12V, 24V. Para la interfaz dentro de una tarjeta electrónica, se emplea el siguiente método:

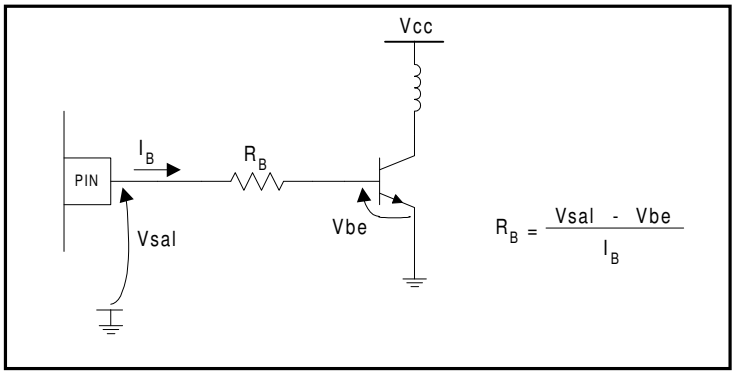
Figura 61: Interfaz de potencia en un pin de puerto



Entonces, el cálculo de la R_B es como sigue:

- La cantidad de corriente que puede entregar cada chip se la obtiene del databook correspondiente.
- La R_B debe asegurar que el V_{CE} sea igual a ‘0V”.
- Para proteger al transistor se debe colocar un diodo rectificador en polarización inversa, entre V_{CC} y el colector del transistor. Esto es para las sobrecorrientes producidas por la bobina del microrelé.

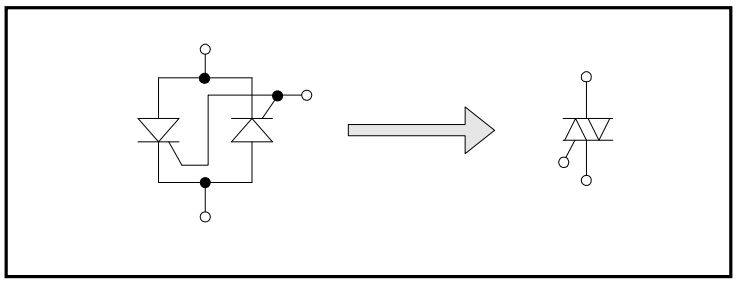
Figura 62: Polarización de un transistor en un pin de puerto



4.3 Semiconductor de Potencia.- TRIAC.

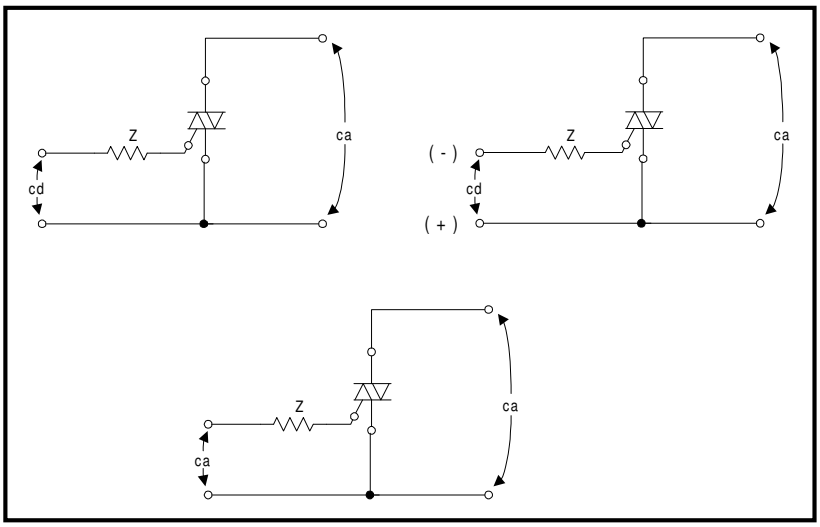
Esta dentro de los semiconductores de 4 capas, en primordialmente es un interruptor tríodo de ca. Su funcionamiento es semejante a dos SCR (*silicon controlled rectificator.- rectificador controlado de silicio*) conectados en antiparalelo:

Figura 63: Equivalente electrónico del triac



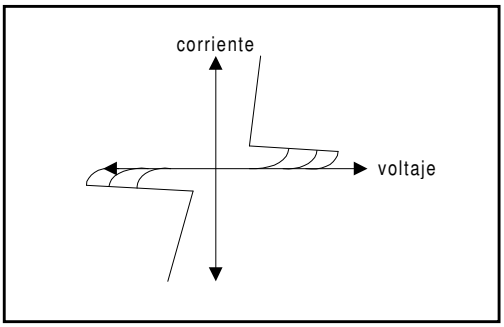
Dado esto puede conducir en cualquier polaridad de voltaje que exista entre sus terminales; y, se lo hace disparar con una señal en la compuerta (gate) de cualquier polaridad:

Figura 64: Métodos de disparo para un triac



Con esta particularidad se lo manipula en los tres cuadrantes:

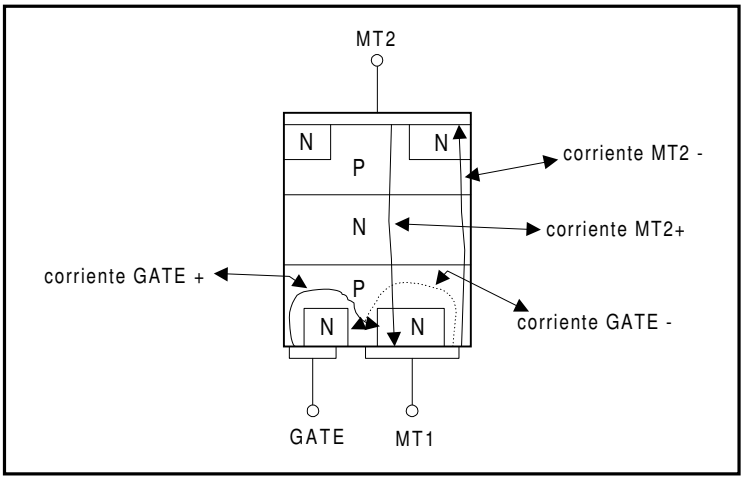
Figura 65: Curva Característica



Métodos de disparo: (*con respecto a MT1*)

1.- MT2 positivo, GATE positiva.
2.- MT2 positivo, GATE negativa.
3.- MT2 negativa, GATE positiva.
4.- MT2 negativa, GATE negativa.

Figura 66: Constitución interna de un triac

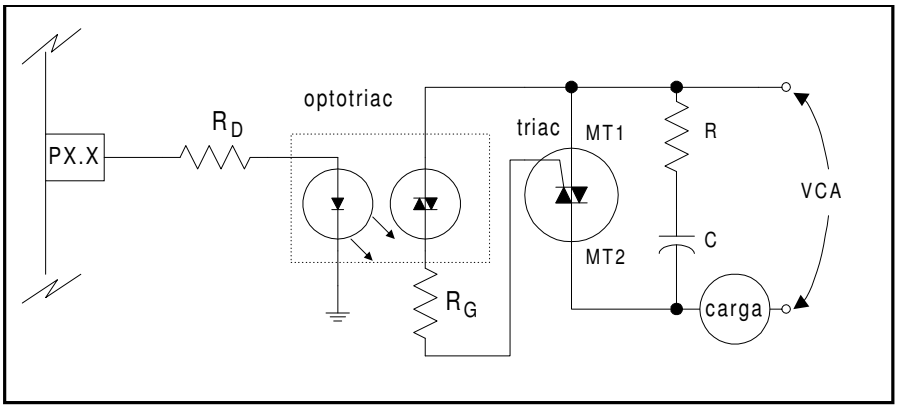


:

Se designa MT a las terminales debido a que el flujo de corriente es bidireccional; por lo que ánodo y cátodo es inapropiado.

Por lo general se emplea la terminal MT1 como la terminal de referencia de medición, por la interacción con el GATE.

Figura 67: Implementación de un SSR en un pin de puerto



4.4 Convertidores de Señales.- ADC y DAC.

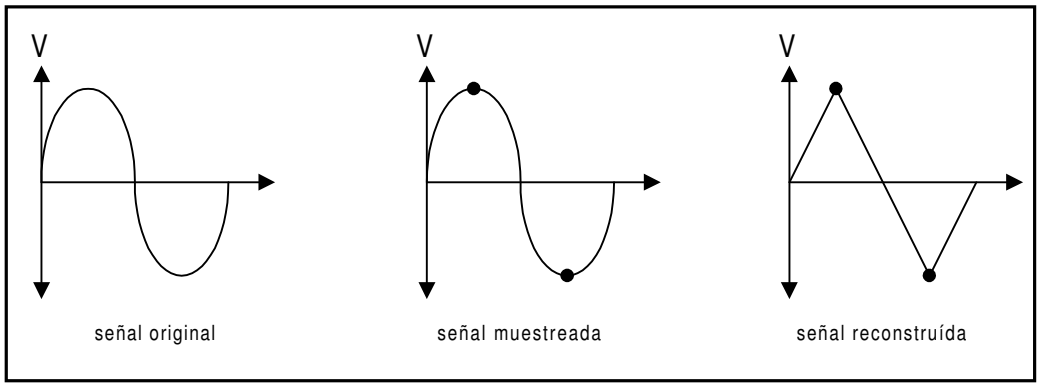
Dentro de la electrónica es necesario adquirir datos y luego mostrarlos acorde a señales que están constantemente variando, lentamente como la temperatura o, tan rápido como una señal de audio; constituyendo una señal análoga que en general es una magnitud cambiante en el tiempo.

Entre tanto los sistemas digitales como los computadores y microcomputadores utilizan señales fijas, es decir patrones binarios, para representar caracteres. Además, en forma binaria es más fácil realizar cálculos, manipular datos ordenadamente.

Por lo que se requiere de los convertidores de señal, tales como ADC y DAC. A continuación se los explica.

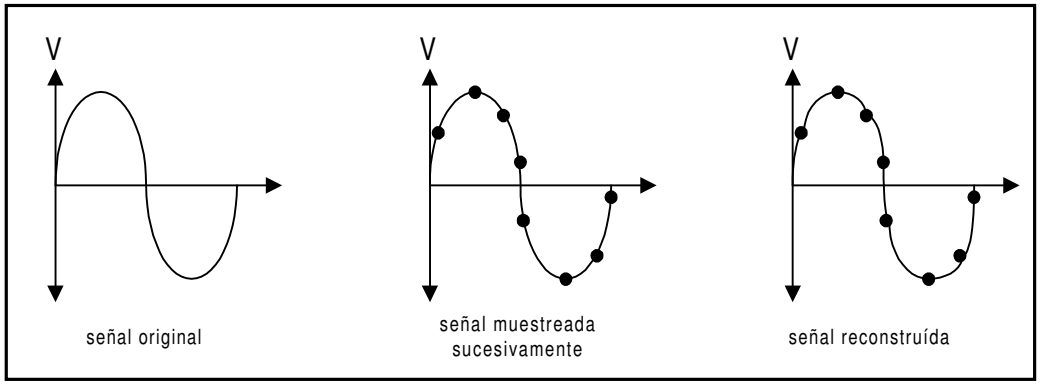
4.4.1 ***Convertidor Análogo / Digital.-*** Para transformar una señal análoga a digital se la tiene que hacer un muestreo sucesivo, por medio de un circuito que se dedica exclusivamente a esto, es decir:

Figura 68: Muestreo de una señal con 2 puntos de referencia



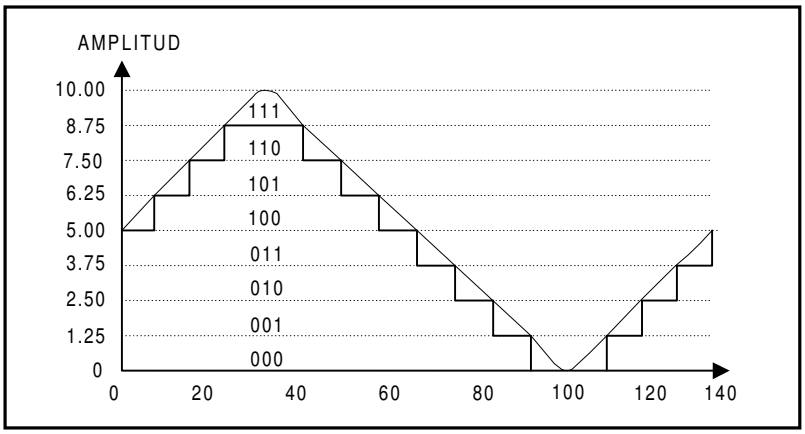
Según el ejemplo, las dos muestras realizadas permiten tener una señal distorsionada de la original; pero con un equivalente binario cada una. Para obtener una mejor idea de la señal original, se efectúan numerosos muestreos:

Figura 69: Muestreo de una señal con varios puntos de referencia



A cada valor muestreado de la señal de origen, se le asigna un número de bits que representa el valor del punto de muestreo:

Figura 70: Digitalización de una onda con 3 bits de resolución



La resolución de cualquier adc se la puede encontrar con 2^n , donde n es el número de salidas digitales; siendo ésta la razón del cambio en el voltaje de entrada que se requiere para cambiar un LSB en el byte de salida. Pero al conocer el voltaje de ingreso al adc en su totalidad, se puede determinar la resolución con:

$$\text{resolución} = \frac{\text{voltaje de ingreso total}}{2^n - 1}$$

De donde la ecuación de entrada – salida del convertidor es:

Código de salida digital = equivalente binario de D

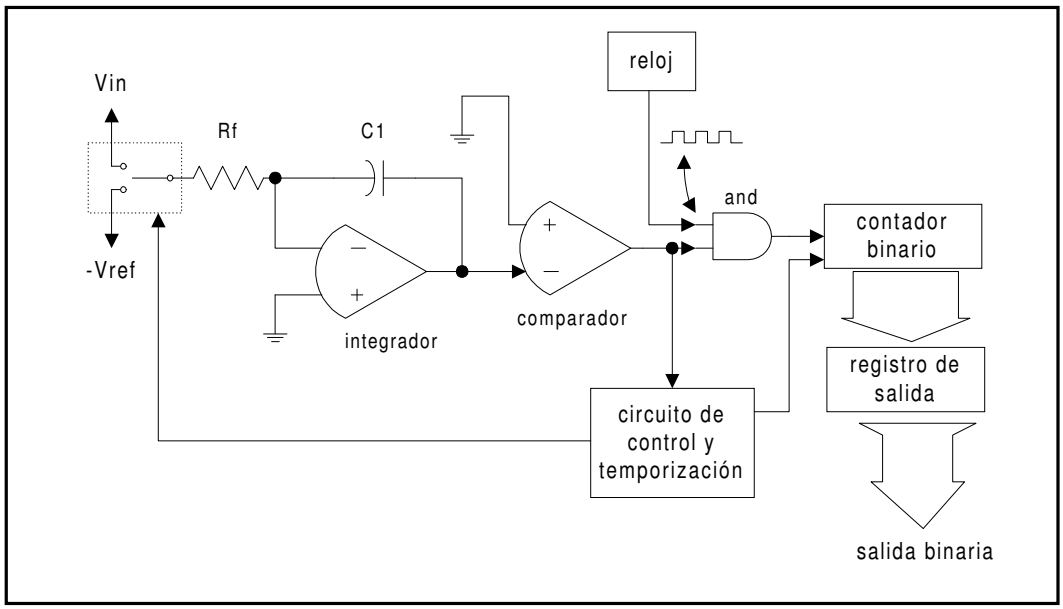
D = al equivalente decimal del número digital; determinándose así:

$$D = \frac{\text{voltaje de ingreso total}}{\text{resolución}}$$

Existen los siguientes convertidores:

4.4.1.1 *Convertidor de doble pendiente*.- "Este tipo de conversor tiene una buena estabilidad, pero muy mala velocidad; elimina el efecto del corrimiento de los voltajes de la rampa a lo largo del tiempo". *Cekit, 1992*.

Figura 71: Circuito demostrativo de un conversor de doble rampa

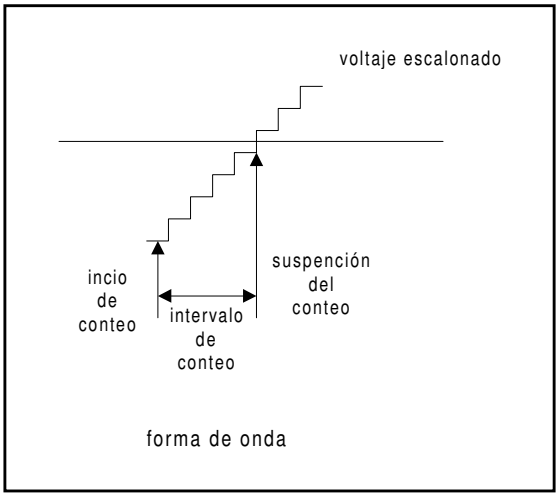


Al aplicarse un voltaje positivo la rampa crece negativamente a la salida del integrador; el comparador coloca a escala alta su salida que con el pulso alto del reloj se tiene un alto en la salida de la compuerta and, permitiendo que el contador avance.

"Luego de un tiempo fijo, se coloca un voltaje negativo en el integrador y a su vez "0L" en el contador. La constante de integración depende de los valores de Rf y C1, y de la magnitud de la señal de ingreso". *Boilestad, 1996.*

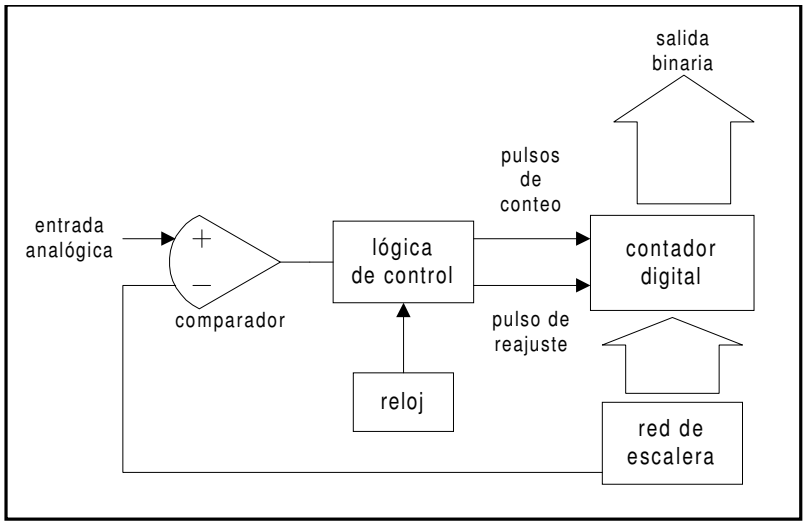
4.4.1.2 *Conversor en rampa*.- Un contador digital avanza a partir de cero mientras un circuito de escalera, manejado por un contador, da un voltaje de salida en escalones. "Que a la vez incrementa voltaje en cada paso del contador". *Boilestad, 1996.*

Figura 72: Forma de onda



El diagrama representativo de éstos convertidores es:

Figura 73: Diagrama lógico de un conversor de red escalera

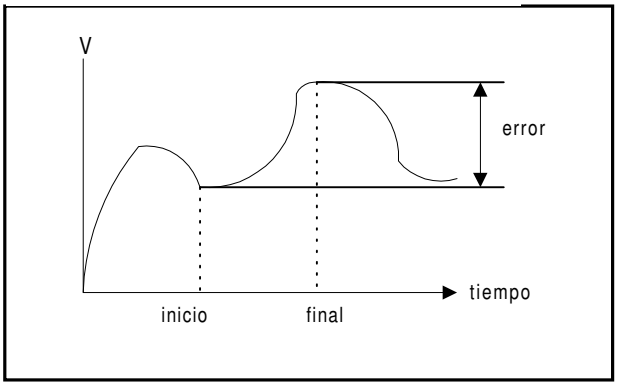


Ahora, la *velocidad de muestreo* debe ser por lo menos el doble de la frecuencia que maneja la señal análoga, conocida comúnmente como *frecuencia de Nyquist*.

El *error de cuantificación* se debe a la incertidumbre que se presenta en los períodos de conversión, dado por el tiempo de microsegundos a milisegundos que se requiere para convertir de análogo a digital un ingreso.

Se da que si la señal es CD no se producen errores; pero si es cambiante en el tiempo, el rango del error es proporcional a la amplitud de la señal.

Figura 74: Error de Cuantificación

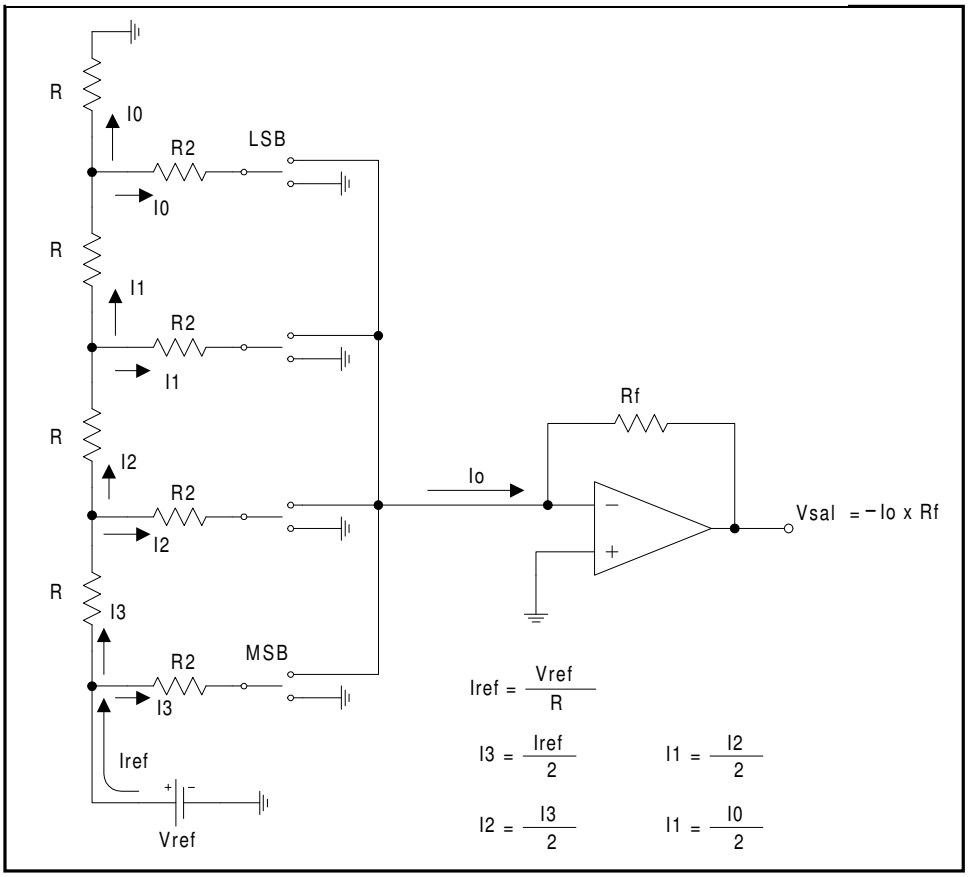


4.4.2 Convertidor Digital – Análogo.- El ADC convierte un código binario en un equivalente decimal compatible con una magnitud física. También, en conjunto con un ADC permite a un sistema operacional computarizado monitorear señales y presentarla a la salida del mismo.

Los convertidores ADC más utilizados son:

4.4.2.1 *Convertidores de Red Escalera R – 2R*.-

Figura 75: Circuito de un conversor R-2R



Al resolver las resistencias se halla que la malla resultante posee una resistencia R, la misma que sirve para determinar la Iref; luego las otras intensidades se las obtiene en función de ésta.

Ahora bien, como lo que ingresa al operacional es una corriente, se requiere una resistencia Rf para convertirla en un voltaje con la ecuación colocada en la salida. El voltaje de salida se lo calcula por:

$$Vo = \left(\frac{Vref}{R} \times \frac{1}{2^n} Rf \right) \times D$$

Vref = voltaje de referencia
RF = resistencia de referencia
D = valor decimal de la entrada digital
n = número de bits desde el MSB al LSB

Vo = voltaje de salida

Se puede distinguir lo siguiente:

- La resolución esta relacionada con el número de niveles análogos de voltaje que puede generar, es decir con el número de bits de la palabra binaria de ingreso.
- El tiempo de estabilización es el tiempo que requiere la salida análoga para estabilizarse después que la palabra binaria aparece en la entrada.
- Exactitud es la variación (+/-) desde la mitad hasta dos veces el valor de 1 LSB. Mientras mas pequeña sea, mejor será la salida.

4.5 Memorias.- EPROM y RAM.

Las memorias RAM son de acceso aleatorio, y pueden ser dinámicas o estáticas. Las SRAM son aquellas constituidas por un flip – flop que actúa como unidad básica de almacenamiento.

Esta es a partir de transistores MOS, al instante de cortar el suministro de voltaje se pierde el dato binario almacenado.

Las DRAM usa como unidad básica de almacenamiento un condensador, como la carga del condensador es muy pequeña se pierde rápidamente, aproximadamente en pocos milisegundos. Entonces se debe refrescar a la celda de memoria cada dos milisegundos aproximadamente.

Figura 76: Constitución de una memoria DRAM

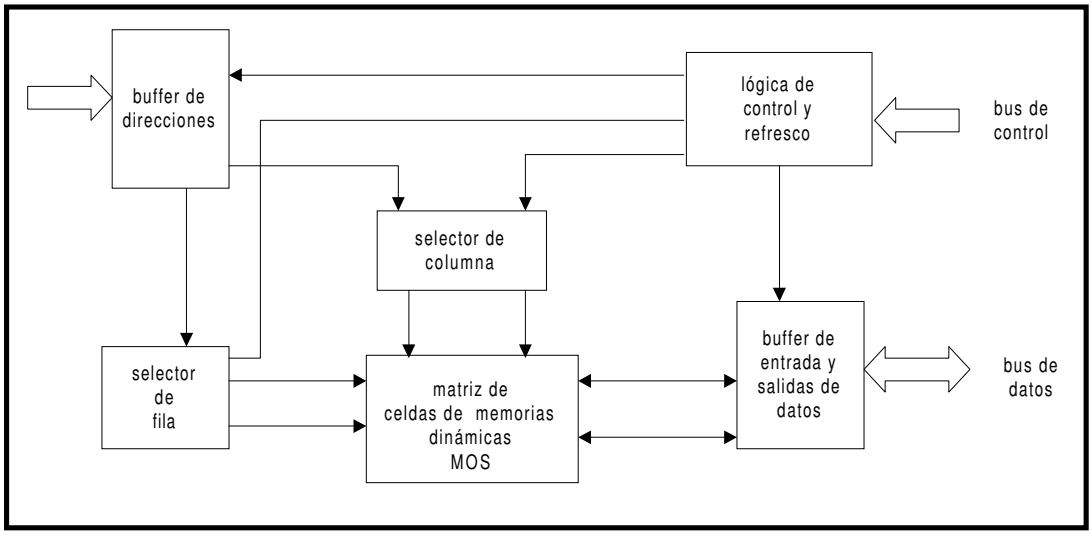
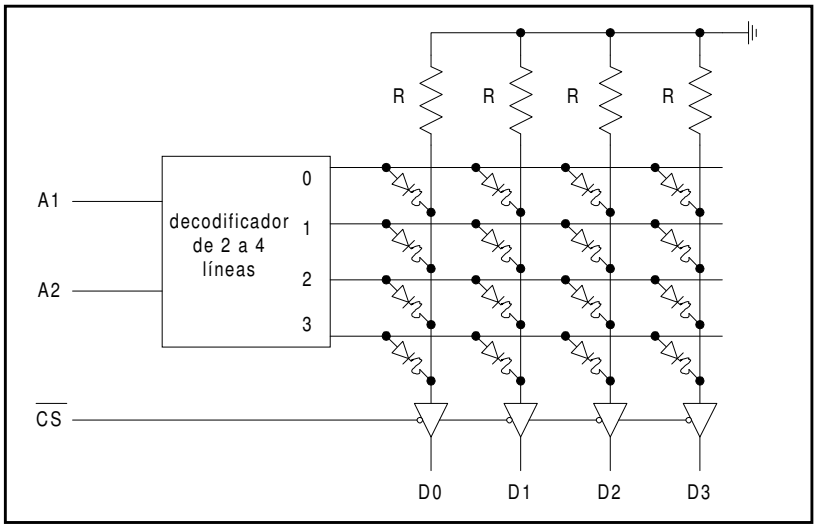


Figura 77: Estructura de una memoria ROM



Al instante de programación se vuelan los fusibles que son después del diodo, este tipo de ROM son programables una sola vez. Pero tienen una subdivisión que permite tener memorias de lectura programables:

∝ Memorias EPROM: memoria de solo lectura programable eléctricamente. Estas se subdividen en:

- UV – EPROM = borrables por radiación ultravioleta
- EEPROM = borrado por medio de electricidad.

Estas memorias no pierden su contenido, considerándolas no volátiles; aún al quedarse sin energía.

Las constituyen:

Bus de datos: es en donde se presenta la información almacenada en la posición de memoria definida.

Bus de direcciones: por medio de estas se ubica la posición de memoria que se desee.

Bus de control: constituye las señales de control que requiere para poder operar.

4.6 Compuertas Schmitt – Trigger.

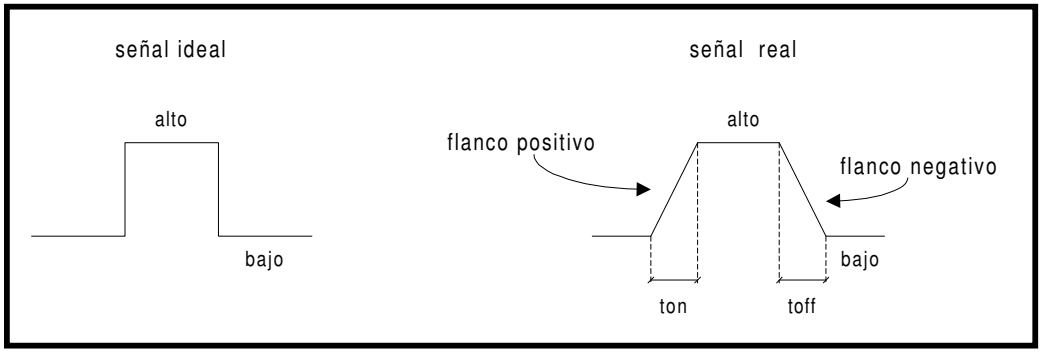
Este tipo de compuertas lógicas se las emplea para convertir señales lentas, imperfectas o con ruido en señales digitales altamente confiables y con una margen de ruido muy reducido. Entre tanto el trabajo que realiza, ya sea AND – NOT, no se altera en lo absoluto.

Entonces se puede definir que los circuitos digitales operan de forma más eficiente cuando las señales que tratan son totalmente digitales, es decir cuadradas.

El principio de operación es:

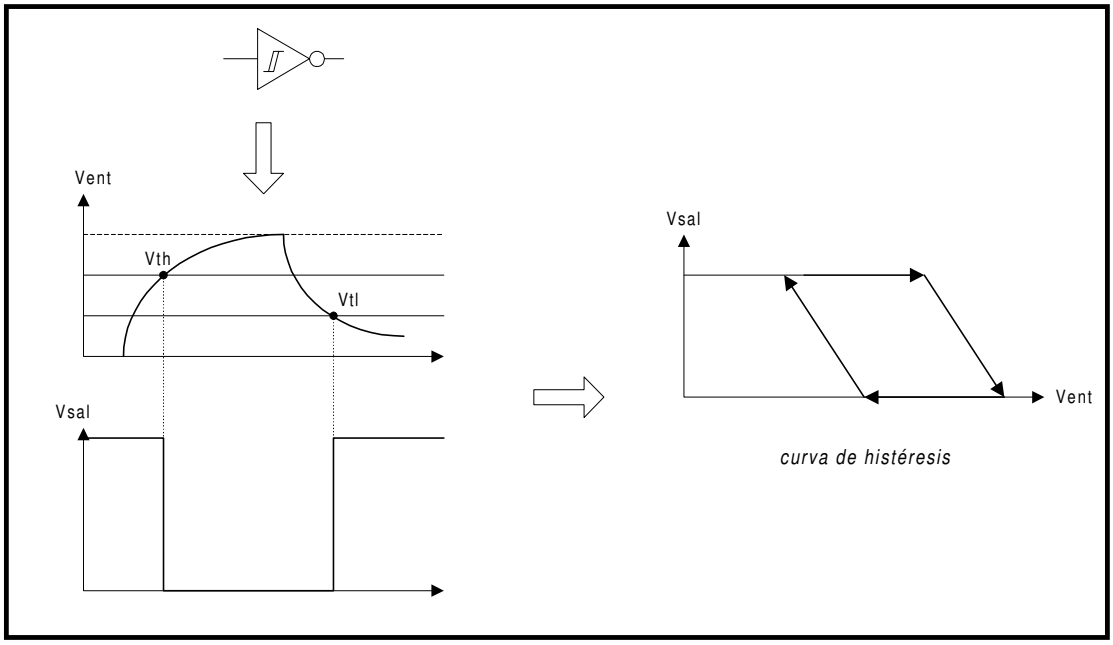
♠ Al producirse una transición de estado, por lo general admite un tiempo de subida o un tiempo de descenso; o la transición es demasiado lenta. Esto podría dar lugar a falsa señales dando inestabilidad al sistema. De la misma manera ocurre si la señal de ingreso no es una onda cuadrada, o si ésta tiene ruido.

Figura 78: Relación de retardo en una onda



Por esto éstas compuertas operan con el principio de *histéresis*, es decir, que las salidas responden a las entradas cuando los valores superan cierto *umbral*.

Figura 79: Ondas de operación de una compuerta schmitt-trigger



+ Los valores más comunes para los integrados TTL son: VTH = 1.6V y VTL = 0.8V.
+ Los valores más comunes para los integrados TTL son: VTH = 5.8V y VTL = 3.8V.

4.7 Registros.

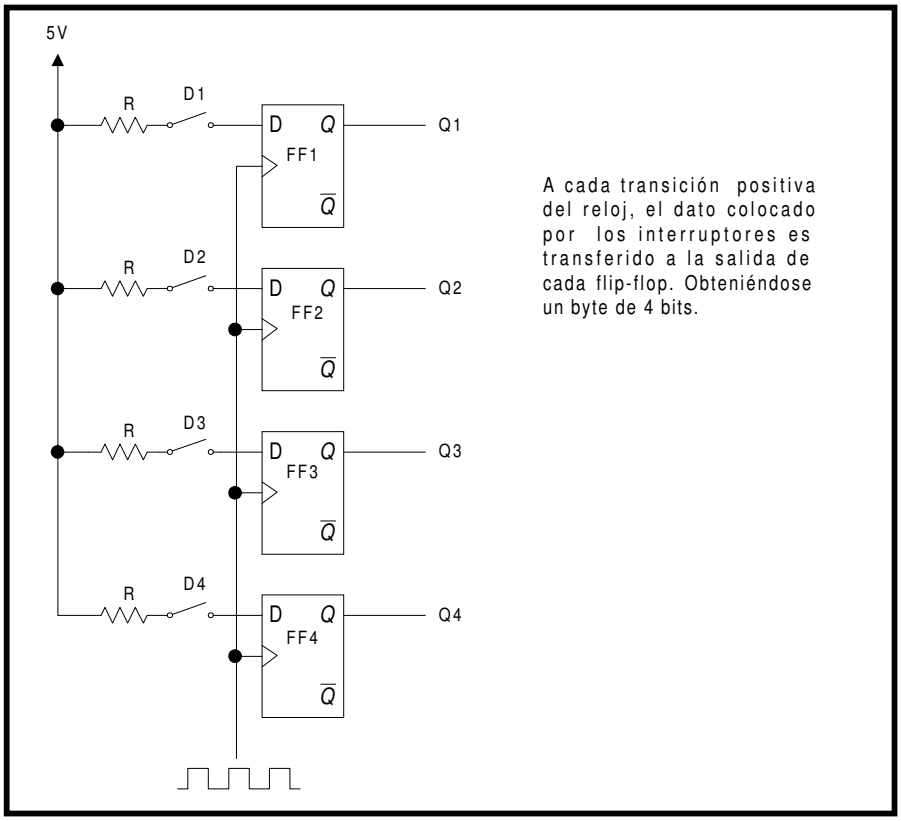
Son basados en las celdas de memoria básicas: *FLIP – FLOP tipo D*; entre los cuales pueden ser de almacenamiento o de desplazamiento. Los registros de almacenamiento se

usan para capturar datos por un tiempo determinado, y los de desplazamiento permiten la recirculación de determinada palabra binaria.

4.7.1 *Registros de almacenamiento.*

También llamado *registro paralelo*, de acuerdo a un arreglo determinado de flip-flops tipo D se logra capturar una serie de bits para luego transferirlos a un sistema en forma de byte. Pudiendo ser una byte de 8 – 16 – 32 y 64 bits.

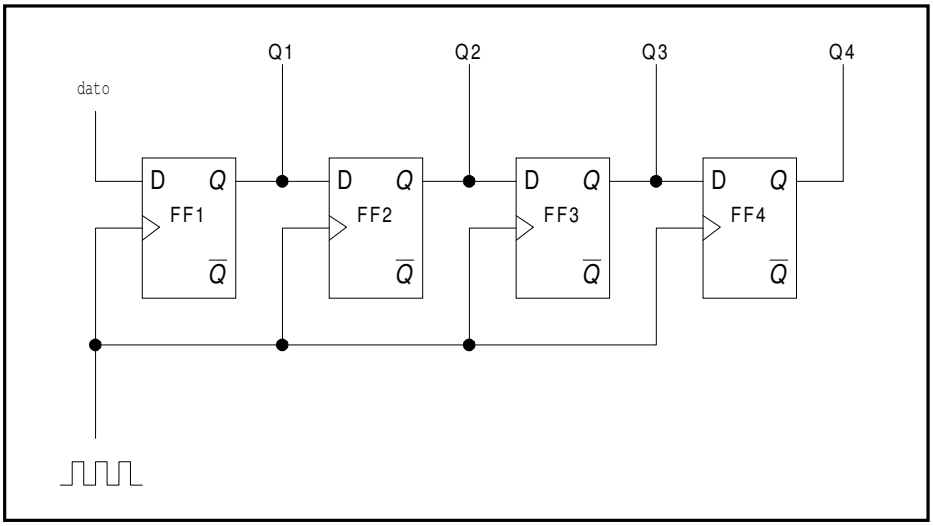
Figura 80: Registro paralelo de 4 bits



4.7.2 *Registros de desplazamiento.*

Su función es de retardar una información, convertir un dato paralelo en serial. Esto se logra transfiriendo el byte en forma de bit a bit con cada transición de reloj adecuada. Los cuales presentan una configuración típica de:

Figura 81: Registro de desplazamiento de 4 bits



Dentro de los registros de desplazamiento se tiene:

- *SISO*: entrada serie – salida serie.
- *SIPO*: entrada serie – salida paralela.
- *PISO:* entrada paralela – salida serie.
- *PIPO:* entrada paralela – salida paralela.

4.8 Optoacopladores.

Es la combinación de una fuente de luz infrarroja (led) con un medio foto sensible (semiconductor de silicio), también conocidos como optoaisladores. Pudiendo ser:

- Optotransistores,
- Optotriacs,
- Optoscr,
- Optodarlington, etc.

Dentro de los parámetros de los optoaisladores se tiene:

1.- *Potencia de salida irradiada, intensidad irradiada y sensibilidad luminosa*.- La potencia de salida irradiada en vatios expresa la salida de los leds infrarrojos; la intensidad irradiada también dice la salida del led y se mide en estereorradián (w/st), siendo el flujo a través de un ángulo sólido unitario.

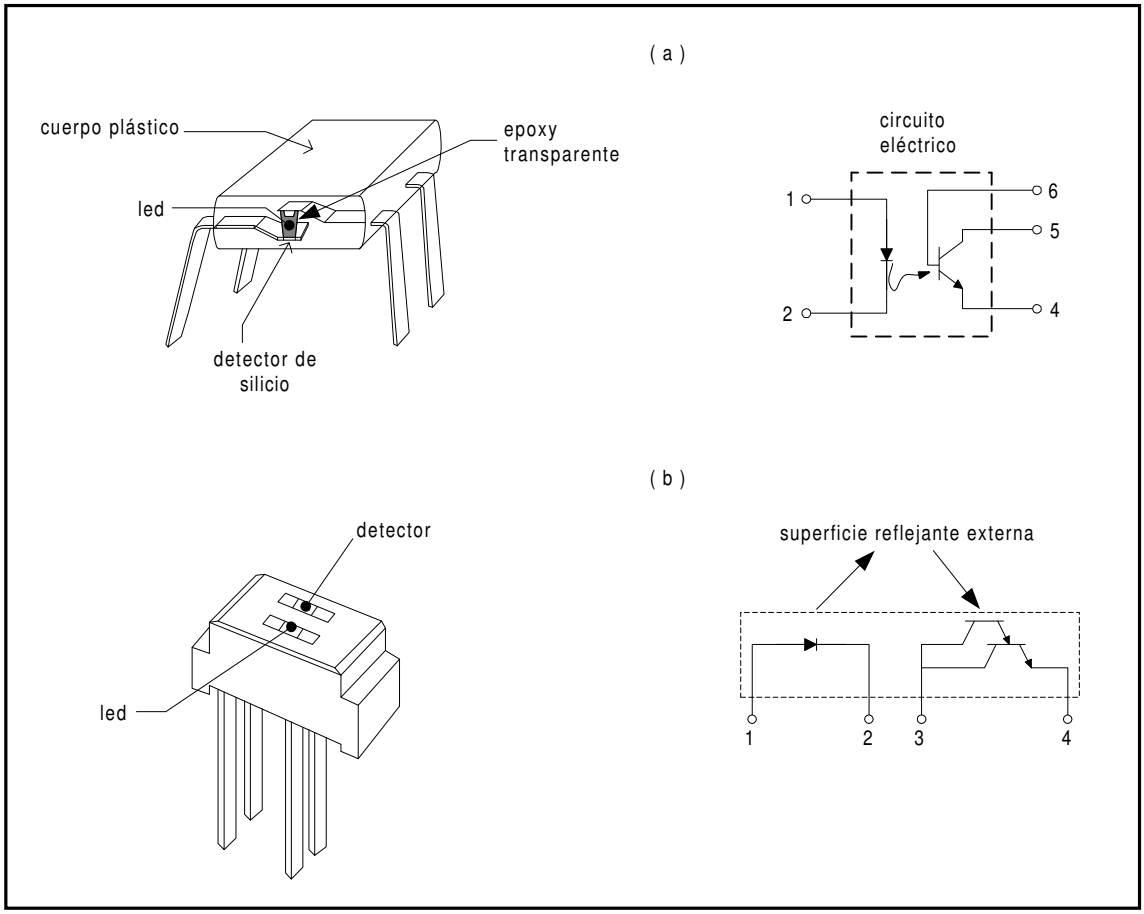
2.- *Corriente obscura: I_{CEO}*.- Corriente de salida en ausencia de una fuente luminosa. Es decir con el led apagado, y debe ser muy pequeña para que no active al optoacoplador.

3.- *Razón de transferencia de corriente*.- Es el cociente de la corriente de salida del optoacoplador y la corriente de entrada al led.

4.- *Resistencia y voltaje de aislamiento*.- Verifican la capacidad de aislamiento eléctrico entre la entrada y la salida del optoacoplador. Siendo el voltaje de aislamiento el voltaje máximo que aplicado a las terminales cortocircuitadas del led de entrada y las terminales de salida, también cortocircuitadas, no genera circulación de corriente ni daña el elemento.

5.- *Velocidad de conmutación y retardos de respuesta*.- Indican el comportamiento dinámico de salida del optoacoplador respecto al pulso de corriente aplicado al led de entrada.

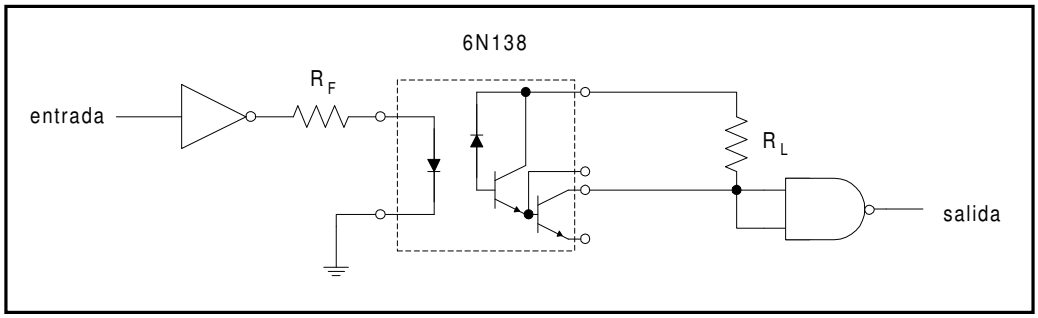
Figura 82: Encapsulados de optoaisladores; a)Típico con vías de transmisión luminosa internas (MCT2); b)Típico con vías de transmisión de luz reflejada (MCA7)



Pueden emplearse en:

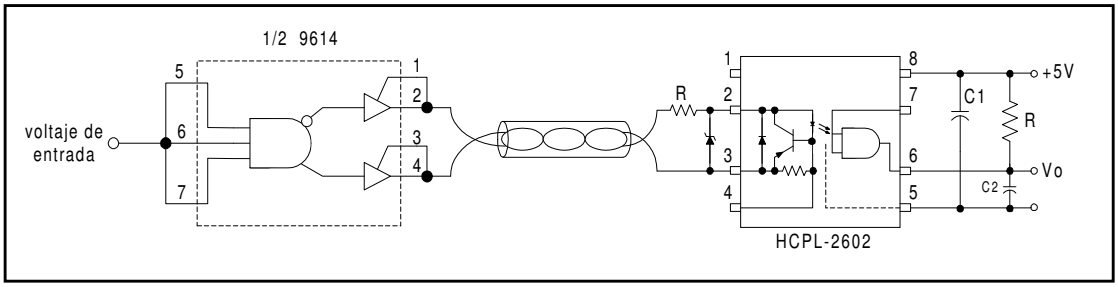
° Interfaz lógica a lógica:

Figura 83: Circuito de interfaz logica



° Interfaz de circuitos lineales:

Figura 84: Circuito de interfaz lineal



4.9 Puertos de Interfaces

Al momento de elaborar un sistema computarizado para asumir tareas complejas y no complejas, se requiere que éste tenga comunicación con el mundo externo. Captando todas

aquellas señales que se generan tal cual son, con los mismos valores, para tan pronto como el sistema lo interprete reaccione según el programa establecido.

Dada ésta razón, los PC compatibles tienen varios puertos de entrada/salida de datos, los que permiten que tener una versatilidad de movimiento al momento de armar un periférico. Por lo general se tiene:

- Puertos paralelos,
- Puertos seriales,
- Puerto para el teclado,
- Buses de expansión, y eventualmente,
- Puerto para joystick.

Como en éste trabajo se prioriza el uso de los puertos serial y paralelo, serán éstos los tratados si no se especifica ninguna otra cosa. Las posiciones de dirección que en el PC maneja cada puerto son:

Tabla 38: Puertos de la PC

Puerto Paralelo	Puerto Serie
LPT1 3BC - 3BF	COM1 3F8 - 3FF
LPT2 378 - 37F	COM2 2F8 - 2FF
LPT3 278 - 27F	COM3 3E0 - 3EF
	COM4 2E8 - 2EF

4.9.1 Puerto Paralelo

Es el principal puerto para todo tipo de aplicaciones por su facilidad de manejo, requiriendo un mínimo de hardware externo para operar, además, sus datos son fácilmente captados. Los niveles de operación del puerto son TTL, con una corriente máxima de salida de 40mA.

A raíz de la norma IEEE 1284 que estandarizó, la cual definía 5 modos de operación, siendo:

1. Modo Compatible, *con Centronics o modo Standar*.
2. Modo Nibble, *ingresa 4 bits en un tiempo por las líneas de estado*.
3. Modo Byte, *ingresa 8 bits por las líneas de datos, llamado también bidireccional*.
4. Modo EPP, *Puerto Paralelo Enganchado, es de comunicación bidireccional para CD-ROM, discos duros, adaptadores de red.*
5. Modo ECP, *Puerto Compatible Extendido, es de comunicación bidireccional para la nueva generación de scanners, impresoras*.

Como el diseño original del puerto fue para comunicarse con impresoras, en la descripción puede notarse ésta situación:

STROBE (pin 1).- Activa en nivel bajo, indica la presencia de datos en D0 –D7 para que la impresora los lea.

D0 – D7 (pin 2 – 9).- Es el bus de datos para expresar el carácter enviado. Es de salida solamente. Si es tipo ECP/EPP se puede escribir en él.

ACKNOWLEDGEMENT (pin 10).- La impresora coloca a 0 para indicar al PC que ha recibido correctamente el dato, y que puede continuar la transmisión.

BUSY (pin 11).- La impresora la pone a 0 para indicar que el buffer está lleno. Entonces el PC para el envío, y lo reanuda al colocarse la línea en 1.

PAPER EMPTY (pin 12).- La impresora indica que se ha quedado sin papel.

SELECT / ON-LINE (pin 13) .- Indica el PC si la impresora está en línea o no.

AUTOFEED (pin 14).- Al estar en 1, la impresora realiza un salto de línea con cada carácter <return> recibido.

ERROR (pin 15).- La impresora indica al PC que ha detectado un error.

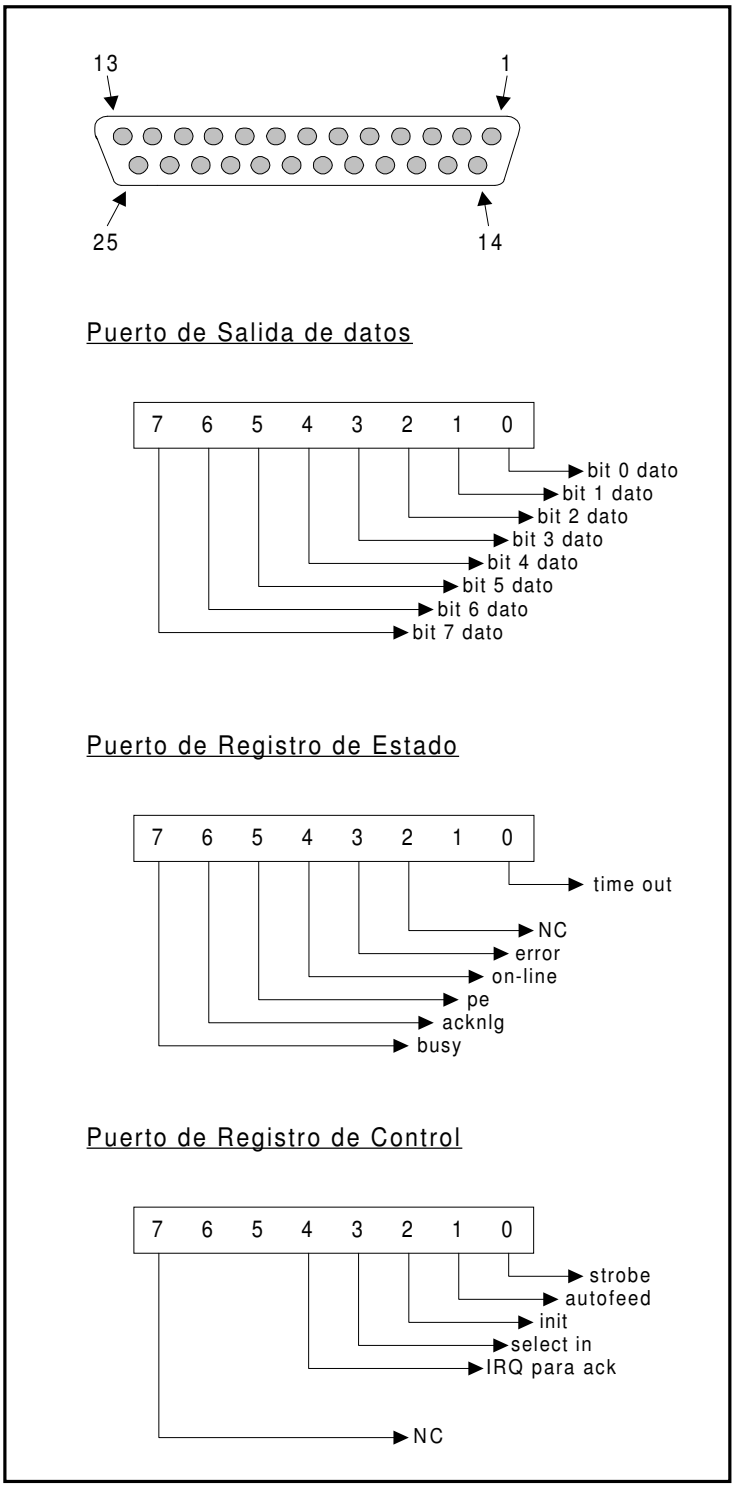
INIT (pin 16).- El PC inicializa la impresora.

SELECT IN (pin 17).- El PC puede poner fuera de línea a la impresora.

GND (pin 18 – 25).- Es la tierra del PC.

Para trabajar con el puerto paralelo, se maneja los siguientes registros para el LPTx:

Figura 85: Puerto paralelo: pinaje y registros



4.9.2 Puerto Serie

Este puerto resulta un poco más difícil manejarlo debido a que necesita de un hardware en donde se permita capturar el tren de bits que corresponde a la transmisión / recepción. Pero su uso resulta más potente y universal; en la PC por lo general vienen 2: COM1 y COM2, el primero de 9 pines y el segundo de 25 pines, pero con las mismas funciones.

Algunas veces para manejarlo se requiere la transformación del byte de dato de paralelo a serie, por lo que se emplea un UART. De igual manera, se requiere manejar ciertos registros para atender su pedido.

Dentro de las ventajas de emplear éste puerto tenemos, entre otras:

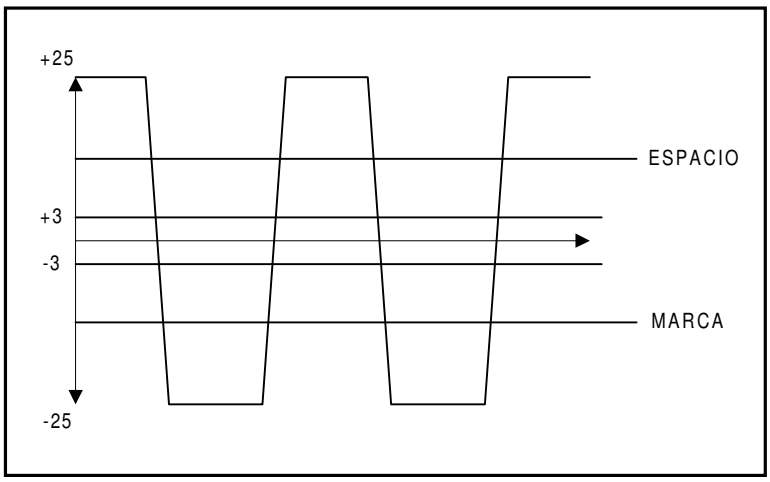
1. Los cables seriales suelen ser más largos que los paralelos.

2. No se necesita demasiados cables para su comunicación.

3. Se emplea en la comunicación con los microcontroladores de reciente aparición.

El puerto del PC emplea para la comunicación niveles de tensión, en donde las señales deben manejar rangos definidos para la interpretación. Estos rangos son:

Figura 86: Valores de voltaje para el puerto serial



Las señales que están entre +3 a +25 comprenden un ESPACIO (0 lógico); y las señales que están entre –3 y –25 comprenden una MARCA (1 lógico). Mientras que la región entre +3 y -3 es indefinida.

Se tendrá en cuenta que:

- El voltaje del circuito de salida nunca excederá los 25 voltios, referidos a tierra.
- Un cortocircuito no excedería los 500mA.

Los conectores tanto machos como hembras tienen el siguiente pinaje:

Figura 87: Pinaje del puerto serial

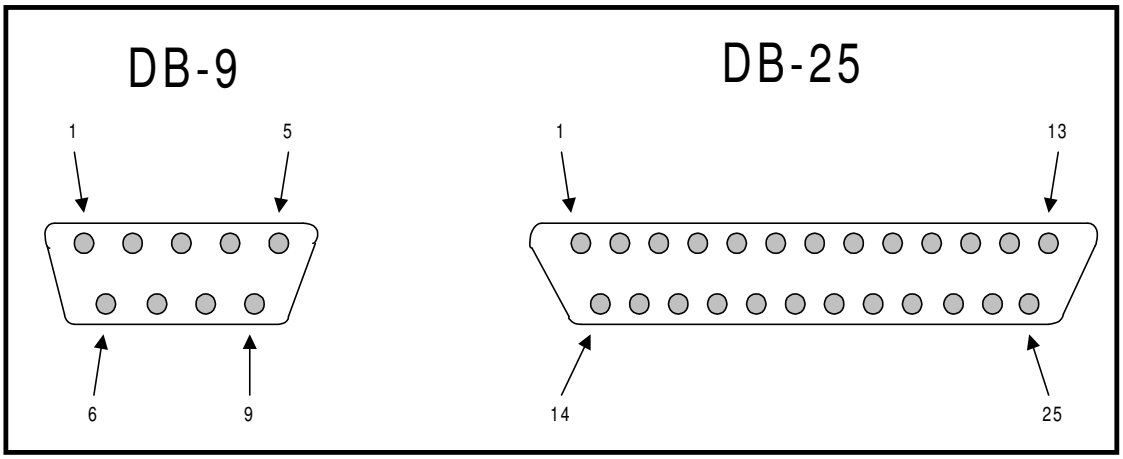
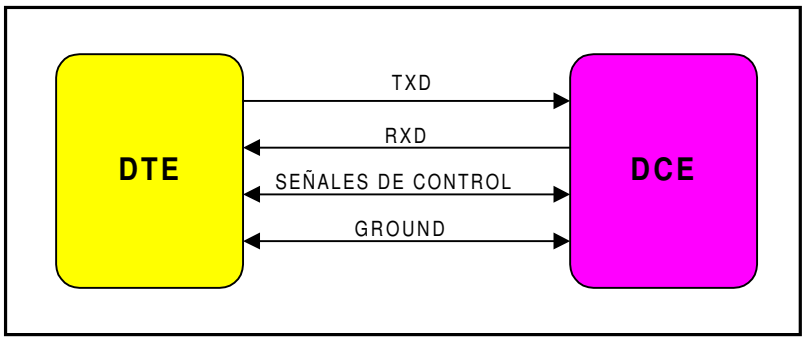


Tabla 39: Descripción del puerto serial

PIN - 9	PIN - 25	DESCRIPCION	
1	8	DCD / CD	detección de la portadora
2	3	RXD / RD	entrada de datos
3	2	TXD / TD	salida de datos
4	20	DTR	data terminal ready: puerto listo
5	7	GND / SG	referencia de tierra a 0v
6	6	DSR	data set ready: listo para recibir datos
7	4	RTS	request to send: petición de envío de datos
8	5	CTS	clear to send: indica deseo de transmisión
9	22	RI	ring indicator: anuncia otra llamada al otro dispositivo
0	9 / 19		no empleados
0	1		masa del chasis

Para la conexión con un dispositivo externo al PC, se maneja lo siguiente:

Figura 88: Enlace de un DTE y un DCE



En donde el DTE es el "Data Terminal Equipment" (*Equipo Terminal de Dato*), que puede ser un computador o un terminal. Mientras que el DCD es el "Data Communications Equipment" (*Equipo de Comunicación de Datos*), que puede ser el MODEM, plotter,etc.

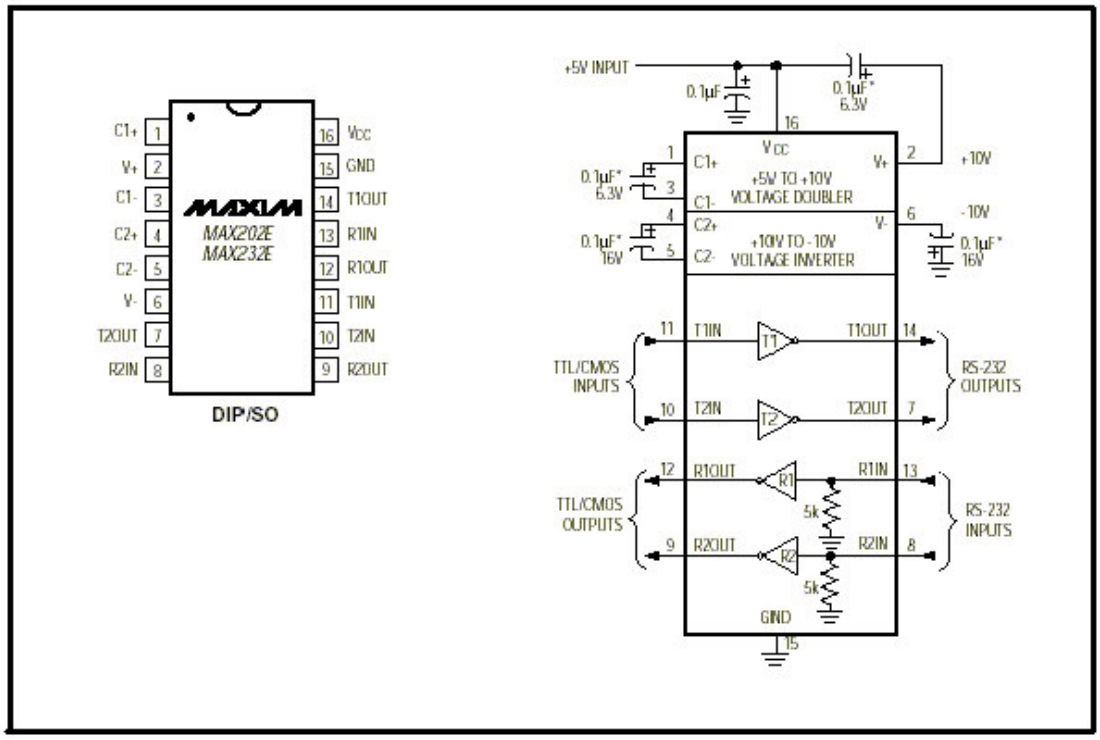
4.9.2.1 Protocolo de Comunicación RS-232

Como el ordenador opera con los rangos antes descritos de voltaje, se requiere de un protocolo de comunicación con el exterior. Esto se logra con el chip RS-232, que permite acoplar lógica TTL / CMOS con el puerto serial.

El RS-232 transforma los niveles lógicos a los voltajes que maneja el puerto en intervalos de tiempo muy pequeños, permitiendo al ordenado actuar de acuerdo a variables externas.

La configuración del chip y su conexión es:

Figura 89: Pinaje y conexión eléctrica del RS-232



El margen del valor de los capacitores no suele ser crítico, pero se puede emplear de 1µF o de 10µF. Los valores de entrada al PC son en el orden de los 10 voltios aproximadamente, que es una tensión valida según los rangos establecidos.

PRACTICAS DE COMPLEMENTACION

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 1

Nombre: _____ Curso: _____ Calificación: _____
Fecha: _____

1 Tema: *Manejo del Software Ensamblador*

2 Objetivos:

2.1 Aprender a operar el Ensamblador para crear el "archivo.LST" y "archivo.HEX.
2.2 Demostrar la operación del Emulador para la verificación del programa.

3 Materiales y Equipos:

1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

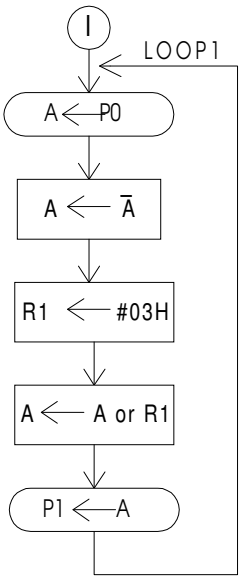
5 Procedimiento:

5.1 Insertar el diskette y abrir el Ensamblador en el PC:

- ¬ Elegir MS-DOS en el PC,
- ¬ Salir de Windows con "cd..",
- ¬ Colocar el drive "A: \edit".

5.2 Escribir el siguiente programa:

FLUJOGRAMA INSTRUCCIONES




```
LOOP1:  MOV     A,P0
        CPL     A
        MOV     R1,#03H
        ORL     A,R1
        MOV     P1,A
        SJMP    LOOP1

        END
```

5.3 Guarde el programa colocándole un "Nombre.A51".

5.4 Cierre y salga del ensamblador; y, coloque:

A:\ASEM NOMBRE.A51

5.5 Ingrese al EDITOR para revisar el resultado del ensamblado, tipeando una de las dos condiciones:

A:\EDIT NOMBRE.LST
A:\EDIT NOMBRE.HEX

5.6 Cierre el EDITOR, y abra el simulador previa selección del tipo de microcontrolador; luego cárguelo:

A:\AVSIM51

⊗ tipo de micro

⊗ Load

⊗ Program: NOMBRE.HEX

- Con las teclas F9 y F10 se puede mover el programa.
- Con la tecla ESC selecta el menú de condiciones del simulador.

6 Conclusiones y Recomendaciones:

7 Evaluación:

7.1 Los controles ensambladores primarios y generales sirven para:

7.2 Describir como se ingresa un programa para compilarlo:

7.3 Indique el proceso para simular un programa:

7.4 Basándose en el programa demostrativo, elabore uno y simúlelo:

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 2

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Movimiento de datos en las posmem del microcontrolador.*

2 Objetivos:

2.1 Comprender y demostrar los modos de direccionamiento del microcontrolador.
2.2 Aprender a leer y/o escribir en los puertos.

3 Materiales y Equipos:

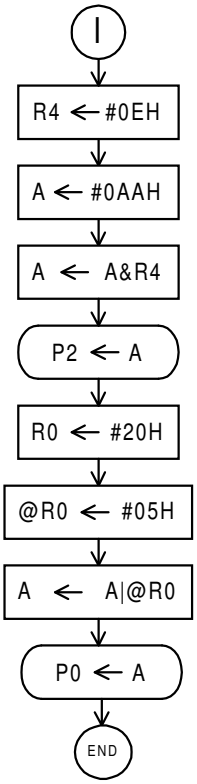
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa:

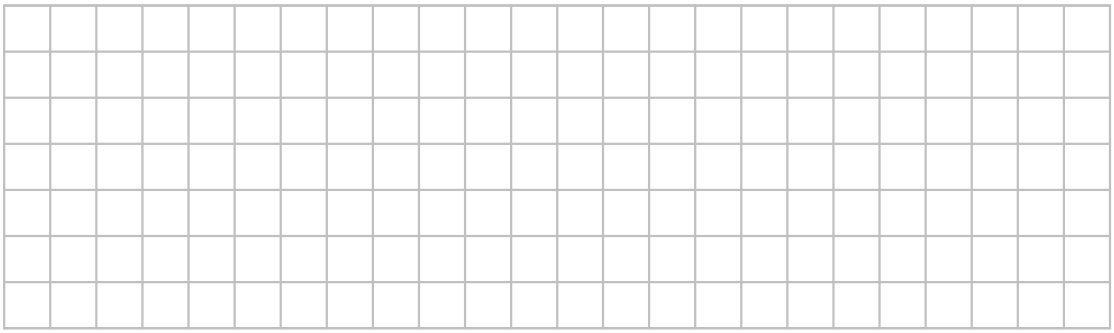
FLUJOGRAMA INSTRUCCIONES



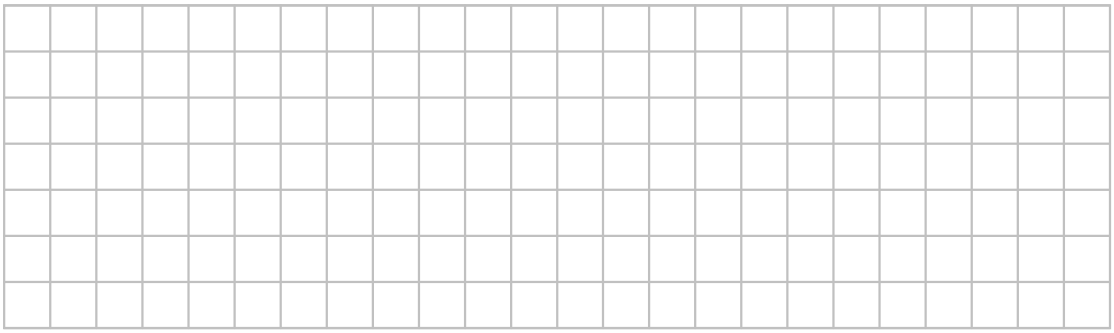

```
MOV     R4,#0EH
MOV     A,#0AAH
ANL     A,R4
MOV     P2,A
MOV     R0,#20H
MOV     @R0,#05H
ORL     A,@R0
MOV     P0,A

    END
```

5.1.2 Describir la tarea del programa.

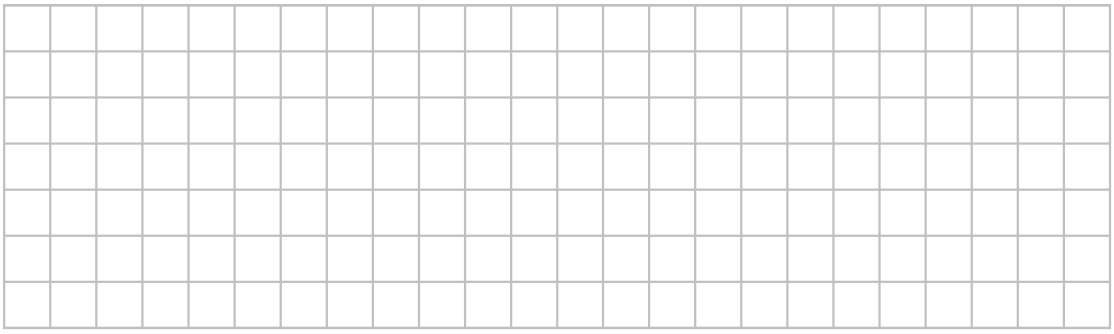
5.1.3 Modifique el programa para leer el puerto.



5.1.4 Indique cuanto demora cada instrucción.



6 **Conclusiones y Recomendaciones:**



7 Evaluación:

7.1 Describir para que sirven los modos de direccionamiento.

7.2 Como se puede escribir y/o leer en los puertos del microcontrolador.

7.3 Elabore un programa y verifique los direccionamientos; luego simúlelo.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 3

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Planteamiento de una sentencia condicional.*

2 Objetivo:

2.1 Conceptuar una sentencia condicional.
2.2 Establecer las condicionantes para que el programa tome decisiones.

3 Materiales y Equipos:

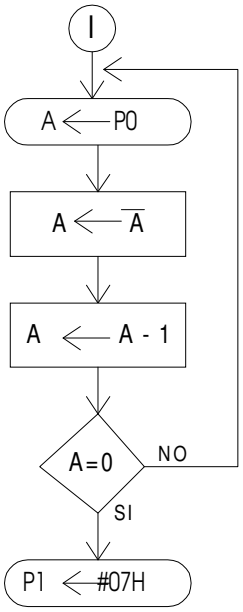
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa con la condición Z, aplicado al acumulador:

FLUJOGRAMA INSTRUCCIONES

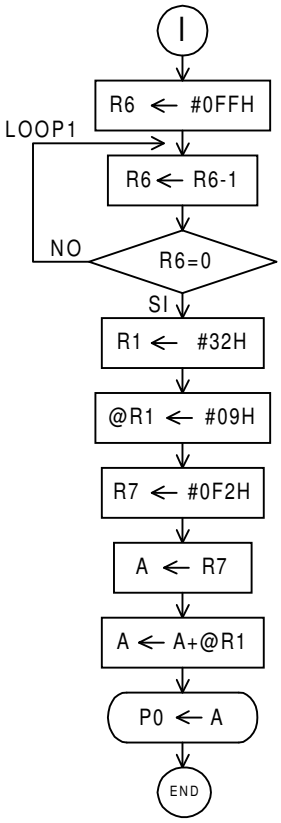



```
LOOP1:  MOV     A,80H
        CPL     A
        DEC     A
        JNZ       LOOP1
        MOV     90H,#07H

        END
```

5.1.1 Describir la tarea del programa y simúlelo.

5.2 Escribir el siguiente programa con la condición "aunque sea una vez", aplicado a un temporizador:

FLUJOGRAMA INSTRUCCIONES

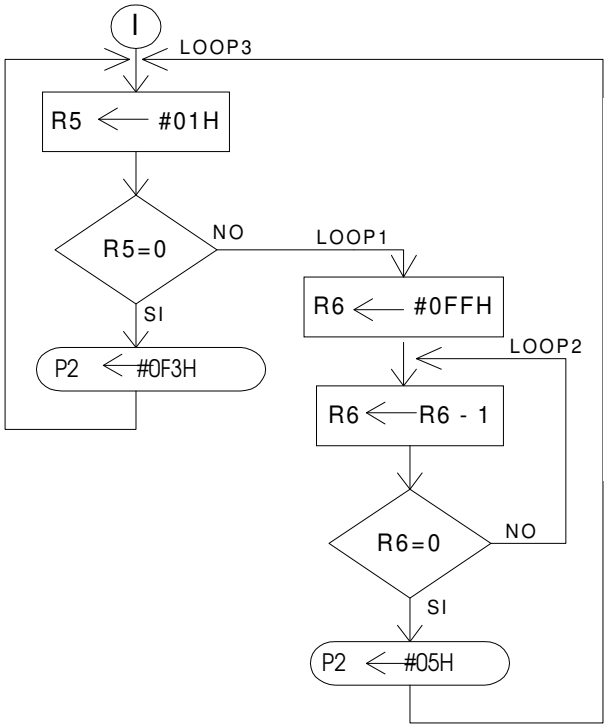



```
        MOV     R6,#0FFH
LOOP1:  DEC     R6
        CJNE    R6,#00H,LOOP1
        MOV     R1,#32H
        MOV     @R1,#09H
        MOV     R7,#0F2H
        MOV     A,R7
        ADD     A,@R1
        MOV     P0,A

        END
```

5.2.1 Describir la tarea del programa y simúlelo.

5.3 Escribir el siguiente programa con la condición "aunque sea una vez":

FLUJOGRAMA INSTRUCCIONES




```
LOOP3:  MOV     R5,#01H
        DJNZ    R5,LOOP1
        MOV     0A0H,#0F3H
        SJMP    LOOP3
LOOP1:  MOV     R6,#0FFH
LOOP2:  DJNZ    R6,LOOP2
        MOV     0A0H,#05H
        SJMP    LOOP3

        END
```

5.3.1 Describir la tarea del programa y simúlelo.

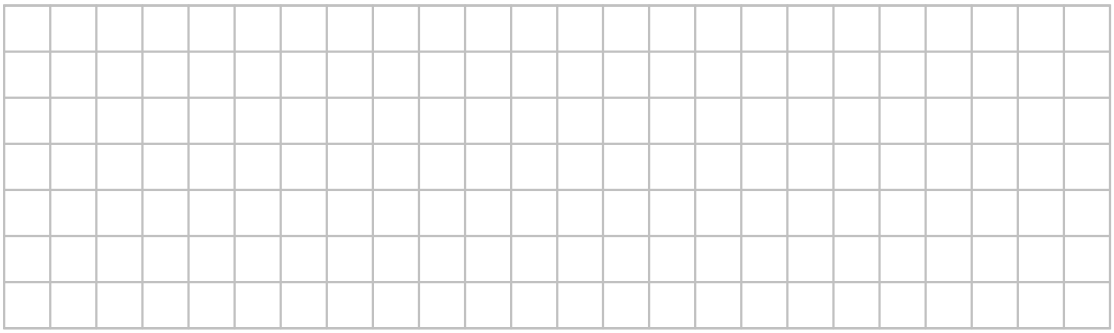
6 Conclusiones y Recomendaciones:

7Evaluación:

7.1 Escriba las instrucciones de bifurcación y explíquelas.

7.2 Basándose en los ejemplos, plantee otros ejemplos para cada uno de ellos.

7.3 Escriba un programa que englobe sentencias condicionales.



"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 4

Nombre: _____ Curso:_____Calificación: _____
Fecha: _____

1 Tema: *Verificación de bytes.*

2 Objetivo:

2.1 Establecer los métodos para chequear bytes.
2.2 Aprender a manipular palabras de 8 bits

3 Materiales y Equipos:

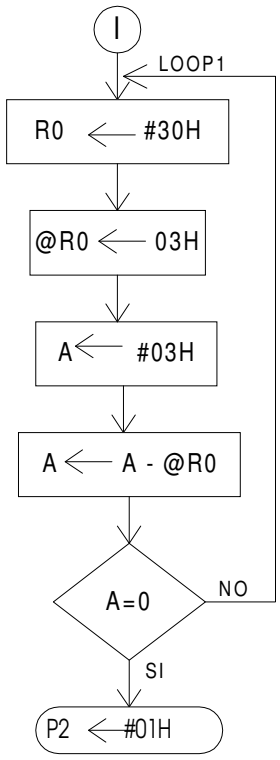
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Verificación por Substracción:

FLUJOGRAMA INSTRUCCIONES

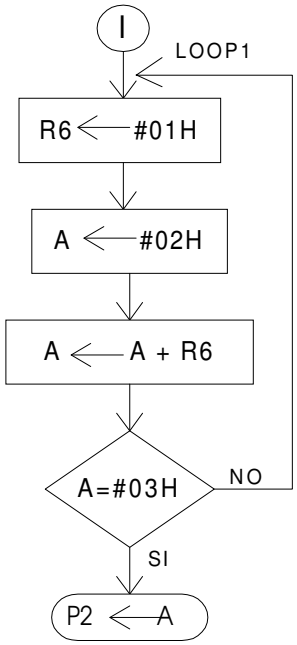



```
LOOP1:  MOV     R0,#30H
        MOV     @R0,03H
        MOV     A,#03H
        SUBB     A,@R0
        JNZ     LOOP1
        MOV     0A0H,#01H

        END
```

5.1.1 Describir la tarea del programa y simúlelo.

5.2 Escribir el siguiente programa con la condición Suma:

FLUJOGRAMA INSTRUCCIONES

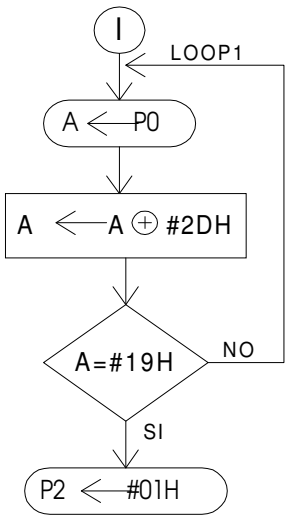



```
LOOP1:  MOV     R6,#01H
        MOV     A,#02H
        ADD     A,R6
        CJNE    R6,#03H,LOOP1
        MOV     0A0H,A

        END
```

5.2.1 Describir la tarea del programa y simúlelo.

5.3 Escribir el siguiente programa para verificar por la lógica X-OR; comparador de desigualdad:

FLUJOGRAMA INSTRUCCIONES




```
LOOP1:  MOV     A,80H
        XRL     A,#2DH
        CJNE    A,#19H,LOOP1
        MOV     0A0H,A

        END
```

5.4 Describir la tarea del programa y simúlelo.

6 Conclusiones y Recomendaciones:

7 Evaluación:

7.1 Explique en que se basan los métodos de verificación.

7.2 Elabore un programa e indique que método aplicaría.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 5

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Verificación de bits.*

2 Objetivo:

2.1 Establecer los métodos para chequear bits de una palabra de determinada.
2.2 Aprender a manipular los bits por separado.

3 Materiales y Equipos:

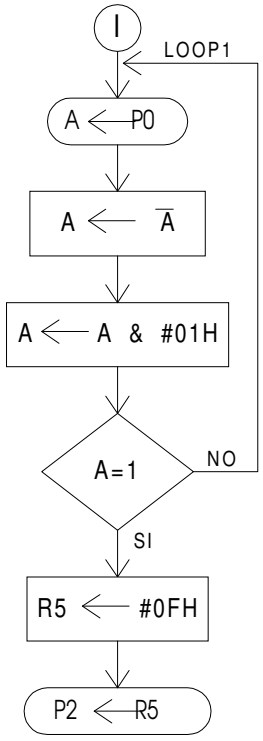
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa por el método lógico:

FLUJOGRAMA INSTRUCCIONES

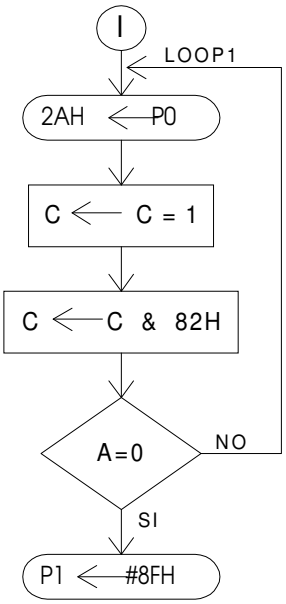



```
LOOP1:  MOV     A,80H
        CPL     A
        ANL     A,#01H
        JNZ     LOOP1
        MOV     R5,#0FH
        MOV     0A0H,R5

        END
```

5.1.1 Explique en que se basa la verificación.

5.2 Escribir el siguiente programa por el método particular:

FLUJOGRAMA INSTRUCCIONES




```
LOOP1:  MOV     2AH,80H
        SETB    C
        ANL     C,82H
        JNC     LOOP1
        MOV     90H,#8FH

        END
```

5.2.1 Explique en que se basa la verificación.

6 Conclusiones y Recomendaciones:

7 Evaluación:

7.1 Actuará igual el primer método con una dirección?

7.2 Podría el segundo método hacerse más directo?

7.3 Elabore un programa en donde se los aplique.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 6

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Temporizadores y Contadores por medio de software.*

2 Objetivo:

2.1 Implementar tiempos de retardo utilizando un conjunto de instrucciones.
2.2 Elaborar contadores por software.

3 Materiales y Equipos:

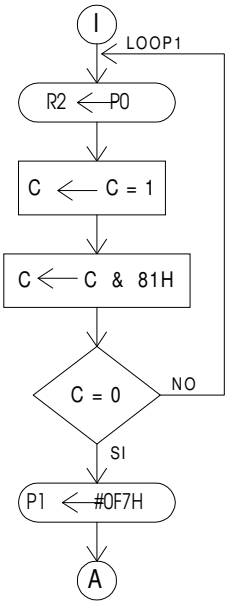
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

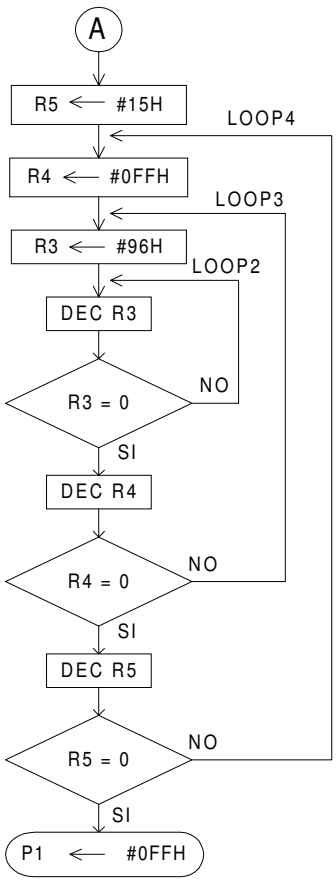
5.1 Escribir el siguiente programa de la aplicación de un temporizador:

FLUJOGRAMA



INSTRUCCIONES

```
LOOP1:  MOV     R2,80H
        SETB    C
        ANL     C,81H
        JNC     LOOP1
        MOV     90H,#0F7H
        MOV     R5,#15H
LOOP4:  MOV     R4,#0FFH
LOOP3:  MOV     R3,#96H
LOOP2:  DJNZ    R3,LOOP2
        DJNZ    R4,LOOP3
        DJNZ    R5,LOOP4
        MOV     90H,0FFH

        END
```

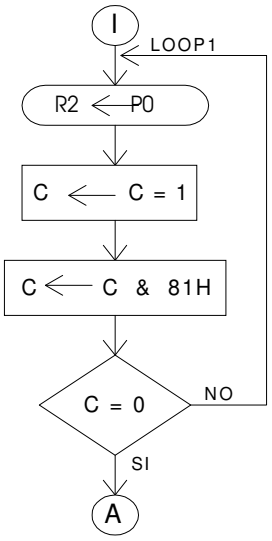


5.1.1 Explique el programa, y simúlelo.

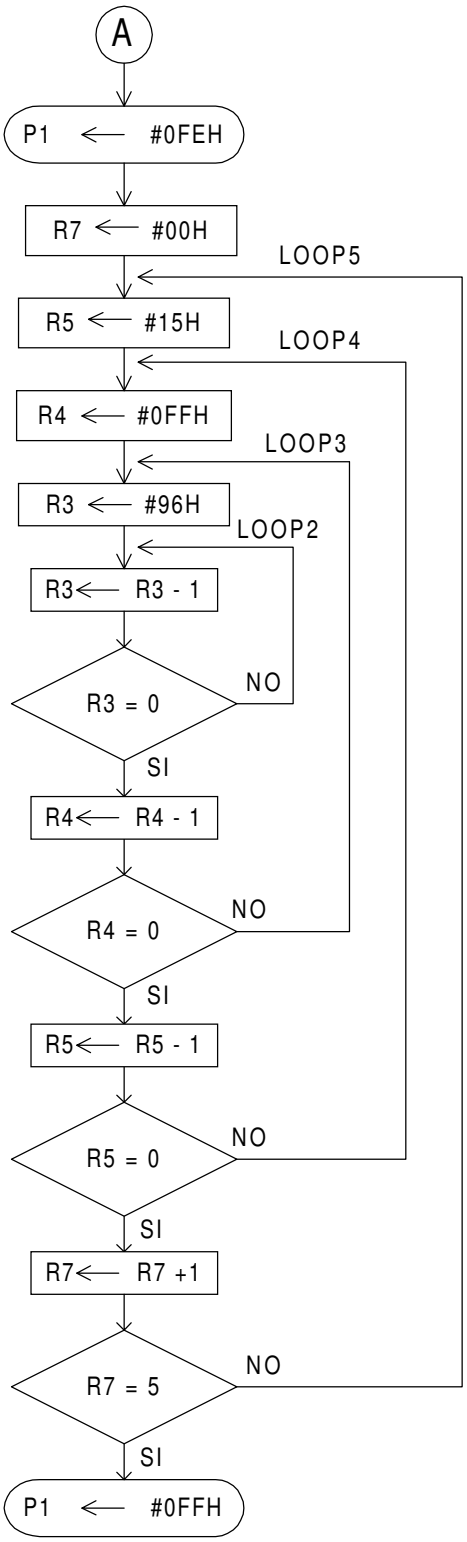
5.1.2 Determine el tiempo que tarda la salida en bascular.

5.2 Escribir el siguiente programa de la aplicación de un contador:

FLUJOGRAMA INSTRUCCIONES




```
LOOP1:  MOV     R2,80H
        SETB    C
        ANL     C,81H
        JNC     LOOP1
        MOV     90H,#0FEH
        MOV     R7,#00H
LOOP5:  MOV     R5,#15H
LOOP4:  MOV     R4,#0FFH
LOOP3:  MOV     R3,#96H
LOOP2:  DJNZ    R3,LOOP2
        DJNZ    R4,LOOP3
        DJNZ    R5,LOOP4
        INC     R7
        CJNE    R7,#05H,LOOP5
        MOV     90H,0FFH

        END
```


A
P1 ⟵ #0FEH
R7 ⟵ #00H
LOOP5
R5 ⟵ #15H
LOOP4
R4 ⟵ #0FFH
LOOP3
R3 ⟵ #96H
LOOP2
R3⟵ R3 - 1
R3 = 0
NO
SI
R4⟵ R4 - 1
R4 = 0
NO
SI
R5⟵ R5 - 1
R5 = 0
NO
SI
R7⟵ R7 +1
R7 = 5
NO
SI
P1 ⟵ #0FFH

5.2.1 Explique como se produce el conteo.

6 Conclusiones y Recomendaciones:

7 Evaluación:

7.1 Plantee un temporizador y, determinar el tiempo que tarda.

7.2 Escriba un temporizador de 1,5 segundos de retardo.

7.3 Elabore un programa que cuente del 0 al 10.

7.4 Explique con un gráfico como realizaría la visualización del conteo.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 7

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Empleo de Subrutinas.*

2 Objetivo:

2.1 Aprender a llamar y a retornar de una subrutina
2.2 Establecer que es una subrutina.

3 Materiales y Equipos:

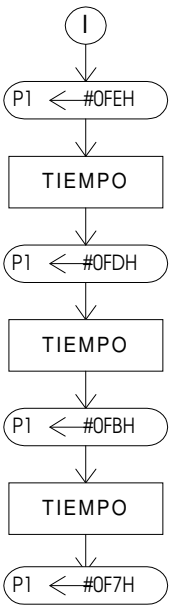
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa:

FLUJOGRAMA

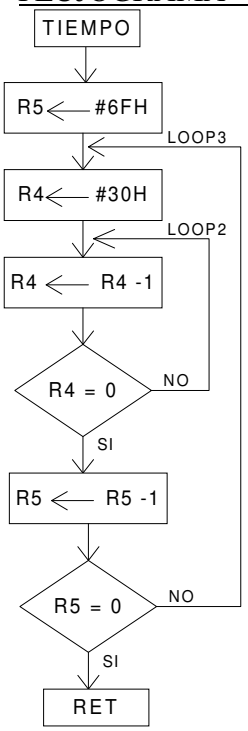


INSTRUCCIONES

```
        ORG     0120H

        MOV     90H,#0FEH
        ACALL   TIEMPO
        MOV     90H,#0FDH
        ACALL   TIEMPO
        MOV     90H,#0FBH
        ACALL   TIEMPO
        MOV     90H,#0F7H

        END
```

5.1.1 Rutina de tiempo:

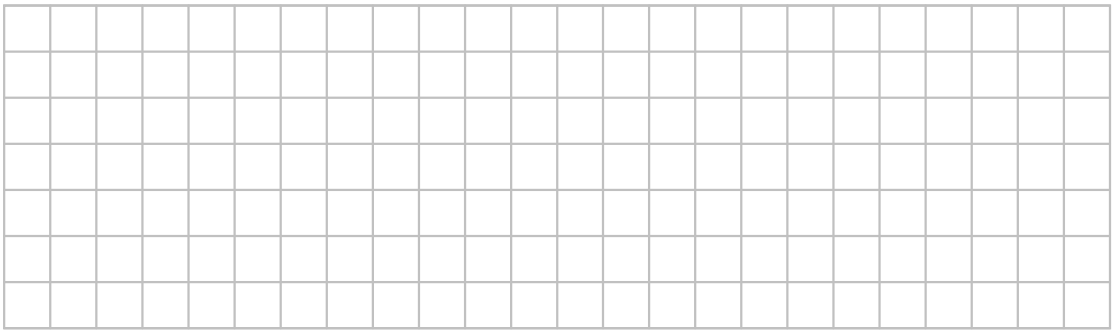
FLUJOGRAMA



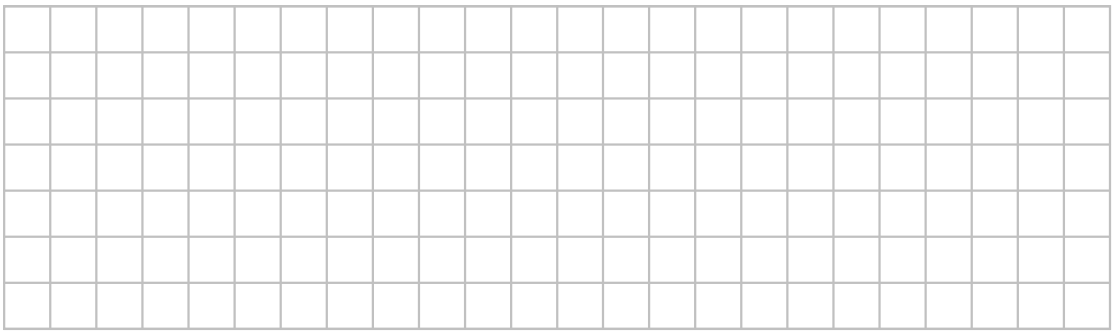
INSTRUCCIONES

```
        ORG     0200H
        MOV     R5,#6FH
LOOP2:  MOV     R4,#30H
LOOP1:  DJNZ    R4,LOOP1
        DJNZ    R5,LOOP2
        RET
```

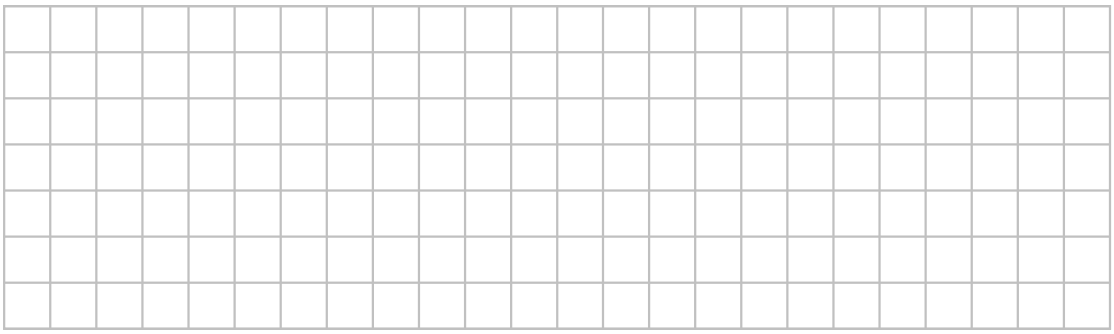
5.2 Que hace el programa.



5.3 Indique que ventaja presenta.



5.4 Conceptualice una subrutina.



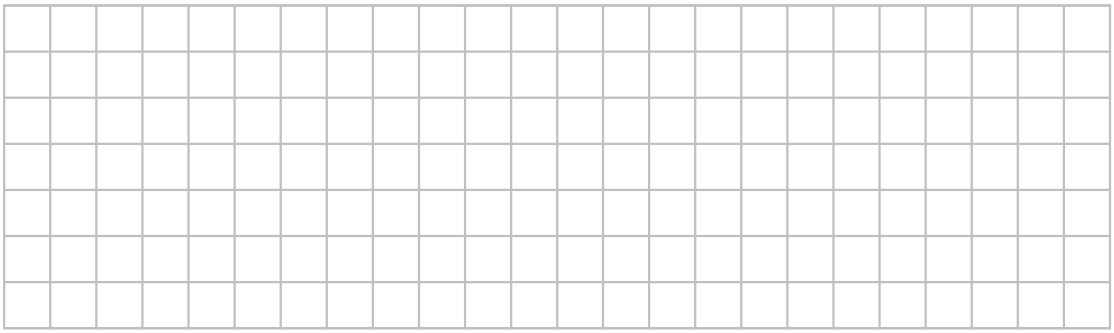
6 Conclusiones y Recomendaciones:

7 Evaluación:

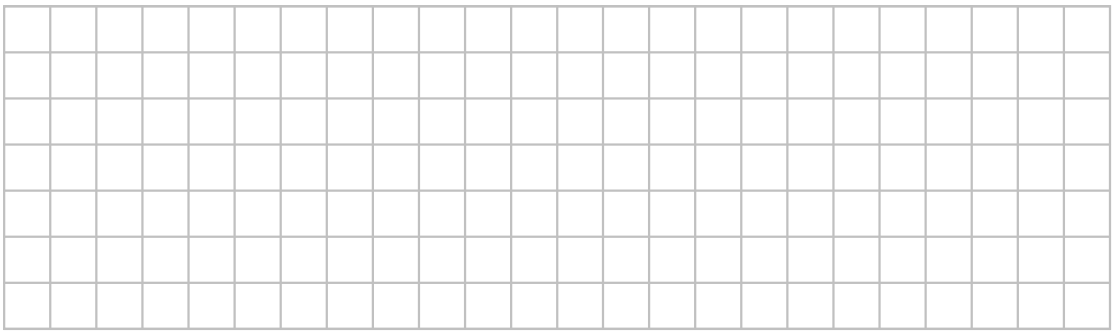
7.1 Para que se emplea una subrutina.

7.2 Cuales son las instrucciones usadas.

7.3 Elabore un programa en el cual se haga indispensable una subrutina.



7.4 Replantee el programa anterior para que cuente en ciclo cerrado izquierda-derecha.



"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 8

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Manejo del puntero de Pila.*

2 Objetivo:

2.1 Aprender a utilizar la pila para guardar datos, para luego recuperarlos.
2.2 Establecer que la pila no solo es manipulada por el microcontrolador sino también por el programador.

3 Materiales y Equipos:

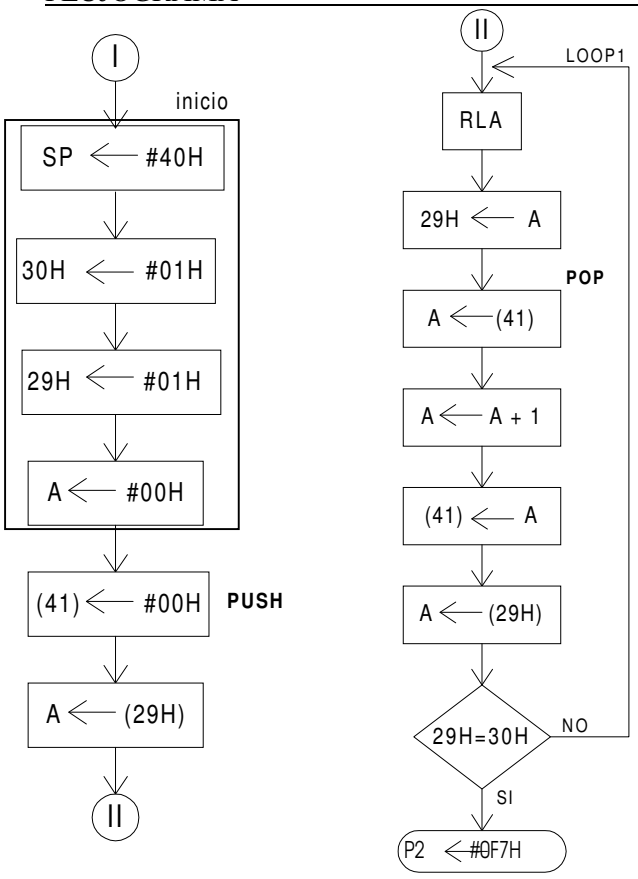
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa aplicado a una rotación de un bit:

FLUJOGRAMA



INSTRUCCIONES

```
; dir1= 30h,  dir2=29h

        MOV     SP,#40H
        MOV     30H,#01H
        MOV     29H,#01H
        MOV     A,#00H
;---------------------------------------
        PUSH    ACC
        MOV     A,29H

LOP1:   RL      A
        MOV     29H,A
        POP     ACC
        INC     A
        PUSH    ACC
        MOV     A,29H
        CJNE    A,30H,LOP1

        MOV     P2,#0F7H

        END
```

5.2 Explique como funciona el programa.

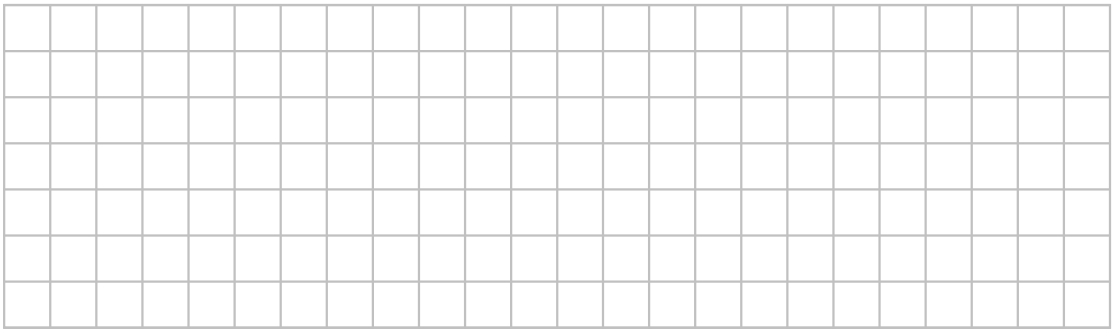
5.3 Como opera la instrucción RL A.

5.4 Como es el anidamiento en la pila.

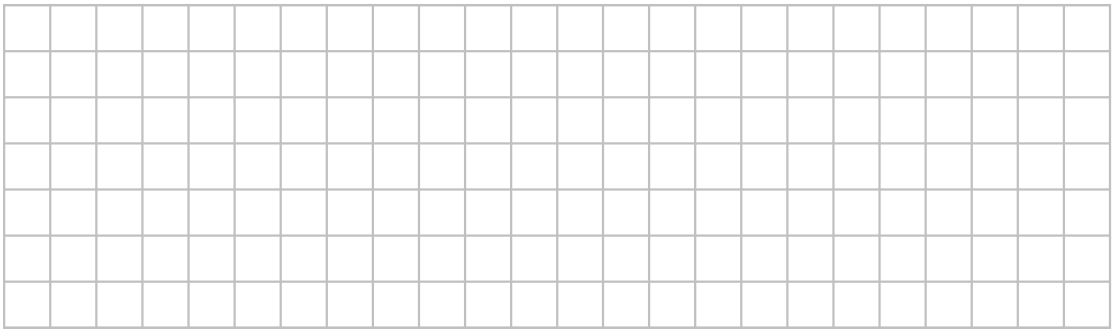
6 Conclusiones y Recomendaciones:

7 Evaluación:

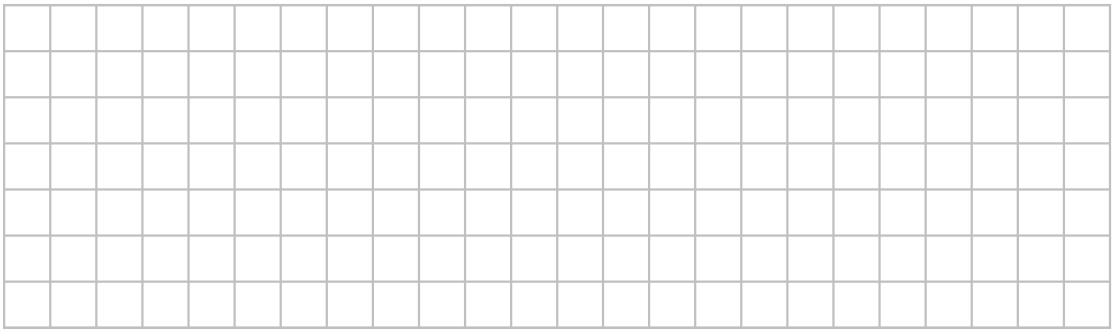
7.1 Describa el orden de gestión de la pila.



7.2 Explique las instrucciones PUSH y POP; describa su empleo.



7.3 Escriba un programa en donde se gestione el stack pointer.



"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 9

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Empleo de los timers para implementar Temporizadores y Contadores.*

2 Objetivo:

2.1 Comprender como se puede implementar temporizadores y contadores con los timers.

2.2 Conocer cuales son los registros que se manipulan para tal efecto.

3 Materiales y Equipos:

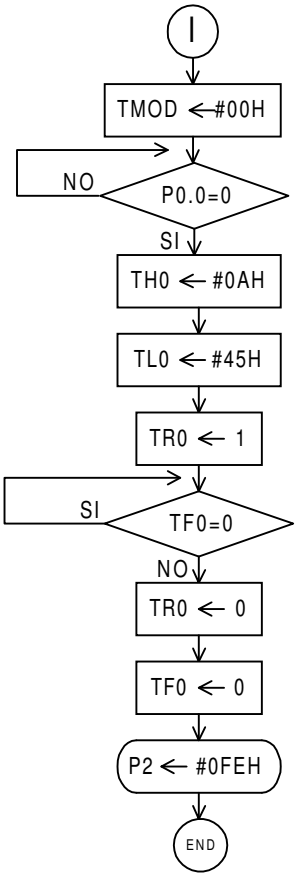
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

4 Marco Teórico:

5 Procedimiento:

5.1 Escribir el siguiente programa aplicado al modo 0, válido para el timer 0 y 1:

FLUJOGRAMA



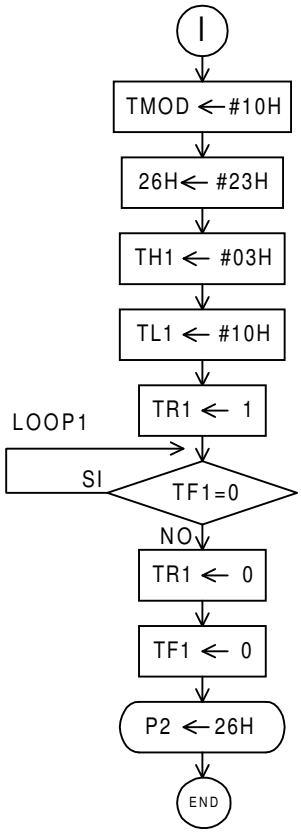
INSTRUCCIONES

```
        MOV     TMOD,#00H
LOOP1:  JB      80H,LOOP1
        MOV     TH0,#0AH
        MOV     TL0,#45H
        SETB    8CH
LOOP2:  JNB     8DH,LOOP2
        CLR     TR0
        CLR     TF0
        MOV     0A0H,#0FEH

        END
```

5.1.1 Explique el proceso del programa.

5.2 Escribir el siguiente programa con el timer 1 en modo 1, también es válido para el timer 0.

FLUJOGRAMA INSTRUCCIONES

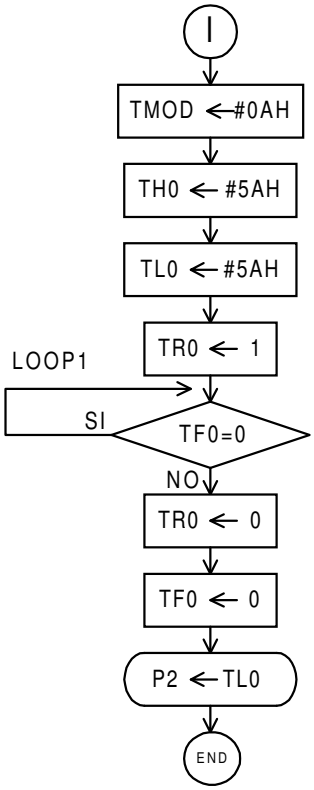



```
        MOV     TMOD,#10H
        MOV     26H,#23H
        MOV     TH1,#03H
        MOV     TL1,#10H
        SETB    8EH
LOOP1:  JNB     8FH,LOOP1
        MOV     90H,23H
        CLR     TR1
        CLR     TF1
        MOV     0A0H,26H

        END
```

5.2.1 Explique el proceso del programa.

5.3 Escribir el siguiente programa con el timer 0 en modo 2, también es válido para el timer 1.

FLUJOGRAMA INSTRUCCIONES




```
        MOV     TMOD,#0AH
        MOV     TH0,#5AH
        MOV     TL0,#5AH
        SETB    8CH
LOOP1:  JNB     8DH,LOOP1
        CLR     TR0
        CLR     TF0
        MOV     P2,TL0

        END
```

5.3.1 Explique el proceso del programa.

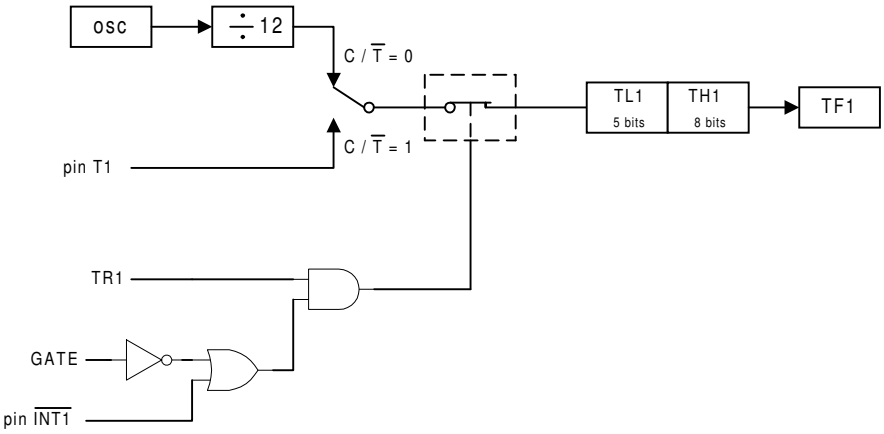
6 Conclusiones y Recomendaciones:

7 Evaluación:

7.1 Describa la finalidad del registro TCON.

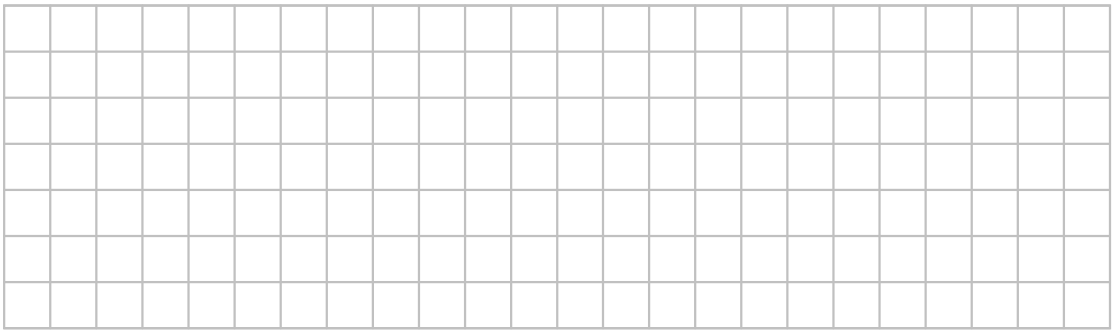
7.2 Describa la finalidad del registro TMOD.

7.3 Basándose en el gráfico siguiente, explicar como haría para que el timer 1 actué como contador:

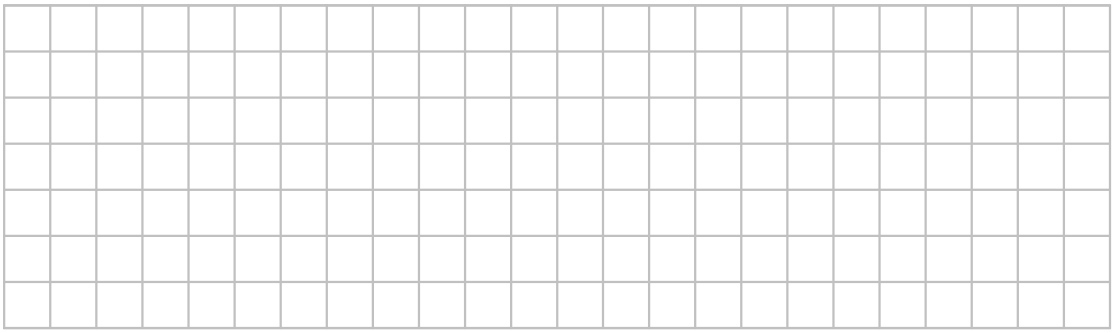


7.4 De acuerdo al ejemplo anterior, haga contador al Timer 0.

7.5 Sintéticamente explique el modo 2.



7.6 Explique el modo 3, y, elabore un programa demostrativo.



"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 10

Nombre: _____ Curso:_____ Calificación: _____
Fecha: _____

1 Tema: *Manejo de las interrupciones externas y de los timers.*

2 Objetivo:

2.1 Comprender como se vectoriza e implementa una interrupción externa.
2.2 Aprender a manejar un timer como generador de interrupción.

3 Materiales y Equipos:

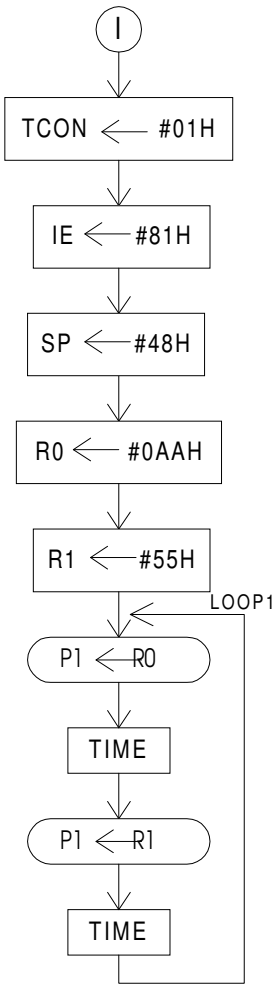
1 Computadora Personal.
1 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.

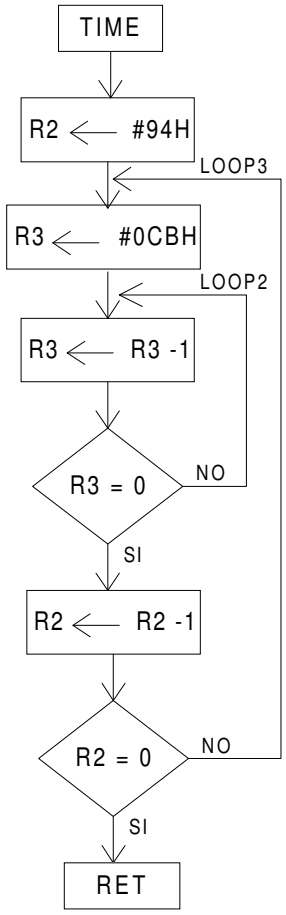
4 Marco Teórico:

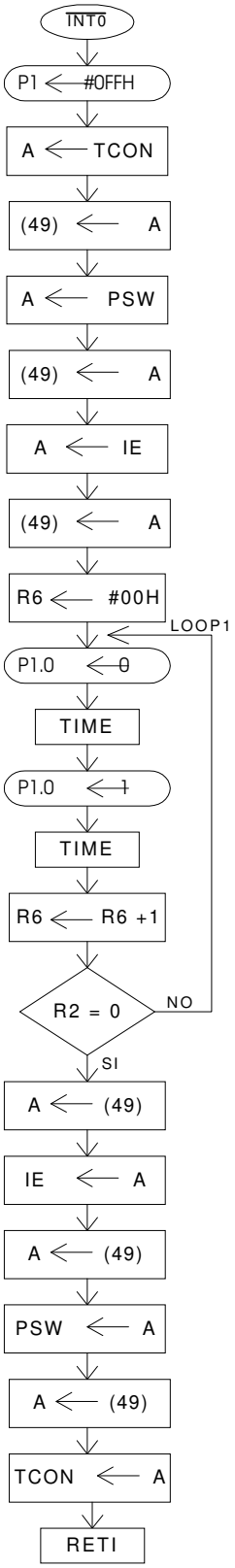
5 Procedimiento:

5.1 Escribir el siguiente programa aplicado al modo 0, válido para el timer 0 y 1:

FLUJOGRAMAS







PROGRAMA PRINCIPAL:

```
        ORG     0100H

        MOV     TCON,#01H
        MOV     IE,#81H
        MOV     SP,#48H
        MOV     R0,#0AAH
        MOV     R1,#55H
LOOP1:  MOV     P1,R0
        ACALL   TIME
        MOV     P1,R1
        ACALL   TIME
        SJMP    LOOP1
```

RUTINA DE TIME:

```
        ORG     0150H

        MOV     R2,#94H
LOOP3:  MOV     R3,#0C8H
LOOP2:  DJNZ    R3,LOOP2
        DJNZ    R2,LOOP3
        RET
```

RUTINA DE INTERRUPCION:

```
        ORG     0200H
        MOV     P1,#0FFH
        MOV     A,TCON
        PUSH    ACC
        MOV     A,PSW
        PUSH    ACC
        MOV     A,IE
        PUSH    ACC
        MOV     R6,#00H
LOOP4:  CLR     P1.0
        ACALL   TIME
        SETB    P1.0
        ACALL   TIME
        INC     R6
        CJNZ    R6,#0AH,LOOP4
        POP     ACC
        MOV     IE,A
        POP     ACC
        MOV     PSW,A
        POP     ACC
        MOV     TCON,A
        RETI
```

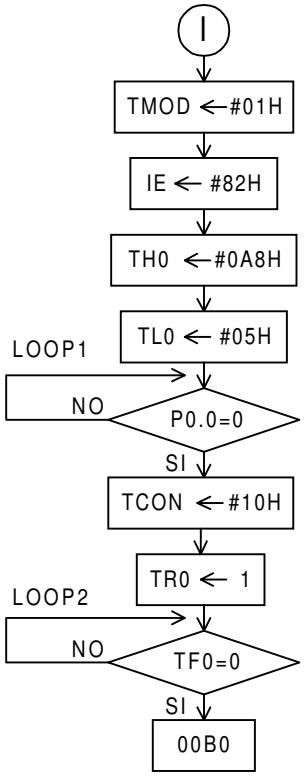
5.1.1 Explique la operación del programa.

5.1.2 Cuanto se demora la subrutina de tiempo.

5.1.3 Encontrar cuanto se demora el programa, describir como se lo calcula.

5.2 Escribir el siguiente programa con la interrupción del timer 0, también es válido para el timer 1.

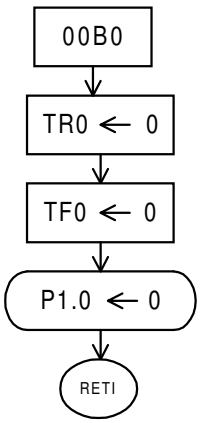
FLUJOGRAMA INSTRUCCIONES



PROGRAMA PRINCIPAL:

```
        ORG     0100H

        MOV     TMOD,#01H
        MOV     IE,#82H
        MOV     TH0,#0A8H
        MOV     TL0,#05H
LOOP1:  JB      P0.0,LOOP1
        MOV     TCON,#10H
        SETB    TR0
LOOP2:  JNB     TF0,LOOP2
```

INTERRUPCION DEL T0:




```
        ORG     000BH

        CLR     TR0
        CLR     TF0
        CLR     P1.0
        RETI

        END
```

5.2.1 En que consiste la programación.

5.2.2 Cuanto se demora el programa.

5.2.3 Demostrar como se logra que la interrupción externa 0 tenga prioridad sobre la del T0.

6 Conclusiones y Recomendaciones:

7 Evaluación:

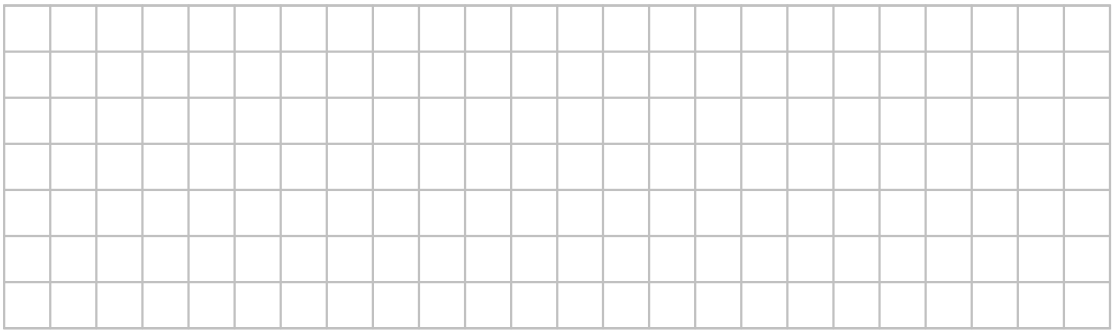
7.1 Como actúan las interrupciones externas.

7.2 Como actúan las interrupciones de los timers.

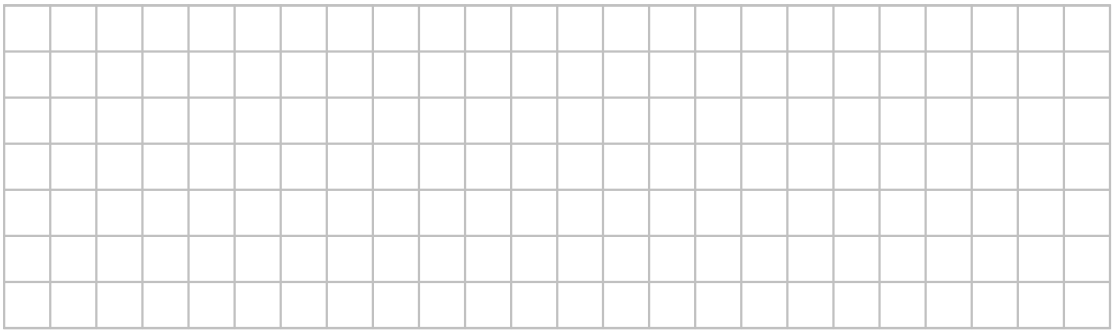
7.3 Al retornar de una interrupción, la posición es:

_____ inicio del programa
_____ final del programa
_____ siguiente instrucción a la que se interrumpió

7.4 Indique los niveles de prioridad de las interrupciones.



7.5 Indique gráficamente los registros de seteo que usan.



PRACTICAS DE IMPLEMENTACION

"COLEGIO TECNICO GUILLERMO MENSI"
LABORATORIO DE ELECTRONICA
PRACTICA No. 1

PRACTICA DE IMPLEMENTACION CON EL MICRO AT89C51

1 Tema: *Mando de dos motores en secuencia temporizada desde 1 puesto.*

2 Materiales y Equipos:

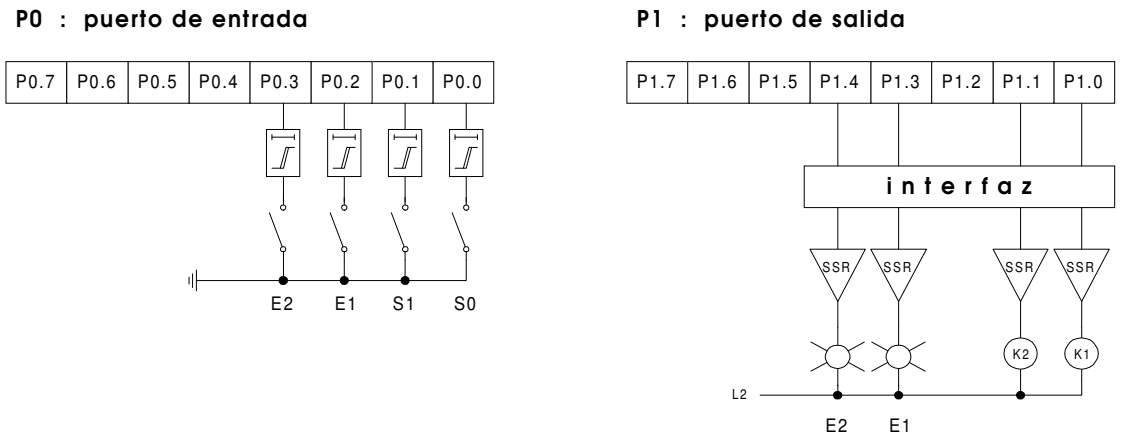
2 Computadora Personal.
2 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.
1 Microcontrolador AT89C51.
5 Pulsantes abiertos.
4 C.I. 74LS132.
5 Resistencias de 1kΩ ½ W +/- 5%.
1 Cristal de 6MHz.
2 Condensadores de 30pF.
1 Resistencias de 10kΩ ½ W +/-5%.
1 Condensador de 2.2µF/16V
1 C.I. 74LS04.
4 Relés de estado sólido.
2 Contactores con bobina 220VCA.

3 Desarrollo:

Se tiene dos motores de una máquina, los cuales se los debe comandar en forma alternada. Para lo cual se establece una secuencia temporizada, es decir automática.

Estableciéndose lo siguiente:

a.- *Situación de los puertos:*



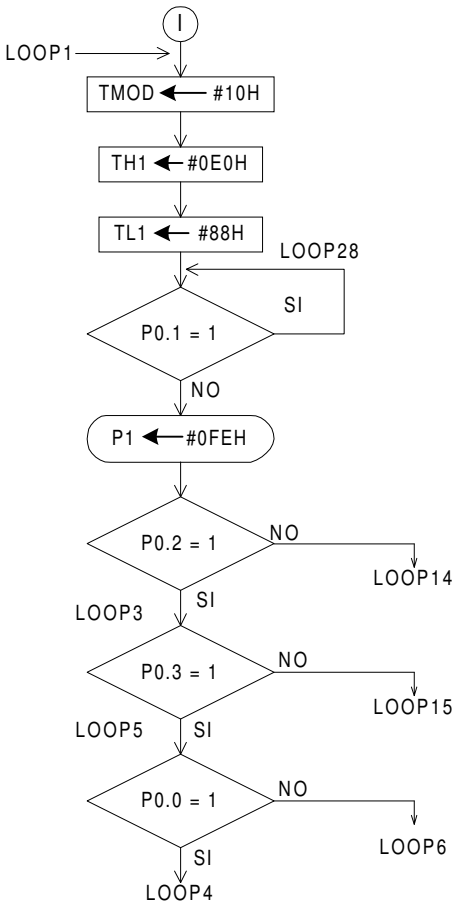
b.- *Condiciones de operación:*

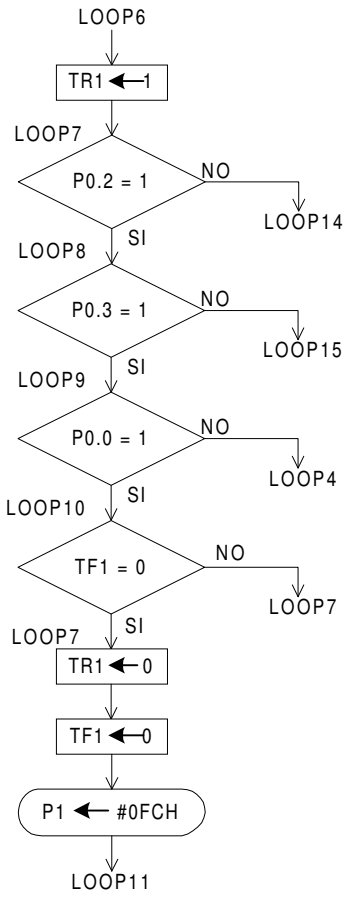
1) Pulsando S1 se enciende k1, luego de un instante de tiempo se energiza k2.
2) El pulsante S0 apaga toda la secuencia en cualquier momento.
3) Cualquiera de los térmicos detiene la secuencia.
4) Para salir del accionamiento de los térmicos, se debe resetear el microcontrolador.
5) La alarma luminosa del disparo de los térmicos es intermitente.

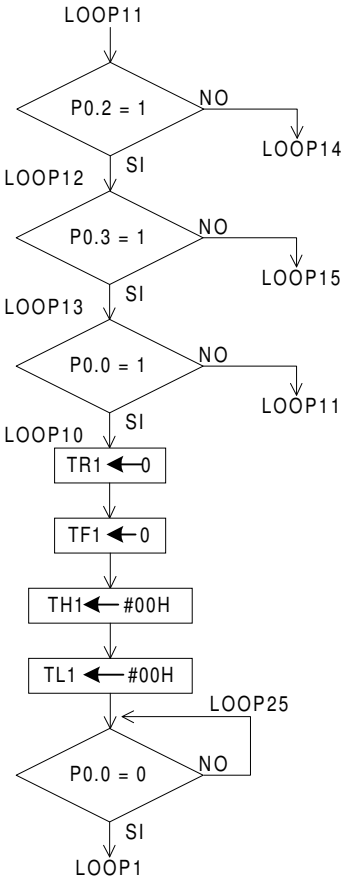
<u>Nota:</u> Pruebe el software en el simulador hasta entender como funciona, y, luego hágalo práctico.

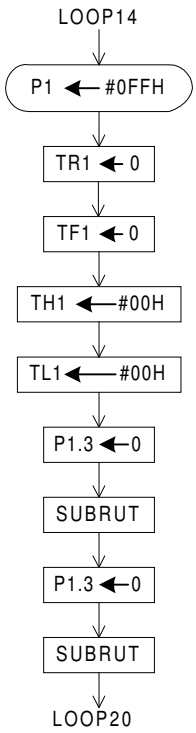
c.- *Elaboración del software de control:*

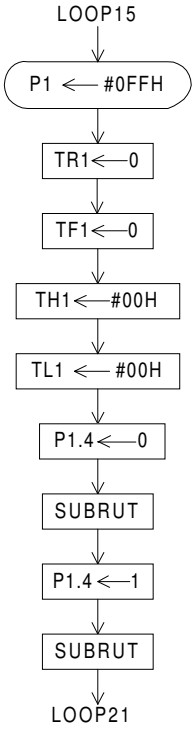
1.- <u>Flujogramas:</u>

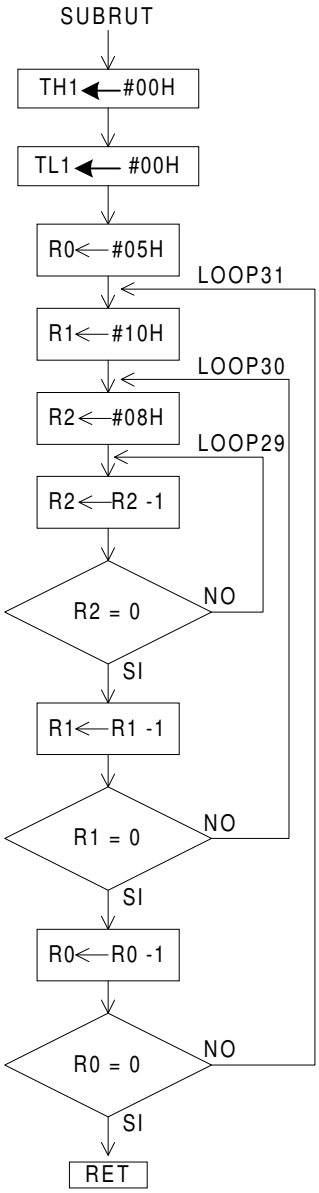





LOOP11
P0.2 = 1
NO
LOOP14
SI
LOOP12
P0.3 = 1
NO
LOOP15
SI
LOOP13
P0.0 = 1
NO
LOOP11
SI
LOOP10
TR1 ← 0
TF1 ← 0
TH1 ← #00H
TL1 ← #00H
LOOP25
P0.0 = 0
NO
SI
LOOP1


LOOP14
P1 ← #0FFH
TR1 ← 0
TF1 ← 0
TH1 ← #00H
TL1 ← #00H
P1.3 ← 0
SUBRUT
P1.3 ← 0
SUBRUT
LOOP20


LOOP15
P1 ← #0FFH
TR1 ← 0
TF1 ← 0
TH1 ← #00H
TL1 ← #00H
P1.4 ← 0
SUBRUT
P1.4 ← 1
SUBRUT
LOOP21



2.- Instrucciones:

```
                ORG     INICIO

LOOP1:          MOV     TMOD,#10H
                MOV     TH1,#0E0H
                MOV     TL1,#88H
```

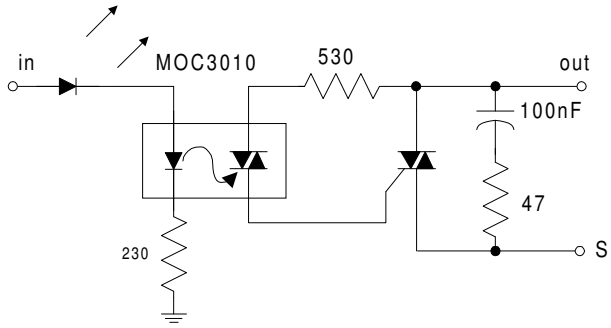
```
LOOP28:   JB      P0.1,LOOP28
          MOV     P1,#0FEH
LOOP2:    JB      P0.2,LOOP3
          SJMP    LOOP14
LOOP3:    JB      P0.3,LOOP5
          SJMP    LOOP15
LOOP5:    JB      P0.0,LOOP6
          SJMP    LOOP4
LOOP6:    SETB    TR1
LOOP7:    JB      P0.2,LOOP8
          SJMP    LOOP14
LOOP8:    JB      P0.3,LOOP9
          SJMP    LOOP15
LOOP9:    JB      P0.0,LOOP10
          SJMP    LOOP4
LOOP10:   JNB     TF1,LOOP7

          CLR     TR1
          CLR     TF1
          MOV     P1,#0FCH
LOOP11:   JB      P0.2,LOOP12
          SJMP    LOOP14
LOOP12:   JB      P0.3,LOOP13
          SJMP    LOOP15
LOOP13:   JB      P0.0,LOOP11
          SJMP    LOOP4
LOOP4:    MOV     P1,#0FFH
          CLR     TR1
          CLR     TF1


          MOV     TH1,#00H
          MOV     TL1,#00H
LOOP25:   JNB     P0.0,LOOP25
          AJMP    LOOP1
LOOP14:   MOV     P1,#0FFH
          CLR     TR1
          CLR     TF1
          MOV     TH1,#00H
          MOV     TL1,#00H
LOOP20:   CLR     P1.3
          ACALL   SUBRUT
          SETB    P1.3
          ACALL   SUBRUT
          JMP     LOOP20
```

```
LOOP15:   MOV     P1,#0FFH
          CLR     TR1
          CLR     TF1
          MOV     TH1,#00H
          MOV     TL1,#00H
LOOP21:   CLR     P1.4
          ACALL   SUBRUT
          SETB    P1.4
          ACALL   SUBRUT
          JMP     LOOP21
```

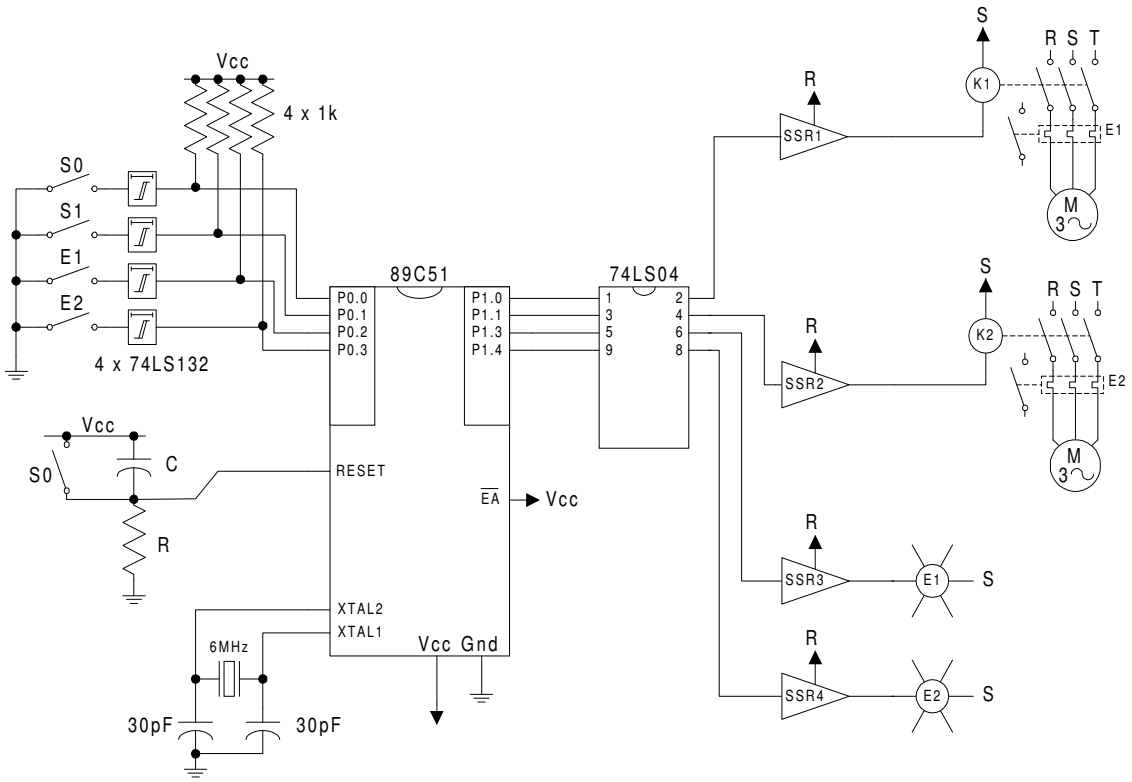
Subrutina de tiempo SUBRUT:
==========

```
          ORG     0300H

          MOV     TH1,#00H
          MOV     TL1,#00H
          MOV     R0,#05H
LOOP31:   MOV     R1,#10H
LOOP30:   MOV     R2,#08H
LOOP29:   DJNZ    R2,LOOP29
          DJNZ    R1,LOOP30
          DJNZ    R0,LOOP31

          RET
;_-------------------------------------------------------------------------

          END
```

3.- Diagrama de conexiones:



Detalle de las conexiones del Rele de Estado Sólido.



<u>En un hoja aparte escriba sus observaciones, y anéxelas a la práctica.</u>

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 2

PRACTICA DE IMPLEMENTACION CON EL MICRO AT89C51

1 Tema: ***Accionamiento de 1 ventilador a determinada temperatura.***

2 Materiales y Equipos:

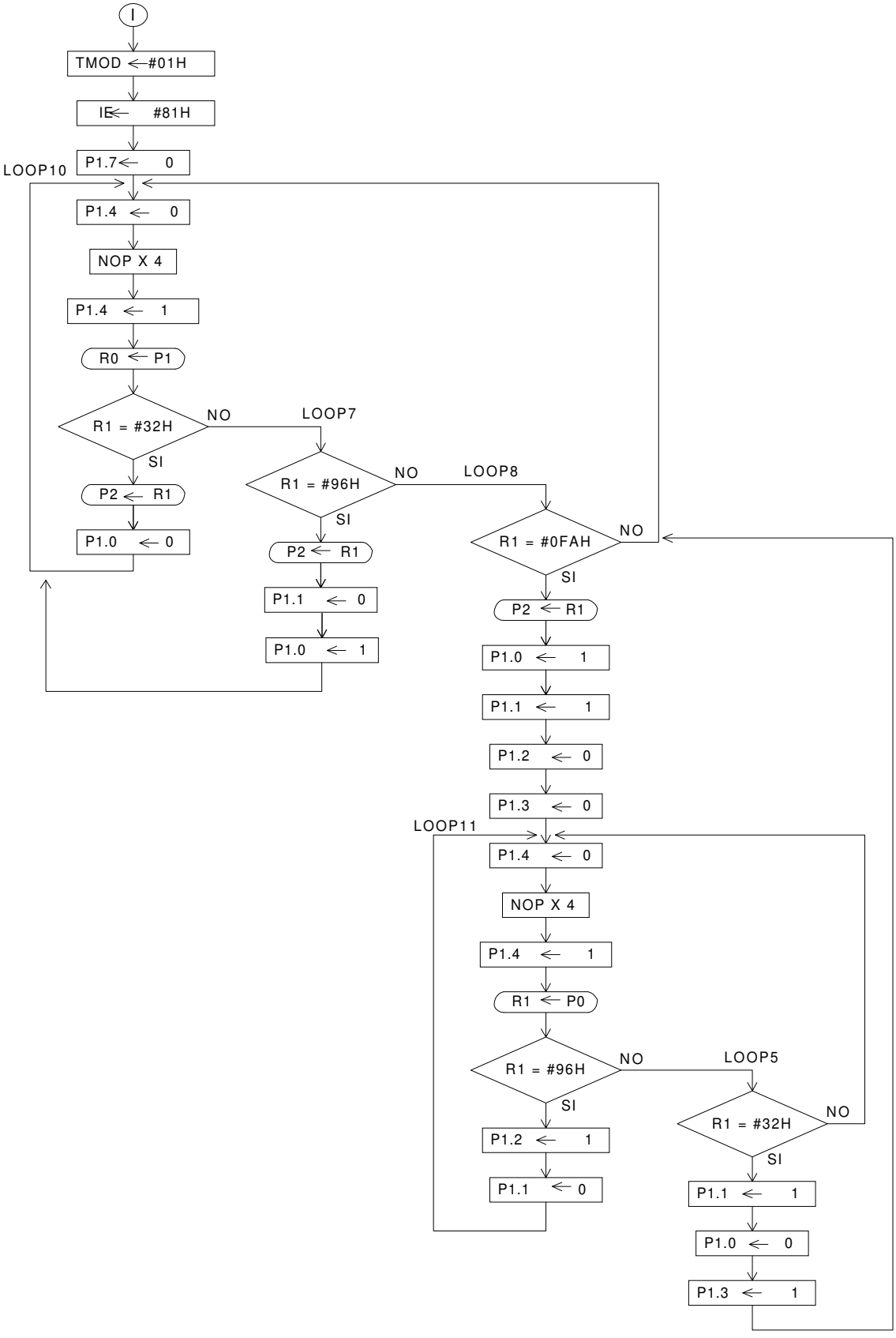
3 Computadora Personal.
3 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.
2 Microcontrolador AT89C51.
3 Cristal de 6MHz.
4 Condensadores de 30pF.
1 Resistencias de 1kΩ ½ W +/-5%.
2 Resistencias de 10kΩ ½ W +/-5%.
2 Pulsantes abiertos.
1 CI 74LS132.
1 Capacitor de 0.1 uF.
1 Capacitor de 10 uF.
10 Leds.
10 Resistencias.
5 Reles de estado sólido.

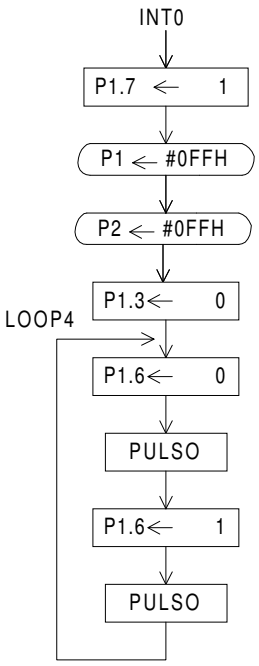
3 Desarrollo:

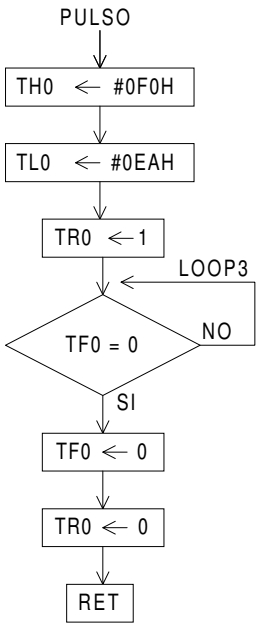
Este circuito es aplicable a un control de temperatura de alguna cámara, espacio físico, etc. Se trata que a cierta temperatura superior a una preestablecida, se acciona el sistema de enfriamiento consistente en un ventilador. Los datos adquiridos son llevados hacia el microcontrolador, el cual los interpreta y realiza el programa requerido. Como la temperatura es una magnitud física, se la debe representar en forma binaria por medio de un ADC. El puerto P0 servirá para introducir los datos y, en los puertos P2 y P1 se hará la visualización del código y el control de los elementos, respectivamente. Los pines P1.6 y P1.7 avisaran si el sistema está interrumpido o es normal.

Cuando el ADC sense 1 voltio se generará un código de 00110010 (32H) que indicará que la temperatura es baja; 3 voltios se generará un código de 10010110 (96H) indicando una temperatura media, y, 5 voltios produciría 11111010 (0FAH) que dirá que la temperatura es alta. Entonces se activará el ventilador permitiendo bajarla, pero al llegar al nivel bajo el ventilador se desconectará; los estados de temperatura se visualizan.

El programa se podrá interrumpir en cualquier instante. A continuación de dan los programas: *Programa Principal*


I
TMOD ←#01H
IE← #81H
P1.7← 0
LOOP10
P1.4 ← 0
NOP X 4
P1.4 ← 1
R0 ← P1
R1 = #32H
NO
LOOP7
SI
P2 ← R1
P1.0 ← 0
R1 = #96H
NO
LOOP8
SI
P2 ← R1
P1.1 ← 0
P1.0 ← 1
R1 = #0FAH
NO
SI
P2 ← R1
P1.0 ← 1
P1.1 ← 1
P1.2 ← 0
P1.3 ← 0
LOOP11
P1.4 ← 0
NOP X 4
P1.4 ← 1
R1 ← P0
R1 = #96H
NO
LOOP5
SI
P1.2 ← 1
P1.1 ← 0
R1 = #32H
NO
SI
P1.1 ← 1
P1.0 ← 0
P1.3 ← 1





<u>Nota:</u> Pruebe en el simulador el software hasta entenderlo, y, luego haga la práctica.

<u>Instrucciones</u>:

```
        ORG   0100H

        MOV     TMOD,#01H
        MOV     IE,#81H

        CLR     P1.7
LOOP10: CLR     P1.4
        NOP
        NOP
        NOP
        NOP
        SETB    P1.4
        MOV     R1,P0
        CJNE    R1,#32H,LOOP7
        MOV     P2,R1
        CLR     P1.0
        SJMP    LOOP10
```

```
LOOP7:  CJNE    R1,#96H,LOOP8
        MOV     P2,R1
        SETB    P1.0
        CLR     P1.1
        SJMP    LOOP10

LOOP8:  CJNE    R1,#0FAH,LOOP10
        MOV     P2,R1
        SETB    P1.0
        SETB    P1.1
        CLR     P1.2
        CLR     P1.3
LOOP11: CLR     P1.4
        NOP
        NOP
        NOP
        NOP
        SETB    P1.4
        MOV     R1,P0
        CJNE    R1,#96H,LOOP5
        SETB    P1.2
        CLR     P1.1
        SJMP    LOOP11

LOOP5:  CJNE    R1,#32H,LOOP11
        SETB    P1.1
        CLR     P1.0
        SETB    P1.3

        AJMP    LOOP10
```

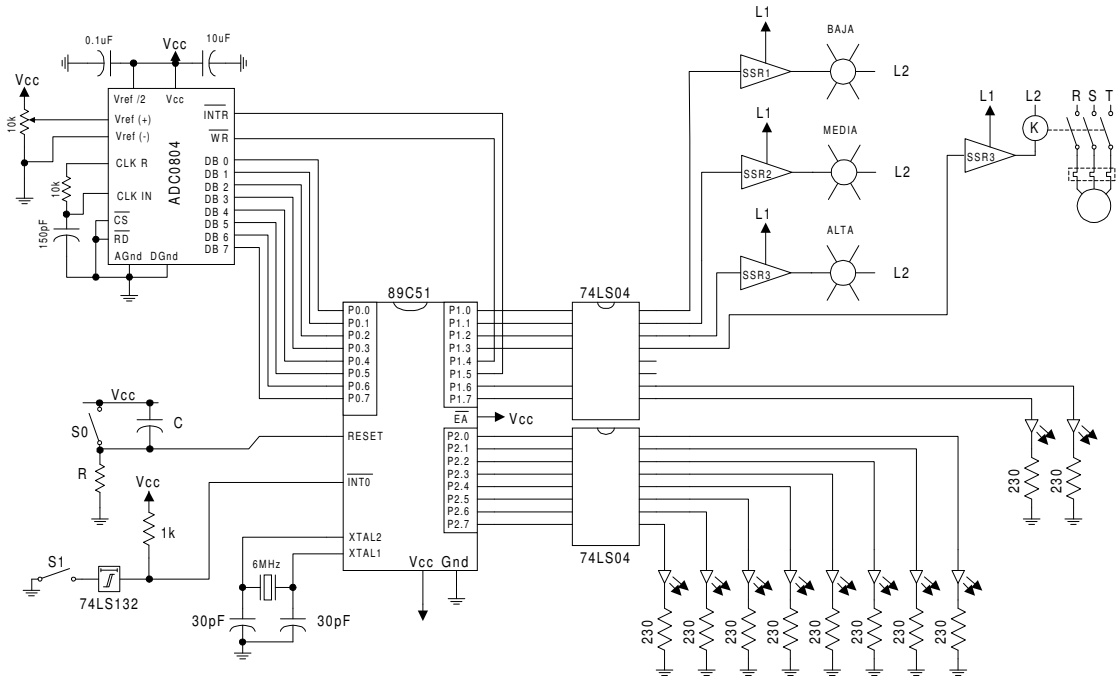
Subrutina de interrupción /INT0:

```
;--------------------------------------

        ORG     0003H

        SETB    P1.7
        MOV     P1,#0FFH
        MOV     P2,#0FFH
        CLR     P1.3
LOOP4:  CLR     P1.6
        ACALL   PULSO
        SETB    P1.6
        ACALL   PULSO
        JMP     LOOP4
```

Subrutina de tiempo PULSO:

```
;-------------------------------------

                ORG         0300H

                MOV         TH0,#0F0H
                MOV         TL0,#0EAH
                SETB        TR0
LOOP3:          JNB         TF0,LOOP3
                CLR         TR0
                CLR         TF0
                RET
;---------------------------------------------------------------------
                END
```

Diagrama de conexiones:



<u>Aparte en un hoja anote sus observaciones y anéxela a la práctica</u>.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 3

PRACTICA DE IMPLEMENTACION CON EL MICRO AT89C51

1 Tema: ***Comunicación serial entre microcontroladores AT89C51.***

2 Materiales y Equipos:

4 Computadora Personal.
4 Software Ensamblador.
1 Diskette de 3 ½ 1.44Mb.
1 Protoboard de 3 bloques.
2 Microcontroladores AT89C51.
3 Cristales de 11.0592 MHz.
4 Condensadores de 30 pF.
1 Dipswitch de 5 pulsadores.
5 Resistencias de 560Ω 1/2W +/-5%
2 Resistencia de 10kΩ 1/2W +/-5%
2 Condensador de 2.2µF/16V
16 Leds
16 Resistencias de 220Ω 1/2W +/-5%
2 CI 74LS240
1 Fuente de +5Vcd 2A
1 Pinza para electrónica
1 Estilete o cuchilla
cable telefónico

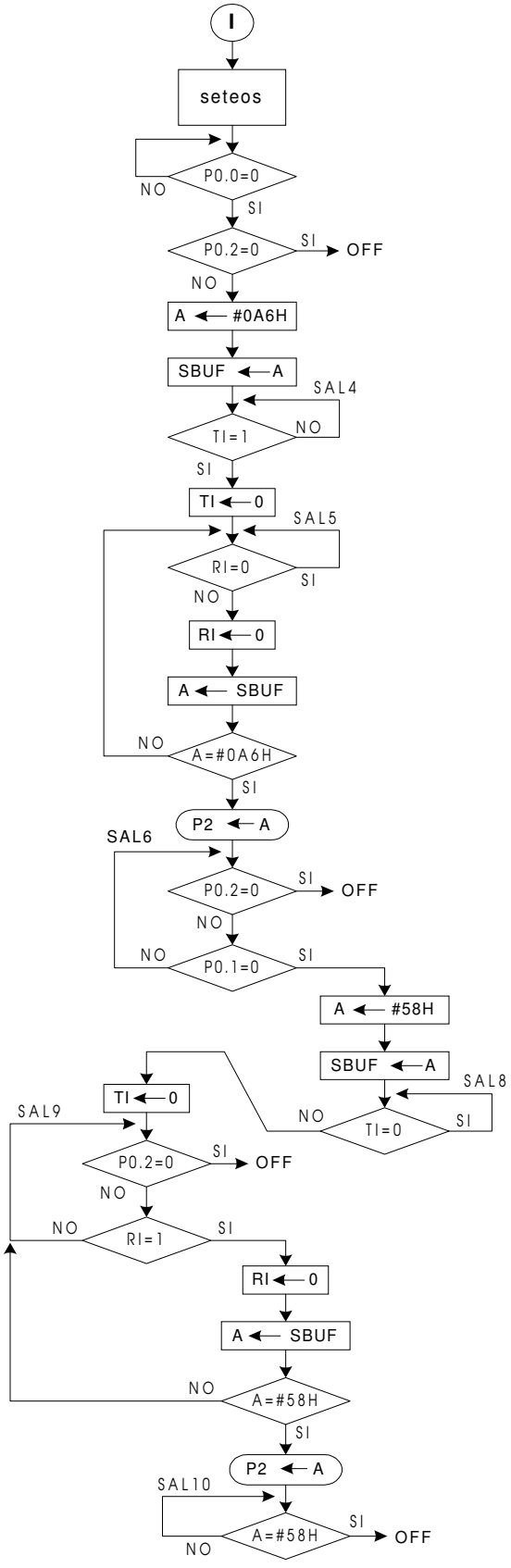
3 Desarrollo:

En esta práctica se establecerá el procedimiento de cómo lograr una comunicación entre dos microcontroladores, la cual funcionará de la siguiente manera:

- Se tiene tres pulsantes conectados al P0.0 – P0.1 – P0.2. El P0.0 activa la primera secuencia; el P0.1, la segunda secuencia; y, P0.2, apaga todo en cualquier instante.
- Al activar el P0.0, se envía un código a l microcontrolador 2, éste lo visualiza y lo regresa al microcontrolador 1 para que también se visualice.
- Al activar el P0.1, se envía otro código al microcontrolador 2 para su visualización, el micro 2 lo regresa al micro 1 y también se visualiza.
- El P0.0 puede apagar a los dos microcontroladores.

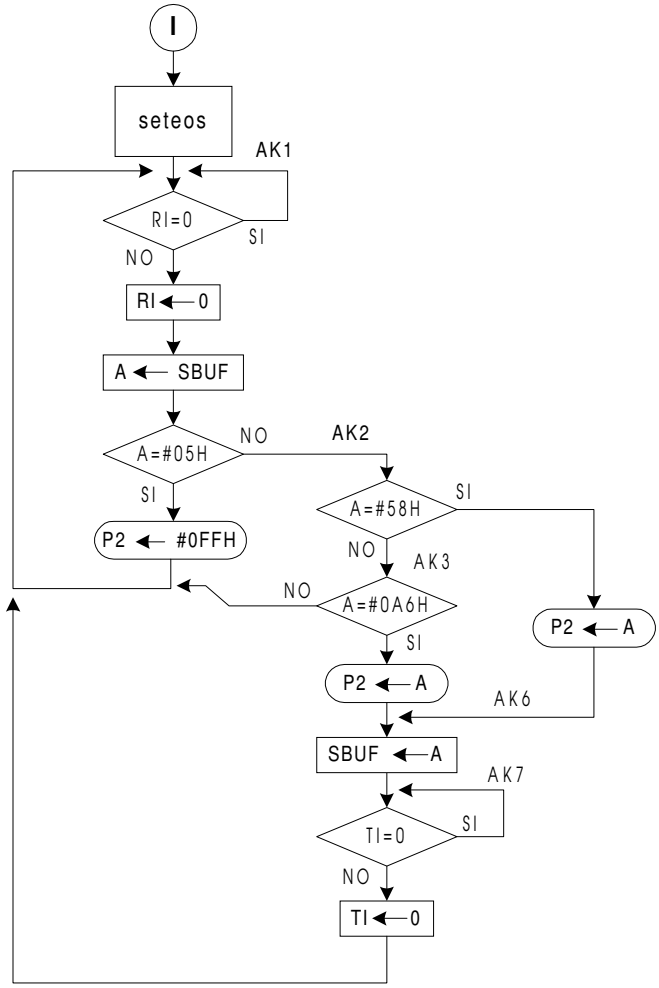
En los seteos se refiere a como se inicializa el micro para generar los baudios de comunicación, y los registros respectivos.

Para implementar, se muestra el Flujograma y la codificación del ejercicio:



Programa para micro1:

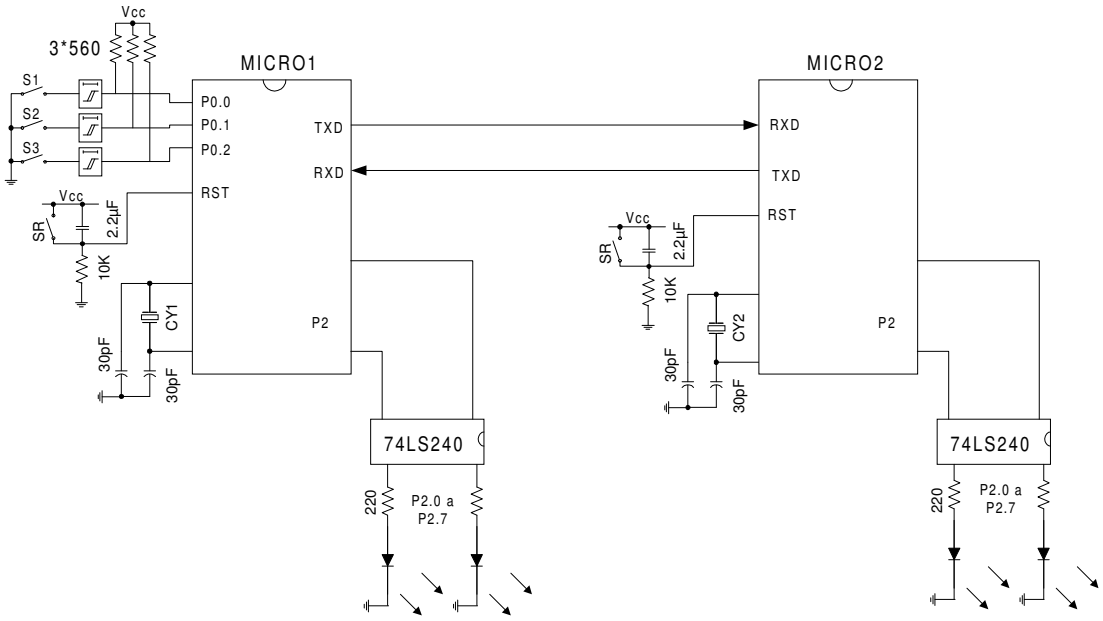
```
SAL1:   JB    P0.0,SAL1
        JNB   P0.2,OFF
        MOV   A,#0A6H
        MOV   SBUF,A
SAL4:   JNB   TI,SAL4
        CLR   TI
SAL5:   JNB   RI,SAL5
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#0A6H,SAL5
        MOV   P2,A
SAL6:   JNB   P0.2,OFF
        JB    P0.1,SAL6
        MOV   A,#58H
        MOV   SBUF,A
SAL8:   JNB   TI,SAL8
        CLR   TI
SAL9:   JNB   P0.2,OFF
        JNB   RI,SAL9
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#58H,SAL9
        MOV   P2,A
SAL10:  JNB   P0.2,OFF
        JMP   SAL10

OFF:    MOV   P2,#0FFH
        MOV   A,#05H
        MOV   SBUF,A
SAL2:   JNB   TI,SAL2
        CLR   TI
SAL3:   JB    P0.2,SAL0
        JMP   SAL3
SAL0:   SJMP  SAL1

        END
```



Programa para micro2:

```
AK1:  JNB   RI,AK1
      CLR   RI
      MOV   A,SBUF
      CJNE  A,#05H,AK2
      MOV   P2,#0FFH
      JMP   AK1

AK2:  CJNE  A,#58H,AK3
      MOV   P2,A
      JMP   AK6

AK3:  CJNE  A,#0A6H,AK1
      MOV   P2,A

AK6:  MOV   SBUF,A

AK7:  JNB   TI,AK7
      CLR   TI
      SJMP  AK1


      END
```

Diagrama de conexiones:



Aparte en un hoja anote sus observaciones, y, anéxelas a la práctica.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

PRACTICA No. 4

PRACTICA DE IMPLEMENTACION CON EL MICRO AT89C51

1 Tema: *Comunicación en formato paralelo del Micro AT89C51 con un PC.*

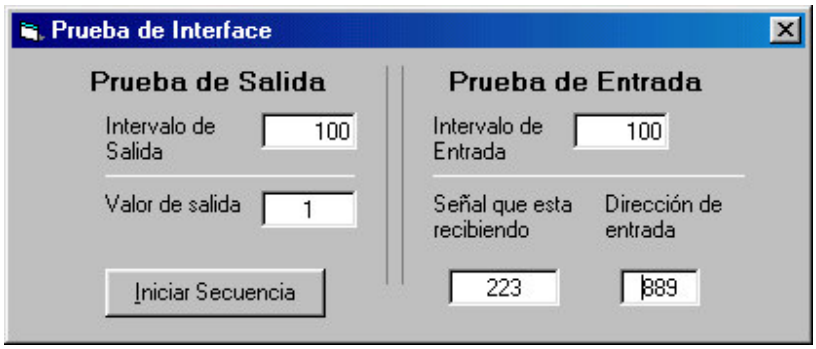
2 Desarrollo:

Para poder realizar ésta práctica se tiene el software Proceso1.exe, que deberá ser ejecutado después de conectado el hardware respectivo. Este presenta las siguientes características:

Pantalla inicial

En la barra de tareas se tiene:

- Prueba de Interface: que permite abrir el subprograma para probar que la interface funcione correctamente.



En la *Prueba de Salida*, el intervalo coloca el tiempo que existe entre el valor que sale y el siguiente. Es programable.

El *Valor de Salida*, es el valor que se envía por el puerto paralelo de la PC hacia el microcontrolador.

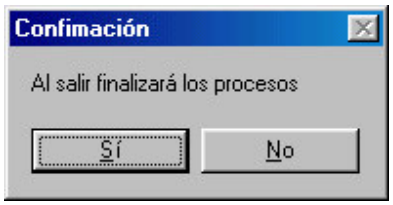
Iniciar Secuencia al pulsarse e iniciarse el envío, cambia a *Terminar Secuencia*.

En la *Prueba de Salida*, el intervalo es el tiempo que establece la lectura de los datos de ingreso. Es programable.

La *Señal que se está recibiendo*, es el valor que está ingresando por el puerto paralelo de la PC.

Dirección de entrada, indica cual es el registro de entrada del puerto paralelo que se está leyendo. Es programable.

- Salir(Alt+F4): cierra el programa, previa confirmación.

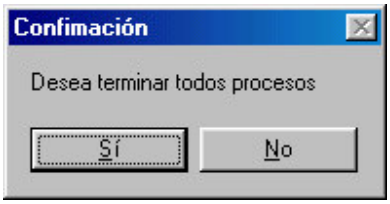


- Acerca de: permite obtener información del software.

Los *Valores de Start*, *Valores de Stop* y *Señal de Apagado* son programables; y, permiten establecer los valores para el cual se requiere activar o detener un proceso. Mientras los

últimos son los valores que detienen fortuitamente un proceso, son enviados por el microcontrolador.

Terminar Todo, detiene todos los procesos que se estén efectuando en el instante de accionamiento., previa confirmación:



En *Señal de Entrada*, se establece los parámetros de la recepción al momento de estar activando procesos.

2.2.1 *Instrucciones del programa del micro en el ASEM51*:

```
; PROGRAMA PARA PRACTICA DE APLICACION  No. 5
; 1. PRIMARY CONTROLS:

$DATE()
$DEBUG
$PAGEWIDTH(90)
$NOXREF
$MOD51
$SYMBOLS

; 2. GENERAL CONTROLS:

$TITLE( APLICACION No. 5/5.2  )
$LIST
$EJECT


; 3. PSEUDO COMMMANDS DEFINITIONS

NAME    PCMICRO1
USING   0

INICIO  EQU     0100H
;/INICIALIZACION:
```

```
ORG     0000H; vectorización del RESET
AJMP    INICIO


; PROGRAMACION EN CURSO:

;Con ,esta secuencia se escoge cual de los receptores
;se va utilizar primero.
;================================================

        ORG     0100H

LOOP1:  MOV     R0,P0
        CJNE    R0,#11H,SAL1
        LJMP    RUTA1

SAL1:   CJNE    R0,#12H,SAL2
        LJMP    RUTA2

SAL2:   CJNE    R0,#13H,LOOP1
        LJMP    RUTA3

;-----------------------------------------------
;-----------------------------------------------

RUTA1:  CLR     P1.5    ;<=====

LOOP2:  JNB     P2.5,EMER
        MOV     R0,P0
        CJNE    R0,#15H,SAL3
        SETB    P1.5
        LJMP    LOOP1
SAL3:   CJNE    R0,#12H,SAL4
        LJMP    RUTA4
SAL4:   CJNE    R0,#13H,LOOP2
        LJMP    RUTA5


RUTA2:  CLR     P1.6  ;<====

LOOP3:  JNB     P2.6,EMER
        MOV     R0,P0
        CJNE    R0,#16H,SAL5
        SETB    P1.6
        LJMP    LOOP1
SAL5:   CJNE    R0,#11H,SAL6
        LJMP    RUTA4
```

```
SAL6:   CJNE    R0,#13H,LOOP3
        LJMP    RUTA7


RUTA3:  CLR     P1.7 ; <=====

LOOP4:  JNB     P2.7,EMER
        MOV     R0,P0
        CJNE    R0,#17H,SAL7
        SETB    P1.7
        LJMP    LOOP1
SAL7:   CJNE    R0,#11H,SAL8
        LJMP    RUTA5
SAL8:   CJNE    R0,#12H,LOOP4
        LJMP    RUTA7

;*****************************************

EMER:   SETB    P1.5
        SETB    P1.6
        SETB    P1.7
DOS:    JNB     P2.5,LOOP21
        JNB     P2.6,LOOP22
        JNB     P2.7,LOOP23

        MOV     R0,P0
        CJNE    R0,#23H,LOOP1
        LJMP    EMER

LOOP21: CLR     P2.0
        SETB    P1.7
        SETB    P1.6
        SETB    P1.5
        JNB     P2.5,LOOP21
        SETB    P2.0
        LJMP    DOS

LOOP22: CLR     P2.1
        SETB    P1.7
        SETB    P1.6
        SETB    P1.5
        JNB     P2.6,LOOP22
        SETB    P2.1
        LJMP    DOS

LOOP23: CLR     P2.2
        SETB    P1.7
```

```
                SETB    P1.6
                SETB    P1.5
                JNB     P2.7,LOOP23
                SETB    P2.2
                LJMP    DOS

;*****************************************

RUTA4:          CLR     P1.5
                CLR     P1.6
LOOP5:          JNB     P2.5,SAL61
                JNB     P2.6,SAL62

                MOV     R0,P0
                CJNE    R0,#15H,SAL11
                SETB    P1.5
                LJMP    RUTA2

SAL11:          CJNE    R0,#16H,SAL12
                SETB    P1.6
                LJMP    RUTA1

SAL12:          CJNE    R0,#13H,LOOP5
                CLR     P1.7

LOOP6:          JNB     P2.5,SAL55
                JNB     P2.6,SAL55
                JNB     P2.7,SAL55
                MOV     R0,P0
                CJNE    R0,#15H,SAL13
                SETB    P1.5
                JMP     RUTA7
SAL13:          CJNE    R0,#16H,SAL14
                SETB    P1.6
                LJMP    RUTA5
SAL14:          CJNE    R0,#17H,LOOP6
                SETB    P1.7
                LJMP    RUTA4

SAL61:          AJMP    EMER1
SAL62:          AJMP    EMER2
SAL55:          AJMP    CAP

;*****************************************

CAP:            SETB    P1.5
                SETB    P1.6
```

```
                SETB    P1.7

SAL15:          JNB     P2.5,SALT1
                SETB    P2.0
                JNB     P2.6,SALT2
                SETB    P2.1
                JNB     P2.7,SALT3
                SETB    P2.2

                MOV     R1,P0
                CJNE    R1,#23H,SAL51
                LJMP    SAL15


SALT1:          CLR     P2.0
                LJMP    SAL15
SALT2:          CLR     P2.1
                LJMP    SAL15
SALT3:          CLR     P2.2
                LJMP    SAL15
SAL51:          AJMP    LOOP1

;*******************************************

EMER1:          SETB    P1.6
                SETB    P1.5
SAL50:          CLR     P2.0
LOOP25:         JNB     P2.6,LOOP26
                JNB     P2.5,LOOP25
                SETB    P2.0
                JNB     P2.6,LOOP26
LOOP28:         SETB    P2.1
                LJMP    LOOP1
LOOP26:         CLR     P2.1
                JNB     P2.5,SAL50
                SETB    P2.0
LOOP27:         JNB     P2.6,LOOP27
                LJMP    LOOP28


EMER2:          SETB    P1.6
                SETB    P1.5
LOOP30:         CLR     P2.1
LOOP29:         JNB     P2.5,LOOP31
                JNB     P2.6,LOOP29
                SETB    P2.1
                JNB     P2.5,LOOP31
```

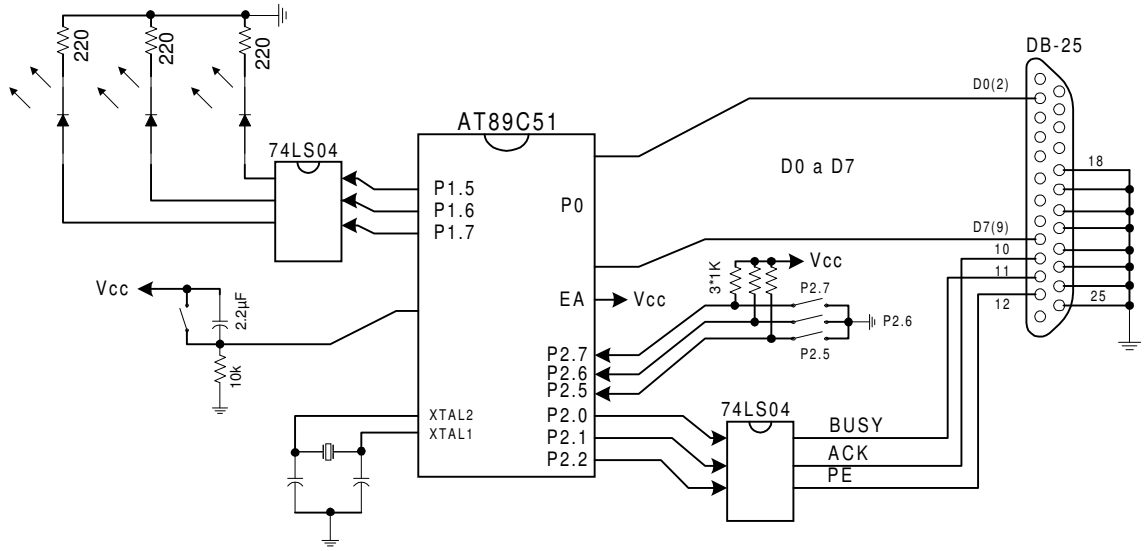
```
LOOP32:   SETB    P2.0
          LJMP    LOOP1
LOOP31:   CLR     P2.0
          JNB     P2.6,LOOP30
          SETB    P2.1
SAL16:    JNB     P2.5, SAL16
          JMP     LOOP32

;***********************************************
;***********************************************

RUTA5:    CLR     P1.5
          CLR     P1.7
LOOP7:    JNB     P2.5,SAL56
          JNB     P2.7,SAL57

          MOV     R0,P0
          CJNE    R0,#15H,SAL17
          SETB    P1.5
          LJMP    RUTA3

SAL17:    CJNE    R0,#17H,SAL18
          SETB    P1.7
          LJMP    RUTA1

SAL18:    CJNE    R0,#12H,LOOP7
          CLR     P1.6
LOOP8:    JNB     P2.5,SAL60
          JNB     P2.6,SAL60
          JNB     P2.7,SAL60
          MOV     R0,P0
          CJNE    R0,#15H,SAL20
          SETB    P1.5
          LJMP    RUTA7
SAL20:    CJNE    R0,#16H,SAL21
          SETB    P1.6
          LJMP    RUTA5;<----------

SAL21:    CJNE    R0,#17H,LOOP8
          SETB    P1.7
          LJMP    RUTA4

SAL56:    AJMP    EMER3
SAL57:    AJMP    EMER4
SAL60:    AJMP    CAP
;-----------------------------------------
```

```
RUTA7:    CLR     P1.6
          CLR     P1.7
LOOP11:   JNB     P2.6,SAL52
          JNB     P2.7,SAL53

          MOV     R0,P0
          CJNE    R0,#16H,SAL22
          SETB    P1.6
          LJMP    RUTA3

SAL22:    CJNE    R0,#17H,SAL23
          SETB    P1.7
          LJMP    RUTA2

SAL23:    CJNE    R0,#11H,LOOP11
          CLR     P1.5

LOOP12:   JNB     P2.5,SAL54
          JNB     P2.6,SAL54
          JNB     P2.7,SAL54
          MOV     R0,P0
          CJNE    R0,#15H,SAL28
          SETB    P1.5
          LJMP    RUTA7
SAL28:    CJNE    R0,#16H,SAL29
          SETB    P1.6
          LJMP    RUTA5
SAL29:    CJNE    R0,#17H,LOOP12
          SETB    P1.7
          LJMP    RUTA4

SAL54:    AJMP    CAP
SAL52:    AJMP    EMER5
SAL53:    AJMP    EMER6

;-------------------------------------------------
;-------------------------------------------------

EMER3:    SETB    P1.5
          SETB    P1.7
SAL30:    CLR     P2.0
SAL31:    JNB     P2.7,SAL32
          JNB     P2.5,SAL31
          SETB    P2.0
          JNB     P2.7,SAL32
SAL34:    SETB    P2.2
          LJMP    LOOP1
```

```
SAL32:  CLR     P2.2
        JNB     P2.5,SAL30
        SETB    P2.0
SAL33:  JNB     P2.7,SAL33
        LJMP    SAL34

EMER4:  SETB    P1.5
        SETB    P1.7
SAL35:  CLR     P2.2
SAL36:  JNB     P2.5,SAL37
        JNB     P2.7,SAL36
        SETB    P2.2
        JNB     P2.5,SAL37
SAL38:  SETB    P2.0
        LJMP    LOOP1
SAL37:  CLR     P2.0
        JNB     P2.7,SAL35
        SETB    P2.2
SAL39:  JNB     P2.5,SAL39
        LJMP    SAL38


EMER5:  SETB    P1.6
        SETB    P1.7
SAL40:  CLR     P2.1
SAL41:  JNB     P2.7,SAL42
        JNB     P2.6,SAL41
        SETB    P2.1
        JNB     P2.7,SAL42
SAL44:  SETB    P2.2
        LJMP    LOOP1
SAL42:  CLR     P2.2
        JNB     P2.6,SAL40
        SETB    P2.1
SAL43:  JNB     P2.7,SAL43
        LJMP    SAL44


EMER6:  SETB    P1.6
        SETB    P1.7
SAL45:  CLR     P2.2
SAL46:  JNB     P2.6,SAL47
        JNB     P2.7,SAL46
        SETB    P2.2
        JNB     P2.6,SAL47
SAL49:  SETB    P2.1
        LJMP    LOOP1
```

```
SAL47:      CLR         P2.1
            JNB         P2.7,SAL45
            SETB        P2.2
SAL48:      JNB         P2.6,SAL48
            LJMP        SAL49

;//////////////////////////////////////////////////
;--------------------------------------------------

            END
```

Diagrama de Conexiones:



<u>En una hoja aparte anote sus observaciones, y, anéxela a la práctica.</u>

MICROPROYECTOS

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

MICROPROYECTO No. 1

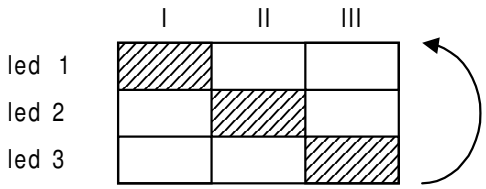
PRACTICA CON EL MICRO
AT89C51

1 Tema: Manejo de 3 leds por medio de una memoria externa EPROM.

2 Materiales:

3 Desarrollo:

El trabajo a realizar será el control de 3 leds en secuencia alternada:



La secuencia puede ser manual o automática. Debe haber un paro general, y contar al menos con una interrupción. Para lo cual tendrá que elaborar el software correspondiente y el hardware de aplicación, y anexarlos. Tenga en consideración la instrucciones MOVX A,@DPTR para dirección a 16 bits; o, MOVX A,@Ri para 8 bits. Así como la combinación de las señales de habilitación.

Luego de completar ésta práctica, anote cuales son sus conclusiones y que recomienda para la aplicación. Anéxelo al microproyecto.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

MICROPROYECTO No. 2

PRACTICA CON EL MICRO
AT89C51

1 Tema: Comunicación en formato serial del microcontrolador con el PC.

2 Materiales:

3 Desarrollo:

Antes de iniciar ésta práctica, cargue el programa SerAt del CD-ROM, póngalo como acceso directo en el escritorio de su PC. Luego realice un software y hardware, anéxelos; para el micro con las siguientes condiciones:

1. Al enviar el PC el número 16, el micro devuelve el número para visualizarse en el monitor, y, se enciende la primera lámpara incandescente de 127v por medio del pin P2.0.
2. Al enviar el PC el número 17, el micro devuelve el número para visualizarse en el monitor, y, se enciende la segunda lámpara incandescente de 127v por medio del pin P2.1.
3. Al enviar el PC el número 18, el micro devuelve el número para visualizarse en el monitor, y, se enciende la tercera lámpara incandescente de 127v por medio del pin P2.2.
4. Al enviar el PC el número 19, el micro devuelve el número para visualizarse en el monitor, y, se apagan todas las lámparas.
5. Luego, la secuencia empieza desde el literal 1, repitiéndose constantemente.

Luego de completar ésta práctica, anote cuales son sus conclusiones y que recomienda para la aplicación.

"COLEGIO TECNICO GUILLERMO MENSI"

LABORATORIO DE ELECTRONICA

MICROPROYECTO No. 3

PRACTICA CON EL MICRO
AT89C51

1 Tema: *Secuencia de luces usando un DAC.*

2 Materiales y Equipos:

3 Desarrollo:

El microcontrolador compara los códigos de ingreso presentados en el puerto P0, y genera de acuerdo a éstos otros códigos que son sacados por el puerto P2. Por medio de un DAC se los transforma en un voltaje, que amplificado es visualizado por una configuración de comparadores conectados a leds.

Los voltajes que se verifican van de 2.2 v en 2.2 v; o sea: 2.2 v – 4.4 v – 6.6 v – 8.8 v. Los códigos de ingreso son producidos por 8 dipswitches.

Luego de realizada ésta práctica, anote sus recomendaciones y conclusiones.

Conclusiones y Recomendaciones

Conclusiones

- Por intermedio de microcontrolador se puede elaborar una aplicación sencilla y práctica para los empleos cotidianos, ahorrando espacio físico y tiempo.

- El microcontrolador AT89C51 posee un repertorio de instrucciones que permiten una rápida programación, muy versátil.

- La gran mayoría de microcontroladores poseen similitud en lo referente a las instrucciones, programación.

- Al dominar un microcontrolador cualquiera, se posee las bases necesarias para comprender otros tipos. Teniendo únicamente que escoger la capacidad de la EPROM.

Recomendaciones

- Al momento de elegir un microcontrolador para una aplicación cualquiera, tomar en consideración la versatilidad del micro.

- Seguir en forma sistemática las prácticas presentadas en este manual, para tener un conocimiento adecuado del AT89C51.

- Incursionar en los lenguajes de programación tales como: C++, Visual Basic, Assembler, etc. Ya que todo microcontrolador se puede interfazar con una PC fácilmente.

- Cuidar la conexiones eléctricas que se emplean en los circuitos que se realizan, porque suelen ser críticas al momento de probar la tarjeta impresa.

- Al momento de emplear un pin de puerto para entrada, utilizar un nivel bajo para indicar un dato presente. Y dicho pin no puede ser usado como salida, peligro de cortocircuito.

- Colocar a cada señal que ingrese al microcontrolador un conformador de pulso, para garantizar un pulso cuadrado integro.

Bibliografía

- **BOYLESTAD** Robert – **NASHELSKY** Louis, Electrónica: Teoría y Circuitos, 6ta Edición, Prentice Hall Hispanoamericana, México, 1996.

- **CEKIT**, Curso Práctico de Circuitos Integrados, Colombia, 1994.

- **COUGHLIN** Robert - **DRISCOLL** Frederick, Amplificadores Operacionales y Circuitos Integrados Lineales, 4ta Edición, Prentice Hall Hispanoamericana, México, 1993.

- **GONZALEZ**, Introducción a los Microcontroladores, McGraw-Hill.

- **HALVORSON** Michael, Microsoft Visual Basic 4 paso a paso, McGraw-Hill.

- **LARCHEVÊQUE** Eric – **LELLU** Laurent, Montajes Avanzados para PC, París, 1997.

- **LILEN** H, Tiristores y Triacs, 4ta Edición, Marcombo S.A, España, 1981.

- **MALONEY** Thimothy, Electrónica Industrial Moderna, 3era Edición, Prentice Hall Hispanoamericana, México, 1997.

- **NATIONAL INSTRUMENTS,** The Measurement and Automation: Catalog 2000.

- **PHILIPS ECG**, Master Replacement Guide, 18ava Edición, 1998.

- **TOCCI** Ronald, Sistemas Digitales: Principio y Aplicaciones, 5ta Edición, Prentice Hall Hispanoamericana, México, 1993.

- **TURBO C++,** Versión 3.0, Borland International Inc, 1992.

- **UNIVERISIDAD CATOLICA DE CUENCA**, Lógica de Programación con Pseudocódigos y Flujogramas, 1era Edición, 1999.

- **VISUAL BASIC 6.0**, Microsoft Corporation, 1998.

- **WILLIAMS** Arthur, Manual de Circuitos Integrados, 1era edición, McGrawHill, USA, 1992.

- **Microcontroladores de 8 bits**, Atmel, 1999, <http://www.atmel.com/>

- **Center´s Design and library**, Protel, 2000, <http://www.Protel.com/>

- **Controlador electrónico aplicado**, Sinectics, 2001, <http://www.sinectis.com.ar/u/exem>

- **Microcontrollers**, Microchip, 2002, <http:microchip.com/>

- **Download´s tutorials**, Guille, 2002, <http://www.guille.com/>

- **Hardware interno del PC I-II-III y IV**, redeya,1998, <http://www.redeya.com/>

- **Interfacing the PC: Serial Port and Parallel Port**, Beyondlogic, 2001, <http://beyondlogic.com>

- **Serial Interface RS232**, Maxim, 2004, <http://maxim.com>

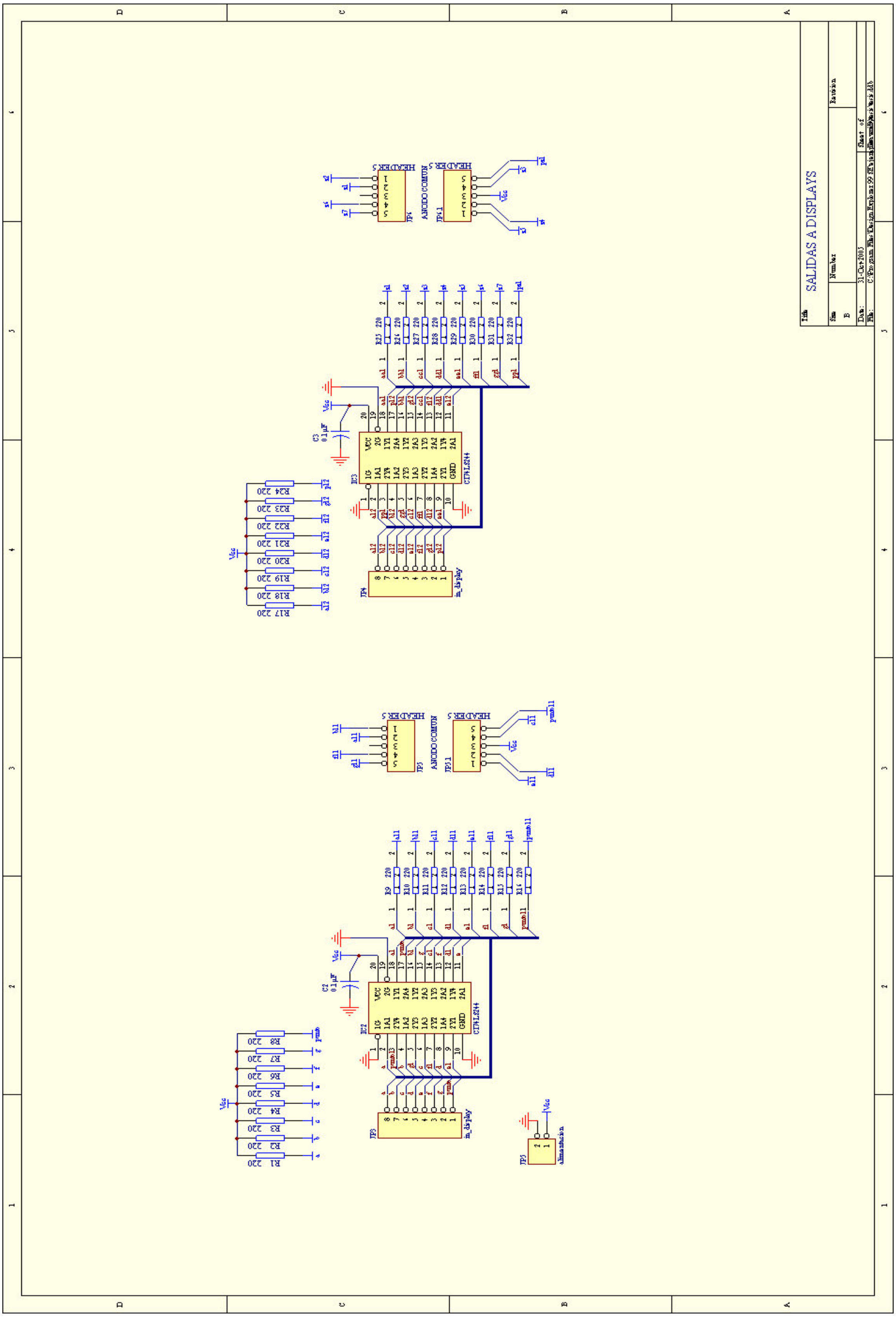
A N E X O

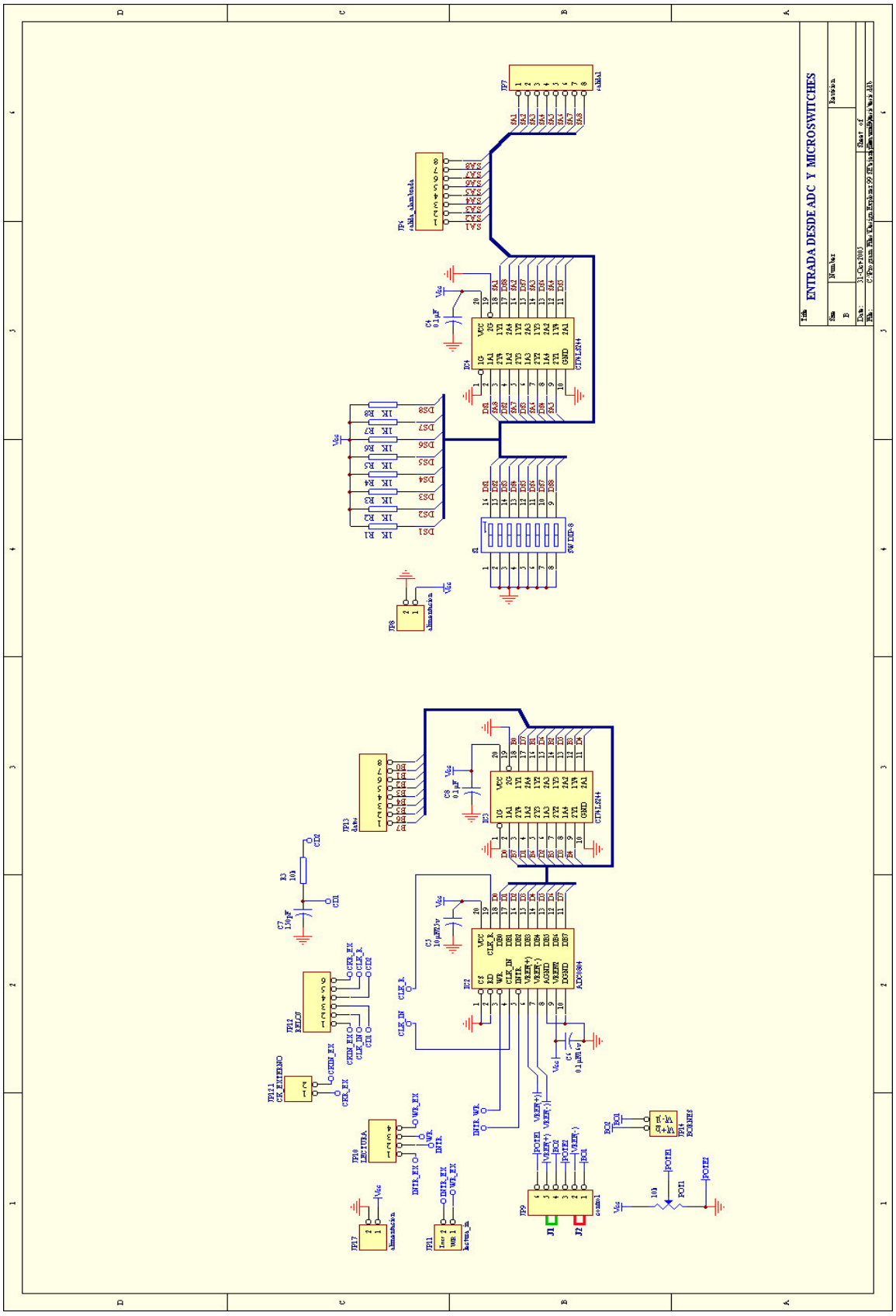
En el presente anexo se encuentra:

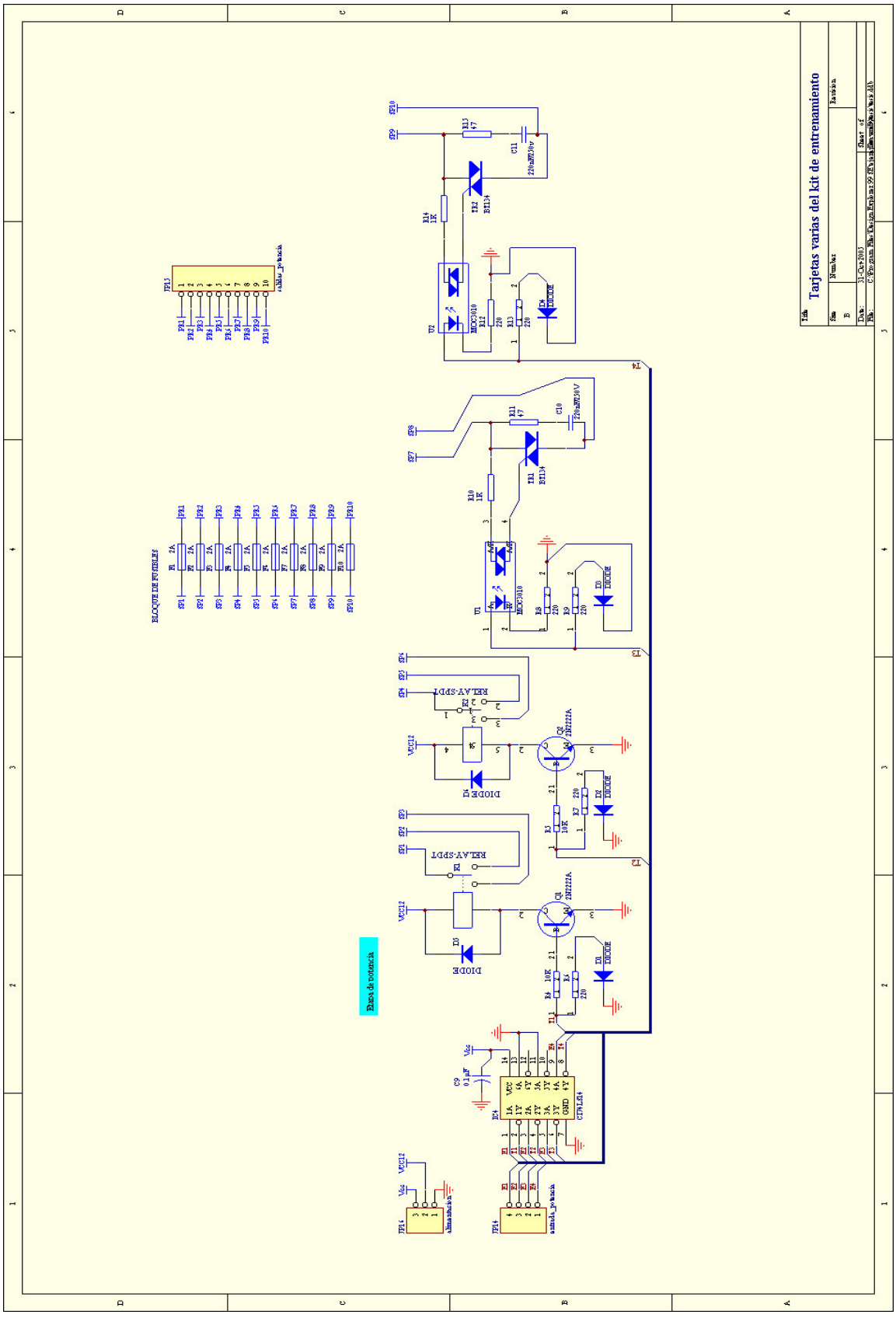
1. Los diagramas esquemáticos de los circuitos que se emplean en cada tarjeta del kit. Los diagramas de PCB de cada uno de los esquemáticos se hallan en la carpeta PCB del CD, tiene que abrirlos con un programa que lea "file.pcb".

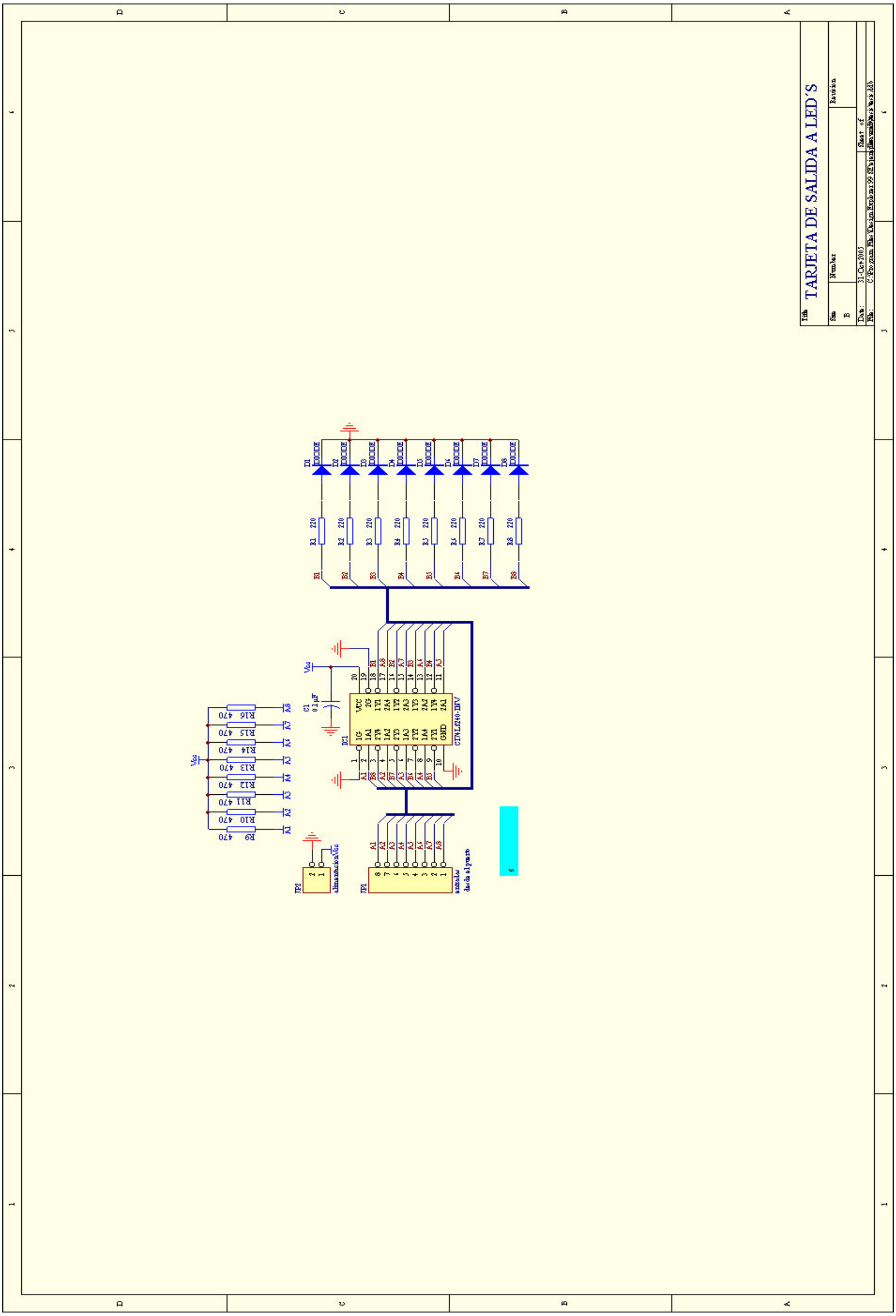
2. Se detallan los gastos que conlleva la elaboración de cada tarjeta; y, de éste trabajo de graduación.

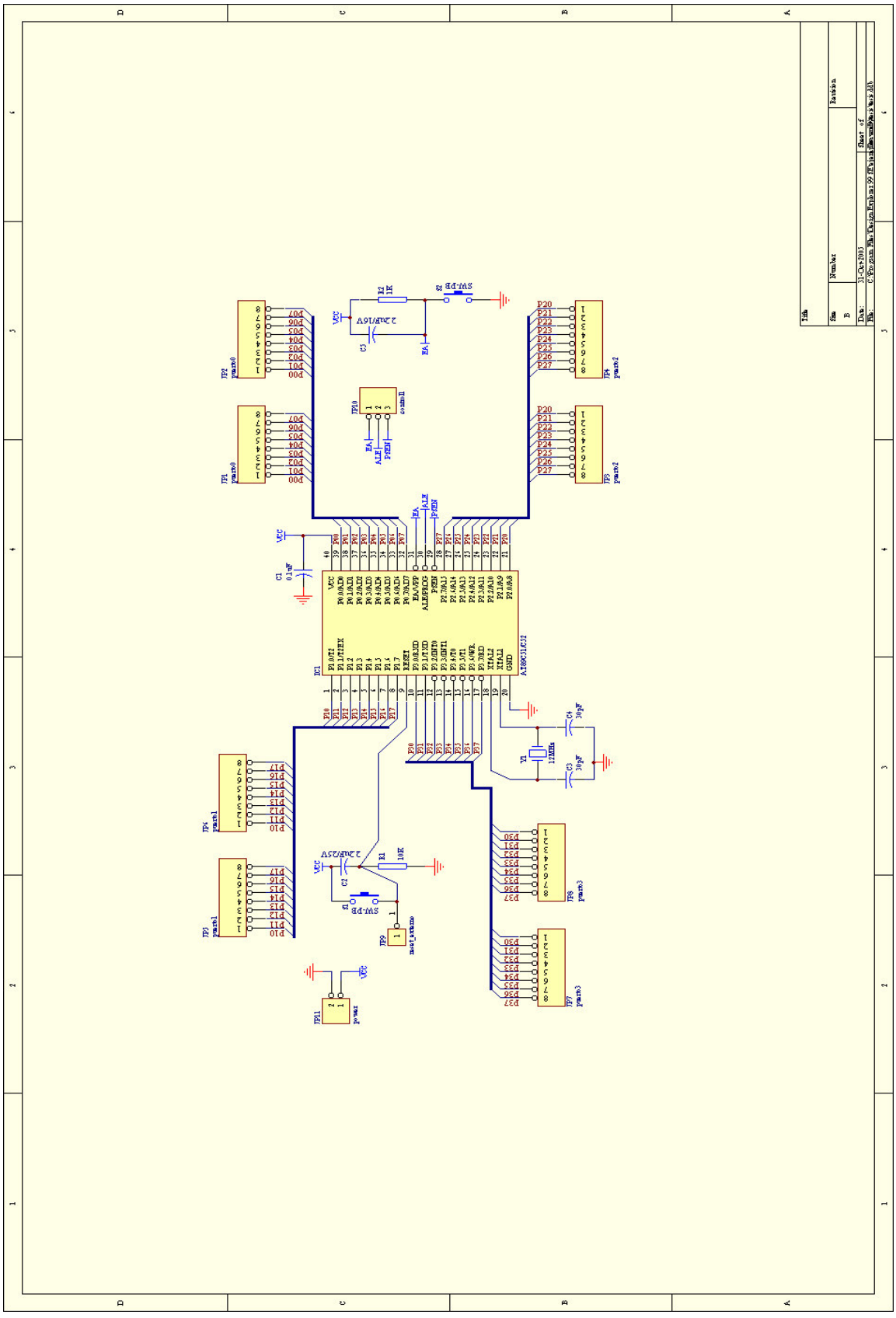
3. Más información relacionada se provee en el CD.

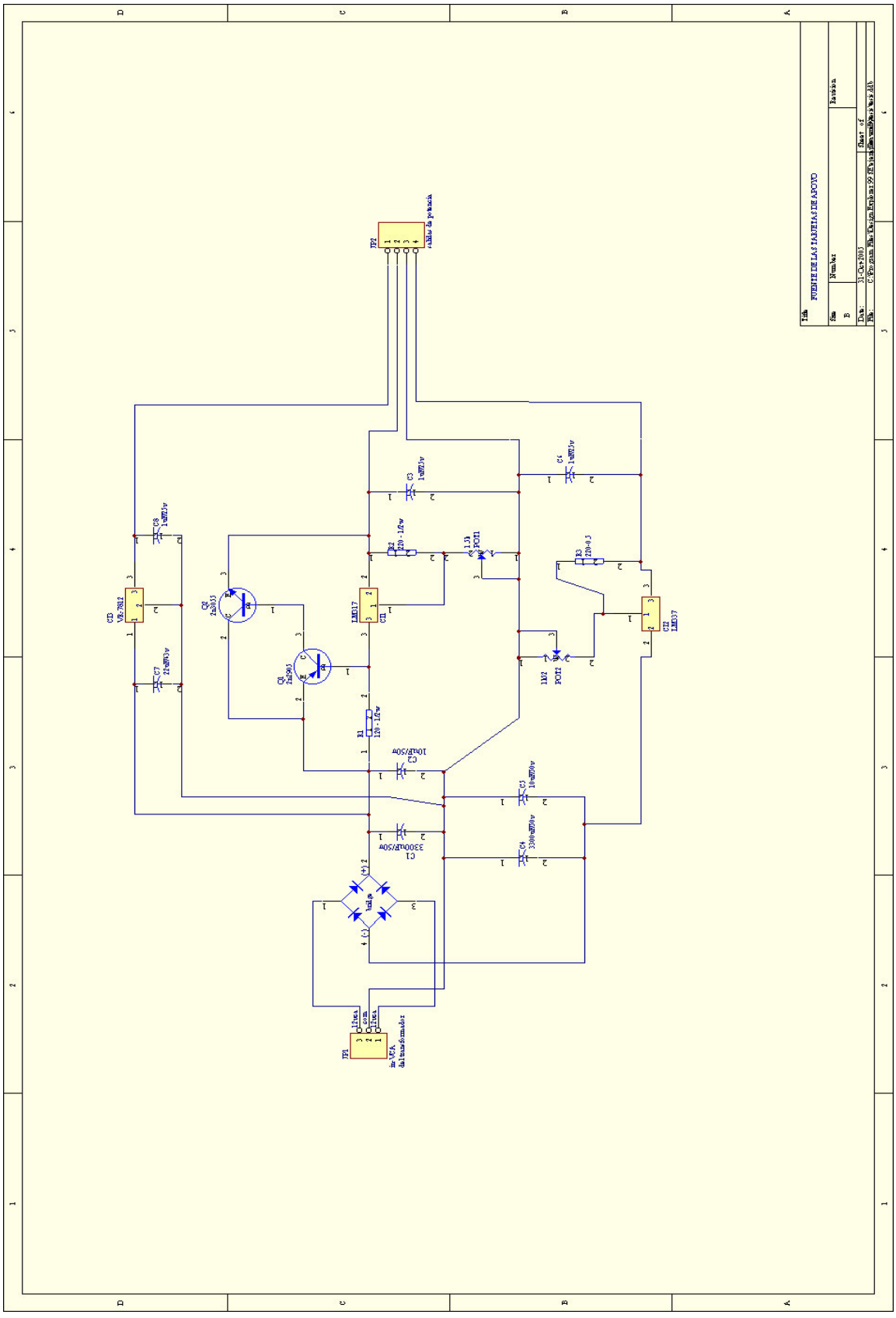

SALIDAS A DISPLAYS

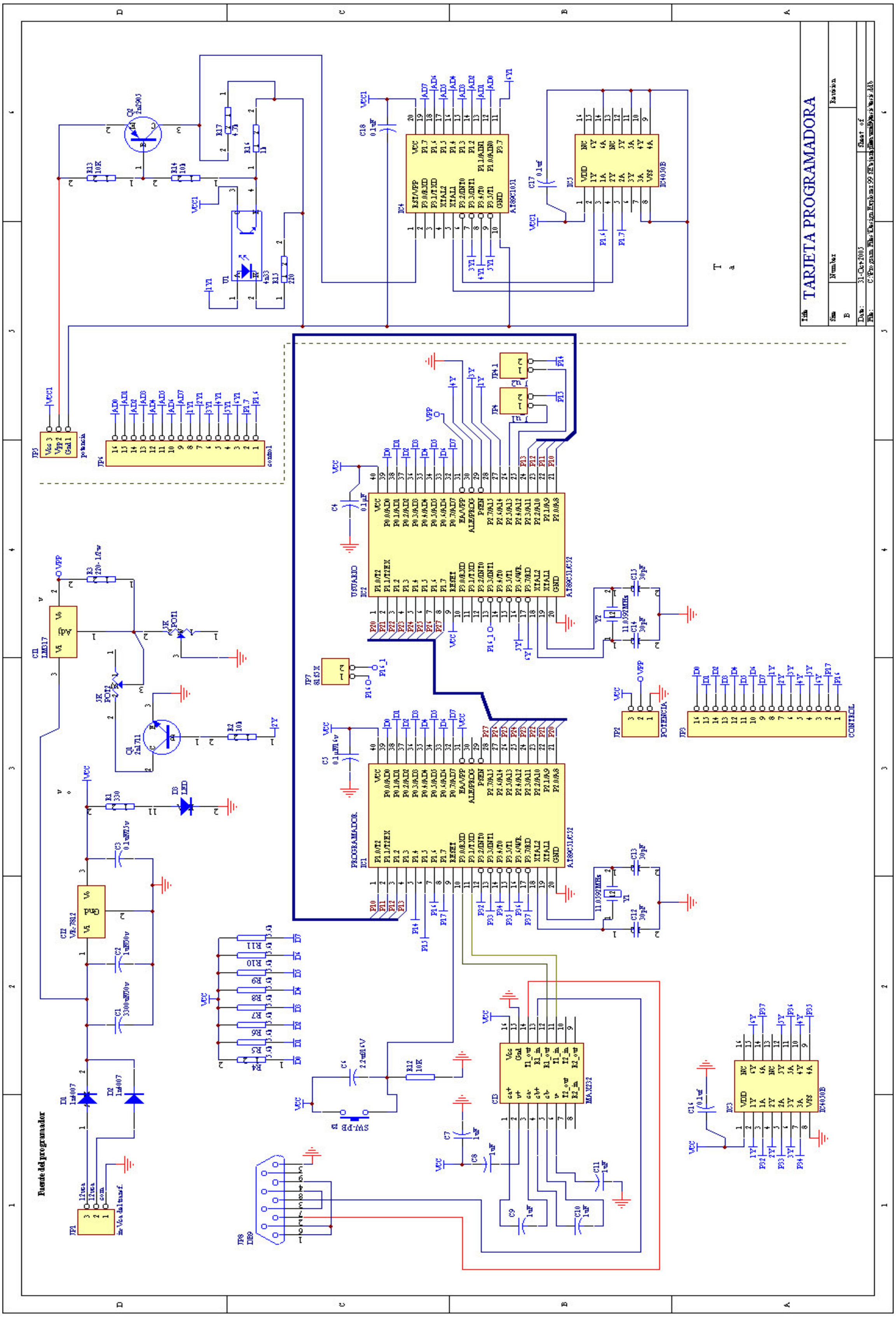

ENTRADA DESDE ADC Y MICROSWITCHES
CTPLS244
ADC0804
DIP-8


Tarjetas varias del kit de entrenamiento
Etapa de potencia
BLOQUE DE FUSIBLES
RELAY-SPDT
DIODE
1N2222A
MOC3010
74LS14


TARJETA DE SALIDA A LED´S






TARJETA PROGRAMADORA

Listado de Gastos por Tarjetas y de Oficina

Tarjeta de display

		valor unitario	valor total
1	display de ánodo común	1	1
16	resistencias de 220 1/2W +/- 5%	0.03	0.48
1	IDC de 8 terminales macho y hembra	1.35	1.35
1	base DIP-20	0.15	0.15
1	CI 74LS244	0.5	0.5
1	capacitor 0.1µF cerámico	0.08	0.08
		total 1 ==> $	**3.56**

Tarjeta de Sal-leds

		valor unitario	valor total
8	leds	0.12	0.96
16	resistencias de 330 1/2W +/-5%	0.03	0.48
1	base DIP-20	0.15	0.15
1	CI 74LS240	0.7	0.7
1	capacitor 0.1µF cerámico	0.08	0.08
1	IDC de 8 terminales macho y hembra	1.35	1.35
		total 2 ==> $	**3.72**

Tarjeta Ent-adc

		valor unitario	valor total
1	IDC de 8 terminales macho y hembra	1.35	1.35
2	bases DIP-20	0.15	0.3
1	CI ADC0804	2.9	2.9
1	CI 74LS244	0.5	0.5
1	capacitor de 0.1µF	0.08	0.08
1	condensador de 10µF/16V	0.08	0.08
1	condensador de 0.1µF/25V	0.08	0.08
1	capacitor de 500pF	0.1	0.1
1	potenciómetro de 10k	0.35	0.35
1	peineta	0.75	0.75
6	jumpers	0.2	1.2
1	base maquinada	0.45	0.45
1	resistencia de 10K 1/2W +/-5%	0.03	0.03
		total 3 ==> $	**8.17**

Tarjeta Ent-microswitches

		valor unitario	valor total
1	dipswitch de 8 pulsantes	0.75	0.75
8	resistencias de 330 1/2W +/-5 %	0.03	0.24
1	capacitor de 0.1µF cerámico	0.08	0.08
1	base maquinada	0.45	0.45
1	IDC de 8 terminales macho y hembra	1.35	1.35
1	base DIP-16	0.06	0.06
		total 4 ==> $	**2.93**

Tarjeta de Sal-potencia

		valor unitario	valor total
1	IDC de 4 pines macho y hembra	0.65	0.65
1	base DIP-14	0.12	0.12
1	CI 74LS04	0.4	0.4
4	leds rojos	0.12	0.48
4	resistencias de 330 1/2W +/-5%	0.03	0.12
4	resistencias de 10K 1/2W +/-5%	0.03	0.12
1	capacitor de 0.1µF cerámico	0.08	0.08
2	transistores 2N1711 (TO-39)	0.7	1.4
2	MOC3010	0.75	1.5
2	bases DIP-6	0.06	0.12
2	díodos rectificadores 1N4007	0.07	0.14
2	microrreles de 12V	1.1	2.2
2	resistencias de 47 1/2W +/-5%	0.03	0.06
2	resistencias de 1K 1/2W +/-5%	0.03	0.06
2	capacitores de 220nF / 250 V	0.25	0.5
2	triacs de 400PRV / 6A	0.75	1.5
10	bases para fusible	0.3	3
10	fusibles de vidrio de 2A / 250V	0.2	2
10	bornes para tarjeta impresa (bornera)	0.35	3.5
		total 5 ==> $	**17.95**

Tarjeta del Usuario

		valor unitario	valor total
4	bases maquinadas	0.45	1.8
4	IDC de 8 pines macho y hembra	1.35	5.4
1	capacitor de 0.1µF	0.08	0.08
2	condensador de 2.2µF/25V	0.08	0.16
1	resistencia de 10K 1/2W +/-5%	0.03	0.03
1	resistencia de 1K 1/2W +/-5%	0.03	0.03
1	base DIP-40	0.2	0.2
1	micropulsante NO	0.12	0.12
1	interrutor on-off	0.35	0.35
2	capacitor de 30pF	0.08	0.16
1	cristal de 12MHz	0.8	0.8
		total 6 ==> $	**9.13**

Tarjeta Programadora del AT89C1051

		valor unitario	valor total
1	conector de 3 terminales	0.3	0.3
1	conector de 17 terminales	0.8	0.8
1	base DIP-16	0.15	0.15
1	base DIP-20	0.15	0.15
2	capacitores de 0.1µF cerámicos	0.08	0.16
2	resistencias de 10k 1/2W +/-5%	0.03	0.06
1	resistencia de 5.6k 1/2W +/-5%	0.03	0.03
1	resistencia de 1k 1/2W +/-5%	0.03	0.03
1	base DIP-6	0.06	0.06
1	optotransistor 4N33	0.75	0.75
1	transistor 2N2905 (TO-39)	0.55	0.55
		total 7 ==> $	**3.04**

Tarjeta Programadora AT89C5X

		valor unitario	valor total
2	bases DIP-16	0.15	0.3
2	bases DIP-40	0.2	0.4
1	CI MAX-232	2.6	2.6
1	CI CD4050D	0.45	0.45
2	cristales 11.0592MHZ	0.8	1.6
4	capacitores de 30pF cerámicos	0.08	0.32
3	capacitores de 0.1µF cerámicos	0.03	0.09
8	resistencias de 5.6K 1/2W +/-5%	0.03	0.24
1	resistencia de 10k 1/4W +/-5%	0.03	0.03
1	resistencia de 10k 1/2W +/-5%	0.03	0.03
2	resistencias de 220 1/2W +/-5%	0.03	0.06
6	condensadores de 1µF/16V	0.08	0.48
1	condensador de 2.2µF/16V	0.1	0.1
1	condensador de 0.1µF/25V	0.08	0.08
1	micropulsante NO	0.12	0.12
1	transistor 2N2222A (TO-39)	0.75	0.75
2	diodos rectificadores 1N4007	0.7	1.4
1	led rojo	0.12	0.12
1	condensador de 3300µF/50V	1.4	1.4
1	CI 7805	0.4	0.4
1	CI LM317	0.6	0.6
2	trimmers de 10k	1.8	3.6
1	conector de 3 terminales	0.3	0.3
1	conector de 17 terminales	0.8	0.8
1	30 cm de cable plano de 24 hilos	0.5	0.5
3	peinetas	0.75	2.25
1	cable BD-9 (macho/hembra)	7.5	7.5
1	CI AT89C51	8.9	8.9
1	terminal DB-9 hembra para tarjeta	1.5	1.5
		total 8 ==> $	**36.92**

Fuente del kit

		valor unitario	valor total
1	transformador de 110/12-0-12 V 3A	3.75	3.75
1	transistor 2N3055	0.95	0.95
1	puente rectificador de 400PRV 100A	0.9	0.9
2	condensadores de 3300µF / 50V	1.4	2.8
2	condensadores de 10µF / 50V	0.1	0.2
1	resistencia de 120 1/2W +/-5%	0.03	0.03
2	resistencias de 220 1/2W +/-5%	0.03	0.06
2	potenciómetros de 1k	0.8	1.6
3	condensadores de 1µF / 50V	0.08	0.24
1	CI LM337	0.75	0.75
1	CI LM317	0.6	0.6
1	transistor 2N2905	0.55	0.55
1	condensador de 22µF / 63V	0.12	0.12
1	CI 7812	0.4	0.4
4	borneras hembras para banana	0.15	0.6
1	terminal de acople para VCA	0.45	0.45
1	cable para 120V	1.5	1.5
		total 9 ==> $	**15.5**

Varios

		valor unitario	valor total
1	tarro de pintura aerosol	2.5	2.5
50	tornillos M4	0.05	2.5
50	tuercas M4	0.05	2.5
1	caja de madera	10	10
1	rollo de estaño	5	5
1	pasta de soldar	2.5	2.5
1	cautín de 110V 40W tipo lápiz	4.5	4.5
4	bases de caucho	0.4	1.6
2	brocas de 1/8 de pulgada	0.85	1.7
1	elaboración de tarjetas formato doble	48	48
1	elaboración de tarjetas formato simple	35	35
1	broca de 3/8 de pulgada	2.5	2.5
		total 10 ==> $	**118.3**

Gastos de Oficina

		valor unitario	valor total
126	horas de consulta en Internet (aprox)	0.85	107.1
1164	copias	0.2	232.8
4	espiralados de copias	2.5	10
4	paquetes de 500 hojas A4 75gr	4.65	18.6
4	diskettes de 3 1/2	0.55	2.2
3	cartuchos de tinta negra	34	102
3	cartuchos de tinta a color	39.55	118.65
3	CD´ s RW	1.75	5.25
	varios		350
		total 11 ==> $	**746.6**

Gasto Total de elaboración de Tesis	USD	**1165.82**

RESUMEN

En la presente obra se empezará con todos los conocimientos teóricos que conlleva el estudio de un microcontrolador, esto es:

- *Hardware de la serie AT89*: corresponde a la constitución interna, relativo a puertos, manejo de memoria RAM, espacio de la memoria FLASH, procedimiento para los vectores de interrupción, seteos que se realizan en los timer/counter, y como se tiene que hacer para efectuar una comunicación serial.
- *Software de la serie AT89*: en ésta parte se estudiará lo relativo a la programación del microcontrolador, dando énfasis al conjunto de instrucciones que se tiene para poder realizar un programa.

De acuerdo a la estructura del manual, cada parte teórica se puede enlazar con una práctica. Permitiendo que la persona que lo lea vaya adquiriendo conocimiento y a la vez vaya practicando; porque al final se da los esquemáticos de los circuitos que debe armar para probar las prácticas. Además, se provee un programador con su respectivo software bajo ambiente windows, que le permitirá grabar un microcontrolador AT89C51, AT89C52, AT89C1051.

Todo esto facilitará el manejo de personas sin experiencia, pero con ganas de aprender.

Autor: Francisco Eduardo Alvarado Correa

ABSTRACT

In this work, I will begin with all the theoric knowledge, which help to study a microcontroller. That is:

- **Hardware series AT89:** belongs to the internal part, related to the parts, memory management RAM, memory flash space, interruption vectors processing, settings done in the timer/counter, and how we have to do to establish a serial comucation.

- **Software series AT89:** this part will have a study related to the microcontroller program, focus on the instructions needed to do a program.

According to the manual organization, each theoric part can be joined to a practical one. Giving people, who read it, an opportunity to acquire new knowledge and practicing then; because at the end they will have the circuit diagrams; which they have to use to practice.

Also, they will have a programmer with its software into windows, which will permit to record a microcontroller AT89C51, AT89C52 y AT89C1051.

All this will make the work easy, and help people to learn.

Author: Francisco Eduardo Alvarado Correa

Observaciones