UNIVERSIDAD DON BOSCO

FACULTAD DE TECNOLÓGICO

EMULADOR DE PLANTAS PROGRAMABLE PARA LA PRUEBA DE SISTEMAS DE CONTROL.

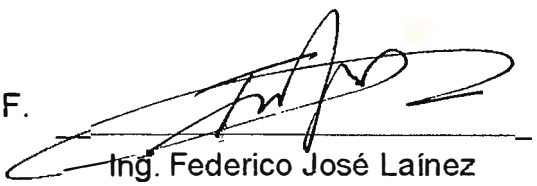
PRESENTADO POR:

HERNÁNDEZ RODRÍGUEZ, EDGAR DAVID

PARA OPTAR AL TÍTULO DE:
TÉCNICO EN INGENIERÍA ELECTRÓNICA

SOYAPANGO EL SALVADOR CENTRO AMÉRICA

JURADO CALIFICADOR:

F. 

Ing. Federico José Laínez

ASESOR

F. 

Tec. César Melgar Acosta

JURADO

F. 

Ing. Héctor Rubén Carías

JURADO

CIUDADELA DON BOSCO, SEPTIEMBRE DE 2003.

AUTORIDADES ACADEMICAS.

Rector:

Ing. Federico Miguel Hughet Rivera

Secretario General:

Lic. Mario Rafael Olmos Argueta

Decano de la Facultad de Estudios Tecnológicos:

Ing. Victor Arnoldo Cornejo Montano

CONTENIDO.

1. INTRODUCCIÓN.

Con la implementación de este proyecto estamos brindando un apoyo a la enseñanza técnica de la universidad Don Bosco en el área de electrónica específicamente en automatización y sistemas de control.

Nuestro objetivo es desarrollar un módulo que se adapte a las necesidades de la universidad para una mejor comprensión del contenido de los cursos, que van desde la aplicación de controles analógicos y/o digitales, hasta aplicaciones con autómatas programables (PLC).

El módulo tiene como base de operaciones el microprocesador Z80 ampliamente utilizado en aplicaciones de circuitos comerciales debido a sus buenas características eléctricas, bajo precio y facilidad de operación. El módulo cuenta también con memorias RAM para almacenar datos temporales y una ROM que contiene la base de operación del módulo en general, también cuenta con un puerto serial para la comunicación de la unidad central con una computadora si es necesario y un puerto paralelo para la conexión de las plantas de emulación.

Nuestro propósito al usar un microprocesador en el desarrollo del módulo, es el de hacer un entrenador dinámico y variable, es decir, que se pueda cambiar la aplicación y emulación de procesos análogos y digitales en un mismo módulo, y todo esto se logra sólo con un software y la adaptación de la pantalla de visualización con su respectiva tarjeta conectada a la unidad central del módulo por medio del puerto paralelo, esto es más práctico ya que no es necesario contar con un gran número de módulos para aplicaciones diferentes, sino un solo módulo para diferentes aplicaciones.

Las aplicaciones con que contará el módulo que hemos desarrollado serán una análoga y una digital.

La aplicación analógica, será la de un tanque de llenado de agua, cuyo caudal de entrada controlará la variación de nivel del líquido contenido en el mismo. Esto es representado, por una columna de leds que dan la apariencia de estar llenando un tanque posicionando un led después de otro sucesivamente.

La aplicación digital consta de un ascensor en un edificio de 8 pisos, el cual la emulación está sujeta a la selección de nivel de piso que se haya escogido. Donde éste se representa, por la columna de leds que se detienen cada tres leds (un piso), repartidos en un total de 8 pisos.

2. OBJETIVOS.

2.1. Objetivo general

- ✓ Brindar una herramienta para la enseñanza de los estudios tecnológicos de la Universidad Don Bosco.

2.2. Objetivos específicos:

- ✓ Diseñar un módulo especializado para el estudio del Control y Sistemas de Automatización que base sus operaciones en el microprocesador Z80.

- ✓ Diseñar un módulo capaz de comunicarse con una computadora por medio del puerto serial.

- ✓ Diseñar un módulo compatible con la lógica de los Autómatas Programables (PLC), para la prueba de sistemas de control.

- ✓ Construir el módulo propuesto con dos tarjetas de simulación implementadas como muestra (una que emule un proceso digital y otra un proceso analógico).

3. ALCANCES Y LIMITACIONES.

Se diseñará y construirá un módulo con las siguientes características:

3.1. Alcances:

- ✓ Sistema basado en el microprocesador Z80.
- ✓ Contará con un sistema monitor base.
- ✓ Contará con un puerto serial con el cual se podrá interconectar la Unidad Central del módulo y una computadora.
- ✓ Contará con un puerto paralelo con el cual se podrá interconectar la Unidad Central del módulo y las plantas de simulación ó para su uso general.
- ✓ Contará con dos puertos analógicos (uno de entrada y uno de salida), para uso general.
- ✓ Contará con una memoria ROM de 32 Kbits (expandible hasta 64 Kbits) que contendrá la base de operación del módulo en general.
- ✓ Contará con una memoria RAM de 16 Kbits (expandible hasta 32 Kbits) para almacenar datos temporales.
- ✓ Contará con leds indicadores para su uso en general y/o para indicar los procesos u operaciones del módulo.
- ✓ Contará con un Bus de expansión, para el manejo y control de dispositivos I/O ó de memoria externa (el cual proveerá las líneas del bus de control del microprocesador, líneas del bus de datos y líneas del bus de direcciones), para uso general.

- ✓ Contará con un botón de RESET, para reiniciar las operaciones del sistema en cualquier momento.

- ✓ El módulo contará con dos aplicaciones implementadas como muestra.

3.2. Limitaciones:

- ✓ El módulo no contará con grabado de memorias EPROM.

- ✓ El módulo no contará con fuente de alimentación, por lo que deberá ser conectado a una fuente externa.

- ✓ El diseño de los programas está limitado a la capacidad de la cual se dispone con el microprocesador Z80, por lo que la complejidad de las simulaciones estarán sujetas a las funciones y operaciones ejecutables por éste.

- ✓ La velocidad máxima del sistema estará restringida a la velocidad de operación del microprocesador Z80.

- ✓ El sistema no contará con adaptación de niveles de voltaje de operación de otros sistemas no compatibles con TTL (por ejemplo los valores de operación de los Autómatas Programables, normalmente de +12V ó +24V), por lo cual cada tarjeta de emulación debe poseer las adaptaciones necesarias para asegurar la compatibilidad con los sistemas para los cuales fue diseñada.

4. JUSTIFICACIÓN.

La propuesta nace ante la necesidad de impartir una mejor enseñanza por parte de la Universidad Don Bosco, al brindar con ella una capacitación más especializada de la que se imparte hasta ahora.

El objetivo que se persigue con su diseño e implementación, es el de reforzar los estudios sobre Control y Sistemas de Automatización al contar con un sistema que se adapte completamente a los objetivos de los cursos en general y de la Universidad.

Este proyecto toma su relevancia al ofrecer ventajas con las que hasta ahora no se cuentan en los laboratorios de Control y Sistemas de Automatización. Entre estas ventajas, esta la de ser un entrenador versátil, es decir, que es un módulo dinámico que puede cambiar su desempeño para funcionar de diferentes formas con sólo cambiar un software y una pantalla de visualización de la planta que se emula. Una de las características más sobresalientes es que el módulo es muy práctico, ya que debido al uso del microprocesador Z80, se pueden emular tanto procesos analógicos como digitales, para una mayor cobertura de los Sistemas de Automatización y de Control.

Otra ventaja sobresaliente es que no es necesario contar con un gran número de módulos con una aplicación diferente cada uno, sino uno sólo con capacidad para diferentes aplicaciones, lo cual significa un importante ahorro de recursos en la adquisición de estos. También es importante resaltar, que el módulo propuesto ofrece enormes ventajas a los cursos impartidos fuera de la Universidad, ya que resulta más conveniente y portátil que los sistemas convencionales (esto se debe a que es un sistema emulado y no necesita requerimientos especiales de energía, temperatura, ó condiciones específicas de operación, sólo una tarjeta de visualización que pueda ser adaptada a la

Unidad Central del módulo por medio del puerto paralelo y un software adecuado)

El sistema propuesto es una innovación que pretende dar solución a las necesidades de la Universidad Don Bosco, al ofrecer una gran ventaja en conveniencia, versatilidad y ahorro en la implementación de éstos. Hasta ahora se ha optado por la adquisición de equipos caros y limitados, que son utilizados para el estudio de determinadas materias y contenidos debido a su limitada aplicación y versatilidad.

5. DESCRIPCIÓN DEL PROYECTO.

5.1. Generalidades.

El módulo se diseñó para la prueba de sistemas de control. Donde un sistema, es una combinación de componentes que actúan conjuntamente y cumplen un determinado objetivo. De estos sistemas de control hay una gran variedad, pero el control para el cual estarán destinadas las emulaciones del tanque y del ascensor de edificio, serán los sistemas de control de procesos. Este es un sistema de regulación con las características de un control automático, el cual, es un sistema de control retroalimentado (lazo cerrado), cuya tarea fundamental, es mantener la salida constante (ó controlada) a pesar de las posibles perturbaciones presentes en la planta. Sin embargo, es fácil apreciar que si no se retroalimenta la salida de la planta, el sistema de control se vuelve de lazo abierto, en el cual por definición, la salida no tiene efecto sobre la accion de control.

El módulo cuenta con un microprocesador como dispositivo controlador del sistema en general, el Z80 CPU de Zilog.

El Z80 es un microprocesador de tercera generación, el cual, es más eficiente en la utilización de la memoria que otros dispositivos de su misma clase. El programador tiene acceso a 208 bits de lectura/escritura de registros internos, también cuenta con un registro de interrupciones, dos registros índices, un registro de refresco (contador), un contador de programa y un puntero de pila; el Z80 es muy versátil y funcional, muestra de ello, es que sólo necesita de +5V para funcionar, todas sus señales de salida son decodificadas y temporizadas de acuerdo al control de las memorias y controladores de periféricos estándares, ya el tiempo es crucial para asegurar que el contenido de la posición de la memoria esté en el bus de datos, cuando el CPU lee dicho bus. Las operaciones aritméticas que el Z80 puede ejecutar son: suma, resta, operación lógica AND, operación lógica OR, operación lógica OR exclusiva,

comparación, desplazamiento hacia la izquierda, desplazamiento hacia la derecha, rotación, incremento y decremento. Los controladores de los buses de datos y direcciones vigilan todas las actividades relacionadas con el intercambio de datos entre el CPU y sus periféricos mediante sus buses respectivos. El bus de datos es bidireccional, el bus de direcciones es unidireccional (esto es porque el CPU no recibe datos en el bus de direcciones).

El diagrama básico del microprocesador Z80, su distribución de pines, la dirección de éstos (de entrada ó salida) y su clasificación (en cuanto a que bus pertenecen), se muestran en la figura siguiente:

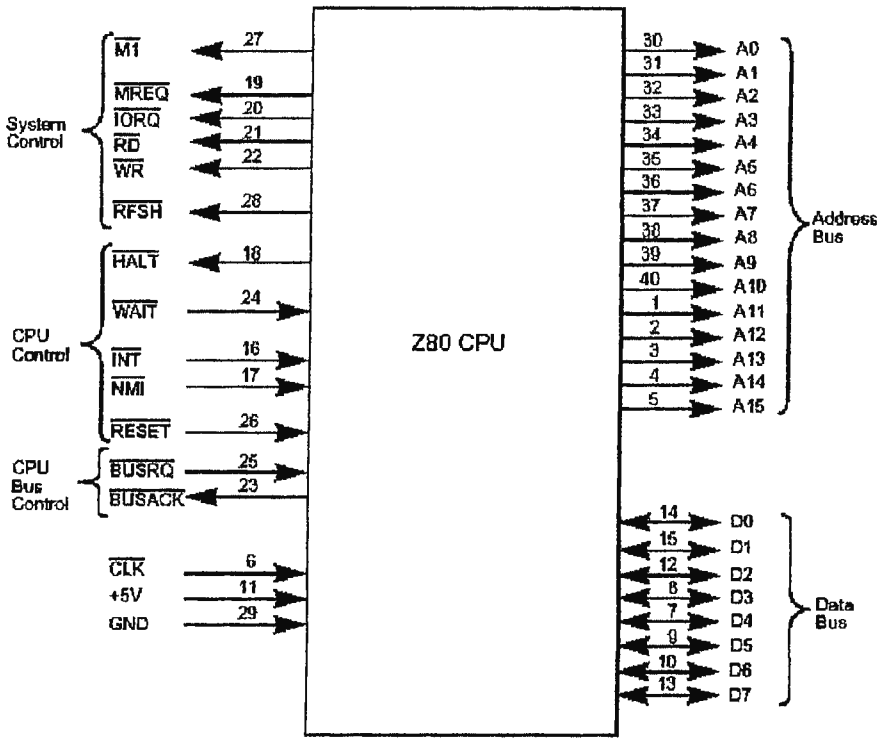


Fig. 1 Configuración de pines del microprocesador Z80

La asignación de pines es la siguiente:

A0 – A15: salida de tres estados, activas en alto; constituyen un bus de direcciones de 16 bits, este proporciona un direccionamiento de hasta 64 k bits

(2^{16} bits), para el intercambio de datos entre dispositivos de I/O. A0 es el bit de direccionamiento menos significativo.

D0 - D7: Entrada/salida de tres estados, activos en alto, constituyen un bus de datos bidireccional de 8 bits. El bus de datos se utiliza para el intercambio de datos con la memoria y los dispositivos de I/O.

M1: Salida activa en nivel bajo. M1 indica que el ciclo de maquina actual es el ciclo de búsqueda del código de operación en la ejecución de una instrucción. M1 también se produce con IORQ para indicar el reconocimiento de un ciclo de interrupción.

MREQ: Salida de tres estados, activa a nivel bajo. La señal de petición de memoria indica que el bus de direcciones mantiene una dirección valida para poder efectuar una operación de lectura ó escritura.

IORQ: Salida de tres estados, activa nivel bajo, la señal de IORQ indica que la mitad baja del bus de direcciones mantiene una dirección valida de I/O para efectuar una operación de lectura/escritura de I/O. también se genera una señal de IORQ con una señal M1 cuando se esta reconociendo una interrupción para indicar que el vector de respuesta de la interrupción puede ser colocado en el bus de datos, las operaciones de reconocimiento de las interrupciones se producen en el tiempo M1 mientras que las operaciones de I/O nunca se producen durante un tiempo M1.

RD: salida de tres estados, activa en nivel bajo. RD indica que el CPU desea leer datos desde la memoria ó desde un dispositivo I/O. El dispositivo de I/O ó la memoria debe utilizar esta señal para dirigir los datos al bus de datos del CPU.

WR: Salida de tres estados, activa en nivel bajo. WR indica que el bus de datos de la CPU mantiene un dato válido para ser almacenado en la memoria direccionada ó en el dispositivo de I/O.

RFSH: Salida activa en nivel bajo. RFSH indica que los 7 bits de menos peso del bus de direcciones contienen una dirección de refresco para las memorias dinámicas y que la señal actual de MREQ debe ser utilizada para efectuar una lectura de refresco para todas las memorias dinámicas.

HALT: Salida activa en nivel bajo. HALT indica que el CPU ha ejecutado una instrucción de software HALT y que está aguardando por una interrupción no enmascarable ó enmascarable mientras ésta señal en ese estado el CPU ejecuta instrucciones NOP para mantener la memoria en estado de refresco.

WAIT: Entrada activa en nivel bajo. WAIT le indica al CPU que la memoria direccionada ó los dispositivos de I/O no están preparados para una transferencia de datos. El CPU continúa entrando estados de espera durante todo el tiempo en que esta señal es activa. Ésta señal permite que se pueda sincronizar con el CPU con memorias ó dispositivos de I/O de cualquier velocidad.

INT: Entrada activa en nivel bajo. La señal de petición de interrupción es generada por los dispositivos de I/O. Cuando la CPU acepta la interrupción, se envía una señal de reconocimiento al principio del ciclo de instrucciones.

NMI: Entrada activa en nivel bajo. La línea de interrupción no enmascarable tiene una prioridad mas alta que INT y siempre es reconocida al final de la instrucción que se está ejecutando.

RESET: Entrada activa en nivel bajo. RESET fuerza al contador de programa a cero e inicializa el CPU. Durante el tiempo de reset, el bus de direcciones y el bus de datos se quedan en un estado de alta impedancia y todas las señales de control de salidas pasan al estado inactivo.

BUSRQ: Entrada activa en nivel bajo. La señal de petición de bus se utiliza para solicitar que el bus de direcciones del CPU, el bus de datos y las señales

de tres estados de control de salida, vayan a un estado de alta impedancia de forma que otros dispositivos puedan controlar estos buses (DMA). Cuando la señal BUSRQ es activada, el CPU colocará los buses en un estado de alta impedancia en el momento que el ciclo de máquina actual del CPU termine.

BUSAK: Salida activa en nivel bajo. El reconocimiento del bus se utiliza para indicar al dispositivo que lo pide, que los buses de dirección del CPU, el bus de datos y las señales de control del bus de tres estados han sido colocadas a su estado de alta impedancia y que el dispositivo externo puede controlar ahora estas señales.

Otro dispositivo que toma relevancia dentro del diseño del módulo (por no ser de uso tan común), es la UART (Universal Asynchronous Receiver/Transmitter, Receptor/Transmisor Universal Asíncrono), el cual convierte los datos que recibe desde algún periférico en forma paralela a datos de forma serial asíncrona a su salida. En nuestro módulo, el UART convierte los datos paralelos que recibe desde el microprocesador Z80 a datos en forma serial asíncrona para la comunicación con una computadora a su salida. El Z80 puede leer el estado de la UART en cualquier momento durante su operación funcional. La información de estado reportada, incluye el tipo y condición de transferencia de las operaciones desarrolladas por la UART, así como también las condiciones de error.

El diagrama básico de la UART, su distribución de pines y la dirección de éstos (de entrada ó salida), se muestran en la figura siguiente:

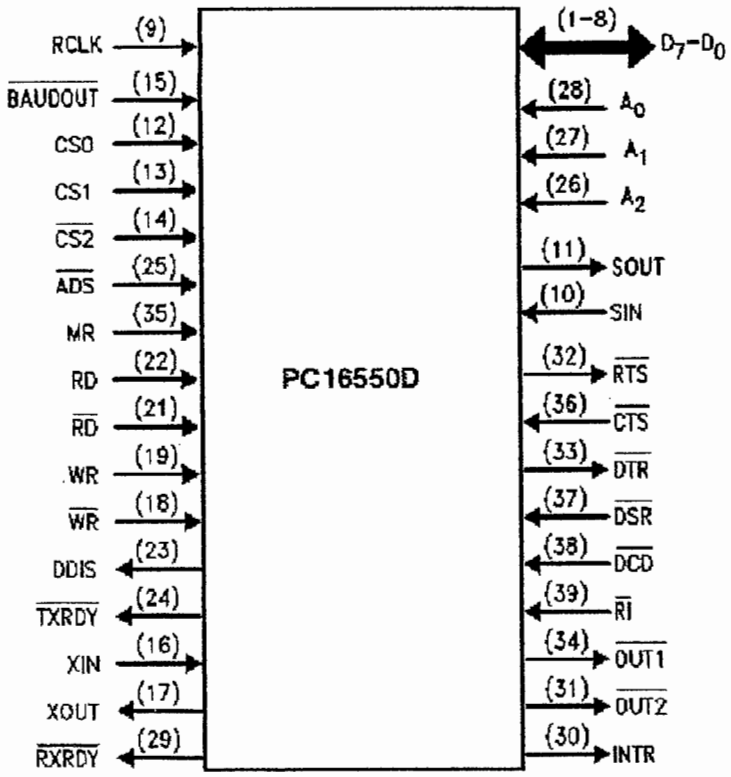


Fig. 2 Configuración de pines del UART

La asignación de pines es la siguiente:

A0, A1, A2: Selección de registros. Señales de entrada para líneas de dirección, que conectadas a estas tres entradas, seleccionan un registro de la UART para que el CPU pueda leer ó escribir dichos registros durante la transferencia de datos.

ADS (adress strobe): la transición positiva de ésta señal, activa la captura de la dirección a sus entradas para la selección de registro (A0, A1, A2) y las señales de selección del chip (CS0, CS1, CS2). Ésta línea se utiliza para mantener estables las señales antes mencionadas internamente en la UART, en el caso de que éstas no pudieran mantenerse estables por mucho tiempo en los pines de entrada correspondientes (durante la transferencia de datos).

BAUD OUT: ésta es una señal de reloj x 16 proveniente de la sección de transmisión de la UART.

CS0, CS1, CS2 (chip select): cuando CS0 y CS1 son altos y CS2 es bajo, el chip es seleccionado y esto habilita la comunicación entre la UART y el CPU.

CTS (clear to send): cuando se encuentra en estado bajo, indica que el módem ó el set de datos está listo para intercambiar datos.

D7–D0 (data bus): este bus comprende ocho líneas de entrada/salida triestados. Este bus, provee comunicación bidireccional entre la UART y el CPU. Datos, palabras de control e información de estado son transferidas a través del bus de datos.

DCD (data carry detect): cuando se encuentra en estado bajo, indica que ha sido detectado una portadora de datos por el módem ó el set de datos.

DDIS (driver disable): cuando ésta señal se va hacia un nivel bajo, indica que el CPU está leyendo datos desde la UART.

DSR (data set ready): cuando se encuentra en estado bajo, indica que el módem ó el set de datos está listo para establecer comunicación con la UART.

DTR (data terminal ready): cuando se encuentra en estado bajo, le indica a el módem ó set de datos que la UART está lista para establecer comunicación.

INTR (interrupción): esta señal pasa a su estado alto, cuando el dispositivo genera una interrupción al tener una condición ó bandera activa y será aceptada y trasladada al CPU, si se encuentran habilitadas la interrupciones por el IER (Registro de Habilitación de Interrupciones).

MR (Master reset): cuando este pin pasa a su estado alto, se limpian todos los registros (excepto el buffer de recepción, transmitting holding y el divisor latch) y el control lógico de la UART.

OUT 1: salida diseñada para ser utilizada por el usuario.

OUT 2: salida diseñada para ser utilizada por el usuario.

RCLK (Receiver clock): es una entrada de 16 x baud rate clock, para la sección de recepción de clock del chip.

RD, RD\ (read): cuando RD es alto ó RD\ es bajo, mientras que el chip es seleccionado, el CPU puede leer la información de los estados ó datos desde el registro seleccionado de la UART.

RI (ring indicator): cuando este pin pasa a su estado bajo, indica que una señal de teléfono ha sido recibida por el módem ó el set de datos.

RTS (request to send): cuando este pin pasa a su estado bajo, informa al módem ó set de datos que la UART esta lista para intercambiar datos.

SIN (serial input): entrada serial de datos.

SOUT (serial out): salida de datos seriales de comunicación.

TXRDY, RXRDY: transmisión y recepción de datos de señales DMA están disponibles a través de estos dos pines.

XIN (external cristal input): esta señal es usada en conjunto con XOUT para formar un circuito realimentado para el generador oscilador de bauds.

XOUT (external cristal output): la señal de salida es usada junto con XIN para formar un circuito de realimentación para el oscilador del generador de bauds.

WR, WR\ (write): cuando WR es alto ó WR\ es bajo, mientras el chip es seleccionado, el CPU puede escribir palabras de control ó datos en el registro seleccionado de la UART.

En el desarrollo del módulo, también se utilizaron convertidores digital/análogos (DAC) y análogos/digitales (ADC). Los DAC se utilizan normalmente, a las salidas digitales de una computadora, porque convierte los valores digitales que recibe (provenientes del microprocesador) y los pasa a un valor de corriente ó voltaje proporcional al dato digital a su salida. El ADC tiene como entrada una señal analógica de corriente ó de voltaje, que convierte a una señal digital que pueda ser leída por una computadora, esto en general, se utilizará para nuestra aplicación en la emulación de una planta analógica que será un tanque de agua, el cual varia el caudal de entrada proporcionalmente con una entrada de voltaje de 0 a 5V, para controlar el nivel de la altura del líquido contenido. En este proceso usaremos un ADC que emplea el método de aproximaciones sucesivas, para tener un tiempo de conversión razonablemente pequeño, además de tener, un tiempo fijo de conversión que no depende de la señal analógica, es decir, el ADC utiliza un registro, el cual es modificado bit por bit hasta que el contenido de éste, se convierte en el equivalente digital de la entrada análoga dentro de la resolución del convertidor.

El módulo cuenta con un puerto paralelo, con el cual se podrán conectar las tarjetas para la emulación de las plantas. Para este fin, utilizamos una interfaz periférica programable (PPI) 8255, que será la encargada de comunicar de forma paralela otros periféricos con el microprocesador Z80, además de ser muy eficiente por tener un puerto con entradas y salidas múltiples que se pueden configurar por software en sus tres diferentes modos. También, es posible ampliar las capacidades del módulo por medio del puerto de expansión, por el cual se puede tener acceso externo a las líneas principales del sistema. En él se encuentran disponibles: el bus de datos, bus de direcciones así como el bus de control; todo esto con el fin de tener acceso a las líneas del microprocesador para hacer del módulo una herramienta fácil de interactuar con el usuario.

El módulo cuenta con memorias de uso general RAM y ROM, también cuenta con la salida serial adaptada al estándar RS232 para la comunicación

con una computadora, además tiene dos puertos analógicos uno de entrada y otro de salida de uso general, leds indicadores y un oscilador generado por cristal que se utiliza como reloj general del sistema.

Este sistema está diseñado para la capacitación y estudio de sistemas de automatización y control, cuenta con un programa monitor base, que regula las actividades del sistema en general (sistema operativo). Por medio de la conexión a una computadora, por el puerto serial del módulo, se pueden regular las actividades de éste, además de brindar la oportunidad de interactuar ambos en forma dinámica.

Para el uso del módulo basta con colocar el software adecuado en la memoria ROM y adaptable la tarjeta de emulación que debe tener el circuito específico de una planta para la visualización del proceso por medio de leds. En las prácticas de laboratorio a realizar basta con tener presentes las indicaciones anteriores y conectar el controlador diseñado en la práctica a la tarjeta de emulación y entonces se podrá dar inicio al proceso de emulación y a la operación de control.

5.2. Estructura interna del módulo.

El módulo está estructurado como se muestra en el siguiente diagrama en bloques:

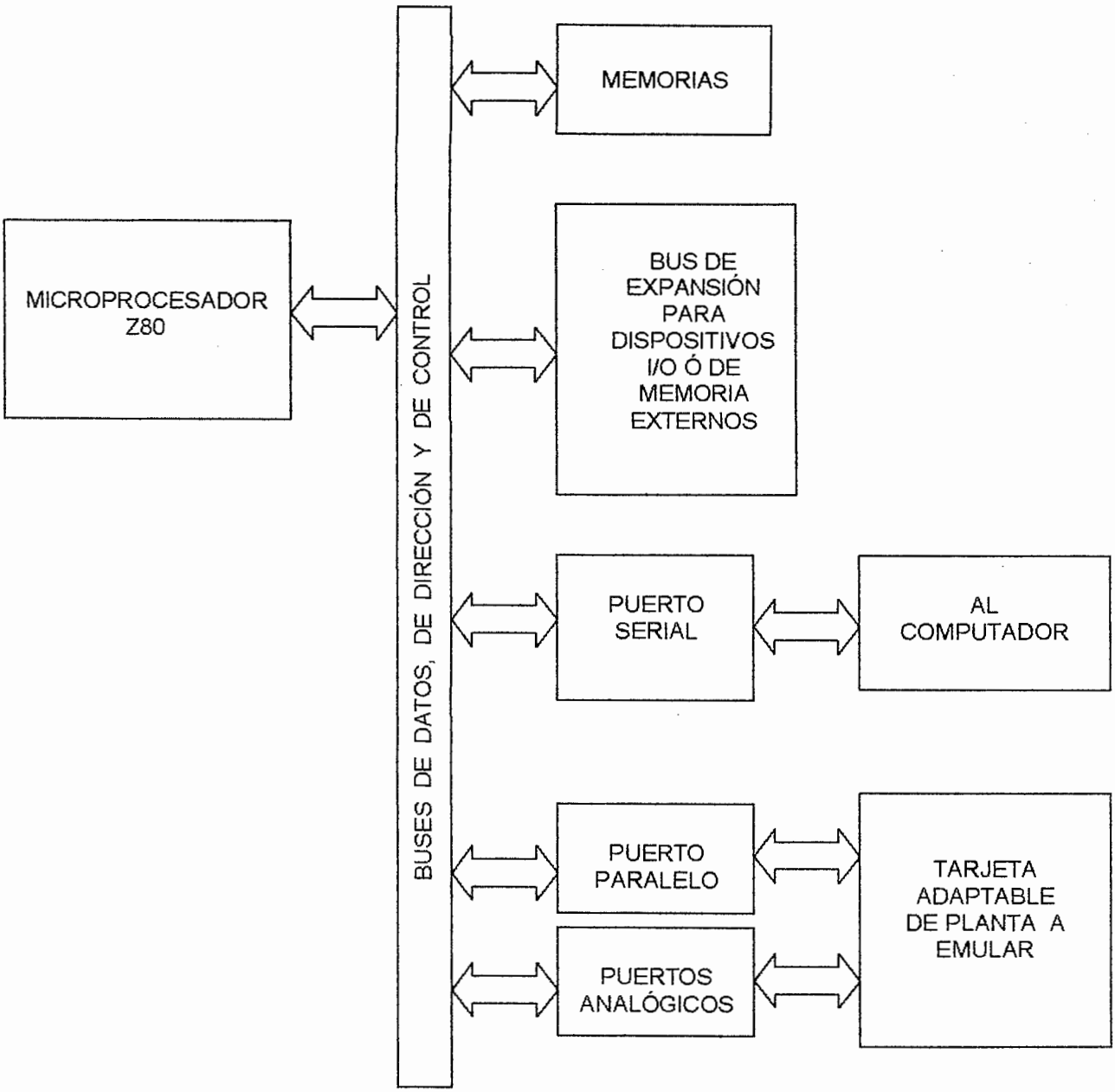


Fig. 3 Diagrama en bloques del módulo.

A continuación se describe las consideraciones que se tomaron para el diseño y la función de cada una de éstas partes.

5.3. Puertos Analógicos.

5.3.1. Puerto Analógico – digital (A/D).

Para el diseño del puerto analógico de entrada, se empleó un

convertidor analógico – digital ADC0804, el cual utiliza tecnología CMOS y lleva a cabo la conversión A/D utilizando el método de conversión por aproximaciones sucesivas. El circuito se muestra en la figura siguiente:

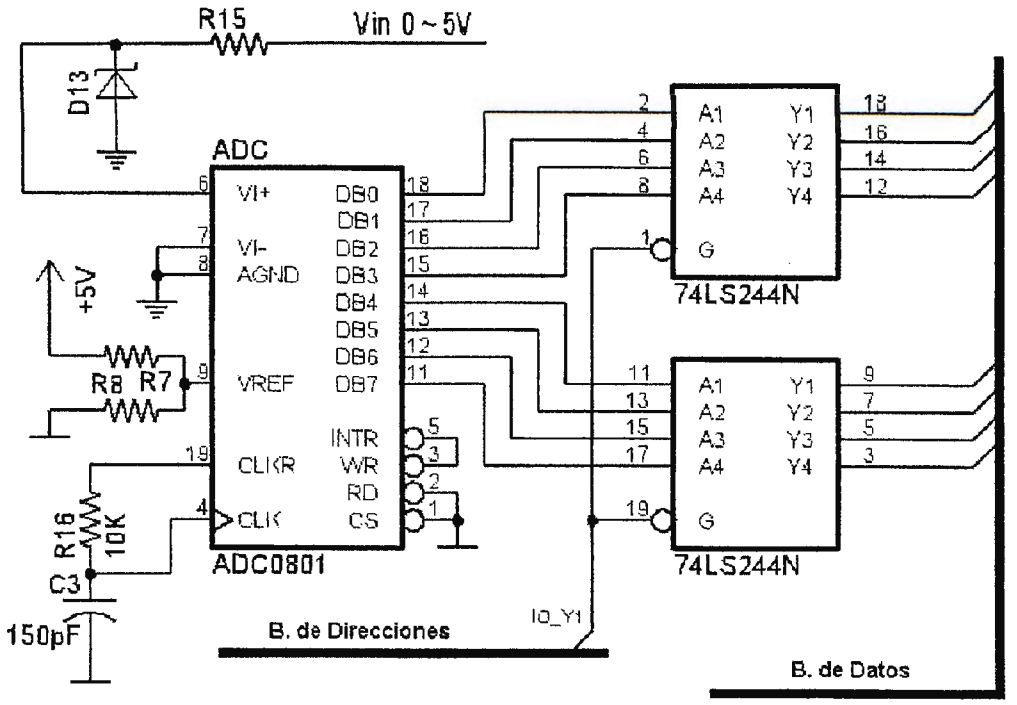


Fig. 4 Puerto Analógico de Entrada

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- El CI ADC0804 está diseñado para ser conectado con facilidad al canal de datos de un microprocesador, es decir, está diseñado básicamente para trabajar con microprocesadores de 8 bits (por ejemplo el Z80).
- Este convertidor A/D es ampliamente utilizado para fines didácticos.
- Es un convertidor fácil de encontrar en el mercado y es de costo muy accesible.
- Ocho bits de resolución, es suficiente para la mayoría de aplicaciones básicas de moderada exactitud (por ejemplo la simulación del llenado de un tanque de agua, la cual es una de las aplicaciones de muestra del funcionamiento del presente módulo).
- El convertidor analógico - digital se configuró en modo de conversión continua (pin $\overline{INTR}$ conectado directamente con $\overline{WR}$, mientras $\overline{RD}$ y

$\overline{CS}$ son conectados a tierra), con ello se logra que el microprocesador no tenga que esperar por el tiempo de conversión del CI (aunque es relativamente bajo) y puede tomar en cualquier momento el último dato de conversión sin recurrir a demoras de tiempo e interrupciones de fin de conversión innecesarias.

- Se accesa a los datos de salida del convertidor por medio de un buffer de tres estados (74LS244), ya que el pin $\overline{CS}$ del ADC0804 está conectado permanentemente a tierra (debido a la configuración free running, expuesta en el literal anterior).
- La frecuencia de reloj es generada por una red RC externa (R16 y C3), con una frecuencia de trabajo, que se calcula por medio de la siguiente fórmula:

$$f \cong \frac{1}{1.1RC} = \frac{1}{1.1*10pF*100K\Omega} = 606KHz$$

Con ello se logra un tiempo de conversión de aproximadamente 100μS, el cual, es suficiente para la mayoría de aplicaciones (por ejemplo fines didácticos).

- Se asignó un voltaje de referencia de Vcc/2 (por medio de un divisor de voltaje generado por R7 y R8), para obtener un rango de voltaje de entrada analógico de 0 a 5V.
- Para la protección de CI ADC0804 contra sobrevoltajes, se empleó un diodo zener y una resistencia zener para la regulación del voltaje de entrada. Donde:

Dz: 1N4733A $Pz_{(max)} = 1W$ $Vz = 5.1V$ $R15 = 47\Omega$

$PRz_{(max)} = 1W$

$Iz_{(max)} = \frac{Pz}{Vz} = \frac{1W}{5.1V} = 196mA$, entonces:

$PRz = (Iz)^2 R15 = (Iz)^2 (47) = 1W \Rightarrow Iz \cong 145mA$, lo cual no excede $Iz_{(max)}$

Luego se calcula el máximo voltaje de entrada que no causa daño al CI:

$Vin_{(max)} = Vz + VR15_{(max)} = 5.1V + (0.145A)(47\Omega) = 5.1V + 6.85V = 11.95V \cong 12V$

5.3.2. Puerto Digital – Analógico (D/A).

Para el diseño del puerto analógico de salida, se empleó un convertidor digital – analógico DAC0808, el cual genera a su salida una corriente (con polaridad negativa) en función del dato digital a su entrada y que cuenta con un tiempo de conversión bajo y constante (150nS típico).

El circuito empleado se muestra en la figura siguiente:

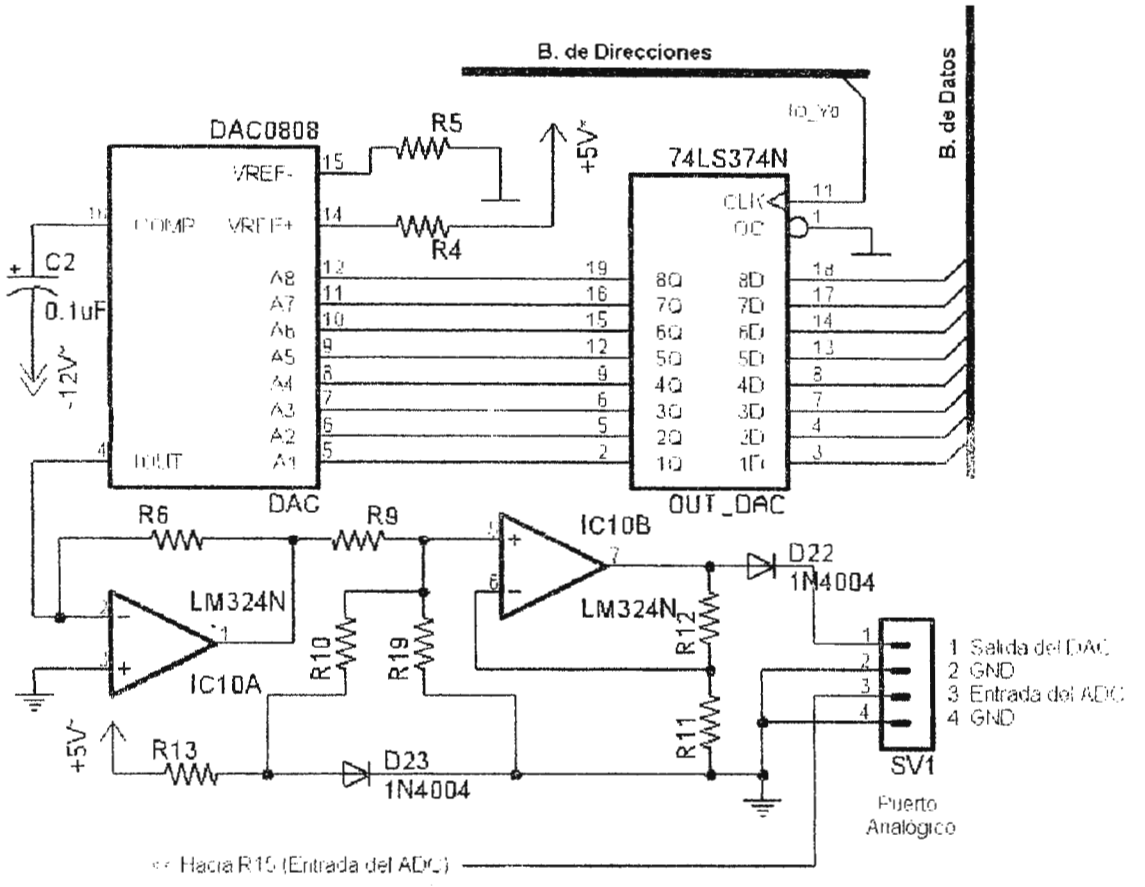


Fig. 5 Puerto Analógico de Salida

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- Este convertidor D/A es ampliamente utilizado para fines didácticos.
- Es un convertidor fácil de encontrar en el mercado y es de costo muy accesible.
- Es un convertidor de alta velocidad, además de tener un tiempo de conversión independiente del dato binario a su entrada, logrando así que éste sea constante (150nS típico).

- Al ser un convertidor de 8 bits de resolución, resulta fácil de adaptar a sistemas con microprocesadores.
- Ocho bits de resolución, es suficiente para la mayoría de aplicaciones básicas de moderada exactitud (por ejemplo la simulación del llenado de un tanque de agua, la cual es una de las aplicaciones de muestra del funcionamiento del presente módulo).
- Los datos provenientes del microprocesador Z80 al convertidor D/A son retenidos por un latch formado por flip-flops tipo D (74LS374), ya que dicho convertidor no cuenta con sistema de retención interno, y los datos desaparecerían (y su voltaje de salida también se desvanecería) tan pronto el microprocesador dejara de tener acceso al convertidor.
- Se configuró con una corriente de referencia de 2.0 mA (por medio de las resistencias R4 y R5 ambas de 2.5 KΩ), que será también la máxima corriente de salida a escala completa, es decir cuando todas sus entradas binarias estén en estado lógico alto.
- Es necesario convertir la corriente de salida del DAC0808 en voltaje por medio de un amplificador operacional en una configuración para tal efecto (IC10A y R6 = 2.5 KΩ para un voltaje máximo de salida = 5V).
- Para proteger el puerto analógico de salida (D/A), es necesario colocar un diodo a su salida (D22), para evitar posibles corrientes en inversa por algún problema ó defecto en algún circuito externo (por ejemplo un cortocircuito).
- Para compensar la caída de voltaje (de 0.7V) a la salida del puerto analógico producida por el diodo mencionado en el inciso anterior, es necesario sumar previamente 0.7V a la salida del convertidor de corriente a voltaje (IC10A), antes de conectarse al diodo de protección. Esto se logra por medio del amplificador operacional IC10B en configuración de sumador no inversor. Donde:

$$R = R_9 = R_{10} = R_{11} = R_{13} = R_{19} = 62K\Omega \quad \text{y} \quad R_{12} = 2R = 124K\Omega$$

Con lo cual se logra que por medio de la suma de la caída de potencial del diodo D23 (de 0.7V) a la salida del convertidor de corriente a voltaje (IC10A), se compense la caída de potencial producida por el

diodo de protección D22.

Los cálculos son los siguientes:

$$V^{+}_{IC10B} = \frac{Vo_{IC10B} \cdot R_{11}}{R_{11} + R_{12}} = \frac{Vo_{IC10B} \cdot R}{R + 2R} = \frac{Vo_{IC10B}}{3}$$

$$\frac{V^{+}_{IC10B} - Vo_{IC10A}}{R} + \frac{V^{+}_{IC10B} - 0.7V}{R} + \frac{V^{+}_{IC10B}}{R} = 0$$

$$\frac{3V^{+}_{IC10B}}{R} = \frac{Vo_{IC10A}}{R} + \frac{0.7V}{R}$$

Como: $V^{+}_{IC10B} = V^{-}_{IC10B}$ entonces:

$$\frac{3Vo_{IC10B}}{3} = Vo_{IC10A} + 0.7V \Rightarrow \boxed{Vo_{IC10B} = Vo_{IC10A} + 0.7V}$$

Con lo cual se demuestra como se compensa la caída de potencial producida por el diodo de protección de D22.

5.4. Puerto Paralelo.

Para el diseño del puerto paralelo se empleó una PPI 8255 (Programmable Peripheral Interface), la cual desempeña junto a un conector tipo macho FC – 40P de 40 pines la función de comunicar de forma paralela otros periféricos con el microprocesador Z80. Este puerto además de contar con los 3 puertos de la PPI 8255, cuenta con líneas de dirección, de control y de datos para su uso en general.

La distribución de pines se muestra en la figura siguiente:

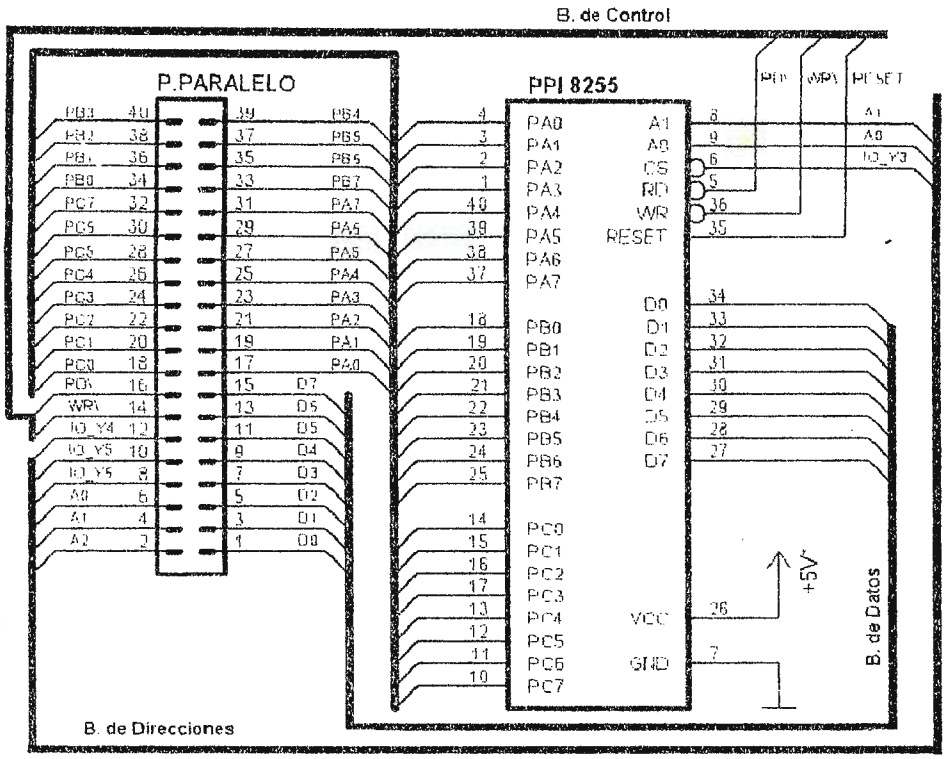


Fig. 7 Puerto Paralelo

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- La PPI es ampliamente utilizada para fines didácticos.
- Es un CI fácil de encontrar en el mercado y es de costo accesible.
- Su amplia versatilidad de configuración controlada por software (en sus tres diferentes modos), permite tener un puerto muy dinámico y muy eficiente ya que con sólo un integrado se pueden tener múltiples puertos I/O, la posibilidad de generar y recibir interrupciones desde y hacia dispositivos externos y hasta un puerto bidireccional con ó sin el uso de handshaking.
- El puerto paralelo cuenta con el bus de datos del sistema para su uso en general, así como también líneas de dirección y de control básicas para que el microprocesador pueda interactuar con dispositivos I/O directamente.
- Uno de los grandes requisitos que debe cumplir este puerto es de ser un medio eficiente para la interconexión con las diferentes plantas de simulación.

5.5. Indicador de Voltaje de Alimentación (Vcc).

Este circuito de diseñó para indicar que la fuente de alimentación (+5V) está funcionando adecuadamente.

La alimentación del módulo es proveída por una fuente que cuente con la capacidad de suministrar: +5V, +12V y -12V, y para ello se recomienda la utilización de las fuentes con las que son construidas las computadoras, ya que éstas, poseen la capacidad de suministrar dichos voltajes, y cuentan con protecciones internas de sobrevoltajes (en la red eléctrica), corte de suministro de alimentación en caso de corto circuito, etc..

Fig. 8 Indicador de Voltaje de Alimentación

5.6. Reset y Leds Indicadores.

Este es un circuito impreso "móvil" (y bajo ciertas condiciones desmontable), que se adapta al circuito principal del módulo por medio de un conector en línea de 13 pines: SV4 (ó por medio de cables directamente soldados a la misma), con el fin de tener un cierto grado de flexibilidad en cuanto a su ubicación dentro del módulo (independiente del circuito impreso principal), para una mejor visualización de los leds ó para la ubicación específica del reset lejos de otros pulsadores que formen parte del funcionamiento del módulo.

El circuito se muestra en la figura siguiente:

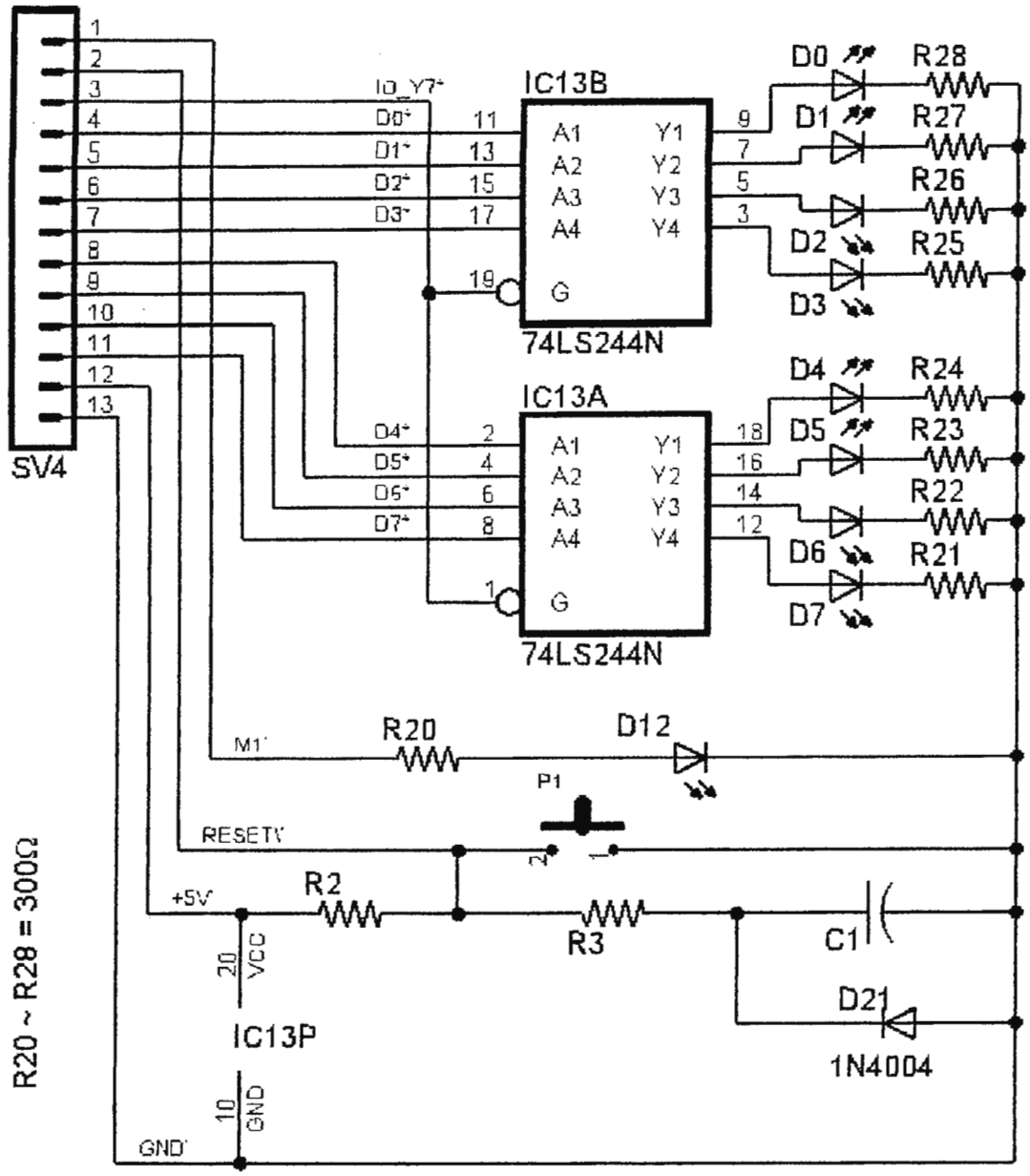


Fig. 9 Circuito de Reset y Leds Indicadores

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- Se diseñó un puerto de salida con un buffer 74LS244 (IC13), con el fin de controlar por medio de él, la activación de los leds indicadores de uso general (que son 8 en total).
- Led indicador de alimentación del microprocesador cuando se encuentra encendido ó indicador de un posible mal funcionamiento de éste cuando

se encuentra apagado (R20 y D12), el cual se conecta con el pin M1 (indicador de ciclo de máquina uno) del microprocesador Z80.

- Para un reset efectivo del sistema, es necesario que el reset se mantenga activo un mínimo de 3 ciclos completos de reloj.
- Se diseñó un sistema de reset, teniendo el cuidado de controlar los efectos de los rebotes en el pulsador (como los picos de corriente por ejemplo), por medio del control de la carga y descarga de un condensador que controla los efectos negativos del reset (D2, R2, R3 y C1). Donde: C1 = 0.1µF R2 = 300Ω R3 = 10Ω

  La constante de tiempo está definida por: $\tau = RC$

  Entonces el tiempo de retardo de la activación del reset está definida por:

$$t_{(activación)} = 3\tau = 3R_3C_1 = 3\times(10\Omega)(0.1\times10^{-6}F) = 6\mu S \text{ Aprox.}$$

  Y la corriente máxima producida tras la activación un reset es aproximadamente:

$$Ip_{(\max)} \approx \frac{Vc_{1(\max)}}{R_3} = \frac{5V}{10\Omega} \approx 0.5A$$

  Entonces el tiempo de retardo de la desactivación del reset está definida por:

$$t_{(desactivación)} = 3\tau = 3R_2C_1 = 3\times(310\Omega)(0.1\times10^{-6}F) = 93\mu S \text{ Aprox.}$$

  Con lo cual se asegura que el sistema tendrá un reset efectivo y libre de picos de corriente tras su activación y desactivación.

5.7. Jumper de Configuración de Interrupciones.

Esta etapa del circuito está destinada a la administración de las interrupciones. Es decir, con la ayuda de un par de jumpers se puede asignar las diferentes interrupciones (INT y NMI), ya sea a la UART, a dispositivos I/O ó a ninguno de ellos (deshabilitadas).

El circuito básico se muestra en la figura siguiente:

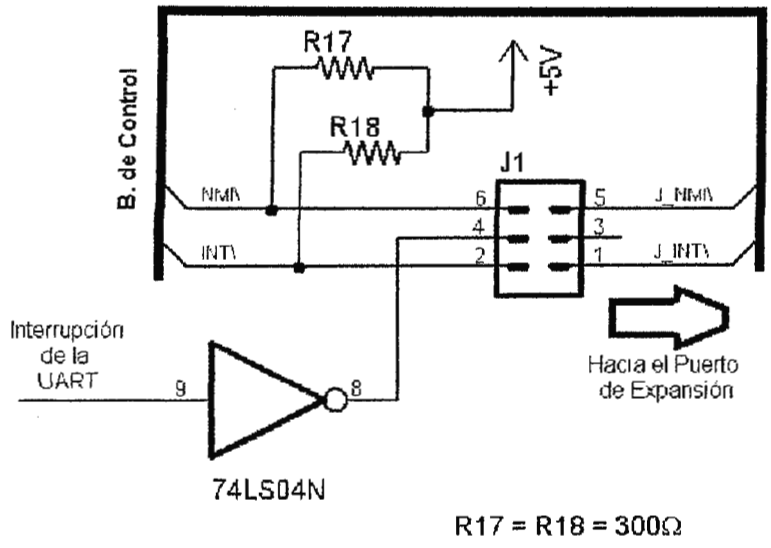


Fig. 10 Jumper de Configuración de Interrupciones

Para el diseño de este circuito habilitador/deshabilitador de interrupciones, se tomaron las siguientes consideraciones:

- Si no se colocan jumpers en J1, se deshabilitan las interrupciones (manteniendo un uno lógico en los pines NMI e INT del microprocesador por medio de las resistencias R17 y R18).
- Si se coloca un jumper entre el pin 6 y el pin 5 de J1, se estará asignando la interrupción no enmascarable (NMI) a los dispositivos periféricos conectados al puerto de expansión.
- Si se coloca un jumper entre el pin 2 y el pin 1 de J1, se estará asignando la interrupción enmascarable (INT) a los dispositivos periféricos conectados al puerto de expansión.
- Si se coloca un jumper entre el pin 6 y el pin 4 de J1, se estará asignando la interrupción no enmascarable (NMI) a la UART.
- Si se coloca un jumper entre el pin 4 y el pin 2 de J1, se estará asignando la interrupción enmascarable (INT) a la UART.
- Se debe asegurar la exclusividad de cada interrupción ya sea ó sólo para la UART ó sólo para los dispositivos externos del bus de expansión (una interrupción no puede responder ó conectarse a ambos sistemas directamente).

5.8. Decodificador de Direcciones.

Como su nombre lo indica, esta etapa del circuito está destinada a decodificar las direcciones básicas para las memorias y los dispositivos I/O del sistema, con lo cual se genera el mapa de memoria.

El circuito se muestra en la figura siguiente:

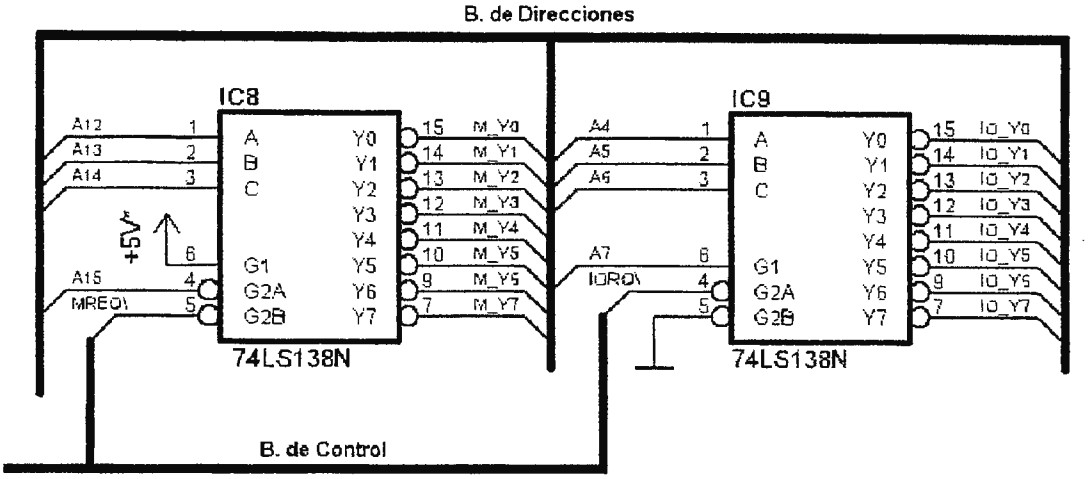


Fig. 11 Decodificador de Direcciones

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- Se empleó el CI decodificador/demultiplexor 74LS138 (3 a 8 líneas).
- Este CI es ampliamente utilizado para fines didácticos.
- Es un CI fácil de encontrar en el mercado y es de costo muy accesible.
- Se emplearon dos CI 74LS138, uno para direccionar memorias (IC8) y otro para direccionar dispositivos I/O (IC9). Esto se debe a que el número de líneas de dirección necesarias para direccionar dispositivos periféricos I/O es menor que las necesarias para direccionar memorias (el microprocesador Z80 sólo puede direccionar hasta 256 dispositivos I/O como máximo, es decir, que sólo requiere para ello 8 líneas de dirección).
- Las líneas decodificadas que no se utilizan para el direccionamiento del sistema (es decir, que no direccionar dispositivos internos para el

funcionamiento del módulo), se encuentran disponibles para su uso general en el puerto paralelo y en el puerto de expansión.

El mapa de memoria, se resume en los dos cuadros siguientes:

74LS138 (IC8) Decodificador de Memorias	
DIRECCIÓN	MEMORIA
0000H – 0FFFH M_Y0	EPROM1
1000H – 1FFFH M_Y1	EPROM2
2000H – 27FFH M_Y2	RAM1
3000H – 37FFH M_Y3	RAM2

74LS138 (IC9) Decodificador de Dispositivos I/O	
DIRECCIÓN	DISPOSITIVO I/O
80H IO_Y0	DAC
90H IO_Y1	ADC
A0H – A7H IO_Y2	UART
B0H – B3H IO_Y3	PPI
FFH IO_Y7	Leds indicadores de uso general

5.9. Generador de Reloj.

Su función específica es brindar el reloj de sincronía del sistema. Este reloj se logra con una configuración de circuito retroalimentado a base de un cristal de 4 MHz.

El circuito se muestra en la figura siguiente:

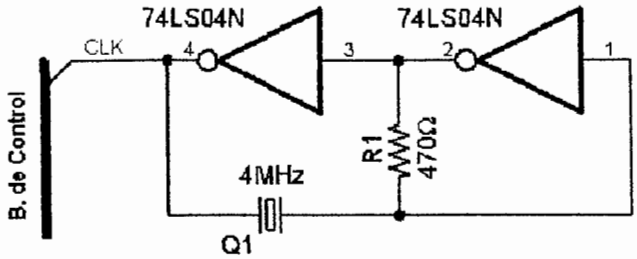


Fig. 12 Circuito generador de reloj

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- Es un circuito sencillo de implementar.
- El costo de su implementación es muy accesible.
- Como es un circuito oscilador basado en cristal, la frecuencia de trabajo es muy estable.
- Requiere solamente de una alimentación de +5V para operar.
- Su salida es TTL.

5.10. Puerto de Expansión.

La función específica de este puerto es brindar una herramienta para la expansión del módulo (como su nombre lo indica), al dotar al usuario de las líneas más importantes del sistema, como lo son: el bus de datos, el bus de direcciones, el bus de control y líneas de dirección ya decodificadas, todas presentes en un conector tipo macho FC – 40P de cuarenta pines.

Su circuito se presenta en la figura siguiente:

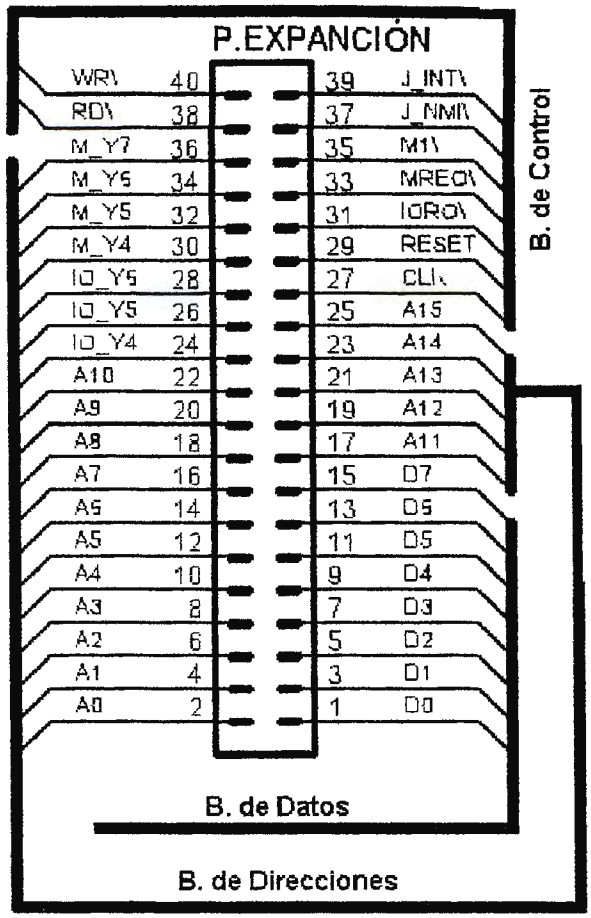


Fig. 13 Puerto de Expansión

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- Es necesario tener un medio por el cual se pueda fácilmente expandir la capacidad de memoria del sistema.
- Es necesario que el módulo posea un medio por el cual se pueda tener acceso a todas las líneas físicas del microprocesador para fines didácticos.
- Es necesario tener un medio por el cual se pueda fácilmente expandir las capacidades de operación del sistema en cuanto a hardware (por ejemplo para brindar al módulo la capacidad de administrar interrupciones por medio de hardware es necesario adaptarle el Controlador de Interrupciones 8259, el cual requiere para su interconexión con el microprocesador dos líneas especialmente diseñadas para su intervención en la respuesta ante interrupciones: M1

y IORQ, las cuales se activan simultáneamente para indicar al dispositivo que solicita servicio que la interrupción ha sido aceptada).

NOTA: es importante tener en cuenta el FAN – OUT, del microprocesador a la hora de crear circuitos de expansión del módulo, debido a que no se han colocado buffers a la salida de éste.

5.11. Puerto de Serie.

La función principal de este puerto, es la de comunicar de forma serie y asíncrona el módulo ya sea con una computadora ó con otros sistemas similares (que soporten comunicación serial asíncrona). Dicha comunicación, puede ser para diversos motivos como lo es por ejemplo el monitoreo y control de las operaciones del módulo.

Para una mayor compatibilidad con otros sistemas que soportan comunicación serial, se adaptó la salida (del puerto serie) al estándar RS232, el cual es extensamente utilizado en sistemas de comunicación serie para asegurar la compatibilidad en cuanto a hardware, entre diversos sistemas que necesiten de comunicación serial.

El circuito se muestra en la figura siguiente:

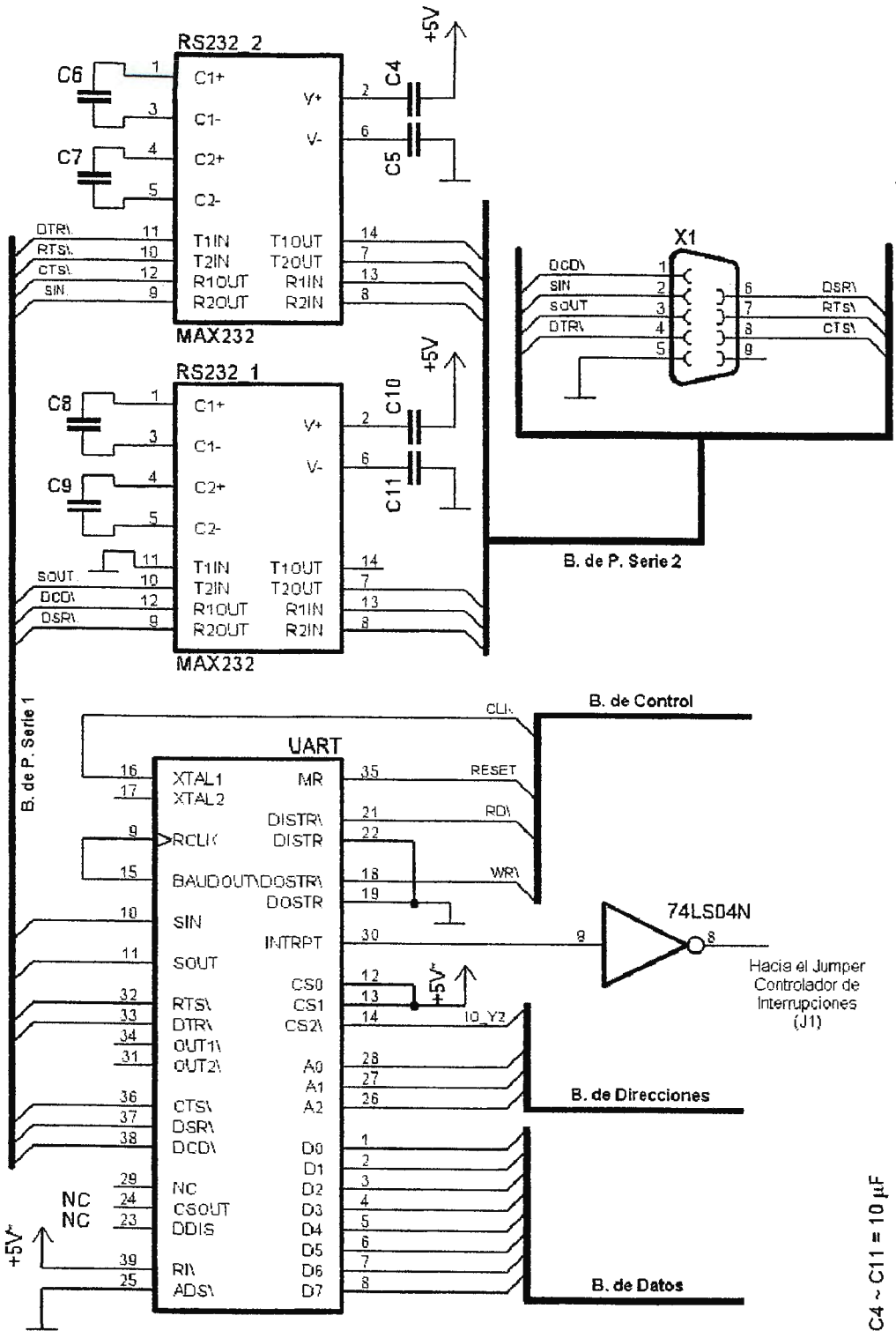


Fig. 14 Puerto de Serie

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- El sistema se basó en un CI de comunicación serie de tipo UART (Universal Asynchronous Receiver/Transmitter, Receptor/Transmisor Universal Asíncrono), el cual es el encargado de convertir los datos paralelos provenientes del microprocesador en datos seriales asíncronos a sus salidas.
- El CI UART puede ser el NS16450N ó su versión mejorada (por ejemplo el PC16550)
- Es un CI fácil de encontrar en el mercado, de costo accesible y muy utilizado actualmente en los sistemas de computadoras que cuentan con comunicación serial (la cual es por defecto asíncrona).
- Este CI está diseñado para trabajar en sistemas basados en microprocesadores (ya que cuenta con pines que pueden conectarse directamente a dichos sistemas, como lo son: líneas de datos, de direcciones y de control).
- Este CI es capaz de agregar (al transmitir) y suprimir (al recibir) automáticamente, bits utilizados como estándares de la comunicación serial asíncrona (como lo son el bit de start, stop y paridad).
- El CI dispone de líneas de control de módem (CTS, RTS, DSR, DTR, RI y DCD), con las cuales se puede establecer un control de flujo por medio de hardware.
- El CI tiene la capacidad de producir interrupciones ante diferentes eventos producidos en la comunicación.
- El CI cuenta con una velocidad de transmisión/recepción programable.
- Para una mayor compatibilidad con otros sistemas de comunicación serie, se adaptó la salida de la UART al estándar RS232, el cual asegura la compatibilidad en cuanto a hardware con dichos sistemas.
- La salida del puerto serie es conectada a un conector tipo hembra DB-9, para su interconexión con otros dispositivos. La distribución es pin por pin compatible con los sistemas seriales de las computadoras para una mayor compatibilidad en la comunicación con éstas.

6. FUNCIONAMIENTO DE LAS PLANTAS A EMULAR.

Como ya se había mencionado el módulo es capaz de emular diferentes plantas para la prueba de sistemas de control. Para su demostración, se diseñaron dos emulaciones de diferentes tipos: una digital (es decir, una planta que requiere señales de control digital) y otra analógica (ó sea, una planta que requiere señales de control analógicas).

Como planta digital, se emulará el ascensor de un edificio, donde el controlador deberá manejar la operación del motor de éste y en cuanto a la planta analógica, el módulo emulará un tanque para almacenar agua, y en este caso, el controlador deberá regular (por medio de la variación del caudal de entrada Qin) el nivel ó altura del líquido contenido por éste, a pesar de las pérdidas causadas por un caudal de salida Qo.

Para el desarrollo de dichas emulaciones, fue necesario diseñar un circuito con una matriz de leds, para la visualización de la secuencia de los procesos llevados a cabo por cada planta, para una mejor comprensión de los efectos de la acción de control.

6.1. Circuito para la Visualización del Proceso.

Para su implementación se diseñó un circuito que se utilizará como tarjeta adaptable al módulo, la cual, no es más que un circuito de expansión del módulo para realizar alguna tarea específica. Su finalidad en este caso, es la de brindar una visualización del proceso llevado a cabo por la planta, al contar con una matriz de leds que refleje el comportamiento de ésta.

El circuito esquemático se muestra en la figura siguiente:

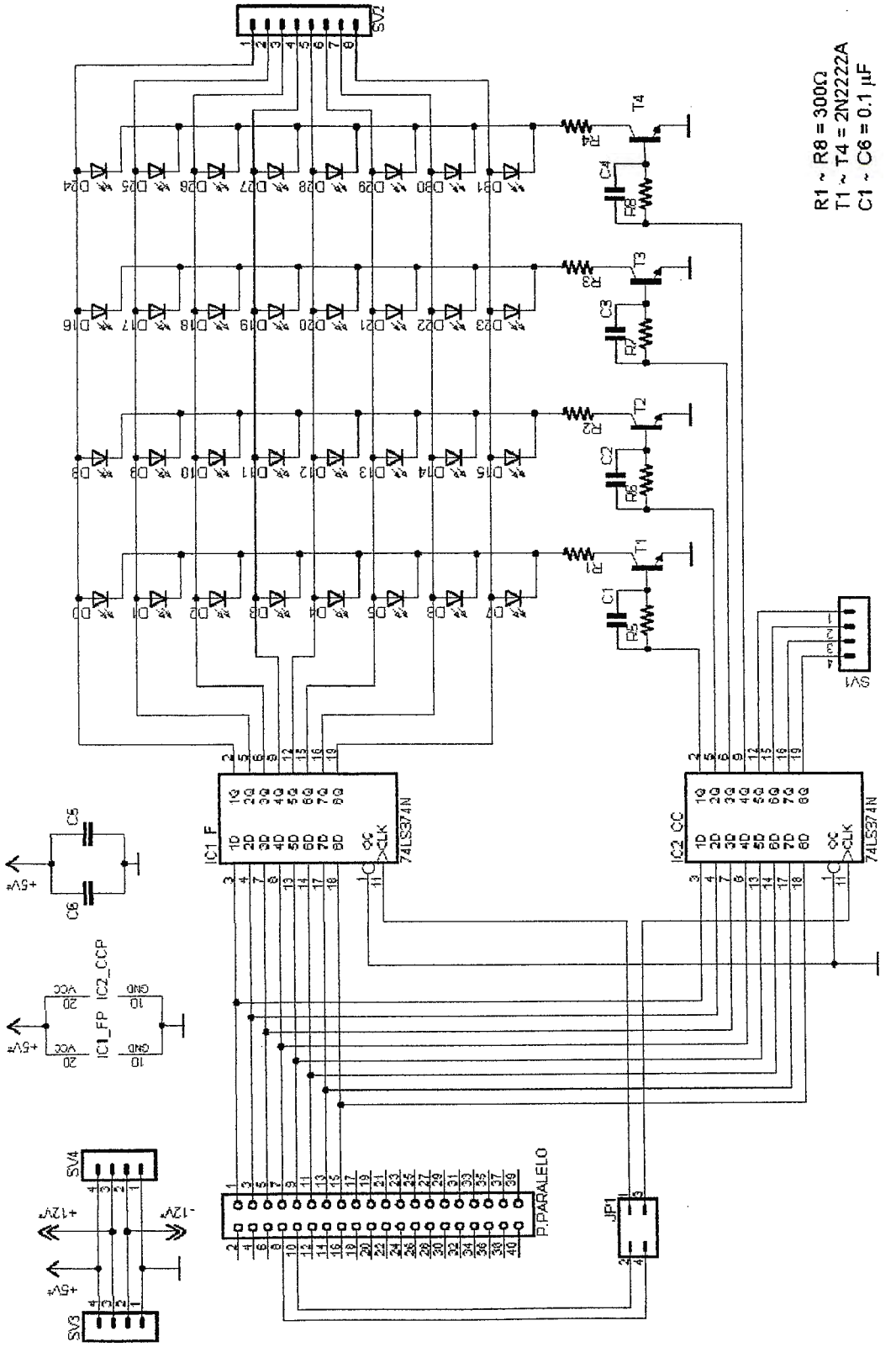


Fig. 15 Matriz de Leds

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- El circuito debe ser capaz de simular los procesos llevados a cabo en ambas plantas (digital y analógica).
- Ésta tarjeta ó circuito adaptable, debe conectarse al módulo por medio del puerto paralelo.
- Los leds estarán colocados en forma matricial, con un total de 32 leds divididos en 4 columnas y 8 filas.
- La matriz se basará en los CI 74LS374, por tanto, los puertos que controlan las filas y las columnas estarán latchadas, para evitar el refrescamiento constante por parte del microprocesador.
- Los puertos de filas y columnas utilizarán direcciones ya decodificadas en el módulo, las cuales, se encuentran disponibles en el puerto paralelo (0DDH y 0EEH respectivamente) con sólo interconectar los pines 1 – 2 y 3 – 4 del JP1.
- Si se deseara utilizar ésta matriz en alguna otra emulación, y las direcciones mencionadas en el inciso anterior se encuentran asignadas a otro circuito, se puede recurrir a las líneas de habilitación disponibles en los pines 1 y 3 de JP1.
- Se encuentran disponibles las líneas de control de las filas y las columnas por medio de los conectores SV1 y SV2, por si se deseara utilizar ésta matriz en alguna otra emulación y se necesitaran agregar más leds a la matriz ó para algún otro propósito general.
- Para la alimentación del circuito se cuenta con dos conectores tipo macho de cuatro pines SV3 y SV4. Donde uno (cualquiera de los dos), se utiliza para la alimentación del circuito (la cual proveniente de las líneas de alimentación disponibles en el módulo en SV5), y el otro para distribuir la alimentación hacia otros circuitos adaptables al módulo.

La distribución y posición de los CI 74LS374, los conectores, el jumper JP1, y los leds de la matriz en el circuito impreso se muestran en la figura siguiente:

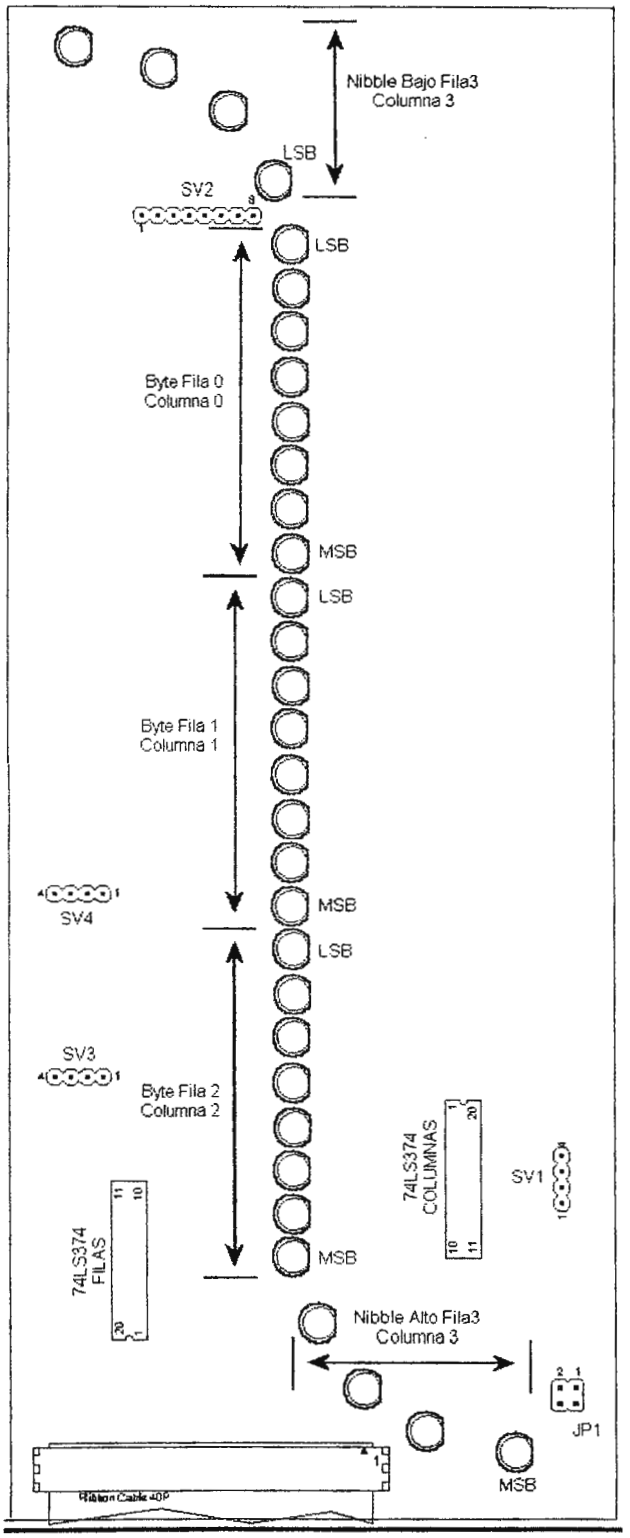


Fig. 16 Distribución de elementos de la Matriz de Leds

6.2. Circuito para la Temporización de las Emulaciones.

Debido a que el circuito de visualización (del inciso anterior) es matricial, es necesario y útil contar con un circuito que temporice el refrescamiento de la pantalla de leds (ó matriz de leds). Y para ello, se diseñó un circuito oscilador basado en un CI LM555 en configuración astable.

El circuito esquemático se muestra en la siguiente figura:

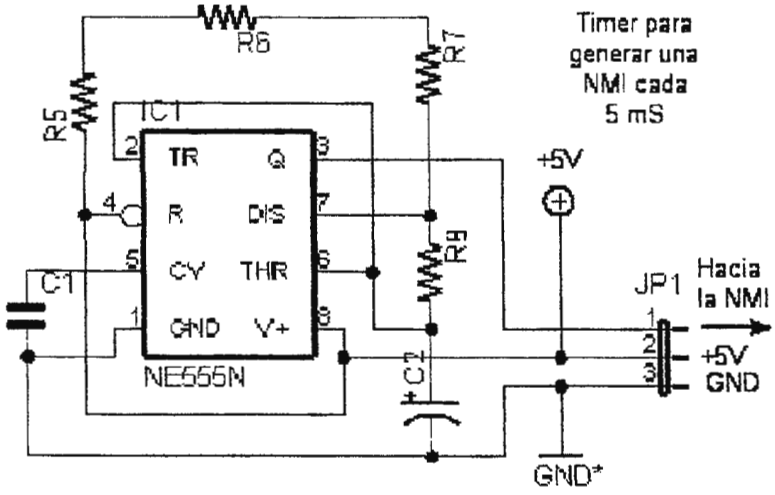


Fig. 17 Temporizador Generador de Interrupción NMI

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- El CI LM555, es ampliamente utilizado para fines didácticos.
- Es un CI muy fácil de encontrar en el mercado, además de tener un costo muy accesible.
- Como puede observarse en el diagrama anterior R5, R6 y R7 se encuentran en serie, y esto es, para conseguir una mayor precisión o para obtener de valores no comunes de resistencias para cuando la aplicación lo requiera.
- El temporizador, está en configuración astable para la generación de un tren de pulsos (los cuales se utilizan para generar una interrupción periódica al microprocesador).
- El temporizador debe tener un tiempo en alto de 5 mS, es decir, entre cada solicitud de la NMI (tiempo necesario para mantener una

visualización aceptable del proceso en la matriz de leds).

- El tiempo en bajo (solicitud de NMI) debe ser menor que el de un ciclo de reloj del sistema (0.25μS), para evitar la anidación de servicios de interrupción y que la ejecución del programa se vuelva inestable.

Los cálculos correspondientes son los siguientes:

$$t_{alto} = 0.7 \times (R_5 + R_6 + R_7 + R_9) \times C_2 \cong 0.7 \times (R_5 + R_6 + R_7) \times C_2$$

$$t_{alto} = 0.7 \times (1.7 \times 10^{6}) \times (4.2 \times 10^{-9}) = 4.99\ mS$$

$$t_{bajo} = 0.7 \times R_9 \times C_2 = 0.7 \times 12 \times (4.2 \times 10^{-9}) = 35nS$$

6.3. Planta Digital: Motor de Ascensor de un Edificio.

La idea central de ésta planta, es la de probar sistemas de control digitales. Para ello, se emula un edificio que cuenta con 8 pisos con habitaciones para huéspedes además de un piso de recepción (piso 0), y que para desplazarse entre piso y piso el edificio cuenta con un ascensor que es gobernado por un motor que sólo necesita una de dos posibles órdenes para operar: subir ó bajar.

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- La planta debe servir para la prueba de sistemas de control digital.
- Se requiere de un acrílico, el cual contenga un diagrama alusivo al proceso de la planta.
- Se requiere un circuito que funcione como tarjeta adaptable al módulo y que se conecte a través del puerto paralelo para la visualización del proceso por medio del leds (que estarán colocados de forma matricial).
- Se requiere un circuito oscilador el cual provea un temporizado de la interrupción NMI (Interrupción No Enmascarable) para el refrescamiento del circuito de visualización (matriz de leds).
- Se requieren 24 leds colocados en línea recta para la visualización del proceso (9 para los pisos y los demás para la simulación del

desplazamiento del ascensor entre cada piso).

- El módulo recibirá las señales u órdenes de control por medio del puerto serie en forma de caracteres ASCII ('s' para subir y 'a' para que el ascensor se desplace hacia abajo).
- El módulo debe proveer al controlador la siguiente información en formato ASCII inmediatamente después de haber recibido una orden ó señal de control 's' ó 'a': 1° el número de led que se encuentra encendido en la matriz (0 ~ 23) y 2° el piso en que se encuentra el ascensor (0 ~ 8).

El diagrama auxiliar a colocarse en el acrílico, para la visualización general de los procesos de la planta, se presenta en la siguiente figura:

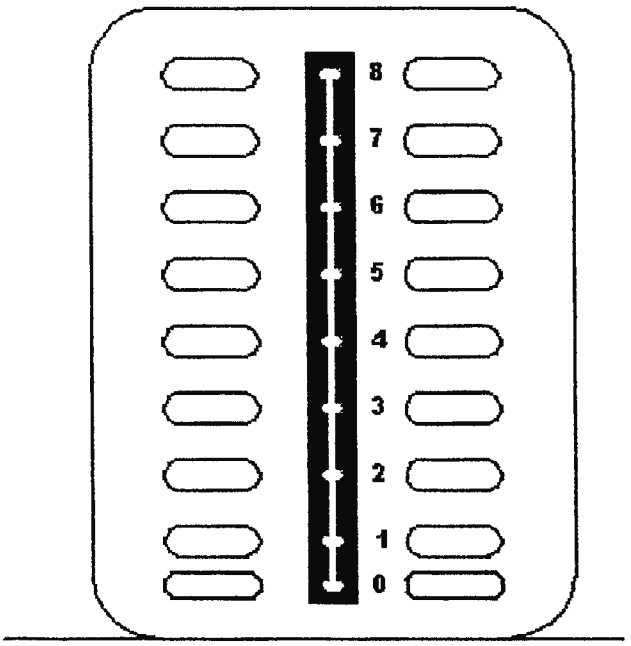
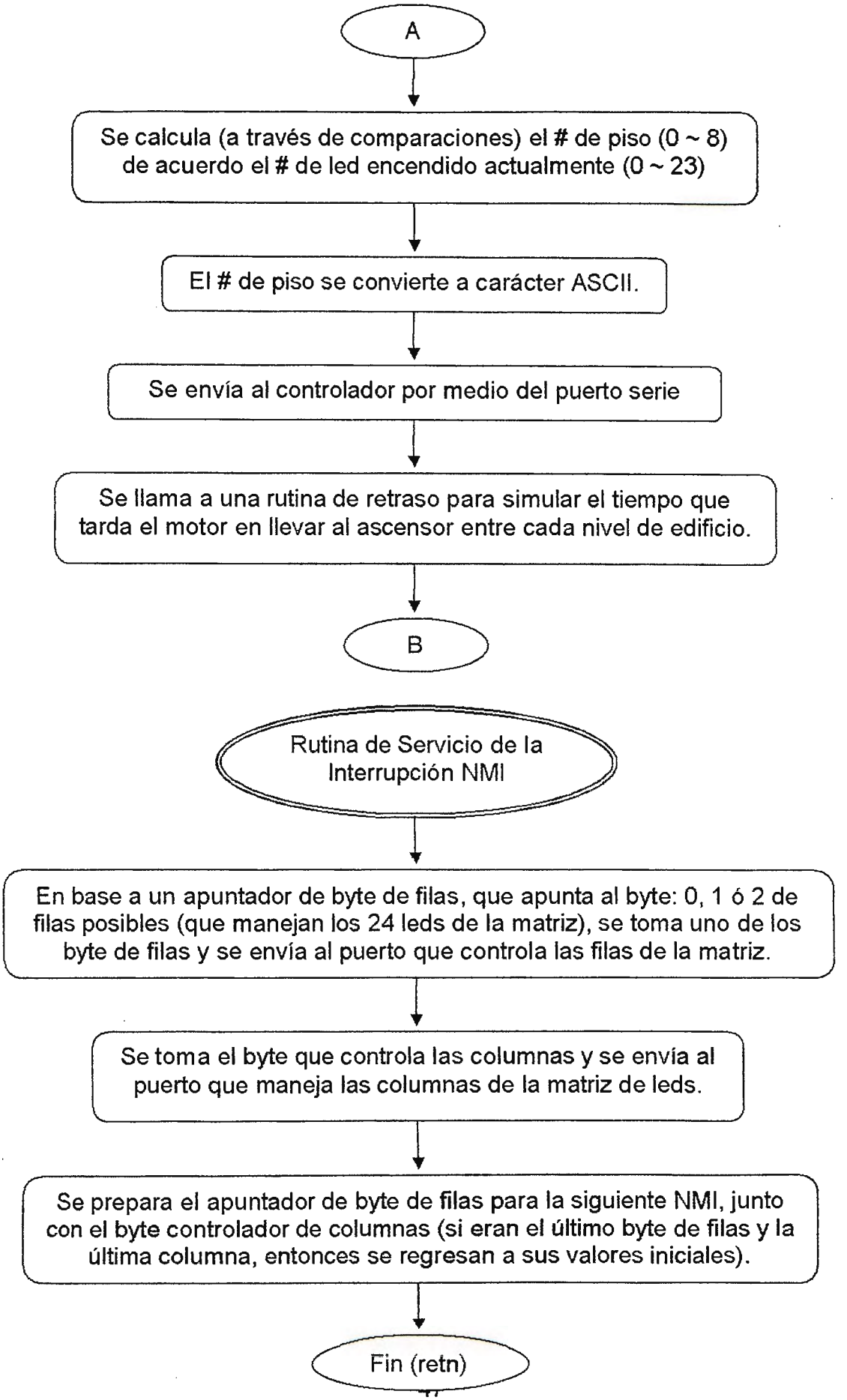
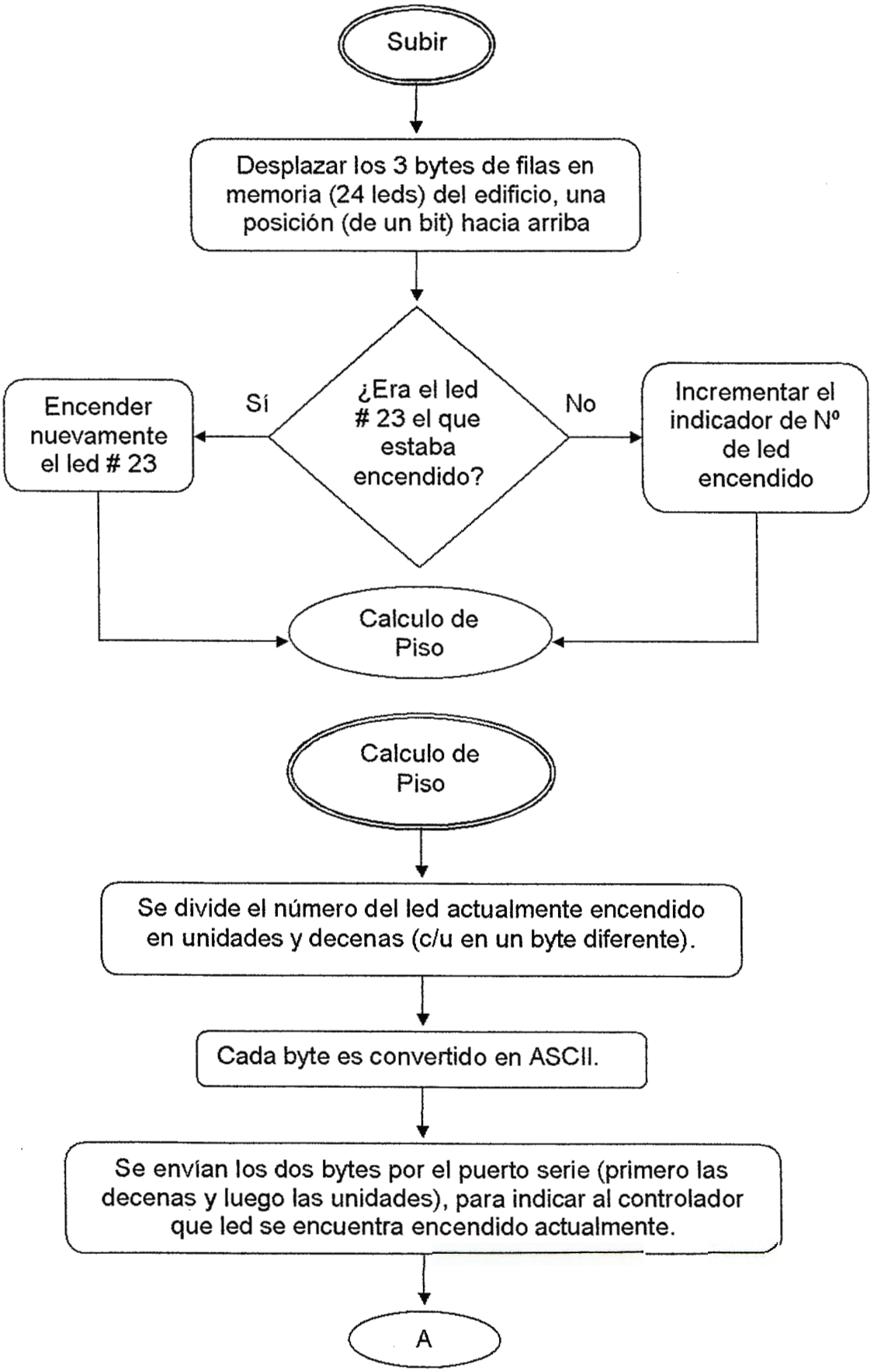


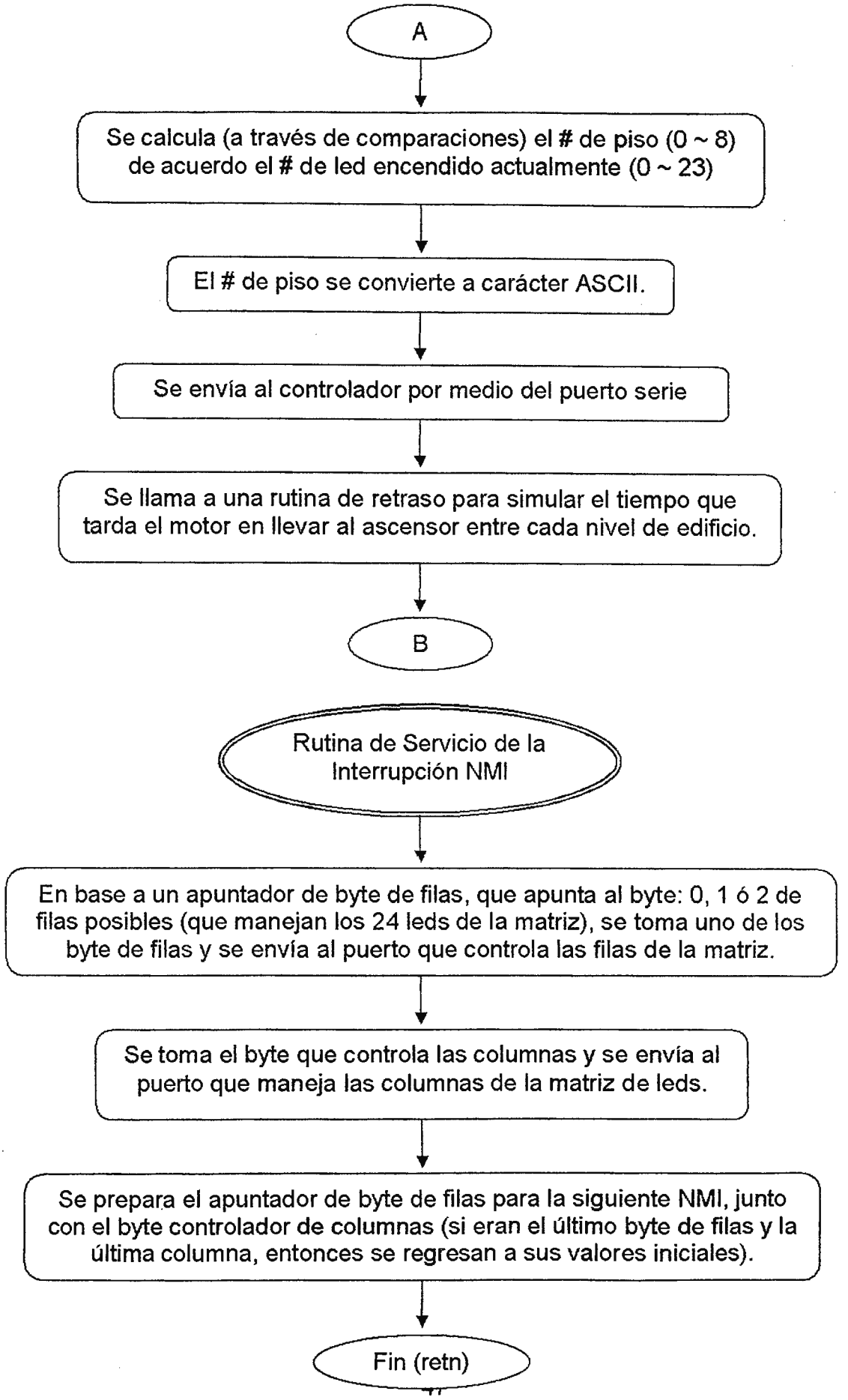
Fig. 18 Visualización de la Planta Digital: Ascensor de Edificio.

Los siguientes flujogramas describen los procesos llevados a cabo por el programa del módulo para la emulación de la planta digital del edificio.

NOTA: todas las variables utilizadas se encuentran descritas en el programa en lenguaje nemotécnico.


A
Se calcula (a través de comparaciones) el # de piso (0 ~ 8) de acuerdo el # de led encendido actualmente (0 ~ 23)
El # de piso se convierte a carácter ASCII.
Se envía al controlador por medio del puerto serie
Se llama a una rutina de retraso para simular el tiempo que tarda el motor en llevar al ascensor entre cada nivel de edificio.
B
Rutina de Servicio de la Interrupción NMI
En base a un apuntador de byte de filas, que apunta al byte: 0, 1 ó 2 de filas posibles (que manejan los 24 leds de la matriz), se toma uno de los byte de filas y se envía al puerto que controla las filas de la matriz.
Se toma el byte que controla las columnas y se envía al puerto que maneja las columnas de la matriz de leds.
Se prepara el apuntador de byte de filas para la siguiente NMI, junto con el byte controlador de columnas (si eran el último byte de filas y la última columna, entonces se regresan a sus valores iniciales).
Fin (retn)


Subir
Desplazar los 3 bytes de filas en memoria (24 leds) del edificio, una posición (de un bit) hacia arriba
¿Era el led # 23 el que estaba encendido?
Sí
No
Encender nuevamente el led # 23
Incrementar el indicador de Nº de led encendido
Calculo de Piso
Calculo de Piso
Se divide el número del led actualmente encendido en unidades y decenas (c/u en un byte diferente).
Cada byte es convertido en ASCII.
Se envían los dos bytes por el puerto serie (primero las decenas y luego las unidades), para indicar al controlador que led se encuentra encendido actualmente.
A


A
Se calcula (a través de comparaciones) el # de piso (0 ~ 8) de acuerdo el # de led encendido actualmente (0 ~ 23)
El # de piso se convierte a carácter ASCII.
Se envía al controlador por medio del puerto serie
Se llama a una rutina de retraso para simular el tiempo que tarda el motor en llevar al ascensor entre cada nivel de edificio.
B
Rutina de Servicio de la Interrupción NMI
En base a un apuntador de byte de filas, que apunta al byte: 0, 1 ó 2 de filas posibles (que manejan los 24 leds de la matriz), se toma uno de los byte de filas y se envía al puerto que controla las filas de la matriz.
Se toma el byte que controla las columnas y se envía al puerto que maneja las columnas de la matriz de leds.
Se prepara el apuntador de byte de filas para la siguiente NMI, junto con el byte controlador de columnas (si eran el último byte de filas y la última columna, entonces se regresan a sus valores iniciales).
Fin (retn)

6.4. Planta Analógica: Tanque de Líquidos.

La idea central de ésta planta, es la de probar sistemas de control analógicos. Y para ello, se emula un tanque utilizado para el almacenamiento de líquidos, en donde, el controlador regulará el caudal de entrada Qin para mantener bajo control el nivel ó altura del líquido contenido por el tanque, el cual también cuenta con un caudal de salida Qo que actúa como las pérdidas del sistema.

La idea básica, consiste en emular un tanque con la siguiente descripción: una altura máxima del líquido contenido de 25.5 m, un área de la base del tanque de 0.02 m^2 (A_t) y una capacidad máxima de 0.511 m^3. Como ya se ha mencionado el controlador variará el caudal de entrada Qin (0 ~ 25.5 m^3/s) proporcionalmente a un voltaje analógico de entrada Vin (0 ~ 5V). Además, el tanque cuenta con un área en la tubería de entrada de 2 m^2 (básicamente sin restricciones de paso en la entrada) y un área en la tubería de salida de 0.0099 m^2 con lo cual se consigue, que el tiempo de llenado del tanque sea de aproximadamente 10 segundos.

Para su diseño se tomaron en cuenta las consideraciones siguientes:

- La planta debe servir para la prueba de sistemas de control analógicos.
- Se requiere de un acrílico, el cual contenga un diagrama alusivo al proceso de la planta.
- Se requiere un circuito que funcione como tarjeta adaptable al módulo y que se conecte a través del puerto paralelo para la visualización del proceso por medio del leds (que estarán colocados de forma matricial).
- Se requieren 32 leds colocados en línea para la visualización del proceso.
- Se requiere un circuito oscilador el cual provea un temporizado de la interrupción NMI (Interrupción No Enmascarable) para que controle el período de muestreo del sistema, así como el temporizado de los cálculos y el refrescamiento del circuito de visualización (matriz de leds).
- El módulo recibirá las señal de control Qin por medio del puerto analógico–digital (0 ~ 5V).

- El módulo proveerá al controlador, la información de la altura del líquido contenido por el tanque h_1, por medio del puerto digital–analógico (0 ~ 5V).

El diagrama auxiliar a colocarse en el acrílico, para la visualización general de los procesos de la planta, se presenta en la siguiente figura:

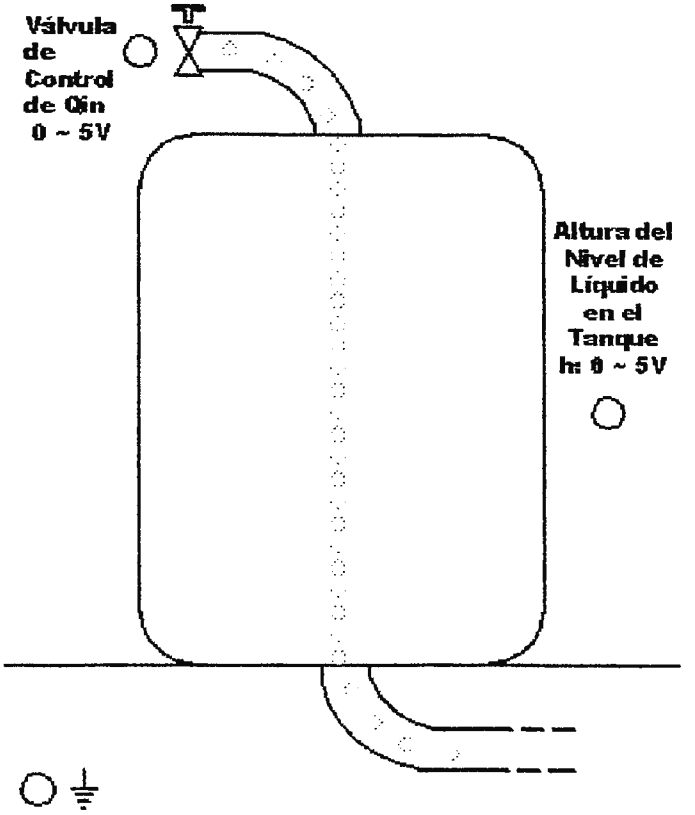


Fig. 19 Visualización de la Planta Analógica: Tanque de Líquidos

Para conseguir dicha emulación por medio del módulo, se desarrollo un programa que calcula periódicamente una ecuación, la cual, no es más que la ecuación de Bernulli, que se expresa de la siguiente forma:

$$\frac{V_1^2}{2g} + \frac{P_1}{\gamma} + h_1 = \frac{V_2^2}{2g} + \frac{P_2}{\gamma} + h_2$$

Donde: V_1 = velocidad del líquido de entrada

V_2 = velocidad del líquido de salida

P_1 = presión en la parte superior del líquido contenido

P_2 = presión del líquido a la salida del tanque

h_1 = altura ó nivel del líquido contenido

h_2 = nivel ó altura de la salida del tanque

γ = densidad del líquido

g = gravedad (9.8 m/s^2)

Como la presión en la superficie del líquido en el tanque, es igual a la presión de salida y la altura de la salida de caudal del tanque (h_2) es la referencia, es decir igual a cero. La ecuación se reduce a:

$$\frac{V_1^2}{2g} + h_1 = \frac{V_2^2}{2g}$$

En esta ecuación, la velocidad del líquido se relaciona con el caudal por medio de la siguiente fórmula:

$$V = \frac{Q}{A}$$

Donde: V = velocidad del líquido

Q = caudal del líquido

A = área de la tubería por la cual se desplaza el líquido

Finalmente para que el módulo pueda resolver con eficiencia el problema, se hace un cambio de unidades. Donde lo que se busca, es aumentar la precisión y mejorar la exactitud y el tiempo involucrado en los cálculos al evitar (lo más que se pueda) el uso de números con puntos decimales. Para ello, todo lo que se necesita es convertir los metros a decímetros en todos los datos.

El proceso en sí, que lleva a cabo el módulo al efectuar los cálculos, es el siguiente:

1° Se lee el valor presente en la entrada análoga del ADC, el cual representa, el valor del caudal de entrada Qin (0 ~ 25.5 m^3/s), pero en unidades de decímetros (es decir: 0 ~ 25,500 $dm^3/s \equiv Qin \times 100$, donde $0 \leq Qin \leq 255$).

2° Se divide el valor de Qin entre el área de la tubería de entrada ($A_1 = 2\ m^2 = 2 \times 100\ dm^2$), para obtener así, el valor de la velocidad del caudal de entrada del tanque (V_1).

3° De la ecuación de Bernulli (reducida), se despeja V_2 para obtener la velocidad del caudal de salida del tanque en función de V_1 y h_1. Obteniendo así:

$$V_2 = \sqrt{(V_1)^2 + (h_1 \times 2g)} = \sqrt{\left(\frac{Q_{in} \times 100}{2 \times 100}\right)^2 + (h_1 \times 2 \times 98)}$$

$$V_2 = \sqrt{\left(\frac{Q_{in}}{2}\right)^2 + (h_1 \times 196)} \left[\frac{dm}{s}\right]$$

Donde: $0 \leq Qin \leq 255$ [dm^3/s] y $0 \leq h_1 \leq 255$ [dm]

4° Se multiplica la velocidad del caudal de salida (V_2) por el área de la tubería de salida de caudal del tanque A_2 (en decímetros, que es de 0.99 dm), para obtener el caudal de salida (Qo).

5° Luego se procede a calcular la cantidad de volumen de líquido remanente en el tanque después de cada cálculo, los cuales se hacen periódicamente por sincronización de un reloj externo que se encarga de causar una NMI (interrupción no enmascarable) al microprocesador Z80. Para calcular el cambio de volumen se emplea el siguiente procedimiento:

i) Se calcula el caudal remanente: $\Delta Q = (Q_{in} - Q_o)$ Si el resultado es negativo, indica que el tanque se está vaciando. Por el contrario, si fue positivo, indica que el tanque se está llenando.

ii) Se obtiene el caudal total de la suma algebraica el caudal remanente con el caudal acumulado (donde éste último, no es más que el caudal "rezagado" de cálculos anteriores que no fue capaz de producir una unidad de volumen de líquido).

iii) De la división del caudal total entre 100, se obtiene el volumen acumulado en litros, cada T segundos. Donde 1/ T =100, y T es el que indica cada cuanto tiempo se efectuarán cálculos de la emulación del tanque (el cual, es igual a 10 mS, es decir, cada dos solicitudes de interrupción de la NMI)

$$\Delta\forall = Q_t \times T = Q_t \times \left(10 \times 10^{-3}\right) = \frac{Q_t}{100} \quad [litros] \quad \text{Donde: } T = 10mS$$

El residuo de la división es el caudal acumulado para la siguiente vez que se efectúen los cálculos. Y el residuo, es la cantidad de litros acumulados en esta rutina de servicio de la NMI.

6º Después, se procede a sumar algebraicamente el resultado del numeral anterior ($\Delta\forall$), al valor del volumen de líquido acumulado por el tanque actualmente ($\forall$), obteniendo un nuevo valor de volumen ($\forall$).

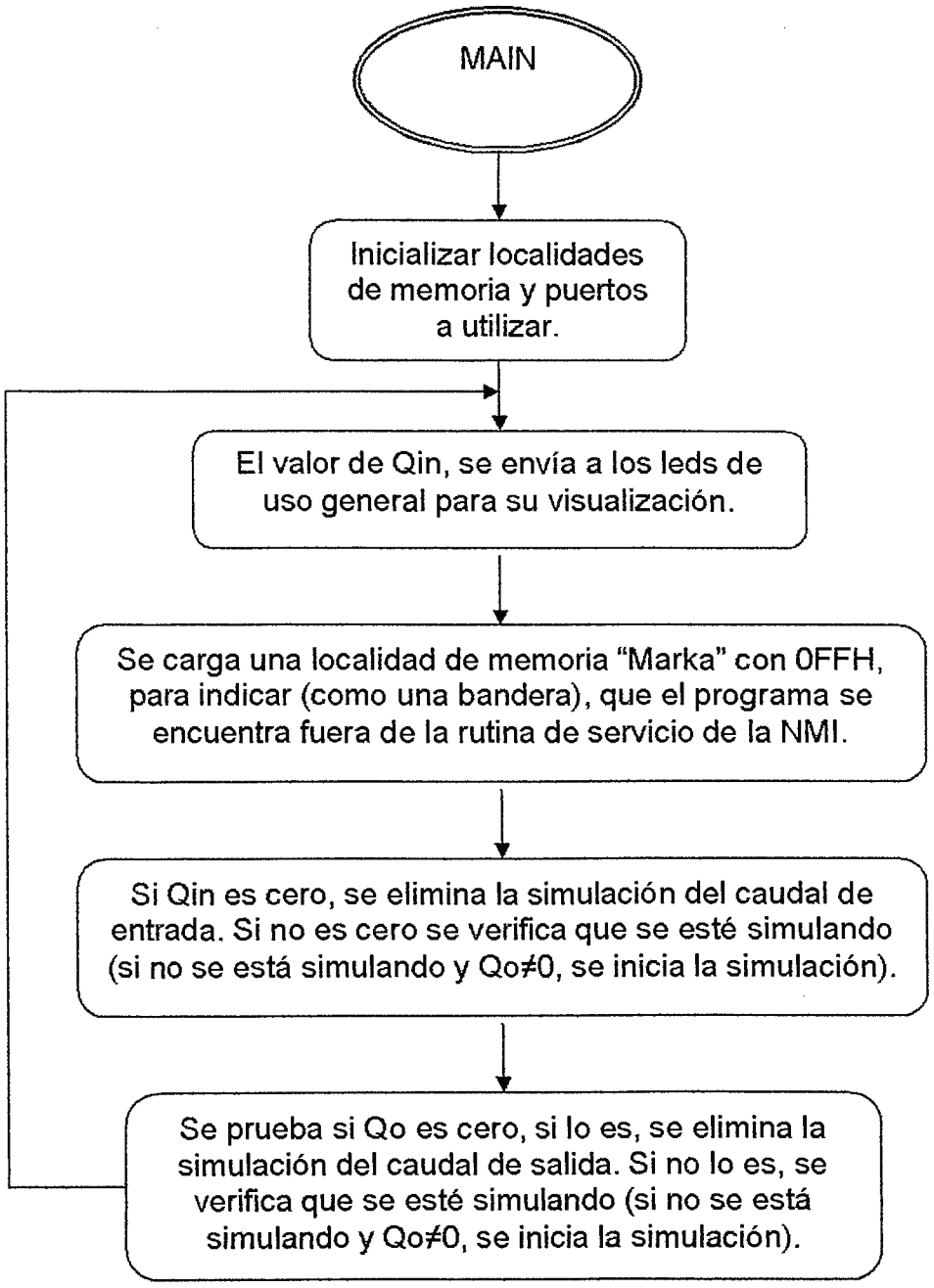
7º Se divide el volumen contenido por el tanque ($\forall$) entre el área de la base del tanque (A_t), para obtener así, la altura del líquido contenido en el tanque (h_1), la cual, es la respuesta que se envía al controlador por medio del puerto digital–analógico.

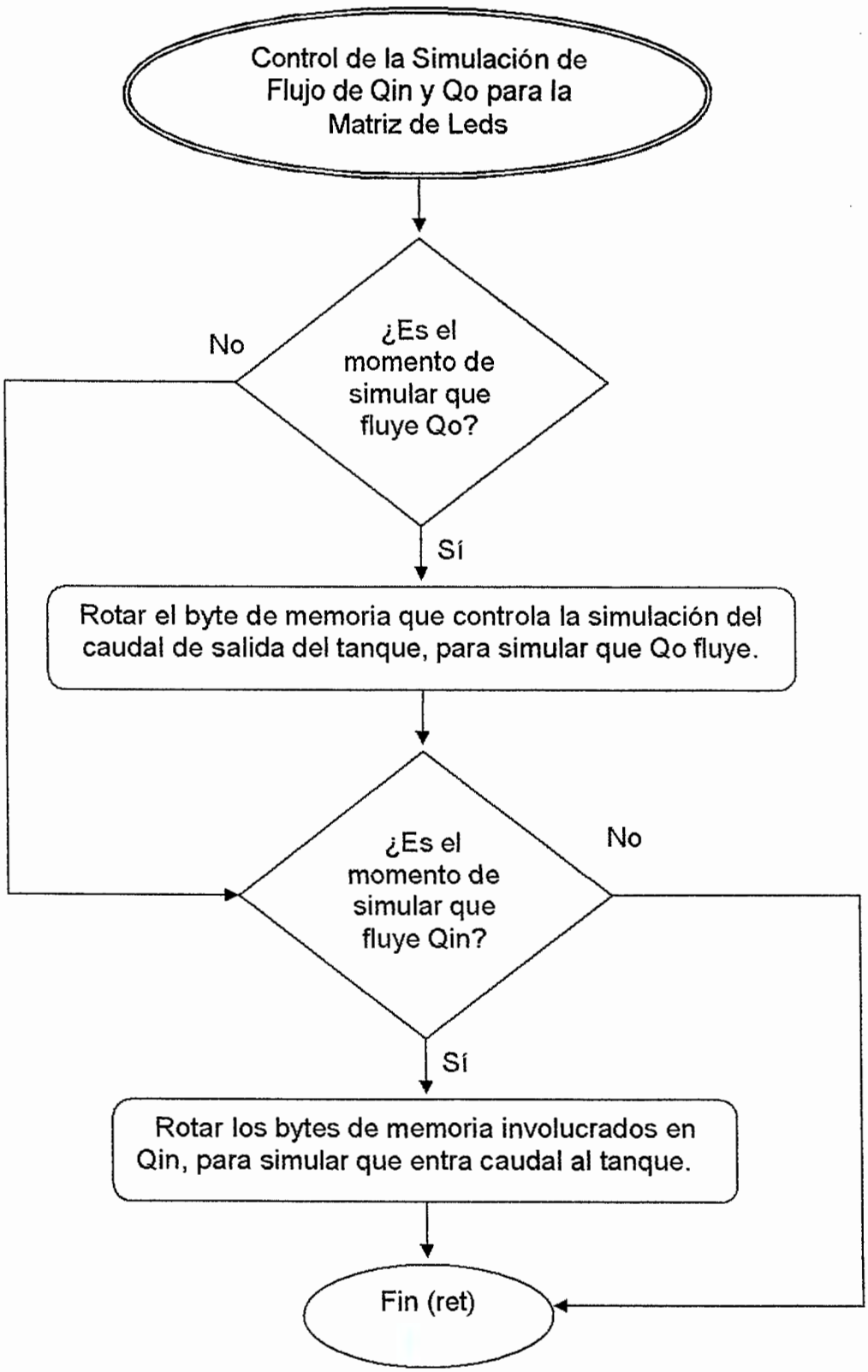
Todo este proceso de cálculos, se efectúa durante el tiempo de una rutina de servicio de la NMI.

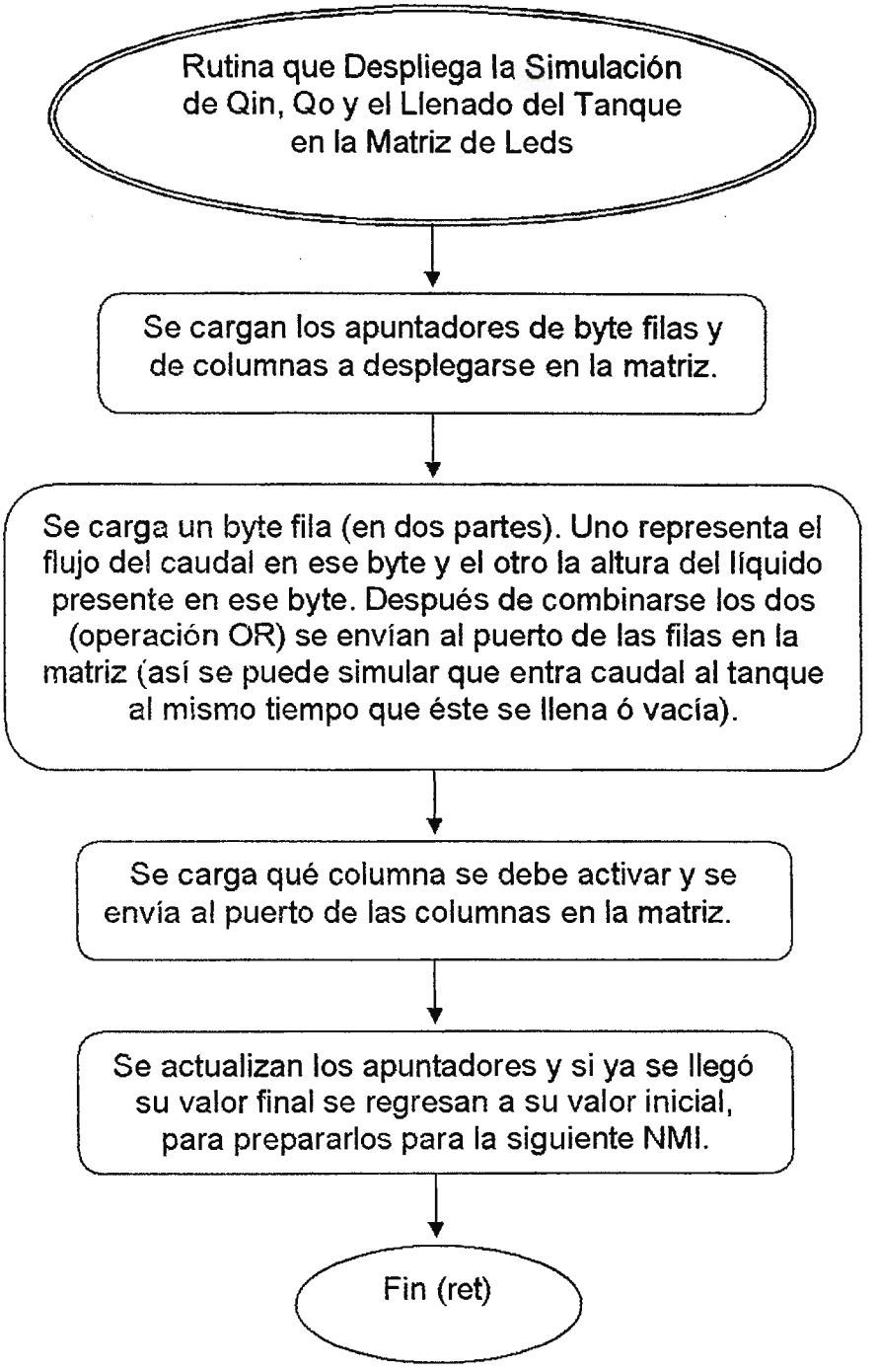
Además, para la implementación de ésta planta, fue necesaria la creación de rutinas auxiliares para realizar operaciones que el microprocesador no puede ejecutar, basándose en algunos métodos "desarrollados" y otros investigados, para llevar a cabo estas operaciones. Entre estas rutinas se encuentran: una básica que se utiliza para permitir la multiplicación al microprocesador, y otras dos que son operaciones más complejas como lo son la división y obtener la raíz cuadrada de un número. Los métodos utilizados por el programa del módulo para realizar dichas operaciones, se presentan en los anexos al final del documento.

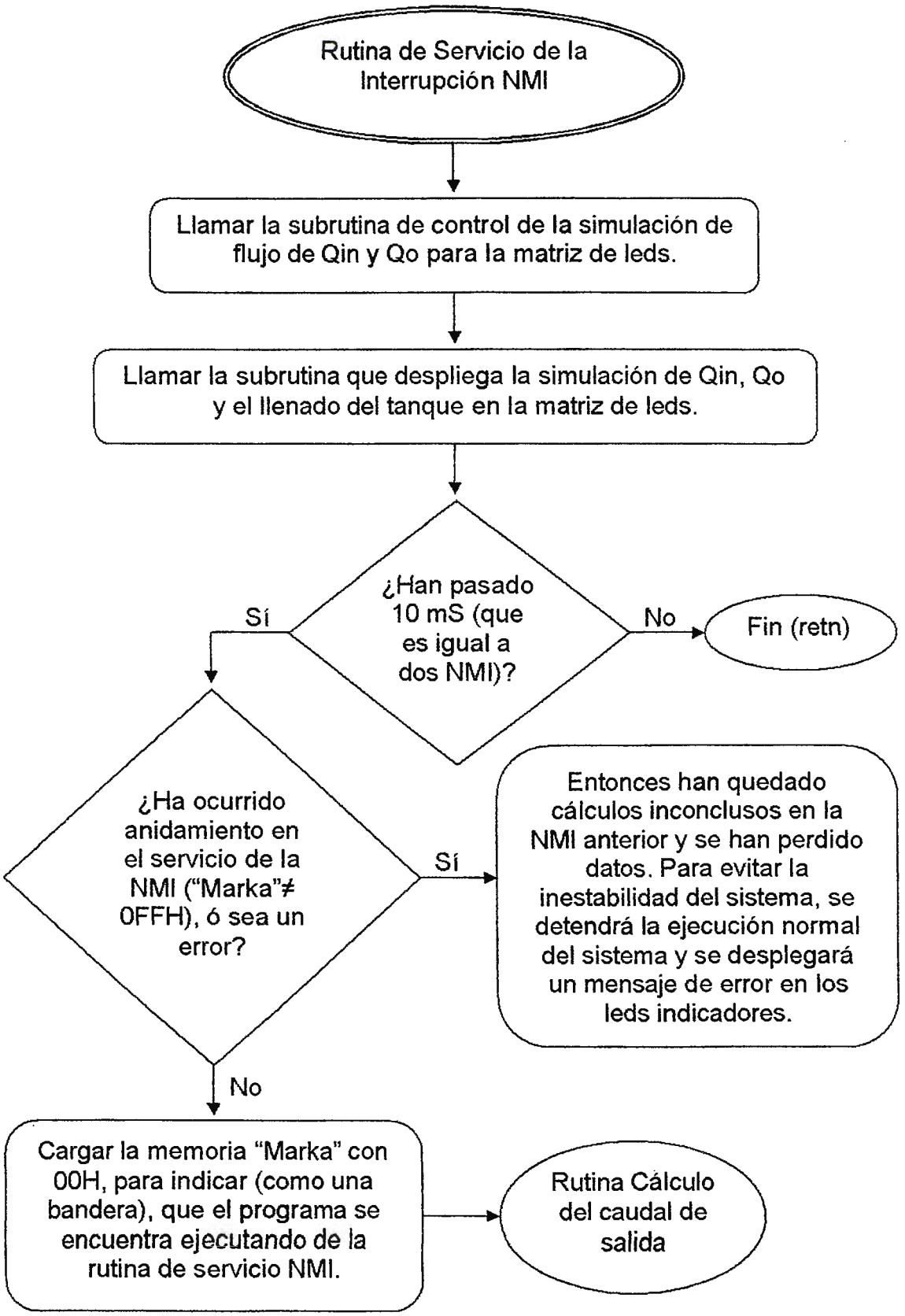
Los siguientes flujogramas describen de forma general, los procesos llevados a cabo por el programa del módulo para la emulación de la planta analógica del tanque de líquidos.

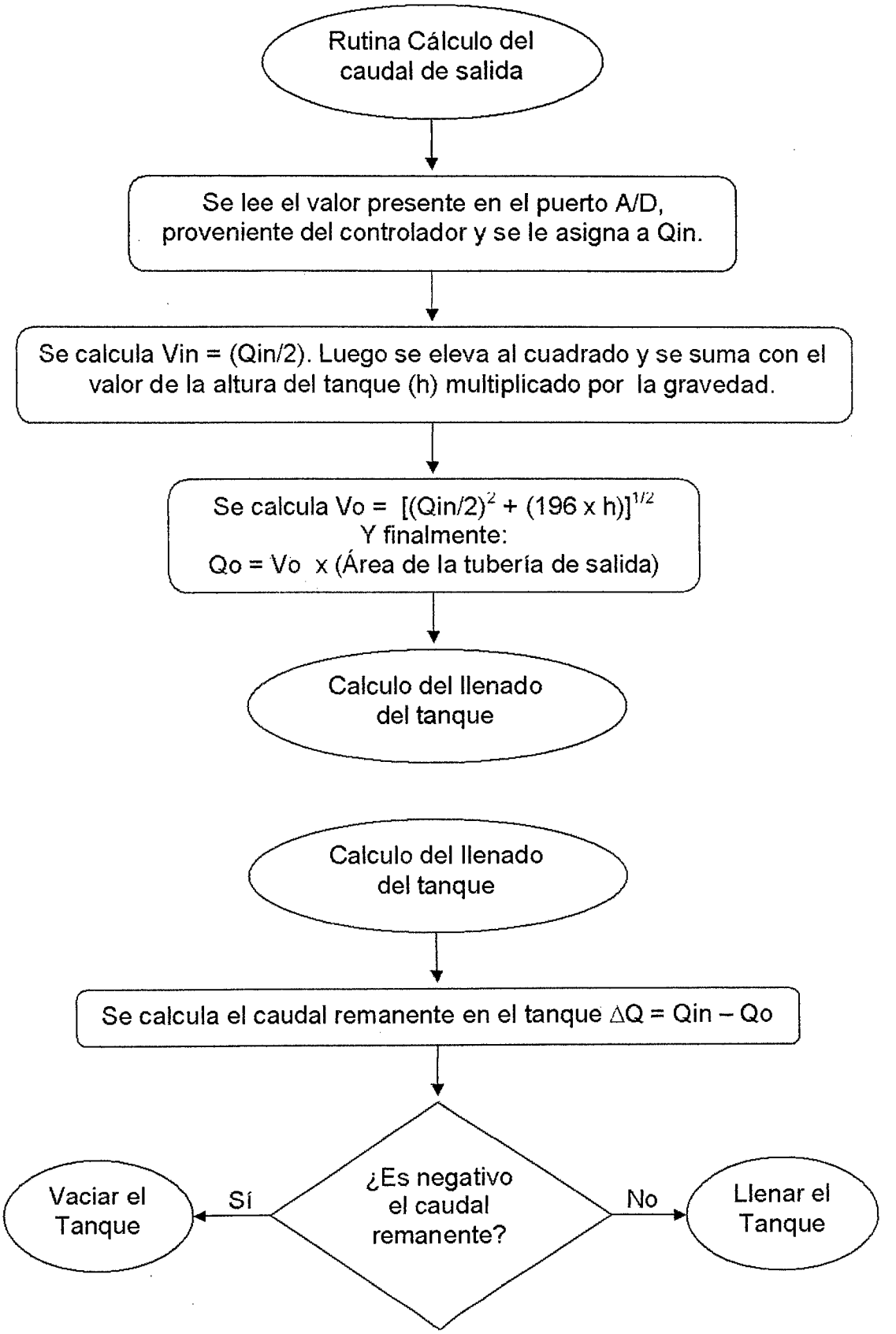
NOTA: todas las variables utilizadas se encuentran descritas en el programa en lenguaje nemotécnico.

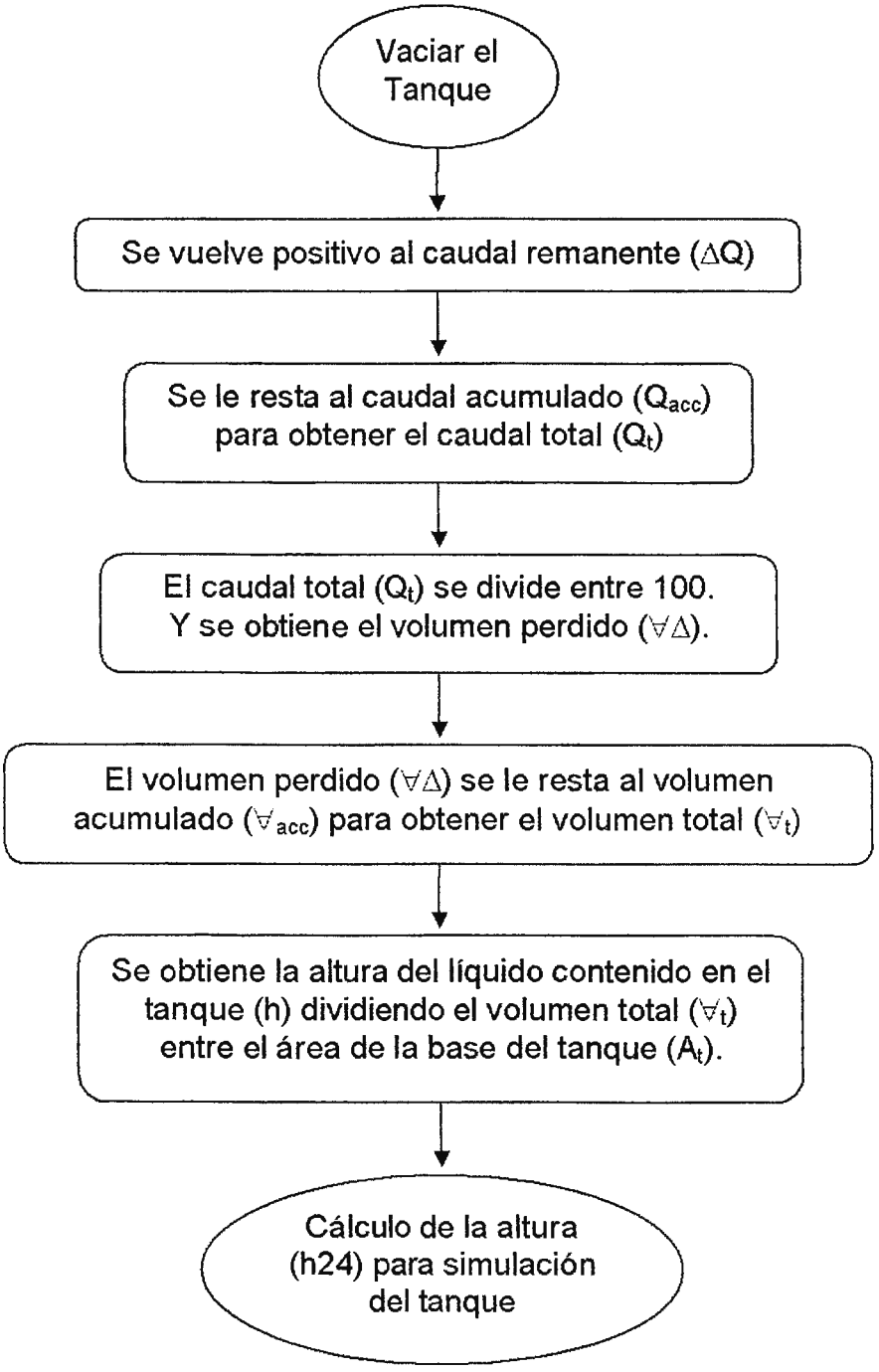


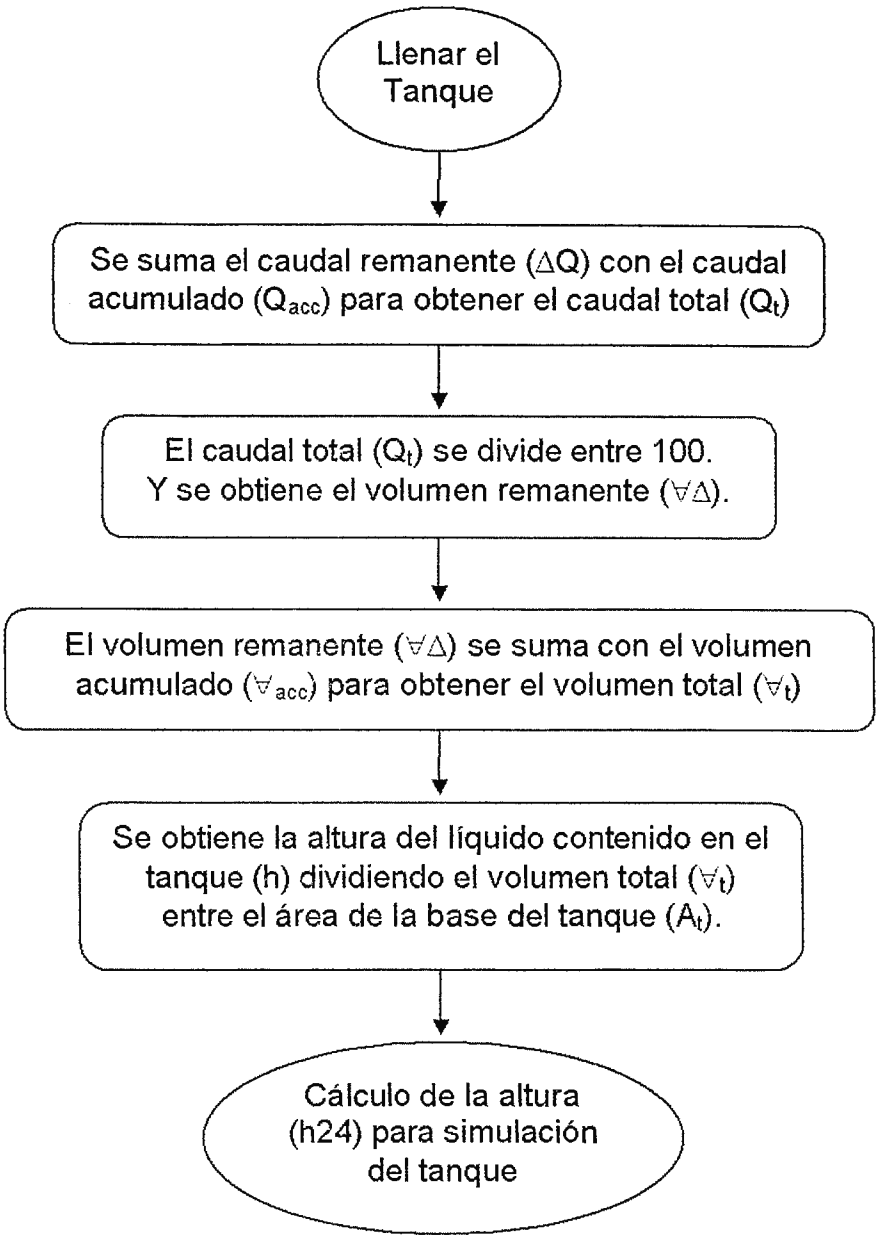

Control de la Simulación de Flujo de Qin y Qo para la Matriz de Leds
¿Es el momento de simular que fluye Qo?
No
Sí
Rotar el byte de memoria que controla la simulación del caudal de salida del tanque, para simular que Qo fluye.
¿Es el momento de simular que fluye Qin?
No
Sí
Rotar los bytes de memoria involucrados en Qin, para simular que entra caudal al tanque.
Fin (ret)

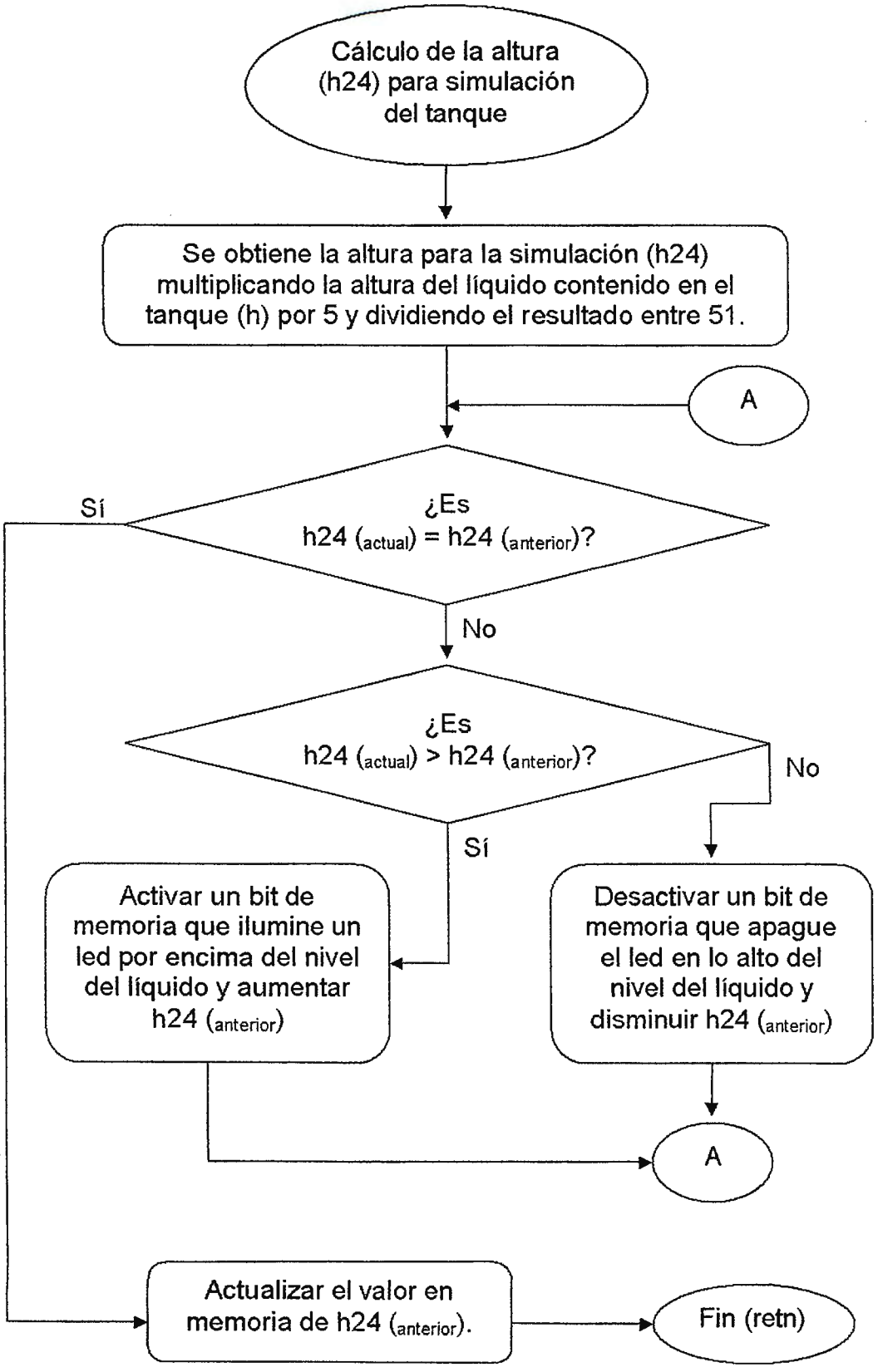

Rutina que Despliega la Simulación de Qin, Qo y el Llenado del Tanque en la Matriz de Leds
Se cargan los apuntadores de byte filas y de columnas a desplegarse en la matriz.
Se carga un byte fila (en dos partes). Uno representa el flujo del caudal en ese byte y el otro la altura del líquido presente en ese byte. Después de combinarse los dos (operación OR) se envían al puerto de las filas en la matriz (así se puede simular que entra caudal al tanque al mismo tiempo que éste se llena ó vacía).
Se carga qué columna se debe activar y se envía al puerto de las columnas en la matriz.
Se actualizan los apuntadores y si ya se llegó su valor final se regresan a su valor inicial, para prepararlos para la siguiente NMI.
Fin (ret)


Rutina de Servicio de la Interrupción NMI
Llamar la subrutina de control de la simulación de flujo de Qin y Qo para la matriz de leds.
Llamar la subrutina que despliega la simulación de Qin, Qo y el llenado del tanque en la matriz de leds.
¿Han pasado 10 mS (que es igual a dos NMI)?
Sí
No
Fin (retn)
¿Ha ocurrido anidamiento en el servicio de la NMI ("Marka"≠ 0FFH), ó sea un error?
Sí
Entonces han quedado cálculos inconclusos en la NMI anterior y se han perdido datos. Para evitar la inestabilidad del sistema, se detendrá la ejecución normal del sistema y se desplegará un mensaje de error en los leds indicadores.
No
Cargar la memoria "Marka" con 00H, para indicar (como una bandera), que el programa se encuentra ejecutando de la rutina de servicio NMI.
Rutina Cálculo del caudal de salida


Rutina Cálculo del caudal de salida
Se lee el valor presente en el puerto A/D, proveniente del controlador y se le asigna a Qin.
Se calcula Vin = (Qin/2). Luego se eleva al cuadrado y se suma con el valor de la altura del tanque (h) multiplicado por la gravedad.
Se calcula Vo = [(Qin/2)^2 + (196 x h)]^1/2
Y finalmente:
Qo = Vo x (Área de la tubería de salida)
Calculo del llenado del tanque
Calculo del llenado del tanque
Se calcula el caudal remanente en el tanque ΔQ = Qin – Qo
¿Es negativo el caudal remanente?
Sí
No
Vaciar el Tanque
Llenar el Tanque


Vaciar el Tanque
Se vuelve positivo al caudal remanente (ΔQ)
Se le resta al caudal acumulado (Q_{acc}) para obtener el caudal total (Q_t)
El caudal total (Q_t) se divide entre 100. Y se obtiene el volumen perdido ($\forall\Delta$).
El volumen perdido ($\forall\Delta$) se le resta al volumen acumulado ($\forall_{acc}$) para obtener el volumen total ($\forall_t$)
Se obtiene la altura del líquido contenido en el tanque (h) dividiendo el volumen total ($\forall_t$) entre el área de la base del tanque (A_t).
Cálculo de la altura (h24) para simulación del tanque


Llenar el Tanque
Se suma el caudal remanente (ΔQ) con el caudal acumulado (Q_{acc}) para obtener el caudal total (Q_t)
El caudal total (Q_t) se divide entre 100. Y se obtiene el volumen remanente ($\forall\Delta$).
El volumen remanente ($\forall\Delta$) se suma con el volumen acumulado ($\forall_{acc}$) para obtener el volumen total ($\forall_t$)
Se obtiene la altura del líquido contenido en el tanque (h) dividiendo el volumen total ($\forall_t$) entre el área de la base del tanque (A_t).
Cálculo de la altura (h24) para simulación del tanque


Cálculo de la altura (h24) para simulación del tanque
Se obtiene la altura para la simulación (h24) multiplicando la altura del líquido contenido en el tanque (h) por 5 y dividiendo el resultado entre 51.
A
¿Es h24 (actual) = h24 (anterior)?
Sí
No
¿Es h24 (actual) > h24 (anterior)?
No
Sí
Activar un bit de memoria que ilumine un led por encima del nivel del líquido y aumentar h24 (anterior)
Desactivar un bit de memoria que apague el led en lo alto del nivel del líquido y disminuir h24 (anterior)
A
Actualizar el valor en memoria de h24 (anterior).
Fin (retn)

7.1. Método de empleado para efectuar la Multiplicación.

El método de multiplicación implementada por el módulo, no es más, que la multiplicación común, con la cual se está familiarizado normalmente. En la cual, primero se multiplica las unidades del multiplicador por el multiplicando, después se multiplican las decenas del multiplicador por el multiplicando, y así sucesivamente... para finalmente sumar éstos productos parciales (respetando su valor posicional) y obtener así el resultado.

Por ejemplo: 55 x 25

$$\begin{array}{r} 55_{10} \times 25_{10} \\ \hline 0125 \\ 125 \\ \hline 1{,}375_{10} \end{array} \equiv \begin{array}{r} 110111_2 \times 11001_2 \\ \hline 00000110111 \\ 0000000000 \\ 000000000 \\ 00110111 \\ 0110111 \\ \hline 10101011111_2 \end{array}$$

7.2. Método desarrollado para efectuar la división.

El método es relativamente sencillo, y consiste en restarle al dividendo el divisor multiplicado por 2^x (para x = n...3, 2, 1, 0), donde si el resultado es positivo, indica que el dos elevado a esa potencia deberá sumarse al cociente para obtener así el resultado, mientras que el dividendo toma el nuevo valor del resultado de la resta. Si el resultado de la resta es negativo, el dividendo conserva su valor y la potencia se descarta y se prueba con su siguiente valor inmediato inferior. El proceso se repetirá hasta que el valor de la potencia sea igual a cero (la cual será la última resta).

Por ejemplo: 55 ÷ 5 = 11.

$$55-(5\times 2^3)=+15 \Rightarrow 2^3+$$
$$15-(5\times 2^2)=-5$$
$$15-(5\times 2^1)=+5 \Rightarrow 2^1+$$
$$5-(5\times 2^0)=0 \Rightarrow \underline{2^0=}$$
$$11$$

7.3. Método de empleado para efectuar la operación de raíz cuadrada.

Este método consiste en el empleo de la siguiente fórmula:

$$G_n = \frac{G_{n-1} + \frac{N}{G_{n-1}}}{2}$$

Donde: n = número de iteraciones (n = 1, 2, 3...)
G = valor de la raíz iterada (G_0 = cualquier valor)
N = número del cual se desea obtener la raíz cuadrada

Su empleo consiste en elegir primero un valor de G_0, donde es recomendable elegir un valor próximo al resultado de la raíz cuadrada para reducir el número de iteraciones (esto sólo si es posible, desde luego). Después se evalúa la fórmula reiteradamente hasta que se consiga que $G_N = G_{N-1}$, en donde éste valor es igual a la raíz cuadrada del número N.

Por ejemplo: $\sqrt{144} = 12$

Primero, se elige un valor inicial de $G_0 = 50$ (que pudo ser cualquier otro valor)
Luego, se procede a calcular:

$$G_1 = \frac{50 + \frac{144}{50}}{2} \cong 26 \qquad G_3 = \frac{15 + \frac{144}{15}}{2} \cong 12$$

$$G_2 = \frac{26 + \frac{144}{26}}{2} \cong 15 \qquad G_4 = \frac{12 + \frac{144}{12}}{2} = 12$$

Como $G_4 = G_3 = 12$ Entonces se detiene el proceso y el resultado de la raíz cuadrada de 144 es 12.

7.4. Ejemplo de un Programa Controlador de Ascensor de Edificio (Desarrollado en Visual Basic)

NOTA: La configuración del puerto serie es la establecida por defecto en VB, al utilizar los controles de dicho puerto (MSComm).

```
Dim ii, z, y, x, control, pis, bajar, subir, buff As Variant
Dim ComEvReceive, i, piso As Integer

Private Sub Form_Load()
   MSComm1.PortOpen = True
   ComEvReceive = 2
   piso = 1
   subir = "s"
   bajar = "a"
   i = 0
End Sub

Private Sub Command0_Click()
  control = 0
   If control < piso Then
      MSComm1.Output = bajar
   End If
End Sub

Private Sub Command1_Click()
   control = 1
   If control < piso Then
      MSComm1.Output = bajar
   End If
   If control > piso Then
      MSComm1.Output = subir
   End If
```

```
End Sub

Private Sub Command2_Click()
   control = 2
   If control < piso Then
      MSComm1.Output = bajar
   End If
   If control > piso Then
      MSComm1.Output = subir
   End If
End Sub

Private Sub Command3_Click()
   control = 3
   If control < piso Then
      MSComm1.Output = bajar
   End If
   If control > piso Then
      MSComm1.Output = subir
   End If
End Sub

Private Sub Command4_Click()
   control = 4
   If control < piso Then
      MSComm1.Output = bajar
   End If
   If control > piso Then
      MSComm1.Output = subir
   End If
End Sub

Private Sub Command5_Click()
```

```
    control = 5
    If control < piso Then
        MSComm1.Output = bajar
    End If
    If control > piso Then
        MSComm1.Output = subir
    End If
End Sub

Private Sub Command6_Click()
    control = 6
    If control < piso Then
        MSComm1.Output = bajar
    End If
    If control > piso Then
        MSComm1.Output = subir
    End If
End Sub

Private Sub Command7_Click()
    control = 7
    If control < piso Then
        MSComm1.Output = bajar
    End If
    If control > piso Then
        MSComm1.Output = subir
    End If
End Sub

Private Sub Command8_Click()
    control = 8
    If control > piso Then
        MSComm1.Output = subir
```

```
    End If
End Sub

Private Sub Command9_Click()
    MSComm1.PortOpen = False
    End
End Sub

Private Sub MSComm1_OnComm()

    If MSComm1.CommEvent = ComEvReceive Then

        MSComm1.InputLen = 2
        ii = MSComm1.Input
        Text1.Text = ii
        MSComm1.InputLen = 1
        pis = MSComm1.Input
        Text2.Text = pis

        i = Val(ii)
        piso = Val(pis)
        Text3.Text = i

        z = i - 1
        y = i + 1

            If (i > 0) And (i < 23) Then
              lbl_a(z).BackColor = &HFFFFFF
              lbl_a(i).BackColor = &HFF00&
              lbl_a(y).BackColor = &HFFFFFF
            End If
            If (i = 0) Then
              lbl_a(i).BackColor = &HFF00&
```

```
        lbl_a(y).BackColor = &HFFFFFF
      End If
      If (i = 23) Then
        lbl_a(z).BackColor = &HFFFFFF
        lbl_a(i).BackColor = &HFF00&
      End If

    lbl_piso.Caption = piso

    If control < piso Then
       MSComm1.Output = bajar
    End If
    If control > piso Then
       MSComm1.Output = subir
    End If

  End If
End Sub
```

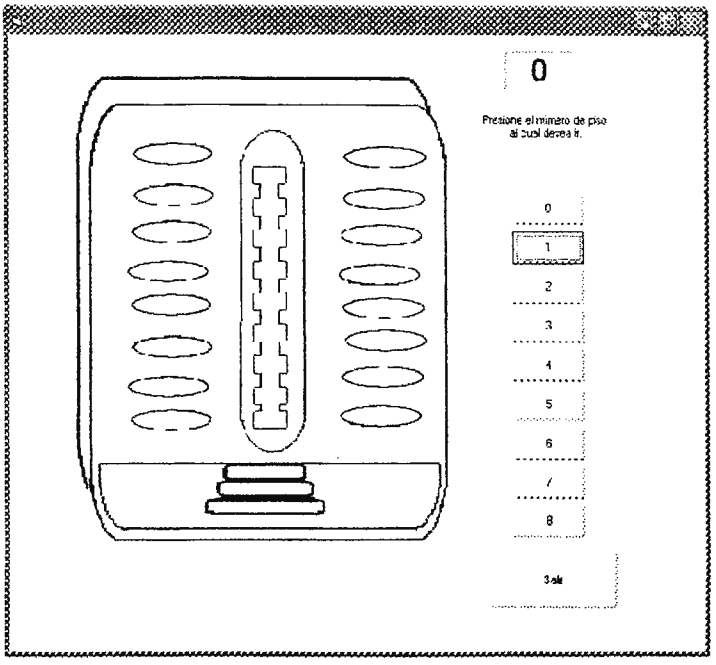


Fig. 20 Visualización del controlador de la Planta Digital elaborado en VB

7.5. Programa en lenguaje nemotécnico de la planta digital: ascensor de edificio.

```
p_disp:     .equ  0FFH     ; dirección de los leds de uso general
filas:      .equ  0DDH     ; dirección del puerto de control de las filas
columnas:   .equ  0EEH     ; dirección del puerto de control de las columnas



;*****************************************************************************

;              MAIN

;-----------------------------------------------------------------------------

    .org  0000H


    xor   A              ; se borra el acc. A
    out   (filas),A      ; se borra cualquier dato presente en el puerto de filas
    out   (columnas),A; se borra cualquier dato presente en el p. de columnas

    xor   A
    out   (0A1H),A       ; sin interrupciones
    ld    A,83H
    out   (0A3H),A
    xor   A
    out   (0A1H),A
    ld    A,1AH
    out   (0A0H),A       ; 9600 bps
    ld    A,03H
    out   (0A3H),A       ; sin paridad, stop1 y 8 bits

inic:
    call  retraso        ; se llama una subrutina de retraso de tiempo
    call  recib_dat      ; llama a la subrutina de espera por dato del p. serie
    cp    'b'            ; se compara si el caracter fue la letra b (para bajar)
    jp    p,subir        ; salta si no se ordena bajar, a la rutina "subir"

    ld    HL,pis0_7      ; carga la dir. del byte que controla los 8 primeros Leds
    sla   (HL)           ; se desplaza a la izq. para simular que el ascensor baja
    inc   L              ; se apunta a la dirección de los siguientes 8 Leds
    rl    (HL)           ; se desplaza a la izq. para simular que el ascensor baja
    inc   L              ; apunta a la dirección de los últimos 8 Leds del edificio
    rl    (HL)           ; se desplaza a la izq. para simular que el ascensor baja
    jp    nc,baj         ; se comprueba si era el último led
    ld    (HL),80H       ; si lo era regresar la rotación
baj:
```

```
    inc     L               ; se apunta al contador indicador de led
    jp      c,baj2          ; si era el último led saltar
    dec     (HL)            ; si no lo era decrementar el contador indicador de led
baj2:
    ld      A,(HL)          ; se carga el contador indicador de led en el acc. A
    jp      calc            ; salta a la rutina de cálculo de piso

subir:
    ld      HL,pis16_23 ; carga la dir. del byte que controla los 8 últimos Leds
    srl     (HL)            ; desplaza a la der. para simular que el ascensor sube
    dec     L               ; se apunta a la dirección de los siguientes 8 Leds
    rr      (HL)            ; desplaza a la der. para simular que el ascensor sube
    dec     L               ; apunta a la dirección de los últimos 8 Leds del edificio
    rr      (HL)            ; desplaza a la der. para simular que el ascensor sube
    jp      nc,sub          ; se comprueba si era el último led
    ld      (HL),01H        ; si lo era regresar la rotación
sub:
    ld      HL,cont         ; se apunta al contador indicador de led
    jp      c,sub2          ; si era el último led saltar
    inc     (HL)            ; si no lo era incrementar el contador indicador de led
sub2:
    ld      A,(HL)          ; se carga el contador indicador de led en el acc. A
    jp      calc            ; salta a la rutina de cálculo de piso

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;
;       Subrutina de retraso
;
;-----------------------------------------------------------------------------

retraso:
    ld      HL,cont_retr ; se carga la dirección del contador de retraso
    ld      (HL),100        ; se carga con 100 el contador
    xor     A               ; se borra el contenido del acc. A
retras:
    cp      (HL)            ; se compara si el contador ya llegó al valor de cero
    jp      nz,retras       ; si no lo es salta
    ret

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;
```

```
;          Rutina de Servicio de Interrupción NMI
;    Mantiene visible la imagen del proceso en la matriz de led

;-------------------------------------------------------------------------------


      .org  0066H

    ex      AF,AF'          ; se almacena el contenido de los acumuladores
    exx                     ; se almacena el contenido de los acumuladores

    ld      HL,(fil)        ; carga la dir. del  grupo de filas a desplegar en la matriz
    ld      A,(HL)          ; se carga el contenido de éstas 8 filas al acc. A
    out     (filas),A       ; se envía al puerto correspondiente

    ld      A,(col)         ; carga la col. que corresponde a las filas ya cargadas
    out     (columnas),A; se envía al puerto correspondiente
    srl     A               ; prepara el apuntador de columna para la siguiente NMI
    ld      (col),A         ; se guarda este nuevo valor en memoria
    ld      A,L             ; carga el apuntador de las filas enviadas al puerto
    dec     A               ; se decrementa

    jp      p,findisp       ; si el valor es positivo (es decir, válido), saltar
    ld      A,04H           ; si no lo es, resetea el apuntador de col. al valor inicial
    ld       (col),A        ; se guarda este valor en memoria
    srl     A               ; se resetea el apuntador de fila al valor inicial
findisp:
    ld      (fil),A         ; se guarda este valor en memoria

    ld      HL,cont_retr ; se carga la dirección del contador de retraso
    dec     (HL)            ; se decrementa el contador

    exx                     ; se recupera el valor original de los acumuladores
    ex      AF,AF'          ; se recupera el valor original de los acumuladores

    retn


;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
;*****************************************************************************
;
;       Rutina de Calculo de Piso a partir del indicador de Led
;         En el acc. A se provee el valor del indicador de led
;
;-----------------------------------------------------------------------------

calc:
    ld      D,A             ; se carga el acc. D, con el valor del acc. A
    ld      B,0             ; se carga el acc. B con cero
calc2:
    cp      0AH             ; se compara el valor del acc. A con 10
    jp      p,decen         ; si es mayor (hay decenas), salta
    add     A,30H           ; si no, se convierte en número ascii.
    ld      H,A             ; se transfiere el valor al acc. H
    ld      A,B             ; se transfiere el valor del acc. B al acc. A
    add     A,30H           ; se convierte en número ascii.
    call    trans_dat       ; llama la subrutina de transmisión de datos por p. serie
    ld      A,H             ; se recupera el valor almacenado en el acc. H
    call    trans_dat       ; llama la subrutina de transmisión de datos por p. serie
    ld      A,D             ; se recupera el valor almacenado en el acc. D
    jp      calc3

decen:
    sub     0AH             ; se resta una decena del acc. A (A = A - 10)
    inc     B               ; se incrementa el contador de decenas
    jp      calc2

calc3:
    ld      B,00H           ; se borra el contenido del acc. B
    cp      B               ; se compara si el acc. A es cero
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $02             ; se compara si el acc. A es 2
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $05             ; se compara si el acc. A es 5
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $08             ; se compara si el acc. A es 8
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $0B             ; se compara si el acc. A es 11
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $0E             ; se compara si el acc. A es 14
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $11             ; se compara si el acc. A es 17
```

```
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $14             ; se compara si el acc. A es 20
    jp      z,pisso         ; si lo es salta
    inc     B               ; si no, incrementa el acc. B (número de piso)
    cp      $17             ; se compara si el acc. A es 23
    jp      z,pisso         ; si lo es salta
    ld      A,(piso)        ; se carga el último nivel de piso enviado
    ld      B,A             ; se transfiere al acc. B
pisso:
    ld      A,B             ; se transfiere al acc. A
    ld       (piso),A       ; se almacena en memoria
    add     A,30H           ; se convierte en ascii
    call    trans_dat       ; llama la subrutina de transmisión de datos por p. serie
    jp      inic            ; salta al inicio

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    .org    2000H           ; dirección de la RAM

pis0_7:         .db   $00
pis8_15:        .db   $00
pis16_23:       .db   $80

cont:           .db    $0
fil:            .dw    pis16_23
col:            .db    $04

cont_retr:      .db    $00
piso:           .db    $00
unid:           .db    $0

        .end
```

7.6. Programa en lenguaje nemotécnico de la planta analógica: tanque de líquidos.

```
pdac: .equ   080H      ; puerto D/A
pad:  .equ   099H      ; puerto A/D
pfil:  .equ  0DDH      ; puerto que controla las filas de la matriz de leds
pcol:  .equ  0EEH      ; puerto que controla las columnas de la matriz de leds
pdisp: .equ  0FFH      ; puerto de leds indicadores de uso general
nmi:  .equ   0066H     ; dirección a la cual accesa el Z80 ante una NMI

    .org  0000H

    jp      0400H      ; saltar al main

    .org  nmi

    jp      0180H      ; saltar a rutina de servicio de NMI


;*****************************************************************************

;              MAIN

;-----------------------------------------------------------------------------

    .org    0400H

    xor     A          ; se borra el acc. A
    out     (pfil),A   ; se borra cualquier dato en el puerto de filas
    out     (pcol),A   ; se borra cualquier dato en el puerto de columnas

    ld      BC,0000H       ; se borra el acc. BC
    ld      (RxMemN),BC    ; se borra el contenido de la memoria RxMemN

    ld      (Qo),A     ; se borra el contenido de la memoria Qo
    ld      (h24),A    ; se borra el contenido de la memoria h24
    ld      (h),A      ; se borra el contenido de la memoria h
    ld      (tank0),A  ; se borra el contenido de la memoria tank0
    ld      (tank11),A ; se borra el contenido de la memoria tank11
    ld      (tank2),A  ; se borra el contenido de la memoria tank2

Inic:
    ld      A,(Qin)    ; se carga el caudal de entrada Qin en el acc. A
    out     (pdisp),A  ; se envía a los leds de uso general
    ld      A,0FFH     ; se carga el acc. A con 255
```

```
    ld      (marka),A    ; se transfiere el acc. A a la memoria marka (bandera)

    ld      A,(Qin)      ; se carga el caudal de entrada Qin en el acc. A
    and     0F8H         ; se eliminan los 3 bits menos significativos
    jp      nz,Inic2     ; si A != 0 entonces salta
    ld      (fi),A       ; si no, se borra el valor a fi..elimina la simulación de Qin
    jp      Inic3        ; salta para la siguiente prueba

Inic2: ld   A,(fi)       ; se transfiere el valor de fi al acc. A
    or      A            ; se verifica si existe simulación actualmente de Qin
    jp      nz,Inic3     ; si la hay salta hacia la siguiente prueba
    ld      A,11H        ; sino se inicia la simulación de Qin, cargando el acc. A
    ld      (fi),A       ; luego se transfiere a la memoria fi

Inic3: ld   A,(h24)      ; se transfiere el valor de h24 al acc. A
    or      A            ; se verifica si la altura del tank es cero (ningún led "on")
    ld      A,(fo)       ; se carga el valor de la simulación de Qo al acc. A
    jp      nz,Inic4     ; si el tank ha ganado altura, salta
    and     0F8H         ; sino, se eliminan los 3 bits menos sig. del acc. A
    ld      (fo),A       ; se transfiere el valor nuevamente a la mem.
    jp      nz,Inic      ; si el valor transferido no fue cero, salta
    ld      A,(fc2)      ; sino, se carga el byte que controla el fondo del tank
    and     080H         ; se prueba si se debe simular una "gota" de Qo
    jp      z,Inic       ; si no hay que simularla, salta
    ld      A,08H        ; si se tiene que simular, carga el acc. A con una "gota"
    ld      (fo),A       ; luego se transfiere a la memoria que controla Qo
    jp      Inic         ; vuelve al inicio

Inic4: or     A          ; se comprueba si Qo = 0
    jp      nz,Inic      ; si no es cero, salta al inicio
    ld      A,88H        ; sino se inicia la simulación de Qo, cargando el acc. A
    ld      (fo),A       ; luego se transfiere a la mem. fo (inicia la simul. de Qo)
    jp      Inic         ; vuelve al inicio

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;
;       Subrutina de control del flujo en la Matriz de Leds
;
;-----------------------------------------------------------------------------

      .org   00A0H

tanque:
    push  AF             ; se guarda el contenido de los acc. en la pila
    push  BC
```

```
    push  HL

    ld    HL,ContQo   ; se carga la dirección del controlador de Veloc. de flujo
    dec   (HL)        ; decrementa el contador controlador de Veloc. de Qo
    ld    HL,fo       ; se apunta al controlador de flujo de salida
    jp    nz,tankef   ; si ContQo != 0 salta
    ld    A,(h)       ; sino, se transfiere la altura del tank al acc. A
    cpl               ; se complementa el acc. A
    srl   A           ; se rota el acc. A a la derecha
    srl   A
    srl   A
    add   A,04H       ; A = A + 4 (4 valor mín. de ContQo p/ Qo = máx.
Veloc.)
    ld    (ContQo),A  ; se transfiere a la mem. ContQo
    rlc   (HL)        ; se rota a la izq. fo
tankef:
    ld    A,(HL)      ; se trasfiere el valor de la mem. fo al acc. A
    and   0F0H        ; se elimina el nibble menos significativo
    ld    B,A         ; se transfiere a B


    ld    A,(ContQi)  ; se carga A con contador controlador de Veloc. de Qin
    dec   A           ; se decrementa
    ld    (ContQi),A  ; se transfiere a la mem. ContQi
    jp    nz,tankef2  ; salta si ContQi != 0
    ld    A,(Qin)     ; sino se carga el valor del caudal de entrada en A
    cpl               ; se complementa el acc. A
    srl   A           ; se rota a la derecha
    add   A,02H       ; A = A + 2 (2 valor mín. de ContQi p/ Qi = máx. Veloc.)
    ld    (ContQi),A  ; se transfiere a la mem. ContQi

    inc   L           ; se apunta al flujo de entrada
    rrc   (HL)        ; se rota a la derecha (simula la entrada del caudal)
    inc   L           ; se apunta a fc0
    rl    (HL)        ; se rota a la izquierda (simula la entrada del caudal)
    inc   L           ; se apunta a fc1
    rl    (HL)        ; se rota a la izquierda (simula la entrada del caudal)
    inc   L           ; se apunta a fc2
    rl    (HL)        ; se rota a la izquierda (simula la entrada del caudal)

tankef2:
    ld    A,(fi)      ; se carga el valor del controlador de flujo de entrada
    and   0FH         ; se elimina el nibble más significativo
    or    B           ; A = A || B
    ld    (fc3),A     ; se transfiere a la mem. fc3 (controla Qi y Qo simulado)

    pop   HL          ; se recupera el valor previo de los acc. guardados en la
    pop   BC          ; pila
    pop   AF
```

```
    ret                     ; retorno de subrutina

;////////////////////////////////////////////////////////////////////////////////////////////////////

;*****************************************************************************
;
;       Subrutina que presenta la simulación
;          del tanque en la Matriz de Leds
;
;-----------------------------------------------------------------------------------------------

      .org 0100H

display:
    exx                     ; se almacena temporalmente el contenido de los acc.
    ex      AF,AF'

    ld      HL,col          ; carga la dirección de la mem. col (columnas)
    ld      DE,(fcx)        ; carga el contenido de la mem. fcx (filas a desplegar)
    push    DE              ; almacena la  direcc. de filas a desplegar (DE) en la pila
    ld      BC,0007H        ; se carga BC con 7

    ld      A,(HL)          ; se carga la columna a activarse en la matriz
    out     (pcol),A        ; se envía al puerto que controla las columnas
    ld      A,(DE)          ; carga el flujo del byte fila que se activará en la matriz
    ex      DE,HL           ; se intercambia el contenido de los acc. HL y DE
    add     HL,BC           ; se apunta a la altura del tank que corresponde a la fila
    ld      C,A             ; guarda temporalmente el flujo del byte fila en el acc. C
    or      (HL)            ; se orea la altura y el flujo (ambos) de éste byte fila
    out     (pfil),A        ; se envía al puerto que controla las filas de la matriz
    ld      A,(HL)          ; se carga el nivel de la altura del tank en el acc. A
    cpl                     ; se complementa el acc. A
    and     C               ; A = A and C (se elimina el flujo cubierto por la altura)
    pop     HL              ; carga el acc. HL con la direcc. del byte fila a desplegar
    ld      (HL),A          ; se transfiere el nuevo flujo a la mem. correspondiente
    ex      DE,HL           ; se intercambia el contenido de los acc. HL y DE

    rlc     (HL)            ; se rota a la izq. la col. p/ la siguiente simulación
    jp      nc,noccol       ; saltar si no era la última columna

    ld      DE,fc0          ; sino reiniciar y apuntar al primer byte fila
    ld      (fcx),DE        ; y cargarla al apuntador de byte fila a desplegar
    jp      findisp         ; saltar al final de la subrutina

noccol:
    inc     E               ; se apunta al siguiente byte fila a desplegar
```

```
    ld      (fcx),DE        ; cargar la mem. del apuntador de byte fila a desplegar

findisp:
    exx                     ; se recupera el valor que tenían previamente los acc.
    ex      AF,AF'

    ret                     ; retorno de subrutina

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

;*****************************************************************************

;       Rutina de Servicio de Interrupción NMI

;-----------------------------------------------------------------------------------------------

      .org    0180H

RutNmi:
    push    AF              ; se guarda el contenido de los acc. en la pila
    push    HL

    call    tanque          ; llama a la subrutina de control de flujo en la matriz
    call    display         ; llama a la subrutina que envía la simulación a la matriz

    ld      A,(contTank) ; se carga el contador controlador de tiempo
    dec     A               ; se comprueba si han pasado 10 mS (= dos NMI)
    jp      nz,FFin         ; si no han pasado no realizar los cálculos y saltar

    ld      A,(marka)       ; se carga la bandera de overflow de NMI
    cp      00H             ; compara si de daba servicio aún a alguna NMI anterior
    jp      z,Error2        ; si hubo overflow salta porque hay un error
    xor     A               ; sino se carga el acc. A con cero
    ld      (marka),A       ; se transfiere a la bandera p/ indicar el inicio de la NMI

    ld      DE,0000H        ; borra el contenido del acc. DE
    ld      (RxMem1),DE     ; se borran datos de cálculos anteriores de la raíz
    ld      (RxMemG),DE


;------- Calculo del caudal de salida ------------
;   Vo (velocidad del caudal de salida) = [ (Qin/2)^2 + (196)(h) ]^(1/2)
;   Qo = Vo  x (Área de la tubería de salida)


    ld      HL,(h)          ; carga L = altura del tanque (h) y H = Qin
```

```
    ld      A,0FCH      ; se carga la máscara 0FCH en el acc. A
    and     H           ; se eliminan los 2 bits menos significativos de Qin
    cp      0FCH        ; se comprueba si los bits restantes son unos
    jp      nz,salto    ; si no lo son salta
    and     L           ; se eliminan los 2 bits menos significativos de h
    cp      0FCH        ; se comprueba si los bits restantes son unos
    jp      nz,salto    ; si no lo son salta
    ld      DE,00FEH    ; sino, entonces se da que Qin = Qin(max) y h = h(max)
    jp      FinQo       ; y se carga DE con Qo(max) = 0FEH, luego salta

salto:
    in      A,(pad)     ; lee del puerto A/D el valor de Qin (del controlador)
    ld      (Qin),A     ; se almacena en la mem. Qin
    ld      H,D         ; borra el contenido del acc. H
    srl     A           ; se divide entre 2 Qin
    ld      L,A         ; se transfiere al acc. L
    call    mult        ; se eleva al cuadrado Qin/2  y HL = resultado
    ex      DE,HL       ; se transfiere el resultado a DE (mientras HL = 0000H)
    ld      L,196       ; carga la gravedad en el acc. L
    ld      A,(h)       ; se carga la altura del líquido del tanque en el acc. A
    call    mult        ; se multiplica la gravedad por la altura y HL = resultado
    xor     A           ; borra el contenido del acc. A
    adc     HL,DE       ; se efectúa la suma: HL = (Qin/2)^2 + (196)(h)
    jp      z,FinRx     ; si el resultado fue cero, salta (no hay que sacar raíz)
    ld      D,A         ; borra el contenido del acc. D

    rr      H
    rr      L           ; se divide entre dos (Qin/2)^2 + (196)(h)
    cp      H           ; prueba si el valor  es mayor a 255
    jp      z,RxL       ; si no lo es, salta
    ld      E,H         ; sino, se transfiere el byte más significativo al acc. E
    ld      A,1FH       ; carga la raíz aprox. más baja para un  num. > 255
AprRx1:
    srl     E           ; se rota 2 veces para comprobar
    srl     E           ; si es la raíz aproximada del número
    jp      z,RxCalc    ; salta si lo es (A = núm. aprox. más cerca de la raíz)
    scf
    rla                 ; sino prueba con la siguiente aproximación
    jp      AprRx1
RxL:
    cp L                ; se prueba si L = 0
    jp      z,FinRx     ; si lo es, salta (no hay que sacar raíz, resultado = 0)
    ld      E,L         ; sino, se transfiere el byte menos significativo al acc. E
    ld      A,01H       ; carga la raíz aprox. más baja para un  num. < 255
    jp      AprRx1

RxCalc:                 ; programa que calcula la raíz cuadrada de un núm.
    ld      (RxMemN),HL ; almacena en mem. el núm. al cual se le sacara raíz
    ld      D,00H       ; se borra el contenido del acc. D
```

```
    ld      E,A             ; se transfiere la aproximación de la raíz al acc. E
PrbRx:
    ld      B,D             ; borra el contenido del acc. B
    ld      C,E             ; se transfiere la aproximación de la raíz al acc. C
    call    div             ; HL = (num. a sacar la raíz) / (aprox. de la raíz)
    ld      D,B
    ld      E,C             ; se transfiere la aproximación de la raíz al acc. DE
    srl     B
    rr      C               ; divide entre dos el núm. aproximado
    add     HL,BC           ; se calcula la siguiente aproximación de la raíz
    ld      (RxMemG),HL     ; se almacena en memoria la última aprox.
    sbc     HL,DE           ; se compara con la aproximación anterior
    jp      z,FinRx         ; si es igual, salta (se encontró la raíz del núm.)
    ld      BC,(RxMem1)     ; sino, se carga la antepenúltima aprox.
    ld      HL,(RxMemG)     ; se carga la última aprox.
    or      A               ; se borra el carry
    sbc     HL,BC           ; se compara con ésta aproximación
    jp      z,FinRx         ; si es igual, salta (se encontró la raíz del núm.)
    ld      (RxMem1),DE     ; sino se actualiza la aprox. anterior en la penúltima
    ld      DE,(RxMemG)     ; se carga la última aprox.
    ld      HL,(RxMemN)     ; se carga el núm. al cual se le sacará la raíz
    jp      PrbRx           ; salta a ejecutar nuevamente el algoritmo

FinRx:
    ld      HL,(RxMemG)     ; carga el resultado de la  raíz cuadrada en HL (=Vo)
    ld      D,H
    ld      E,L             ; se transfiere a DE
    ld      BC,0064H        ; se carga BC con 100
    xor     A               ; borra la bandera de carry
    sbc     HL,BC           ; prueba si el resultado excedió el valor de 100
    jp      m,FinQo         ; si no es así salta (se ha calculado Qo)
    dec     DE              ; si era mayor a 100 se aplica un factor de corrección
    sbc     HL,BC           ; prueba si el resultado excedía el valor de 200
    jp      m,FinQo         ; si no es así salta (se ha calculado Qo)
    dec     DE              ; si era mayor a 200 se aplica otro factor de correcc.

FinQo:
    Ld      A,E             ; el resultado final (Qo) se transfiere al acc. A
    ld      (Qo),A          ; se almacena en memoria como Qo


;-------- Calculo del llenado del tanque ----------


Tankq:
    ld      A,(Qin)         ; se carga el valor de Qin en el acc. A
    or      A               ; prueba si el valor es cero
    jp      nz,Tankq2       ; si no lo es, salta (inicia los cálculo de la altura del tank)
    cp      E               ; si es cero se prueba si Qo es cero también
```

```
    jp      nz,Tankq2   ; si no lo es, salta (inicia los cálculo de la altura del tank)

    ld      HL,0000H    ; sino, es que Qin = Qo = 0, y se borra el acc. HL
    ld      (Caudal),HL ; Caudal = 0, porque no existe caudal remanente
    ld      (Vol),HL    ; también se borra la mem. de volumen temporal
    out     (pdac),A    ; se borra cualquier dato presente en el puerto D/A
    jp      fintank     ; salta al final de ésta etapa, porque no hay cálculos

Tankq2:
    ld      L,A         ; transfiere Qin al acc. L
    xor     A           ; borra el acc. A
    ld      H,A         ; borra el acc. H
    sbc     HL,DE       ; calcula el caudal remanente en el tank (HL = Qin – Qo)
    ld      DE,(Caudal) ; DE = "Caudal" ya acumulado en el tanque
    jp      m,Vaciar    ; si el resultado fue negativo, saltar a la etapa: vaciar

    add     HL,DE       ; sino, HL = caudal remanente + caudal acumulado
    ld      A,0FEH      ; se carga una máscara en el acc. A
    and     H           ; se prueban los 7 bits más sig. [si caudal(actual) > 511]
    jp      z,Tank1     ; si no se ha sobrepasado Caudal(max) = 511, salta
    ld      HL,01FFH    ; si es mayor a 511, entonces se transfiere 511 a HL
    jp      Tank1       ; salta a Tank1
Vaciar:
    ld      H,A         ; borra el acc. H
    or      L           ; se transfiere el acc. L al acc. A y se borra el carry
    neg                 ; se obtiene el complemento A dos del acc. A
    ld      L,A         ; se transfiere el resultado al acc. L
    ex      DE,HL       ; HL = caudal acumulado y DE = caudal remanente
    xor     A           ; borra el acc. A y borra el carry
    sbc     HL,DE       ; HL = caudal acumulado - caudal remanente

    ld      (Caudal),HL ; almacena el valor actualizado del caudal acumulado
    ld      DE,(Vol)    ; carga el volumen acumulado en el tanque al acc. DE
    ld      BC,100      ; carga BC = 100
    xor     A           ; borra el carry
    adc     HL,BC       ; prueba si ya se vació un litro (al volumen del tank)
    jp      p,contin    ; si no es así, salta
    ld      (Caudal),HL ; sino, guarda el valor actualizado del caudal acumulado
    dec     DE          ; se decrementa el volumen en un litro
    xor     A           ; borra el carry
    adc     HL,BC       ; prueba si se ha vaciado un litro (al volumen del tank)
    jp      p,temp1     ; si no es así, salta
    ld      (Caudal),HL ; sino, guarda el valor actualizado del caudal acumulado
    dec     DE          ; se decrementa el volumen en un litro
    xor     A           ; borra el carry
    adc     HL,BC       ; prueba si se ha vaciado un litro (al volumen del tank)
    jp      p,temp1     ; si no es así, salta
    ld      (Caudal),HL ; sino, guarda el valor actualizado del caudal acumulado
    dec     DE          ; se decrementa el volumen en un litro
```

```
    xor     A               ; borra el carry
    adc     HL,BC           ; prueba si se ha vaciado un litro (al volumen del tank)
    jp      p,temp1         ; si no es así, salta
    dec     DE              ; se decrementa el volumen en un litro
    ld      (Caudal),HL     ; sino, guarda el valor actualizado del caudal acumulado
    jp      temp1           ; salta a temp1


Tank1:
    ld      (Caudal),HL     ; almacena el valor actualizado del caudal acumulado
    ld      DE,(Vol)        ; carga el volumen acumulado en el tanque al acc. DE
    ld      BC,100          ; carga BC = 100
    xor     A               ; borra el carry
    sbc     HL,BC           ; prueba si se ha acumulado un litro al volumen del tank
    jp      m,contin        ; si no es así, salta
    ld      (Caudal),HL     ; sino, guarda el valor actualizado del caudal acumulado
    inc     DE              ; se incrementa el volumen en un litro
    sbc     HL,BC           ; prueba si se ha acumulado un litro al volumen del tank
    jp      m,temp2         ; si no es así, salta
    inc     DE              ; se incrementa el volumen en un litro
    ld      (Caudal),HL     ; almacena el valor actualizado del caudal acumulado

temp2:
    ld      A,0FEH          ; se carga una máscara en el acc. A
    and     D               ; se prueban los 7 bits más sig. [si vol(actual) > 511]
    jp      z,temp1         ; si es menor ó igual a 511, salta
    ld      DE,01FFH        ; sino, se carga el volumen mayor [Vol(max) = 511]

temp1:
    ld      (Vol),DE        ; se almacena en mem. el valor actualizado del volumen
    ld      H,D
    ld      L,E             ; se copia el volumen al acc. HL
    srl     D
    rr      E               ; se calcula h -altura del tanque - (DE = h = vol/2)
    ld      A,E             ; se copia el valor de h al acc. A
    ld      (h),A           ; se almacena en mem. el valor actualizado de la altura

    ld      C,08H           ; se carga un contador p/ conversión
rotar:
    rra                     ; se rota el valor de h
    rl      B               ; se rota el dato proveniente de h
    dec     C               ; decrementa el contador
    jp      nz,rotar        ; si no es cero el contador, salta a rotar
    ld      A,B             ; se transfiere el resultado al acc. A
    out     (pdac),A        ; se envía el resultado al puerto de salida D/A

    xorA                    ; se borra el contenido del acc. A
    cp      D               ; prueba si h(calculada) > h(max), donde h(max) = 255
```

```
    jp      z,NivCal        ; si no se ha excedido el valor máximo, salta

    jp      Error           ; sino, ha ocurrido un error y salta a la rutina de error 1


;-------- Calculo de la altura para simulación (h24) del tanque ----------

NivCal:
NivelH:
    ld      A,(h24)         ; se carga el valor en mem. de la altura en leds al acc A
    ld      B,A             ; se transfiere al acc. B
    ld      A,E             ; se carga la altura h en el acc. A
    cp      00H             ; prueba si h es cero
    jp      z,tankk         ; si lo es salta

    sla     L
    rl      H
    add     HL,DE           ; se HL = h * 5
    ld      DE,51
    call    div             ; se convierte h en h24 [ HL = (h * 5) / 51 = h24]
    ld      A,L             ; se transfiere h24 al acc. A

tankk:
    cp      B               ; prueba si h24(actual) > h24(anterior)
    jp      z,fintank       ; si es igual no se cambia la simulación de llenado, salta
    jp      m,vaxiar        ; si es menor, salta para disminuir la altura (# de leds)
    inc     B               ; sino, incrementa la altura h24 (# de leds encendidos)
    ld      HL,tank2        ; carga la dirección de la parte baja del tank (fila/byte 2)
    scf                     ; se activa el carry (bit que será transferido al tanque)
    rr      (HL)            ; simula que el nivel de agua sube en la parte baja
    dec     HL              ; apunta a la dirección de la parte media del tank (fila 1)
    rr      (HL)            ; simula que el nivel de agua sube en la parte media
    dec     HL              ; apunta a la dirección de la parte alta del tank (fila 0)
    rr      (HL)            ; simula que el nivel de agua sube en la parte alta
    jp      tankk           ; salta a la etapa tankk del programa

vaxiar:
    dec     B               ; decrementa la altura h24 (# de leds encendidos)
    ld      HL,tank0        ; carga la dirección de la parte alta del tank (fila/byte 0)
    sla     (HL)            ; simula que el nivel de agua baja en la parte alta
    inc     HL              ; apunta a la dirección de la parte media del tank (fila 1)
    rl      (HL)            ; simula que el nivel de agua baja en la parte media
    inc     HL              ; apunta a la dirección de la parte baja del tank (fila 2)
    rl      (HL)            ; simula que el nivel de agua baja en la parte baja
    jp      tankk           ; salta a la etapa tankk del programa


fintank:
    ld      (h24),A         ; se actualiza el valor de la altura en base al # de leds
```

```
contin:
    ld      A,02H           ; carga el valor del controlador de tiempo de cálculos
FFin:
    ld      (contTank),A ; actualiza el valor del controlador de tiempo de cálculos

    pop     HL
    pop     AF              ; recupera el valor de los acc. guardados en la pila

    retn                    ; fin de interrupción no enmascarable



Error:
    ld      A,0F0H          ; carga el mensaje de error #1
Error11:
    out     (pdisp),A       ; se despliega en los leds de uso general
    jp      Error11         ; salta a desplegar nuevamente el mensaje

Error2:
    ld      A,0FH           ; carga el mensaje de error #2
Error22:
    out     (pdisp),A       ; se despliega en los leds de uso general
    jp      Error22         ; salta a desplegar nuevamente el mensaje

Error3:
    ld      A,0C3H          ; carga el mensaje de error #3
Error33:
    out     (pdisp),A       ; se despliega en los leds de uso general
    jp      Error33         ; salta a desplegar nuevamente el mensaje


      .org  2000H


fo: .db00H    ; mem. que controla la simulación flujo del caudal de salida
fi:  .db00H   ; mem. que controla la simulación flujo del caudal de entrada

fc0:    .db 00H   ; control del flujo de entrada en byte fila 0 del tank (parte alta)
fc1:    .db 00H   ; control del flujo de entrada en byte fila 1 del tank (parte med.)
fc2:    .db 00H   ; control del flujo de entrada en byte fila 2 del tank (parte baja)
fc3:    .db18H    ; control de caudal in/out en byte fila 3 del tank (tuberías)

fcx:    .dw    fc0     ; mem. que controla la fila a desplegarse en la matriz
col:    .db    11H     ; mem. que controla la columna a desplegarse en la matriz

tank0:     .db    00H   ; control de altura del tanque en byte fila 0 (parte alta)
tank11:    .db    00H   ; control de altura del tanque en byte fila 1 (parte media)
tank2:     .db    00H   ; control de altura del tanque en byte fila 2 (parte baja)
tank3:     .db    00H   ; control de altura del tank en byte fila 3 (tuberías in/out)
```

```
h24:   .db    00H       ; altura en base al num. de leds de simulación (0 ~ 24)
h:     .db    00H       ; altura del nivel de líquido en el  tanque (0 ~ 255)
Qin:   .db    0FFH      ; caudal de entrada del tanque (0 ~ 255)
Qo:    .db    00H       ; caudal de salida del tanque ( 0 ~ 255)
Caudal:  .dw    0000H    ; caudal acumulado en el tanque

RxMem1:    .dw    0000H    ; mem. 1 de datos temporales de la rutina de raíz
RxMemG:    .dw    0000H    ; mem. 2 de datos temporales de la rutina de raíz
RxMemN:    .dw    0000H    ; núm. del que se obtendrá raíz  cuadrada

Vol:       .dw    0000H    ; volumen de liquido contenido por el tanque
marka:     .db    00H      ; bandera para indicar que finalizó la NMI (completa)
marka2:    .db    00H      ;
contTank:  .db    02H      ; mem. que controla que pasen 10ms p/ los cálculos
ContQo:    .db    04H      ; mem. que controla la velocidad del flujo de Qo
ContQi:    .db    04H      ; mem. que controla la velocidad del flujo de Qin

;////////////////////////////////////////////////////// ////////////////////////////////////////////////////////////////

;*****************************************************************************************
;     Rutina de Multiplicación

;          A * (H)L = HL
;-----------------------------------------------------------------------------------------


     .org    0500H

mult:
   push  DE          ; almacena en la pila el valor del acc. DE

   or    A           ; borra la bandera de carry
   ld    DE,0000H    ; borra el contenido del acc. DE
   sbc   HL,DE       ; prueba si el multiplicando es cero
   jp    z,FinMult   ; si lo es finaliza la multiplicación y salta al fin
   ex    DE,HL       ; sino, HL = 0000H y DE = multiplicando
   or    A           ; prueba si el multiplicador es cero
   jp    nz,InicM    ; si no lo es, salta para iniciar la multiplicación
   jp    FinMult     ; sino, finaliza la multiplicación y salta al fin

StBit:
   sla   E
   rl    D           ; se rota el multiplicando (se multiplica por 2)
InicM:
   srl   A           ; se rota el multiplicador a la derecha (se divide entre 2)
   jp    nc,StBit    ; si el bit LSB rotado no es 1 lógico, salta
```

```
    add   HL,DE       ; sino, se suma al producto el multiplicando (específico)
    or    A           ; prueba si el multiplicador es cero
    jp    nz,StBit    ; si no es cero salta y continúa la multiplicación

FinMult:
    pop   DE          ; recupera el valor del acc. DE guardado en la pila

    ret               ; retorna de la subrutina

;////////////////////////////////////////////////////////////////////////////////////////////////////

;*************************************************************************************
;
;         Rutina de División

;    HL / DE = HL     (DE: residuo)
;-------------------------------------------------------------------------------------

     .org   0600H

div:
    push  AF          ; se almacena en la pila el contenido de los acc.
    push  BC
    push  IX

    ld    IX,0000H    ; borra el contenido del reg. Índice IX
    ld    BC,0000H    ; borra el contenido del acc. B
    xor   A           ; borra el contenido del acc. A y borra el carry
    sbc   HL,BC       ; prueba si el dividendo es cero
    jp    nz,PbDiv1   ; si no lo es, salta para continuar con la división
    ld    D,H
    ld    E,L         ; sino se borra el divisor
    jp    FinDiv1     ; salta para finalizar la rutina

PbDiv1:
    ex    DE,HL       ; DE = dividendo y HL = divisor
    sbc   HL,BC       ; prueba si el divisor es cero
    ex    DE,HL       ; HL = dividendo y DE = divisor
    jp    nz,PbDiv2   ; si el divisor no es cero, salta para continuar
    jp    Error3    ; sino ha habido un error (división entre cero) y salta

PbDiv2:
    ld    (Divisor),HL ; se almacena en mem. el divisor
    sbc   HL,DE       ; HL = divisor - dividendo
    jp    m,FinDiv    ; si el resultado es negativo finaliza la división (salta)
    ld    HL,(Divisor) ; sino, se carga el valor del divisor al acc. HL
    ld    B,D
    ld    C,E         ; se copia el dividendo al acc. BC
```

```
    cp      H           ; prueba si el divisor es mayor a 255
    jp      nz,SiH      ; si el divisor es mayor a 255, salta
    ld      A,01H       ; sino, carga el menor exponente (dividendo - divisor)
Rot1:
    sla     L           ; se multiplica por 2 el divisor
    jp      c,FnRot1    ; si ya excedió el valor de 255, salta
    sla     E           ; sino, se multiplica por 2 el dividendo
    jp      Rot1        ; salta a Rot1 y repite la operación nuevamente
Rot2:
    sla     C           ; C = dividendo x 2
    rlca                ; multiplica el exponente por 2 (valor en potencias de 2)
FnRot1:
    sla     E           ; E = dividendo x 2
    jp      nc,Rot2     ; si no se ha excedido el valor de 255, salta a Rot2
    ld      E,A         ; se transfiere la diferencia de los exponentes al acc. E
ndetec:
    inc     D           ; D=D+1 (calcula la dif. de exp. ya no en potencias de 2)
    rrca                ; divide la diferencia de exp. (en potencias de 2) entre 2
    jp      nc,ndetec   ; si no ha llegado a cero la diferencia de exp., salta
    ld      A,D         ; la diferencia de exp. se transfiere al acc. A
    ld      D,00H       ; borra el contenido del acc. D
    jp      Div1        ; salta para iniciar la división


SiH:
    ld      A,01H       ; carga el contador del exponente dividendo - divisor
    sla     H           ; se multiplica por 2 el divisor
    jp      c,FnRot3    ; si ya excedió el valor de 0FFFFH, salta
    sla     E
    rl      D           ; sino, se multiplica por 2 el dividendo
    jp      SiH         ; salta a SiH  y repite la operación nuevamente
FnRot3:
    ld      HL,0001H    ; sino, carga el menor exponente (dividendo - divisor)
    jp      SiH2        ; salta a calcular la diferencia de los exponentes
Rot4:
    inc     A           ; incrementa el contador del exp. dividendo - divisor
    sla     C
    rl      B           ; BC = dividendo x 2
    sla     L
    rl      H           ; HL = divisor x 2
SiH2:
    sla     E
    rl      D           ; DE = dividendo x 2
    jp      nc,Rot4     ; si no se ha excedido el valor de 0FFFFH, salta a Rot4
    ex      DE,HL       ; DE = divisor     y   HL = dividendo

Div1:
    ld      HL,(Divisor) ; se carga en HL el valor del divisor (valor inicial)
Div2:
```

```
    dec     A           ; decrementa el contador (n) del exp. dividendo - divisor
    jp      m,FinDiv    ; si su valor es neg. ha terminado la división y salta
    and     A           ; sino, borra el carry
    sbc     HL,BC       ; HL = divisor – [ dividendo x (2 elevado a la n)]
    jp      m,NoDiv     ; si el resultado es negativo, salta

    add     IX,DE       ; IX(cociente) = cociente + [dividendo x (2 elev. a la n)]
    ld      (Divisor),HL ; se actualiza la mem. del divisor con su nuevo valor
    jp      z,FinDiv    ; si HL = a cero, es el fin de la división y salta
    srl     D
    rr      E           ; DE = divisor / 2
    srl     B
    rr      C           ; BC = [ dividendo x (2 elevado a la n)] / 2
    jp      Div2        ; salta para continuar con el proceso de división

NoDiv:
    srl     D
    rr      E           ; DE = divisor / 2
    srl     B
    rr      C           ; BC = [ dividendo x (2 elevado a la n)] / 2
    jp      Div1        ; salta para continuar con el proceso de división

FinDiv:
    ld      DE,(Divisor) ; DE = residuo de la división
    push    IX          ; se transfiere el cociente a la pila
    pop     HL          ; HL = cociente (= resultado de la división)
FinDiv1:
    pop     IX          ; se cargan los valores de los acc. guardados en la pila
    pop     BC
    pop     AF

    ret                 ; retorno de la subrutina


  .org   2100H


Divisor:   .dw    0000H    ; mem. temporal para el cálculo de la división

;////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


      .end
```

7.7. Diagramas y Circuitos Impresos.

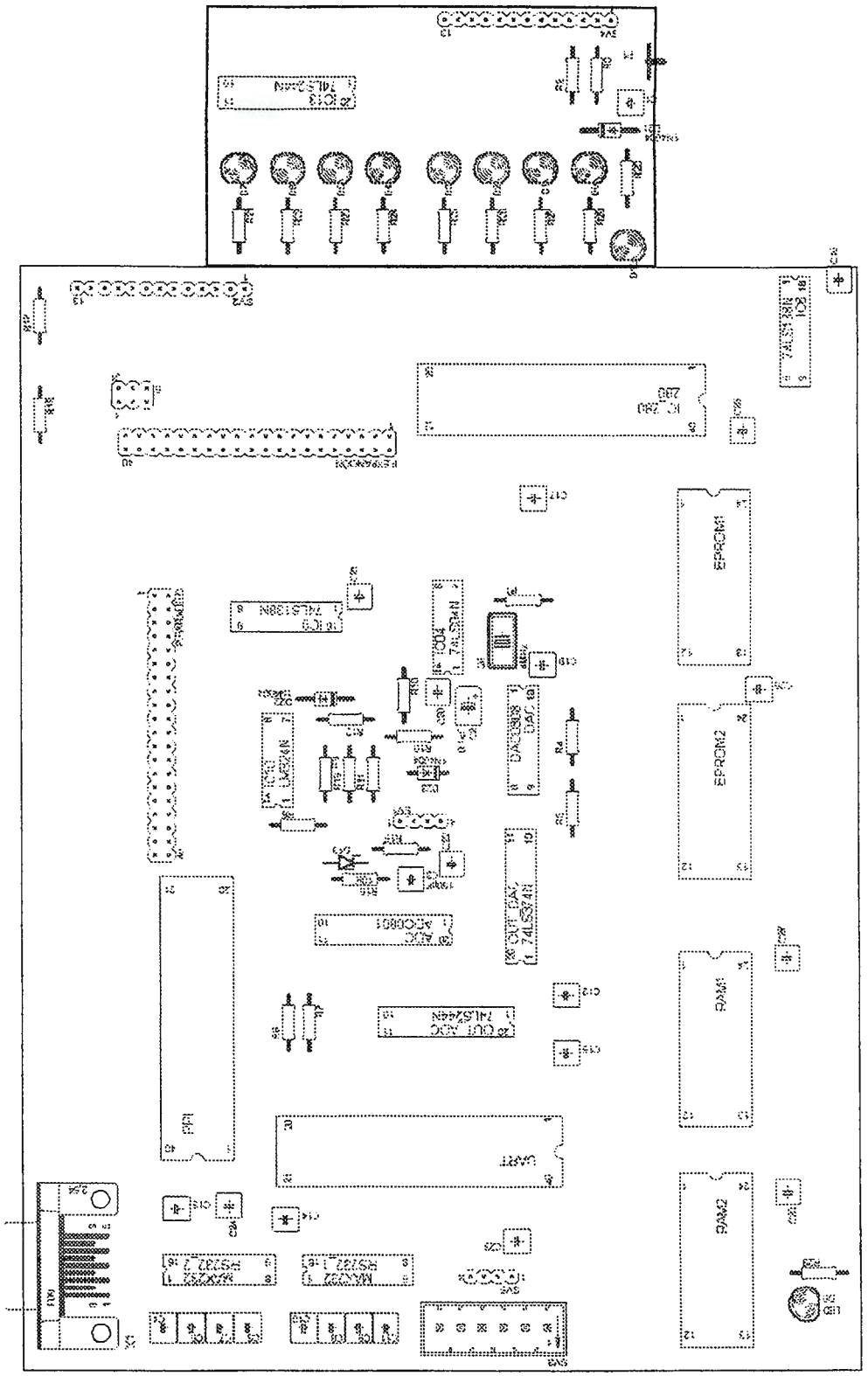


Fig. 21 Ubicación de los Componentes en el Circuito impreso del Módulo

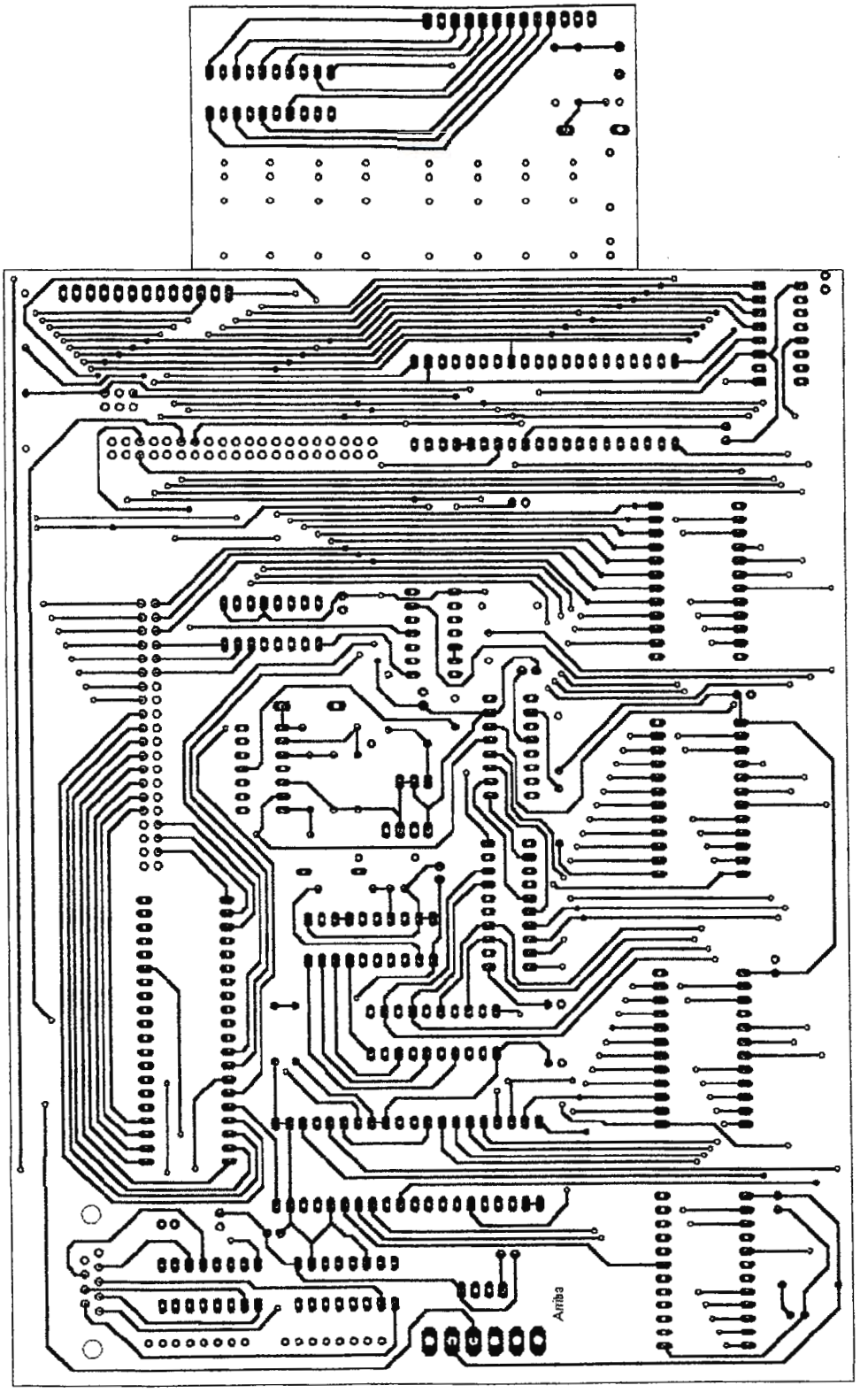


Fig. 22 Circuito impreso del CPU del Módulo. Cara 1: Componentes.

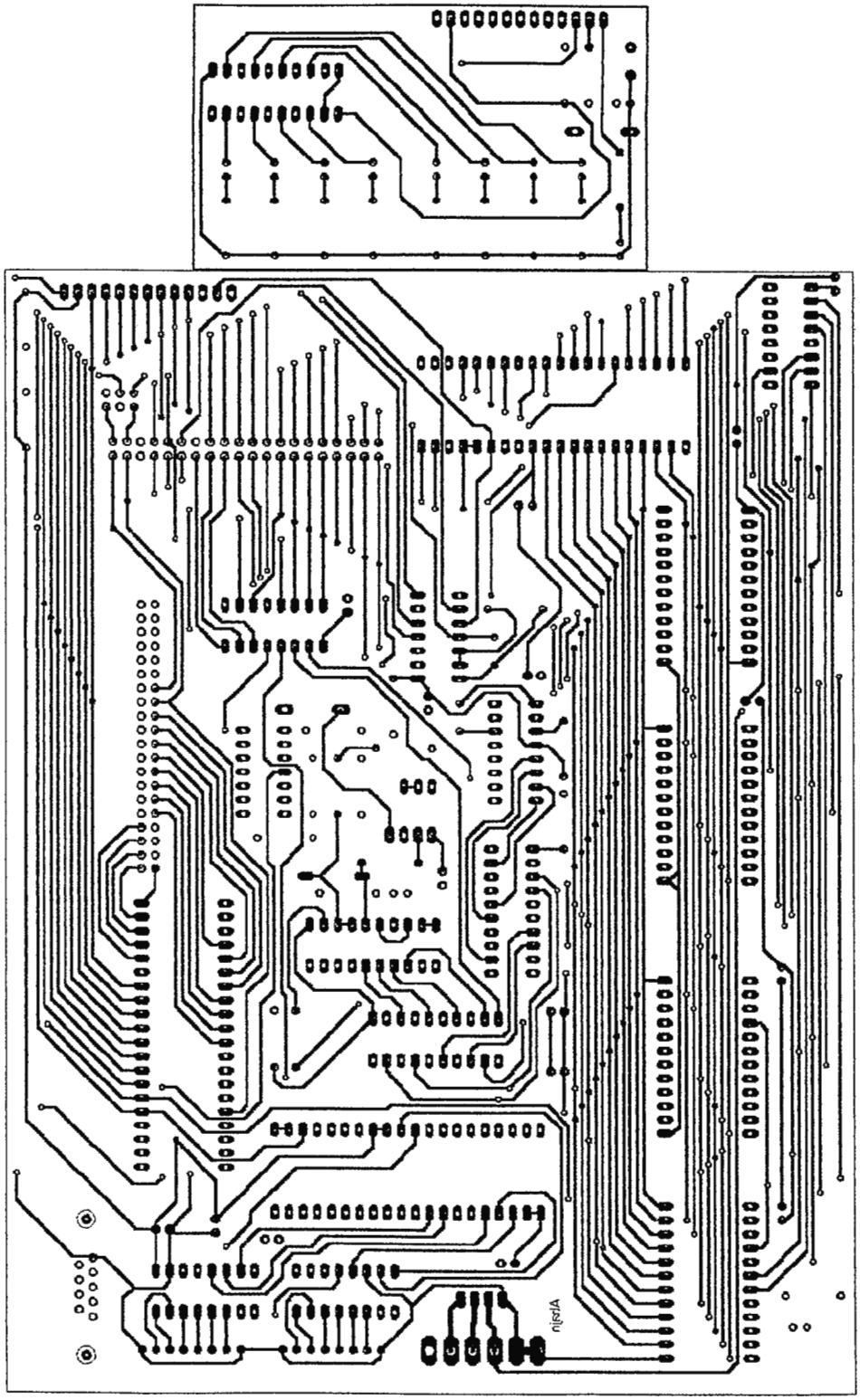


Fig. 23 Circuito impreso del Módulo. Cara 2: Soldaduras.

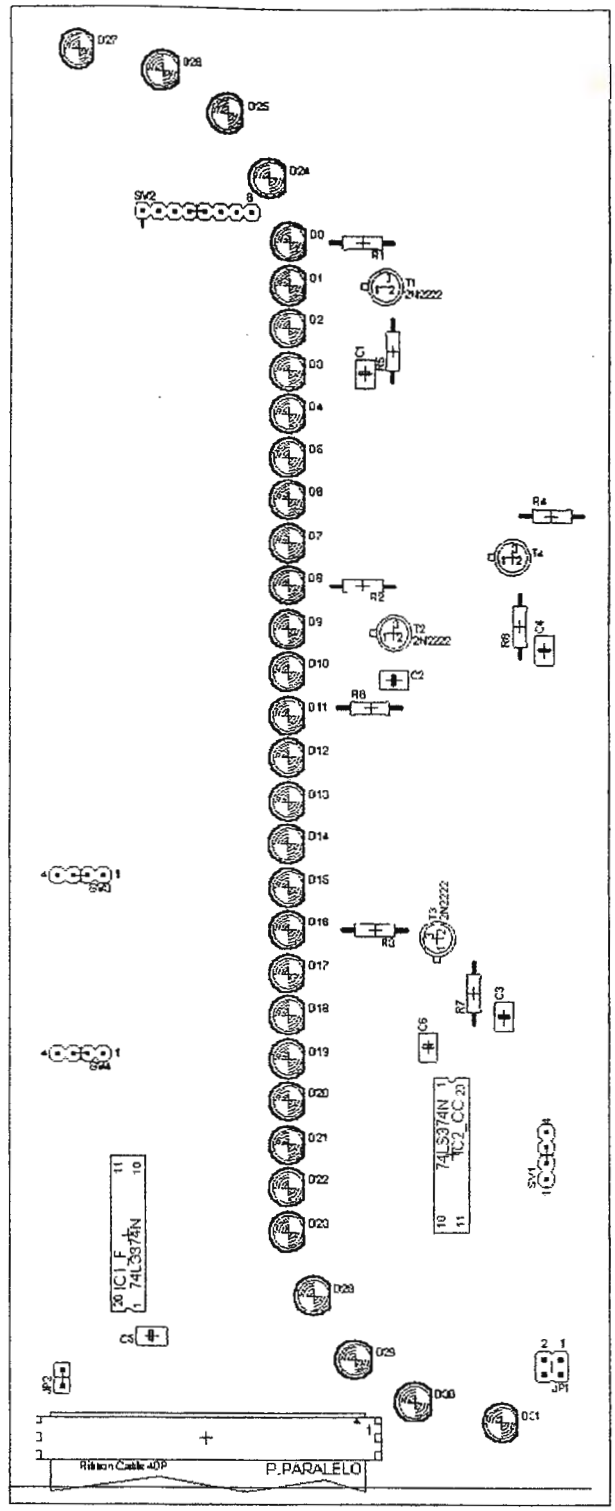


Fig. 24 Ubicación de los Componentes en el Circuito impreso de la Matriz de Leds.

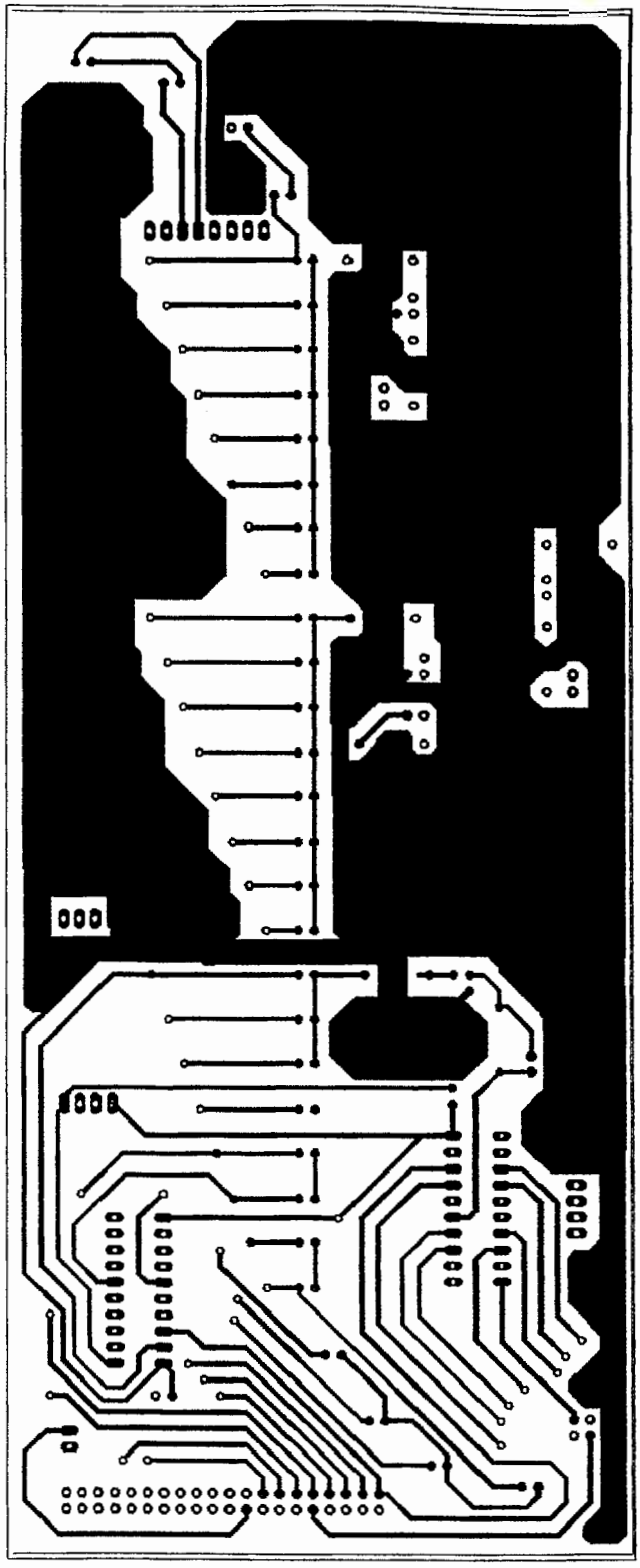


Fig. 25 Circuito impreso de la Matriz de Leds. Cara 1: Componentes.

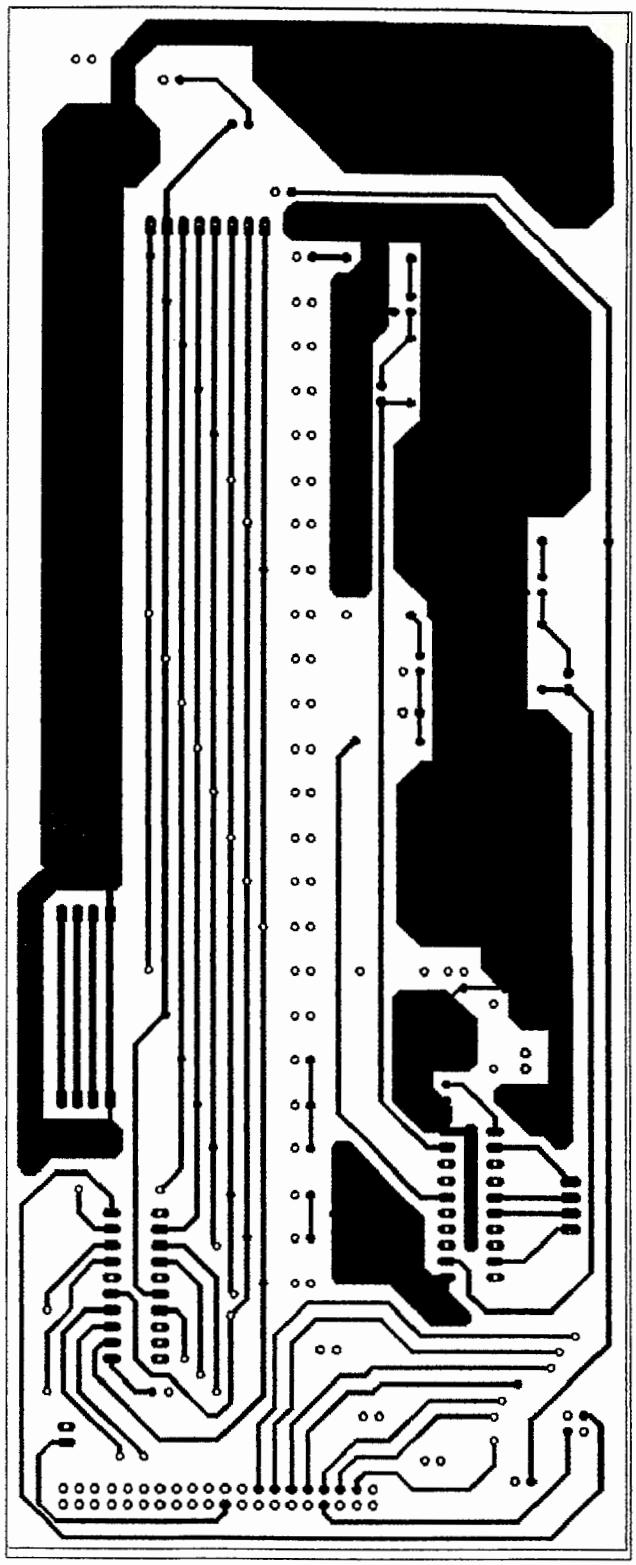
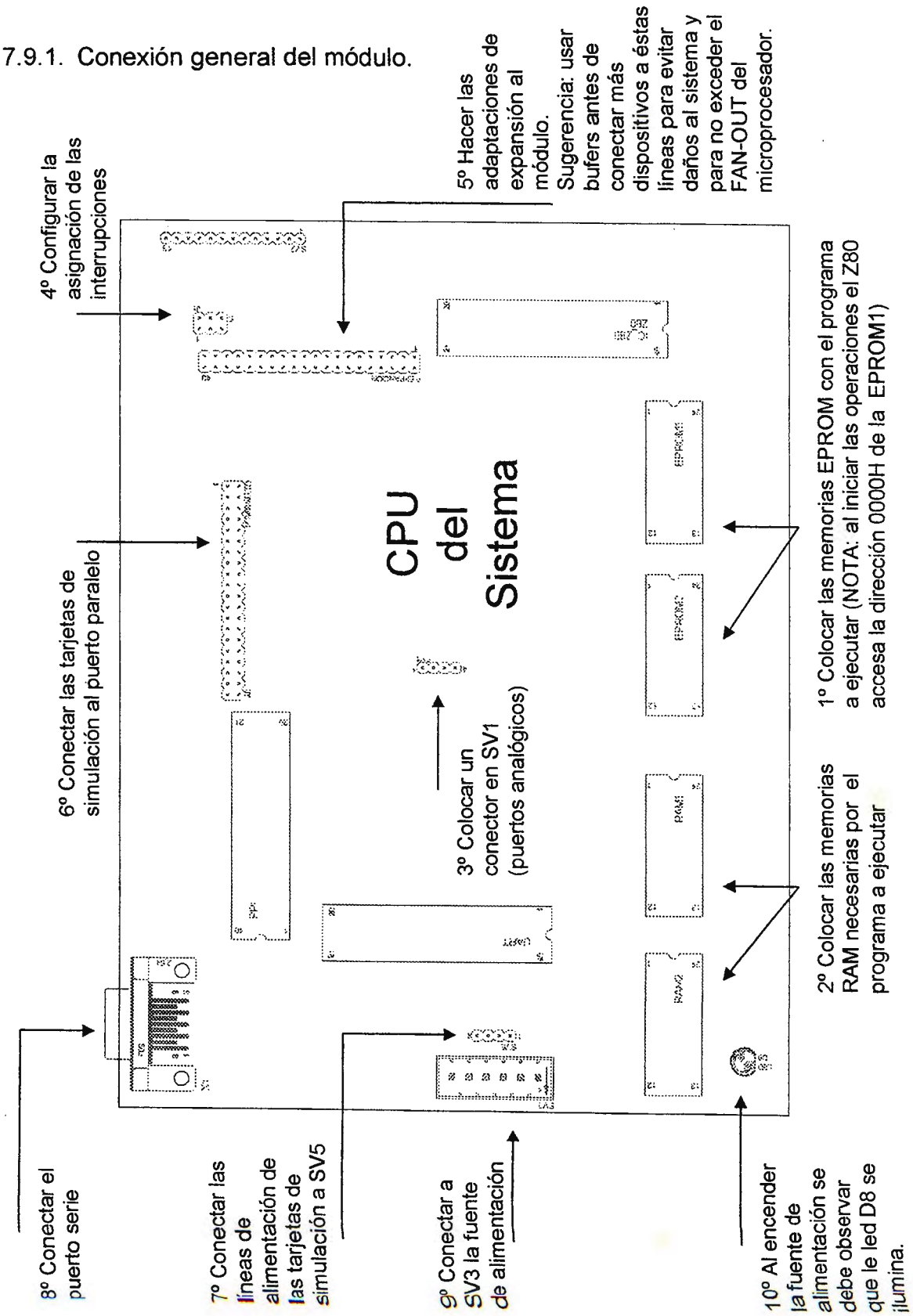


Fig. 26 Circuito impreso de la Matriz de Leds. Cara 2: Soldaduras.

7.8. Presupuesto.

Elemento	Reemplazo NTE	Precio Unitario	Cantidad	Precio
Z80CPU	3880	$4.00	1	$4.00
7404	7404	$1.13	1	$1.13
EPROM	2732A	$11.43	1	$11.43
RAM	2117	$6.45	1	$6.45
RS232	MAX232	$5.00	2	$10.00
74374	74374	$2.05	3	$6.15
74138	74138	$1.68	2	$3.36
74244	74244	$1.00	2	$2.00
LM324		$2.11	1	$2.11
82C50A		$10.00	1	$10.00
82C55A	8255A	$14.03	1	$14.03
ADC0804	2056	$9.18	1	$9.18
DAC0804		$4.07	1	$4.07
Conectores DB9		$0.51	1	$0.51
Cristal 4MHz		$5.16	1	$5.16
Circuito impreso		$20.00	1	$20.00
Bases: 14 pines		$0.56	2	$1.12
16 pines		$0.45	5	$2.25
20 pines		$0.68	6	$4.08
24 pines		$0.68	4	$2.72
40 pines		$0.56	3	$1.68
2N2222A	123A	$0.61	4	$2.44
1N4001	1N4001	$0.26	2	$0.52
1N4733A Diodo Zener 5.1V 1W		$0.61	1	$0.61
Pulsador		$0.50	1	$0.50
Interruptor		$0.50	1	$0.50
Condensador 10uF electrolítico		$0.10	4	$0.40
LM555		$0.64	1	$0.64
Condensador 0.1uF cerámico		$0.10	24	$2.40
Leds		$0.25	40	$10.00
Resistencias (Valor promedio)		$0.08	35	$2.80
			TOTAL:	$142.24

7.9.1. Conexión general del módulo.



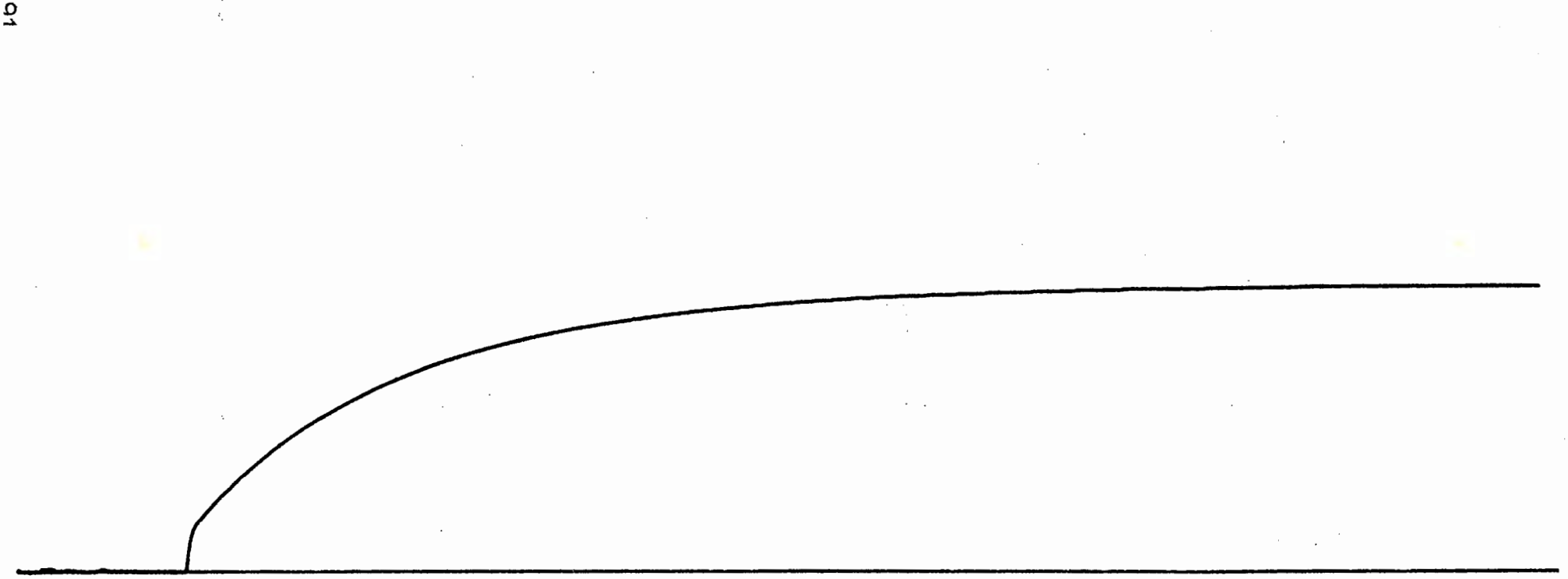
7.9.2. Configuración del módulo para la emulación de la planta digital: Ascensor de Edificio.

- Del inciso anterior realizar las conexiones de los ítems: 1º, 2º, 6º, 7º, 8º y 9º.
- En el inciso 6º el circuito la planta a utilizar es el de la matriz de leds presentado en la fig. 24.
- Para el refrescamiento de la matriz de leds es necesario conectar la terminal NMI\ del microprocesador (pin 6 de J1) a la salida del circuito temporizador generador de interrupciones NMI de la fig. 17 (pin 1 de JP1)
- En el literal 8º la conexión se realizará con una PC al COM1.
- Cargar en la PC el programa Controlador de Ascensor de Edificio (Desarrollado en Visual Basic), expuesto en la sección 7.4
- Encender el módulo (de ser necesario resetear el módulo)

7.9.3. Conexión del módulo para la emulación de la planta analógica: Tanque de Líquidos.

- Del inciso 7.9.1 realizar las conexiones de los ítems: 1º, 2º, 3º, 6º, 7º y 9º.
- En el inciso 6º el circuito la planta a utilizar es el de la matriz de leds presentado en la fig. 24.
- Para el refrescamiento de la matriz de leds es necesario conectar la terminal NMI\ del microprocesador (pin 6 de J1) a la salida del circuito temporizador generador de interrupciones NMI de la fig. 17 (pin 1 de JP1)
- Encender el módulo (de ser necesario resetear el módulo)

RESPUESTA AL ESCALÓN DE LA PLANTA ANALÓGICA: ANQUE DE LÍQUIDOS



8. BIBLIOGRAFÍA.

- ✓ TOCCI, Ronald. SISTEMAS DIGITALES. Principios y Aplicaciones. Prentice Hall. Quinta Edición. Méjico1993.

- ✓ BREY, Barry. LOS MICROPROCESADORES DE INTEL. Prentice Hall. Tercera Edición. Impreso en Méjico 1994.

- ✓ Tesis. ENTRENADOR PARA MICROPROCESADOR Z80. Universidad Don Bosco.

- ✓ STREETER, Víctor L. MECÁNICA DE LOS FLUIDOS. Mc Graw Hill. Novena edición. Méjico

- ✓ OGATA, Katsuhiko. INGENIERÍA DE CONTROL MODERNA. Prentice Hall. Segunda Edición. Impreso en Méjico 1993.